

Presentation 1

The software quality challenge

- The uniqueness of software quality assurance
- The environments for which SQA methods are developed

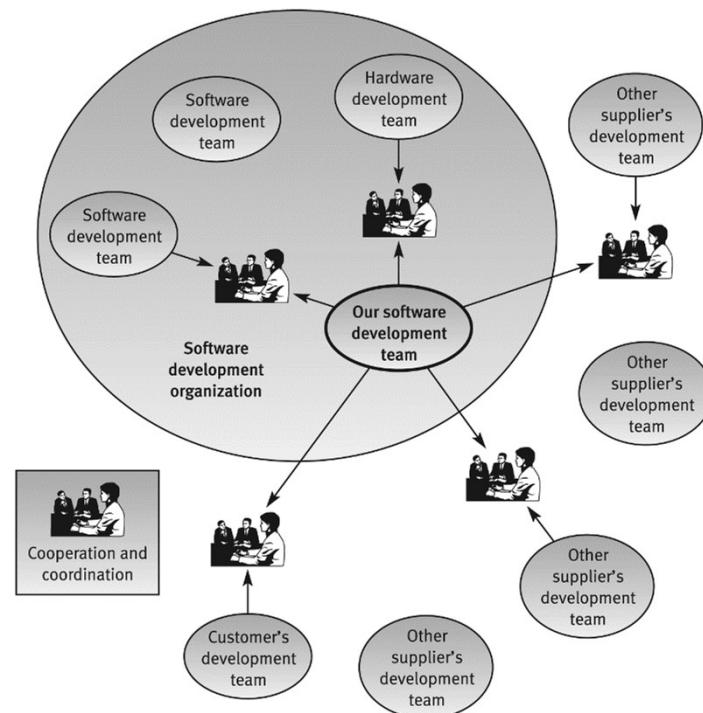
The uniqueness of the software development process

- High complexity
- Invisibility of the product
- Limited opportunities to detect defects (“bugs”)

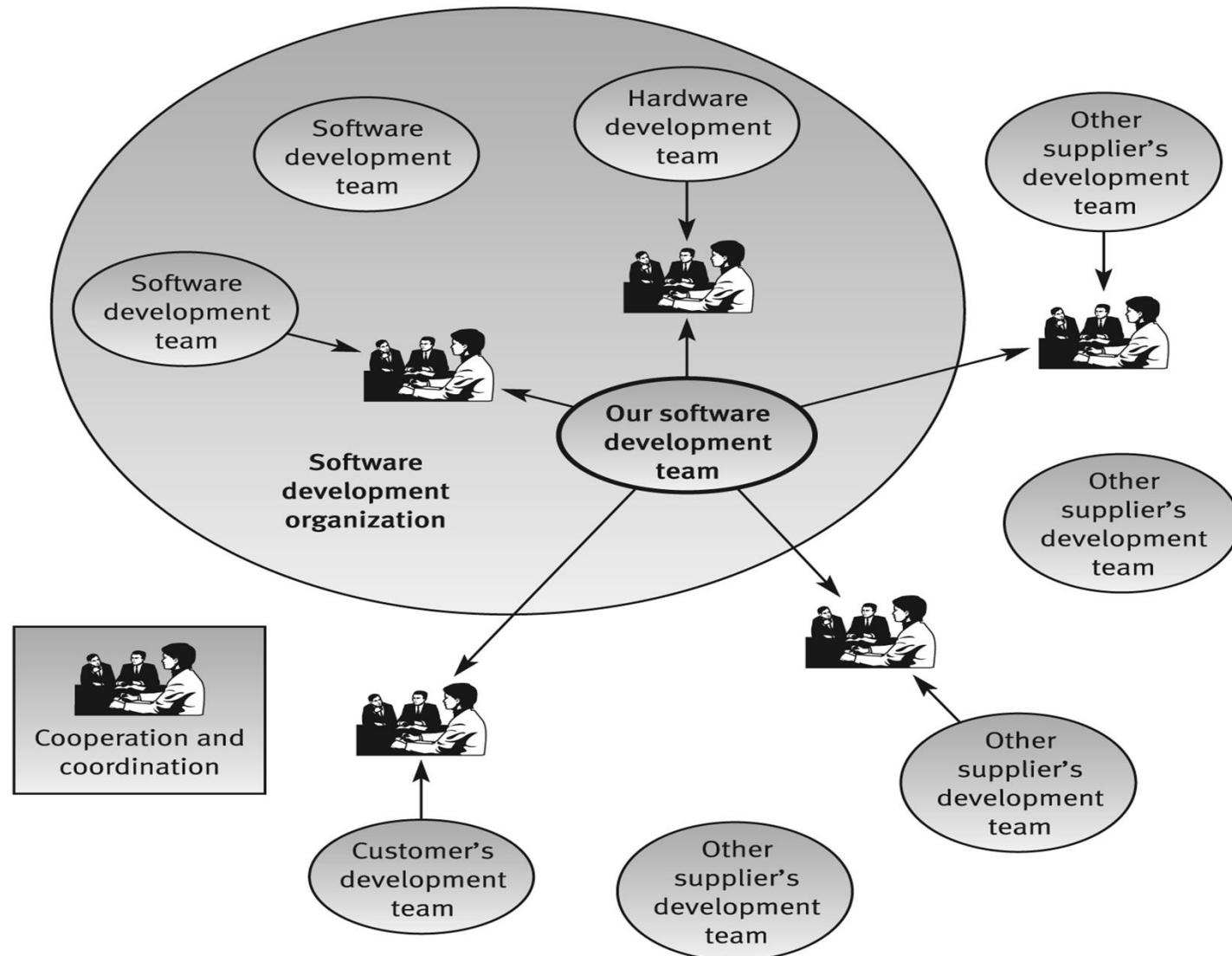
The characteristics of the SQA environment process

- Being contracted
- Subjection to customer-supplier relationship
- Requirement for teamwork
- Need for cooperation and coordination with other development teams
- Need for interfaces with other software systems
- Need to continue carrying out a project while the team changes
- Need to continue maintaining the software system for years

Cooperation and coordination scheme for a software development project team

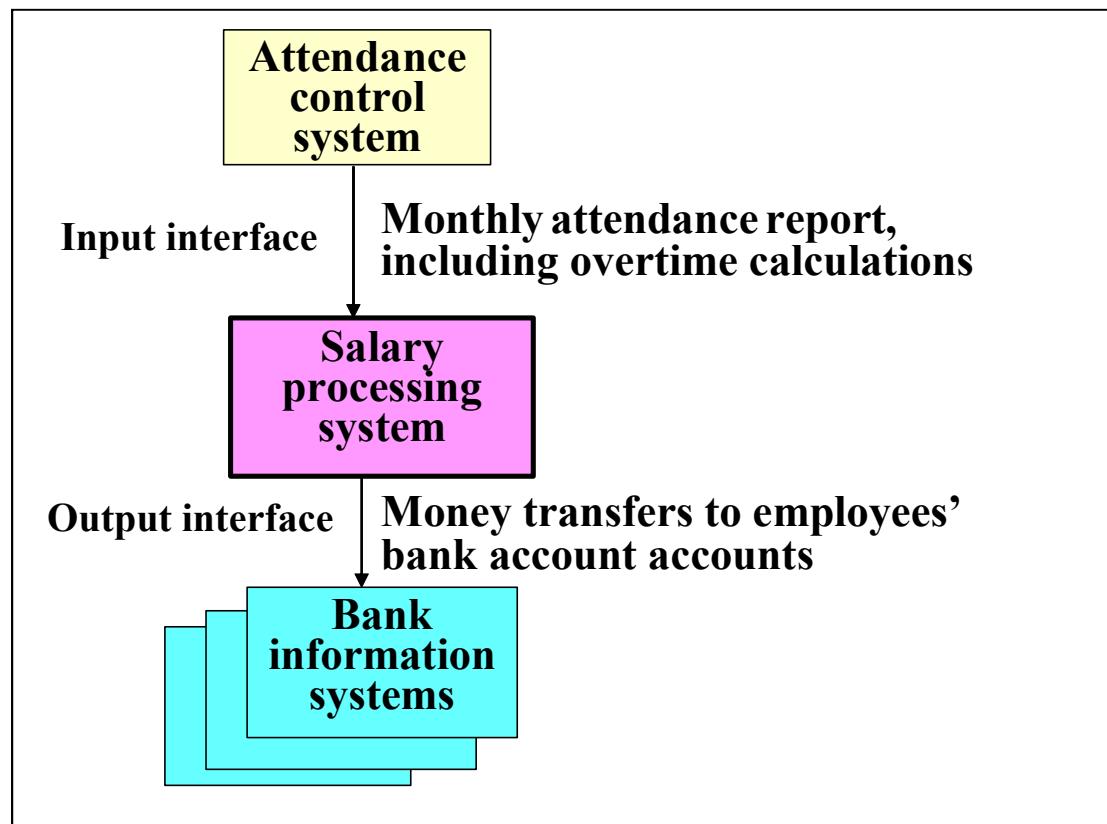


OHT 1.5



SQA from theory to implementation

Salary software system - an example of software interfaces



What is software quality?

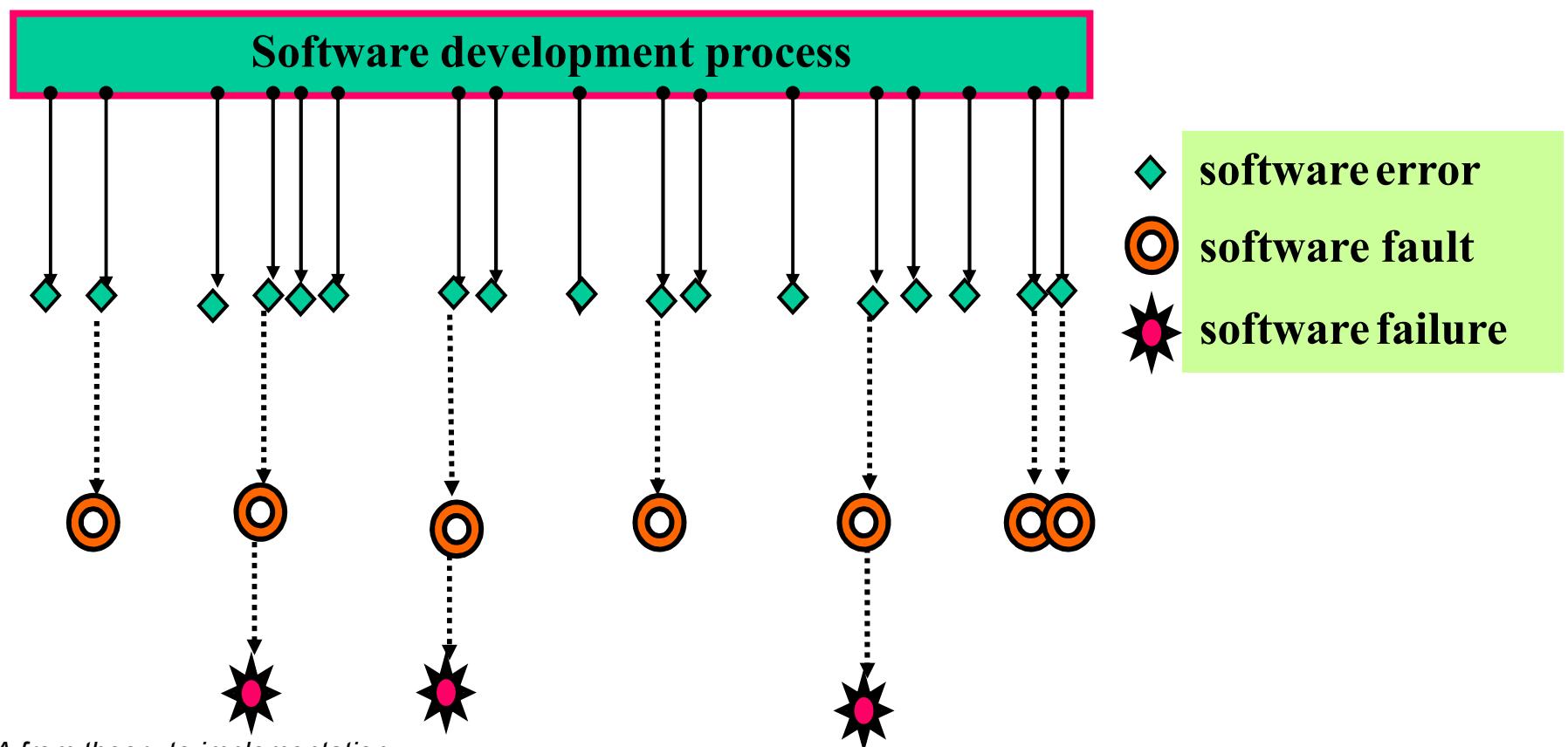
- **What is software?**
- **Software errors, faults and failures**
- **Classification of the causes of software errors**
- **Software quality – definition**
- **Software quality assurance – definition and objectives**
- **Software quality assurance and software engineering**

Software - IEEE definition

Software is:

Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.

Software errors, software faults and software failures



The nine causes of software errors are:

- 1. Faulty requirements definition**
- 2. Client-developer communication failures**
- 3. Deliberate deviations from software requirements**
- 4. Logical design errors**
- 5. Coding errors**
- 6. Non-compliance with documentation and coding instructions**
- 7. Shortcomings of the testing process**
- 8. User interface and procedure errors**
- 9. Documentation errors**

Software quality - IEEE definition

Software quality is:

- (1) The degree to which a system, component, or process meets specified requirements.**
- (2) The degree to which a system, component, or process meets customer or user needs or expectations.**

Software quality - Pressman's definition

Software quality is :

Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

SQA - IEEE definition

Software quality assurance is:

1. A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.
2. A set of activities designed to evaluate the process by which the products are developed or manufactured. Contrast with: quality control.

SQA - expanded definition

Software quality assurance is:

A systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines.

The objectives of SQA activities in software development

- (1) Assuring an acceptable level of confidence that the software will conform to functional technical requirements.
- (2) Assuring an acceptable level of confidence that the software will conform to managerial scheduling and budgetary requirements.
- (3) Initiation and management of activities for the improvement and greater efficiency of software development and SQA activities.

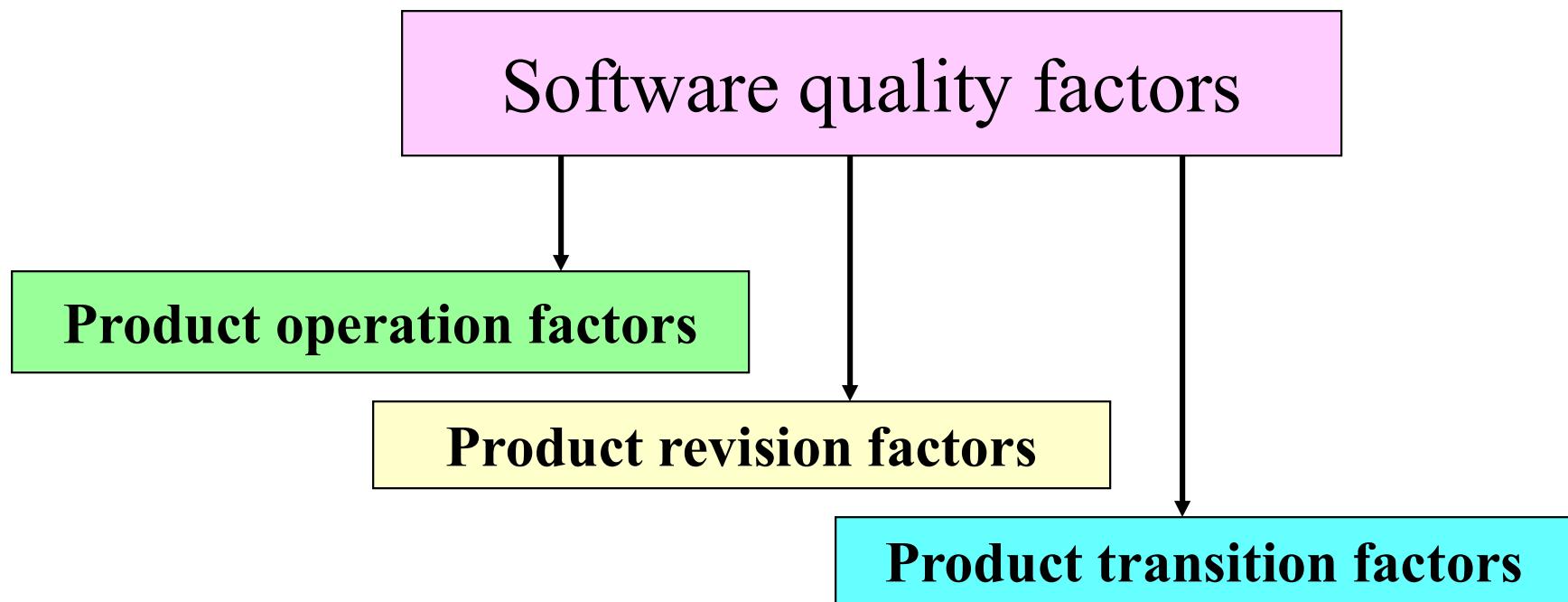
The objectives of SQA activities in software maintenance

- (1) Assuring an acceptable level of confidence that the software maintenance activities will conform to the functional technical requirements.
- (2) Assuring an acceptable level of confidence that the software maintenance activities will conform to managerial scheduling and budgetary requirements.
- (3) Initiate and manage activities to improve and increase the efficiency of software maintenance and SQA activities.

Software quality factors

- The need for comprehensive software quality requirements
- Classification of requirements into software quality factors
- Product operation factors
- Product revision factors
- Product transition factors
- Alternative models of software quality factors
- Who is interested in defining quality requirements?
- Software compliance with quality factors

McCall's software quality factors model



Requirement Document

- A comprehensive definition of requirement
- The need to define requirement that belong to each quality factor
- Requirement documents are expected to differ in the emphasis placed on the various factors. (Sometimes for comparisons)
- Not all the factors will be represented.

Product operation factors

- Correctness
- Reliability
- Efficiency
- Integrity
- Usability

Correctness

- Defined in a list of the software system's required outputs.
- Output specifications are usually multidimensional, including:
 - output mission
 - required accuracy of the outputs
 - the completeness of the output information
 - the up-to-dateness of the information
 - the availability of the information
 - the standards for coding & documenting

Reliability

- Deals with failures to provide services.
- Usually refer to the maximum allowed software system failure rate that can be the entire system or one or more of its functions.

Efficiency

- Deals with the hardware resources needed to perform all the functions
- May include the maximum values at which the hardware resources will be applied in the software system.
- May also deal with the time between recharging of the system's units.

Integrity

- Deals with the system security, namely, the prevention of the access to unauthorized persons.
- Also deals with distinguishing the privilege (read, copy, write, ... permit) to the information of the personnel.

Usability

- Deals with the scope of staff resources needed to train a new employee and to operate the system.

Product revision factors

- Deals with software maintenance activities: corrective maintenance, adaptive maintenance, perfect maintenance.

- Maintainability
- Flexibility
- Testability

Maintainability

- Determines the efforts that will be needed by users and maintenance personnel to identify the reasons for failures, to correct the failures, and to verify the success of correctness.
- Refers to modular structure (size), program documentation (standards), manuals, etc.

Flexibility

- Refers to the capabilities & efforts required to support adaptive maintenance & perfect maintenance, including human resources, extents of activities, etc.
- E.g., The system should be suitable for teachers of all subjects and all school levels.

Testability

- Deals with the testing of the system as well as with the operation.
 - For ease of testing, such as providing predefined intermediate results and log files.
 - For operation, such as automatic diagnostics performed by the system prior to starting the system; automatic generating report about detected faults.
 - For maintenance, automatic diagnostic checks applied by maintenance technicians to detect the causes of failures.

Product transition factors

- Refers to adaptation to other environments and interaction with other systems.

- Portability
- Reusability
- Interoperability

Portability

- Refers to adaptation to other environments consisting of different hardware, operating systems, and so forth.

Reusability

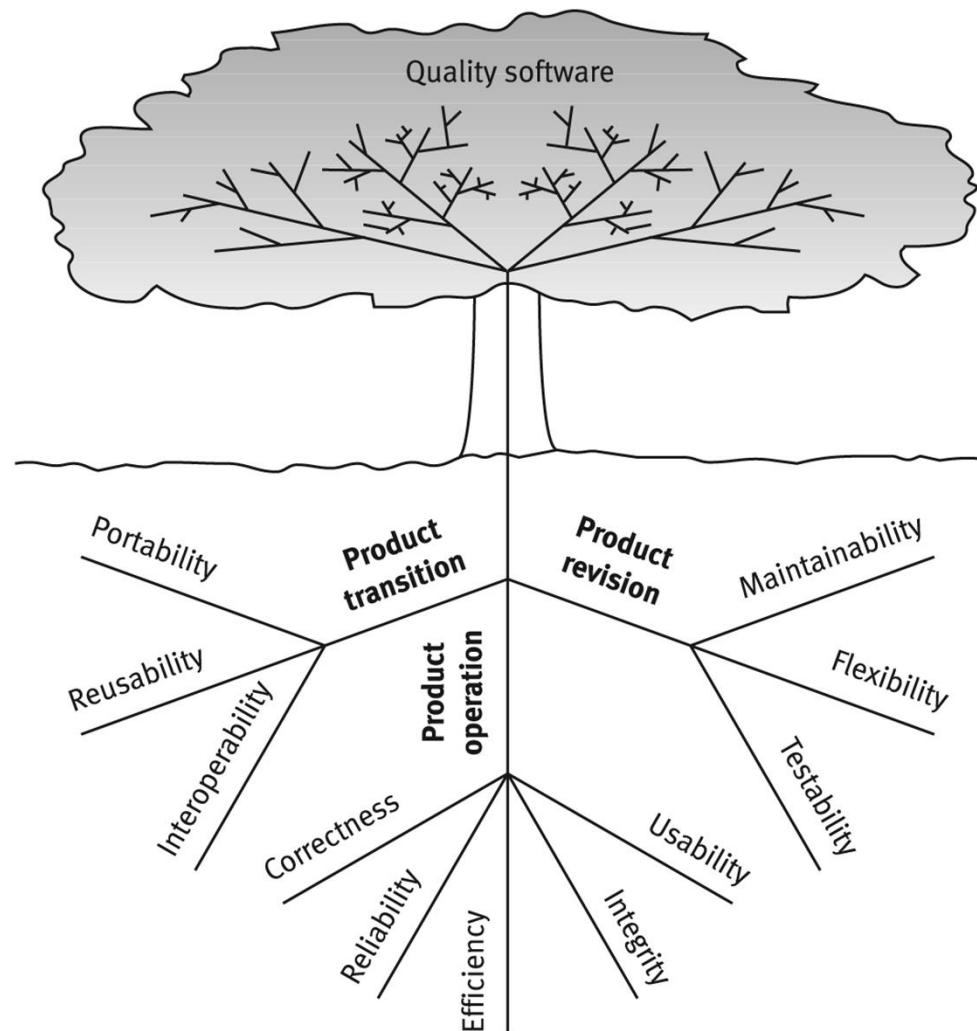
- Deals with the use of software modules originally designed for one project in a new software project being developed.
- Can save resources, shorten time and provide higher quality modules.

Interoperability

- Focuses on creating interfaces with other software systems or with other equipment firmware.
- Can specify the names of the software or firmware for which interface is required.
- Can also specify the output structure accepted as standard.

OHT 1.34

McCalls factor model tree



SQA from theory to implementation

OHT 1.35

McCall's factor model and alternative models

No.	Software quality factor	McCall's classic model	Alternative factor models	
			Evans and Marciak model	Deutsch and Willis model
1	Correctness	+	+	+
2	Reliability	+	+	+
3	Efficiency	+	+	+
4	Integrity	+	+	+
5	Usability	+	+	+
6	Maintainability	+	+	+
7	Flexibility	+	+	+
8	Testability	+		
9	Portability	+	+	+
10	Reusability	+	+	+
11	Interoperability	+	+	+
12	Verifiability		+	+
13	Expandability		+	+
14	Safety			+
15	Manageability			+
16	Survivability			+

SQA from theory to implementation

Verifiability

- Defines design and programming features.
- Refer to modularity and simplicity, and adherence to documentation and programming guidelines.
- Improves the design reviews and software tests.

Expandability

- Refers to future efforts that will be needed to serve larger populations.

Safety

- To eliminate conditions hazardous to operations of equipment as a result of errors, including providing alarm signals.

Manageability

- Refers to the administrative tools that supports software modification, such as CM.

Survivability

- Refers to the continuity of service. These define the minimum time allowed between failures, and the maximum time permitted for recovery of service.

OHT 1.41

Who is interested in defining quality requirements?

- A project will be carried out according to two requirement documents:
 - The client's requirement document
 - The developer's additional requirement document

Software compliance with quality factors

- Examine the development process by reviews, inspections, tests, etc.
- Verify and validate the quality by measuring.
- Sub-factors are defined for quality factors.
- Please see Table 3.3 suggested by Evans and Marciak (1987).

OHT 1.43

Presentation 4

Components of software quality assurance system overview

SQA from theory to implementation

Why do we need a wide range of SQA components ?

- Multitude of source of errors
 - various style of source of errors will affect the SQA components
- * The environment in which software development & maintenance is undertaken directly influences the SQA components.

Six Classes of SQA Components

- Pre-project components
- Software project life cycle components
- Infrastructure components for error prevention and improvements
- Management SQA components
- SQA standards, system certification and assessment components
- Organizing for SQA – the human components

Pre-project SQA components

- **Contract reviews**
 - To assure commitments have been properly defined under resources, schedule and budget constraints.
 - detailed examination of the project proposal draft
 - detailed examination of the contract draft
 - include: requirements, schedule, resource, staff's capacity, customer capacity, risks.
- **Development and quality plans**
 - To assure plans have been correctly determined.

Development and quality plans

- Development plan
 - Schedule
 - manpower & resources
 - risk evaluation
 - organizational related issues
 - project methodology, development tools;
 - reuse plans
- Quality plan
 - quality goal
 - criteria for starting & ending of stage
 - list of reviews, tests, scheduled V&V activities

Software project life cycle components

- The development and the operation-maintenance stages

- **Reviews** (Formal design reviews & Peer reviews)
- **Expert opinions** (Formal design reviews)
- **Software testing**
- **Software maintenance components**
- **Assurance of the quality of external participants' work**

Infrastructure components for prevention and improvement

- Use of computerized and automatic tools

- Procedures and work instruction (based on experience and knowledge)
- Templates and checklists (supporting devices)
- Staff training, retraining and certification
- Preventive and corrective actions
- Configuration management
- Documentation control

Documentation control activities

SQA requires the application of measure to ensure long-term availability of major documents

- Def. of the types of controlled documents
- Specification of the formats, identification methods
- Def. of review and approval processes for each controlled document
- Def. of archive storage methods.

Management SQA components

- Project progress control
- Software quality metrics
- Software quality costs

Project progress control

Objective: To detect the appearance of any situation that may induce deviation from the project plan.

Focus on:

1. resource usage
2. schedule
3. risk management activities
4. the budget

Software quality metrics

- Quality of software development and maintenance activities
- Development teams' productivity
- Help desk and maintenance teams' productivity
- Software faults density
- Schedule deviations.

Software quality costs

- Cost of control (prevention costs, appraisal costs, managerial preparation and control costs) + costs of failure
- Expanding the resources allocated to control activities yields much larger savings in failure costs while reducing total quality costs

SQA standards, system certification and assessment components

- Project process standards (focus on “how”)
- Quality management standards (focus on “what”)

Objectives:

- Utilization of international professional knowledge
- Improvement of coordination with other organizations’ quality systems
- Objective professional evaluation and measurement of the organization’s SQA achievement

Organizing for SQA - the human components

- Management's role in SQA
- The SQA unit
- SQA trusties
- SQA committees
- SQA forums

Considerations guiding construction of organization's SQA system

- The SQA organizational base
- The SQA components to be implemented within the organization and the extent of their use

OHT 1.58

The main considerations affecting the use of the components (1/2)

Organizational considerations

- Type of software development clientele
- Type of software maintenance clientele
- Range of software products
- Size of the organization
- Degree and nature of cooperation with other organizations carrying out related projects
- Optimization objective (quality, productivity, efficiency, savings)

Project and maintenance service considerations

- Level of complexity and difficulty
- Degrees of experience with the project technology
- Extent of software reuse in the new projects

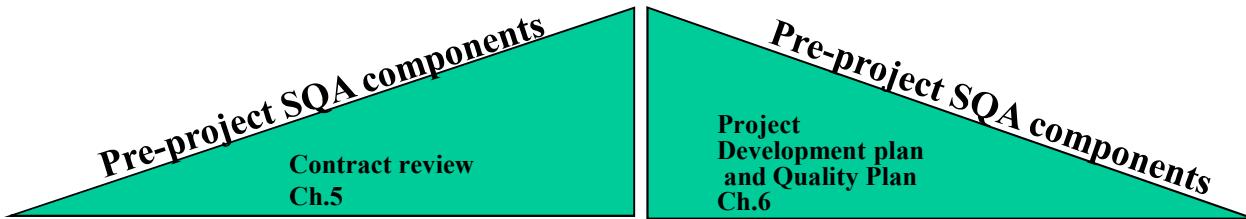
(2/2)

Professional staff considerations

- Professional qualifications
- Level of acquaintance with team members

OHT 4.60

The Software Quality Shrine



Project Life Cycle SQA components

Formal Design Reviews
Sec. 8.2

Peer Reviews
Sec. 8.3

Experts Opinion
Sec. 8.5

Software Testing
Chs. 9-10

Software Maintenance
Ch. 11

SQA of External Participants
Ch 12

Quality Infrastructure components

Procedures
Ch. 14

Supporting
Devices
Ch. 15

Training
Instruction
Ch. 16

Preventive
Actions
Ch.17

Configuration
Management
Ch. 18

Documentation
Control
Ch. 19

Quality Management

Project
Progress
Control
Ch. 20

Software
Quality
Metrics
Ch. 21

Software
Quality
Costs
Ch. 22

Standards

Quality
Management
Standards
Ch. 23

Project
Process
Standards
Ch.24

Organizational Base – Human components

Management - Ch. 25

SQA Unit - Sec. 26.1

SQA Trustees – Sec. 26.2

SQA Committees – Sec. 26.2

SQA Forums – Sec 26.4

SQA from theory to implementation

Contract review

- Contract review process and stages
- Contract review objectives
- Implementation of contract review
- Contract review subjects
- Contract review for internal projects

Common contract situations

- Participation in a tender
- Proposal submission according to customer's RFP
- Receipt of an order from a company's customer
- Internal request from another department in the organization

OHT 1.63

Contract review stages

Proposal draft review

+

Contract draft review

Contract review

Proposal draft review - Objectives

To make sure that the following activities have been satisfactorily carried out:

1. Customer requirements clarified and documented
2. Alternative approaches for carrying out the project examined
3. Formal aspects of the relationship between the customer and the software firm specified
4. Development risks identified
5. Project resources and timetable adequately estimated
6. The firm's capacity with respect to the project examined
7. The customer's capacity to fulfill his commitments examined
8. Partner and subcontractor's participation conditions defined
9. Protection of proprietary rights defined

Contract draft review - Objectives

To make sure that the following activities have been satisfactorily carried out:

1. No unclarified issues remain in the contract draft
2. All understandings reached subsequent to the proposal are correctly documented
3. No “new” changes, additions, or omissions have entered the contract draft

Implementation of a contract review (1/2)

- Factors affecting the extent of a contract review
 - project magnitude, technical complexity, acquaintance/experience with/in the project area, project organizational complexity
- Who performs a contract review? (In ascending order with complexity)
 - the leader or another member of the proposal team
 - the members of the proposal team
 - an outside professional or a company staff
 - a team of outside experts

(2/2)

- Implementation of a review for major proposals
 - Def. of major proposals
 - The difficulties of carrying out contract reviews
 - *time pressures, requires professional work, time conflict
- Recommended avenues for implementing major contract reviews
 - the review should be scheduled.
 - a team should carry out the review.
 - a review team leader should be appointed.

Contract Review Subjects

- Based on the contract review objectives.
- Checklists are useful devices.
- Contract review team determines the list of subjects pertinent for the project.

Contract reviews for internal projects

- Types of internal projects

(1) Administrative or operative software to be applied internally

(2) Software packages originally intended to be sold to the public as “off-the-shelf” packages

(3) Firmware to be embedded in the company’s products

OHT 1.70

"Loose relationship" internal projects – Disadvantages to internal customers

<i>Subject</i>	<i>Disadvantages to the internal customer</i>
(1) Inadequate definition of project requirements	* Implementation deviates from the needed applications * Low satisfaction
(2) Poor estimate of the required resources	* Unrealistic expectations about project feasibility
(3) Poor timetable	* Missing scheduled dates for beginning the distribution of new products
(4) Inadequate awareness of development risks	* Customer unprepared for project risks and their consequences

OHT 1.71

"Loose relationship" internal projects – Disadvantages to internal developers

<i>Subject</i>	<i>Disadvantages to the internal developers</i>
(1) Inadequate definition of project requirements	* Higher change requirements * Wasted resources due to introducing avoidable changes
(2) Poor estimate of the required resources	* Substantial deviations from budget * Friction between units induced by requirements for budget additions
(3) Poor timetable	* Development activities are under time pressures and suffer from low quality * Delays in freeing staff for their next project
(4) Inadequate awareness of development risks	* Tardy initiation of efforts to overcome difficulties

Development and quality plans

- **Development plan and quality plan objectives**
- **The elements of the development plan**
- **Elements of the quality plan**
- **Development and quality plans for small and for internal projects**
- **Software development risks and software risk management**

Development and quality plans - Objectives

Planning is meant to prepare adequate foundations for successful and timely completion of the project. The planning process includes:

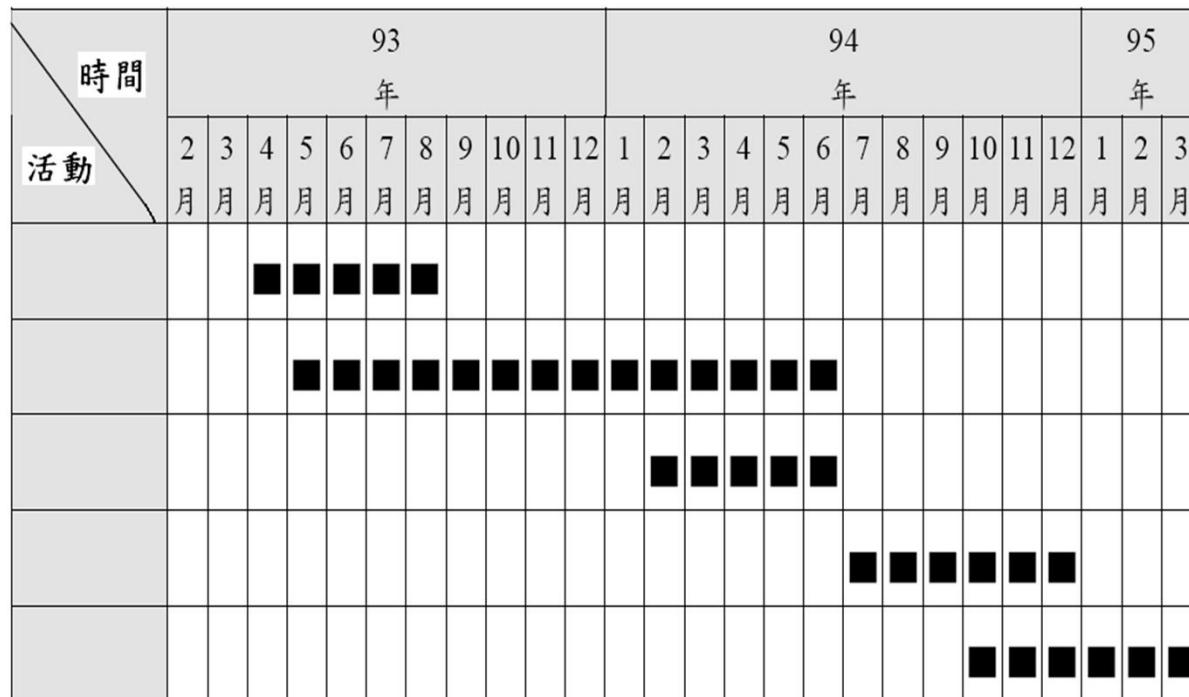
- 1. Scheduling development activities and estimating the required manpower resources and budget**
- 2. Recruiting team members and allocating development resources**
- 3. Resolving development risks**
- 4. Implementing required SQA activities**
- 5. Providing management with data needed for project control**

The elements of a development plan

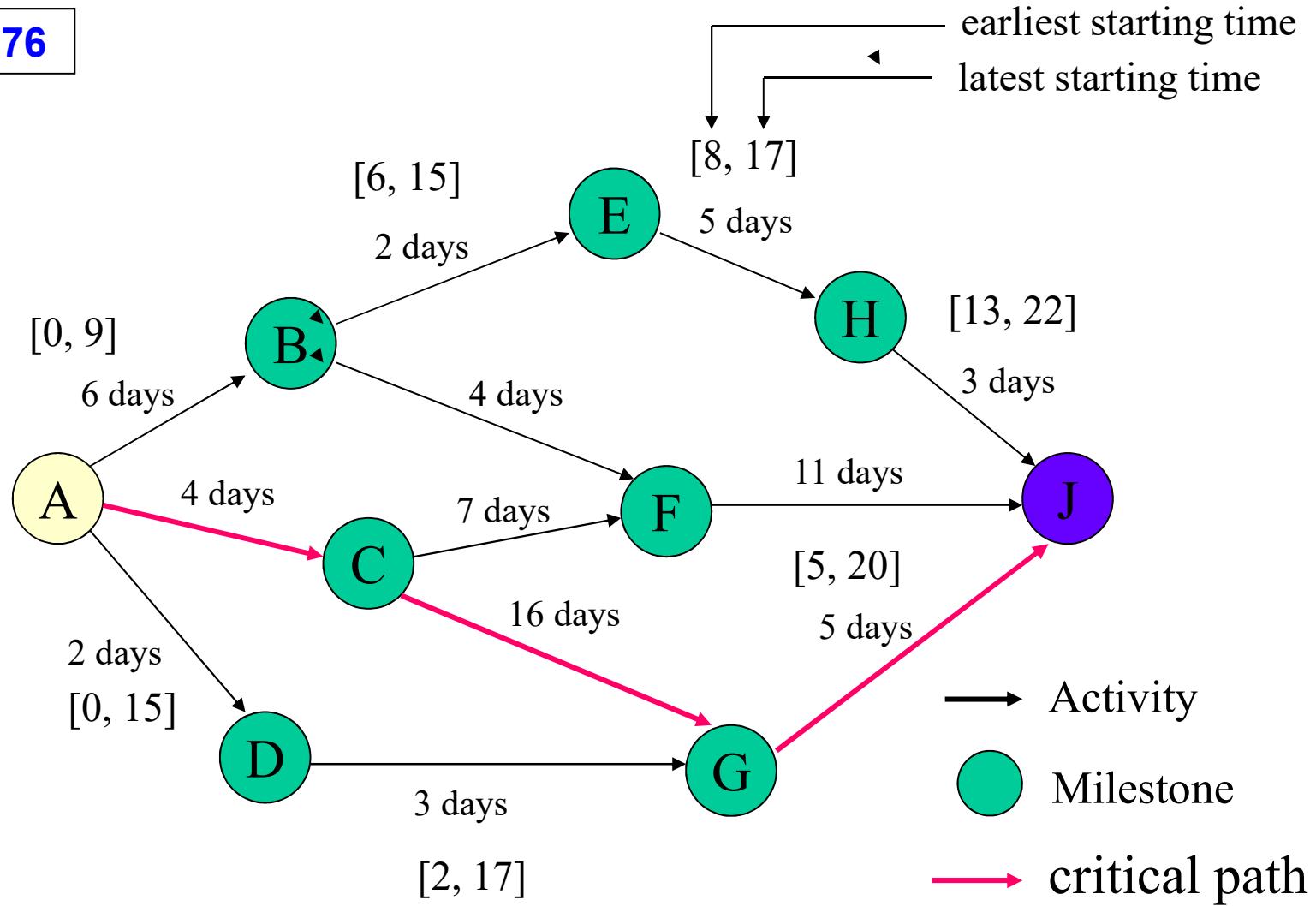
1. Project products, specifying “deliverables”
2. Project interfaces
3. Project’s methodology and development tools
4. Software development standards and procedures
5. Map of the development process
6. Project milestones
7. Project staff organization and coordination with external participants
8. Required development facilities
9. Development risks and risk management actions
10. Control methods
11. Project cost estimates

OHT 1.75

GANTT Chart (甘特圖)

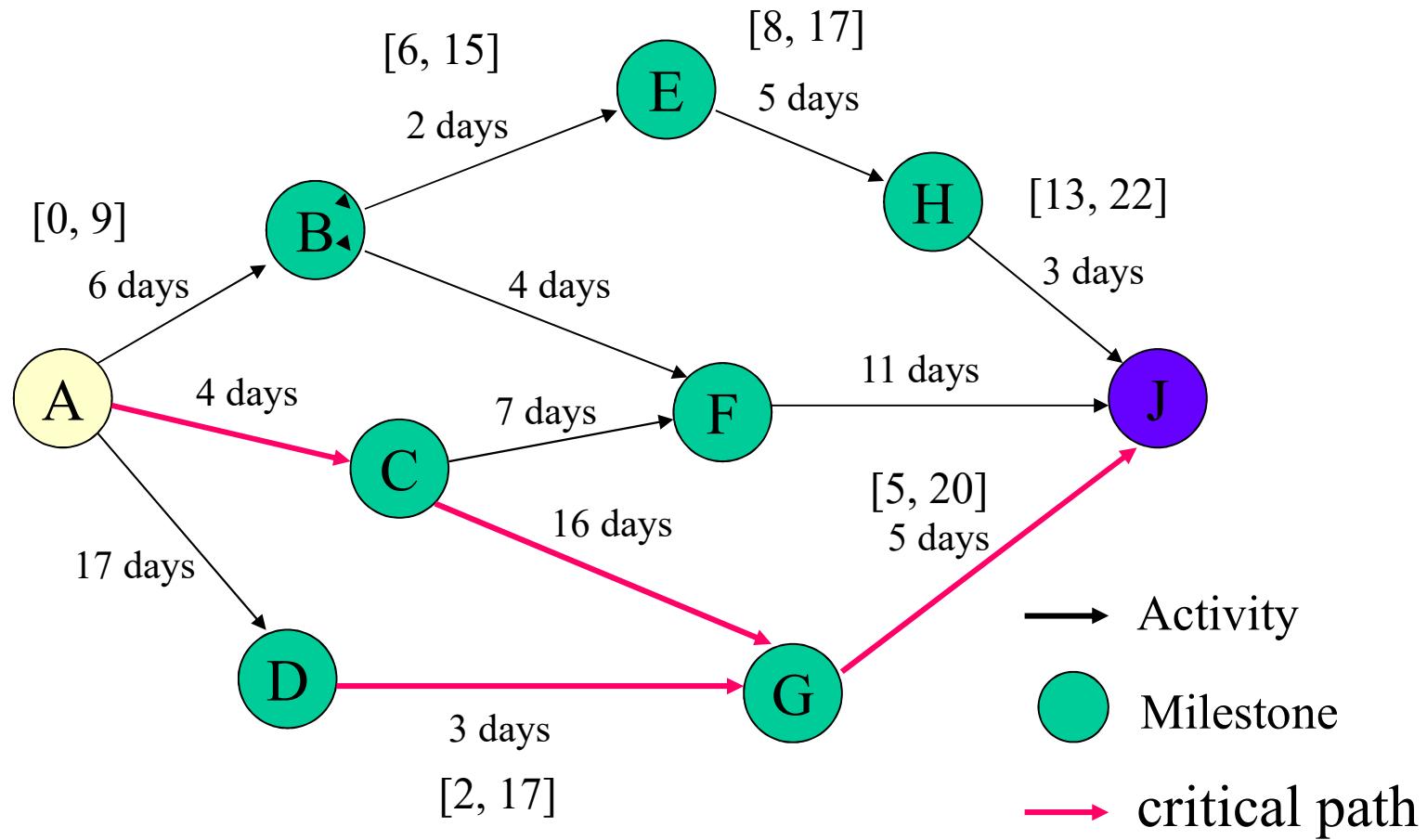


OHT 1.76



PERT Chart

OHT 1.77



Updated PERT Chart at day 17

The elements of a development plan for small projects

1. Project products, indicating “deliverables”
2. Project benchmarks
3. Development risks
4. Project cost estimates

The elements of a software quality plan

1. List of quality goals
2. Review activities
3. Software tests
4. Acceptance tests for software externally developed
5. Configuration management plans: tools, procedures and dates for version release

Some quality plan's elements must be coordinated with the development process

- The schedule of review and software activities should be coordinated with the schedule for completion of the software products (documents and code).
- Correction should also be coordinated with the review and test activities.
- The configuration management activities and especially the release of planned baseline versions of the software system must be coordinated with the progress of the development process and the completion of the production of the relevant software products.

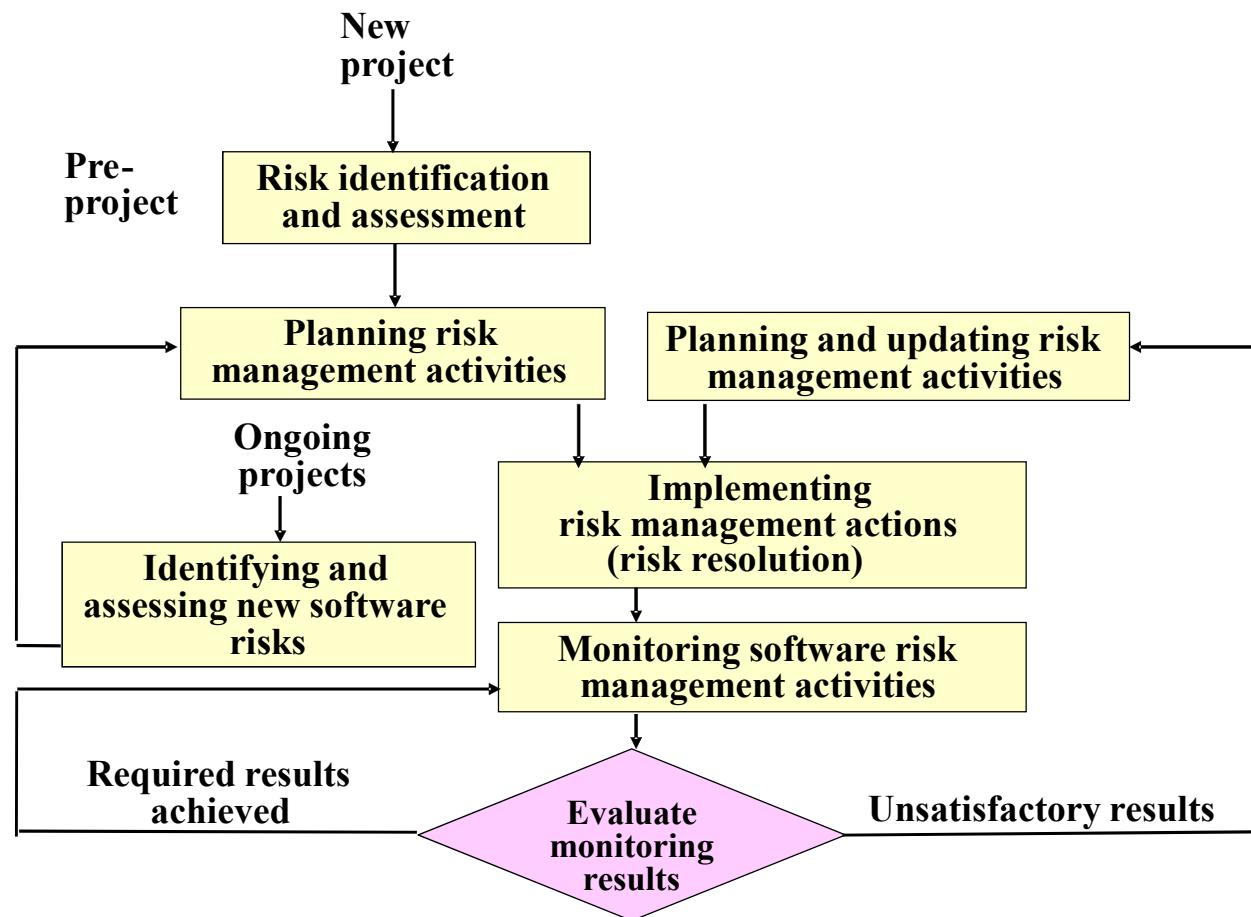
Classes of software development risks

- A. Scheduling and timing risks
- B. System functionality risks
- C. Subcontracting risks
- D. Requirement management risks
- E. Resource usage and performance risks
- F. Personnel management risks

Boehm and Ross's Top 10 software risk items

1. Unrealistic schedules and budgets (A)
2. Developing wrong software functions (B)
3. Developing wrong user interface (B)
4. Shortfalls in externally furnished components (C)
5. Shortfalls in externally performed tasks (C)
6. Gold plating (D)
7. Continuing stream of requirement changes (D)
8. Real-time performance shortfalls (E)
9. Straining computer science capabilities (E)
10. Personnel shortfalls (F)

The software risk management process



Integrating quality activities in the project life cycle

- Software development methodologies:
 - The software development life cycle (SDLC) model
 - The prototyping model
 - The spiral model
 - The object-oriented model
- Factors affecting intensity of SQA activities
- Verification, validation and qualification
- Development and quality plans for small and for internal projects
- A model for SQA defect removal effectiveness and cost

OHT 1.85

The seven SDLC phases, as presented in Fig. 7.1 are:

- Requirements definition
- Analysis
- Design
- Coding
- System tests
- Installation and conversion
- Operation and maintenance

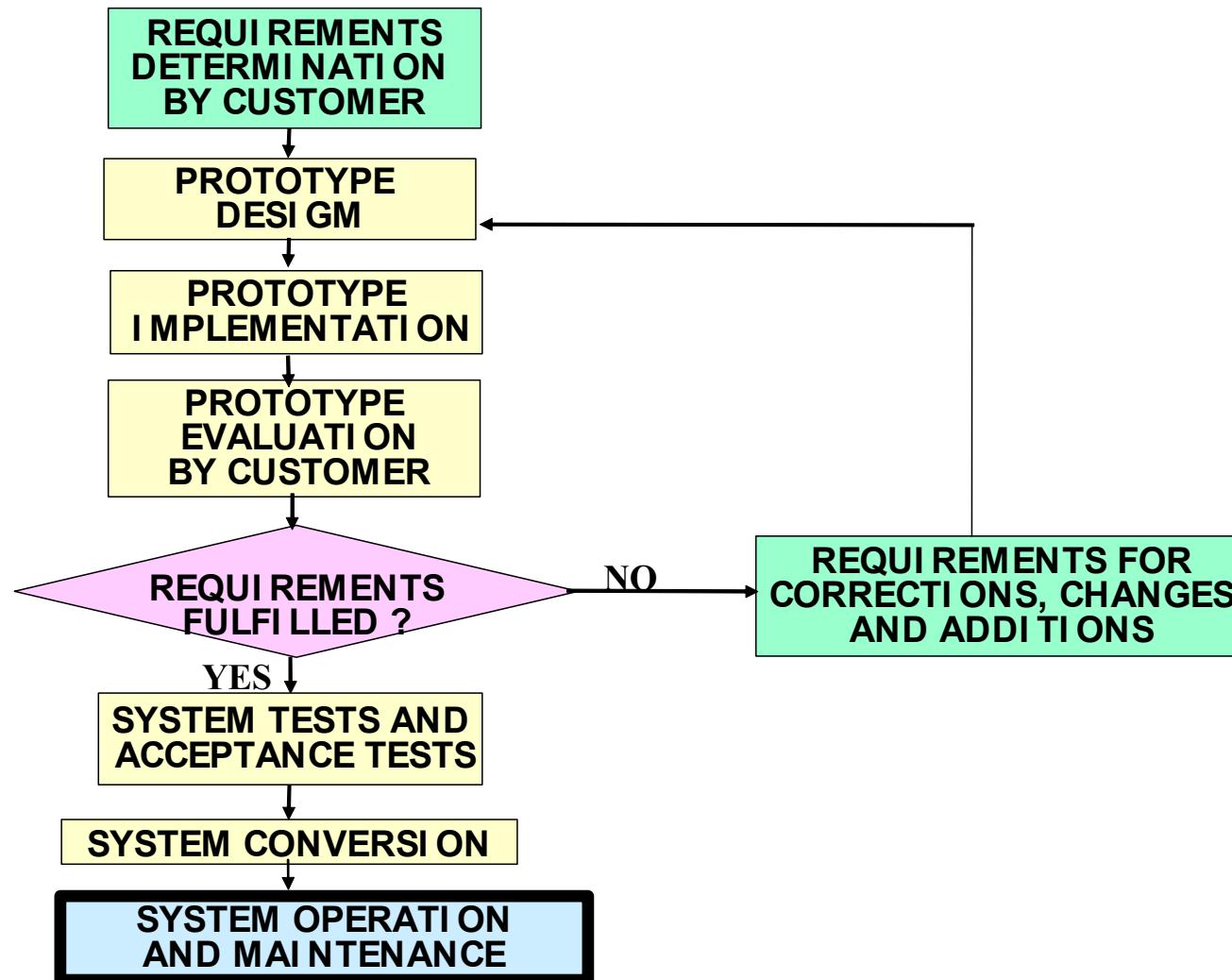
OHT 1.86

Typical inputs and outputs of each phase are listed in the following table:

Development phase	Inputs	Outputs
Requirement definition	-----	a. System requirement document. b. Software requirement document.
Analysis	a. System requirement document. b. Software requirement document.	a. Preliminary design document. b. Software test plan.
Design	a. Preliminary design document. b. Software test plan.	a. Critical design document. b. Software test procedure. c. Software installation plan. d. Software maintenance manual.
Coding	Critical design document.	a. Software test report. b. Project code (complete version prepared for system tests).
Testing	a. Software test plan. b. Software test procedure. c. Project code (complete version prepared for system tests).	a. Tested and corrected version of the project code (approved for installation). b. Software user manual.
Installation and conversion	a. Tested and corrected version of the project code (approved for installation). b. Software installation plan.	

OHT 1.87

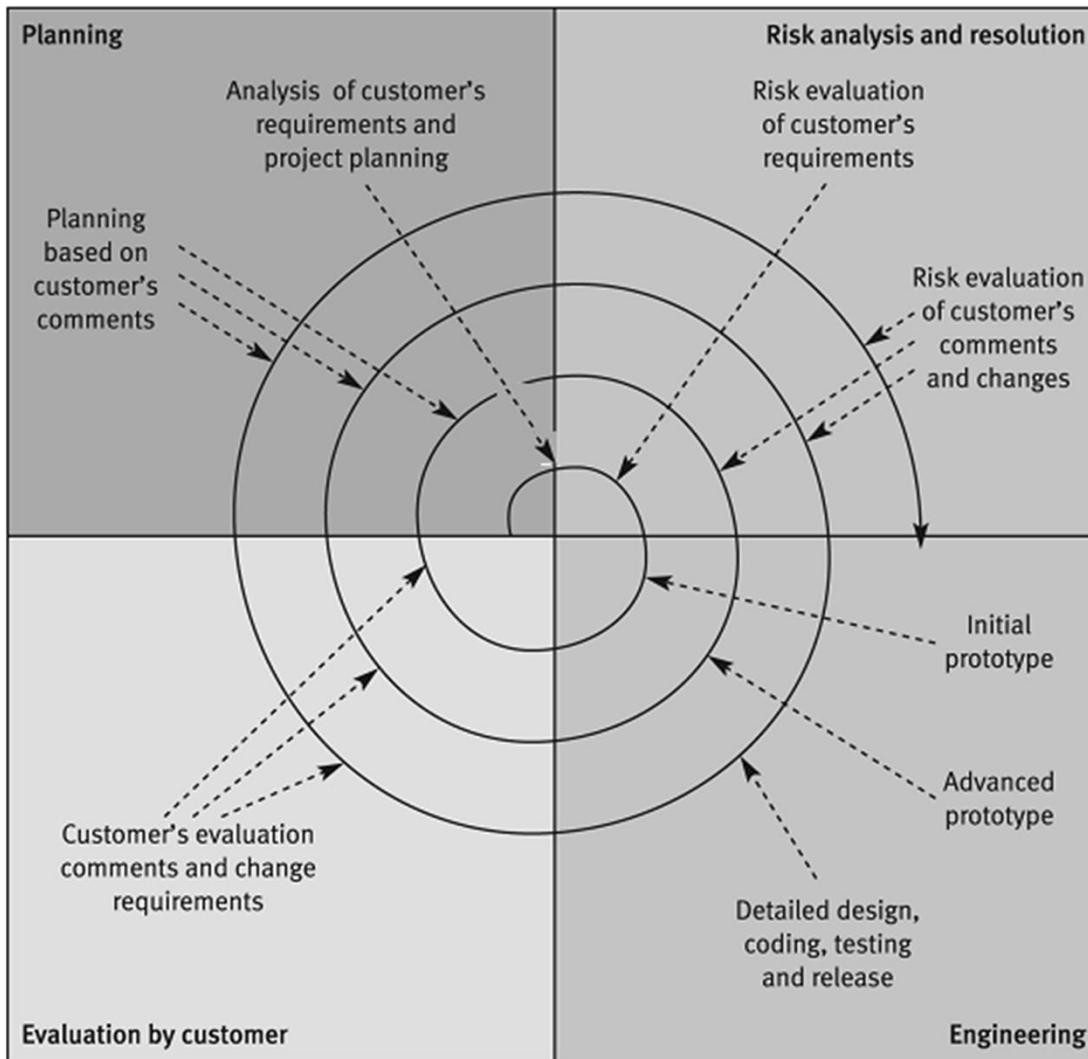
The prototyping model



SQA from theory to implementation

OHT 1.88

The Spiral Model

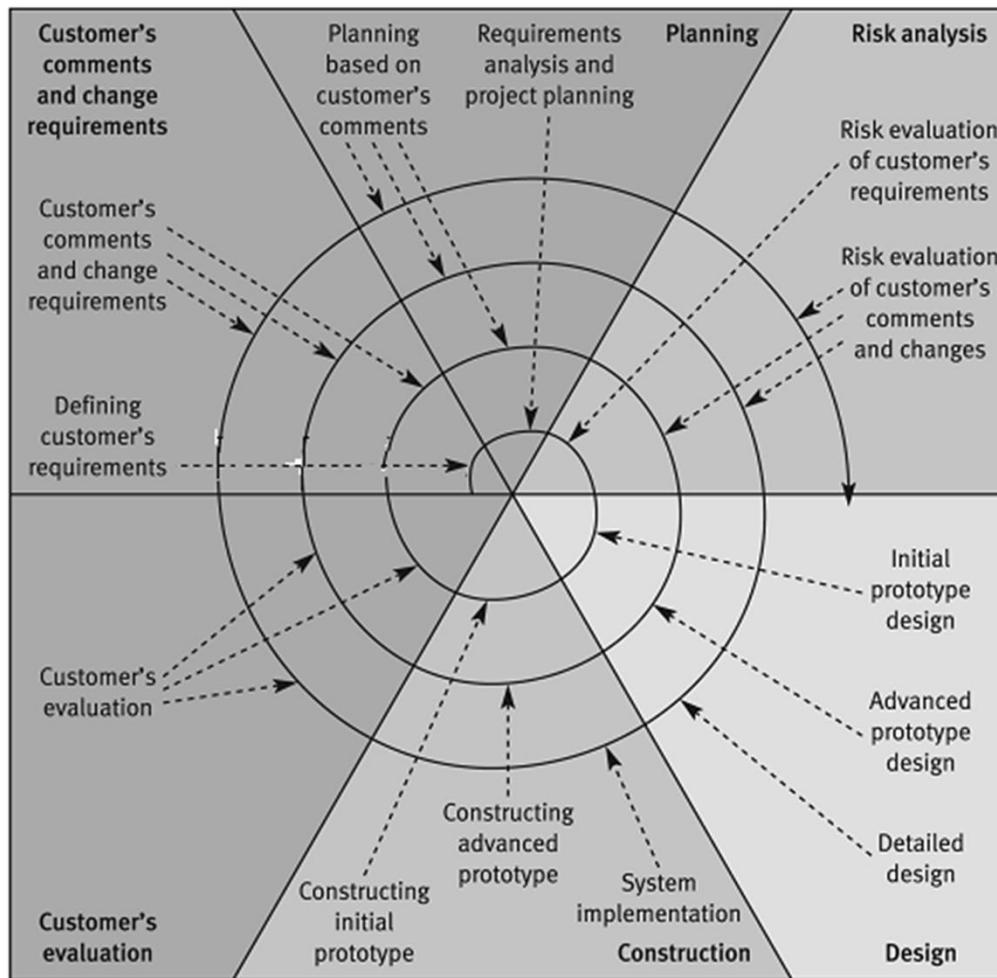


Source: After Boehm 1988 (© 1988 IEEE)

SQA from theory to implementation

OHT 1.89

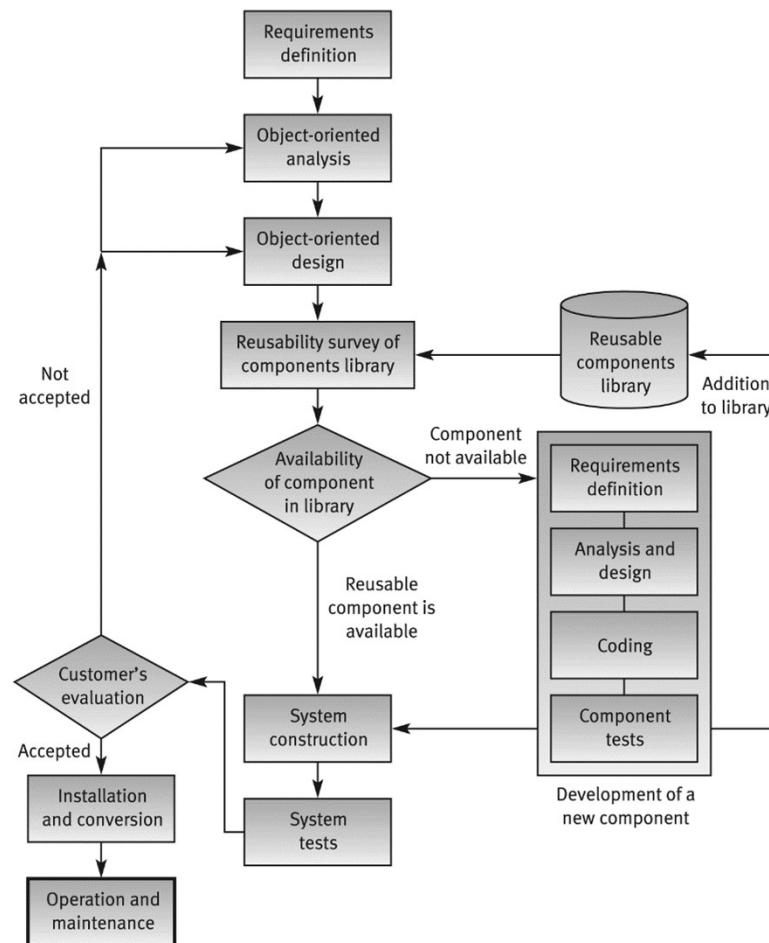
The Advanced Spiral model The Win-Win Model



998 (© 1988 IEEE)

OHT 1.90

Object Oriented Development Model



SQA from theory to implementation

Factors affecting the required intensity of SQA activities

Project factors:

- Project's magnitude
- Project's technical complexity and difficulty
- Extent of reusable software components
- Severity of failure outcomes if the project fails

Team factors:

- The professional qualification of the team members
- Team acquaintance with the project and its experience in the area
- Availability of staff members that can professionally support the team
- Familiarity with the team members, in other words, the percentage of new staff members in the team

OHT 1.92

See Example 1 and Table 7.1 as well as
Example 2 and Table 7.2

OHT 1.93

Three aspects of quality assurance of software product are examined: verification, validation, and qualification.

OHT 1.94

Verification, validation and qualification

Verification – The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase

Validation - The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements

Qualification - The process used to determine whether a system or component is suitable for operational use

IEEE Std 610.12-1990 (IEEE 1990)

SQA from theory to implementation

Model for SQA defect removal effectiveness and cost

The model's quantitative results:

- a. The SQA plan's total effectiveness in removing project defects
- b. The total costs of removal of project defects

OHT 1.96

Defects originating and defect removal costs

Software development phase	Average % of defects originating in phase	Average relative defect removal cost
Requirement specification	15%	1
Design	35%	2.5
Unit coding	30%	6.5
Integration coding	10%	16
Documentation	10%	40
System testing	-----	40
Operation	-----	110

OHT 1.97

Defects removal effectiveness for quality assurance plans

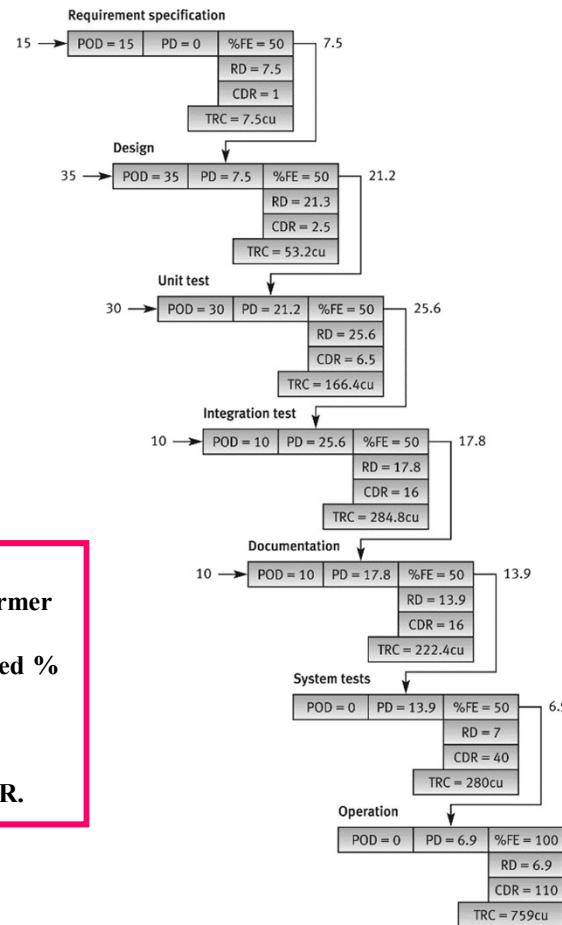
Quality assurance activity	Defects removal effectiveness for standard SQA plan	Defects removal effectiveness for comprehensive SQA plan
Specification requirement review	50%	60%
Design inspection	-----	70%
Design review	50%	60%
Code inspection	-----	70%
Unit test	50%	40%
Integration tests	50%	60%
Documentation review	50%	60%
System test	50%	60%
Operation phase detection	100%	100%

SQA from theory to implementation

OHT 1.98

The standard quality assurance plan

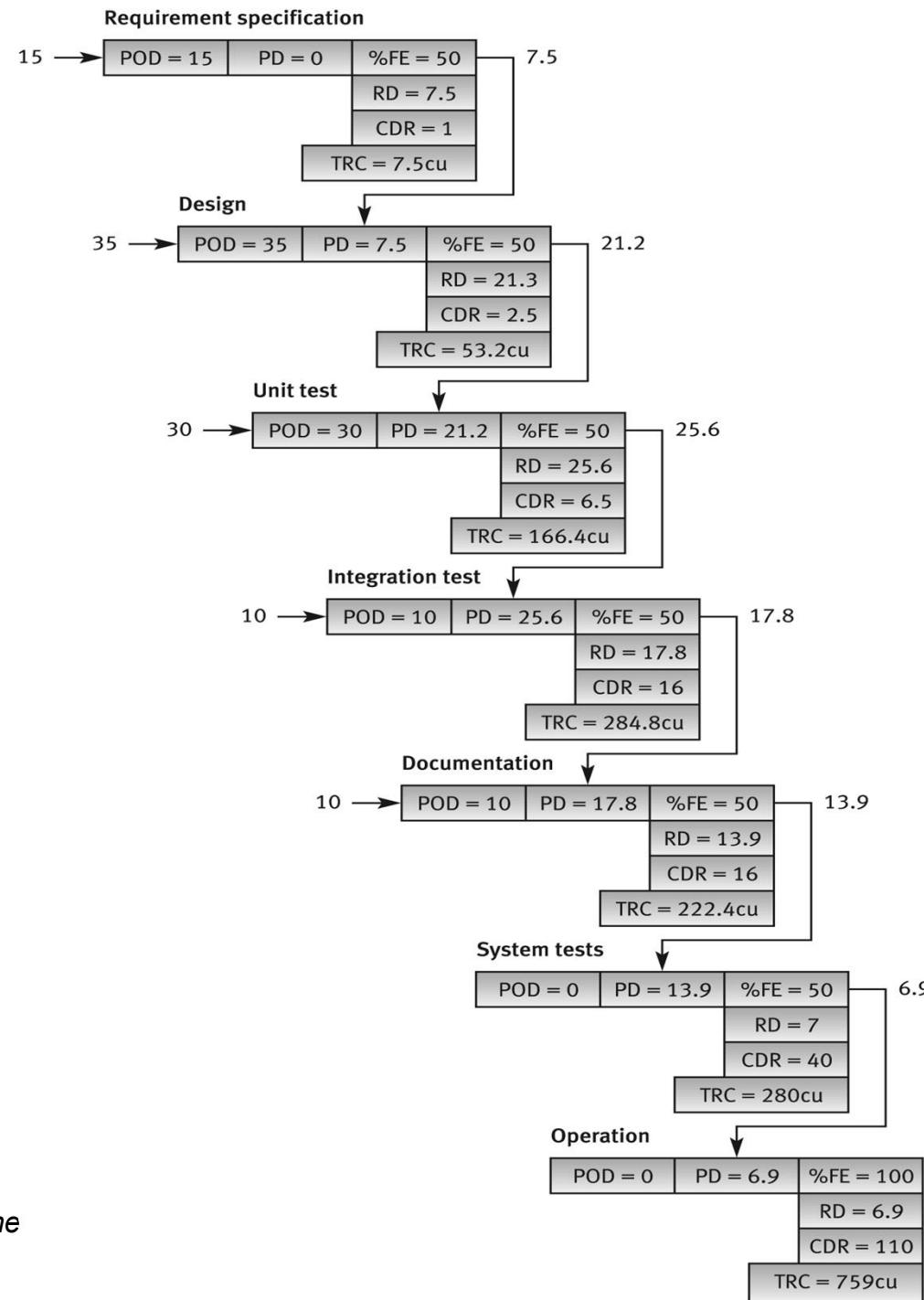
The process of removing 100 defects



- **POD** = Phase Originated Defects
- **PD** = Passed Defects (from former phase or former quality assurance activity)
- **%FE** = % of Filtering Effectiveness (also termed % screening effectiveness)
- **RD** = Removed Defects
- **CDR** = Cost of Defect Removal
- **TRC** = Total Removal Cost. $TRC = RD \times CDR$.

SQA from theory to implementation

OHT 1.99



SQA from theory to impleme

OHT 1.100

- See Figure 7.8 and Table 7.8

Presentation 8

Reviews

- **Review objectives**
- **Formal design reviews (FDRs)**
 - Participants
 - Preparations
 - The FDR session
 - Post-review activities
- **Peer reviews (inspections and walkthroughs)**
 - Participants
 - Preparations
 - The FDR session
 - Post-review activities
 - Peer review coverage
- **Comparison of peer reviews methods**
- **Expert opinions**

Review objectives

Direct objectives

- a. To detect analysis and design errors as well as subjects where corrections, changes and completions are required
- b. To identify new risks likely to affect the project.
- c. To locate deviations from templates, style procedures and conventions.
- d. To approve the analysis or design product. Approval allows the team to continue on to the next development phase.

Indirect objectives

- a. To provide an informal meeting place for exchange of professional knowledge about methods, tools and techniques.
- b. To record analysis and design errors that will serve as a basis for future corrective actions.

Some common formal design (technical) reviews (DRs)

DPR – Development Plan Review

SRSR – Software Requirement Specification Review

PDR – Preliminary Design Review

DDR – Detailed Design Review

DBDR – Data Base Design Review

TPR – Test Plan Review

STPR – Software Test Procedure Review

VDR – Version Description Review

OMR – Operator Manual Review

SMR – Support Manual Review

TRR – Test Readiness Review

PRR – Product Release Review

IPR – Installation Plan Review

Formal design reviews (FDRs) focus on:

- **Participants** (review leader + review team)
- **Preparations**
- **The FDR session**
- **Post-review activities**

Characteristics of a DR leader

- * Knowledge and experience in development of projects of the type reviewed.
Preliminary acquaintance with the current project is not necessary.
- * Seniority at a level similar if not higher than that of the project leader.
- * A good relationship with the project leader and his team.
- * A position external the project team

Review Team

(Non-project staff to make up the majority of the team)

- Senior members of the project
- Senior professionals of other projects & departments
- Customer-user representatives
- Consultants

Team Size: 3-5 members is to be efficient.

Preparations for a DR

- Review leader
 - Appoints the member
 - Schedules the sessions
 - Distributes the document
- Review team
 - To review the document and list the comments (use checklist)
- Development team
 - To prepare a short presentation of the document.

The Agenda of a DR session

- a. A short presentation of the design document.
- b. Comments made by members of the review team.
- c. Verification and validation of comments is discussed to determine the required action items (corrections, changes and additions).
- d. Decisions about the design product (document), which determines the project's progress:
 - *Full approval.*
 - *Partial approval.*
 - *Denial of approval.*

DR post-review activities

a. Preparation of the DR report.

The report's major sections:

- *A summary of the review discussions.*
- *The decision about continuation of the project.*
- *A full list of the required action items — corrections, changes and additions. For each action item, completion date and project team member responsible are listed.*
- *The name(s) of the review team member(s) assigned to follow up.*

b. Follow up performance of the corrections and to examine the corrected sections.

Pressman's 13 golden guidelines for successful DR - 1

Design Review Infrastructure

- Develop checklists for each or common types of design documents.
- Train senior professionals serve as a reservoir for DR teams.
- Periodically analyze past DR effectiveness.
- Schedule the DRs as part of the project plan.

The Design Review Team

- Review teams size should be limited, with 3–5 members being the optimum.

OHT 1.111

Pressman's 13 golden guidelines for successful DR - 2

The Design Review Session

- Discuss professional issues in a constructive way refraining from personalizing the issues.
- Keep to the review agenda.
- Focus on detection of defects by verifying and validating the participants' comments. Refrain from discussing possible solutions.
- In cases of disagreement about an error - end the debate by noting the issue and shifting its discussion to another forum.
- Properly document the discussed comments, and the results of their verification and validation.
- The duration of a review session should not exceed two hours.

Post-Review Activities

- Prepare the review report, including the action items
- Establish follow-up procedure to ensure the satisfactory performance of all the list of action items

Participants of peer reviews

Inspection (more formal)

- Review leader
- The author
- Specialized professionals:
 - Designer
 - Coder or implementer
 - Tester

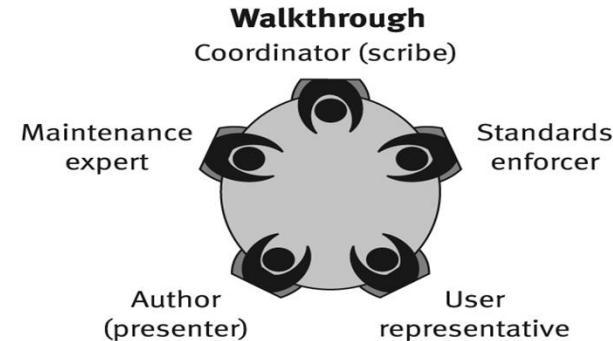
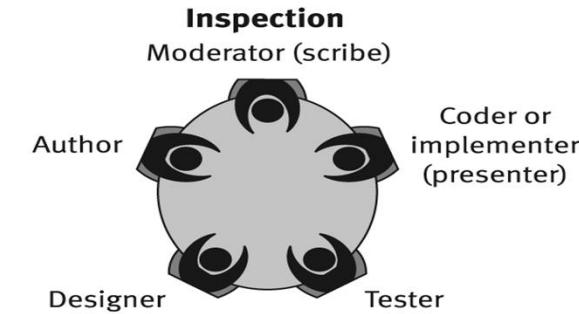
Walkthrough

- Review leader
- The author
- Specialized professionals:
 - Standards enforcer
 - Maintenance expert
 - User representative

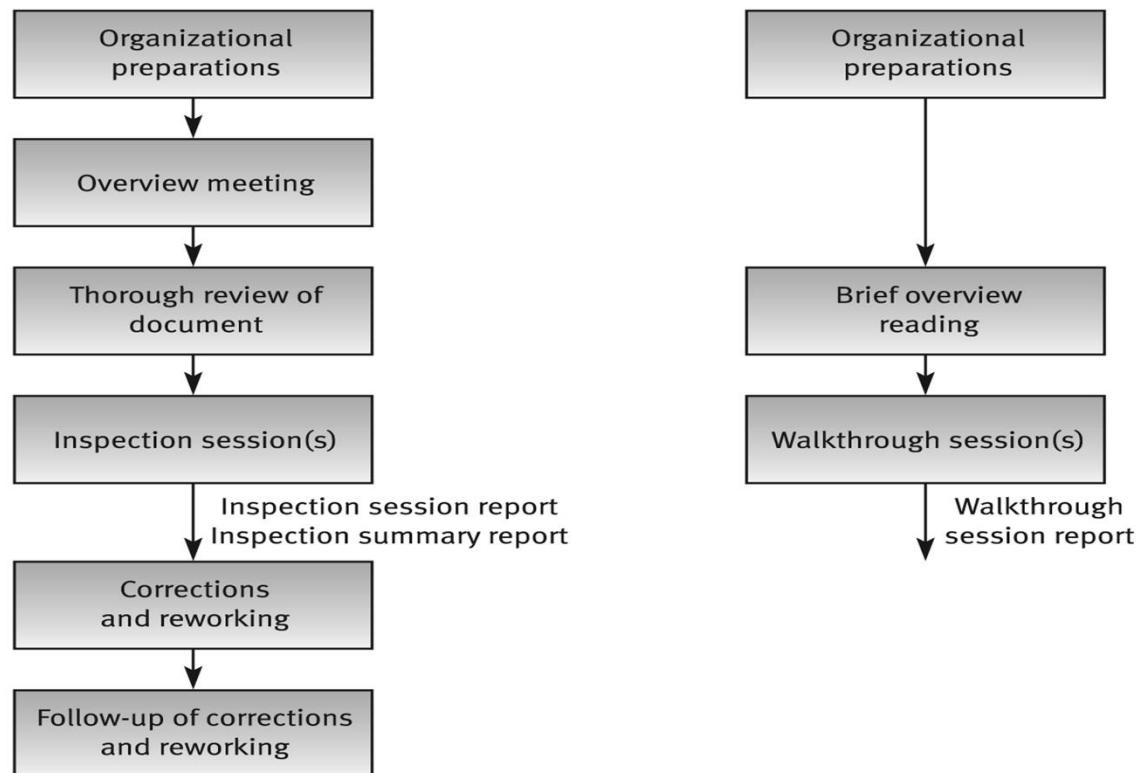
OHT 1.1.3

Inspection vs. Walkthrough

PARTICIPANTS



PROCESS



OHT 1.114

Code inspection effectiveness at Fujitso (Cusumano)

Year	Defect detection method			Defects per 1000 lines of maintained code
	Test %	Design review %	Code inspection %	
1977	85	---	15	0.19
1978	80	5	15	0.13
1979	70	10	20	0.06
1980	60	15	25	0.05
1981	40	30	30	0.04
1982	30	40	30	0.02

SQA from theory to implementation

Sections recommended to be included in or omitted from peer reviews

Sections recommended for inclusion

- Sections of complicated logic
- Critical sections, where defects severely damage essential system capability
- Sections dealing with new environments
- Sections designed by new or inexperienced team members

Sections recommended for omission

- “Straightforward” sections (no complications)
- Sections of a type already reviewed by the team in similar past projects
- Sections that, if faulty, are not expected to effect functionality
- Reused design and code
- Repeated parts of the design and code

OHT 1.116

Comparison of review methodologies - Process of review

Properties	Design review	Inspection	Walkthrough
Overview meeting	No	Yes	No
Participant's preparations	Yes - thorough	Yes - thorough	Yes - brief
Review session	Yes	Yes	Yes
Follow-up of corrections	Yes	Yes	No
Formal training of participants	No	Yes	No
Participant's use of checklists	No	Yes	No
Error-related data collection	Not formally required	Formally required	Not formally required
Review documentation	Formal design review report	1) Inspection session findings report 2) Inspection session summary report	

SQA from theory to implementation

Situations beneficial for expert's participation in reviews

- Insufficient in-house professional capabilities in a specialized area.
- Temporary lack of in-house professionals for review team.
- Indecisiveness caused by major disagreements among the organization's senior professionals.
- In small organizations, where the number of suitable candidates for a review team is insufficient.

Software testing - strategies

- Definitions and objectives
- Software testing strategies
- Software test classifications
- White box testing
 - Data processing and calculation correctness tests
 - Correctness tests and path coverage
 - Correctness tests and line coverage
 - McCabe's cyclomatic complexity metrics
 - Software qualification and reusability testing
 - Advantages and disadvantages of white box testing
- Black box testing
 - Equivalence classes for output correctness tests
 - Other operation factor testing classes
 - Revision factor testing classes
 - Transition factor testing classes
 - Advantages and disadvantages of black box testing

Software tests - definition

Software testing is a **formal** process carried out by a **specialized testing team** in which a software unit, several integrated software units or an entire software package are examined by **running the programs on a computer**. All the associated tests are performed according to **approved test procedures** on **approved test cases**.

Software testing objectives

Direct objectives

- a. To identify and reveal as many errors as possible in the tested software
- b. To bring the tested software, after correction of the identified errors and retesting, to an acceptable level of quality.
- c. To perform the required tests efficiently and effectively, within the limits budgetary and scheduling limitation.

Indirect objectives

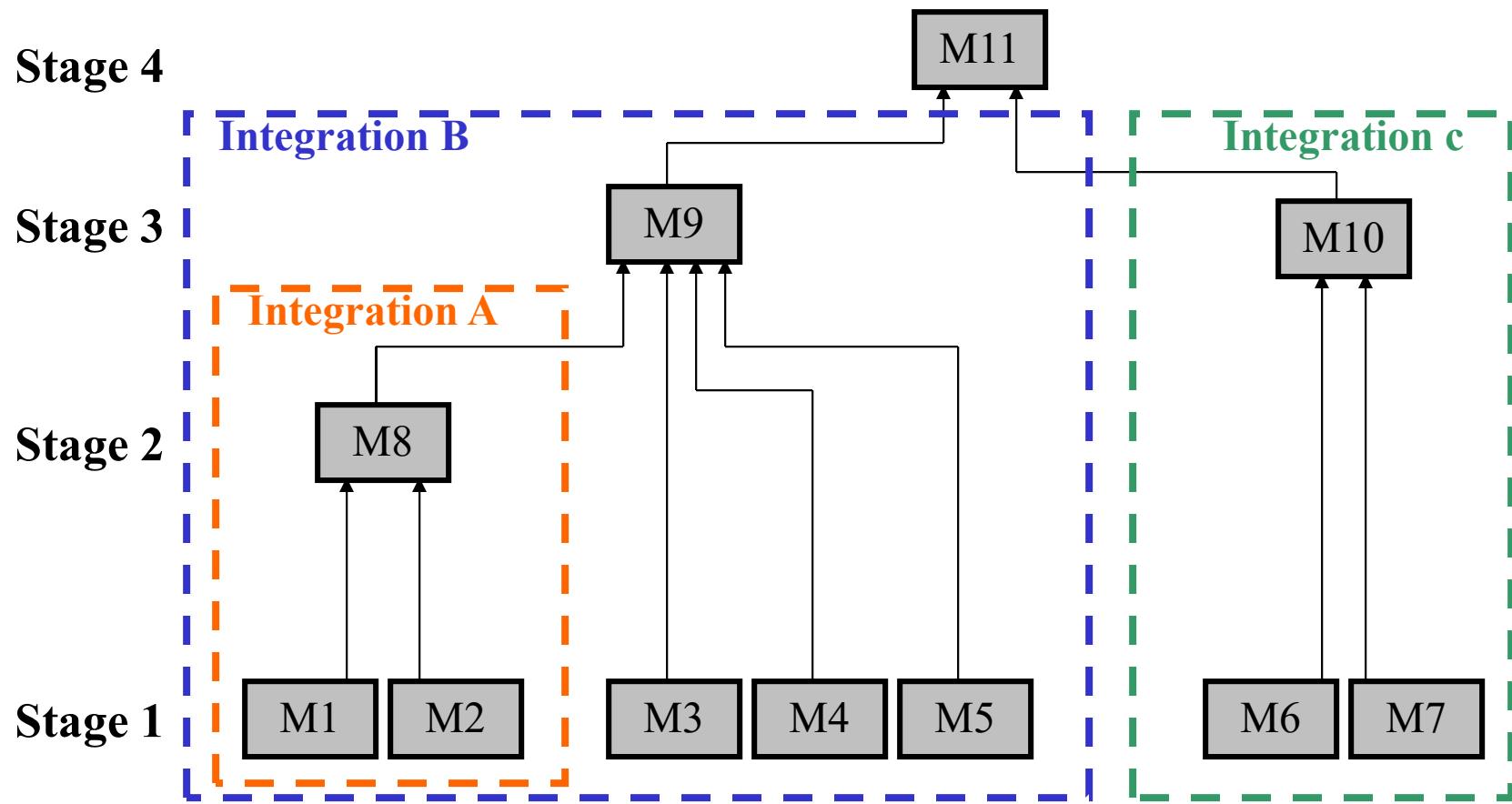
- a. To compile a record of software errors for use in error prevention (by corrective and preventive actions)

Software testing strategies

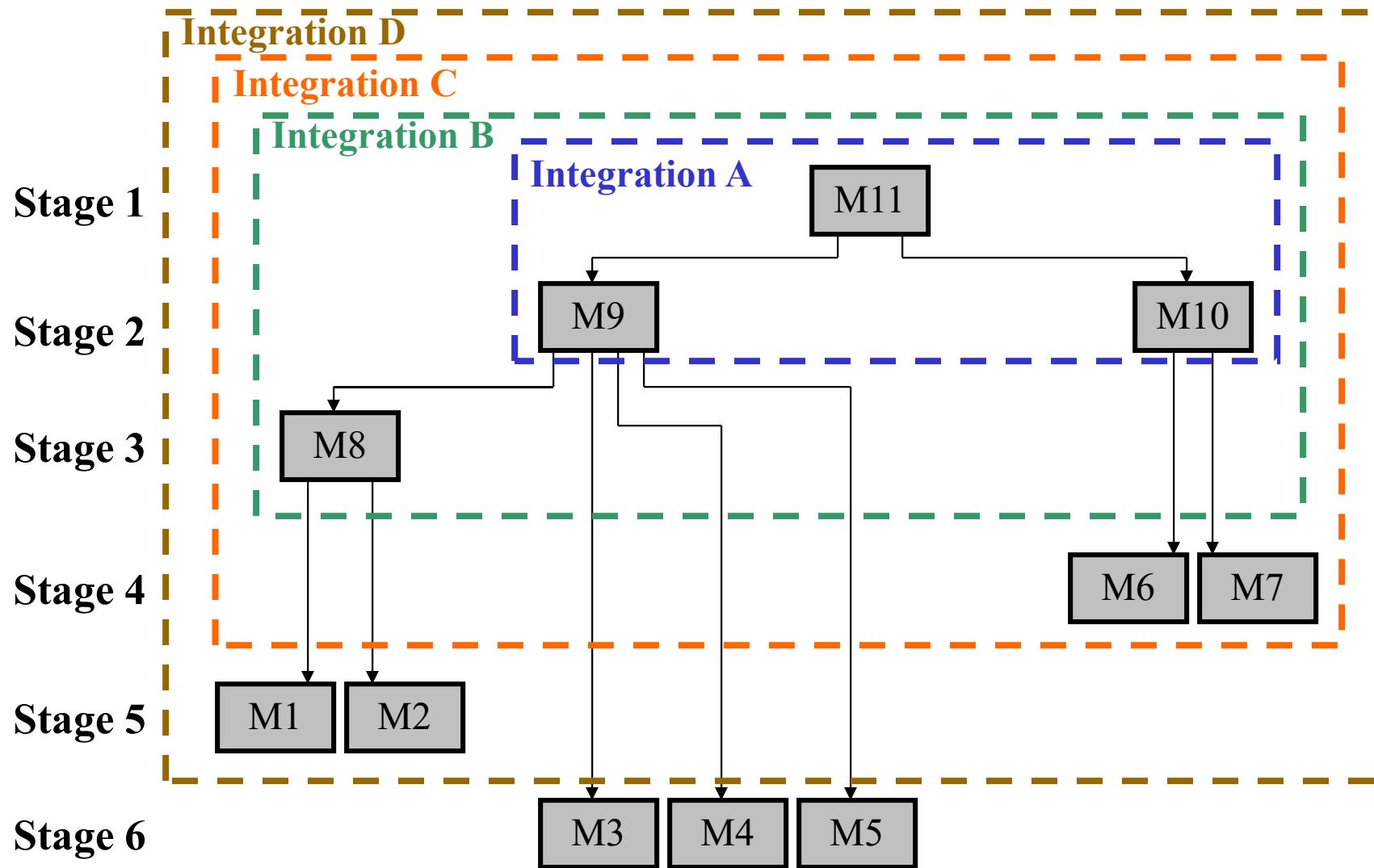
- *Incremental testing strategies:*
 - Bottom-up testing
 - Top-down testing
- *Big bang testing*

OHT 1.122

Bottom-up testing



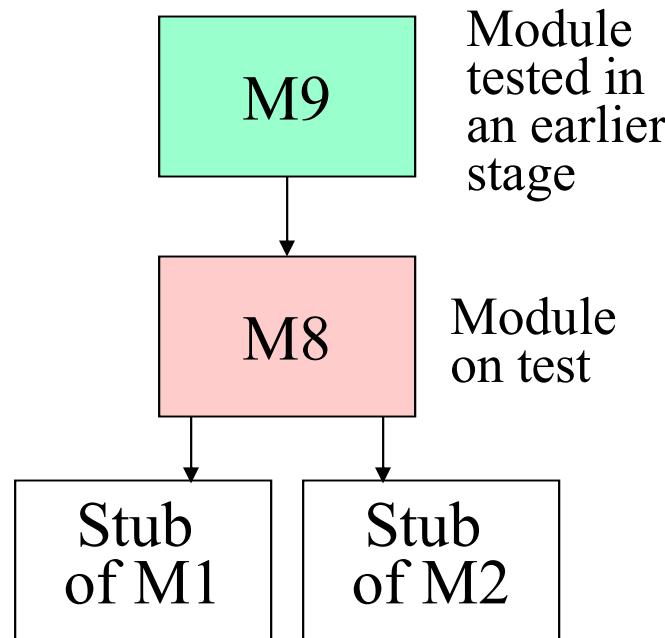
Top-down testing



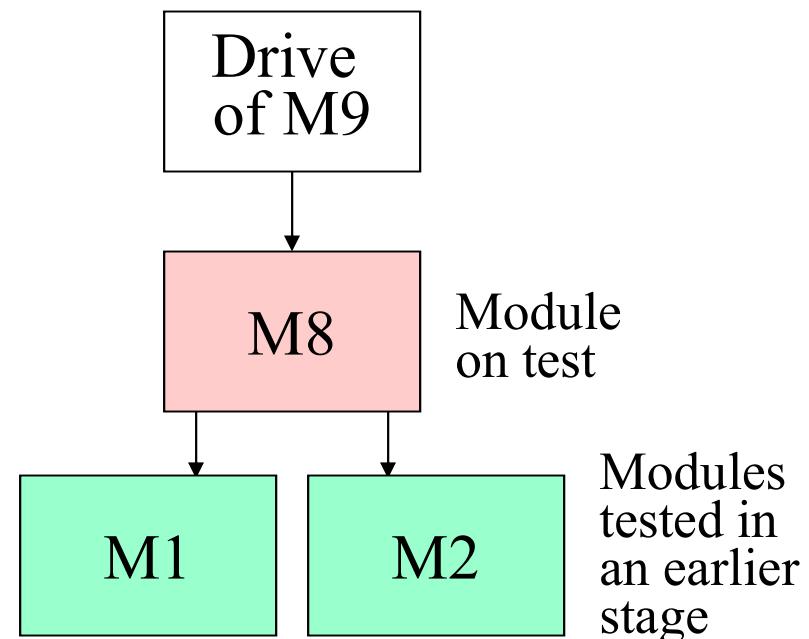
OHT 1.124

Use of stubs and drivers for incremental testing

Top-down testing of module M8



Bottom-up testing of module M8



OHT 1.125

Black box and white box - IEEE definitions

Black box testing

1. Testing that ignores the internal mechanism of the system or component and focuses solely on the outputs in response to selected inputs and execution conditions
2. Testing conducted to evaluate the compliance of a system or component with specified functional requirements

White box testing

Testing that takes into account the internal mechanism of a system or component

White box testing

"Path" vs "line" coverage

Path coverage

- Path coverage of a test is measured by the percentage of all possible program paths included in planned testing.

Line coverage

- Line coverage of a test is measured by the percentage of program code lines included in planned testing.

OHT 1.127

The Imperial Taxi Services (ITS) taximeter

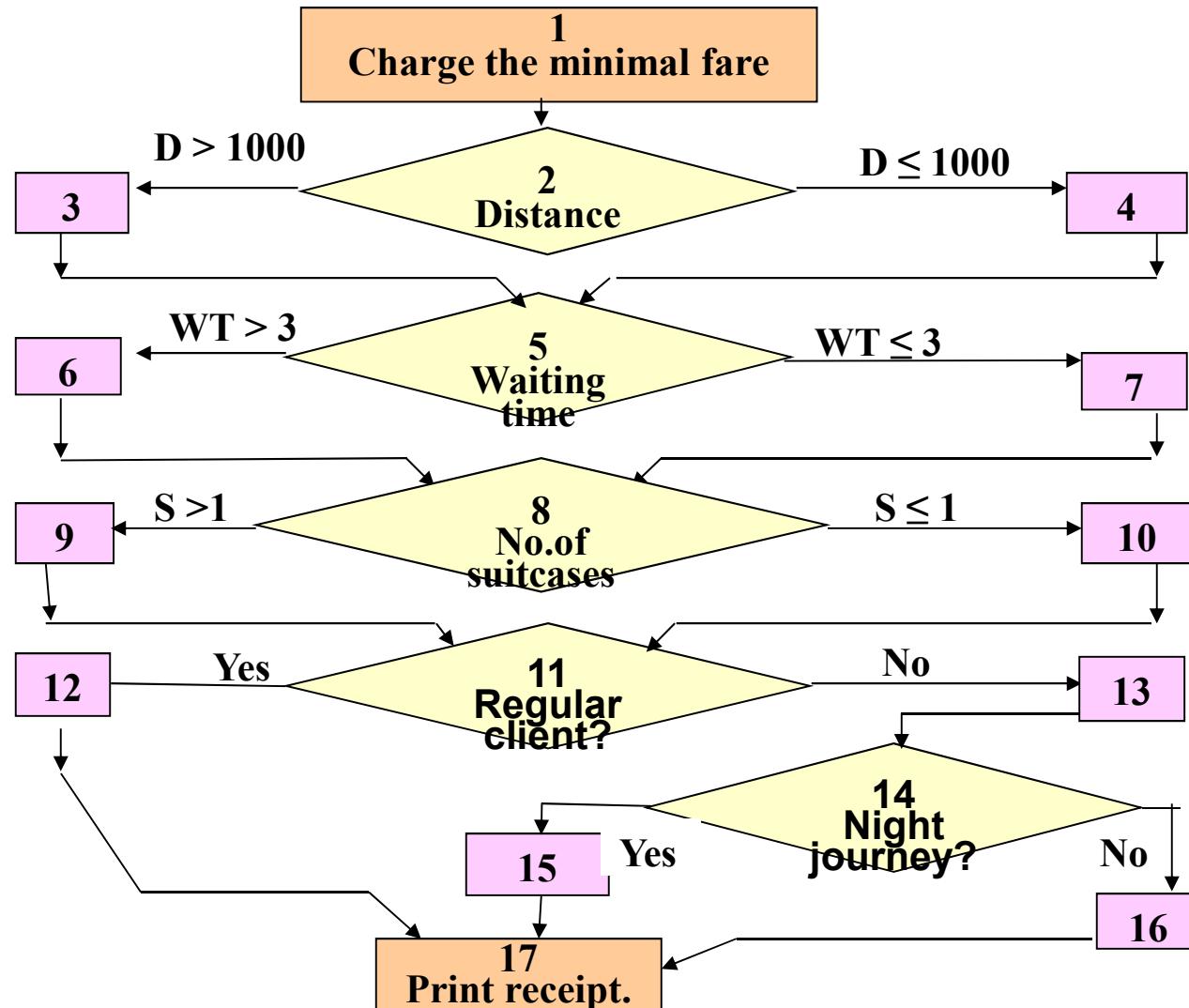
EExample ITS taxi fares for one-time passengers are calculated as follows:

1. Minimal fare: \$2. This fare covers the distance traveled up to 1000 yards and waiting time (stopping for traffic lights or traffic jams, etc.) of up to 3 minutes.
2. For every additional 250 yards or part of it: 25 cents.
3. For every additional 2 minutes of stopping or waiting or part thereof: 20 cents.
4. One suitcase: 0 change; each additional suitcase: \$1.
5. Night supplement: 25%, effective for journeys between 21.00 and 06.00.

Regular clients are entitled to a 10% discount and are not charged the night supplement.

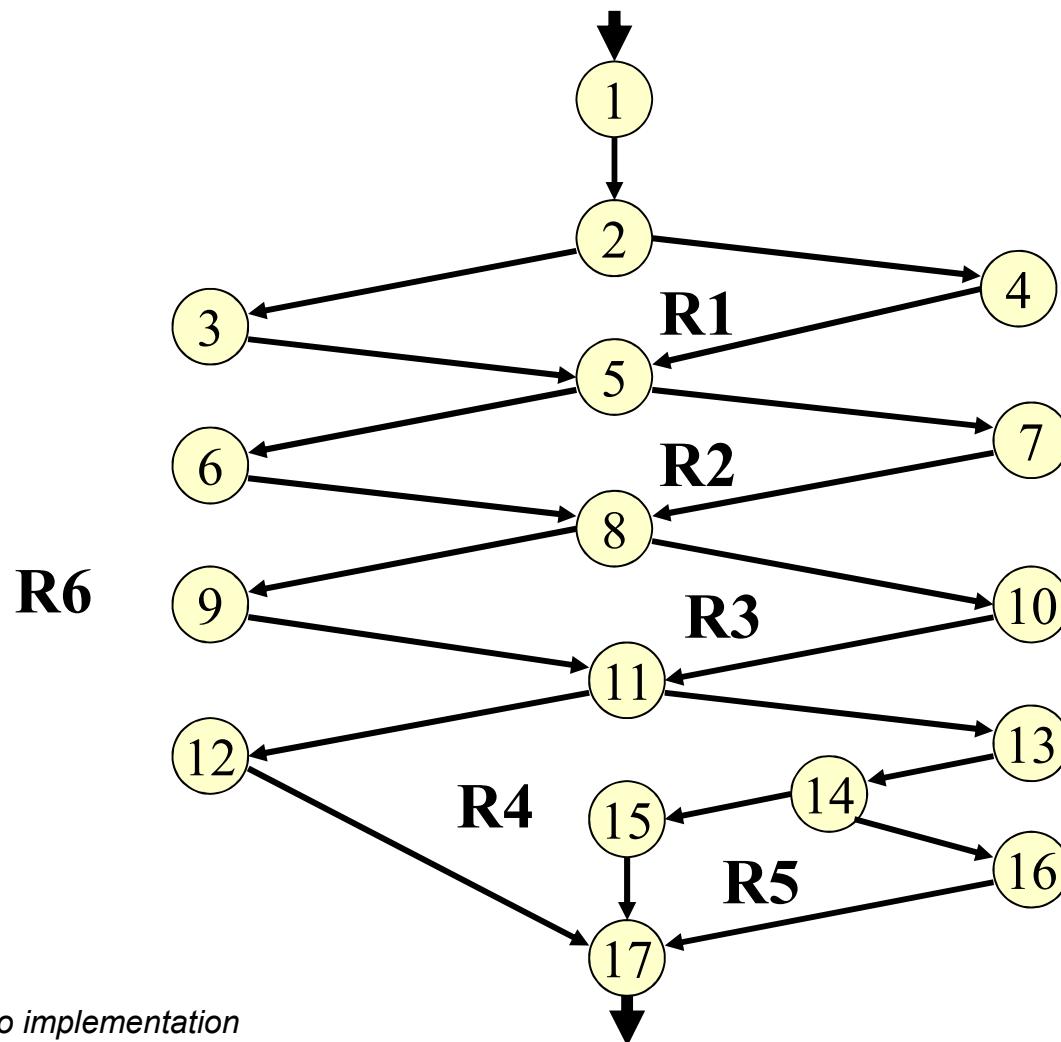
OHT 1.128

ITS - Flow chart



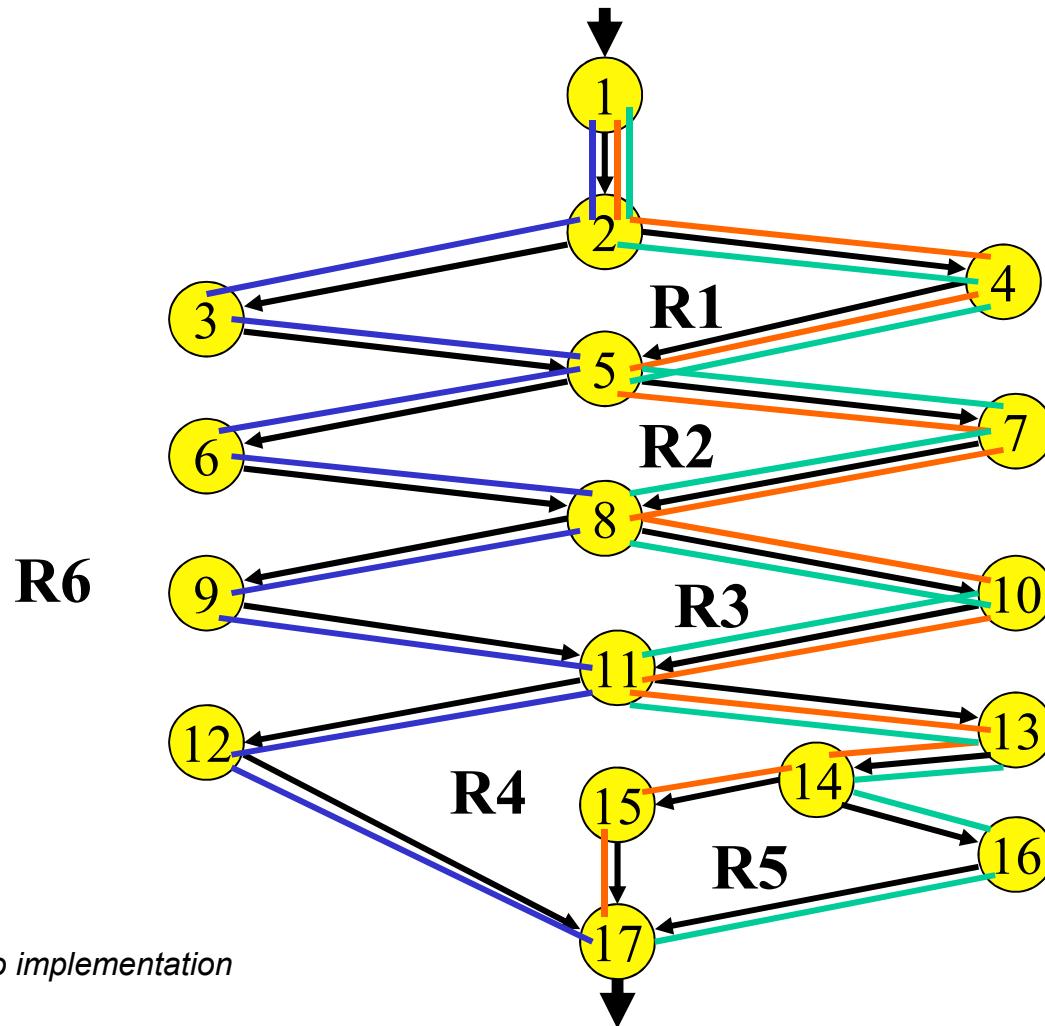
OHT 1.129

ITS - Program flow graph



OHT 1.130

ITS - The minimum number of paths for full line coverage



OHT 1.131

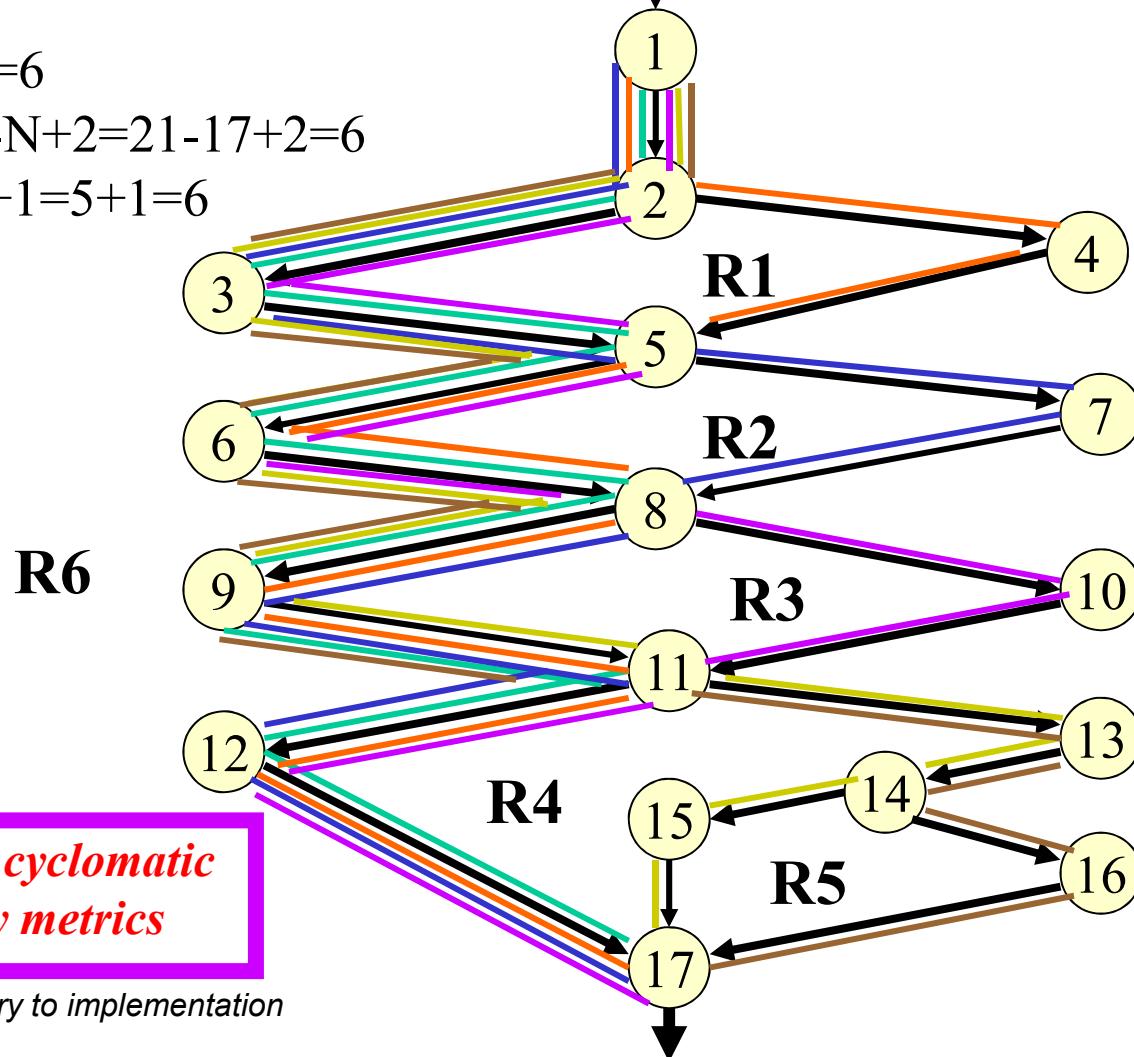
ITS - The maximum set of independent paths

$$V(G)=R=6$$

$$V(G)=E-N+2=21-17+2=6$$

$$V(G)=P+1=5+1=6$$

R=Regions
N=Nodes
E=Edges
P=Decisions



McCabe's cyclomatic complexity metrics

SQA from theory to implementation

Advantages and disadvantages of white box testing

Advantages:

- * Direct determination of software correctness as expressed in the processing paths, including algorithms.
- * Allows performance of line coverage follow up.
- * Ascertains quality of coding work and its adherence to coding standards.

Disadvantages :

- * The vast resources utilized, much above those required for black box testing of the same software package.
- * The inability to test software performance in terms of availability (response time), reliability, load durability, etc.

Equivalence class partitioning (EC)

A black box method aimed at increasing the efficiency of testing and, at the same time, improving coverage of potential error conditions.

Equivalence class partitioning (EC)

- An equivalence class (EC) is a set of input variable values that produce the same output results or that are processed identically.
- EC boundaries are defined by a single numeric or alphabetic value, a group of numeric or alphabetic values, a range of values, and so on.
- An EC that contains only valid states is defined as a "valid EC," whereas an EC that contains only invalid states is defined as the "invalid EC."
- In cases where a program's input is provided by several variables, valid and invalid ECs should be defined for each variable.

Equivalence class partitioning (EC)

According to the equivalence class partitioning method:

- Each valid EC and each invalid EC are included in at least one test case.
- Definition of test cases is done separately for the valid and invalid ECs. 34,751
- In defining a test case for the valid ECs, we try to cover as many as possible “new” ECs in that same test case.
- In defining invalid ECs, we must assign one test case to each “new” invalid EC, as a test case that includes more than one invalid EC may not allow the tester to distinguish between the program’s separate reactions to each of the invalid ECs.
- Test cases are added as long as there are uncovered ECs.

OHT 1.136

Entrance ticket price table - The Pool

Day	Mon., Tue., Wed., Thu, Fri.				Sat., Sun.			
Visitor's status	Ot	Ot	Mem	Mem	Ot	Ot	Mem	Mem
Entry hour	6.00-19.00	19.01-24.00	6.00-19.00	19.01-24.00	6.00-19.00	19.01-24.00	6.00-19.00	19.01-24.00
Age: 0.00-16.00	\$5	\$6	\$2.50	\$3	\$7.50	\$9	\$3.50	\$4
Age 16.01-60.00	\$10	\$17	\$5	\$6	\$15	\$18	\$7	\$8
Age: 60.01-120.00	\$8	\$8	\$4	\$4	\$12	\$12	\$5.50	\$5.50

SQA from theory to implementation

OHT 1.137

Test cases - The ticket price module

Test case type	Test case no.	Day of week	Visitor's status	Entry hour	Visitor's age	Test case result
Valid ECs	1	Mon.	Ot	7.55	8.4	\$5
	2	Sat.	Mem	20.44	42.7	\$8
	3	Sat.	Mem	22.44	65.0	\$5.50
	4	Sat.	Mem	6.00	0.0	\$3.50
	5	Sat.	Mem	19.00	16.0	\$3.50
	6	Sat.	Mem	19.01	16.01	\$8
	7	Sat.	Mem	19.01	60.0	\$8
	8	Sat.	Mem	24.00	60.01	\$5.50
	9	Sat.	Mem	24.00	120.0	\$5.50
Invalid ECs	10	Mox.	Ot	7.55	8.4	Invalid day
	11	Mon.	88	7.55	8.4	Invalid visitor's status
	12	Mon.	Ot	4.40	8.4	Invalid entry hour
	13	Mon.	Ot	8@	8.4	Invalid entry hour
	14	Mon.	Ot	7.55	TTR	Invalid visitor's age
	15	Mon.	Ot	7.55	150.1	Invalid visitor's age

SQA from theory to implementation

Other operation factor testing classes

Quality requirement factor	Test class
Correctness	(1) Documentation tests (2) Availability tests (reaction time)
Reliability	Reliability tests
Efficiency	Stress tests
Integrity	Software system security tests
Usability	(1) Training usability tests (2) Operational usability tests

Revision factor testing classes

- Maintainability tests
 - structure
 - documentation
- Flexibility tests
 - modular structure
 - parametric options
- Testability tests
 - ease of testing, intermediate results, log files

Transition factor testing classes

- Portability tests
 - standards adhered to
 - estimate the sources required for transfer
- Reusability tests
 - if reusability standards are adhered to
- Test for interoperability tests

Advantages and disadvantages of black box testing

Advantages:

- * Allows us to carry out the majority of testing classes, most of which can be implemented solely by black box tests, i.e. load tests and availability tests.
- * For testing classes that can be carried out by both white and black box tests, black box testing requires fewer resources.

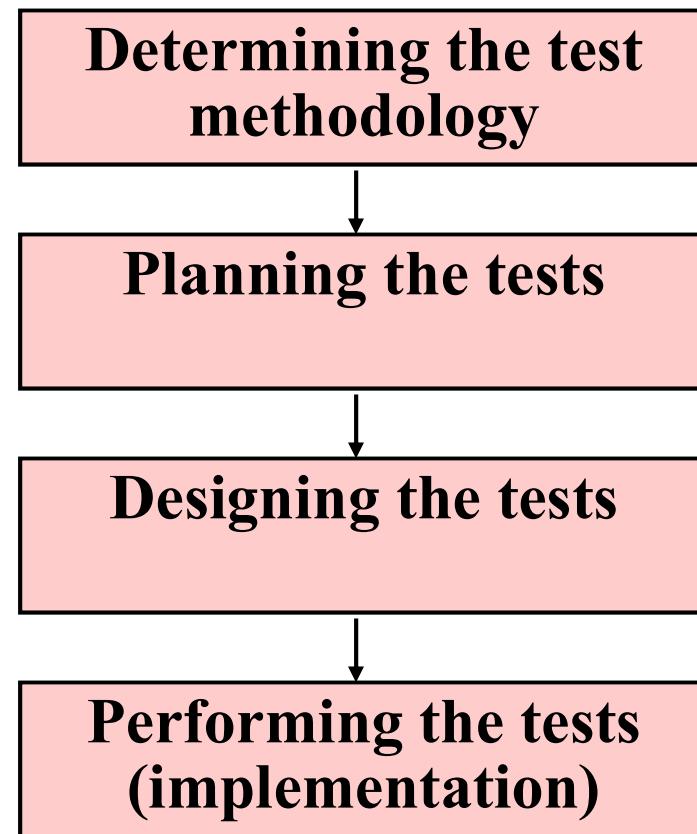
Disadvantages:

- * Possibility that coincidental aggregation of several errors will produce the correct response for a test case, and prevent error detection.
- * Absence of control of line coverage. There is no easy way to specify the parameters of the test cases required to improve coverage.
- * Impossibility of testing the quality of coding and its strict adherence to the coding standards.

Software testing - Implementation

- **The testing process**
 - Determining the test methodology phase
 - Planning the tests
 - Test design
 - Test implementation
- **Test case design**
 - Test case data components
 - Test case sources
- **Automated testing**
 - The process of automated testing
 - Types of automated testing
 - Advantages and disadvantages of automated testing
- **Alpha and beta site testing programs**

The testing process



Classification of software failure damages – Damages to Customers and Users

- 1. Endangers the safety of human beings**
- 2. Affects an essential organizational function with no system replacement capability available**
- 3. Affects functioning of firmware, causing malfunction of an entire system**
- 4. Affects an essential organizational function but a replacement is available**
- 5. Affects proper functioning of software packages for business applications**
- 6. Affects proper functioning of software packages for a private customer**
- 7. Affects functioning of a firmware application but without affecting the entire system.**
- 8. Inconveniences the user but does not prevent accomplishment of the system's capabilities**

Classification of software failure damages – **Damages to the software developer**

1. Financial losses

- * Damages paid for physical injuries
- * Damages paid to organizations for malfunctioning of software
- * Purchase cost reimbursed to customers
- * High maintenance expenses for repair of failed systems

2. Non-quantitative damages

- * Expected to affect future sales
- * Substantially reduced current sales

Issues affecting software risk level

Module/application issues

1. Magnitude
2. Complexity and difficulty
3. Percentage of original software (vs. percentage of reused software)

Programmer issues

4. Professional qualifications
5. Experience with the module's specific subject matter.
6. Availability of professional support (backup of knowledgeable and experience).
7. Acquaintance with the programmer and the ability to evaluate his/her capabilities.

OHT 1.147

Super Teacher – alternative combined rating methods

Application	Damage Severity Factor A	Damage Severity Factor B	Combined rating method		
			A +B	7xA+2xB	A x B
1. Input of test results	3	2	5 (4)	25 (5)	6 (4)
2. Interface for input and output of pupils' data to and from other teachers	4	4	8 (1)	36 (1)	16 (1)
3. Preparation of lists of low achievers	2	2	4 (5-6)	18 (7)	4 (5-6)
4. Printing letters to parents of low achievers	1	2	3 (7-8)	11 (8)	2 (8)
5. Preparation of reports for the school principal	3	3	6 (3)	27 (3)	9 (3)
6. Display of a pupil's achievements profile	4	3	7 (2)	34 (2)	12 (2)
7. Printing of pupil's term report card	3	1	3 (7-8)	23 (6)	3 (7)
8. Printing of pupil's year-end report card	4	1	4 (5-6)	26 (4)	4 (5-6)

SQA from theory to implementation

Software test plan (STP) - template

1 Scope of the tests

- 1.1 The software package to be tested (name, version and revision)
- 1.2 The documents that provide the basis for the planned tests

2 Testing environment

- 2.1 Sites
- 2.2 Required hardware and firmware configuration
- 2.3 Participating organizations
- 2.4 Manpower requirements
- 2.5 Preparation and training required of the test team

Software test plan (STP) - template (cont.)

3 Tests details (for each test)

- 3.1 Test identification
- 3.2 Test objective
- 3.3 Cross-reference to the relevant design document and the requirement document
- 3.4 Test class
- 3.5 Test level (unit, integration or system tests)
- 3.6 Test case requirements
- 3.7 Special requirements (e.g., measurements of response times, security requirements)
- 3.8 Data to be recorded

4 Test schedule (for each test or test group) including time estimates for:

- 4.1 Preparation
- 4.2 Testing
- 4.3 Error correction
- 4.4 Regression tests

Software test description (STD) - template

1 Scope of the tests

- 1.1 The software package to be tested (name, version and revision)
- 1.2 The documents providing the basis for the designed tests (name and version for each document)

2 Test environment (for each test)

- 2.1 Test identification (the test details are documented in the STP)
- 2.2 Detailed description of the operating system and hardware configuration and the required switch settings for the tests
- 2.3 Instructions for software loading

3. Testing process

- 3.1 Instructions for input, detailing every step of the input process
- 3.2 Data to be recorded during the tests

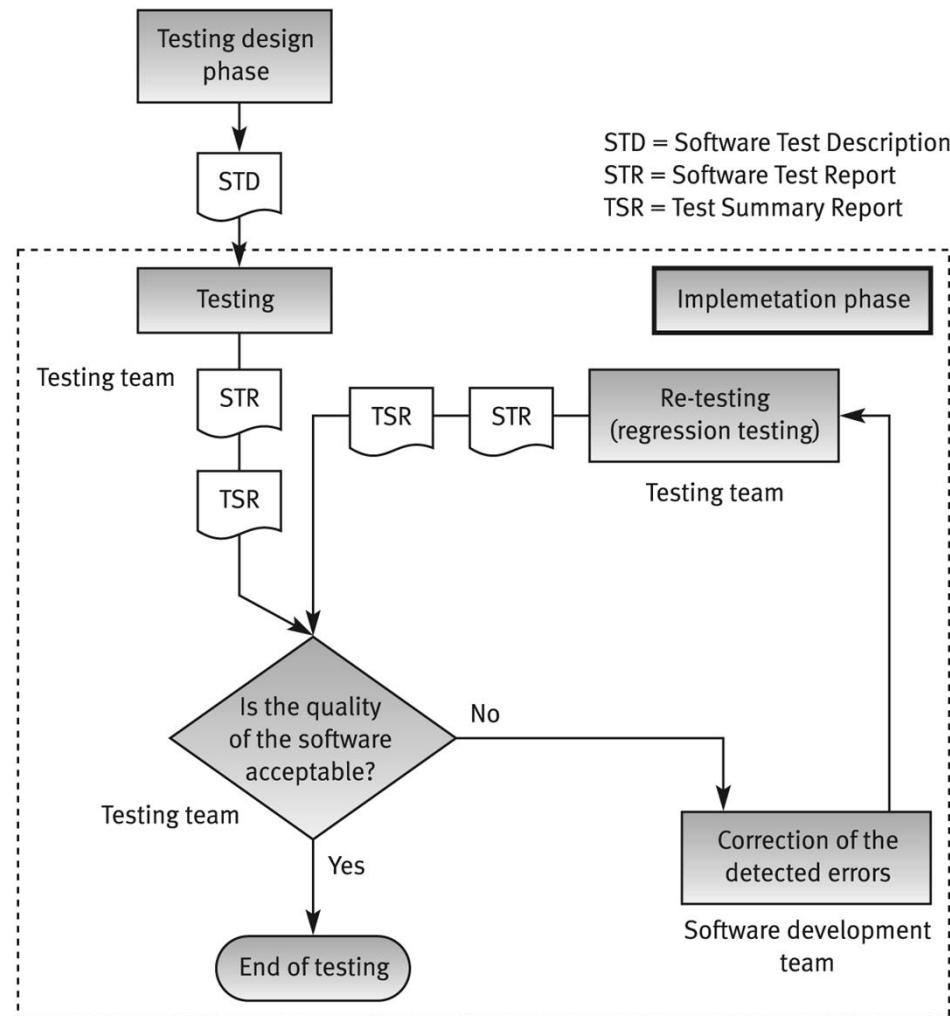
4. Test cases (for each case)

- 4.1 Test case identification details
- 4.2 Input data and system settings
- 4.3 Expected intermediate results (if applicable)
- 4.4 Expected results (numerical, message, activation of equipment, etc.)

5. Actions to be taken in case of program failure/cessation

6. Procedures to be applied according to the test results summary

Implementation phase activities



Software test report (STR) - template

1. Test identification, site, schedule and participation

- 1.1 The tested software identification (name, version and revision)
- 1.2 The documents providing the basis for the tests (name and version for each document)
- 1.3 Test site
- 1.4 Initiation and concluding times for each testing session
- 1.5 Test team members
- 1.6 Other participants
- 1.7 Hours invested in performing the tests

2. Test environment

- 2.1 Hardware and firmware configurations
- 2.2 Preparations and training prior to testing

OHT 1.153

Software test report (STR) - template (cont.)

3. Test results

- 3.1 Test identification
- 3.2 Test case results (for each test case individually)

4. Summary tables for total number of errors, their distribution and types

- 4.1 Summary of current tests
- 4.2 Comparison with previous results (for regression test summaries)

5. Special events and testers' proposals

- 5.1 Special events and unpredicted responses of the software during testing
- 5.2 Problems encountered during testing.
- 5.3 Proposals for changes in the test environment, including test preparations
- 5.4 Proposals for changes or corrections in test procedures and test case files

Test cases - types of expected results

Management information systems - expected results:

- Numerical
- Alphabetic (name, address, etc.)
- Error message. Standard output informing user about missing data, erroneous data, unmet conditions, etc.

Real-time software and firmware - expected results:

- Numerical and/or alphabetic messages displayed on a monitor's screen or on the equipment display.
- Activation of equipment or initiation of a defined operation.
- Activation of an operation, a siren, warning lamps and the like as a reaction to identified threatening conditions.
- Error message. Standard output to inform the operator about missing data, erroneous data, etc.

Test case sources

- **Random samples** of real life cases
(Preferable – Stratified sampling of real life cases)
- **Synthetic** test cases (simulated test cases)

OHT 1.156

Comparison of automated and manual testing by phase

Testing process phase	Automated testing	Manual testing
Test planning	M	M
Test design	M	M
Preparing test cases	M	M
Performance of the tests	A	M
Preparing the test log and test reports	A	M
Regression tests	A	M
Preparing the tests log and test reports including comparative reports	M	M
Test planning	M	M
Test design	A	M

M = phase performed manually, A= phase performed automatically

Advantages and disadvantages of automated testing

Advantages

- Accuracy and completeness of performance.
- Accuracy of results log and summary reports.
- Comprehensiveness of information.
- Few manpower resources required for performing of tests.
- Shorter duration of testing.
- Performance of complete regression tests.
- Performance of test classes beyond the scope of manual testing.

Disadvantages

- High investments required in package purchasing and training.
- High package development investment costs.
- High manpower requirements for test preparation.
- Considerable testing areas left uncovered.

Advantages and disadvantages of beta site tests

Advantages

- Identification of unexpected errors.
- A wider population in search of errors.
- Low costs.

Disadvantages

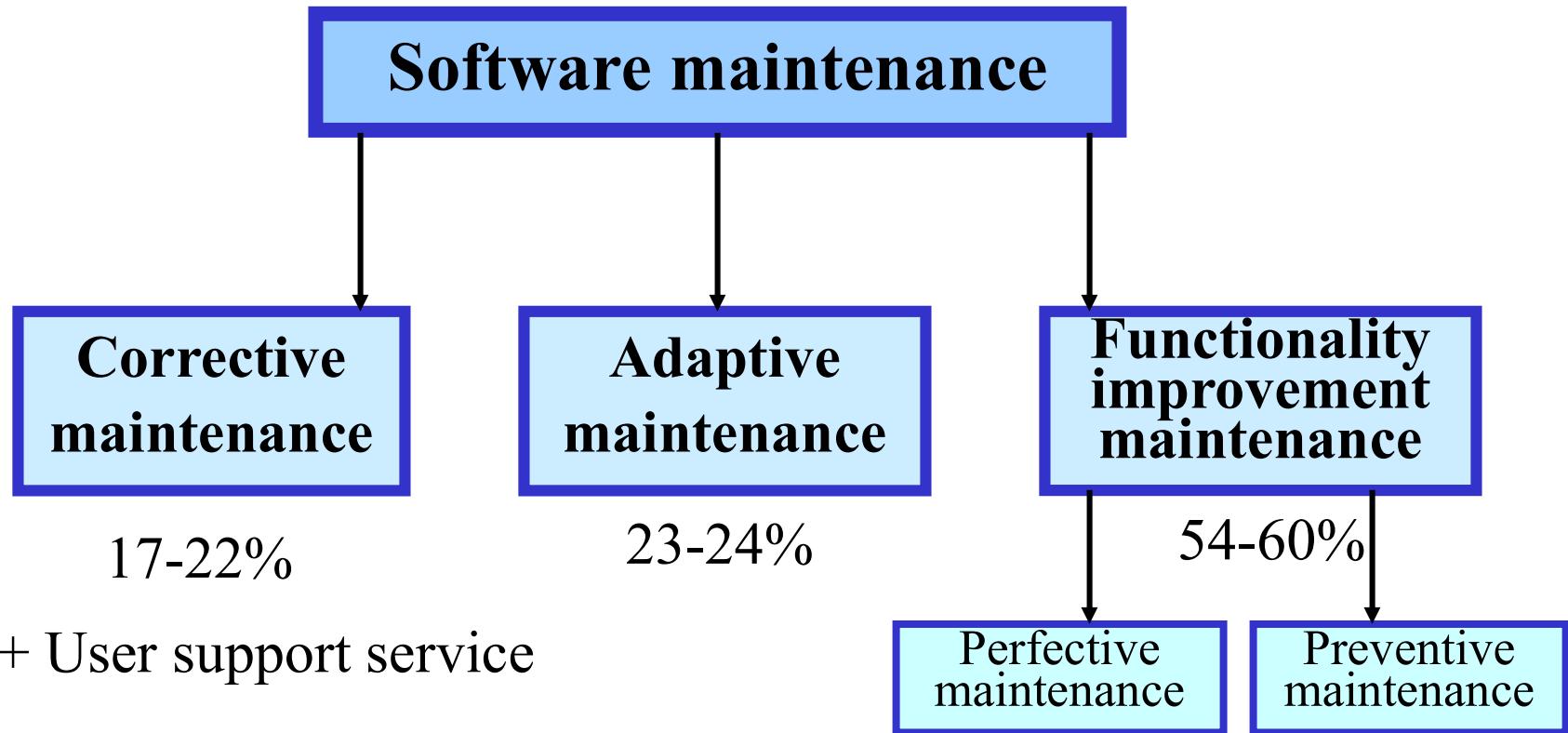
- A lack of systematic testing.
- Low quality error reports.
- Difficult to reproduce the test environment.
- Much effort is required to examine reports.

Assuring the quality of software maintenance components

- **Introduction**
- **The foundations of high quality maintenance**
 - Foundation 1: software package quality
 - Foundation 2: maintenance policy
- **Pre-maintenance software quality components**
 - Maintenance contract review
 - Maintenance plan
- **Maintenance software quality assurance tools**
 - SQA tools for corrective maintenance
 - SQA tools for functionality improving maintenance
 - SQA infrastructure components for software maintenance
 - Managerial SQA tools for software maintenance

OHT 1.160

Software maintenance components



User's difficulties

- Software failure
- Documentation failure
- Incomplete, vague or imprecise documentation
- User's insufficient knowledge

Foundation 1:

Software package quality

Quality factors of high impact on software maintenance

Quality factor	Corrective maintenance	Adaptive maintenance	Functionality improvement maintenance
Correctness – Output	High		
Correctness - Documentation	High	High	High
Correctness – Coding qualification	High	High	High
Reliability	High		
Maintenability	High	High	High
Flexibility		High	
Testability	High		
Portability		High	
Interoperability		High	

Foundation 2: Maintenance policy

- Version development policy
 - Sequential version policy
 - Tree version policy (branching and new version rules)
- Change policy (permissive/balanced)

範例

在一些年的應用之後，Inventory Perfect這一依據樹狀方針(tree policy)所開發的財產目錄管理套件，已經發展為七版本軟體套件並有這些主要分支：藥房、電子業、醫院、書店、超市、汽車自動修理廠、以及化學工廠。每一分支包含四或五個次分支，差異在軟體模組數目、實行層級或是特定的客戶導向應用。例如，書局版本有下列五個次分支(版本)：連鎖書局、單一書店、進階管理書店、LP連鎖書局的特別版本、CUCB(City University Campus Bookstores)的特別版本。軟體維護小組同時維護軟體套件的總共30個不同版本，每一版本定期依據客戶請求以及小組的技術革新來修正。

維護小組的日常經驗因此包含克服由已經超出軟體本身關連的套件結構所產生的困難：

■由特定客戶使用的目前版本之模組架構的不適當確認所引發的不正確的修正。

■由稍後證明為不適合整合到客戶套件版本的其他版本之一個模組，因不正確替換模組所引發的不正確的修正。

■說服客戶來更新他們的軟體套件以加入新開發的模組或是由新版本替換現有版本的努力。緊接著在成功說服客戶更新他們的軟體套件的努力之後，當嘗試整合新開發的模組或是由進階的模組版本替換目前版本之時，所帶來的問題及失敗。

維護小組的領導人時常提到她羨慕她的同行熟人，Inventory Star維護小組的領導人堅持他的公司所開發的軟體套件只提供所有客戶唯一一個綜合版本。

明顯地，Inventory Star所採用的連續方針(sequential policy)需要更少的維護；因為只有一個版本需要被維護，很容易來維持它的品質層級。

Maintenanace contract review

- Customer requirements clarification
- Review of alternative approaches to maintenance provision
- Review of estimates of required maintenance resources
- Review of maintenance services to be provided by subcontractors and/or the customer
- Review of maintenance costs estimates

Preparation of a Maintenance plan

- A list of the contracted maintenance services (external and internal customers)
- A description of the maintenance team's organization
- A list of maintenance facilities
- A list of identified maintenance service risks
- A list of required software maintenance procedures and controls
- The software maintenance budget

SQA tools for corrective maintenance

- Mini life cycle SQA tools – Mini testing
- Contractor-Subcontractor contract (for outsourcing maintenance services)
 - Procedures for handling maintenance calls
 - Full documentation of the service procedures
 - Availability of records
 - Certification for the contractor
 - Quality-related conditions requiring penalties

OHT 1.169

SQA tools for functionality improving maintenance

- Project life cycle tools (reviews & testing)
Tools are also implemented for large-scale adaptive maintenance tasks.

OHT 1.170

SQA infrastructure components for software maintenance

- Maintenance procedure and work instruction
- Supporting quality devices
- Training and certification of maintenance teams
- Preventive and corrective actions
- Configuration management
- Maintenance documentation and quality record control

OHT 1.171

Managerial SQA tools for software maintenance

- Performance controls for corrective maintenance services
- Quality metrics for corrective maintenance
- Cost of software maintenance quality

OHT 1.172

Cost of software maintenance quality

- Cost of prevention
- Cost of appraisal
- Cost of managerial preparation and control
- Cost of internal failure
- Cost of external failure
- Cost of managerial failure

Assuring the quality of external participants' contributions

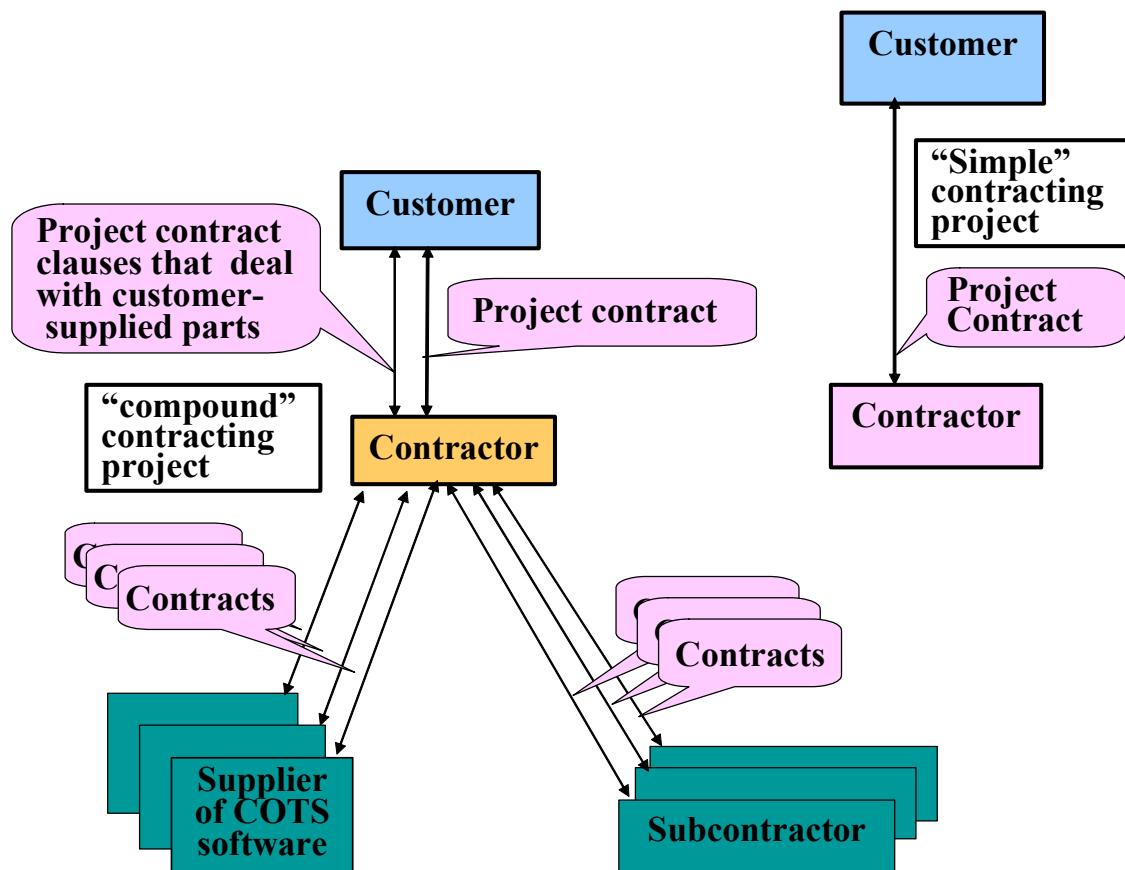
- Introduction
- Types of external participants
- Risks and benefits of introducing external participants
- Assuring the quality of external participants' contribution: objectives
- SQA tools for assuring the quality of external participants' contributions

Types of external participants

- **Subcontractors**
(currently called “outsourcing” organizations)
- **Suppliers of COTS software and reused software modules**
- **The customers themselves as participants in performing the project**

Software development projects

Typical contracting structures



OHT 1.176

Risks and benefits of introducing external participants

Main risks to project quality

- (1) Delays in completion of the project
- (2) Low quality of project parts supplied by external participants
- (3) Future maintenance difficulties
- (4) Loss of control over project parts

Main benefits for the contractor:

- (1) Budget reductions.
- (2) Remedy of professional staff shortages.
- (3) Shorter project schedule.
- (4) Acquisition of expertise in specialized areas

Main benefits for the customer (as external participant):

- (1) Protecting the customer's commercial secrets.
- (2) Provision of employment to internal software development department.
- (3) Acquisition of project know-how for self-supplied maintenance.
- (4) Project cost reductions.

OHT 1.177

Assuring the quality of external participants' contribution: objectives

- To prevent delays and ensure early alert of anticipated delays.
- To assure acceptable quality levels and receive early warning of quality requirement.
- To assure adequate documentation
- To assure comprehensive control over external participants' performance.

OHT 1.178

SQA tools applied to external participants in a software development project

- Requirements document reviews
- Evaluation of choice criteria regarding external participants
- Establishment of project coordination and joint control committee
- Participation in design reviews
- Participation in software testing
- Formulation of special procedures
- Certification of supplier's team leaders and members
- Preparation of progress reports of development activities
- Review of deliverables (documents) and acceptance tests.

Requirements document reviews

Issues to be dealt with include:

Functionality – functional requirement, interface,
performance, maintenance services

Formal & Staff – qualification, joint control committee,
delivered documents, criteria for
completion, financial arrangement

SQA – participation of design reviews & software
testing

OHT 1.180

Evaluation of choice criteria regarding external participants

- Previous experience & performance
- Quality assurance system
- Survey of opinions
 - requires systematic reporting by a dept.
- Systematic evaluation by a evaluation committee or a responsible manager

OHT 1.181

Establishment of project coordination and joint control committee

Activities:

- Confirmation of timetable and milestones
- Follow-up according to progress report
- Meeting with team leader and others
- Making decisions identified in design reviews, software tests, and follow-up
- Solving disagreements

OHT 1.182

Participation in design reviews

- extent

Participation in software testing

Include, when required,

- Planning
- Design of the tests
- Reviews of the test results
- Follow-up meeting for corrections and regression testing

Formulation of special procedures

- Supported by templates, checklists and forms.
- Objectives
 - preparation of requirement documents
 - choice of subcontractor or supplier
 - audit of the subcontractor's SQA system
 - appointment of the committee
 - progress reporting requirements

OHT 1.185

Certification of supplier's team leaders and members

- Qualification and certification
- Implementation
- Changes and replacement of team member
are to be approved

OHT 1.186

Preparation of progress reports of development activities

Contents:

- Follow-up of the risks
- Follow-up of the schedule
- Follow-up of the usage of resources
- Follow-up of the budget

OHT 1.187

Review of deliverables (documents) and acceptance tests

- Review of software development documents
- Testing of the software components of the external participant's products

Procedures and work instructions

- Software quality infrastructure components
- The need for procedures and work instructions
- Procedures and procedures manuals
- Work instructions and work instruction manuals
- Procedures and work instructions: preparation, implementation and updating

Infrastructure

- Infrastructure components are tools employed to prevent software errors and promote the quality level of the entire organization.

Typical infrastructure components

- * Procedures and work instruction.
- * Quality support devices like templates and checklists.
- * Staff SQA training and certification activities.
- * Preventive and corrective actions.
- * Software configuration management.
- * Documentation and quality records control.

SQA procedures

- Management quality review
- Annual quality planning
- Contract review
- Development and quality plans
- Quality assurance of the design
- Document control
- Subcontractors and suppliers file management
- Pre-contract review for subcontract proposal
- Acceptance tests for subcontracted software
- Acceptance tests for customer-supplied software
- Software development process

OHT 1.192

- Configuration management
- Unit tests and integration tests
- Software system tests
- Customer acceptance tests
- Progress control for software development project
- Control of design and code corrections
- Installation and delivery
- Control of quality records
- Training and certification of employees
- Maintenance plan
- Change request management
- Dealing with customers' complaint

OHT 1.193

SQA work instruction subjects (examples)

Departmental work instructions

- Audit process for new software development subcontractors (supplier candidates)
- Priorities for handling corrective maintenance tasks
- Annual evaluation of software development subcontractors
- On-the-job instructions and follow-up for new team members
 - Design documentation templates and their application
 - C++ programming instructions

OHT 1.194

- Project management work instructions
 - Coordination and cooperation with the customer
 - Weekly progress reporting by team leaders
 - Special design report template and their application in the project
 - Follow-up of beta site reporting
 - Monthly progress reporting to the customer
 - Coordination of installation and customer's team instructions

OHT 1.195

一般軟體發展包含活動 (例)

專案發展計畫

訂定軟體需求規格

軟體設計

資料庫設計

程式開發

測試

系統整合

製作操作手冊

製作維護手冊

建立安裝計畫

OHT 1.196

一般軟體發展包含活動程序 (例)

專案發展計畫程序(procedure)

軟體需求規格訂定程序

軟體設計程序

資料庫設計程序

程式開發程序

測試程序

系統整合程序

操作手冊製作程序

維護手冊製作程序

安裝計畫發展程序

OHT 1.197

軟體發展活動中預先建立之發展指引文件 (例)

專案發展計畫發展準則(criteria)

軟體需求規格發展準則

系統設計發展準則

資料庫設計發展準則

原始程式碼發展準則

測試計畫發展準則

測試報告建立準則

系統整合發展準則

操作手冊發展準則

維護手冊發展準則

安裝計畫發展準則

OHT 1.198

軟體發展活動中產出的工作產品 (例) (1/2)

- 專案發展計畫書
- 軟體需求規格書
- 系統設計文件
- 資料庫設計文件
- 原始程式碼
- 測試計畫書
- 測試報告書
- 系統整合文件
- 操作手冊
- 維護手冊
- 安裝計畫書

軟體發展活動中預先建立之產品樣板(例)

專案發展計畫書樣板(template)

軟體需求規格書樣板

系統設計文件樣板

資料庫設計文件樣板

原始程式碼撰寫樣板

測試計畫書樣板

測試報告書樣板

系統整合文件樣板

操作手冊樣板

維護手冊樣板

安裝計畫書樣板

軟體發展中可包含的品質保證活動(審查)

專案發展計畫書審查(Review)

軟體需求規格書審查

系統設計文件審查

資料庫設計文件審查

原始程式碼審查

測試計畫書審查

測試報告審查

系統整合文件審查

操作手冊審查

維護手冊審查

安裝計畫書審查

軟體發展中可包含的品質保證活動程序(審查)

專案發展計畫書審查程序(**Review Procedure**)

軟體需求規格書審查程序

系統設計文件審查程序

資料庫設計文件審查程序

原始程式碼審查程序

測試計畫書審查程序

測試報告審查程序

系統整合文件審查程序

操作手冊審查程序

維護手冊審查程序

安裝計畫書審查程序

OHT 1.202

軟體發展活動中預先建立之審查指引文件 (例)

專案發展計畫書審查準則(criteria)

軟體需求規格書審查準則

系統設計文件審查準則

資料庫設計文件審查準則

原始程式碼審查準則

測試計畫書審查準則

測試報告審查準則

系統整合文件審查準則

操作手冊審查準則

維護手冊審查準則

安裝計畫書審查準則

OHT 1.203

軟體發展活動中預先建立之檢查表單(例)

專案發展計畫書檢查表單(**checklist**)

軟體需求規格書檢查表單

系統設計文件檢查表單

資料庫設計文件檢查表單

原始程式碼檢查表單

測試計畫書檢查表單

測試報告檢查表單

系統整合文件檢查表單

操作手冊檢查表單

維護手冊檢查表單

安裝計畫書檢查表單

OHT 1.204

軟體發展活動中預先建立之檢查報告樣板(例)

專案發展計畫書檢查報告樣板
軟體需求規格書檢查報告樣板
系統設計文件檢查報告樣板
資料庫設計文件檢查報告樣板
原始程式碼檢查報告樣板
測試計畫書檢查報告樣板
測試報告檢查報告樣板
系統整合文件檢查報告樣板
操作手冊檢查報告樣板
維護手冊檢查報告樣板
安裝計畫書檢查報告樣板

OHT 1.205

Please see Appendix 14A (P. 324)

應建立之標準 (例)

Product standards

Design review form

Requirements document structure

Method header format

Java programming style

Project plan format

Change request form

Process standards

Design review conduct

Submission of documents to CM

Version release process

Project plan approval process

Change control process

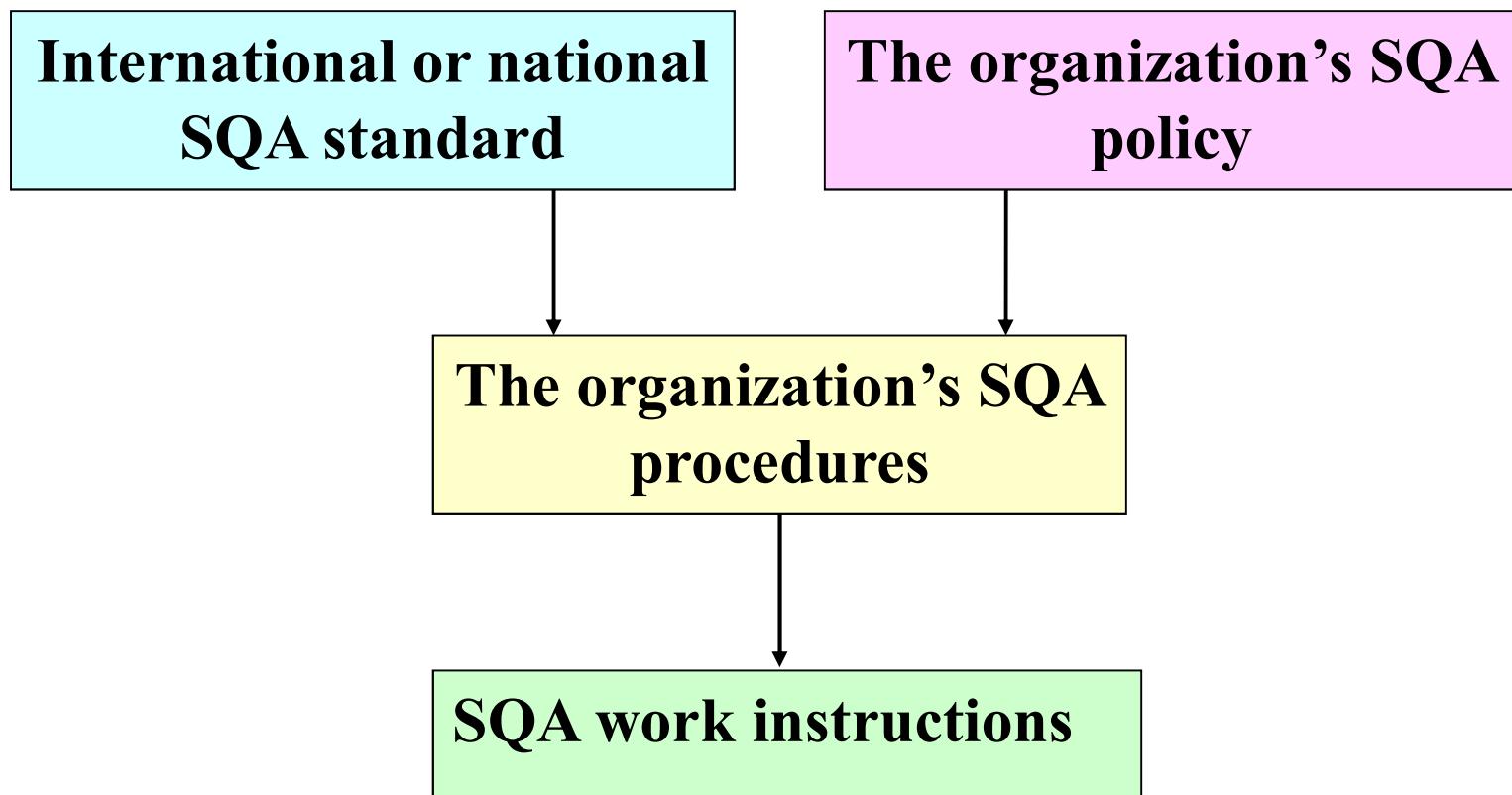
Test recording process

Procedures and work instructions - benefits

- Performance of tasks, processes or activities in the most effective and efficient way.
- Effective and efficient communication between development and maintenance teams that reduces the misunderstandings which lead to software errors.
- Simplified coordination between tasks and activities performed by various teams that means fewer errors.

OHT 1.208

Conceptual hierarchy of procedures and work instructions



SQA from theory to implementation

Issues resolved by procedures

The Five W's

- * **W**hat activities have to be performed
- * **HoW** should each activity be performed
- * **W**hen should the activity be performed
- * **W**here should the activity be performed
- * **W**ho should perform the activity

OHT 1.210

Typical fixed table of contents for procedure

1. Introduction *
2. Purpose
3. Terms and abbreviations *
4. Applicable documents
5. Method
6. Quality records and documentation
7. Reporting and follow up *
8. Responsibility for implementation *
9. List of appendices *
- Appendices *

* Sections included only if applicable

OHT 1.211

Please see Appendix 14A (P. 322)

Factors affecting the contents of the SQA procedures manual

- * The types of software development and maintenance activities carried out by the organization
- * The range of activities belonging to each activity type.
- * The range of customers.
- * The conceptions for the choice of method applied by the organization to achieved SQA objectives.

Motivations for updating existing procedures

- ◇ Technological changes in development tools, hardware, communication equipment, etc.
- ◇ Changes in the organization's areas of activity
- ◇ User proposals for improvement
- ◇ Analysis of failures as well as successes
- ◇ Proposals for improvements initiated by internal audit reports
- ◇ Learning from the experience of other organizations
- ◇ Experiences of the SQA team

Supporting quality devices

- Templates
 - The contribution of templates to software quality
 - The organizational framework for preparing, implementing and updating templates
- Checklists
 - The contribution of checklists to software quality
 - The organizational framework for preparing, implementing and updating checklists

Examples of Templates

- **Software test plan**
- **Software test description**
- **Software test report**
- **Software change request**
- **Version description document**
- **Software requirement specification**
- **System design description**
- **Computer operator manual**
- **Interface design description**
-

The contribution of templates to software quality

- For development teams:
 - * Facilitates the process of preparing documents.
 - * Documents prepared by developer are more complete.
 - * Provides for easier integration of new team members.
 - * Facilitates review of documents.
- For software maintenance teams:
 - * Enables easier location of the information.

Information sources in preparing a template

- **Informal templates already in use**
- **Template examples found in professional publications**
- **Templates used by similar organizations**

Sources for updating templates

- * User proposals and suggestions.
- * Changes in the organization's areas of activity.
- * Proposals initiated by design review and inspection teams.
- * Analysis of failures as well as successes.
- * Other organizations' experience.
- * SQA team initiatives

Examples of checklists

- Subject checklist for Proposal draft reviews
- Subject checklist for Contract draft review
- Checklist for requirement specification documents review
- Checklist for installation of a software package
- Checklist for performance of quality audits at subcontractors' sites
-

The advantages of checklists

- To development teams:
 - * Helps developers carrying out self-checks of documents or software code prior completion.
 - * Assists developers in their preparations for tasks.
- To review teams:
 - * Assures completeness of document reviews by review team members.
 - * Facilitates improves efficiency of review sessions.

Information sources in preparing a checklist

- Informal checklists already in use
- Checklist examples found in professional publications or books
- Checklist s used by similar organizations

Sources for updating checklists

- * User proposals and suggestions.
- * Changes in technology, areas of activity and clientele.
- * Proposals initiated by design review and inspection teams emanating from document reviews.
- * Analysis of failures as well as successes.
- * Other organizations' experience.
- * SQA team initiatives

Staff training and certification

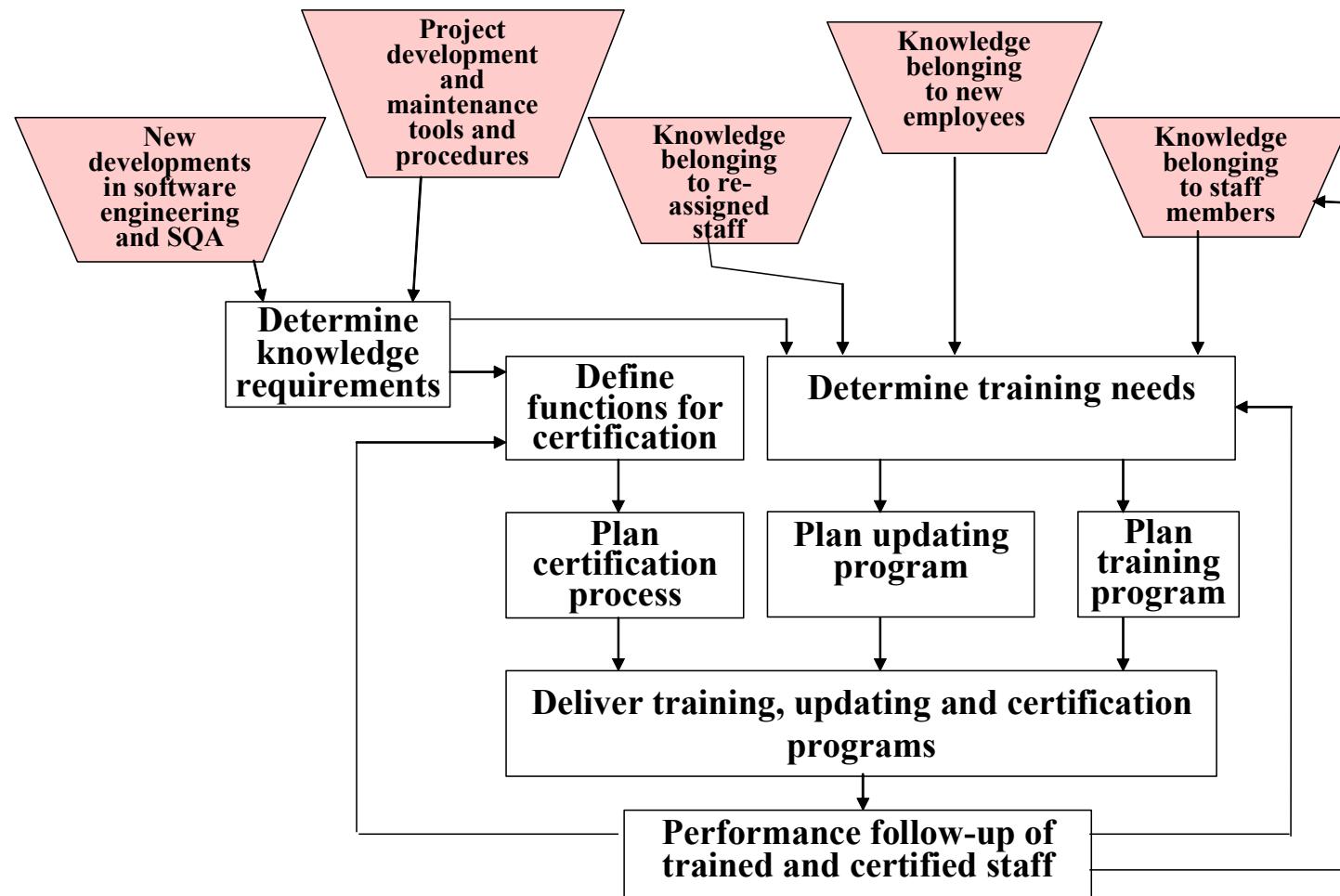
- The objectives of training and certification
- The training and certification process
- Determine professional knowledge requirements
- Determine training and updating needs
- Planning training and updating programs
- Define positions requiring certification
- Planning the certification processes
- Delivery of training and certification programs
- Follow-up subsequent to training and certification

The objectives of training and certification

- *** To develop the knowledge and skills new staff need to perform software development & maintenance tasks.
- *** To assure conformity to the organization's standards for software products (documents and code).
- *** To update the knowledge and skills of veteran staff.
- *** To transmit knowledge of SQA procedures
- *** To assure that candidates for key positions are adequately qualified

OHT 1.225

The training and certification process



OHT 1.226

Determine professional knowledge requirements

- **Profession:** system analyst, programmer, software development team leader, programming team leader, software maintenance technician, software tester, software testing team leader.
- **Knowledge requirements:** knowledge and skills of software engineering topics; knowledge of SQA topics.

Determine training and updating needs

- Training: for new employee
- Retraining: for employees assigned to new positions or receiving new assignments
- Updating: for staff members as demanded by their position
- Follow-up provides major input in redefining training needs.

Positions requiring certification

- Examples: software development team leader, programming team leader, software maintenance technician, software testing team leader, internal quality auditor.
- - Varies by firms or organization.

Typical certification requirements

- ** Professional education: academic or technical degrees
- ** Internal training courses
- ** Professional experience in the organization (may be partially or completely replaced by experience in other organizations)
- ** Assessment of achievements and ability in periodic appraisals
- ** Evaluation by the candidate's direct superior
- ** Demonstration of knowledge and skills by means of a test or a project
- ** Mentor's supervision for a specified period of time

Functions of the certification committee

- ** To perform certification process and grant certification to those who qualify
- ** To follow-up certification activities (such as mentoring) carried out by others
- ** To update certification requirements in response to developments in the organization as well as the profession
- ** To revise the list of positions requiring certification

Delivery of training and certification programs

- - Topics include software engineering, SQA & management skills, as needed by the firm.
- Courses can be short lectures, demonstrations, lengthy courses.
- Can be conducted by organization's training unit, academic institutions, vocational institutions.

Follow-up of trained and certified staff performance

- Collection of regular performance metrics
- Questionnaires completed by trained and certified staffs, superiors, customers and others.
- Analysis of outstanding achievements & failures.
- Specialized review of software products

Corrective and preventive actions

- Corrective and preventive actions — definitions
- The corrective and preventive actions process
- Information collection
- Analysis of collected information
- Development of solutions and their implementation
- Follow-up of activities
- Organizing for preventive and corrective actions

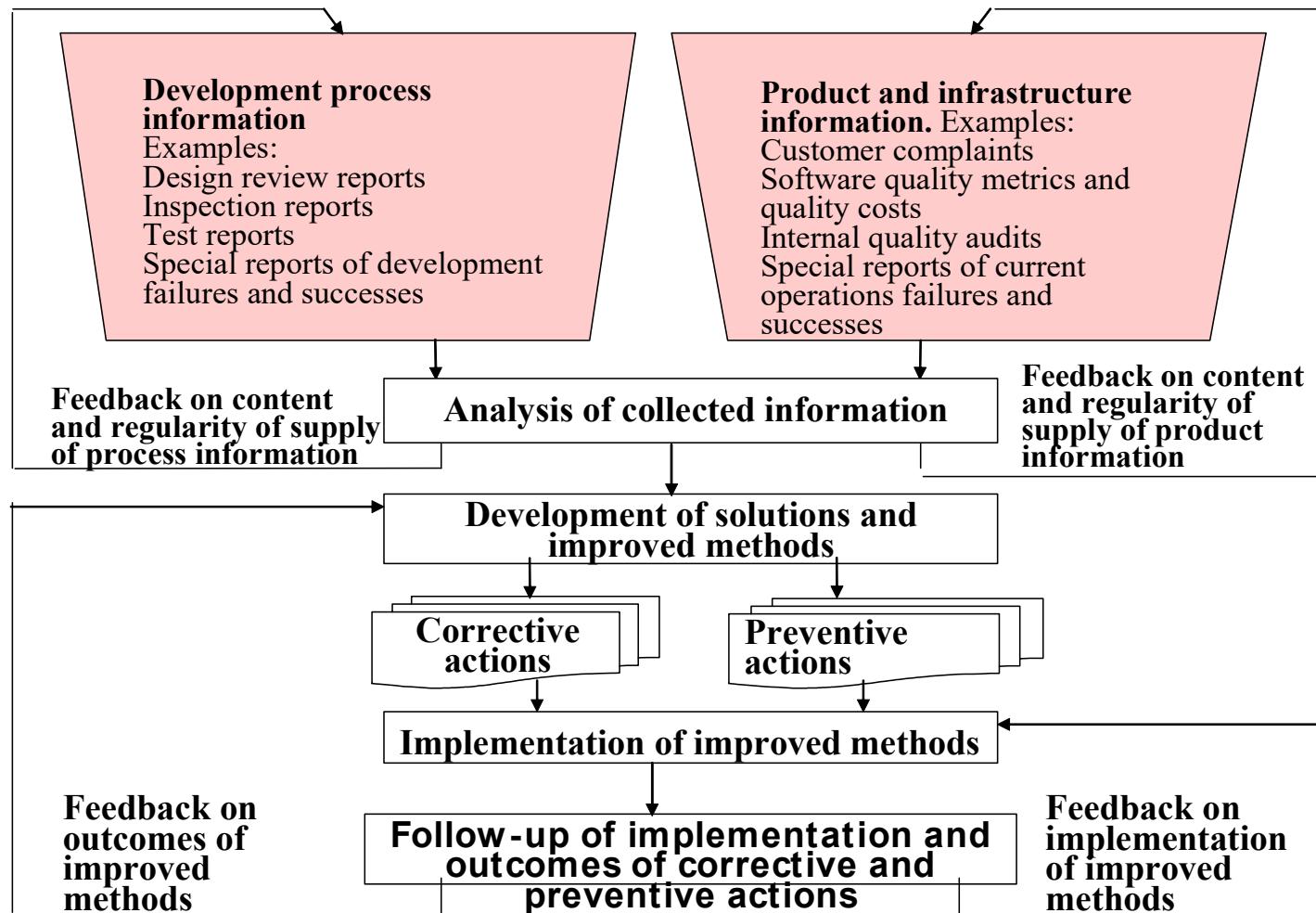
Corrective and preventive actions - definitions

- **Corrective actions**
- A regularly applied feedback process that includes collection of information on quality non-conformities, identification and analysis of sources of irregularities as well as development and assimilation of improved practices and procedures, together with control of their implementation and measurement of their outcomes.
- **Preventive actions**
- A regularly applied feedback process that includes collection of information on potential quality problems, identification and analysis of departures from quality standards, development and assimilation of improved practices and procedures, together with control of their implementation and measurement of their outcomes.

Sources of CAPA information

- Quality records
- Service reports
- Internal quality audits
- Project risk reviews
- Software risk management reports
-

The corrective and preventive action process



Sources of CAPA information

- *Internal information sources*
- Software development process
 - * Software risk management reports
 - * Design review reports
 - * Inspection reports
 - * Walkthrough reports
 - * Experts opinion reports
 - * Test reviews
 - * Special reports on development failures
- and
- successes
- * Proposals suggested by staff members
- Software maintenance
 - * Customer applications statistics
 - * Software change requests initiated by customer applications
 - * Software change requests initiated by maintenance staff
 - * Special reports on maintenance failures
- and
- successes

SQA from theory to implementation
Proposals suggested by staff members

- SQA infrastructure class of sources
 - * Internal quality audit reports
 - * External quality audit reports
 - * Performance follow-up of trained and certified staff
 - * Proposals suggested staff members
- Software quality management procedures class of sources
 - * Project progress reports
 - * Software quality metrics reports
 - * Software quality cost reports
 - * Proposals of staff members
- *External information sources*
 - * Customer complaints
 - * Customer service statistics
 - * Customer-suggested proposals

Corrective actions board (CAB) – the tasks

- * Collecting CAPA records from the various sources.
- * Screening the collected information.
- * Nominating ad hoc CAPA teams to attend to given subjects or head the teams.
- * Promoting implementation of CAPA
- * Following up information collection, data analysis, progress made by ad hoc teams, implementation as well as outcomes of improved CAPA methods.

Configuration management

- Software configuration, software configuration items and software configuration management
- Software configuration management – tasks and organization
 - The tasks of the software configuration management
 - The software configuration authority
- Software change control
 - Approval to carry out proposed changes
 - Quality assurance of software changes
- Release of software configuration versions
 - Types of software configuration releases
 - Software configuration management plans
 - Software configuration evolution models
 - Documentation of software configuration versions
- Provision of SCM information services
- Software configuration management audits
- Computerized tools for managing software configuration

The need for configuration management

- <> “What is the correct version of the software module that I have to continue its coding?”
- <> “Who can provide me with an accurate copy of the last year’s version 4.1 of the TMY software package?”
- <> “What version of the design document matches the software version we are adapting to a new customer?”
- <> “What version of the software system is installed at ABC Industries?”
- <> “What changes have been introduced in the version installed at the ABC Industries’ site?”
- <> “What changes have been introduced in the new version of the software?”
- <> “Where can I find the full list of customers that use version 6.8 of our software?”
- <> “Can we be sure that the version installed at Top Com Ltd. does not include undocumented changes?”

OHT 1.241

Software configuration management - definitions

- **Software configuration item (SCI)**
- An approved unit of software code, a document or piece of hardware that is designed for configuration management and treated as a distinct entity in the software configuration management process.
- **Software configuration item version (SCI version)**
- The approved state of an SCI at any given point of time during the development or maintenance process
- **Software configuration version**
- An approved selected set of documented SCI versions, that **constitute** a software system or document at a given point of time, where the activities to be performed are controlled by *software configuration management* procedures.

Common types of software configuration items

- **Design documents**
- **Software code**
 - * Source code
 - * Object code
 - * Prototype software
- **Data files**
 - * Test cases and test scripts
 - * Parameters, codes, etc.
- **Software development tools** (the versions applied in the development and maintenance stages)
 - * Compilers and debuggers
 - * Application generators
 - * CASE tools

OHT 1.243

Design documents

- Software development plan (SDP)
- System requirement document
- Software requirement document (SRD)
- Interface design specifications
- Preliminary design document (PDD)
- Critical design document (CDD)
- Database description
- Software test plan (STP)
- Software test procedure (STPR)
- Software test report (STR)
- Software user manual
- Software maintenance manual
- Software installation plan (SIP)
- Software maintenance request (including problem reports)
- Software change request (SCRs) and software change order
- Version description document (VDD)

OHT 1.244

Example: 2 software configuration versions of PMT software

SCI Version	Release and release date	
	PMT Version 6.0 January 6, 2002 SCI Version in the Release	PMT Version 7.0 January 22, 2003 SCI Version in the Release
SRD	Ver. 1	Ver. 1
CDD	Ver. 3	Ver. 4
STP	Ver. 3	Ver. 4
SIP	Ver. 2	Ver. 2
VDD	Ver. 6	Ver. 7
Code Module 1	Ver. 3	Ver. 5
Code Module 2	Ver. 8	Ver. 8
Code Module 3	Ver. 2	Ver. 2
Test cases file	Ver. 3	Ver. 4
CL compiler	Ver. 5	Ver. 7
Software user manual	Ver. 6	Ver. 7

SQA from theory to implementation

Software configuration management - definition

- An SQA component responsible for applying (computerized and non-computerized) technical tools and administrative procedures that enable completion of the tasks required to maintain SCIs and software configuration versions.

The tasks of the software configuration management

- *** Control software change
- *** Release of SCI and software configuration versions
- *** Provision of SCM information services
- *** Verification of compliance to SCM procedures

Factors affecting approval of proposed change

- * Expected contribution of the proposed change
- * Urgency of the change
- * Effect of the proposed change on project timetables, level of service, etc.
- * Efforts required in making the change operational
- * Required software quality assurance efforts
- * Estimated required professional resources and cost of performing the change

The need to release a new software configuration version

1. Defective SCIs
2. Special features demanded by new customers
3. Team's initiatives to introduce SCI improvements

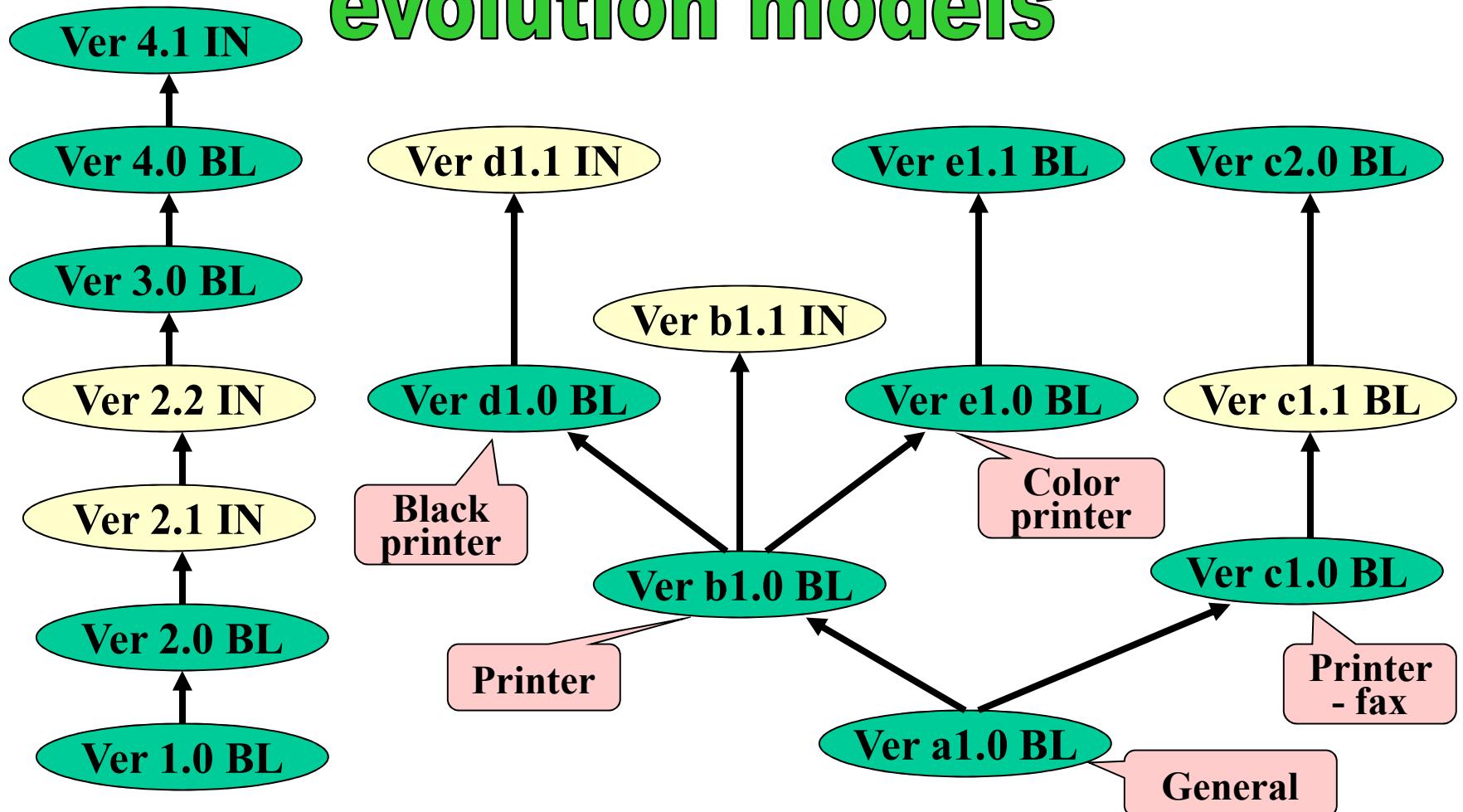
OHT 1.249

Software configuration management plan (SCMP)

- The plan includes:
- * **A list of scheduled baseline version releases.**
- * **A list of SCIs (documents, code, etc.) to be included in each version.**
- * **A table identifying the relationship of software development project plans and maintenance plans to scheduled releases of new SCIs or SCI versions.**
- * **A list of assumptions about the resources required to perform the SCMP.**
- * **Estimates of the human resources and budget needed to perform the SCMP.**

OHT 1.250

Software configuration evolution models



Linear evolution model

SQA from theory to implementation

Tree evolution model

OHT 1.251

Software configuration release documentation

VDD Template

- a. **Identification and installations**
 - *
 - *
- * Release version and revision number, including date
- * List of installations where the release was installed
- b. **Configuration of the released version**
 - *
 - *
 - *
 - *
 - *
 - *
- * List of SCIs (including SCI's version) in the released software version
- * List of hardware configuration items required for operating the specified version
- * List of interfacing software and hardware systems
- * Installation instructions for the new release

Software configuration release documentation

VDD Template - cont

- C. **Changes in the new version**
 - * Previous software configuration version
 - * List of SCIs that have been changed, new SCIs, and deleted SCIs
 - * Short description of introduced changes.
 - * Operational and other implications of changes in the release.
- D. **Further development issues**
 - * List of software system problems that have not been solved in the new version.
 - * List of delayed SCRs and proposals for development of the software system.

SCM information services

- *Information related to software change control:*
 - *
 - Change request status information
 - *
 - Change order progress information
- *Information about SCIs and software configuration versions:*
 - *
 - Accurate copies of SCI versions (code SCIs, document SCIs, etc.) and entire software configuration versions.
 - *
 - Full reports of changes between successive releases (versions and/or revisions) of code SCIs and between successive releases of other types of SCIs.
 - *
 - Copies of SCI version documentation and software configuration version documentation (VDDs).
 - *
 - Detailed version and revision history for SCIs and software configurations.
 - *
 - Progress information about planned versions and releases
 - *
 - Information correlated about versions installed at a given site and about the site itself.
 - *
 - List where a given software configuration version is installed.

Documentation control

- Controlled documents and quality records
 - Definitions and objectives
 - Documentation control procedures
- The controlled documents list
- Controlled document preparation
- Issues of controlled document approval
- Issues of controlled document storage and retrieval

Controlled document and quality record - Definitions

- **Controlled document**
- A document that is currently vital or may become vital for the development and maintenance of software systems as well as for the management of current and future relationships with the customer. Hence, its preparation, storage, retrieval and disposal are controlled by documentation procedures.
- **Quality record**
- A quality record is a special type of controlled document. It is a customer-targeted document that may be required to demonstrate full compliance with customer requirements and effective operation of the software quality assurance system throughout the development and maintenance processes.

Objectives for managing controlled documents

- * To assure the quality of the document.
- * To assure its technical completeness and compliance with document structure procedures and instructions
- * To assure the future availability of documents that may be required for software system maintenance, or responses to the customer's future complaints.
- * To support investigation of software failure causes as part of corrective and other actions.

Typical components of documentation control procedures

- * Definition of the list of the document types and updates to be controlled (some classified as quality records).
- * Document preparation requirements.
- * Document approval requirements.
- * Document storage and retrieval requirements, including controlled storage of document versions, revisions and disposal, document security.

Decision issues regarding controlled documents list

- * Deciding which document type be categorized as a controlled document and which controlled document types be classified as quality record.
- * Deciding about the adequate level of control for each controlled document type.
- * Following up of compliance with the controlled document types list.
- * Analyzing follow-up findings and initiating the required updates, changes, removals and additions to the controlled documents types list.

Project progress control

- The components of project progress control
- Progress control of internal projects and external participants
- Implementation of project progress control regimes
- Computerized tools for software progress control

The Components of project progress control

- Control of risk management activities
- Project schedule control
- Project resource control
- Project budget control

Project budget control

Main budget items

- Human resources
- Development and testing facilities
- Purchase of COTS software
- Purchase of hardware
- Payments to subcontractors.

OHT 1.262

Computerized project progress control

Control of risk management

- * Lists of software risk items by category and their planned solution dates

- * Lists of exceptions of software risk items – overrun solution dates

OHT 1.263

Computerized project progress control

Project schedule control

- * Classified lists of delayed activities.
- * Classified lists of delays of critical activities – delays that can affect the project's completion date.
- * Updated activity schedules, according to progress reports and correction measures.
- * Classified lists of delayed milestones.
- * Updated milestones schedules, according to progress reports and applied correction measures.

OHT 1.264

Computerized project progress control

Project resource control

- * **Project resources allocation plan**
 - for activities and software modules
 - for teams and development units
 - for designated time periods, etc.
- * **Project resources utilization – as specified above**
- * **Project resources utilization exceptions – as specified above**
- * **Updated resource allocation plans generated according to progress reports and reaction measures applied**

OHT 1.265

Computerized project progress control

Project budget control

- * **Project budget plans**
 - for activity and software module
 - for teams and development units
 - for designated time periods, etc.
- * **Project budget utilization reports — as specified above**
- * **Project budget utilization deviations – by period or accumulated – as specified above**
- * **Updated budget plans generated according to progress reports ports and correction measures**

Software quality metrics

- Objectives of quality measurement
- Classification of software quality metrics
- Process metrics
- Product metrics
- Implementation of software quality metrics
- Limitations of software metrics
- The function point method

IEEE definitions of software quality metrics

- (1) **A quantitative measure** of the degree to which an item possesses a given quality attribute.
- (2) **A function** whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.

Main objectives of software quality metrics

1. **Facilitate** management control, planning and managerial intervention.

Based on:

- Deviations of actual from planned performance.
- Deviations of actual timetable and budget performance from planned.

2. **Identify** situations for development or maintenance process improvement (preventive or corrective actions). Based on:

- Accumulation of metrics information regarding the performance of teams, units, etc.

Software quality metrics – Requirements

General requirements

- Relevant
- Valid
- Reliable
- Comprehensive
- Mutually exclusive

Operative requirements

- Easy and simple
- Does not require independent data collection
- Immune to biased interventions by interested parties

Classifications of software quality metrics

Classification by phases of software system

- Process metrics – metrics related to the software development process
- Product metrics – metrics related to software maintenance

Classification by subjects of measurements

- Quality
- Timetable
- Effectiveness (of error removal and maintenance services)
- Productivity

Software size (volume) measures

- **KLOC** — classic metric that measures the size of software by thousands of code lines.
- **Number of function points (NFP)** — a measure of the development resources (human resources) required to develop a program, based on the functionality specified for the software system.

OHT 1.272

Error counted measures

Number of code errors (NCE) vs. weighted number of code errors (WCE)

Error severity class	Calculation of NCE		Calculation of WCE	
	Number of Errors	Relative Weight	Weighted Errors	
a	b	c	D = b x c	
low severity	42	1	42	
medium severity	17	3	51	
high severity	11	9	99	
Total	70	---	192	
NCE	70	---	---	
WCE		---	192	

Process metrics categories

- **Software process quality metrics**
 - Error density metrics
 - Error severity metrics
- **Software process timetable metrics**
- **Software process error removal effectiveness metrics**
- **Software process productivity metrics**

OHT 1.274

Error density metrics

Code	Name	Calculation formula
CED	Code Error Density	$CED = \frac{NCE}{KLOC}$
DED	Development Error Density	$DED = \frac{NDE}{KLOC}$
WCED	Weighted Code Error Density	$WCED = \frac{WCE}{KLOC}$
WDED	Weighted Development Error Density	$WDED = \frac{WDE}{KLOC}$
WCEF	Weighted Code Errors per Function Point	$WCEF = \frac{WCE}{NFP}$
WDEF	Weighted Development Errors per Function Point	$WDEF = \frac{WDE}{NFP}$

NCE = The number of code errors detected by code inspections and testing.

NDE = total number of development (design and code) errors detected in the development process.

WCE = weighted total code errors detected by code inspections and testing.

WDE = total weighted development (design and code) errors detected in development process.

Error severity metrics

Code	Name	Calculation formula
ASCE	Average Severity of Code Errors	ASCE = $\frac{WCE}{NCE}$
DED	Average Severity of Development Errors	ASDE = $\frac{WDE}{NDE}$

NCE = The number of code errors detected by code inspections and testing.

NDE = total number of development (design and code) errors detected in the development process.

WCE = weighted total code errors detected by code inspections and testing.

WDE = total weighted development (design and code) errors detected in development process.

OHT 1.276

Software process timetable metrics

Code	Name	Calculation formula
TTO	Time Table Observance	$TTO = \frac{MSOT}{MS}$
ADMC	Average Delay of Milestone Completion	$ADMC = \frac{TCDAM}{MS}$

MSOT = Milestones completed on time.

MS = Total number of milestones.

TCDAM = Total Completion Delays (days, weeks, etc.) for all milestones.

Error removal effectiveness metrics

Code	Name	Calculation formula
DERE	Development Errors Removal Effectiveness	$\text{DERE} = \frac{\text{NDE}}{\text{NDE} + \text{NYF}}$
DWERE	Development Weighted Errors Removal Effectiveness	$\text{DWERE} = \frac{\text{WDE}}{\text{WDE} + \text{WYF}}$

NDE = total number of development (design and code) errors detected in the development process.

WCE = weighted total code errors detected by code inspections and testing.

WDE = total weighted development (design and code) errors detected in development process.

NYF = number software failures detected during a year of maintenance service.

WYF = weighted number of software failures detected during a year of maintenance service.

OHT 1.278

Process productivity metrics

Code	Name	Calculation formula
DevP	Development Productivity	$DevP = \frac{DevH}{KLOC}$
FDevP	Function point Development Productivity	$FDevP = \frac{DevH}{NFP}$
CRe	Code Reuse	$CRe = \frac{ReKLOC}{KLOC}$
DocRe	Documentation Reuse	$DocRe = \frac{ReDoc}{NDoc}$

DevH = Total working hours invested in the development of the software system.

ReKLOC = Number of thousands of reused lines of code.

ReDoc = Number of reused pages of documentation.

NDoc = Number of pages of documentation.

Product metrics categories

- * **HD quality metrics:**
 - * HD calls density metrics - measured by the number of calls.
 - * HD calls severity metrics - the severity of the HD issues raised.
 - * HD success metrics – the level of success in responding to HD calls.
- * **HD productivity metrics.**
- * **HD effectiveness metrics.**
- * **Corrective maintenance quality metrics.**
 - * Software system failures density metrics
 - * Software system failures severity metrics
 - * Failures of maintenance services metrics
 - * Software system availability metrics
- * **Corrective maintenance productivity and effectiveness metrics.**

OHT 1.280

HD calls density metrics

Code	Name	Calculation Formula
HDD	HD calls density	$\text{HDD} = \frac{\text{NHYC}}{\text{KLMC}}$
WHDD	Weighted HD calls density	$\text{WHYC} = \frac{\text{WHYC}}{\text{KLMC}}$
WHDF	Weighted HD calls per function point	$\text{WHDF} = \frac{\text{WHYC}}{\text{NMFP}}$

NHYC = the number of HD calls during a year of service.

KLMC = Thousands of lines of maintained software code.

WHYC = weighted HD calls received during one year of service.

NMFP = number of function points to be maintained.

OHT 1.281

Severity of HD calls metrics

Code	Name	Calculation Formula
ASHC	Average severity of HD calls	$\text{ASHC} = \frac{\text{WHYC}}{\text{NHYC}}$

NHYC = the number of HD calls during a year of service.

WHYC = weighted HD calls received during one year of service.

OHT 1.282

HD success metrics

Code	Name	Calculation Formula
HDS	HD service success	$HDS = \frac{NHYOT}{NHYC}$

NHYNOT = Number of yearly HD calls completed on time during one year of service.
NHYC = the number of HD calls during a year of service.

OHT 1.283

HD productivity and effectiveness metrics

Code	Name	Calculation Formula
HDP	HD Productivity	$HDP = \frac{HDYH}{KLNC}$
FHDP	Function Point HD Productivity	$FHDP = \frac{HDYH}{NMFP}$
HDE	HD effectiveness	$HDE = \frac{HDYH}{NHYC}$

HDYH = Total yearly working hours invested in HD servicing of the software system.

KLMC = Thousands of lines of maintained software code.

NMFP = number of function points to be maintained.

NHYC = the number of HD calls during a year of service.

OHT 1.284

Software system failures density metrics

Code	Name	Calculation Formula
SSFD	Software System Failure Density	$\text{SSFD} = \frac{\text{NYF}}{\text{KLMC}}$
WSSFD	Weighted Software System Failure Density	$\text{WSSFD} = \frac{\text{WYF}}{\text{KLMC}}$
WSSFF	Weighted Software System Failures per Function point	$\text{WSSFF} = \frac{\text{WYF}}{\text{NMFP}}$

NYF = number of software failures detected during a year of maintenance service.

WYF = weighted number of yearly software failures detected during one year of maintenance service.

NMFP = number of function points designated for the maintained software.

KLMC = Thousands of lines of maintained software code.

OHT 1.285

Software system failure severity metrics

Code	Name	Calculation Formula
ASSSF	Average Severity of Software System Failures	ASSSF = $\frac{WYF}{NYF}$

NYF = number of software failures detected during a year of maintenance service.

WYF = weighted number of yearly software failures detected during one year.

OHT 1.286

Failures of maintenance services metrics

Code	Name	Calculation Formula
MRepF	Maintenance Repeated repair Failure metric -	$MRepF = \frac{\text{RepYF}}{\text{NYF}}$

NYF = number of software failures detected during a year of maintenance service.

RepYF = Number of repeated software failure calls (service failures).

OHT 1.287

Software system availability metrics

Code	Name	Calculation Formula
FA	Full Availability	$FA = \frac{NYSerH - NYFH}{NYSerH}$
VitA	Vital Availability	$VitA = \frac{NYSerH - NYVitFH}{NYSerH}$
TUA	Total Unavailability	$TUA = \frac{NYTFH}{NYSerH}$

NYSerH = Number of hours software system is in service during one year.

NYFH = Number of hours where at least one function is unavailable (failed) during one year, including total failure of the software system.

NYVitFH = Number of hours when at least one vital function is unavailable (failed) during one year, including total failure of the software system.

NYTFH = Number of hours of total failure (all system functions failed) during one year.

NYFH \geq NYVitFH \geq NYTFH.

1 – TUA \geq VitA \geq FA

SQA from theory to implementation

OHT 1.288

Software corrective maintenance productivity and effectiveness metrics

Code	Name	Calculation Formula
CMaP	Corrective Maintenance Productivity	$CMaP = \frac{CMaiYH}{KLMC}$
FCMP	Function point Corrective Maintenance Productivity	$FCMP = \frac{CMaiYH}{NMFP}$
CMaE	Corrective Maintenance Effectiveness	$CMaE = \frac{CMaiYH}{NYF}$

CMaiYH = Total yearly working hours invested in the corrective maintenance of the software system.

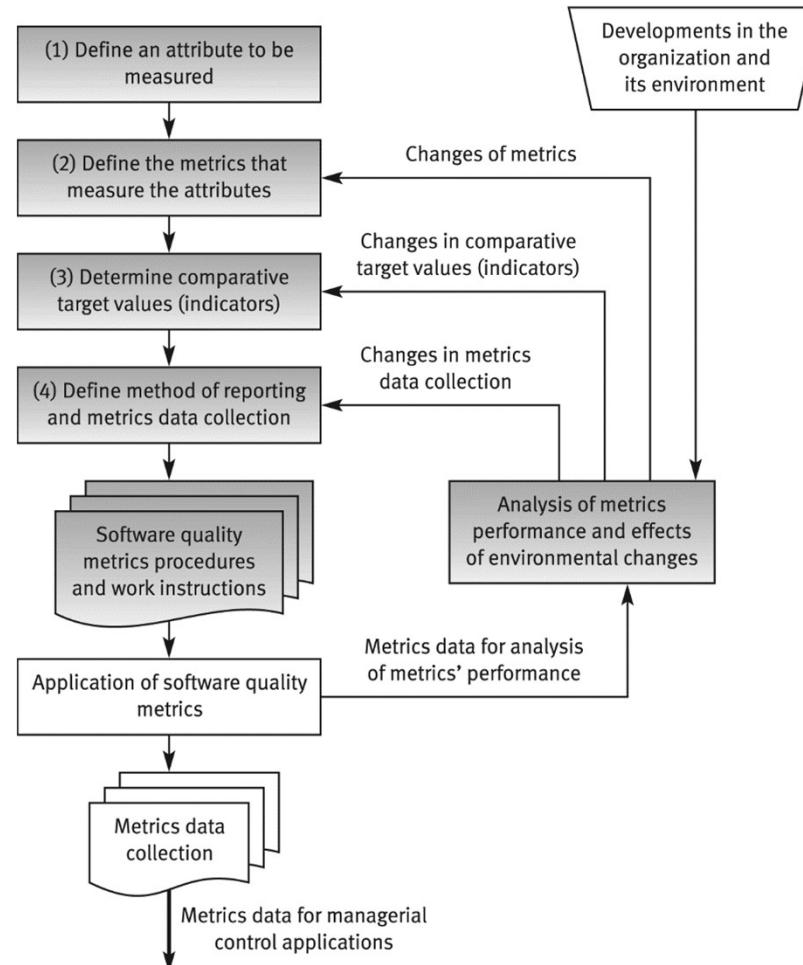
NYF = number of software failures detected during a year of maintenance service.

NMFP = number of function points designated for the maintained software.

KLMC = Thousands of lines of maintained software code.

OHT 1.289

The process of defining software quality metrics



SQA from theory to implementation

General limitations of quality metrics

- * **Budget constraints** in allocating the necessary resources.
- * **Human factors**, especially opposition of employees to evaluation of their activities.
- * **Validity** Uncertainty regarding the data's, partial and biased reporting.

Examples of software metrics that exhibit severe weaknesses

- * Parameters used in development process metrics:
KLOC, NDE, NCE.
- * Parameters used in product (maintenance) metrics:
KLMC, NHYC, NYF.

Factors affecting parameters used for development process metrics

- a. Programming style (**KLOC**).
- b. Volume of documentation comments (**KLOC**).
- c. Software complexity (**KLOC, NCE**).
- d. Percentage of reused code (**NDE, NCE**).
- e. Professionalism and thoroughness of design review and software testing teams: affects the number of defects detected (**NCE**).
- f. Reporting style of the review and testing results: concise reports vs. comprehensive reports (**NDE, NCE**).

Factors affecting parameters used for product (maintenance) metrics

- a. Quality of installed software and its documentation (**NYF, NHYC**).
- b. Programming style and volume of documentation comments included in the code be maintained (**KLMC**).
- c. Software complexity (**NYF**).
- d. Percentage of reused code (**NYF**).
- e. Number of installations, size of the user population and level of applications in use: (**NHYC, NYF**).

The function point method

The function point estimation process:

- **Stage 1:** Compute crude function points (**CFP**).
- **Stage 2:** Compute the relative complexity adjustment factor (**RCAF**) for the project. RCAF varies between 0 and 70.
- **Stage 3:** Compute the number of function points (**FP**):

$$FP = CFP \times (0.65 + 0.01 \times RCAF)$$

OHT 1.295

Crude function points (CFP) – calculation form

Software system components	Complexity level									Total CFP	
	Simple			average			complex				
	Count	Weight Factor	Points	Count	Weight Factor	Points	Count	Weight Factor	Points		
	A	B	C= Ax B	D	E	F= DxE	G	H	I= GxH		
User inputs		3			4			6			
User outputs		4			5			7			
User online queries		3			4			6			
Logical files		7			10			15			
External interfaces		5			7			10			
Total CFP											

OHT 1.296

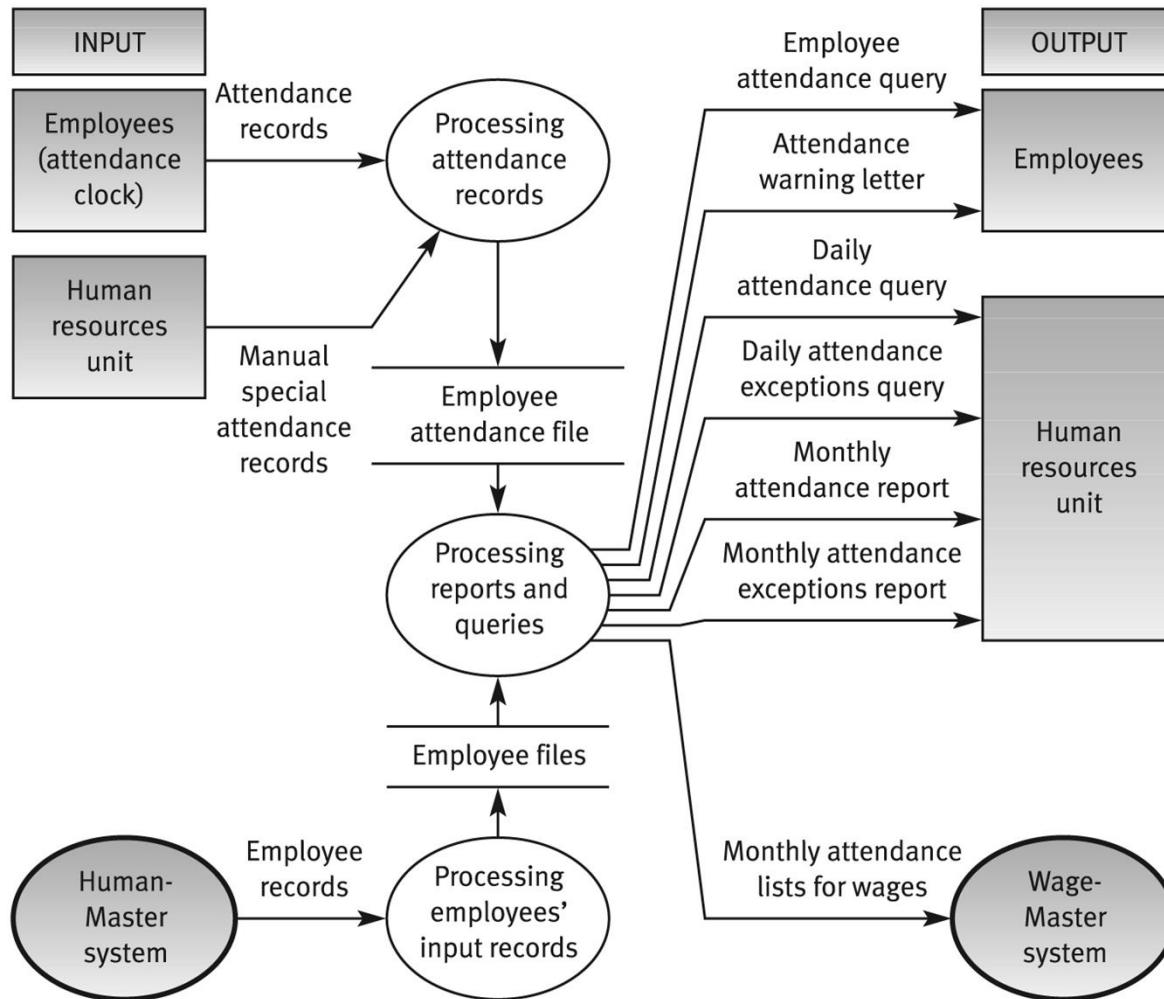
Relative complexity adjustment factor (RCAF) – form

No	Subject	Grade
1	Requirement for reliable backup and recovery	0 1 2 3 4 5
2	Requirement for data communication	0 1 2 3 4 5
3	Extent of distributed processing	0 1 2 3 4 5
4	Performance requirements	0 1 2 3 4 5
5	Expected operational environment	0 1 2 3 4 5
6	Extent of online data entries	0 1 2 3 4 5
7	Extent of multi-screen or multi-operation online data input	0 1 2 3 4 5
8	Extent of online updating of master files	0 1 2 3 4 5
9	Extent of complex inputs, outputs, online queries and files	0 1 2 3 4 5
10	Extent of complex data processing	0 1 2 3 4 5
11	Extent that currently developed code can be designed for reuse	0 1 2 3 4 5
12	Extent of conversion and installation included in the design	0 1 2 3 4 5
13	Extent of multiple installations in an organization and variety of customer organizations	0 1 2 3 4 5
14	Extent of change and focus on ease of use	0 1 2 3 4 5
	Total = RCAF	

SQA from theory to implementation

OHT 1.297

The ATTEND MASTER - Data Flow Diagram



SQA from theory to implementation

OHT 1.298

The ATTEND MASTER

CFP calculation form

Software system components	Complexity level									Total CFP	
	Simple			average			complex				
	Count	Weight Factor	Points	Count	Weight Factor	Points	Count	Weight Factor	Points		
	A	B	C= AxB	D	E	F= DxE	G	H	I= GxH		
User inputs	1	3	3	---	4	---	1	6	6	9	
User outputs	---	4	---	2	5	10	1	7	7	17	
User online queries	1	3	3	1	4	4	1	6	6	13	
Logical files	1	7	7	---	10	---	1	15	15	22	
External interfaces	---	5	---	---	7	---	2	10	20	20	
Total CFP										81	

OHT 1.299

The ATTEND MASTER

RCAF calculation form

No	Subject	Grade
1	Requirement for reliable backup and recovery	0 1 2 3 4 5
2	Requirement for data communication	0 1 2 3 4 5
3	Extent of distributed processing	0 1 2 3 4 5
4	Performance requirements	0 1 2 3 4 5
5	Expected operational environment	0 1 2 3 4 5
6	Extent of online data entries	0 1 2 3 4 5
7	Extent of multi-screen or multi-operation online data input	0 1 2 3 4 5
8	Extent of online updating of master files	0 1 2 3 4 5
9	Extent of complex inputs, outputs, online queries and files	0 1 2 3 4 5
10	Extent of complex data processing	0 1 2 3 4 5
11	Extent that currently developed code can be designed for reuse	0 1 2 3 4 5
12	Extent of conversion and installation included in the design	0 1 2 3 4 5
13	Extent of multiple installations in an organization and variety of customer organizations	0 1 2 3 4 5
14	Extent of change and focus on ease of use	0 1 2 3 4 5
	Total = RCAF	41

SQA from theory to implementation

OHT 1.300

The ATTEND MASTER – function points calculation

$$FP = CFP \times (0.65 + 0.01 \times RCAF)$$

$$FP = 81 \times (0.65 + 0.01 \times 41) = 85.86$$

The function point method – advantages and disadvantages

Main advantages

- Estimates can be prepared at the pre-project stage.
- Based on requirement specification documents (not specific dependent on development tools or programming languages), the method's reliability is relatively high.

Main disadvantages

- FP results depend on the counting instruction manual.
- Estimates based on detailed requirements specifications, which are not always available.
- The entire process requires an experienced function point team and substantial resources.
- The evaluations required result in subjective results.
- Successful applications are related to data processing. The method cannot yet be universally applied.

Costs of software quality

1. Objectives of cost of software quality metrics
2. The classic model of cost of software quality
 - Prevention costs
 - Appraisal costs
 - Internal failure costs
 - External failure costs
3. Galin's extended model for cost of software quality
 - Managerial preparation and control costs
 - Managerial failure costs
4. Application of a cost of software quality system
 - Definition of a cost of software quality model
 - Definition of the cost data collection method
 - Implementation of a cost of software quality system
 - Problems in the application of cost of software quality metrics
5. Problems in the application of cost of software quality metrics

OHT 1.303

Cost of software quality metrics — Objectives

- In general – it enables management to achieve economic control over SQA activities and outcomes.
The specific objectives are:
 - * Control organization-initiated costs to prevent and detect software errors.
 - * Evaluation of the economic damages of software failures as a basis for revising the SQA budget.
 - * Evaluation of plans to increase or decrease of SQA activities or to invest in SQA infrastructure on the basis of past economic performance.

Performance comparisons for Managerial control over SQA costs

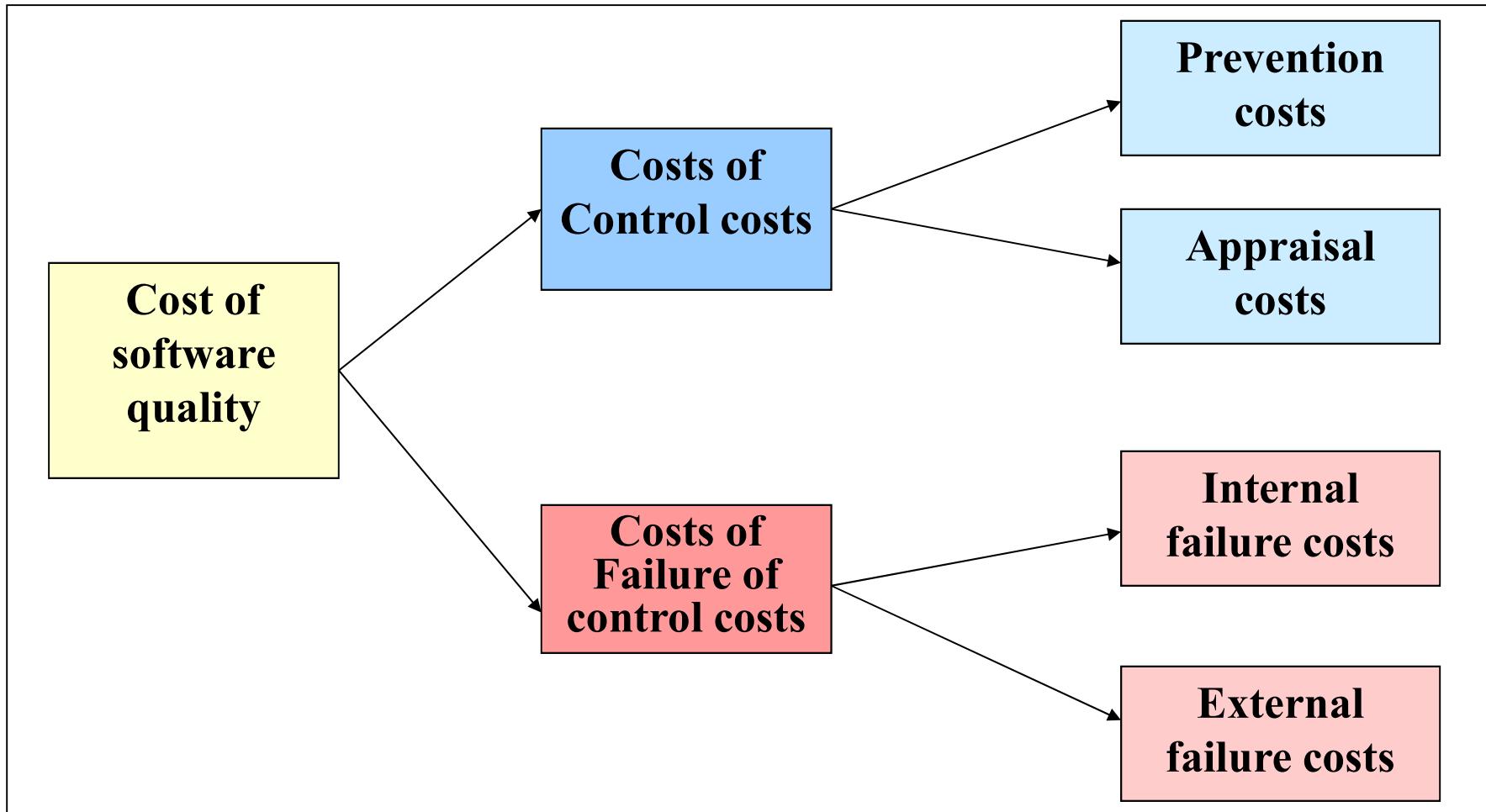
- * Control Budgeted expenditures (for SQA prevention and appraisal activities).
- * Previous year's failure costs
- * Previous project's quality costs (control costs and failure costs).
- * Other department's quality costs (control costs and failure costs).

Cost metrics for evaluating SQA systems - examples

- * Percentage of cost of software quality out of total software development costs.
- * Percentage of software failure costs out of total software development costs.
- * Percentage of cost of software quality out of total software maintenance costs.
- * Percentage of cost of software quality out of total sales of software products and software maintenance.

OHT 1.306

The classic model of cost of software quality



Prevention costs

- **a. Investments in development of SQA infrastructure components**
 - * Procedures and work instructions
 - * Support devices: templates, checklists etc
 - * Software configuration management system
 - * Software quality metrics
 - **b. Regular implementation of SQA preventive activities:**
 - * Instruction of new employees in SQA subjects
 - * Certification of employees
 - * Consultations on SQA issues to team leaders and others
 - **c. Control of the SQA system through performance of:**
 - * Internal quality reviews
 - * External quality audits
 - * Management quality reviews

SQA from theory to implementation

Appraisal costs

- **(a) Costs of reviews:**
 - * Formal design reviews (DRs)
 - * Peer reviews (inspections and walkthroughs)
 - * Expert reviews
- **(b) Costs of software testing:**
 - * Unit, integration and software system tests
 - * Acceptance tests (carried out by customers)
- **(c) Costs of assuring quality of external participants**

Internal failure costs

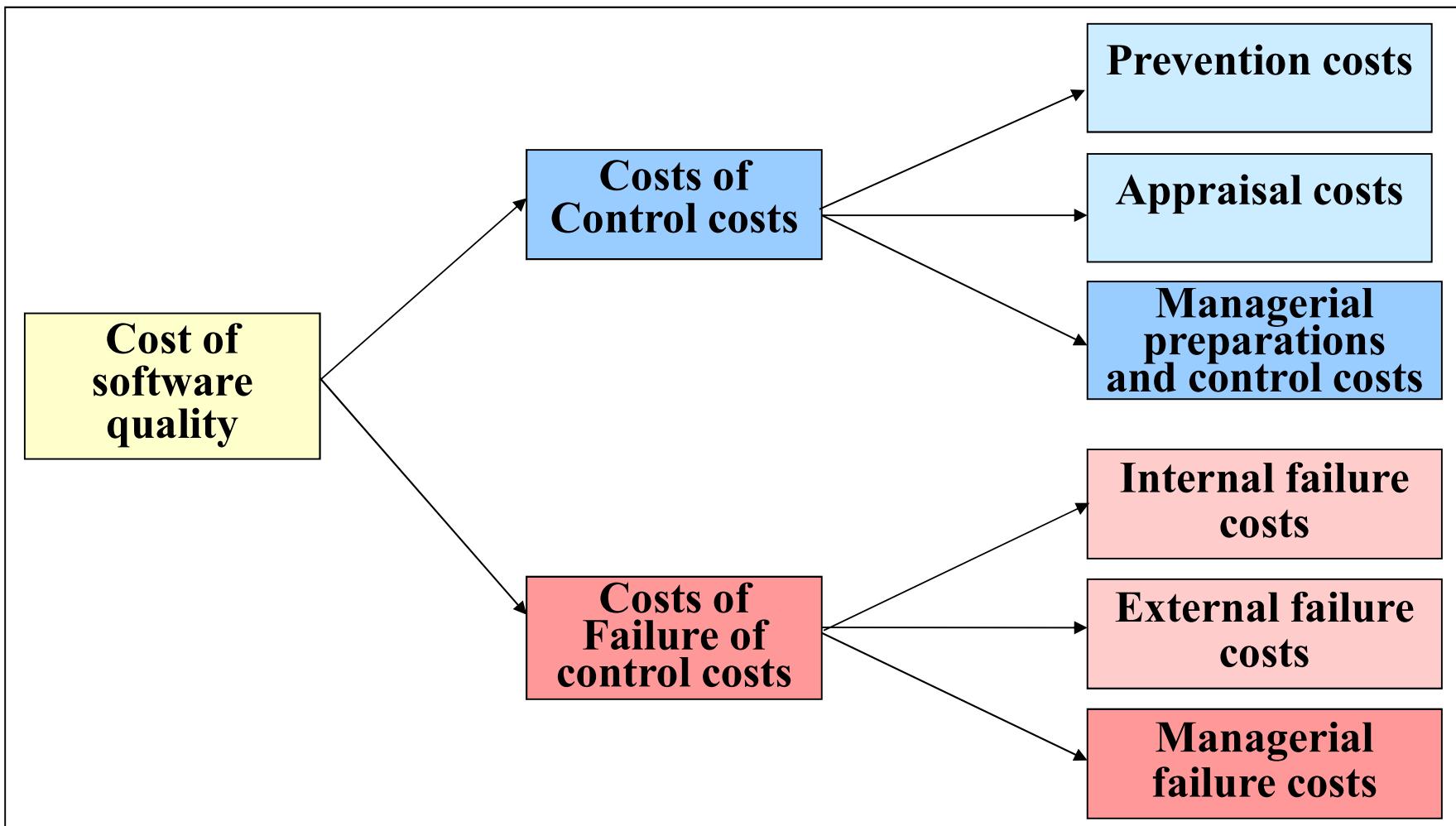
- * Costs of redesign or design corrections subsequent to design review and test findings
- * Costs of re-programming or correcting programs in response to test findings
- * Costs of repeated design review and re- testing (regression tests)

External failure costs

- Typical external failure costs cover:
 - * Resolution of customer complaints during the warranty period.
 - * Correction of software bugs detected during regular operation.
 - * Correction of software failures after the warranty period is over even if the correction is not covered by the warranty.
 - * Damages paid to customers in case of a severe software failure.
 - * Reimbursement of customer's purchase costs.
 - * Insurance against customer's claims.
- Typical examples of hidden external failure costs:
 - * Reduction of sales to customers that suffered from software failures.
 - * Severe reduction of sales motivated by the firm's damaged reputation.
 - * ~~Increased investment in sales promotion to counter the effects of past software failures.~~
- * Reduced prospects to win a tender or, alternatively, the need to

OHT 1.311

Galin's extended mode for cost of software quality



OHT 1.312

Managerial preparation and control costs

- * Costs of carrying out contract reviews
- * Costs of preparing project plans, including quality plans
- * Costs of periodic updating of project and quality plans
- * Costs of performing regular progress control
- * Costs of performing regular progress control of external participants' contributions to projects

Managerial failure costs

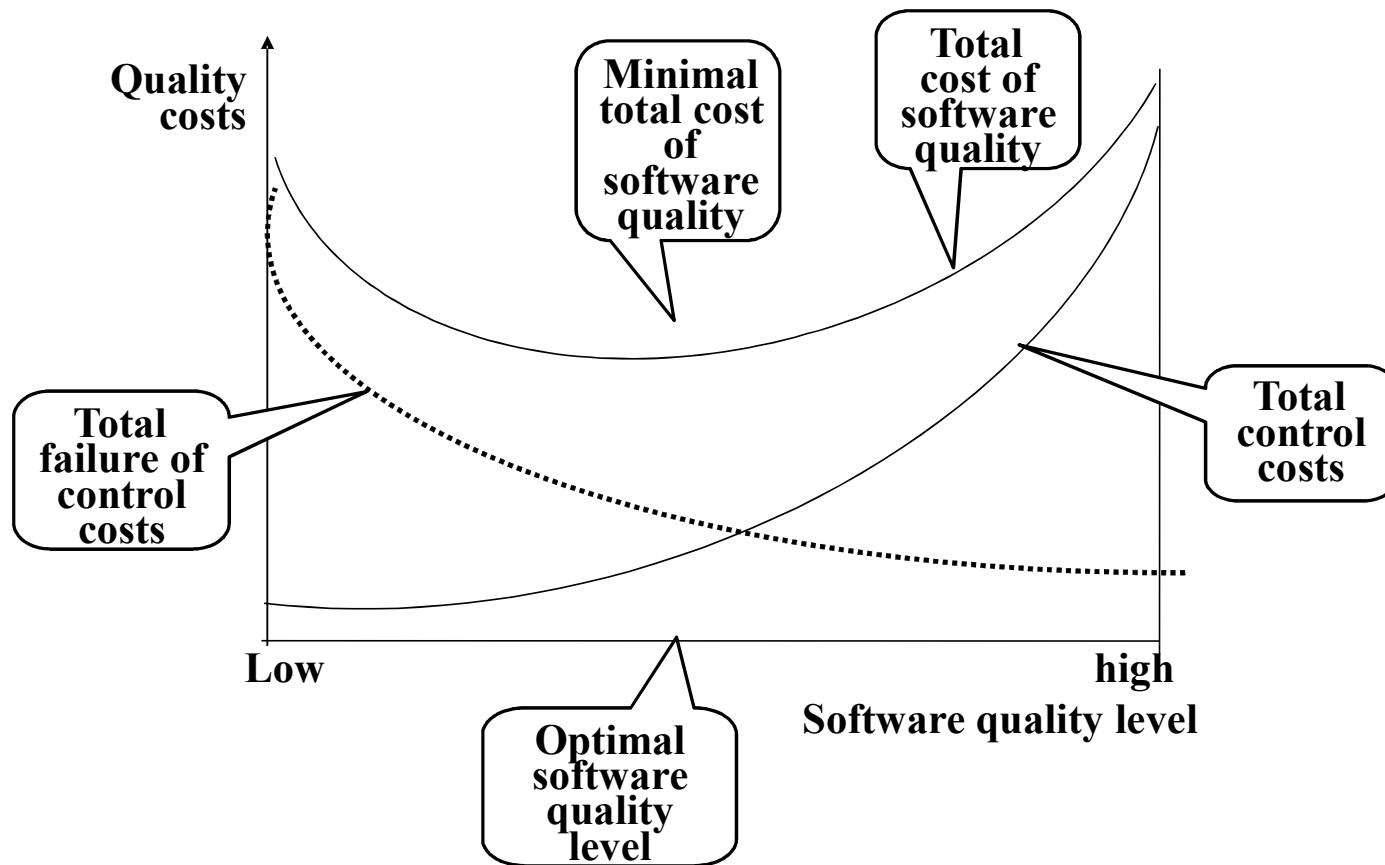
- * Unplanned costs for professional and other resources, resulting from underestimation of the resources in the proposals stage.
- * Damages paid to customers as compensation for late project completion, a result of the unrealistic schedule in the Company's proposal.
- * Damages paid to customers as compensation for late completion of the project, a result of management's failure to recruit team members.
- * *Domino effect*: Damages to other projects planned to be performed by the same teams involved in the delayed projects. The domino effect may induce considerable hidden external failure costs.

Application of a cost of software quality system

- * Definition of a cost of software quality model and specification of cost items.
- * Definition of the method of data collection for each cost item.
- * Application of a cost of software quality system, including thorough follow up.
- * Actions taken in response to the findings.

OHT 1.315

Cost of software quality balance by quality level



Problems in the application of cost of software quality metrics

- ***General problems***
 - * Inaccurate and/or incomplete identification and classification of quality costs.
 - * Negligent reporting by team members
 - * Biased reporting of software costs, especially of “censored” internal and external costs.
 - * Biased recording of external failure costs - “camouflaged” compensation of customers for failures.
- ***Problems arising when collecting data on managerial costs:***
 - * Contract review and progress control activities are performed in a “part-time mode”. The reporting of time invested is usually inaccurate and often neglected.
 - * Many participants in these activities are senior staff members who are not required to report use of their time resources.
 - * Difficulties in determination of responsibility for schedule failures.
 - * *Payment* of overt and formal *compensation* usually occurs quite some time after the project is completed, and much too late for efficient application of the lessons learned.

Quality management standards

- The benefits of use of standards
- The organizations involved in standards development
- The classification of standards
- The scope of quality management standards
- ISO 9001 and ISO 9000-3
- Certification according to ISO 9000-3
- Capability Maturity Models
- The SPICE project and the ISO/IEC 15504 software process assessment standard

The benefits of use of standards

- * The ability to apply methodologies and procedures of the highest professional level.
- * Better mutual understanding and coordination among development teams but especially between development and maintenance teams.
- * Greater cooperation between the software developer and external participants in the project.
- * Better understanding and cooperation between suppliers and customers, based on the adoption of standards as part of the contract.

OHT 1.319

Classes of SQA standards - comparison

Characteristics	Quality Management Standards	Project Process Standards
The target unit	Management of software development and/or maintenance and the specific SQA units	A software development and/or maintenance project team
The main focus	Organization of SQA systems, infrastructure and requirements	Methodologies for carrying out software development and maintenance projects
Standard's objective	“What” to achieve	“How” to perform
Standard's goal	Assuring supplier’s software quality and assessing its software process capability	Assuring the quality of a specific software project’s products

SQA from theory to implementation

Organizations involved in SQA standards development

Most prominent developers of SQA standards:

- ◊ **IEEE (Institute of Electric and Electronic Engineers Computer Society)**
- ◊ **ISO (International Standards Organization)**
- ◊ **DOD (US Department of Defense)**
- ◊ **ANSI (American National Standards Institute)**
- ◊ **IEC (International Electrotechnical Commission)**
- ◊ **EIA (Electronic Industries Association)**

OHT 1.321

The scope of quality management standards

Certification standards

- * **Enable** a software development organization to demonstrate consistent ability to assure acceptable quality of its software products or maintenance services. Certification is granted by an external body.
- * **Serve** as an agreed-upon basis for customer and supplier evaluation of the supplier's quality management system. Accomplished by performance of a quality audit by the customer.
- * **Support** the organization's efforts to improve its quality management system through compliance with the standard's requirements.

The scope of quality management standards

Assessment standards

- * **Serve** organizations as a tool for self-assessment of their ability to carry out software development projects.
- * **Serve** for improvement of development and maintenance processes by application of the standard directions
- * **Help** purchasing organizations determine the capabilities of potential suppliers.
- * **Guide** training of assessor by delineating qualifications and training program curricula.

ISO 9000-3 principles

- Customer focus
- Leadership
- Involvement of people
- Process approach
- System approach to management
- Continual improvement
- Factual approach to decision making
- Mutually supportive supplier relationships

OHT 1.324

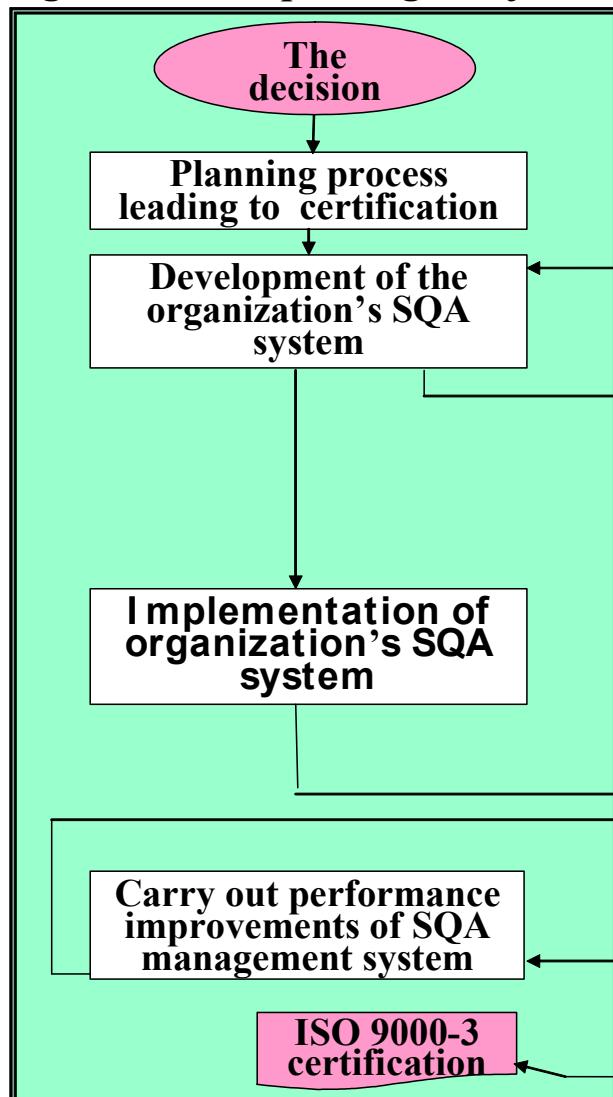
ISO 9000-3 - Requirements classification

Requirement Subjects	Requirement Subjects
4. Quality management system	4.1 General requirements 4.2 Documentation requirements
5. Management responsibilities	5.1 Management commitments 5.2 Customer focus 5.3 Quality policy 5.4 Planning 5.5 Responsibility, authority and communication 5.6 Management review
6. Resource management	6.1 Provision of resources 6.2 Human resources 6.3 Infrastructure 6.4 Work environment
7. Product realization	7.1 Planning of product realization 7.2 Customer-related processes 7.3 Design and development 7.4 Purchasing 7.5 Production and service provision 7.6 Control of monitoring and measuring devices
8. Measurement, analysis and improvement	8.1 General 8.2 Monitoring and measurement 8.3 Control of nonconforming product 8.4 Analysis of data 8.5 Improvement

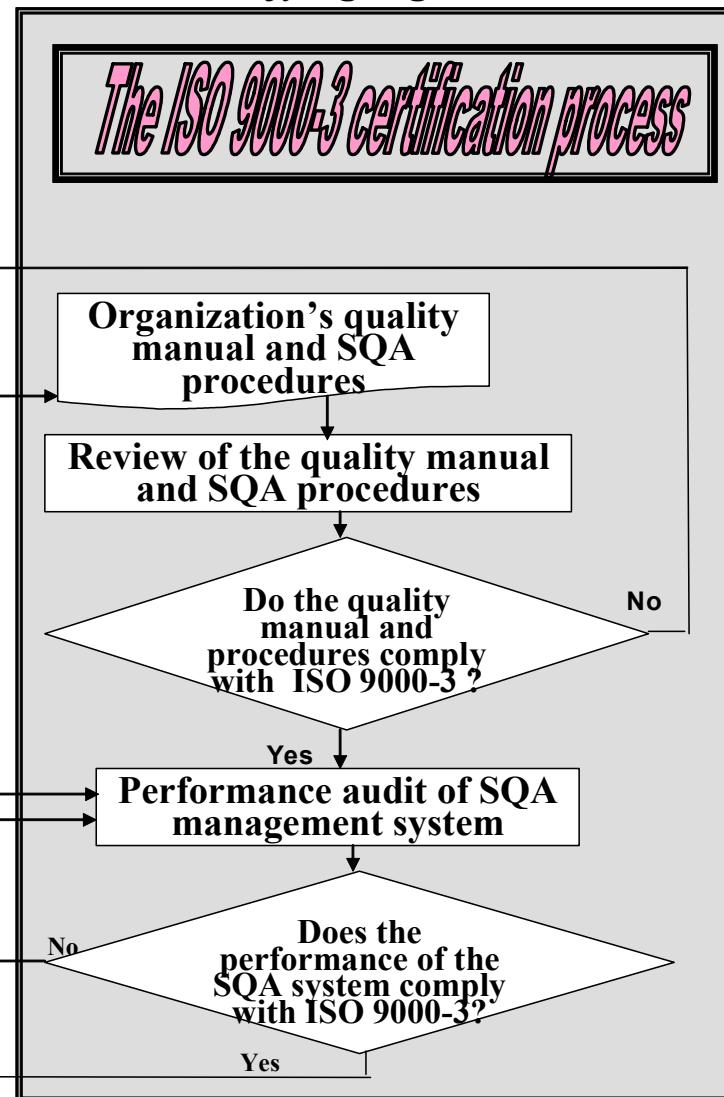
SQA from theory to implementation

OHT 1.325

Organization requesting certification



The certifying organization



SQA from theory to implementation

The principles of CMM assessment

- ◊ Quantitative management methods increases the organization's capability to control the quality and improve the productivity.
- ◊ Application of the five-level capability maturity model that enables to evaluate the achievements and determine the efforts needed to reach the next capability.
- ◊ Generic process areas that define the “what” — not “how” enables the model's applicability to a wide range of implementation organizations:
 - It allows use of any life cycle model.
 - It allows use of any design methodology, development tool and programming language.
 - It does not specify any particular documentation standard.

OHT 1.327

Time required to progress to the next CMM level

Capability level transition	Mean time (months)	No. of organizations
Level 1 to level 2	24	125
Level 2 to level 3	21.5	124
Level 3 to level 4	33	18
Level 4 to level 5	18	19

Source: Based on Gartner Inc. (2001)

OHT 1.328

Project resources distribution by CMM capability level

The case of Raytheon

CMM capability level	Percentage of project resources		
	Original work	Reworking	Testing and quality assurance
1	34	41	25
2	55	18	27
3	67	11	22
4	76	7	17

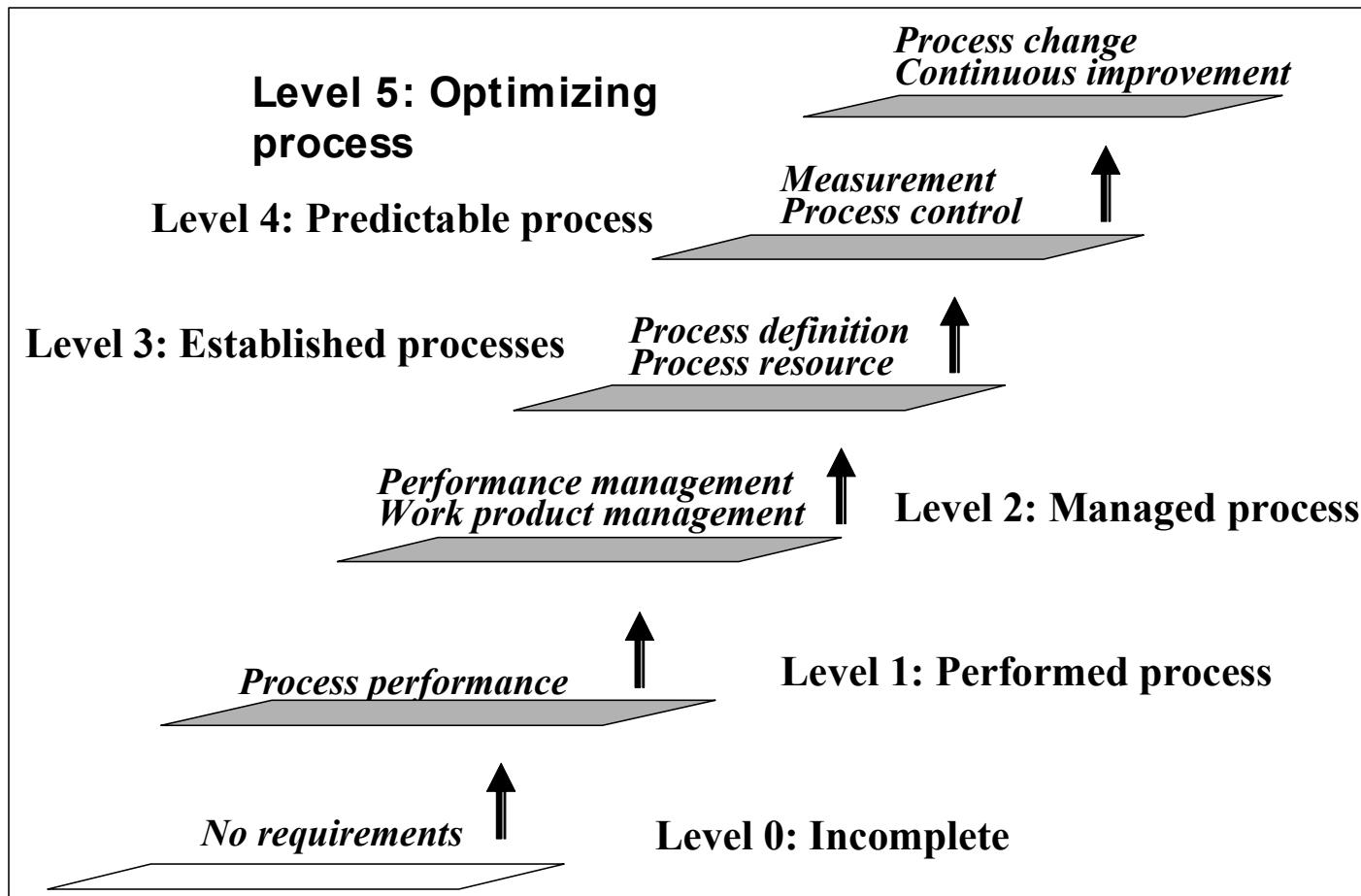
SQA from theory to implementation

Versions of CMMI (Capability Maturity Model Integration)

- **CMMI-SE/SW**
 - System Engineering CMM (SE-CMM)
 - Software engineering CMM (SW-CMM)
- **CMMI-SE/SW/IPPD/SS**
 - System Engineering CMM (SE-CMM)
 - Software engineering CMM (SW-CMM)
 - Integrated Product/Process Development (IPPD-CMM)
 - Supplier Sourcing
- **CMMI-SE/SW/IPPD**
 - System Engineering CMM (SE-CMM)
 - Software engineering CMM (SW-CMM)
 - Integrated Product/Process Development (IPPD-CMM)

OHT 1.330

ISO/IEC 15504 process assessment model



SQA from theory to implementation

Principles of the ISO/IEC 15504 assessment model

- ** **Harmonize** the many existing “independent” assessment methodologies by providing a comprehensive framework model (“what” has to be accomplished rather than “how” it has to be done).
- ** **Be universal** to serve all or almost all categories of software suppliers, customers and software categories.
- ** **Be highly professional.**
- ** **Aim at reaching international acceptance** as world standard. To save suppliers' resources by eliminating the need to perform several different capability assessments in response to different customer requirements.

The goals of the SPICE project trials

- ◊ To validate the ISO/IEC 15504 model's conformity with current standards.
- ◊ To verify its usability in determining whether software satisfies user requirements.
- ◊ To gain experience in applying the ISO/IEC 15504 model.

SQA project process standards

IEEE software engineering standards

- The structure and content of IEEE software engineering standards
- IEEE/EIA Std. 12207 – Software life cycle processes
- IEEE Std. 1012 - verification and validation
- IEEE Std. 1028 - reviews

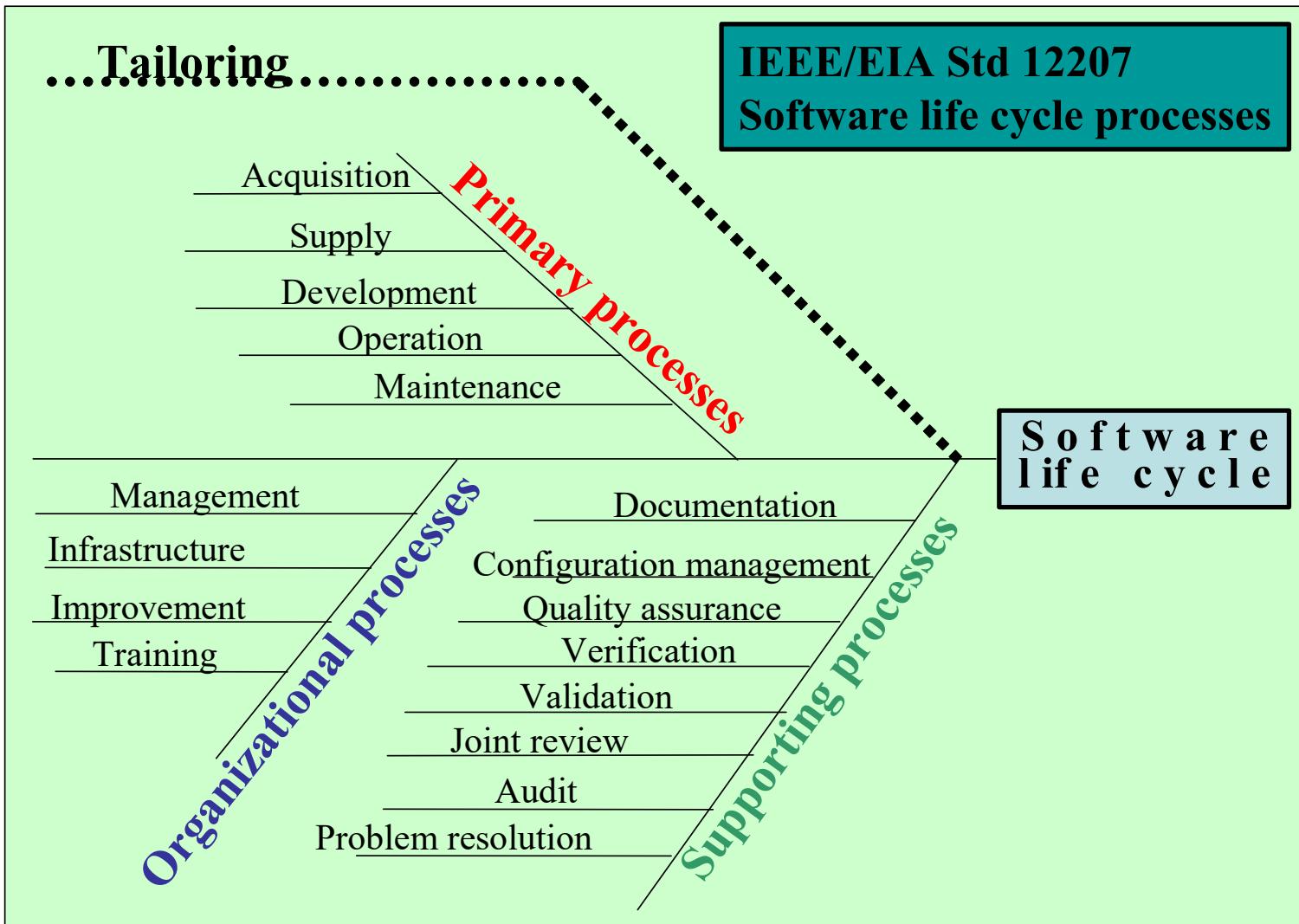
Classes of IEEE standards

- **A. Conceptual standards.** Guiding principles and overall approach
 - * *IEEE 1061 – Software Quality Metrics Methodology*
 - * *IEEE/EIA 12207.0 — Information Technology Software Life Cycle Processes*
- **B. Prescriptive standards of conformance.** Requirements to which a software developer must conform.
 - * *IEEE 829 — Software Test Documentation*
 - * *IEEE 1012 – Software Verification And Validation*
 - * *IEEE 1028 – Software Reviews*
- **C. Guidance standards.** Implementation of class B standards.
 - * *IEEE 1233 – Guide for Developing System Requirement Specifications*
 - * *IEEE/EIA 12207.1 – Guide, Information technology – Software Life Cycle Processes – Life Cycle Data*

The purpose of IEEE/EIA Std 12207

- ◊ To establish an internationally recognized model of common software life cycle processes that can be referenced by the software industry worldwide.
- ◊ To promote understanding among business parties by application of commonly recognized processes, activities and tasks.

OHT 1.336



Source: IEEE (1992). From IEEE Std 10 45-19992. Copyright 1992 IEEE. All rights reserved.

SQA from theory to implementation

IEEE/EIA Std 12207.0 concepts

General concepts

- Applicability of the standard in general and its adaptation by tailoring
- Applicability for all participants in the software life cycle
- Flexibility and responsiveness to technological change
- Software links to the system
- TQM consistency
- No certification requirements
- Baselining

Task-related concepts

- Responsibility for activities and tasks
- Modularity of components of software life cycle
- Levels of required conformance
- Nature of evaluation task

The purpose of IEEE Std 1012

- * **Establish** a common framework for V&V activities and tasks for all software life cycle processes.
- * **Define** V&V requirements, including their inputs and outputs.
- * **Define** software integrity levels and the appropriate V&V tasks.
- * **Define** the content of a SVVP (software V&V Plan) document.

IEEE Std 1012 concepts

- Broad definition of V&V activities
- Software integrity levels and their V&V requirements
- Prescriptive requirements
 - * Detailed description of the performance methodology.
 - * Required inputs.
 - * Required outputs.
 - * Definition of integrity levels for which performance of the task is not mandatory.
 - * Optional V&V tasks to be performed during selected life cycle process.
- Independence of V&V activities
- Compliance and compatibility with international standards
- Special characteristics of reusable software V&V
- Application of V&V metrics
- Quantitative criteria for V&V tasks

The processes covered by IEEE Std 1012

- (1) Management
- (2) Acquisition
- (3) Supply
- (4) Development
- (5) Operation
- (6) Maintenance

A three level tree architecture:

- Processes (each includes 1-6 activities)
- Activities (each includes 3-10 tasks)
- Tasks

Types of reviews

covered by IEEE Std. 1028

- Management reviews
- Technical reviews (in the book
“formal design reviews”)
- Inspections
- Walkthroughs
- Audits

The purpose of IEEE Std 1028

To define systematic review procedures
that are:

- * Applicable for reviews performed throughout the software life cycle
- * Conform with the review requirements defined by other standards

IEEE Std 1028 concepts

- High formality
- Follow-up of corrections
- Compliance with international and IEEE standards

Review requirements of IEEE Std.1028

Document structure:

- (1) **Introduction**
- (2) **Responsibilities** The responsibilities of the participants in the review.
- (3) **Input** Mandatory and optional data items.
- (4) **Entry criteria** Common criteria: a. a statement of the review's objectives. b. Availability of the required input data.
- (5) **Procedure** Required to include: management preparations, planning, team preparation, examination of the products, follow up of corrections.
- (6) **Exit criteria** What must be accomplished before the review can be concluded.
- (7) **Output items**
- (8) **Data collection recommendations** To be used to study the effectiveness and efficiency of current practices.
- (9) **Improvements** Formulate improved procedures, checklists and development processes.

Management and its role in software quality assurance

- The quality assurance organizational framework
- Top management's quality assurance activities
 - Software quality policy
 - The executive in charge of software quality
 - Management review
- Department management responsibilities for quality assurance processes
- Project management responsibilities for quality assurance

OHT 1.346

The quality assurance organizational framework

List of participants (“actors”)

Managers

- * Top management executives, especially the executive in charge of SQA
- * Software development and maintenance department managers
- * Software testing department managers
- * Project managers and team leaders of development and maintenance projects
- * Leaders of software testing teams

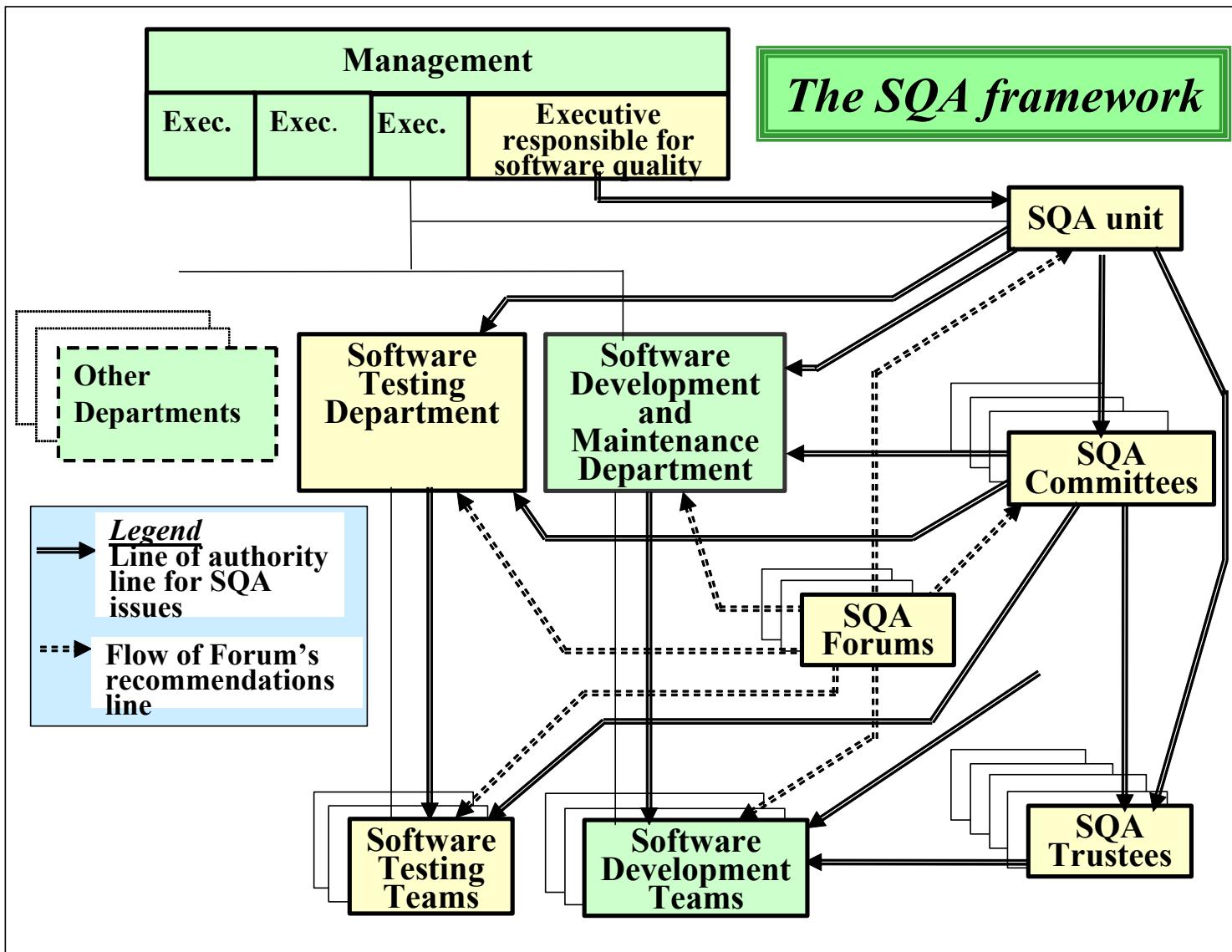
Testers

- * Members of software testing teams

SQA professionals and interested practitioners

- * SQA trustees
- * SQA committee members
- * SQA forum members
- * SQA unit team members

OHT 1.347



SQA from theory to implementation

OHT 1.348

Top management's overall responsibilities for software quality

- * Assure the quality of the Company's software products and software maintenance services.
- * Communicate the importance of product and service quality in addition to customer satisfaction to employees.
- * Assure full compliance with customer requirements.
- * Ensure that SQA objectives are established and accomplished.
- * Initiate planning and oversee implementation of changes to adapt the SQA system to changes related to the organization's clientele, competition and technology.
- * Intervene directly to resolve of crisis situations and minimize damages.
- * Ensure availability of resources required by SQA systems.

Software quality policy requirements

Conformity to the organization purpose and goals

Commitment to:

- * General software quality assurance concepts
- * The quality standards adopted by the organization
- * Allocate adequate resources for software quality assurance
- * Continuous improvement of the organizations quality and productivity

OHT 1.350

The responsibilities of the executive in charge of software quality

- **Responsibility** for preparation of an annual SQA activities program and budget
- **Responsibility** for preparation of SQA system development plans
- **Overall control** of implementation of the annual SQA regular activities program and planned SQA development projects
- **Presentation** and advocacy of SQA issues to executive management

OHT 1.351

Typical items contained in management review reports

***Management review* is the name given to the periodic meeting convened to allow executives to obtain an overview of their organization's software quality issues.**

- * Periodic performance reports, including quality metrics
- * Customer satisfaction feedback
- * Follow up reports for SQA annual regular activity program and SQA development projects
- * Summary of special quality events related to customers, suppliers, subcontractors, etc.
- * Review of significant findings of internal and external quality audits as well as special surveys
- * Identification of new software quality risks and unsolved pre-existing risks
- * Recommendations for software quality management improvements.

The objectives of management reviews

- **Assess** achievement of quality objectives set for the organization's software quality management system
- **Initiate** updates and improvements of the software quality management system and its objectives
- **Outline directions** for remedying major SQA deficiencies and software quality management problems.
- **Allocate** additional resources to the software quality management system.

Department management responsibilities for quality assurance (1)

The quality system-related responsibilities

Preparation of the department's annual SQA activities program and budget, based on recommended SQA unit program.

Preparation of the department's SQA systems development plans, based on recommended SQA unit plan.

Control of performance of the department's annual SQA activities program and development projects

Presentation of the department's SQA issues to the executive in charge of software quality.

Department management responsibilities for quality assurance (2)

Project-related responsibilities

- **Control** of compliance to quality assurance procedures in the department's units, including CAB, SCM and SCCA bodies
- **Detailed** follow up of contract review results and proposal approvals
- **Review** of unit performance of planned review activities; approval of project documents and project phase completion
- **Follow up** of software tests; approval of project's software products.
- **Follow up** of progress of software development project schedules and budget deviations. Advise and support project managers in resolving difficulties.
- **Follow up** of quality of maintenance services
- **Detailed follow up** of project risks and their solutions
- **Follow up** of project's compliance with customer requirements and customers satisfaction.
- **Approval** of large software change orders and significant deviations from project specifications.

OHT 1.355

Project management responsibilities for quality assurance

Professional hands-on tasks:

Preparation of project and quality plans and their updates.

Participation in joint customer-supplier committee

Close follow up of project team staffing, including recruitment, training and instruction.

Management tasks The **follow up** issues:

- * Performance of review activities and the consequent corrections, including participating in some reviews.
- * Software development and maintenance units' performance with respect to development, integration and system test activities, corrections and regression tests and acceptance tests
- * Software installation in customer sites and the running-in of the software system by the customer
- * SQA training and instruction of project team members
- * Schedules and resources allocated to project activities.
- * Customer requests and satisfaction
- * Evolving project development risks, application of solutions and control of results.

The SQA unit and other actors in the SQA system

The SQA unit

- Tasks performed by the head of the SQA unit
- SQA sub-unit tasks related to the project life cycle
- SQA sub-unit infrastructure operations tasks
- SQA sub-unit audit and certification tasks
- SQA sub-unit support tasks
- SQA sub-unit standards and procedures: Development and maintenance tasks
- SQA sub-unit information system tasks

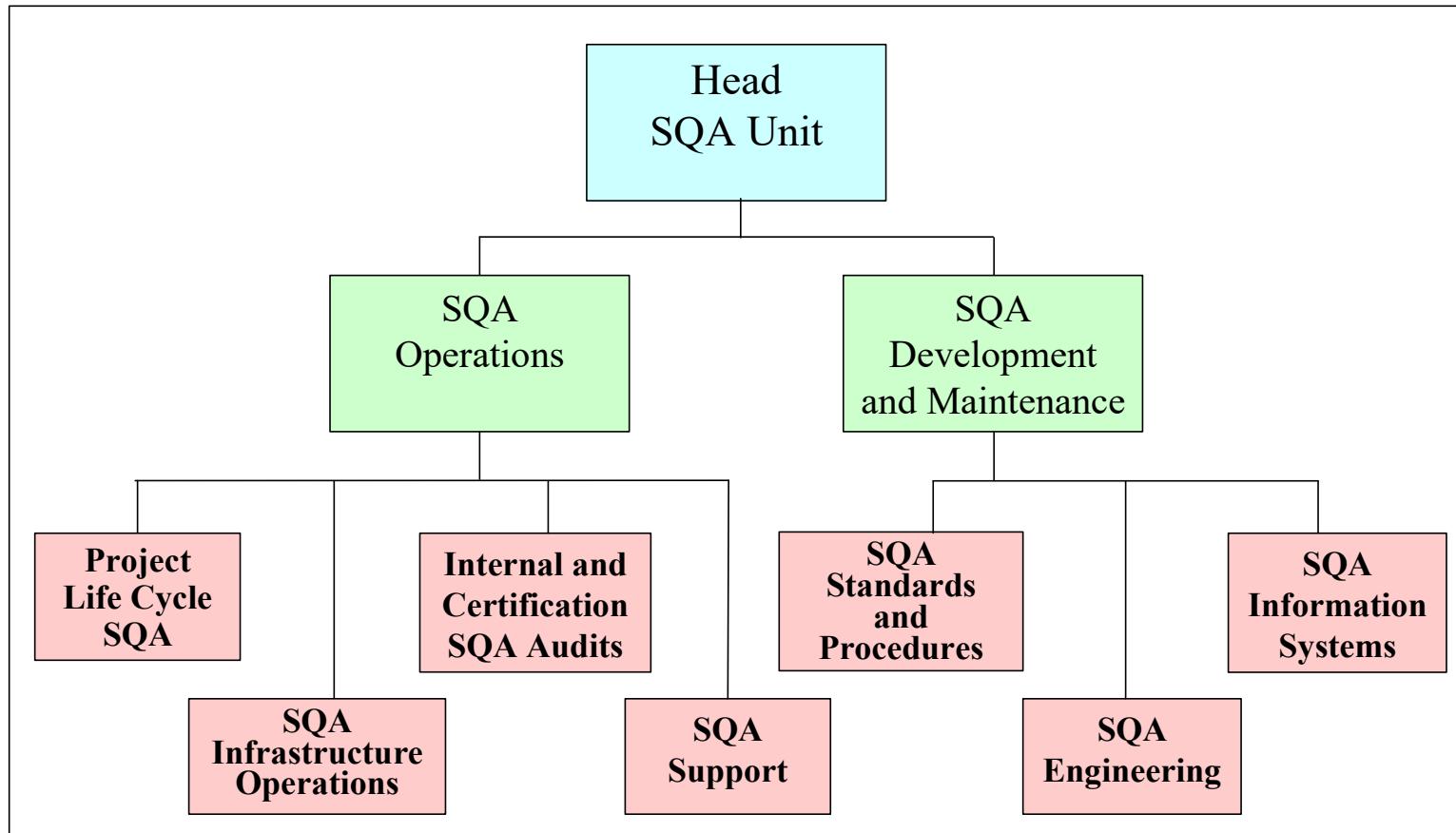
SQA trustees and their tasks

SQA committees and their tasks

SQA forums — tasks and methods of operation

OHT 1.357

Proposed model for an SQA Unit's organizational structure



SQA from theory to implementation

Tasks performed by the head of the SQA unit (1)

Planning tasks

- Preparation of proposals for the Unit's annual activity program and budget
- Planning and updating the organization's software quality management system and recommended annual SQA activities programs for the software development and maintenance departments.
- Preparation of recommended SQA systems development plans for the software development and maintenance departments.

Management tasks

- Management of SQA team's activities
- Monitoring implementation of the SQA activity program
- Nomination of team members, SQA committee members and SQA trustees
- Preparation of special and periodic status and performance reports.

Tasks performed by the head of the SQA unit (2)

Contacts with customers and other external bodies and the executive in charge of software quality

- Serving as the customer's address for software quality issues of software products and services supplied
- Representation of the organization before external bodies regarding software quality issues
- Drafting the management review reports
- Raising SQA organizational issues and preparing requested material for top management's consideration

SQA professional activities

- Participation in project joint committees
- Participation in formal design reviews
- Review and approval of deviations from specifications
- Consultation to project managers and team leaders
- Participation in SQA committees and forums

SQA sub-unit tasks related to the project life cycle

Project life cycle control tasks

- Follow up of development and maintenance teams' compliance with SQA procedures and work instructions
- Approval or recommendation of software products (design reports and code).
- Monitoring delivery of software maintenance services to internal and external customers
- Monitoring customer satisfaction (surveys, etc.) and maintaining contact with customer's SQA representatives

Participation tasks participation in:

- Contract reviews
- Preparation and updating of project development and project quality plans
- Formal design reviews
- Subcontractors' formal design reviews
- Software testing, including customer acceptance tests
- Software acceptance tests of subcontractors' software products
- Installation of new software products

OHT 1.361

SQA sub-unit

infrastructure operations tasks

- Publication of updated versions of procedures, work instructions, templates, checklists, etc., with their circulation.
- Training and instruction to new and current staff and SQA trustees regarding SQA procedures, work instructions, new and revised procedures, development tools and methods, etc.
- Monitoring and supporting implementation of new and revised SQA procedures
- Follow up of staff certification activities
- Proposal of subjects requiring preventive and corrective actions, including participation in CAB committees
- Follow up of configuration management activities, including participation in CCA committees
- Follow up of compliance with documentation procedures and work instructions

SQA sub-unit

infrastructure operations tasks

- Publication of updated versions of procedures, work instructions, templates, checklists, etc., and their circulation
- Transmission of training and instruction regarding application of SQA procedures, work instructions and similar items to new and current staff
- Instruction of SQA trustees regarding new and revised procedures, development tools and methods, etc.
- Monitoring implementation of new and revised SQA procedures
- Follow up of staff certification activities
- Proposal of preventive and corrective actions, including participation in CAB committees
- Follow up of configuration management activities, including participation in CCA committees
- Follow up of compliance with documentation procedures and work instructions

Types of SQA audits carried out in or by software organization

- Internal audits
- Audits of subcontractors and suppliers to evaluate their SQA systems
- External audits performed by certification bodies
- External audits performed by customers who wish to evaluate the SQA system prior to accepting the organization as a supplier

OHT 1.364

SQA sub-unit audit and certification tasks

- Preparation of annual programs for SQA audits
- Performance of SQA audits
- Follow up of corrections
- Preparation of periodic summary reports
- Collection of data on the performance of the audited organization from internal and external sources
- Periodic evaluation of the audited organization
- Coordination of the external audit's contents and schedule
- Preparation of documents as specified by external auditors
- Instruction of the audited teams and performance of preparations for external audits
- Participation in the audit

SQA sub-unit support tasks

- Preparation of project development plans and project quality plans
- Staffing review teams
- Choice of development methodologies and tools that reflect the accumulated failure experience
- Choice of measures to solve identified software development risks
- Choice of measures to solve schedule delays and budget overruns
- Choice of SQA metrics and software costs components
- Use of SQA information systems

SQA sub-unit

standards and procedures

development and maintenance tasks

- Prepare an annual program for development of new procedures and procedure updates
- Responsibility for development of new procedures and procedure updates, including participation in appropriate committees and forums
- Follow up of developments and changes in SQA and software engineering standards; introduction of additional relevant procedures and changes
- Initiation of updates and adaptations of procedures in response to changes in professional standards, including adoption or deletion of standards applied by the organization.

SQA sub-unit

engineering development and maintenance tasks

- Testing quality and productivity aspects with respect to new development tools and new versions of currently used development tools
- Evaluation of quality and productivity of new and improved development and maintenance methods
- Development of solutions to difficulties confronted in application of currently used software development tools and methods
- Development of methods for measuring software quality and team productivity
- Provision of technological support to CAB committees during analysis of failures and formulation of solutions

SQA sub-unit information system tasks

- **Development of SQA information systems for software development and maintenance units for:**
 - Collection of activity data.
 - Processing of information delivered by the units: periodic reports, lists, exception reports, queries and estimates of software quality metrics and software quality costs.
- **Updating of SQA information systems**
- **Development and maintenance of the organization's SQA Intranet/Internet site**

SQA trustees and their tasks

Unit-related tasks:

- Support their colleagues' attempts to solve difficulties in the implementation of SQA procedures and work instructions
- Help their unit manager in performing his or her SQA tasks Promote compliance and monitor implementation of SQA procedures and work instructions by colleagues
- Report substantial and systematic non-compliance events to the SQA unit
- Report severe software quality failures to the SQA unit

Organization-related tasks

- Initiate changes and updates of organization-wide SQA procedures and work instructions
- Initiate organization-wide improvements of development and maintenance processes and applications to the CAB for solutions to recurrent failures observed in their units
- Identify organization-wide SQA training needs and propose an appropriate training or instruction program

SQA committees and their tasks

Permanent committees commonly deal with:

- SCC (software change control),
- CA (corrective actions),
- Procedures,
- Development of method, tools and quality metrics.

Ad-hoc committees commonly deal with specific cases:

- Updates of a specific procedure,
- Analysis and solution of a software failure,
- Elaboration of software metrics for a targeted process or product,
- Updating software quality costs,
- Data collection methods for a specific issue.

SQA forums — tasks

SQA forums typically focus on:

- SQA procedures improvements and implementation
- Quality metrics
- Corrective actions – analysis of failure and success cases
- Quality system issues – development and implementation of new tools
- Quality line management problems – daily operational software quality problems

SQA forums — participants

Members of an open forum may include:

- SQA unit members
- SQA trustees
- Software development and maintenance staff
- SQA and software engineering consultants/experts
- Customer representatives