Overview of Software Quality Assurance & Testing

1 Overview	2 Life cycle components	3 Infrastructure components	4 Management components	5 Standards and Organizing
6 Static tesing	7 Dynamic testing	8 Test management	9 Tools	

Learning objectives

- Define software, software quality and software quality assurance
- Explain the structure (categories and factors) of McCall’s classic factor model
- Show the components of the SQA system
- Explain the fundamental principles in testing

References

- Galin (2004). *Software quality assurance from theory to implementation.* Pearson Education Limited
- Dorothy Grahamet, Erik van Veenendaal, Isabel Evans, Rex Black. *Foundations of software testing: ISTQB Certification*

Contents

What is software product?

- Software product is

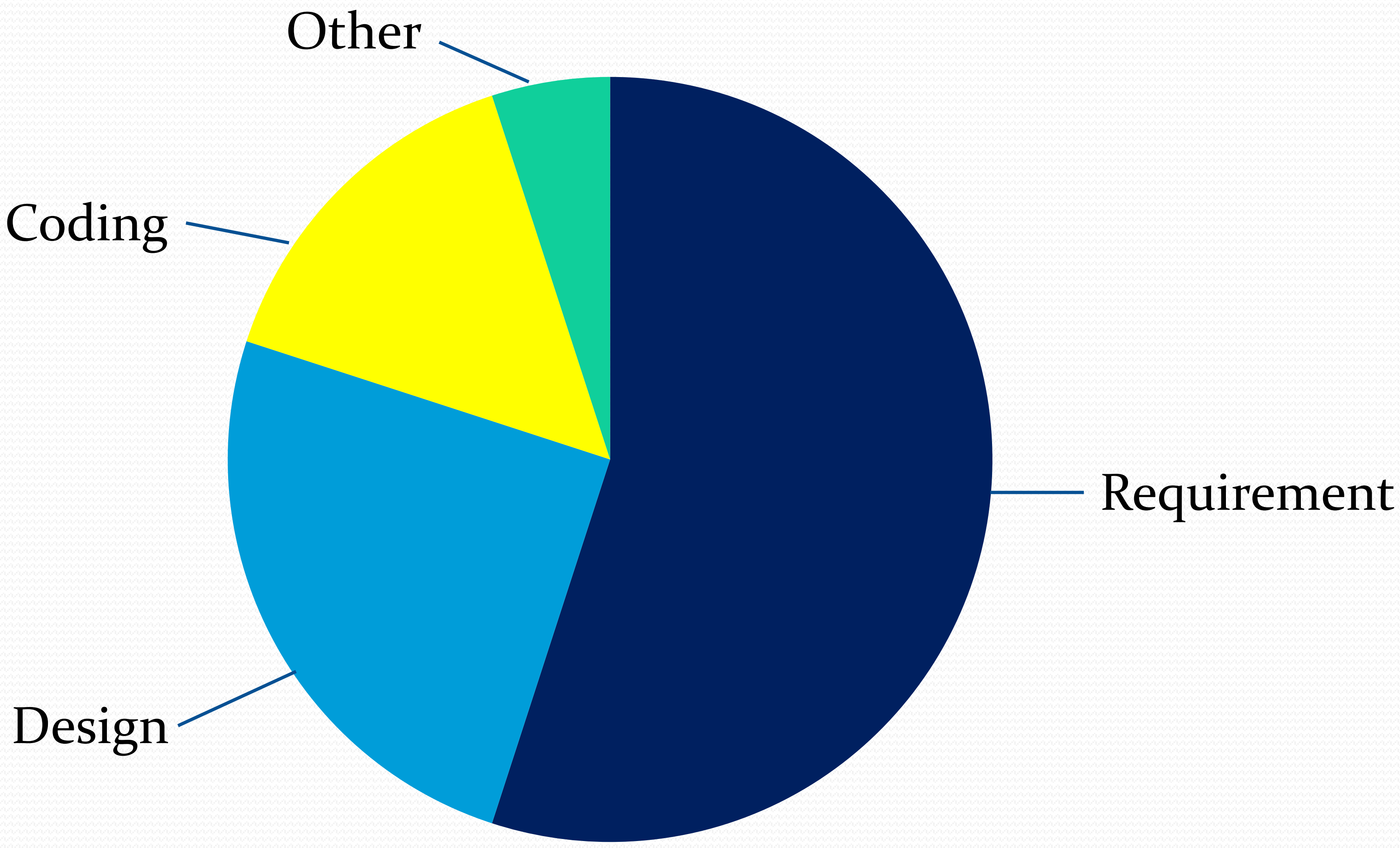
  Set of computer programs, procedures, and possibly associated documentation and data.

[ISO/IEC/IEEE Std. 90003:2014]

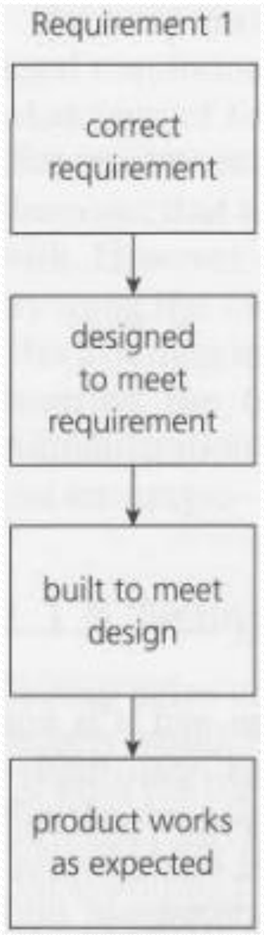
Causes of software errors

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. Procedure errors
9. Documentation errors

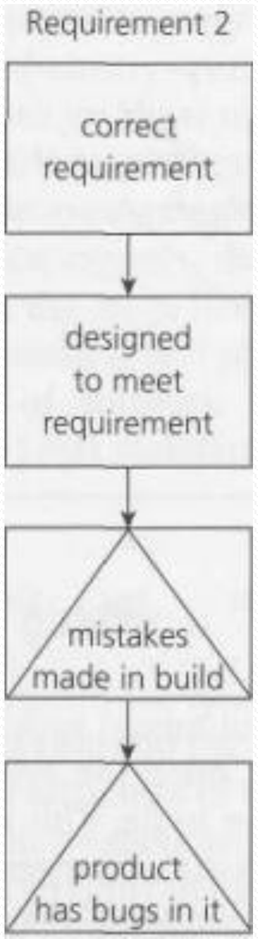
Cause of defect



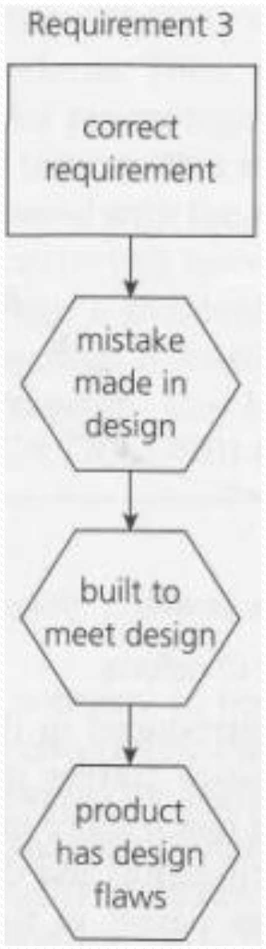
Defects in requirement, design, build



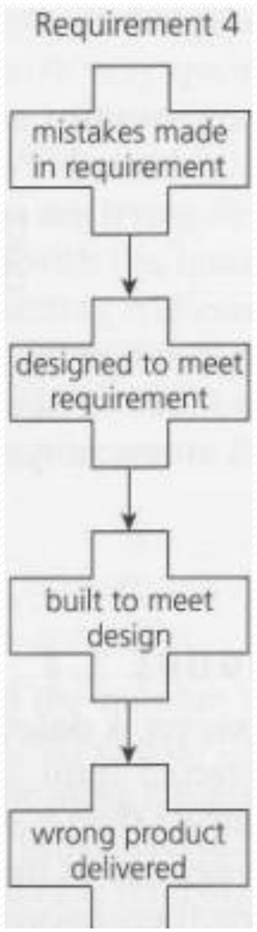
correct functional and non-functional attributes delivered



correctable defects

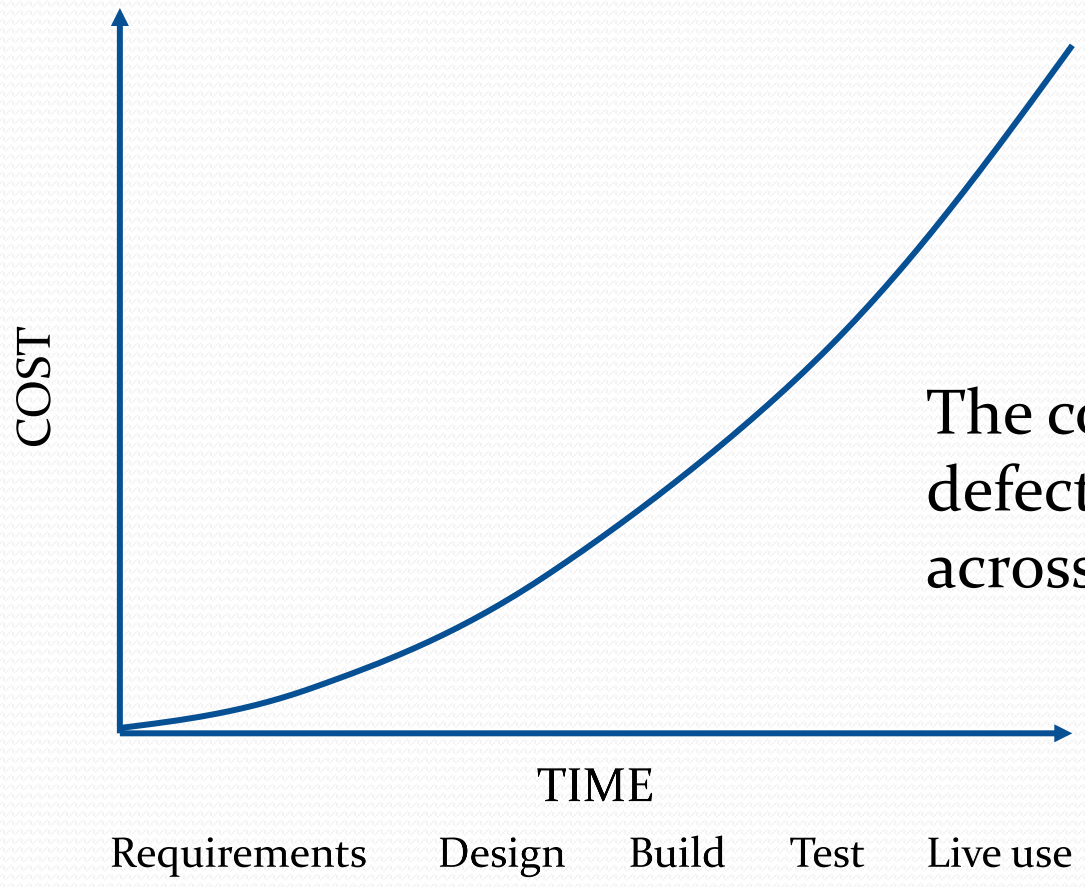


redesign to correct defects



defects may be hidden from the IT team including testers

The cost of defects



The cost of finding and fixing defects rises considerably across the life cycle

Software quality

- Software quality is:

The degree to which a software product meets established requirements; however, quality depends upon the degree to which established requirements accurately represent stakeholder needs, wants, and expectations.

[IEEE Std. 730-2014]

1	2	3	4	5
6	7	8	9	

Contents

Definition of SQA

- Software Quality Assurance is:

A set of activities that define and assess the adequacy of software processes *to provide evidence that establishes confidence that the* ***software processes*** *are appropriate for and produce* ***software products*** *of suitable quality for their intended purposes*. A key attribute of SQA is the objectivity of the SQA function with respect to the project. The SQA function may also be organizationally independent of the project; that is, free from technical, managerial, and financial pressures from the project.

[IEEE Std.730-2014]

SQA vs SQC

	QA	QC
Definition	SQA is a set of activities for ensuring quality in software engineering **processes**. The activities establish and evaluate the processes that produce products.	QC is a set of activities for ensuring quality in software **products**. The activities focus on identifying defects in the actual products produced.
Focus	Process focused	Product focused
Orientation	Prevention oriented	Detection oriented
Scope	Relates to all products that will ever be created by a process	Relates to specific product
Activities	• Process definition and implementation • Audits • Training…	• Reviews • Testing

Objectives of SQA

- Assuring, with acceptable levels of confidence, conformance to **functional technical requirements**
- Assuring, with acceptable levels of confidence, conformance to managerial requirements of **scheduling** and **budgets**
- Initiating and managing activities for the improvement and greater efficiency of software development and SQA activities

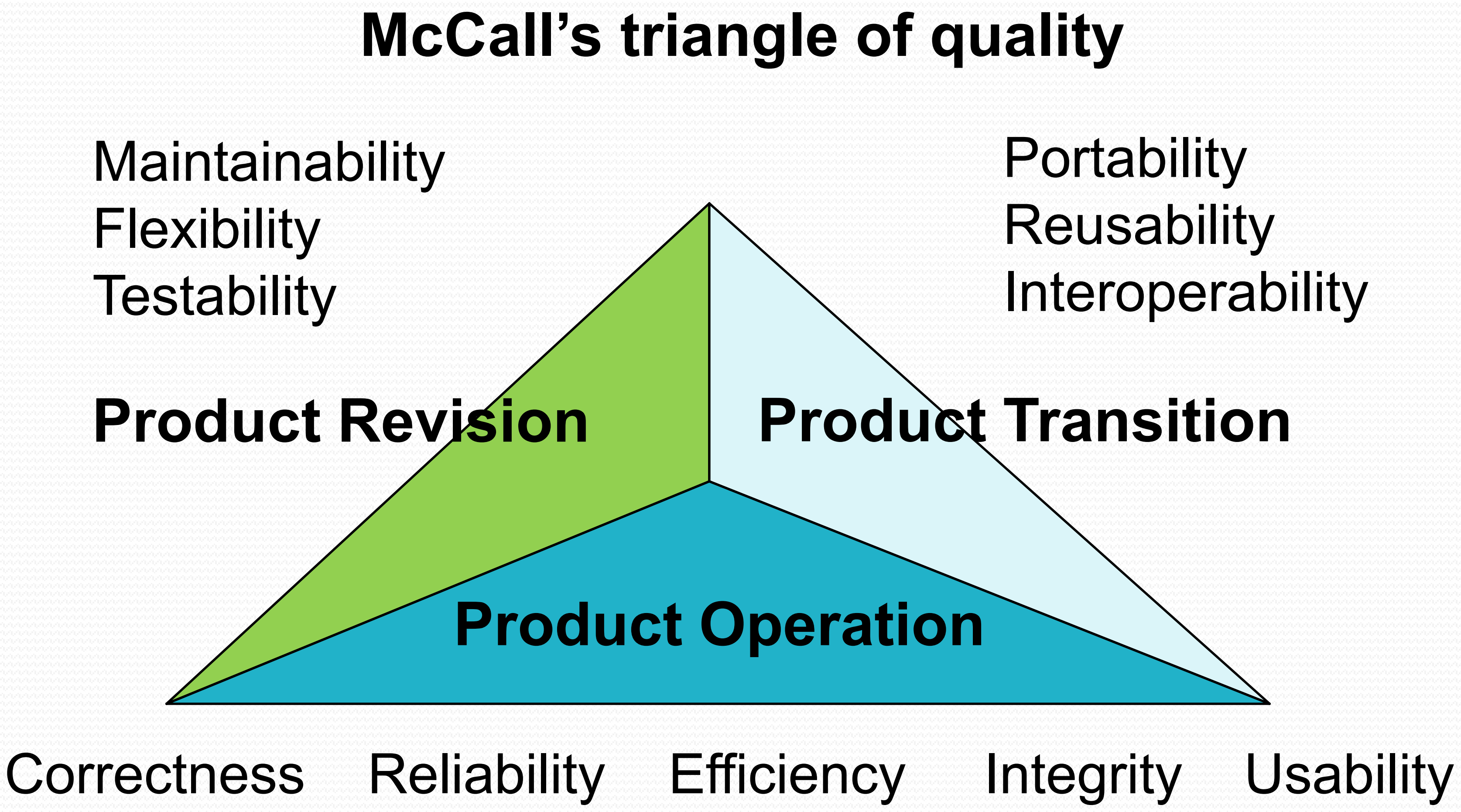
Contents

The need for comprehension software quality requirements

- Many cases of low customer satisfaction are situations which suffering from **poor performance** in other important areas such as maintenance, reliability, software reuse, or training
- One of the main causes: **lack of defined requirements** pertaining to these aspects of software functionality

  → need for **comprehensive definition of requirements** that will cover all aspects of software use throughout all stages of the software life cycle

The structure (categories and factors)



Product operation software quality factors

- Correctness
  - accuracy, completeness of required output
  - up-to-dateness, availability of the information
- Reliability
  - determine maximum allowed failure rate
- Efficiency
  - resources needed to all the perform software functions
- Integrity
  - software system security, access rights
- Usability
  - ability to learn, perform required task

Product revision software quality factors

- Maintainability
  - errors can be easily corrected as and when they show up
  - new functions can be easily added to the product
  - functionalities of the product can be easily modified, etc.
- Flexibility
  - degree of adaptability (to new customers, tasks, etc)
- Testability
  - support for testing (e.g. log files, automatic diagnostics, etc.)

Product transition software quality factors

- Portability
  - adaptation to other environments (hardware, software)
- Reusability
  - use of software components for other projects
- Interoperability
  - ability to interface with other components/systems

Review question

- Fill in the name of the factor that best fits the requirement

No	Section taken from the software requirement document	The requirement factors
1	The probability that the "Super-lab" software system will be found in a state of failure during peak hours (9 am to 4 pm) is required to be below 0.5%.	
2	The "Super-Lab" software system will enable direct transfer of laboratory results to those files of hospitalized patients managed by the "MD-File" software package.	

"Super-lab": A software system for managing a hospital laboratory

Review question

3	The "Super-Lab" software system will include a module that prepares a detailed report of the patient's laboratory test results during his current hospitalization. (This report will serve as an Appendix to the family physician's file). The time required to obtain this printed report will be less than 30 seconds; the level of accuracy and completeness will be at least 99%.	
4	The "Super-Lab" software to be developed for hospital laboratory use may be adapted later for private laboratory use.	

Review question

5	The training of a laboratory technician, requiring no more than 3 days, will enable the technician to reach level C of "Super-Lab" software usage. This means he or she will be able to manage reception of 20patients per Hour.	
6	The "Super-Lab" software system will record a detailed user's Log. In addition, the system will report attempts by unauthorized persons to obtain medical information from the laboratory test results data base. *The report will include the following informations: The network identification of the applying terminal, the system code of the employee who requested that information, the day and time of attempt and the type of attempt.*	

Review question

7	The system software documentation should be clear, self descriptive, and have a high degree of consistency.	
8	The software system should be able to serve 12 workstations and 8 automatic testing machines with a single model AS20 server and a cs25 communication server that will be able to serve 25 communication lines. This hardware system should conform to all availability requirements as listed in appendix C	
9	The “Super-Lab” software package developed for the Linux Operating System should be compatible for applications in a window NT environment.	

Alternative models of software quality factors

- The Evans and Marciniak factor model (Evans & Marciniak, 1987)
- The Deutsch and Willis factor model (Deustch & Willis, 1988)

Comparison

- Both exclude testability factors
- Evan & Marciniak[1]: 12 factors, classified into 3 categories
- Deustch & Willis[2]: 15 factors, classified into 4 categories
- 5 new factors: verifiability (both), expandability (both), safety[2], manageability[2], survivability[2]

Additional factors

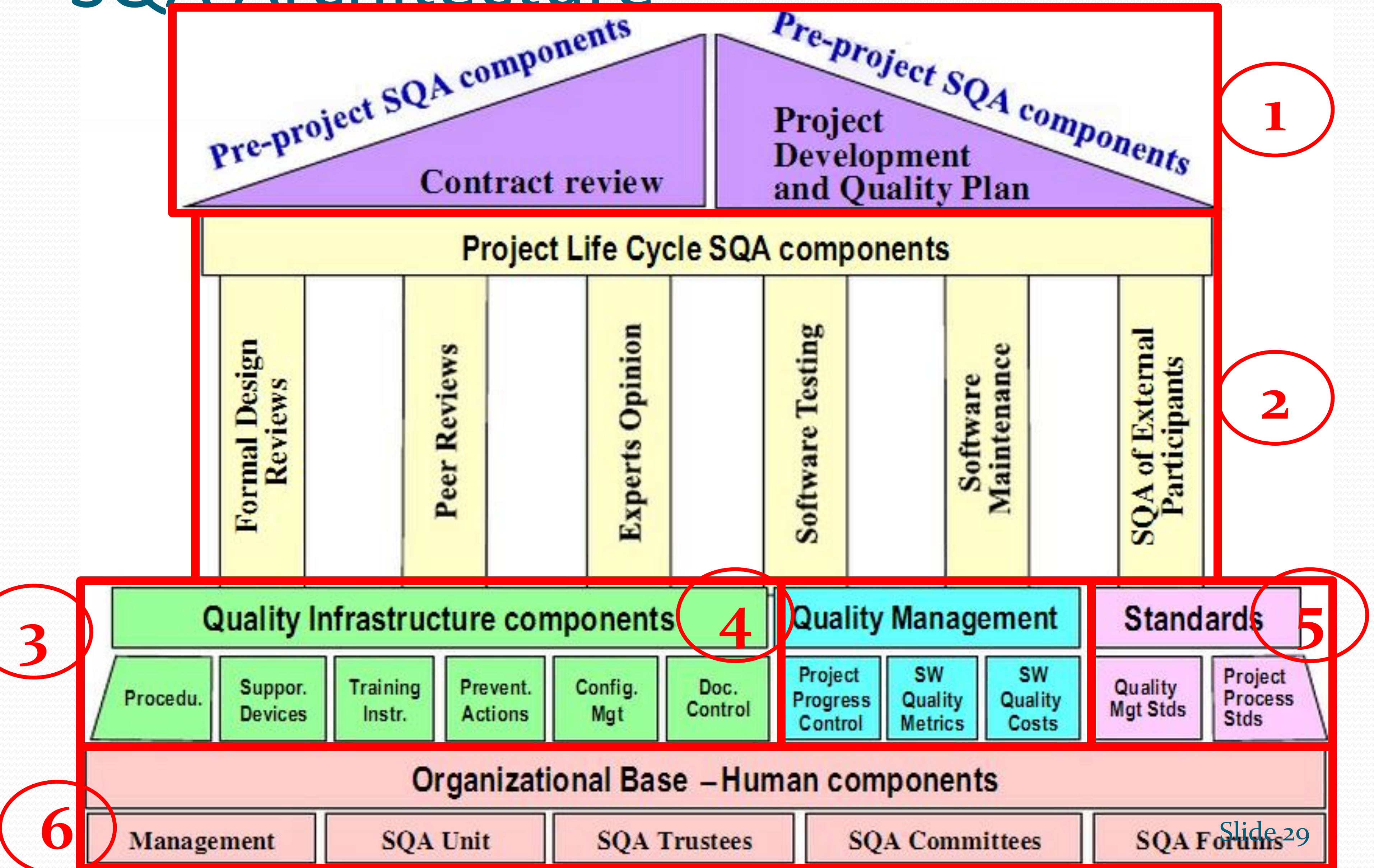
- Verifiability: design and programming features that enable efficient verification of the design and programming
- Expandability: future efforts to improve service, or add new applications to improve usability
- Safety: eliminate condition hazardous to operators of equipment as a results of errors in process control software
- Manageability: administrative tools that support software modification
- Survivability: continuity of service

1	2	3	4	5
6	7	8	9	

Contents

- Software, error and software quality
- Definitions and objectives of SQA
- Software quality factors
- **The components of the SQA system**
- What is testing?
- Testing principles
- Independent testing

SQA Architecture



Pre-project SQA components

- The preparatory steps taken prior to initiating work on the project itself
- To assure that
  - the project commitments have been adequately defined considering **resource, schedule** and **budget**
  - the **development** and **quality plans** have been correctly determined
- Two components in pre-project phase
  - contract review
  - development and quality plan

Pre-project: Contract review

- Must include a detailed examination of the project <u>proposal draft</u> and the <u>contract drafts</u>
  - Stage One – **Proposal** draft review: review of the final proposal draft prior to submission to the potential customer
  - Stage Two – **Contract** draft review: review of contract draft prior to signing

Pre-project: Contract review Proposal draft review objectives

- Objectives
  1. **customer requirements** have been clarified and documented
  2. **alternative approaches** for carrying out the project have been examined
  3. formal aspects of the relationship between the customer and the software firm have been specified
  4. identification of development **risks**
  5. adequate estimation of project **resources** and **timetable**
  6. examination of the **company's capacity** with respect to the project
  7. examination of the **customer's capacity** to meet his commitments
  8. definition of **partner** and **subcontractor** participation
  9. definition and protection of **proprietary rights**

Pre-project: Contract review
Contract draft review objectives

- Objectives
  - no un-clarified issues remain in the contract draft
  - all the understandings reached between the customer and the firm are to be fully and correctly documented in the contract and its appendices

Pre-project: Dev. and Quality plan

- Objectives
  - **scheduling** development activities, **estimating** the required manpower resources and budget
  - **recruiting** team members and **allocating** development resources
  - **resolving** development **risks**
  - implementing required SQA activities
  - providing management with data needed for project control

Pre-project: Dev. and Quality plan Elements of the 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

Pre-project: Dev. and Quality plan Elements of the quality plan

1. Quality goals
2. Planned review activities
3. Planned software tests
4. Planned acceptance tests for externally developed software
5. Configuration management plans

Pre-project: Dev. and Quality plan Quality plan: Quality goals [1/2]

- Refers to the developed software system's substantive quality requirements
- When choosing quality goals:
  - quantitative measures – more objective assessments of software performance
  - qualitative measures

Pre-project: Dev. and Quality plan
Quality plan: Quality goals [2/2]

- Example: Help desk system (HDS)

HDS qualitative requirements	Related quantitative quality goals
The HDS should be user friendly	A new help desk operator should be able to learn details of the HDS following a course lasting less than 8 hours, and to master operation of the HDS in less than 5 working days.
The HDS should be very reliable	HDS availability should exceed 99.5% (HDS downtime should not exceed 30 minutes per week).
The HDS should operate continuously	The system's recovery time should not exceed 10 minutes in 99% of cases of HDS failure.
…	…

Pre-project: Dev. and Quality plan
Quality plan: Planned review activities

- The quality plan should provide **a complete listing** of all planned review activities: design reviews (DRs), design inspections, code inspections, etc., with the following determined for each review activity:
  - the **scope**
  - the **type**
  - the **schedule**
  - the specific **procedures** to be applied
  - **who** is responsible for carrying out

Pre-project: Dev. and Quality plan
Quality plan: Planned software tests

- The quality plan should provide **a complete list** of planned software tests:
  - the **unit, integration** or the complete **system** to be tested
  - the **type** of testing activities
  - the test **schedule**
  - the specific **procedures** to be applied
  - **who** is responsible for carrying out

Pre-project: Dev. and Quality plan Quality plan: Planned acceptance tests for externally developed software

- Items to be included:
  - purchased software
  - software developed by subcontractors
  - customer-supplied software

Pre-project: Dev. and Quality plan
Quality plan: Configuration management

- The quality plan should specify **configuration management tools** and **procedures**, including those change-control procedures meant to be applied throughout the project

Contents

Why is testing necessary?

- Testing is necessary because we all make <u>mistakes</u>
  - we know something, but not everything
  - we have skills, but aren’t perfect
- Some of those mistakes can be unimportant; some of them can be costly and damaging (loss of money, time or business reputation), even may result in injury or death
- We need to *check everything and anything* we produce

So how testing is changed?

History of Software Testing
What? I've done the coding and now you want to test it. Why? We haven't got time anyway.
1960s - 1980s
Constraint
OK, maybe you were right about testing. It looks like a nasty bug made its way into the Live environment and now costumers are complaining.
1990s
Need
Testers! you must work harder! Longer! Faster!
2000+
Asset

60'– 80': Nice To Have | **90': Should Have** | **00': Must Have**

60'– 80': Nice To Have

- Black-box testing
- System testing
- Functional testing
- Ensure no obvious broken
- Part-time tester

90': Should Have

- Black/**grey-box** testing
- System/**Integration** testing
- Functional testing
- **Ensure requirements are meet**
- **Independent tester**

00': Must Have

- Black/grey/**white-box** testing
- **System-system testing**
- **Non-functional** testing
- Ensure requirements are meet and **Fit-for-Use**
- **Professional tester**

What is testing? [ISTQB definition]

- Testing is the process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products **to** determine that they satisfy specified requirements, **to** demonstrate that they are fit for purpose and **to** detect defects

process – involve a series of activities

all life cycle activities – testing takes place throughout the software development life cycle

static and dynamic – test and find defects when executed code and without executing code

What is testing? [ISTQB definition]

- Testing is the process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products
  to determine that they satisfy specified requirements,
  to demonstrate that they are fit for purpose and **to** detect defects

planning – to **manage** the testing

preparation – selecting **test conditions** and designing **test cases**

evaluation – check the results and evaluate the **completion criteria**,

software products and related work products - codes, requirements and design specifications, manual…

What is testing? [ISTQB definition]

- Testing is the process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products **to** determine that they satisfy specified requirements, **to** demonstrate that they are fit for purpose and **to** detect defects

  determine... – some of the testing we do is focused on checking products against the specification for the product

  demonstrate... –whether the software does what the user might reasonably expect

  detect defects – with root cause analysis, help to improve the development processes and make fewer error in future work

What is testing?

- Software testing is NOT:
  - a final exam
  - just finding broken code
  - correction of defects
  - “debugging”: the process that developers go through to identify the cause of defects in code and undertake corrections

1	2	3	4	5
6	7	8	9	

Contents

Testing principles

- **Principle 1 –** Testing shows presence of defects
- **Principle 2 –** Exhaustive testing is impossible
- **Principle 3 –** Early testing
- **Principle 4 –** Defect clustering
- **Principle 5 –** The pesticide paradox
- **Principle 6 –** Testing is context dependent
- **Principle 7 –** Absence of errors fallacy

Testing principles
P1 – Testing shows presence of defects

- Testing can show that defects are present, but cannot prove that there are no defects

- Testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness

Testing principles
P2 - Exhaustive testing is impossible

- Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases
  - e.g. you have 10 input fields to test, each having 5 possible values, the number of combinations to be tested would be 5^{10} = 9.765.625
- Instead of exhaustive testing, focus on RISK analysis and priorities, use risk to determine:
  - what to test first
  - what to test most
  - what not to test

Testing principles
P2 - Exhaustive testing is impossible

- How to prioritise? - Possible ranking criteria (all risk based)
  - test where a failure would be most severe
  - test where failures would be most visible
  - test where failures are most likely
  - ask the customer to prioritise the requirements
  - what is most critical to the customer’s business
  - areas changed most often
  - areas with most problems in the past
  - most complex areas, or technically critical

Testing principles
P3 - Early testing

- Testing activities should start as early as possible in the software or system development life cycle, and should be focused on defined objectives
  - errors identified late in the development process generally cost more to resolve
    - e.g. an error in a product specification may be fairly easy to fix, but if that error is transferred to the coding, then fixing the mistake could become costly and time-consuming

Testing principles
P4 - Defect clustering

- A small number of modules contain most of the defects
  - defects tend to cluster around narrow areas or functions ('hot spots')
    - 80–20 rule: approximately 80% of the problems are found in about 20% of the modules
  - by identifying and focusing on these clusters, testers can efficiently test the sensitive areas while concurrently testing the remaining areas

Testing principles
P5 – The pesticide paradox

- If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new bugs
- To overcome this 'pesticide paradox', the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects

Testing principles
P6 – Testing is context dependent

- Testing is done differently in different contexts
  - the same tests should not be applied across the board because different software products have varying requirements, functions and purposes
    - for example, *safety-critical software* is tested differently from an *e-commerce site*
  - not all software systems carry the same level of risk and not all problems have the same impact when they occur
    - suppose a user interface has typographical defects. Does this matter
      - if it’s in my personal family-tree website?
      - if it’s in the company website?

Testing principles
P7 – Absence of errors fallacy

If we don't find defects does that mean the users will accept the software?

- Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations
  - a test that finds no errors is different than concluding that the software is error-free
  - testers should assume that all software contains some faults, even if faults are hidden

Contents

Who is tester?

- A variety of different people may be involved in testing process
  - programmers do test their own code
  - business analysis should review their own requirements…
- It is difficult to find our own mistakes: find 30% - 50% of your own faults
- Testing can be **more effective** if it is **not undertaken by the creator** because of the different mindset
  - if we are building something we are working positively to solve problems
  - when we test or review a product, we are looking for defects in the product

Independent testing

- Several stages of reviews and testing may be **carried out independently**
  - professional testers, specialists, users,..
- This degree of independence avoids author bias and is often more effective at finding defects and failures

Levels of independence

- None: tests designed by the person who wrote the software
- Tests designed by a different person within the same team (e.g. another programmer)
- Tests designed by someone from a different department or team (e.g. test team)
- Tests designed by someone from a different organisation (e.g. outsourced testing, agency)
- Tests generated by a tool (low quality tests?)

Review

1. Nêu các khái niệm: chất lượng phần mềm (IEEE), đảm bảo chất lượng phần mềm (IEEE), kiểm thử phần mềm (ISTQB).
2. Cho 1 vd về sự cải tiến sản phẩm để nâng cao chất lượng.
3. Lợi ích của SQA?
4. Nêu các nguyên nhân của lỗi phần mềm và cho ví dụ.
5. Phân loại các yếu tố chất lượng theo McCall và giải thích các yếu tố đó. Nêu bổ sung các yếu tố chất lượng khác mà bạn biết.
6. Nêu và giải thích 7 nguyên tắc kiểm thử PM.
7. Liệt kê các thành phần trong hệ thống đảm bảo chất lượng phần mềm.

Preparation

- Đọc trước chương 2
- Tìm hiểu và trả lời các câu hỏi cuối chương 2 (có cộng điểm)

Any
Questions