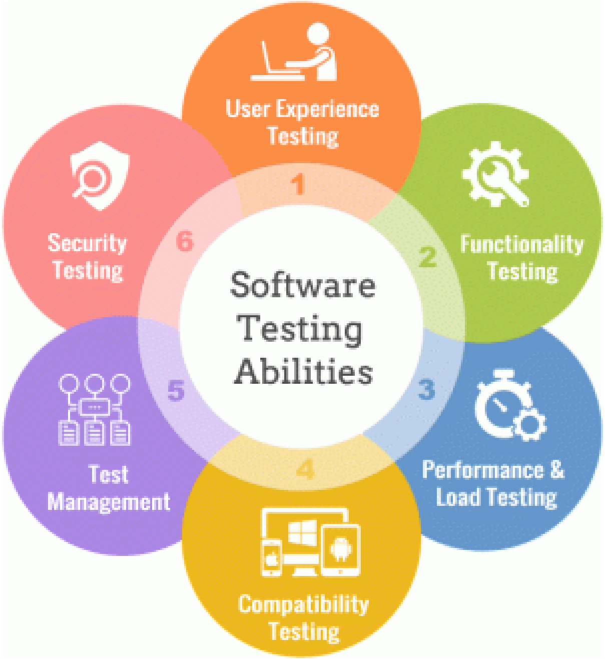
Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Security by Design

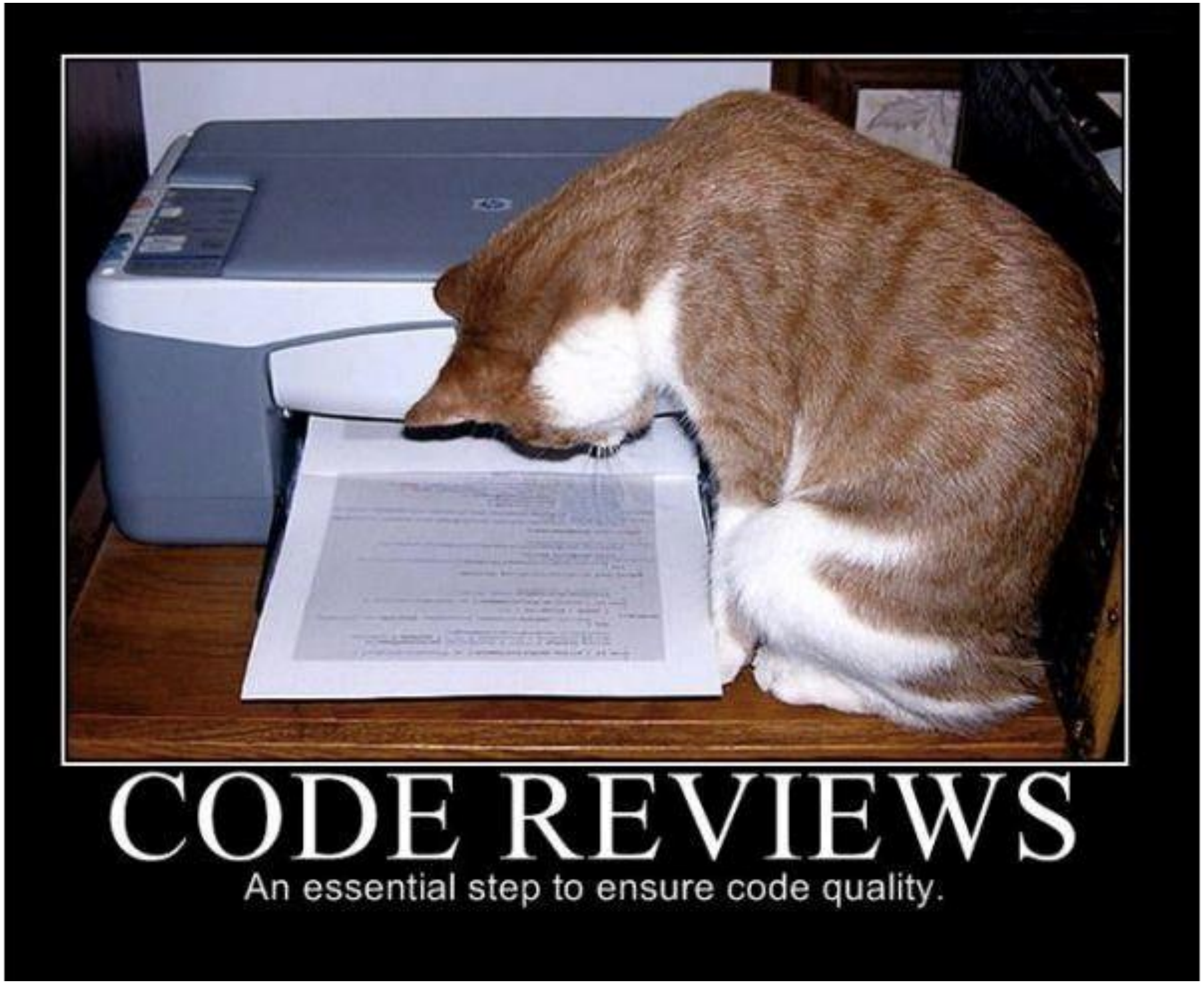
Quality Assurance

SUPSI DTI
Angelo Consoli
2025 - 2026

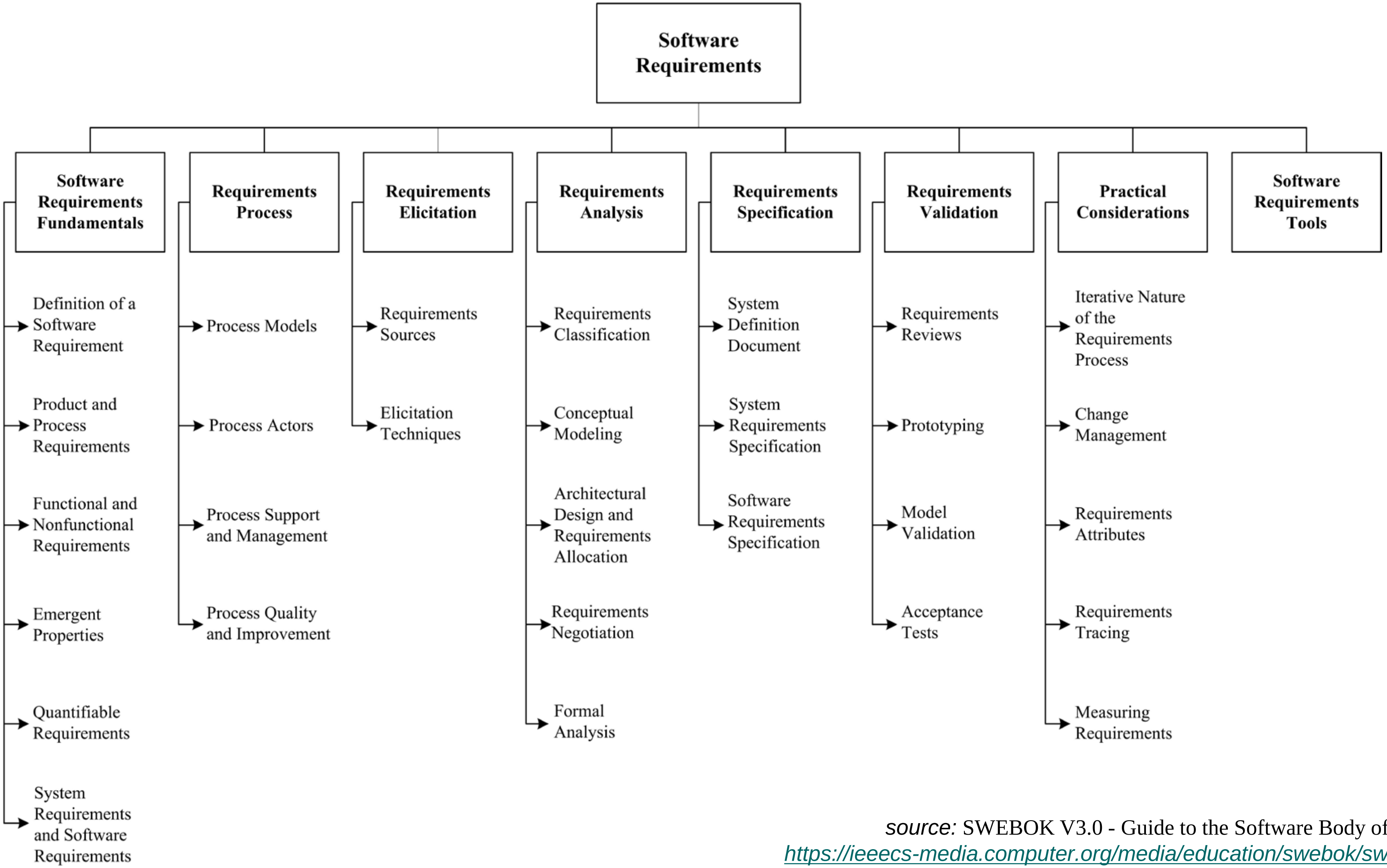


PART ONE - SOFTWARE QUALITY ASSURANCE (SQA) STANDARDS

SQA is an important step toward better software development



SQA - Breakdown of Topics for the Improved Software Requirements



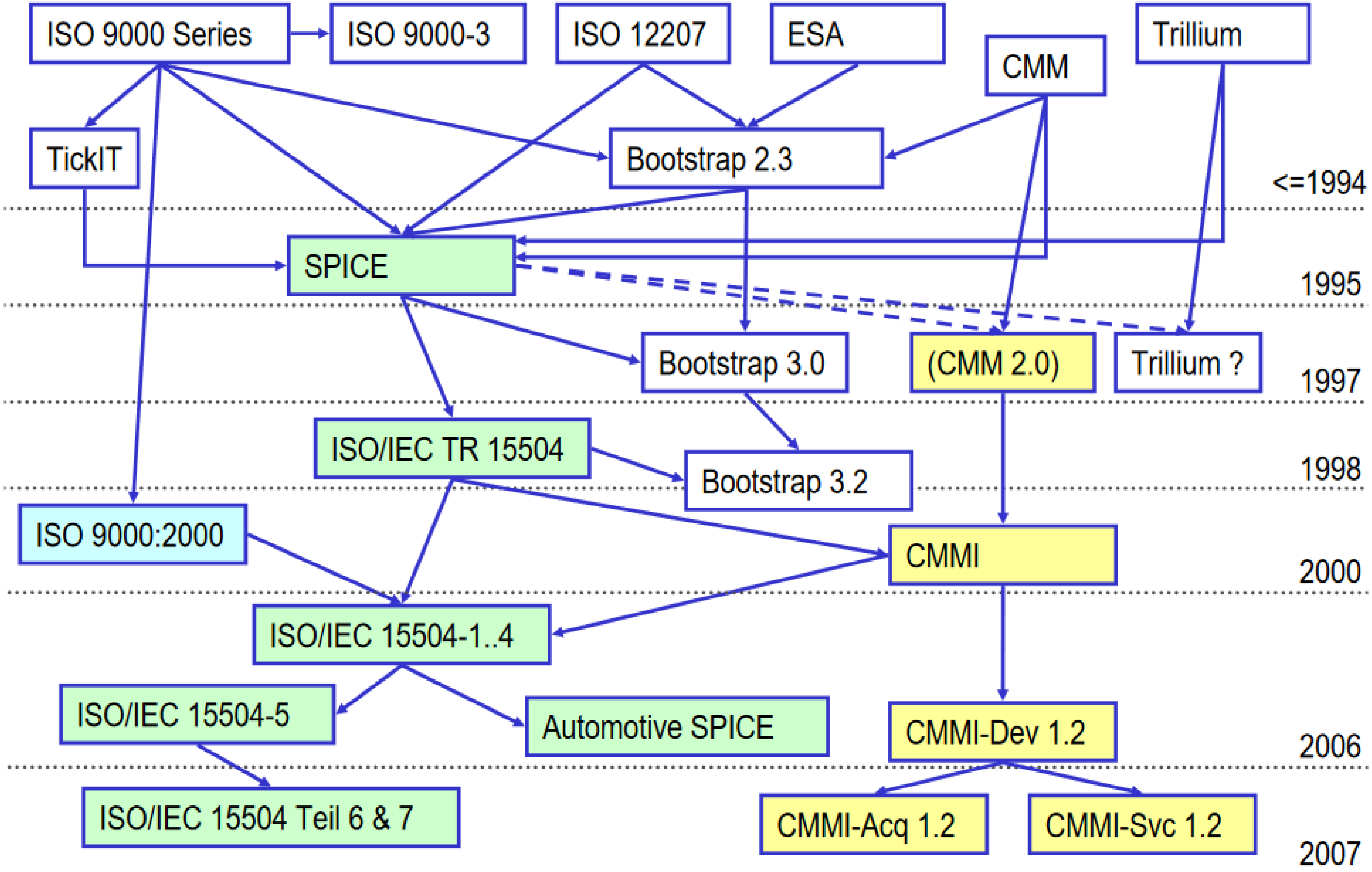
- **Software Requirements**
  - **Software Requirements Fundamentals**
    - Definition of a Software Requirement
    - Product and Process Requirements
    - Functional and Nonfunctional Requirements
    - Emergent Properties
    - Quantifiable Requirements
    - System Requirements and Software Requirements
  - **Requirements Process**
    - Process Models
    - Process Actors
    - Process Support and Management
    - Process Quality and Improvement
  - **Requirements Elicitation**
    - Requirements Sources
    - Elicitation Techniques
  - **Requirements Analysis**
    - Requirements Classification
    - Conceptual Modeling
    - Architectural Design and Requirements Allocation
    - Requirements Negotiation
    - Formal Analysis
  - **Requirements Specification**
    - System Definition Document
    - System Requirements Specification
    - Software Requirements Specification
  - **Requirements Validation**
    - Requirements Reviews
    - Prototyping
    - Model Validation
    - Acceptance Tests
  - **Practical Considerations**
    - Iterative Nature of the Requirements Process
    - Change Management
    - Requirements Attributes
    - Requirements Tracing
    - Measuring Requirements
  - **Software Requirements Tools**

source: SWEBOK V3.0 - Guide to the Software Body of Knowledge;
https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf

Some base-standards

- ISO/IEC 9126 - ISO/IEC 25010 ; ISO/IEC 25010
- ISO/IEC 15504
- Capability Maturity Model Integration CMMI
- And many others …

Some baseline-standards - Overview



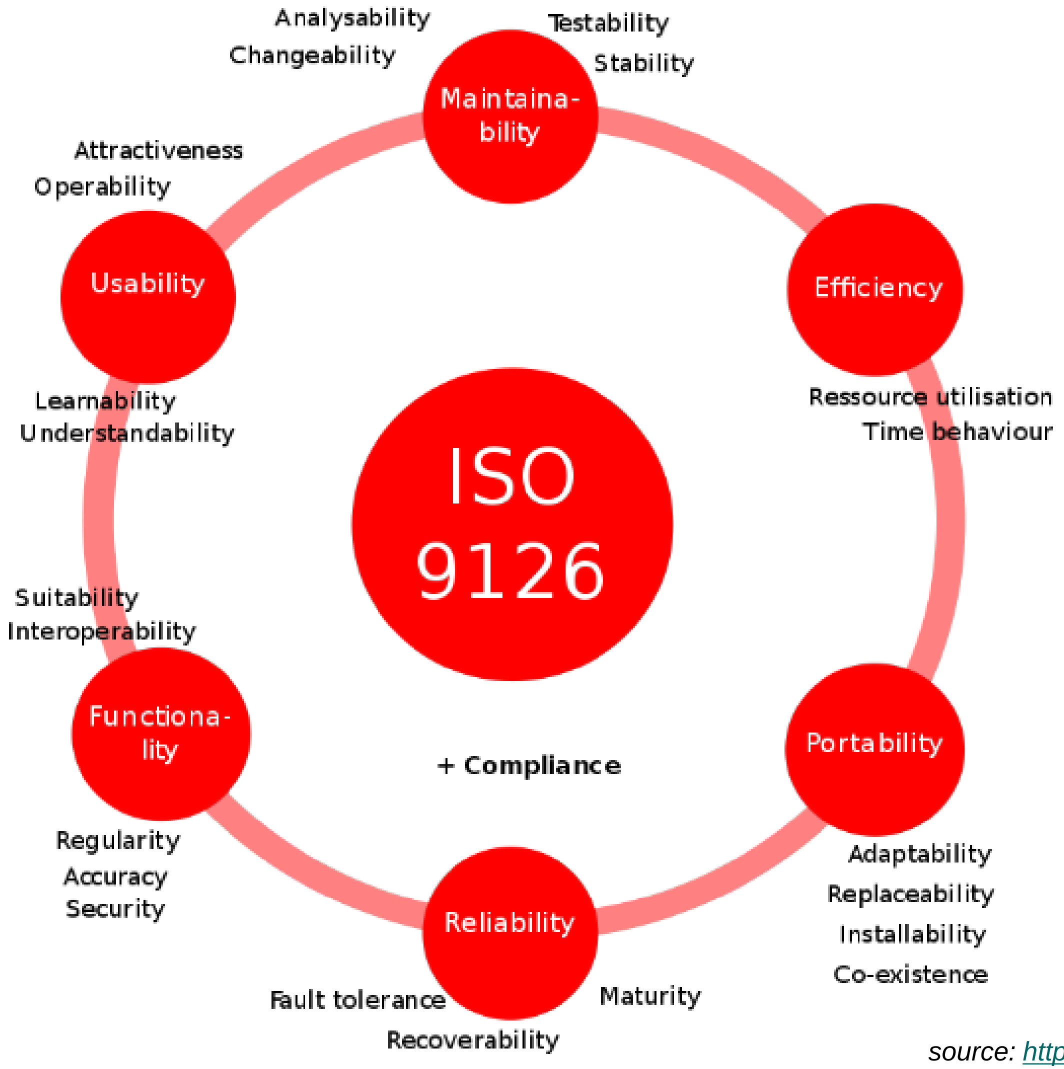
source: https://www.flecsim.de/images/download/fs-ueberblickspicecmmi.pdf

ISO/IEC 9126

Software engineering - Product quality is an international standard dictated by the ISO/IEC (International Organization for Standardization/International Electrotechnical Commission (ITU-T M 3000)).

The concept of compliance revolves around 6 main characteristics each characteristic is then divided in more detailed sub-characteristics.

ISO/IEC 9126 Quality Criteria



source: *https://en.wikipedia.org/wiki/ISO/IEC_9126*

ISO/IEC 9126 - Characteristics

Characteristic	**Sub-Characteristic**	**Example of practical questions**
Functionality	Suitability (F1)	Can software perform the tasks required?
	Accurateness (F2)	Is the result as expected?
	Interoperability (F3)	Can the system interact with another system?
	Compliance (F4)	Is the system compliant with standards?
	Security (F5)	Does the system prevent unauthorized access?
Usability	Understandability (U1)	Does the user comprehend how to use the system easily?
	Learnability (U2)	Can the user learn to use the system easily?
	Operability (U3)	Can the user use the system without much effort?
	Attractiveness (U4)	Does the interface look good?
Maintainability	Analyzability (M1)	Can faults be easily diagnosed?
	Changeability (M2)	Can the software be easily modified?
	Stability (M3)	Can the software continue functioning if changes are made?
	Testability (M4)	Can the software be tested easily?

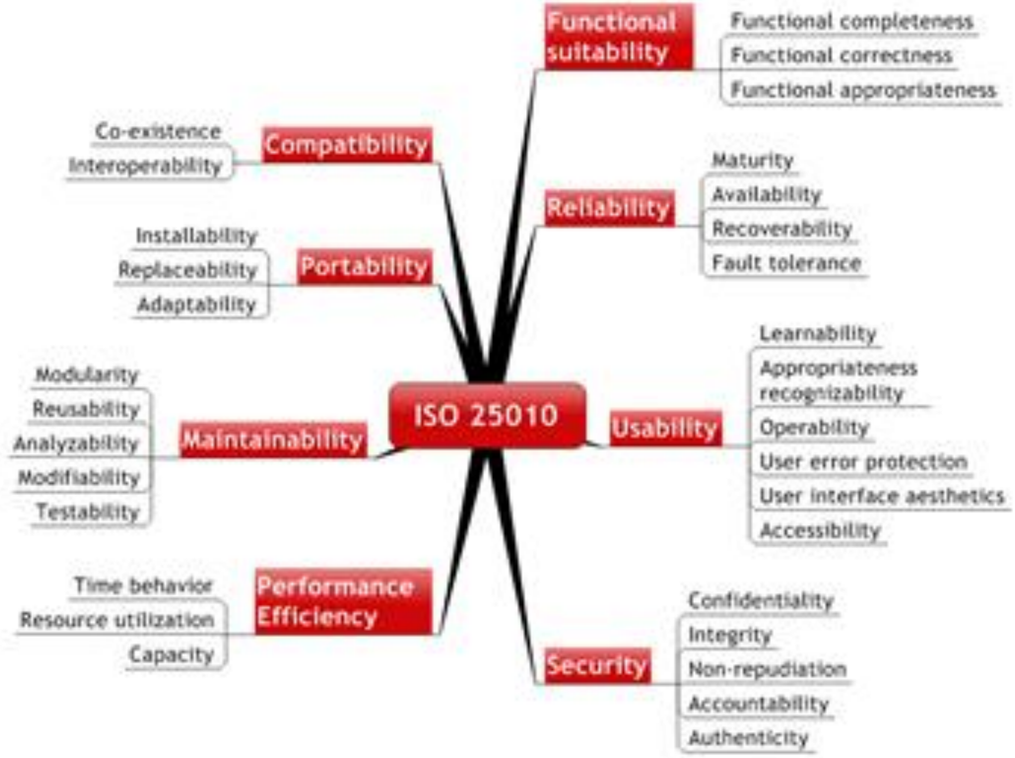
ISO/IEC 9126 - Characteristics

Characteristic	Sub-Characteristic	Example of practical questions
Reliability	Maturity (R1) Fault tolerance (R2) Recoverability (R3)	Have most of the faults in the software been eliminated over time? Is the software capable of handling errors? Can the software resume working & restore lost data after failure?
Efficiency	Time Behaviour (E1) Resource utilization (E2)	How quickly does the system respond? Does the system utilize resources efficiently?
Portability	Adaptability (P1) Installability (P2) Conformance (P3) Replaceability (P4)	Can the software be moved to other environments? Can the software be installed easily? Does the software comply with portability standards? Can the software easily replace other software?

ISO/IEC 9126 to ISO/IEC 25010

ISO/IEC 9126 was surpassed in 2011 by ISO/IEC 25010 which modify and adds a set of new characteristics to evaluate software, making it 8 in in total as listed below:

1. Portability
2. Maintainability
3. **Security**
4. Reliability
5. Usability
6. **Compatibility**
7. **Functional suitability**
8. **Performance efficiency**



ISO/IEC 9126 - Interesting readings

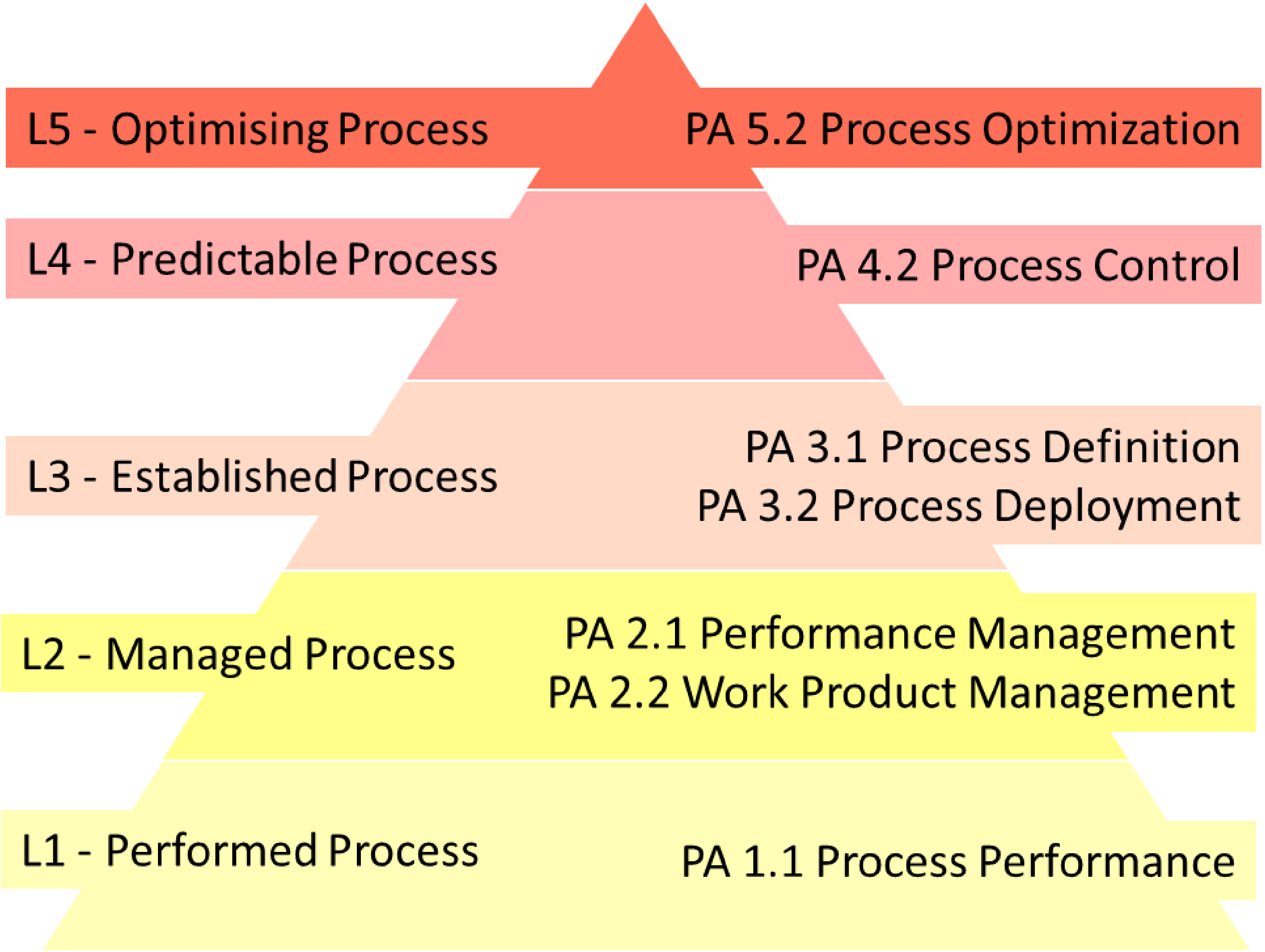
- “ISO/IEC 9126 in practice: what do we need to know?”, P. Botella, X. Burgués, J.P. Carvallo, X. Franch, G. Grau, J. Marco, C. Quer

- “Code Quality Evaluation Methodology Using the ISO/IEC 9126 Standard”, Yiannis Kanellopoulos, Panos Antonellis, Dimitris Antoniou, Christos Makris, Evangelos Theodoridis, Christos Tjortjis, and Nikos Tsirakis, International Journal of Software Engineering & Applications (IJSEA), Vol.1, No.3, July 2010

ISO/IEC 15504

ISO/IEC 155504 : **Information technology - Process assessment**, also known as Software Process Improvement and Capability Determination (**SPICE**) is a set of technical standards defining a precise capability level hierarchy.

SPICE Capability Levels and Process Attributes



L5 - Optimising Process

PA 5.2 Process Optimization

L4 - Predictable Process

PA 4.2 Process Control

L3 - Established Process

PA 3.1 Process Definition
PA 3.2 Process Deployment

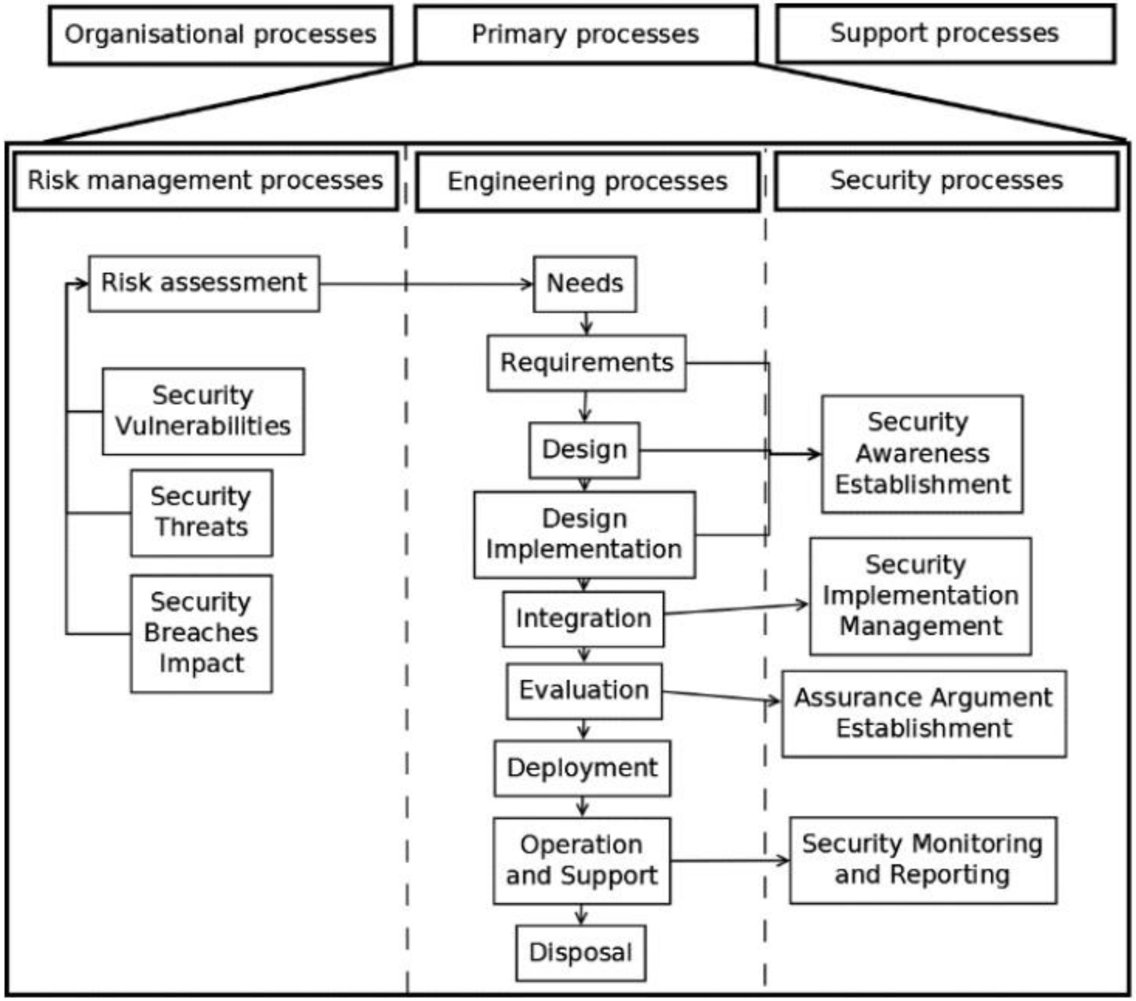
L2 - Managed Process

PA 2.1 Performance Management
PA 2.2 Work Product Management

L1 - Performed Process

PA 1.1 Process Performance

Relationship of the primary process category processes



source: https://www.researchgate.net/publication/272321908_Security_Process_Capability_Model_Based_on_ISOIEC_15504_Conformant_Enterprise_SPICE

SPICE Example

Legend PA

- 🟩 Fully achieved
- 🟨 Largely achieved
- 🟧 Partially achieved
- 🟥 Not achieved

	PA 1.1	PA 2.1	PA 2.2	PA 3.1.	PA 3.2	LEVEL
ENG.2 Sys. req. analysis	🟥	🟥	🟥			0
ENG.3 Sys. Arch. design	🟧	🟥	🟥			0
ENG.4 SW requ. analysis	🟨	🟥	🟥			1
ENG.5 SW design	🟩	🟩	🟧			1
ENG.6 SW construction	🟩	🟨	🟨			2
ENG.7 SW integration (test)	🟩	🟨	🟨			2
ENG.8 SW testing	🟨	🟨	🟧			1
ENG.9 Sys. integration (test)	🟨	🟧	🟧			1
ENG.10 Sys. testing	🟧	🟧	🟧			0
ACQ.4 Supplier monitoring	🟩	🟩	🟩	🟨	🟨	3
SUP.1 Quality assurance	🟨	🟧	🟧			1
SUP.8 Configuration man.	🟧	🟧	🟧			0
SUP.9 Problem res. man.	🟨	🟧	🟧			1
SUP.10 Change requ. Man.	🟨	🟨	🟧			1
MAN.3 Proj. Management	🟩	🟩	🟨			2

Lowest possible rating to reach level 1

Lowest possible rating to reach level 2

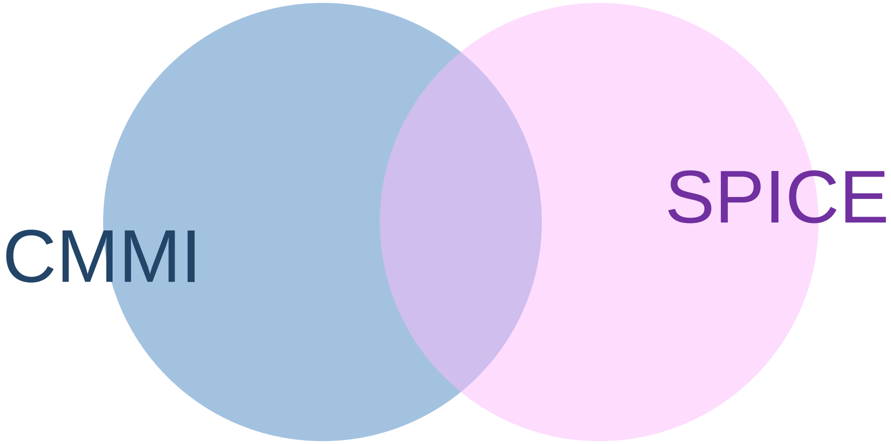
Lowest possible rating to reach level 3

Source: https://www.flecsim.de/images/download/fs-ueberblickspicecmmi.pdf

Capability Maturity Model Integration CMMI

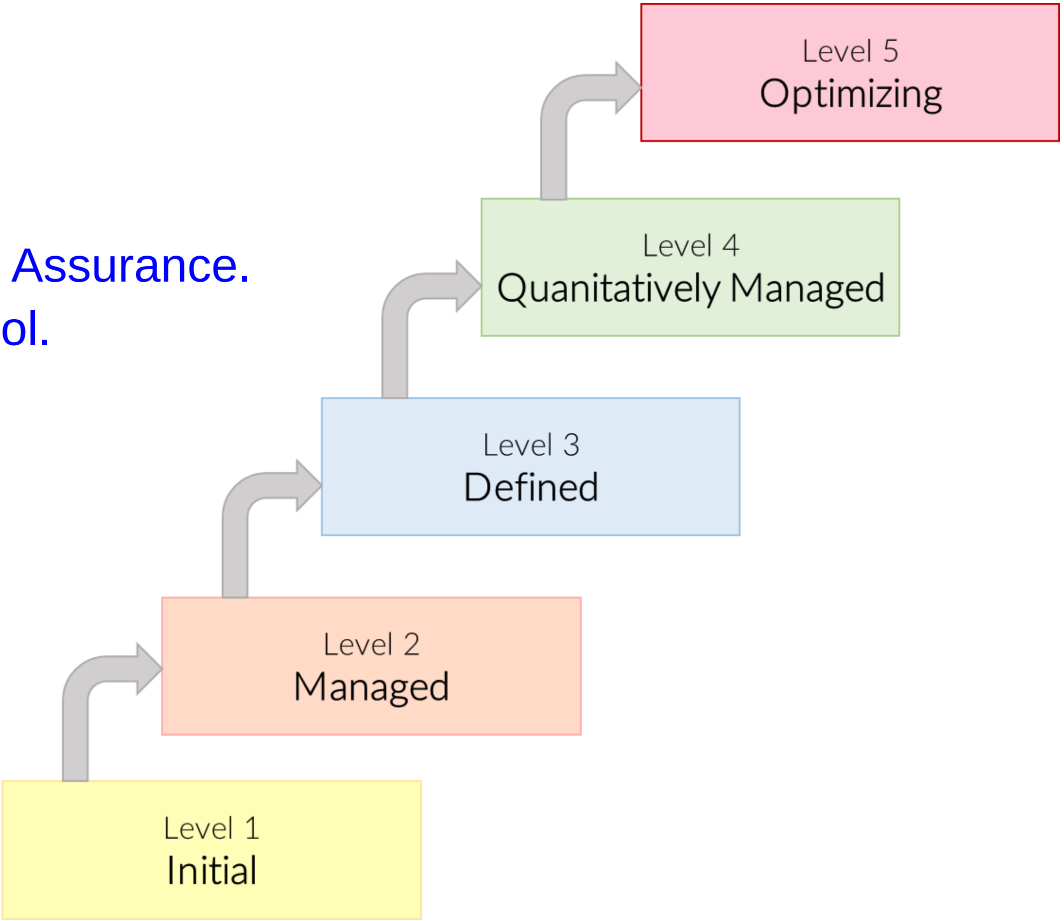
The Capability Maturity Model Integration is a model for optimizing development processes, like many others the standard is not focused on projects (such as application development) but mainly covers business processes. CMMI provides 5 maturity levels.

Relationship with SPICE



CMMI Maturity Levels

- Level 1: Initial
- Level 2: Managed
  - Configuration Management.
  - Process and Product Quality Assurance.
  - Project Monitoring and Control.
  - Project Planning.
- Level 3: Defined
  - Risk Management.
  - Organizational Training.
  - Requirements Validation.
- Level 4: Quantitatively Managed
- Level 5: Optimizing



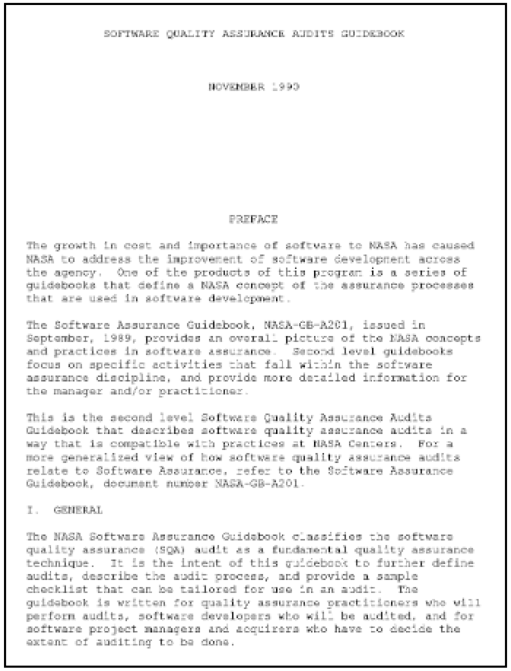
Another example … by NASA

NASA = National Aeronautics
and Space Administration

Software Quality Assurance Audits Guidebook 1990

A little bit dated … but still valid !

At NASA Software Quality Assurance is of paramount importance.



SOFTWARE QUALITY ASSURANCE AUDITS GUIDEBOOK

NOVEMBER 1990

PREFACE

The growth in cost and importance of software to NASA has caused NASA to address the improvement of software development across the agency. One of the products of this program is a series of guidebooks that define a NASA concept of the assurance processes that are used in software development.

The Software Assurance Guidebook, NASA-GB-A201, issued in September, 1989, provides an overall picture of the NASA concepts and practices in software assurance. Second level guidebooks focus on specific activities that fall within the software assurance discipline, and provide more detailed information for the manager and/or practitioner.

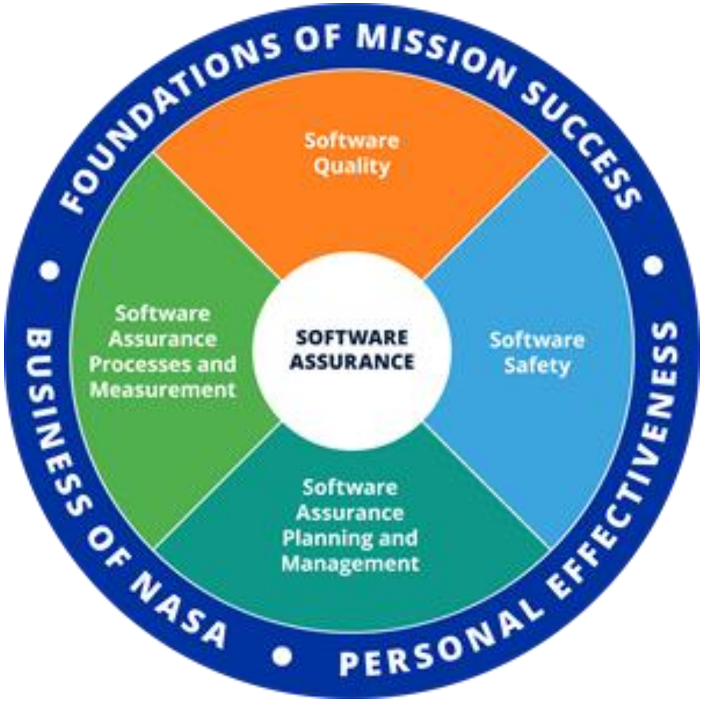
This is the second level Software Quality Assurance Audits Guidebook that describes software quality assurance audits in a way that is compatible with practices at NASA Centers. For a more generalized view of how software quality assurance audits relate to Software Assurance, refer to the Software Assurance Guidebook, document number NASA-GB-A201.

I. GENERAL

The NASA Software Assurance Guidebook classifies the software quality assurance (SQA) audit as a fundamental quality assurance technique. It is the intent of this guidebook to further define audits, describe the audit process, and provide a sample checklist that can be tailored for use in an audit. The guidebook is written for quality assurance practitioners who will perform audits, software developers who will be audited, and for software project managers and acquirers who have to decide the extent of auditing to be done.

NASA SQA - Macro Phases

I. Conducting a SQA audit.
II. SQA audit scheduling.
III. SQA audits during the software life cycle.
IV. Preparing a checklist.
V. Auditing in the absence of standards and procedures.
VI. Qualities of an auditor.
VII. Techniques and tools.


FOUNDATIONS OF MISSION SUCCESS
PERSONAL EFFECTIVENESS
BUSINESS OF NASA
Software Quality
Software Safety
Software Assurance Planning and Management
Software Assurance Processes and Measurement
SOFTWARE ASSURANCE