KHMELNITSKY NATIONAL UNIVERSITY
SYSTEM PROGRAMMING DEPARTMENT

EVALUATION OF MUTUAL INFLUENCES OF SOFTWARE QUALITY CHARACTERISTICS BASED ISO 25010:2011

Tetiana Hovorushchenko

Actuality

Software quality is the ability of the software to meet the stated and predicted needs when using under certain conditions

[ISO 25010, ISO 25030].

One of the most important causes of poor quality of large software projects are the increasing the number of components (subsystems) and the interfaces between them, and uncontrolled complexity of software systems

[CHAOS Report, The Standish Group International].

Actuality: Statistics 1

Success of small, moderate, medium, large and grand software projects is significantly different:

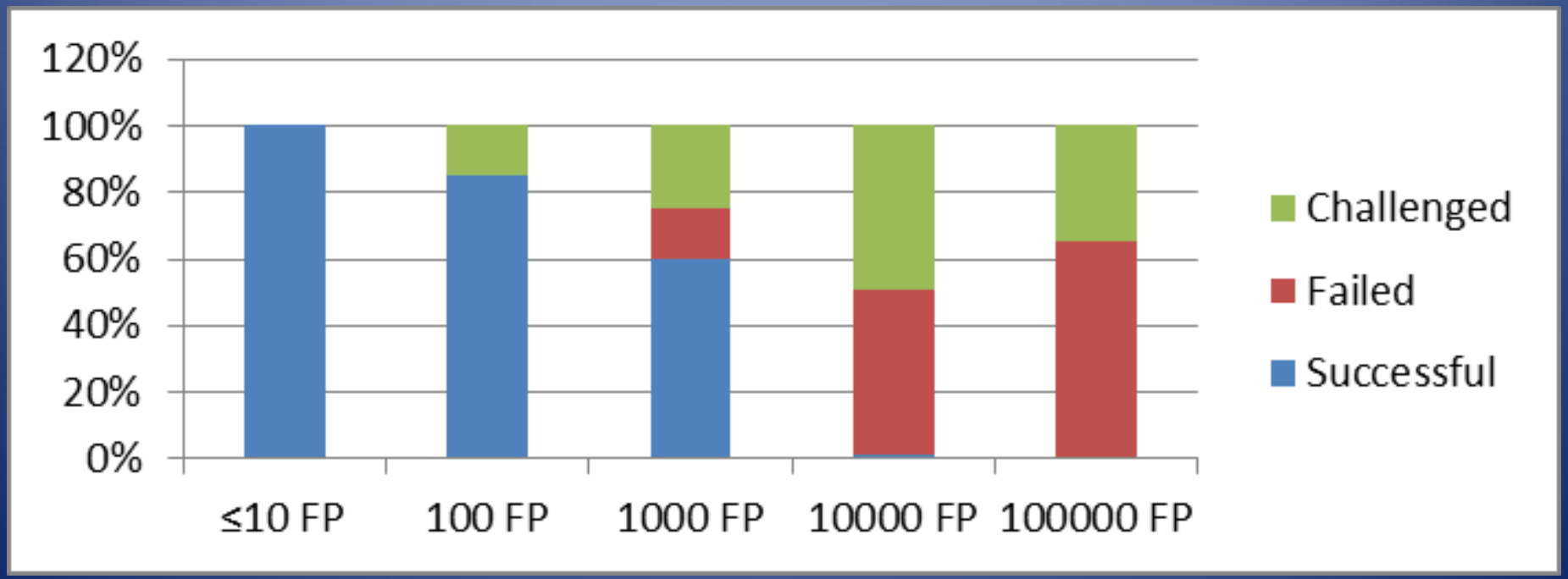
CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

Actuality: Statistics 2

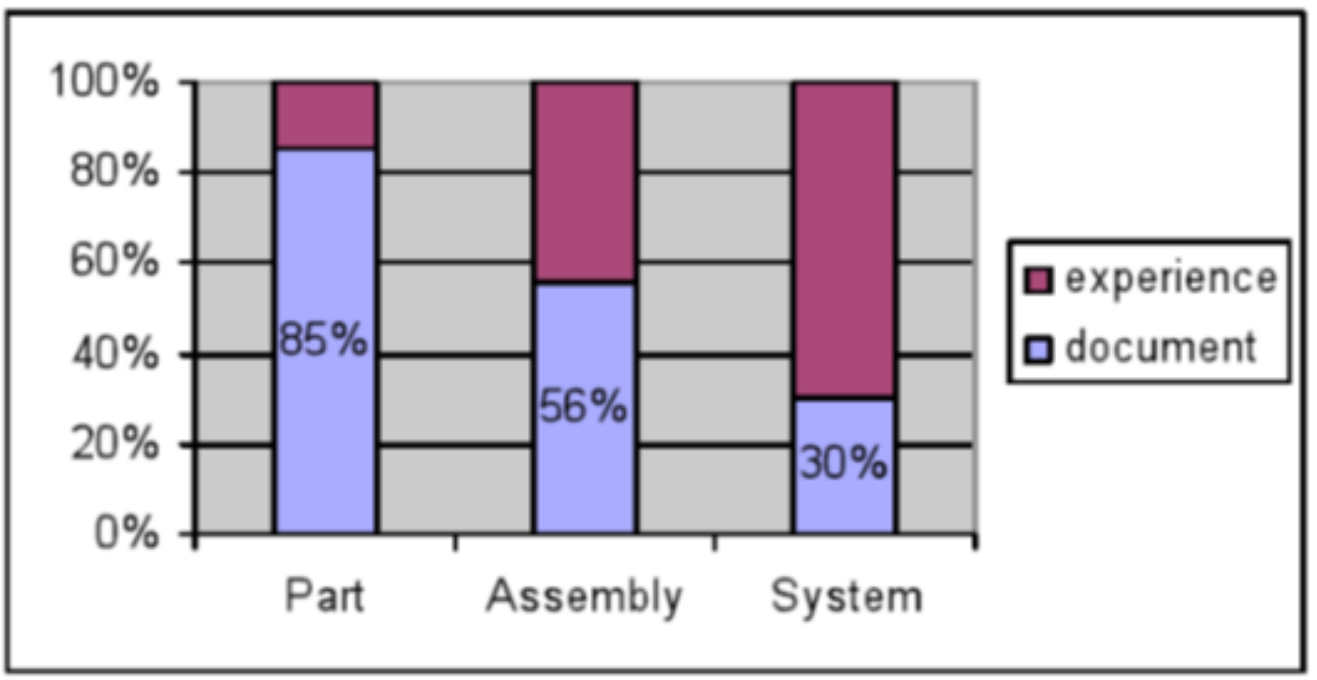
Software projects on the basis of the function points as the main modern units of software size:



Informational Indeterminacy

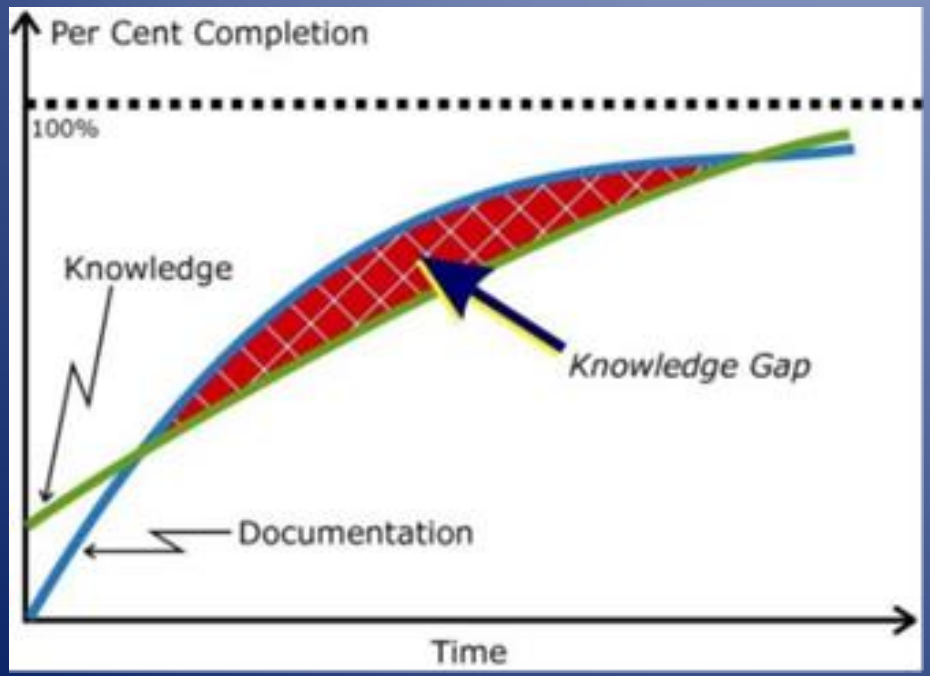
During the software project, we often can't estimate the share of the informational indeterminacy of the project.

The cause of appearance of informational indeterminacy is the low level of knowledge documentation, especially at the system level :

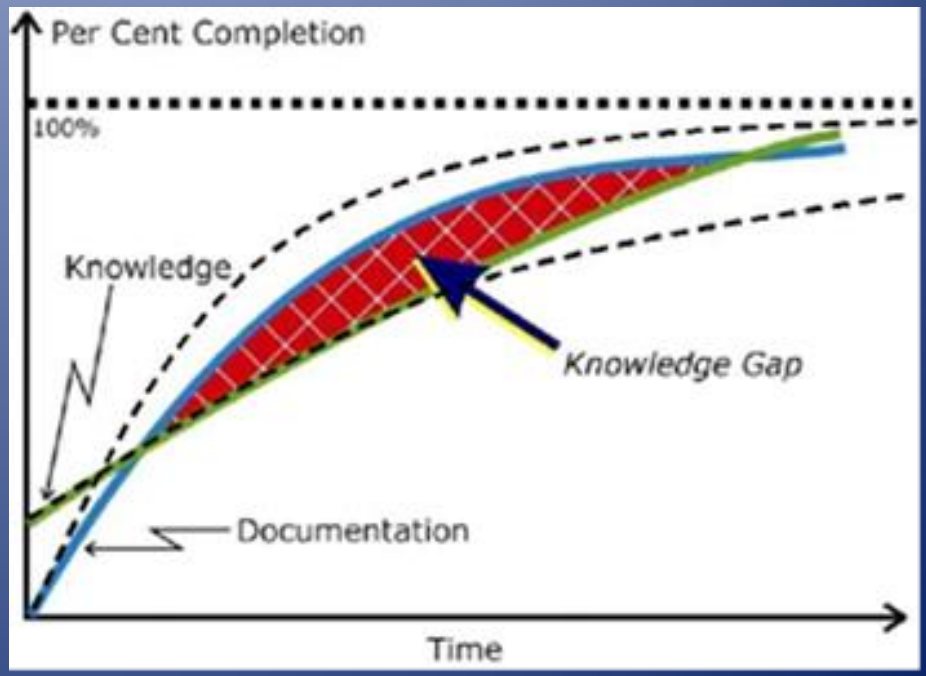


Knowledge gap

Premature design decisions and their documentation, prior to understanding the design – area, referred to as the **"knowledge gap"** (the result of the low level of knowledge documentation and the root cause of many engineering failures):



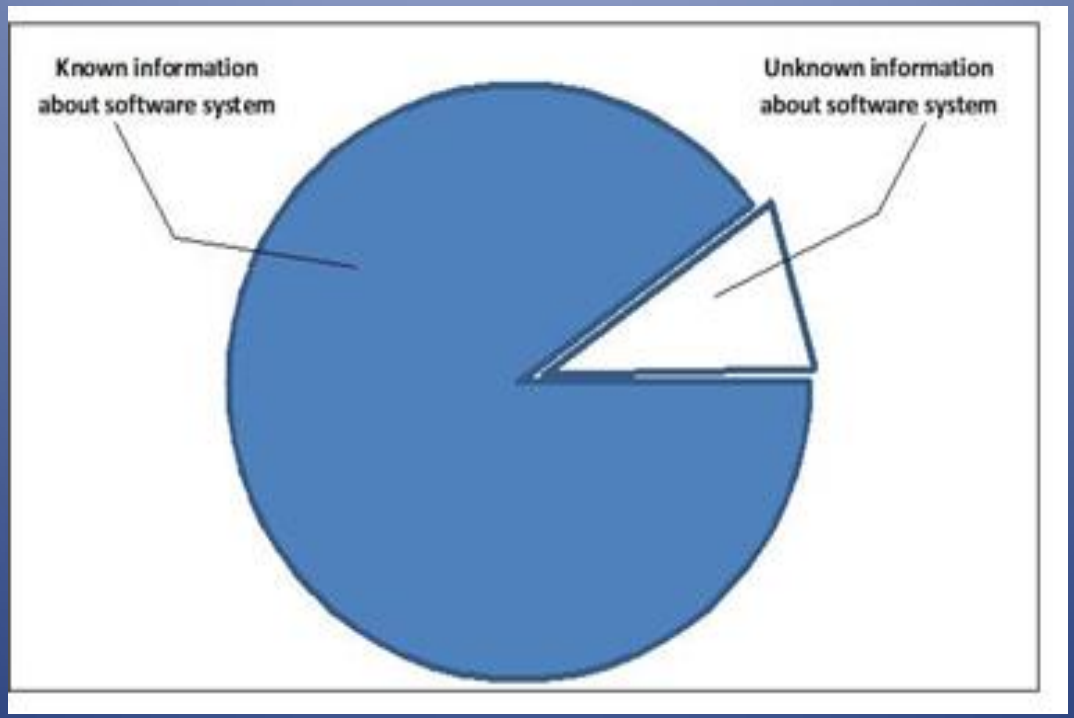
Patterson's vision



Our vision

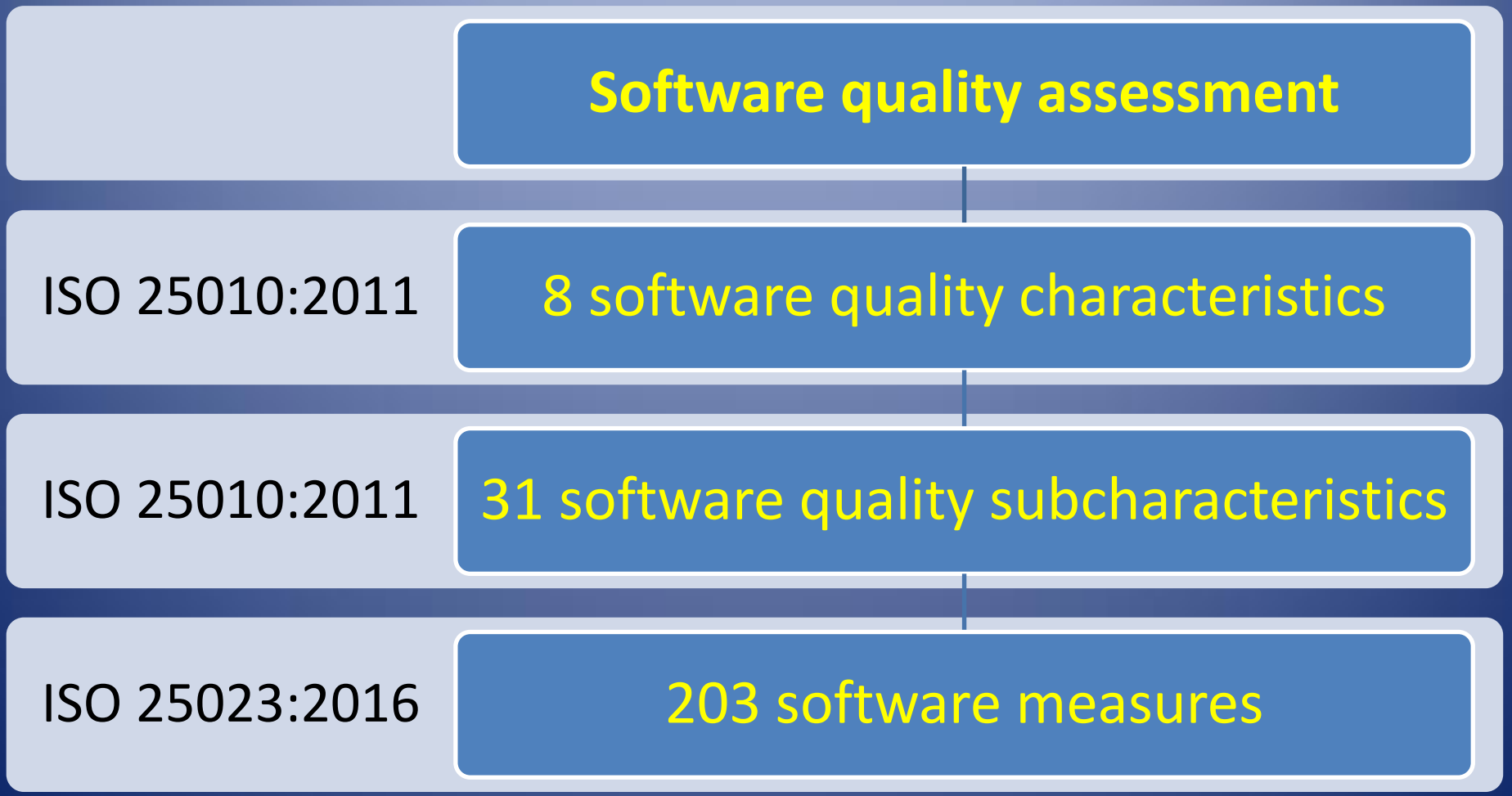
Unknown subject domain information

All available knowledge and information about the software system can be represented as the diagram, which has **the sector that reflects the volume of insufficient (unknown) information (knowledge gap):**

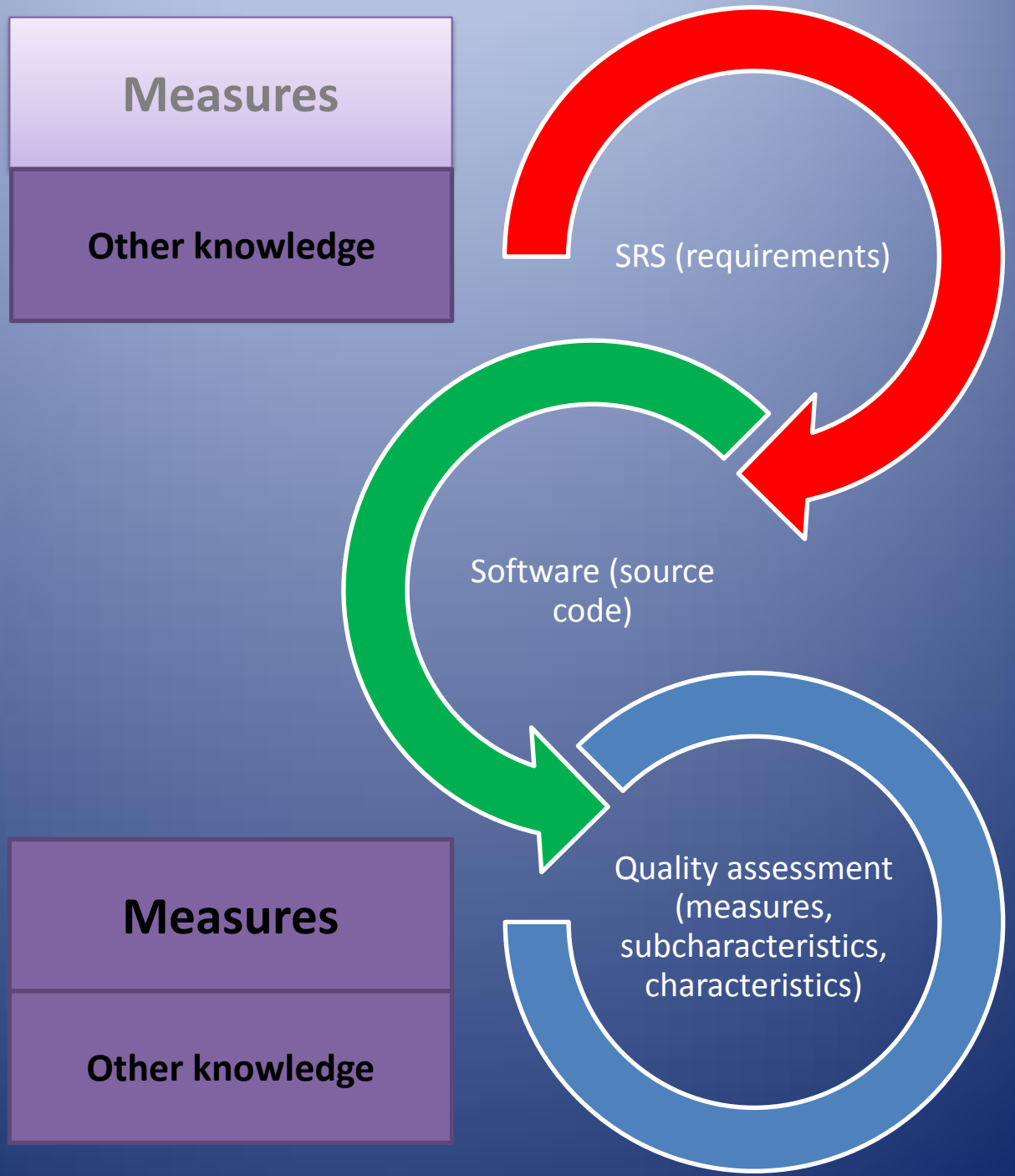


This sector consists of unconsidered subject domain information.

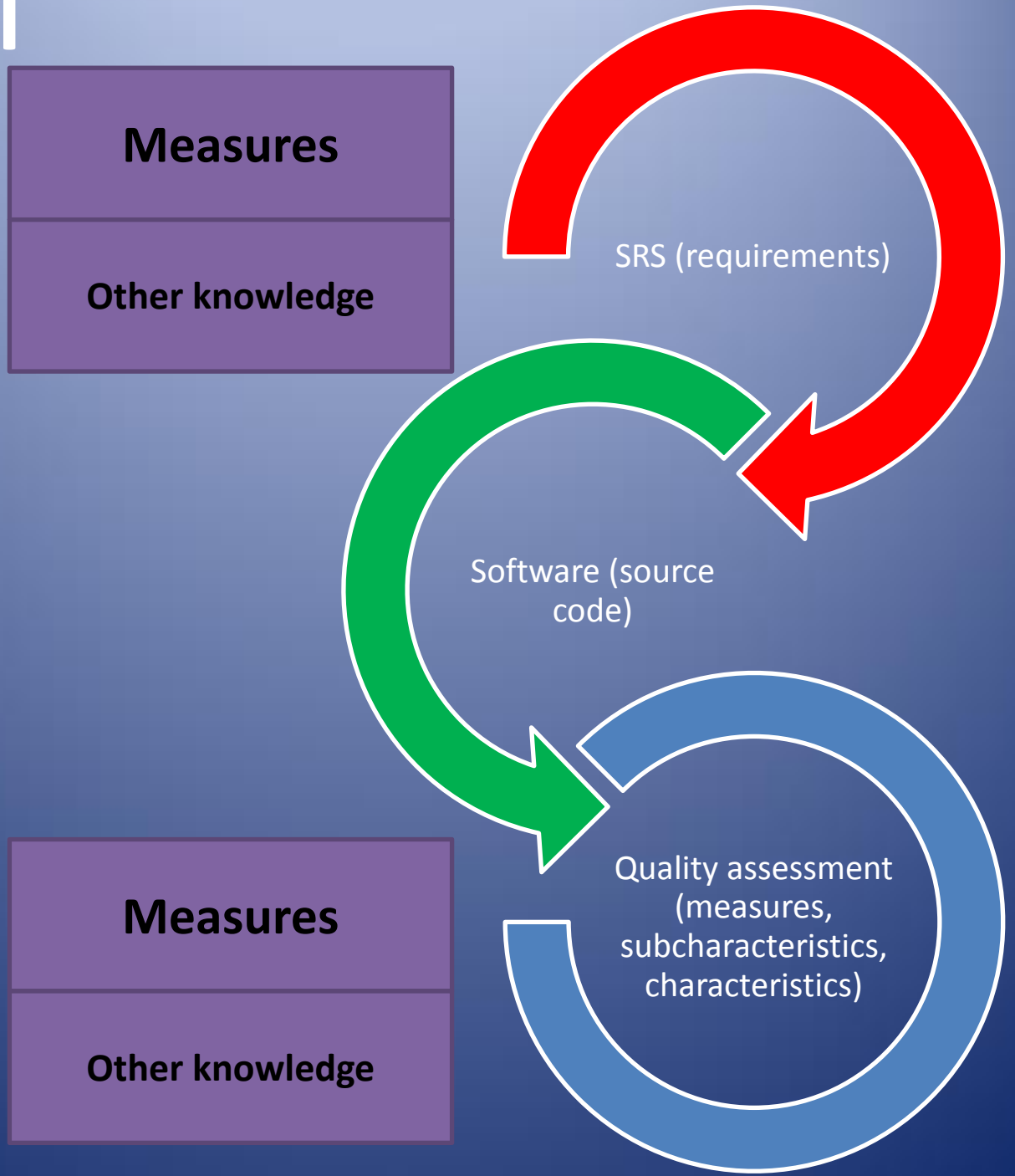
Software quality model



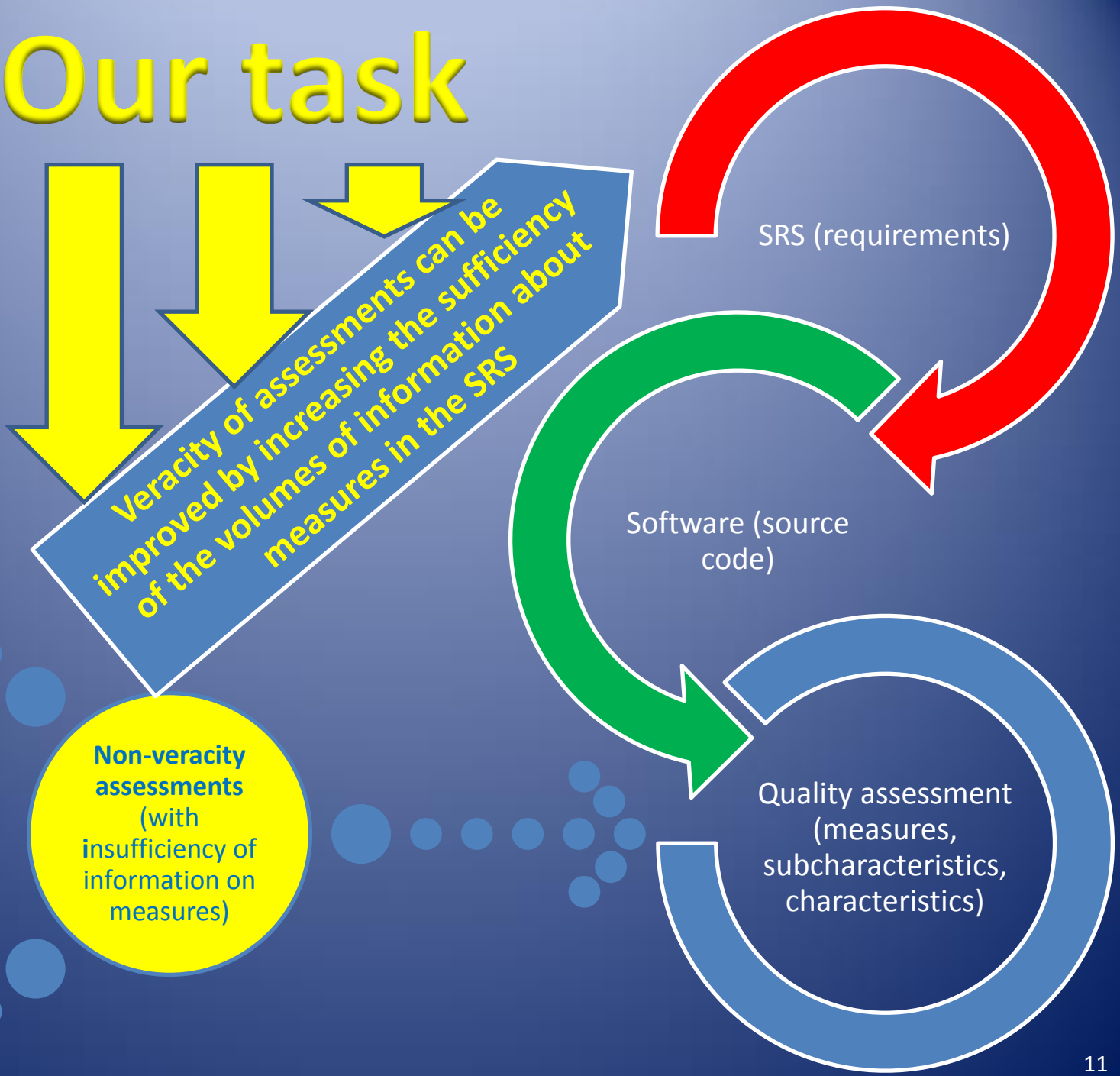
Now


Measures
Other knowledge
SRS (requirements)
Software (source code)
Quality assessment (measures, subcharacteristics, characteristics)
Measures
Other knowledge

Proposal


Measures
Other knowledge
SRS (requirements)
Software (source code)
Quality assessment (measures, subcharacteristics, characteristics)
Measures
Other knowledge

Our task


Veracity of assessments can be improved by increasing the sufficiency of the volumes of information about measures in the SRS
SRS (requirements)
Software (source code)
Quality assessment (measures, subcharacteristics, characteristics)
Non-veracity assessments (with insufficiency of information on measures)
Joint measures
Quality subcharacteristics and characteristics cross-correlation

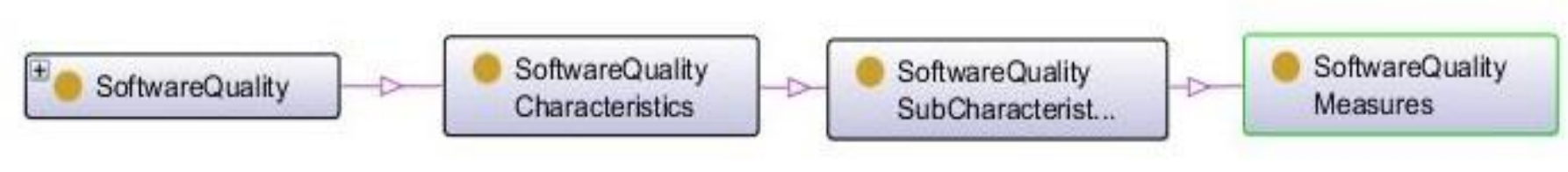
Concrete task:

evaluation of mutual influences of software quality characteristics (determination of the degree of mutual influences of software quality characteristics based ISO 25010)

To accomplish this we should to calculate the weights of measures which define the software quality characteristics.

Mutual Influences of Software Quality Characteristics

Idea of base ontology for subject domain “Software engineering” (part “Software quality”):



Quantity of subcharacteristics and measures for software quality characteristics:

Software Quality Characteristic	Quantity of Subcharacteristics	Quantity of Measures
Functional Suitability	3	15
Reliability	4	30
Usability	6	49
Security	5	23
Performance Efficiency	3	26
Maintainability	5	33
Portability	2	9
Compatibility	3	18

Dependences of the quality characteristics on the joint measures:

1) measure Operation Time is part of all 8 software quality characteristics;

2) measure Number Of Functions is part of Functional Suitability, Reliability, Usability, Maintainability, Compatibility, Portability;

3) measure Number Of Data Items is part of Functional Suitability, Performance Efficiency, Maintainability, Security, Compatibility, Portability;

4) measure Number Of Faults is part of Reliability, Portability;

5) measure Number Of Failures is part of Reliability, Performance Efficiency, Maintainability, Compatibility;

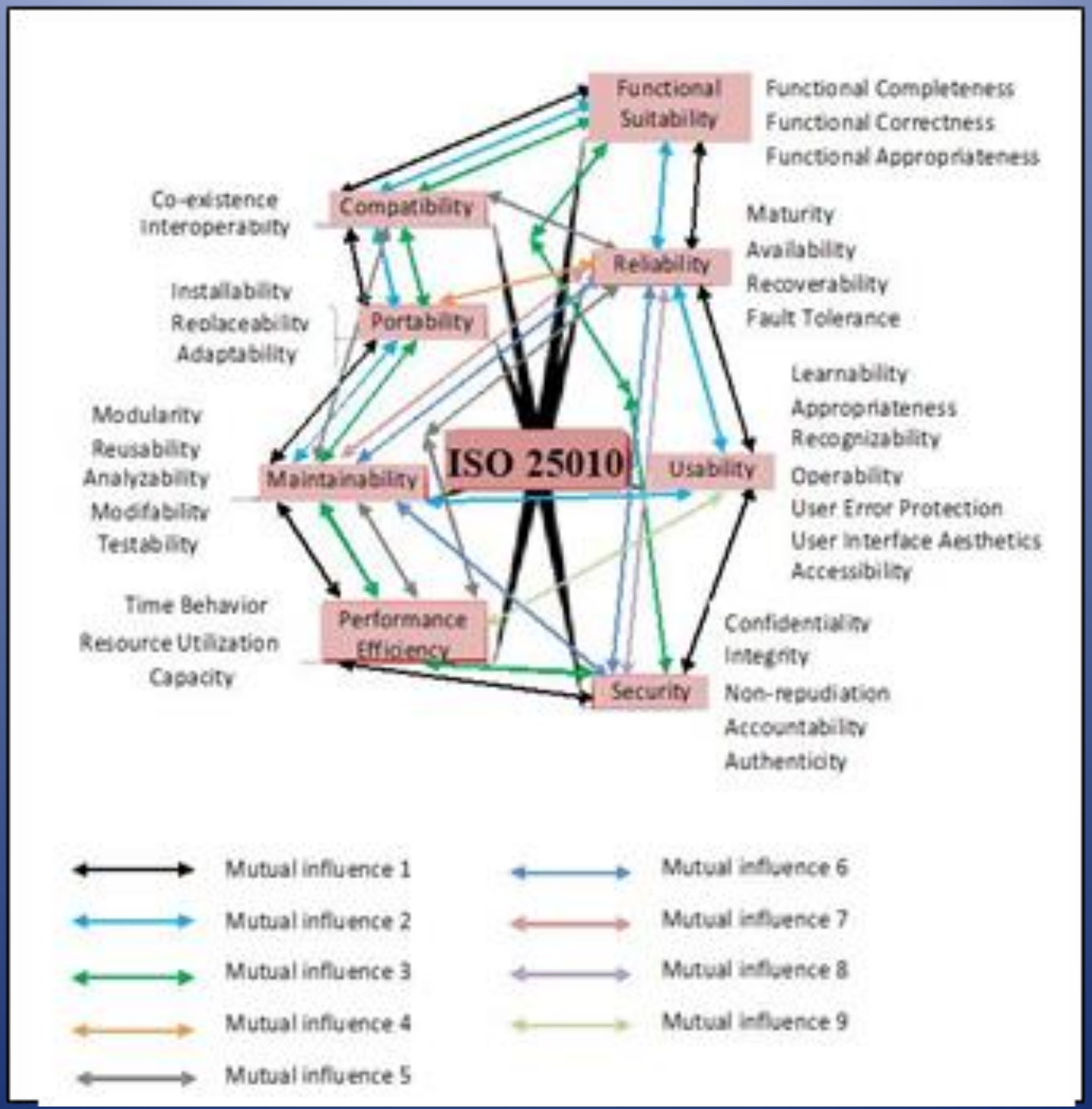
6) measure Number Of Test Cases is part of Reliability, Maintainability, Security;

7) measure Number Of Resolved Failures is part of Reliability, Maintainability;

8) measure Number Of Illegal Operations is part of Reliability, Security;

9) measure Number Of Tasks, measure Number Of IO-Related Errors is part of Performance Efficiency, Usability.

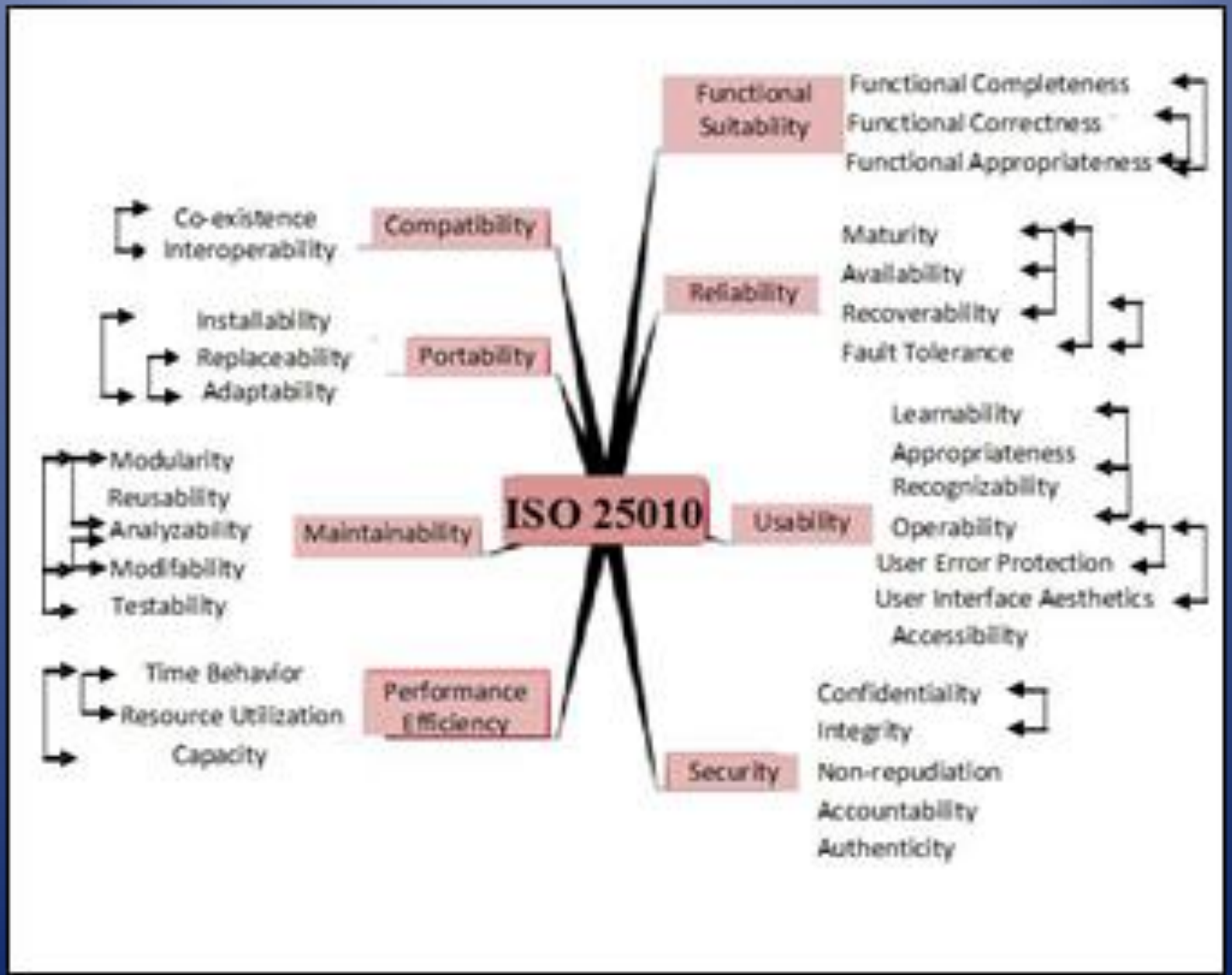
The mutual influences of software quality characteristics:



Pair wise comparison of software quality characteristics for number of joint measures:

	Functional Suitability	Reliability	Usability	Security	Performance Efficiency	Maintainability	Portability	Compati-bility
Functional Suitability		2	2	2	2	3	3	3
Reliability	2		2	3	2	5	3	3
Usability	2	2		1	2	2	2	2
Security	2	3	1		2	3	2	2
Performance Efficiency	2	2	2	2		3	2	3
Maintainability	3	5	2	3	3		3	4
Portability	3	3	2	2	2	3		3
Compatibility	3	3	2	2	3	4	3	

The mutual influences of software quality subcharacteristics within each quality characteristic:



Dependence of quality subcharacteristics on joint measures:

	Functional Suitability		
	Functional Completeness	**Functional Correctness**	**Functional Appropriateness**
Number Of Functions	+		+
Functional Implementation Completeness	+		+
Functional Adequacy	+		+
Functional Implementation Coverage	+		+
Operation Time		+	+
Precision		+	+
Number Of Data Items		+	

	Reliability			
	Maturity	**Availability**	**Recover-ability**	**Fault Tolerance**
Operation Time	+	+	+	
Number Of Failures	+			+
Number Of Test Cases	+			+
Number Of Breakdowns			+	+
Number Of Functions				+
Number Of Faults	+			
Number Of Resolved Failures	+			
Number Of Illegal Operations				+

The existence of relationships between characteristics and subcharacteristics affect the significance and importance of the software quality characteristics, that is expressed as weights.

Because there are measures that affect more than one software quality subcharacteristic, we will evaluate the weight of the i-th measure by the following expression:

$$w_{mi} = k_{schmi}/k_m \quad (1)$$

where k_{schmi} – quantity of subcharacteristics, that depends on i-th measure; k_m – total quantity of measures (analysis of ISO 25023 shows, that nowadays the software quality characteristics and subcharacteristics depend on the total 203 measures, i.e. k_m=203).

Weights of joint measures:

Measure	Weight	Measure	Weight
Number Of Functions	11/203	Functional Implementation Coverage	2/203
Functional Implementation Completeness	2/203	Operation Time	17/203
Functional Adequacy	2/203	Precision	2/203
Number Of Data Items	8/203	Number Of Tasks	2/203
Number Of Failures	6/203	Number Of IO-Related Errors	2/203
Number Of Test Cases	5/203	Number Of Instances Of Data Corruption	2/203
Number Of Breakdowns	2/203	Number Of Access Types	2/203
Number Of Faults	3/203	Number Of Controllability Requirements	2/203
Number Of Resolved Failures	4/203	Access Controllability	2/203
Number Of Illegal Operations	3/203	Number Of Evaluations	2/203
Number Of User Errors Or Changes	2/203	Mean Amount Of Throughput	2/203
Number Of Interface Elements	2/203	Error Time	2/203

Conclusions

- The correlation of software quality characteristics and subcharacteristics should be taken into account during the evaluation of the weights of software quality characteristics. In this paper **the weights of measures are calculated**.
- During assessing the software quality characteristics the ensuring **the presence of the measures with larger weights in the software requirements specification is very important** with purpose to ensure the appropriate level of veracity of software quality assessment. Thus, **the calculated weights of measures provide prioritize of addition of measures in the SRS**.
- **Further research** of the authors focused on development of the methods of evaluation of the software quality characteristics for ISO 25010 considering the established mutual influences of the software quality characteristics.

THANK YOU FOR ATTENTION!

Tetiana Hovorushchenko

Doctor, Senior Researcher, Associate of Professor,

Lecturer of System Programming Department

tat_yana@ukr.net , govorushchenko@gmail.com

Oksana Pomorova,

Doctor of Science, Full Professor

Head of System Programming Department

o.pomorova@gmail.com