SS ZG622: Software Project Management (Lecture #10)

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

T V Rao, BITS-Pilani Off-campus Centre, Hyderabad

Text Books

T1: Bob Hughes, Mike Cotterel, and Rajib Mall, Software Project Management, 5th Edition, McGraw Hill, 2011

T2: Pressman, R.S. Software Engineering : A Practitioner's Approach, 7th Edition, McGraw Hill, 2010

R1: Sommerville, I., Software Engineering, Pearson Education, 9th Ed., 2010

R2: Capers Jones., Software Engineering Best Practices, TMH ©2010

R3: Robert K. Wysocki, Effective Software Project Management, John Wiley & Sons © 2006

R4: George Stepanek, Software Project Secrets : Why Software Projects Fail, Apress ©2012

R5: A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Fifth Edition by Project Management Institute Project Management Institute © 2013

R6: Jake Kouns and Daniel Minoli, Information Technology Risk Management in Enterprise Environments. John Wiley & Sons © 2010

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

L10: Sotware Quality –

Achieving Quality - SQA, Reviews, Several Approaches

Source Courtesy: Some of the contents of this PPT are sourced from materials provided by publishers of prescribed books

Achieving Software Quality

Software Engineering Methods

- Understand the problem to be solved
- Create design that conforms to the problem
- Design and Software exhibit quality dimensions

Project Management Techniques

- Use estimation for achievable delivery dates
- Understand schedule dependencies and avoid short cuts
- Conduct risk planning

Achieving Software Quality

Quality Control

- Reviews
- Inspection
- Testing
- Measurements and Feedback

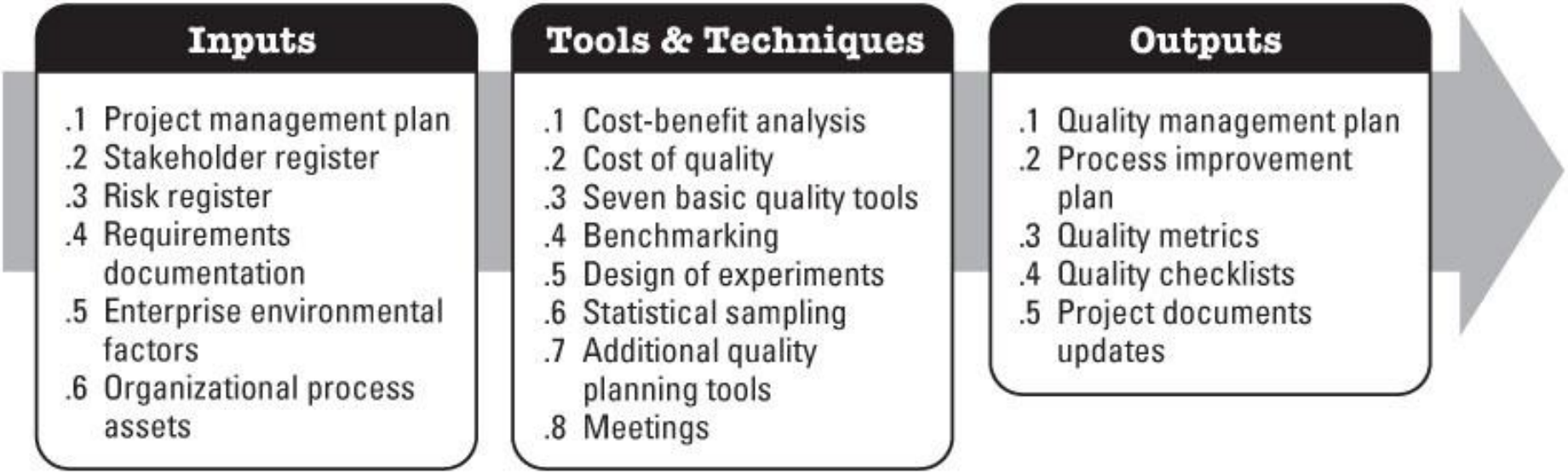
Quality Assurance

- Establish infrastructure for software engineering
- Audit effectiveness and completeness of quality control

Project Quality Management (PMBOK)

- **Plan Quality Management**—The process of identifying quality requirements and/or standards for the project and its deliverables and documenting how the project will demonstrate compliance with quality requirements.
- **Perform Quality Assurance**—The process of auditing the quality requirements and the results from quality control measurements to ensure that appropriate quality standards and operational definitions are used.
- **Control Quality**—The process of monitoring and recording results of executing the quality activities to assess performance and recommend necessary changes.

Project Quality Management (PMBOK)



Plan Quality Management

Project Quality Management (PMBOK)

Tools & Techniques

- **Cost-Benefit Analysis**
  - The primary benefits of meeting quality requirements include less rework, higher productivity, lower costs, increased stakeholder satisfaction, and increased profitability. A cost-benefit analysis for each quality activity compares the cost of the quality step to the expected benefit
- **Cost of Quality**
  - Cost of quality includes all costs incurred over the life of the product by investment in preventing nonconformance to requirements, appraising the product or service for conformance to requirements, and failing to meet requirements (rework). Failure costs are often categorized into internal (found by the project) and external (found by the customer).
- **Benchmarking**
  - Benchmarking involves comparing actual or planned project practices to those of comparable projects to identify best practices, generate ideas for improvement, and provide a basis for measuring performance
- **Design of Experiments**
  - Design of experiments (DOE) is a statistical method for identifying which factors may influence specific variables of a product or process under development

Project Quality Management (PMBOK)

Tools & Techniques

- **Seven Basic Quality Tools**

The seven basic quality tools, also known in the industry as 7QC Tools, are used to solve quality-related problems

- *Cause-and-effect diagrams*, (aka fishbone diagrams or Ishikawa diagrams) The problem statement placed at the head of the fishbone is used as a starting point to trace the problem's source back to its actionable root cause
- *Flowcharts*, display the sequence of steps and the branching possibilities that exist for a process that transforms one or more inputs into one or more outputs. Flowcharts prove useful in understanding and estimating the cost of quality in a process
- *Checksheets*, (aka tally sheets) used as a checklist when gathering data
- *Control charts*, are used to determine whether or not a process is stable or has predictable performance; control charts may also be used to monitor cost and schedule variances, volume, and frequency of scope changes etc.
- *Pareto diagrams*, identify the vital few sources causing most effects
- *Histograms*, *Scatter diagrams*

Project Quality Management (PMBOK)

Perform Quality Assurance

Project Quality Management (PMBOK)

Control Quality

innovate achieve lead

Reviews in Software Projects

Reviews

... there is no particular reason

why your friend and colleague

cannot also be your sternest critic.

Jerry Weinberg, Author of The Psychology of Computer Programming

What Are Reviews?

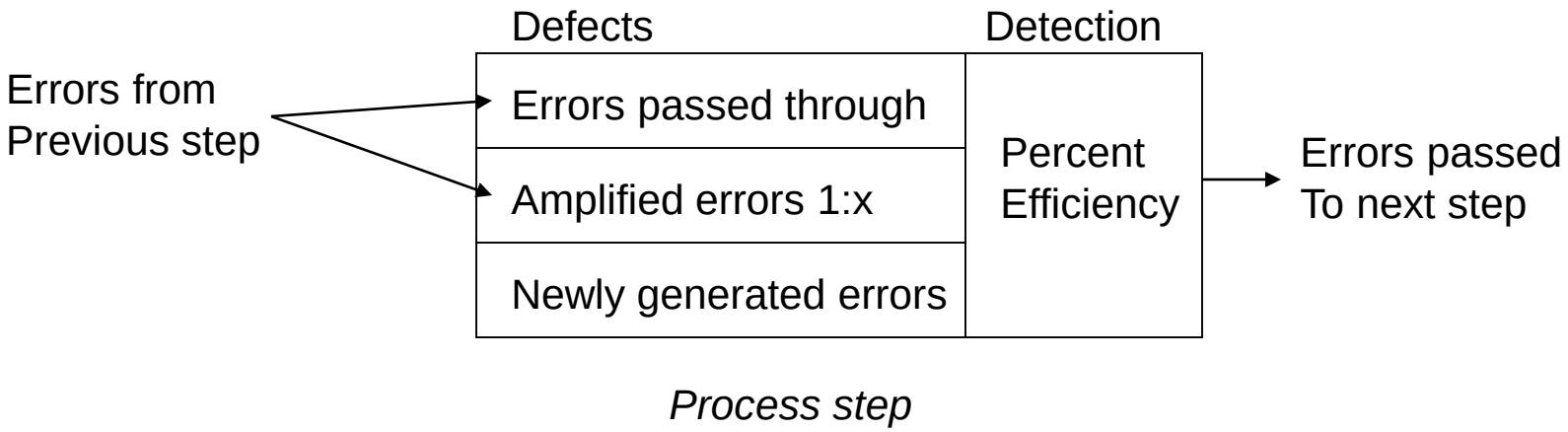
- a meeting conducted by technical people for technical people
- a technical assessment of a work product created during the software engineering process
- a software quality assurance mechanism
- a training ground

What Reviews Are Not

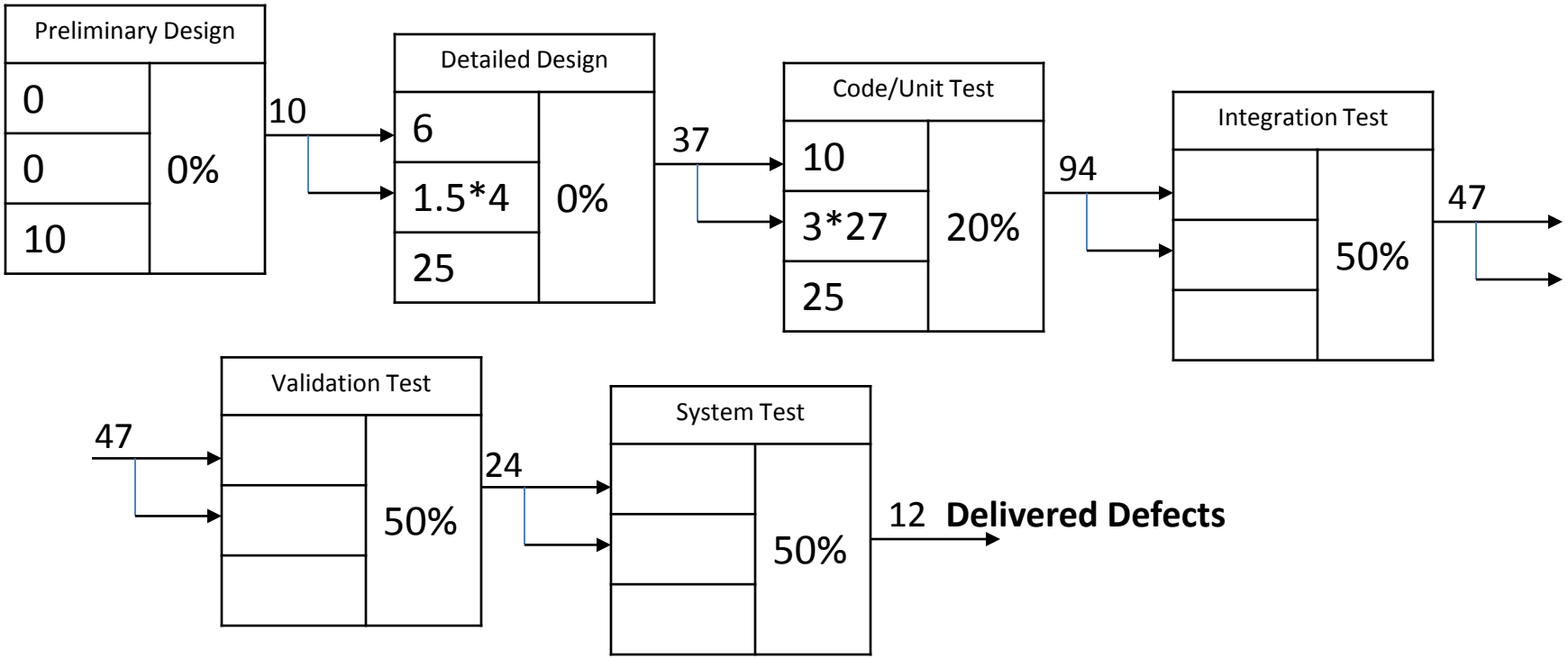
- A project summary or progress assessment
- A meeting intended solely to impart information
- A mechanism for political or personal reprisal!

Defect Amplification

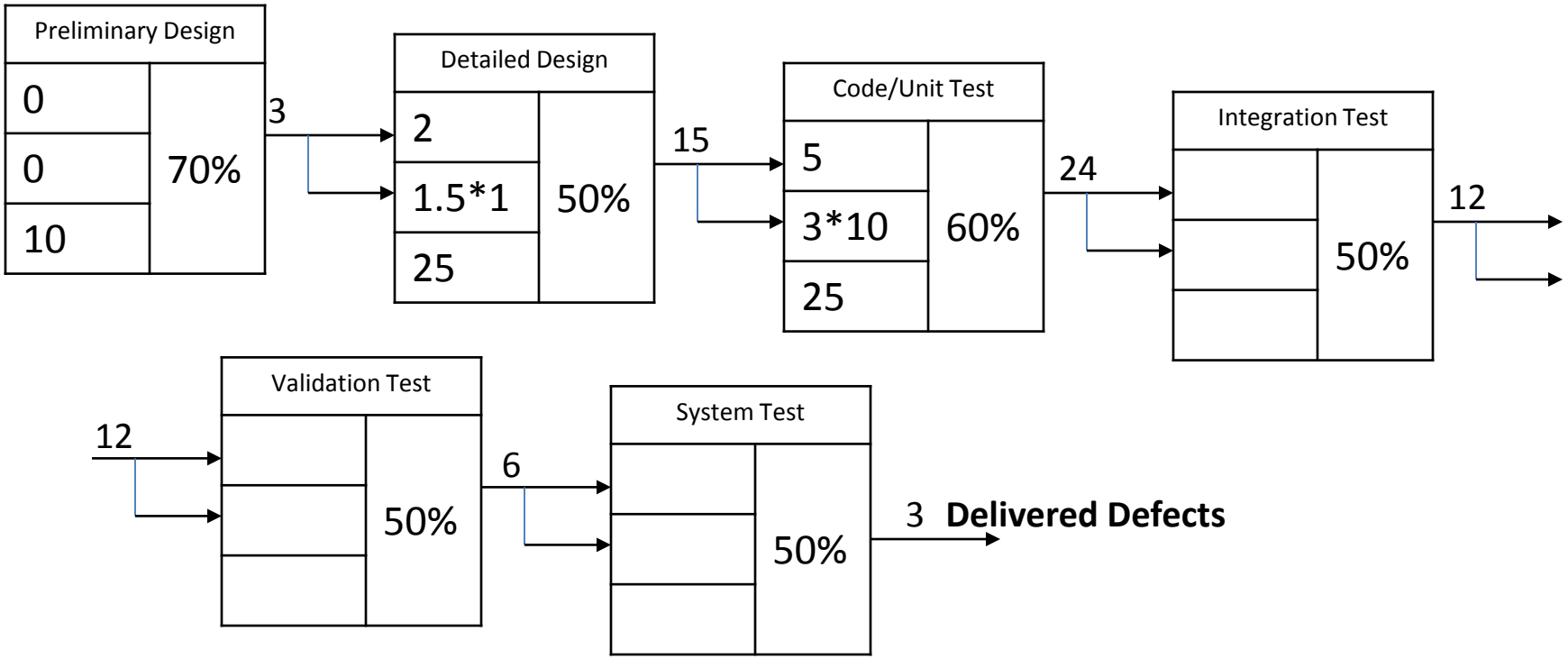
- A *defect amplification model* (proposed by IBM) can be used to illustrate the generation and detection of errors during the design and code generation actions of a software process.



Defect Amplification – No/Ineffective Reviews



Defect Amplification – Effective Reviews



Defect Amplification

- In the example provided by Pressman 7/e in Section 15.2,
  - a software process that does NOT include reviews,
    - yields 94 errors at the beginning of testing and
    - Releases 12 latent defects to the field
  - a software process that does include reviews,
    - yields 24 errors at the beginning of testing and
    - releases 3 latent defects to the field
  - A cost analysis(in next slide) indicates that the process with NO reviews costs approximately 3 times more than the process with reviews, taking the cost of correcting the latent defects into account

Defect Removal Costs Analysis

Assuming cost of defect removal to be

- 1.5 units for design
- 6.5 units before test
- 15 units during tests
- 67 units after release

- Cost in 1st case (with no/ineffective reviews) = 67*12+15(12+23+47)+6.5*22+1.5*0=2177 units
- Cost in 2nd case (with effective reviews) = 67*3+15(3+6+12)+6.5*36+1.5*(15+7)=783 units

Review Metrics

- *Preparation effort,* E_p—the effort (in person-hours) required to review a work product prior to the actual review meeting
- *Assessment effort,* E_a— the effort (in person-hours) that is expending during the actual review
- *Rework effort,* E_r— the effort (in person-hours) that is dedicated to the correction of those errors uncovered during the review
- *Work product size, WPS*—a measure of the size of the work product that has been reviewed (e.g., the number of UML models, or the number of document pages, or the number of lines of code)
- *Minor errors found,* Err_{minor}—the number of errors found that can be categorized as minor (requiring less than some pre-specified effort to correct)
- *Major errors found,* Err_{major}— the number of errors found that can be categorized as major (requiring more than some pre-specified effort to correct)

Review Metrics

- The total review effort and the total number of errors discovered are defined as:
  - $E_{review} = E_p + E_a + E_r$
  - $Err_{tot} = Err_{minor} + Err_{major}$
- *Defect density* represents the errors found per unit of work product reviewed.
  - Defect density = Err_{tot} / WPS

Example Metrics

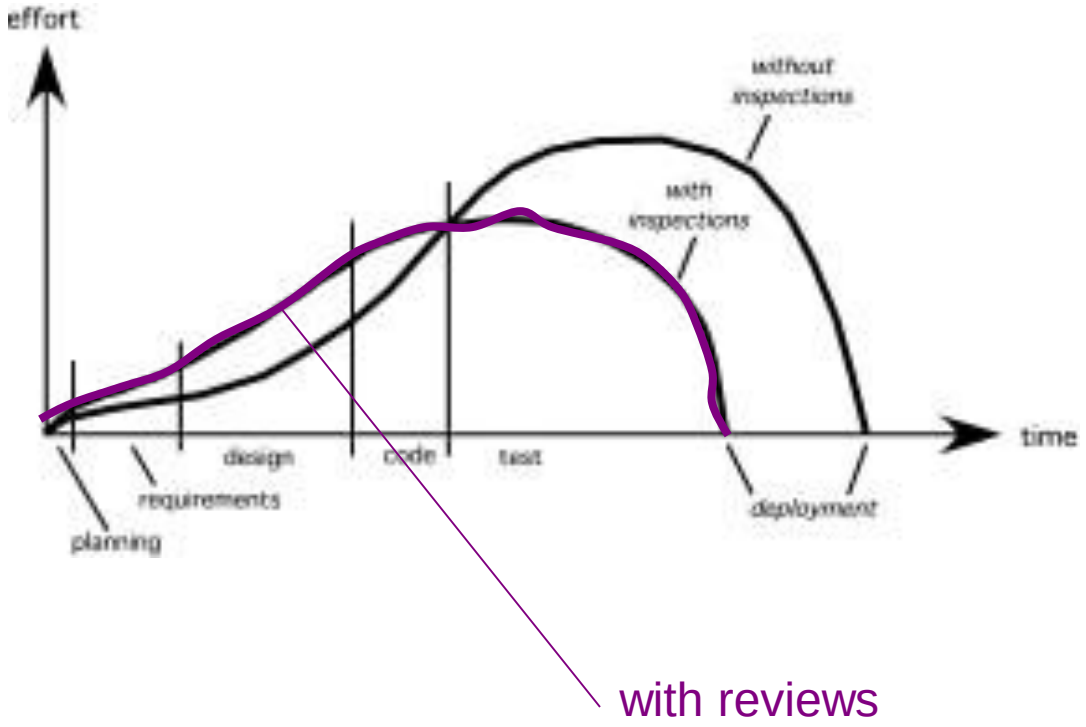
- If past history indicates that
  - the average defect density for a requirements model is 0.6 errors per page, and a new requirement model is 32 pages long,
  - a rough estimate suggests that your software team will find about 19 or 20 errors during the review of the document.
  - If you find only 6 errors, you've done an extremely good job in developing the requirements model *or* your review approach was not thorough enough.

Example Metrics

- The effort required to correct a minor model error (immediately after the review) was found to require 4 person-hours.
- The effort required for a major requirement error was found to be 18 person-hours.
- Examining the review data collected, you find that minor errors occur about 6 times more frequently than major errors. Therefore, you can estimate that the average effort to find and correct a requirements error during review is about 6 person-hours.
- Requirements related errors uncovered during testing require an average of 45 person-hours to find and correct. Using the averages noted, we get:
- Effort saved per error = $E_{testing} - E_{reviews}$
- $45 - 6 = 39$ person-hours/error
- Since 22 errors were found during the review of the requirements model, a saving of about 860 person-hours of testing effort would be achieved. And that's just for requirements-related errors.

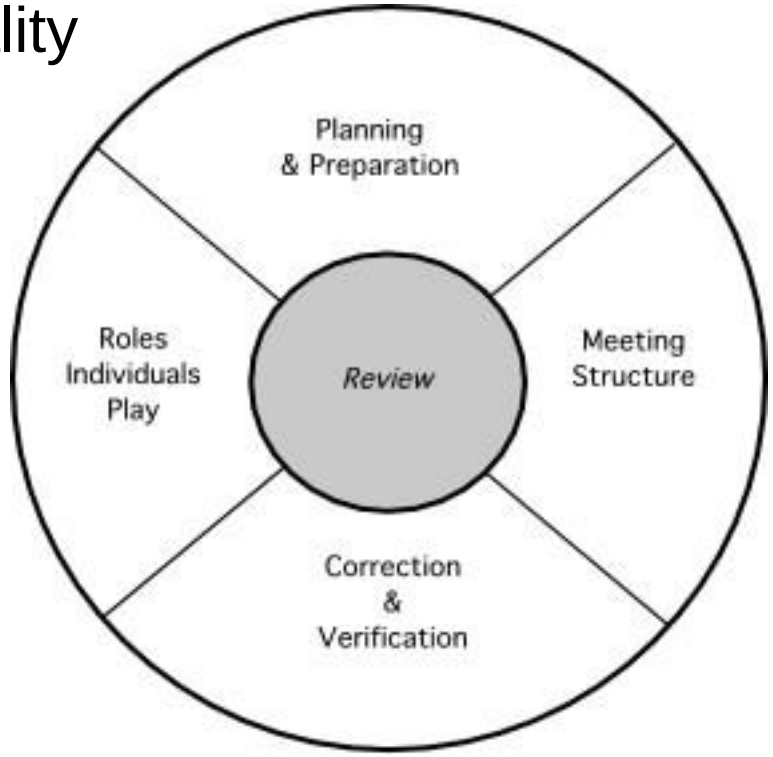
Overall benefits of Reviews

- Effort expended with and without reviews



Formal Review Model

Dimensions that
Contribute to formality



- Laitenberger

Informal Reviews

- Informal reviews include:
  - a simple desk check of a software engineering work product with a colleague
  - a casual meeting (involving more than 2 people) for the purpose of reviewing a work product, or
  - the review-oriented aspects of pair programming
- *pair programming* encourages continuous review as a work product (design or code) is created.
  - The benefit is immediate discovery of errors and better work product quality as a consequence.

Formal Technical Reviews

- The objectives of an FTR are:
  - to 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
  - to make projects more manageable
- The FTR is actually a class of reviews that includes *walkthroughs* and *inspections.*

The Review Meeting

- Between three and five people (typically) should be involved in the review.
- Advance preparation should occur but should require no more than two hours of work for each person.
- The duration of the review meeting should be less than two hours.
- Focus is on a work product (e.g., a portion of a requirements model, a detailed component design, source code for a component)

Players at Review

- *Producer*—the individual who has developed the work product
  - informs the project leader that the work product is complete and that a review is required
- *Review leader*—evaluates the product for readiness, generates copies of product materials, and distributes them to two or three *reviewers* for advance preparation.
- *Reviewer(s)*—expected to spend between one and two hours reviewing the product, making notes, and otherwise becoming familiar with the work.
- *Recorder*—reviewer who records (in writing) all important issues raised during the review.

Etiquette for a Review

- *Review the product, not the producer.*
- *Set an agenda and maintain it.*
- *Limit debate and rebuttal.*
- *Enunciate problem areas, but don't attempt to solve every problem noted.*
- *Take written notes.*
- *Limit the number of participants and insist upon advance preparation.*
- *Develop a checklist for each product that is likely to be reviewed.*
- *Allocate resources and schedule time for FTRs.*
- *Conduct meaningful training for all reviewers.*
- *Review your early reviews.*

Review Options Matrix

	IPR*	WT	IN	RRR
trained leader	no	yes	yes	yes
agenda established	maybe	yes	yes	yes
reviewers prepare in advance	maybe	yes	yes	yes
producer presents product	maybe	yes	no	no
"reader" presents product	no	no	yes	no
recorder takes notes	maybe	yes	yes	yes
checklists used to find errors	no	no	yes	no
errors categorized as found	no	no	yes	no
issues list created	no	yes	yes	yes
team must sign-off on result	no	yes	yes	maybe

*IPR—informal peer review WT—Walkthrough
IN—Inspection RRR—round robin review

Sample-Driven Reviews (SDRs)

- SDRs attempt to quantify those work products that are primary targets for full FTRs.

To accomplish this …

- Inspect a fraction a_i of each software work product, *i.* Record the number of faults, f_i found within a_i.
- Develop a gross estimate of the number of faults within work product *i* by multiplying f_i by $1/a_i$.
- Sort the work products in descending order according to the gross estimate of the number of faults in each.
- Focus available review resources on those work products that have the highest estimated number of faults.

Metrics Derived from Reviews

- inspection time per page of documentation
- inspection time per KLOC or FP
- inspection effort per KLOC or FP
- errors uncovered per reviewer hour
- errors uncovered per preparation hour
- errors uncovered per SE task (e.g., design)
- number of minor errors (e.g., typos)
- number of major errors (e.g., nonconformance to req.)
- number of errors found during preparation

Software Quality Assurance

Elements of SQA

- Standards
- Reviews and Audits
- Testing
- Error/defect collection and analysis
- Change management
- Education
- Vendor management
- Security management
- Safety
- Risk management

Role of the SQA Group-I

- **Prepares an SQA plan for a project.**
  - The plan identifies
    - evaluations to be performed
    - audits and reviews to be performed
    - standards that are applicable to the project
    - procedures for error reporting and tracking
    - documents to be produced by the SQA group
    - amount of feedback provided to the software project team
- **Participates in the development of the project's software process description.**
  - The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.

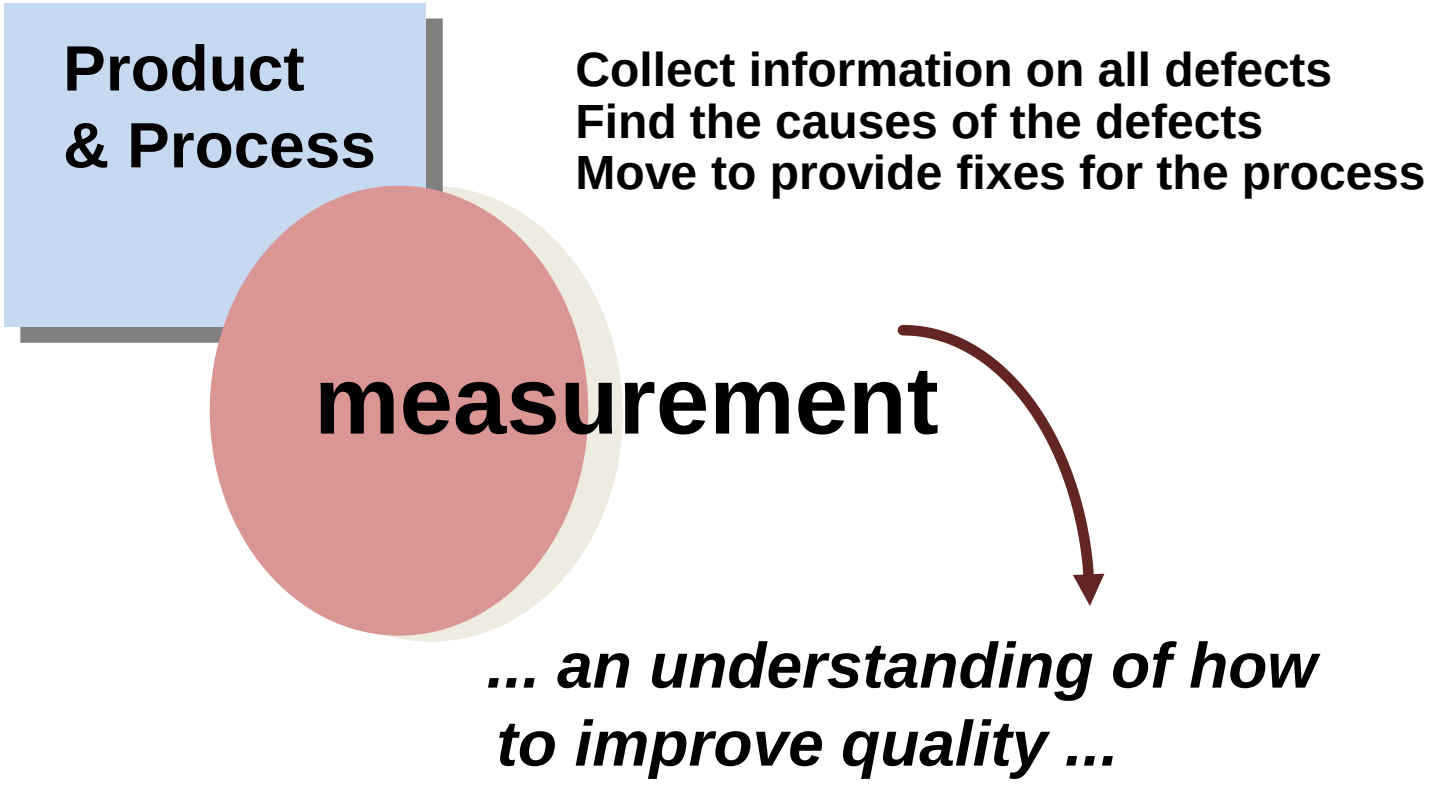
Role of the SQA Group-II

- **Reviews software engineering activities to verify compliance with the defined software process.**
  - identifies, documents, and tracks deviations from the process and verifies that corrections have been made.
- **Audits designated software work products to verify compliance with those defined as part of the software process.**
  - reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made
  - periodically reports the results of its work to the project manager.
- **Ensures that deviations in software work and work products are documented and handled according to a documented procedure.**
- **Records any noncompliance and reports to senior management.**
  - Noncompliance items are tracked until they are resolved.

SQA Goals

- **Requirements quality.** The correctness, completeness, and consistency of the requirements model will have a strong influence on the quality of all work products that follow.
- **Design quality.** Every element of the design model should be assessed by the software team to ensure that it exhibits high quality and that the design itself conforms to requirements.
- **Code quality.** Source code and related work products (e.g., other descriptive information) must conform to local coding standards and exhibit characteristics that will facilitate maintainability.
- **Quality control effectiveness.** A software team should apply limited resources in a way that has the highest likelihood of achieving a high quality result.

Statistical SQA



Statistical SQA

- Information about software errors and defects is collected and categorized.
- An attempt is made to trace each error and defect to its underlying cause (e.g., non-conformance to specifications, design error, violation of standards, poor communication with the customer).
- Using the Pareto principle (80 percent of the defects can be traced to 20 percent of all possible causes), isolate the 20 percent (the *vital few*).
- Once the vital few causes have been identified, move to correct the problems that have caused the errors and defects.

Statistical SQA – Root Cause Samples

Type of Error

- Incomplete or Erroneous Specification
- Misinterpretation of Customer Communication
- Intentional deviation from specifications
- Violation of programming standards
- Error in data representation
- Inconsistent component interface
- Error in design logic
- Incomplete or erroneous testing
- Inaccurate or incomplete documentation
- Miscellaneous

Severity of Error

- Serious
- Moderate
- Minor

Six-Sigma for Software Engineering

- Six Sigma is the most widely used Statistical SQA technique
- The term “six sigma” is derived from six standard deviations—3.4 instances (defects) per million occurrences—implying an extremely high quality standard.
- The Six Sigma methodology defines three core steps:
  - *Define* customer requirements and deliverables and project goals via well-defined methods of customer communication
  - *Measure* the existing process and its output to determine current quality performance (collect defect metrics)
  - *Analyze* defect metrics and determine the vital few causes.

Six-Sigma for Software Engineering

- Six Sigma has two variants of approach depending on current status of the process
  - DMAIC (Define-Measure-Analyze-Improve-Control) is used for measuring the current performance of a business process whereas DMADV (Define-Measure-Analyze-Design-Verify) is used for measuring the customer needs and specifications.
  - DMAIC, analyzes business process to find the root cause of defects. DMADV, analyzes business process for finding options to satisfy the customer needs and specifications.
  - DMAIC improves business process by eliminating or reducing defects. DMADV designs an appropriate business model to meet customer requirements.

ISO 9001:2000 Standard

- The ISO 9000 family of standards consists of three standards: namely ISO 9000, ISO 9001, and ISO 9004
  - The ISO 9000 standard covers the fundamentals and vocabulary of quality management systems
  - The ISO 9001 standard specifies the requirements of a quality management system, and applies to manufacturing, software, and service organizations
  - The ISO 9004 standard (Managing for the Sustained Success of the Organization) provides guidance for performance improvement, and it is helpful in assisting organizations in the implementation of ISO 9001.

ISO 9001:2000 Standard

- There are five main clauses in ISO 9001 standard, with each clause having several sub-clauses

Clause	Description
Quality Management System	This clause refers to the documentation and implementation of the Quality Management System (QMS)
Resource Management	This is concerned with the provision of the resources required to implement the QMS.
Product or Service Realization	This is concerned with the provision of processes to implement the product or service.
Management Responsibility	This is concerned with defining the responsibilities of management in implementing the quality system.
Measurement, Analysis and Improvement	This is concerned with the establishment of a measurement programme, and using measures for continuous improvement.

ISO 9001:2000 Standard

- **Quality Management System**
  - Quality Manual
  - Control of Documents
  - Control of records
  - Internal Audit
  - Control of Non-conforming product
  - Corrective Action Procedure
  - Preventive Action Procedure.
- **Management Responsibility**
  - Management Commitment
  - Customer Focus
  - Defining Quality Policy and Quality Manual
  - Planning and setting quality objectives
  - Responsibility, Authority and Communication
  - Management Review

ISO 9001:2000 Standard

- **Resource Management**
  - Provision of Resources
  - Human Resources
  - Infrastructure
  - Work Environment
- **Measuring, Analysis, and Improvement**
  - Measurement of customer satisfaction
  - Internal audits
  - Measurement and monitoring of processes
  - Measurement and monitoring of product
  - Control of nonconformity
  - Analysis of data
  - Continual improvement
  - Corrective action
  - Preventative action
- **Product or Service Realization**
  - Planning of product realization
  - Customer related processes
  - Customer requirements
  - Customer communication
  - Design and development
  - Control of changes
  - Purchasing process (information and verification)
  - Production and service provision
  - Validation of processes
  - Identification and traceability
  - Customer property
  - Preservation of product
  - Control of measuring and monitoring devices

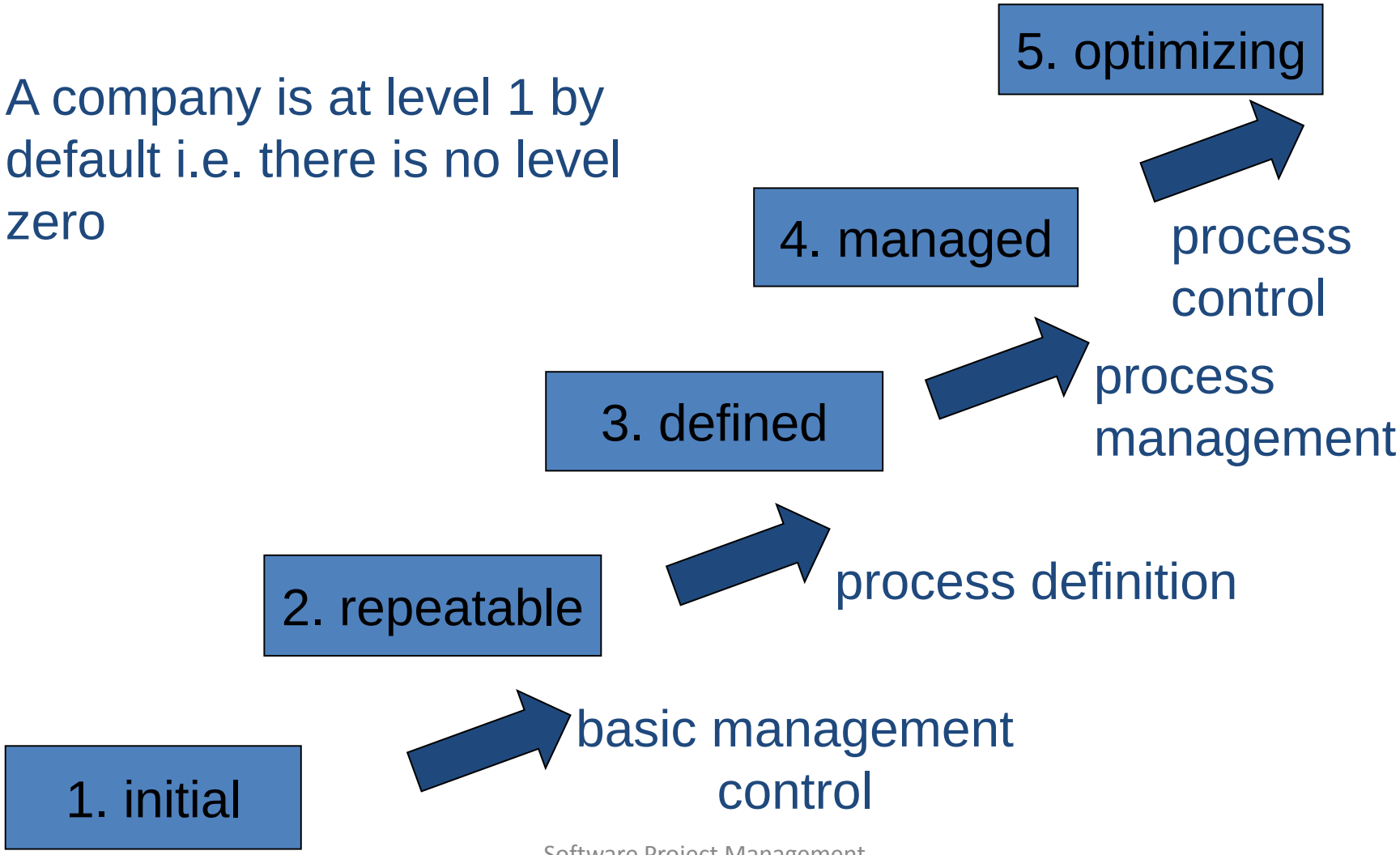
Capability Maturity Model

Capability maturity model

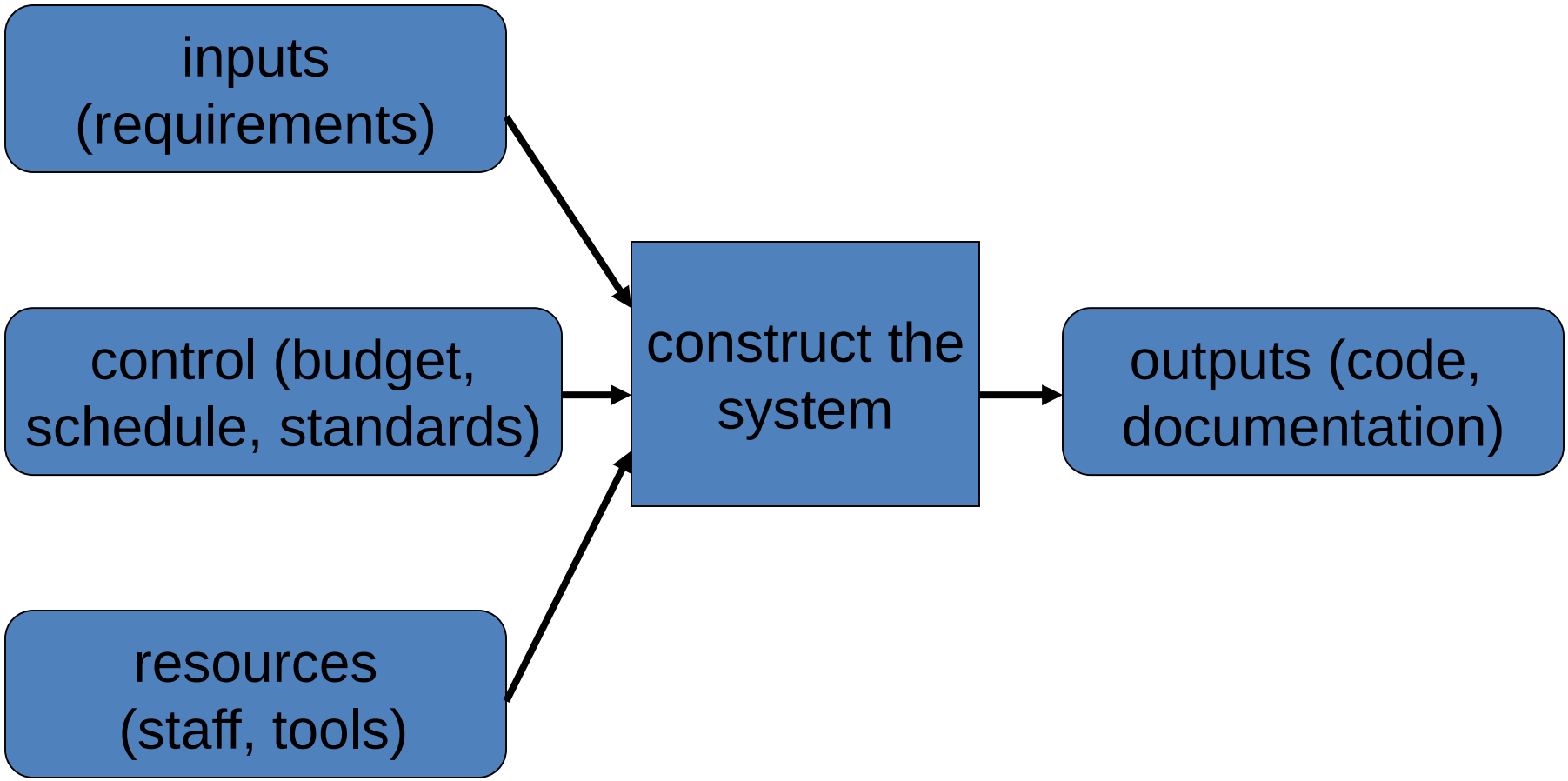
- Created by Software Engineering Institute, Carnegie Mellon University
- CMM developed by SEI for US government to help procurement
- Assessment is by questionnaire and interview
- Different versions have been developed for different environments

Process maturity levels

A company is at level 1 by default i.e. there is no level zero



A Repeatable Model

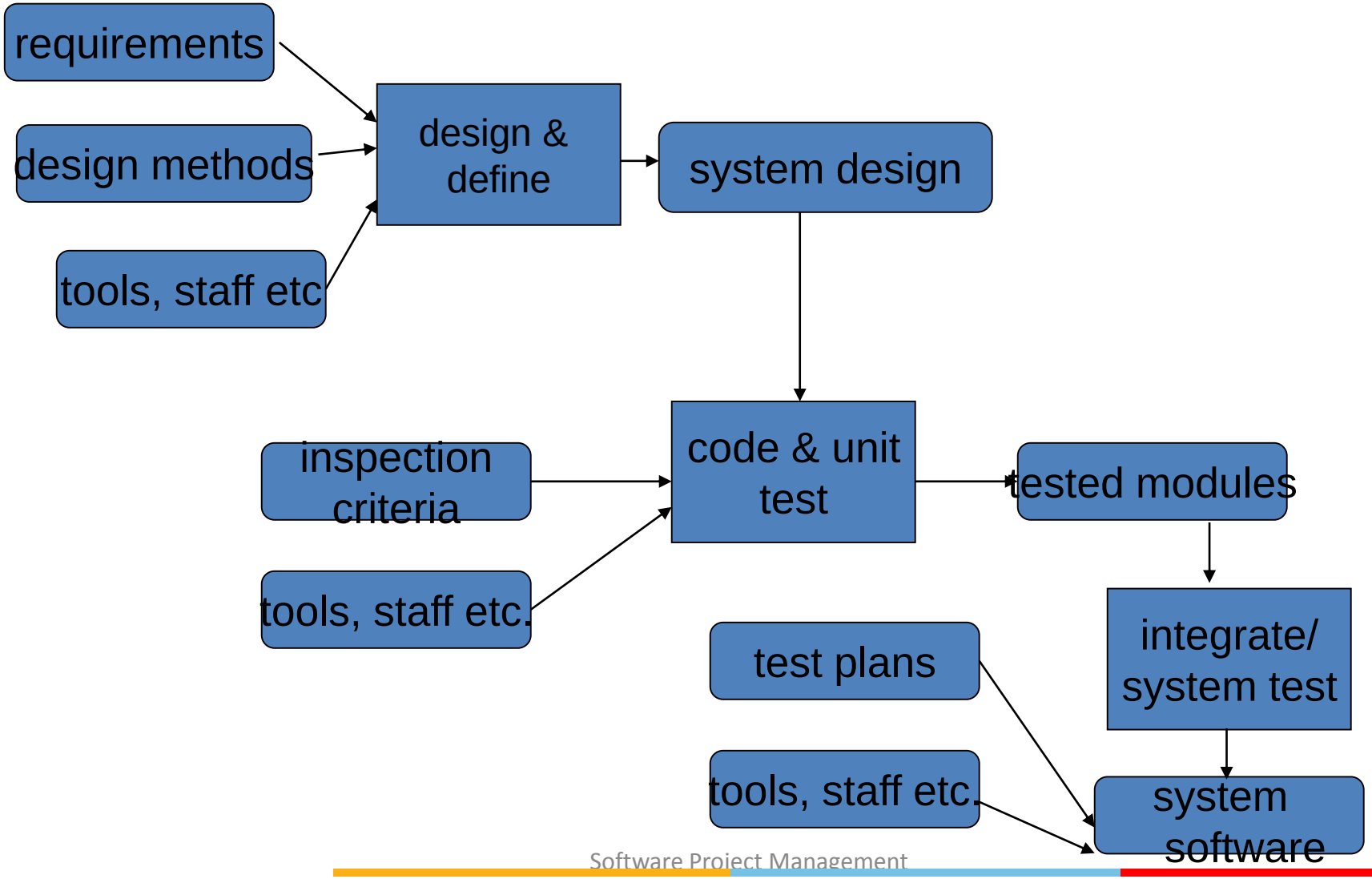


Repeatable model KPAs

To move to this level concentrate on:

- Configuration management
- Quality assurance
- Sub-contract management
- Project planning
- Project tracking and oversight
- Measurement and analysis

A Defined Process

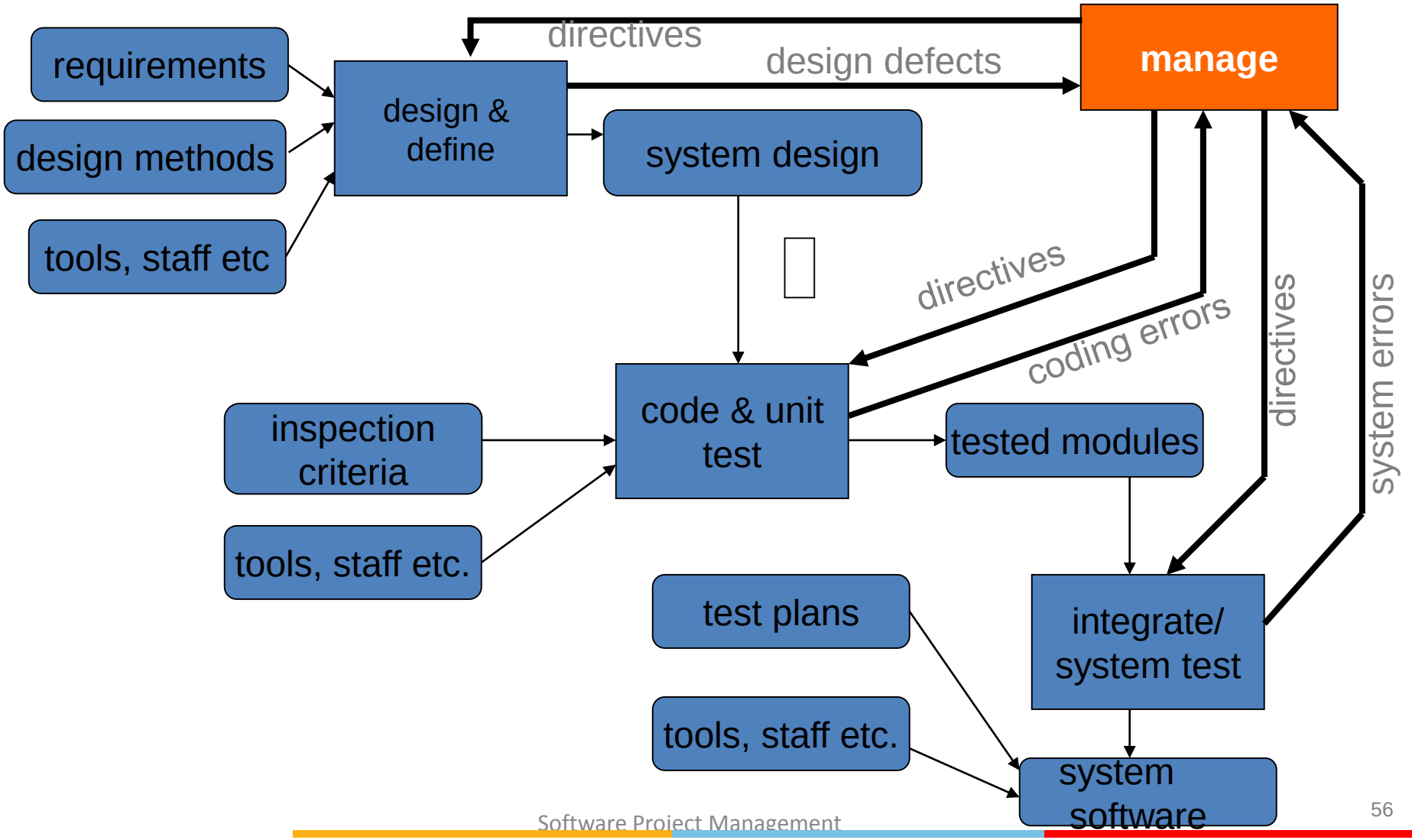


Repeatable to defined KPAs

Concentrate on

- Requirements development and technical solution
- Verification and validation
- Product integration
- Risk management
- Organizational training
- Organizational process focus
- Decision analysis and resolution
- Process definition
- Integrated project management

A Managed Process

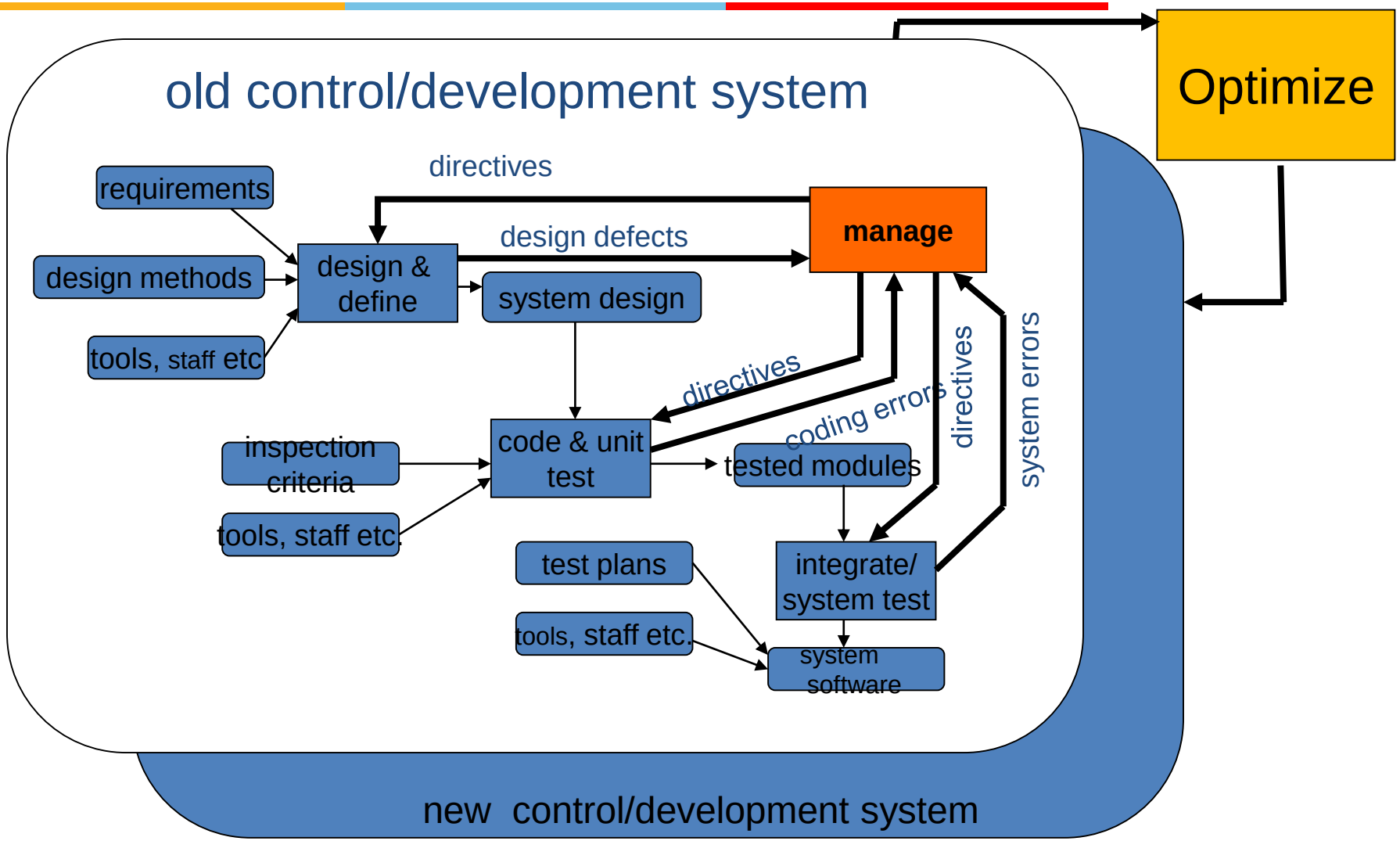


Defined to Managed KPAs

Concentrate on:

- Organizational process performance
- Quantitative project management

Optimizing



Managing to Optimizing: KPAs

concentrate on:

- Causal analysis and resolution
- Organizational innovation and deployment

Some faqs about CMMI

- Currently CMMI process areas are organized into 3 constellations
  - **Development:** For improving the development of (product or complex service) solutions (almost as described in the last 10 slides)
  - **Acquisition:** For improving the purchasing of products, services and/or solutions
  - **Services:** For improving delivery of services and creation of service systems (say, to operate a solution but not buy it or build it in the first place)
- Being Agile with CMMI: they can co-exist
  - nothing in CMMI requires that everything be "documented"
  - nothing among the agile values or practices insist that a team produces "no" documentation

http://www.cmmifaq.info/

Thank You

Any Questions?