What is Software Quality?

- Quality, simplistically, means that a product should meet its specification
- This is problematical for software systems
  - **Tension** between
    - customer quality requirements
      - efficiency, reliability, etc.
    - and developer quality requirements
      - maintainability, reusability, etc.
  - Some quality requirements are difficult to specify in an unambiguous way
  - Software specifications are usually incomplete and often inconsistent

5

The Questions for us

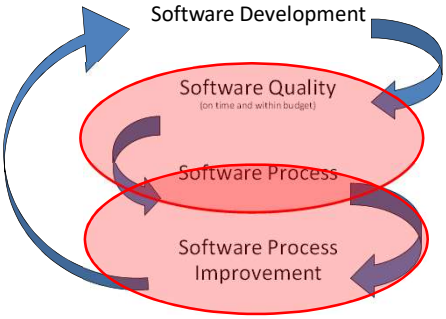
How do we create a

Quality Software Product

Quality Software Development Process is part of that

6

Context



7

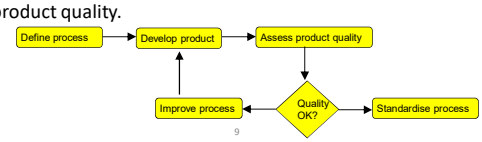
Process and Product quality

- The quality of a developed **product** is influenced by the quality of the production **process**.
- This is important in software development as some product quality attributes are hard to assess.
- However, **there is a very complex and poorly understood relationship between software processes and product quality**.

8

Process-based quality

- There is a straightforward link between process and product in manufactured goods.
- More complex for software because:
  - The application of individual skills and experience is particularly important in software development;
  - External factors such as the novelty of an application or the need for an accelerated development schedule may impair product quality.
- Care must be taken not to impose inappropriate process standards - these could reduce rather than improve the product quality.



9

Quality and Standards

- Standards are the key to effective quality management.
- They may be international, national, organizational or project standards.
- **Product standards** define characteristics that all components should exhibit e.g. a common programming style.
- **Process standards** define how the software process should be enacted.

10

Introduction to Software Quality

- Next Topics…
  - What is Software Quality?
  - Why is Software Quality important?
  - What is Software Quality Assurance ?
  - Software Quality factors
  - Elements of Software Quality Assurance
  - Development and quality plans
  - Process Standards

- Based on D. Galin Ch1 – 5 ++

11

Some Causes of Software Errors

- Faulty requirements definition
- Client-developer communication failures
- Deliberate deviations from software requirements
- Logical design errors
- Coding errors
- Non-compliance with documentation and coding instructions
- Shortcomings of the testing process
- User interface and procedure errors
- Documentation errors
- Many more…

13

What is Software Quality ?

- **Conformance to requirements**
- Lack of bugs
- Low defect rate (# of defects/size unit)
- High reliability (number of failures per n hours of operation)
  - Measured as Mean Time To Failure (MTTF) probability of failure-free operation in a specified time

15

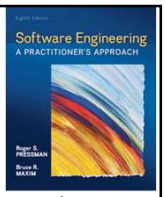
What is Software Quality ?

- According to the IEEE Software quality is:
  - The degree to which a system, component, or process meets specified requirements.
  - The degree to which a system, component, or process meets customer or user needs or expectations.

IEEE

16

What is Software Quality ?



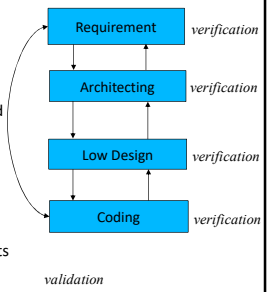
- According to Roger Pressman

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

17

Software Quality – 2 Core Terms

- Software Quality includes
- **Verification**
  - are we building the product right ?
  - performed at the end of a phase to ensure that requirements established during previous phase have been met
- **Validation**
  - are we building the right product ?
  - performed at the end of the development process to ensure compliance with product requirements



18

Importance of Software Quality

- Software is a major component of computer systems (significant % of the overall cost) – used for
  - communication (e.g. phone system, email system)
  - health monitoring,
  - transportation (e.g. automobile, aeronautics),
  - economic exchanges (e.g. e-commerce),
  - entertainment,
  - etc.
- Software defects are extremely costly in term of
  - money
  - reputation
  - loss of life

Importance of Software Quality

- Numerous historic disasters attributed to software
- Some good if somewhat dated examples are here: http://www.cse.psu.edu/~gxt29/bug/softwarebug.html
- Many more examples are reported on an ongoing basis.

Importance of Software Quality

- Monetary impact of poor software quality (Standish group report)
- 175,000 software projects/year - Average Cost per project
  - Large companies - $US 2,322,000
  - Medium companies - $US 1,331,000
  - Small companies - $US 434,000
- 31.1% of projects canceled before completed
  - cost $81 billion
- 52.7% of projects exceed their budget - costing 189% of original estimates
  - cost $59 billion
- 16.2% of software projects completed on-time and on-budget (9% for larger companies)
- Large companies - delivered systems have approximately only 42% of originally-proposed features and functions
- 78.4% of smaller companies projects get deployed with at least 74.2% of their original features and functions.

The Software Quality Challenge

- The uniqueness of the software product
  - High complexity
  - Invisibility of the product
  - Limited opportunities to detect defects ("bugs")
    - only opportunity is Product development
- The environments in which software is developed
  - 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

Software Quality in the SDLC

- **Software Quality activities should be integrated into the software development plan**
- The software development plan will implement one or more SDLC's
- The intensity and number of quality assurance activities is affected by project and team factors…

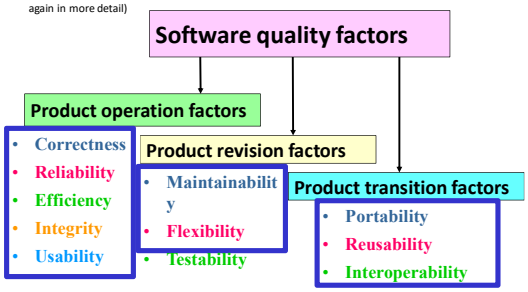
- SDLC phases
  - Requirements
  - Specification (Analysis)
  - Design
  - Implementation
  - Integration
  - Maintenance
  - Retirement
- In waterfalls, spirals, iterations, etc.

Factors affecting the required intensity of Software Quality 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

Software Quality Factors

- McCall's software quality factor model (we'll see this again in more detail)



29

Software Quality Factors

- Correctness
  - accuracy, completeness of required output, up-to-dateness, availability of the information
- Reliability
  - maximum failure rate
- Efficiency
  - resources needed to perform software function
- Integrity
  - software system security, access rights
- Usability
- ability to learn, perform required task
- Maintainability
  - effort to identify and fix software failures
- Flexibility
  - degree of adaptability (to new customers, tasks, etc)
- Testability
  - support for testing (e.g. log files, automatic diagnostics, etc)
- Portability
  - adaptation to other environments (hardware, software)
- Reusability
  - use of software components for other projects
- Interoperability
  - ability to interface with other components/systems

30

31

Software Quality Assurance?

- According to the IEEE 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.

32

What is Software Quality Assurance?

- According to D. Galin 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.

33

Objectives of SQA in 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.

34

Objectives of SQA in 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.

35

Three General Principles of QA

- Know what you **are doing**
- Know what you **should be doing**
- Know how to **measure** the **difference**

36

Three General Principles of QA

- Know what you are doing
  - understand **what** is being built, **how** it is being built and what it currently **does**
  - suppose a software development process with
    - management structure (milestones, scheduling)
    - reporting policies
    - tracking

37

Three General Principles of QA

- Know what you should be doing
  - having explicit **requirements** and **specifications**
  - suppose a software development process with
    - requirements analysis,
    - acceptance tests,
    - frequent user feedback

38

Three General Principles of QA

- Know how to measure the difference
  - having explicit **measures** comparing what is being done from what should be done
  - Some complementary methods:
    - **formal methods** – verify mathematically specified properties
    - **testing** – explicit input to exercise software and check for expected output
    - **inspections** – human examination of requirements, design, code, ... based on checklists
    - **metrics** – measures a known set of properties related to quality

39

SQA and Risk Management

- Risk management is an integral part of the process to develop software quality
  - Quality is not free…QA activities costs time and money…trade-offs are necessary
  - Quality assurance activities are risk reduction efforts
  - A “one-size-fits-all” quality assurance plan is rarely feasible
    - QA process tailoring is necessary
    - Risk management is an approach to intelligent process tailoring
- Risk Management
- Risk Identification
  - Risk Analysis and Assessment
  - Risk Planning & Mitigation
  - Risk Tracking

54

SQA and Risk Management

- **SEI Software Risk Taxonomy**

- **Use risk taxonomy as a checklist to identify risks and potential risk mitigation activities (i.e., QA process and products)**

55

Software Quality Control

- Process Enforcement
  - Independent agent (i.e., Software Quality Assurance manager/team)
  - Quantitative Metrics
  - Data Collection / Archiving
  - Review / Audit (process and products)
  - Authority to act
- Process Assessment
- Process Improvement Feedback
- Has QC changed in agile software development?

56

Software Metrics

- Necessary, under-used and hard
  - Forms one of the pillars of "prove-able" quality
  - Software and its development is complex and multi-dimensional, hard to understand and measure
- Scope of software metrics – process, products or resources
  - Project management
  - Cost and level-of-effort estimation
  - Productivity measures
  - Quality attributes
  - Reliability measures
  - Performance
  - Defect tracking
  - Structural and complexity metrics
  - Software engineering process metrics

57

SQ Goals
The Bottom Line…

- Defect Prevention
  - prevents defects from occurring in the first place
  - Activities: training, planning, and simulation
- Defect detection
  - finds defects in a software artifact
  - Activities: inspections, testing or measuring
- Defect removal
  - isolation, correction, verification of fixes
  - Activities: fault isolation, fault analysis, regression testing

59

Many challenges exist in S/W development…

Q. What is the right level of quality?
A. At any point in time there is an optimum QA/ROI proposition – but this is very difficult to achieve because our business is complex. This difficulty is amplified because change is unpredictable and (increasingly?) unavoidable.

Q. Manage a surplus or a deficit?
A. There are two types of struggle: (i) with deficient quality levels (which is soul destroying), (ii) struggle to justify SQA costs. Up to you which struggle to have – but is seems easier to manage a surplus!

The pursuit of appropriate levels of quality is effectively a personal philosophy. However, sub-standard work habits (be they of a technical or of an interactive nature) are eventually found out.

60

The aim of a professional S/W developer…

⇒ The aim should be to develop high quality, maintainable code through respectful collaboration with others… if everyone does this then we can all reasonably expect to be able to have rewarding professional lives.

⇒ Some thoughts that may assist you:

⇒ "It is easier to point a finger than it is to lend a hand"

⇒ You don't really understand another person's role until you have "walked a mile in their shoes"

⇒ Laziness in your approach to quality in all aspects of your professional life wont go unnoticed and will cost you in the long run… whenever you find yourself in a position of trying to correct some issue, always take the time to understand the root cause and implement a preventative action, otherwise…

61

Models of Software Quality Factors

- There are many models of software quality factors
  - **ISO/IEC 9126 (now ISO/IEC 25010)**
    - Its really a framework for the evaluation of software quality
  - **McCall's software quality factors model**
    - Dated from 1977 and is still relevant today
  - Boehm's Quality Model – 1978
  - Evans and Marciniak factor model (1987)
  - Deutsch and Willis (1988)

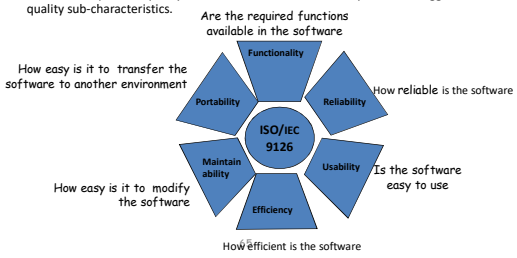
62

ISO/IEC 9126 Software engineering — Product quality

63

The Quality Factors defined by ISO/IEC 9126

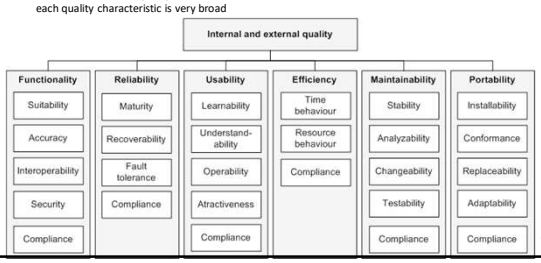
- The Quality Factors defined by ISO/IEC 9126 (see also ISO/IEC 25000)
  - Objective of standard is to provide a framework for the evaluation of software quality.
  - It does not provide requirements for software, but it defines a quality model which is applicable to every kind of software.
  - It defines six product quality characteristics and in an annex provides a suggestion of quality sub-characteristics.



65

ISO/IEC 9126 Quality Model

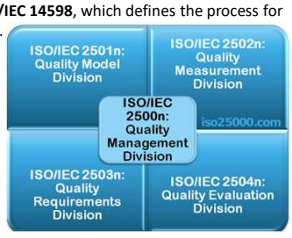
- ISO 9126/IEC provides the definition of the characteristics and associated quality evaluation process to be used when specifying the requirements for and evaluating the quality of software products throughout their life cycle.
- This standard does not provide subcharacteristics and metrics, nor the method for measurement, rating and assessment) .
- ISO 9126 sets out six quality characteristics, which are intended to be exhaustive. From this it follows that each quality characteristic is very broad



66

ISO/IEC 25000

- The series of standards **ISO/IEC 25000**, also known as **SQuaRE** (System and Software Quality Requirements and Evaluation), has the goal of creating a framework for the evaluation of software product quality.
- **ISO/IEC 25000** is the result of the evolution of several other standards; specifically from **ISO/IEC 9126**, which defines a quality model for software product evaluation, and **ISO/IEC 14598**, which defines the process for software product evaluation.
- The series of standards ISO/IEC 25000 consists of five divisions.



67

ISO/IEC 25000 series consists of five divisions.

- **Quality Management Division**
  - The standards that form this division define all common models, terms and definitions referred further by all other standards from SQuaRE series
- **Quality Model Division**
  - The standards that form this division present detailed quality models for computer systems and software products, quality in use, and data.
- **Quality Measurement Division**
  - The standards that form this division include a software product quality measurement reference model, mathematical definitions of quality measures, and practical guidance for their application.
  - Presented measures apply to software product quality and quality in use
- **Quality Requirements Division**
  - The standard that forms this division helps specifying quality requirements. These quality requirements can be used in the process of quality requirements elicitation for a software product to be developed or as input for an evaluation process.
- **Quality Evaluation Division**
  - The standards that form this division provide requirements, recommendations and guidelines for software product evaluation.

68

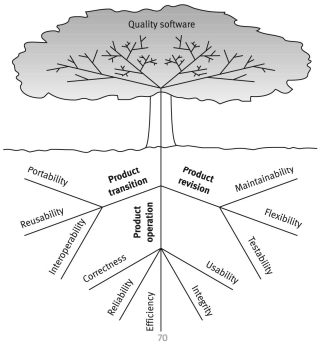
ISO/IEC 25010

Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models

- The quality model is the cornerstone of a product quality evaluation system.
  - The quality model determines which quality characteristics will be taken into account when evaluating the properties of a software product.
- The quality of a system is the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value.
  - Those stakeholders' needs (functionality, performance, security, maintainability, etc.) are precisely what is represented in the quality model, which categorizes the product quality into characteristics and sub-characteristics.
- The product quality model defined in ISO/IEC 25010 comprises the eight quality characteristics

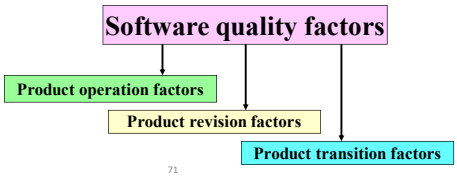
McCalls factor model tree

Galin Chapter 3 pp. 38

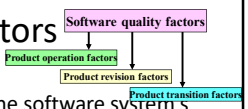


McCall's quality factor model

- 11 quality factors grouped into 3 categories
  - Product operation factors: Correctness, Efficiency, Integrity, Usability
  - Product revision factors: Maintainability, Flexibility, Testability
  - Product transition factors: Portability, Reusability, Interoperability



Product operation factors



- Correctness: Defined in a list of the software system's required output
  - The output mission (e.g. red alarms when temperature rises to 100 °C)
  - Required accuracy of the output (e.g. non-accurate output will not exceed 1%)
  - Completeness of the output info (e.g. probability of missing data less than 1%)
  - The up-to-dateness of the info (e.g. it will take no more than 1s for the information to be updated)
  - The availability of the info (e.g. reaction time for queries will be less than 2s on average)
  - The required standards and guidelines (the software and its docs must comply with the client's guidelines)

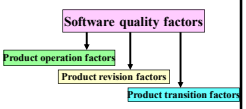
72

Product operation factors

(cont.)

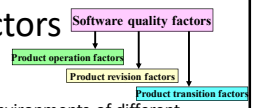
- **Efficiency**:
  - Focus is on hardware resources needed to perform the operations to fulfill the requirements (memory, storage, CPU speed, etc.)
- **Integrity**:
  - Requirements to prevent access to unauthorized persons
  - Rights management (e.g. limit the "write permit" to key personnel)
- **Usability**:
  - Deals with the scope of staff resources needed to train a new employee and to operate the software system
  - E.g. training of a new employee to operate the system will take no more than 2 working days (16h)

Product revision factors



- Maintainability:
  - Determines the effort needed to identify the causes for software failures, to correct them, and to verify the success of the corrections
  - Refers to the modular structure of the software as well as to the manuals and documentations
- Flexibility:
  - The effort required to support adaptive maintenance activities
    - E.g. man-days required to adapt a software package to a variety of customers of the same trade
- Testability:
  - Testability requirements include automatic diagnostics checks and log files, etc.
    - E.g. a standard test must be run every morning before the production begins

Product transition factors



- Portability:
  - Adaptation of the system to other environments of different hardwares, OS, etc.
- Reusability:
  - Mostly the developer himself will initiate the reusability requirement by recognizing the potential benefit of a reuse
  - It's expected to save development resources, shorten the development time and provide higher quality modules
- Interoperability:
  - Focuses on developing interfaces with other software systems or with other equipment firmwares
  - E.g a laboratory equipment is required to process its results (output) according to a standard data structure, which the laboratory information system can then use as an input

Comparing models

Galin Chapter 3 pp. 45

No.	Software quality factor	McCall's classic model	Alternative factor models	
			Evans and Marciniak 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			+

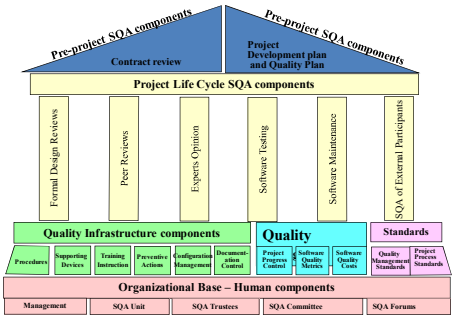
Models -> System

Question

How do we translate these SQA models into an SQA system?

The Software Quality Shrine

Galin Chapter 4 pp. 59



The SQA system

- Goal is to minimize the number of software errors and to achieve an acceptable level of software quality
- Can be divided into to six classes:
  - Pre-project components
  - Components of project life cycle activities assessment
  - Components of infrastructure error prevention and improvement
  - Components of software quality management
  - Components of standardization, certification, and SQA system assessment
  - Organizing for SQA-the human components

Summary

- Software Quality is important
  - Software controls life-and-death decisions, has enormous economic consequences, affects reputations, etc.
- An independent Quality Management System is vital component of an effort to produce quality software
  - Quality needs a "stakeholder"
- Think about quality requirements and a quality process.
- Quality assurance is a product of effective risk management
  - Tailor project management and software engineering practices to mitigate quality-related risks
- Measurement program is necessary
  - Demonstrate progress toward quality objectives using metrics, testing and other measurements

Software standards

- Standards define the required attributes of a product or process.
- They play an important role in quality management.
- Standards may be international, national, organizational or project standards.

83

Importance of standards

- Encapsulation of best practice - avoids repetition of past mistakes.
- They are a framework for defining what quality means in a particular setting i.e. that organization's view of quality.
- They provide continuity - new staff can understand the organisation by understanding the standards that are used.

84

Product and process standards

- **Product standards**
  - Apply to the software product being developed.
- **Process standards**
  - These define the processes that should be followed during software development.

85

Product and process standards

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of new code for system building
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

86

Benefits

- The use of standards has many potential benefits for any organization
  - **Improved management** of software
    - Schedules and budgets are more likely to be met
    - Quality goals are likely to be reached
    - Employee training and turnover can be managed
  - Visible **certification** can attract new customers or be required by existing ones
  - **Partnerships** and co-development, particularly in a global environment, are enhanced

87

More business benefits

- **Regulation**
  - –Cost effective compliance
  - –Customer assurance
  - –Reduce product liability
  - –Risk management
  - –Governance
- **Cost Optimization**
  - –Reduced transaction costs
  - –Product/process interoperability
  - –Flexibility in supply chain
  - –Best practice & management systems
- **Maximizing Revenue**
  - –Improve speed to market
  - –Product acceptance
  - –Product life cycle management
- **Business Opportunities**
  - –Develop new markets & future sales
  - –Influence technology change
  - –Influence industry evolution
  - –Structure regional/international competition

88

Importance of standards

- Encapsulation of best practice
  - – avoids repetition of past mistakes
- Framework for quality assurance process
  - – it involves checking standard compliance
- Provide continuity
  - – new staff can understand the organisation by the standards applied

89

Problems with standards

- There is evidence that the majority of small software organizations are not adopting existing standards as they perceive them as being orientated towards large organizations.
- Studies have shown that small firms' negative perceptions of process model standards are primarily driven by negative views of cost, documentation and bureaucracy
- it has been reported that VSEs find it difficult to relate standards to their business needs and to justify the application of the international standards in their operations

90

Who is the ISO?

International Organization for Standardization

- International Organization for Standardization is the world's largest developer of International Standards
- ISO is a network of the national standards institutes of 162 countries
  - one member per country
  - **one vote per country**
- ISO is a non-governmental organization that forms a bridge between the public and private sectors
  - Many of its member institutes are part of the governmental structure of their countries, or are mandated by their government
  - Other members have their roots uniquely in the private sector, having been set up by national partnerships of industry associations
- This enables ISO to reach a consensus on solutions that meet both the requirements of business and the broader needs of society

91

Who develops ISO standards

- ISO standards are developed by technical **committees**, (or subcommittees) comprising experts from the industrial, technical and business sectors
- These **experts** may be joined by representatives of government agencies, consumer associations, non-governmental organizations and academic circles, etc.
- Experts participate as national delegations, chosen by the ISO national member body for the **country** concerned.

92

How ISO standards are developed

- The national delegations of experts of a committee meet to discuss, debate and argue until they reach consensus on a draft agreement
- The resulting document is circulated as a Draft International Standard (DIS) to all ISO's member bodies for voting and comment
- If the voting is in favor, the document, with eventual modifications, is circulated to the ISO members as a Final Draft International Standard (FDIS)

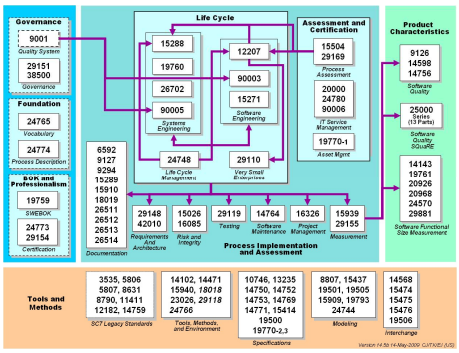
93

What ISO/IEC Standards are available?

- There are a large collection of standards covering a range of domains
- For example:
  - ISO 9126 for the evaluation of software quality
  - ISO 20926 a functional size measurement method
  - ISO 26513 for testers and reviewers of user documentation

94

JTC 1 SC7 Standards Collection



95

ISO 9000 Series

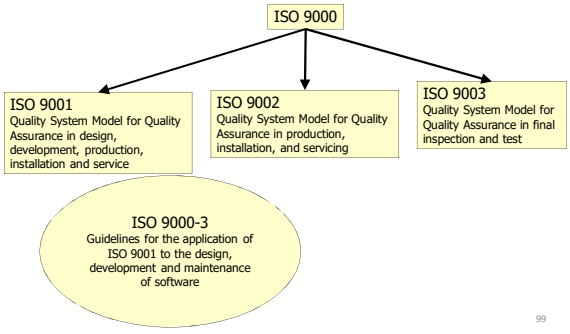
96

ISO 9000 Philosophy

- Document what you do
  - in conformance with the requirements of the applicable standard
- Do what you document
- Record what you did
- Prove it
  - maintenance of registration requires audits every three years, with mini-audits every six months

97

ISO 9000 Structure



99

Quality management

- ISO 9001 is for **quality management**.
- **Quality** refers to all those features of a product (or service) which are required by the customer.
- **Quality management** means what the organization does to
  - ensure that its products or services satisfy the customer's **quality requirements** and
  - comply with any **regulations** applicable to those products or services.
- Quality management also means what the organization does to
  - enhance **customer satisfaction**, and
  - achieve **continual improvement** of its performance

100

Generic standard

- ISO 9001 is a generic standard
- Generic means that the same standards can be applied:
  - to **any organization**, large or small, whatever its product or service,
  - In **any sector** of activity, and
  - whether it is a business enterprise, a public administration, or a government department.
- Generic also signifies that signifies that
  - no matter what the organization's scope of activity
  - if it wants to establish a **quality management system**, ISO 9001 gives the essential features

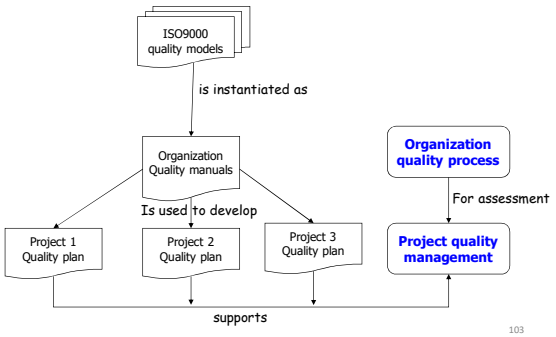
101

Processes, not products

- ISO 9001 concerns **the way an organization goes about its work**
  - Its not a product standard
  - Its not a service standard
  - It's a **process** standard
- It can be used by product manufacturers and service providers.
- Processes affect final products or services.
- **ISO 9001** gives the requirements for what the organization must do to manage **processes affecting quality** of its products and services

102

ISO 9000 and Quality Management



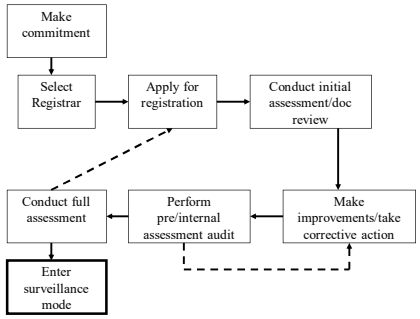
103

Certification and registration

- **Certification** is known in some countries as **registration**.
- It means that an **independent, external body** has audited an organization's management system and verified that it conforms to the requirements specified in the standard (ISO 9001 or ISO 14001).
- **ISO does not carry out certification** and does not issue or approve certificates,

104

Certification Process



105

Certification not a requirement

- **Certification is not a requirement of ISO 9001**
- The organization can implement and benefit from an ISO 9001 system without having it certified
- The organization can implement them for the **internal benefits** without spending money on a certification programme

106

Certification is a business decision

- Certification is **a decision to be taken for business reasons**:
  - if it is a contractual, regulatory, or market requirement,
  - If it meets customer preferences
  - it is part of a risk management programme, or
  - if it will motivate staff by setting a clear goal.

107

ISO does not certify

- **ISO does not carry out ISO 9001 certification**
- ISO does not issue certificates
- ISO does not accredit, approve or control the certification bodies
- ISO develops **standards and guides to encourage good practice** in accreditation and certification

108

Is ISO 9000 Worth it?

Opponents claim that it is only for documentation. Proponents believe that if a company has documented its quality systems, then most of the paperwork has already been completed (Barnes, 2000)

- A common criticism of ISO 9000 is the amount of money, time and paperwork required for registration
- ISO 9000 can be difficult to apply in a software environment
- Unlike CMMI, ISO 9000 does not provide a road map for improvement beyond the adherence to quality management documents
- ISO 9000 is primarily applied in the software domain because of its market credibility
- Some argue it negates the advantages accruing to small software firms

109

ISO 15504 and 330XX Series

110

Remember this?

- Remember we already covered SPICE…
- ISO/IEC **15504 / 330XX**
- **SPICE** (Software Process Improvement and Capability Determination)
  - Is a **framework for the assessment of processes**
  - **PRM – Process reference model,**
  - **PAM Process assessment method**

111

Key points

- Software quality management is concerned with ensuring that software has a low number of defects and that it reaches the required standards of maintainability, reliability, portability etc.
  - Software standards are important for quality assurance as they represent an identification of 'best practice'.
  - When developing software, standards provide a solid foundation for building good quality software.
- Reviews of the software process deliverables involve a team of people who check that quality standards are being followed. Reviews are the most widely used technique for assessing quality.
- Agile quality management relies on establishing a quality culture where the development team works together to improve software quality.

115