ISO standards
ISO 12207, ISO 15504 & ISO 9126

ISACA – CETIC Meeting

23 May 2007

Introduction

Process standards

- ISO 12207 = common framework for the lifecycle of the software
  - Architecture of the software lifecycle processes (processes, activities, tasks)
- ISO 15504 also known as SPICE (Software Process Improvement and Capability Determination) = "framework for the assessment of software processes"
  - Derived from 12207 and CMMI

Introduction (2)

Product standard

- ISO 9126 = set of characteristics to describe software product quality
  - Internal, external and use-related features
  - Each characteristic = subcharacteristics + metric to assess conformance with requirements

ISO 12207
Software lifecycle processes

Agenda

1. Context and Purpose
2. Scope
3. History
4. Basic concepts

1. Context and Purpose

- Domain : software engineering
- Focus : software lifecycle processes
- Purpose : to establish a common framework for the life cycle of software

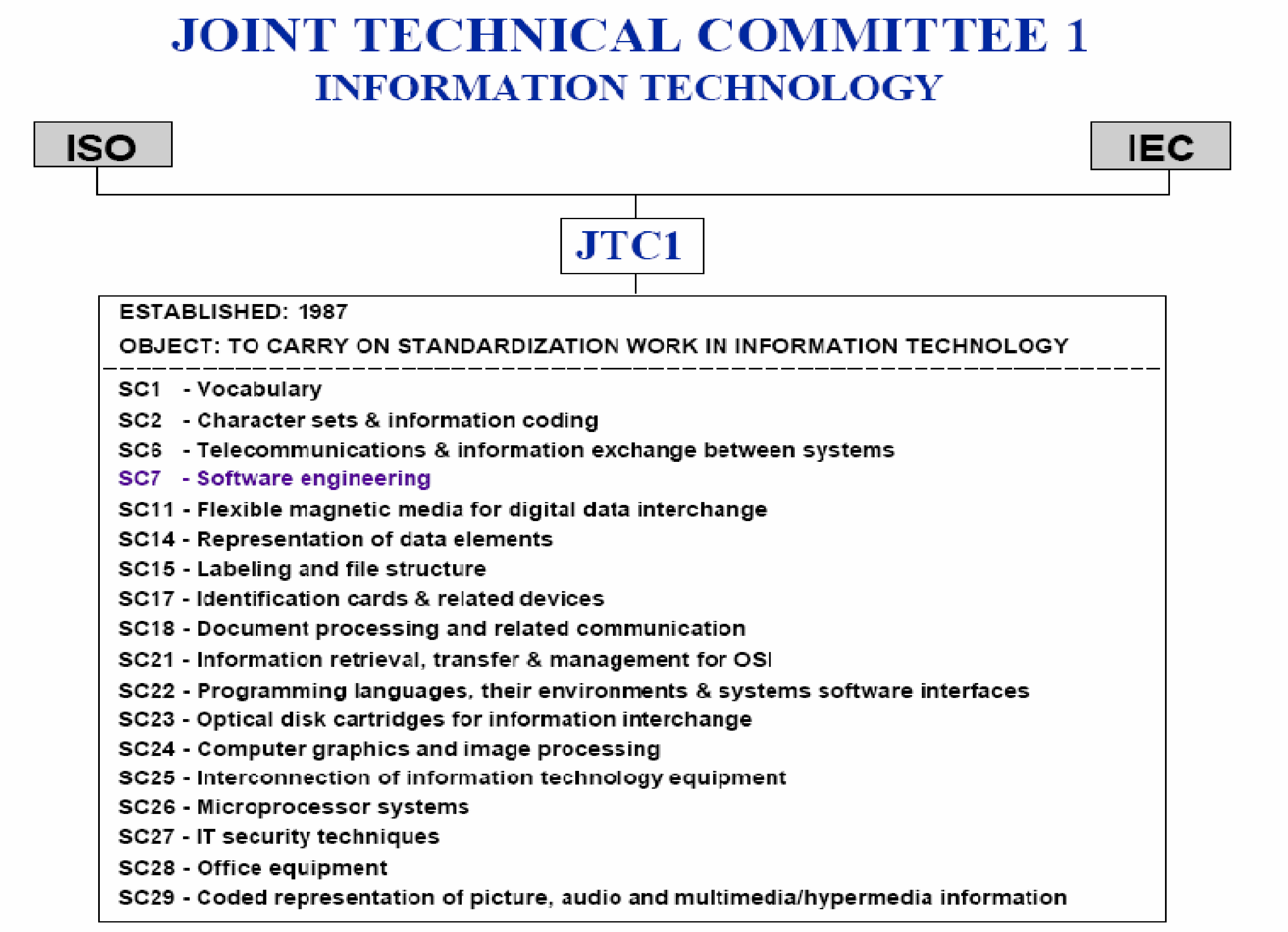
  → to foster mutual understanding among business parties

  → to acquire, supply, develop, operate and maintain software

2. Scope

- Stakeholders: acquirers, suppliers, users etc
- Application: corporate processes related to project products and project services
- ISO 12207 covers process definitions and descriptions

3. History



JOINT TECHNICAL COMMITTEE 1
INFORMATION TECHNOLOGY

ESTABLISHED: 1987

OBJECT: TO CARRY ON STANDARDIZATION WORK IN INFORMATION TECHNOLOGY

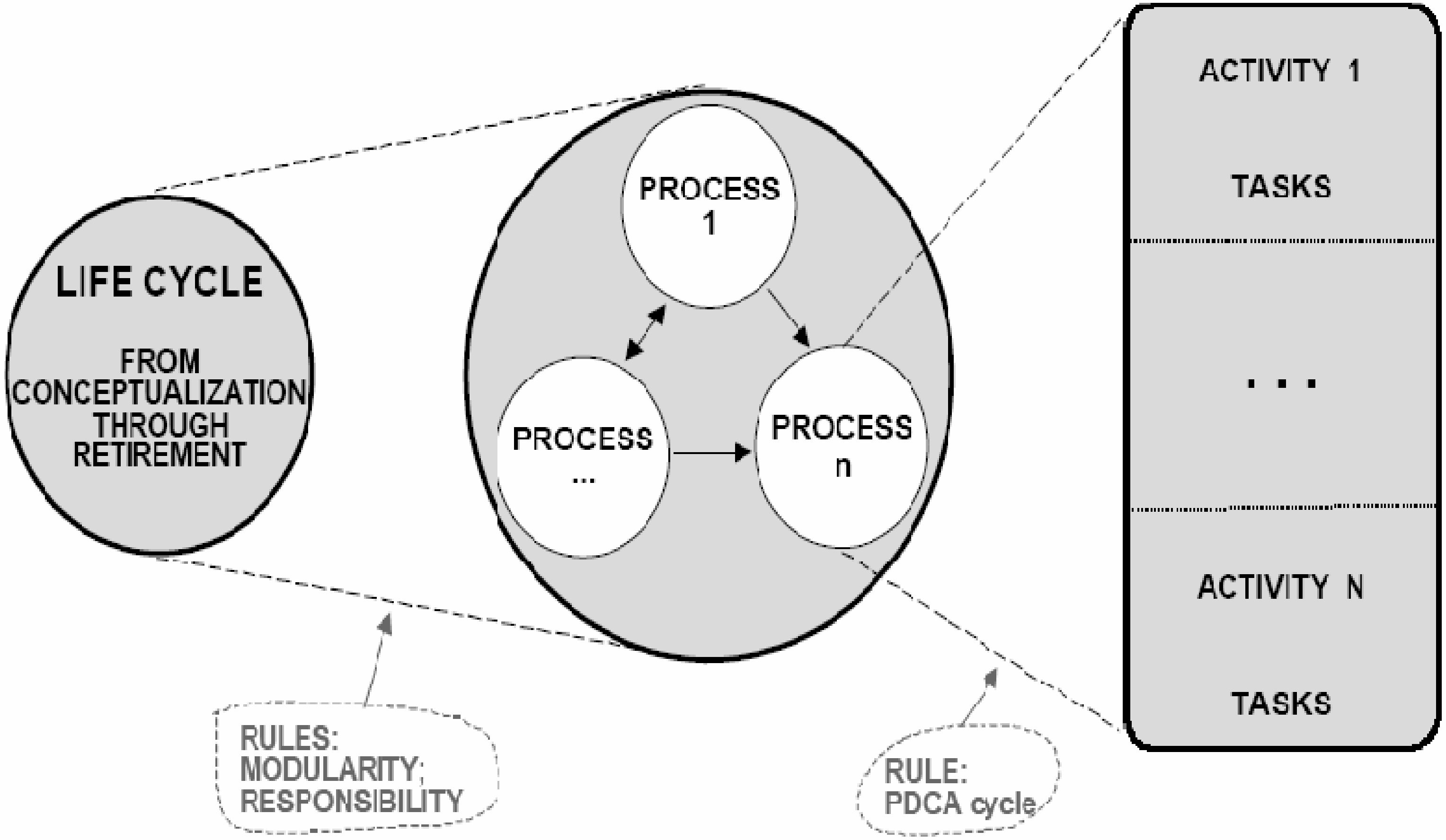
SC1 - Vocabulary
SC2 - Character sets & information coding
SC6 - Telecommunications & information exchange between systems
SC7 - Software engineering
SC11 - Flexible magnetic media for digital data interchange
SC14 - Representation of data elements
SC15 - Labeling and file structure
SC17 - Identification cards & related devices
SC18 - Document processing and related communication
SC21 - Information retrieval, transfer & management for OSI
SC22 - Programming languages, their environments & systems software interfaces
SC23 - Optical disk cartridges for information interchange
SC24 - Computer graphics and image processing
SC25 - Interconnection of information technology equipment
SC26 - Microprocessor systems
SC27 - IT security techniques
SC28 - Office equipment
SC29 - Coded representation of picture, audio and multimedia/hypermedia information

3. History (2)

- ISO/IEC 12207 **Sponsor** :
  - Joint Technical Committe 1 (JTC1) (Information Technology) of International Organization for Standardization (ISO) and International Electrotechnical Commission 7 (IEC).
  - Developer: Subcommittee 7 (SC7) (Software Engineering)
- Proposed in June 1988
- Published 1 August 1995
- Participants: Australia, Canada, Denmark, Finland, France, Germany, Ireland, Italy, Japan, Korea, Netherlands, Spain, Sweden, UK, USA

4. Basic Concepts – Life cycle and architecture

- THE ARCHITECTURING OF THE LIFE CYCLE:



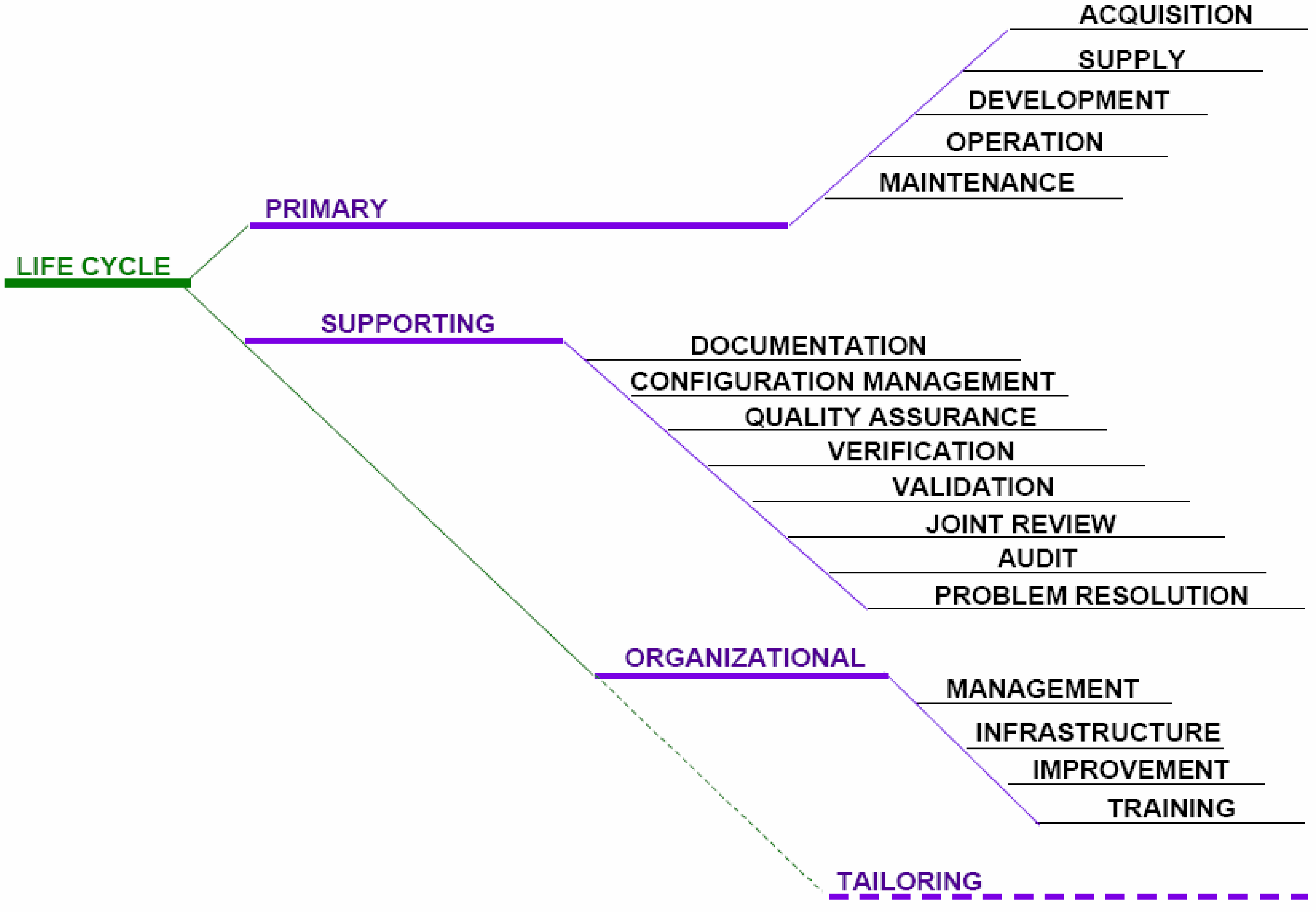
ISO 12207

4. Basic Concepts – Rules for partitioning the life cycle

- Modularity
  - Cohesion (Functional): Tasks in a process must be functionally related
  - Coupling (Internal): Links between processes must be minimal
- Association
  - If a function is used by more than one process, then the function becomes a process in itself
  - If Process X is invoked by Process A and Process A only, then Process X belongs to Process A
- Responsibility
  - Each process is under a responsibility
  - A function with parts under different responsibilities shall not be a process

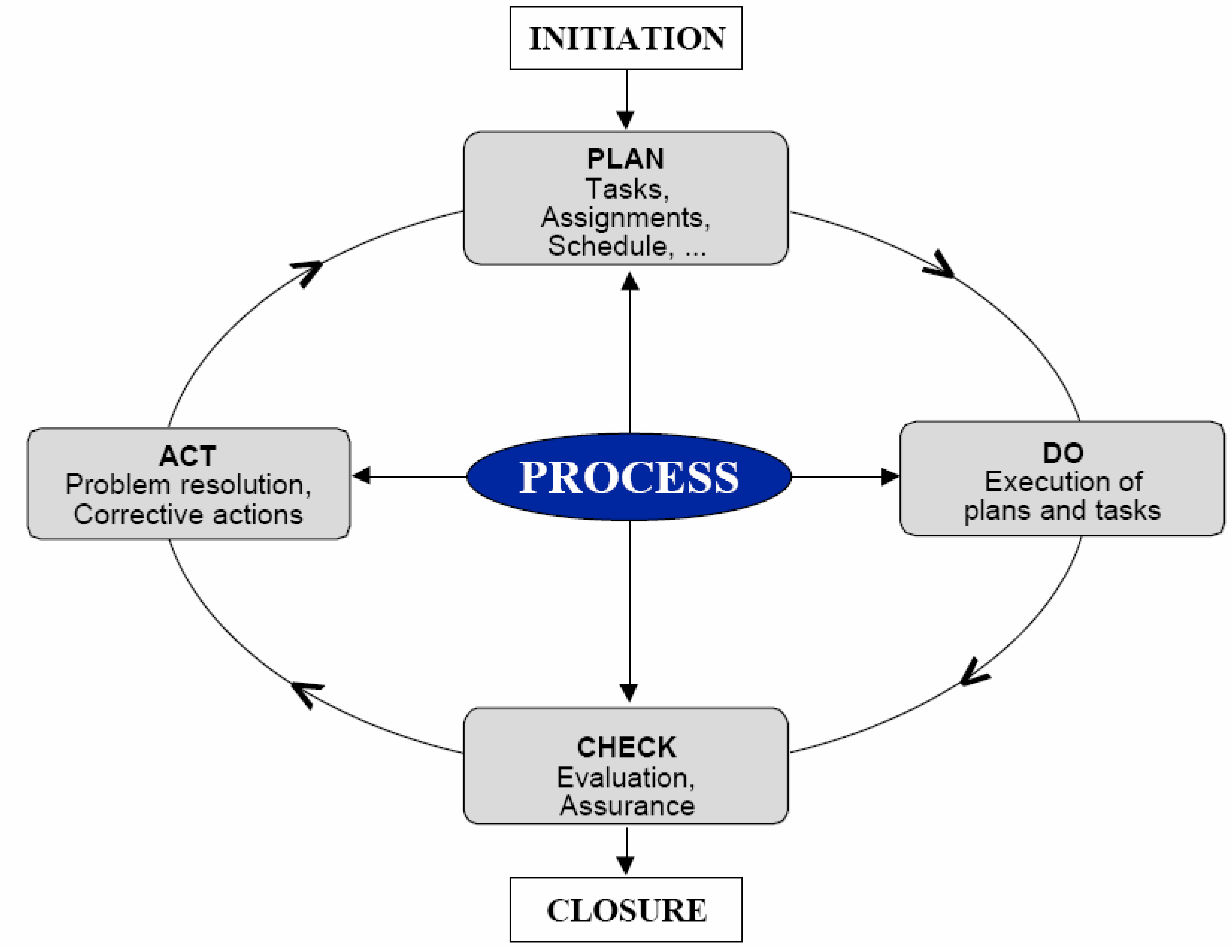
4. Basic Concepts – The Process Tree

THE PROCESS TREE



4. Basic Concepts – Rules for partitioning a process

- A process is partitioned into PDCA activities based on the PDCA-cycle principles



4. Basic Concepts – Activity and Tasks

- An activity is divided into tasks, which are grouped into similar actions

- Based on TQM Principles
  - Each party/participant has appropriate responsibility

4. Basic Concepts – What 12207 is not

- Not certifying
- Not prescriptive, no how-tos
- Not a standard for methods, techniques & models
  - does not prescribe management and engineering methods
  - does not prescribe computer languages
  - Etc
- Not a standard for metrics
  - many tasks need metrics and indicators
  - but prescribes no specific metrics/indicators
  - references ISO/IEC 9126 for guidance

ISO 15504 (SPICE)
Software Quality

Agenda

1. Context and Purpose
2. History
3. Basic concepts
4. CETIC products derived from ISO 15504

1. Context and Purpose

- Normalized structure devoted to managing requirements related to a software development process

- Model for process management + set of requirements/guidelines to assess/improve those processes

2. History

- Early 1990's: process improvement and capability determination methods developed in several countries
  - → International consensus on the urgent need for a public domain standar for software process assessment

- June 1991 in London, Joint Technical Committee 1/Sub-Committee 7 of the ISO/IEC: resolution to develop an international standard on software process assessment

3. Basic concepts - Process

- 5 process categories
  - Customer-Provider
    - Acquisition process (process for selectiong provider)
    - Process for support to customer
  - Engineering
    - Process for analyzing requirements and designing the system
  - Support
    - Documentation process
  - Management
    - Risk management process
  - Organization
    - Process for managing human resources

3. Basic concepts - Process

- 6 maturity levels for assessing the processes
  - 5 : optimizing
  - 4 : quantitatively managed
  - 3 : defined
  - 2 : managed
  - 1 : initial
  - 0 : incomplete

- To assess a process, we define it as follows:
  - Purpose/goal
  - Results/attributes that should be met to reach a successful implementation of the process

ISO 15504

3. Basic concepts - Process and maturity levels (2)

Example : process for software testing(1/2)

- Purpose: to test the integrated software
- Result of a successful implementation of the process
  - Acceptance criteria are developed in order to verify compliance with requirements
  - The integrated software is verified using the defined acceptance criteria
  - The testing results are taken in
  - A non-regression strategy is established in order to test the integrated software again if software is modified
  - The regression testing is performed when necessary

3. Basic concepts - Assessing each process

- For each attribute:
  - N = not implemented ⇔ 0 % → 15 %
  - P = partly implemented ⇔ 16 % → 50 %
  - L = largely implemented ⇔ 51 % → 85 %
  - F = fully implemented ⇔ 86 % → 100 %

- A level is achieved if
  - The attribute(s) of this level = **L** or **F**
  - Attributes of lower levels = F

3. Basic concepts - Example of process assessment

		Requirements analysis	design	building	testing	Quality assurance	configuration management
level 3	PA3.2	P	P	N	P	L	P
	PA3.1	P	P	N	P	P	P
level 2	PA2.2	L	L	P	L	L	P
	PA2.1	L	L	L	P	P	N
level 1	PA1.1	L	F	L	P	P	L
=>achieved level		1	2	1	0	0	1

ISO 15504

3. Basic concepts - Conclusion

SPICE is very interesting to prepare an improvement plan

- Can be applied to the way a team works
- Gives the opportunity to deploy progressively the action plan:
  - By targeting first and foremost the most critical **processes**
  - By targeting the levels in ascending order
    - On a mid-term: target = **level 2**
    - On a long term: target = **level 3**

4. CETIC products - OWPL
(Observatoire Wallon des Pratiques Logicielles)

- Model based on CMM and SPICE (ISO 15504)
- Adapted to SMO's
- **goal:** improve software production processes

4. CETIC products - OWPL (2)

- model structure:
  - 10 processes (each split up in practices):
    - ➢ requirements management,
    - ➢ project planification,
    - ➢ project follow-up,
    - ➢ development,
    - ➢ documentation,
    - ➢ test,
    - ➢ configuration management,
    - ➢ outsourcing management,
    - ➢ quality managemenet,
    - ➢ process for capitalizing knowledge
  - Success factors organized in 4 categories:
    - ➢ organization within the processes take place,
    - ➢ the management policy,
    - ➢ the human resources
    - ➢ the « used » technical tools

4. CETIC products - OWPL (3)

- Success story: PEPITe
  - CETIC has assessed the PEPITo software with OWPL
  - Goal: inform PEPITe about their software development practices to improve them

    → **Improve their products and services**
  - CETIC has provided a complete assessment report + recommendations to improve their development practices

4. CETIC products - NOEMI

- based on existing standards such as ISO/IEC15504
- The NOEMI assessment method has been developed by Centre HENRI TUDOR (Luxemburg).
- Two goals:
  - improve the perception of computer maturity in SMO's or VSMO's
  - methodological tool for improving those companies' SI

4. CETIC products - NOEMI (2)

- Assessment according to an exhaustive list of the typical computer activities in SMO’s/VSMO’s divided in 5 fields:
  - infrastructure
  - support
  - management
  - security
  - documentation

4. CETIC products - NOEMI (3)

- Success Story: GREISCH (Liège), Architects office
  - Interviews conducted with 3 types of users:
    - One responsible within the computer department
    - The director of the computer department
    - 3 end-users (architects)
  - CETIC has provided GREISCH with an assessment report on their practices within the computer department and the quality of the services/products delivered to the end-users (the architects) by the computer scientists
  - CETIC has also provided recommendations to improve their products and services

ISO 9126
Software Product Quality

Agenda

1. Scope
2. History
3. Basic concepts
4. CETIC products derived from ISO 9126

1. Scope

- **ISO 9126** is an international standard for the evaluation of software.
- It will be overseen by the project SQuaRE, ISO 25000:2005, which follows the same general concepts
- four parts:
  - quality model;
  - external metrics;
  - internal metrics;
  - and quality in use metrics.

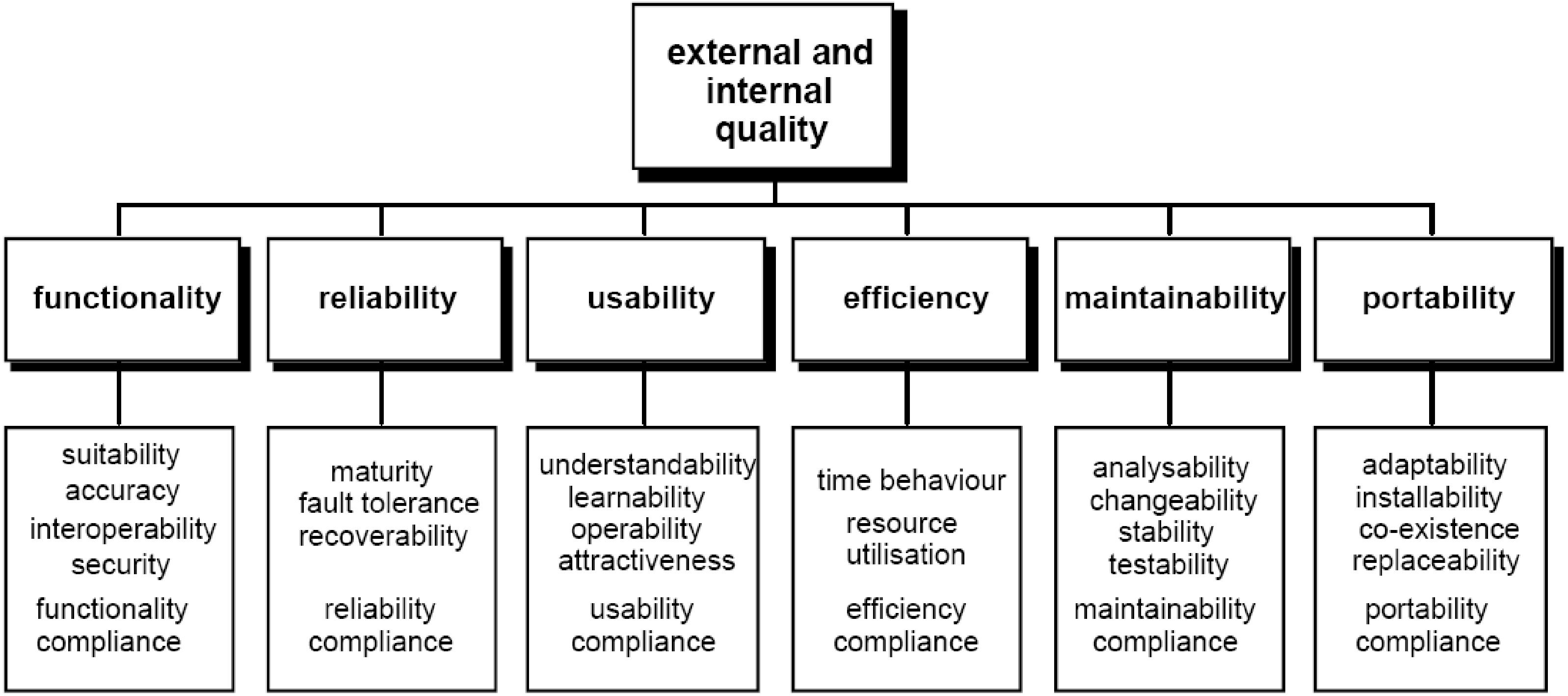
2. History

- Late 1980's: need for a framework assessing the quality of a software product
- 1991: Joint Technical Committee of the ISO/IEC develops ISO 9126
- Standard revised in 2001
- Will be overseen by SQuaRE (ISO 25000:2005)

3. Basic concepts – First part

- **The quality model established in the first part of the standard, ISO 9126-1, classifies software quality in a structured set of characteristics and sub-characteristics as follows:**

3. Basic concepts – First part



ISO 9126

3. Basic concepts – First part (2)

- **Functionality** - *A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.*
- **Reliability** - *A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.*
- **Usability** - *A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.*

ISO 9126

3. Basic concepts – First part (3)

- **Efficiency** - *A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.*

- **Maintainability** - *A set of attributes that bear on the effort needed to make specified modifications.*

- **Portability** - *A set of attributes that bear on the ability of software to be transferred from one environment to another.*

3. Basic concepts – First part (4)

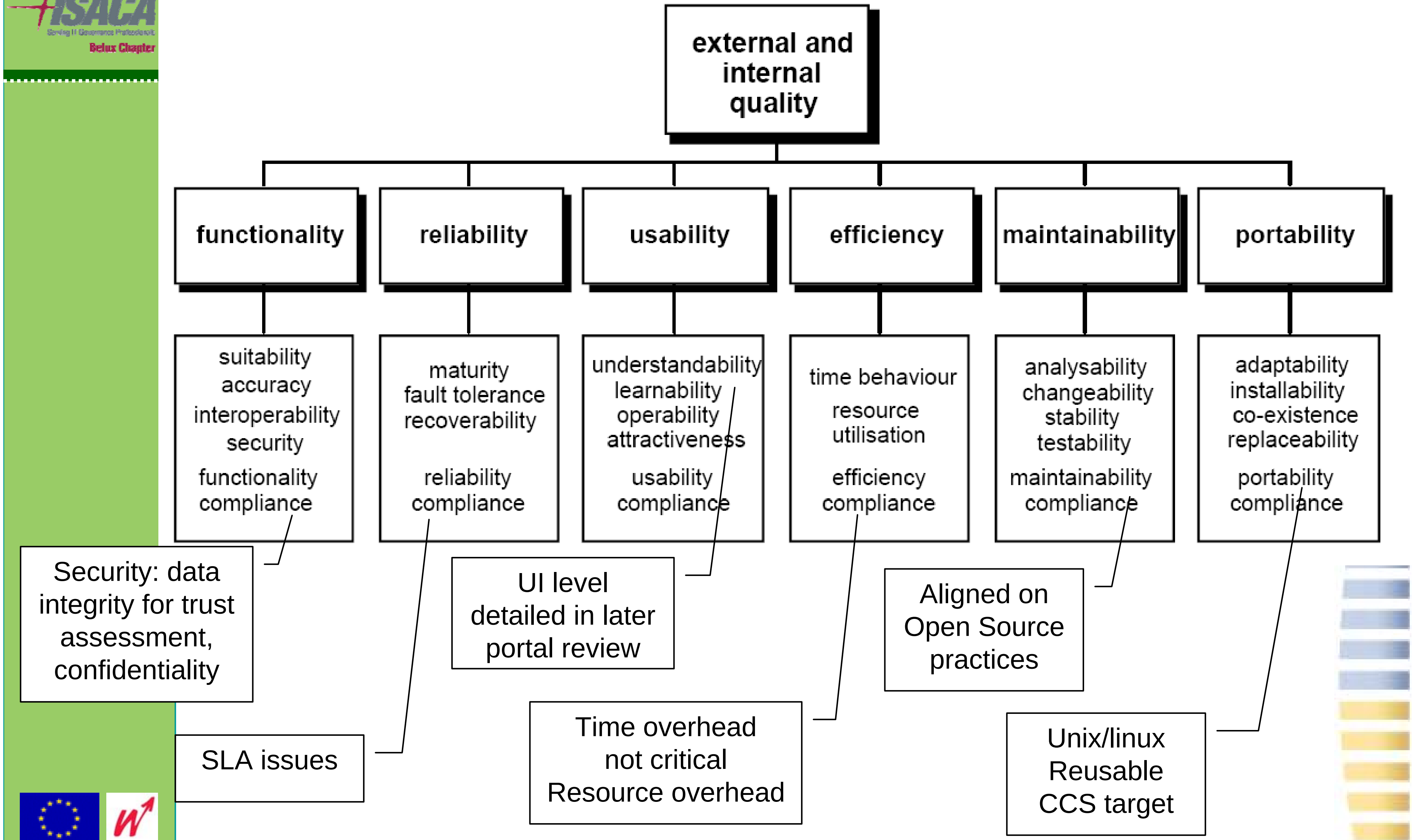
- Each quality sub-characteristic (as adaptability) is further divided into attributes.

- An attribute is an entity which can be verified or measured in the software product.

- Attributes are not defined in the standard, as they vary between different software products.

3. Basic concepts – First part (5)

AssessGrid - Non functional using requirements: ISO-9126



external and internal quality

functionality
- suitability
- accuracy
- interoperability
- security
- functionality compliance

reliability
- maturity
- fault tolerance
- recoverability
- reliability compliance

usability
- understandability
- learnability
- operability
- attractiveness
- usability compliance

efficiency
- time behaviour
- resource utilisation
- efficiency compliance

maintainability
- analysability
- changeability
- stability
- testability
- maintainability compliance

portability
- adaptability
- installability
- co-existence
- replaceability
- portability compliance

Security: data integrity for trust assessment, confidentiality

SLA issues

UI level detailed in later portal review

Time overhead not critical Resource overhead

Aligned on Open Source practices

Unix/linux Reusable CCS target

3. Basic concepts - Description of the standard

- Internal metrics are those which do not rely on software execution (static measures).
- External metrics are applicable to running software.
- Quality in use metrics are only available when the final product is used in real conditions
- Ideally, the internal quality determines the external quality and external quality determines quality in use.

3. Basic concepts – Internal metric

Metric Name: Data corruption prevention

Purpose: how complete is the implementation of data corruption prevention

Method of application: Count the number of implemented instances of data corruption prevention as specified and compare with the number of instances of operations/access specified in requirements as capable of currption/destroying data

Measurement, formula and data element computations: X=A/B with A= number of implemented instances of data corruption prevention as specified confirmed in review and B = Number of instances of operation/access identified in requirements as capable of corruption/destroying data Note: consider security levels when using this metric

Interpretation of measured value: 0<=X<=1 with the closer to 1, the more complete

Metric scale type: absolute

Measure type: X=count/count

A = count

B = count

Input to measurement : Requirement specification, Design, Source code, Review report

3. Basic concepts – External metric

Metric name: maintainability compliance

Purpose of the metric: how compliant is the maintainability of the product to be applicable regulations, standards and conventions

Method of application: count the number of items requireing compliance that have been met and compare with the number of items reuquiring compliance in the specification

Measurement, formula and data element computations: X = 1-A/B with A=Number of maintainability compliance items specified that have not been implemented during testing and B = Total number of maintainability compliance items specified

Interpretation of measured value: 0<=X<=1 The closer to 1.0 is the better

Metric scale type: absolute

Measure type: A = count, B=count and X=count/count

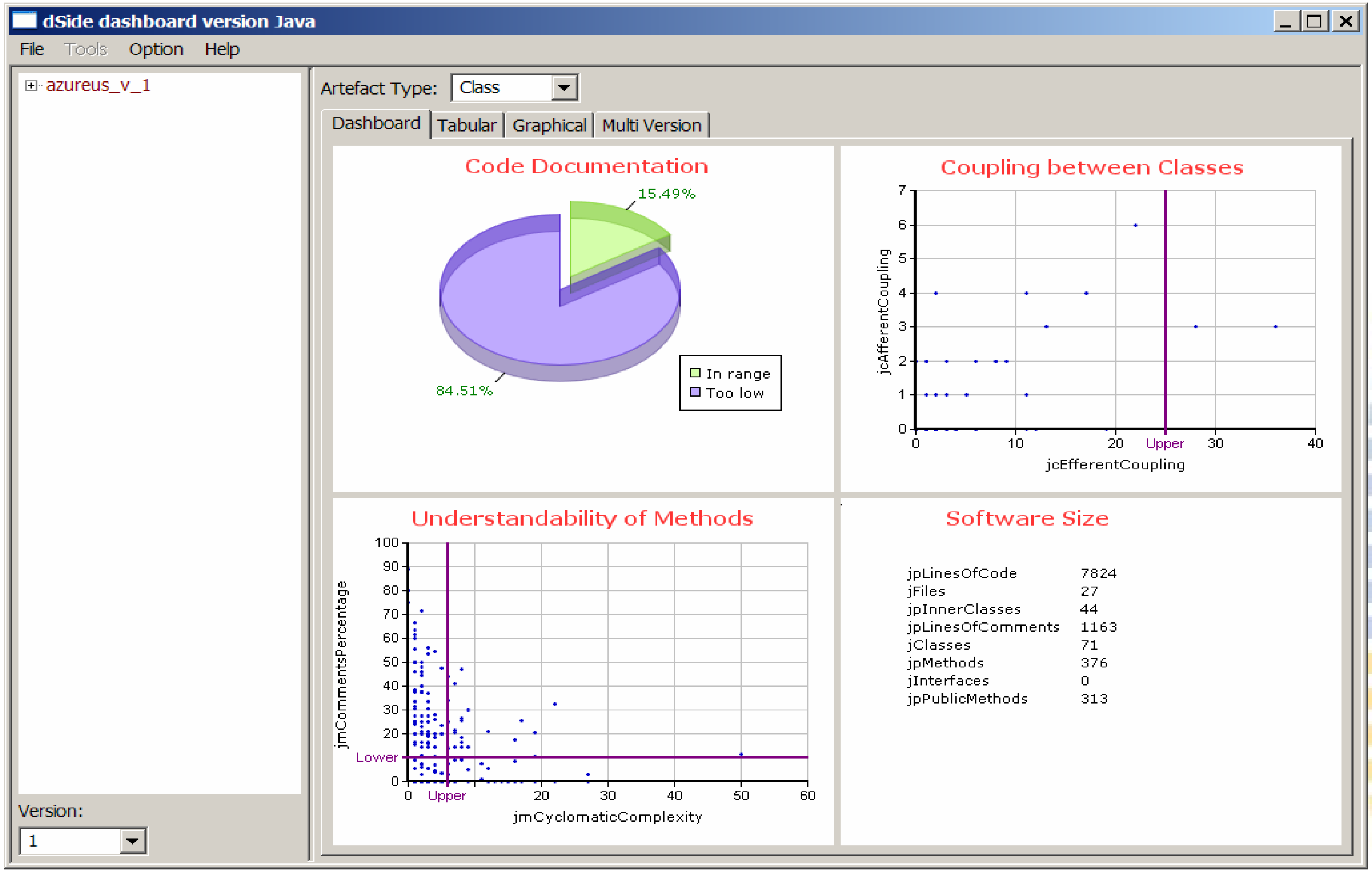
Input to measurement: product description (user manual or Specification) of complance and related standards, conventions or regulations. Test specification and report

Target audience: supplier, user

4. CETIC product - D-SIDE Dashboard

- CETIC has developed a measurement software tool in the framework of research in software quality

  → **D-SIDE Dashboard**



Artefact Type: Class

```
jpLinesOfCode        7824
jFiles               27
jpInnerClasses       44
jpLinesOfComments    1163
jClasses             71
jpMethods            376
jInterfaces          0
jpPublicMethods      313
```

Version: 1

4. CETIC product - D-SIDE Dashboard
Frequent questions by Project Leader

- Where should we concentrate the testing effort?
- Which classes are used the most?
- Which classes are error-prone?
- Which classes/methods are difficult to understand/test/maintain?
- Which classes are impacted when a modification occurs, what do we have to test again?
- Which classes are difficult to debug?

4. CETIC product - D-SIDE Dashboard

Most used metrics

- Comments rate:
  - Classes/Methods
- Afferent and efferent coupling
  - Classes/Methods
- Cyclomatic complexity
  - Classes/Methods
- Depth of Inheritance
  - Classes
- Number of Children
  - Classes

ISO 9126

4. CETIC product - D-SIDE Dashboard

Benefits from D-SIDE Dashboard

- Rapid graphical identification of abnormal code in order to target
  - Unit tests
  - Code reviews
- Quick overview of an application
  - Commented?
  - Modular?
  - volume?

4. CETIC product - D-SIDE Dashboard
Benefits from D-SIDE Dashboard (2)

- Definition of quality models, according to (for example):
  - The application type (framework, GUI, etc.)
  - The sector
- Definition of new metrics
- Plug-in Architecture allowing to add:
  - New parsers (other languages)
  - External measurers