Virtual University of Pakistan

Federal Government University

Software process Improvement

Course Code – CS-724

Course Instructor

Dr. Ghulam Ahmad Farrukh
Ph.D Software Engineering
George Mason University, USA

Reference Books:

1. Software Quality: Analysis and Guidelines for Success by Capers Jones
2. Software Assessments, Benchmarks, and Best Practices by Capers Jones
3. Customer-oriented Software Quality Assurance by Frank P. Ginac
4. Software Engineering by Sommerville
5. Software Engineering: A Practitioner's Approach by Roger S. Pressman
6. Requirements Engineering: Processes and Techniques by Kotonya and Sommerville
7. Inroads to Software Quality by Alka Jarvis and Vern Crandell
8. Software Requirements: Objects, States, and Functions by Alan M. Davis
9. Software Engineering Quality Practices by Ronald K. Kandt
10. High Quality Low Cost Software Inspections by Ronald A. Radice

Video Links: http://www.vumultan.com/CS724Video.aspx

Published By: www.vumultan.com

Lecture 1
Software Process Improvement

Agenda

- Objectives of this course
- What is a process?
- What is a software process? Why we need them
- Process models
- Process improvement

Objectives of This Course

- To introduce students to the basics of software process and process improvement
- To teach students about the activities and issues software process engineering
- To teach students different software process improvement approaches
- To teach students concept of measurements and how it applies to software processes
- To introduce students advance and potential research topics in software process engineering
- Provide an academic/theoretical background in the class room about an interesting area of research and practice
- Let students explore the local industry and understand their processes and suggest an improvement strategy

Processes

- Processes are part of all aspects of life and are an essential mechanism for coping with complexity in the world
- Description of processes are very important because they allow knowledge to be reused

What is a Process?

- Once someone has worked out how to solve a problem, they can document the way in which that solution was derived as a process
- This then helps other people faced with similar problems to get started on their own solutions
- Examples of Processes
  - ✓ An instruction manual for a kitchen dishwasher describes the process of using that machine to clean dishes
  - ✓ A cookery book describes a set of processes to prepare and cook various different types of meals
  - ✓ A procedures manual in a bank describes the ways in which different banking processes such as agreeing a personal loan, correcting errors, etc. should be carried out
  - ✓ A quality manual for software development describes the processes which should be used to assure the quality of the software. It may include descriptions of standards which are basis for the quality checking

Software Processes

Software Problems

- Technical
  - ✓ Need a technical solution
  - ✓ Easy to understand and solve
- Systemic
  - ✓ Difficult to understand and solve
  - ✓ Need a process/management solution

Solution to Software Problems

- Treat the software task as a process that can be controlled, measured, and improved
- Relate the required tasks, tools, methods with skill, training, and motivation of people involved

Software Processes

- Software engineering, as a discipline, has many processes
- These processes help in performing different software engineering activities in an organized manner
- The software process is the set of tools, methods, and practices that people use to produce (develop and maintain) software and associated products , e.g., project plans, design documents, code, test cases, and user manuals
- Where are the people? Aren't they the most important part of an organization?

Characteristics of Software Processes

- Requires creativity
- Interactions between a wide range of different people
- Engineering judgment
- Background knowledge
- Experience

Examples of Software Processes

- Software engineering development process (SDLC)
- Requirements engineering process
- Quality assurance process
- Change management process
- Design process.

Process Management

Basic Concepts

- The objectives of software process management are produce products according to plan while simultaneously improving the organization's capability to produce better products
- The basic principles are those of statistical process control, which have been used successfully in many fields
- A process is said to be stable or under statistical control if its future performance is predictable within established statistical limits
- When a process is under statistical control, repeating the work in roughly the same way will produce roughly the same result
- To obtain consistently better results, it is thus necessary to improve the process
- If the process is not under statistical control, sustained progress is not possible until it is
- The basic principle behind statistical control is measurement
- **Lord Kelvin Said:**
  - ✓ When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of the knowledge, but you have scarcely in your thoughts advanced to the stage of science
- You cannot just start to use numbers to control things
- The numbers must properly represent the process being controlled, and they must be sufficiently well defined and verified to provide a reliable basis for action
- While process measurements are essential for orderly improvement, careful planning and preparation are required or the results are likely to be disappointing
- The mere act of measuring human processes changes them
- Since people's fears and motivations are involved, the results must be viewed in a different light from data on natural phenomenon. It is thus essential to limit the measurements to those with predefined use
- Measurements are both expensive and disruptive; overzealous measuring can degrade the process we are trying to improve

Process Models

Process Models

- A process model is a simplified description of a process presented from a particular perspective
- There may be several different models of the same process
- No single model gives a complete understanding of the process being modeled

Variations in Process Models

- A process model is produced on the anticipated need for that model
- A model to help explain how process information has been organized
- A model to help understand and improve a process
- A model to satisfy some quality management standard

Types of Process Model

- Coarse-grain activity models
  - ✓ This type of model provides an overall picture of the process
  - ✓ Describes the context of different activities in the process
  - ✓ It doesn't document how to enact a process
- Fine-grain activity models
  - ✓ These are more detailed models of a specific process, which are used for understanding and improving existing processes

Enactment of Processes

- Different people usually enact the process in different ways
- Sometimes the same person will enact the same process in different ways at different times
- Different people have different backgrounds

Process Enactment

- People with experience may change the order of stages in a process or combine stages because they understand the consequences of what they are doing. However, inexperienced people follow the stages as described because they don't have this background knowledge
- Different software support in different environments

Process Improvement

Process Improvement Objectives

- Quality improvement
- Schedule reduction
- Resource reduction

Process Improvements Planning

- What are the improvement goals?
- How can we introduce process improvements to achieve these goals?
- How should improvements be controlled and managed?

Six Steps to Software Improvements

- Understand the current status of development processes
- Develop a vision for the desired process
- Establish a list of required process improvement actions in order of priority
- Produce a plan to accomplish the required actions
- Commit the resources to execute the plan
- Start over at step 1

Process Maturity Levels

- Initial
- Repeatable
- Defined
- Managed
- Optimizing

References

- Managing the Software Process, Chapter 1, by Watts Humphrey
- Requirements Engineering, Chapter 2 [2.0, 2.1, 2.4], by Gerald Kotonya and Ian Sommerville

Lecture 2
Software Quality Assurance

SQA Lecture Agenda

- Quality and its need
- SQA tasks
- SQA group skills and responsibilities
- SQA Reviews
- SQA Reporting

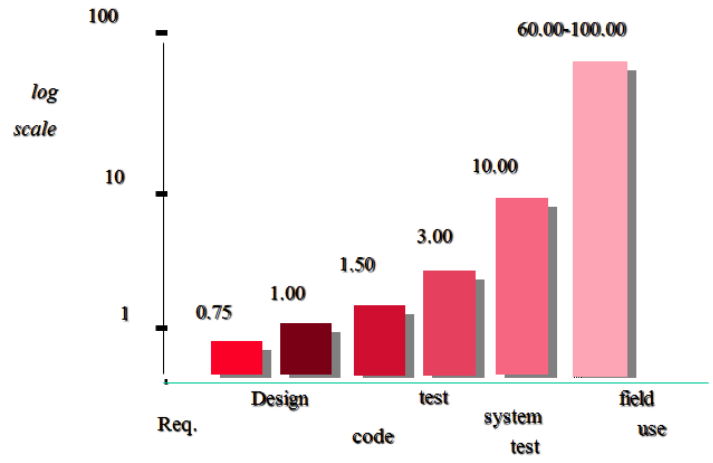
Software Quality

- Low levels of defects when deployed, ideally approaching zero
- High reliability, or the capability of running without crashes or strange results

Why Software Quality?

- Reduces time to market for new products
- Enhances market share compared to direct competitors
- Minimizes "scrap and rework" expenses
- Attracts and keeps "top-gun" personnel
- Minimizes the risk of serious litigation
- Minimizes the risk of serious operating failures and delays
- Minimizes the risk of bankruptcy or business failures, which may be attributed directly to poor quality or poor software quality

Cost to Find and Fix a Defect



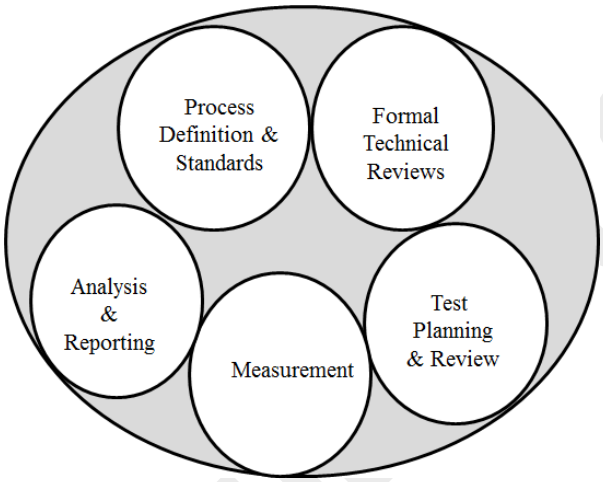
Software Quality Assurance

- So, there is a need for an active and independent group within every organization, who is devoted to assuring quality of software products
- This group is called SQA

Goals of SQA

- To improve software quality by appropriately monitoring both the software and development process that produces it
- To ensure full compliance with the established standards and procedures for the software and the software process
- To ensure that any inadequacies in the product, the process, or the standards are brought to management's attention so they can be fixed

SQA in pictorial form



Skills of SQA Group

- Statistical methods
- Quality control principles
- Software process
- An ability to deal effectively with people in contentious situations

Role of SQA

- The people responsible for the software projects are the only people who can be held responsible for the quality of software
- The role of SQA is to monitor the way these groups perform their responsibilities
- This seems like a contradiction

Pitfalls

1. It is a mistake to assume that the SQA people themselves can do anything about quality
2. The existence of an SQA group does not ensure that the standards and procedures are followed
3. Unless management periodically demonstrates its support for SQA by following their recommendations, SQA will be ineffective
4. Unless line management requires that SQA try to resolve their issues with project management before escalation, SQA and development will not work together effectively

SQA Responsibilities

- Review all development and quality plans for completeness
- Participate as inspection moderators in design and code inspections
- Review all test plans for adherence to standards
- Review a significant sample of all test results to determine adherence to plans
- Periodically audit SCM performance to determine adherence to standards
- Participate in all project quarterly and phase reviews and register non-concurrence if the appropriate standards and procedures have not been reasonably met

SQA Reviews

What is a Review?

- A process or meeting during which a work product, or a set of work products, is presented to project personnel, managers, users, or other interested parties for comment or approval. Types include code review, design review, formal qualification review, requirements review, test readiness review
  - ✓ IEEE Std. 610.12-1990

Objectives of Formal Technical Reviews (FTRs)

- Uncover errors in function, logic, or implementation for any representation of the software
- To verify that the software under review meets its requirements
- To ensure that the software has been represented according to predefined standards
- To achieve software that is developed in a uniform manner
- Make projects more manageable
- Ownership transfers from individual to group
- In addition, the FTR serves as a training ground, enabling junior engineers to observe different approaches to software analysis, design, and implementation
- Also serves as a means of corporate backup

What Technical Reviews Are Not!

- A project budget summary
- A scheduling assessment
- An overall progress report
- A mechanism for reprisal or political intrigue!!

Review Roles

- Facilitator
- Author
- Recorder
- Reviewer
- Observer

Responsibilities of Roles

Responsibilities of Facilitator

- Responsible for providing the background of the work and assigning roles to attendees
- Encourages all attendees to participate
- Keeps the meeting focused and moving
- Responsible for gaining consensus on problems

Responsibilities of Author

- Responsible for the readiness and distribution of material to be reviewed
- During the meeting, the author paraphrases the document a section at a time
- Responsible for
  - ✓ scheduling the review
  - ✓ selecting the review participants
  - ✓ determining if the entry criteria for the review are met
  - ✓ providing information about the product during all stages
  - ✓ clarifying any unclear issues
  - ✓ correcting any problems identified
  - ✓ providing dates for rework and resolution

Responsibilities of Recorder

- Collects and records each defect uncovered during the review meeting
- Develops an issues list and identifies whose responsibility it is to resolve each issue
- Records meeting decisions on issues; prepares the minutes; and publishes the minutes, and continually tracks the action items

Responsibilities of Reviewer

- Spends time prior to the meeting reviewing information
- Makes notes of defects and becomes familiar with the product to be reviewed
- Identifies strengths of the product
- Verifies that the rework is done
- Insists upon clarifying any issues that are not clear

Responsibilities of Observer

- A new member to the project team, who learns the product and observes the review techniques

Review Activities

- Review planning
  - ✓ Distribute review package one week in advance
    - Document to be reviewed
    - Review agenda
    - Identification of the individual who will manage the agenda and schedule

- Exit and entrance criteria for the review
- Objective of the review
- Names of attendees, their roles and responsibilities
- Review location
- Date and time of review
- List of classifications that will be used for defects discovered (defect type, defect origin, and defect severity)
- Procedures for handling issues raised during the review and escalation phase

- Review meeting
  - ✓ Facilitator begins the meeting with an introduction of agenda, people, and description of their roles
  - ✓ Author of the document proceeds to explain the materials, while reviewers raise issues based on advance preparation
  - ✓ When valid problems, issues, or defects are discovered, they are classified according to their origin or severity and then recorded
  - ✓ These are accompanied with the names of individuals who are responsible for resolution and the time frame during which the item will be resolved
  - ✓ Related recommendations are also recorded
  - ✓ Guidelines for Reviewers
    - Be prepared - evaluate product before the review meeting
    - Review the product, not the producer
    - Keep your tone mild, ask questions instead of making accusations
    - Stick to the review agenda
    - Raise issues, don't resolve them
    - Avoid discussions of style - stick to technical correctness
  - ✓ Decisions at the End of a Review Meeting
    - All attendees must decide whether to
      - Accept the product without further modification
      - Reject the product due to severe errors
      - Accept the product provisionally
      - Hold a follow-up review session

- Review report
  - ✓ Published by the recorder, with approval from all attendees, after a week of the review meeting
  - ✓ Review report consists of
    - Elements reviewed
    - Names of individuals who participated in the review
    - Specific inputs to the review
    - List of unresolved items
    - List of issues that need to be escalated to management
    - Action items/ownership/status
    - Suggested recommendations

- Rework
  - ✓ It is the responsibility of project manager to ensure that all defects identified in the review are fixed and retested
- Follow-up
  - ✓ During the follow-up, that all discrepancies identified are resolved and the exit criteria for the review have been met
  - ✓ Document lessons learned during the final report also

SQA Reporting

- SQA should not report to the project manager
- SQA should report somewhere within the local office and division office
- There should typically be no more than one management position between SQA and the senior location manager
- SQA should always have a "dotted-line" relationship to a senior corporate quality executive
- Whenever possible, SQA should report to someone who has a vested interest in software quality, like the staff head responsible for field service

References

- Software Engineering 5th Edition by Roger Pressman, Chapter 8
- Inroads to Software Quality by Alka Jarvis and Vern Crandall, PH 1997 (Ch. 7)

Lecture 3
Configuration Management

Agenda

- Change
- Software configuration management
- SCM Functions

Change

- Changes will happen in all work products and during all processes during software development and maintenance
- Change increases the level of confusion among software engineers who are working on a software project

Sources of Change

- New business or market conditions dictate changes in product requirements or business rules
- New customer needs demand modification of data produced by information systems, functionality delivered by products, or services delivered by computer-based system
- Reorganization or business growth / downsizing causes changes in project priorities or software engineering team structure
- Budgetary or scheduling constraints cause a redefinition of the system or product

Why All This Modification?

- As time passes, all constituencies know more
  - ✓ About what they need
  - ✓ Which approach would be best
  - ✓ How to get it done and still make money
- Most changes are justified!

Confusion

- Confusion arises when changes are not
- Analyzed before they are made
- Recorded before they are implemented
- Reported to those who need to know
- Controlled in a manner that will improve quality and reduce errors
- We need to minimize this confusion, or else our projects will get out of control

Configuration Management

- The art of coordinating software development to minimize confusion is called configuration management
- The goal is to maximize productivity by minimizing mistakes

Software Configuration Management

- SCM is a set of activities designed to control change by identifying the work products that are likely to change
  - ✓ establish relationships among them
  - ✓ defining mechanisms for managing different versions of these work products
  - ✓ controlling the changes imposed
  - ✓ auditing and reporting on the changes

Purpose of SCM Activities

- Identify change
- Control change
- Ensure that the change is being properly implemented
- Report changes to others who may be interested

SCM provides Cover against

- Lack of visibility
- Lack of control
- Lack of traceability
- Lack of monitoring
- Uncontrolled change

Software Configuration

- The items that comprise all information produced as part of the software process are collectively called a software configuration
  - ✓ Computer programs (source and executable)
  - ✓ Documents that describe the computer programs
  - ✓ Data
- Software configuration items will grow

Baseline every Software Configuration Item

How to Manage Change?

- A baseline is a software configuration management concept that helps us to control change without seriously impeding justifiable change

IEEE definition of Baseline

- A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures
  - ✓ IEEE Std. No. 610.12-1990
- Before a software configuration item becomes a baseline, changes may be made quickly and informally

- However, once a baseline is established, changes can be made, but a specific, formal procedure must be applied to evaluate and verify each change request
- In the context of software engineering, a baseline is a milestone in the development of software that is marked by the delivery of one or more software configuration items and approval of these software configuration items is obtained through a formal technical review or inspection
- Typical, work products that are base-lined are
- System specification
  - ✓ Software requirements
  - ✓ Design specification
  - ✓ Source code
  - ✓ Test plans/procedures/data
  - ✓ Operational system

SCM Questions

1. How does an organization identify and manage the many existing versions of a configuration item in a manner that will enable change to be accommodated efficiently?
2. How an organization control changes does before and after software is released to a customer?
3. Who has the responsibility for approving and ranking changes?
4. How can we ensure that changes have been made properly?
5. What mechanism is used to apprise others of changes that are made?

SCM Functions

1. Identification of software configuration items
2. Version control
3. Change control
4. Configuration audit
5. Status accounting/reporting

SCM Function 1: Identification

- To control and manage software configuration items, each item must be separately named or numbered
- The identification scheme is documented in the software configuration management plan
- The unique identification of each SCI helps in organization and retrieval of configuration items

SCM Function 2: Version Control

- Version control combines procedures and tools to manage different versions of configuration items that are created during the software process
- The naming scheme for SCIs should incorporate the version number
- Configuration management allows a user to specify alternative configurations of the software system through the selection of appropriate versions
- This is supported by associating attributes with each software version, and then allowing a configuration to be specified [and constructed] by describing the set of attributes

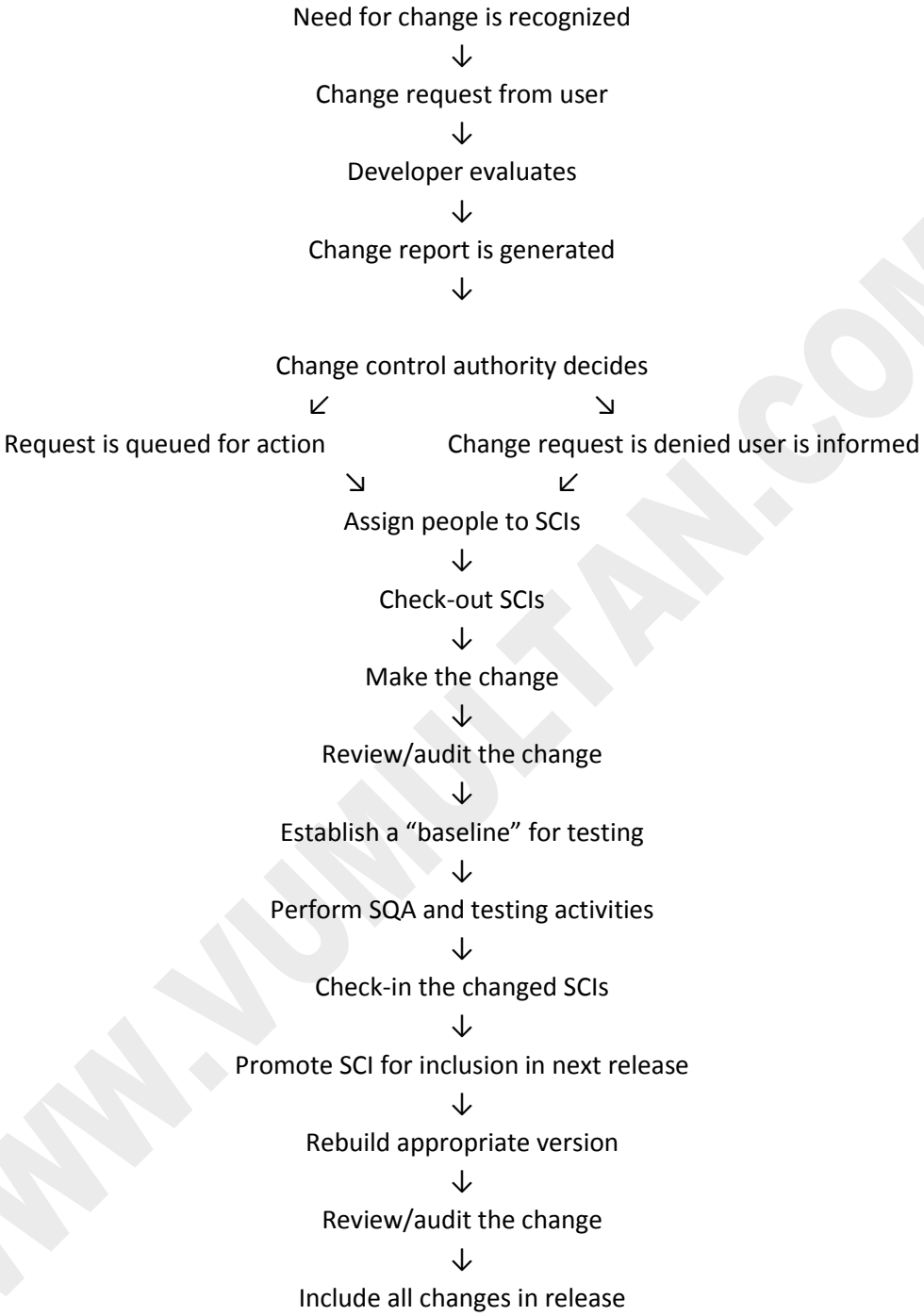
SCM Function 3: Change Control

- Change control is vital
- Too much change control and we create problems. Too little, and we create other problems (elaborate this point more)
- For large projects, uncontrolled change rapidly leads to chaos
- For medium to large projects, change control combines human procedures and automated tools to provide a mechanism for the control of change

Change Control Process

- Proposed changes to software work-products are reviewed, then subjected to the agreement of project participants, and finally incorporated into the currently approved software configuration
- This requires that the separate authority, which reviews and approves all change requests be established for every project from among the project participants. This authority is known as change control authority or change control board
- A CCA/CCB plays an active role in the project level change control and formal change control activities
- Typically, a CCA consists of representatives from software, hardware, database engineering, support, and marketing, etc., depending on the size and nature of the project
- For every change request, the change control authority/board assesses the
  - ✓ Technical merit
  - ✓ Potential side effects
  - ✓ Overall impact on other configuration items and system functions
  - ✓ Projected cost of the change
- The CCA/CCB has the authority to approve or reject a change request
- It can delay the request for consideration and inclusion in the next iteration also
- For every approved change request, an engineering change order (or ECO) is generated, which describes
  - ✓ The change to be made
  - ✓ The constraints that must be respected
  - ✓ The criteria of review and audit
- As you can notice, that formal change control process is very elaborate and comprehensive process
- The “check-in” and “check-out” activities implement two important elements of change control
  - ✓ access control
  - ✓ synchronization control

The Change Control Process



Access and Synchronization Control

- Access control governs which software engineers have the authority to access and modify a particular configuration item
- Synchronization control helps to ensure that parallel changes, performed by two different people, don't overwrite one another
- We need to implement both

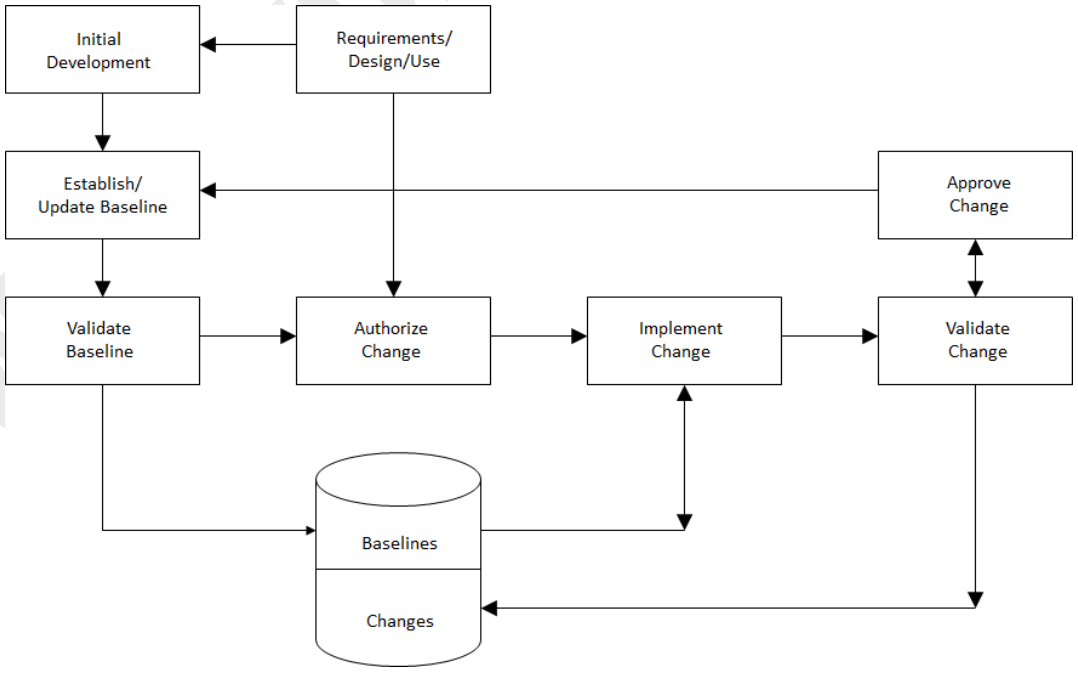
SCM Function 4: Configuration Audit

- A software configuration audit complements the formal technical reviews/inspections by assessing a configuration item for characteristics that are generally not considered during review
- The SCM audit is conducted by the quality assurance group
- Has the change specified in the ECO been made? Have any additional modifications been incorporated?
- Has a formal technical review been conducted to assess technical correctness?
- Has the software process been followed and have software engineering standards been properly applied?
- Has the change been "highlighted" in the SCI? Have the change date and author been specified? Do the attributes of the configuration object reflect the change?
- Have SCM procedures for noting the change, recording it, and reporting it been followed?
- Have all related SCIs been properly updated?

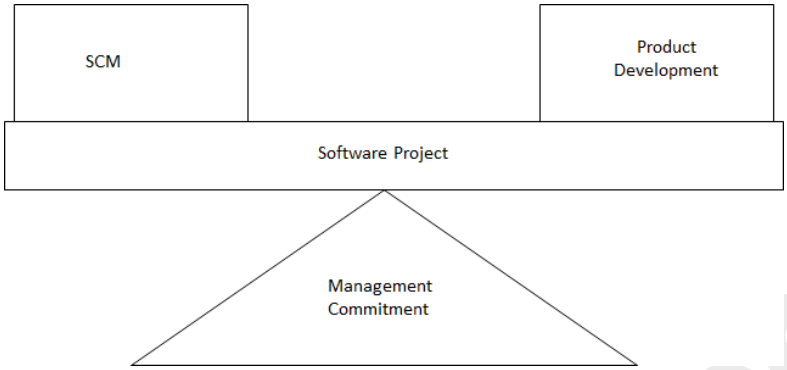
SCM Function 5: Status Accounting/Reporting

- The status accounting function provides a corporate memory of project events that supports accomplishment of other configuration management items
  - ✓ What happened?
  - ✓ Who did it?
  - ✓ When did it happen?
  - ✓ What else will be affected?

Software Configuration Management Overview



Management Commitment for SCM



Summary

- Software configuration management is an umbrella activity that is applied throughout the software process
- SCM identifies, controls, audits, and reports modifications that invariably occur while software is being developed and after it has been released
- SCM saves a project from total chaos

References

- Software Engineering: A Practitioner's Approach by Roger S. Pressman (Chapter 9.1 – 9.9)
- A Handbook of Software Quality Assurance edited by G. Gordon Schulmeyer and James L. McManus, PH, 1999, (Chapter 10.1-10.2)
- Managing the Software Process by Watts S. Humphrey, AW, 1989, page 177 (figure 7.2 SCM Overview)

Lecture 4
Introduction to Project Planning

Software Project Planning

- Software development can be exceedingly complex and it needs to be planned in such a way that all required resources are made available at the right time for the necessary duration

Logic of Software Project Planning

- While requirements are initially vague and incomplete, a quality program can only be built from an accurate and precise understanding of the users' needs. The project plan thus starts by mapping the route from vague and incorrect requirements to accurate and precise ones
- A conceptual design is then developed as a basis for planning. This initial structure must be produced with care since it generally defines the breakdown of the product into units, the allocation of functions to these units, and the relationships among them. Since this provides the organizational framework for planning and implementing the work, it is almost impossible to recover from a poor conceptual design
- At each subsequent requirements refinement, resource projections, size estimates, and schedules are also refined
- When requirements are sufficiently clear, a detailed design and implementation strategy is developed and incorporated in the plan
- As various parts of the project become sufficiently well understood, implementation details are established and documented in further plan refinements
- Throughout this cycle, the plan provides the framework for negotiating the time and resources to do the job

Goal of Project Planning

- The overall goal of project planning is to establish a pragmatic strategy for controlling, tracking, and monitoring a complex technical project. Why?
- So the end result gets done on time, with quality

Objectives of Project Planning

- The objective of software project planning is to provide a framework that enables the manager to make reasonable estimates of resources, cost, and schedule
- These estimates are made within a limited time frame at the beginning of a software project and should be updated regularly as the project progresses
- Estimates should attempt to define best case and worst case scenarios so that the project outcomes can be bounded

Steps of Project Planning

- Software scope
- Estimation
- Risk
- Schedule
- Control strategy

Software Scope

- Understand the problem and the work that must be done – in a nutshell
- Software scope describes the data and control to be processed, function, performance, constraints, interfaces, and reliability
- Project scope must be unambiguous and understandable at the management and technical levels
- A statement of software scope must be bounded – in other words
- At the beginning of a project, things are very hazy and nothing is clear
- Good and open communication is required between developers and customer to define the scope of the project
- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefit of a successful solution?
- Is there another source for the solution?

To Understand Scope

- Understand the customer needs
- Understand the business context
- Understand the project boundaries
- Understand the customer's motivation
- Understand the likely paths for change
- Understand that even when you understand, nothing is guaranteed

Feasibility

- Once scope has been identified (with the concurrence of the customer), it is reasonable to ask: "Can we build software to meet this scope? Is the project feasible?"
- Often times this question is overlooked and developers face serious problems because of that

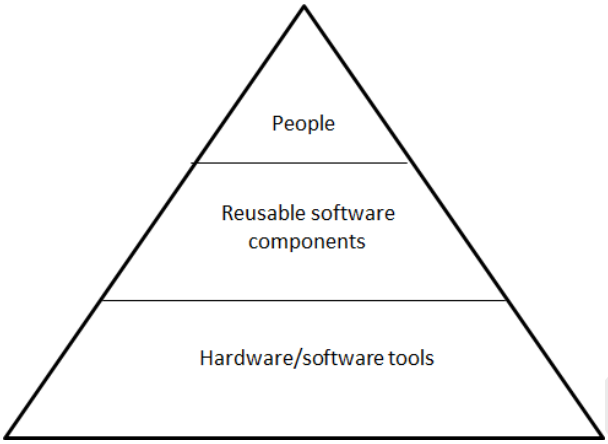
Estimation

- How much effort?
- How much time?
- Resource and time estimation

Cost Estimation

- Project scope must be explicitly defined
- Task and/or functional decomposition is necessary
- Historical measures (metrics) are very helpful
- At least two different techniques should be used
- Remember that uncertainty is inherent

Project Resources

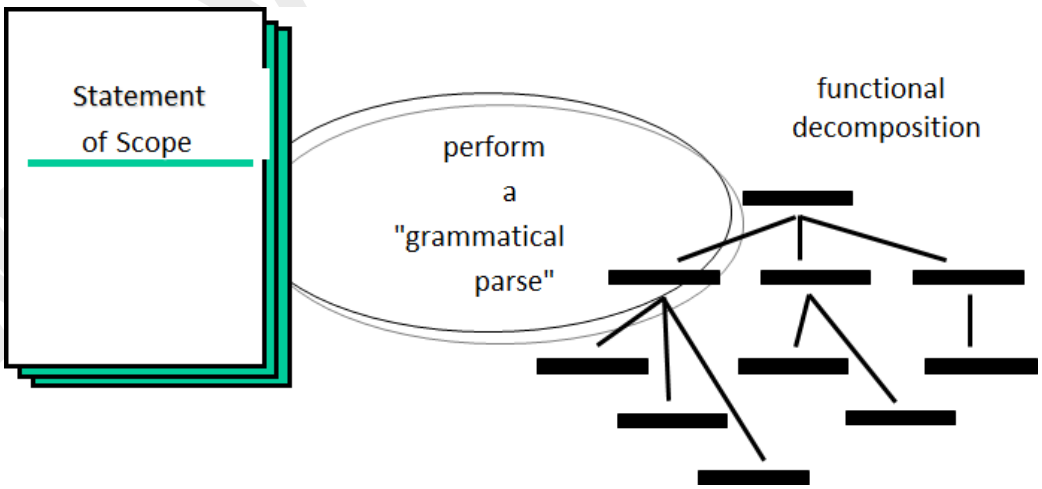


- Discussion on people, reusable software components, and hardware/software tools
- Software cost and effort estimation will never be an exact science, as too many variables – human, technical, environmental, political – can affect the ultimate cost of software and effort applied to develop it

Options for Reliable Cost and Effort Estimates

- Delay estimation until late in the project
- Base estimates on similar projects that have already been completed
- Use relatively simple decomposition techniques to generate project cost and effort estimates
- Use one or more empirical models for software cost and effort estimation

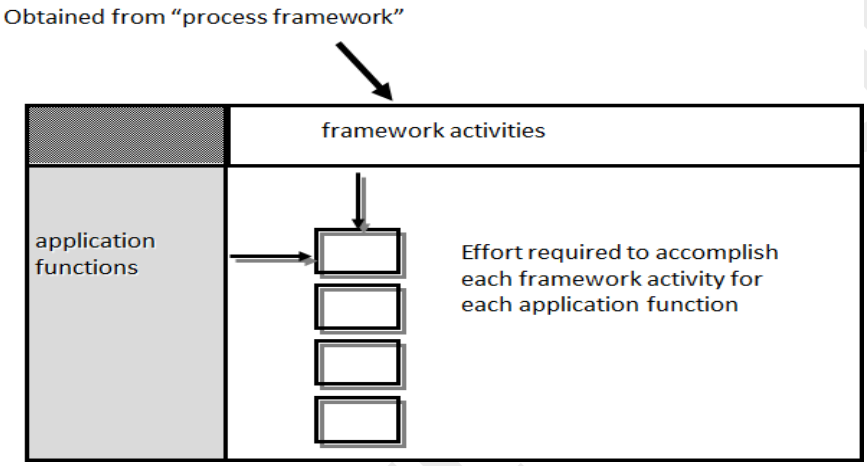
Functional decomposition – divide and conquer



Estimation Techniques

- Past similar projects experience
- Conventional estimation techniques
  - ✓ Task breakdown and effort estimates
  - ✓ Size (e.g., FP) estimates
  - ✓ Tools (e.g., Checkpoint)

Creating a Task Matrix



Estimation based on Empirical Models

- $d = f(v_i)$
- d is one of a number of estimated values (e.g., effort, cost, project duration) and v_i are selected independent parameters (e.g., estimated LOC or FP)

COCOMO model

- The COCOMO model

Risk

- What can go wrong?
- How can we avoid it?
- What can we do about it?

Schedule

- How do we allocate resources along the time line?
- What are the milestones?

Control Strategy

- How do we control quality?
- How do we control change?

References

- Software Engineering 5th Edition by Roger Pressman, Chapter 5

Lecture 5
Process Models

Why Software Process?

- Software development can be exceedingly complex and there are often many alternative ways to perform the various tasks
- There is a need for a organized set of activities, which when performed accurately, will result in an orderly development effort

Software Process

- A defined process can help guide the software professionals through tasks, which can be performed in a number of ways, in an orderly way
- This results in the establishment of a process definition they can understand
- This enable software professionals to better understand
  - ✓ What they should do
  - ✓ What they expect from their co-workers, and what they are expected to provide in return
- This allows them to focus on doing their jobs; contract between co-workers
- Operational definitions are "something everyone can communicate about and work to" – Deming
- They provides organizations with a consistent working framework while permitting individual adjustments to unique needs

Software Engineering

- Software engineering is not a routine activity that can be structured and regimented like repetitive manufacturing or clerical procedure
- It is an intellectual process that must dynamically adjust to the creative needs of the professionals and their tasks; trade-off is needed

Process Trade-off Factors

- Since software projects have differences, their software engineering processes must have differences as well
- In the absence of a universal software engineering process, organizations and projects must define processes that meet their own unique needs
- The process used for a given project must consider the experience level of the members, current product status and the available tools and facilities

Process Standardization

- Elaborate on the need to have standardization and the need to have individual creativity
- Example: music, artistic painting, electrical circuits, etc.
- Let's see what are the compelling reasons for process standardization

Process Standards

- Process standardization helps to reduce the problems of training, review, and tool support
- With standard methods, each project's experiences can contribute to overall process improvement
- Process standards provide the basis for process and quality measurements
- Since process definitions take time and effort to produce, it is impractical to produce new ones for each project

Customization and Standardization

- The conflicting needs for customization and standardization can often be resolved by establishing a process architecture, which consists of a standard set of unit or "kernel" process steps with rules for describing and relating them
- Customization is achieved through appropriate interconnections of these standard elements into tailored process models

Software

- The term software refers to a program and all the associated information and materials needed to support its installation, operation, repair, and enhancement

Definitions

- Software Engineering Process
  - ✓ The total set of software engineering activities needed to transform a user's requirements into software
- Software Process Architecture
  - ✓ A framework within which project-specific software processes are defined
- Software Process Model
  - ✓ One specific embodiment of a software process architecture

Software Processes

- Software processes can be categorized according to the level of details organized in these processes
- Some can be high-level (with little detail), while others can be low-level processes with lots of details; still others may fall in between
- Careful analysis results in …

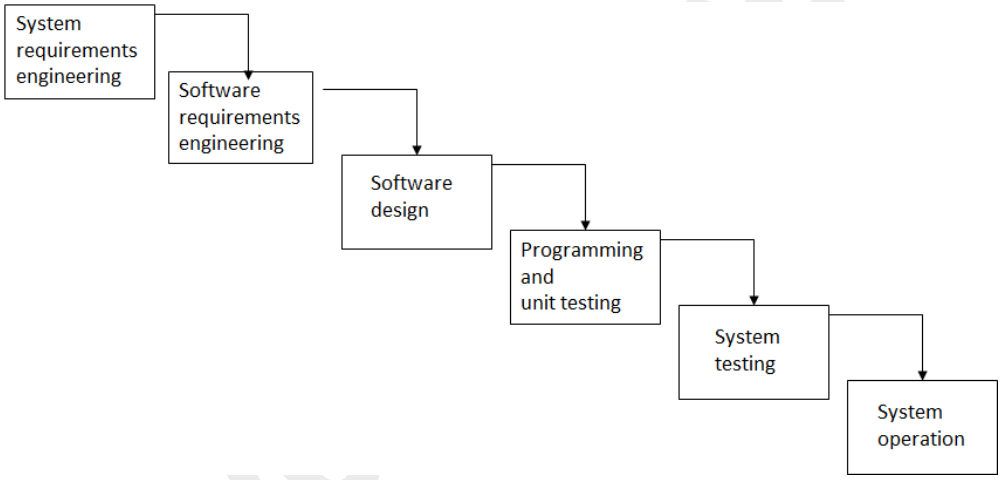
Levels of Software Process Models

- Universal or U Process Models
- Atomic or A Process Models
- Worldly or W Process Models

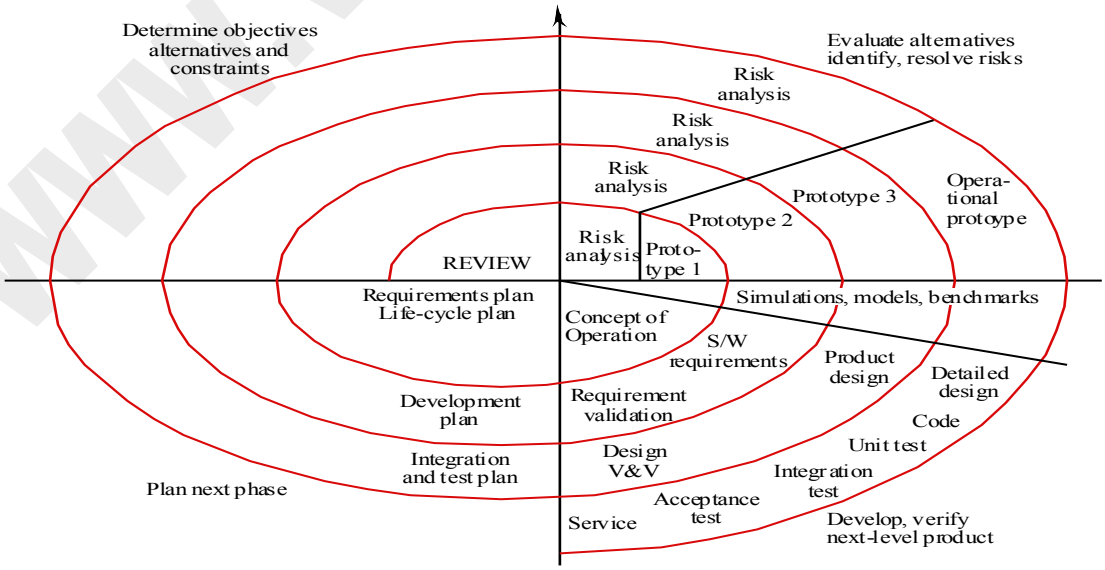
Universal or U Process Models

- U process models provide high-level overview
- Traditional Waterfall, Spiral, and other later process models fall in this category
- Typically task-oriented
- The fundamental problem with simplistic U –level software process models is that they do not accurately represent what is really done
- The reason is that traditional process models are extremely sensitive to task sequence; consequently, simple adjustments can destroy the entire framework
- Take the example of the waterfall model, which has been used in the software industry for a long time
- Provides a very simplified view of activities, which can lead to inaccurate model development

A Version of Waterfall Model



Spiral model of the software process



Problems with Universal Model

- This results from an over-emphasis on modeling tasks
- It limits human flexibility and tends to arbitrarily impose rigidity
- It becomes progressively less helpful as the number of possible task sequences increases
- The model becomes very difficult if not impossible to understand
- When such process models are used to guide the process automation, project management, or contract administration, the resulting process rigidity can cause serious problems
- The question, therefore, is not "what is the right way to model the process?" but "what is the most appropriate way to model this process for this purpose?"

Worldly or W Process Models

- W process models are most useful to practicing software engineers
- It guides the sequence of their working tasks and defines task prerequisites and results
- They look like procedures
- They specify who does what when
- Where appropriate, they reference the A level that specifies standard task definitions or tool usage
- For each task, W models define the anticipated results, the appropriate measures, and the key checkpoints

Atomic or A Process Models

- "A" process models are enormously detailed
- They are used to automate specific process activity or use a standardized method or procedure to guide execution of a task
- Precise data definitions, algorithmic specifications, information flows, and user procedures are essential at this level
- The amount of details to be included in such models must be determined by their use
- For example, an experienced developer who is repeating known manual tasks will not need as detailed a standard as a new trainee
- When the task is to be automated, however, a great deal of detail is generally required
- Atomic process definitions are often embodied in process standards and conventions, which can be treated as process abstractions in the higher level "W" or "U" process models

Relations of Software Process Models

- U process models embody policies
  - ✓ What are policies?
- W process models embody procedures
  - ✓ What are procedures?
- A process models embody standards and/or tools
  - ✓ What are standards and/or tools?

Policies

- Policies establish a high level framework and set of principles that guide the overall behavior of organizations
- They are particularly helpful in unanticipated circumstances where no precedents have been established
- Some policy-level statements might be:

Policy-level Statements

- All work will be subjected to an inspection before it is incorporated in a baseline
- The quality of each product, at the time of new shipment, shall be better than its predecessor or leading competitor
- All commitments for software cost or delivery will be supported by a documented and approved software engineering plan
- Quality Assurance (QA) will review the software development process to assure senior management that the work is done according to established standards and procedures and in conformance with the intent of the stated policies

"W"-level, Procedures

- At the "W"-level, procedures are established to implement the policies
- This W-level process model refers to any available Atomic-level standards that define precisely how tasks are to be performed
- At the W-level, for example, a procedure might define the points at which Quality Assurance reviews are to be conducted and how resulting issues are to be handled
- This might specify what percent of work is to be reviewed, how statistical samples are to be selected, and whether, when, how SQA independently tests or monitors the software engineering work as it is being done

Atomic level standards

- Atomic level standards serve as the basis for directing the work and for the SQA review
- For example, a code inspection standard would specify what code is to be reviewed, when, the methods to be used, the reports to be produced, and the acceptable performance limits
- The developers would use this standard to guide their actions
- The SQA people would review their actions and work products against this standard

Critical Software Process Issues

- Quality
- Product technology
- Requirements instability
  - ✓ Unknown requirements
  - ✓ Unstable requirements
  - ✓ Misunderstood requirements
- Complexity

References

- Managing the Software Process by Watts S. Humphrey (Chapter 13.1-13.6)

Lecture 6
ETVX: Process Modeling Technique

Critical Software Process Issues

- Quality
- Product technology
- Requirements instability
  - ✓ Unknown requirements
  - ✓ Unstable requirements
  - ✓ Misunderstood requirements
- Complexity

Critical Software Process Issues

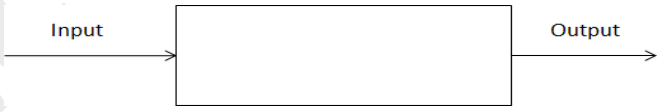
- Organizations that face the issues of quality, product technology, requirements stability, and/or complexity need to define ways to address them
- A process architecture permits these organizations to represent and manipulate the process at the U level and then selectively to refine it to the W and A level

Process Framework

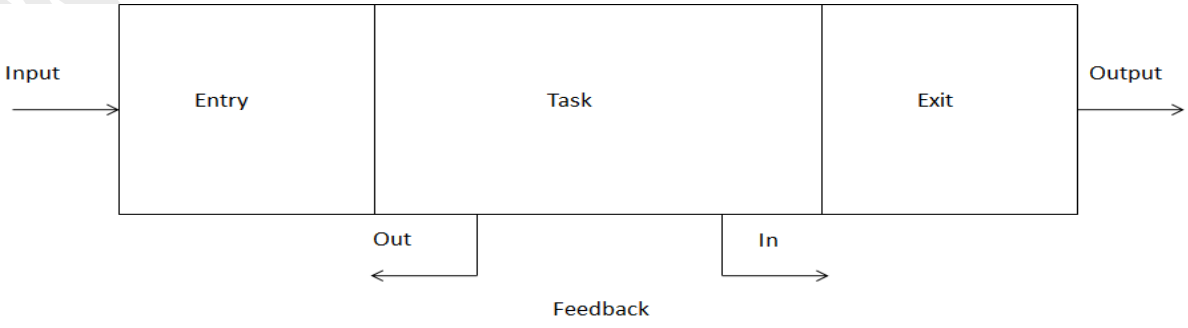
- The universal, worldly, and atomic levels of process models can be presented in the process framework – in other words
- Thus, the framework would give established policies, procedures and standards
- The architectural framework of a process helps in providing a definition for
- basic elements
- how they interact
- how they are decomposed into increasing level of detail

A Unit Cell

- The basic element of the process architecture is the unit cell
- Each cell is defined to accomplish a specified task and is uniquely identified



Detailed description of Unit Cell



Entry Conditions of a Unit Cell

- Each cell has required entry conditions specified for task initiation that include the inputs (one or more with their sources)
- Task standards, procedures, methods, responsibilities, and required measures are also defined

Exit Conditions of a Unit Cell

- Exit conditions define the results produced, their level of validation, and any post-task conditions
- Cell feedback refers to any data provided to or received from other stages in the process

Basic Unit Cell Specification

Specification	Activities
Entry	Conditions to be met before task initiation
Exit	Results produced
Feedback	In: Feedback from other stages Out: Feedback to other stages
Task	What is to be done (by whom, how, and when, including standards, procedures, and responsibilities
Measurements	Task (activities, resources, time), output (number, size, quality), and feedback (number, size, quality) measures

ETVX

- ETVX as a modeling language
- Introduced by IBM in 1980

Objectives of ETVX

- The objective of ETVX is to illustrate the effective process performance
- It can be applied to as low a level as required to control process
- A model developed in ETVX is expressed as a set of interconnected activities each of which has four sets of attributes

ETVX Representation

Entry (E)	The Entry section defines the entry criteria that must be satisfied for the process to be initiated, and list the work products that must be available as inputs to the process
Task (T)	The Entry section defines the entry criteria that must be satisfied for the process to be initiated, and list the work products that must be available as inputs to the process
Verification (V)	The verification section defines steps for verifying that the process has been properly executed, and that the associated work products meet project objectives
Exit (X)	The Exit section defines the exit criteria that must be satisfied for the process to be terminated. The exit criteria usually define completion and verification work products, in terms of qualitative aspects of the products

ETVX model

- An ETVX model indicates the relationship and flow among the four aspects of an activity and between activities
- The notion of formal entry, exit, and criteria go back to the evolution of the waterfall development process
- The idea is that every process step, inspection, function test, or software design has a precise entry and exit criteria

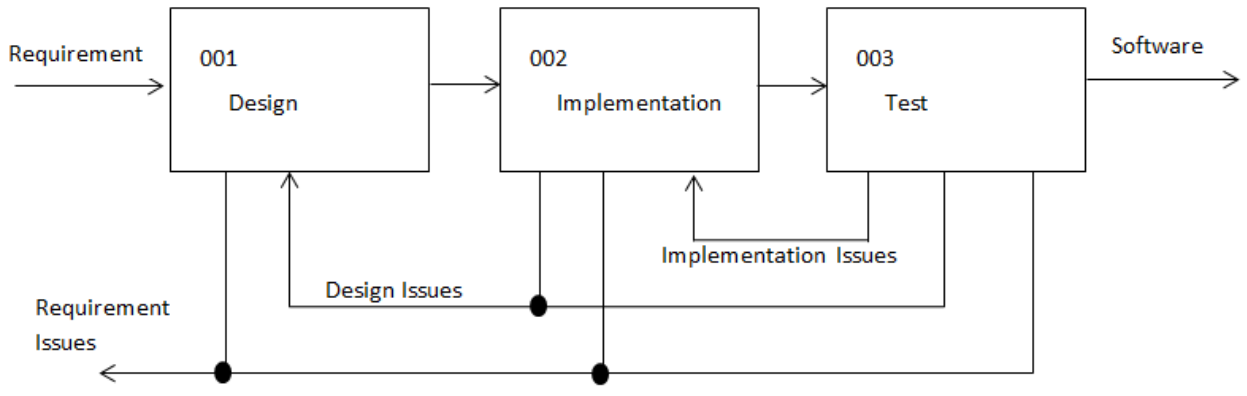
Characteristics of ETVX

- ETVX is a task-based model
- Each task must be explicitly defined
- The basic cells can be combined to create process
- "bottom-up" approach (A level to W and U levels)

Example of ETVX

- The generic activities of different process models from the U-level's perspective can be listed as "design", "implementation", and "test"
- When these activities are broken into more detail, however, significant differences show up.
- Even with all these variations at the W level, however, many software activities can relatively standardized across different projects
- It is thus possible to establish some basic process cells that can be interconnected in different ways to meet project-unique needs
- The detailed structures of these standard cells are then further defined by A-level models as needed
- Let's look at the "design", "implementation", and "test" activities in any process model using the ETVX technique

U-Level Developmental Process



Unit Cell Definitions for Process Model

Cell	001	002	003
Entry	Approved requirements, changes and development plan	Inspected and approved design and changes	Inspected and approved code and changes
Exit	Inspected and approved design changes	Inspected and approved code and changes	Inspected, tested and approved software
Feedback in	Design Issues	Implementation Issues	
Feedback out	Requirements Issues	Requirements and Design Issues	Requirements, Design Issues, and Implementation Issues
Task	Design	Implementation, Inspection and Unit test	Testing: Integration, Function, System, and Acceptance
Measures	Resources, Product; Changes, Errors, Design, Document Pages	Resources, Product: Changes, Errors, Code, Document Pages	Resources, Product: Changes, Errors, Code, Document Pages, Test Suite

Lecture 7

Application of ETVX

- Let's use software inspection process as an example of the application of the ETVX model
- We'll consider only a selected set of activities from within the detailed inspection process

Agenda

- Preparation
- Inspection Meeting
- Rework
- Follow-up
- Data recording and reports

Preparation

Preparation: Entry Criteria

- The overview, if needed, has been satisfactorily completed
- Any open issues identified for the overview have been closed and addressed in the work product or are documented as open issues and provided as ancillary material for the preparation
- Open issues not closed are documented for tracking within the change control system used by the project
- The producer determines that the work product is ready for inspection
- The work product has reached closure and the code complies with defined standards, style guides, and templates for format
- All necessary ancillary material have been made available well in advance
- The work product includes all base-lined function and approved changes for this planned work product completion date
- The amount of time needed for preparation has been confirmed with the inspectors and is available to them
- Predecessor and dependent work products are available, have been inspected, and meet exit criteria
- The moderator and producer have defined the coverage of material to be inspected
- The work products allow easy identification of defects by location in the material
- The moderator agrees that the work product is inspectable

Preparation: Tasks

- Each inspector uses the scheduled time to complete the preparation in a style and format they are comfortable with
- The material to be inspected is marked with questions, concerns, and possible defects, both major and minor, found during inspection
- The minor defects are either recorded on a separate sheet that will be delivered to the moderator at the start of the inspection meeting or they are clearly noted in the marked material that will be delivered to the moderator at the end of the inspection meeting. Each minor defect should be noted by location in the work product when using a minor list

Preparation: Validation/Verification

- The moderator uses the preparation entry criteria and procedure
- The moderator uses the minor defect information to determine if all inspectors have properly performed preparation
- The inspectors have confirmed that they have prepared
- The SQA group ensures that the moderator has used the preparation procedure and that the inspectors performed sufficient preparation. This can be done via audits of the process records or sampling of inspections
- Data gathered in this activity
  - ✓ How much time was spent in preparation
  - ✓ How long a period between notification of the inspection and the preparation
  - ✓ How many inspection meetings required rescheduling due to insufficient preparation
  - ✓ The number of major and minor defects found during preparation

Preparation: Exit Criteria

- Each inspector has completed sufficient preparation based on organization and project preparation time criteria
- Minor defect inputs are complete
- Preparation notes are recorded on the work product materials or defect lists

Inspection Meeting

Inspection Meeting: Entry Criteria

- The inspection team members are sufficiently present in number and role assignments
- Inspection materials were available for preparation with sufficient time for study and review before the inspection meeting, including necessary reference material
- Inspectors have adequately prepared
- Inspectors have submitted their minor defects list at the start of the meeting or have marked the work products that will be provided at the end of the meeting
- Scope of the inspection meeting has been defined
- Recorder and a data recording system are available
- Other roles; e.g., reader have been assigned
- The producer has identified any new potential problem areas

Inspection Meeting: Tasks

- Brief introduction (moderator)
- Preparedness check (moderator)
- Read the work product (reader)
- Identify defects (inspectors)
- Record defects (recorder)
- Determine disposition of material (inspection team)
  - ✓ Accept the material

- ✓ Accept the material after verification with follow-up inspector
- ✓ Request the work product to be re-inspected after rework
- ✓ Recommend re-engineering of the work product followed by a new inspection

Inspection Meeting: Validation/Verification

- The moderator, using the inspection meeting entry criteria and procedure, determines if the team has properly performed the inspection
- The inspectors participated in an effective meeting
- The SQA group ensures that inspection meeting procedure and that the inspectors performed sufficient preparation. This can be done via audits of the process records or sampling of inspections
- Data gathered during this activity
  - ✓ How much time was spent in the inspection meeting
  - ✓ How long a period between the preparation and the inspection meeting
  - ✓ How many inspection meetings required rescheduling due to insufficient preparation
  - ✓ How many inspections required re-inspection
  - ✓ How many defects were found
  - ✓ How long the meeting took
  - ✓ How many inspectors were in attendance

Inspection Meeting: Exit Criteria

- The inspection materials have been inspected and coverage of the work product is completed as planned
- The inspection results fall within expected tolerance of performance for
  - ✓ Time spent during preparation
  - ✓ Time spent at the inspection meeting
  - ✓ Defect density
- The defects and the conduct of the inspection have been recorded and the team concurs with the contents
- Open issues have been recorded for follow-up during rework
- The moderator or a designee has been appointed to perform follow-up with the producer
- Data is available to update the process data base
- Any associated deviations or risks are noted
- Decisions to re-inspect or not have been reviewed against criteria
- Decision on re-engineering has been addressed
- Process defects have been recorded, as appropriate, as well as product defects
- The locations of the defects of the inspected work product are clearly noted to facilitate repair
- A decision is taken on the timeframe by which defect repairs and open issues will be resolved
- The inspection satisfies the criteria to be indicated as performed

Rework

Rework: Entry Criteria

- The list of defects and open issues is provided to the producer for resolution
- The moderator or someone assigned meets with the producer to review rework and open issues
- The inspection report is completed, is on file, and available

Rework: Tasks

- The producer repairs accepted defects identified during the inspection meeting
- The producer resolves any open issues
- The moderator meets with the producer to discuss resolutions of open issues
- Change requests are written for any open issues or defects not resolved during the rework activity
- Either the minor defect list or marked work products with minor defects noted are used to repair the minor defects

Rework: Validation/Verification

- The follow-up activity is scheduled; where the rework will be verified by the moderator or assigned designee
- SQA has reviewed sample results of this activity in the project
- Data gathered during this activity
  - ✓ How much time was spent in rework
  - ✓ How many open issues were resolved and accepted as defects
  - ✓ How many open issues became submitted change requests

Rework: Exit Criteria

- The producer resolves all defects and open issues
- Inspected work product materials are updated to account for repairs

Follow-Up

Follow-up: Entry Criteria

- Rework of defects has been completed; i.e., fixed or identified with a decision to not fix
- The producer has completed the rework for defects and open issues resolved to be defects
- Change requests are written for any defects or open issues not resolved
- The moderator concurs with the producer's decisions on defects, open issues, and change requests

Follow-up: Tasks

- The moderator and producer discuss and agree on compliance with respect to defects and open issues
- In case of disagreement, the issue would be resolved by the project lead

- The producer updates the work product to reflect the fixes to defects found and open issues accepted as defects
- The producer writes any change requests that may be required
- The moderator completes the inspection report and marks the inspection as closed

Follow-up: Validation/Verification

- The moderator concurs with the defect repairs and open issue closures
- The producer reviews the final inspection report
- SQA group reviews the final inspection report
- Data gathered during this activity
  - ✓ How much time was spent in follow-up
  - ✓ How many open issues were disputed

Follow-up: Exit Criteria

- Any change requests resulting from unresolved open issues have been submitted to the change approval process for handling
- The inspection report is completed and the producer agrees
- If necessary, a re-inspection is scheduled
- If necessary, issues are escalated to the project lead for resolution
- The inspection is noted as closed

Data Recording and Reports

- To record the data about the defects and conduct of the inspection
- This activity is held concurrently with other activities, including at the end of the inspection process

Data Recording and Reports: Entry Criteria

- The optional overview meeting was held
- The inspection meeting was held
- The optional analysis meeting was held

Data Recording and Reports: Tasks

- Record data from overview, if held
- Record data at the inspection meeting, including preparation data
- Record data at the optional analysis meeting
- Record data during the follow-up activity, including sign-off to close the inspection

Data Recording and Reports: Validation/Verification

- The inspection verifies the data at the end of the inspection meeting and optional analysis meeting
- SQA review sampled reports

- The producer reviews the report completed by the moderator
- Data should be considered for this activity; e.g., how much effort is used for recording and reporting

Data Recording and Reports: Exit Criteria

- The data are complete and agreed to by the inspection meeting and analysis meeting participants
- The data gathered during the follow-up activity are complete and agreed to by the producer and moderator

Lecture 8
IDEF0: Process Modeling Technique

IDEF0

- Integration Definition for Function Modeling is a function modeling methodology for describing manufacturing functions
- This functional modeling language is used for the analysis, development, reengineering, and integration of information systems; business processes; or software engineering analysis
- IDEF0 is part of the IDEF family of modeling languages in the field of software engineering, and is built on the functional modeling language Structured Analysis and Design Technique (SADT)
- The IDEF0 Functional Modeling method is designed to model the decisions, actions, and activities of an organization or system
- In its original form, IDEF0 includes both a definition of a graphical modeling language (syntax and semantics) and a description of a comprehensive methodology for developing models

- **SADT** was designed by Douglas T. Ross and SofTech Inc in the 1970s
- Purpose of SADT by US Air Force
  - ✓ "to develop a function model method for analyzing and communicating the functional perspective of a system
  - ✓ IDEF0 should assist in organizing system analysis and promote effective communication between the analyst and the customer through simplified graphical devices"

- IDEF0 is used to show data flow, system control, and the functional flow of life cycle processes
- IDEF0 is capable of graphically representing a wide variety of business, manufacturing and other types of enterprise operations to any level of detail
- It provides rigorous and precise description, and promotes consistency of usage and interpretation
- It is well-tested and proven through many years of use by government and private industry
- It can be generated by a variety of computer graphics tools
- Numerous commercial products specifically support development and analysis of IDEF0 diagrams and models

Function Model

- A function model is a structured representation of the functions, activities or processes within the modeled system or subject area
  - ✓ IDEF0 describes function models
  - ✓ IDEF1 describes information models
  - ✓ IDEF2 describes dynamic models

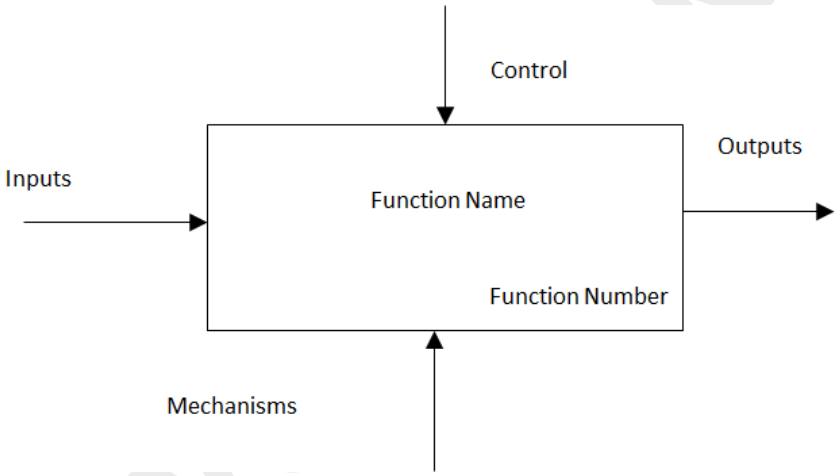
IDEF0 Approach

- IDEF0 may be used to model a wide variety of automated and non-automated systems
- For new systems, it may be used first to define the requirements and specify the functions, and then to design an implementation that meets the requirements and performs the functions
- For existing systems, IDEF0 can be used to analyze the functions the system performs and to record the mechanisms (means) by which these are done
- The result of applying IDEF0 to a system is a model that consists of a hierarchical series of diagrams, text, and glossary cross-referenced to each other
- The two primary modeling components are functions (represented on a diagram by boxes) and the data and objects that inter-relate those functions (represented by arrows).[

ICOM

- Inputs Control Outputs and Mechanisms

IDEF0



IDEF0 Description

- Each activity is described by a verb based label placed in a box
- Inputs are shown as arrows entering the left side of the activity box
- Output are shown as exiting arrows on the right side of the box
- Controls are displayed as arrows entering the top of the box
- Mechanisms are displayed as arrows entering from the bottom of the box
- Inputs, Controls, Outputs, and Mechanisms are all referred to as concepts
- **I**nputs
  - ✓ Entities which are transformed or consumed by the function to produce outputs
- Outputs
  - ✓ Outputs are the data or objects produced by the function
- **C**ontrols
  - ✓ Controls specify the conditions required for the function to produce correct outputs

- Mechanisms
  - ✓ Anything which is used to help perform the activity. Upward pointing arrows identify some of the means that support the execution of the function
  - ✓ Call (downward) arrows enable the sharing of detail between models or between portions of the same model

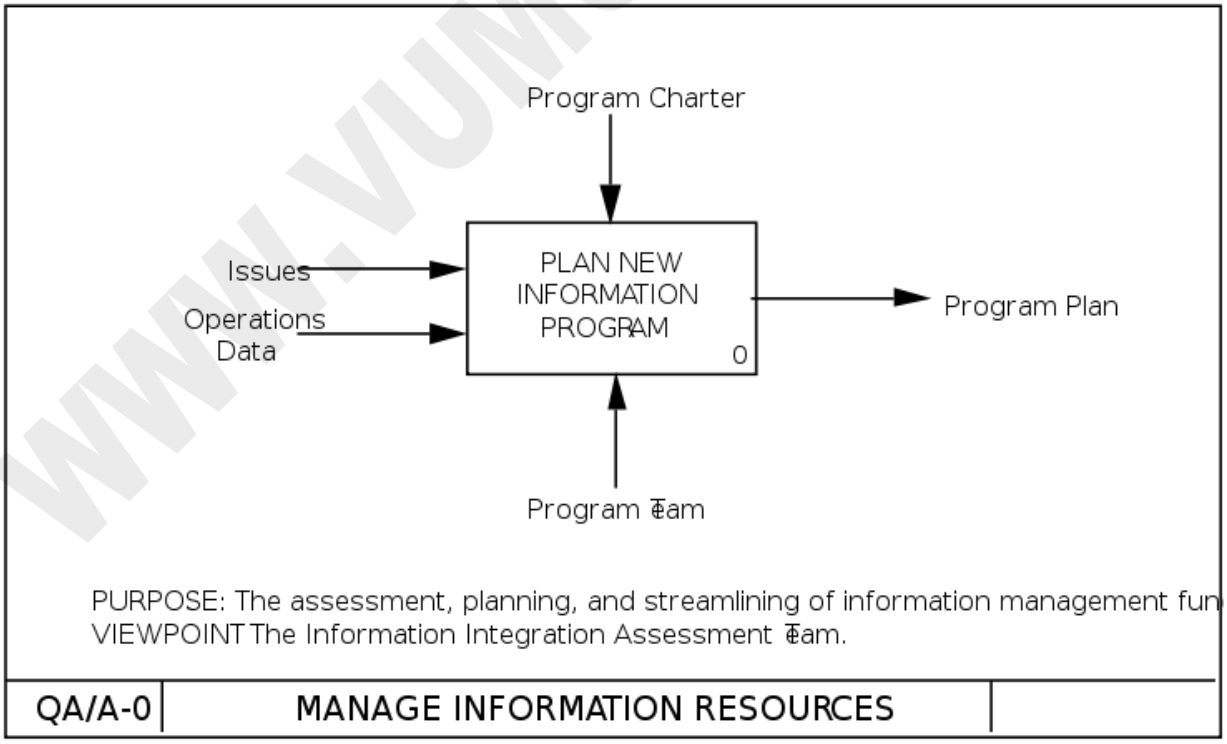
IDEF0 Diagrammatic Notation

- IDEF0 is a model that consists of a hierarchical series of diagrams, text, and glossary cross referenced to each other. The two primary modeling components are:
  - ✓ functions (represented on a diagram by boxes)
  - ✓ data and objects that interrelate those functions (represented by arrows)

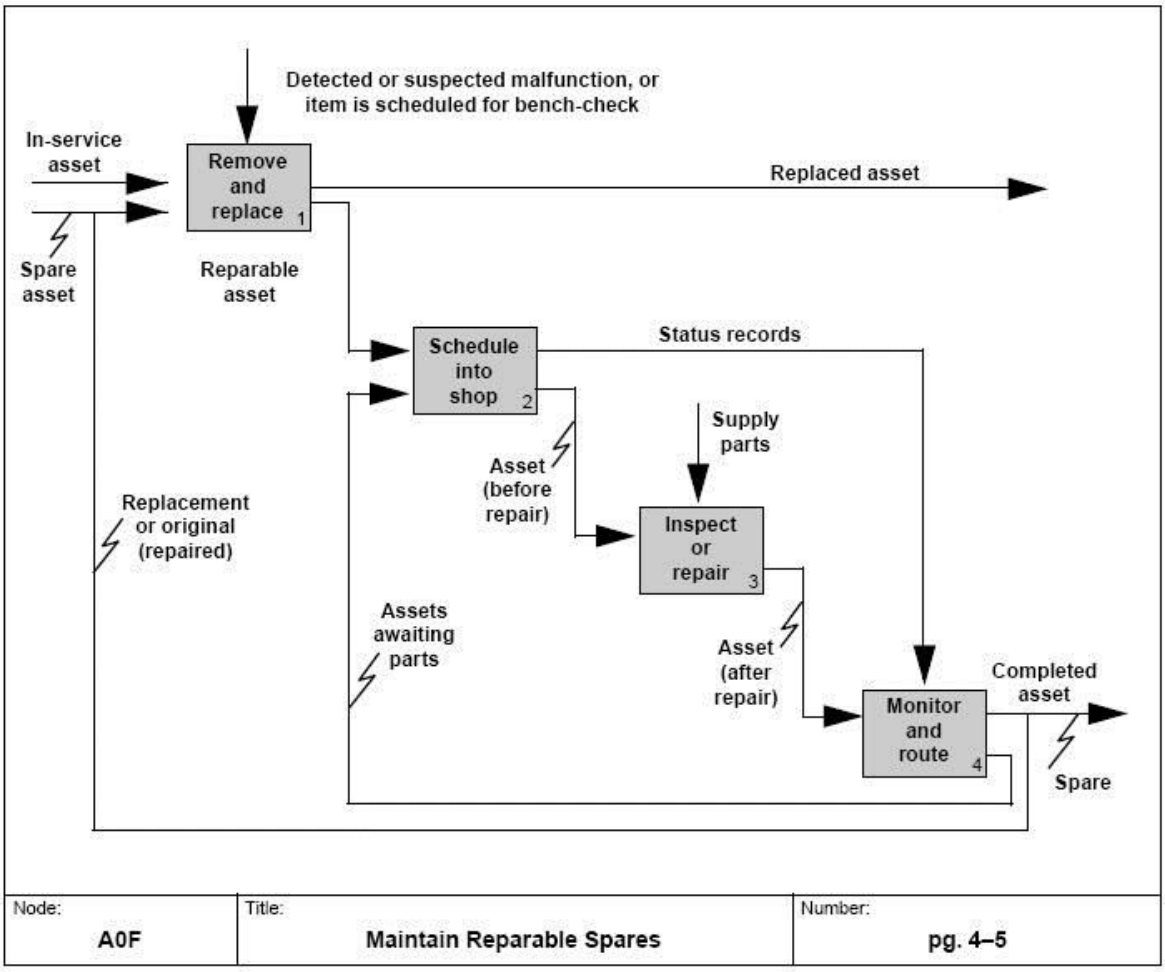
IDEF0 Process

- The IDEF0 process starts with the identification of the prime function to be decomposed
- This function is identified on a "Top Level Context Diagram," that defines the scope of the particular IDEF0 analysis
- This diagram is known as "A-0"
- A-0 diagram also sets the model scope or boundary and orientation
- Shall present brief statements specifying the model's viewpoint and purpose
- From this diagram lower-level diagrams are generated, called a "child" in IDEF0 terminology

IDEF0 Context Diagram



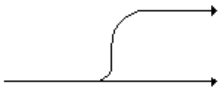
IDEF0 Child Diagram



IDEF0 Fundamentals

- Diagrams based on simple box and arrow graphics, arrows convey data or objects
- Text labels to describe boxes and arrows and glossary and text to define the precise meanings of diagram elements
- Box name shall be a verb or verb phrase, such as "Perform Inspection", arrows are nouns
- Gradual exposition of detail, with the major functions at the top and with successive levels of sub functions revealing well-bounded detail breakout.
- The limitation of detail to no more than six sub functions on each successive function.
- A "node chart" that provides a quick index for locating details within the hierarchic structure of diagrams.

IDEF0 - Rules

- There is always an A-0 context diagram, and its box number is always 0. This is a strict part of the standard that helps identify the overall description of the system.
- A non-context diagram has from 3 to 6 boxes. This helps us manage detail.
- Each box is numbered in its lower right corner, generally going upper left to lower right on the diagram. This gives us a consistent way to lay out the diagram.
- Arrows have horizontal and/or vertical segments, never diagonal. This makes the diagrams more readable.
- Each box has at least one control and output. This keeps us from using boxes with little purpose.
- Only one call arrow is allowed. This is another way to manage detail.
- Successive detail diagrams are numbered by "building up" diagram and box numbers. This is one of the most important concepts, that leads to a collection of easy-to-understand diagrams rather than a single confusing one.
- Unconnected ends of boundary arrows are identified by their ICOM codes. This helps identify arrows when moving from one diagram to another.
- Fork and/or 
- join arrows, 
- Rather than using parallel arrows for the same object. This reduces clutter, and reduces the likelihood that an arrow could be overlooked because it is in another part of the diagram.
- Arrow : A directed line, composed of one or more arrow segments, that models an open channel or conduit conveying data or objects from source to
- There are 4 arrow classes: Input Arrow, Output Arrow, Control Arrow, and Mechanism Arrow (includes Call Arrow). See Arrow Segment, Boundary Arrow, Internal Arrow
- Box : A rectangle, containing a name and number, used to represent a function
- Context : The immediate environment in which a function (or set of functions on a diagram) operates
- Decomposition : The partitioning of a modeled function into its component functions
- Fork: The junction at which an IDEF0 arrow segment (going from source to use) divides into two or more arrow segments. May denote unbundling of meaning
- Function : An activity, process, or transformation (modeled by an IDEF0 box) identified by a verb or verb phrase that describes what must be accomplished
- Join: The junction at which an IDEF0 arrow segment (going from source to use) merges with one or more other arrow segments to form a single arrow segment. May denote bundling of arrow segment meanings
- Node: A box from which child boxes originate; a parent box. See Node Index, Node Tree, Node Number, Node Reference, Diagram Node Number
- **IDEF0** - Function and process modeling
- Answers the question - **WHAT DO I DO?**

IDEF0 Benefits

- IDEF0 reveals redundant and non-value-added processes.
- IDEF0 documents the AS-IS-- the processes, the relationships between them, and the logical breakdown of those functions into their sub-processes and properties for baseline evaluation and further analysis
- IDEF0 begins the road map from the AS-IS to the TO-BE
- IDEF0 provides a means for communicating and presenting results
- IDEF0 establishes a forum and a structure for data gathering and knowledge acquisition
- IDEF0 identifies opportunities for improvements
- IDEF0 reveals data relationships and incongruities
- IDEF0 identifies and categorizes information entities which form the foundation for information modeling (IDEF1x)

IDEF is not a substitute for...

- Thinking...
- Intelligence...
- Experience,
- But it can expose their absence

Modeling Exercise

- Build system models for Requirements gathering, analysis, specification, and review process using:
  - ✓ IDEF0

References

- Managing the Software Process by Watts S. Humphrey (Chapter 13)

Lecture 9
Intro to Process Management

- Why is Important to Know Your Current Process?
- Pontificate on this point for a while

Technology Innovations

- Initially, new technology innovations and improvements (languages, compilers, CASE tools, and so forth) were viewed as the "silver bullet"
- Technology helped, but it did not provide the breakthrough desired

Deming's approach

- Other industries began to adopt the approach advocated by W. Edwards Deming and successfully implemented by many companies in Japan
- Deming's approach, greatly influenced by the work of Walter A. Shewhart, deals with the notions of process management and continual improvement
- Deming's approach contends that to be competitive, improve quality, and increase productivity, the following actions are required

- Focus on the processes that generate the products and services to improve quality and productivity. Consider the task of building the product or providing the service as a series of integrated and interconnected processes
- Ensure that the processes are properly supported
- Manage poorly behaving processes by fixing the process, not blaming the people
- Recognize that variation is present in all processes and that existence of variation is an opportunity for improvement. Improvement comes from reducing variation
- Take variation into account in the decision-making process. Management action uses data from the process to guide decisions

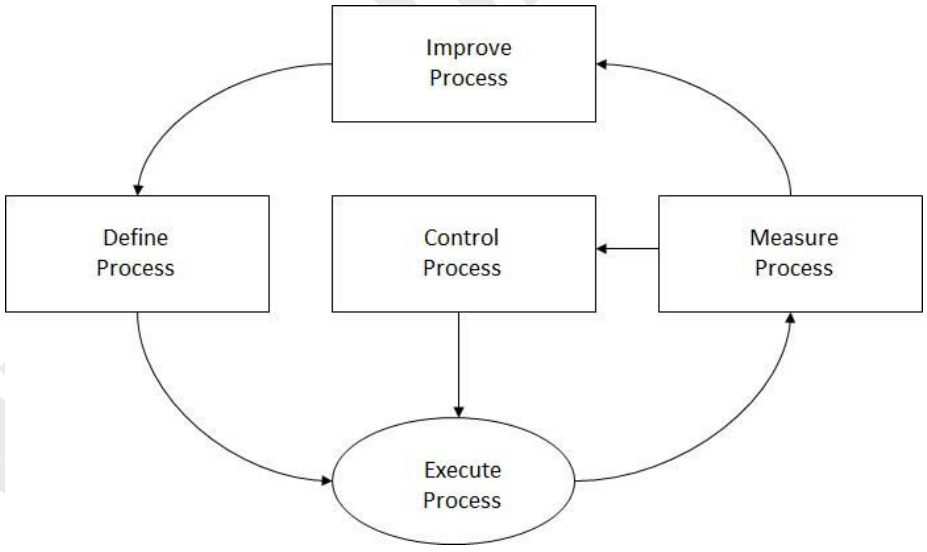
Process-Thinking Principles

- Requires unique relationship within the organization, involving its management and management philosophy, its people, and the management of the processes used to produce products and provide services
- How SW industry has adopted these principles on the next slide
- These concepts, methods, and practices embody a way of thinking, a way of acting, and a way of understanding the data generated by processes that collectively result in improved quality, increased productivity, and competitive products
- The acceptance and use of these process-thinking approach by much of the software industry has required us to consider ways to measure software processes that are responsive to questions relating to process performance (effectiveness, efficiency, timeliness, predictability, improvements, and product quality) in quantitative terms

- A process is a sequence of steps performed by people with the aid of tools and equipment to transform raw material into product
- Software process management is about successfully managing the work processes associated with developing, maintaining, and supporting software products and software-intensive systems
- This concept of process management is founded on the principles of statistical process control
- These principles hold that, by establishing and sustaining stable levels of variability, processes will yield predictable results
- Controlled processes are stable processes, and stable processes enable you to predict results
- This, in turn, enables us to prepare achievable plans, meet cost estimates, and scheduling commitments, and deliver required product functionality and quality with acceptable and reasonable consistency
- If a controlled process is not capable of meeting customer requirements or other business objectives, the process must be improved

Process Management Responsibilities

- Define the process
- Measure the process
- Control the process
  - ✓ Ensure variability is stable. Why? So that the results are predictable
- Improve the process



Defining the Process

- Defining a software process creates the disciplined and structured environment required for controlling and improving the process
- Management's responsibility to define each process inherently includes responsibilities for implementing and sustaining a process

Objectives of Process Definition

- Design processes that can meet or support business and technical objectives
- Identify and define the issues, models, and measures that relate to the performance of the processes
- Provide infrastructures (the set of methods, people, and practices) that are needed to support software activities
- Ensure that the software organization has the ability to execute and sustain the processes
  - ✓ Skills
  - ✓ Training
  - ✓ Tools
  - ✓ Facilities
  - ✓ Funds

Measuring the Process

- Measurements are the basis for detecting deviations from acceptable performance
- They are also basis for identifying opportunities for process improvement

Objectives of Process Measurements

- Collect the data that measure the performance of each process
- Analyze the performance of each process
- Retain and use data:
  - ✓ To assess process stability and capability
  - ✓ To interpret the results of observations and analyses
  - ✓ To predict future costs and performance
  - ✓ To provide baselines and benchmarks
  - ✓ To plot trends
  - ✓ To identify opportunities for improvements

Controlling the Process

- Controlling a process means keeping the process within its normal (inherent) performance boundaries – that is, making the process behave consistently
  - ✓ This involves: Measurement, Detection, Correction
- Measurement
  - ✓ Obtaining information about process performance
- Detection
  - ✓ Analyzing the information to identify variations in the process that are due to assignable causes
- Correction
  - ✓ Taking steps to remove variation due to assignable causes from the process and to remove the results of process drift from the product

- In other words, we can formulate Actions for Process Control

Actions for Process Control

- Determine whether or not the process is under control (is stable with respect to the inherent variability of measured performance)
- Identify performance variations that are caused by process anomalies (assignable causes)
- Eliminate the sources of assignable causes so as to stabilize the process
- Once a process is under control, sustaining activities must be undertaken to forestall the effects of entropy
- Without sustaining activities, processes can easily fall victim to the forces of ad hoc change or disuse and deteriorate to out-of-control states
- This requires reinforcing the use of defined processes through continuing management oversight, measurement, benchmarking, and process assessments

Improving the Process

- Even though a process may be defined and under control, it may not be capable of producing products that meets customer needs or organizational objectives
- For most organizations, processes must be technologically competitive, adaptable, and timely, and they must produce products that consistently meet customer and business needs
- Resources must be available and capable of performing and supporting the processes
- Processes can be approved improved by making changes that improve their existing capabilities or by replacing existing sub-processes with others that are more effective or efficient

Objectives of Process Improvement

- Understand the characteristics of existing processes and the factors that affect process capability
- Plan, justify, and implement actions that will modify the processes so as to better meet business needs
- Assess the impacts and benefits gained, and compare these to the costs of changes made to the processes

Issues on the Road to Process Improvement

Process Improvement

- All processes are designed to produce results. The products and services they deliver and the ways they deliver them have measureable attributes that can be observed to describe the quality, quantity, cost, and timeliness of the results produced
- If we know the current values of these attributes, and if a process is not delivering the qualities we desire, we will have reference points to start from when introducing and validating process adjustments and improvements
- So, our first concern on the road to process improvement is to understand the existing performance of the processes we use
- What is our existing processes producing now?

- Knowing how a process is performing will enable us to assess the repeatability of the process and whether or not it is meeting is internal and external needs
- We used "how," not "how well". Our purpose is to just get the facts.

Process Performance

- What is the process producing now with respect to measurable attributes of quality, quantity, cost, and time?
- Measures of process performance quantify and make visible the ability of a process to deliver products with qualities, timeliness, and costs that customers and businesses require
- When measurements of process performance vary erratically and unpredictably over time, the process is not in control
- To attain control, we must ensure first that we have a process whose variability is stable, for without stability we cannot predict results
- So, another important property associated with any process is that of process stability

Process Stability

- Is the process that we are managing behaving predictably?
- How do we know whether a process is stable?
- We examine process performance through the use of process behavior charts that allow us to determine whether the process is stable (within limits) and hence predictable
- If process performance is erratic and unpredictable, we must take action to stabilize that process
- Stability of a process depends on support for and faithful operation of the process
- Three questions that should concern people responsible for processes are:

Three Questions for Process Managers

- Is the process supported such that it will be stable if operated according to the definition?
- Is the process, as defined, being executed faithfully?
- Is the organization fit to execute the process?

Process Compliance

- Are the processes sufficiently supported?
- Are they faithfully executed?
- Is the organization fit to execute the process
- Having a stable and compliant process does not mean that process performance is satisfactory
- The process must be capable, meaning that variations in the characteristics of the product and in the operational performance of the process, when measured over time, fall within the ranges required for business success
- Measures of process capability relate the performance of the process to the specifications that the product or process must satisfy

Process Capability

- Is the process capable of delivering products that meet requirements?
- Does the performance of the process meet the business needs of the organization?
- If a software process is not capable of consistently meeting product requirements and business needs, or if an organization is to satisfy ever-increasing demands for higher quality, robustness, complexity, and market responsiveness while moving to new technologies and improving its competitive position, people in the organization will be faced with the need to continually improve process performance
- Understanding the capability of the sub-processes that make up each software process is the first step in making progress toward process improvement

Process Improvement

- What can we do to improve the performance of the process?
- What would enable us to reduce variability?
- What would let us move the mean to a more profitable level?
- How do we know that the changes we have introduced are working
- Resolution of these process improvement issues revolves around measurement and analysis of process performance, which leads to the question of process measurement

Process Measurement

- Sequential measurements of quality attributes of products and process can provide an effective foundation for initiating and managing process improvement activities
- Measurement is a mechanism for creating a corporate memory and an aid in answering a variety of questions associated with the enactment of any software process
- To determine the strengths and weaknesses of the current processes and products
- To evaluate the quality of specific processes and products
- Measurements also enable people to detect trends and anticipate problems, thus providing better control of costs, reducing risks, improving quality, and ensuring that business objectives are met
- Using measurements to manage and improve software processes show how quality characteristics of software products and processes can be quantified, plotted, and analyzed
- In turn, the performance of activities that produce the products can be predicted, controlled, and guided to achieve business and technical goals

Measurement Questions

- What should be measured?
- How should the characteristics measured?
- What is the data telling us?
- How is our process behaving?
- What are the signals that we should be reacting to?
- How do we know that something is a signal?
- What should we do when we recognize a signal?

Issues in Road to SPI

- Performance
- Stability
- Capability
- Improvement
- Measurement

References

- Measuring the Software Process by William Florac and Anita Carleton, Chapter 1

Lecture 10
Introduction to CMM

CMM Structure

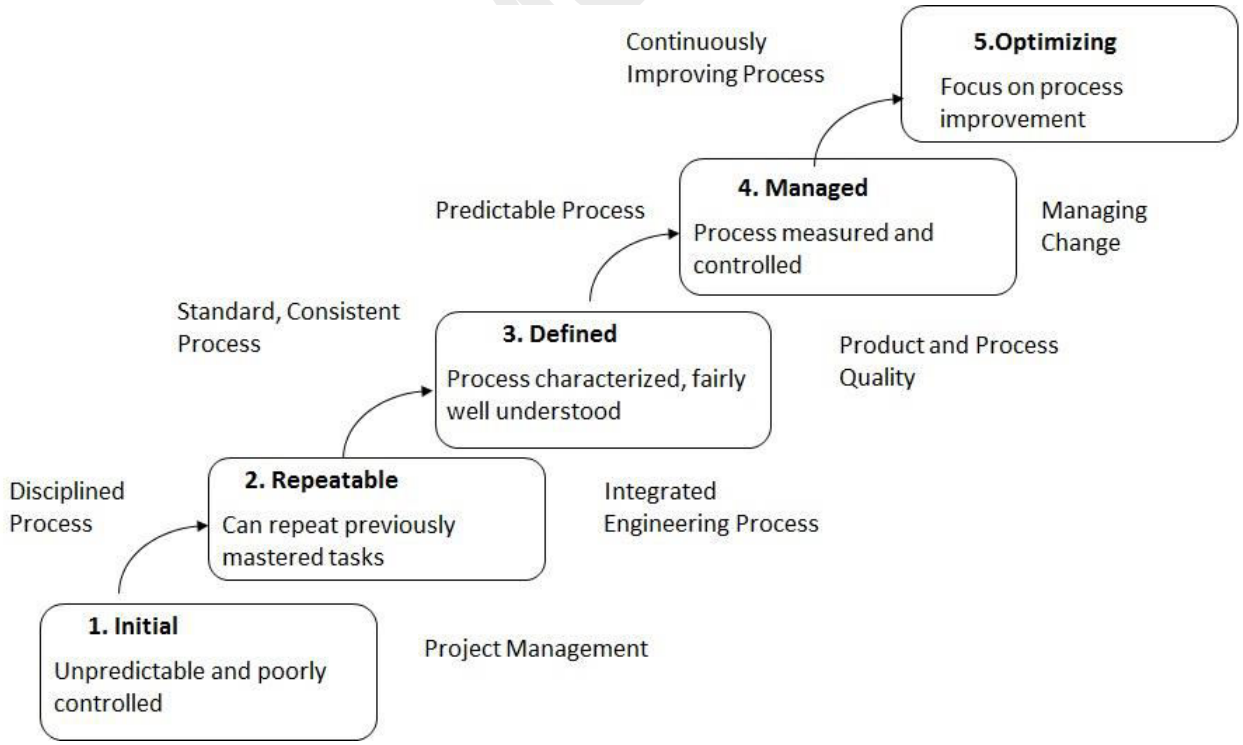
Software State-of-the-Art in 1984

- More than half of the large software systems were late in excess of 12 months
- The average costs of large software systems was more than twice the initial budget
- The cancellation rate of large software systems exceeded 35%
- The quality and reliability levels of delivered software of all sizes was poor
- Software personnel were increasing by more than 10% per year
- Software was the largest known business expense which could not be managed

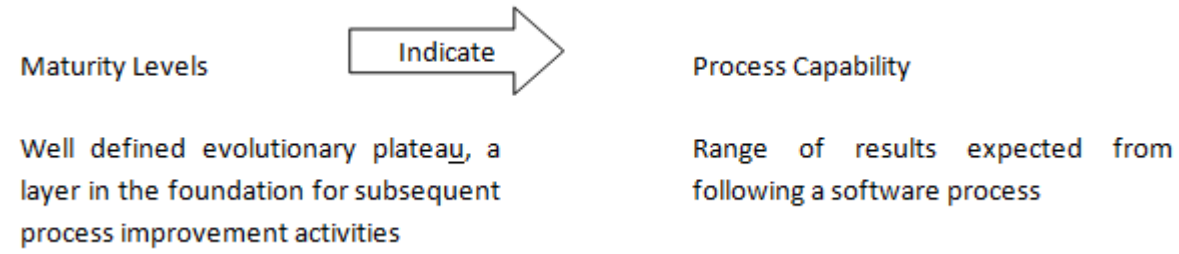
Software Engineering Institute

- A research facility, located in University of Carnegie Mellon, Pennsylvania
- Primarily funded by US DoD to explore software issues, and especially topics associated with defense contracts
- US DoD is the largest producer and consumer of software in the world
- SEI developed a Capability Maturity Model (CMM) for software systems and an assessment mechanism

The Five Levels of Software Process Maturity



Maturity Levels



- Well defined evolutionary plateau, a layer in the foundation for subsequent process improvement activities
  - ✓ Each level is a layer in the foundation for continuous process improvement
  - ✓ There are five maturity levels in the CMM
  - ✓ Achieving each level establishes a different component in the software process
  - ✓ Maturity levels are described in terms of 18 key process areas

Key Process Areas

Key Process Area Goals

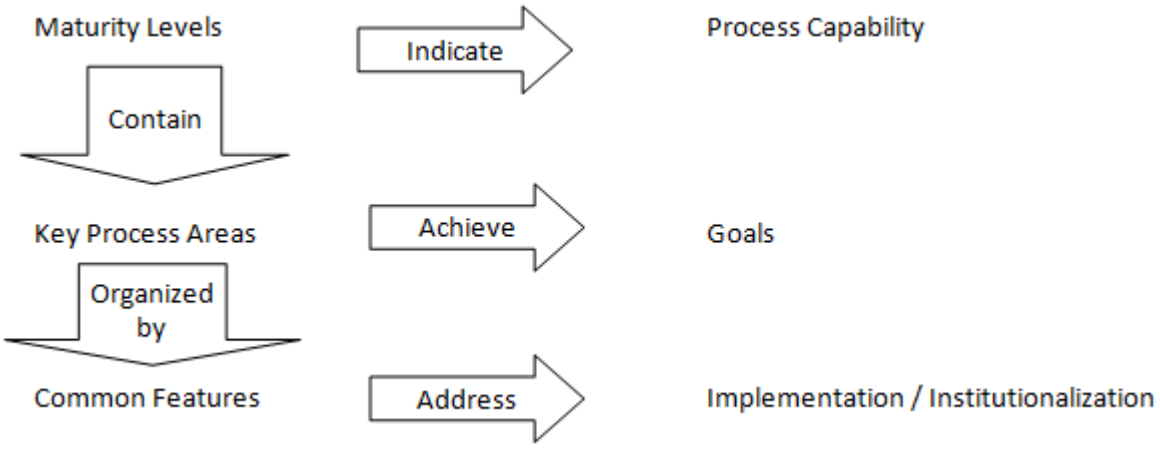
- Goals summarize the key practices of the key process areas.
  - ✓ They are considered important for enhancing process capability for that level of maturity
  - ✓ They can be used to guide organizations and appraisal teams in assessing alternative ways to implement key process areas
  - ✓ Each Key Practice maps to one or more goals

The CMM Key Process Areas

- Identify a cluster of related activities that, when performed collectively, achieve a set of goals considered important for enhancing process capability
  - ✓ Defined to reside at a single maturity level
  - ✓ Identify the issues that must be addressed to achieve a maturity level
  - ✓ 18 KPAs in the CMM

Levels/Process Categories	Management	Organizational	Engineering
5 Optimizing		Technology Change Management Process Change Management	Defect Prevention
4 Managed	Quantitative Software Management	Technology Change Management Process Change Management	Software Quality Management
3 Defined	Integrated Software Management Intergroup Coordination	Organization Process Focus Organization Process Definition Training Program	Software Product Engineering Peer Reviews
2 Repeatable	Requirements Management Software Project Planning Software Project Tracking and Oversight Software Subcontract Management Software Quality Assurance Software Configuration Management		
1 Initial		Ad Hoc Processes	

Common Features



- Used to organize the key practices in each key process area
- Attribute that ensure the processes are defined, documented and understood

- Common features are
  - ✓ Commitment to perform
  - ✓ ability to perform
  - ✓ activities performed
  - ✓ measurement and analysis
  - ✓ verifying implementation

Institutionalization

- The organization outlives those who leave it
- The organizational culture must convey the process
- Indicator of whether the KPA is effective, repeatable, and lasting
- The CMM has four common features that focus on institutionalizing the process
  - ✓ Commitment to perform
  - ✓ Ability to perform
  - ✓ Measurement and analysis
  - ✓ Verifying implementation
- Activities performed focus on implementing the process

Commitment to Perform

- Describes the actions the organization must take to ensure that the process is established and will endure
- Typically includes
  - ✓ Policies
  - ✓ Leadership

Ability to Perform

- Describes the preconditions that must exist in the project or organization to implement the software process competently
- Typically includes
  - ✓ Function / tools
  - ✓ Resources
  - ✓ Delegation
  - ✓ Training
  - ✓ Orientation
- Describes the roles and procedures necessary to implement a key process area
- Typically includes
  - ✓ Establishing plans and procedures
  - ✓ Performing the work
  - ✓ Tracking it
  - ✓ Taking corrective actions as necessary

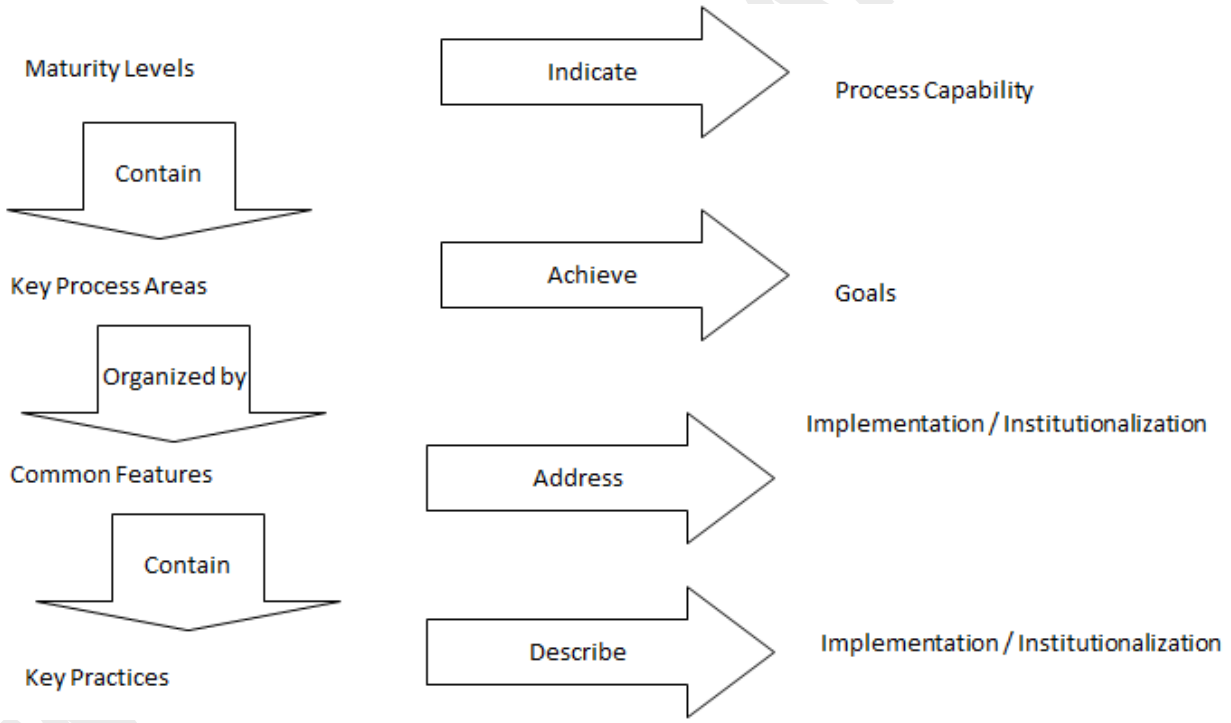
Measurement and Analysis

- Describes the need to measure the process and analyze the measurements
- Typically includes
  - ✓ Examples of the measurements that could be taken to determine the status and effectiveness of the activities performed common feature

Verifying Implementation

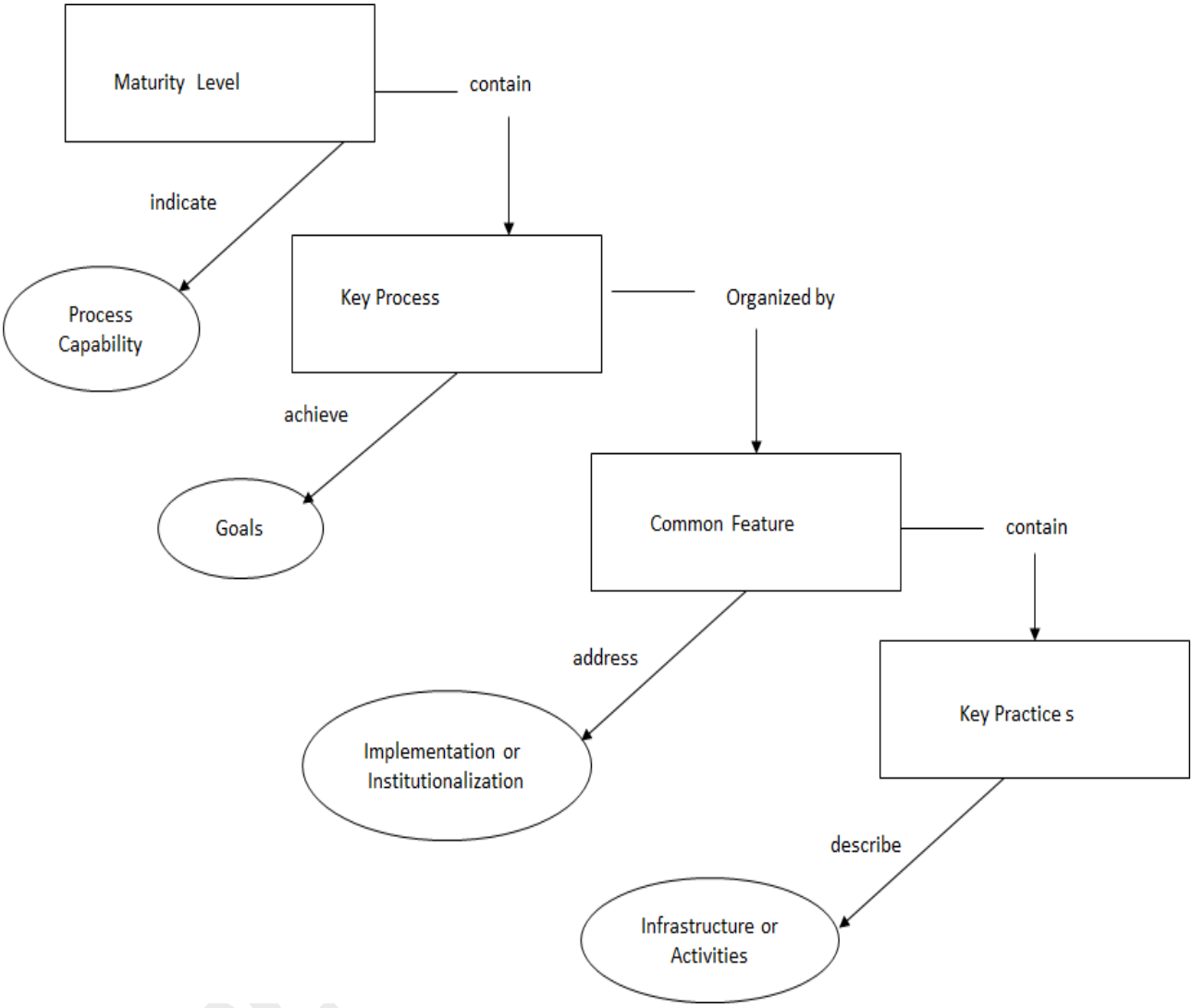
- Describes the steps to ensure that the activities are performed in compliance with the process that has been established
- Typically includes reviews and audits by
  - ✓ Senior Management
  - ✓ Project management
  - ✓ SQA

Key Practices

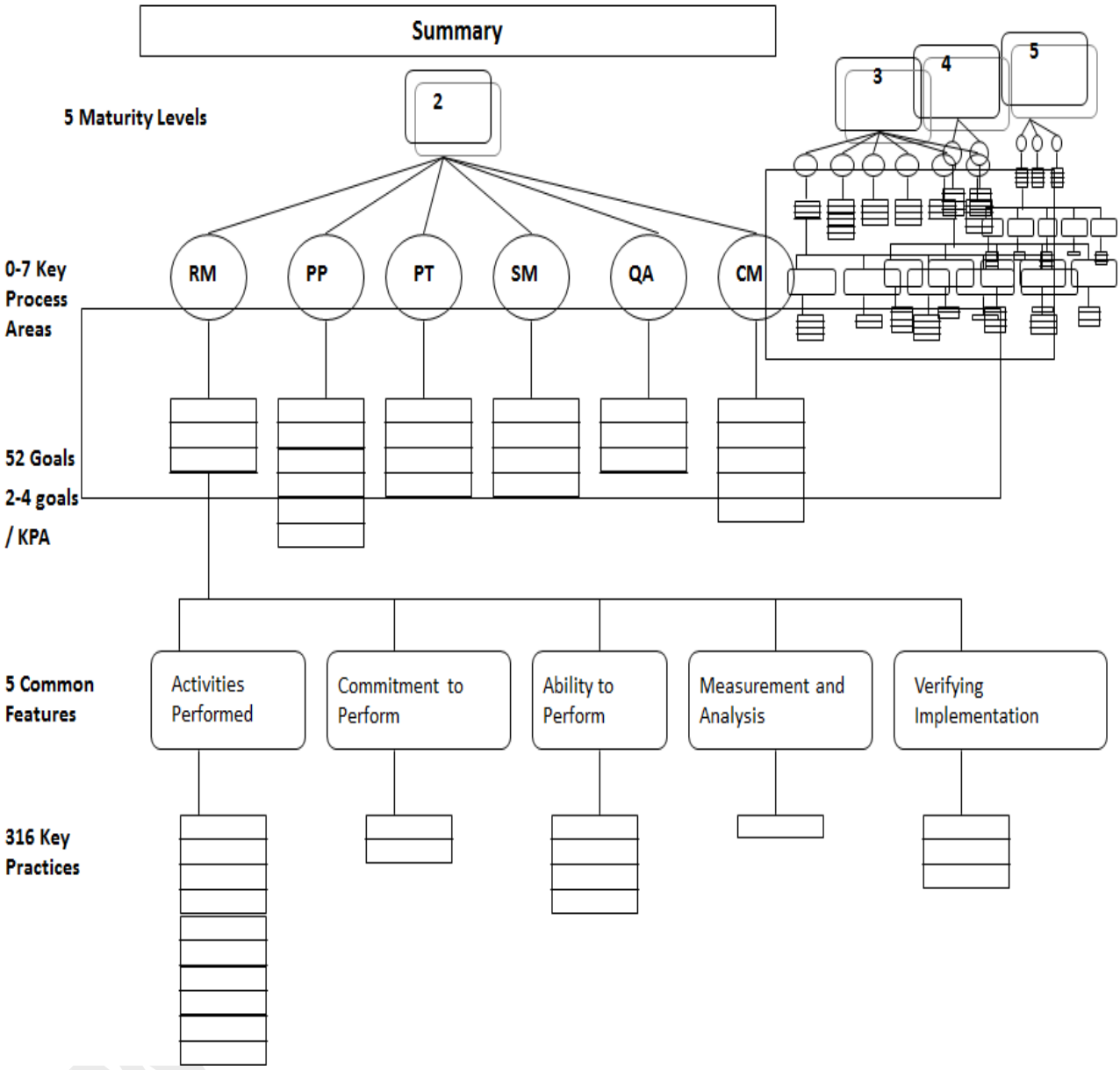


- State the fundamental policies, procedures, and activities for a key process area. Policies and procedures make up the infrastructure
- Describe "what" is to be done, but they should not be interpreted as mandating "how"
- Are organized by common features
- 316 key practices in the CMM

CMM Structure



Summary


Summary
5 Maturity Levels
2
3
4
5
0-7 Key Process Areas
RM
PP
PT
SM
QA
CM
52 Goals
2-4 goals / KPA
5 Common Features
Activities Performed
Commitment to Perform
Ability to Perform
Measurement and Analysis
Verifying Implementation
316 Key Practices

CMM Level 2
The Repeatable Level

Moving from Level 1 to Level 2

- At level 2, a software project management system is in place
- The organization sets expectations via policies
- At level 2 projects have disciplined processes
- Organizations have introduced at least some rigor into project management and technical development tasks
- Approaches such as formal cost estimating are noted for project management, and formal requirements gathering are often noted during development
- Compared to initial level, a higher frequency of success and a lower incidence of overruns and cancelled projects can be observed

KPA's - Level 2

- Requirements Management
- Software Project Planning
- Software Project Tracking and Oversight
- Software Subcontract Management
- Software Quality Assurance
- Software Configuration Management

Requirements Management

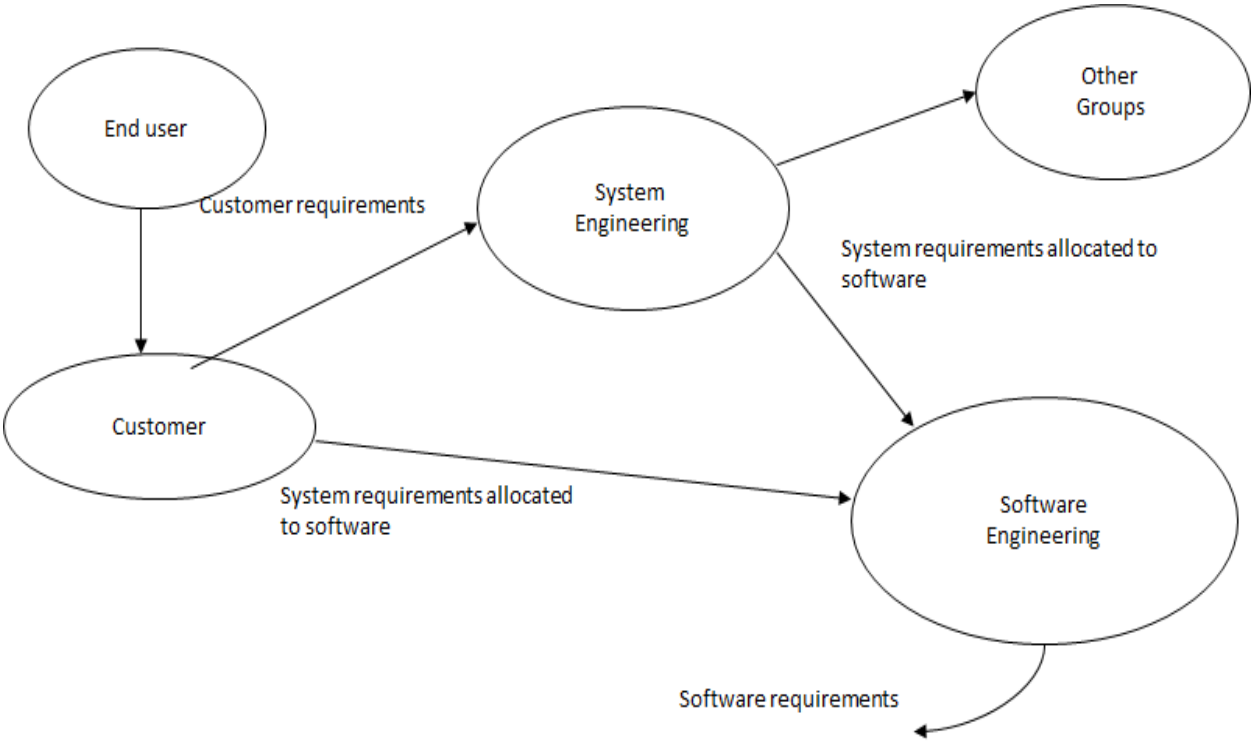
Purpose

- The purpose is to establish a "common understanding" between the customer and organization
- Involves
  - ✓ Document and control of customer requirements
  - ✓ Keeping plans, products, and activities consistent with requirements

Software Requirements versus Allocated Requirements

- The scope of Requirements management is control of requirements allocated to software
- Requirements allocated to software come from systems engineering
- Software requirements are derived from requirements allocated to software

What are Requirements?



Who is Customer?

- Customers may be external or internal
- Systems engineering
- marketing
- user
- The customer and / or end user identifies the problem that software will address

Documentation of requirements

- The system requirements assigned to software Engineering must be documented
- Documenting system requirements can be as simple as a memo or as elaborate as a multi volume specification
- If requirements change, the changes must be documented and all resulting necessary changes in other documents must be tracked and verified

Review of Requirements

- The software engineering group ensures that the system requirements allocated to software are documented and controlled. For which, the software engineering group reviews the initial and revised system requirements allocated to software to resolve issues

Software Project Planning

Purpose

- The purpose is to establish reasonable plans for performing the software engineering and managing the project
- Involves:
  - ✓ Developing estimates for the work
  - ✓ Establishing necessary commitments
  - ✓ Defining the plan to perform the work

Managing Based on Plan

- Plan provides the basics for initiating the software effort and managing the work.
- Other names for this plan include:
  - ✓ Software development plan
  - ✓ Software project management plan
  - ✓ Software project plan
  - ✓ Software engineering management plan

What is a Software Development Plan?

- A software development plan specifies many or all of the following:
- projects chosen software life cycle
- list of products to be developed
- Schedules
- Estimates for level of effort (number of people), cost, etc.
- Facilities, support tools, and hardware
- Project risks

Packaging the Software Development Plan

- There are many ways the software plan can be developed.
- There may be separate documents entitled:
  - ✓ Software development plan
  - ✓ Software quality assurance plan
  - ✓ Software configuration management plan
  - ✓ Risk management plan
  - ✓ Software test plan
  - ✓ Project training plan

Plans are based on Estimates

- In creating estimates for size, effort, cost, schedule, and/or computer resources
- use historical data, where available
- Document assumptions and estimates
- Good estimating depends on the skills and judgment of the estimator

Meeting Commitments

- Commitment - a pact that is freely assumed, visible, and expected to be kept by all parties.
- Commitment is necessary to achieve plans.
- Feasible commitments are made when plans are realistic.
- Commitment is a process.

Software Project Tracking and Oversight

Purpose

- To provide adequate visibility into actual progress so that management can take effective actions when the software projects performance deviates significantly from the software plans
- Involves:
  - ✓ Tracking and reviewing software accomplishments and results against documented estimates, commitments and plans
  - ✓ Adjusting plans based on actual accomplishments and results

Manage to a Plan

- Progress must be tracked against plans and specifications, including
  - ✓ Product size
  - ✓ Project effort, cost and schedule
  - ✓ Activities
  - ✓ Risks
- Mechanisms to track progress against plans include both internal reviews and formal reviews with the customer.

Taking Corrective Action

- If and when discrepancies between plans and actual progress occur, a judgment must be made about whether to:
  - ✓ Change the work the works being done
  - ✓ Adjust the plans
- This judgment results in corrective action
- Archives of original and adjusted plans should be kept.

Software Subcontract Management

Purpose

- To select qualified software subcontractors and manage them effectively.
- Involves
  - ✓ Selecting a software subcontractor
  - ✓ Establishing commitments with the subcontractor
  - ✓ Tracking and reviewing the subcontractor performance and results

Prime vs Subcontractor

- The prime contractor is the organization responsible for building a system
- The prime contractor may contract out part of the work to another contractor, the subcontractor
- Performance of the prime contractor may critically depend on performance of the subcontractor

Planning for the Subcontract

- In selecting and managing subcontractor, prime contractors must perform activities additional to the ordinary project management effort.
- Specify the work to be performed and the procedures to be followed by the subcontractor
  - ✓ Statements of work
  - ✓ Requirements
  - ✓ Products to be delivered
  - ✓ Standards
  - ✓ Procedures
- Specify criteria for selecting and evaluating subcontractors

Selecting the Subcontractor

- The qualifications of a subcontractor may depend on many factors
- Process capability
  - ✓ Software engineering expertise
  - ✓ Application domain knowledge
  - ✓ Strategic business alliances

Managing a Subcontract

- The prime contractor must manage subcontract
- Ensures subcontractor follow software development plans, standards and procedures
- Monitoring software quality assurance by subcontractor
- Monitoring software configuration management by the subcontractor

Software Quality Assurance

Purpose

- Purpose is to provide management with appropriate visibility in to the process being used by software project and products being built.
- Involves
  - ✓ Reviewing and auditing products and activities to ensure that they comply with applicable procedures and standards
  - ✓ Providing the software project and other appropriate managers with the results of those reviews and audits

Provides Visibility to Everyone

- The value of SQA is that it provides an independent view of the project's activities, process and product.
- SQA serves the "eyes and ears" of the management.
- Most key process areas contain SQA practices in Verifying Implementation

Independent vs Objective

- Commitment 1.2 (the SQA group has an "independent reporting channel" to the senior management) indicates that an independent SQA group is normally expected.
- Goal 2 of SQA (verify adherence objectively) provides latitude in defining SQA.
- Without an independent SQA group, can the organization demonstrate objective means of adherence?

Resolving Noncompliance

- There are three possible ways for resolving a noncompliance issue:
  - ✓ Make the product or process satisfy the standard, procedure or requirement
  - ✓ Change the standard or procedure to make it useable
  - ✓ Make an executive decision not to satisfy the standard, procedure or requirement

SQA Evolves

- Proactive SQA functions as a value adding member of the project team
  - ✓ SQA starts early in the project
  - ✓ SQA helps to prepare and review procedures, plans and standards
- At higher levels, the CMM provides flexibility for SQA to function proactively to drive improvements in the software process and product.

Software Configuration Management

Purpose

- Purpose is to establish and maintain the integrity of the products of the software projects throughout the project's software life cycle
- Involves:
  - ✓ Identifying configuration items / units
  - ✓ Systematically controlling changes
  - ✓ Maintaining integrity and traceability of the configuration throughout the life cycle

Using Baselines

- SCM relies on baselining software work products.
- A baseline is a specification or product that
  - ✓ Has been formally reviewed and agreed on
  - ✓ Serves as the basis for further work
  - ✓ Can be changed only through formal change control procedures

Controlling Change

- SCM provides a stable working environment.
- Uncontrolled change of work product is a chaotic process.
- SCM provides a memory of the status of software work products via baselines.
- When many individuals are working on the same product, SCM coordinates access to and change of the software work products.

Software Product and Software Work Product

- **Software Product** - the complete set, or any of the individual items of the set, of computer programs, procedures, associated documentation, and data designated for delivery to a customer or end user.
- **Software Work Product** - any artifact created as part of defining, maintaining, or using a software process, including process descriptions, plans, procedures, computer programs, and associated documents, which may or may not be intended for delivery top a customer or end user.

Baseline Vs Developmental Configuration Management

- In baseline CM, baselines for identified software work products are established at predetermined points
- In developmental CM, configuration control is exercised by the developers as they perform their work
- The SCM KPA can be satisfied using at a minimum baseline configuration management

Managed and Controlled

- Some software work products do not need the formality of configuration management but do need to be placed under some form of
  - ✓ Version control
  - ✓ Change control
- This is referred to as “managed and controlled” in the key practices.
- Managed and controlled is a subset of developmental configuration management.

References

- The Capability Maturity Model: Guidelines for Improving Software Process

Lecture 11
CMM Level 3
The Defined Level

CMM Capability Maturity Model

- Initial
- Repeatable
- Defined
- Managed
- Optimizing

The Defined Level

Moving from Level 2 to Level 3

- At level 2, the focus is on projects
- At level 3, the emphasis shifts to the organization
  - ✓ Best practices are gathered across the organization
  - ✓ Processes are tailored as appropriate
- Organization supports the project by establishing
- Common processes, common measurements and training
- Organizations have mastered a development process that can often lead to successful large systems
- Over and above the project management and technical approaches found in Level 2 organizations, the Level 3 groups have a well-defined development process that can handle all sizes and kinds of projects

KPA's - Level 3

- Organization Process Focus
- Organization Process Definition
- Training Program
- Integrated Software Management
- Software Product Engineering
- Intergroup Coordination
- Peer Reviews

Organization Process Focus

Purpose

- Purpose is to establish the organizational responsibility for software process activities that improve the organization's overall process capability
- Involves
  - ✓ Developing and maintaining an understanding of organization's and projects software processes
  - ✓ Coordinating the activities to assess, develop, maintain, and improve these processes

Dedicating People to Process

- A dedicated group of people is responsible for the organization's software process activities; e.g.
  - ✓ Appraisals
  - ✓ Software process improvement plans
- Includes maintaining the organization's software process database and providing training about the organization's software process

Dedicated Groups May Vary

- A software engineering process group (SEPG) is the typical means of providing a process focus for the organization.
- Other ways of focusing on process are possible:
  - ✓ process review boards
  - ✓ quality circles
  - ✓ process steering committees
  - ✓ software quality assurance
- These mechanisms may work in conjunction with, or in place of, an SEPG.

Responsibilities of SEPGs

- Establish process standards
- Maintain the process database
- Serve as the focal point for technology transition
- Provide process education
- Provide project consultation
- Make periodic assessment and status reports

Organization Process Definition

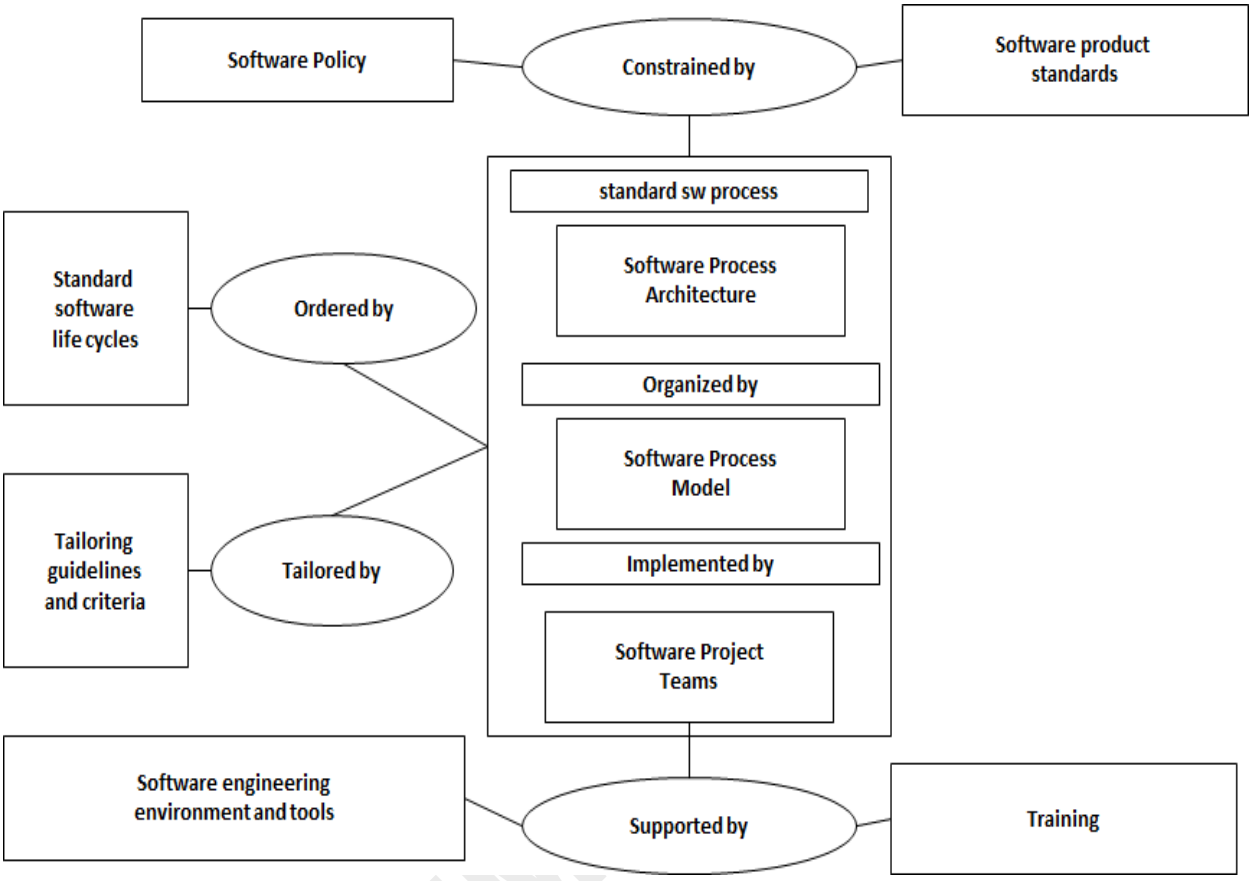
Purpose

- Purpose is to develop and maintain a useable set of software process assets that improve process performance and provide a basis for cumulative and long term benefits.
- Involves
  - ✓ Developing and maintaining the organization's standard software process and related process assets

Software Process Assets

- A Collection of entities, maintained by an organization for use by projects in developing, tailoring, maintaining and implementing their software practices.
- Software process assets include
  - ✓ The organization's standard software process
  - ✓ Descriptions of software life cycles approved for use
  - ✓ Guidelines and criteria for tailoring the organization's standard software process
  - ✓ The organization's software process database
  - ✓ A library of software process related documentation

Context for Software Process Assets in an Organization



Process Building Blocks

- he organization's standard software process includes process elements for activities such as
  - ✓ Software estimating
  - ✓ Design
  - ✓ Coding
  - ✓ Peer review
  - ✓ Process elements as described in terms of standards, procedures, templates, abstractions, etc., which can be "hooked together" to form a process.

Relating Process Elements

- Process elements have
  - ✓ An order in which they can be connected (perhaps more than one)
  - ✓ Interfaces
  - ✓ Interdependencies
- These relationships are sometimes referred to as a software process architecture.
- Many different software processes can be built using the architecture of the organization's standard software process.

Software Life Cycles

- The organization may support several software life cycles models such as
  - ✓ Waterfall
  - ✓ Spiral
  - ✓ Incremental
  - ✓ Combination
  - ✓ Prototype
  - ✓ Rational unified process
  - ✓ Agile processes

Tailoring Guidelines

- Guidelines for tailoring the organization's standard software process are available to individual projects.
- What can be tailored out? What cannot?
- How much can a process element be modified?
- What part of a process element should be considered for tailoring?

Organizations software Process Database: Measuring Data

- A central repository for organization measurement data.
- Contains
  - ✓ Actual measurement data (the numbers) from individual projects
  - ✓ The related information needed to understand the measurement data and apply it to the new projects
- This is where the planning and re-planning data for the organization as a whole are kept

Library of software process related documentation

- The library where best documents used on past projects are kept
- Also contains lessons learnt, reports, example documents and document fragments
- In general contains any document that can be used as model or example for future projects

Relation to Organization Process Definition

- Organization Process Focus and Organization Process Definition are tightly coupled
- Organizational Process Focus focuses on the who (roles and people).
- Organization Process Definition focuses on the what.

Training Program

Purpose

- Purpose is to develop the skills and knowledge of individuals so that they can perform their roles effectively and efficiently.
- Involves
  - ✓ Identifying the training needs of the organization, projects and individuals
  - ✓ Developing and/or procuring training to address the identified needs

Training

- Software training is identified, developed, and established by an organizational training group
- The training group
  - ✓ Begins by reviewing each software project's training needs and plans.
  - ✓ Analyzes the skills needed by the organization and how and when needed training will occur
  - ✓ Prepares, develops, and maintains training courses
  - ✓ Maintains training records

Organizational Training

- Organizational training needs are identified based on the
  - ✓ Organization's standard software processes
  - ✓ Project needs
  - ✓ Organization's culture
  - ✓ Individual needs

Waivers

- The purpose of training is to impart skills and knowledge
- individuals want to build their professional skills
- Individuals already possessing a skill do not need training
- A waiver procedure should exist to waive people who are already trained

Required Training

- At Level 2 a phrase "receive training" is used
  - ✓ Training at level 2 may not be institutionalized across the organization
- At Level 3 and above, the phrase "receive required training" is used
  - ✓ Institutionalization of training is expected.

Alternative Vehicles

- Training may include formal as well as informal instruction to transfer skills and knowledge
- Mentors and on the job training can be very effective if implemented properly
- Remember that informal vehicles are frequently abused

Integrated Software Management

Purpose

- Purpose is to integrate the project's software engineering and management activities into a coherent, defined software process that is tailored from the organization's standard software process and related process assets.
- Involves
  - ✓ Developing the projects defined software process by tailoring the organization's standard software process
  - ✓ Managing the software project according to this defined software process

Tailoring Organizational Processes

- At Level 3, each project tailors the organization's standard software process for its particular needs
  - ✓ Includes software life cycles, standards, procedures, etc..
  - ✓ Uses the lessons learned and data from previous projects
  - ✓ Feeds back appropriate measurement data and documents to the organization for use by other projects

Project Management Evolves

- Software development plan is now based on the project's defined software process
- Projects can use and share process data and lessons learnt
- Integrated software management is the evolution of software project planning and software project tracking and oversight

Software Product Engineering

Purpose

- Purpose is to consistently perform a well-defined engineering process that integrates all the software engineering activities to produce correct, consistent software products effectively and efficiently.
- Involves
  - ✓ Performing the engineering tasks to build and maintain the software using the appropriate methods and tools.

Software Life cycles processes

- Development and maintenance activities include
- software requirement analysis
  - ✓ Software design
  - ✓ Coding
  - ✓ Testing
  - ✓ Integration
- Testing activities include
  - ✓ Unit
  - ✓ Integration
  - ✓ System
  - ✓ Acceptance

Documentation

- Addresses needs that span life cycle
- Customer and end-user documentation includes
  - ✓ User manuals
  - ✓ Training materials
  - ✓ Operator manuals
- Developer / maintainer documentation includes
  - ✓ Software requirements documents
  - ✓ Design documents
  - ✓ Test documents
  - ✓ Maintenance manuals

Consistency and Traceability

- Consistency and traceability among documents are critical to their usefulness
- Inconsistent specifications are not useable
- If the relationship between documents is not clearly traceable, their usefulness is minimized

Inter group Coordination

Purpose

- Purpose is to establish a means for software engineering group to participate actively with the other engineering groups, so that the project is better able to satisfy the customer needs effectively and efficiently.
- Involves
  - ✓ Disciplined interaction and coordination of the projects engineering groups with each other to address system-level requirements, objectives and plans

Ongoing working Relationship

- "Commitments" among groups are documented and agreed to by all groups
- Inter group coordination can be
  - ✓ As minimal as one or two meetings between project groups
  - ✓ As extensive as integrated product teams

Interdisciplinary Groups

- The software engineering group actively interfaces with a variety of groups.
- Examples of these groups, to whom the interface must be managed include;
  - ✓ Systems engineering
  - ✓ Marketing
  - ✓ Training
  - ✓ Sub contract management
  - ✓ Documentation
- Intergroup coordination is a first step on the road to concurrent engineering

Peer Reviews

Purpose

- Purpose is to remove defects from the software work products early and efficiently. An important corollary effect is to develop a better understanding of the software work products and of defects that might be prevented
- Involves
  - ✓ Methodical examination of work products by the producer's peers to identify defects and areas where changes are needed.
  - ✓ Identifying products that will undergo a peer review in the project's defined software process

Performing Peer Reviews

- Plan and schedule peer reviews
- Train leaders and participants
- Give material to the reviewers in advance
- Assign each reviewer a specific role
- Specify criteria to begin and end reviews
- Use pre planned checklists
- Identify action items and track to closure
- Collect and analyze data for product and peer review process

Alternative Peer Review Methods

- Possible alternative ways of implementing peer reviews include;
- Fagan-style inspections
- Structured walkthroughs
- Active reviews
- Phased inspections

References

- The Capability Maturity Model: Guidelines for Improving Software Process

Lecture 12
CMM Level 4
The Managed Level

Moving from level 3 to 4

- At level 3, measurements have been defined and collected systematically
- At level 4, decisions are made based on the data collected
  - ✓ Common measurement
  - ✓ Data analysis
- Organizations have established a firm quantitative basis for project management and utilize both effective measurements and also effective cost and quality estimates

KPAs - Level 4

- Software Quality Management
- Quantitative Process Management

Quantitative Process Management

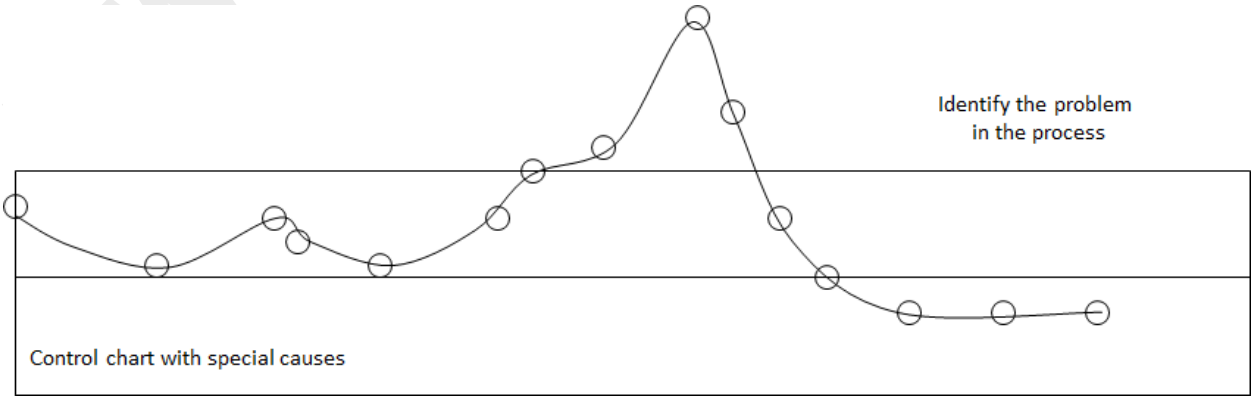
Purpose

- Purpose is to control the process performance of software project quantitatively
- Involves
  - ✓ Establishing goals for the process performance
  - ✓ Measuring the performance of the project
  - ✓ Analyzing these measurements
  - ✓ Making adjustments to maintain process performance within acceptable limits.

Controlling Special Causes of Variation

- An important concern is identifying variations in performance that are not within the normal range of process performance
- "extraordinary" events outside the bounds of process capability

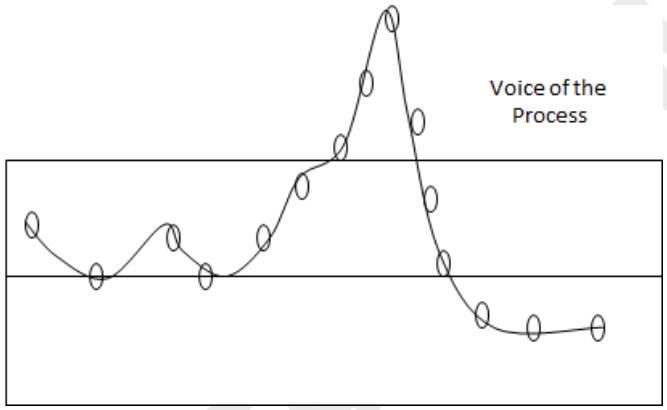
"Extraordinary" events outside the bounds of process capability



Taking Corrective Action at Level 4

- Quantitative Process Management focuses on the process
- Process capability is quantitatively known
- When performance falls outside the limits:
  - ✓ Identify the reason
  - ✓ Take corrective action when appropriate

Control chart with special causes



Meaning of Quantitative Control

- Quantitative control in the CMM implies any quantitative or statistically based technique.
- The words "statistical" and "quantitative" imply data
- Data reflect facts
- Fact-based management results in objective decisions

Basic Statistics

- Quantitative, or statistical, techniques need to be sophisticated to be useful
- Preto analysis for example is very simple
- Quantitative techniques do require
  - ✓ Measurable data
  - ✓ Consistent data collection
  - ✓ Defined comparable measurements

Seven (Some) Basic Tools of Statistical Process Control

- There are seven quantitative tools considered basic to statistical process or quality control.
  - ✓ Histograms
  - ✓ Cause and effect diagrams
  - ✓ Check sheets
  - ✓ Pareto diagrams
  - ✓ Run charts
  - ✓ Control charts
  - ✓ Scatter diagrams

Software Quality Management

Purpose

- Purpose is to develop a quantitative understanding of the quality of the project's software products and achieve specific quality goals
- Involves
  - ✓ Defining quality goals for the software products,
  - ✓ Establishing plans to achieve these goals
  - ✓ Monitoring and adjusting software plans and work products

Building High-Quality Products

- Software Quality Management Focuses on the product.
- Measurable quality goals for the product are defined.
- The product is ready when the goals are achieved.

Quality Evolves

- At level 2 the focus of quality is conformance to requirement
- By level 4 there is emphasis on understanding needs of the
  - ✓ Customer
  - ✓ End users
  - ✓ Organization (supplier)
- Ultimately customer determines what quality is or is not
- TQM revolves around customer satisfaction

The CMM Level 5
The Optimizing Level

Moving from Level 4 to 5

- At level 4 , the process is quantitatively understood
- At level 5, continues process improvement is a way of life
- In immature organizations, no one may be responsible for process improvement
- Mature organizations usually have 70 to 80% participation in improvement activities at any point in time – everyone is involved.
- Organizations are assumed to have mastered the current state-of-the-art of software project management and development

KPAs - Level 5

- Defect Prevention
- Technology Change Management
- Process Change Management

Defect Prevention

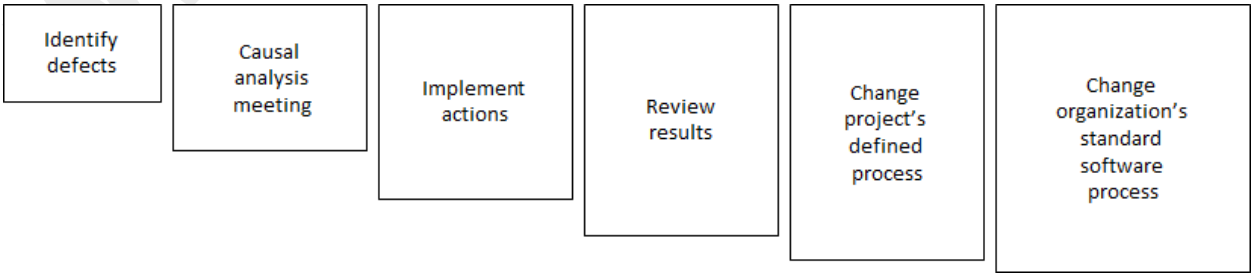
Purpose

- Purpose is to identify the cause of defects and prevent them from reoccurring
- Involves:
- Analyzing defects that were encountered in the past
- Taking specific actions to prevent the occurrence of these types of defects in the future

Fixing Problems Before They Happen

- Focus is on causal analysis
  - ✓ What in the process permitted the defect to occur
  - ✓ What in the process needs to be corrected to prevent the defect from occurring in the future

Fixing Problems Before They Happen



Technology Change Management

Purpose

- Purpose is to identify new technologies (i.e. tools, methods and processes) and track them into the organization in an orderly manner
- Involves
  - ✓ Identifying, selecting, and evaluating new technologies,
  - ✓ And incorporating effective technologies into organization

Reacting to Innovation

- Technology changes occur at all maturity levels
- At level 5 innovation is:
  - ✓ Introduced into the process in a disciplined way
  - ✓ Part of the culture (institutionalized)

Process Change Management

Purpose

- Purpose is to continually improve the software process used in the organization with the intent of improving software quality, increasing productivity, and decreasing the cycle time for product development
- Involves
  - ✓ Defining process improvement goals and,
  - ✓ Systematically identifying, evaluating, and implementing improvements to the organization's standards software process, and project's defined software processes

Change Is a Process

- Disciplined change is the key to success
- improvement includes
  - ✓ Planning
  - ✓ Evaluating improvement proposals and planning actions
  - ✓ Establishing process improvement teams
  - ✓ Conducting pilot programs for process improvement
  - ✓ Updating procedures, training etc
- Improvements are transferred into everyday practice across the organization

Level 5 is not the destination

- Level 5 is the foundation for building an ever improving capability
- level 5 organizations continuously improve
  - ✓ Incremental (Kaizen)
  - ✓ Revolutionary
- everyone in level 5 organization is involved in improvement

Tying It All Together

Themes in the CMM

- Continuous improvement
- Defined, documented and used processes
- Commitment by senior management
- Stable processes
- Measured processes
- Controlled processes
- Evolving processes
- Team work

Measurement by Levels

1. Measurement is haphazard, but investigation may yield cost and effort data
2. Projects collect management data about costs, effort, size, schedule, quality etc
3. There are consistent affiliations from one project to the next. Data are collected across the organization both management and quality data are collected
4. Data analysis is based on the principles of statistical process control. Actual measurements are compared to expected values of mean and variance
5. Continuous improvement is based on business objectives and cost benefit analysis

How is improvement achieved?

- Know where you are
- Know where you want to go
- assign resources and responsibilities
- Try progress

Keys to Improvements

- Senior management commitment
- Organizational consensus on important issues
- Believe that improvement is possible
- Action oriented frame work for improvement

Key Adjectives

- CMM is not exhaustive
- there are software management and engineering processes that are not described in the CMM
- key includes a focus on the major leverage points
  - ✓ key process area
  - ✓ key practices

What does a good software organization look like?

- Processes are defined, documented, and used
- Management plans, monitors and communicates
- Roles and responsibilities are clear
- Product and processes are measured
- Quality cost and schedule are predictable
- Technology is planned and used effectively
- Continuous improvement is a way of life

ISO 9001:2008

Process-based Quality Management System

- Talk a little about efforts of ISO in improving quality
- Focus on quality management system
- The adoption of a quality management system should be a strategic decision of an organization
- The design & implementation of an organization's quality management system is influenced by
  - ✓ Its organizational environment, changes in that environment, and the risks associated with that environment
  - ✓ Its varying needs
  - ✓ Its particular objectives,
  - ✓ The products it provides,
  - ✓ The processes it employs,
  - ✓ Its size and organizational structure
- This International Standard promotes the adoption of a process approach when developing, implementing and improving the effectiveness of a quality management system, to enhance customer satisfaction by meeting customer requirements
- For an organization to function effectively, it has to determine and manage numerous linked activities
- An activity or set of activities using resources, and managed in order to enable the transformation of inputs into outputs, can be considered as a process
- Often the output from one process directly forms the input to the next
- The application of a system of processes within an organization, together with the identification and interactions of these processes, and their management to produce the desired outcome, can be referred to as the "process approach"
- An advantage of the process approach is the ongoing control that it provides over the linkage between the individual processes within the system of processes, as well as over their combination and interaction

Eight Principles of Management

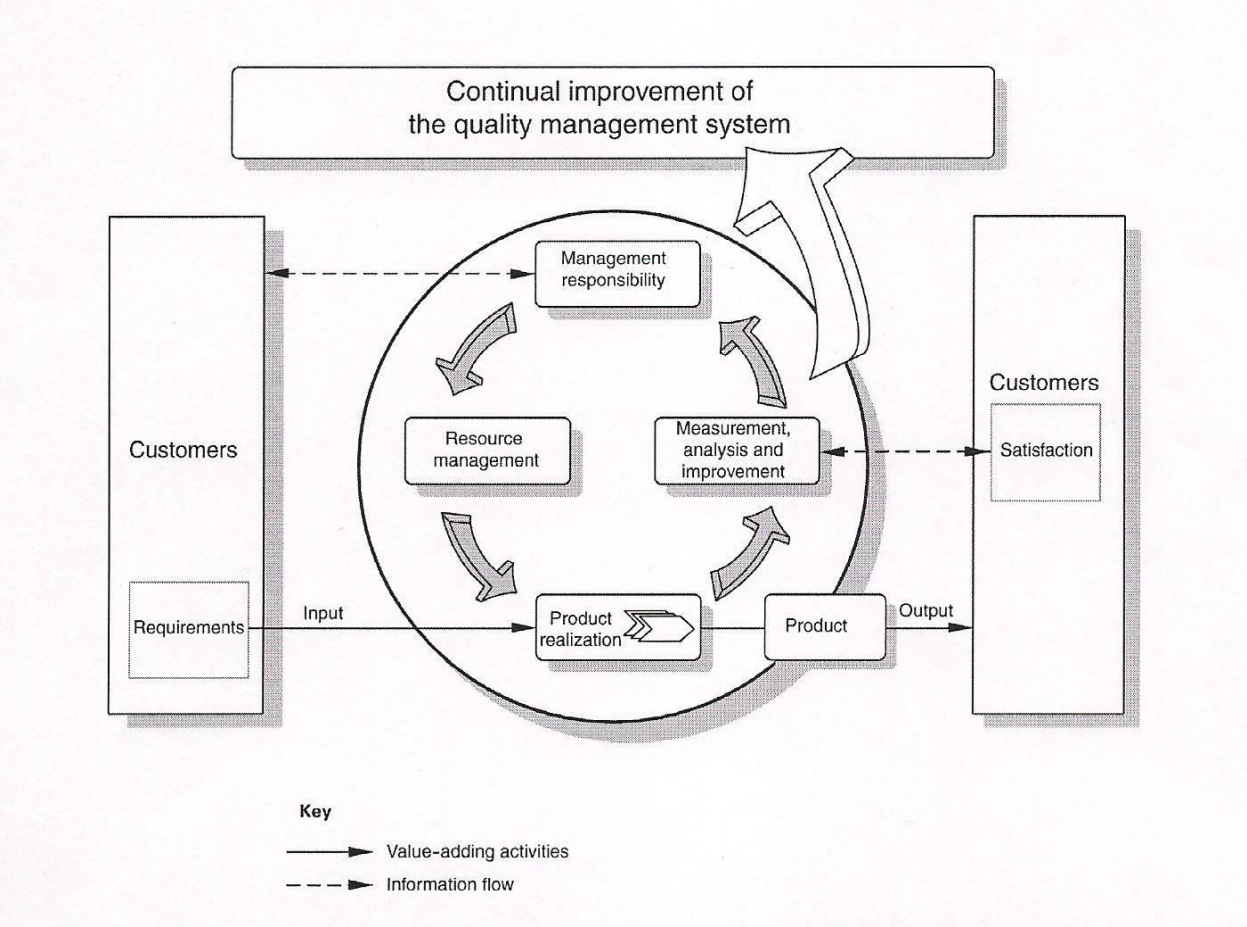
- Customer focus
- Leadership
- Involvement of people
- Process approach
- System approach
- Continual improvement
- Factual approach
- Mutually beneficial supplier relationship

Quality Management System

- When used within quality management system, such an approach emphasizes the importance of
  - ✓ Understanding and meeting requirements,
  - ✓ The need to consider processes in terms of added value
  - ✓ Obtaining results of process performance and effectiveness, and
  - ✓ Continual improvement of processes based on objective measurement

Model of a Process-based Quality Management System

- The figure that I am going to show you next, is that of process-based quality management system
- This illustration shows that customers play a significant role in defining requirements as inputs



Model of a Process-based Quality Management System

- Monitoring of customer satisfaction requires the evaluation of information relating to customer perception as to whether the organization has met the customer requirements
- This figure shows all the requirements of this International Standard, but does not show processes at a detailed level
- All requirements of this International Standard are generic and are intended to be applicable to all organizations, regardless of type, size and product provided and are applicable to IT organizations
- Where any requirements of this International Standard cannot be applied due to the nature of an organization and its product, this can be considered for exclusion

QMS - Requirements

- General requirements
- Documentation requirements
  - ✓ General
  - ✓ Quality manual
  - ✓ Control of documents
  - ✓ Control of records

QMS: Management Responsibility

- Management commitment
- Customer focus
- Quality policy
- Planning
- Responsibility, authority and communication
- Management review

QMS: Resource Management

- Provision of resources
- Human resources
- Infrastructure
- Work environment

QMS: Product Realization

- Planning of product realization
- Customer-related processes
- Design and development
- Purchasing
- Production and service provision
- Control of monitoring and measuring equipment

QMS: Measurement, Analysis and Improvement

- General
- Monitoring and measurement
- Control of nonconforming product
- Analysis of data
- Improvement

References

- Quality Management Systems – Requirements by ISO (.pdf file)
- The Capability Maturity Model: Guidelines for Improving Software Process

Lecture 13
Introduction to CMMI

- CMMI is an acronym for Capability Maturity Model Integration
- Some people would say the CMMI is a model, while others would describe it as a set of models
- But most will agree that the CMMI is a merger of process improvement models for systems engineering, software engineering, hardware engineering, and integrated teams
- Some of the goals of the CMMI are to provide a common vocabulary across the set of models and to provide clarification on how these areas interrelate
- The integrated model has both a continuous and staged perspective
- Let's discuss the differences between the CMMI, a process model, a process, a process description, a procedure, and an appraisal
- Some people have a problem because they think CMMI is a process or a set of processes
- It is neither

What is CMMI?

- CMMI is a process model
- The CMMI cannot just be copied as is and serve as an organization's processes
- CMMI is a collection of best practices from highly functioning organizations collected to help you improve your processes by describing what things or activities should be done in your organization

What is a Process?

- A process is what you really do in your organization
- The process is written down as the activities or steps you go through to perform work
- Your organization's processes must be documented to be real, to be performed consistently, and to be improved
- If your organization has undocumented processes and you believe that these processes are used consistently then you are living in a world of denial
- The CMMI recommends certain attributes that are necessary when documenting processes

What is a Process Description?

- A process description is a "documented expression of a set of activities to achieve a given purpose"
- The process description documents the steps of the process performed, that is, what you do when you perform a task
- Closely related are procedures that describe the step-by-step instructions on how to perform the steps in the process

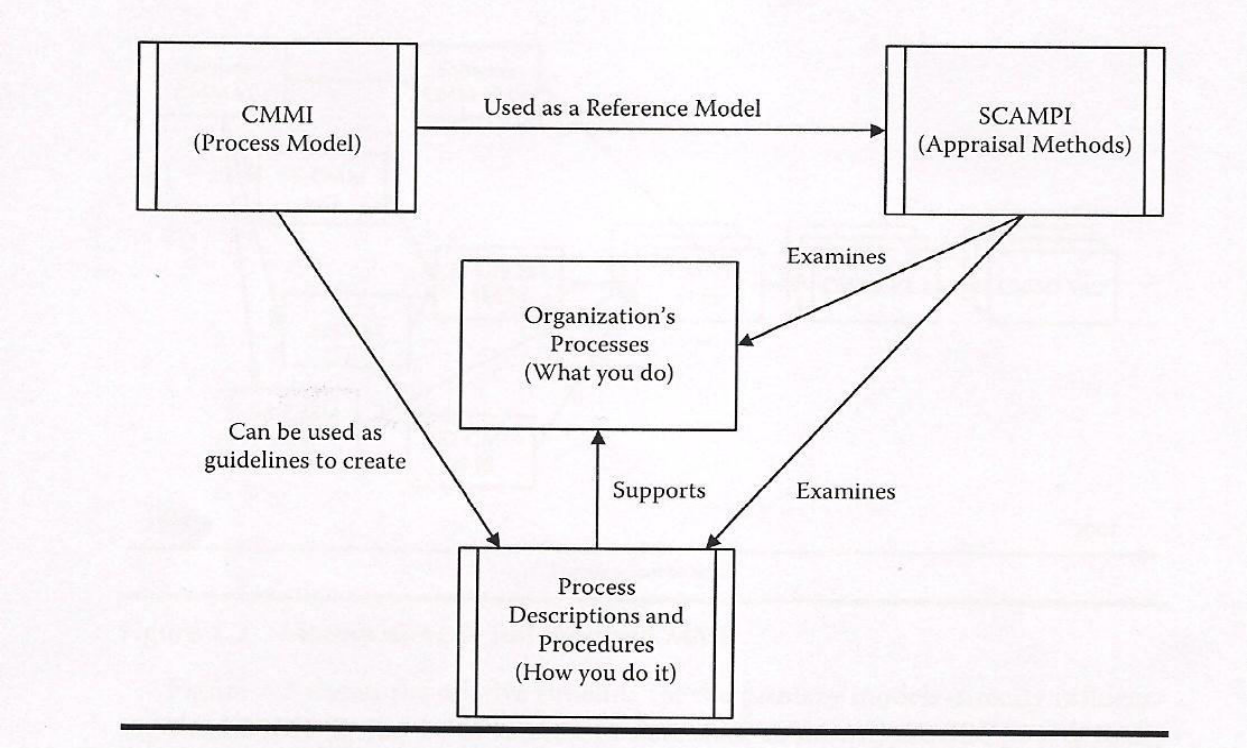
What is an appraisal?

- An appraisal is a way to evaluate the organization's processes and procedures, how they were implemented, to what extent they are followed, and to what extent they map to the CMMI practices

CMMI

- In the CMMI world, SCAMPI (Standard CMMI Appraisal Method for Process Improvement) is a set of appraisal methods that differ in their intent and rigor
- The SCAMPI can be used for internal process improvement, supplier selection, process monitoring, and for maturity or capability level ratings for contract awards
- The CMMI is used as a guideline to create processes and procedures
- CMMI is used as the reference model for the SCAMPI appraisal method to examine the organization's processes and procedures

Relationship between Key Terms



Short History of CMMI

- People were successfully using the Capability Maturity Model, which was designed for improving software processes and measuring the maturity of software processes in an organization
- This success brought more and more attention to model-based process improvement in other areas

What Models Do I Use?

- Historically: Depends on the discipline that you want to model.
  - ✓ Software Engineering
  - ✓ Systems Engineering
  - ✓ Software Acquisition
  - ✓ Systems Security
  - ✓ etc.

What is CMM?

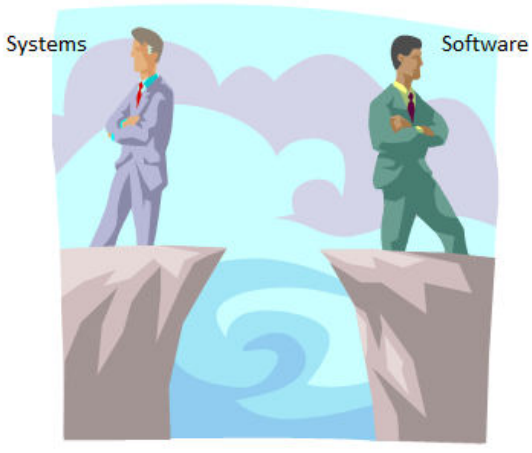
- Capability Maturity Model: A reference model of mature practices in a specified discipline, used to assess a group's capability to perform that discipline
- CMMs differ by
  - ✓ Discipline (software, systems, acquisition, etc.)
  - ✓ Structure (staged versus continuous)
  - ✓ How Maturity is Defined (process improvement path)
  - ✓ How Capability is Defined (institutionalization)

Commonly Used CMMs

Software CMM	staged	software development
System Engineering CMM	continuous	system engineering
System Engineering Capability Model	continuous	system engineering
Software Acquisition CMM	staged	software acquisition
System Security Engineering CMM	continuous	security engineering
Personal Software Process	staged	individual software development
FAA-iCMM	continuous	software engineering, systems engineering, and acquisition
IPD-CMM	hybrid	integrated product development
People CMM	staged	workforce
SPICE Model	continuous	Software development

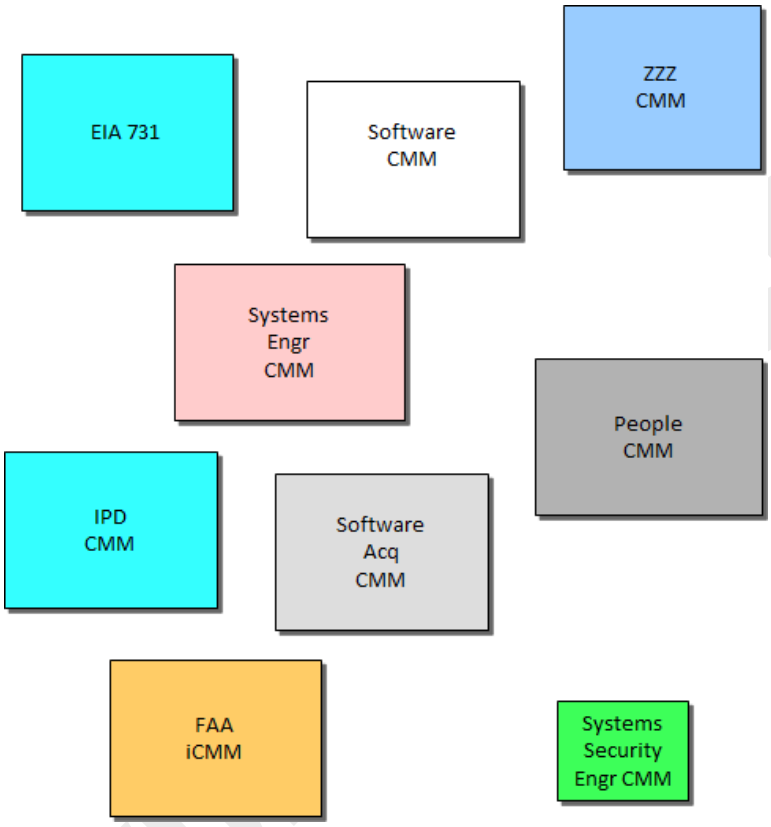
System and Software Divide

- Systems and software disciplines have traditionally not been well integrated
- The importance of software in systems has increased dramatically
- There was need to make the systems/software interface more seamless



So Many Models, So Little Time

- Different structures, formats, terms, ways of measuring maturity
- Causes confusion, especially when using more than one model
- Hard to integrate them in a combined improvement program
- Hard to use multiple models in supplier selection

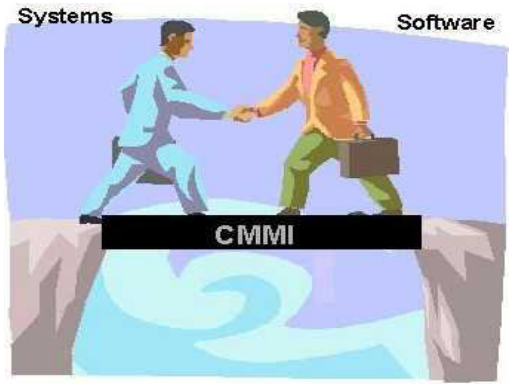


CMMI Comes to Rescue

- Integrates systems and software disciplines into one process improvement framework
- Provides a framework for introducing new disciplines as needs arise

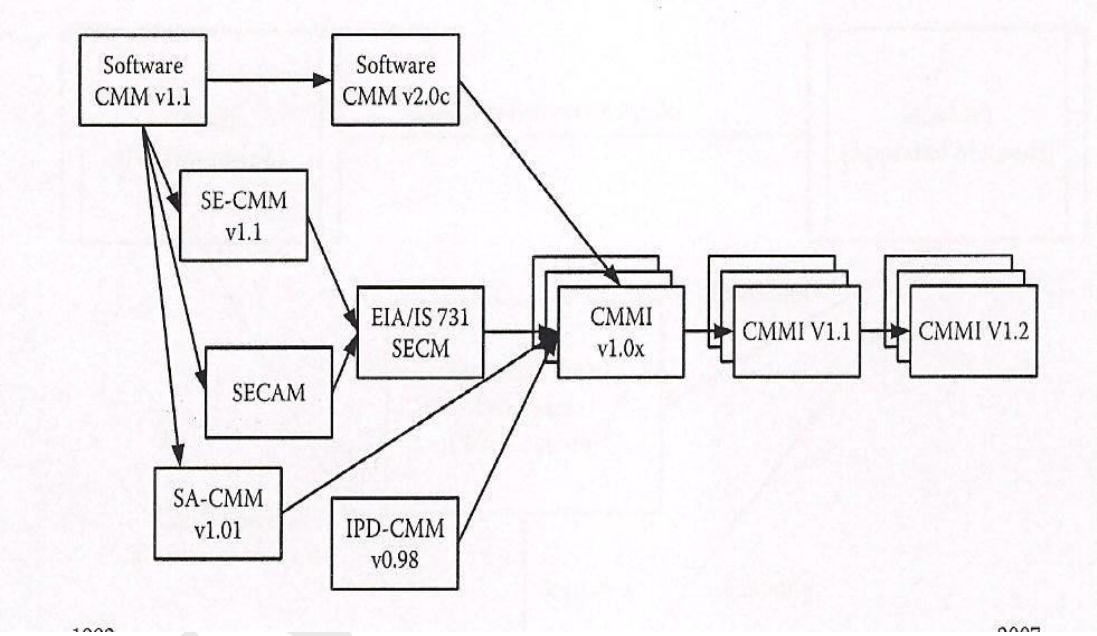
Bridging the Divide

- Systems engineering and software engineering processes are integrated
- Integrates systems and software disciplines into one process improvement framework
- Provides a framework for introducing new disciplines as needs arise

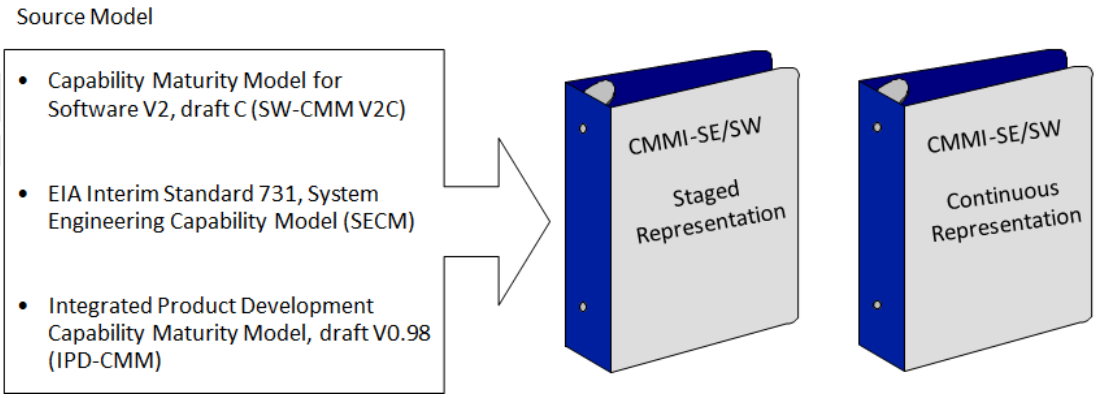


- Some organizations see themselves as performing just one discipline
  - ✓ Software
  - ✓ Systems
  - ✓ Acquisition
- But…
  - ✓ Software always must be part of some kind of system
  - ✓ Systems that don't have software are rare
  - ✓ Acquisition can involve both
- Communication and cooperation with other disciplines, even if they are external to our organization is vital

Models directly influencing CMMI

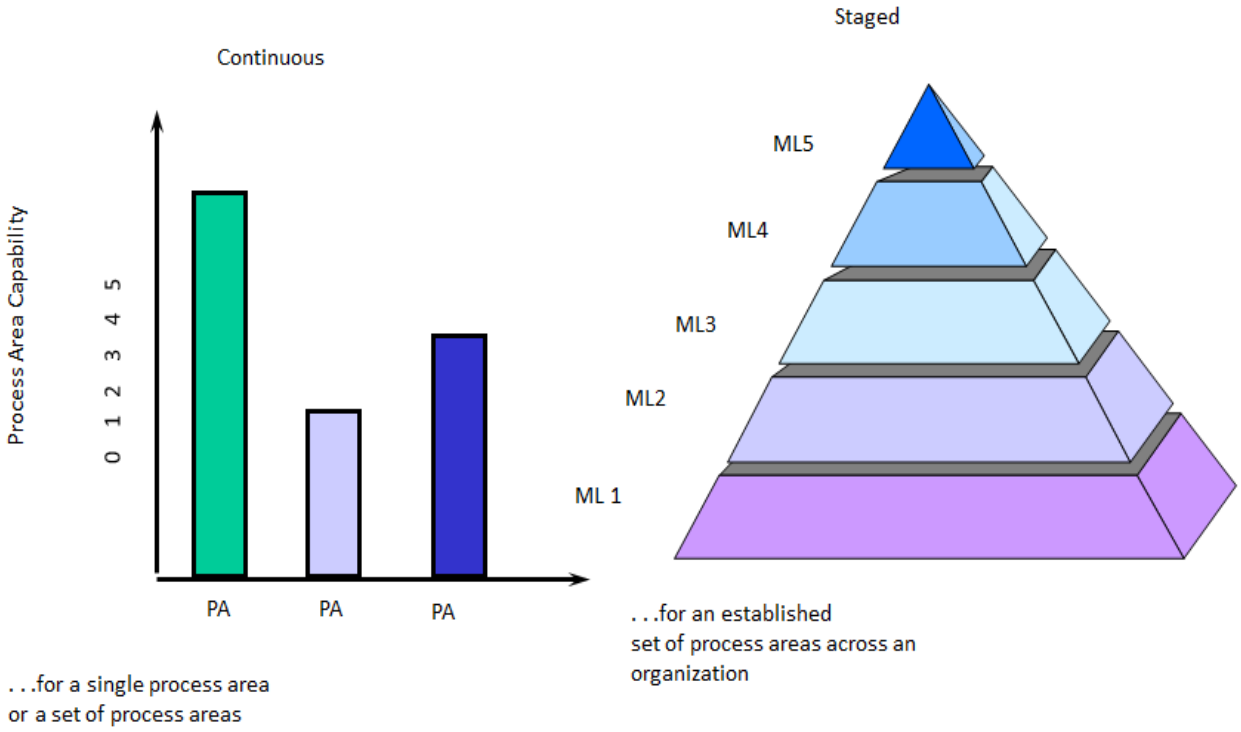


CMMI Models



- Combined System Engineering/Software Engineering model
- Can be applied to
  - ✓ Just the software engineering projects in an organization
  - ✓ Just the system engineering projects in an organization
  - ✓ Both

Comparing Model Representations



Staged Representation

- The staged representation is the approach used in the Software CMM. It is an approach that uses predefined sets of process areas to define an improvement path for an organization. This improvement path is described by a model component called a maturity level
- A maturity level is a well-defined evolutionary plateau toward achieving improved organizational processes

Continuous Representation

- The continuous representation is the approach used in the SECM and the IPD-CMM. This approach allows an organization to select a specific process area and improve relative to it
- The continuous representation uses capability levels to characterize improvement relative to an individual process area

Process Area Capability and Organizational Maturity

- Process area capability and organizational maturity are similar concepts
- The difference between them is that process area capability deals with a set of processes relating to a single process area or specific practice, while organizational maturity pertains to a set of process areas across an organization

Remember

- A model is not a process
- The model shows what to do, NOT how to do it or who does it

Why do we have Two Representations?

- Source Model Heritage
  - ✓ Software CMM--Staged
  - ✓ SECM--Continuous
  - ✓ IPD CMM—Hybrid
- Proponents for each type of representation were part of CMMI product development team
- Selecting a single representation approach became “too hard”
- A compromise was made to initially support two representations of the model with equivalent content

Advantages of Staged Representation

- Provides a roadmap for implementing
  - ✓ Groups of process areas
  - ✓ Sequencing of implementation
- Familiar structure for those transitioning from the SW-CMM

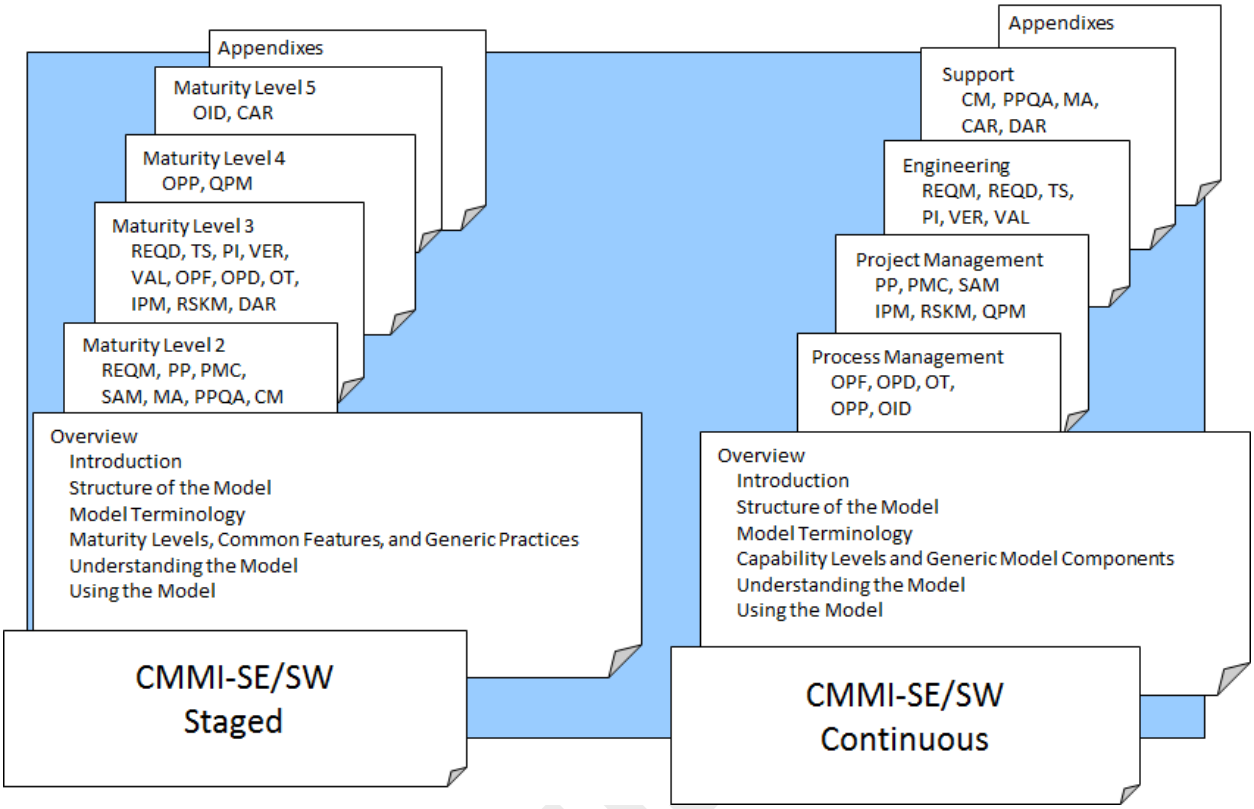
Advantages of Continuous Representations

- Provides maximum flexibility for focusing on specific process areas according to business goals and objectives
- Familiar structure for those transitioning from the systems engineering community

CMMI in a Nutshell

- A CMMI model provides a structured view of process improvement across an organization
- CMMI can help
  - ✓ Set process improvement goals and priorities
  - ✓ Provide guidance for quality processes
  - ✓ Provide a yardstick for appraising current practices

CMMI Structure: One Model, Two Representations



CMMI Model Structure

- Maturity Levels (staged representation) or Capability Levels (continuous representation)
- Process Areas
- Goals – Generic and Specific
- Practices – Generic and Specific
- The CMMI product suite is at the forefront of process improvement because it provide the latest best practices for product and service development and maintenance
- The CMMI models improve the best practices of previous models in many important ways

Benefits of CMMI

- CMMI best practices enable organizations to do the following
  - ✓ More explicitly link management and engineering activities to their business objectives
  - ✓ Expand the scope of and visibility into the product lifecycle and engineering activities to ensure that the product or service meets customer expectations
  - ✓ Incorporate lessons learned from additional areas of best practices (e.g., measurement, risk management, and supplier management)
  - ✓ Implement more robust high-maturity practices
  - ✓ Address additional organizational functions critical to their products and services
  - ✓ More fully comply with relevant ISO standards

- Use CMMI in process improvement activities as a
  - ✓ Collection of best practices
  - ✓ Framework for organizing and prioritizing activities
  - ✓ Support for the coordination of multi-disciplined activities that might be required to successfully build a product
  - ✓ Means to emphasize the alignment of the process improvement objectives with organization business objectives

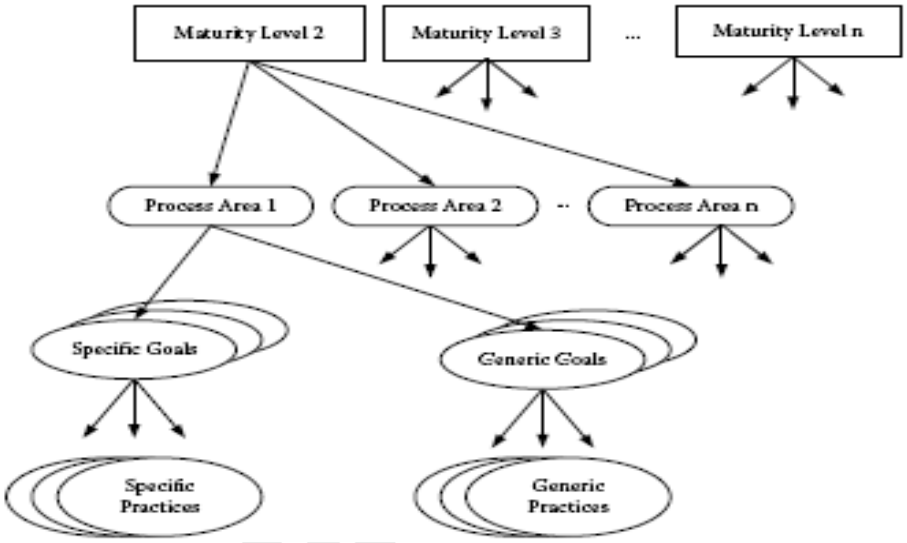
References

- http://www.sei.cmu.edu/general/index.html
- Interpreting the CMMI: A Process Improvement Approach, Second Edition, by Margaret K. Kulpa and Kent A. Johnson, Auerbach Publication, 2008 (electronic file), (Chapter 1-4)

Lecture 14
Introduction to CMMI Staged Representation

Staged Representation

- The staged representation is the approach used in the Software CMM. It is an approach that uses predefined sets of process areas to define an improvement path for an organization. This improvement path is described by a model component called a maturity level
- A maturity level is a well-defined evolutionary plateau toward achieving improved organizational processes
- CMMI Model Components in the Staged Representation

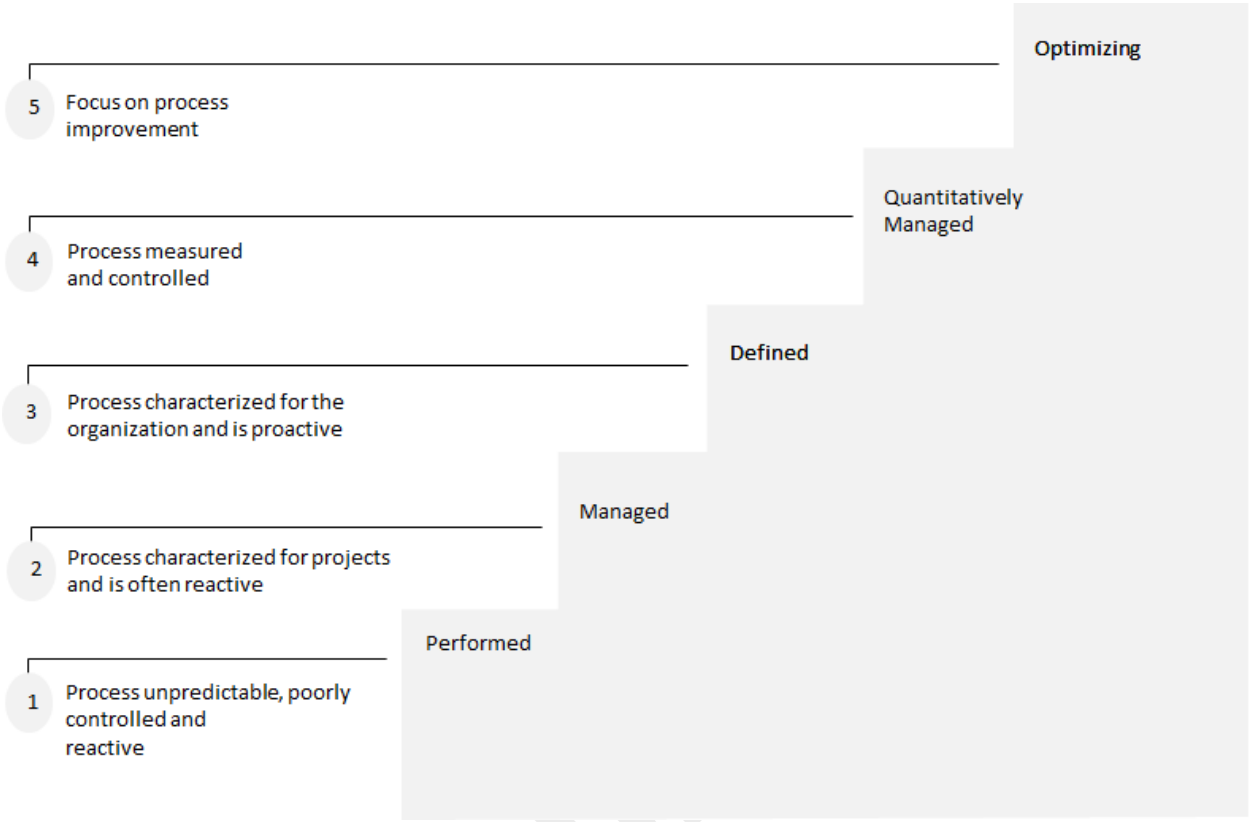


- The staged representation offers a systematic, structured way to approach process improvement one step at a time. Achieving each stage ensures that an adequate improvement has been laid as a foundation for the next stage
- Process areas are organized by maturity levels that take much of the guess work out of process improvement. The staged representation prescribes the order for implementing each process area according to maturity levels, which define the improvement path for an organization from the initial level to the optimizing level

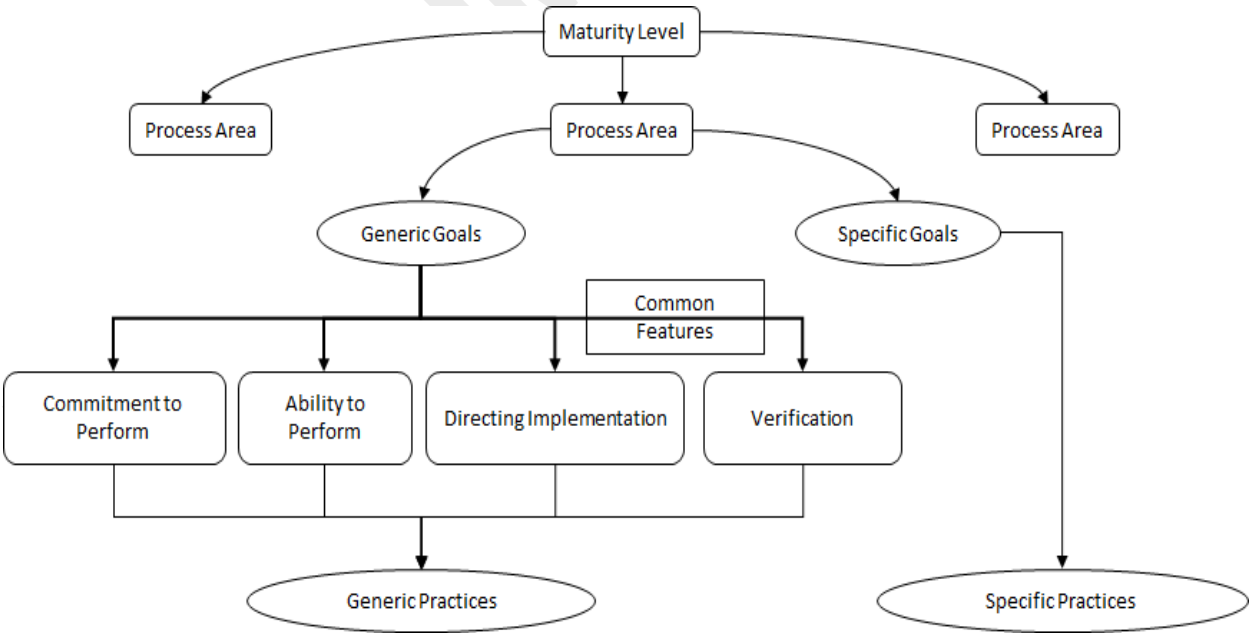
Maturity Levels

- Achieving each maturity level ensures that an adequate improvement foundation has been laid for the next maturity level and allows for lasting, incremental improvement
- Maturity level signifies the level of performance that can be expected from an organization
- A maturity level is a well-defined evolutionary plateau on the path to becoming a mature organization
- There are five maturity levels
- Each level is a layer in the foundation for continuous process improvement
- Performed, Managed, Defined, Quantitatively Managed, Optimizing

The Maturity Levels



Structure of the CMMI Staged Representation



- Commitment to Perform: creates policies and secures sponsorship for process improvement efforts
- Ability to Perform: ensures that the project and/or organization has the resources it needs to pursue process improvement
- Directing Implementation: collects, measures, and analyzes data related to processes
- Verification: verifies that the projects and/or organization's activities conform to requirements, processes, and procedures

- All old common features got rolled into Level 2 generic practices.
- Both activities and common features are present in each KPA, where the common features include Commitment to Perform, Ability to Perform, Measurement and Analysis, and Verifying Implementation.

Maturity Levels Cannot Be Skipped

- A level provides a necessary foundation for effective implementation of processes at the next level.
  - ✓ Higher level processes are easily sacrificed without the discipline provided by lower levels.
  - ✓ The effect of innovation is obscured in a noisy process.
- Higher maturity level processes may be performed by organizations at lower maturity levels, with risk of not being consistently applied in a crisis.

Process Areas

- Process Areas (PAs) are clusters of relatedpractices performed collectively to achieve a set of goals.
- They are the major building blocks in establishing the process capability of an organization.
- Each process area has been defined to reside at a given maturity level.
-
- Anwar: Please delete the mention of "Integrated Supplier Management" from the audio discussion of the Process Areas of CMMI.
- You will need to re-number the Pas. Thanks. This PA has been removed from the CMMI version 2

Process Areas by Maturity Level

Level	Focus	Process Areas
5 Optimizing	Continuous process improvement	Organizational Innovation and Deployment Causal Analysis and Resolution
4 Quantitatively Managed	Quantitative management	Organizational Process Performance Quantitative Project Management

3 Defined	Process standardization	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Integrated Project Management Risk Management Decision Analysis and Resolution
2 Managed	Basic Project management	Requirements Management Project Planning Project Monitoring and Control Supplier Agreement Management Measurement and Analysis Process and Product Quality Assurance Configuration Management

Level 1: Performed

Level 2: Managed

- Adhering to organizational policies
- Following a documented plan and process description
- Applying adequate funding and resources
- Maintaining appropriate assignment of responsibility and authority
- Training people in their appropriate processes
- Placing work products under appropriate configuration management
- Monitoring and controlling process performance, and taking corrective action
- Objectively reviewing the process, work products, and services, and addressing noncompliance
- Reviewing the activities, status, and results of the process with appropriate levels of management, and taking corrective action
- Identifying and interacting with relevant stakeholders

Level 3: Defined

- The organization has achieved all of the goals of Level 2
- There is an organizational way of doing business, with tailoring of this organizational method allowed under predefined conditions
- The organization has an organization's set of standard processes (OSSP)
- The following characteristics of the process are clearly stated
  - ✓ Purpose
  - ✓ Inputs
  - ✓ Entry criteria
  - ✓ Activities
  - ✓ Roles
  - ✓ Measures
  - ✓ Verification steps
  - ✓ Outputs
  - ✓ Exit criteria

- Level 3 continues with defining a strong, meaningful, organization-wide approach to developing products
- An important distinction between Level 2 and Level 3 is that at Level 3, processes are described in more detail and more rigorously than at Level 2
- Processes are managed more proactively, based on a more sophisticated understanding of the interrelationships and measurements of the processes and parts of the processes
- Level 3 is more sophisticated, more organized, and establishes an organizational identity—a way of doing business particular to this organization

Level 4: Quantitatively Managed

- For Maturity Level 4, the organization has achieved all of the goals of Levels 2 and 3
- The organization controls its processes by statistical and other quantitative techniques
- Product quality, process performance, and service quality are understood in statistical terms and are managed throughout the life of the processes
- Level 4 focuses on using metrics to make decisions and to truly measure whether progress is occurring and your product is improving
- Distinctions between Level 3 and Level 4 are that at Level 3, processes are qualitatively predictable
- At Level 4, processes are quantitatively predictable
- Level 4 addresses special causes of process variation and takes corrective action

Level 5: Optimizing

- The organization has achieved all of the goals of Levels 2, 3, and 4
- Processes are continually improved based on an understanding of common causes of variation within the processes
- Level 5 is nirvana
- Everyone is a productive member of the team, defects are reduced, and your product is delivered on time and within the estimated budget

Maturity Levels

- In the staged representation, the maturity levels serve as process boundaries— meaning that the efforts documented in that maturity level relate only to that maturity level
- For example, Requirements Management is a Level 2 process area. The next process area in Level 2 is Project Planning. Then there is Project Monitoring and Control, Supplier Agreement Management, Measurement and Analysis, Process and Product Quality Assurance, and Configuration Management. So, to be considered a Maturity Level 2 organization, the projects undergoing process improvement need to satisfy the goals for all of the process areas for Maturity Level 2
- An organization seeking to be Maturity Level 3 would need to structure their process improvement program to satisfy the goals for both Level 2 and Level 3
- The point to note is that the process areas listed in Level 2 are not listed in Level 3, and vice versa

- The same holds true for Maturity Level 4 and Maturity Level 5
- You must satisfy all of the goals in the previous levels plus the goals for the current level in order to attain the maturity level rating
- Each maturity level consists of process areas. Each process area contains goals that must be satisfied. Each goal has certain practices or actions associated with it

Process Areas

- Each maturity level consists of several process area
- A process area is a group of practices or activities performed collectively in order to achieve a specific objective

Goals

- Each PA has several goals that need to be satisfied in order to satisfy the objectives of the PA. There are two types of goals:
  - ✓ Specific goals (SG): goals that relate only to the specific PA under study
  - ✓ Generic goals (GG): goals that is common to multiple process areas throughout the model. These goals help determine whether the PA has been institutionalized

Practices

- Practices are activities that must be performed to satisfy the goals for each PA. Each practice relates to only one goal. There are two types of practices:
  - ✓ Specific practices (SP): practices that relate to specific goals
  - ✓ Generic practices (GP): practices associated with the generic goals for institutionalization

Example: Specific Goal and Specific Practice

- Specific Goal (from Requirements Management PA)
  - ✓ Requirements are maintained and accurately reflected in project plans, activities and products.
- Specific Practice (from Requirements Management PA)
  - ✓ Maintain the traceability of requirements to their source requirements.
- Generic Goal (from Maturity Level 2)
  - ✓ Institutionalize a Managed Process.
- Generic Practice (from Maturity Level 2)
  - ✓ Establish an Organizational Policy.

Common Features

- Common features are a means of categorizing Generic practices.
- Commitment to perform: establishment of management policies
- Ability to perform: establishment and maintenance of plans, resources, assigned responsibility and authority, and training
- Directing implementation: measurement, control, and performance practices
- Verification: ensure implementation and compliance

Another way to look at Common Features

- Common feature categories are very similar across process areas.
- They are referred to as Institutionalization Common Features because they:
  - ✓ Ensure the process areas are effective, repeatable and lasting
  - ✓ Provide needed infrastructure support

Common Feature Examples

- Commitment to perform:
  - ✓ Establish and maintain an organizational policy for planning and performing the requirements management process.
- Ability to perform:
  - ✓ Train the people performing or supporting the requirements management process as needed.
- Directing implementation:
  - ✓ Place designated work products of the requirements management process under appropriate levels of configuration management.
- Verification:
  - ✓ Review the activities, status, and results of the requirements management process with higher level management and resolve issues.

Summary

- Staged
  - ✓ Structured for implementation based on proven grouping and ordering of processes
  - ✓ The CMMI model should be applied using intelligence, common sense, and professional judgment

References

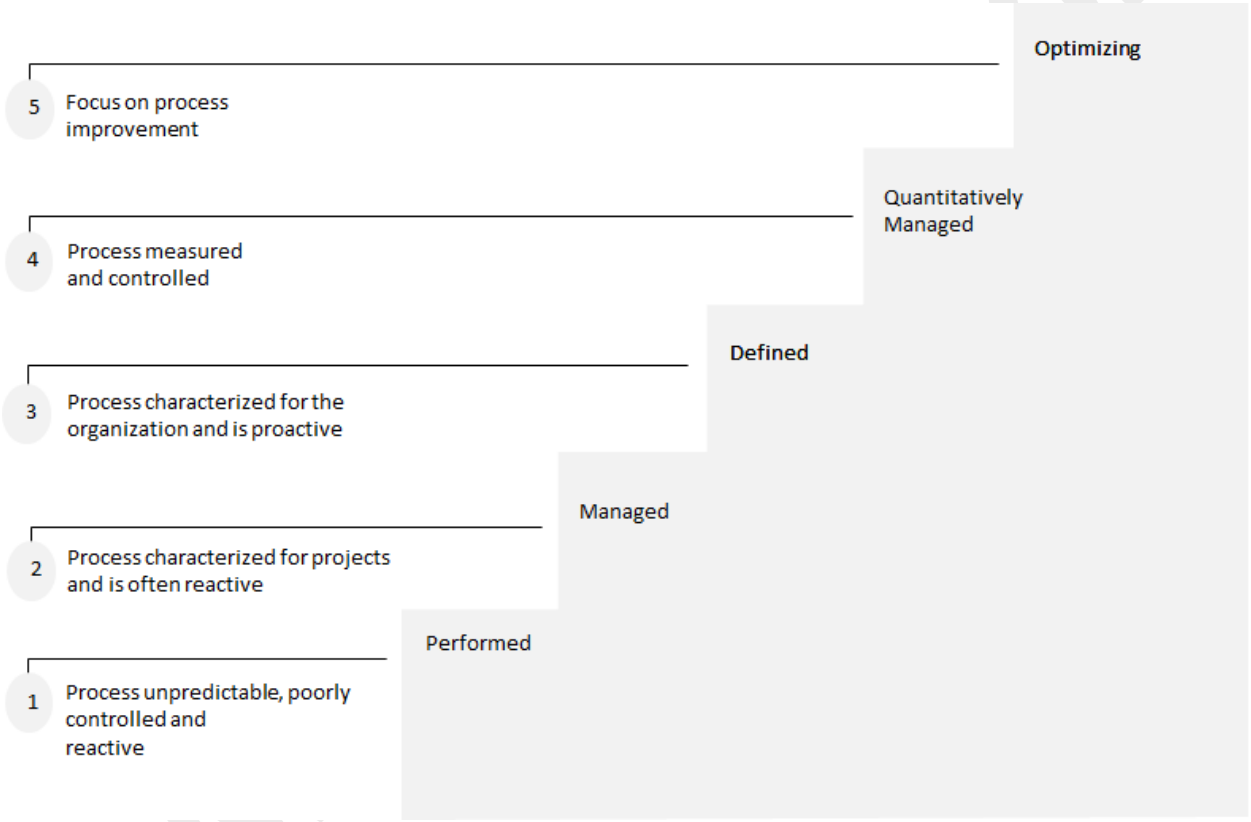
- http://www.sei.cmu.edu/general/index.html
- Interpreting the CMMI: A Process Improvement Approach, Second Edition, by Margaret K. Kulpa and Kent A. Johnson, Auerbach Publication, 2008 (electronic file), (Chapter 1-4)

Lecture 15
CMMI Staged – Maturity Level 2

Staged Representation

- The staged representation is the approach used in the Software CMM. It is an approach that uses predefined sets of process areas to define an improvement path for an organization. This improvement path is described by a model component called a maturity level
- A maturity level is a well-defined evolutionary plateau toward achieving improved organizational processes

The Maturity Levels



- The biggest hurdle that most organizations face when embarking on the journey from an immature organization to a more mature one is the jump from Level 1 to Level 2
- Level 1 is characterized by ad hoc processes; that is, processes that the people doing the work have created themselves to accomplish their tasks
- The problem with this method (creating individual processes) is that redundant work is often done, people do not share their methods across the organization, and some approaches are in opposition to actually making the organization run more effectively
- While some individual approaches may work for a particular individual, that person's approach may actually conflict with work being done downstream
- The results are more rework, delays, and frustration

- Turf wars are common, and the organization functions due to the heroics of its people. When these people move on (or burn out), the organization suffers
- Level 2 is characterized by individuals sharing their lessons learned and best practices, and devising preliminary processes that will function at the project level, and in some cases, across the organization as a whole
- Level 2 focuses on management issues that affect normal, day-to-day work routines. Level 2 consists of seven process areas that contribute to project management efficiencies
- There are seven process areas (PAs) that make up Level 2
    - ✓ Requirements Management
    - ✓ Project Planning
    - ✓ Project Monitoring and Control
    - ✓ Supplier Agreement Management
    - ✓ Measurement and Analysis
    - ✓ Process and Product Quality Assurance
    - ✓ Configuration Management
- Each process area has specific goals (specific to that process area) and generic goals that are applied to every process area in that maturity level
- These generic goals lead to institutionalizing the process area; that is, when an organization ensures that these goals are practiced consistently across the organization, the process area associated with these goals will continue to be applied appropriately in the organization even after those who created the procedures for this area have left

Generic Goal Level 2: Institutionalize a Managed Process

- Generic Practice 2.1 Establish an Organizational Policy
- Generic Practice 2.2 Plan the Process
- Generic Practice 2.3 Provide Resources
- Generic Practice 2.4 Assign Responsibility
- Generic Practice 2.5 Train People
- Generic Practice 2.6 Manage Configurations
- Generic Practice 2.7 Identify and Involve Relevant Stakeholders
- Generic Practice 2.8 Monitor and Control the Process
- Generic Practice 2.9 Objectively Evaluate Adherence
- Generic Practice 2.10 Review Status with Higher Level Management

PA 1 – Requirements Management

Requirements Management (REQM)

- The purpose of Requirements Management (REQM) is to manage the requirements of the project's products and product components and to identify inconsistencies between those requirements and the project's plans and work products
- Specific goal (SG1) for REQM is Manage Requirements

Specific Practices for REQM

- SP 1.1 Obtain an Understanding of Requirements
- SP 1.2 Obtain Commitment to Requirements
- SP 1.3 Manage Requirements Changes
- SP 1.4 Maintain Bidirectional Traceability of Requirements
- SP 1.5 Identify Inconsistencies between Project Work and Requirements

Discussion on REQM

- Requirements Management for Level 2 is all about managing already existing requirements; that is, those requirements that have been elicited from the customer, are documented, and are ready to be worked or are in the process of being worked
- It is at this level because if you don't set the scope you cannot control how much work will need to be done by the project
- Difference between Requirements Development PA at Level 3 and this PA
- There are no generic practices that directly map to this PA

Requirements Management

- Requirements Management includes understanding the requirements for all parts and components of the system (not just software); obtaining customer, management, and developer commitment to the requirements; managing changes to the requirements; maintaining traceability from the requirements forward in the development cycle, as well as backward to discover where changes may have introduced problems; and identifying inconsistencies between the requirements, ensuing work products, and ensuing activities
- Requirements Management feeds into many process areas
- This process area forms the basis of project development, so it touches just about every process area in the model
- Remember, the scope of the model has greatly expanded. Not only are you devising and managing requirements for software, but for all parts of the product

PA 2 – Project Planning

Project Planning (PP)

- The purpose of Project Planning (PP) is to establish and maintain plans that define project activities
- Specific goals for this process area
  - ✓ SG1 Establish Estimates
  - ✓ SG2 Develop a Project Plan
  - ✓ SG3 Obtain Commitment to the Plan

SG1 Establish Estimates – Specific Practices

- SP 1.1 Estimate the Scope of the Project
- SP 1.2 Establish Estimates of Work Product and Task Attributes

- SP 1.3 Define Project Life Cycle
- SP 1.4 Determine Estimates of Effort and Cost

SG2 Develop a Project Plan – Specific Practices

- SP 2.1 Establish the Budget and Schedule
- SP 2.2 Identify Project Risks
- SP 2.3 Plan for Data Management
- SP 2.4 Plan for Project Resources
- SP 2.5 Plan for Needed Knowledge and Skills
- SP 2.6 Plan Stakeholder Involvement
- SP 2.7 Establish the Project Plan

SG3 Obtain Commitment to the Plan – Specific Practices

- SP 3.1 Review Plans That Affect the Project
- SP 3.2 Reconcile Work and Resource Levels
- SP 3.3 Obtain Plan Commitment

Discussion on PP

- The generic practice that most closely matches this PA is GP 2.2 Plan the Process
- Other GPs that can trace guidance back to this PA include GP 2.3 Provide Resources, GP 2.4 Assign Responsibility, GP 2.5 Train People, and GP 2.7 Identify and Involve Relevant Stakeholders

PA 3 – Project Monitoring and Control

Project Monitoring and Control (PMC)

- The purpose of Project Monitoring and Control (PMC) is to provide an understanding of the project's progress so that appropriate corrective actions can be taken when the project's performance deviates significantly from the plan
- Specific goals for this process area
  - ✓ SG1 Monitor Project against Plan
  - ✓ SG2 Manage Corrective Action to Closure

SG1 Monitor Project against Plan – Specific Practices

- SP 1.1 Monitor Project Planning Parameters
- SP 1.2 Monitor Commitments
- SP 1.3 Monitor Project Risks
- SP 1.4 Monitor Data Management
- SP 1.5 Monitor Stakeholder Involvement
- SP 1.6 Conduct Progress Reviews
- SP 1.7 Conduct Milestone Reviews

SG2 Manage Corrective Actions to Closure – Specific Practices

- SP 2.1 Analyze Issues
- SP 2.2 Take Corrective Action
- SP 2.3 Manage Corrective Action

Things People Forget

- This process area is tied to the Project Planning PA. Without having a credible project plan, this process area cannot be satisfied. The project plan coming out of Project Planning is the basis for activities in the Project Monitoring and Control PA
- Document your reviews with management, and any other reviews and status meetings that you feel may be necessary. Informal meetings do not need to be documented; however, the results of those meetings may need to be documented, especially if the results will impact schedules, planning, staffing, or strategies

Discussion on PMC

- The generic practice that most closely matches this PA is GP 2.8 Monitor and Control the Process. This is where the reader is referred to consider taking corrective actions when deviations from a process are found
- Other GPs that relate to this PA are GP 2.7 Identify and Involve Relevant Stakeholders, and GP 2.10 Review Status with Higher Level Management
- Guidance given includes that reviews should be both periodic (that is, routinely scheduled and held as scheduled) and event-driven (when something unexpected happens or when an "event"—such as reaching a milestone—has occurred)
- Project Monitoring and Control includes monitoring the attributes of the project specified in the project plan; monitoring the staff availability and stakeholder involvement; monitoring and revising project risks; monitoring information handling; conducting reviews to ensure progress is being made (usually conducted at least at major milestone completion); bringing to light potential problems and issues, and analyzing those issues; and taking corrective action to ameliorate project issues

PA 4 – Supplier Agreement Management

The purpose of Supplier Agreement Management (SAM)

- The purpose of Supplier Agreement Management (SAM) is to manage the acquisition of products from suppliers
- Specific goals for this process area
  - ✓ SG1 Establish Supplier Agreements
  - ✓ SG2 Satisfy Supplier Agreements

SG1 Establish Supplier Agreements – Specific Practices

- SP 1.1 Determine Acquisition Type
- SP 1.2 Select Suppliers
- SP 1.3 Establish Supplier Agreements

SG2 Satisfy Supplier Agreements

- SP 2.1 Execute the Supplier Agreement
- SP 2.2 Monitor Selected Supplier Processes
- SP 2.3 Evaluate Selected Supplier Work Products
- SP 2.4 Accept the Acquired Product
- SP 2.5 Transition Products

Supplier Agreement Management (SAM)

- This process area does not apply to projects where the supplier or his employee is integrated into the project team, uses the same processes, and reports to the same management as the product developers/project members
- If the supplier or his employees are directly incorporated into the project team, and the employee is treated just like any other member of the team (except that his paycheck is signed by the supplier organization), then this PA does not apply to you
- If your organization does not have external suppliers (including contractors providing services) this process area may be not applicable for your organization
- However, considering the expanded nature of the model, this N/A seems less and less likely
- Your organization probably will have formal agreements for delivery or development or installation of hardware, tools, COTS products, simulators, and so forth. This area is not just about subcontracting, as in the CMM for Software
- Some guidance for releasing the product built by suppliers, as well as evaluating project risks and acceptance testing is described in this process area
- Things People Forget
    - ✓ Even though the wording specifies "products and product components," the practices also include services provided
    - ✓ Your products being developed need to be managed by the practices in the Supplier Agreement Management process area or the practices in the Project Planning and Project Monitoring and Control process areas
- This PA is the only PA that may be tailored out—that means that you do not have to implement this PA if it does not pertain to your organization
- This PA does not map to any generic practice

PA 5 – Measurement and Analysis

- The purpose of Measurement and Analysis (M&A) is to develop and sustain a measurement capability that is used to support management information needs
- Specific goals for this process area are
    - ✓ SG1 Align Measurement and Analysis Activities
    - ✓ SG2 Provide Measurement Results

SG1 Align Measurement & Analysis Activities - Specific Practices

- SP 1.1 Establish Measurement Objectives
- SP 1.2 Specify Measures
- SP 1.3 Specify Data Collection and Storage Procedures
- SP 1.4 Specify Analysis Procedures

SG2 Provide Measurement Results - Specific Practices

- SP 2.1 Collect Measurement Data
- SP 2.2 Analyze Measurement Data
- SP 2.3 Store Data and Results
- SP 2.4 Communicate Results

Measurement and Analysis

- This process area describes what to do when instituting a measurement process in your organization, not just which measurements to collect
- This process area should be considered global because all processes should be measured, and most work products also produce meaningful metrics

Why was M&A a Separate PA in CMMI?

- Because high-maturity organizations stated quite clearly that the key to success in process improvement is measurement and actually using those measures to make decisions and monitor the process improvement effort

Measurement and Analysis

- While instituting a measurement approach and repository is not an easy task, it is so necessary for success of any process improvement initiative that it is important that this task should be initiated early in the process improvement effort
- There is a difference between beginning and successfully implementing
- Organizations were good at collecting measurements but not in actually using the measurements. While creating this process area as a separate entity has not really ended that problem, it has made organizations more aware of the importance of measurements and tracking against those measurements
- Things People Forget
  - ✓ You have to tie your measurements back to meaningful business goals—not just collect something because it has always been collected or not just collect something because the book tells you to in the examples. So don't measure everything listed as an example—but, also, don't just pay lip service to this area. Prove that you are collecting the right measurements that will truly help track progress and failure of your process improvement effort and of your business activities
  - ✓ This process area is about defining a measurement effort in your organization. Don't forget to measure your measures. That is, you also must track the time and effort and effectiveness of your measurement effort itself

- The generic practice that most closely matches this PA is GP 2.8 Monitor and Control the Process
- The M&A process area, plus the Project Monitoring and Control PA must both be implemented to satisfy GP 2.8
- Measurements should focus on the process, its products, and its services

Basic Step-by-Step Approach to Measurement

- Select the process to measure
- Select the measures
- Determine when to collect data
- Determine how to collect data
- Record and store the information
- Analyze data for consistency and accuracy
- Chart the results
- Review the chart
- Do something (corrective actions)

PA 6 – Process and Product Quality Assurance

Process and Product Quality Assurance

- The purpose of Process and Product Quality Assurance (PPQA) is to provide staff and management with objective insight into processes and associated work products
- The specific goals for this process area are
  - ✓ SG1 Objectively Evaluate Processes and Work Products
  - ✓ SG2 Provide Objective Insight

SG1 Objectively Evaluate Processes and Work Products – Specific Practices

- SP 1.1 Objectively Evaluate Processes
- SP 1.2 Objectively Evaluate Work Products and Services

SG2 Provide Objective Insight – Specific Practices

- SP 2.1 Communicate and Ensure Resolution of Noncompliance Issues
- SP 2.2 Establish Records

Things People Forget

- You also need someone to independently review your QA activities and results. This review can be done by another organization in your company, outside your own organization, or by companies that specialize in performing IV&V or other formal external audits. Some organizations also use the SCAMPI appraisal itself as part of this external review
- PPQA is the most often abused process area that we see when performing appraisals. Do not abuse or misuse this PA! If you decide that it is too much trouble to properly implement PPQA in

your organization (that is, as defined in the CMMI), then don't bother to introduce the CMMI into your organization

Process and Product Quality Assurance

- The generic practice that most closely matches this PA is GP 2.9 Objectively Evaluate Adherence
- Process and Product Quality Assurance includes providing a strategy and procedures for objectively evaluating processes and products; identifying personnel to fulfill this role objectively; reporting quality issues and noncompliance; and producing reports that provide verification that quality reviews were conducted and their results

PA 7 – Configuration Management

Configuration Management

- The purpose of Configuration Management (CM) is to establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting, and configuration audits
- Specific goals for this process area are
  - ✓ SG1 Establish Baselines
  - ✓ SG2 Track and Control Changes
  - ✓ SG3 Establish Integrity

SG1 Establish Baselines – Specific Practices

- SP 1.1 Identify Configuration Items
- SP 1.2 Establish a Configuration Management System
- SP 1.3 Create or Release Baselines

SG2 Track and Control Changes – Specific Practices

- SP 2.1 Track Change Requests
- SP 2.2 Control Configuration Items

SG2 Establish Integrity – Specific Practices

- SP 3.1 Establish Configuration Management Records
- SP 3.2 Perform Configuration Audits

Things People Forget

- The organization needs to define when each type of baseline (formal or developmental) should be used

Discussion on CM

- The generic practice that most closely matches this PA is GP 2.6 Manage Configurations. The term configuration control is used instead of the term configuration management, when describing appropriate levels of configuration activities
- Configuration Management includes defining configuration items; developing or implementing support tools, techniques, procedures, and storage media; generating and maintaining baselines; tracking change requests and controlling changes; documenting the results of the configuration management effort; and performing configuration audits

References

- Interpreting the CMMI: A Process Improvement Approach, Second Edition, by Margaret K. Kulpa and Kent A. Johnson, Auerbach Publication, 2008 (electronic file), (Chapter 5)

Lecture 16
CMMI Staged – Maturity Level 3

Maturity Level

- Maturity Level 3 differs from Level 2 in that now an organizational way of doing business has been developed. What that means is that the best practices and lessons learned from the projects have bubbled up to the organizational level to create an organizational identity
- There are common, shared approaches for performing daily tasks on each project. For example, estimating the size of a project may be done using the Delphi Technique (basically subject matter experts discussing best-case and worst-case estimates), a standard metric may have been institutionalized (such as using function points instead of lines of code), and a standard tool may be in use
- to actually calculate the size
- This organizational way of doing business is documented in the Organization's Set of Standard Processes (OSSP)
- However, should a project need to tailor this OSSP to more adequately fit its needs, then that tailoring request is bought to a decision-making body (usually the Engineering Process Group [EPG]), and if appropriate, the request is granted
- An example may be a legacy system that calculates size by lines of code instead of by function point. Rather than reengineer the millions of lines of code in the system in order to use a tool to count function points, and rather than do it manually, the project is simply allowed to continue calculating size by lines of code. The Delphi Technique is still used, but lines of code is the measurement
- To perform at Level 3, an organization must have satisfied all of the goals for all of the process areas (PAs) in both Level 2 and Level 3
- Sometimes exceptions may be made. Caution should be exercised however
- Entire process areas are generally not allowed to be tailored out of consideration. Practices may be tailored out if replaced by sufficient alternative practices. Remember: the more tailoring done, the less likely an organization is to achieve improvement, and the less likely the organization is to achieve a maturity level through an appraisal

Attributes of a Process suggested by CMMI

Attributes of a Process

- Purpose—Purpose of the process.
- Inputs—Work products, plans, approval memoranda (usually nouns).
- Entry criteria—What must be triggered before this process can start? (Usually the exit criteria of the previous step or process. Usually stated as a verb.)
- Activities—Tasks that must be performed. These tasks are usually later broken down into the detailed procedures for how to perform the tasks
- Roles—Who does what? (Usually by position.)
- Measures—What measures does this process produce?

- Verification steps—What reviews are performed to determine that this process is followed and is producing the correct results? (Usually management and quality assurance reviews, but sometimes can include customer, peer, and project team reviews.)
- Outputs—Work products, plans, approved products (can be the completed inputs)
- Exit criteria—How do we know when it is time to stop this process? (Usually expressed in verbs, and usually becomes the entry criteria for the next process step or next process.)

Managed Process

- Another distinction is made concerning processes. A managed process is a process that tackles project management efforts, is planned and executed according to a policy, is monitored, and reviewed to ensure adherence to its process description
- This is the type of process expected at Maturity Level 2

Defined Process

- A defined process builds upon a managed process by creating an organizational process that is then tailored to fit the needs of a particular project and involves gathering information related to improvement efforts undertaken by the organization in order to improve both the organization-level process and the project-level process

Process Areas for Maturity Level 3

- Requirements Development
- Technical Solution
- Product Integration
- Verification
- Validation
- Organizational Process Focus
- Organizational Process Definition (with IPPD)
- Organizational Training
- Integrated Project Management (with IPPD)
- Risk Management
- Decision Analysis and Resolution

Maturity Level

- To satisfy the goals for Level 3, the goals for Level 2 must be satisfied as well
- This mandate holds true for both the specific goals (listed in each process area below) and the generic goals
- There are two generic goals for levels 2 and 3

Generic Goals for Levels 2 & 3

- GG2 Institutionalize a Managed Process
- GG3 Institutionalize a Defined Process

GG2 Institutionalize a Managed Process – Generic Practices

- GP 2.1 Establish an Organizational Policy
- GP 2.2 Plan the Process
- GP 2.3 Provide Resources
- GP 2.4 Assign Responsibility
- GP 2.5 Train People
- GP 2.6 Manage Configurations
- GP 2.7 Identify and Involve Relevant Stakeholders
- GP 2.8 Monitor and Control the Process
- GP 2.9 Objectively Evaluate Adherence
- GP 2.10 Review Status with Higher Level Management

GG3 Institutionalize a Defined Process – Generic Practices

- GP 3.1 Establish a Defined Process
- GP 3.2 Collect Improvement Information

PA 1 – Requirements Development (RD)

- The purpose of Requirements Development (RD) is to produce and analyze customer, product, and product component requirements
- Specific goals for this process area
  - ✓ SG1 Develop Customer Requirements
    - SP 1.1 Elicit Needs
    - SP 1.2 Develop the Customer Requirements
  - ✓ SG 2 Develop Product Requirements
    - SP 2.1 Establish Product and Product Component Requirements
    - SP 2.2 Allocate Product Component Requirements
    - SP 2.3 Identify Interface Requirements
  - ✓ SG3 Analyze and Validate Requirements
    - SP 3.1 Establish Operational Concepts and Scenarios
    - SP 3.2 Establish a Definition of Required Functionality
    - SP 3.3 Analyze Requirements
    - SP 3.4 Analyze Requirements to Achieve Balance
    - SP 3.5 Validate Requirements
- Requirements Development is where requirements are created and documented
- Requirements Management at Level 2 is where changes are administered
- Requirements Development gathers requirements and then must usually refine these requirements in some way—by stating them more clearly, determining whether they are redundant or inconsistent with other requirements, and breaking them down into more detailed and traceable requirements
- Some readers may think that they can skip this process area because they are not doing "development," that is, "new" development, because they are a Maintenance shop. The CMMI makes it very clear that no matter whether you are in new development or in maintenance

mode, there are always new requirements or changes to existing requirements that require definition, refinement, user interaction, and so forth. That is, all of the behaviors necessary to produce complete and accurate requirements, and discussed in this process area

- This process area introduces the Concept of Operations or, using the CMMI term, operations concept. This document is usually the first document written when determining whether to pursue development of a product. It is usually a high-level statement of the desired functionality, and a preliminary plan and schedule for development, to justify initiating this new project to higher-level management
- In the commercial these documents are generally not used. This can be a problem when following the CMMI because this document is expected in several practices throughout the model. Organizations that are not DOD-like generally rely on object-oriented techniques, use cases, or man–machine interface documentation to satisfy this concept
- Things People Forget
  - ✓ Maintenance shops also do development
  - ✓ Even if you get your requirements directly from the customer, some refinement and analysis of them is always necessary
  - ✓ Developing and analyzing requirements never stops until the entire application is either delivered to the customer for final handoff, or is dead and buried
- There are no generic practices that directly map to this process area
- Requirements Development includes collecting and eliciting customer requirements from all involved parties at all levels; breaking down those high-level requirements into lower-level, more detailed requirements, and assigning them to categories for further development; defining interfaces among the requirements and any other areas necessary to fulfill the requirements; more adequately defining and documenting the operational need, concept, scenario, and functionality desired; ensuring that the requirements are complete, required, and consistent; negotiating needs versus wants versus constraints; and validating the requirements against risk in the early phases of the project

PA 2 – Technical Solution (TS)

- The purpose of Technical Solution (TS) is to design, develop, and implement solutions to requirements. Solutions, designs, and implementations encompass products, product components, and product-related life-cycle processes either singly or in combination as appropriate
- Specific Goals for TS
  - ✓ SG1 Select Product Component Solutions
    - SP 1.1 Develop Alternative Solutions and Selection Criteria
    - SP 1.2 Select Product Component Solutions
  - ✓ SG2 Develop the Design
    - SP 2.1 Design the Product or Product Component
    - SP 2.2 Establish a Technical Data Package
    - SP 2.3 Design Interfaces Using Criteria
    - SP 2.4 Perform Make, Buy, or Reuse Analyses

- ✓ SG3 Implement the Product Design
  - SP 3.1 Implement the Design
  - SP 3.2 Develop Product Support Documentation

- In this process area, when the model refers to processes, the model generally does not mean process improvement–type processes, but design processes. That is, the processes they are talking about in this PA focus on the technical steps necessary to produce the product
- Peer reviews are mentioned in this process area
- Things People Forget
  - ✓ Alternative solutions should be investigated
- There are no generic practices that directly map to this process area
- Technical Solution includes determining how to satisfy the requirements via analysis of different alternatives and methods; creating operational scenarios; selecting solutions and follow-on designs; generating a technical data package that may include development methodologies, bills of material, life-cycle processes, product descriptions, requirements, conditions of use, rationale for decisions made, and verification criteria; defining and documenting detailed interface information; determining whether to make, buy, or reuse; and implementing the design and generating supporting documentation

PA 3 – Product Integration

- The purpose of Product Integration (PI) is to assemble the product from the product components, ensure that the product, as integrated, functions properly, and deliver the product
- Specific goals for this process area are
  - ✓ SG1 Prepare for product integration
    - SP 1.1 Determine Integration Sequence
    - SP 1.2 Establish the Product Integration Environment
    - SP 1.3 Establish Product Integration Procedures and Criteria
  - ✓ SG2 Ensure Interface Compatibility
    - SP 2.1 Review Interface Descriptions for Completeness
    - SP 2.2 Manage Interfaces
  - ✓ SG3 Assemble Product Components and Deliver the Product
    - SP 3.1 Confirm Readiness of Product Components for Integration
    - SP 3.2 Assemble Product Components
    - SP 3.3 Evaluate Assembled Product Components
    - SP 3.4 Package and Deliver the Product or Product Component
- This is where the product comes together and you get to see the results of your work
- It is also where you deliver the product, and that means you get paid
- This process area expects the project to demonstrate each step to the user
- This process area is not release management. Releasing products is covered somewhat in Configuration Management, Supplier Agreement Management, and Technical Solution
- Things People Forget
  - ✓ The nightly build does not satisfy all of the Product Integration process area practices. Just running the automated build nightly does not constitute planning an integration

sequence. You must analyze the problems that occur and see if any of the integrated components conflict with remaining components or modules or requirements. Also, the emphasis is on being proactive and identifying problems early, rather than being reactive to problems that pop up during the nightly run

- There are no generic practices that directly map to this process area
- Product Integration includes determining how to assemble the product and what the sequence of assembly should be; creating an operational environment in which to satisfactorily deploy the product; documenting procedures and criteria for integrating the product; ensuring adequate integration of interfaces; and delivering the product
- There are no generic practices that directly map to this process area

References

- Interpreting the CMMI: A Process Improvement Approach, Second Edition, by Margaret K. Kulpa and Kent A. Johnson, Auerbach Publication, 2008 (electronic file), (Chapter 6)

Lecture 17
CMMI Staged – Maturity Level 3
(Continue….)

Process Areas for Maturity Level 3

- Requirements Development
- Technical Solution
- Product Integration
- Verification
- Validation
- Organizational Process Focus
- Organizational Process Definition (with IPPD)
- Organizational Training
- Integrated Project Management (with IPPD)
- Risk Management
- Decision Analysis and Resolution

PA 4 – Verification

- The purpose of Verification (VER) is to ensure that selected work products meet their specified requirements
- Specific goals for this process area are
  - ✓ SG1 Prepare for Verification
  - ✓ SG2 Perform Peer Reviews
  - ✓ SG3 Verify Selected Work Products
- SG1 Prepare for Verification – Specific Practices
  - ✓ SP 1.1 Select Work Products for Verification
  - ✓ SP 1.2 Establish the Verification Environment
  - ✓ SP 1.3 Establish Verification Procedures and Criteria
- SG2 Perform Peer Reviews – Specific Practices
  - ✓ SP 2.1 Prepare for Peer Reviews
  - ✓ SP 2.2 Conduct Peer Reviews
  - ✓ SP 2.3 Analyze Peer Review Data
- SG3 Verify Selected Work Products – Specific Practices
  - ✓ SP 3.1 Perform Verification
  - ✓ SP 3.2 Analyze Verification Results
- The Verification PA allows the usage of test setups and test simulators. Sometimes, the same test setups and simulators may be used for Validation as well—you just use them for different purposes, looking for different things
- Acceptance testing is mentioned here
- Things People Forget
  - ✓ You don't need to test everything
  - ✓ You do need to test almost everything
  - ✓ You cannot test in quality
  - ✓ You must do both peer reviews and testing. You can peer review and test the same products or different products. Try to ensure total coverage of the product, by one means or the other, if possible

- ✓ What is a peer? If you are peer reviewing code, a peer is a coder. If you are peer reviewing a project plan, a peer is another project manager. Do not mix people of different job status—for example, do not mix project managers in with coders. They are not peers
- ✓ A peer review board is not one person
- ✓ Peer reviews are not the place to philosophic discussions. Keep it short and sweet and focus on finding errors
- ✓ You must ensure that the errors found during a peer review are resolved. Another peer review may be necessary. Quality Assurance should also check to see that these problems have been resolved

- Generic Practices
  - ✓ There are no generic practices that directly map to this process area
- Verification includes selecting which work products are to be verified; creating the environment necessary for verification of those products; documenting procedures and criteria for verification and then following those procedures; conducting peer reviews; and verifying the product and taking any corrective actions needed

PA 5 – Validation

- The purpose of Validation (VAL) is to demonstrate that a product or product component fulfills its intended use when placed in its intended environment
- Specific goals for this process area are
  - ✓ SG1 Prepare for Validation
  - ✓ SG2 Validate Product or Product Components
- SG1 Prepare for Validation – Specific Practices
  - ✓ SP 1.1 Select Products for Validation
  - ✓ SP 1.2 Establish the Validation Environment
  - ✓ SP 1.3 Establish Validation Procedures and Criteria
- SG2 Validate Product or Product Components – Specific Practices
  - ✓ SP 2.1 Perform Validation
  - ✓ SP 2.2 Analyze Validation Results
- Discussion on VAL
  - ✓ Validation involves creating an environment as close as possible to the environment in which the product will be used in order to perform final testing of the product. However, this is not always a logical, practical thing to do
- Things People Forget
  - ✓ You don't need to test everything. You do need to test almost everything. You cannot test in quality
  - ✓ Validation is not performed at the end of the project. It can be performed throughout the development life cycle, depending on the product being produced
  - ✓ Make sure you plan in detail any tests to be performed by the users or in the user environment

- ✓ If you are simulating the user environment, make sure it truly replicates the actual environment. This will require planning and testing of the environment itself
- ✓ If you are developing a Web-based system using the Web that your users will use, then you may have satisfied Validation without knowing about it. Basically, if you are developing and testing the system using the same resources and procedures that the users will use (or are using), you may have this PA covered

- General Practices
  - ✓ There are no general practices that directly map to this process are
- Validation asks: are you building the right product? It includes selecting products and approaches for validating products, generating the validation environment, documenting validation criteria and procedures; conducting the validation activities; and analyzing the results and any issues that arise from conducting the validation process

PA 6 – Organizational Process Focus

- The purpose of Organizational Process Focus (OPF) is to plan, implement, and deploy organizational process improvements based on a thorough understanding of the current strengths and weaknesses of the organization's processes and process assets
- SG1 Determine Process Improvement Opportunities – Specific Practices
  - ✓ SP 1.1 Establish Organizational Process Needs
  - ✓ SP 1.2 Appraise the Organization's Processes
  - ✓ SP 1.3 Identify the Organization's Process Improvements
- SG2 Plan and Implement Process Improvements – Specific Practices
  - ✓ SP 2.1 Establish Process Action Plans
  - ✓ SP 2.2 Implement Process Action Plans
- SG3 Deploy Organizational Process Assets and Incorporate Lessons Learned – Specific Practices
  - ✓ SP 3.1 Deploy Organizational Process Assets
  - ✓ SP 3.2 Deploy Standard Processes
  - ✓ SP 3.3 Monitor Implementation
  - ✓ SP 3.4 Incorporate Process-Related Experiences into the Organizational Process Assets
- Organizational Process Focus
  - ✓ OPF introduces who should be doing process improvement and what process improvement is. The Engineering Process Group (EPG) is mentioned here
  - ✓ The EPG is the group responsible for planning process improvement and implementing the plans
- Process Area
  - ✓ The process area essentially describes how to initiate, diagnose, evaluate, act, and learn from process improvement in an organization
  - ✓ Like IDEAL model
- Things People Forget
  - ✓ This PA probably has the most "churn" in the beginning of any process improvement effort. That is because the people who staff it (the EPG) are as unfamiliar with process improvement as everyone else in the organization. If you decide to staff this group with

personnel who have supposedly already "done" process improvement somewhere else, make sure they are flexible
    - ✓ Don't try to change everything at once. Do a little at a time
    - ✓ After one or two years on the EPG, rotate some of the members out! You need new blood, new ways of thinking, new ways of meeting new challenges. Plus, the members of the EPG are more likely to create and implement usable processes if they must return to their previous jobs and follow those processes
- Generic Practice
    - ✓ The generic practice that most closely maps to this process area is GP 3.2 Collect Improvement Information (most directly mapping to SP 3.4 Incorporate Process-Related Experiences into the Organizational Process Assets)
- Organizational Process Focus includes establishing a fundamental understanding of what process is and why it is important to an organization; assessing current processes in the organization and identifying areas in need of improvement; creating and following action plans for improvement; determining how to institute process improvement in the organization and what plans and documentation will be needed; and reviewing the process improvement effort itself and instituting improvements in this area

PA 7 – Organizational Process Definition (with IPPD)

- The purpose of Organizational Process Definition (OPD) is to establish and maintain a usable set of organizational process assets and work environment standards.
- For IPPD: Organizational Process Definition + IPPD also covers the establishment of organizational rules and guidelines that enable conducting work using integrated teams
- Specific Goal for Organizational Process Definition
    - ✓ SG1 Establish Organizational Process Assets
- SG1 Establish Organizational Process Assets – Specific Practices
    - ✓ SP 1.1 Establish Standard Processes
    - ✓ SP 1.2 Establish Life Cycle Model Descriptions
    - ✓ SP 1.3 Establish Tailoring Criteria and Guidelines
    - ✓ SP 1.4 Establish the Organization's Measurement Repository
    - ✓ SP 1.5 Establish the Organization's Process Asset Library
    - ✓ SP 1.6 Establish Work Environment Standards
- We'll discuss software only; IPPD is applicable to DOD projects specifically
- Things People Forget
    - ✓ How to document processes that can be used. Individuals create processes that are either too complex to properly implement or too high-level to properly implement. You will need to rewrite, re-plan, and re-pilot your processes and procedures
    - ✓ There are differences between the types of documentation you will need to produce
    - ✓ IPPD is optional
- Generic Practice
    - ✓ The generic practice that most closely matches this process area is GP 3.1 Establish a Defined Process

- ✓ Other generic practices that can trace guidance back to this process area include GP 3.2 Collect Improvement Information

- Organizational Process Definition includes generating the Organization's Set of Standard Processes (OSSP); describing various life cycles approved for use; documenting tailoring criteria and guidelines; and creating and maintaining the measurement repository and process asset library

PA 8 – Organizational Training

- The purpose of Organizational Training (OT) is to develop the skills and knowledge of people so they can perform their roles effectively and efficiently
- Specific goals for this process area are
  - ✓ SG1 Establish an Organizational Training Capability
  - ✓ SG2 Provide Necessary Training
- SG1 Establish an Organizational Training Capability – Specific Practices
  - ✓ SP 1.1 Establish the Strategic Training Needs
  - ✓ SP 1.2 Determine Which Training Needs Are the Responsibility of the Organization
  - ✓ SP 1.3 Establish an Organizational Training Tactical Plan
  - ✓ SP 1.4 Establish Training Capability
- SG2 Provide Necessary Training – Specific Practices
  - ✓ SP 2.1 Deliver Training
  - ✓ SP 2.2 Establish Training Records
  - ✓ SP 2.3 Assess Training Effectiveness
- Process Area
  - ✓ This PA expects a Strategic Training Plan coming from the OSSP, as well as business plans, process improvement plans, defined skill sets of existing groups, missing skill sets of existing groups, skill sets needed for any nonexistent groups necessary to be formed, mission statements, and vision statements
  - ✓ And all of these things are tied into training plans. That's a lot of documentation that many smaller organizations do not have and do not need. Small organizations tend to operate at the tactical planning level and not at the strategic level. This PA expects a Strategic Plan that leads to a Tactical Plan.
  - ✓ Organizational-level training plans bubble up from project-level training plans and needs. Organizations also need to evaluate the effectiveness of the training received
  - ✓ This process area is not about knowledge management. While you may define core competencies and certifications necessary, the concepts are not that similar. For that consult People CMM
- Things People Forget
  - ✓ A strategic vision is usually more than one year. So, strategic plans should include at least a vision of what the Training Organization or training vision should be in the future, with schedules and budgets for the coming year

- ✓ Tactical Plans may be included as part of Project Planning in the Project Plan. That is, if the project needs to plan for training, it will include the schedules and budget for that training in the Project Plan
- ✓ Some organizations have a two- to five-year strategic plan, which is more of a vision than a plan; then, a tactical plan for the coming year, which focuses on specific classes to be held, when, and how much they will cost; and then, the Project Plan (or Project Training Plan) that uses the strategic and tactical plans as input to create their training plan based on the needs of the project
- ✓ It doesn't matter what you call these plans, as long as this information is contained clearly and completely somewhere. We would hope that you would document this stuff in a straightforward, easily accessible manner

- Generic Practice
  - ✓ The generic practice that most closely matches this process area from an organizational point of view is GP 2.5 Train People
  - ✓ GP 2.2 Plan the Process matches this process area from a project perspective
- Organizational Training
  - ✓ Organizational Training includes determining the strategic training needs of the organization and how to achieve them; procuring or delivering the training; and tracking its effectiveness

PA 9 – Integrated Project Management (with IPPD)

- The purpose of Integrated Project Management (IPM) is to establish and manage the project and the involvement of the relevant stakeholders according to an integrated and defined process that is tailored from the organization's set of standard Processes
- For IPPD: Integrated Project Management + IPPD also covers the establishment of a shared vision for the project and the establishment of integrated teams that will carry out the objectives of the project
- Specific Goals for IPM
  - ✓ SG1 Use the Project's Defined Process
  - ✓ SG2 Coordinate and Collaborate with Relevant Stakeholders
- SG1 Use the Project's Defined Process – Specific Practices
  - ✓ SP 1.1 Establish the Project's Defined Process
  - ✓ SP 1.2 Use Organizational Process Assets for Planning Project Activities
  - ✓ SP 1.3 Establish the Project's Work Environment
  - ✓ SP 1.4 Integrate Plans
  - ✓ SP 1.5 Manage the Project Using the Integrated Plans
  - ✓ SP 1.6 Contribute to the Organizational Process Assets
- SG2 Coordinate and Collaborate with Relevant Stakeholders – Specific Practices
  - ✓ SP 2.1 Manage Stakeholder Involvement
  - ✓ SP 2.2 Manage Dependencies
  - ✓ SP 2.3 Resolve Coordination Issues
- We'll not talk about IPPD related specific practices

- Process Area
  - ✓ This PA is supposed to be the evolution of Project Planning, and Project Monitoring and Control from Level 2, plus more sophistication for Level 3. That means that this PA involves more rigorous techniques for planning and monitoring projects within the organization
  - ✓ In this PA, each project reviews the OSSP and tailors the OSSP to fit a project's specific needs. The result is called the project's defined process, and yes, it must be documented. This process is then used to help build the project plan
  - ✓ The difference between the standard processes, tailoring guidelines, and procedures mentioned in Organizational Process Definition (OPD) and here in Integrated Project Management (IPM) is that the documentation is created and stored in OPD and used in IPM on the projects
  - ✓ The difference between management at Level 2 and at Level 3 is that Level 3 uses a set of organizational plans, processes, and assets (templates, checklists) based on best practices and lessons learned
- Things People Forget
  - ✓ The Project's Defined Process (PDP). This process is actually one or more documents that describe how the project functions in a step-by-step manner (usually either by process area or by life-cycle phase). It uses the OSSP as input and tailors the OSSP to fit the project. In some cases, especially when the organization is small and non-diverse, the OSSP can be used as the PDP, as appropriate
- Generic Practice
  - ✓ The generic practice that most closely matches this process area is GP 3.1 Establish a Defined Process
  - ✓ Other generic practices that can trace guidance back to this process area include GP 2.7 Identify and Involve Relevant Stakeholders and GP 3.2 Collect Improvement Information
- Integrated Project Management includes defining a process or processes at the project-level when necessary that are tailored from the organizational process(es); using the processes and documentation developed from the organization; integrating all plans (including plans for each process area as well as project management plans) with the project's defined process; managing the project according to the plan; incorporating measurements, documentation, and improvements into the project or organizational-level repositories and processes; ensuring stakeholder involvement; tracking critical dependencies; and resolving issues

PA 10 – Risk Management

- The purpose of Risk Management (RSKM) is to identify potential problems before they occur so that risk-handling activities can be planned and invoked as needed across the life of the product or project to mitigate adverse impacts on achieving objectives
- Specific Goals for Risk Management
  - ✓ SG1 Prepare for Risk Management
  - ✓ SG2 Identify and Analyze Risks
  - ✓ SG3 Mitigate Risks

- SG1 Prepare for Risk Management – Specific Practices
  - ✓ SP 1.1 Determine Risk Sources and Categories
  - ✓ SP 1.2 Define Risk Parameters
  - ✓ SP 1.3 Establish a Risk Management Strategy
- SG2 Identify and Analyze Risks – Specific Practices
  - ✓ SP 2.1 Identify Risks
  - ✓ SP 2.2 Evaluate, Categorize, and Prioritize Risks
- SG3 Mitigate Risks – Specific Practices
  - ✓ SP 3.1 Develop Risk Mitigation Plans
  - ✓ SP 3.2 Implement Risk Mitigation Plans
- Process Area
  - ✓ The Risk Management process area is much more proactive, involving identification of risk parameters, formal strategies for handling risks, preparing risk mitigation plans, and structured risk assessments
  - ✓ Periodic and event-driven reviews should occur on the project to summarize the most critical risks that may occur. Make sure you review risks during periodic reviews
  - ✓ Why? Discussing risks as a events exposes projects – discuss this clearly
  - ✓ Plus, risk probability changes over time and over the course of the project
  - ✓ Disaster Recovery may be included as part of an organization's risk management culture and is included in a sub-practice. Considering the risks associated with Continuity of Operations has also been added
  - ✓ A risk repository can be built at the organization level
- Things People Forget
  - ✓ This PA is the big time of risk management. This PA is not just about listing risks and reviewing them at project meetings. It is about studying the risks and measuring their impact and probability on project activities
  - ✓ The main focus of this PA is on project risks. However, the same concepts can be applied to organizational risks
- There are no generic practices that directly map to this process area
- Risk Management includes identifying and categorizing risks; generating a risk management strategy; analyzing risks; documenting risk mitigation plans; mitigating risks; and monitoring the risk effort

PA 11 – Decision Analysis and Resolution

- The purpose of Decision Analysis and Resolution (DAR) is to analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria
- Specific goal for this process area is
  - ✓ SG1 Evaluate Alternatives
- SG1 Evaluate Alternatives – Specific Practices
  - ✓ SP 1.1 Establish Guidelines for Decision Analysis
  - ✓ SP 1.2 Establish Evaluation Criteria
  - ✓ SP 1.3 Identify Alternative Solutions

- ✓ SP 1.4 Select Evaluation Methods
- ✓ SP 1.5 Evaluate Alternatives
- ✓ SP 1.6 Select Solutions

- Process Area
  - ✓ Why is this process area needed? The rationale is to provide managers and analysts with a mechanism to make decisions. This mechanism requires a formal approach to determine which issues need the formal approach of DAR and what that mechanism should be
  - ✓ Difficulty of using this PA
- Things People Forget
  - ✓ You must define when this PA should be used. Otherwise, some projects will use it for every decision, and some projects will not use it at all
  - ✓ You must create and document "established criteria." These criteria are used to judge proposed alternatives. Your criteria may already have been established as part of a technical or contractual requirement. However, the criteria for allowing or disallowing an alternative may change, depending on changes to the project involved with budget, personnel, schedule, safety, and other unanticipated factors. Document all changes to the criteria, why the change was made, who initiated the change, and the impact and results of the change
- There are no generic practices that directly map to this process area
- Decision Analysis and Resolution includes determining which decisions will be part of a formal decision-making evaluation process; creating evaluation criteria; determining the types of evaluation methods to use; and determining alternative solutions

Level 3's Process Areas

- Engineering PAs: Requirements Development, Technical Solution, Product Integration, Verification, and Validation
- Process Management PAs: Organization Process Focus, Organization Process Definition, and Organizational Training
- Project Management PAs: Integrated Project Management and Risk Management
- Support PAs: Decision Analysis and Resolution

References

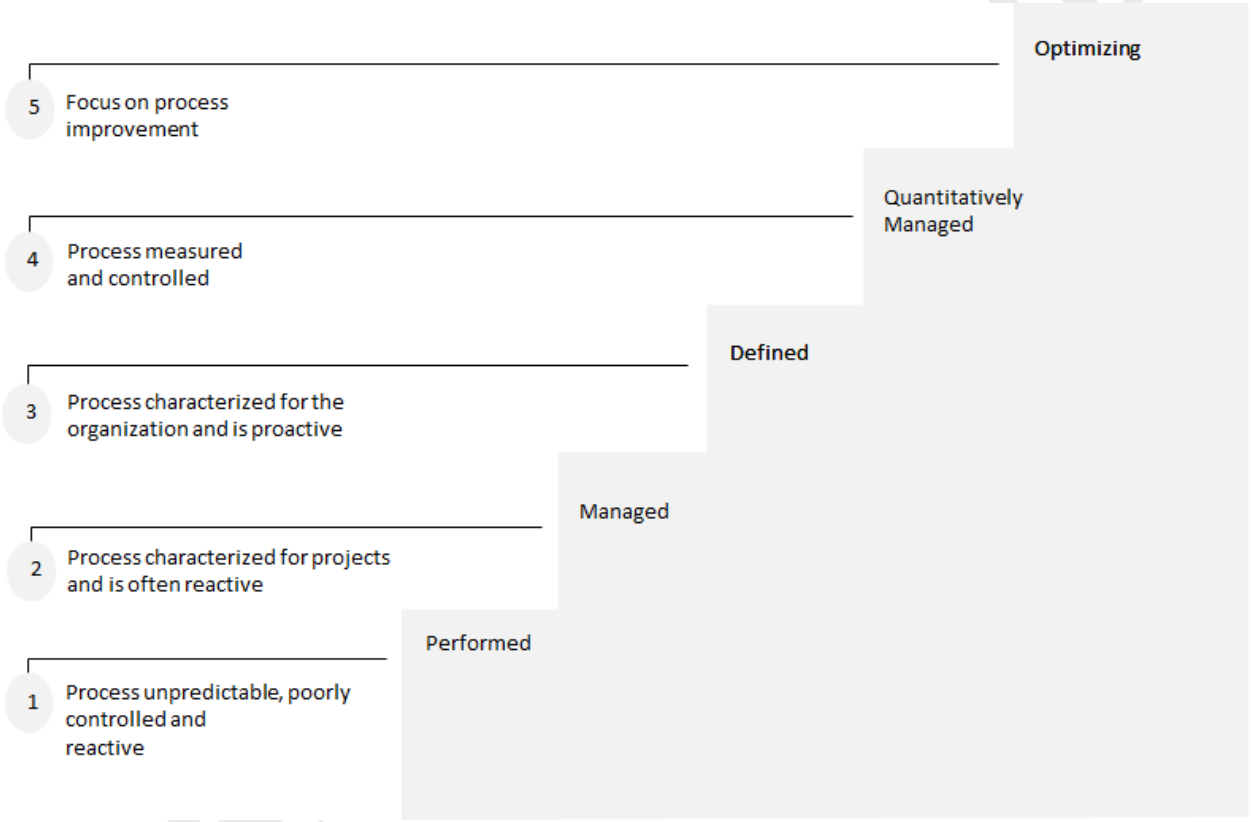
- Interpreting the CMMI: A Process Improvement Approach, Second Edition, by Margaret K. Kulpa and Kent A. Johnson, Auerbach Publication, 2008 (electronic file), (Chapter 6)

Lecture 18
CMMI Staged – Maturity Level 4

Staged Representation

- The staged representation is the approach used in the Software CMM. It is an approach that uses predefined sets of process areas to define an improvement path for an organization. This improvement path is described by a model component called a maturity level
- A maturity level is a well-defined evolutionary plateau toward achieving improved organizational processes

The Maturity Levels



Level 4

- Level 4 is all about numbers
- The projects are managed "by the numbers." Organizational decisions are made "by the numbers"
- Processes, services, product quality are all measured "by the numbers"
- At Level 4, the organization has achieved all of the goals of Levels 2 and 3
- Processes, although qualitatively stable and predictable at Level 3, can be proved to be quantitatively stable and predictable at Level 4
- Level 4 analyzes the data collected, determines special causes of variation from the norm, and supports quantitative management and control

- This is done to make your processes predictable
- To get "good" data, an organization usually has to collect data for several years, or at least through several projects and several life cycles of the projects. And when you first begin collecting data, they will not be consistent data
- Measurements data are collected beginning with Level 2 in the staged representation, and in most organizations, actually begin being collected at Level 1. The problem is the data are not clean and consistent because the processes used on the projects (where the data are collected and used) are not yet stable and consistent. Data in and of themselves are not magical. They simply reflect what is going on in the projects
- The point is, an organization cannot go to Level 4 overnight
- What problems do we see in organizations when they decide to move from Level 3 to Level 4?
- At Level 3, measures are collected and preliminary thresholds are established, usually relating to size and effort. If the thresholds are exceeded, some sort of corrective action is undertaken
- At Level 4, the control limits are based on years of historical data and trends analyses done on those data. More data are collected, and, therefore, more limits are established, monitored, and refined as necessary
- At Level 3, the data may be somewhat inconsistent and "dirty." Although in a perfect world we would like to see "clean" data at Level 3, the focus on Level 3 is on organizational process, not necessarily on stabilized, normalized, statistically accurate data—which is exactly what Level 4 expects
- One problem that we see in some organizations that have barely met the criteria for Level 3 is that the processes are not always followed consistently across the organization

Process Areas at Level 4

- Organizational Process Performance
- Quantitative Project Management

- At level 4, there are no additions to the list of generic goals at Level 4 from Level 3. What makes this maturity level different are the two process areas
  - ✓ GG2 Institutionalize a Managed Process
  - ✓ GG3 Institutionalize a Defined Process
- To satisfy the goals for Level 4, the goals for Levels 2 and 3 must be satisfied as well. This mandate holds true for both the specific goals and the generic goals

PA 1 – Organizational Process Performance

- The purpose of Organizational Process Performance (OPP) is to establish and maintain a quantitative understanding of the performance of the Organization's Set of Standard Processes (OSSP) in support of quality and process–performance objectives, and to provide the process–performance data, baselines, and models to quantitatively manage the organization's projects

Specific Goal for Level 4

- SG1 Establish Performance Baselines and Models

SG1 Establish Performance Baselines and Models – Specific Practices

- SP 1.1 Select Processes
- SP 1.2 Establish Process–performance Measures
- SP 1.3 Establish Quality and Process–performance Objectives
- SP 1.4 Establish Process–performance Baselines
- SP 1.5 Establish Process–performance Models

Process Area

- This process area includes measurements for both process and product. Additionally, service has also been specifically presented as an area appropriate for measurement. This process area combines these measures to determine both the quality of the process and the product in quantitative terms
- Process Performance Baselines
    - ✓ Process performance baselines and process performance models are included in goals for this process area
    - ✓ A process performance baseline (PPB) documents the historical results achieved by following a process. A PPB is used as a benchmark for comparing actual process performance against expected process performance
    - ✓ A process performance model (PPM) describes the relationships among attributes (for example, defects) of a process and its work products
    - ✓ A PPM is used to estimate or predict a critical value that cannot be measured until later in the project's life—for example, predicting the number of delivered defects throughout the life cycle
    - ✓ At Level 2, measures are collected, stored in a database per project, bubble up to an organizational database in Level 3, are reviewed for consistency and accuracy at Level 3, and then, at Level 4, have statistically-based controls applied to them
    - ✓ The OSSP must be understood from a statistical point of view
- The most common measurements in use for this process area are size, effort, cost, schedule, and product defect density
- The measurements for these data points are usually displayed in ranges and not by absolute points
- Performance-related measurements can include schedule variance (lateness), effort variance, and unplanned tasks
- Quality-related measurements may include rework and defects. These defects can be collected during all life-cycle phases, including requirements inspections, design inspections, code inspections, unit testing, integration testing, and system testing
- Common process-related measures are related to Productivity at the different phases of life cycles

- For example, in Testing, how many hours were spent deriving test cases versus how many tests were actually completed?
- Generic Goals
  - ✓ There are no new generic goals for Level 4 in the staged representation because the process areas include the basic tenets
- This process area covers both project-level and organization-level activities. Selecting processes to measure and selecting appropriate measures themselves can be iterative to meet changing business needs. Establishing quality and process objectives can be iterative as well, based on fixing special causes of variation
- Measurements should be tied to business objectives of the organization. So, if you are highly driven by time-to-market, you would focus on product defects and the scheduling effort. Decisions to release the product with an "appropriate" limit of defects would be made by senior management in order to make the schedule date. That "appropriate" part should be determined based on historical data (and analysis of that data and your measurement repository) for the number of defects that can be released into the marketplace, and the types of defects that can be released into the marketplace and still satisfy the customer and make the product work

Things People Forget

- The organization either builds too many models or not enough. This approach is the same approach we see when trying to select appropriate measures to collect at Level 2 and in trying to write effective process documentation when beginning the process improvement effort
- Nobody really understands the models that are built (except maybe the guy who built them)
- The models aren't really used to make decisions

PA 2 – Quantitative Project Management

- The purpose of Quantitative Project Management (QPM) is to quantitatively manage the project's defined process to achieve the project's established quality and process–performance objectives

Specific Goals for this PA

- SG1 Quantitatively Manage the Project
- SG2 Statistically Manage Sub-process Performance

SG1 Quantitatively Manage the Project – Specific Practices

- SP 1.1 Establish the Project's Objectives
- SP 1.2 Compose the Defined Process
- SP 1.3 Select the Sub-processes That Will Be Statistically Managed
- SP 1.4 Manage Project Performance

SG2 Statistically Manage Sub-process Performance – Specific Practices

- SP 2.1 Select Measures and Analytic Techniques
- SP 2.2 Apply Statistical Methods to Understand Variation
- SP 2.3 Monitor Performance of the Selected Sub-processes
- SP 2.4 Record Statistical Management Data

Process Area

- In this process area, usage of the organizational level measurement repository is refined. This PA describes what projects need to do to manage quantitatively
- Experienced managers and measurement personnel identify measures
- Senior level project personnel collect the measures
- Projects use the measures
- Training for each role needs to be addressed
- Project managers should do at least a weekly review of the project measures and how they are being used. This information is usually communicated to senior management
- A measurement group is usually needed to support measurement activities. Collection of data is easier if automated tools are used
- There can be several organizational measurement repositories, or layers within one overall repository, so as to not mix data that may lead to misleading numbers and bad decisions. Repositories require years of historical data using the same, normalized data, and reviews and analyses of these data. Training and practice in this effort need to occur. Running projects quantitatively is not an overnight transition

Things People Forget

- Make sure the project manager (or whoever will actually use these data to manage the project) is involved in determining which measures he will need
- Train the people collecting the data and the people who will use the data
- Make sure that decisions are made that show that these data and models were used. If they are not used, they do not really exist. Show how the project's defined process was changed, reviewed, or influenced by these data
- Make sure you have examples of using process, product, and quality measures and models

Quantitative Project Management

- Quantitative Project Management includes quantitatively defining project objectives; using stable and consistent historical data to construct the project's defined process; selecting sub-processes of the project's defined process that will be statistically managed; monitoring the project against the quantitative measures and objectives; using analytical techniques to derive and understand variation; and monitoring performance and recording measurement data in the organization's measurement repository
- Level 4 is where senior management commitment and participation really come to the fore. Business decisions are supposed to be made based on the numbers

- Have you ever sat in any senior- and executive-level meetings? You are lucky if you get ten minutes with these people. And they are not overly fond of viewing slide after slide of esoteric charts and graphs. They want to know the bottom line—are we making money? And the one chart that they all love, which is not particularly popular in the world of statistics is the pie chart
- Most small organizations will find this level very difficult to implement as written, based on the number of people needed to make this run smoothly and based on the type of expertise needed. And this level may not prove all that beneficial (using cost-benefit analyses) to these organizations anyway

CMMI Staged – Maturity Level 5

Staged Representation

- The staged representation is the approach used in the Software CMM. It is an approach that uses predefined sets of process areas to define an improvement path for an organization. This improvement path is described by a model component called a maturity level
- A maturity level is a well-defined evolutionary plateau toward achieving improved organizational processes

Level 5

- At Level 5, an organization has achieved all of the goals of Levels 2, 3, and 4
- The major difference between Level 4 and the next level, Level 5, is that Level 4 analyzes the data collected, determines special causes of variation from the norm, and supports quantitative management and control
- Level 5 addresses common causes of variation. So, for Level 4, an organization needs data that are stable and consistent. The major preoccupation of assessors when reviewing Level 4 is, "Did this organization mix apples and oranges? Are the data really accurate? Did this organization collect the right data and did they collect the data right?"
- Level 5 concentrates on improving the overall quality of the organization's processes by
  - ✓ identifying common causes of variation
  - ✓ determining root causes of the conditions identified
  - ✓ piloting process improvements
  - ✓ incorporating the improvements and corrective actions into the Organization's Set of Standard Processes (OSSP) or, as appropriate, just the project's defined process
- Although innovative, radical approaches to introduce change into an organization are often undertaken, most organizations have found that an incremental approach works better and has longer-lasting results
- Process Areas at Level 5
  - ✓ Organizational Innovation and Deployment
  - ✓ Causal Analysis and Resolution
- There are no additions to the list of generic goals at Level 5 from Level 3. What makes this maturity level different is the two process areas

- ✓ GG2 Institutionalize a Managed Process
- ✓ GG3 Institutionalize a Defined Process

- To satisfy the goals for Level 5, the goals for Levels 2, 3, and 4 must be satisfied as well. This rule holds true for both the specific goals and the generic goals

PA 1 – Organizational Innovation and Deployment

- The purpose of Organizational Innovation and Deployment (OID) is to select and deploy incremental and innovative improvements that measurably improve the organization's processes and technologies. The improvements support the organization's quality and process–performance objectives as derived from the organization's business objectives

Specific Goals for Level 5

- SG1 Select Improvements
- SG2 Deploy Improvements

SG1 Select Improvements – Specific Practices

- SP 1.1 Collect and Analyze Improvement Proposals
- SP 1.2 Identify and Analyze Innovations
- SP 1.3 Pilot Improvements
- SP 1.4 Select Improvements for Deployment

SG2 Deploy Improvements – Specific Practices

- SP 2.1 Plan the Deployment
- SP 2.2 Manage the Deployment
- SP 2.3 Measure Improvement Effects

Organizational Innovation and Deployment

- In OID, the proposals are subjected to quantitative analysis of proposed improvements. Metrics residing in the historical database, as well as defects and where they were introduced, are reviewed as well in order to determine where, when, and how to make improvements. Costs and benefits of the proposed versus actual improvements are also stud

Steps in this PA

- Submitting improvement proposals
- Reviewing and analyzing the proposals (including a cost-benefit review)
- Piloting the proposed improvement
- Measuring the improvement to see whether it has been effective in the pilot
- Planning the deployment of the improvement
- Deploying the improvement
- Measuring the effectiveness of the improvement across the organization or project

Things People Forget

- You must pilot your improvement! Only piloting will be able to accurately predict whether this improvement has a chance of being accepted and working in your organization. Most organizations do not want to take the time and trouble to pilot their improvements. Senior management does not understand, or does not want to understand, the importance of piloting. They feel, "Well, the stuff has been written—just throw it out to the projects. The project managers will make it work. That's why they are project managers." This is the wrong approach.
- You must plan the pilot and document critical success factors in a quantitative manner. Then a decision should be made, after reviewing the results against the critical success factors, whether the improvement should be implemented in the organization and what approach should be used to implement it (e.g., phased approach or en masse). More piloting may also be necessary

Organizational Innovation and Deployment

- Organizational Innovation and Deployment involves coordinating process improvement proposals submitted from the staff at various levels (improvement proposals may be related to innovative technology improvements); piloting selected improvements; planning and implementing the deployment of improvements throughout the organization; and measuring the effects of the improvements implemented

PA 2 – Casual Analysis and Resolution

- The purpose of Causal Analysis and Resolution (CAR) is to identify causes of defects and other problems and take action to prevent them from occurring in the future
- Specific goals for this process area are
  - ✓ SG1 Determine Causes of Defects
  - ✓ SG2 Address Causes of Defects

SG1 Determine Causes of Defects – Specific Practices

- SP 1.1 Select Defect Data for Analysis
- SP 1.2 Analyze Causes

SG2 Address Causes of Defects – Specific Practices

- SP 2.1 Implement the Action Proposals
- SP 2.2 Evaluate the Effect of Changes
- SP 2.3 Record Data

Process Area

- This process area looks at defects and determines their root cause. The simplest definition of a root cause is simply the one, most basic reason why the defect occurred (or the source of the defect) and if that cause is removed, the defect vanishes
- This PA identifies the root cause(s) and addresses the cause using a structured approach

Steps in this PA

- Look at the defects and problems in the organization
- Select data to analyze
- Analyze causes
- Prepare proposals to address the problems
- Implement the proposals
- Evaluate the effects of the changes
- Users of the CMMI for process improvement also review the CMM for Software for more advice or suggestions on other activities to include in order to satisfactorily complete this PA
- Kickoffs, sharing among projects, roles in the organization, rolling up project data/causes/problems into organizational level data, and integrating changes into the processes are described somewhat in the CMM model, and may benefit you in understanding this PA

Things People Forget

- You are analyzing defects of the process. The defects of the process may result in product defects, which are probably more noticeable in your organization
- The CMMI now makes very clear that the processes under review here should be quantitatively managed processes—that is, those processes that were studied, produced, and implemented in the organization using the concepts of Maturity Level 4. So, the processes, sub-processes, models, and benchmarks that the organization developed and used as part of their statistical control effort are the primary ones that are scrutinized here. So, your outputs from Level 4 form your inputs to Level 5

Causal Analysis and Resolution

- Causal Analysis and Resolution includes identifying defects and where in the process they were introduced; determining the causes of defects and their resolution; and defining methods and procedures to avoid introducing defects into the processes in the future

Summary of Level 5

- The focus at Level 5 is on improving processes, but now the OSSP is the controlling document that provides the primary focus. By using the OSSP (which by now must surely reflect true organizational functioning), process improvement can be engineered into the organization in a much more reasonable and efficient manner
- Level 5 tries to find common causes and fix them, which will result in overall improvements. Measurements are used to select improvements and reliably estimate the costs and benefits of attempting the improvements. Measurements are used to prove the actual costs and benefits of the improvements. These same measurements can be used to justify future improvement efforts
- There are two types of improvement strategies—innovative and incremental. Incremental builds on the foundation of earlier improvements. Innovative tends to introduce more radical and drastic methods of improvement

- Both can work in an organization, depending on the culture of the organization and the strength of its leaders, both politically and charismatically. However, most organizations have reported that incremental approaches to process improvement tend to have more long-lasting effects and lead to easier institutionalization
- At Level 5, the focus is on constantly reviewing and improving the processes, but these improvements must be introduced in a disciplined manner in order to manage and maintain process stability

References

- Interpreting the CMMI: A Process Improvement Approach, Second Edition, by Margaret K. Kulpa and Kent A. Johnson, Auerbach Publication, 2008 (electronic file), (Chapter 8)

Lecture 19
Introduction to CMMI Continuous Representation

Continuous Representation

- The continuous representation is the approach used in the SECM and the IPD-CMM. This approach allows an organization to select a specific process area and improve relative to it
- The continuous representation uses capability levels to characterize improvement relative to an individual process area

CMMI Model Components in the Continuous Representation

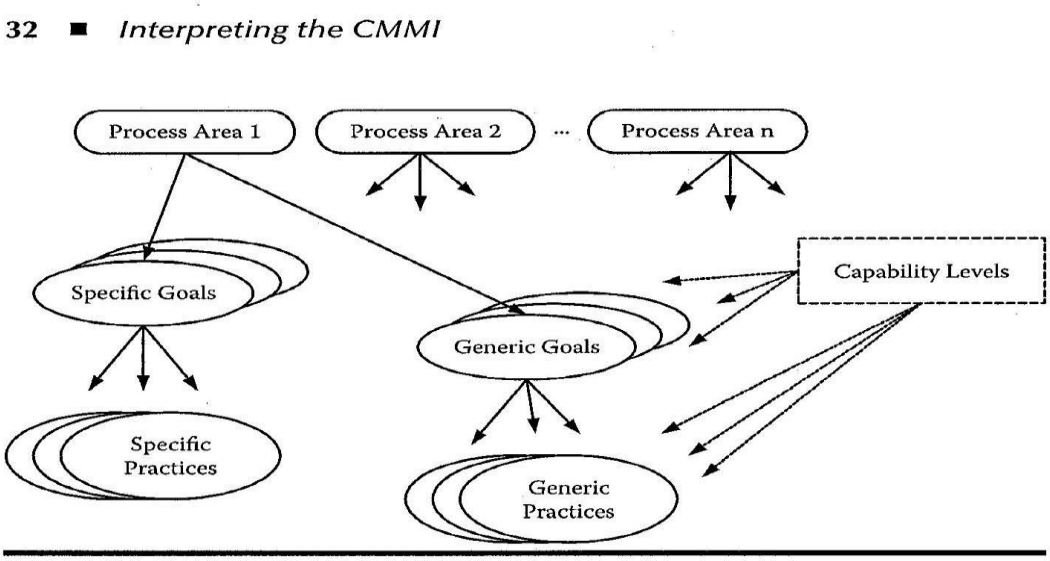


Figure 3.2 CMMI model components in the continuous representation.

Continuous Representation

- The continuous representation has specific goals that organize specific practices and generic goals that organize generic practices.
- Each specific and generic practice corresponds to a capability level
- Specific goals and specific practices apply to individual process areas
- The continuous representation uses the same basic structure as the staged representation
- However, each PA belongs to a Process Area Category
- A Process Area Category is just a simple way of arranging PAs by their related, primary functions

Process Area Categories

- Process Management
- Project Management
- Engineering
- Support

Process Management

- PAs in Process Management PA category consist of common functions related to defining, planning, implementing, and monitoring a process
- Notice that they all reside at the organizational level, not the project level, and are listed in increasing order of sophistication and complexity
- Process Management PAs
  - ✓ Organizational Process Focus
  - ✓ Organizational Process Definition (with Integrated Product and Process Development—IPPD)
  - ✓ Organizational Training
  - ✓ Organizational Process Performance
  - ✓ Organizational Innovation and Deployment

Project Management

- These PAs consist of functions related to planning, tracking, and controlling projects, and are listed in increasing order of sophistication and complexity
- Project Management PAs
  - ✓ Project Planning
  - ✓ Project Monitoring and Control
  - ✓ Supplier Agreement Management
  - ✓ Integrated Project Management (with Integrated Product and Process Development—IPPD)
  - ✓ Risk Management
  - ✓ Quantitative Project Management
- These process areas are all related in some way. They are categorized as Project Management Processes. Although an organization may select which processes to focus improvement efforts on when using the continuous representation, this representation seems to suggest that if your organization needs help in improving its project management approach to product development, start with these process areas in this order

Engineering

- These PAs consist of technical functions related to building and delivering a product, and are listed in increasing order of sophistication and complexity
- Engineering PAs
  - ✓ Requirements Development
  - ✓ Requirements Management
  - ✓ Technical Solution
  - ✓ Product Integration
  - ✓ Verification
  - ✓ Validation

Support

- These PAs consist of related support functions related to managing changes, ensuring quality, measuring results and activities, and structured decision making
- They are listed in increasing order of sophistication and complexity
- Support PAs
  - ✓ Configuration Management
  - ✓ Process and Product Quality Assurance
  - ✓ Measurement and Analysis
  - ✓ Decision Analysis and Resolution
  - ✓ Causal Analysis and Resolution
- The continuous representation does not overtly suggest a sequence to use for process improvement; however, a sequence is implied

Goals and Practices

- Specific goals and practices relate to specific process areas and relate to tasks that make sense for that process area only
- For example, Project Planning requires a project plan. Quantitative Project Management requires a process performance baseline
- Generic goals and practices relate to multiple process areas
- So, Requirements Management, if desiring a Capability Level 2, would have to establish an organizational policy, plan the process, and train people. To attain Capability Level 3, Requirements Management must do the capability level 2 actions, as well as define the process and collect improvement information relating to the process
- Project Planning and Organizational Process Focus would have to do the same as well
- So the generic goals and practices can be applied to all process areas
- Both specific goals and practices and generic goals and practices must be satisfied to achieve the capability level
- Basically, each capability level has one generic goal associated with it. Each generic practice maps to only one generic goal. The generic goals and generic practices map directly to one or more PAs, and basically summarize the concepts of each PA

Generic Goal 1: Achieve Specific Goals of Each Process Area

Generic Practices	Related Process Areas
GP 1.1: Perform Specific Practices of the Process Area	Specific practices of one or more Process Areas, depending on the extent of the model to be achieved

Generic Goal 2: Institutionalize a Managed Process

Generic Practices	Related Process Areas
GP 2.1: Establish an Organizational Policy	Project Planning

GP 2.2: Plan the Process	Project Planning
GP 2.3: Provide Resources	Project Planning
GP 2.4: Assign Responsibility	Project Planning
GP 2.5: Train People	Organizational Training Project Planning
GP 2.6: Manage Configurations	Configuration Management
GP 2.7: Identify and Involve Relevant Stakeholders	Project Planning Project Monitoring and Control Integrated Project Management (with IPPD)
GP 2.8: Monitor and Control the Process	Project Monitoring and Control Measurement and Analysis
GP 2.9: Objectively Evaluate Adherence	Process and Product Quality Assurance
GP 2.10: Review Status with Higher Level Management	Project Monitoring and Control

Generic Goal 3: Institutionalize a Defined Process

Generic Practices	Related Process Areas
GP 3.1: Establish a Defined Process	Integrated Project Management Organizational Process Definition
GP 3.2: Collect Improvement Information	Integrated Project Management Organizational Process Focus Organizational Process Definition

Generic Goal 4: Institutionalize a Quantitatively Managed Process

Generic Practices	Related Process Areas
GP 4.1: Establish Quantitative Objectives for the Process	Quantitative Project Management Organizational Process Performance
GP 4.2: Stabilize Sub-process Performance	Quantitative Project Management Organizational Process Performance

Generic Goal 5: Institutionalize an Optimizing Process

Generic Practices	Related Process Areas
GP 5.1: Ensure Continuous Process Improvement	Organizational Innovation and Deployment
GP 5.2: Correct Root Causes of Problems	Causal Analysis and Resolution

Continuous Representation

- The continuous representation has the same basic information as the staged representation, just arranged differently; that is, in capability levels not maturity levels, and process area categories
- The continuous representation focuses process improvement on actions to be completed within process areas, yet the processes and their actions may span different levels

What is a Capability Level?

- Capability levels focus on maturing the organization's ability to perform, control, and improve its performance in a process area
- This ability allows the organization to focus on specific areas to improve performance of that area

Capability Levels

- There are six capability levels:
  - ✓ Level 0: Incomplete
  - ✓ Level 1: Performed
  - ✓ Level 2: Managed
  - ✓ Level 3: Defined
  - ✓ Level 4: Quantitatively Managed
  - ✓ Level 5: Optimizing

Capability Level 0: Incomplete

- An incomplete process does not implement all of the Capability Level 1 specific practices in the process area that has been selected
- This is tantamount to Maturity Level 1 in the staged representation

Capability Level 1: Performed

- A Capability Level 1 process is a process that is expected to perform all of the Capability Level 1 specific practices
- Performance may not be stable and may not meet specific objectives such as quality, cost, and schedule, but useful work can be done. This is only a start, or baby step, in process improvement. It means you are doing something, but you cannot prove that it is really working for you

Capability Level 2: Managed

- A managed process is planned, performed, monitored, and controlled for individual projects, groups, or stand-alone processes to achieve a given purpose
- Managing the process achieves both the model objectives for the process as well as other objectives, such as cost, schedule, and quality
- You are actively managing the way things are done in your organization
- You have some metrics that are consistently collected and applied to your management approach

Capability Level 3: Defined

- A defined process is a managed process that is tailored from the organization's set of standard processes. Deviations beyond those allowed by the tailoring guidelines are documented, justified, reviewed, and approved

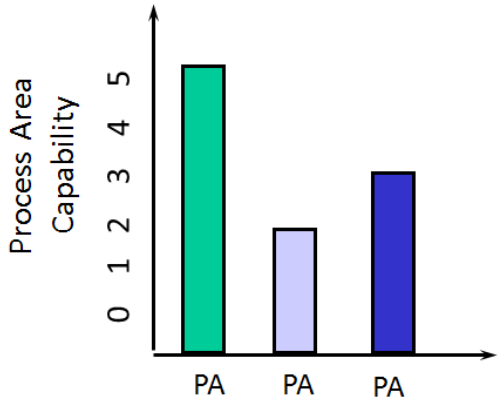
Capability Level 4: Quantitatively Managed

- A quantitatively managed process is a defined process that is controlled using statistical and other quantitative techniques. Product quality, service quality, process performance, and other business objectives are understood in statistical terms and are controlled throughout the life cycle

Capability Level 5: Optimizing

- An optimizing process is a quantitatively managed process that is improved based on an understanding of the common causes of process variation inherent in the process. It focuses on continually improving process performance through both incremental and innovative improvements. Both the defined processes and the organization's set of standard processes are targets of improvement activities. Level 4 focuses on establishing baselines, models, and measurements for process performance
- Level 5 focuses on studying performance results across the organization or entire enterprise, finding common causes of problems in how the work is done (the process(es) used), and fixing the problems in the process

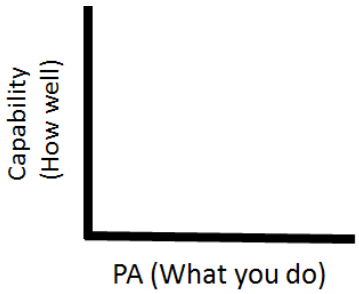
Process Areas in Continuous Representation



. . .for a single process area or a set of process areas

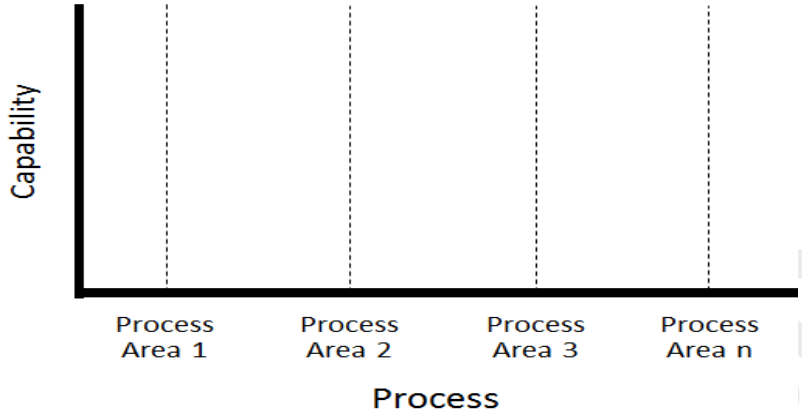
Process Area Capability Profile

- A process area capability profile may be represented by a set of points in two dimensions.
  - ✓ the process dimension
    - "What" you do
  - ✓ the capability dimension
    - "How well" you do it



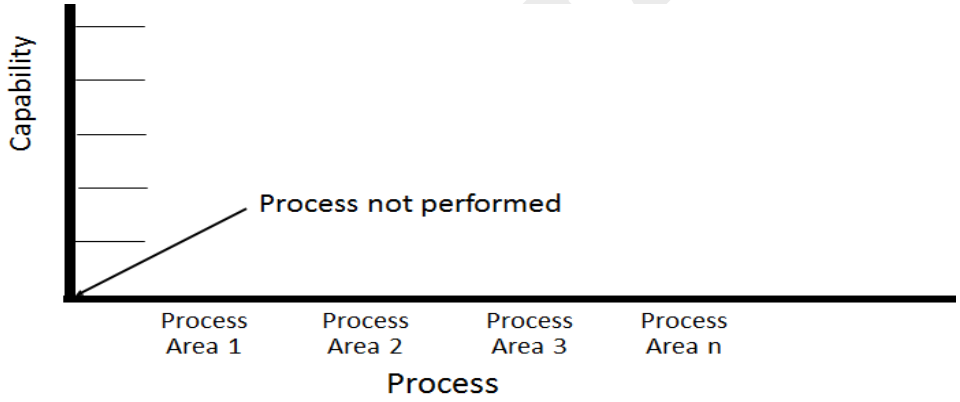
The Process Dimension

- The values on this axis describe what processes (described within Process Areas) you perform



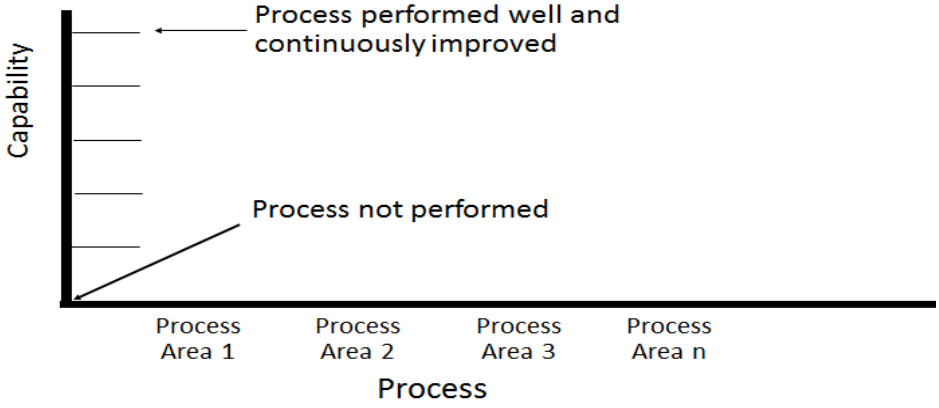
The Capability Dimension - 1

- The values on this axis describe how well you perform a process (called Capability Levels)

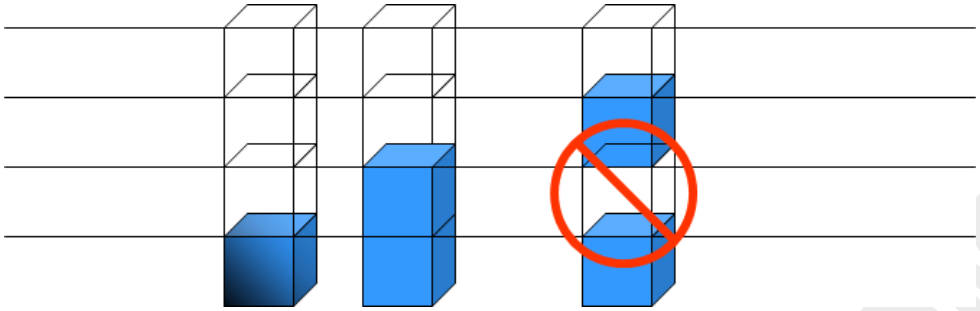


Capability Dimension - 2

- The values on this axis describe how well you perform a process (called Capability Levels)

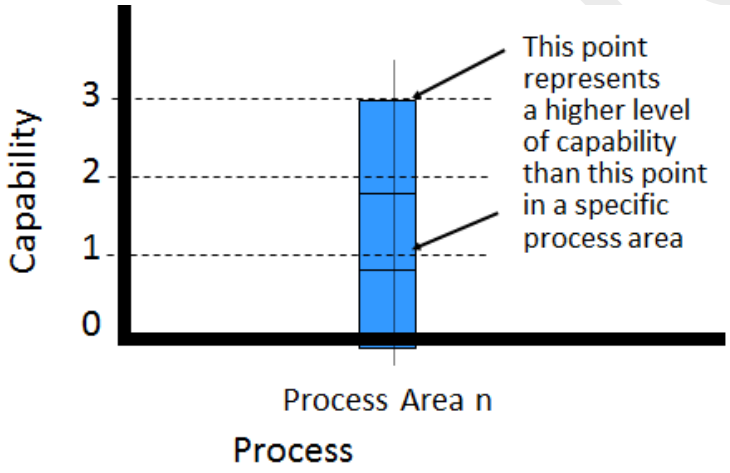


Capability Levels are Cumulative

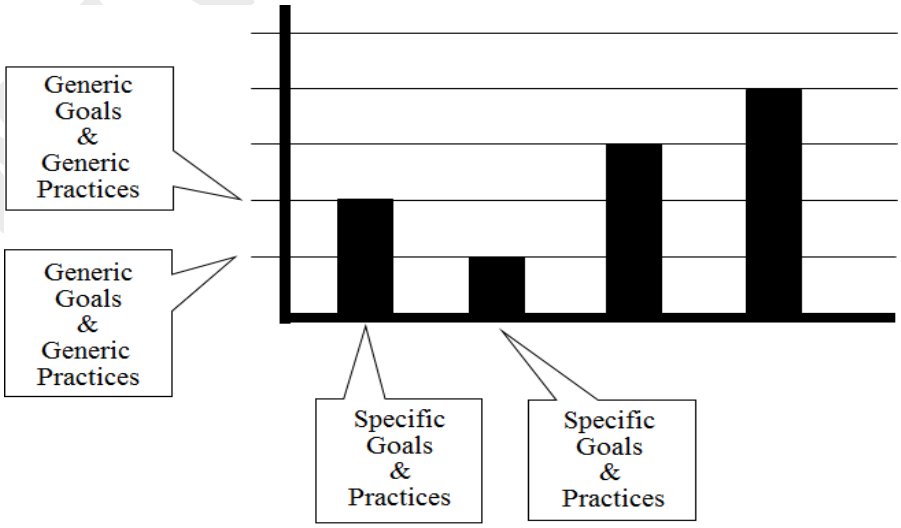


Representing Capability Levels

- The capability of a implemented process can be represented by a bar



Structure of the CMMI Continuous Representation



Target Profile

- A target profile is a list of process areas and their corresponding capability levels
- One example is when comparing maturity levels to capability levels. Capability Level 3 can only be determined as equivalent to Maturity Level 3 in the staged representation when all of the goals for all of the process areas at Maturity Levels 2 and 3 in the staged representation have been met. So the target profile 3 would include satisfying seven process areas at Maturity Level 2 plus eleven process areas at Maturity Level 3
- Target staging is a sequence of target profiles that describe the path of process improvement the organization will take. Care should be taken to ensure that dependencies between the generic practices and process areas are implemented. This is where the organization documents the PAs it will focus on, justifies this approach, and tracks the PAs back to business objectives

Achievement Profile

- Used in the continuous representation, this profile is basically a bar chart of each process area that shows how much of the PA has been achieved and how much has not been achieved

Generic Practices

- How do generic practices relate to process areas, maturity levels, and capability levels?
- Process areas at Maturity Level 2 in the staged representation include entire process areas for planning, managing changes, ensuring quality, and tracking progress. The generic practices for Capability Level 2 in the continuous representation also include statements for the same things—planning, managing changes, ensuring quality, and tracking progress
- Generic practices are used to determine whether generic goals have been satisfied. Generic goals must be satisfied to achieve a level—either a capability level or a maturity level
- What this means is that because an organization is expected to implement the generic practices, an organization using the continuous representation and selecting separate process areas to focus on must in reality also satisfy the basic concepts of Maturity Level 2 in the staged representation
- An organization may decide to use the continuous representation. This organization selects the process area of Technical Solution and is seeking a Capability Level 2 for it. To achieve this capability level, all of the specific practices for Technical Solution are expected to be implemented, and all of the generic practices for this level are expected to be instituted. That means that activities surrounding planning, managing changes, ensuring quality, and tracking progress for Technical Solution must be addressed
- To address these issues and institute the generic practices, the organization discovers that it must actually backup a bit and also focus on the process areas of Project Planning, Project Monitoring and Control, and Configuration Management. So, although it may appear that an organization can skip process areas when using the continuous representation, the actions of those process areas must be met
- The continuous representation was designed to tailor the model and process improvement approaches focusing on specific categories that may match those areas in your organizations where problems exist

- For example, let's say your organization does only Independent Verification and Validation. You do not write software. You produce no product. As an independent third party, you simply review those products produced by some other party, and you write your report. This type of organization might want to focus on the Verification, Validation, and Product Integration process areas

References

- http://www.sei.cmu.edu/general/index.html
- Interpreting the CMMI: A Process Improvement Approach, Second Edition, by Margaret K. Kulpa and Kent A. Johnson, Auerbach Publication, 2008 (electronic file), (Chapter 1-4)

Lecture 20
Personal Software Process (PSP)

Concept of PSP

- Many viewed the CMM as designed for large organizations and did not see how it could be applied to individual work or to small project teams….The SEI thus started a process research project to examine ways individual software engineers could apply level-5 process principles. After several years of research, means were devised to adapt 12 of the 18 CMM key process areas to the work of individual software engineers
  - ✓ Watts Humphrey

What is PSP?

- The PSP is a defined and measured software process designed to be used by an individual software engineer
- The PSP was developed by Watts Humphrey
- Its intended use is to guide the planning and development of software modules or small programs, but it is adaptable to other personal tasks

PSP

- Like the SEI Capability Maturity Model for Software, the PSP is based on process improvement principles
- While the CMM-SW was focused on improving organizational capability, the focus of the PSP is the individual engineer
- To foster improvement at the personal level, PSP extends process management and control to the software engineer
- With PSP, engineers develop software using a disciplined, structured approach
- Individual engineers follow a defined process, plan, measure, and track their work, manage product quality, and apply quantitative feedback to improve their personal work processes, leading to
  - ✓ better estimating
  - ✓ better planning and tracking
  - ✓ protection against over-commitment
  - ✓ a personal commitment to quality
  - ✓ the engineers' involvement in continuous process improvement
- Thus, both the individual and the organization's capability are improved

Cost and Schedule Management

- A critical business need for all organizations that develop software is better cost and schedule management
- Cost and schedule problems often begin when projects make commitments that are based on inadequate estimates of size and development resources

- PSP addresses this problem by showing engineers how to make better size and resource estimates using statistical techniques and historical data
- A more balanced estimation error means that the errors tend to cancel, rather than compound, when multiple estimates are combined

Quality Management

- A second critical business need is for improved software quality
- Poor quality management now limits our ability to field many critical systems, increases software development costs, and makes development schedules even harder to predict
- Many of these quality problems stem from the practice of relying on testing to manage software quality. But finding and fixing defects in test is costly, ineffective, and unpredictable
- The most efficient and effective way to manage software quality is through a comprehensive program focused on removing defects early, at the source
- The PSP helps engineers to find and remove defects where they are injected, before compile, inspection, unit test, integration test, or system test
- With fewer defects to find and remove in integration and system test, test costs are reduced sharply, schedules are more predictable, fewer defects are released to the field, maintenance and repair costs are reduced, and customer satisfaction is increased

Cycle Time

- A third critical business need is for reduced product cycle time. Cycle time can be reduced through better planning and the elimination of rework through improvements in product quality
- Accurate plans allow for tighter scheduling and greater concurrency among planned activities
- Better quality through early defect removal reduces waste and further increases planning accuracy by reducing a source of variation, the discovery and repair of defects in integration and system test
- With PSP, engineers learn how to gather the process data needed to minimize cycle time
- The data helps them to build accurate plans, eliminate rework, and reduce integration and system test by as much as four to five times

Organizational Process Improvement

- A fourth critical business need is process improvement. It is well understood that process improvement can increase competitive advantage, but it has been difficult to involve software engineers
- They often view process improvement and product quality as a management or staff activity, not as their personal responsibility
- With PSP, engineers gain personal experience with process improvement. They become process owners, directly involved in the measurement, management, and improvement of the software process

Background and Introduction

- All businesses are becoming software businesses. That is, more and more businesses now develop and incorporate software in the products they produce, or develop software to support the design, manufacture, or delivery of the products and services they provide
- Many are finding that as the software component of their business grows, schedule delays, cost overruns, and quality problems caused by software are becoming their number one business problem
- Software development has become the high-risk element in their business plans. Moreover, despite their best management efforts, they find their risk of failure increasing with each increase in the size or complexity of the software they produce
- Software businesses have at least one thing in common. The business is dependent on people. Software products are made of hundreds to millions of individual computer instructions, each one handcrafted by a software engineer
- And so, the technical practices and experience of their engineers largely determine the outcome of the development process, as has been widely recognized
- Software businesses also share a common set of needs: better cost and schedule management, improved software quality, and reduced software development cycle time
- PSP directly addresses these needs by improving the technical practices and individual abilities of software engineers, and by providing a quantitative basis for managing the development process
- By improving individual performance, PSP can improve the performance of the organization, as individual improvements in aggregate are reflected at the organizational level

How

- Data from the PSP improve planning and tracking of software projects
- Early defect removal results in higher quality products, as well as reductions in test costs and cycle time
- PSP provides a classroom setting for learning and practicing process improvement. Short feedback cycles and personal data make it easier to gain understanding through experience
- PSP helps engineers and their managers learn to practice quantitative process management. They learn to use defined processes and collect data to manage, control, and improve the work
- Finally, PSP exposes engineers to 12 of the CMM Key Process Areas (KPAs). They are better prepared to participate in CMM-based improvement
- The PSP provides a framework for teaching engineers about the software process, and a starting point from which they can evolve their own personal processes. The PSP is based on the same industrial practices that are found in the SEI CMM, but scaled-down for individual use
- It is a defined and measured, individualized process for consistently and efficiently developing high quality software modules or small programs
- It has been adapted to many other kinds of software engineering tasks, such as developing software requirements, software specifications, and test cases
- The PSP can also be scaled up to support small projects by integrating individual personal processes with a project process that was based on the PSP architecture

- In general, once learned, the principles and concepts in the PSP can be applied to any structured, repetitive task

Key Process Areas for PSP

- Software project planning
- Software project tracking/oversight
- Software process focus
- Software process definition
- Integrated software management
- Software product engineering
- Peer reviews
- Quantitative process management
- Quality management
- Defect prevention
- Technology change management
- Process change management

PSP in a Nutshell

- Baselining your existing personal software process
- Developing a planning process for your software development
- Measuring software size as part of the planning process
- Estimating software size in advance
- Estimating the required schedule and resources for software
- Conducting appropriate measures of one's process
- Conducting meaningful design and code reviews
- Performing software quality management
- Perform software design in a more formal fashion
- Verifying the design, using methods using methods such as object state machines, program tracing, etc.
- Scaling up the PSP to larger problems

Personal Improvement Process

- Define the quality goals
- Measure product quality
- Understand the process
- Adjust the process
- Use the adjusted process
- Measure the results
- Compare the results with the goal

References

- SEI website: www.sei.cmu.edu
- Introduction to the Personal Software Process by Watts Humphrey
- The Personal Software Process (PSPSM): An Empirical Study of the Impact of PSP on Individual Engineers (e-file), SEI TR, 1997
- The Personal Software Process by Watts Humphrey, SEI TR 0022, November 2000

Lecture 21
Implementing PSP

PSP

- The Personal Software Process (PSP) provides engineers with a disciplined personal framework for doing software work
- The PSP process consists of a set of methods, forms, and scripts that show software engineers how to plan, measure, and manage their work
- The PSP extends the improvement process to the people who actually do the work—the practicing engineers
- The PSP concentrates on the work practices of the individual engineers
- The principle behind the PSP is that to produce quality software systems, every engineer who works on the system must do quality work
- The PSP is designed to help software professionals consistently use sound engineering practices
- It shows them how to plan and track their work, use a defined and measured process, establish measurable goals, and track performance against these goals
- The PSP shows engineers how to manage quality from the beginning of the job, how to analyze the results of each job, and how to use the results to improve the process for the next project

Implementing Personal Software Process

- Managing time
- Managing commitments
- Key part of implementing personal software process is managing time accurately
- Let's talk about time management issues and learn ways to record and store time

Logic of Time Management

- You will likely spend your time this week much the way you spent it last week
- To make realistic plans, you have to track the way you spend time
- To check the accuracy of your time estimates and plans, you must document them and later compare them with what you actually do
- To make more accurate plans, determine where your plans were in error and what you could have done better
- To manage your time, plan your time and then follow the plan

Product Planning

- The size and important features of the product to be produced
- An estimate of the time required to do the work
- A projection of the schedule
- Relationship between product and period planning

How Time is Spent?

- Categorize your major activities
- Record the time spent on each major activity
- Record time in a standard way
- Keep the time data in a convenient place
  - ✓ Engineering Notebook
  - ✓ Electronic hand-held devices

Tracking Time in PSP

- Date
- Start time
- Stop time
- Interruption time
- Delta time
- Activity
- Comments
- Completed (or not)?
- Units

Hints on Logging Your Time

- Keep the engineering notebook/hand-held device with you at all times
- When you occasionally forget to record the start time, stop time, or interrupt duration, make an estimate as soon as you remember
- You may use a stopwatch to track interruptions
- Summarize your time promptly

Managing Your Time

- Decide how you want to spend your time
- Make a time budget
- Track the way you spend time against this budget
- Decide what changes to make to bring your actions into agreement with the budget

Time Stealers

- Interruptions
  - ✓ Telephone
  - ✓ Personal visitors
- Meetings
- Tasks you should have delegated
- Procrastination and indecision
- Acting with incomplete information
- Dealing with team members
- Crisis management (fire fighting)
- Unclear communication
- Inadequate technical knowledge
- Unclear objectives and priorities
- Lack of planning
- Stress and fatigue
- Inability to say 'No'
- Desk management and personal disorganization

Suggestion on Managing Variable Time

- What are your highest priority items?
- Are there some tasks that should be done at specific times?
- Are there activities you want to do as soon as you have time?

Time Management Strategies

- Always define your objectives as clearly as possible
  - ✓ Written goals, which can be reviewed regularly
  - ✓ Long term goals should impact daily activities
  - ✓ Without a goal or object, people tend to just drift personally and professionally
- Analyze your use of time
  - ✓ 'What is the most important use of my time, right now?'
- Have a plan
  - ✓ Successful people make lists constantly
  - ✓ Change priorities on a regular basis
- Action plan analysis
  - ✓ Problems will occur
  - ✓ A good plan identifies them early and seek out solutions
  - ✓ Good time management enables you to measure the progress towards your goals
  - ✓ What you can measure, you can control

- Let's now talk about commitments management
- Keep discussion on commitments short and sweet and move to PSP process levels

Defining Commitments

- What will be done
- The criteria for determining that it is done
- Who will do it
- When it will be done
- The compensation or other consideration to be given in return
- Who will provide this compensation or consideration

Responsibly Made Commitments

- Analyze the job before agreeing to the commitment
- Support the commitment with a plan
- Document the agreement
- If unable to meet the commitment, promptly tell the other party and try to minimize the impact on that party

Consequences of Not Managing Commitments

- Work required exceeds time available
- Failure to meet commitments
- Misplaced priorities
- Poor quality work
- Loss of trust
- Loss of respect for your judgment

PSP Process Levels

- The Baseline Personal Process
- Personal Project Management
- Personal Quality Management
- The Cyclic Personal Process

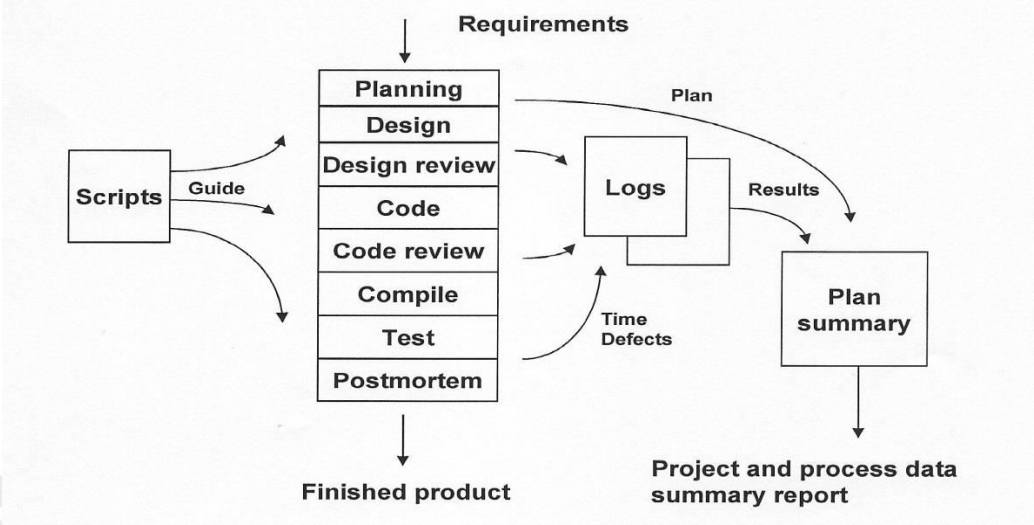
The Baseline Personal Process

- The baseline personal process provides an introduction to the PSP and establishes an initial base of historical size, time, and defect data
- Basic process measurement and planning are introduced here
- Development time, defects, and program size are measured and recorded on forms. A simple plan summary form is used to document planned and actual results
- A form for recording process improvement proposals (PIPs) is also introduced. This form provides engineers with a convenient way to record process problems and proposed solutions

Steps in Baseline PSP

Step	Phase	Description
1	Plan	Plan the work and document the plan
2	Design	Design the program
3	Code	Implement the design
4	Compile	Compile the program and fix and log all defects found
5	Test	Test the program and fix and log all defects found
6	Postmortem	Record actual time, defect, and size data on the plan

PSP Process Flow



Personal Project Management

- Personal project management techniques are the focus at this level, introducing size and effort estimating, schedule planning, and schedule tracking methods
- The estimated size of the newly developed code is the sum of all new objects, plus any modifications or additions to existing base code
- Predicted program size and effort are estimated using the statistical method linear regression
- Linear regression makes use of the historical relationship between prior estimates of size and actual size and effort to generate predicted values for program size and effort

- Finally, a prediction interval is calculated that gives the likely range around the estimate, based on the variance found in the historical data
- The prediction interval can be used to assess the quality of the estimate
- PSP uses the earned value method for schedule planning and tracking
- The earned value method is a standard management technique that assigns a planned value to each task in a project
- A task's planned value is based on the percentage of the total planned project effort that the task will take
- As tasks are completed, the task's planned value becomes earned value for the project
- The project's earned value then becomes an indicator of the percentage of completed work
- When tracked week by week, the project's earned value can be compared to its planned value to determine status, to estimate rate of progress, and to project the completion date for the project

Personal Quality Management

- Quality management methods are added to the PSP at this level
- These include: personal design and code reviews, a design notation, design templates, design verification techniques, and measures for managing process and product quality
- The goal of quality management in the PSP is to find and remove all defects before the first compile. The measure associated with this goal is yield
- Yield is defined as the percent of defects injected before compile that were removed before compile. A yield of 100% occurs when all the defects injected before compile are removed before compile
- Design and code reviews help engineers achieve 100% yield. These are personal reviews conducted by an engineer on his/her own design or code
- They are structured, data-driven review processes that are guided by personal review checklists derived from the engineer's historical defect data
- Engineers also begin using the historical data to plan for quality and control quality during development
- Their goal is to remove all the defects they inject before the first compile
- During planning, they estimate the number of defects that they will inject and remove in each phase. Then they use the historical correlation between review rates and yield to plan effective and efficient reviews
- During development, they control quality by monitoring the actual defects injected and removed versus planned, and by comparing actual review rates to established limits
- With sufficient data and practice, engineers are capable of eliminating 60% to 70% of the defects they inject before their first compile

The Cyclic Personal Process

- The cyclic personal process addresses the need to efficiently scale the PSP up to larger projects without sacrificing quality or productivity

- Productivity is highest between some minimum and maximum size range. Below this range, productivity declines due to fixed overhead costs. Above this range, productivity declines because the process scalability limit has been reached
- At this level, this scalability limit is addressed by introducing a cyclic development strategy where large programs are decomposed into parts for development and then integrated
- This strategy ensures that engineers are working at their maximum productivity and product quality levels, with only incremental, not exponential, increases in overhead for larger projects
- At this level high-level design, high-level design review, cycle planning, and development cycles are introduced based on the personal quality management processes

PSP Measures

- There are three basic measures in the PSP:
  - ✓ Development time
  - ✓ Size
  - ✓ Defects
- All other PSP measures are derived from these three basic measures

Development Time Measure

- Minutes are the unit of measure for development time
- Engineers track the number of minutes they spend in each PSP phase, less time for any interruptions such as phone calls, coffee breaks, etc.
- Using minutes is precise and simplifies calculations involving development time
- Recording interruptions to work reduces the number of time log entries, provides a more accurate measure of the actual time spent, and a more accurate basis for estimating actual development time.
- Tracking interruption time separately can help engineers deal objectively with issues that affect time management, such as a noisy work environment or inappropriate mix of responsibilities (e.g., software development and help desk support)
- Time log entries take substantially less than a minute to record, but provide a wealth of detailed historical data for planning, tracking, and process improvement

Defect Measure

- A defect is defined as any change that must be made to the design or code in order to get the program to compile or test correctly. Defects are recorded on the Defect Recording Log as they are found and fixed
- The example Defect Recording Log shows the information that is recorded for each defect: the date, sequence number, defect type, phase in which the defect was injected, phase in which it was removed, fix time,3 and a description of the problem and fix
- When an engineer injects a new defect while trying to fix an existing defect, proper accounting of fix time becomes more complicated
- A common mistake is to include the fix time for the new defect twice
- Each defect is classified according to a defect type standard
- A standard and easy to use classification scheme should be used

Size Measure

- The primary purpose of size measurement in the PSP is to provide a basis for estimating development time
- Lines of code or function points can be used
- Size is also used to normalize other data, such as productivity (LOC per hour) and defect density (defects per KLOC)

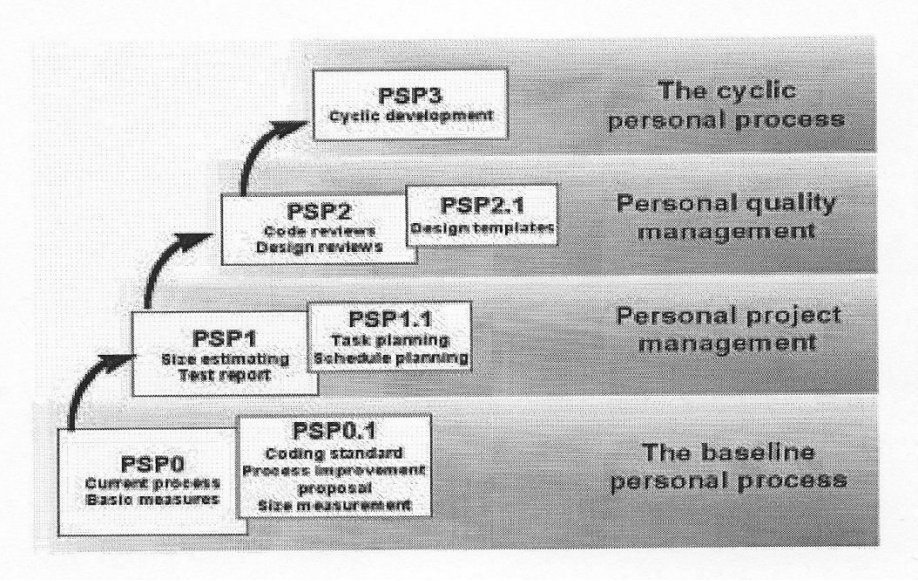
PSP Derived Measures

- PSP introduces new measures to help engineers manage and improve their performance
- These measures are derived from the three basic PSP measures: development time, defects, and size

Why PSP?

- When you trust your life to someone, you want them to behave professionally. Now consider your professional life. Should you entrust it to someone who is out of date? If you are looking for career guidance, would you ask someone who has not cracked a technical book since college? Probably not. The skill, knowledge, and ability you bring to your job will determine your future. Your future is in your hands
  - ✓ Watts Humphrey

PSP Process Levels



References

- SEI website: www.sei.cmu.edu
- Introduction to the Personal Software Process by Watts Humphrey
- The Personal Software Process (PSPSM): An Empirical Study of the Impact of PSP on Individual Engineers (e-file), 1997 TR
- The Personal Software Process by Watts Humphrey, SEI TR 0022, November 2000

Lecture 22
Review Up To Mid-Term

Process

- Once someone has worked out how to solve a problem, they can document the way in which that solution was derived as a process
- This then helps other people faced with similar problems to get started on their own solutions

Examples of Processes

- An instruction manual for a kitchen dishwasher describes the process of using that machine to clean dishes
- A cookery book describes a set of processes to prepare and cook various different types of meals
- A procedures manual in a bank describes the ways in which different banking processes such as agreeing a personal loan, correcting errors, etc. should be carried out
- A quality manual for software development describes the processes which should be used to assure the quality of the software. It may include descriptions of standards which are basis for the quality checking

Software Problems

- Technical
  - ✓ Need a technical solution
  - ✓ Easy to understand and solve
- Systemic
  - ✓ Need a process/management solution
  - ✓ Difficult to understand and solve

Solution to Software Problems

- Treat the software task as a process that can be controlled, measured, and improved
- Relate the required tasks, tools, methods with skill, training, and motivation of people involved

Software Processes

- Software engineering, as a discipline, has many processes
- These processes help in performing different software engineering activities in an organized manner
- The software process is the set of tools, methods, and practices that people use to produce (develop and maintain) software and associated products , e.g., project plans, design documents, code, test cases, and user manuals
- Where are the people? Aren't they the most important part of an organization?

Characteristics of Software Processes

- Requires creativity
- Interactions between a wide range of different people
- Engineering judgment
- Background knowledge
- Experience

Examples of Software Processes

- Software engineering development process (SDLC)
- Requirements engineering process
- Design process
- Quality assurance process
- Change management process

Process Improvement Objectives

- Quality improvement
- Resource reduction
- Schedule reduction

Process Improvements Planning

- What are the improvement goals?
- How can we introduce process improvements to achieve these goals?
- How should improvements be controlled and managed?

Six Steps to Software Improvements

- Understand the current status of development processes
- Develop a vision for the desired process
- Establish a list of required process improvement actions in order of priority
- Produce a plan to accomplish the required actions
- Commit the resources to execute the plan
- Start over at step 1

Levels of Software Process Models

- Universal or U Process Models
  - ✓ U process models provide high-level overview
  - ✓ Traditional Waterfall, Spiral, and other later process models fall in this category
  - ✓ Typically task-oriented
- Worldly or W Process Models
  - ✓ W process models are most useful to practicing software engineers
  - ✓ It guides the sequence of their working tasks and defines task prerequisites and results
  - ✓ They look like procedures
- Atomic or A Process Models
  - ✓ "A" process models are enormously detailed
  - ✓ They are used to automate specific process activity or use a standardized method or procedure to guide execution of a task

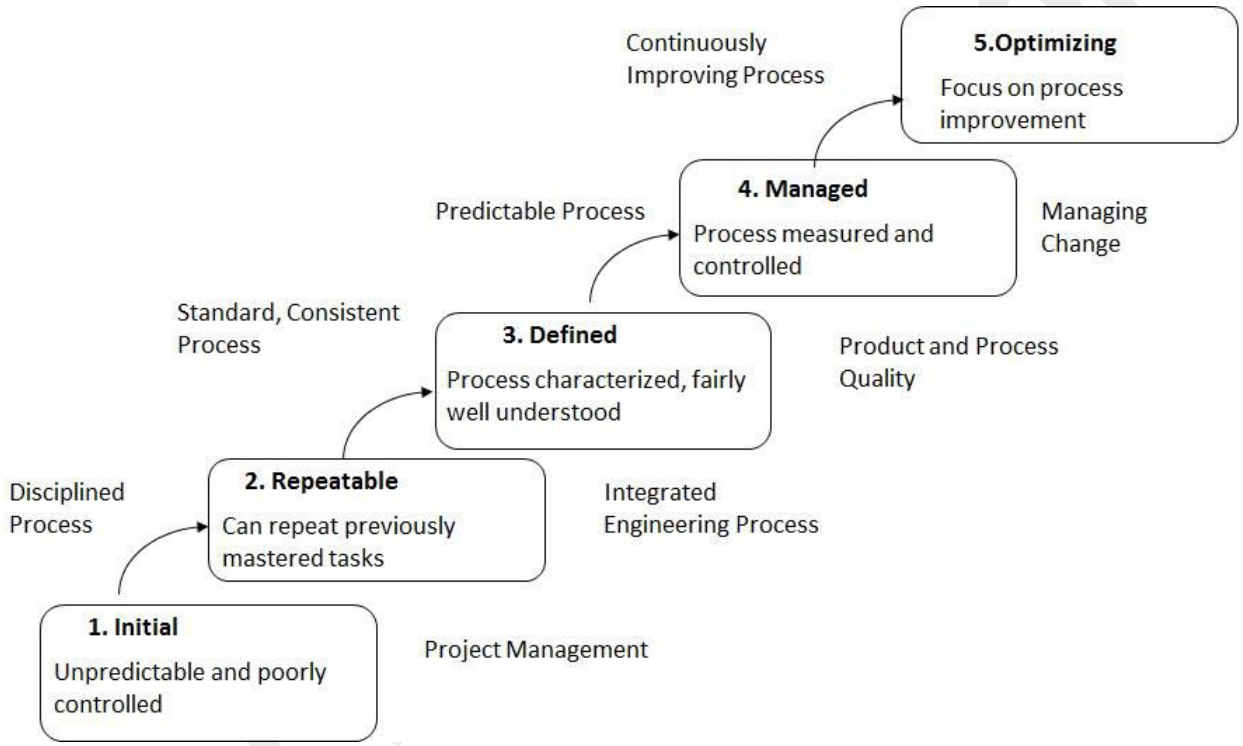
Process Modeling

- ETVX
- IDEF0

CMM-SW Levels

- Initial
- Repeatable
- Defined
- Managed
- Optimizing

The Five Levels of Software Process Maturity



CMMI

- People were successfully using the Capability Maturity Model, which was designed for improving software processes and measuring the maturity of software processes in an organization
- This success brought more and more attention to model-based process improvement in other areas

What Models Do I Use?

- Historically: Depends on the discipline that you want to model.
  - ✓ Software Engineering
  - ✓ Systems Engineering
  - ✓ Software Acquisition
  - ✓ Systems Security
  - ✓ etc.

What is CMM?

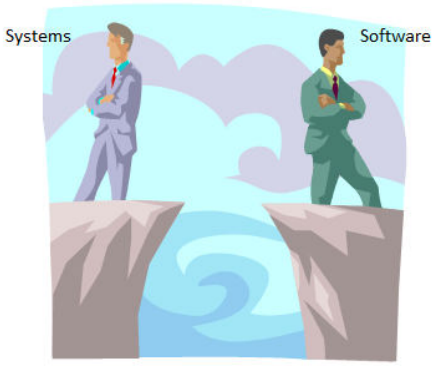
- Capability Maturity Model:
  - ✓ A reference model of mature practices in a specified discipline, used to assess a group's capability to perform that discipline
- CMMs differ by
  - ✓ Discipline (software, systems, acquisition, etc.)
  - ✓ Structure (staged versus continuous)
  - ✓ How Maturity is Defined (process improvement path)
  - ✓ How Capability is Defined (institutionalization)

Commonly Used CMMs

Software CMM	staged	software development
System Engineering CMM	continuous	system engineering
System Engineering Capability Model	continuous	system engineering
Software Acquisition CMM	staged	software acquisition
System Security Engineering CMM	continuous	security engineering
Personal Software Process	staged	individual software development
FAA-iCMM	continuous	software engineering, systems engineering, and acquisition
IPD-CMM	hybrid	integrated product development
People CMM	staged	workforce
SPICE Model	continuous	Software development

System and Software Divide

- Systems and software disciplines have traditionally not been well integrated
- The importance of software in systems has increased dramatically
- There was need to make the systems/software interface more seamless



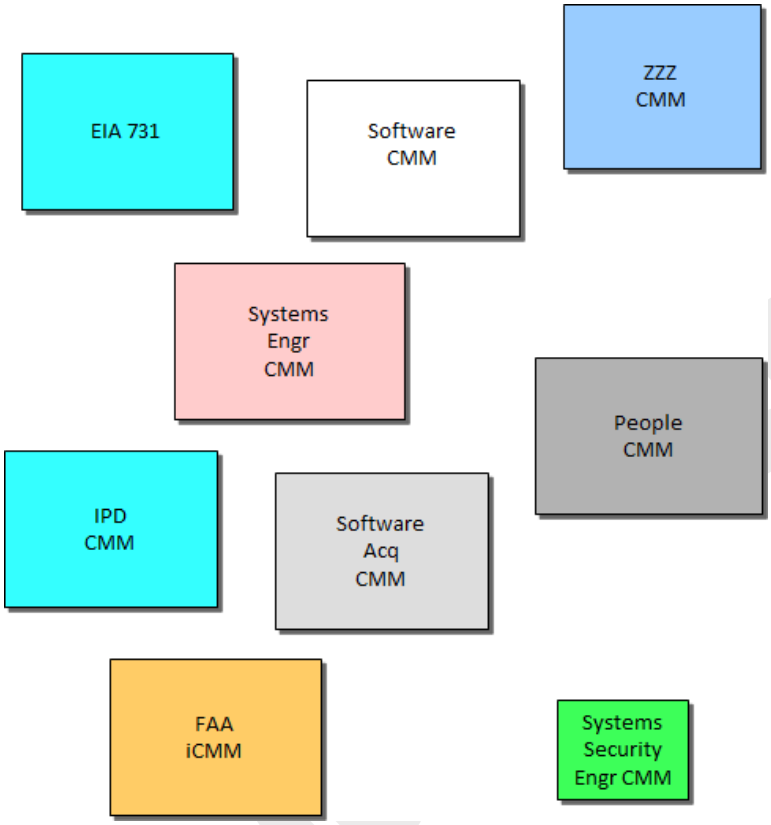
So Many Models, So Little Time

- Different structures, formats, terms, ways of measuring maturity
- Causes confusion, especially when using more than one model
- Hard to integrate them in a combined improvement program
- Hard to use multiple models in supplier selection

CMMI Comes to Rescue

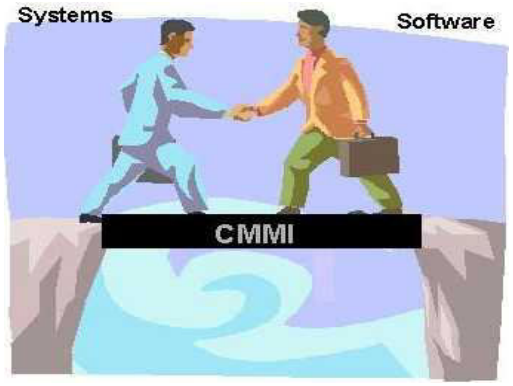
- Integrates systems and software disciplines into one process improvement framework

- Provides a framework for introducing new disciplines as needs arise



Bridging the Divide

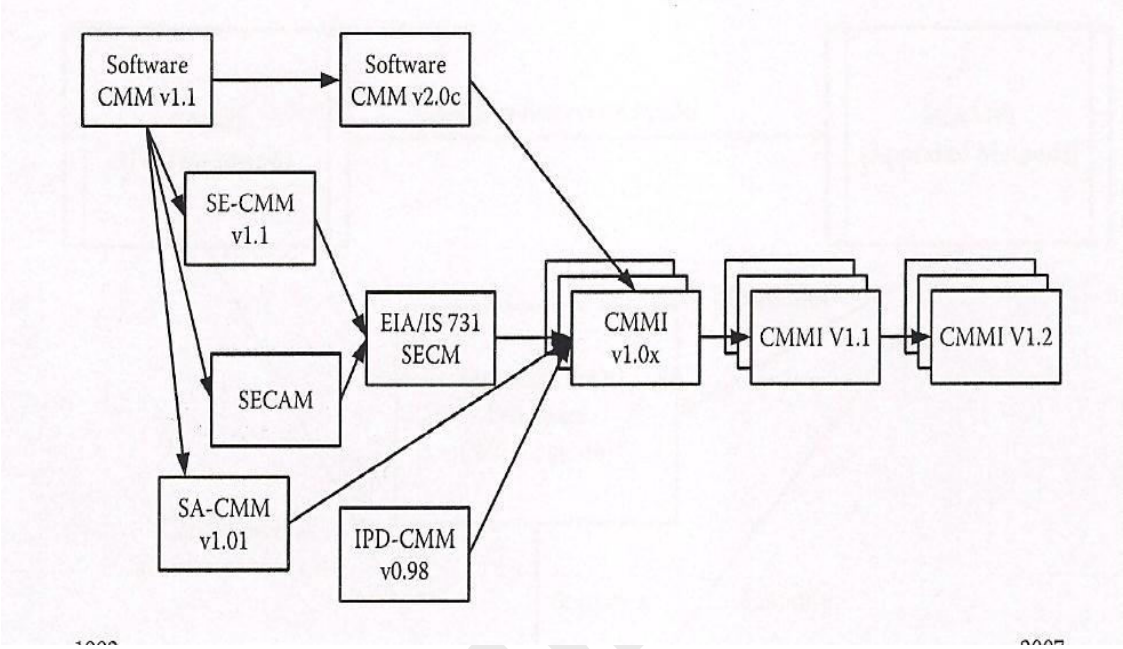
- Systems engineering and software engineering processes are integrated
- Integrates systems and software disciplines into one process improvement framework
- Provides a framework for introducing new disciplines as needs arise
- Some organizations see themselves as performing just one discipline
  - ✓ Software
  - ✓ Systems
  - ✓ Acquisition



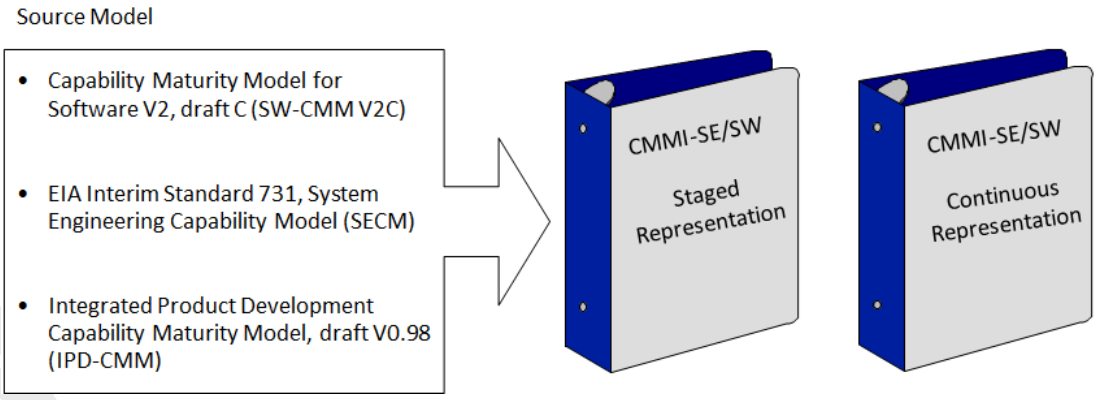
- But…
  - ✓ Software always must be part of some kind of system
  - ✓ Systems that don't have software are rare
  - ✓ Acquisition can involve both

- Communication and cooperation with other disciplines, even if they are external to our organization is vital

Models directly influencing CMMI

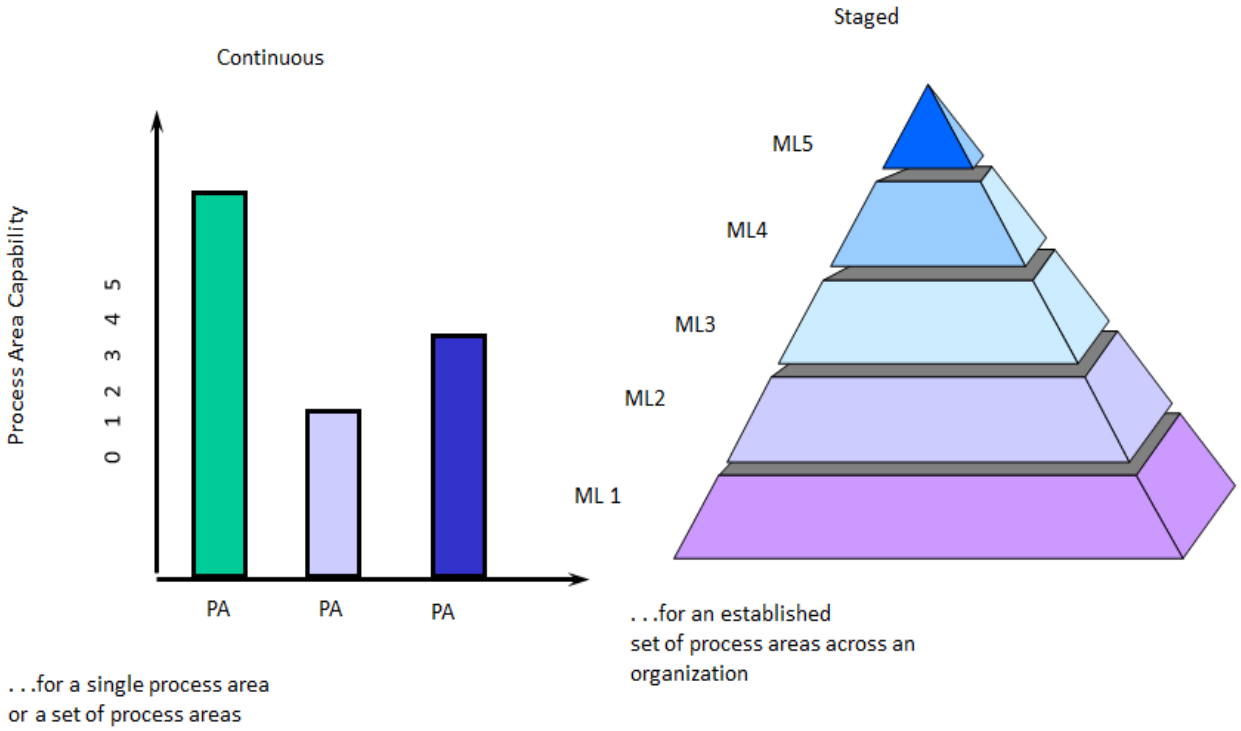


CMMI Models



- Combined System Engineering/Software Engineering model
- Can be applied to
  - ✓ Just the software engineering projects in an organization
  - ✓ Just the system engineering projects in an organization
  - ✓ Both

Comparing Model Representations



Staged Representation

- The staged representation is the approach used in the Software CMM. It is an approach that uses predefined sets of process areas to define an improvement path for an organization. This improvement path is described by a model component called a maturity level
- A maturity level is a well-defined evolutionary plateau toward achieving improved organizational processes

Continuous Representation

- The continuous representation is the approach used in the SECM and the IPD-CMM. This approach allows an organization to select a specific process area and improve relative to it
- The continuous representation uses capability levels to characterize improvement relative to an individual process area

Process Area Capability and Organizational Maturity

- Process area capability and organizational maturity are similar concepts
- The difference between them is that process area capability deals with a set of processes relating to a single process area or specific practice, while organizational maturity pertains to a set of process areas across an organization
- Remember
  - ✓ A model is not a process
  - ✓ The model shows what to do, NOT how to do it or who does it

Why Do we have Two Representations?

- Source Model Heritage
  - ✓ Software CMM--Staged
  - ✓ SECM--Continuous
  - ✓ IPD CMM—Hybrid
- Proponents for each type of representation were part of CMMI product development team
- Selecting a single representation approach became "too hard"
- A compromise was made to initially support two representations of the model with equivalent content

Advantages of Staged Representation

- Provides a roadmap for implementing
  - ✓ groups of process areas
  - ✓ sequencing of implementation
- Familiar structure for those transitioning from the SW-CMM

Advantages of Continuous Representations

- Provides maximum flexibility for focusing on specific process areas according to business goals and objectives
- Familiar structure for those transitioning from the systems engineering community

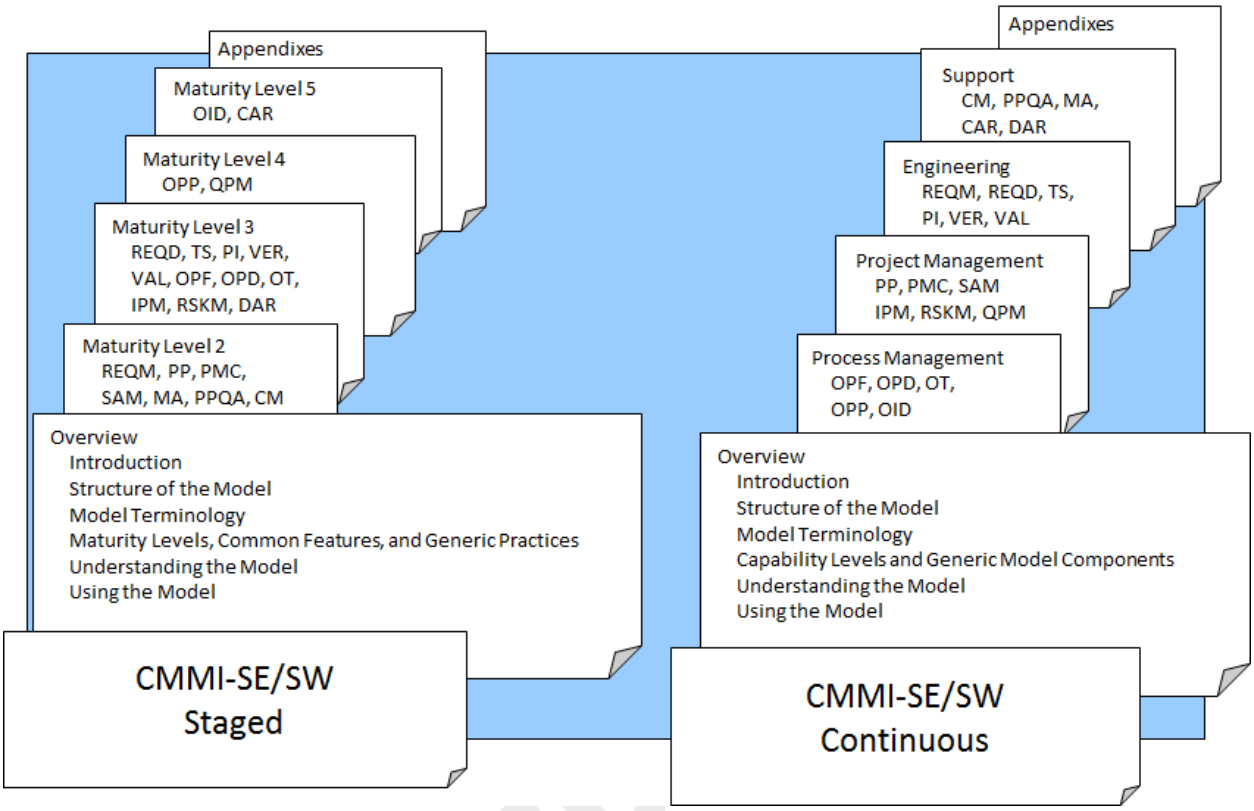
CMMI in a Nutshell

- A CMMI model provides a structured view of process improvement across an organization
- CMMI can help
  - ✓ set process improvement goals and priorities
  - ✓ provide guidance for quality processes
  - ✓ provide a yardstick for appraising current practices

CMMI Model Structure

- Maturity Levels (staged representation) or Capability Levels (continuous representation)
- Process Areas
- Goals – Generic and Specific
- Practices – Generic and Specific
- The CMMI product suite is at the forefront of process improvement because it provide the latest best practices for product and service development and maintenance
- The CMMI models improve the best practices of previous models in many important ways

CMMI Structure: One Model, Two Representations



Benefits of CMMI

- CMMI best practices enable organizations to do the following
  - ✓ More explicitly link management and engineering activities to their business objectives
  - ✓ Expand the scope of and visibility into the product lifecycle and engineering activities to ensure that the product or service meets customer expectations
  - ✓ Incorporate lessons learned from additional areas of best practices (e.g., measurement, risk management, and supplier management)
  - ✓ Implement more robust high-maturity practices
  - ✓ Address additional organizational functions critical to their products and services
  - ✓ More fully comply with relevant ISO standards
- Use CMMI in process improvement activities as a
  - ✓ Collection of best practices
  - ✓ Framework for organizing and prioritizing activities
  - ✓ Support for the coordination of multi-disciplined activities that might be required to successfully build a product
  - ✓ Means to emphasize the alignment of the process improvement objectives with organization business objectives

What is PSP?

- The PSP is a defined and measured software process designed to be used by an individual software engineer
- The PSP was developed by Watts Humphrey
- Its intended use is to guide the planning and development of software modules or small programs, but it is adaptable to other personal tasks
- Like the SEI Capability Maturity Model for Software, the PSP is based on process improvement principles
- While the CMM-SW was focused on improving organizational capability, the focus of the PSP is the individual engineer

Key Process Areas for PSP

- Software project planning
- Software project tracking/oversight
- Software process focus
- Software process definition
- Integrated software management
- Software product engineering
- Peer reviews
- Quantitative process management
- Quality management
- Defect prevention
- Technology change management
- Process change management

PSP in a Nutshell

- Baselining your existing personal software process
- Developing a planning process for your software development
- Measuring software size as part of the planning process
- Estimating software size in advance
- Estimating the required schedule and resources for software
- Conducting appropriate measures of one's process
- Conducting meaningful design and code reviews
- Performing software quality management
- Perform software design in a more formal fashion
- Verifying the design, using methods using methods such as object state machines, program tracing, etc.
- Scaling up the PSP to larger problems

Lecture 23
Introduction to TSP – I

Why Projects Fail

- Reasons
  - ✓ Seldom due to technical reasons
  - ✓ People problems
  - ✓ Excessive schedule pressure
- Solution
  - ✓ People problems can be solved with the help of good communication and good teams
  - ✓ Good teams guide in handling pressure
- The best team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels, and the overall problem difficulty
- The performance of a team is inversely proportional to the amount of communication that must be conducted
- The length of time that the team will "live together" affects team morale
- Constantine suggests four organizational paradigms for software engineering teams:

Closed Paradigm

- A closed paradigm structures a team along a traditional hierarchy of authority
- Such teams can work well when producing software that is quite similar to past efforts, but they will be less likely to be innovative when working within the closed paradigm

Random Paradigm

- The random paradigm structures a team loosely and depends on individual initiative of the team members
- When innovation or technological breakthrough is required, teams following the random paradigm will excel
- But such teams may struggle when "orderly performance" is required

Open Paradigm

- The open paradigm attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
- Work is performed collaboratively, with heavy communication and consensus-based decision making the trademarks of open paradigm teams
- Open paradigm team structures are well suited to the solution of complex problems but may not perform as efficiently as other teams

Synchronous Paradigm

- The synchronous paradigm relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves
- To achieve a high-performance team:
  - ✓ Team members must have trust in one another
  - ✓ The distribution of skills must be appropriate to the problem
  - ✓ Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained
- A jelled team is a group of people so strongly knit that the whole is greater than the sum of the parts…
- Once a team begins to jell, the probability of success goes way up. The team can become unstoppable, a juggernaut for success

Factors that Foster Toxic Team Environment

- A frenzied work atmosphere in which team members waste energy and lose focus on the objectives of the work to be performed
- High frustration caused by personal, business, or technological factors that causes friction among team members
- "Fragmented or poorly coordinated procedures" or a poorly defined or improperly chosen process model that becomes a roadblock to accomplishment
- Unclear definition of roles resulting in a lack of accountability and resultant finger-pointing
- "Continuous and repeated exposure to failure" that leads to a loss of confidence and a lowering of morale

Team Software Process (TSP)

- After this basic introduction about teams, let's talk about Team Software Process or TSP
- Development of Team Software Process (TSP) is another important step in software process improvement, after the development of CMM and PSP
- TSP was also developed at Software Engineering Institute
- The TSP provides a disciplined context for engineering work
- The principal motivator for the development of the TSP was the conviction that engineering teams can do extraordinary work, but only if they are properly formed, suitably trained, staffed with skilled members, and effectively led
- The objective of the TSP is to build and guide such teams
- Teams are required for most engineering projects. Although some small hardware or software products can be developed by individuals, the scale and complexity of modern systems is such, and the demand for short schedules so great, that it is no longer practical for one person to do most engineering jobs
- Systems development is a team activity, and the effectiveness of the team largely determines the quality of the engineering
- There are different kinds of teams

- In sports, for example, a basketball team's positions are dynamic while baseball/cricket team members have more static roles. However, in both cases the members must all work together cooperatively
- Conversely, wrestling and track teams are composed of individual competitors who do not dynamically interact, although the members support each other socially and emotionally
- In engineering, development teams often behave much like baseball or basketball teams. Even though they may have multiple specialties, all the members work toward a single objective
- However, on systems maintenance and enhancement teams, the engineers often work relatively independently, much like wrestling and track teams
- A team is more than just a group of people who happen to work together
- Teamwork takes practice and it involves special skills
- Teams require common processes; they need agreed upon goals; and they need effective guidance and leadership
- The methods for guiding and leading such teams are well known, but they are not obvious
- The Software Engineering Institute (SEI) is supporting the TSP as a way to guide engineers and their managers in using effective teamwork methods

Conditions for Teamwork

What is a Team?

- A team is a group of people who share a common goal
- They must all be committed to this goal and have a common working framework
- A team consists of at least two people
- The members are working toward a common goal
- Each person has a specific assigned role
- Completion of the mission requires some form of dependency among the group members

Team

- The four parts of this definition of a team are all important
- For example, it is obvious that a team must have more than one member, and the need for common goals is also generally accepted
- However, it is not as obvious why team members must have roles?
- Roles provide a sense of ownership and belonging
- They help to guide team members on how to do their jobs; they prevent conflicts, duplicate work, and wasted effort; and they provide the members with a degree of control over their working environment
- Such a sense of control is a fundamental requirement for motivated and energetic team members

Interdependence

- Interdependence is also an important element of teamwork. It means that each team member depends to some degree on the performance of the other members

- Interdependence improves individual performance because the members can help and support each other
- E.g., Design teams

Social Support of Membership

- Team performance is further enhanced by the social support of membership. Human beings are social animals and few people like to work entirely by themselves—at least not for very long
- Because of the social context of teams, the members will generally make a special effort to meet their obligations to the rest of the team
- Through mutual support and interdependence, teams become more than just the sum of their individual members
- To be effective, teams must be properly skilled and be able to work as cohesive units
- Effective teams have certain common characteristics

Characteristics of Effective Teams

- The members are skilled
- The team's goal is important, defined, visible, and realistic
- The team's resources are adequate for the job
- The members are motivated and committed to meeting the team's goal
- The members cooperate and support each other
- The members are disciplined in their work
- The members are innovative

Innovation

- Innovation is more than just thinking up bright ideas; it requires creativity and a lot of hard work. Just about every engineering task is part of an innovative endeavor
- Innovative teams must have skilled and capable people who are highly motivated. They must be creative, flexible, and disciplined
- They must strive to meet demanding schedules while adjusting to changing needs
- They must also control costs and schedules while keeping management informed of their progress
- To be innovative and effective, engineering teams must work in a trusting and supportive environment
- Engineering teams are composed of extremely capable people who can quickly sense a lack of trust
- When managers do not trust their teams to make aggressive schedules or to strive to meet these schedules, the engineers will know it
- When engineers do not feel trusted and respected, they often feel antagonized and manipulated. These engineers no longer feel loyal to the organization and can easily lose their commitment to the team
- Since people generally work harder when they face an important and meaningful challenge, it is appropriate for management to challenge their teams with aggressive goals

- But when the teams respond to the challenge with a plan, management must be willing to negotiate realistic commitments that the engineers believe they can meet
- Few people will work diligently to meet a seemingly hopeless schedule
- To perform effectively, teams must believe that their project is important and that the schedule is achievable
- The TSP is designed to establish the conditions that characterize effective teams

References

- SEI website: www.sei.cmu.edu
- Introduction to the Team Software Process by Watts Humphrey (Chapters Ch 1-2, 3.1-3.4, 4.1-4.5,5.1, 10.1-10.3)
- The Team Software Process by Watts Humphrey, SEI Technical Report 0023, November 2000

Lecture 24
Introduction to TSP – II

Teambuilding

- The teambuilding principles used in the TSP to establish conditions for effective teams

TSP Teambuilding Principles

- The team members establish common goals and defined roles
- The team develops an agreed-upon strategy
- The team members define a common process for their work
- All team members participate in producing the plan, and each member knows his or her personal role in that plan
- The team negotiates the plan with management
- Management reviews and accepts the negotiated plan
- The team members do the job in the way that they have planned to do it
- The team members communicate freely and often
- The team forms a cohesive group: the members cooperate, and they are all committed to meeting the goal
- The engineers know their status, get feedback on their work, and have leadership that sustains their motivation
- Effective team formation requires that the members truly understand what they are supposed to do, agree on how to do the job, and believe that their plan is achievable
- These conditions can all be established by involving the engineers in producing their own plans. Then, assuming that these plans are competently made, teams can almost always sell their plans to management
- While all these conditions are necessary for effective teamwork, the specific ways for establishing these conditions are not obvious
- The TSP provides the explicit guidance that organizations need to build effective engineering teams

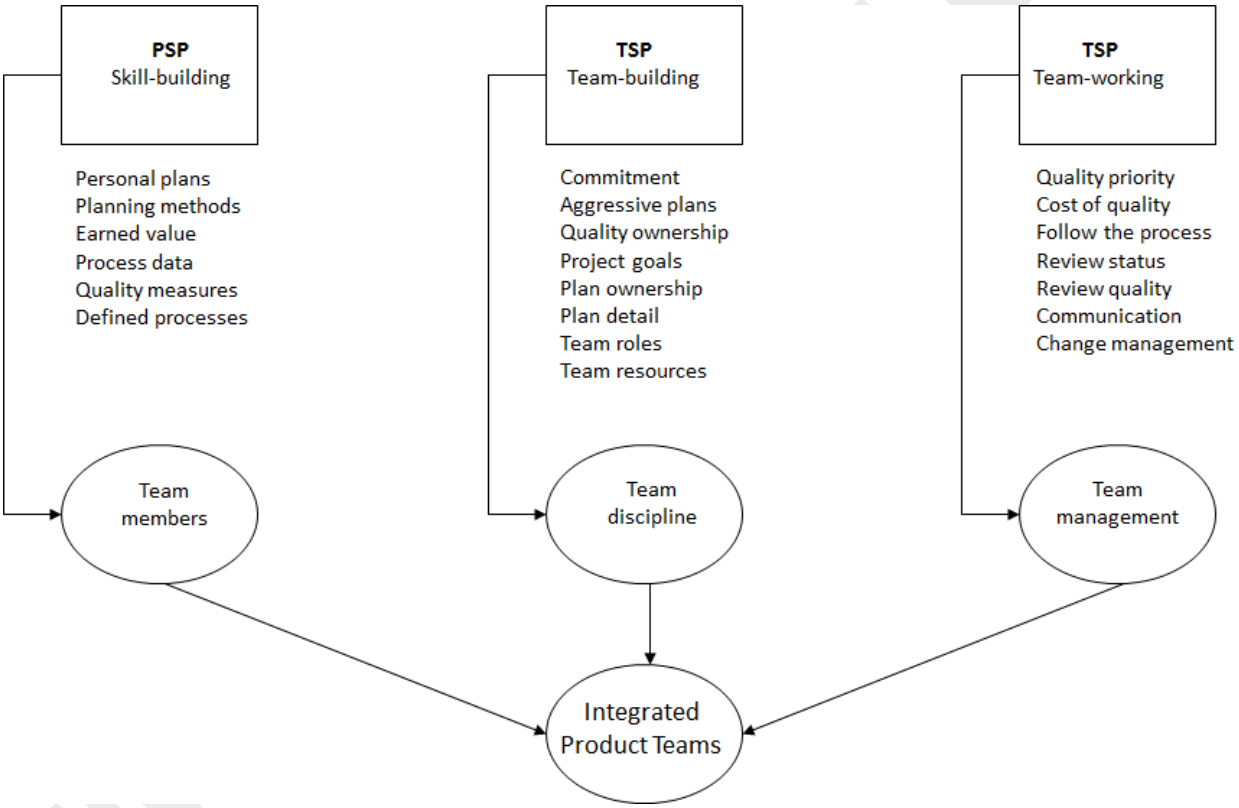
Team Operational Processes

- To do disciplined work, engineers need what Deming calls “operational processes”
- These are processes that define precisely how the work is to be done
- An operational process is more like a script. It is designed to be used by the team members when they do the work
- The TSP provides a defined operational process to guide engineers and managers through the team-building steps
- This process specifies the steps needed to establish an effective team-working environment
- Without specific guidance, engineers must work out the details of team-building and team-working for themselves
- Since defining these details involves considerable skill and effort, and since few engineers have the experience or time to work out all of the necessary details, engineering teams generally

follow ad-hoc team-building and teamwork processes. This wastes time and it often produces poorly functioning teams

- With a defined process and a plan that follows that process, engineers can be highly efficient
- If they don't have such a process, they must stop at each step to figure out what to do next and how to do it
- Most engineering processes are quite complex and involve many steps. Without specific guidance, engineers are likely to skip steps, to do steps in an unproductive order, or to waste time figuring out what to do next
- The TSP provides the operational processes needed to form engineering teams, to establish an effective team environment, and to guide teams in doing the work

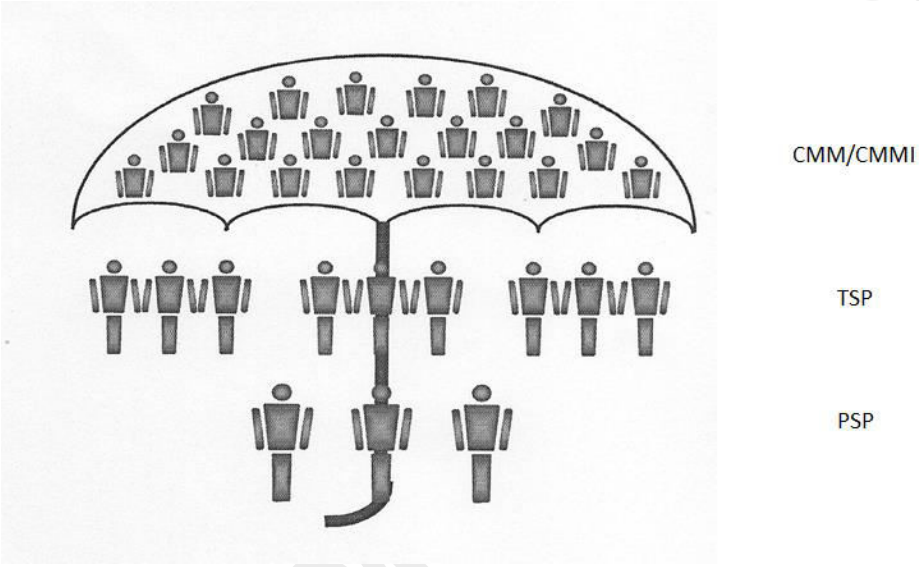
Principle Elements of TSP



- Before team members can participate on a TSP team, they must know how to do disciplined work
- Training in the Personal Software Process (PSP) is required to provide engineers with the knowledge and skills to use the TSP
- PSP training includes learning how to make detailed plans, gathering and using process data, developing earned value plans, using earned value to track a project, measuring and managing product quality, and defining and using operational processes

- Engineers must be trained in these skills before they can participate in TSP team building or follow the defined TSP process
- The Capability Maturity Model (CMM) or Capability Maturity Model Integration (CMMI) provides the overall improvement framework needed for effective engineering work
- The Personal Software Process (PSP) provides the engineering disciplines that engineers need for consistently using a defined, planned, and measured process

Tying Process Improvement Efforts



- The TSP couples the principles of integrated product teams with the PSP and CMM/CMMI methods to produce effective teams
- In essence, the CMM/CMMI and PSP provide the context and skills for effective engineering while the TSP guides engineers in actually doing the work
- Thus, the TSP capitalizes on the preparation provided by the PSP and CMM/CMMI, while also providing explicit guidance on how to do the work

Back to TSP

- While there are many ways to build teams, they all require that the individuals work together to accomplish some demanding task
- In the TSP, this demanding team-building task is a four- day planning process that is called the team launch
- In a launch, all the team members develop the strategy, process, and plan for doing their project. After completing the launch, the team follows its own defined process to do the job

TSP Process

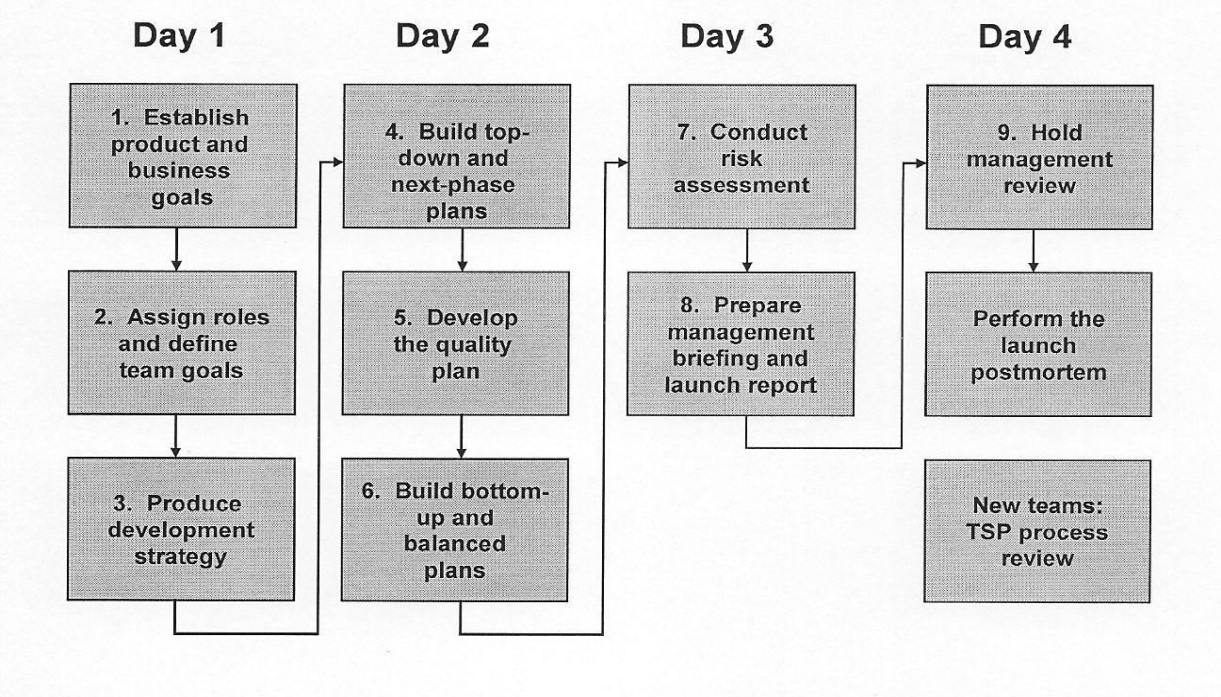
- Launching a team project
- Development strategy

- Development plan
- Design process
- Implementation process
- Test plan
- Postmortem

Launching a Team Project

- The process of forming team and building team does not happen by accident, and it takes time
- Teams need to establish their working relationships, determine member roles, and agree on goals
- The launch is the first step
- An hour or so spent on team-building issues at the beginning of the project saves time later
- Defining the roles and agreeing on who will handle each role is an essential first step in team formation
- In the TSP launch, teams make an overall plan and a detailed plan for about the next three to four months
- After the team members have completed all or most of the next project phase or cycle, they revise the overall plan if needed and make a new detailed plan to cover the next three to four months

The TSP Launch Process



- Teams generally need professional guidance to properly complete the launch process. This guidance is provided by a trained launch coach who leads the team through the launch process
- While the TSP scripts provide essential guidance, every team has unique problems and issues, so a simple process cannot possibly provide all the material needed to guide an inexperienced team through the launch process
- Unless teams are very experienced and have a team leader who has completed several TSP projects, they generally need the support of a trained launch coach
- In launch meeting 1, the team, team leader, and launch coach meet with senior management and marketing representatives
- The objective of meeting 1 is to inform all the team members about the job, to describe management's goals for the team, and to convince the team members that management is relying on them to do this important project
- In launch meetings 2 through 8, the team, team leader, and coach meet with no observers or visitors
- During these meetings, the team is led through a series of steps that are designed to produce the conditions for effective teamwork
- In launch meeting 2, the team documents its goals and selects the team member roles. The standard TSP team roles are team leader, customer interface manager, design manager, implementation manager, test manager, planning manager, process manager, quality manager, and support manager. Other possible roles can be assigned as needed
- In launch meetings 3 and 4, the team makes the overall project strategy and plan. The engineers produce a conceptual design, devise the development strategy, define the detailed process they will use, and determine the support tools and facilities they will need
- A list of products to be produced, along with estimate of size and time required to process step is generated
- A schedule is generated from these data
- Once they have an overall plan, the engineers produce the quality goals and plan in launch meeting 5
- This plan defines the quality actions the team plans to take and it provides a measurable basis for tracking the quality of the work as it is done
- In making quality plans, team members estimate the number of defects they will inject and remove in each phase, and how many defects will be left for system test, customer acceptance testing, and final product delivery
- In meeting 6, the team members make detailed next-phase plans and then review the entire team workload to ensure that these plans evenly distribute the tasks among the members. The result is then what is called a balanced team plan
- During meeting 7, the engineers identify the major project risks and rank them for likelihood and impact. The team also assigns a team member to track each risk and it prepares a mitigation plan for the most significant risks
- After the team has completed its plan, the members hold meeting 8 to prepare for the management review and then they conduct the review with management in meeting 9

- During this meeting, the team explains the plan, describes how it was produced, and demonstrates that all the members agree with and are committed to the plan
- If the team has not met management's objectives, it should generally prepare and present alternate plans that show what could be done with added resources or requirements changes
- The principal reason for showing alternate plans is to provide management with options to consider in case the team's plan does not meet business needs
- At the end of the TSP launch, the team and management should agree on how the team is to proceed with the project
- In the final postmortem step, the team reviews the launch process and submits process improvement proposals (PIPs) on suggested process improvements. The team also gathers and files the launch data and materials for later use

References

- SEI website: www.sei.cmu.edu
- Introduction to the Team Software Process by Watts Humphrey (Chapters Ch 1-2, 3.1-3.4, 4.1-4.5,5.1, 10.1-10.3)
- The Team Software Process by Watts Humphrey, SEI Technical Report 0023, November 2000

Lecture 25
Introduction to TSP – III

The TSP Team working Process

- Once the TSP team is launched, the principal need is to ensure that all team members follow the plan. This includes the following major topics
- Leading the team
- Process discipline
- Tracking issues
- Communication
- Management reporting
- Maintaining the plan
- Estimating project completion
- Rebalancing team workload
- Relaunching the project
- TSP quality management

Leading the Team

- The team leader is responsible for guiding and motivating the team members, handling customer issues, and dealing with management
- This includes the day-to-day direction of the work, protecting team resources, resolving team issues, conducting team meetings, and reporting on the work
- The team leader's principal responsibility is to maintain the team's motivation and energy and to ensure that it is fully effective in doing its work
- Productive teamwork requires a combination of specific goals, a supportive working environment, and capable coaching and leadership. The project goal is to build a working product
- TSP provides the supportive environment, one of your team members will be the team leader, and the instructor provides the coaching
- When you have struggled with actual project problems and have been guided to effective solutions, you will appreciate the benefits of sound development practices. Without the precise guidance of the TSP, however, you could waste considerable time in defining your own practices, methods, and roles
- Instruction is most effective when it builds on the available body of prior knowledge. There has been a great deal of experience with software teams and software team courses. TSP builds on this foundation

Process Discipline

- One key leadership responsibility is maintaining process discipline. Here, the team leader ensures that the engineers do the job the way they had planned to do it
- Almost every project faces heavy schedule and resource pressure, so there is always a temptation to cut corners
- However, when teams stop following their defined processes, they have no way to tell what they are supposed to do or where they stand on the job

Tracking Issues

- Another important team leader responsibility is ensuring that all of the issues that the team members identify are managed and tracked
- Task hours
- Earned value status
- Milestone status
- Quality status

Communication and Feedback

- Learning is most effective when you follow a defined process and get rapid feedback. The TSP scripts and forms provide a defined, measured, and repeatable framework for team software engineering
- TSP provides rapid performance feedback because the team produces the product in several short development cycles and evaluates results after each cycle
- The team leader is responsible for maintaining open and effective team communication. When team members do not know the project's status, understand what their team mates are doing, or know what challenges lie ahead, it is hard for them to stay motivated
- Communication is a key part of maintaining the team's energy and drive and facilitating communication is a key part of the team leader's responsibilities

Rebalancing Team Workload

- Unbalanced workload can cause a team to be inefficient. This occurs when some engineers have much more work than others have

Re-launching the Project

- In TSP, teams periodically replan their work. This is necessary because it is difficult to make detailed plans that run for more than few months
- The relaunch is generally scheduled as part of the project and conducted whenever the team feels it would be helpful

TSP Quality Management

- To manage quality, teams must establish quality measures, set quality goals, establish plans to meet these goals, measure progress against the plans, and take remedial action when the goals are not met
- The elements of TSP quality management are making a quality plan, identifying quality problems, and finding and preventing quality problems

TSP Goals and Goal-Setting

Team Goals

- Goal setting is an essential step in team formation and should usually be done at the start of each project

- Establishing goals is simple in concept but often difficult in practice
- Goals should be precisely measured, and few of us are in the habit of being precise about our work

Goal-Setting Considerations

- Teams should not be measured on whether or not they actually meet their goals. Instead, they should be evaluated on their willingness to set measurable and aggressive goals and on their efforts to meet them
- Too much emphasis on meeting goals is counterproductive because it motivates teams to set goals that they are sure they can meet

Goal-Setting Steps

- Write down the goals you wish to use instead
- Specify how to measure these goals
- Describe why you have selected these goals instead of the ones provided by the TSP
- Give a copy of your revised goals to the team and to the instructor
- Have a team leader put a copy of your goals in the project notebook

Goal Setting for TSP

- Produce a quality product
- Run a productive and well-managed project
- Finish on time

TSP Process

- Launching a team project (already discussed)
- Development strategy
- Development plan
- Design process
- Implementation process
- Test plan
- Postmortem

The Development Strategy

- One of the principal problems in software development is to figure out how to build big systems. Some years ago it was common to build big modules and components, but no on advocates this approach any more
- Steps in Developing Strategy
    - ✓ Define the strategy criteria
    - ✓ Determine the possible alternative strategies
    - ✓ Identify the risks and benefits of each alternative strategy
    - ✓ Make a comparative evaluation of these alternatives
    - ✓ Make the strategic decision
    - ✓ Document the selected strategy

Ask Four Questions

- Based on what I know, how would I build this product?
- What are the principal components I will need to build this product?
- What functions must these components provide?
- How big do I think these components will be?

Some Risks

- You may encounter one or more functions that you don't know how to design
- You may run into support system problems that delay your work
- The product may be so defective that testing takes too long
- You might lose control of the product or product changes and waste time reconstructing programs you have already developed
- Your team may not be able to work effectively

Managing Risks

- Too large a product
  - ✓ Start with a small kernel product and then add functions in later development cycles
- Difficult or complex functions
  - ✓ Prototype these functions at the beginning of the project, when you have time to consider alternatives
- Support system problems
  - ✓ Build an early prototype or small product version to make sure that you know how the support system works
- Testing time
  - ✓ If you consistently follow the TSP and use disciplined PSP methods, your product will have few defects and testing will not be a problem
- Product control
  - ✓ It is easy for teams to lose control of their products. This is why TSP addresses configuration management at the beginning of the project
- Teamwork problems
  - ✓ If your team is having trouble working together, discuss the problems openly to see whether you can resolve them. If the problems persist, get help from the instructor

The Development Plan

- Why make plans?
- Balanced plans
- Tracking progress against the plan
- Planning in detail
- Handling unplanned tasks
- Estimating level
- Implementation planning

The TSP Design Decisions

- Provide a simple framework that builds on the foundation of the Personal Software Process (PSP)
- Develop products in several cycles
- Establish standard measures for quality and performance
- Provide precise measures for teams and students
- Use role and team evaluations
- Require process discipline
- Provide guidance on teamwork

The Postmortem

- Why we need a postmortem
- What a postmortem can do for you
- The process improvement proposal

References

- SEI website: www.sei.cmu.edu
- Introduction to the Team Software Process by Watts Humphrey (Chapters Ch 1-2, 3.1-3.4, 4.1-4.5,5.1, 10.1-10.3)
- The Team Software Process by Watts Humphrey, SEI Technical Report 0023, November 2000

Lecture 26
TSP Body of Knowledge

TSP Body of Knowledge

- SEI has drafted the Team Software Process Body of Knowledge (TSP BOK) document, which defines the fundamental knowledge and skills that set TSP-trained individuals apart from other software professionals
- It helps individual practitioners to assess and improve their own skills, provides employers with an objective baseline for assessing the process improvement skills and capabilities of their development team members, and guides academic institutions that want to incorporate TSP into their software engineering curriculum
- The TSP body of knowledge is composed of six competency areas, each with several knowledge areas
- We'll talk about these in a little while, but let's talk about some basic terminologies associated with the body of knowledge

Competency Area

- A competence area is a group of closely-related knowledge areas that a practitioner is well qualified to perform intellectually or physically

Knowledge Area

- A knowledge area is the sum or range of specific understanding and ability gained through study of a set of concepts or through experience with a set of skills

Concept

- An explanatory principle applicable to a specific instance or occurrence within a particular knowledge area

Skill

- Proficiency, facility, or dexterity of performance that is acquired or developed through training or experience in a particular knowledge area

TSP BOK Competence Areas

- TSP Foundations and Fundamentals
- Team Foundations
- Project Planning with TSP
- Project Implementation and Tracking with TSP
- Gathering and Using TSP Data
- Scaling Up the TSP

TSP Foundations and Fundamentals

- This competency area outlines the foundational knowledge on which TSP is built and describes the fundamental concepts that a TSP practitioner must understand in order to successfully implement and practice the TSP methodology
- The knowledge areas composing the TSP Foundations and Fundamentals competency area are
  - ✓ Knowledge Work
    - PSP and TSP are practices designed to facilitate and improve both the process and the outputs of knowledge work, which is the interpretation, development, and implementation of information by skilled professionals within a specific subject area
    - This knowledge area discusses the nature of knowledge work and the team and workplace characteristics required for such work
  - ✓ TSP Prerequisite Knowledge
    - This knowledge area outlines the fundamental concepts and skills that individuals must master before implementing the TSP methodology as a member of a TSP team
    - Although this area calls out some of the specific Knowledge Areas of the PSP BOK, the PSP BOK in its entirety is considered to be prerequisite knowledge for implementing the TSP in practice
  - ✓ TSP Principles
    - This knowledge area outlines the basic principles underlying the Team Software Process
    - The key concepts identify the elements that are common to and required for successful outcomes of work done by teams to produce software products and/or software-intensive systems
  - ✓ TSP Process Elements and Measures
    - This knowledge area describes the process elements and measures that are used in the TSP. (Where applicable, overlaps with or differences from PSP process elements and measures are noted.)
  - ✓ TSP Quality Practices
    - This knowledge area describes the specific quality practices added in the TSP to build on the individual quality practices used by PSP practitioners

Team Foundations

- A team consists of a group of people who act in cooperation to achieve a common purpose
- When teams are effective in achieving their goals, it is because they are composed of members with complementary skills that work together to create a synergistic effort; effective teams achieve a kind of gestalt, in which the members' strengths are maximized and the weaknesses minimized so that the team as a whole becomes greater than the sum of its parts
- Teams are especially appropriate for doing work of a highly complex nature (such as knowledge work) and for accomplishing large-scale tasks with many interdependent subtasks
- The knowledge areas composing the Team Foundations competency area are:

- ✓ Teams and Teambuilding
  - This knowledge area describes the characteristics of teams and explains concepts for building high-performing project teams
- ✓ Team Types, Styles, and Dynamics
  - This knowledge area describes some of the models of team types, team styles, and team dynamics that provide a foundational understanding of team work in the TSP
- ✓ Team Formation and Membership
  - This knowledge area describes important parameters that should be considered when forming and populating TSP teams
- ✓ Team Member Responsibilities
  - This knowledge area describes the team's specific responsibilities for helping teams to be fully effective
- ✓ Team Member Roles
  - This knowledge area discusses the types of roles on TSP teams and delineates the responsibilities required of the various team member roles
- ✓ Team Leader Role
  - This knowledge area lists the roles that a TSP team leader must fulfill and discusses some of the tasks and responsibilities that accompany the various roles
- ✓ Coach Role
  - This knowledge area describes the general roles and responsibilities of the TSP coach

Project planning with TSP

- Project planning with TSP begins when an organization makes the decision to implement the new technology using one or more small pilot projects
- The organization works with a certified TSP coach to choose pilot teams and to provide them with training in PSP or process improvement techniques and in TSP project planning and tracking methodologies
- Once it is trained, the team plans the project work by participating in a TSP launch
- The TSP launch process is actually a series of teambuilding activities that are organized into ten meetings
- The TSP coach works with the team's members, leader, and managers to form an understanding of the project requirements, determine team goals, select team member roles, and make plans for producing the product
- The team presents its plan to management and after receiving management acceptance for the plan, or an alternative plan, begins to work on the project
- The knowledge areas in this competency area describe the requirements, activities, and guideposts for implementing pilot TSP projects and launching TSP teams

- The knowledge area are:
  - ✓ Change Management Fundamentals
    - This knowledge area describes some of the fundamental concepts of change management and technology introduction that can help to facilitate the introduction of new technologies, processes, and practices into an organization
  - ✓ Piloting TSP in an Organization
    - This knowledge area describes the general requirements and guidelines for initiating and conducting TSP pilot projects
  - ✓ Preparing Management and Teams for TSP Implementation
    - This knowledge area describes the training and pre-launch preparations required to prepare executives, managers, team leaders, and team members for effective participation in TSP implementations
  - ✓ The TSP Launch Meetings
    - This knowledge area provides an overview of the TSP launch and a description of each of the meetings that make up a TSP launch
  - ✓ The TSP Relaunch
    - The team relaunch is nearly the same as the launch except that it is done by a team that has already completed an initial launch of the same project
    - This knowledge area describes how the relaunch differs from the launch, explains when and how to conduct a relaunch, and delineates the inputs and outputs for the relaunch

Project Implementation and Tracking with TSP

- Following a TSP team launch or relaunch, the team members must work to implement their plan by completing the tasks which they have been assigned
- The TSP does not mandate the use of any particular approach to project implementation; rather, it stipulates that team members should all use PSP (for engineering work) or best professional practices (for other types of work done in support of the project) to enable team members to produce high-quality product components or project elements
- TSP principles also require the team members to observe process discipline by following the processes and strategies that the team agreed to use; by requiring that the team members all record their data accurately, honestly, and in real time; and that the team members individually and collectively maintain a constant focus on quality
- Faithful and precise data collection enables teams to monitor their progress against the plan
- Upto-date and accurate data can be analyzed to enable team members to identify potential problems with project schedule or product quality earlier in the process, rather than later
- This allows teams to address these issues, fix the problems, or seek help before small problems become big problems that can derail the project
- Using data for status tracking also provides the information necessary to generate reports for its stakeholders on the team's progress, and to advise management of potential problems so that they can provide assistance or additional resources to put the team back on track
- This competency area is composed of the following knowledge areas

- ✓ Weekly Meetings
  - The weekly team meeting is one of the elements that differentiates TSP from typical software projects. It keeps the team members focused on the project work and allows regular opportunities for teamworking and team building
- ✓ Checkpoints
  - Checkpoints are a series of meetings between the team leader, team members, and coach that allow everyone to discuss issues or problems such as process implementation, data gathering, or data analysis, and to generate solutions for any identified problems
- ✓ Communicating with Stakeholders
  - Management is the primary stakeholder of the TSP team, although there may also be internal customers, external customers, or both. In addition, the team members themselves are stakeholders in the process and its outcome
  - Whoever the stakeholders may be, it is important that the TSP team communicates frequently with them and reports useful and timely information about status, schedule, and product quality
- ✓ Replanning
  - Plans are subject to frequent change, particularly on new or inexperienced TSP teams. When plans change frequently it is easy for the team members to lose track of the overall team plan or their personal roles in meeting the plan
- ✓ Phase, Cycle, and Project Postmortems
  - To ensure that the team stays on track and maintains its motivation, it may be desirable to hold periodic short replanning meetings, in which the team leader reviews the current management priorities, and the team members summarize their status and adjust their individual and team plans as needed to address those priorities

- The postmortem analyzes the collected data on schedule, task hours, estimation, quality, and process
- The results are used to provide personal, team and organizational planning data for use in an upcoming launch or relaunch and to identify improvements to process elements
- The focus is on using data for planning and process improvement

Gathering and Using TSP Data

- In the same way that PSP relies on individual data, the TSP methods rely on data collection and analysis to enable teams to understand what they do and how they do it, thereby enabling teams to identify effective procedures that should continue to be used and to pinpoint those areas in which improvements are needed
- Data provide a baseline for making estimates and plans, tracking progress against the plan, and measuring the effects of changes implemented as part of a process improvement effort
- Therefore, it is critical to the successful implementation of TSP that teams collect data as they work, for use in later analyses and estimations

- The knowledge areas composing this competency area are as follows
  - ✓ Data Recording
    - Data are used to provide teams with a baseline for making estimates and plans, tracking progress against the plan, and measuring the effects of changes implemented as part of a process improvement effort
    - Therefore, it is critical to the successful implementation of TSP that teams collect data accurately and in a timely manner as the project work progresses
  - ✓ Gathering and Using Size Data
    - Size and time are often correlated, and when they are, size estimates can be used to estimate effort and then create plans based on the size and effort estimates
    - Size data are also useful for tracking development effort and assessing product quality (when defect data are normalized based on size)
  - ✓ Gathering and Using Schedule Data
    - Schedule data are used to predict the project's likely completion date and to track actual progress against the planned schedule
  - ✓ Gathering and Using Quality Data
    - The primary focus of the TSP is producing a high quality product, and it is required to learn the principles, measures, tools, and techniques for using data to manage quality
  - ✓ Gathering and Analyzing Postmortem Data
    - The postmortem provides the team with an opportunity to learn from its work to improve their product quality, design practices, planning and tracking, and teamwork
- By gathering, compiling, and analyzing the available data at the end of a phase, cycle or project when it can most economically, rapidly, and accurately be obtained, the team members can learn from their strengths and weaknesses and capitalize on this learning when they start another phase, cycle or project
- This knowledge area discusses the types of data that are gathered and analyzed, identifies the various team member responsibilities for gathering and preparing data for the postmortem, explains how data are used in the postmortem meeting to identify strengths and areas for improvement, and lists the data that should be captured in the postmortem report

Scaling Up the TSP

- Successful introduction of TSP into an organization begins with one or two small-scale pilot projects
- Pilot projects provide the management support and compelling evidence needed to convince the organization's general population that the TSP methods are effective and will be beneficial to the organization
- If the pilot projects are successful, management often decides to implement TSP in one or more divisions or organizations

- As with the introduction of the pilot projects, special considerations must be addressed to increase the likelihood that organization-wide implementation will be successful
- Whether or not organizations choose to implement TSP throughout the company, they all have unique needs that may require some tailoring of the TSP applications
- This is particularly true if the TSP project team requires more 15 to 20 members, if the members of the team have different professional capabilities or specialties that must work together to produce the product, or if some of the team members work at locations apart from most of the team
- The knowledge areas in this competency area describe the activities of scaling up the TSP for entire organizations or very large TSP project teams, and the adaptations to the basic TSP process that may be needed to address the needs of specialized TSP teams
- Knowledge areas in this competence area are
    - ✓ Organizational Implementation
        - This knowledge area describes the process of scaling the TSP implementation up from use on a few pilot projects to full introduction of TSP across the organization
    - ✓ TSP Process Variations
        - In development work, teams typically are classified as either project teams or functional teams
        - A project team is one that is formed to accomplish a specific project objective and, when that objective has been completed, the team is either disbanded or given another project assignment
        - A functional team is one that has a continuing mission responsibility
        - Additional variations in team type are due to team size or physical location
        - TSP can be adapted to fit the needs of functional teams (TSPf), integrated project teams (TSPI), distributed teams (TSPd), multiple TSP teams working in tandem (TSPm), TSP teams that also use CMMI (TSP+), and academic (student) teams (TSPi)
    - ✓ Large-scale TSP Teams
        - This knowledge area describes the characteristics and considerations unique to large-scale TSP teams

References

- TSP BOK, SEI, July 2010

Lecture 27

Software Process Improvement Using PDCA

Software Process Improvement

- There are a number of models for instituting software process improvement programs in organizations
- All these models have to be incorporated in orderly and cyclic manner
- These introductions cannot and must not be made abruptly

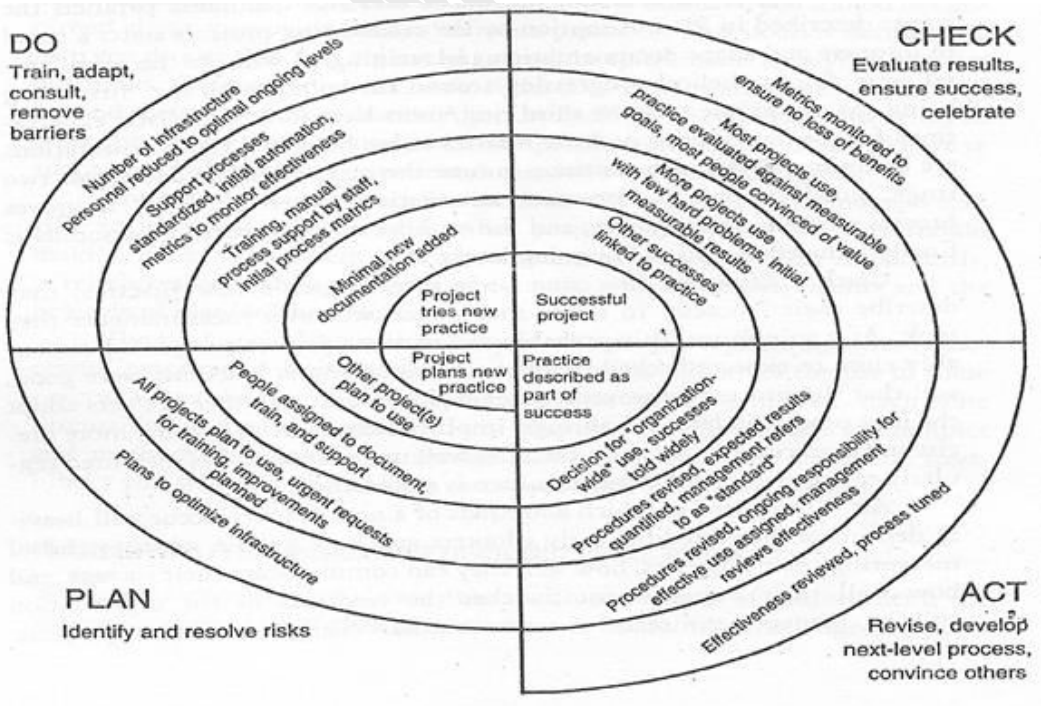
Shewart cycle

- The Shewart cycle is one such way of introducing software process improvement program in organization. It was was popularized by W. Edwards Deming
- The Shewart cycle has four steps
  - ✓ Plan
  - ✓ Do
  - ✓ Check
  - ✓ Act

PDCA

- Plan: Identify and resolve risks
- Do: Train, adapt, consult, remove barriers
- Check: Evaluate results, ensure success, celebrate
- Act: Revise, develop next-level process, convince others
- The PDCA model can be combined with the famous spiral process model, specially adapted for process improvements
- This results in a new spiral process improvement model for software

Spiral Model for Process-Improvement Adoption



- Let's discuss this picture in some detail cycle by cycle

Cycle I: SPI Spiral Model

- Project plans new practice (plan)
- Project tries new practice (do)
- Successful project (check)
- Practice described as part of success (act)

Cycle II: SPI Spiral Model

- Other project(s) plan to use (plan)
- Minimal new documentation added (do)
- Other successes linked to practice (check)
- Decision for "organization-wide" use, successes told widely (act)

Cycle III: SPI Spiral Model

- People assigned to document, train, and support (plan)
- Training manual, process support by staff, initial process metrics (do)
- More projects use with few hard problems, initial measurable results (check)
- Procedures revised, expected results quantified, management refers to as "standard" (do)

Cycle IV: SPI Spiral Model

- All projects plan to use, urgent requests for training, improvements planned (plan)
- Support processes standardized, initial automation, metrics to monitor effectiveness (do)
- Most projects use, practice evaluated against measurable goals, most people convinced to value (check)
- Procedures revised, ongoing responsibility for effective use assigned, management reviews effectiveness (act)

Cycle V: SPI Spiral Model

- Plans to optimize infrastructure (plan)
- Number of infrastructure personnel reduced to optimal ongoing levels (do)
- Metrics monitored to ensure no loss of benefits (check)
- Effectiveness reviewed, process tuned (act)

- The figure shows a progression from first-time practice usage (in the center, "project plans new practice") to widespread organizational adoption
- Normal product (development) projects aren't organized to go through this entire spiral. At best, there may be a product-line architecture that could ideally do so if all planned products were done
- This figure shows some useful insights for process improvement projects that can help us assess to what extent an improvement has been adopted
- The model includes several patterns across the plan/do/check/act quadrants. These reflect
  - ✓ Increasing people and resource investments
  - ✓ Greater management understanding, attention, and commitment, and
  - ✓ Better control with process metrics
- Let's now talk about the steps/quadrants of the Shewart cycle

Plan

- Most new improvements are first tried by one project team. If it is a success, these improvements are spread to other projects
- As usage spreads, planning aspects of new adoptions include ways to provide better documentation, training, and support
- Eventually, demand outpaces the ability of the people helping the new adopters to keep pace, and plans for an ongoing infrastructure are developed and tuned

Plan: Identifying Risks and Deciding What to Do

- While P/D/C/A always starts with some form of commitment, the nature and strength of that commitment is constantly challenged as plans solidify
- Anticipating these challenges helps you to strengthen plans to help ensure success
- Business planning includes changes that makes good business sense to you and link to your goals. The closer the tie they make between you and your competitive needs, the easier it will be for you to decide and act
- You need to plan the scope, timing, cost, and payback of each change. Without such scoping, you have no basis for comparing proposed process-improvement projects to any other business proposals
- Understand your organization's readiness and enthusiasm for each proposed change
- Your culture, your existing processes, and your people's underlying beliefs all affect changes
- If you don't consider these aspects, even a simple process improvement can fail or not be cost effective
- Focus on developing core-competence is extremely important for organizations in software business
- By thinking about these needs strategically, managers more easily tie improvement objectives to business needs
- Such objectives are compelling and more likely to sustain management commitment than ones justified solely around some methodology or technology that is poorly understood by decision-making managers
- Core-competence planning is just as important as business planning in setting an organization's long-term strategy
- The key results of core-competence planning process are a set of clearly stated goals and action plans with clear links to business needs
- With these plans and goals, you have a strategic framework that explains what software abilities your organization must develop to remain competitive, and you have a way to describe the business aspects of any process-improvement recommendations
- Knowing which competencies your organization needs provides a framework for strategic decisions
- Some process-improvement projects are more costly, risky, timely, and so forth, than others
- An investment model of your software processes will help you to evaluate trade-off
- No matter how well you analyze the business urgency of a particular process improvement and its investment value, your organization might just not be ready for it

- If not, a decision to mandate people to make such changes could be very costly and ultimately unsuccessful
- A good subjective assessment should give you some sense of how widespread feelings are about key issue areas and how motivated people are to change
- Another approach is to directly evaluate the business and organizational forces that influence your organization's management commitment to potential improvements

Management Commitment Influencers

	Business	**Organizational**
Strategic	Vision Business Approach Core Competence	Organizational Maturity Process Improvement Infrastructure
Tactical	Customer Perceptions Market Share Product Cycle Time Profitability	Organizational Inertia Stability Cost/Time Alignment

- This table shows just one possible way to model many aspects that can be positive influencers. Unfortunately, when they aren't positive, they can distract and undermine your manager's best intentions
- One can think of situations when one of these influencers has seemed to stop process-improvement progress in its tracks
  - ✓ Cost cutting due to declining profits (profitability)
  - ✓ Reorganization that splits R&D into two different divisions or combines with another (stability)
  - ✓ Improvement-project resources pulled to help shorten a development schedule due to competitive product announcement (product cycle time)
- Any one of the twelve aspects can temporarily slow or even stop process improvements. It is useful to consciously evaluate each one to understand potential risks in your organization
- Strategic business aspects most reflect leadership of your organization's management
- Tactical business aspects are a constant source of tension, since they are tied to commonly accepted evaluations of business health and are reviewed monthly
- Strategic organizational aspects can be the strongest negative influencers. They are often victims of management responses to tactical business pressures
- Tactical organizational aspects are the most controllable from lower in the organization. If any of these are negative influences, it is often possible to work around them
- There are many reasons to select various processes to improve. These include critical customer dissatisfiers, ease of solution, organizational readiness and enthusiasm, return on investment (ROI), and reinforcement of basic organizational values
- A strong selection process with a well-thought-out balance of tactical and strategic projects with clearly defined goals shows people your rationale and helps to motivate them toward successful change

Do

- Progress in this and the following quadrants parallels the stages described in the plan
- Adoption by the second ring projects starts a trend to improve and adapt documentation and training
- By the third ring, users have formed realistic expectations for the new process and use metrics to see how well these expectations are being met
- They also continue to tune the process
- In the outermost two rings, good support is common, automation (where feasible) improves process efficiency and quality, and the number of infrastructure personnel is finally reduced to optimal ongoing levels

Do: Structuring for Success

- After countless years of high pressure and late projects, software managers have a well-developed sense of urgency
- They are also used to frequent and rapid changes
- However, they seem to respond to many proposed process-improvement projects with skepticism
- Software managers have seen many improvements begun with perfectly good intentions that have died due to lack of management or organizational follow-through
- What causes such breakdowns, and what can you do help prevent them
- One of the most powerful approaches is to work with people to mentally shift to a future desired state
- When they make such a shift, they will use words like “can, might, should, or will”
- The spiral model shows that improvements progress by stages. The amount of management attention and influence also varies by stage
- One aspect is the key people in an organization that require special attention

Project Managers

- They play a key role by ensuring that all the involved people work together productively
- This takes positive feedback, open listening, a person the team can go to who will help to fairly resolve disagreements, and someone who can sense when people are stuck and get them unstuck

Champion

- A project manager is generally also the project’s champion
- The champion promotes a project, removes obstacles, enthusiastically supports implementers and users, and leads through active involvement

Sponsor

- A sponsor is a person in a higher organizational position who ensures funding and high-level public enthusiasm and encouragement
- A sponsor also helps coach to keep things progressing smoothly

Opinion Leader

- Sometimes there are also less obvious technical leaders, and are called "opinion leaders"
- They may not be as vocal as technical leaders, but before other people will even consider a change, they look to whether opinion leaders accept it

Infrastructure Members

- When people first work with new process, ready access to someone to answer questions and remove roadblocks is invaluable
- Later, these same key infrastructure members are continuing contact points to capture and exchange process adjustments and improvements
- Minimizing negative commitment influences and optimizing support from key people are already big challenges
- The single most important step you can take is to create and manage from a good process-improvement project plan
- Sometimes project plans aren't easy to do
- A set of storyboard pictures or maps of where you plan to go is also a very useful way to start a project plan
- Think in terms of a measurable framework and a clear picture of the intermediate steps that lead to widespread adoption
- Storyboards can be used to organize your plan/do/check/act thoughts and to create a slideset that you will evolve and present many times over the evolution and adoption of your process improvement

Typical Storyboard Outline

DIV Situation, Challenges	Plan
Project Selection Criteria, Planned Changes, Objectives	
Cost/Benefit Analysis	
Project Milestones	Do
Methods for Measuring and Validating	Check
Approach for Standardizing Process Changes	Act

- Structuring a process-improvement project for success is a big challenge. Fortunately, there are useful techniques that will help you understand
  - ✓ The forces in your organization and its business that influence improvement efforts
  - ✓ The people who are most likely to sponsor and lead the efforts and how to keep them motivated, and
  - ✓ The best ways to plan and execute improvement projects to evolve specific processes and abilities into strategically important assets

Check

- After the first ring project team uses a new practice, they describe their "success" in terms that match whatever measurements they took
- As a minimum, this probably means their project completed reasonably close to expected schedules, that initial customer reactions were good, and that the process improvement seemed to work
- As other projects adopt the new process in later spiral rings, improvement metrics become more precise until process performance becomes well understood and is measured regularly to ensure optimum performance is maintained

Check: Measuring Success

- Let's say you have completed your project and some managers have asked you to discuss it
- What will they want to know?
- What kinds of questions will they ask?
- What kinds of savings did you get?
- What did it cost you?
- How does your project compare with others?
- What did you learn that should be passed on to other projects?
- If you started with a storyboard that included the "check" part, you are probably in good shape
- If you didn't, then some of these questions might be tough to answer. Your answers will be no better than those from the "process-improvement" projects with no experimental validation
- The amount of effort you might need to set up measures for your process-improvement project depends on what measures already exist in your organization
- Non-Comment Source Statements (NCSS)
- Engineering Months (EM)
- Defects
- Calendar Months (CM)
- If you don't measure these four primitive metrics, it will be difficult to frame expectations and even more difficult to describe success
- You need to collect relevant data

Act

- The speed at which adoptions of a new practice occur will heavily depend on how credible early adopters are, how good a job they did of measuring their processes, how well they can communicate their success, and how well their improvement matches the readiness of the organization and its management team to make similar changes on other projects
- Gaining ever-increasing management commitment as you move outward on these segments suggests the critical elements of follow-through and "sales" that are necessary for process improvements
- You can think of this sequence as pursuing "market share" for adopting a practice

Act: Leveraging Success

- The essence of the "Act" step is to revise and spread improved processes
- This is done by describing the successes so that different audiences understand and get excited
- No process should ever be "final". This means we must
  - ✓ Model improvements on previous successes,
  - ✓ Create similar situations and environments, and
  - ✓ Reinforce behaviors that move people as quickly as possible to improved conditions
- The best way to summarize what is needed to extend best-practice adoption and effective usage outward around the process-improvement spiral is
  - ✓ Proactively identify and seek support of process-improvement champions and sponsors,
  - ✓ Reinforce management awareness and commitment with a strong business case for each desired process improvement,
  - ✓ Build an infrastructure strong enough to achieve and hold software core competence, and
  - ✓ Measure the extent of adoption of each desired process improvement until it is operating effectively, efficiently, and across all appropriate parts of the organization
- Your challenge is to change your management's thinking from solely tactically oriented process-improvement projects to include strategically oriented organizational capabilities
- Such capabilities require business-decision makers to lead an evolution of key abilities from isolated individual knowledge to organizational processes and infrastructure that capture that knowledge
- The challenges of software process improvements are complex and many-faceted
- PDCA model provide an effective approach to adopting improved software engineering processes, methods, and tools

Conclusions

- Use the PDCA model to help you:
  - ✓ Make sure you have clear goals and plans for what you want at the end of each of multiple spirals
  - ✓ Understand how much you can expect to successfully do in stages
  - ✓ Measure progress and results that emphasize your incremental successes
  - ✓ Evaluate management and organizational commitment influencers and regularly reenlist and reinforce support on your continued exciting journey

References

- Successful Software Process Improvement by Robert R. Grady, Chapter 1

Lecture 28

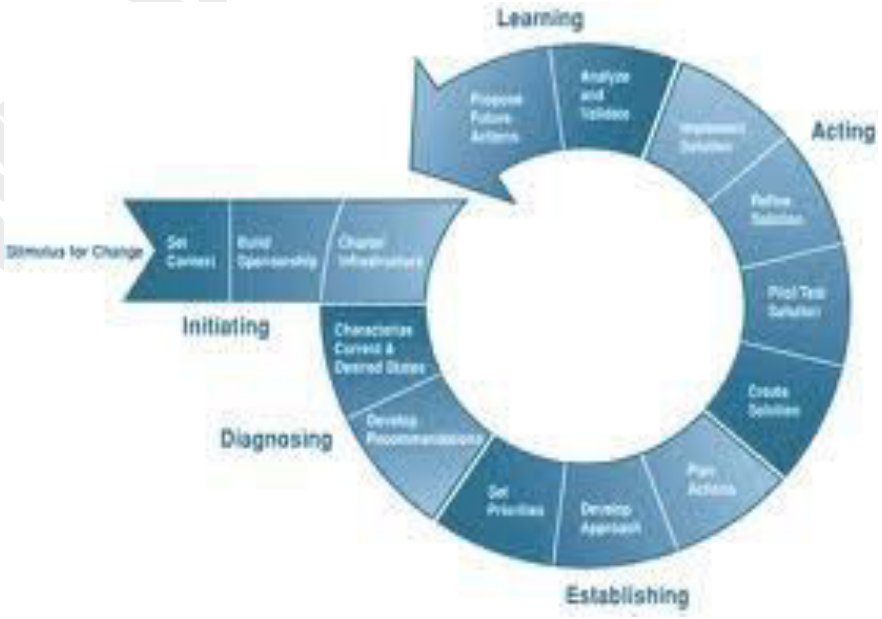
Software Process Improvement Using SEI's IDEAL Model

Introduction

- The IDEAL model is an organizational improvement model that serves as a roadmap for initiating, planning, and implementing improvement actions
- The IDEAL model is named for the five phases it describes: initiating, diagnosing, establishing, acting, and learning
- The IDEAL model as originally conceived was a life-cycle model for software process improvement based upon the Capability Maturity Model® (CMM®) for Software, and for this reason the model used process improvement terms
- The SEI has revised the IDEAL Model for broader application
- IDEAL model provides a usable, understandable approach to continuous improvement by outlining the steps necessary to establish a successful improvement program
- Following the phases, activities, and principles of the IDEAL model has proven beneficial in many improvement efforts
- The model provides a disciplined engineering approach for improvement, focuses on managing the improvement program, and establishes the foundation for a long-term improvement strategy
- As described, the model consists of five phases. Let's introduce these phases now:

Initiating	Laying the groundwork for a successful improvement effort
Diagnosing	Determining where you are relative to where you want to be
Establishing	Planning the specifics of how you will reach your destination
Acting	Doing the work according to plan
Learning	Learning from the experience and improving your ability to adapt

The IDEAL Model



The Initiating Phase

- Critical groundwork is completed during the initiating phase. The business reasons for undertaking the effort are clearly articulated
- The effort's contributions to business goals and objectives are identified, as are its relationships with the organization's other work
- The support of critical managers is secured, and resources are allocated on an order-of-magnitude basis. Finally, an infrastructure for managing implementation details is put in place
- Stimulus for Change
  - ✓ It is important to recognize the business reasons for changing an organization's practices. The stimulus for change could be unanticipated events or circumstances, an edict from someone higher up in the organization, or the information gained from benchmarking activities as part of a continuous improvement approach
  - ✓ Whatever the stimulus, it can have far-reaching influence on the effort's visibility, conduct, and ultimate success, change for the sake of change rarely results in significant improvement. In general, when the business reasons for change are more evident, there is greater buyin throughout the organization and there are greater chances for success
- Steps in the Initiating Phase
- Set Context
  - ✓ Once the reasons for initiating change have been clearly identified, the organization's management can set the context for the work that will be done
  - ✓ "Setting context" means being very clear about where this effort fits within the organization's business strategy
  - ✓ What specific business goals and objectives will be realized or supported by this change?
  - ✓ How will it affect other initiatives and ongoing work?
  - ✓ What benefits (such as return on investment or improved capabilities and morale) will result?
  - ✓ Context and implications often become more evident as the effort proceeds, but it is important to be as clear as possible regarding these issues early in the effort
- Build Sponsorship
  - ✓ Effective sponsorship is one of the most important factors for improvement efforts
  - ✓ It is necessary to maintain sponsorship levels throughout an improvement effort, but because of the uncertainty and chaos facing the organization in the beginning of the effort, it is especially important to build critical management support early in the process
  - ✓ The commitment of essential resources is an important element of sponsorship, but effective sponsors often do much more than this
  - ✓ Sponsors can be most effective if they give personal attention to the effort and stick with it through difficult times
- Charter Infrastructure
  - ✓ Once the reason for the change and the context are understood and key sponsors are committed to the effort, the organization must set up a mechanism for managing the implementation details for the effort

- ✓ The infrastructure may be temporary or permanent, and its size and complexity may vary substantially depending on the nature of the improvement
- ✓ For a small effort, the infrastructure may be a single part-time employee; for a large and complex effort, such as software process improvement, it may involve 2-3% of the organization's people across a number of groups
- ✓ Chartering the infrastructure involves developing explicit written agreements that document and clarify expectations and describe responsibilities
- ✓ The activities of the initiating phase are critical
- ✓ If they are done completely and well, subsequent activities can proceed with minimal disruption
- ✓ If they are done poorly, incompletely, or haphazardly, then time, effort, and resources will be wasted in subsequent phases

The Diagnosing Phase

- The diagnosing phase builds upon the initiating phase to develop a more complete understanding of the improvement work
- During the diagnosing phase two characterizations of the organization are developed: the current state of the organization and the desired future state
- These organizational states are used to develop an approach for improving business practice
- Characterize Current and Desired States
- Develop Recommendations
- Characterize Current and Desired States
  - ✓ Characterizing the current and desired states is similar to identifying the origin and destination of a journey
  - ✓ Characterizing these two states can be done more easily using a reference standard such as the CMM for Software
  - ✓ Where such a standard is not available, a good starting point is the factors identified as part of the "stimulus for change" activity
  - ✓ This activity should focus on elements critical to the changes being introduced, rather than every aspect of an organization's work
- Develop Recommendations
  - ✓ The recommendations that are developed as a part of this activity suggest a way of proceeding in subsequent activities
  - ✓ The diagnosing phase activities are most often performed by a team with experience and expertise relevant to the task at hand
  - ✓ Their recommendations often weigh heavily in the decisions made by key managers and sponsors

The Establishing Phase

- The purpose of the establishing phase is to develop a detailed work plan
- Priorities are set that reflect the recommendations made during the diagnosing phase as well as the organization's broader operations and the constraints of its operating environment

- An approach is then developed that honors and factors in the priorities
- Finally, specific actions, milestones, deliverables, and responsibilities are incorporated into an action plan
- Step in Establishing Phase
- Set Priorities
  - ✓ The first activity of this phase is to set priorities for the change effort
  - ✓ These priorities must take many factors into account: resources are limited, dependencies exist between recommended activities, external factors may intervene, and the organization's more global priorities must be honored
- Develop Approach
  - ✓ Combining increased understanding of the scope of work (gained in the diagnosing phase) with a set of priorities leads to the development of a strategy for accomplishing the work and identifying resource availability
  - ✓ Technical factors might include the specifics of installing the new technology and new skills and knowledge required for using a technology
  - ✓ Non-technical factors, including the organization's culture, likely sources of resistance, sponsorship levels, and market forces, also must be considered
- Plan Actions
  - ✓ With the approach defined, a detailed implementation plan can be developed
  - ✓ This plan includes schedule, tasks, milestones, decision points, resources, responsibilities, measurement, tracking mechanisms, risks and mitigation strategies, and any other elements required by the organization

The Acting Phase

- The activities of the acting phase help an organization implement the work that has been conceptualized and planned in the previous three phases
- These activities will typically consume more calendar time and more resources than all of the other phases combined
- Step in Acting Phase
- Create Solution
  - ✓ The acting phase begins with bringing all available key elements together to create a "best guess" solution to address the previously identified organizational needs
  - ✓ These key elements might include existing tools, processes, knowledge, and skills, as well as new knowledge, information, and outside help
  - ✓ The solution, which may be quite complex and multi-faceted, is often created by a technical working group
- Pilot/Test Solution
  - ✓ Once a solution has been created, it must be tested, as best guess solutions rarely work exactly as planned
  - ✓ This is often accomplished through a pilot test, but other means may be used

- Refine Solution
  - ✓ Once the paper solution has been tested, it should be modified to reflect the knowledge, experience, and lessons that were gained from the test
  - ✓ Several iterations of the test-refine process may be necessary to reach a satisfactory solution
  - ✓ A solution should be workable before it is implemented, but waiting for a "perfect" solution may unnecessarily delay the implementation
- Implement Solution
  - ✓ Once the solution is workable, it can be implemented throughout the organization
  - ✓ Various roll-out approaches may be used for implementation, including top-down (starting at the highest level of the organization and working down) and just-in-time (implementing project-by-project at an appropriate time in its life cycle)
  - ✓ No one roll-out approach is universally better than another; the approach should be chosen based on the nature of the improvement and organizational circumstances
  - ✓ For a major change, implementation may require substantial time and resources

The Learning Phase

- The learning phase completes the improvement cycle
- One of the goals of the IDEAL Model is to continuously improve the ability to implement change
- In the learning phase, the entire IDEAL experience is reviewed to determine what was accomplished, whether the effort accomplished the intended goals, and how the organization can implement change more effectively and/or efficiently in the future
- Records must be kept throughout the IDEAL cycle with this phase in mind
- Step in Learning Phase
- Analyze and Validate
  - ✓ This activity answers several questions:
    - In what ways did the effort accomplish its intended purpose?
    - What worked well?
    - What could be done more effectively or efficiently?
  - ✓ Lessons are collected, analyzed, summarized, and documented
  - ✓ The business needs identified during the initiating phase are reexamined to see if they have been met
- Propose Future Actions
  - ✓ During this activity, recommendations based on analysis and validation are developed and documented
  - ✓ Proposals for improving future change implementations are provided to appropriate levels of management for consideration

Useful Tips on SPI

- Proactively identify and seek support of process-improvement champions and sponsor
- Reinforce management awareness and commitment with a strong business case for each desired process improvement

- Build an infrastructure strong enough to achieve and hold software core competence
- Measure the extent of adoption of each desired process improvement until it is effectively, efficiently, and across all appropriate parts of the organization

Conclusions

- IDEAL model provides an effective approach to adopting improved software engineering processes, methods, and tools

References

- www.sei.cmu.edu/ideal/ideal.html
- The IDEAL Model: A Practical Guide for Improvement by Jennifer Gremba and Chuck Myers

Lecture 29

Software Process Assessments

- Since software applications are built by human beings, the methods, tools, and practices used for software have become subject to study and analysis
- Software process assessments are normally carried out on-site by independent assessors
- In-house assessments are also common
- Formal software assessments appear to have originated within the IBM corporation in the early 1970s
- An assessment finds all the strengths and weaknesses associated with software
- The primary goal of software process assessments is to gather qualitative information about the practices, methods, tool suites, and organizational structures used for software
- Some organizations gather additional information on topics such as office space and ergonomic factors used by the organizations undergoing the assessment
- As the software has become a worldwide business phenomenon, the need for software process assessments has become more and more important
- Software is a very troublesome and difficult technology
- For large systems, failures outnumber successes by a considerable margin
- Software process assessment data is normally gathered by means of structured interviews with managers and technical personnel
- To ensure consistency among enterprises, standard questionnaires are utilized by most of the assessment methods, although the questions vary from method to method
- Scripts and questionnaires for data collection used by many of the major kinds of software assessments are now utilized throughout the world
- Many forms of software assessments exist, following is a partial list of assessment methods
- Software Process Assessment Methodologies
    - ✓ The SPR assessment methodology
    - ✓ The SEI assessment methodology
    - ✓ The ISO assessment methodology
    - ✓ The SPICE assessment methodology
    - ✓ The TickIT assessment methodology
    - ✓ The Trillium assessment methodology
    - ✓ The Howard Rubin Associates assessment methodology
    - ✓ The Gartner Group assessment methodology
- A software process assessment is somewhat equivalent to undergoing an annual medical examination
- The annual physical examination is not intended to cure any specific disease. The main purpose is to find out about the health of the patient
- If the physical examination finds any serious problems, then the physician can prescribe a therapy program
- However, the examination itself is performed for diagnostic purposes, not for therapeutic purposes

- Unfortunately, this analogy of medical checkups is overlooked by clients of software process assessments
- Companies regard the assessment as an end in itself, rather than a means to an end
- Assessments by themselves will not improve software quality or productivity, or cure any specific problems that are found
- But without the assessment, many problems would remain invisible
- Assessments should be viewed as nothing more than annual checkups to ensure that no new problems have occurred and that progress is being made on curing problems found during the last assessment
- Out of a total of 250 factors that can affect the outcome of a software development project, an individual project would be affected by 15 to 20 factors
- These factors are collected from projects in the domains of system software, information systems, military software, commercial software, outsourced projects, and end user applications
- The major topics that are covered by software process assessment can be summarized by ten key process areas that are common among many of the various assessment methods
- These ten areas are not specific to any individual kind of assessment, but are the topics examined by several forms of assessment because they have a very significant impact on software project schedules, productivity, and quality levels
- Key Process Areas
  - ✓ Project management methods such as estimation
  - ✓ Requirements-gathering and analysis methods
  - ✓ Design and specification methods
  - ✓ Coding methods
  - ✓ Reusability methods
  - ✓ Change control methods
  - ✓ User documentation methods
  - ✓ Pretest defect removal methods such as inspections
  - ✓ Testing methods and tools
  - ✓ Maintenance methods and tools
- Software process assessments attempts to identify practices that can lead to successful outcomes on software projects
- Conversely, process assessments also seek to identify harmful practices that might lead to delays or unacceptable outcomes
- As software projects are very labor intensive and require exceptional skills on the part of technical and management personnel, software assessments may also examine the factors that are concerned with hiring and supporting software personnel
- Following is a list of ten key personnel-related topics that may be examined during software process assessments
- Key Personnel-Related Topics
  - ✓ Staff hiring practices
  - ✓ Staff training and education
  - ✓ Management training and education

- ✓ Specialists and occupational groups
- ✓ Compensations levels
- ✓ Office ergonomics
- ✓ Organizational structures
- ✓ Morale surveys and results
- ✓ Work patterns and overtime
- ✓ Staff turnover rates

Introduction to Some Assessment Methods

- Because software development is still a largely manual activity performed by skilled craftsmen, it is obvious that success on software projects requires very capable personnel led by capable managers
- Good teams must also utilize methods and practices that are known to produce excellent results, and should avoid practices that have led to failures

SEI Assessment Approach

- The SEI assessment data is collected by means of on-site interviews using both standard questions, observations, and information data
- Once collected, the assessment data are analyzed and used to place a software organization on one of the five well-known maturity levels or capability levels of the SEI CMMI
- Although the assessment and maturity-level concepts can be considered separately, most people regard the two as being a linked set
- Formal SEI assessments are performed by the SEI or by a number of licensed consulting companies
- Informal "SEI-style" assessments that companies perform on their own or that are done by consultants who are not licensed by the SEI but who use assessment approaches derived from the published SEI materials
- There is a large body of knowledge related to SEI assessments
- SEI assessment approach originally dealt primarily with the software processes and methodologies used by very large companies that produced very large systems
- SEI has now broadened the sets of factors included under the assessment
- Software process improvements cannot be successful if they overlook key topics or emphasize one topic and ignore others
- Thus, the broad range of topics included in SEI's programs emphasize that success in software requires being good in many activities

The SPR Assessment Approach

- The SPR assessment interview approach is a group activity for each project analyzed
- The SPR consultant meets with the project manager and as many as seven technical staff members at the same time
- All the participants have seen the questionnaires ahead of time, so they know the kinds of sought information

- The "official" assessment questionnaire is filled out by the SPR consultant using the group consensus as the source of information
- Sometimes there may be debate or polarized opinions among team members, as can occur on any human activity. A full set of responses is recorded in this case
- These differences are then resolved
- The SPR assessment approach uses structured interviews based on a set of scripted questions
- The SPR questionnaires make use of sets of 100 of more than 250 multiple-choice questions
- Some of the SPR questions overlap the same topics as the SEI questions, although the forms are different
- Once a sample of projects has been analyzed using the questionnaires, the answers to the questions are input directly into a tool that performs a statistical analysis of the results and shows the mean values and standard deviations of all factors
- Although strengths are very laudable, the real value of a process assessment for most clients is the discovery and the objective analysis of areas in which the client is lagging and needs improvement
- Identifying these areas of weakness is the key to a successful process improvement program
- Once an assessment is completed, the patterns of strengths and weaknesses observed usually lead to a desire to improve software development processes
- Once the areas of weakness have been analyzed and discussed, the client is in an excellent position to begin a successful improvement program
- Then organizations can follow a six-stage improvement program. These six phases provide an overall framework for software improvement strategies

Six-Stage Improvement Program

- Stage 1: Focus on management technologies
  - ✓ Stage 1 concentrates on management issues, which are critical to all downstream activities. Software process improvement is a fairly expensive undertaking that can stretch out over three to five years
  - ✓ Unless the management community is at state-of-the-art levels in cost estimating, value analysis, and risk analysis, funding for process improvement may be denied
  - ✓ Management needs to be trained in critical technologies such as planning, sizing, estimating, tracking, measurement, risk analysis, and value analysis
  - ✓ Managers collect ROI
  - ✓ Managers collect data that demonstrate progress
- Stage 2: Focus on software processes and methodologies
  - ✓ Stage 2 of improvement programs involves the introduction of specific process technologies, e.g., formal code and design inspections
  - ✓ Stage 2 improvements require training and cultural adjustments, but are not very expensive in terms of capital equipment or major investments
  - ✓ Solid approaches for dealing with requirements, design, development, and quality control

- ✓ According to a study, projects that used no formal methods at all lagged in terms of productivity, schedules, defect prevention, and defect removal efficiencies, against those projects which used them

- Stage 3: Focus on new tools and approaches
  - ✓ Stage 3 improvements may involve fairly costly tool suites and capital equipment, such as better workstations
  - ✓ As tools support processes, rather than the other way around, one can see why process improvements precede changes in tool suites
  - ✓ Use various tools and explore new or advanced technologies
  - ✓ Study shows the difference between leading and lagging companies are the use of quality assurance tools, testing tools, and project management tools
- Stage 4: Focus on infrastructure and specialization
  - ✓ Stage 4 improvements deal with corporate "infrastructures". An example of such a change is the establishment of formal quality assurance departments, formal measurement departments, and testing and maintenance departments
  - ✓ As infrastructure changes are both expensive and complex, they need to be deferred until the value of the previous kinds of changes has started to become obvious
  - ✓ This stage addresses issues of organization and specialization, and begins to move toward the establishment of specialized teams for handling testing, maintenance, integration, and configuration control
  - ✓ Formal quality assurance departments
  - ✓ Policies on continuing education
- Stage 5: Focus on reusability
  - ✓ Stage 5 improvements are the most valuable in terms of improving quality and productivity, but they are also the most challenging
  - ✓ Stage 5 moves toward a full and formal software reusability program
  - ✓ However, reuse is a double-edge sword
  - ✓ If the reusable materials are of top quality, then reuse has the best positive ROI of any software technology
  - ✓ However, if the reusable materials are error prone and filled with defects, then the ROI of software reuse switches from positive to sharply negative
  - ✓ Reusability has the best ROI of any technology, but effective reuse is not a game for amateurs
  - ✓ Successful software reuse demands state-of-the-art quality control, software measurements, and software management training
  - ✓ Some Reusable Components
    - Reusable architecture
    - Reusable requirements
    - Reusable plans
    - Reusable estimates
    - Reusable designs and specifications
    - Reusable interfaces

  - Reusable data
  - Reusable screens or screen elements
  - Reusable source code
  - Reusable user documents
  - Reusable test plans
  - Reusable test cases
- Stage 6: Focus on industry leadership
  - ✓ Stage 6 involves using software excellence for competitive purposes, such as winning a Baldrige Award and using this award to aid in gaining market share
  - ✓ Executives who understand and support software
  - ✓ Software project managers of exceptional ability
  - ✓ Software technical staff of exceptional ability
  - ✓ More specialists than average organizations
  - ✓ Better measurements than average organizations
  - ✓ Better than average customer satisfaction levels
  - ✓ Better than average staff morale levels
  - ✓ Powerful project management tool suites
  - ✓ Powerful software development tool suites
  - ✓ Powerful quality assurance and testing tool suites
  - ✓ Powerful geriatric tool for aging software

The Cost of Process Improvements

- Every company needs to create an individualized plan and budget for its improvement strategy. However, estimates are reported in literature
- Cost elements include training, consulting fees, capital equipment, software licenses, and improvement in office conditions

The Timing of Process Improvements

- Smaller companies move much more rapidly than larger corporations and government agencies
- Process improvement is a multi-year undertaking
- When there is polarization of opinion or political opposition, progress can be very slow or nonexistent

The Value of Process Improvements

-
-

References

- Software Assessments, Benchmarks, and Best Practices by Capers Jones (Chapters 2, 3, and 6)

Lecture 30
Software Process Benchmarks

Benchmarks

- Standard
- Yardstick
- Target
- Scale
- Point of reference

Software Benchmarks

- Benchmarks compare a company against industry norms
- Benchmarks collect quantitative and qualitative data on a number of important topics, including investments, staffing levels, development schedules, staff efforts, costs, quality, and customer satisfaction
- All of these factors are related to software process improvement initiatives
- Benchmark studies often include supplemental information on specific tool suites, programming languages, and formal methods utilized
- Thus, benchmark studies and assessment studies overlap to a degree
- Benchmarks are primarily comparisons between a specific company and other companies in the same kind of business
- Benchmark studies or comparisons of quantitative data are older than the software industry. Indeed, comparisons between the companies have been taking place for centuries. E.g., studies of market shares, staff compensation levels, executive pay, and customer satisfaction have long been performed
- Software benchmark is a formal comparison of software methods and results against those of other organizations
- Software benchmarks can be traced back to the 1960s, when studies of data center performance and downtimes started to be performed
- These were soon followed by corporate-level studies of investments in information technology, compensation studies of software personnel, and studies of software schedules, cost, and process
- Within the software industry, there are a number of levels of benchmark studies that have been noted to occur
- It would be a good idea to consider the various kinds of benchmarks

Macroeconomic Benchmarks

- Macroeconomic benchmarks are concerned with very large-scale issues such as impact of computers and information technology on economic health of nations and industries
- Macroeconomic benchmarks are usually performed by national governments and sometimes by economists or universities

Economic Benchmarks

- Economic benchmarks are concerned with topics such as whether investments in computers, software, factory automation, or process improvement benefit the profitability and market shares of companies that spend more than others
- *Productivity paradox*
- Economic benchmarks are performed by economists, universities, and by management consulting groups

Corporate IT Benchmarks

- Corporate information technology benchmarks deal with a host of interesting comparisons among companies
- Some of the comparisons that are studied under corporate benchmarks include
  - ✓ Percent of corporate employees in IT
  - ✓ Number of user supported per staff member in IT
  - ✓ Annual corporate spending for computer hardware equipment
  - ✓ Revenues per IT employee
  - ✓ Sales volumes correlated to IT expenses
  - ✓ Profitability correlated to IT expenses
  - ✓ Average salaries for IT workers
  - ✓ Number of specialist occupations employed within IT
  - ✓ Corporate revenues per IT worker
- IT benchmarks are produced by many universities, by consulting organizations, and sometimes by universities and consultants together
- Corporate benchmarks are also produced by software journals such as the annual software compensation and salary levels published by Computer-World

Customer Satisfaction Benchmarks

- Customer satisfaction benchmarks also predate the computer era and can be traced back more than 100 years. In the context of software applications, customer satisfaction surveys for specific products are often carried out by the vendors themselves
- However, comparative benchmarks between classes of similar products require more extensive data

Project-Level Benchmarks

- Project-level benchmarks for software have been carried out since the 1970s
- With hundreds of companies performing scores of benchmarks it is not surprising that there are many variations in how the work is performed

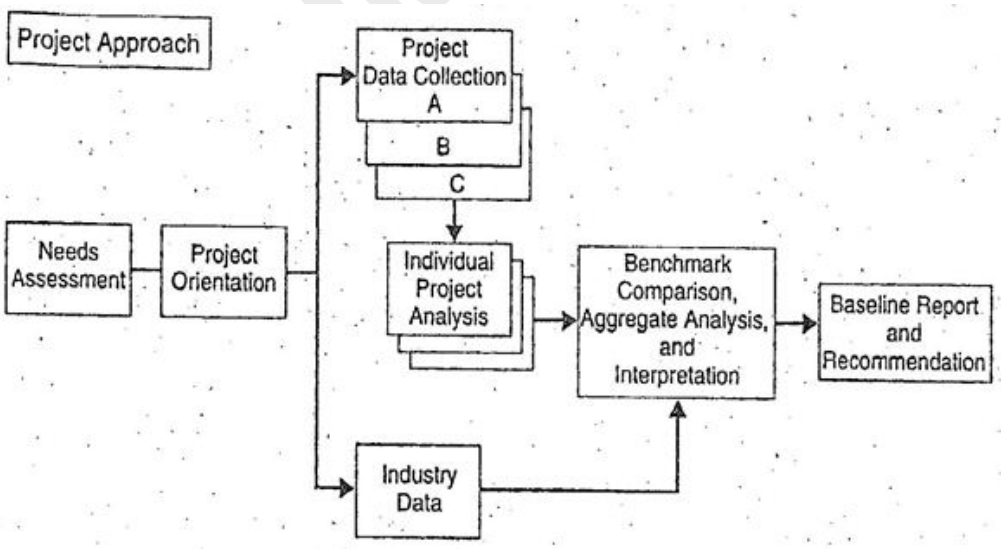
Benchmark studies

- Benchmark studies can be carried out in a variety of ways, including
  - ✓ Blind studies in which none of the participating companies are identified
  - ✓ Hybrid studies with a mix of named and unnamed participants
  - ✓ Open studies in which all participants are aware of one another
  - ✓ Targeted studies between two specific companies
  - ✓ General studies between one company and industry averages
- Benchmarks by means of mailed/e-mailed survey instruments
- Benchmarks by means of telephone surveys
- Benchmarks by means of on-site interviews

Software Benchmark Analysis

- The goal of a software benchmark analysis is to show clients exactly where they stand in the context of their own industry
- Normally, clients care about the average results from their industry, but care more about best-in-class results
- Simply placing the client against the background of their industry is not enough
- A good benchmark should also uncover information on why the results are the way they are
- Also, because benchmarks are precursors to process improvement programs, the benchmark should indicate some of the steps that might be needed to improve the client's standing against data

Basic Sequence of Software Benchmarks



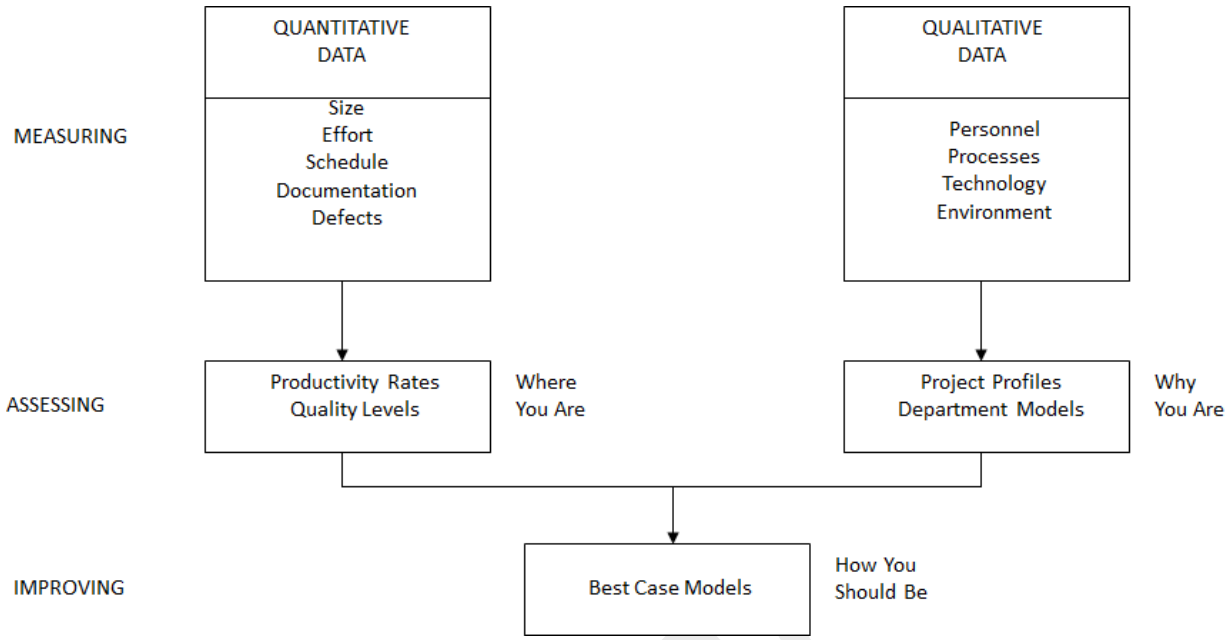
- Benchmarks are even more effective than assessments in causing companies to start software process improvement programs
- If a company is well below average ranges in productivity and quality, this situation presents tangible data that executives can understand easily
- For example, the virtues of moving up from an SEI CMMI level 1 to a level 3 might be abstract and difficult to explain, but the need to improve productivity rates from five function points per staff month to ten to reach industry norms is fairly unambiguous
- Accurate benchmarking of software projects was difficult for many years due to problems with the LOC software metric, which was unreliable for studies involving multiple programming languages
- The advent of the function point metric has opened a door to more accurate benchmark studies of software projects
- Benchmark studies can explore a number of different topics
- Very interesting software benchmarks can be carried out at the project level. These project-level benchmarks can generate very useful information. However, project-level benchmarks require a great deal of care to ensure an apples-to-apples comparisons

Most FAQs

- What are best-in-class productivity rates in our industry?
- What are best-in-class development costs per function point in our industry?
- What are best-in-class development schedules?
- What are best-in-class maintenance assignment scopes in our industry?
- What are best-in-class quality levels in our industry?
- What are the best development processes for software like ours?
- What are the best development tools for software like ours?
- What does it cost to move from SEI CMMI level 1 to SEI CMMI level 3?
- What are the differences in results between SEI CMMI level 1 and CMMI level 3?

- As can be seen from these very common questions, the most interesting topics from benchmarks are those associated with best practices and best-in-class results
- Averages are interesting and useful, but benchmark studies are often commissioned by companies with goals that are to be much better than average
- For benchmarks to be useful, or even for the comparisons to be valid, the projects that are compared during the benchmark have to be similar in size and in nature
- For benchmarking purposes, it is not a normal practice to compare unlike software applications, such as comparing the productivity level of a small Web applet with the productivity level of a large military system

Benchmarking Overview



Software Baselines

- A baseline is a milestone in the development of software
- Software benchmarks and baselines collect similar data
- Baselines compare a company against its own history for several years in the past
- Baselines collect quantitative and qualitative data on a number of important topics, including investments, staffing levels, development schedules, staff efforts, costs, quality, and customer satisfaction
- Software baselines are normally used as a starting points when beginning process improvement programs
- Accurate baseline provides a firm quantitative basis for productivity, schedules, costs, quality, and user satisfaction to judge rate of improvement
- Another very common use of baselines is when outsource agreements include a contractual obligation for the vendor to improve quality or productivity
- Baselines are technically very challenging and difficult to create with acceptable accuracy
- This is because most companies do not routinely collect accurate quality and productivity data. In fact most companies don't even collect accurate cost data on how much they spend on software projects
- In addition to the fact that the effort and costs for many activities are incomplete, there are several other significant cost elements that are seldom recorded at all
- One of the most important missing pieces of information is the amount of unpaid overtime applied to software projects. This is very important factor for international software comparisons

- Another missing piece of information is the work that software clients themselves perform during development cycles
- Because leakage is so common, initial baselines are often very unrealistic.
- For example, if a software organization's actual productivity is 5.0 FPs per staff months but their cost tracking system only captures low-level design, coding, and testing costs, their apparent productivity might look like 15.0 FPs per staff months
- Many software cost tracking systems omit requirements, high-level design, project management, creation of user manuals and documents, and the work of specialists in areas such as software quality assurance, database administration, change control, and administrative support
- Sometimes, the activities omitted actually accumulate more costs than the activities that are tracked
- The bottom line is that the baselines are very challenging to create well and accurately
- If they are going to serve a purpose in contracts between outsource vendors and clients without leading to litigation, then baselines need to be accurate and reliable

References

- Software Assessments, Benchmarks, and Best Practices by Capers Jones (Chapter 3)

Lecture 31
Agile Software Process

- Agile software engineering combines a philosophy and a set of development guidelines
- The philosophy encourages customer satisfaction and early incremental delivery of software, small; highly motivated project teams; informal methods; minimal software engineering work products, and overall development simplicity
- The development guidelines stress delivery over analysis and design (although these activities are not discouraged), and active and continuous communication between developers and customers
- There are a number of agile process models
  - ✓ Adaptive Software Development
  - ✓ Extreme Programming
  - ✓ Feature-Driven Development
  - ✓ SCRUM
  - ✓ Agile Modeling
  - ✓ Crystal
- Two key features of these development methods are
  - ✓ They accept that change will occur – changes in requirements and changes in technology; they therefore adopt a more 'adaptive' or iterative approach to planning
  - ✓ In suggesting how projects should be organized, they make a point of trying to work with human nature rather than ignoring it
- One consequence of both of these is that they tend to create less documentation than conventional methods – it is not that they don't value documentation – just that they actively avoid creating it for its own sake
- Proponents of agile methods have published a 'Manifesto for Agile Software Development'

The Agile Manifesto

- We are uncovering better ways of developing software by doing it and helping other does it. Through this work we have come to value;

individuals and interactions	over	process and tools
working software	over	comprehensive documentation
customer collaboration	over	contract negotiation
responding to change	over	following a plan

- That is, while there is value in items on the right, we value the items on the left more
- Note that they do state that there is value in the items on the right
- The Agile Manifesto is based on the following twelve principles:

Agile Principles

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
- Business people and developers must work together daily throughout the project
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
- Working software is the primary measure of progress
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
- Continuous attention to technical excellence and good design enhances agility
- Simplicity--the art of maximizing the amount of work not done--is essential
- The best architectures, requirements, and designs emerge from self-organizing teams
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Agile Methods

- Not all agile methods are the same
- XP focuses on design, coding and testing activities, whereas Adaptive Software Development and the Crystal family both aim to cover a wider scope. They both place great emphasis on scalability – using appropriate methods for the project in hand. Crystal in particular is based on a matrix of methods depending on the size and criticality of the project
- As a general principle, most agile methods focus on issues such as overall approaches to programming, testing, documentation, planning and team interaction, rather than the details of, for example, how you carry out your configuration management
- In practice, therefore, an organization wanting to adopt agile methods need to fill in gaps from elsewhere
- The key high-level messages for us from 'agile methods' are:
  - ✓ Don't assume the waterfall life cycle is the only approach
  - ✓ Work with your customers and users, not against them
  - ✓ Accept that the real world (from which your requirements arise) does change, and manage that change
  - ✓ Don't try to be more rigorous than the situation demands
- Software process improvement has been practiced for over two decades, but picked up after the introduction of CMM-SW and later many model based methods for software process improvement were introduced

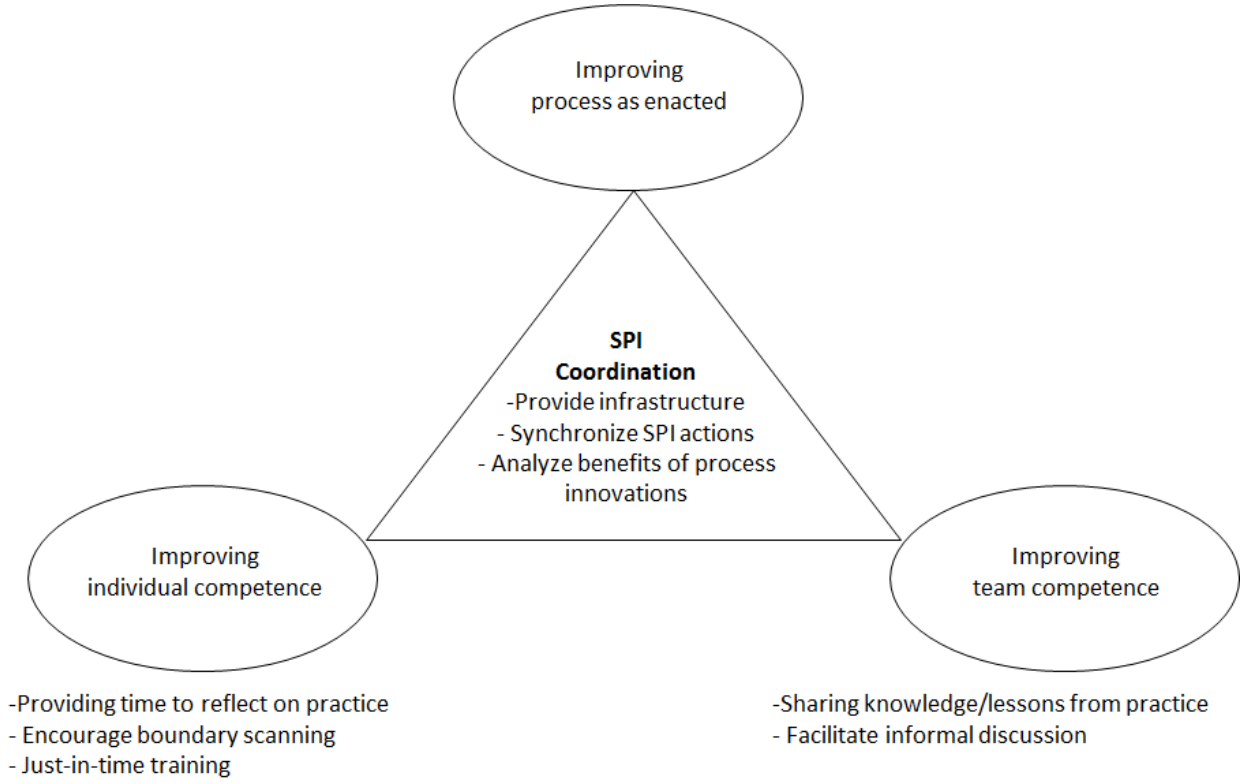
- Whilst agile methods place considerable emphasis on process improvement ("At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly") the agile methods movement is seen by some as the antithesis of capability maturity, and a recipe for anarchy
- The 'manifesto' is certainly intended as a counter-blast to, for example, the higher 'maturity levels' of CMM/CMMI
- But for serious professionals concerned with efficient system development, responding to customer needs, and avoidance of bureaucracy which does not serve a useful purpose, this manifesto must give us a pause for thought
- It is also worth noting that reaching CMMI capability level 4 or 5 for a process area is conceptually feasible but may not be economical except, perhaps, in situations where the product domain has become very stable for an extended period of time
- How many of us work in such a product domain?
- ISO 9001:2000 requires that an "organization shall continually improve the effectiveness of the quality management system through the use of quality policy, quality objectives, audit results, analysis of data, corrective and preventive actions and management review."
- Agile methods do not conflict with this
- It would therefore appear that agile methods are entirely consistent with process improvement, both in general, and as interpreted by ISO 9001:2000/2008

Agility Applied to Process Improvement

- Agility need not only apply to software development. It can also apply to our own process improvement activities
- Some of the agile principles can be changed to the following
  - ✓ Our highest priority is to satisfy the user through early and continuous delivery of valuable productivity aids
  - ✓ Welcome changing requirements
  - ✓ Software developers and process improvers must work together closely throughout the program
  - ✓ Recognize the importance of motivated individuals; give them the environment and support they need, and trust them to get the job done
  - ✓ Simplicity – the art of maximizing the amount of work not done – is essential
  - ✓ At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly
- Process improvement should be carried out in an agile manner to an iterative life cycle, taking full account of the way in which people actually work, and taking care to work closely with the people at the programming shop
- One of the principles which 'agile process improvement' may need to challenge is the inexorable (unyielding/unalterable) link between measurement and process improvement
- We should see quantitative data as a valuable bonus for process improvement, rather than an absolute prerequisite

- New software development technology is still appearing at an astounding rate
- Some of it is self-evidently good
- Some of it doomed to the annals of interesting (or less interesting) culs-de-sac (dead end streets)
- We should adopt the following approach to process improvement
  - ✓ If people believe something needs doing, then do it
  - ✓ If something looks like a good approach, then try it
  - ✓ If you can cost-effectively collect data to support your process improvement activities, then so much the better, but don't hold your breath
  - ✓ If your new approach requires spending lots of money, then of course you will need to justify it, but make sure that by 'lots of money' you really mean lots of money - $1,000 on a $5,000,000 project is not 'lots of money'. Be (informally) quantitative and logical about deciding whether to be (formally) quantitative
  - ✓ Estimating is an area which is by definition quantitative, and you can't get away from the fact that people like to know in advance how much you're asking them to spend. Don't use this section as a justification for avoiding methodical approaches to estimating
- The messages of agile methods are:
  - ✓ While we value and use processes and tools, we should not allow them to deter us from talking to our customers and users
  - ✓ We should never forget that working (and usually, maintainable) software is the key objective – we should always ask whether every piece of our documentation serves a useful purpose
  - ✓ While we need to agree with our customers what we are delivering, and document that agreement, many projects in the past should benefited from more collaboration and less haggling
  - ✓ No-one benefits from following a plan that no longer meets the real requirements – sure we have to plan, but we also need to know when to re-plan
- Engineering professionalism should not prevent us from having good relationships with our users and customers and responding to their needs
- Some of you will see all of this as the thin end of the wedge, working inexorably to separate them from the holy grail of CMMI level 5. Others recognize that a lively debate is an essential component of continual improvement
- An agile approach to SPI would be responsive and flexible to local needs, encourage innovation in the process, build SPI projects around those who are motivated, encourage self-organizing competent teams, and promote sustainable development of the processes
- The following figure conceptualizes the features considered necessary to support an agile approach to SPI

A Model of an Agile SPI Method



- Improvement in process management is 'a continuous effort to design an efficacious process by understanding the relationship between process configurations and process outcomes and embedding the knowledge in the process through routines and formal process definitions'

Factors in Ensuring a Successful SPI Program

- Changes must be aligned with business strategy and priorities should be based on the business context
- Commitment to improve and investment required
- Must come from top
- SPI requires the consensus of all stakeholders and they need to have ownership of the change
- Current processes must be understood
- Change must be continuous
- Improvement infrastructure is required to identify and manage the implementation of changes
- The results of the SPI should be monitored
- Explicitly moving away from a software process philosophy of stability to one that is adaptive and flexible to local concerns will encourage a change in the actor's intentions, as the norms of the organization will change from remaining consistent to finding the most appropriate means of undertaking a task

- This is not to suggest that methodologies or process life-cycles should be abandoned, rather the aim should be to produce a successful product rather than following a pre-defined methodology at all costs
- The objective is not change for its own sake, as there is a need to balance between optimizing the current approaches and experimenting with new ideas
- The primary means for adjusting the process to suit the product development needs will be through learning from practice as individuals identify adjustments in the life cycle
- To achieve this ongoing learning from practice we need to do more than just recognize that it is happens gradually, we need to encourage innovation in the process
- Encouragement to innovate, whether by bringing in new ideas from the software profession or developing approaches individually, requires a suitable mixture of experimentation, training, boundary scanning, and time to reflect on the current experience
- To support the learning and innovation activity, the SPI initiative can be supported through a suitable infrastructure
- The SEI view of setting up process action teams under the software engineering process group (SEPG) is a model that can be adapted to suit this more emergent view of software process improvement
- Rather than the SEPG being the sole identifiers of process areas that need to be introduced, they could act as a facilitating group. By supporting through training, encouragement, resources, and progress management the SEPG can enable the introduction of significant changes to the defined process
- SEPG can also act as a research and development group, facilitating boundary scanning, thus helping an organization to be more aware of and ready to take on external innovations
- It could also act as the control group for process actions, synchronizing SPI actions over time. New initiatives can be incorporated into the improvement action plan
- One way to improve the relevance of the SPI activity is to align it with the emergent business strategy
- Both planned and improvised improvements could be coordinated to focus upon the areas that are expected to be important to the business to engender a sense of purpose over the longer term
- A final point: whilst good processes are fundamental to success in the software business, good people will bring their own processes with them, and can therefore produce good work in an 'immature' organization
- Good processes without good people will achieve very little

References

- "Agile process improvement?" by Keith Southwell, in TickIT 3Q02, pp. 3-14
- "Towards an Agile Approach to Software Process Improvement: Addressing the Changing Needs of Software Products" by Ian Allison, Communications of the IIMA, 2005 Volume 5 Issue 1, pp. 67-76

Lecture 32
Process Patterns

Process

- A series of actions in which one or more inputs are used to produce one or more outputs

Patterns

- A general solution to a common (recurring) problem or issue, one from which a specific solution may be derived
- Patterns can exist at all scales
    - ✓ Christopher Alexander

Process Patterns

- A process pattern is a collection of general techniques, actions, and/or tasks (activities) for developing software
- A process pattern describes what you should do but not the exact details of how you should do something
- When applied together in an organized manner, process patterns can be used to construct software process for your organization
- Since process patterns do not specify how to perform a given task, they can be used as reusable building blocks from which you may tailor a software process that meets the specific needs of your organization
    - ✓ Why?
- Software development patterns come in many flavors, including but not limited to analysis patterns, design patterns, organizational patterns, and process patterns
- A process pattern is a pattern which describes a proven, successful approach and/or series of actions for developing software
- It can be argued that process patterns and organizational patterns go hand in hand
- Organizational patterns describe common management techniques or organizational structures
- Organization patterns describe proven, successful approaches for organizing and managing people involved with the software process
- An important feature of process patterns is that it is possible to develop them for all aspects of development
- The point to be made is that the scope of a single process pattern may range from a high-level view of how applications are developed to a more-detailed view of a specific portion of the software process
- There are at least three types of process patterns that we are interested in
    - ✓ Task process patterns
        - Task process patterns depict the detailed steps to perform a specific task, such as the Technical Review and Reuse First process patterns

- ✓ Stage process patterns
  - Stage process patterns depict the steps, which are often performed iteratively, of a single project stage
  - A project stage is a higher-level form of process pattern, one that is often composed of several task process patterns
  - A stage process pattern is presented for each project stage such as the Program and Rework stages
- ✓ Phase process patterns
  - Phase process patterns depict the interactions between the stage process patterns for a single project phase, such as Initiate and Delivery phases
  - A phase process pattern is a collection of two or more stage process patterns
  - Project phases are performed in a serial manner, this is true of both structured development and object development
  - Phase process patterns are performed in serial order, made up of stage process patterns which are performed iteratively

Documenting Process Patterns

- We can combine the existing work in process patterns with that of the existing process improvement community
- The implication is the need for a format/template that both communities understand and agree to
- In that light, let's discuss one way of documenting a process pattern

Process Patterns Format

- Name
  - ✓ Provide a concise, strong name for the pattern, such as Program or Reuse First
- Forces
  - ✓ Applicable forces motivating the process pattern
- Initial context/Entry conditions
  - ✓ Indicate the situation to which the pattern solution applies, and if applicable the entry conditions that must be true before the process may begin
- Solution
  - ✓ Describe in detail how to perform the steps/activities of the process pattern. You may also choose, particularly for phase and stage process patterns, to describe management, quality assurance, and risk management issues, as well as indicate potential metrics to collect when working the process
- Resulting context/Exit conditions
  - ✓ Indicate the situation/context which will result from performing the process pattern solution, including if applicable the conditions that must be true for the process to be considered complete

- Related Patterns
  - ✓ Indicate other patterns that this pattern is composed of, is a part of, or is associated to
- Known Uses/Examples
  - ✓ Indicate where/how the process pattern has been applied in use. For example, the Technical Review task process pattern can be applied to the management of peer reviews, code reviews, model reviews, and management reviews

Example of a Task Process Pattern

Technical Review

- The Technical Review task process pattern describes how to organize, conduct, and follow through on the review of one or more deliverables

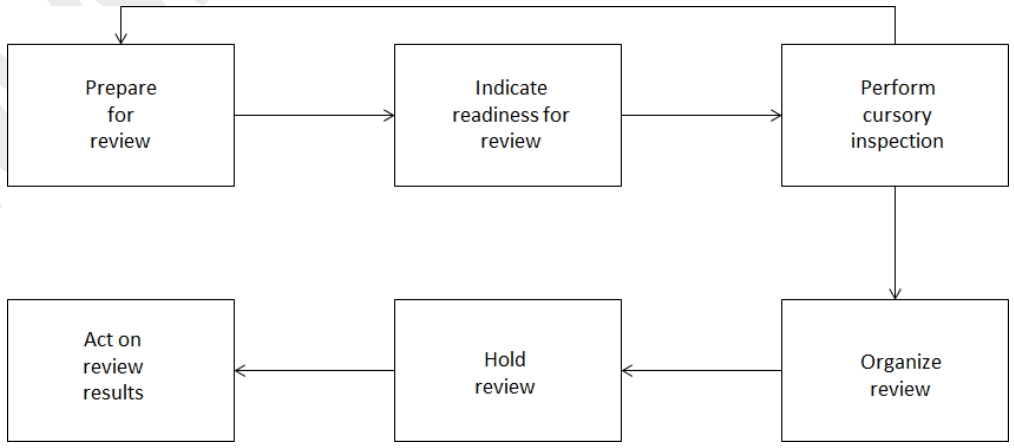
Technical Review – Forces

- First, the deliverables (models, prototypes, documents, source code, …) produced during the development process help to define the software and related products to be released to your user community, therefore you should validate that each deliverable is of sufficient quality before building on it
- Second, because the cost of fixing defects increases the later they are detected in the development life cycle as a result of the error snowballing throughout your work, you want to detect defects as early as possible so you may fix them early (and inexpensively)
- Third, because it is difficult to review your own work you want "a second set of eyes" to review a deliverable
- Fourth, you want to communicate your work to others, and one way to do that is to have your teammates review the deliverables that you produce

Technical Review – Initial Context

- There are one or more deliverables to be reviewed, the deliverables are ready to be reviewed, and the development team is ready to have the deliverables reviewed

Technical Review – Basic Steps



Technical Review – Solution

- There are six basic steps to technical reviews process pattern (model reviews, document reviews, prototype reviews, requirement reviews, and code inspections are all specific processes that follow the Technical Review process pattern).
- Prepare for review
  - ✓ The item(s) that are to be reviewed are gathered, organized appropriately, and packaged so that they may be presented to the reviewers
- Indicate readiness for review
  - ✓ The development team must inform the review manager, often a member of quality assurance, when they are ready to have their work reviewed as well as what they intend to have reviewed
- Perform cursory inspection
  - ✓ The first thing that the review manager must do is determine if the development team has produced work that is ready to be reviewed. The manager will probably discuss the development team's work with the team leader and do a quick rundown of what they have produced. The main goal is to ensure that the work to be reviewed is good enough to warrant getting a review team together
- Organize review
  - ✓ The review manager must schedule a review room and any equipment needed for the review, invite the proper people, and distribute any materials ahead of time that are needed for the review
- Hold review
  - ✓ Technical reviews should take about two hours so as not to overwhelm the people involved. The entire development team should attend, or at least the people responsible for what is being reviewed, to answer questions and to explain/clarify their work. There are typically between three to five reviewers, as well as the review manager, all of whom are responsible for doing the review
  - ✓ It is important that all material is reviewed. It is too easy to look at something quickly and assume that it is right. It is the job of the review facilitator to ensure that everything is looked at, that everything is questioned
- Act on review results
  - ✓ A document is produced during the review describing both the strengths and weaknesses of the work being reviewed. This document should provide both a description of any weakness, why it is a weakness, and provide an indication of what needs to be addressed to fix the weakness
  - ✓ This document will be given to the development team so that they can act on it, and to the review manager to be used in follow-up reviews in which the work is inspected again to verify that the weaknesses were addressed

Technical Review – Resulting Context

- Senior management is assured that the development team has produced quality deliverables that meet the needs of their user community
- The development team, and the reviewers, have a better understanding of the deliverables that they are building and how their work fits into the overall software project
- Individual team members and reviewers are likely to learn new techniques during the review, either techniques applied to the deliverable itself, management techniques applied during the review, or development techniques suggested during the review to improve the deliverable

Example of a Stage Process Pattern

Program

- An important aspect of software development, one of many to be exact, is the actual development of the source code
- As experienced developers know, there is far more to programming than just sitting down at a computer and typing in source code
- The Program stage process pattern describes the iterative tasks/activities of programming

Program – Forces

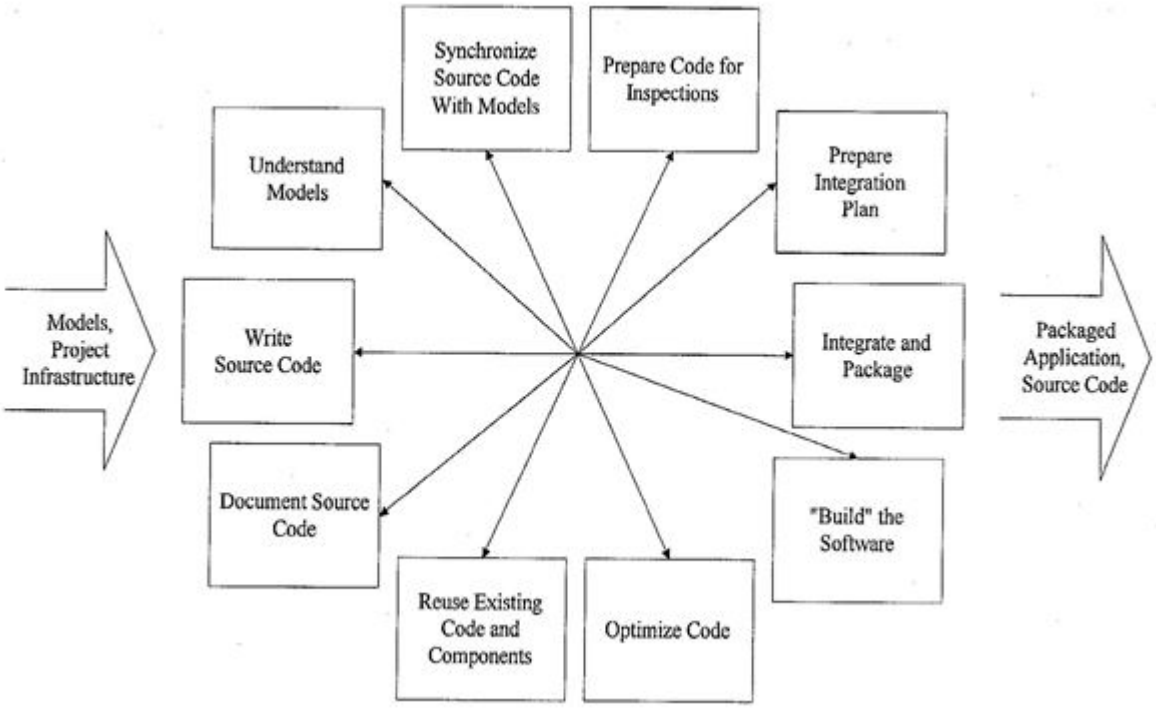
- Programmers need to develop software that meets the needs of their user communities, needs that have been reflected in the form of models and documents developed during the Model stage and the Define and Validate Initial Requirements stage
- The developed source code should reflect the information contained in these deliverables, yet at the same time may drive changes to them as programmers gain a detailed understanding of the domain
- Furthermore, most organizations want software to be developed in a timely and efficient manner but at the same time want it to be maintainable and extensible so that changes in the future may be made efficiently and quickly

Program – Initial Context/Entry Conditions

- Several conditions must be met before coding may begin. First, your design models should be in place for the code that you intend to write
- Second, your project infrastructure should be in place, defined during the Define Infrastructure stage of the Initiate phase.
- The infrastructure includes the development and supporting tools that your programmers will use as well as the standards and guidelines that they will follow
- Third, programmers must be available to do the work

Program - Solution

- The Program Process Pattern



- This figure depicts the process pattern for the Program stage, showing that there is more to this stage than simply writing source code: you need to understand the models, then seek out reusable artifacts to reduce your work load, then document what you are going to write, then write the code, then inspect and improve it, then test and fix it, and then finally package it

Program – Resulting Context/Exit Conditions

- First, your code should have passed inspection
- Second, the code should work (it passed testing)
- Third, the code should have been optimized sufficiently
- Fourth, if applicable the software should be integrated and packaged for delivery

Example of a Phase Process Pattern

Construct Phase

- The main goal of the Construct phase is to build working software that is ready to be tested and delivered to your user community
- This software will be accompanied by the models and source code that was used to develop it, a test plan to verify that the software works, any reusable artifacts that can be used on future projects, and the initial documentation and training plans supporting the software

Construct Phase – Forces

- There are several forces applicable to the Construct Phase, including a lack of understanding of how to work the phase by both senior management and by developers; an unwarranted focus on programming to the neglect of modeling, testing, and generalization; and a penchant by everyone involved to cut corners and take shortcuts that more often than not result in poor quality software that is late and over budget anyway

Construct Phase – Initial Context/Existing Conditions

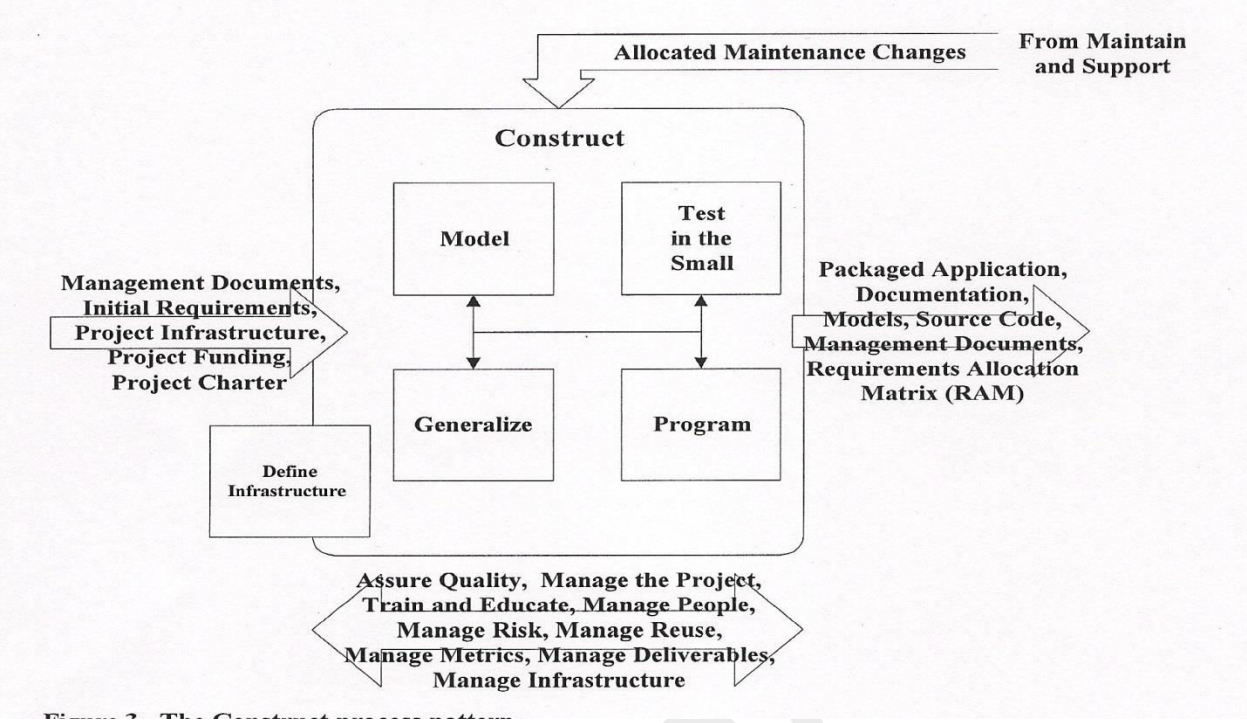
- The Construct phase can be entered two different ways, either from the Initiate phase or from the Maintain and Support phase
- Regardless, there are several conditions that must be met before the Construct phase may begin
- First, the key project management documents (project plan, estimate, schedule, risk assessment, …) should be available and up-to-date
- Second, the project infrastructure should be defined, or at least a good portion of it is defined, so that the tools, processes, and standards are available to your development team
- Third, the high-level requirements for your software should be in place as well as the project charter for your team
- Fourth, maintenance changes applicable to the software you are constructing should be allocated to the release that you are working on (this is applicable only for existing software that is being updated)
- Finally, your development team should be selected and made available (as best as possible) for when they are needed by your project

Construct Phase – Solution

- An important point is that you are not starting from scratch when you enter the Construct phase
  - ✓ important management documents such as the project plan and initial risk assessment have been defined, the initial requirements for your application should have been defined, the project infrastructure is (mostly) defined, and the funding and charter for the project have been obtained

The Construct Phase Process Pattern

- The four iterative stages of the Construct phase are highly interrelated
- The Model stage concentrates on the abstraction of the technical and/or problem domain via the use of diagrams, documents, and prototypes
- The Program stage focuses on the development and documentation of program source code
- The Generalize Stage is critical to your organization's reuse efforts as it focuses on identifying reusable items, or items that may become reusable once modified, from a software project
- This is effectively "opportunistic reuse" because you attempt to develop reusable items by harvesting your work after the fact, instead of "systematic reuse" in which you design software during modeling to be reusable



- The goal of the Test In The Small Stage is to verify and validate the deliverables developed by the other stages of the Construct Phase
- In many ways this stage is the equivalent of unit testing from the structured world combined with quality assurance techniques such as code inspections and technical reviews

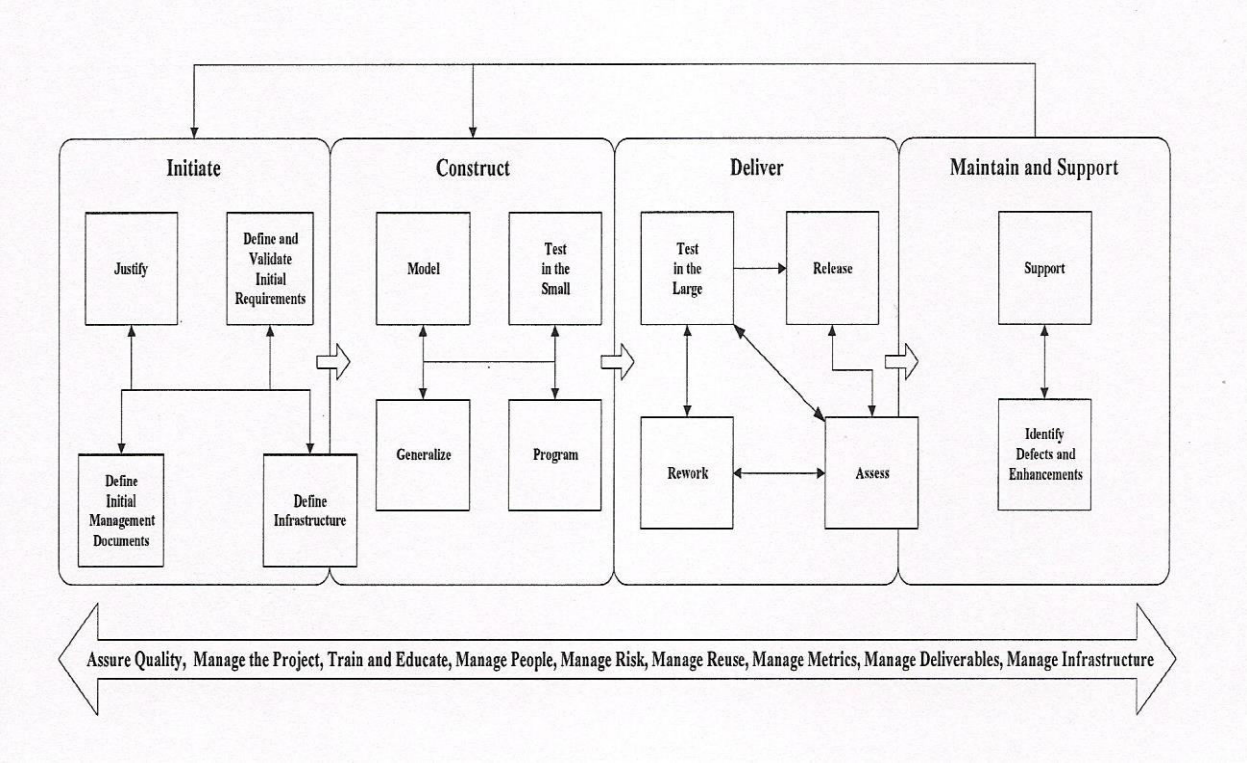
Construct Phase – Resulting Context/Exit Conditions

- The Construct phase effectively ends when a code/development freeze has been declared
- For a code/development freeze to be official, the following deliverables must be in place (when applicable):
  - ✓ Models (Class Model, Use-Case Model, Sequence Diagrams, …), Requirements Allocation Matrix (RAM), Source code, Master Test/QA Plan, User Documentation, Operations Documentation, Support Documentation, the software itself, Training Plan, Release Plan, and Lessons Learned
- At this point your software is ready to move on to the Deliver phase where it will be tested in the large, reworked as needed, and released to your user community

Why Process Patterns?

- They are an excellent mechanism for communicating approaches to software development that have proven to be effective in practice
- They are reusable building blocks from which an organization may tailor a mature software process

The Object-Oriented Software Process



- There are four project phases within the OOSP – Initiate, Construct, Deliver, and Maintain and Support – each of which is described by a phase process pattern
- There are 14 project stages in the OOSP – Justify, Define and Validate Initial Requirements, Define Initial Management Documents, Define Infrastructure, Model, Program, Test In The Small, Generalize, Test In The Large, Rework, Release, Assess, Support, and Identify Defects and Enhancements – each of which is described by a stage process pattern
- Project stages are performed in an iterative manner within the scope of a single project phase
- Project phases, on the other hand, are performed in a serial manner within the OOSP
- Process patterns are a key enabler for tailoring/defining a mature software process for your organization
- The reality of process improvement, however, is that you cannot make all of the changes that you want to immediately; it is simply too great a change for your organization to absorb at once
- Software process improvement promoters suggest that you prioritize the process improvements that your organization needs to make, expect that it will take several years to make the needed changes, and expect that you will experience difficulties while doing so
- Experience shows that organizations that try to make immediate, large-scale process changes are likely to fail doing so

Process Anti-Patterns

- Approaches to software development that are proven to be ineffective in practice, such as hacking

Summary

- Process patterns are reusable building blocks from which your organization can tailor a mature software process
- Process patterns describe proven approaches to developing software, approaches that can be used within your organization to increase the quality, maintainability, and extensibility of software

References

- Ambler, S. W., 'An Introduction to Process Patterns,' – www.ambysoft.com
- Ambler, S. W., 'Process Patterns'
- Ambler, S. W., 'More Process Patterns'

Lecture 33

ISO/IEC 12207:2008

- Information Technology Software Life Cycle Processes
- ISO: International Standards Organization
- IEC: International Electro-technical Commission

ISO 12207

- ISO 12207 is an ISO standard for software lifecycle processes
- ISO/IEC 12207 is the first International Standard to provide a comprehensive set of life cycle processes, activities and tasks for software that is part of a larger system, and for standalone software products and services
- It was first introduced on August 1, 1995
- The ISO 12207 standard establishes a process of lifecycle for software, including processes and activities applied during the acquisition and configuration of the services of the system
- Each Process has a set of outcomes associated with it
- The standard has the main objective of supplying a common structure so that the buyers, suppliers, developers, maintainers, operators, managers and technicians involved with the software development use a common language
- This common language is established in the form of well-defined processes
- ISO/IEC 12207 also provides a process that can be employed for defining, controlling, and improving software life cycle processes
- The structure of the standard was intended to be conceived in a flexible, modular way so as to be adaptable to the necessities of whoever uses it
- The standard is based on two basic principles: modularity and responsibility

Modularity

- Modularity means processes with minimum coupling and maximum cohesion

Responsibility

- Responsibility means to establish a responsibility for each process, facilitating the application of the standard in projects where many people can be legally involved
- The set of processes, activities and tasks can be adapted according to the software project
- These processes are classified in three types: basic, for support and organizational

Try not to use this

- The support and organizational processes must exist independently of the organization and the project being executed
- The basic processes are instantiated according to the situation

- This International Standard can be used in one or more of the following modes
  - ✓ By an organization
    - To help establish an environment of desired processes
    - These processes are to be supported by an infrastructure of methods, procedures, techniques, tools and trained personnel
    - The organization may then employ this environment to perform and manage its projects and progresses systems through their life cycle stages
    - In this mode this International Standard is used to assess conformance of a declared, established set of life cycle processes to its provisions
  - ✓ By a project
    - To help select, structure and employ the elements of an established set of life cycle processes to provide products and services
    - In this mode this International Standard is used in the assessment of conformance of the project to the declared and established environment
  - ✓ By an acquirer and a supplier
    - To help develop an agreement concerning processes and activities
    - Via the agreement, the processes and activities in this International Standard are selected, negotiated, agreed to and performed
    - In this mode this International Standard is used for guidance in developing the agreement
  - ✓ By organizations and assessors
    - To perform assessments that may be used to support organizational process improvement

Life Cycle Process Groups

- System Context Processes
- Software Specific Processes
- Each process of this standard is described in terms of the following attributes:
  - ✓ The title conveys the scope of the process as a whole
  - ✓ The purpose describes the goals of performing the process
  - ✓ The outcomes express the observable results expected from the successful performance of the process
  - ✓ The activities are a list of actions that are used to achieve the outcomes
  - ✓ The tasks are requirements, recommendations, or permissible actions intended to support the achievement of the outcomes

The Life Cycle Processes of 12207:2008

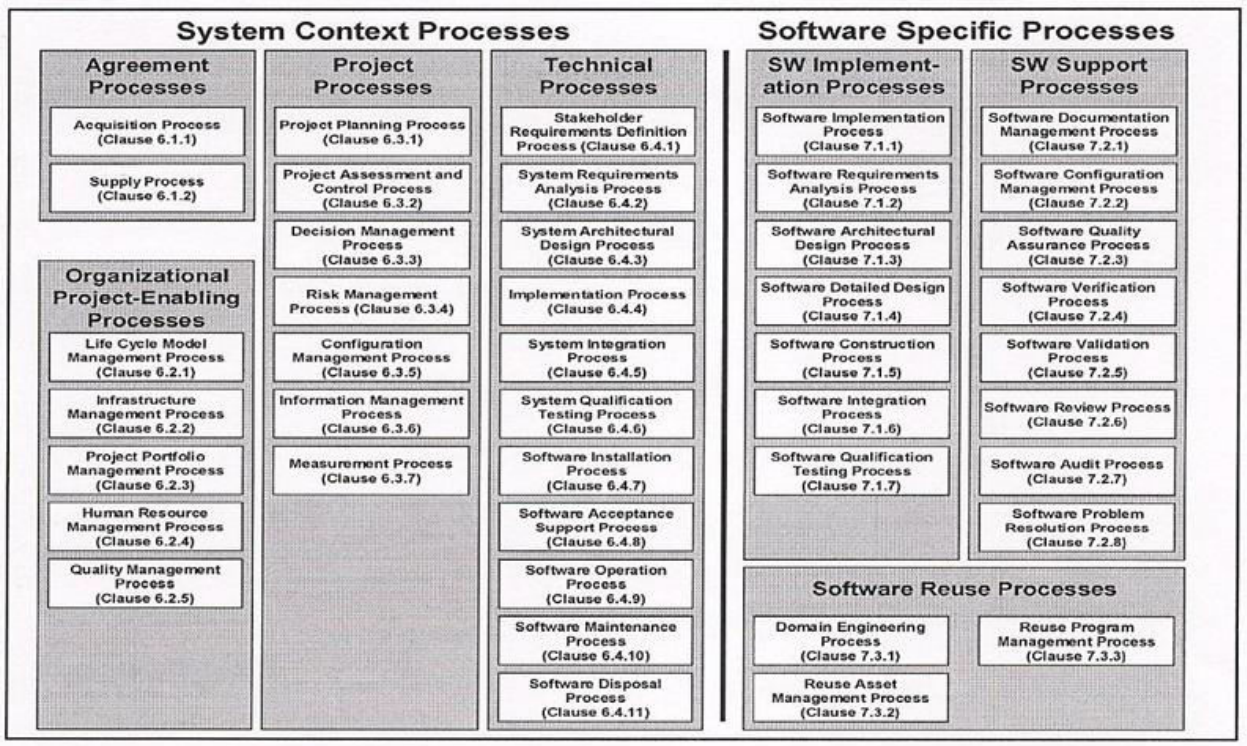


Figure 1 — Life Cycle Process groups

Agreement Processes

- These processes define the activities necessary to establish an agreement between two organizations
  - ✓ There are two agreement processes
  - ✓ Acquisition Process
  - ✓ Supply Process
- If the Acquisition Process is invoked, it provides the means for conducting business with a supplier of products that are supplied for use as an operational system, of services in support of an operational system, or of elements of a system being developed by a project
- If the Supply Process is invoked, it provides the means for conducting a project in which the result is a product or service that is delivered to the acquirer

Organizational Project-Enabling Processes

- There are five organizational project-enabling processes
  - ✓ Life Cycle Model Management Process
  - ✓ Infrastructure Management Process
  - ✓ Project Portfolio Management Process
  - ✓ Human Resource Management Process
  - ✓ Quality Management Process

- The Organizational Project-Enabling Processes manage the organization's capability to acquire and supply products or services through the initiation, support and control of projects
- They provide resources and infrastructure necessary to support projects and ensure the satisfaction of organizational objectives and established agreements
- They are not intended to be a comprehensive set of business processes that enable management of the organization's business

Project Processes

- In this International Standard, the project has been chosen as the context for describing processes concerned with planning, assessment and control
- The principles related to these processes can be applied in any area of an organization's management
- There are two categories of Project Processes
  - ✓ Project Management Processes
    - The Project Management Processes are used to establish and evolve project plans, to assess actual achievement and progress against the plans and to control execution of the project through to fulfillment
    - Individual Project Management Processes may be invoked at any time in the life cycle and at any level in a hierarchy of projects, as required by project plans or unforeseen events
    - Simply put, the Project Management Processes are used to plan, execute, assess and control the progress of a project
    - There are two Project Management Processes
    - Project Planning Process
    - Project Assessment and Control Process
  - ✓ Project Support Processes
    - The Project Support Processes provide a specific focused set of tasks for performing a specialized management objective
    - They are all evident in the management of any undertaking, ranging from a complete organization down to a single life cycle process and its tasks
    - There are five Project Support Processes
    - Decision Management Process
    - Risk Management Process
    - Configuration Management Process
    - Information Management Process
    - Measurement Process

Technical Processes

- There are ten technical processes
  - ✓ Stakeholder Requirements Definition Process
  - ✓ System Requirements Analysis Process
  - ✓ System Architectural Design Process

- ✓ Implementation Process
- ✓ System Integration Process
- ✓ System Qualification Testing Process
- ✓ Software Installation Process
- ✓ Software Acceptance Support Process
- ✓ Software Operation Process
- ✓ Software Maintenance Process
- ✓ Software Disposal Process

- The Technical Processes are used to define the requirements for a system, to transform the requirements into an effective product, to permit consistent reproduction of the product where necessary, to use the product, to provide the required services, to sustain the provision of those services and to dispose of the product when it is retired from service
- The Technical Processes define the activities that enable organizational and project functions to optimize the benefits and reduce the risks that arise from technical decisions and actions
- These activities enable products and services to possess the timeliness and availability, the cost effectiveness, and the functionality, reliability, maintainability, usability and other qualities required by acquiring and supplying organizations
- They also enable products and services to conform to the expectations or legislated requirements of society, including health, safety, security and environmental factors

Software-Specific Processes

- Software Implementation Processes
- Software Support Processes
- Software Reuse

Software Implementation Processes

- The Software Implementation Processes are used to produce a specified system element (software item) implemented in software
- This process transforms specified behavior, interfaces and implementation constraints into actions that create a system element implemented as a software product or service, otherwise known as a "software item"
- This process results in a software item that satisfies architectural design requirements through verification and stakeholder requirements through validation
- For the purpose of clear description, processes are sometimes decomposed into smaller pieces
- Some processes are decomposed into activities and/or lower-level processes. A lower-level process is described when the decomposed portion of the process itself satisfies the criteria to be a process

Sub-Processes of Software Implementation Process

- Software Requirements Analysis Process
  - ✓ The Software Requirements Analysis Process in this International Standard is a lower-level process of the Software Implementation Process

- ✓ The purpose of Software Requirements Analysis Process is to establish the requirements of the software elements of the system

- Software Architectural Design Process
  - ✓ The Software Architectural Design Process in this International Standard is a lower-level process of the Software Implementation Process
  - ✓ The purpose of the Software Architectural Design Process is to provide a design for the software that implements and can be verified against the requirements
- Software Detailed Design Process
  - ✓ The Software Detail Design Process in this International Standard is a lower-level process of the Software Implementation Process
  - ✓ The purpose of the Software Detailed Design Process is to provide a design for the software that implements and can be verified against the requirements and the software architecture and is sufficiently detailed to permit coding and testing
- Software Construction Process
  - ✓ The Software Construction Process in this International Standard is a lower-level process of the Software Implementation Process
  - ✓ The purpose of the Software Construction Process is to produce executable software units that properly reflect the software design
- Software Integration Process
  - ✓ The Software Integration Process in this International Standard is a lower-level process of the Software Implementation Process
  - ✓ The purpose of the Software Integration Process is to combine the software units and software components, producing integrated software items, consistent with the software design, that demonstrate that the functional and non-functional software requirements are satisfied on an equivalent or complete operational platform
- Software Qualification Testing Process
  - ✓ The Software Qualification Testing Process in this International Standard is a lower-level process of the Software Implementation Process
  - ✓ The purpose of the Software Qualification Testing Process is to confirm that the integrated software product meets its defined requirements

Software Support Processes

- The Software Support Processes provide a specific focused set of activities for performing a specialized software process
- A supporting process assists the Software Implementation Process as an integral part with a distinct purpose, contributing to the success and quality of the software project. There are eight of these processes
  - ✓ Software Documentation Management Process
  - ✓ Software Configuration Management Process
  - ✓ Software Quality Assurance Process
  - ✓ Software Verification Process
  - ✓ Software Validation Process

- ✓ Software Review Process
- ✓ Software Audit Process
- ✓ Software Problem Resolution Process

Software Reuse Processes

- The Software Reuse Process Group consists of three processes that support an organization's ability to reuse software items across project boundaries
- These processes are unique because, by their nature, they operate outside the bounds of any particular project
  - ✓ Domain Engineering Process
  - ✓ Reuse Asset Management Process
  - ✓ Reuse Program Management Process
- This International Standard allows tailoring of processes in certain situations and supports tailoring process

Tailoring Process

- Tailoring is not a requirement for conformance to the standard. In fact, tailoring is not permitted if a claim of "full conformance" is to be made. If a claim of "tailored conformance" is made then tailoring is to be performed as required by this process
- The purpose of the Tailoring Process is to adapt the processes of this International Standard to satisfy particular circumstances or factors that:
  - ✓ Surround an organization that is employing this International Standard in an agreement
  - ✓ Influence a project that is required to meet an agreement in which this International Standard is referenced
  - ✓ Reflect the needs of an organization in order to supply products or services
- It should be noted that tailoring may diminish the perceived value of a claim of conformance to this standard
- This is because it is difficult for other organizations to understand the extent to which tailoring may have deleted desirable provisions
- An organization asserting a single-party claim of conformance to this standard may find it advantageous to claim absolute conformance to a smaller list of processes rather than tailored conformance to a larger list of processes

References

- System and Software Engineering – Software Life Cycle Processes sponsored by Software & Systems Engineering Standards Committee of the IEEE Computer Society (.pdf file)

Lecture 34

SPICE – ISO/IEC 15504

- SPICE is a major international initiative to support the development of an International Standard for Software Process Assessment
- SPICE is also known as ISO/IEC TR Standard 15504
- SPICE is acronym for Software Process Improvement and Capability dEtermination

Principal/Original Goals of SPICE Project

- To develop a working draft for a standard for software process assessment
- To conduct industry trials of the emerging standard
- To promote the technology transfer of software process assessment into the software industry world-wide
- The first goal of the project was achieved in June 1995, with the release of Version 1 of a draft standard for software process assessment
- The SPICE documents have been carried through the international standardization process and have been published as ISO/IEC TR 15504 - Software Process Assessment
- The methods of Software Process Assessment are coming more generally into use in the management of software development, acquisition and utilization, in the face of substantial evidence of the success of such methods in driving improvements in both quality and productivity
- The primary impetus for the use of assessment has come not from the mainstream of the software development industry, but rather from acquirers of large, critical software-intensive systems - notably in the defense and telecommunications sectors
- At the same time, there has always been a recognition that process assessment can be a strong and effective driver for process improvement
- The major focus of use of the CMM/CMMI has been on improvement, and most acquirers use assessment approaches as part of a partnership approach with their suppliers, focusing on improvement
- In addition, methods have been developed with a specific focus on improvement
- As experience with the technique grows, substantial empirical evidence has accumulated demonstrating the benefits that can be derived from an assessment-based improvement programs
- The increasing number of assessment approaches available, and the increasing use of assessments in commercially-sensitive areas, were the key motivating factors behind the development and acceptance of this International Standard for software process assessment
- All industries now depend on software for competitive advantage. Growth will only be achieved if industry meets and even exceeds international standards and worlds best- practice

ISO/IEC 15504

- ISO/IEC 15504 is the reference model for the maturity models (consisting of capability levels which in turn consist of the process attributes and further consist of generic practices) against

which the assessors can place the evidence that they collect during their assessment, so that the assessors can give an overall determination of the organization's capabilities for delivering products (software, systems, and IT services)

- The reference model defines a process dimension and a capability dimension

Processes

- The process dimension shows processes as divided into the five process categories of:
  - ✓ Customer-supplier
  - ✓ Engineering
  - ✓ Supporting
  - ✓ Management
  - ✓ Organization
- It is expected that the process categories will expand, particularly for IT service process categories and enterprise process categories

Capability Levels and Process Attributes

- For each process, ISO/IEC 15504 defines a capability level on the following scale:

Level	Name
5	Optimizing Process
4	Predictable Process
3	Established Process
2	Managed Process
1	Performed Process
0	Incomplete Process

- The capability of processes is measured using process attributes. The international standard defines nine process attributes
- Process Attributes
  - ✓ 1.1 Process Performance
  - ✓ 2.1 Performance Management
  - ✓ 2.2 Work Product Management
  - ✓ 3.1 Process Definition
  - ✓ 3.2 Process Deployment
  - ✓ 4.1 Process Measurement
  - ✓ 4.2 Process Control
  - ✓ 5.1 Process Innovation
  - ✓ 5.2 Process Optimization
- Each process attribute consists of one or more generic practices, which are further elaborated into practice indicators to aid assessment performance

Assessment of Process Attributes

- Each process attribute is assessed on a four-point (N-P-L-F) rating scale:
  - ✓ Not achieved (0 - 15%)
  - ✓ Partially achieved (>15% - 50%)
  - ✓ Largely achieved (>50% - 85%)
  - ✓ Fully achieved (>85% - 100%)
- The rating is based upon evidence collected against the practice indicators, which demonstrate fulfillment of the process attribute

Assessments

- ISO/IEC 15504 provides a guide for performing an assessment. This includes:
  - ✓ The assessment process
  - ✓ The model for assessment
  - ✓ Any tools used in the assessment

Assessment Process

- A conformant assessment method must be used for the assessment process
- The actual method is not specified in the standard
- The standard places requirements on qualification and competence of assessors
- The standard provides general guidance to assessors and this must be supplemented by undergoing formal training and detailed guidance during initial assessments
- Steps in Assessments
  - ✓ Initiate an assessment (assessment sponsor)
  - ✓ Select assessor and assessment team
  - ✓ Plan the assessment, including processes and organizational unit to be assessed (lead assessor and assessment team)
  - ✓ Pre-assessment briefing
  - ✓ Data collection
  - ✓ Data validation
  - ✓ Process rating
  - ✓ Reporting the assessment result
- An assessor can collect data on a process by various means, including interviews with persons performing the process, collecting documents and quality records, and collecting statistical process data
- The assessor validates this data to ensure it is accurate and completely covers the assessment scope
- The assessor assesses this data (using their expert judgment) against a process's base practices and the capability dimension's generic practices in the process rating step
- Process rating requires some exercising of expert judgment on the part of the assessor and this is the reason that there are requirements on assessor qualifications and competency
- The process rating is then presented as a preliminary finding to the sponsor (and preferably also to the persons assessed) to ensure that they agree that the assessment is accurate
- In a few cases, there may be feedback requiring further assessment before a final process rating is made

Assessment Model

- The process assessment model (PAM) is the detailed model that is used for an actual assessment
- This is an elaboration of the process reference model (PRM) provided by the process lifecycle standards

- The process assessment model is based on the process reference model for systems: ISO/IEC 15288
- The standard allows other models to be used instead, if they meet ISO/IEC 15504's criteria, which include a defined community of interest and meeting the requirements for content (i.e. process purpose, process outcomes and assessment indicators)

Tools Used in the Assessment

- There exist several assessment tools. The simplest comprise paper-based tools that are manually used. In general, they are laid out to incorporate the assessment model indicators, including the base practice indicators and generic practice indicators. Assessors write down the assessment results and notes supporting the assessment judgment
- There are a limited number of computer based tools that present the indicators and allow users to enter the assessment judgment and notes in formatted screens, as well as automate the collated assessment result (i.e. the process attribute ratings) and creating reports

Assessor Qualification and Competency

- For a successful assessment, the assessor must have a suitable level of the relevant skills and experience. These include:
  - ✓ Personal qualities such as communication skills
  - ✓ Relevant education, training and experience
  - ✓ Specific skills for particular categories, e.g. management skills for the management category
  - ✓ ISO/IEC 15504 related training and experience in process capability assessments

Uses of SPICE

- ISO/IEC 15504 can be used in two contexts:
  - ✓ Process improvement
  - ✓ Capability determination (= evaluation of supplier's process capability

Process Improvement

- ISO/IEC 15504 can be used to perform process improvement within a technology organization
- Process improvement is always difficult, and initiatives often fail, so it is important to understand the initial baseline level (process capability level), and to assess the situation after an improvement project
- ISO 15504 provides a standard for assessing the organization's capacity to deliver at each of these stages
- In particular, the reference framework of ISO/IEC 15504 provides a structure for defining objectives, which facilitates specific programs to achieve these objectives
- It specifies requirements for improvement programs and provides guidance on planning and executing improvements

Capability Determination

- An organization considering outsourcing software development needs to have a good understanding of the capability of potential suppliers to deliver
- ISO/IEC 15504 can also be used to inform supplier selection decisions. The ISO/IEC 15504 provides a framework for assessing proposed suppliers, as assessed either by the organization itself, or by an independent assessor
- The ISO/IEC 15504 provides a framework for assessing proposed suppliers, as assessed either by the organization itself, or by an independent assessor
- ISO/IEC 15504 specifies the high level requirements for covering target process profiles
- Target process profiles are particularly important in contexts where the organization (for example, a government department) is required to accept the cheapest qualifying vendor
- This also enables suppliers to identify gaps between their current capability and the level required by a potential customer, and to undertake improvement to achieve the contract requirements (i.e. become qualified)

Acceptance of ISO/IEC TR 15504

- ISO/IEC 15504 is publicly available through National Standards Bodies
- It has the support of the international community
- Thousands of assessments have been performed to date
- Major sectors are leading the pace such as automotive, space and medical systems with industry relevant variants
- Domain-specific models like Automotive SPICE and SPICE 4 SPACE can be derived from it
- There have been many international initiatives to support take-up such as SPICE for small companies
- On the other hand, ISO/IEC 15504 has not yet been as successful as the models developed by Software Engineering Institute
- There are several reasons for this: (next slide)
- ISO/IEC 15504 is not available as free download but must be purchased from the ISO (Automotive SPICE on the other hand can be freely downloaded.) CMM and CMMI are available as free downloads from the SEI website
- The CMMI is actively sponsored by the US Department of Defense
- The CMM was created first, and reached critical 'market' share before ISO 15504 became available
- The CMM has subsequently been replaced by the CMMI, which incorporates many of the ideas of ISO/IEC 15504, but also retains the benefits of the CMM
- Like the CMM, ISO/IEC 15504 was created in a development context, making it difficult to apply in a service management context. But work has started to develop an ITIL-based process reference model that can serve as a basis for a process assessment model
- You can join the SPICE users' group www.spiceusergroup.org
- Every year an international conference is held on SPICE, and it is named SPICE Conference

References

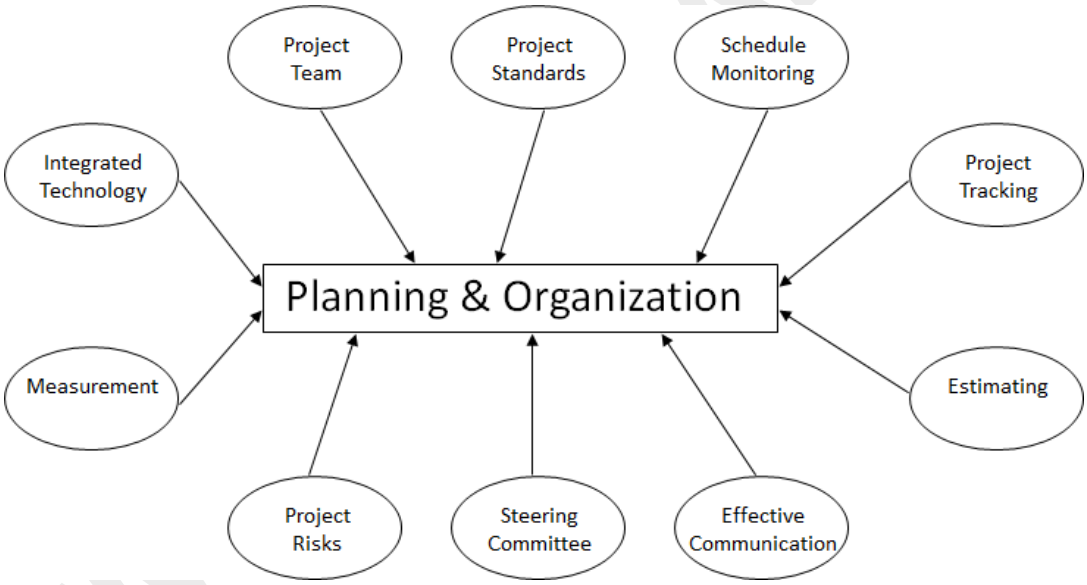
- http://en.wikipedia.org/wiki/ISO/IEC_15504

Lecture 35

Process Assurance

- Process assurance consists of the collective activities carried out while developing a product to ensure that the methods and techniques used are integrated, consistent, and correctly applied
- Emphasis is given to cost, time, technical requirements, testing measurements, and prototyping
- Process assurance involves the interrelationships of several different components
- Depending on how these are managed, they can have a major positive impact on the products
- Once an effective process assurance program is put in place and shown to be beneficial, then emphasis can be placed in making verification and validation strategies effective and in improving the quality of the products
- Successful process assurance is based on planning and organization
- There are several important components of planning and organization

Components of Planning and Organization



1. Project Team

- Project team is the project manager's only means of reaching the project goals
- Selection of team members is a vital step to the success of the project
- Size of the team depends on the size and complexity of the project
- It is important to identify the blend of the technical knowledge and the experience required for the successful completion of the project
- Special attention should be paid to creating team composition that fosters mutual respect among team members and maintains good team morals

2. Project Standards

- Before the project is started, the team should establish standards for activities such as requirements gathering, developing design, and conducting unit tests
- Standards or guidelines should also be established for quality control activities such as walk-throughs, reviews, and inspections
- Many companies follow IEEE Software Engineering standards or they have their own internally developed standards
- The standards should be flexible enough to be applied to large or small projects
- Any deviations from the standards should be approved by the project team and the reason for such deviation should be noted in the minutes of the project meetings

3. Schedule Monitoring

- Stringent deadlines for the project are frequently established by management, end users, a project sponsor, or a client with no regard to the reality of achievement. The project manager is then designated to meet unrealistic expectations of the project completion date
- For this reason, the project start date, milestones, and completion date should be negotiated upfront
- If the unrealistic date is accepted and the project activities are then made to fit within this time frame, the quality of the project activities are then made to fit within this time frame, the quality of the project certainly will suffer
- The key to an "on-time" completion of project lies in the ability to identify the critical path before starting the project
- The critical path of a project is where problems that may affect the overall schedule are faced
- To define the critical path, you should develop systematic work breakdown structures which identify task groupings that consist of tasks that can be combined together, task sequences, and entrance/exit criteria for each task
- The define tasks, follow the guidelines of the system development methodology used by your organization
- In the absence of a development methodology, obtain copies of task lists and task dependencies from other projects and customize them to suit your needs of current project
- Clearly defined work breakdown structures will assist in selecting the correct skilled resources. At the same time, using the breakdown structures also ensures that no activity is forgotten
- The technique of breaking down activities into smaller tasks takes an impossibly complex project and reorganizes it into manageable pieces under the direction of the project manager
- Once you have defined a critical path, review the tasks and schedule with the project team members and other significantly impacted individuals. Since these people are the stakeholders and are affected by the project in one or more of the following ways
    - ✓ Their budget is charged for all or part of the project
    - ✓ The department's resources are used by the project
    - ✓ The department has either existing projects or ongoing projects that are affected by the new project

- Avoid the most common mistake of adding another resource to shorten or meet the schedule
- This usually results in increasing the overhead of the project and reducing the efficiency of the team members since additional communication and guidance are required for the new added resources
- Overtime is often viewed as the alternative to meet the schedule. This can be effective for a short period
- However, longer periods of overtime can exhaust the team, thus reducing the ability to function efficiently
- The cost of overtime should also be considered to finish the project within budget
- It is possible to minimize the risk by how you manage your resources, especially your best programmers
- *Never* put your best programmers on the critical path

4. Project Tracking

- There are several project-tracking tools available on the market which give project progress information and provide a view of the total project schedule at a glance
- Some of these tools have features to do task dependencies and estimations
- These features allow the project manager to evaluate the impact of a delay in the completion of any individual activity within the project
- Such tools are excellent vehicles in giving early warnings for project delays so there are no surprises at the end

5. Estimation

- The project manager must be capable of understanding any assumptions in the project, what is possible, and what is financially desirable
- This will result in a fine-tuning of the schedule estimates, thus creating more realistic estimates
- Realistic estimates allow you to discuss alternative approaches at the start of the project
- Time estimates are not foolproof; activities are many times not clearly understood or defined and not all the situations can be predicted at the time of estimation
- Allow time for resource management and unforeseen events like illness of a team member
- Such contingencies should be viewed as reserved time which may be required for the project. If unused, this time will result in early project completion
- Evaluate the impact of unforeseen events so there are no catastrophes
- Revise estimates at the end of each major phase and if there are any changes in scope, cost, or schedules encountered, notify management and obtain appropriate approvals

6. Effective Communication

- Two-way communication between the management and the project team is a critical interpersonal skill
- An effective project manager must have the skills to listen, observe, and give guidance to the team

- During a time constraint situation, the project manager should be able to delegate responsibilities to the team members but should also be able to give them guidance in order to control the stress level
- The success of team harmony and good rapport between project team members, users, and integrated technology depends on the ability of the project manager to encourage open communication frequently and resolve any conflict thorough informal negotiation

7. Steering Committee

- A committee responsible for defining project policy, reviewing the project milestones, and evaluating the risk factors must be established
- Members of the committee should represent all the impacted areas of the business. They should be knowledgeable enough to make informed technological decisions and should not be just passive observers but should change the course, if need be
- If the features of the project are changed or new features are added halfway through the project, the steering committee is responsible for evaluating the impact on the overall project and authorizing or rejecting the changes or deferring the changes to a later version
- The steering committee should be in charge of the system management tasks such as
  - ✓ Estimating the time that will be required to maintain the system
  - ✓ Deciding on the type of support required from the operations for the running of the system
  - ✓ Deciding when the data will be available and how it will be managed, reported, and used
  - ✓ Forming a Configuration Control Board (CCB) that manages the impact of changes

8. Project Risks

- It is a known fact that systems projects are risky if the project expectations are not controlled or if the outcome of the project was ill defined and misunderstood
- While the risk of the project cannot be eliminated, many companies have opted to identify and address the risk factors upfront
- All the risk factors are fully discussed with the project team, management and end users
- These companies have evaluated the causes of the risks and implemented measurements to the risk element and have designed processes to minimize the risk by giving new projects additional management attention
- Project risks can be assessed in three categories
  - ✓ Business risk
  - ✓ Technology risk
  - ✓ Project size risk
- The risks can be minimized by
  - ✓ Implementing controls from the initiation stage and by ensuring preestablished development standards are followed
  - ✓ Providing project management training
  - ✓ Reducing the scope of the project by incremental development or phased development

- Technical risk is encountered when the project team utilizes a new technology like new hardware or new development methodology for the first time. The technical risks can be controlled by
  - ✓ Appointing a qualified technical project leader
  - ✓ Implementing a strong, independent quality control group to evaluate the progress of the project and project deliverable
  - ✓ Getting additional technical expertise from outside consultants who have expertise and the knowledge to make a difference in the ultimate quality of the project

9. Measurement

- Establishing measurement criteria, against which each phase of the project will be evaluated, is vital
- When the exit criteria is well defined, it is sufficient to evaluate the outcome of each phase against the exit criteria and move forward
- If the outcome of each phase does not meet the performance criteria, the project manager should be able to control the project by evaluating the problems, identifying the deviations, and implementing new processes to address the deviations
- The preestablished quality goals for the project can also serve as criteria against which the project can be measured. In order to meet the goals, processes should be established to
  - ✓ Enable the organization to address customer complaints
  - ✓ Give the organization statistics regarding the types of customer calls
  - ✓ Incorporate reporting and handling of customer problems
  - ✓ Enable management to make staffing decisions based on the number of customer calls
- It is important to know there are certain elements which cannot be controlled by project management, such as selection of the wrong technology (wrong hardware or operating environment), the decision to go ahead with the project without understanding the needs of the business, the underestimation of the learning curve, and a lack of strategic plans for the future which means that the new project becomes obsolete without giving the full return on the investment

10. Integrated Technology

- Strategy for Integrated Technology (IT) should be considered by management in relation to the other business needs
- This will empower the management to react to the operational needs of the business and, at the same time, take an inventory of the current status of various systems, projects, and the quality of technical staff to support any future projects
- The Integrated Technology trends, competitors, and demands of the customers should be visible to the management
- When planning a project, the system architecture with its data groups, structures, processes, and dependencies must be considered
- Parts of the new system that will be interfacing with existing systems should be identified so that the impact can be evaluated

- If the technology is new and not well understood, allowances to incorporate experiments should be made in the overall project plan and schedule

Causes of Failure in Process Assurance

- There are many reasons why process assurance efforts fail. Most of them are related to management failures rather than technical failures
    1. Lack of Management Support
    2. Lack of User Involvement
    3. Lack of Project Leadership
    4. Lack of Measures of Success

1. Lack of Management Support

- Management very often assigns a project and then forgets about it. At other times management assigns it, feeling that it is important, but fails to allocate adequate resources for the project to be done correctly
- This lack of project sponsorship leads to serious morale problems and lost resources
- The project sponsor is usually a member of the senior management team on whose behalf the project is initiated
- This person is responsible for researching and resolving regulatory issues, obtaining project funding, and managing the business implication for the project. Since the project sponsor is responsible for acting as the main advocate of the project and ensuring that the project is implemented as agreed upon, lack of support from such an individual is extremely serious to the process assurance activity

2. Lack of User Involvement

- When the end users of the project are not involved in the design and development phases of a project, lots of assumptions are made regarding the requirements
- This is especially true where the requirements are ambiguous and user approval for the requirements is not obtained
- The situation can be corrected by assigning a user liaison who is responsible for getting support for the project from within the end-user organization, providing detailed user requirements, providing feedback to the project team, managing user expectations, and identifying user acceptance criteria

3. Lack of Project Leadership

- The single most important cause of failure is due to lack of project leadership. There are many consequences of not having a good leader such as
    - ✓ IT and development staff members not understanding the business needs of the project
    - ✓ The project's scope and goals not being fully defined and the project team not being sure of the project's ultimate results. Often in this case, the solution is derived before the problem is identified and coding starts before any clarity of the project goals is

obtained. The schedule is set before the full extent of the project is understood, thereby setting unrealistic schedule extimates

- ✓ Performance not being controlled and checkpoint meetings not being conducted to monitor the progress
- ✓ Improper selection of project team members and unavailability of potential team members during the project, resulting in the project manager being overburdened with various tasks

- Ensuring that the project is in the hands of a good project manager and that the project is progressing according to the preestablished schedule is vital for the success of the project
- Selecting an individual who is goal-oriented, possessing the right traits with a positive attitude and competence for managing the project is a challenge
- However, even if an excellent project manager is allocated, the success or failure of a project may not have anything to do with the individual who is assigned to manage the project because other factors still influence the project's outcome
- There are three attributes that must be treated equally:
  - ✓ The individual assigned to the project must have the responsibility to manage it (if an individual assigned to a project has no authority to make any decisions on the project, he or she will have very little control over the project)
  - ✓ The authority of that manager must be equal to the responsibility
  - ✓ The manager must have access to the necessary resources for successfully completing the project
- There are projects whose outcome depends heavily on the project manager's performance and motivation, for example, information systems projects often have different characteristics and require a different management style than real-time systems
- For this type of project, it is important to select a project manager who is most suitably matched in skills and personality to handle the requirements of the project and also the team members who are assigned to the project
- Although technical knowledge is necessary, the knowledge of human interactions and relationships is more important since it allows the project manager to deal with unexpected and adverse situations
- The project manager should be flexible, responsive, and effective in coordinating day-to-day activities
- For some reason, if the project schedule is sliding or if there are situations that need attention and decision making, the project manager should be competent enough to deal with them
- Careful planning and early process control can avoid pitfalls at the later stages of the project
- Changing work environment and technology in the middle of the project can pose problems
- In such situations, the project manager must be in a position to make effective decisions and implement them
- The project manager should also be the content expert, thus allowing him or her to develop quick solutions and implement decisions and changes that are best for the success of the project
- The project manager should be credible and respected by other team members to be able to orchestrate the strategies for the project and carry out negotiations

- When under time constraints, the project manager should possess the skills to evaluate the schedule, risks, and resources, and arrive at the conclusions that would be best for the project
- If the goals of the project are unclear, the project manager should be able to work with the team members to go through the entire scope of the project, evaluate the benefits, and finalize the goals

4. Lack of Measures of Success

- In addition to the causes of failures listed above, another major contributor to failure is when you define the project success based on development efficiency measures only, such as on time and within budget
- Effectiveness measures for the project are completely ignored, such as technical performance and quality
- To evaluate the success of the project, objective measurements should be developed, for example, the project is completed on time, within budget, and with all the required features
- Successful companies enforce standards which require consensus to evaluate effectiveness and success criteria of the project
- Each element of the success criteria must be defined and agreed upon prior to starting the project

References

- Inroads to Software Quality by Alka Jarvis and Vern Crandall, Chapter 1

Lecture 36

People Capability Maturity Model – 1

- Do you ask yourself these questions?
  - ✓ Do I get full advantage from my development capability?
  - ✓ Can I attract, train, deploy, and retain the people I need to develop software in today's competitive market?
- Then People CMM can help you
- The guidance provided by the People CMM helps you manage and develop your organization's workforce
- The People Capability Maturity Model is a tool that helps you successfully address the critical people issues in your organization
- The People CMM employs the process maturity framework of the highly successful Capability Maturity Model for Software (SW-CMM) as a foundation for a model of best practices for managing and developing an organization's workforce
- Based on the best current practices in fields such as human resources, knowledge management, and organizational development, the People CMM guides organizations in improving their processes for managing and developing their workforce
- The People CMM helps organizations characterize the maturity of their workforce practices, establish a program of continuous workforce development, set priorities for improvement actions, integrate workforce development with process improvement, and establish a culture of excellence
- Since its release in 1995, thousands of copies of the People CMM have been distributed, and it is used world-wide by organizations, small and large, such as IBM, Boeing, BAE Systems, Tata Consultancy Services, Ericsson, Lockheed Martin and QAI (India) Ltd
- The People CMM evolutionary improvement path guides movement from an ad hoc approach; to managing the workforce; to a mature, disciplined development of the knowledge, skills, and motivation of the people that fuels enhanced business performance
- The People CMM helps organizations to do the following:
  - ✓ Evaluate how well you utilize your human resources
  - ✓ Set priorities for improving the competence of its workforce
  - ✓ Integrate competence growth with process improvement
  - ✓ Establish a culture of workforce excellence
- The People CMM's primary objective is to improve the capability of the workforce.
- Workforce capability can be defined as the level of knowledge, skills, and process abilities available for performing an organization's business activities
- Workforce capability indicates an organization's
  - ✓ Readiness for performing its critical business activities,
  - ✓ Likely results from performing these business activities, and
  - ✓ Potential for benefiting from investments in process improvement or advanced technology

- In order to measure and improve capability, the workforce in most organizations must be divided into its constituent workforce competencies
- Each workforce competency represents a unique integration of knowledge, skills, and process abilities acquired through specialized education or work experience
- Strategically, an organization wants to design its workforce to include the various workforce competencies required to perform the business activities underlying its core competencies
- Each of these workforce competencies can be characterized by its capability—the profile of knowledge, skills, and process abilities available to the organization in that competency
- The People CMM describes an evolutionary improvement path from ad hoc, inconsistently performed workforce practices, to a mature infrastructure of practices for continuously elevating workforce capability
- The philosophy implicit in the People CMM can be summarized in the ten principles (discussed next)

Principles of People CMM – 1

- In mature organizations, workforce capability is directly related to business performance
- Workforce capability is a competitive issue and a source of strategic advantage
- Workforce capability must be defined in relation to the organization's strategic business objectives
- Knowledge-intense work shifts the focus from job elements to workforce competencies
- Capability can be measured and improved at multiple levels of the organization, including individuals, workgroups, workforce competencies, and the organization
- An organization should invest in improving the capability of those workforce competencies that are critical to its core competency as a business
- Operational management is responsible for the capability of the workforce
- The improvement of workforce capability can be pursued as a process composed from proven practices and procedures
- The organization is responsible for providing improvement opportunities, and individuals are responsible for taking advantage of them
- Because technologies and organizational forms evolve rapidly, organizations must continually evolve their workforce practices and develop new workforce competencies
- The People CMM is an evolutionary framework that guides organizations in selecting high-priority improvement actions based on the current maturity of their workforce practices
- The benefit of the People CMM is in narrowing the scope of improvement activities to those vital few practices that provide the next foundational layer for developing an organization's workforce
- The practices required to attract, develop, and retain outstanding talent have been understood for decades. In his acclaimed book, *The Human Equation, Jeffrey Pfeffer of the* Stanford Graduate School of Business identified seven principles of workforce management that distinguished the companies that had exhibited the largest percentage stock market returns over the past quarter century

Seven Principle of Workforce Management

1. Employment security
2. Selective hiring of new personnel
3. Self-managed teams and decentralization of decision making
4. Comparatively high compensation contingent on organizational performance
5. Extensive training
6. Reduced status distinctions and barriers
7. Extensive sharing of financial and performance information

- These principles characterize organizations that no longer expect employees merely to execute orders, but rather to act as independent centers of intelligent action coordinated toward a common purpose
- Deep technical and business knowledge is required to make rapid decisions that are not only correct, but also consistent with decisions made by colleagues
- The development and coordination of a modern workforce requires an integrated set of practices that address attracting, developing, organizing, motivating, and retaining outstanding individuals

Critical Success Factors for Managing Human Capital Strategically

Four Human Capital Cornerstones

- Leadership
  - ✓ Commitment to Human Capital Management
  - ✓ Role of the Human Capital Function
- Strategic Human Capital Planning
  - ✓ Integration and Alignment
  - ✓ Data-Driven Human Capital Decisions
- Acquiring, Developing, and Retaining Talent
  - ✓ Targeted Investments in People
  - ✓ Human Capital Approaches Tailored to Meet Organizational Needs
- Results-Oriented Organizational Cultures
  - ✓ Empowerment and Inclusiveness
  - ✓ Unit and Individual Performance Linked to Organizational Goals

Process Maturity Framework

- Humpherey designed the process maturity framework to enable an organization to achieve a state of continuous process improvement in five stages
- Because of this staging, the process maturity framework is more than a process standard comprising a list of best practices
- Rather, it integrates improved practices into a staged model that guides an organization through a series of cultural transformations, each of which supports the deployment of more sophisticated and mature processes
- Over a dozen years of experience with the People CMM demonstrates that this process maturity framework is also applicable to the management and improvement of workforce practices within an organization

First Level of Maturity

- At the first level of maturity, the Initial Level, an organization has no consistent way of performing its work. Since most work processes are ad hoc, they are constantly reinvented on each project, and frequently appear chaotic
- Without well-understood ways of conducting their work, managers have no reliable basis for estimating the effort required to complete a project
- In a rush to meet overly aggressive deadlines, the project staff begins cutting corners on sound practices and making mistakes that are not detected until it is much more time consuming and costly to remove them than to have prevented them
- As a result, projects lose control of their schedule, costs, and product quality. Since work is chronically overcommitted in low-maturity organizations, their results depend largely on the skills of exceptional individuals and on excessive overtime

Second Level of Maturity

- At the second level of maturity, organizations must establish a foundation on which they can deploy common processes across the organization
- Before being able to successfully implement many advanced practices, management must first establish a stable environment in which to perform professional work
- They must ensure that people are not constantly rushing about pell-mell, cutting corners, making mistakes due to hasty work, and fighting the fires that characterize overcommitted organizations
- The primary objective of a Maturity Level 2 environment is to enable people to repeat practices they have used successfully

Third Level of Maturity

- At the third level of maturity, the organization identifies its best practices and integrates them into a common process
- Once people are able to perform their work using practices they have found to work, the organization has the ability to identify which practices work best in its unique environment
- These practices are documented and integrated into a common process in which the entire organization is then trained
- Measures of the critical practices in this process are defined and collected into a repository for analysis

Fourth Level of Maturity

- At the fourth level of maturity, the organization begins managing its processes through the data that describes its performance
- The performance of the organization's critical processes is characterized statistically so that the historical performance of the process can be used to predict and manage its future performance

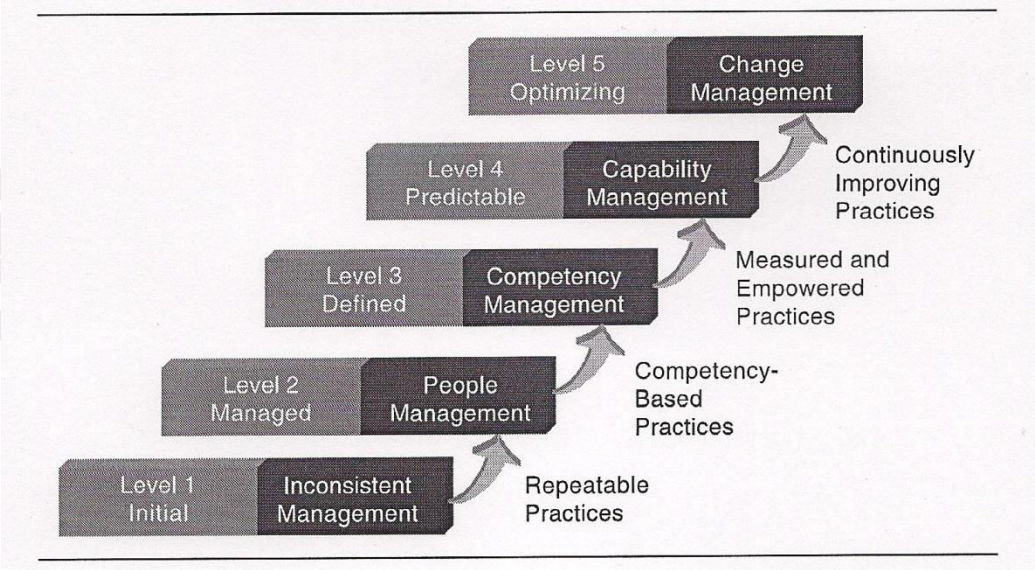
Fifth Level of Maturity

- At the fifth and highest level of maturity, the organization uses its profound, quantitative knowledge to make continuous improvements in its processes. Based on its data, the organization can identify which processes can benefit most from improvement actions
- In the abstract, the maturity framework builds an environment in which
  - ✓ Practices can be repeated
  - ✓ Best practices can be transferred rapidly across groups
  - ✓ Variations in performing best practices are reduced
  - ✓ Practices are continuously improved to enhance their capability
- The People CMM has been designed to increase the capability of the workforce, just as the SW-CMM increased the capability of the organization's software development processes
- The People CMM has been applied in numerous industries and types and sizes of organizations around the globe
- Industries using the People CMM range from high-tech and information technology companies, to pharmaceuticals and hospitality, to construction and government agencies

Types of organizations around the world using the People CMM

- Business Process Outsourcing
- Information Technology
- Hospitality Consulting
- Construction
- Defense Contractors
- Insurance Pharmaceuticals
- Government Agencies/Defense Agencies
- Energy/Utilities
- Software Development
- Banking/Financial Services Management
- Information Systems

Maturity Levels of the People CMM



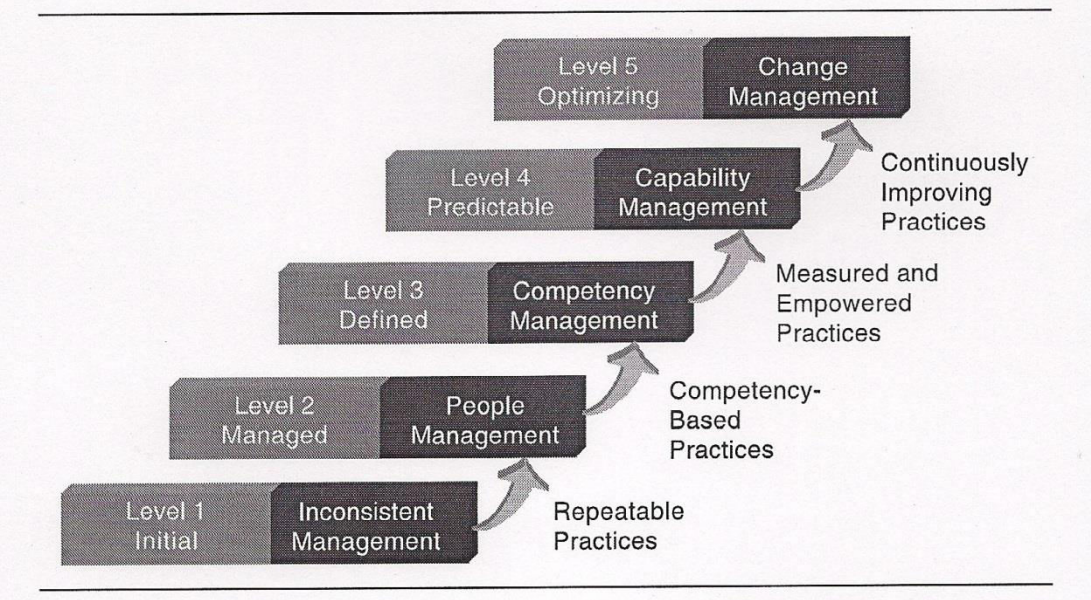
- The People CMM applies the principles of the process maturity framework to the domain of workforce practices
- Each of the People CMM's five maturity levels represents a different level of organizational capability for managing and developing the workforce
- Each maturity level provides a layer in the foundation for continuous improvement and equips the organization with increasingly powerful tools for developing the capability of its workforce
- When institutionalized and performed with appropriate regularity, these workforce practices create new capabilities within the organization for managing and developing its workforce

References

- People CMM: A Framework for Human Capital Management, 2nd Edition, by Bill Curtis, William E. Hefley, and Sally A. Miller, Addison-Wesley, 2010 (Chapter 1)

Lecture 37
People CMM – 2

Maturity Levels of the People CMM



The Initial Level: Maturity Level 1

- Organizations at the Initial Level of maturity usually have difficulty retaining talented individuals
- Even though many low-maturity organizations complain about a talent shortage, the inconsistency of their actions belies whether they actually believe it
- Low-maturity organizations are poorly equipped to respond to talent shortages with anything other than slogans and exhortations
- Despite the importance of talent, workforce practices in low-maturity organizations are often ad hoc and inconsistent
- In some areas, the organization has not defined workforce practices, and in other areas, it has not trained responsible individuals to perform the practices that exist
- Organizations at the Initial Level typically exhibit four characteristics:
  - ✓ Inconsistency in performing practices
  - ✓ Displacement of responsibility
  - ✓ Ritualistic practices
  - ✓ An emotionally detached workforce
- Generally managers and supervisors in low-maturity organizations are ill prepared to perform their workforce responsibilities
- Their management training is sparse
- Low-maturity organizations implicitly assume that management skill either is innate or is acquired by observing other managers
- However, if managers are inconsistent in managing their people, nascent managers will be learning from inconsistent role models

- Management capability should ultimately be defined as a competency, just like other critical skill sets that are required by the organization
- Since low-maturity organizations rarely clarify the responsibilities of managers, inconsistencies are to be expected
- Consequently, the way people are treated depends largely on personal orientation, experience, and the individual "people skills" of their managers, supervisors, or team leaders
- Although some managers perform their workforce responsibilities diligently, others perform some workforce activities with little forethought and ignore other responsibilities altogether.
- Studies have consistently shown that one of the major causes for voluntary turnover is related to individuals' relationships with their managers or supervisors
- The first step in changing this state of affairs is to get managers to take responsibility for the capability and development of those who report to them

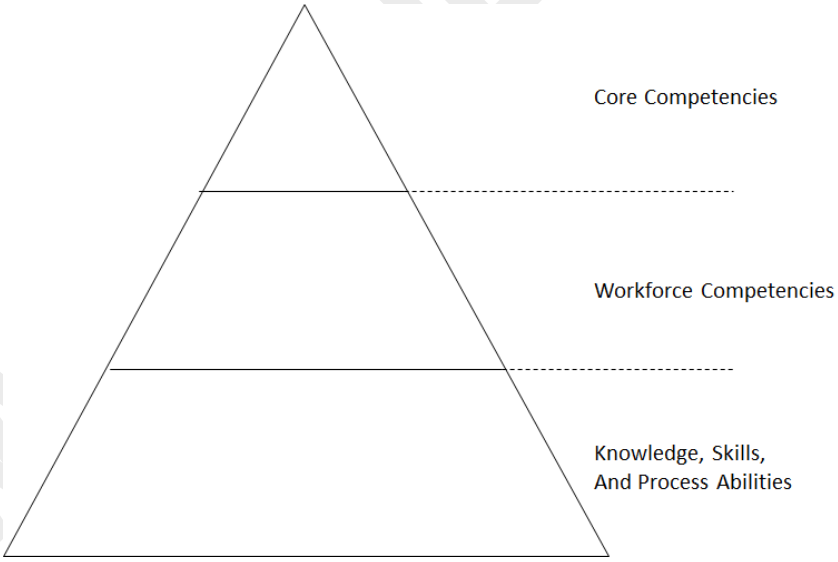
The Managed Level: Maturity Level 2

- The first step toward improving the capability of the workforce is to get managers to take workforce activities as high-priority responsibilities of their job
- They must accept personal responsibility for the performance and development of those who perform the unit's work
- The practices implemented at Maturity Level 2 focus a manager's attention on unit level issues such as staffing, coordinating commitments, providing resources, managing performance, developing skills, and making compensation decisions
- Building a solid foundation of workforce practices in each unit provides the bedrock on which more sophisticated workforce practices can be implemented at higher levels of maturity
- Maturity Level 2 focuses on establishing basic practices in units that address immediate problems and prepare managers to implement more sophisticated practices at higher levels
- In a Maturity Level 2 organization, managers are vigilant for problems that hinder performance in their units
- These problems are:
  - ✓ Work overload
  - ✓ Environmental distractions
  - ✓ Unclear performance objectives or feedback
  - ✓ Lack of relevant knowledge or skill
  - ✓ Poor communication
  - ✓ Low morale
- The effort to ensure that workforce practices are performed in each unit begins when executive management commits the organization to continuously improve the knowledge, skills, motivation, and performance of its workforce
- Executive management manifests these commitments in policies and provides the resources needed to support unit-level implementation of basic workforce practices
- Through policies and accountability, executive management communicates that managers are to accept personal responsibility for ensuring that workforce practices are implemented effectively in their units

- In applying the People CMM, it is important to distinguish between management and managers
- There are responsibilities that need to be managed and there are people called managers, but there is no required one-to-one mapping between them
- At Maturity Level 2, units become stable environments for performing work. Units are able to balance their commitments with available resources
- At Maturity Level 2, an organization's capability for performing work is best characterized by the capability of units to meet commitments
- This capability is achieved by ensuring that people have the skills needed to perform their assigned work and that performance is regularly discussed to identify actions that can improve it
- Measurements of status and performance of these workforce activities provide management with a means of monitoring and ensuring appropriate performance of workforce practices
- One of the first benefits organizations experience when they implement improvements guided by the People CMM is a reduction in voluntary turnover
- At Maturity Level 2, the People CMM addresses one of the most frequent causes of turnover—poor relations with the immediate supervisor
- Process Areas of Level 2
    - ✓ Staffing
        - The purpose of Staffing is to establish a formal process by which committed work is matched to unit resources and qualified individuals are recruited, selected, and transitioned into assignments
    - ✓ Communication and Coordination
        - The purpose of Communication and Coordination is to establish timely communication throughout the organization and to ensure that the workforce has the skills to share information and coordinate activities efficiently
    - ✓ Work Environment
        - The purpose of Work Environment is to establish and maintain physical working conditions and to provide resources that allow individuals and workgroups to perform their tasks efficiently without unnecessary distractions
    - ✓ Performance Management
        - The purpose of Performance Management is to establish objectives related to committed work against which unit and individual performance can be measured, to discuss performance against these objectives, and to continuously enhance performance
    - ✓ Training and Development
        - The purpose of Training and Development is to ensure that all individuals have the skills required to perform their assignments and are provided relevant development opportunities
    - ✓ Compensation
        - The purpose of Compensation is to provide all individuals with remuneration and benefits based on their contribution and value to the organization

The Defined Level: The Level 3

- Once a foundation of basic workforce practices has been established in the units, the next step is for the organization to develop an organization-wide infrastructure building on these practices that ties the capability of the workforce to strategic business objectives
- The primary objective of the Defined Level is to help an organization gain a competitive advantage by developing the various competencies that must be combined in its workforce to accomplish its business activities
- These workforce competencies represent the critical pillars that support the strategic business plan; their absence poses a severe risk to strategic business objectives
- In tying workforce competencies to current and future business objectives, the improved workforce practices implemented at Maturity Level 3 become critical enablers of business strategy
- The concept of workforce competencies implemented in the People CMM differs from the concept of "core competency"
- Core competency refers to an organization's combination of technology and production skills that create its products and services and provide its competitive advantage in the marketplace
- In the People CMM, workforce competencies reside one level of abstraction below an organization's core competency
- Hierarchy of Competency Abstractions



- Workforce Competencies
  - ✓ Each workforce competency represents a distinct integration of the knowledge, skills, and process abilities required to perform some of the business activities that contribute to an organization's core competency
  - ✓ The range of workforce competencies an organization must integrate depends on the breadth and type of business activities that comprise its core competencies

- ✓ Therefore, these workforce competencies are a strategic underpinning (foundation) of the organization's core competencies
- ✓ By defining process abilities as a component of a workforce competency, the People CMM becomes linked with the process frameworks established in other CMMs and with other process-based methods, such as business process reengineering
- ✓ A process ability is demonstrated by performing the competency-based processes appropriate for an individual's required workforce competency
- ✓ To define the process abilities incorporated in each workforce competency, the organization defines the competency-based processes that an individual would be expected to perform in accomplishing his or her committed work
- ✓ Within a workforce competency, a competency-based process defines how individuals apply their knowledge, perform their skills, and apply their process abilities in the context of the organization's defined work processes
- ✓ At Maturity Level 3, the organization builds an organization-wide framework of workforce competencies that establishes the architecture of the organization's workforce
- ✓ Each workforce competency is an element of the workforce architecture, and dependencies among competency-based processes describe how these architectural elements interact
- ✓ The architecture of the organization's workforce must evolve as business conditions and technologies change
- ✓ Because workforce competencies are strategic, the organization must develop strategic workforce plans for ensuring the required capability in each of its current or anticipated workforce competencies
- ✓ When the processes to be performed by each workforce competency are defined, the organization has a new foundation for developing workgroups
- ✓ Competency-based processes form a basis for defining workgroup roles and operating processes
- ✓ Workgroups can now organize themselves by tailoring and applying standard competency-based processes
- ✓ A common organizational culture typically develops as the organization achieves the Defined Level
- ✓ This culture is best described as one of professionalism, since it is built from common understanding of the knowledge and skills that need to be developed to achieve superior levels of performance and a definition of the competency-based processes that such individuals perform
- ✓ Since these workforce competencies are strategic to the business, the organization reinforces their importance by developing and rewarding them
- ✓ As a result, the entire workforce begins to share responsibility for developing increasing levels of capability in the organization's workforce competencies

- ✓ The workforce practices that were implemented at Maturity Level 2 are now standardized and adapted to encourage and reward growth in the organization's workforce competencies

- Process Areas of Level 3
  - ✓ Competency Analysis
    - The purpose of Competency Analysis is to identify the knowledge, skills, and process abilities required to perform the organization's business activities so that they may be developed and used as a basis for workforce practices
  - ✓ Workforce Planning
    - The purpose of Workforce Planning is to coordinate workforce activities with current and future business needs at both the organizational and unit levels
  - ✓ Competency Development
    - The purpose of Competency Development is to enhance constantly the capability of the workforce to perform its assigned tasks and responsibilities
  - ✓ Career Development
    - The purpose of Career Development is to ensure that individuals are provided opportunities to develop workforce competencies that enable them to achieve career objectives
  - ✓ Competency-Based Practices
    - The purpose of Competency-Based Practices is to ensure that all workforce practices are based in part on developing the competencies of the workforce
  - ✓ Workgroup Development
    - The purpose of Workgroup Development is to organize work around competency-based process abilities
  - ✓ Participatory Culture
    - The purpose of a Participatory Culture is to enable the workforce's full capability for making decisions that affect the performance of business activities

The Predictable Level: Maturity Level 4

- At the Predictable Level, the organization manages and exploits the capability created by its framework of workforce competencies
- This framework is sustained through formal mentoring activities
- The organization is now able to manage its capability and performance quantitatively
- The organization is able to predict its capability for performing work because it can quantify the capability of its workforce and of the competency based processes they use in performing their assignments
- There are at least three ways in which the framework of workforce competencies enables the organization to more fully use the capabilities of its workforce
- First, when competent people perform their assignments using proven competency-based processes, management trusts the results they produce
- This trust enables the organization to preserve the results of performing competency-based processes and develop them as organizational assets to be reused by others

- In essence, people trust the asset because they trust the methods through which it was produced
- When these assets are created and used effectively, learning spreads rapidly through the organization and productivity rises when reuse replaces redevelopment
- Second, this trust also gives managers the confidence they need to empower workgroups
- Managers will transfer responsibility and authority for committed work into workgroups only if they believe the members of the workgroup are competent to perform the work and use processes that have been proven effective
- In achieving Maturity Level 4, management senses less risk in empowering workgroups and is willing to delegate increasingly greater levels of authority for managing day-to-day operations and for performing some of their own workforce practices
- Increasingly free of managing operational details, managers at Maturity Level 4 are able to turn their attention to more strategic issues
- Third, when members of each workforce competency community have mastered their competency-based processes, the organization is able to integrate different competency-based processes into a single multidisciplinary process
- An example would be the integration of software and hardware design processes into a single product design process in which different competency-based processes are interwoven at every point where they share a potential dependency
- Such multidisciplinary processes have proven to accelerate business results
- In addition to exploiting the possibilities enabled by the competency framework, the organization begins to manage its capability quantitatively
- Within each unit or workgroup, the performance of competency-based processes most critical for accomplishing business objectives is measured
- These measures are used to establish process performance baselines that can be used to manage competency-based processes and assess the need for corrective action
- The combined availability of workforce capability baselines and process capability baselines for competency-based processes enables both unit and organizational performance to become more predictable
- These data allow management to make more accurate predictions about performance and better decisions about tradeoffs involving workforce capability or process performance issues
- The quantitative management capabilities implemented at Maturity Level 4 provide management with better input for strategic decisions, while encouraging delegation of operational details to people close to the processes
- Process Areas of Level 4
    - ✓ Competency Integration
        - The purpose of Competency Integration is to improve the efficiency and agility of interdependent work by integrating the process abilities of different workforce competencies

- ✓ Empowered Workgroups
  - The purpose of Empowered Workgroups is to invest workgroups with the responsibility and authority to determine how to conduct their business activities most effectively
- ✓ Competency-Based Assets
  - The purpose of Competency-Based Assets is to capture the knowledge, experience, and artifacts developed in performing competency-based processes for use in enhancing capability and performance
- ✓ Quantitative Performance Management
  - The purpose of Quantitative Performance Management is to predict and manage the capability of competency-based processes for achieving measurable performance objectives
- ✓ Organizational Capability Management
  - The purpose of Organizational Capability Management is to quantify and manage the capability of the workforce and of the critical competency-based processes it performs
- ✓ Mentoring
  - The purpose of Mentoring is to transfer the lessons of greater experience in a workforce competency to improve the capability of other individuals or workgroups

The Optimizing Level: Maturity Level 5

- At the Optimizing Level, the entire organization is focused on continual improvement
- These improvements are made to the capability of individuals and workgroups, to the performance of competency-based processes, and to workforce practices and activities
- The organization uses the results of the quantitative management activities established at Maturity Level 4 to guide improvements at Maturity Level 5
- Although several individuals may be performing identical competency-based processes, they frequently exhibit individual differences in the methods and work styles they use to perform their assignments
- At Maturity Level 5, individuals are encouraged to make continuous improvements to their personal work processes by analyzing their work and making necessary process enhancements. So is true for workgroups
- Although individuals and workgroups continually improve their performance, the organization must be vigilant to ensure that performance at all levels remains aligned with organizational objectives
- At Maturity Level 5, the process performance data collected across the organization is evaluated to detect instances of misalignment
- Further, the impact of workforce practices and activities is evaluated to ensure that they encourage rather than discourage alignment
- Corrective action is taken to realign performance objectives and results when necessary

- Process Areas of Level 5
  - ✓ Continuous Capability Improvement
    - The purpose of Continuous Capability Improvement is to provide a foundation for individuals and workgroups to continuously improve their capability for performing competency-based processes
  - ✓ Organizational Performance Alignment
    - The purpose of Organizational Performance Alignment is to enhance the alignment of performance results across individuals, workgroups, and units with organizational performance and business objectives
  - ✓ Continuous Workforce Innovation
    - The purpose of Continuous Workforce Innovation is to identify and evaluate improved or innovative workforce practices and technologies, and implement the most promising ones throughout the organization

References

- People CMM: A Framework for Human Capital Management, 2nd Edition, by Bill Curtis, William E. Hefley, and Sally A. Miller, Addison-Wesley, 2010 (Chapter 2 and 3.2)

Lecture 38
Introduction to Measurements

Software measurement

- Software measurement, once an obscure and esoteric specialty has become essential to good software engineering
- Many of the best software developers measure characteristics of the software to get some sense of whether the requirements are consistent and complete, whether the code is ready to be tested
- Effective project managers measure attributes of process and product to be able to tell when the software will be ready for delivery and whether the budget will be exceeded
- Informed customers measure aspects of the final product to determine if it meets the requirements and is of sufficient quality
- Measurement lies at the heart of many systems that govern our lives
- Economic measurements determine price and pay increases
- Measurements in radar systems enable us to detect aircraft when direct vision is obscured
- Medical system measurements enable doctors to diagnose specific illnesses
- Measurements in atmospheric systems are the basis for whether prediction
- Without measurement, technology cannot function. But measurement is not solely the domain of professional technologists. Each of *us* uses it in everyday life
- Examples of Measurements
    - ✓ Prices acts as a measure of value of an item in a shop
    - ✓ We use height, weight, and size measurements to ensure that our clothing will fit properly
    - ✓ When making a journey, we calculate distance, choose our route, measure our speed, and predict when we will arrive at our destination (and perhaps when we need to refuel)
    - ✓ So measurement helps us to understand our world, interact with our surroundings and improve our lives
    - ✓ These examples present a picture of the variety in how we use measurement
    - ✓ There is a common thread running through each of the described activities: in every case, some aspect of a thing is assigned a descriptor that allows us to compare it with others
    - ✓ In a shop, we can compare the price of one item with another
    - ✓ In a clothing store, we contrast sizes
    - ✓ And on our journey, we compare distance travelled to distance remaining
    - ✓ The rules for assignment and comparisons are not explicit in the examples, but it is clear that we make our comparisons and calculations according to a well-defined set of rules
    - ✓ So, we can define measurement formally

Measurement

- Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules
- Thus, measurement captures information about attributes of entities
- An entity is an object or an event in the real world. For example;
  - ✓ A person or a room
  - ✓ A journey or the testing phase of software project
- We want to describe the entity by identifying characteristics that are important to us in distinguishing one entity from another
- An attribute is a feature or property of an entity. Typical attributes include the area or color (of a room), the cost (of a journey), or the elapsed time (of the testing phase)
- We often talk about entities and their attributes interchangeably
  - ✓ 'It is cold today' (air temperature is cold today)
  - ✓ 'Usman is taller than Ali' (height is greater than)
- Such loose terminology is acceptable for everyday speech, but it is incorrect and unsuitable for scientific endeavors. Thus, it wrong to say that we measure things or that we measure attributes; in fact, we measure attributes of things
- It is ambiguous to say that we "measure a room", since we can measure its length, area, or temperature
- It is likewise ambiguous to say that we "measure the temperature," since we measure the temperature of a specific geographical location under specific conditions
- In other words, what is commonplace in common speech is unacceptable for engineers and scientists
- We can describe entities by using attributes, we often define the attributes using numbers or symbols
- Price can be designated as a number of Rupees or dollars, etc.
- Clothing size may be "small", "medium", or "large"
- These numbers and symbols are abstractions that we use to reflect our perceptions of the real world
- For example, in defining numbers and symbols, we try to preserve certain relationships that we see among entities. Thus, someone who is six feet in height is taller than someone who is five feet in height
- Similarly, a "medium" T-shirt is smaller than a "large" T-shirt
- This number or symbol can be very useful and important
- We can make judgments about entities solely by knowing and analyzing their attributes
- Example of seven feet man, if he ever come to meet us in our office. He would need to stoop (bend forward and down from the waist or the middle of the back)
- Measurement is a process whose definition is far from clear-cut
- Many different authoritative views lead to different interpretations about what constitutes measurement
- To understand what measurement is, we must ask a host of questions that are difficult to answer. For example:

1. We have noted that color is an attribute of a room. In a room with blue walls, is "blue" a measure of the color the room?
2. The height of a person is commonly understood attribute that can be measured. But what about other attributes of people, such as intelligence? Is intelligence adequately measured by an IQ test score. Similarly, a soft drink can be measured in terms of caffeine content ("proof is in the eating of the pudding"), but can a soft drink quality be measured using the ratings of experts?
3. The accuracy of a measure depends on the measuring instrument as well as on the definition of the measurement. For example, length can be measured accurately as long as the ruler is accurate and used properly. But some measures are not likely to be accurate, either because the measurement is imprecise or because it depends on the judgment of the person doing the measuring. For instance, proposed measured of human intelligence or soft drink quality appear to have likely error margins. Is this a reason to reject them as bona fide measurements?
4. Even when the measuring devices are reliable and used properly, there is margin of error in measuring the best understood physical attributes. For example, we can obtain vastly different measures for a person's height, depending on whether we make allowances for the shoes being worn or the standing posture. So how do we decide which error margins are acceptable and which are not?
5. We can measure height in terms of meters, inches or feet. These different scales measure the same attribute. But we can also measure height in terms of miles and kilometers – appropriate for measuring the height of a satellite above earth, but not for measuring the height of a person. When is a scale acceptable for the purpose to which it is put?
6. Once we obtain measurements, we want to analyze them and draw conclusions about entities from which they were derived. What kind of manipulations can we apply to the results of measurement? For example, why is it acceptable to say that Fred is twice as tall as Joe, but not acceptable to say that it is twice as hot today as it was yesterday? And why is it meaningful to calculate the mean of a set of heights (to say, for example, that the average height of a London building is 200 meters), but not the mean of the football jersey numbers of a team?

- Examples
  - ✓ Kal 'moosla dhar' barish huee
  - ✓ Kal mein 'moosla dhar' bhaga
- It is important to remember that the concepts of time, temperature, and speed, once un-measurable by primitive peoples, are now not only commonplace but also easily measured by almost everyone; these measurements have become part of the fabric of our existence
- To improve the rigor of measurement in software engineering, we need not restrict the type or range of measurements we can make
- Indeed, measuring the un-measurable should improve our understanding of particular entities and attributes, making software engineering as powerful as other engineering disciplines

- Some software engineers may continue to claim that important software attributes like dependability, quality, usability, and maintainability are simply not quantifiable, we prefer to try to use measurement to advance our understanding of these attributes
- We should note that there are two kinds of quantification: measurement and calculation
- Measurement is a direct quantification of something, as in measuring the height of a tree or weight of a shipment of bricks
- Calculation is indirect, where we take measurements and combine them into a quantified item that reflects some attribute whose value we are trying to understand.
    - ✓ Example: valuation of a house/shop for tax purposes
- Example:
    - ✓ In the decathlon athletics event, we measure the time to run various distances as well as the length covered in various jumping activities. These measures are subsequently combined into an overall score, computed using a complex weighting scheme that reflects the importance of each component measure
- Software engineering activities include managing, costing, planning, modeling, analyzing, specifying, designing, implementing, testing, and maintaining
- We need measurements in all these areas
- Underpinning the scientific process is measurement: measuring variables to differentiate cases, measuring the changes in behavior, and measuring the causes and effects

Neglect of Measurements in SE

- Measurement has been considered a luxury in software engineering
- We fail to set measurable targets for our software products. For example, we promise that the product will be user-friendly, reliable and maintainable without specifying clearly and objectively what these terms mean
- As a result, when the project is complete, we cannot tell if we have met our goals
- Projects without clear goals will not achieve their goals clearly
- We fail to understand and quantify the component costs of software projects. For example, most projects cannot differentiate the cost of design from the cost of coding and testing
- Since excessive cost is a frequent complaint from many of our customers, we cannot hope to control costs if we are not measuring the relative components of cost
- We do not quantify or predict the quality of the products we produce
- Thus, we cannot tell a potential user how reliable a product will be in terms of likelihood of failure in a given period of use, or how much work will be needed to port the product to a different machine environment
- We allow anecdotal evidence to convince us to try yet another revolutionary new development technology, without doing a carefully controlled study to determine if the technology is efficient and effective

Objectives for Software Measurements

- Even when a project is not in trouble, measurement is not only useful but necessary
- After all, how can you tell if your project is healthy if you have no measures of its health?

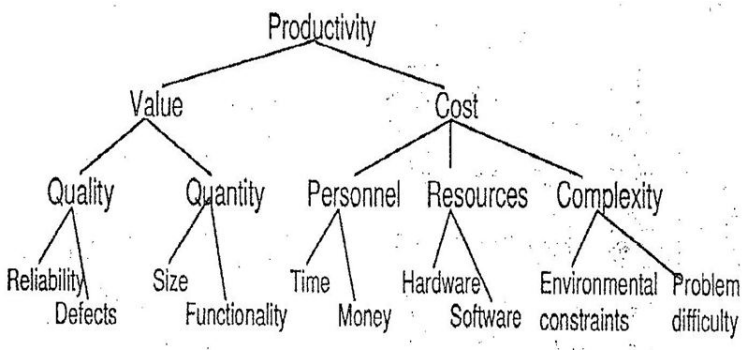
- So measurement is needed at least for assessing the status of your projects, products, processes, and resources
- Let's talk about objectives of software engineering measurements from the perspectives of management as well as engineering staff
- Managers' Perspectives
  - ✓ What does each process cost?
  - ✓ How productive is the staff?
  - ✓ How good is the code being developed?
  - ✓ Will the user be satisfied with the product?
  - ✓ How can we improve?

Importance of Measurement

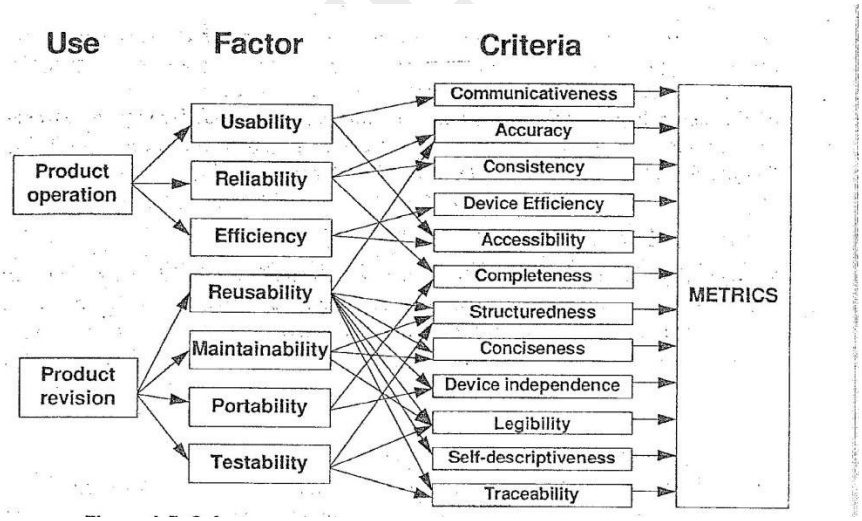
- Measurement is important for three basic activities
  - ✓ First, there are measures that help us to understand what is happening during development and maintenance
  - ✓ We assess the current situation, establishing baselines that help us to set goals for future behavior. In this sense, the measurements make aspects of process and product more visible to us, giving us a better understanding of relationships among activities and the entities they affect
  - ✓ Second, the measurement allows us to control what is happening on our projects
  - ✓ Using our baselines, goals and understanding of relationships, we predict what is likely to happen and make changes to processes and products that help us to meet our goals. For example, we may monitor the complexity of code modules, giving through review only to those that exceed acceptable bounds
  - ✓ Third, measurement encourages us to improve our processes and products
  - ✓ For instance, we may increase the number or type of design reviews we do, based on measures of specification quality and predictions of likely design quality
  - ✓ You can neither predict not control what you cannot measure

Scope of Software Metrics

- Software metrics is a term that embraces many activities, all of which involve some degree of software measurement:
  - ✓ Cost and effort estimation
    - Managers provided the original motivation for deriving and using software measures. They wanted to be able to predict costs during early phases in the software lifecycle
    - Many models are available to express cost and size
  - ✓ Productivity models and measures
    - The pressing needs of management have also resulted in numerous attempts to define measures and models for assessing staff productivity during different software processes and in different environments
    - A Productivity Model



- ✓ Data collection
  - The quality of any measurement program is clearly dependent on careful data collection
  - Consistent and accurate data collection is very difficult
  - Metrics data collection must be planned and executed in a careful and sensitive manner
- ✓ Quality models and measures
  - Productivity cannot be viewed in isolation. Without an accompanying assessment of product quality, speed of production is meaningless
  - Therefore, a number of models of quality whose measurements can be combined with those of productivity models



- ✓ Reliability models
  - Most quality models include reliability as a component factor, but the need to predict and measure reliability itself has led to a separate specialization in reliability modeling and prediction
  - This provides better understanding and control of software products
- ✓ Performance evaluation and models
  - Performance is another aspect of quality. Performance evaluation includes externally observable system performance characteristics, such as response times and completion rates

- Performance specialists investigate the internal workings of a system, including the efficiency of algorithms as embodied in embodied and algorithmic complexity

✓ Structural and complexity metrics
- Desirable quality attributes like reliability and maintainability cannot be measured until some operational version of the code is available
- Yet we wish to be able to predict which parts of the software system are likely to be less reliable, more difficult to test, or require more maintenance than others, even before the system is complete
- As a result, we measure structural attributes of representations of the software which are available in advance of (or without the need for) execution; then, we try to establish empirically predictive theories to support quality assurance, quality control, and quality prediction

✓ Management by metrics
- Measurement is becoming an important part of software project management
- Customers and developers alike rely on measurement-based charts and graphs to help them decide if the project is on track
- It is important to collect measures in a standard format for comparisons and contrast

✓ Evaluation of methods and tools
- In order to select a method or tool for adoption, it is important to investigate them before incorporating them into an organization
- These investigations can be in the form of experiments, case studies, or suveys
- These investigation cannot be done without careful, control measurement and analysis

✓ Capability maturity assessment
- SEI proposed capability maturity assessments to evaluate contractors have been in use since 1980s, using quality models
- Capability maturity assessment is based on key practices that every good contractor should be using
- Other organizations are using their own capability assessments

Summary

- Measurement pervades our everyday life
- Measurement is essential for good engineering in other disciplines; it should likewise become an integral part of software engineering practice

References

- Software Metrics: A Rigorous & Practical Approach, by Norman E. Fenton and Shari L. Pleeger, 2nd Edition, PWS Publishing Company, 1997 (Chapter 1)

Lecture 39

Basics of Measurements

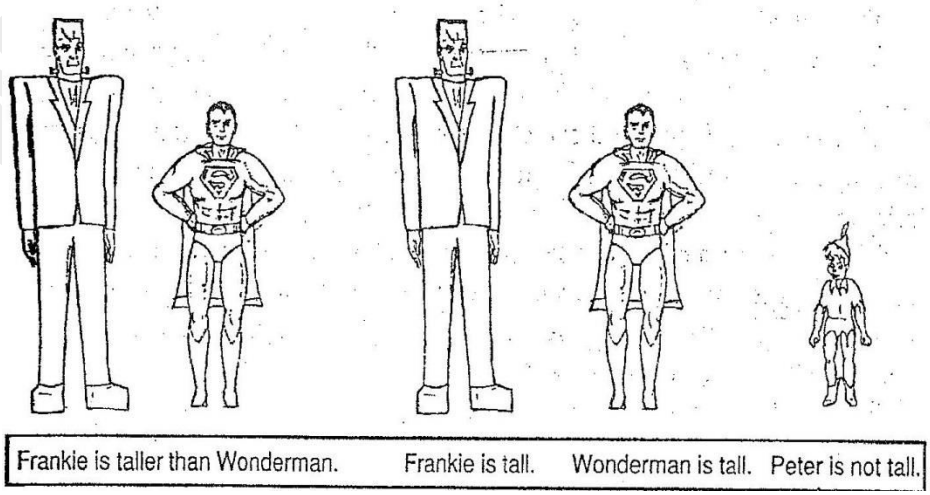
- We have a had good grounding on measurements. Today, we'll talk more about measurement
- Ordinarily, when we measure things, we do not think about the scientific principles we are applying
- We measure attributes such as the length of physical objects, the timing of events, and the temperature of liquids or of the air
- To do the measuring, we use both tools and principles that we now take for granted
- However, these sophisticated measuring devices and techniques have been developed over time, based on the growth of understanding of the attributes we are measuring
- For example, using the length of a column of mercury to capture information about temperature is a technique that was not at all obvious to the first person who wanted to know how much hotter it is in summer than in winter
- As we understood more about temperature, materials, and the relationships between them, we developed a framework for describing temperature as well as tools for measuring it
- Unfortunately, we have no comparably deep understanding of software attributes
- Nor do we have the associated sophisticated measurement tools
- Questions that are relatively easy to answer for non-software entities are difficult for software
- Question # 1:
  - ✓ How much *must* we know about an attribute before it is reasonable to consider measuring it?
  - ✓ For instance, do we know enough about "complexity" of programs to be able to measure it?
- Question # 2:
  - ✓ How do we *know* if we have really measured the attribute we wanted to measure?
  - ✓ For instance, does a count of the number of "bugs" found in a system during integration testing measure the quality of the system? If not, what does the count tell us?
- Question # 3:
  - ✓ Using measurement, *what* meaningful statements can we make about an attribute and the entities that posses it?
  - ✓ For instance, is it meaningful to talk about doubling a design's quality? If not, how do we compare two different designs?
- Question # 4:
  - ✓ What meaningful *operations* can we perform on measures?
  - ✓ For instance, is it sensible to compute average productivity for a group of developers, or the average quality of a set of modules?
- To answer these questions, we must establish the basics of a theory of measurement
- Let's start by examining formal measurement theory, developed as a classical discipline from the physical sciences
- We see how the concepts of measurement theory apply to software, and we explore several examples to determine when measurements are meaningful and useful

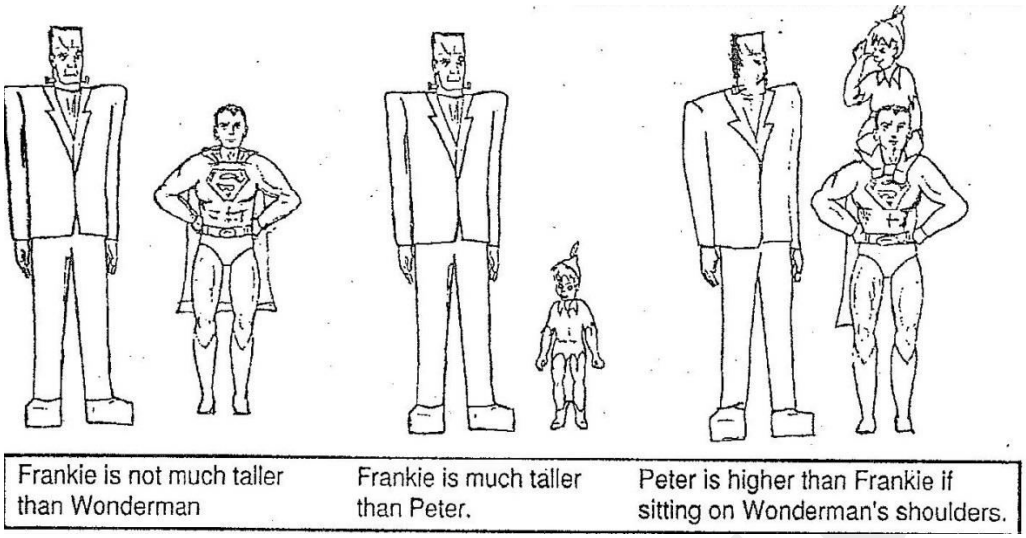
The Representational Theory of Measurement

- In any measurement activity, there are rules to be followed
- The rules help us to be consistent in our measurement, as well as providing a basis for interpreting data
- Measurement theory tells us the rules, laying the groundwork for developing and reasoning about all kinds of measurement
- This rule-based approach is common in many sciences
- For example, the mathematicians learned about the world by defining axioms for a geometry. Then by combining axioms and using their results to support or refute their observations, they expanded their understanding and the set of rules that govern the behavior of objects
- In the same way, we can use the rules about measurement to codify our initial understanding, and then expand our horizons as we analyze our software
- There are different kinds of geometry (Euclidean and non-Euclidean), and depending on the sets of rules chosen, there are several theories of measurement, we are talking about representational theory of measurements here
- The representational theory of measurement seeks to formalize our intuition about the way the world works
- That is, the data we obtain as measures should represent attributes of the entities we observe, and manipulation of the data should preserve relationships that we observe among the entities
- Thus, our intuition is the starting point for all measurement
- Which brings us to the concept of empirical relations

Empirical Relations

- Consider the way we perceive the real world
- We tend to understand things by comparing them, not by assigning numbers to them
- Let's talk about height of people
- We observe that certain people are taller than others without actually measuring them
- Some Empirical Relations for the Attribute "Height"





- It is easy to see that Frankie is taller than Wonderman who in turn is taller than Peter
- However, our observation reflects a set of rules that we are imposing on the set of people. We form pairs of people and define a binary relation on them. In other words, "taller than" is a binary relation defined on the set of pairs of people
- Given any two people, x and y, we can observe that
  - ✓ x is taller than y, or
  - ✓ y is taller than x
- Therefore, we say that "taller than" is an empirical relation for height
- A (binary) empirical relation is one for which there is a reasonable consensus about which pairs are in the relation
- We can define more than one empirical relation on the same set (e.g., "Much taller than")
- Most of us would agree that both Frankie and Wonderman are much taller than Peter
- Empirical relations need not be binary. We can define a relation on a single element of a set, or on a collections of elements
- Many empirical relations are unary, meaning that they are defined on individual entities. The relation "is tall" is an example of a unary relation
- Similarly, we can define a ternary relationship by comparing groups of three
- We can think of these relations as mappings from the empirical world to the formal mathematical world
- We have entities and their attributes in the real world, and we define a mathematical mapping that preserves the relationships we observe
- Thus, height can be considered as a mapping from a set of people to a set of real numbers
- If we agree that Frankie is taller than Wonderman, then any measure of height should assign a higher number to Frankie than to Wonderman
- This preservation of intuition and observation is the notion behind the representation condition of measurement
- Let's consider the example of four different word processors: A, B, C, and D
- Let's try to compare there functionality and user-friendliness

- The comparison is based on a survey of 100 independent computer users
- Comparisons of "Functionality" in Four Products

	A	B	C	D
A	-	80	10	80
B	20	-	5	50
C	90	95	-	96
D	20	50	4	-

- Each cell of the table represents the percentage of respondents who preferred the row's program to the column's program

- Comparisons of "User-Friendliness" in Four Products

	A	B	C	D
A	-	45	50	44
B	55	-	52	50
C	50	48	-	52
D	54	50	49	-

- Each cell of the table represents the percentage of respondents who preferred the row's program to the column's program

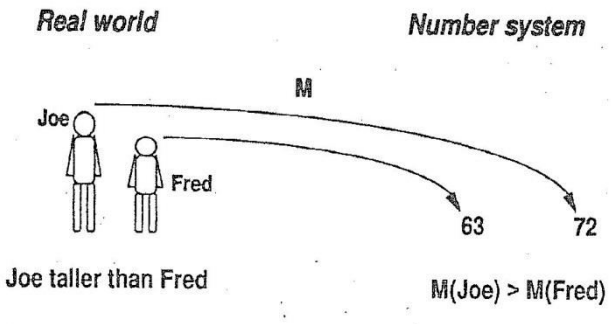
- Can we make any judgments about "functionality" (greater or lesser) and "user-friendliness" of these software products
- We can begin to understand the world by using relatively unsophisticated relationships that require no measuring tools
- Once we develop an initial understanding and have accumulated some data, we may need to measure in more sophisticated ways and with special tools
- Analyzing results often leads to the clarification and re-evaluation of the attribute, and yet more sophisticated empirical relations. In turn, we have improved accuracy and increased understanding
- Measurement is a mapping from the empirical world to the formal, relational world
- Measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute
- There are several rating formats
  - ✓ Likert scale (Strongly Agree, Agree, …, Strongly Disagree)
  - ✓ Forced ranking (give n alternatives, ordered from 1 (best) to n (worst))
  - ✓ Verbal frequency scale (Always, Often, …, Never)
  - ✓ Ordinal scale (1…., 2…., …., n….)
  - ✓ Comparative scale (Very superior, …, Very inferior)
  - ✓ Numerical scale (1,2,…10)

Rules of Mapping

- We have seen how a measure is used to characterize an attribute
- We begin in the real world, studying an entity and trying to understand more about it. Thus, the real world is the domain of the mapping, and the mathematical world is the range
- When we map the attribute to a mathematical system, we have many choices for the mapping and the range
- We can use real numbers, integers, or even a set of non-numeric symbols
- A measure must specify the domain and range as well as the rule for performing
- Sometimes a measure is associated with a number, the assumptions about the mapping are well-known, and our terminology is imprecise
- For example, we say "Felix's age is 11," or "Felix is 11." In expressing ourselves in this way, we really mean that we are measuring age by mapping each person into years in such a way that we count only whole years since birth
- We encounter some of the problems in measuring software. For example, many organizations measure the size of their source code in terms of the number of lines of code in a program
- But the definition of a line of code must be made clear
- SEI has developed a checklist to assist developers in deciding exactly what is included in a line of code

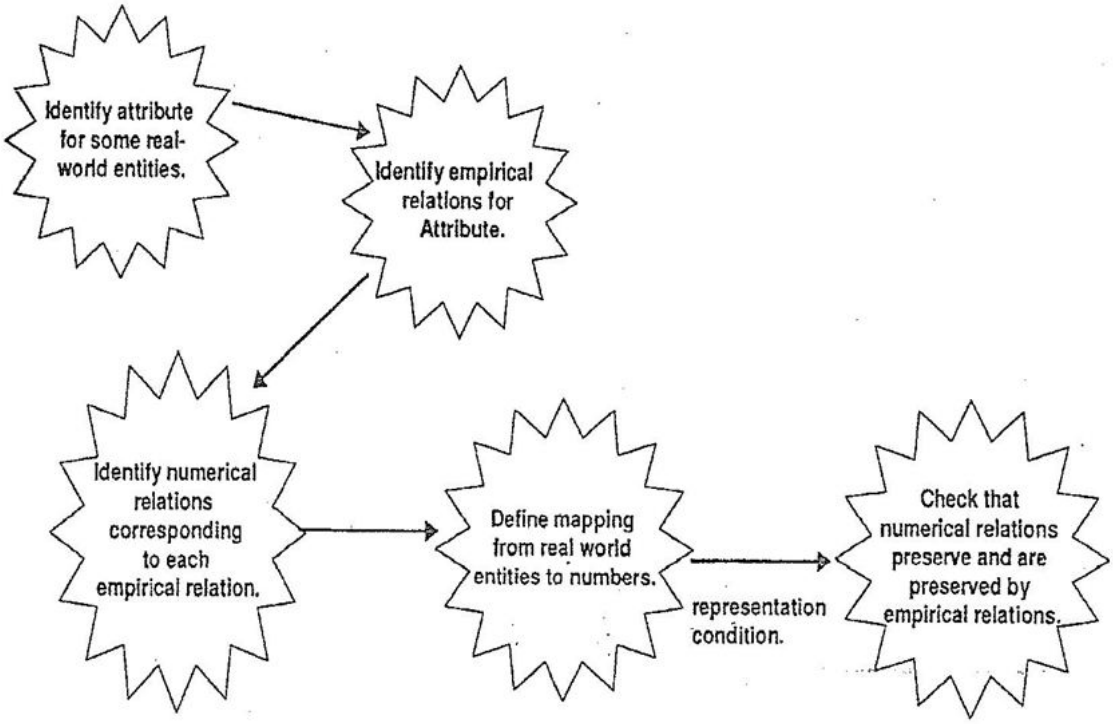
Representation Condition of Measurement

- By definition, each relation in the empirical relational system corresponds via the measurement to an element in a number system
- We want the behavior of the measures in the number system to be the same as the corresponding elements in the real world, so that by studying the numbers, we learn about the real world
- Thus, we want the mapping to preserve the relation
- This rule is called the representation condition
- Representation condition asserts that a measurement mapping M must map entities into numbers and empirical relations into numerical relations in such a way that the empirical relations preserve and are preserved by the numerical relations
- Representation Condition



A is taller than B if and only if M(A)>M(B)
Empirical relation preserved under M as Numerical relation

- We can define numerical relations for binary, unary, or ternary empirical relations
- "taller than"
- "is tall"
- "much taller than"
- Explain with the help of example of Frankie, Wonderman, and Peter
- Key Stages of Formal Measurements



Measurement and Models

- Defining attributes
- Direct and indirect measurement (length of an entity vs. density of a physical object)
  - ✓ Length of source code
  - ✓ Duration of testing process
  - ✓ Number of defects discovered during testing process
  - ✓ Time a programmer spends on a project
- Indirect measurement is often useful in making visible the interactions between direct measurement
- That is, it is sometimes easier to see what is happening on a project by using combinations of measures

Common Indirect Measures is SWE

- Programmer productivity = LOC produced/person month effort
- Module defect density = number of defects/module size
- Defect detection efficiency = number of defects detected/total number of defects

- Requirements stability = number of initial requirements/total number of requirements
- Test effectiveness ration = number of items covered/total number of items
- System spoilage = effort spent on fixing faults/total project effort

- Where no previous measurement has been performed, direct measurement constitutes the natural process of trying to understand entities and the attributes they possess
- However, simple models of direct measurement do not preclude the possibility of more accurate subsequent measurement that will be achieved indirectly
- For example, temperature can be measured as the length of a column of mercury under given conditions
- This measure is indirect because we are examining the column, rather than the entity whose temperature we want to know
- When we talk about measuring something, we usually mean that we wish to assess some entity that already exists
- This measurement for assessment is very helpful in understanding what exists now or what has happened in the past
- However, in many circumstances, we would like to predict an attribute of some entity that does not yet exist
- For example, early assurance on reliability of a very sophisticated software system
- There are other attributes as well, for example, cost, performance, etc.
- So, we predict things based on models, and we make measurements of prediction
- The distinction between measurement for assessment and predictions is not always clear-cut
- For example distance on globe between Washington and London
- And making a prediction on the distance we will be travelling on a trip between Washington and London

A Prediction System

- A prediction system consists of a mathematical model together with a set of prediction procedures for determining unknown parameters and interpreting results

Measurement Scales and Scale Types

- We know that direct measurement of an attribute assigns a representation or mapping M from an observed relation system to some numerical relation system
- The purpose of performing the mapping is to be able to manipulate data in the numerical system and use the results to draw conclusions about the attribute in the empirical system
- We do this sort of analysis all the time
- For example, we use a thermometer to measure air temperature, and then we conclude that it is hotter today than yesterday
- The numbers tell us the characteristic of the air
- But not all measurement mappings are the same, and the differences among the mappings can restrict the kind of analysis we can do

- To understand these differences, we introduce the notion of a measurement scale, and then we use the scale to help us understand which analyses are appropriate

Measurement Scales

- Nominal
- Ordinal
- Interval
- Ratio
- Absolute

References

- Software Metrics: A Rigorous & Practical Approach, by Norman E. Fenton and Shari L. Pfleeger, 2nd Edition, PWS Publishing Company, 1997(Chapter 2.1-2.2)

Lecture 40
Measurement Scales

1. Nominal
2. Ordinal
3. Interval
4. Ratio
5. Absolute

1 – Nominal Scale

- We define classes or categories, and then place each entity in a particular class or category, based on the value of the attribute
- This is nominal measurement
- Classes are not ordered
- The empirical relation system consists only of different classes; there is no notion of ordering among the classes
- Any distinct numbering or symbolic representation of the classes is an acceptable measure, but there is no notion of magnitude associated with the numbers or symbols
- Example
  - ✓ We are trying to capture the location of software faults (specification, code, design)

2 – Ordinal Scale

- The ordinal scale is often useful to augment the nominal scale with information about an ordering of the classes or categories
- The ordering leads to analysis not possible with nominal measures
- The empirical relation system consists of classes that are ordered with respect to the attribute
- Any mapping that preserves the ordering (that is, any monotonic function) is acceptable
- The numbers represent ranking only, so addition, subtraction, and other arithmetic operations have no meaning
- Example
  - ✓ Complexity of software modules is described as:
    - Trivial
    - Simple
    - Moderate
    - Complex
    - Incomprehensible

3 – Interval Scale

- The interval scale carries more information still, making it more powerful than nominal or ordinal scales
- This scale captures information about the size of the intervals that separate the classes, so that we can in some sense understand the size of the jump from one class to another
- An interval scale preserves order, as with an ordinal scale
- An interval scale preserves differences but not ratios. That is, we know the difference between any two of the ordered classes in the range of the mapping, but computing the ratio of two classes in the range does not make sense

- Addition and subtraction are acceptable on the interval scale, but not multiplication and division
- Example
  - ✓ Complexity of software modules is described as:
    - Trivial 0
    - Simple 2
    - Moderate 4
    - Complex 6
    - Incomprehensible 8

4 – Ratio Scale

- We would like to be able to say that one liquid is twice as hot as another, or that one project took twice as long as another
- This need for ratios gives rise to ratio scale
- It is a measurement mapping that preserves ordering, the size of intervals between entities, and ratios between entities
- There is a zero element, representing total lack of the attribute
- The measurement mapping must start at zero and increase at equal intervals, known as units
- All arithmetic can be meaningfully applied to the classes in the range of the mapping
- Example
  - ✓ The length of software code is also measurable on a ratio scale

5 – Absolute Scale

- There is only one way in which the measurement can be made, so M and M' must be equal
- The absolute scale is the most restrictive of all
- The measurement for an absolute scale is made simply by counting the number of elements in the entity set
- The attributes always takes the form "number of occurrences of x in the entity"
- There is only one possible measurement mapping, namely the actual count
- All arithmetic analysis of the resulting count is meaningful
- Example
  - ✓ LOC is an absolute scale measure of the attribute "number of lines of code" of a program
  - ✓ However, LOC is not an absolute-scale measure of length, because there are different ways to measure length (such as thousands of LOC, number of characters, and number of bytes)

Scales of Measurement

Scale Type	Admissible transformations	Examples
Nominal	1-1 mapping from M to M'	Labeling, classifying entities
Ordinal	Monotonic increasing function from M to M', that is, M(x) >= M(y) implies M'(x) >= M'(y)	Preference, hardness, air quality, intelligence tests (raw scores)

Interval	M' = aM + b (a>0)	Relative time, temperature (Fahrenheit, Celsius), intelligence tests (standardized scores)
Ratio	M' = aM (a>0)	Time interval, length, temperature (Kelvin)
Absolute	M' = M	Counting entities

Meaningfulness in Measurement

- Can we deduce meaningful statements about the entities being measured?
- The number of errors discovered during the integration testing of program X was at least 100
- The cost of fixing each error in program X is at least 100 (meaningless without reference to a particular scale)
- A semantic error takes twice as long to fix as a syntactic error
- A semantic error is twice as complex as a syntactic error (not meaningful without clarifying complexity)
- A statement involving measurement is meaningful if its truth value is invariant of transformations of allowable scales

A Goal-based Framework for Software Measurement

- Now, we'll talk about a conceptual framework for the diverse software-measurement activities that contribute to an organization's software practices
- These practices may include not only your usual development and maintenance activities, but also any experiments and case studies you may perform as you investigate new techniques and tools
- This framework is based on three principles:
  - ✓ Classifying the entities to be examined
  - ✓ Determining relevant measurement tools
  - ✓ Identifying the level of maturity that your organization has reached

Classifying Software Measures

- The first obligation of any software measurement activity is identifying the entities and attributes we wish to measure
- In software, there are three such classes:
  - ✓ Processes
    - These are collections of software-related activities
  - ✓ Products
    - These are any artifacts, deliverables or documents that result from a process activity
  - ✓ Resources
    - These are entities required by a process activity

- A process is usually associated with some timescale, i.e., the activities in the process are ordered or related in some way that depends on time, so that one activity must be completed before another can begin
- The timing can be explicit, as when design must be complete by October 31, or implicit, as when a flow diagram shows that design must be completed before coding can begin
- Resources and products are associated with the process
- Each process activity has resources and products that it uses, as well as products that are produced
- Thus, the product of one activity may feed another activity, e.g., the design document
- Before, we discuss these entities in detail, let's distinguish between internal and external attributes:
    - ✓ Internal attributes
        - Internal attributes of a product, process or resource are those that can be measured purely in terms of the product, process or resource itself
        - In other words, an internal attribute can be measured by examining the product, process or resource on its own, separate from its behavior
    - ✓ External attributes
        - External attributes of a product, process or resource are those that can be measured only with respect to how the product, process or resource relates to its environment
        - Here, the behavior of the process, product or resource is important, rather than the entity itself
- Examples of Internal and External Attributes
    - ✓ Some attributes of software can be determined without executing it, like, software size, its complexity, and the dependencies among modules
    - ✓ Some attributes of software can only be measured when code is executed, like, the number of failures experienced by a user, difficulty in navigating screens, database search time
    - ✓ There is a clear need for internal attribute measurements to support measurement and decision making about external attributes
    - ✓ We need to identify the relationships among internal and external attributes, as well as to find new and useful methods for measuring directly the attributes of interest

Processes (3.1.1)

- We often have questions about our software development activities and processes that measurement can help us answer
- We want to know how long it takes for a process to complete, how much it will cost, whether it is effective or efficient, and how it compares with other processes that we could have chosen
- However, only a limited number of internal process attributes can be measured directly
- These measures include:
    - ✓ The duration of the process or one of its activities;
    - ✓ The effort associated with the process or one of its activities;

- ✓ The number of incidents of a specified type arising during the process or one of its activities

- For example, we may be reviewing our requirements to ensure their quality before turning them over to the designers
- To measure the effectiveness of the review process we can measure the number of requirements errors found during specification
- We can measure the number of requirements errors found during integration testing to determine how well we are doing
- We can get insight into the resources needed for development process by measuring the number of personnel working on the project between May 1 and September 30

- Average Cost of each Defect Detected during the process
  - ✓ Cost / Number of Errors Found
  - ✓ This is an indirect measure

- Effectiveness of Software Inspections
  - ✓ Measure the average amount of effort expended per thousand lines of code reviewed

- Effectiveness of Software Testing Process
  - ✓ The testing process may be composed of unit testing, integration testing, system testing, and acceptance testing
  - ✓ Each component process can be measured to determine how effectively it contributes to overall testing effectiveness
  - ✓ We can track the number of errors identified in each sub-process, along with the duration and cost of identifying each error, to see if each sub-process is cost-effective

- Cost is not the only process measure we can examine
- Controllability, observability, and stability are also important in managing large projects
- These attributes are clearly external
- We often use propose objective measures of external attributes in terms of internal attributes
- For example, measures of the effectiveness of code maintenance can be defined in terms of the number of faults discovered and the number of faults corrected
- Attributes of Process

Entities	**Internal Attributes**	**External Attributes**
Constructing specification	Time, effort, number of requirements changes, …	Quality, cost, stability, …
Detailed design	Time, effort, number of specification faults found, …	Cost, cost-effectiveness, …
Testing	Time, effort, number of coding faults found, …	Cost, cost-effectiveness, stability, …
…	…	…

Products (3.1.2)

- Products are not restricted to the items that management is committed to deliver to the customer
- Any artifact or document produced during the software life cycle can be measured and assessed. Examples are prototypes and test harnesses
- Documents, e.g., user guide/customer specification document, can be measured for size, quality, and other things
- External Product Attributes
  - ✓ An external product attribute depends on both product behavior and environment, each attribute measure should take these characteristics into account
  - ✓ E.g., if we are interested in measuring the reliability of code, we must consider the machine on which the program is run as well as the mode of operational usage
  - ✓ E.g., a word-processors and databases
  - ✓ The understandability of a document depends on the experience and credentials of the person reading it; a nuclear engineer reading the specification for power-plant software is likely to rate its understandability higher than a mathematician reading the same document
  - ✓ Maintainability of a system may depend on the skills of the maintainers and the tools available to them
  - ✓ Usability, integrity, efficiency, testability, reusability, portability, and interoperability are other external attributes that we can measure
  - ✓ These attributes describe not only the code but also other documents that support the development effort
- Internal Product Attributes
  - ✓ Internal product attributes are sometimes easy to measure
  - ✓ We can determine the size of a product by measuring the number of pages it fills or the number of words it contains
  - ✓ Since products are concrete, we have a better understanding of attributes like size, effort, and cost
  - ✓ Other internal product attributes are more difficult to measure, because opinions differ as to what they mean and how to measure them
  - ✓ E.g., there are many aspects of code complexity, and no consensus about what best measures it
  - ✓ Since products are relatively easy to examine in an automated fashion, there is a set of commonly used internal attributes
  - ✓ Specifications can be assessed in terms of their length, functionality, modularity, reuse, redundancy, and syntactic correctness
  - ✓ Formal designs and code can also be measured in the same way
  - ✓ We can also measure attributes such as structuredness (of control and data flow, for example) as well as module coupling and cohesiveness
  - ✓ Some people consider many of these unimportant, focusing only on functionality, quality, and utility of software

- ✓ However, internal attributes can be very helpful in suggesting what we are likely to find as the external attributes
- ✓ Consider the example of buying a used car, where we make decisions based on dynamic testing (driving in different conditions). These conditions are not available at all times, so supplement our decision-making process by certain static properties
- ✓ These internal attributes provide insight into external attributes
- ✓ Changes to the inspection process, for instance, may be based on measures of faults found, even though the ultimate goal of reliability is based on failures, not faults
- ✓

- An Example
  - ✓ A software system design document may consist of a large number of module design documents
  - ✓ Average module size, an attribute of the system design, can be derived by calculating the size of each module
  - ✓ We can measure size by the number of DFD bubbles in each diagram

- Size of one Module = No of DFD Bubbles
- Average Module Size = $1/n \sum_{i=1 \text{ to } n}$ No of DFD Bubbles$_i$

- Almost all design methodologies give structure to software products, which make them easier to understand, analyze, and test
- The structure involves two aspects of development
  - ✓ The development process, since certain products need to be produced at certain stages
  - ✓ The products themselves, since the products must confirm to certain structural principles
- In particular, product structure is usually characterized by levels of internal attributes such as modularity, coupling, or cohesiveness
- Quality is frequently used by software engineers to describe an internal attribute of design or code
- However, quality is multi-dimensional; it does not reflect a single aspect of a particular product
- The concept of quality is very similar to that of gross national product (GNP) of a country
- Economists look at the trend in GNP over time, hoping that it will rise – that the country is becoming more productive
- The GNP is a weighted combination of the values of key goods and services produced; the weights reflect the priorities and opinions of the economists defining the measure
- The value of individual goods and services can be measured directly, but the GNP itself is an indirect measure
- We measure and control the quality of our products by measuring and controlling a number of internal (structural) product attributes
- We must also articulate specific attributes that contribute to the general notion of quality, including but not limited to cognitive and structural notions of complexity, maintainability, and usability

- Attributes of Products

Entities	Internal Attributes	External Attributes
Specifications	Size, reuse, modularity, redundancy, functionality, syntactic correctness, …	Comprehensibility, maintainability, …
Designs	Size, reuse, modularity, coupling, cohesiveness, functionality, …	Quality, complexity, maintainability, …
Code	Size, reuse, modularity, coupling, functionality, algorithmic complexity, control-flow structuredness, …	Reliability, usability, maintainability, …
Test data	Size, coverage, level, …	Quality, …
…	…	…

Resources (3.1.3)

- The resources that we are likely to measure include any input for software production
- These can be
  - ✓ Personnel (individuals and teams)
  - ✓ Materials (including office supplies)
  - ✓ Tools (both software and hardware)
  - ✓ Methods
- We measure resources to determine their magnitude (how many staff are working on this project?), their cost (how much are we paying for testing tools?), and their quality (how experienced are our designers?)
- These measures help us to understand and control the process by telling us how the process is using and changing inputs to outputs
- For example, if we are producing poor-quality software, resource measurements may show us that the software quality is the result of too few people or people with the wrong skills
- Cost is often measured across all types of resources, so that managers can see how the cost of the inputs affects the cost of the outputs
- Productivity is always important, and managers are keen not only to measure it but also to understand how to improve it
- Although a measure of staff, productivity is an external resource attribute, since it depends on the underlying development process
- A productive worker using one process may become less productive if the process changes

- Productivity = Amount of Output / Effort Input
  - ✓ This resource measure combined a process measure (input) with a product measure (output)

- There are many other staff attributes that we can measure, whose values may have an influence on the process or the product
- For example, the experience, age, or intelligence of a developer may affect the quality of the design or code

- Similarly, the size, structure, and communication patterns of the development team are important
- We can classify and analyze tools and methods. Languages are block-structured or not, object-oriented or not, and so on
- Techniques can be rated as manual or automated, and tools can require specialized training or experience
- These attributes of resources help us to understand how to use tools and methods in more effective ways
- No developer has the time to measure everything
- It is important to focus measurements activities on those areas needing the most visibility, understanding, and improvement
- Attributes of Resources

Entities	Internal Attributes	External Attributes
Personnel	Age, price, …	Productivity, experience, intelligence, …
Teams	Size, communication level, structuredness, …	Productivity, quality, …
Software	Price, size, …	Usability, reliability, …
Hardware	Price, speed, memory, size, …	Reliability, …
Offices	Size, temperature, light, …	Comfort, quality, …
…	…	…

Determining What to Measure?

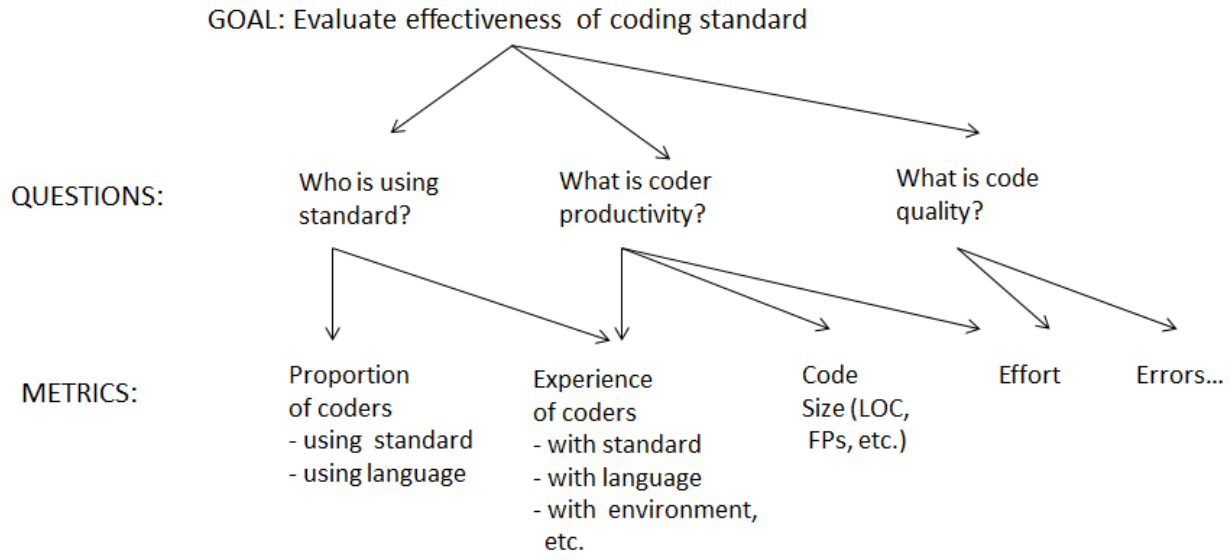
- Utility is in the eye of the beholder. In other words, a particular measurement is useful only if it helps you to understand the underlying process or one of its resultant products
- In turn, recognizing improvement of the process and products can occur only when the project has clearly defined goals for process and products
- That is, you cannot tell if you are going in the right direction until you know where you want to go
- The Goal-Question-Metric (GQM) approach to process and metrics has proven to be a particularly effective approach to selecting and implementing metrics
- To use GQM, you express the overall goals of your organization
- Your organization can encompass the entire corporation for which you work, or only the quality assurance group to which you are assigned; the scope of metrics and goals is up to you and your needs
- Then, you generate questions whose answers you must know in order to determine if your goals are being met
- Finally, you analyze each question in terms of what measurements you need in order to answer each question
- Once you have a list of possible measures, you must assess your organization to see whether it is capable of providing useful information for the measurement

- For example, if your developers cannot tell when design ends and coding begins, then measuring the duration of the design activity will be impossible
- To understand when measurement is appropriate for your organization, you must know the process maturity of your organization
- The more mature your process, the more that is visible and therefore measurable

GQM Paradigm

- A measurement program can be more successful if it is designed with the goals of the project in mind
- The GQM approach provides a framework involving three steps
- Steps of GQM Framework
  - ✓ List the major goals of the development or maintenance project
  - ✓ Derive from each goal the questions that must be answered to determine if the goals are being met
  - ✓ Decide what must be measured in order to be able to answer the questions adequately
  - ✓ By deriving the measurements in this way, it becomes clear how to use the resulting data
  - ✓ Several metrics might be generated from a single goal
  - ✓ One metric may be supporting several goals
  - ✓ Suppose your overall goal is to evaluate the effectiveness of using a coding standard
  - ✓ First, it is important to know who is using the standard, so that you can compare the productivity of those who do not. You would also want to compare the quality of the code produced with the standard with the quality of non-standard code
  - ✓ Second, you must analyze each question to determine what must be measured in order to answer the question
  - ✓ For example, to understand who is using the standard, it is necessary to know what proportion of coders is using the standard. It is also important to have an experience profile of coders, explaining how long they have worked with the standard, the environment, the language, and other factors that will help evaluate the effectiveness of the standard
  - ✓ The productivity question requires a definition of productivity, which is usually some measure of effort divided by some measure of product size (in LOC, FPs, …)
  - ✓ Quality may be measured in terms of the number of errors found in the code, plus any other quality measures that you would like to use
  - ✓ In this way, you generate only those measures that are related to the goal
  - ✓ In many cases, several measurements may be needed to answer a single question
  - ✓ Likewise, a single measurement may apply to more than one question
  - ✓ The goal provides the purpose for collecting the data, and the questions tell you and your project how to use the data

Deriving Metrics from Goals and Questions



References

- Software Metrics: A Rigorous & Practical Approach, by Norman E. Fenton and Shari L. Pfleeger, 2nd Edition, PWS Publishing Company, 1997(Chapter 3.1-3.2 – up to page 85)

Lecture 41
Goal-Question-Metric (GQM)

Goal Question Metric Approach

- The Goal Question Metric (GQM) approach is based upon the assumption that for an organization to measure in a purposeful way
  - ✓ it must first specify the goals for itself and its projects, then
  - ✓ it must trace those goals to the data that are intended to define those goals operationally, and finally
  - ✓ provide a framework for interpreting the data with respect to the stated goals
- The result of the application of the Goal Question Metric approach is the specification of a measurement system targeting a particular set of issues and a set of rules for the interpretation of the measurement data
- The resulting measurement model has three levels:
  1. Conceptual
  2. Operational
  3. Quantitative

1 – Conceptual Level (Goal)

- A goal is defined for an object, for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment
- Objects of measurement are
  - ✓ Products
    - Artifacts, deliverables and documents that are produced during the system life cycle; E.g., specifications, designs, programs, test suites
  - ✓ Processes
    - Software related activities normally associated with time; E.g., specifying, designing, testing, interviewing
  - ✓ Resources
    - Items used by processes in order to produce their outputs; E.g., personnel, hardware, software, office space

2 – Operational Level (Question)

- A set of questions is used to characterize the way the assessment/achievement of a specific goal is going to be performed based on some characterizing model
- Questions try to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint

3 – Quantitative Level (Metric)

- A set of data is associated with every question in order to answer it in a quantitative way
- The data can be
  - ✓ Objective

- If they depend only on the object that is being measured and not on the viewpoint from which they are taken; E.g., number of versions of a document, staff hours spent on a task, size of a program

✓ Subjective

- If they depend on both the object that is being measured and the viewpoint from which they are taken; E.g., readability of a text, level of user satisfaction
- A number of templates are available to aid in generating the goals, questions, and metrics

Template for Goal Definition

- Template for Purpose
  - ✓ To (characterize, evaluate, predict, motivate, etc.) the (process, product, model, metric, etc.) in order to (understand, assess, manage, engineer, learn, improve, etc.) it
  - ✓ Example: To *evaluate* the *maintenance* process in order to *improve* it
- Template for Perspective
  - ✓ Examine the (cost, effectiveness, correctness, defects, changes, product measures, etc.) from the viewpoint of the (developer, manager, customer, etc.)
  - ✓ Example: Examine the *cost* from the viewpoint of the *manager*
- Template for Environment
  - ✓ The environment consists of the following: process factors, people factors, problem factors, methods, tools, constraints, etc.)
  - ✓ Example: The maintenance staff are poorly motivated programmers who have limited access to tools

Goal Question Metric Approach (continue)

- The GQM approach also provides separate guidelines for defining product-related and process-related questions
- Steps involve defining the process or product, defining the relevant attributes, and obtaining feedback related to the attributes
- What constitute a goal or question may be vague, and several levels of refinement may be required for certain goals and questions; that is, a goal may first have to be related to a series of sub-goals before questions can be derived
- We can relate the GQM templates to attribute framework
- A goal or a question can be associated with at least one pair of entities and attributes. Thus, a goal is stated, leading to a question that should be answered so that we can tell if we have met our goal; the answer to the question requires that we measure some attribute of an entity (and possibly several attributes of an entity or several entities)
- The use of the measure is determined by the goals and questions, so that assessment, prediction, and motivation are tightly linked to the data analysis and reporting
- Thus, the GQM complements the entity-attribute measurement framework
- The results of a GQM analysis are a collection of measurements related by goal tree and overall model

- However, GQM does not address issues of measurement scale, objectivity, or feasibility
- So the GQM measures should be used with care, remembering the overall goal of providing useful data that can help to improve our processes, products, and resources

Example of GQM

- AT&T used GQM to help determine which metrics were appropriate for assessing their inspection process
- Their goals, with questions and metrics derived are shown next
  - ✓ AT&T Goals, Questions, and Metrics for Inspection Process – Plan

Goal	Questions	Metrics
Plan	How much does the inspection process cost? How much calendar time does the inspection process take?	1. Average effort per KLOC 2. Percentage of reinspections 3. Average effort per KLOC 4. Total KLOC inspected

  - ✓ AT&T Goals, Questions, and Metrics for Inspection Process – Monitor and Control

Goal	Questions	Metrics
Monitor and Control	What is the quality of inspected software? To what degree did the staff conform to the procedures? What is the status of the inspection process?	1. Average faults detected per KLOC 2. Average inspection rate 3. Average preparation rate 4. Average inspection rate 5. Average preparation rate 6. Average lines of code inspected 7. Percentage of reinspections 8. Total KLOC inspected

  - ✓ AT&T Goals, Questions, and Metrics for Inspection Process – Improve

Goal	Questions	Metrics
Improve	How effective is the inspection process? What is the productivity of the inspection process?	1. Defect removal efficiency 2. Average faults detected per KLOC 3. Average inspection rate 4. Average preparation rate 5. Average lines of code inspected 6. Average effort per fault detected 7. Average inspection rate 8. Average preparation rate 9. Average lines of code inspected

GQM-Goal

- A GQM model is a hierarchical structure starting with a goal (specifying purpose of measurement, object to be measured, issue to be measured, and viewpoint from which the measure is taken)
- So a GQM goal has three coordinates and a purpose

GQM-Question

- The goal is refined into several questions, that usually break down the issue into its major components
- In general, we will ask at least three groups of questions:
  - ✓ Group 1. How can we characterize the object (product, process, or resource) with respect to the overall goal of the specific GQM model?
  - ✓ Group 2. How can we characterize the attributes of the object that are relevant with respect to the issue of the specific GQM model?
  - ✓ Group 3. How do we evaluate the characteristics of the object that are relevant with respect to the issue of the specific GQM model?

GQM-Metric

- Once the questions have been developed, we proceed to associating the question with appropriate metrics. The factors we consider in doing this are many; among them:
  - ✓ Amount and quality of the existing data: we will try to maximize the use of existing data-sources if they are available and reliable;
  - ✓ Maturity of the objects of measurement: we will apply objective measures to more mature measurement objects, and we will use more subjective evaluations when we deal with informal or unstable objects
  - ✓ Learning process: GQM models need always refinement and adaptation, therefore the measures we define must help us in evaluating not only the object of measurement but also the reliability of the model used to evaluate it

Another Example of GQM

Improve timelines of change request processing

GQM - Goals

- A GQM model is a hierarchical structure starting with a goal (specifying purpose of measurement, object to be measured, issue to be measured, and viewpoint from which the measure is taken)
- So a GQM goal has three coordinates:
  1. Issue: Timeliness
  2. Object (process): Change request processing
  3. Viewpoint: Project manager

Goal	Improve timelines of change request processing
Purpose	Improve
Issue	The timelines
Object	Change request processing
Viewpoint	Project Manager

GQM - Questions

- The goal is refined into several questions, that usually break down the issue into its major components
- We'll try to ask at least three groups of questions

GQM – Questions-Group 1

- How can we characterize the object (product, process, or resource) with respect to the overall goal of the specific GQM model?
- Example:
  - ✓ What is the current change request processing speed?
  - ✓ Is the (documented) change request process actually performed?

GQM – Questions-Group 2

- How can we characterize the attributes of the object that are relevant with respect to the issue of the specific GQM model?
- Example:
  - ✓ What is the deviation of the actual change request processing time from the estimated one?
  - ✓ Is the performance of the process improving?

GQM – Questions-Group 3

- How do we evaluate the characteristics of the object that are relevant with respect to the issue of the specific GQM model?
- Example:
  - ✓ Is the current performance satisfactory from the viewpoint of the project manager?
  - ✓ Is the performance visibly improving?

GQM – Metrics

- Once the questions have been developed, we proceed to associating the question with appropriate metrics. The factors we consider in doing this are many; among them:
  - ✓ Amount and quality of the existing data: we will try to maximize the use of existing data sources if they are available and reliable;
  - ✓ Maturity of the objects of measurement: we will apply objective measures to more mature measurement objects, and we will use more subjective evaluations when we deal with informal or unstable objects

- ✓ Learning process: GQM models need always refinement and adaptation, therefore the measures we define must help us in evaluating not only the object of measurement but also the reliability of the model used to evaluate it.

GQM – Model

Goal		
	Purpose	Improve
	Issue	The timelines of
	Object	Change request processing
	Viewpoint	Project Manager
Question	Q1	What is the current change request processing speed?
Metrics	M1	Average Cycle time
	M2	Standard deviation
	M3	%age cases outside of upper limit
Question	Q2	Is the (documented) change request process actually performed?
Metrics	M4	Subjective rating by Project Manager
	M5	%age of exceptions identified during reviews
Question	Q3	What is the deviation of the actual change request processing time from the estimated?
Metrics	M6	((current average cycle time – estimated average cycle time)/Current average cycle time)*100
	M7	Subjective rating by Project Manager
Question	Q4	Is the performance of the process improving?
Metrics	M8	(current average cycle time/Baseline average cycle time)*100
Question	Q5	Is the current performance satisfactory from the viewpoint of the project manager?
Metrics	M9	Subjective rating by Project Manager
Question	Q6	Is the performance visibly improving?
Metrics	M10	(Current average cycle time/Baseline average cycle time)*100

References

- Software Metrics: A Rigorous & Practical Approach, by Norman E. Fenton and Shari L. Pfleeger, 2nd Edition, PWS Publishing Company, 1997(Chapter 3.2.1-3.2.2)

Lecture 42

Software-Metrics Data Collection

- Software measurement is only as good as the data that are collected and analyzed
- In other words, we cannot make good decisions with bad data

Moroney, 1950

- "Data should be collected with clear purpose in mind. Not only a clear purpose but a clear idea as to the precise way in which they will be analyzed so as to yield the desired information. It is astonishing that men who in other respects are clear-sighted, will collect absolute hotchpotches of data in the blithe and uncritical belief that analysis can get something out of it"
- We need to ask questions, like
  - ✓ What constitutes good data

What is Good Data?

- Measures should be well-defined and valid: that our measures should reflect the attributes we claim they do
- But even when we have a well-defined measure that maps a real world attribute to a formal, relational system in appropriate ways, we need to ask several questions about data
- Are they Correct?
  - ✓ Correctness means that the data were collected according to the exact rules of definition of the metric
  - ✓ For example, if a lines of code count is supposed to include everything but comments, then a check for correctness assures us that no comments were counted
  - ✓ A measure of process duration is correct if time is measured from the beginning of one specified activity and ends at the completion of another specified activity
- Are they Accurate?
  - ✓ Accuracy refers to the difference between the data and the actual value
  - ✓ For example, time measured using an analog clock may be less accurate than time measured using a digital clock
- Are they Appropriately Precise?
  - ✓ Precision deals with the number of decimal places needed to express the data
  - ✓ For instance, activity duration need not be reported in hours, minutes, and seconds; hours or days are usually sufficient
  - ✓ Likewise, it is not necessary to calculate the mean cyclomatic number to several decimal places; since cyclomatic number expresses one plus the number decisions in a module, fractions of a decision to several decimals places are meaningless
- Are they Consistent?
  - ✓ Data should be consistent from one measuring device or person to another, without large differences in value. Thus, two evaluators should calculate the same or similar function-point values from the same requirements document

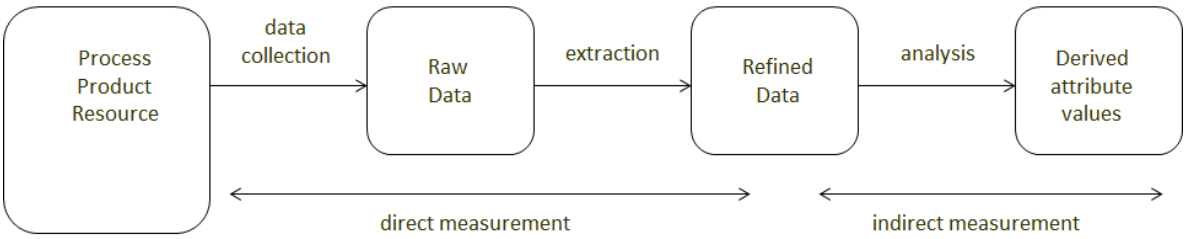
- ✓ Similarly, when the same data value is computed repeatedly over time, the data should be captured in the same way
- ✓ For example, we often measure the growth in size of a product over time, especially during maintenance
- ✓ We want to be sure that the size measure is calculated the same way each time, so that the resulting measures are comparable

- Are they Associated with a Particular Activity or Time Period
  - ✓ If so, then the data should be time-stamped, so that we know exactly when they were collected
  - ✓ This association of values allows us to track trends and compare activities
- Can they be Replicated?
  - ✓ Data are often collected to support surveys, case studies and experiments
  - ✓ These investigations are frequently repeated under different circumstances, and the results compared
  - ✓ At the very least, project histories and study results are stored in a historical database, so that baseline measures can be established and organizational goals set
  - ✓ Unless the data collection from one study can be replicated on others, this amalgamation is impossible
  - ✓ Thus, it is very important to assess the quality of data and data collection, before data collection begins
  - ✓ Your measurement program must specify not only what metrics to use, but also what precision is required, what activities and time periods are to be associated with data collection, and what rules govern the data collection (such as usage of specific tool for data collection)
  - ✓ Of critical importance is the definition of the metric
  - ✓ Terminology must be clear and detailed, so that all involved understand what the metric is and how to collect it

How to Define the Data?

- There are two kinds of data with which we are concerned
  - ✓ Raw data
  - ✓ Refined data
- Raw data results from the initial measurement of process, product, or resource
- But through the refinement process, raw data are transformed into refined data by extracting essential data elements
- The analysts can, then, derive values about attributes. This is shown as a figure

The Role of Data Collection in Software Measurement



- To see the difference, consider the measurement of developer effort
- The raw effort data may consist of weekly timesheets for each staff member working on a project
- To measure the effort expended on the design so far, we must select relevant timesheets and add up the figures
- This refined data is still a direct measurement of effort
- We may derive other, indirect measures as part of our analysis: average effort per staff member, or effort per design component

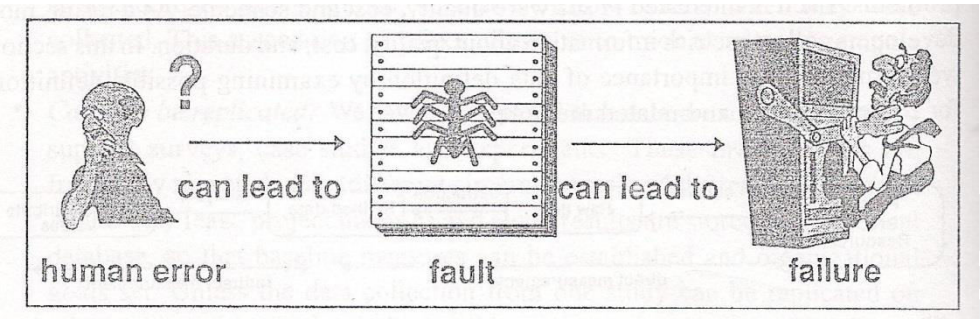
Back to defining data for collection

- Deciding what to measure is the first step
- We must specify which direct measures are needed, and also which indirect measures may be derived from the direct ones
- Sometimes, we begin with indirect measures. From a GQM analysis, we understand which indirect measures we want to know; from those, we must determine which direct measures are required to calculate them
- Most organizations are different from each other, not only in terms of their business goals but also in terms of corporate cultures, development preferences, staff skills, and more
- So, a GQM analysis of apparently similar projects may result in different metrics at different companies
- That is exactly why GQM is preferable to adopting a "one-size-fits-all" standard measurement set
- Nevertheless, most organizations share similar problems
- Each is interested in software quality, cost, and schedule
- As a result, most developers collect metrics information about quality, cost, and duration

The problem with problems

- No software developer consistently produces perfect software the first time
- Thus, it is important for developers to measure different aspects of software quality
- Such information can be useful for determining
  - ✓ How many problems have been found with a product;
  - ✓ How effective are the prevention, detection, and removal processes;
  - ✓ Whether the product is ready for release to the next development stage or to the customer;
  - ✓ How the current version of a product compares in quality with previous or competing versions

Software Quality Terminology



- A fault occurs when a human error results in a mistake in some software product. That is, the fault is the encoding of the human error
- A failure is the departure of a system from its required behavior. Failures can be discovered both before and after system delivery, as they can occur in testing as well as in operations
- We need to use and understand terminology consistently, especially those supply, collect, analyze, and use the data
- For example, we need to clearly define "faults", "errors", "processing errors", "anomalies", "defects", "bugs", "failures", "crashes", and etc.
- We also need a good, clear-way of describing what we do in reaction to problems
- For example, if an investigation of a failure results in the detection of a fault, then we make a change (or changes in many places) to the product to remove it
- A change (or changes in many places) can also be made if a fault is detected during a review or inspection process
- Whenever a problem is observed, we want to record its key elements, so that we can then investigate causes and cures
- In particular, we want to know the following:
  - ✓ Location: where did the problem occur?
  - ✓ Timing: when did it occur?
  - ✓ Symptom: what was observed?
  - ✓ End result: which consequences resulted?
  - ✓ Mechanism: how did it occur?
  - ✓ Cause: why did it occur?
  - ✓ Severity: how much was the user affected?
  - ✓ Cost: how much did it cost?

How to Collect Data?

- Since software development is an intellectual activity, the collection of data requires human observation and reporting
- Managers, systems analysts, programmers, testers, and users must record raw data on forms (both manual or computer-based)
- This manual recording is subject to bias (deliberate or unconscious), error, omission, and delay

- Automatic data capture is, therefore, desirable, and sometimes essential, such as in recording the execution time of real-time software
- Unfortunately, in many instances, there is no alternative to manual data collection
- To ensure that the data are accurate and complete, we must plan our collection effort before we begin to measure and capture data
- Ideally we should:
  - ✓ Keep procedures simple;
  - ✓ Avoid unnecessary recording;
  - ✓ Train staff in the need to record data and in the procedures to be used;
  - ✓ Provide the results of data capture and analysis to the original providers promptly and in a useful form that will assist them in their work;
  - ✓ Validate all data collected at a central collection point
- The last point is especially important, as some studies show that half the data collected was corrupted in some way and therefore not useful for analysis
- Planning for data collection involves several steps
- First, you must decide which products to measure, based on your GQM analysis
- You need to measure several products that are used together, or you may measure one part or subsystem of a larger system
- The GQM analysis suggests which attributes and measures you would like to evaluate
- Once you are committed to a measurement program, you must decide exactly which attributes to measure and how indirect measures will be derived
- This will determine what raw data will be collected, and when
- The direct and indirect measures may be related by a measurement model, defining how metrics relate to one another
- Sometime, equations, graphs, or relationship diagrams are used to depict the ways in which metrics are calculated or related
- Once the set of metrics is clear, and the set of components to be measured has been identified, you must devise a scheme for identifying each entity involved in the measurement process
- That is, you must make clear how you will denote products, versions, installations, failures, faults, and other things on your data-collection forms
- This step enables you to proceed to *form design*, including only the necessary and relevant information on each form
- Finally, you must establish procedures for handling the forms, analyzing the data, and reporting the results
- Define who fills in what, when, and where, and describe clearly how the completed forms are to be processed
- In particular, set up a central collection point for data forms, and determine who is responsible for the data each step of the way
- If no one person or group has responsibility for a given step, the data collection and analysis process will stop, as each developer assumes that another is handling the data
- Analysis and feedback will ensure that the data are used, and useful results will motivate staff to record information

When to Collect Data

- It is clear that data-collection planning must begin when project planning begin, and that careful forms (both manual computer-based), design, and organization is needed to support good measurement
- The actual data collection takes place during many phases of development
- Some data can be collected at the start of the project, while other data collection is done later
- Some data are collected when reviews and inspections are conducted
- Data about faults found by inspections and tests should be recorded consistently so that the relative effectiveness of these activities can be determined
- Similarly, data about changes made to enhance the product, as opposed to corrected faults, can be collected as enhancements are performed
- In general, data should be collected at the beginning of the project to establish initial values, and then again as the initial values change to reflect activities and resources being studied
- It is essential that the data-collection activities become part of the regular development process
- If developers do not know when they are supposed to be measuring, the measurement will not occur
- Thus, it is helpful to compare a model of the normal development process with a list of desired measurements, and then map the measurements directly to the process model
- In this way, if something needs to be measured and there is no obvious place to collect the data, the process can be modified to enable measurement
- Likewise, if there is a process activity that is important and should be analyzed, the measurement list can be appended to address it

How to Store and Extract Data

- Raw software-engineering data should be stored on a database
- DBMS is an automated tool for organizing, storing, and retrieving data, and it has many advantages over manual record-keeping

References

- Software Metrics: A Rigorous & Practical Approach, by Norman E. Fenton and Shari L. Pfleeger, 2nd Edition, PWS Publishing Company, 1997(Chapter 4)

Lecture 43
Process Metrics

- Process metrics are those that can be used for improving the software development and maintenance processes
- Examples include the effectiveness of defect removal during development, the pattern of testing defect arrival, and the response time of the fix process
- Compared to end-product quality metrics, process quality metrics are less formally defined, and their practices vary greatly among software developers
- Some organizations pay little attention to process quality metrics, while others have well-established software metrics programs that cover various parameters in each phase of the development cycle

Process Metrics

- Defect density
  - ✓ Software defect density never follows the uniform distribution. It is an indication of defect injection in code or other software products
  - ✓ If a piece of code or a product has higher testing defects, either it is due to more effective testing or it is because of higher latent defects in the code
  - ✓ It is also possible that the more defects found during testing, the more defects will be found later
  - ✓ Defects per KLOC (or other denominator)
  - ✓ This simple metric of defect density is a good indicator of quality while the software is still under testing
  - ✓ It is especially useful to monitor subsequent releases of the same product within the same development organization
  - ✓ Therefore, release-to-release comparisons are not contaminated by other extraneous factors
- Defect arrival rate
  - ✓ It is the number of defects found during testing measured at regular intervals over some period of time
  - ✓ Rather than a single value, a set of values is associated with this metric
  - ✓ When plotted on a graph, the data may rise, indicating a positive defect arrival rate; it may stay flat, indicating a constant defect arrival rate; or decrease, indicating a negative defect arrival rate
  - ✓ Interpretation of the results of this metric can be very difficult
  - ✓ Intuitively, one might interpret a negative defect arrival rate to indicate that the product is improving since the number of new defects found is declining over time
  - ✓ To validate this interpretation, you must eliminate certain possible causes for the decline
  - ✓ For example, it could be that test effectiveness is declining over time

- ✓ In other words, the tests may only be effective at uncovering certain types of problems. Once those problems have been found, the tests are no longer effective
- ✓ Another possibility is that the test organization is understaffed and consequently is unable to adequately test the product between measurement intervals
- ✓ They focus their efforts during the first interval on performing stress tests that expose many problems, followed by executing system tests during the next interval where fewer problems are uncovered

- Test effectiveness
  - ✓ To measure test effectiveness, take the number of defects found by formal tests and divide by the total number of formal tests
    - $TE = D_n / T_n$
  - ✓ When calculated at regular intervals and plotted, test effectiveness can be observed over some period of time
  - ✓ If the graph rises over time, test effectiveness may be improving. On the other hand, if the graph is falling over time, test effectiveness may be waning
- Defects by phase
  - ✓ The defects by phase metric is a variation of the defect arrival rate metric
  - ✓ At the conclusion of each discreet phase of the development process, a count of the new defects is taken and plotted to observe a trend
  - ✓ It is much less expensive in terms of resources and reputation to eliminate defects early instead of fix late
  - ✓ At the conclusion of each discreet phase of the development process, graph is plotted to observe a trend
  - ✓ If the graph appears to be rising, you might infer that the methods used for defect detection and removal during the earlier phases are not effective since the rate at which new defects are being discovered is increasing
  - ✓ On the other hand, if the graph appears to be falling, you might conclude that early defect detection and removal is effective
  - ✓ Explain the snowball effect
- Defect removal effectiveness
  - ✓ Defect removal effectiveness can be calculated by dividing the number of defects removed prior to release by the sum of defects removed prior to release and the total number of defects that remain in the product at release. When multiplied by 100, this value can be expressed as a percentage:
    - $DRE = D_r/(D_r + D_t)$

Metrics for Software Maintenance Process

- When a software product has completed its development and is released to the market, it enters into the maintenance phase of its life cycle
- During this phase the defect arrivals by time interval and customer problem calls (which may or may not be defects) by time interval rate are the *de facto* metrics

- However, the number or problem arrivals is largely determined by the development process before the maintenance phase
- Not much can be done to alter the quality of the product during this phase. Therefore, the *de facto* metrics, while important, do not reflect the quality of software maintenance
- What can be done during maintenance phase is to fix the defects as soon as possible and with excellent fix quality
- Such actions, although still not able to improve the defect rate of the product, can improve customer satisfaction to a large extent

Metrics in Software Maintenance

- Defect backlog
  - ✓ The defect backlog metric is a count of the number of defects in the product following its release that require a repair
  - ✓ It is usually measured at regular intervals of time and plotted for trend analysis
  - ✓ By itself, this metric provides very little useful information
  - ✓ For example, what does a defect backlog count of 128 tells you?
  - ✓ Can you predict the impact of those defects on customers?
  - ✓ Can you estimate the time it would take to repair those defects?
  - ✓ Can you recommend changes to improve the development process?
  - ✓ A more useful way to represent the defect backlog is by defect severity
  - ✓ By calculating the defect backlog by severity level, you can begin to draw useful conclusions from your measurements
- Backlog management index
  - ✓ As the backlog is worked, new problems arrive that impact the net result of your team's efforts to reduce the backlog
  - ✓ If the number of new defects exceeds the number of defects closed over some period of time, your team is losing ground to the backlog. If, on the other hand, your team closes problems faster than new ones are opened, they are gaining ground
  - ✓ The backlog management index (BMI) is calculated by dividing the number of defects closed during some period of time by the number of new defects that arrived during that same period of time
    - $BMI = D_c / D_n$
  - ✓ If the result is greater than 1, your team is gaining ground; otherwise it is losing
  - ✓ When measurements are taken at regular intervals and plotted, a trend can be observed indicating the rate at which the backlog is growing or shrinking
- Fix response time
  - ✓ The fix response time metric is determined by calculating the average time it takes your team to fix a defect
  - ✓ It can be measured several different ways
  - ✓ In some cases, it is the elapsed time between the discovery of the defect and the development of an unverified fix

- ✓ In other cases, it is the elapsed time between the discovery and the development of verified fix
- ✓ A better alternative to this metric is fix response time by severity

- Percent delinquent fixes
  - ✓ A fix is delinquent if exceeds your fix response time criteria
  - ✓ For example, if you have established a maximum fix response time of 48 hours; then fix response times that exceed 48 hours are considered delinquent
  - ✓ To calculate the percent delinquent fixes; divide the number of delinquent fixes by the number of non-delinquent fixes and multiply by 100
    - PDF = $(F_d / F_n) * 100$
  - ✓ This metric is also measured better by severity since the consequences of having a high delinquent fixes for severe defects is typically much greater than for less severe or minor defects
- Defective fixes
  - ✓ A defect for which a fix has been prepared that later turns out to be defective or worse, creates one or more additional problems, is called a defective fix
  - ✓ The defective fixes metric is a count of the number of such fixes
  - ✓ To accurately measure the number of defective fixes, your organization must not only keep track of defects that have been closed and then reopened but must also keep track of new defects that were caused by a defect fix
  - ✓ This metric is also known as fix quality

Strategic Application of Software Metrics

- Sensitive usage of metrics data
- Process improvement through defect analysis
- Validation of best practices

Use the concept of public and private data to drive your decisions on how to collect and analyze data

Private Data

- Understanding of who owns what data and when it is appropriate to share the data with others
  - ✓ People prefer to keep defect data private
  - ✓ How do we analyze defects to improve processes rather than blame people?
  - ✓ Private data becomes public at well-defined handoffs or transitions
  - ✓ Accountability

Inspections

- Inspections are such transitions
- If inspections are properly run, these defects are less embarrassing than defects that are found later
- Data is no longer private
- Focus of accountability shifts from individual to team

Project's Private Data

- Time data
- Principle of "Information Hiding" is applied
- "Information Hiding" during project review

Public Data

- Metrics for public data
  - ✓ Calendar times
  - ✓ Defect rates
  - ✓ Project costs
  - ✓ Measure of functionality

Private and Public Data

Private (to individual)	Private (to project team) Public (to team members)	Public (to company)
Defect rates (by individual) Defect rates (by module) Defect rates (under development) Number of compiles	Defect rates (beyond individual) NCSS/module Estimated NCSS/module No. of re-inspections Defects/modules (pre-release)	Defect rates (by project) NCSS (by product) Effort (by project) Calendar times Defects/module (post-release) Effort/defect (average)

- Apply your best management sensitivity to the interpretation and use of software metrics data

Functional Management

- Don't allow anyone in your organization to use metrics to measure individuals
- Set clear goals and get your staff to help define metrics for success
- Understand the data that your people take pride in reporting; don't ever use it against them; don't ever even hint that you might
- Don't emphasize one metric to the exclusion of others
- Support your people when their reports are backed by data useful to the organization

Project Management

- Don't try to measure individuals
- Gain agreement with your team on the metrics that you will track, and define them in a project plan
- Provide regular feedback to the team about the data they help collect
- Know the strategic focus of your organization and emphasize metrics that support the strategy in your reports

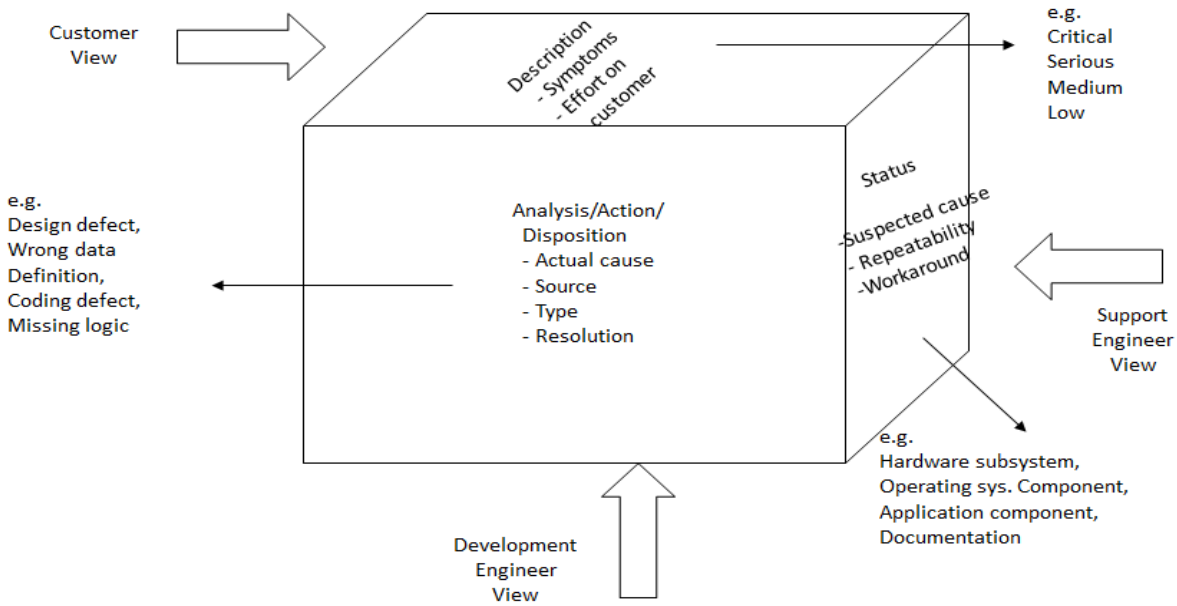
Project Team

- Do your best to report accurate, timely data
- Help your managers to focus project data on improving your processes
- Don't use metrics data to brag about how good you are or you will encourage others to use other data to show the opposite
- Use software failure analysis to help drive process improvement decisions and to measure the impact of those decisions

Dissecting Software Defects

- Customer's view of defects
- Development engineer's view of defects
- Support engineer's view of defects

Defect Cube



Key Questions to Learn from Mistakes

- What development or maintenance process failed?
- How often do such failures occur?
- How expensive is it to fix such failures?
- Which components are most subject to failures?
- What process change will detect or eliminate these failures?
- Evaluate all software process changes in terms of measurable ROI

References

- Metrics and Models in Software Quality Engineering by Stephan H. Kan (Chapter 4.2-4.4) [pp.99-117]
- Practical Software Metrics for Project Management and Process Improvement by Robert Grady (Part II)

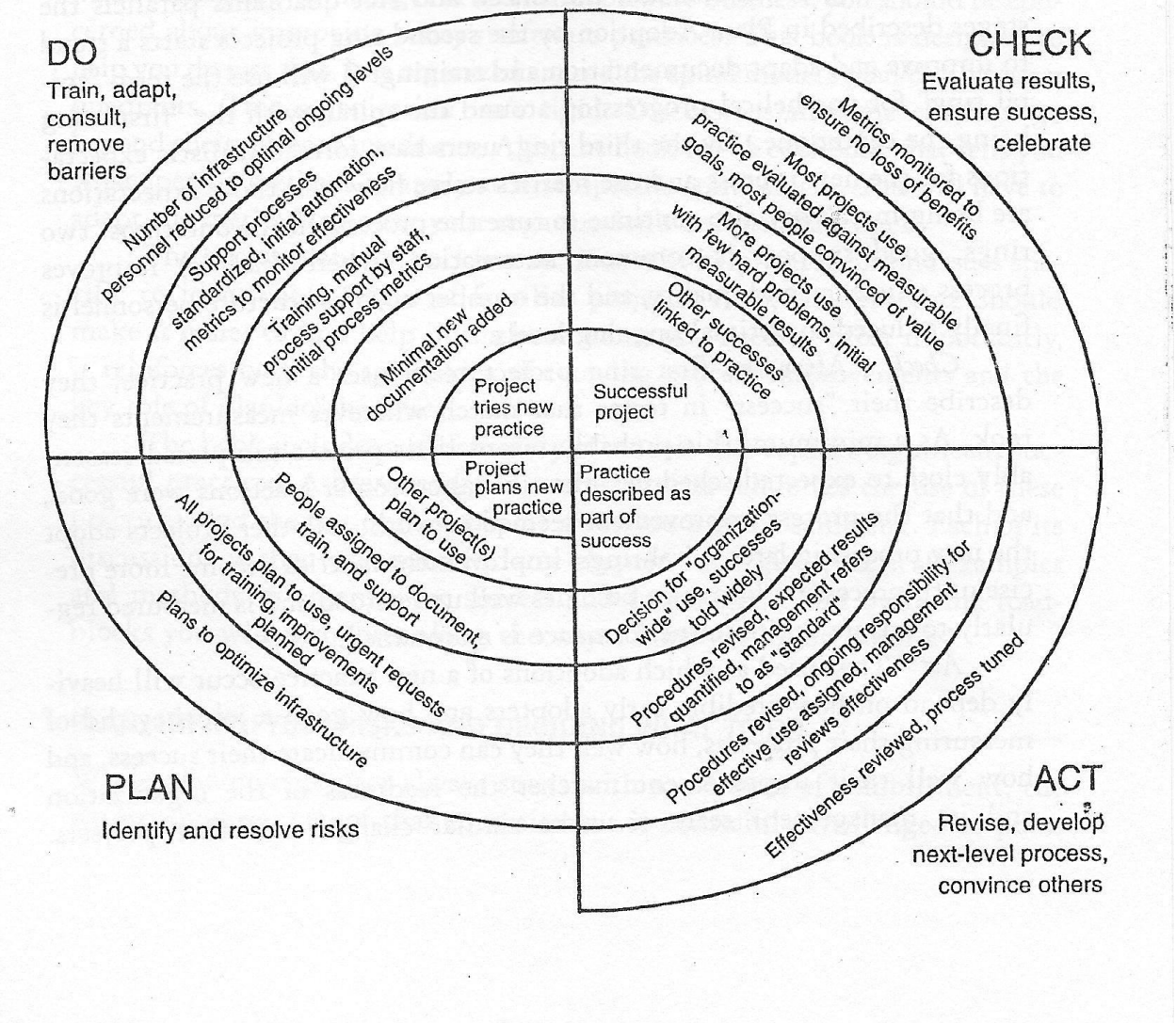
Lecture 44

Software Process Improvement:

Your Mileage May Vary

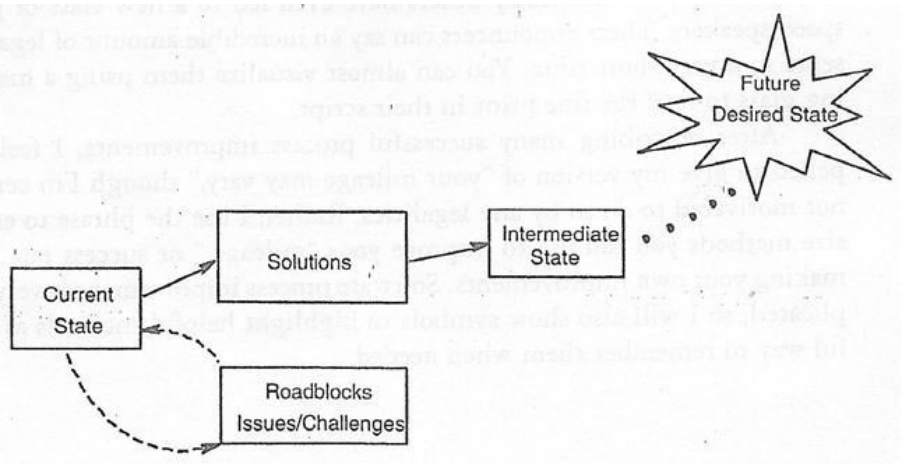
- If you own a car or a motorbike, you are all too well familiar with the concept of mileage
- Your mileage may vary
- Let's see if we can use this with process improvement efforts and programs
- Here, the mileage may refer to as "success rate"
- The significance of a spiral model for process improvement is very clear, as it shows that software process improvement is an unending journey
- While each improvement can be viewed as a tactical project, you must follow through with others to remain competitive

Spiral Model for Process-Improvement Adoption



- If we were to unwrap the spiral into a linear model, here's how it will look like

The Software-Improvement Journey



- We start moving toward a desired future state from a current state
- Almost always, we encounter roadblocks, but gradually we move ahead with solutions
- An intermediate state represents an acknowledged success. But we do not stop here, so the dots represent other projects, or later progressions from current to intermediate states for other spiral loops

Define Process Improvement in Terms of the Strategic Future

- Your starting point is very important, which is where you want to go
- It represents your organization's strategic future, so the first way to improve your mileage (chances for, and speed of, success) is to define that clearly
- The most succinct definition of a future desired state is a vision

Improving Mileage #1

- Understand Business Needs Well
  - ✓ A process that adds structure and commitment to your organization's vision is the core-competence planning process
  - ✓ Its major contribution is that it elevates process-improvement thinking to a strategic level
  - ✓ It also expands top management's competitive view from a traditional product focus to include key abilities the organization needs to create those products
  - ✓ Ideally, the core-competence planning process is as strong as your business-planning process
  - ✓ Just as business planning isn't always ideal, core-competence planning isn't either
  - ✓ Think of core-competence planning as a spiral process also
- Get a Clear Picture of Your Current State
  - ✓ Once you have a good picture of your desired future, there are thee key methods by which to gain a similar understanding of your current state

Improving Mileage #2

- Understand Your Software Development Costs and Their Leverage Points
  - ✓ The software-improvement investment model gives you a simple way to model major development, rework, and knowledge recovery costs
  - ✓ It helps to
    - Give your management a better understanding of how software developers work,
    - Change people's opinions of the relative importance of proposed improvements, and
    - Set realistic expectations for proposed cost savings
  - ✓ You get an even better picture of improvement opportunities when you also analyze your major defect sources
  - ✓ This information is especially powerful when you combine it with the investment-model information. It
    - Further clarifies costs
    - Gives you an excellent way to evaluate the potential of proposed improvements, and
    - Gives you an effective way to track and communicate the successful changes that you make

Improving Mileage #3

- Understand Your Organization's Readiness for Change
  - ✓ One way to get a clear picture of your current state is to do an assessment that is well matched to your organization's needs
  - ✓ Pick an assessment approach that matches your business needs, level of sponsorship, scope of needed change, and your best guess of organizational readiness
  - ✓ An assessment does three key things for you
    - Gives you a reference point of current strengths and opportunities,
    - Motivates people to change, and
    - Models how to approach improvements
  - ✓ It isn't easy to get a clear picture of an organization's readiness for change
  - ✓ Assessments help us in this regard
- Avoid and Minimize Potential Roadblocks
  - ✓ Process-improvement projects face complex sociological issues not normally faced in product projects
  - ✓ Extra analysis and planning can go a long way in helping to avoid roadblocks

Improving Mileage #4

- Understand Your Business and Organizational Forces
  - ✓ A force-field analysis of management commitment gives you the best snapshot of potential varying "mileage"
  - ✓ The force-field analysis helps you steer towards simple risk-minimizing strategies

Improving Mileage #5

- Understand the Origins of Resistance to Change
  - ✓ People develop reasonable beliefs based on experiences. Proposed changes can be dissimilar enough to their past experiences that when they try to draw analogies, the analogies won't reflect probable outcomes
  - ✓ One way to avoid being judgmental about such derived beliefs is to think of their resistance as a "standard reason" not to do things
  - ✓ This can help you open your viewpoint so you can think about both your and their ladders of inference
  - ✓ Resulting discussions will help you to create ways to remove roadblocks

Improving Mileage #6

- Strengthen Plans
  - ✓ More complex projects require better planning and risk management, and even the simplest process improvement isn't easy
  - ✓ Thus, you especially need to plan to motivate people and to optimize their working environment
  - ✓ Process improvements also more fully use the PDCA cycle
  - ✓ This requires activities that many team members aren't experienced with. These added risks mandate at least a minimal, written, process-improvement plan

Improving Mileage #7

- Communicate Solutions to Maximize Success
  - ✓ You will need to effectively describe your project and its benefits often
  - ✓ A good approach is to imagine its completion early and create a storyboard that envisions how that will appear
  - ✓ This is a powerful way to describe your project clearly as part of the organization's future desired state
  - ✓ Another useful tool that ensures strong ties to business goals is the goal/question/metric (GQM) process
  - ✓ These ties lead to more easily accepted communications

Improving Mileage #8

- Set Reasonable Expectations for Results
  - ✓ The first part of setting reasonable improvement expectations is to calculate cost savings. This is a double-edged sword
  - ✓ If you don't do these estimates or if the way you communicate them makes seem low, your project is unlikely to be approved
  - ✓ If you estimate too high, your project may be cancelled, because you aren't believed
  - ✓ Also, if your estimates are high and you are allowed to finish, your project may still be considered a failure

- ✓ Doing a cost-savings analysis early in your planning forces you document assumptions. This analysis enables you to make your intermediate steps and improvement gains clearer
- ✓ These will generally be both believable and high enough

Improving Mileage #9

- Frame Expectations for Different Audiences
  - ✓ A key step in gaining and holding project sponsorship is to translate your cost savings into benefits

Improving Mileage #10

- Reinforce Success with Measured Results
  - ✓ There is nothing like quantified intermediate results to eliminate doubts and solidify support
  - ✓ By imagining how your results will look during your initial planning/storyboarding, you will ensure that you collect the right data to report successes confidently

Your Improvement Future

- The spiral model emphasizes looking toward a future desired state
- As we continue to mature software engineering, improvement techniques will get even better, and we will understand our sociology better

References

- Successful Software Process Improvement by Robert R. Grady, Chapter 14

Lecture 45
Review from Mid-Term to Final

Software Problems

- Technical
  - ✓ Need a technical solution
  - ✓ Easy to understand and solve
- Systemic
  - ✓ Need a process/management solution
  - ✓ Difficult to understand and solve

Solution to Software Problems

- Treat the software task as a process that can be controlled, measured, and improved
- Relate the required tasks, tools, methods with skill, training, and motivation of people involved

Software Processes

- Software engineering, as a discipline, has many processes
- These processes help in performing different software engineering activities in an organized manner
- The software process is the set of tools, methods, and practices that people use to produce (develop and maintain) software and *associated products* , e.g., project plans, design documents, code, test cases, and user manuals
- Where are the people? Aren't they the most important part of an organization?
- Characteristics of Software Processes
  - ✓ Requires creativity
  - ✓ Interactions between a wide range of different people
  - ✓ Engineering judgment
  - ✓ Background knowledge
  - ✓ Experience

Introduction to TSP

Why Projects Fail

- Reasons
  - ✓ Seldom due to technical reasons
  - ✓ People problems
  - ✓ Excessive schedule pressure
- Solution
  - ✓ People problems can be solved with the help of good communication and good teams
  - ✓ Good teams guide in handling pressure
- After this basic introduction about teams, let's talk about Team Software Process or TSP
- Development of Team Software Process (TSP) is another important step in software process improvement, after the development of CMM and PSP

- The TSP provides a disciplined context for engineering work
- The principal motivator for the development of the TSP was the conviction that engineering teams can do extraordinary work, but only if they are properly formed, suitably trained, staffed with skilled members, and effectively led
- The objective of the TSP is to build and guide such teams
- A team is more than just a group of people who happen to work together
- Teamwork takes practice and it involves special skills
- Teams require common processes; they need agreed upon goals; and they need effective guidance and leadership

What is a Team?

- A team consists of at least two people
- The members are working toward a common goal
- Each person has a specific assigned role
- Completion of the mission requires some form of dependency among the group members
- The four parts of this definition of a team are all important
- For example, it is obvious that a team must have more than one member, and the need for common goals is also generally accepted
- However, it is not as obvious why team members must have roles?
- Roles provide a sense of ownership and belonging
- They help to guide team members on how to do their jobs; they prevent conflicts, duplicate work, and wasted effort; and they provide the members with a degree of control over their working environment
- Such a sense of control is a fundamental requirement for motivated and energetic team members
- Interdependence is also an important element of teamwork. It means that each team member depends to some degree on the performance of the other members
- Interdependence improves individual performance because the members can help and support each other
    - ✓ E.g., Design teams
- Team performance is further enhanced by the social support of membership. Human beings are social animals and few people like to work entirely by themselves—at least not for very long
- Because of the social context of teams, the members will generally make a special effort to meet their obligations to the rest of the team
- Through mutual support and interdependence, teams become more than just the sum of their individual members
- To be effective, teams must be properly skilled and be able to work as cohesive units
- Effective teams have certain common characteristics
- Innovation is more than just thinking up bright ideas; it requires creativity and a lot of hard work. Just about every engineering task is part of an innovative endeavor
- Innovative teams must have skilled and capable people who are highly motivated. They must be creative, flexible, and discipline

- The best team structure depends on the management style of your organization, the number of people who will populate the team and their skill levels, and the overall problem difficulty
- The performance of a team is inversely proportional to the amount of communication that must be conducted
- The length of time that the team will "live together" affects team morale
- Constantine suggests four organizational paradigms for software engineering teams:
  - ✓ Closed paradigm – traditional, good for déjà vu projects
  - ✓ Random paradigm – loose, innovative projects
  - ✓ Open paradigm – mix of traditional and random
  - ✓ Synchronous paradigm – natural compartmentalization of problem

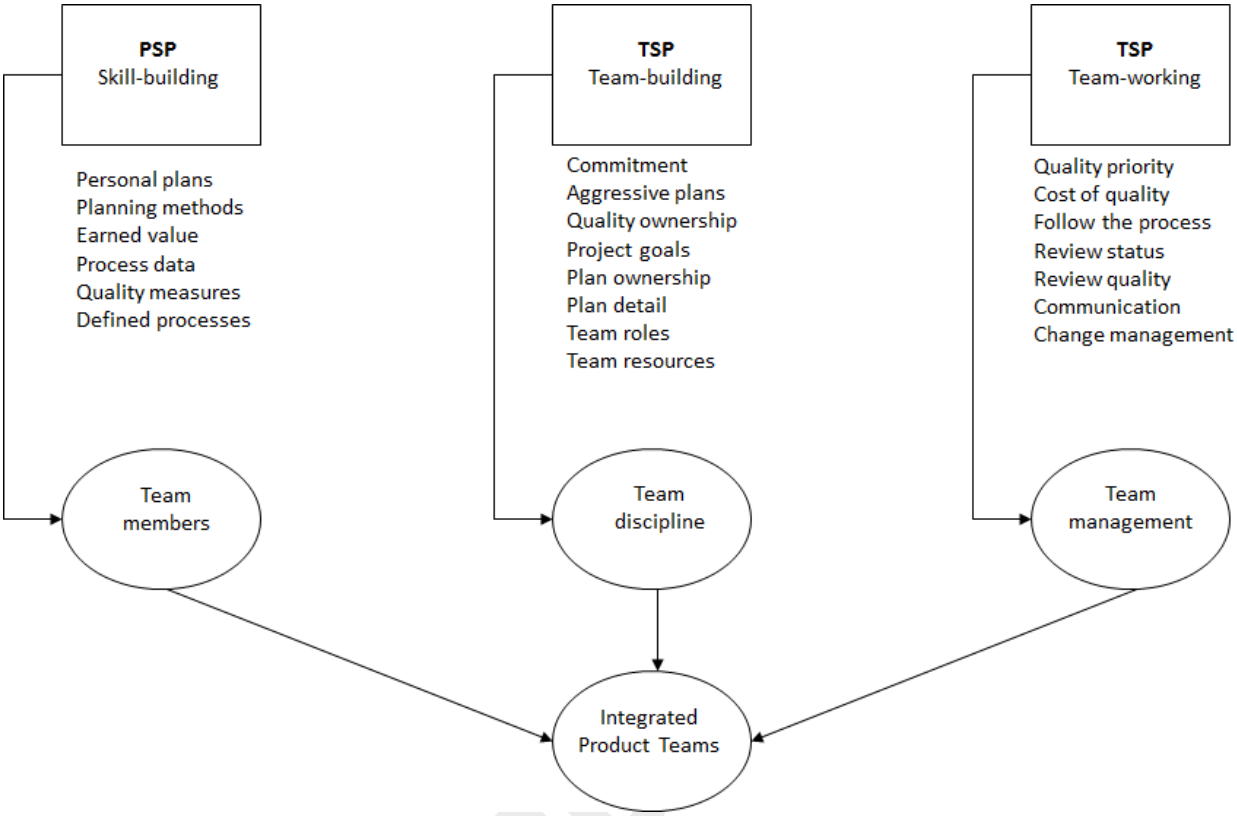
High-Performance Team

- Team members must have trust in one another
- The distribution of skills must be appropriate to the problem
- Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained
- A jelled team is a group of people so strongly knit that the whole is greater than the sum of the parts…
- Once a team begins to jell, the probability of success goes way up. The team can become unstoppable, a juggernaut for success

Factors that Foster Toxic Team Environment

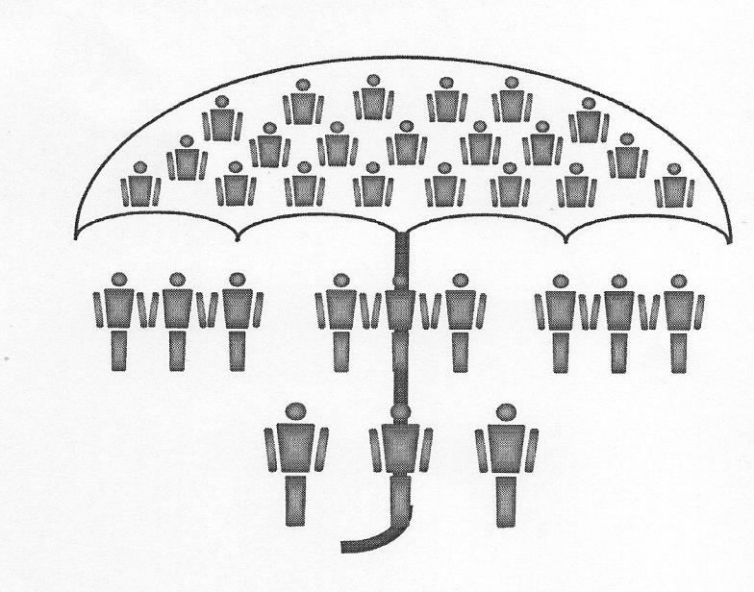
- A frenzied work atmosphere in which team members waste energy and lose focus on the objectives of the work to be performed
- High frustration caused by personal, business, or technological factors that causes friction among team members
- "Fragmented or poorly coordinated procedures" or a poorly defined or improperly chosen process model that becomes a roadblock to accomplishment
- Unclear definition of roles resulting in a lack of accountability and resultant finger-pointing
- "Continuous and repeated exposure to failure" that leads to a loss of confidence and a lowering of morale

Principle Elements of TSP



- Before team members can participate on a TSP team, they must know how to do disciplined work
- Training in the Personal Software Process (PSP) is required to provide engineers with the knowledge and skills to use the TSP
- PSP training includes learning how to make detailed plans, gathering and using process data, developing earned value plans, using earned value to track a project, measuring and managing product quality, and defining and using operational processes
- Engineers must be trained in these skills before they can participate in TSP team building or follow the defined TSP process
- The Capability Maturity Model (CMM) or Capability Maturity Model Integration (CMMI) provides the overall improvement framework needed for effective engineering work
- The Personal Software Process (PSP) provides the engineering disciplines that engineers need for consistently using a defined, planned, and measured process

Tying Process Improvement Efforts



- The TSP couples the principles of integrated product teams with the PSP and CMM/CMMI methods to produce effective teams
- In essence, the CMM/CMMI and PSP provide the context and skills for effective engineering while the TSP guides engineers in actually doing the work
- Thus, the TSP capitalizes on the preparation provided by the PSP and CMM/CMMI, while also providing explicit guidance on how to do the work

TSP Teambuilding Principles

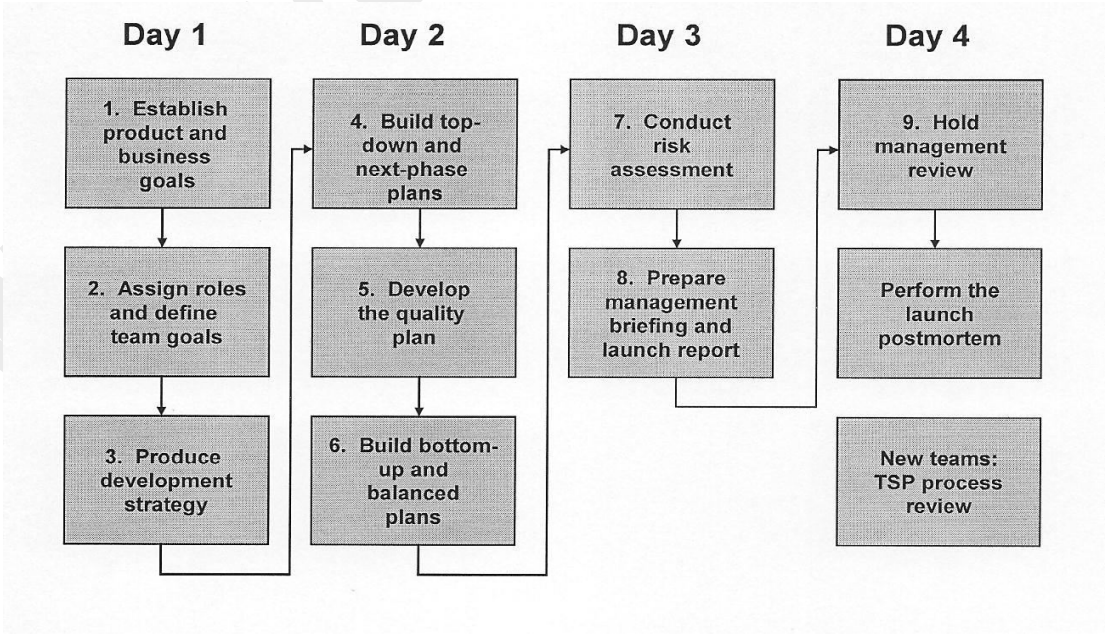
- The team members establish common goals and defined roles
- The team develops an agreed-upon strategy
- The team members define a common process for their work
- All team members participate in producing the plan, and each member knows his or her personal role in that plan
- The team negotiates the plan with management
- Management reviews and accepts the negotiated plan
- The team members do the job in the way that they have planned to do it
- The team members communicate freely and often
- The team forms a cohesive group: the members cooperate, and they are all committed to meeting the goal
- The engineers know their status, get feedback on their work, and have leadership that sustains their motivation
- Effective team formation requires that the members truly understand what they are supposed to do, agree on how to do the job, and believe that their plan is achievable
- These conditions can all be established by involving the engineers in producing their own plans

- While all these conditions are necessary for effective teamwork, the specific ways for establishing these conditions are not obvious
- The TSP provides the explicit guidance that organizations need to build effective engineering teams
- While there are many ways to build teams, they all require that the individuals work together to accomplish some demanding task
- In the TSP, this demanding team-building task is a four- day planning process that is called the team launch
- In a launch, all the team members develop the strategy, process, and plan for doing their project. After completing the launch, the team follows its own defined process to do the job

TSP Process

- TSP processes are
  - ✓ Launching a team project
  - ✓ Development strategy
  - ✓ Development plan
  - ✓ Design process
  - ✓ Implementation process
  - ✓ Test plan
  - ✓ Postmortem
- The process of forming team and building team does not happen by accident, and it takes time
- Teams need to establish their working relationships, determine member roles, and agree on goals
- The launch is the first step

The TSP Launch Process



- Teams generally need professional guidance to properly complete the launch process. This guidance is provided by a trained launch coach who leads the team through the launch process
- While the TSP scripts provide essential guidance, every team has unique problems and issues, so a simple process cannot possibly provide all the material needed to guide an inexperienced team through the launch process
- Unless teams are very experienced and have a team leader who has completed several TSP projects, they generally need the support of a trained launch coach

Introduction to Measurements

- Software measurement, once an obscure and esoteric specialty has become essential to good software engineering
- Many of the best software developers measure characteristics of the software to get some sense of whether the requirements are consistent and complete, whether the code is ready to be tested
- Effective project managers measure attributes of process and product to be able to tell when the software will be ready for delivery and whether the budget will be exceeded
- Informed customers measure aspects of the final product to determine if it meets the requirements and is of sufficient quality
- Measurement lies at the heart of many systems that govern our lives
- Economic measurements determine price and pay increases
- Measurements in radar systems enable us to detect aircraft when direct vision is obscured
- Medical system measurements enable doctors to diagnose specific illnesses
- Measurements in atmospheric systems are the basis for whether prediction
- Without measurement, technology cannot function. But measurement is not solely the domain of professional technologists. Each of *us* uses it in every day life
- Examples of Measurements
    - ✓ Prices acts as a measure of value of an item in a shop
    - ✓ We use height, weight, and size measurements to ensure that our clothing will fit properly
    - ✓ When making a journey, we calculate distance, choose our route, measure our speed, and predict when we will arrive at our destination (and perhaps when we need to refuel)
    - ✓ So measurement helps us to understand our world, interact with our surroundings and improve our lives