SOFTWARE QUALITY ASSURANCE

UNIT III

Assuring the quality of software maintenance components: the foundation of high quality – pre maintenance software quality components -maintenance software quality assurance tools. Assuring the quality of external participant contributions: types of external participants – risks and benefits of introducing external participants – assuring the quality of external participant contribution: objectives. CASE tools and their effect on software quality: CASE tool –contribution of CASE tool to software project quality, to software maintenance quality to improved project management.

Assuring the quality of software maintenance components:

Introduction

The following three components of maintenance service are all essential for success:

- **Corrective maintenance** – user support services and software corrections.
- **Adaptive maintenance** – adapts the software package to differences in new customer requirements, changing environmental conditions and the like.

- **Functionality improvement maintenance** – combines
  (1) **perfective maintenance** of new functions added to the software so as to enhance performance, with
  (2) **preventive maintenance** activities that improve reliability and system infrastructure for easier and more efficient future maintainability.

Software maintenance QA activities: objectives

1. Assure, with an accepted level of confidence, that the software maintenance activities conform to the functional technical requirements.
2. Assure, with an accepted level of confidence, that the software maintenance activities conform to managerial scheduling and budgetary requirements.
3. Initiate and manage activities to improve and increase the efficiency of software maintenance and SQA activities.

The foundations of high quality

Foundation 1: software package quality

The two product operation factors are as follows.

(1) **Correctness** – includes:

- **Output correctness:** The completeness of the outputs specified (in other words, no pre-specified output is missing), the accuracy of the outputs (all system's outputs are processed correctly), the up-to-datedness of the outputs (processed information is up

to date as specified) and the availability of the outputs (reaction times do not exceed the specified maximum values, especially in online and real-time applications).

- **Documentation correctness.** The quality of documentation: its completeness, accuracy, documentation style and structure. Documentation formats include hard copy and computer files – printed manuals as well as electronic "help" files – whereas its scope encompasses installation manuals, user manuals and programmer manuals.
- **Coding qualification.** Compliance with coding instructions, especially those that limit and reduce code complexity and define standard coding style.

(2) **Reliability**. The frequency of system failures as well as recovery times.

The three product revision factors are as follows.

(1) **Maintainability.** These requirements are fulfilled first and foremost by following the software structure and style requirements and by implementing programmer documentation requirements.

(2) **Flexibility**. Achieved by appropriate planning and design, features that provide an application space much wider than necessary for the current user population. In practice, this means that room is left for future functional improvements.

(3) **Testability.** Testability includes the availability of system diagnostics to be applied by the user as well as failure diagnostics to be applied by the support center or the maintenance staff at the user's site.

The two product transition factors are as follows.

(1) **Portability.** The software's potential application in different hardware and operating system environments, including the activities that enable those applications.

(2) **Interoperability.** The package's capacity to interface with other packages and computerized equipment. High interoperability is achieved by providing capacity to meet known interfacing standards and matching the interfacing applied by leading manufacturers of equipment and software.

Foundation 2: maintenance policy

Version development policy

This policy relates mainly to the question of how many versions of the software should be operative simultaneously.

Change policy

Change policy refers to the method of examining each change request and the criteria used for its approval.

Pre-maintenance software quality components

Like pre-project SQA components, the pre-maintenance SQA activities to be completed prior to initiating the required maintenance services are of utmost importance. These entail:

- Maintenance contract review
- Maintenance plan construction.

Maintenance contract review

(1) Customer requirements clarification
(2) Review of alternative approaches to maintenance provision
(3)Review of estimates of required maintenance resources
(4) Review of maintenance services to be provided by subcontractors and/or the customer
(5) Review of maintenance costs estimates

Maintenance plan

Maintenance plans should be prepared for all customers, external and internal.

The plan includes the following:

(1) A list of the contracted maintenance services
(2) A description of the maintenance team's organization
(3) (3) A list of maintenance facilities
(4) (4) A list of identified maintenance service risks

(5) A list of required software maintenance procedures and controls
(6) The software maintenance budget

Maintenance software quality assurance tools

1.SQA tools for corrective maintenance
2.SQA tools for functionality improvement maintenance
3.SQA infrastructure components for software Maintenance

- Maintenance procedures and work instructions
- Supporting quality devices
- Training and certification of maintenance teams
- Preventive and corrective actions
- Configuration management
- Documentation and quality record control.

4.Managerial control SQA tools for software maintenance

- Performance controls for corrective maintenance services
- Quality metrics for corrective maintenance

- Costs of software maintenance quality.

Costs of prevention – Costs of error prevention, i.e. costs of instruction and training of maintenance team, costs of preventative and corrective actions.

- **Costs of appraisal** – Costs of error detection, i.e. costs of review of maintenance services carried out by SQA teams, external teams and customer satisfaction surveys.
- **Costs of managerial preparation and control** – Costs of managerial activities carried out to prevent errors, i.e. costs of preparation of maintenance plans, maintenance team recruitment and follow-up of maintenance performance.
- **Costs of internal failure** – Costs of software failure corrections initiated by the maintenance team (prior to receiving customer complaints).
- **Costs of external failure** – Costs of software failure corrections initiated by customer complaints.
- **Costs of managerial failure** – Costs of software failures caused by managerial actions or inaction, i.e. costs of damages resulting from shortage of maintenance staff and/or inadequate maintenance task organization.

Costs of external failure of software corrective maintenance activities

(1) For software corrections:

- All costs of software correction initiated by users during the warranty period are external quality costs because they are considered to result directly from software development failures; hence, the developer is responsible for their correction during this period.
- Software corrections performed during the contracted maintenance period are considered part of regular service, as the responsibility of the developer for corrections is limited to the warranty period. As such, the costs of these services are considered regular service costs and not quality costs.
- During the contracted maintenance period, only costs of re-correction after failure of the initial correction efforts are considered external failure costs as the software technician failed in his regular maintenance service.

(2) For user support services:

- During the warranty period, user support services are considered to be an inherent part of the instruction effort, and therefore should not be considered external failure costs.
- During the contracted maintenance period, all types of user support services, whether dealing with an identified software failure or consultations about

application options, are all part of regular service, and their costs are not considered external failure costs.

- During both maintenance periods, an external failure is defined as a case where a second consultation is required after the initial consultation proves to be inadequate. The costs of furnishing the second and further consultations for the same case are considered external failure costs.

Assuring the quality of external participants' contributions

Types of external participants

External participants can be classified into three main groups:

(1) **Subcontractors** (currently called "outsourcing" organizations) that undertake to carry out parts of a project, small or large, according to circumstances. Subcontractors usually offer the contractor at least one of the following benefits: staff availability, special expertise or low prices.

(2) **Suppliers of COTS software and reused software modules.**

The advantages of integrating these ready elements are obvious, ranging from timetable and cost reductions to quality. One expects that integration of these ready-for-use elements will achieve savings in development resources, a

shorter timetable and higher quality software. Software of higher quality is expected as these components have already been tested and corrected by the developers as well as corrected according to the faults identified by previous customers. The characteristics of COTS software and quality problems involved in their use are discussed by Basili and Boehm (2001).

(3) **The customer themselves as participant in performing the project.**

It is quite common for a customer to perform parts of the project: to apply the customers' special expertise, respond to commercial or other security needs, keep internal development staff occupied, prevent future maintenance problems and so forth. This situation does have drawbacks in terms of the customer–supplier relationship necessary for successful performance of a project, but they are overweighed by the inputs the customer makes. Hence, the inevitability of this situation has become a standard element of many software development projects and contractual relations.

Risks and benefits of introducing external participants

(1) **Delays in completion of the project.** In those cases where external participants are late in supplying their parts to the software system, the project as a whole will be delayed. These delays are typical for subcontractors'parts

and customers' parts but less so for COTS software suppliers. In many cases the control over subcontractors' and the customers' software development obligations is loose, a situation that causes tardy recognition of expected delays and leaves no time for the changes and reorganization necessary to cope with the delays and to limit their negative effects on the project.

(2) **Low quality of project parts supplied by external participants.** Quality problems can be classified as (a) defects: a higher than expected number of defects, often more severe than expected; and (b) non-standard coding and documentation: violations of style and structure in instructions and procedures (supposedly stipulated in any contract). Low quality and non-standard software are expected to cause difficulties in the testing phase and later in the maintenance phase. The extra time required to test and correct low-quality software can cause project delays even in cases when external participants complete their tasks on time.

(3) **Future maintenance difficulties.** The fact that several organizations take part in development but only one of them, the contractor, is directly responsible for the project creates two possibly difficult maintenance situations:

(a) One organization, most probably the contractor, is responsible for maintenance of the whole project, the arrangement commonly stipulated in the tender itself. The contractor may then be faced with incomplete and/or non-standard coding and documentation supplied by the

external participants, causing lower-quality maintenance service delivered by the maintenance team and higher costs to the contractor.

(b) Maintenance services are supplied by more than one organization, possibly the subcontractors, suppliers of COTS software and occasionally the customer's software development department. Each of these bodies takes limited responsibility, a situation that may force the customer to search for the body responsible for a specific software failure once discovered.

Damages caused by software failures are expected to grow in "multi-maintainer" situations. Neither of these situations contributes to good and reliable maintenance unless adequate measures are taken in advance, during the project's development and maintenance planning phases.

(4) **Loss of control over project parts.** Whether intentionally or not, the control of software development by external bodies may produce an unrealistically optimistic picture of the project's status. Communication with external participants' teams may be interrupted for several weeks, a situation that prevents assessment of the project's progress. As a result, alerts about development difficulties, staff shortages and other problems reach the contractor belatedly. The possibilities for timely solution of the difficulties – whether by adaptations or other suitable changes – are thereby often drastically reduced.

Introduction of external participants: benefits and risks

Benefits

For the contractor:

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

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

Risks

For the contractor and the customer:

1. Delayed completion of the project caused by delays in completion of parts supplied by external participants
2. Low quality of parts supplied by external participants
3. Increased probability of difficulties in maintaining parts supplied by external participants
4. Loss of control over development of specific project parts

Assuring the quality of external participants' contributions: objectives

(1) To prevent delays in task completion and to ensure early alert of anticipated delays.
(2) To assure acceptable quality levels of the parts developed and receive early warnings of breaches of quality requirements.

(3) To assure adequate documentation to serve the maintenance team.
(4) To assure continuous, comprehensive and reliable control over external participants' performance.

CASE tools and their effect on software quality

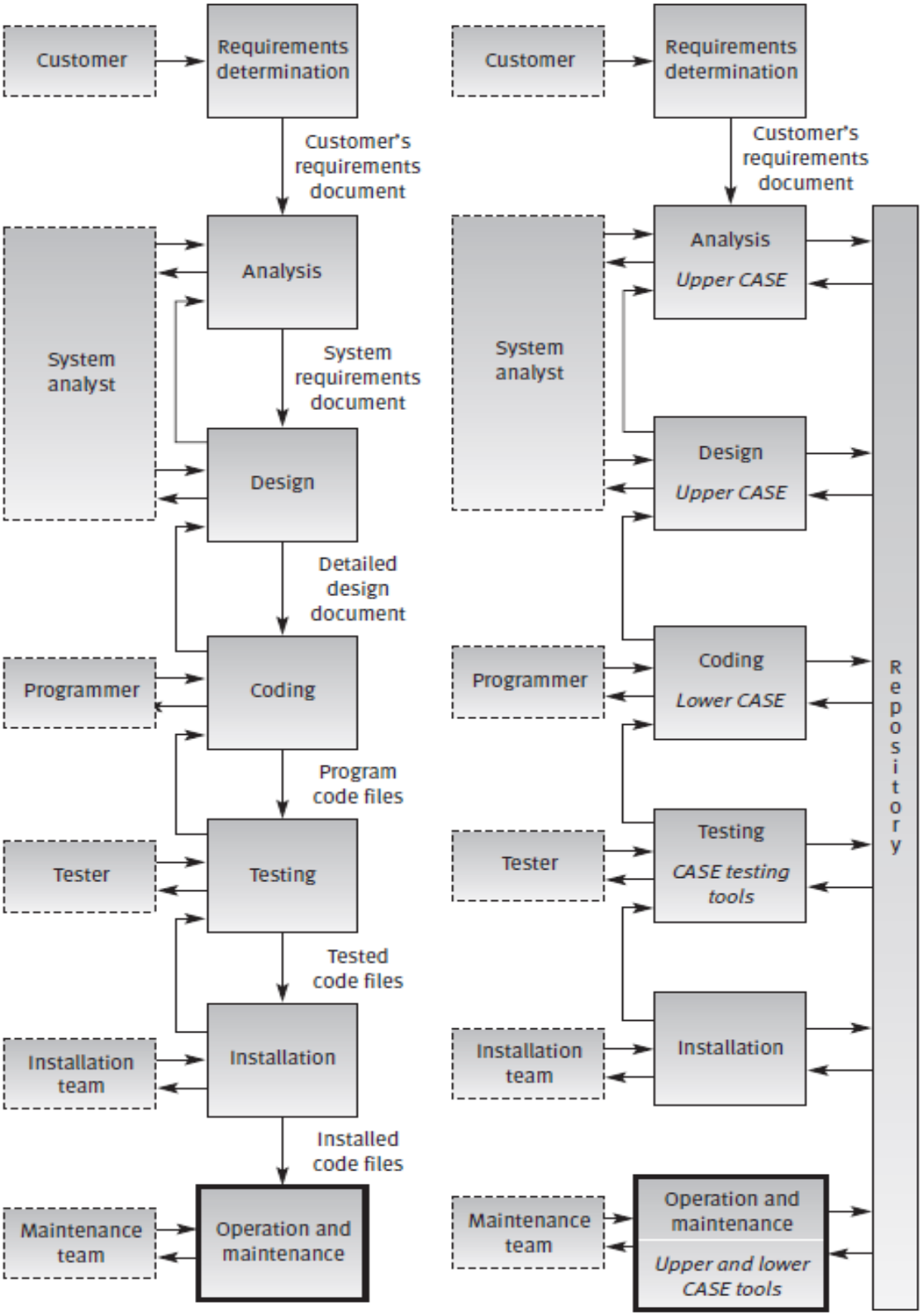
What is a CASE tool?

CASE tools – definition

CASE tools are computerized software development tools that support the developer when performing one or more phases of the software life cycle and/or support software maintenance.

The definition's generality allows compilers, interactive debugging systems, configuration management systems and automated testing systems to be considered as CASE tools. In other words, well-established computerized software development support tools (such as interactive debuggers, compilers and project progress control systems) can readily be considered *classic* CASE tools, whereas the new tools that support the developer for a succession of several development phases of a development project are referred to as real CASE tools.

Type of CASE tool	Support provided
Editing and diagramming	Editing text and diagrams, generating design diagrams according to repository records
Repository query	Display of parts of the design texts, charts, etc.; cross-referencing queries and requirement tracing
Automated documentation	Automatic generation of requested documentation according to updated repository records
Design support	Editing design recorded by the systems analyst and management of the data dictionary
Code editing	Compiling, interpreting or applying interactive debugging code for specific coding language or development tools
Code generation	Transformation of design records into prototypes or application software compatible with a given software development language (or development tools)
Configuration management	Management of design documents and software code versions, control of changes in design and software code*
Software testing	Automated testing, load testing and management of testing and correction records, etc.
Reverse engineering (re-engineering)	Construction of a software repository and design documents, based on code: the "legacy" software system. Once the repository of the legacy software is available, it can be updated and used to automatically generate new versions of the system. As new re-engineered software version is generated, it can be easily maintained and its documentation automatically updated
Project management and software metrics	Support progress control of software development projects by follow-up of schedules and calculation of productivity and defects metrics



(a) Traditional development life cycle

(b) Real CASE tool-supported development life cycle

The contribution of CASE tools to software product quality

CASE tools contribute to software product quality by reducing the number of errors introduced in each development phase.

Cause of software errors	Extent and manner of contribution to quality: Classic CASE tools	Extent and manner of contribution to quality: Real CASE tools
1. Faulty requirements definition		**Almost no contribution** Computerized examination of requirements consistency or correctness is rarely possible.
2. Client–developer communication failures		**Almost no contribution** In most cases, computerized identification of communication failures is impossible. Communication failures can be located or prevented only when a change or other information is found to be inconsistent with repository information.
3. Deliberate deviations from software requirements		**High contribution** Based on information stored in the repository, deviations from recorded requirements are identified as inconsistent and labeled as errors. Such deviations can also be identified by repository-based requirements tracing tools and cross-referenced query tools.
4. Logical design errors		**High contribution** (1) Re-engineering enables automated generation of the design of legacy systems and their recording in a repository.

Cause of software errors	Extent and manner of contribution to quality Classic CASE tools	Real CASE tools
4. Logical design errors		**High contribution** (2) Use of the repository is expected to identify design omissions, changes and additions inconsistent with repository records.
5. Coding errors	**Very high contribution** Application of compilers, interpreters and interactive debuggers.	**Very high contribution** Application of lower CASE tools for automated code generation achieves full consistency with the design recorded in the repository. In addition, as coding is automatic, no coding errors are expected.
6. Non-compliance with coding and documentation instructions	**Limited contribution** Use of text editors and code auditing supports the standardization of structure and style of texts and code and facilitates identification of non-compliance.	**Very high contribution** Application of lower CASE tools for automated code generation assures full compliance with documentation and coding instructions.
7. Shortcomings in the testing process	**High contribution** Automated testing tools perform full regression and automated load testing. Computerized management of testing and corrections reduces errors by improvement follow-up.	**High contribution** Application of lower CASE but especially of integrated CASE tools prevents coding errors and reduces design errors. Application of repository tools (cross-referenced queries and performance consistency checks) to corrections and changes during the development process prevent most software errors.
8. Procedural errors	**High contribution** Control of versions, revisions and software installation by means of software configuration management tools.	**Limited contribution** Use of updated and full documentation is expected to prevent many of the maintenance errors caused by incomplete and/or inaccurate documentation, especially if the design has been revised several times.
9. Documentation errors	**Limited contribution** Application of text editors only	**High contribution** Use of repository automatically generates full and updated documentation prior to each correction or change.

The contribution of CASE tools to software maintenance quality

Corrective maintenance:

- CASE-generated full and updated documentation of the software enables easier and more reliable identification of the cause for software failure.
- Cross-referenced queries enable better identification of anticipated effects of any proposed correction.
- Correction by means of lower CASE or integrated CASE tools provides automated coding, with no expected coding errors as well as automated documentation of corrections.

Adaptive maintenance:

- Full and updated documentation of the software by CASE tools enables thorough examination of possible software package adaptations for new users and applications.

Functional improvement maintenance:

- Use of the repository enables designers to assure consistency of new applications and improvements with existing software systems.
- Cross-referenced repository queries enable better planning of changes and additions.

- Changes and additions carried out by means of lower CASE or integrated CASE tools enable automated coding, with no expected coding errors as well as automated documentation of the changes and additions.

The contribution of CASE tools to improved project management

Let us compare two projects of similar nature and magnitude: Project A is carried out by conventional methods, Project B by advanced CASE tools.

The following results were obtained after comparison of the planning and implementation phases:

Method of development	Project A Conventional tools	Project B CASE tools
Planned resources (man-months)	35	20
Actual resources invested	42	27
Planned completion time (months)	15	9
Actual completion time	18	12

Two items quickly attract our attention:

(1) The advanced CASE method was much more economical than the conventional method.

(2) The quality of management in both projects was similar, with resources and schedule estimated at below the required levels.

In general, application of CASE tools is expected to reduce project budgets and development time ("shorter time to market").