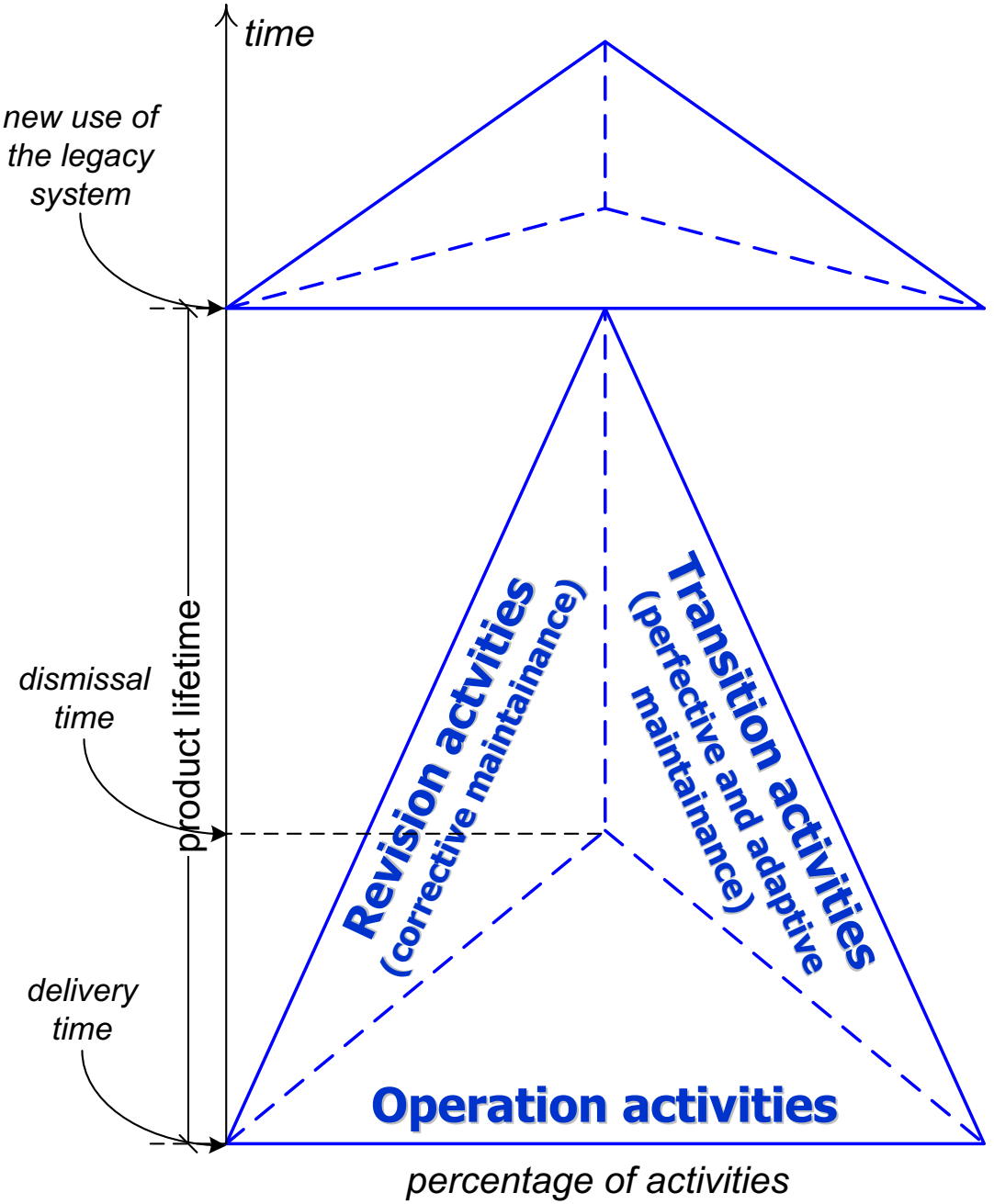
Software Quality

- IEEE Definition (Std 1061-1998, *Standard for a Software Quality Metrics Methodology*):

  Software quality is the degree to which software possesses a desired combination of attributes

- Viewpoints:
  - Transcendent
    - innate excellence
  - User
    - fit for use (meet user's needs), enjoy to use, user satisfaction
  - Product
    - desired attributes of software (reliability, correctness, etc.)
    - conformance to requirements
  - Organization
    - costs and profits (increased efficiency, increased effectiveness, added value to organization, marketable product)

The Quality Triangle (McCall quality model)



Quality Indices

i_6 maintainability
i_7 flexibility
i_8 testability

Revision actvities
(corrective maintainance)

i_9 portability
i_{10} reusability
i_{11} interoperability
i_{12} evolubility

Transition activities
(perfective and adaptive maintainance)

Operation activities

i_1 correctness
i_2 reliability
i_3 efficiency
i_4 integrity
i_5 usability

Quality Indices (Operation)

- **i_1 Correctness**
  - extent to which a product satisfies spec & fulfills users' objectives (does it do what I want?)
- **i_2 Reliability**
  - extent to which a product can be expected to perform its intended function with required precision (does it do it accurately all of the time?)
- **i_3 Efficiency**
  - amount of computing resources and code required by a product to perform a function (will it run on my hardware as well as it can?)
- **i_4 Integrity**
  - extent to which access to software or data by unauthorized persons can be controlled (is it secure?)
- **i_5 Usability**
  - effort required to learn, operate, prepare input, and interpret output of a product (can I run it?)

Quality Indices (Revision)

i_6 Maintainability

- effort required to locate and fix an error in an operational program (can I fix it?)

i_7 Testability

- effort required to test a product to ensure that it performs its intended function (can I test it?)

i_8 Flexibility

- effort required to modify an operational product (can I change it?)

Quality Indices (Transition)

i_9 Portability

- effort required to transfer a product from one hardware and/or software environment to another (will I be able to use it on another machine?)

i_{10} Reusability

- extent to which a product (or parts thereof) can be reused in other applications (will I be able to reuse some of the software?)

i_{11} Interoperability

- effort required to couple one product with another (will I be able to interface it with another system?)

i_{12} Evolubility

- effort required to update the product in order to fulfill new requirements (is it easy to update when requirements change?)

Software Quality Indices and Attributes

- Software quality is defined in terms of quality indices

$$q = (i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}, i_{11}, i_{12})$$

- Each quality index can be in turn defined in terms of quality attributes

$$i_j = (a_1, a_2, ..., a_n)$$

- Each attribute can be in turn defined in terms of sub-attributes, which are measured in order to obtain the attribute value
- Attribute values contribute to quantify the relevant indices
- The evaluation of quality indices allows to verify the system ability to meet the desired levels of quality (for the indices of interest)

Quality Attributes

a_1 Complexity

level of understandability and verifiability of elements of the software and their interactions.

a_2 Accuracy

precision of computations and output

a_3 Completeness

full implementation of the required functionalities

a_4 Consistency

use of uniform design and implementation techniques and notations

a_5 Error tolerance

continuity of operation ensured under adverse conditions

a_6 Traceability

degree to which a relationship can be established between two or more products of the development process

a_7 Expandability

storage or functions can be expanded

a_8 Generality

breadth of potential applications

a_9 Modularity

provisions of highly independent modules

a_{10} Auto-documentation

in-line docs

Software Quality Indices and Attributes

(+/- denotes positive/negative impact)

i_1 Correctness

+ Completeness
+ Consistency
+ Traceability

i_7 Flexibility

+ Traceability
+ Consistency
- Complexity
+ Modularity
+ Generality
+ Auto-documentation

i_2 Reliability

+ Error Tolerance
+ Consistency
+ Accuracy
- Complexity

i_{12} Evolubility

+ Consistency
- Complexity
+ Modularity
+ Expandability
+ Generality
+ Auto-documentation

$$i_1 = (a_3, a_4, a_6)$$

$$i_7 = (a_1, a_4, a_6, a_8, a_9, a_{10})$$

$$i_2 = (a_1, a_2, a_4, a_5)$$

$$i_{12} = (a_1, a_4, a_7, a_8, a_9, a_{10})$$

Checklist Method

- The quality level of each attribute is evaluated by use of techniques based on *checklists*
- The checklist result is elaborated according by appropriate algorithms to obtain a normalised score, indicating the quality level of each attribute
- Once the score of each attribute has been evaluated, the relevant values are grouped for each attribute, summarised and normalised in order to obtain the quality level for each index. The attribute has to be considered with its own sign (positive or negative)

Checklist evaluation

- A checklist is composed by several questions, and for each question four answers are foreseen with a given value
- The calculation will not take into account questions identified as *Not Applicable*
- If the question is *Not Valuable*, then the score assigned to that question is the lowest among the possible ones
- The *checklist evaluation team* is composed by at least four people, with different skills so that various points of view of can be compared and the final evaluation is the most precise one
- The recommended composition of the team is:
  - One Quality Assurance Specialist
  - One Project Leader
  - One System Engineer
  - One Software Analyst

Calculation of the score of attributes

- For each attribute (*except for Modularity*) the score calculation is performed according the following formula:

$$V_{attribute} = \frac{\sum_{i=1}^{\#\ questions} V_{answer_i}}{\sum_{i=1}^{\#\ questions} max(V_{answer_i})}$$

- where V_{answer_i} is the value given to the answer chosen for question *i*, and $max(V_{answer_i})$ is its maximum value

Evaluation of the quality index

- Using the results of calculation of attributes score and the impact of each attribute on the relevant index it is possible to calculate the quality level of the index as follows:

$$V_{index} = \frac{\sum V_{attribute(+)} + \sum \left(1 - V_{attribute(-)}\right)}{number\ of\ attr.\ associated\ to\ index}$$

- where $V_{attribute(+)}$ and $V_{attribute(-)}$ are the scores of the attributes impacting positively and negatively on the index, respectively

Acceptance Level Scale

- The output of the formula yielding V_{index} is a value between 0 and 1
- The following scheme can be used in order to identify threshold values for quality acceptance:

V_{index}	*Quality Level*
$0.66 < V \leq 1$	High
$0.33 < V \leq 0.66$	Medium
$0 < V \leq 0.33$	Low

Example checklists

- Let assume the quality level of a *system architecture* is to be evaluated (at the architectural design phase)
- Example checklists for scoring the following attributes are illustrated in the next slides:

  a1 Complexity

  a8 Generality

  a9 Modularity

Complexity checklist

General aspects

1) Are modules, procedures and structure names significant or conform to a standard, if any.

   0 : Yes, almost always

   1 : Quite often

   2 : Rarely

   3 : No

2) Is the formalism utilised to describe the system architecture only one, or multiple ones are present.

   0 : Unique formalism, standard and properly chosen

   1 : Few formalisms, standard and properly chosen

   2 : Few formalisms and someone out of standard

   3 : A lot of formalisms, someone out of standard

3) Is the global system design properly structured and easy understandable by people without specific knowledge about the system.

   0 : Yes

   1 : Almost positive

   2 : Not properly structured

   3 : Bad structured, and hardly understandable

Complexity checklist (2)

4) Is it possible to deduce the class of information of each single data present into the logical model. Are all the utilisation of this data declared in the documentation and realised with the purpose to read or write that class of information.

    0 : No

    1 : Rarely

    2 : Quite often

    3 : Approximately always.

Metrics applications

5) If is possible to perform measurements about the program complexity, based on the program calling graph, then use the metrics below defined.

    5.1) *Hierarchical* complexity: is the average number of modules per calling-tree level, that is the total number of modules divided by the number of tree levels

        0 : 0 up to 4
        1 : 4 up to 8
        2 : 8 up to 12
        3 : Less than 2 or greater than 12

    5.2) *Structural* complexity: is the average number of call for each module, that is the number of inter-module calls divided by the number modules.

        0 : Less than 2
        1 : 2 up to 4
        2 : 4 up to 8
        3 : Greater than 8

Generality checklist

1) Are all functions offered by modules (and their interfaces) properly planned so that is possible to reuse them in some implementations that wasn't planned at the start time of the project. (i.e. : a specific percentage [18%], or any percentage of any value)

    0 : No
    1 : Rarely
    2 : Quite often
    3 : Always

2) Concerning modules that haven't enough generality, is it possible to make them more generic with low effort (near 10% of global effort to obtain each of them).

    0 : No
    1 : Rarely
    2 : Quite often
    3 : Always

3) Are all the data structure manipulations combined into modules specifically dedicated to this.

    0 : No
    1 : Rarely
    2 : Quite often
    3 : Always

Modularity checklist

Cohesion

1) Which is the percentage distribution of system modules according the following four types:
   - 1A modules with coincidental cohesion
   - 1B modules with logical or temporal cohesion
   - 1C modules with procedural or communicational cohesion
   - 1D modules with informational or functional cohesion

Coupling

2) Which is the percentage distribution of communication modes (coupling) between modules according to the following four types:
   - 2A pairs of modules with content coupling
   - 2B pairs of modules with common coupling
   - 2C pairs of modules with control coupling
   - 2D pairs of modules with stamp or data coupling

Modularity checklist (2)

General

3) Is the system decomposition organised in a hierarchical way

    0 : No

    1 : Rarely

    2 : Yes, enough

    3 : Yes

4) Which is the percentage of hierarchical structures (each program contains a comparable number of modules)

    0 : less then 70%

    1 : within 70% and 80%

    2 : within 80% and 90%

    3 : more than 90%

Score for cohesion and coupling questions

$$V_{answer_1} = \frac{(0 \times \%1A) + (1 \times \%1B) + (2 \times \%1C) + (3 \times \%1D)}{50}$$

$$V_{answer_2} = \frac{(0 \times \%2A) + (1 \times \%2B) + (2 \times \%2C) + (3 \times \%2D)}{50}$$

where *%X* is the percentage of modules (for answer 1) or pairs_of_modules (for answer 2) of type *X* with respect to the whole set of modules/pairs_of_modules of the system

$$V_{Modularity} = \frac{V_{answer_1} + V_{answer_2} + V_{answer_3} + V_{answer4}}{\max(V_{answer_1}) + \max(V_{answer_2}) + \max(V_{answer_3}) + \max(V_{answer_4})}$$

Procedure for attribute scoring

a) The documentation must be deeply analysed in order to get a sufficiently precise answer to each question contained in the check list. Each question foresees four possible answers and every one of them is associated to a score.

b) Each reviewer belonging to the evaluation team must answer the questions of the checklist independently. The reviewer must not know the answer of the other ones before.

   Questions identified as *Not Applicable* must be explicitly marked in this way.

   When a question has been identified as *Not Valuable*, it is analysed in more detail to verify if it is applicable or not. If at the end of the analysis the question is considered applicable, then the lowest score among the possible ones is assigned to the question itself. This event also gives an indication of non conformity of the examined documentation.

c) When all members of the evaluation team give different answers to a given question, it is necessary they reach a common agreed answer.

d) At the end of the discussion a sole official checklist will be completed with the answers agreed upon by the evaluation team members.

e) Calculate the score of the sub-attribute according the calculation formula

Template for attribute scoring

Attribute:			
Question #	N/A	Not Valuable	Score

Total Score (see formula on slide 12)

Template for index evaluation

Index:		
Attribute	Weight (+/-)	Score
Evaluation (see formula on slide 13)		
Quality level (see acceptance scale on slide 14)		

Software Quality Assurance

- Software quality Assurance (SQA) is a planned and systematic approach to ensure that both software process and software product conform to the established standards, processes, and procedures
- The goals of SQA are to improve software quality by monitoring both software and the development process to ensure full compliance with the established standards and procedures
- Steps to establish an SQA program
  - Get the top management's agreement on its goal and support (without management support, SQA can not be effective)
  - Identify SQA issues, write SQA plan, establish standards and SQA procedures, implement the SQA plan and evaluate SQA program

SQA role

- The role of SQA is to give management the assurance that the officially established process is actually being implemented
- SQA ensures that:
  - An appropriate development methodology is in place
  - The projects use standards and procedures in their work
  - Reviews and audits are conducted
  - Documents are produced to support maintenance and enhancement
  - Software configuration management is set up to control change.
  - Testing is performed and passed
  - Deficiencies and deviations are identified, documented and brought to management's attention

SQA goals

- The goals of SQA is to reduce the risks by:
  - Appropriately monitoring the software and the development process
  - Ensuring full compliance with standards and procedures for software and process
  - Ensuring that inadequacies in product, process, or standards are brought to management's attention so that they can be fixed
- *SQA is not responsible for producing quality products*. It is responsible for auditing (i.e., looking at the product in depth, comparing it with established standards and procedures) the quality actions and for alerting management to any deviations

SQA standards and procedures

- *Standards* are the criteria to which software products are compared (*i.e.*, standards define what should be done)
  - Minimum requirement for standards include:
    - ***Documentation Standards:*** specify form and content for planning, control, and product documentation and provide consistency throughout a project
    - ***Design Standards:*** specify the form and content of the design product. They provide rules for translating the software requirements into the software design and for representing it in the design documentation
    - ***Code Standards:*** specify the language in which the code is to be written and define any restrictions on use of language features. They define legal language structure, style conventions, rules for data structures, and internal code documentation
- *Procedures* are the criteria to which development and control processes are compared (*i.e.*, procedures define how the work is actually to be done, by whom, when and what is done with the results)

SQA plan

- The SQA plan specifies SQA *goals*, *tasks* to be performed, and the *standards* and *procedures* against which the development work is to be measured
- IEEE standard for SQA plan preparation (730-1998) includes:
  - Management
  - Documentation
  - Standards, Practices, and Conventions
  - Reviews and Audits
  - Software Configuration Management
  - Problem Reporting and Corrective Action
  - Tools, Techniques, and Methodologies
  - Code Control
  - Media Control
  - Supplier Control
  - Records Collection, Maintenance, and Retention