softarex technologies

Meet deadline with best software product

Methods, Tools, and Approaches for Source Code Analysis and Quality

Alexander Lipanov
CEO of Softarex Technologies Inc,
PhD in Applied Mathematics
alex@softarex.com

Alexander Chelombitko
Head of Department,
Softarex Technologies Inc
achelombitko@softarex.com

ORACLE Gold Partner

OMG WE SET THE STANDARD™

Open Health Tools

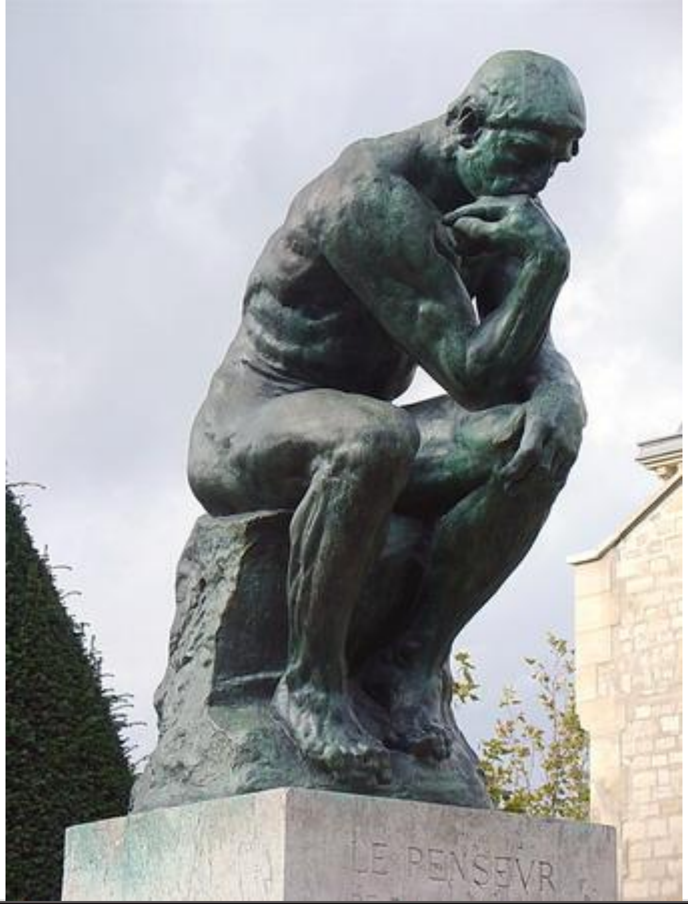
nvtc NORTHERN VIRGINIA TECHNOLOGY COUNCIL

SEND YOUR QUESTIONS TO JAVAONE@SOFTAREX.COM

In Presentation

- **Why need have quality software?**
  - Software Quality and its influence to business
  - Software Quality definitions
  - Who and what influencing to software quality
  - Software Quality and Testing Maturity Model
- **Quality Target Point**
  - Definition of Quality Target Point
  - Best software product definition through set of Quality Target points
  - Parameters of Quality Target Point and its measures: Statistical metrics; Object oriented metrics; CISQ measures for quality characteristics; Additional parameters
- **Software development process with Quality Target Point**
  - Rules for definition of Quality Target Point
  - Preparing your process for usage of Quality Target Point
  - Integration of Quality Target Point into SCRUM process
- **Practical aspects of Quality Target Point usage**
  - Tools for measuring of parameters for Quality Target Point
  - codeNforcer is a tool for Software Quality improvements
  - Quality Target Point on example of Java code
  - How looks in code problems which can be defined during code analysis by codeNforcer (some examples)
- **Conclusion**

Why need have quality software?



Software Quality and its influence to business

Business Outcome

Management

=

Functionality: Meet clients expectations

Security: Keep safely all customers information

Reliability: Keep and grow your clients base, grow your revenue

Performance: Grow clients base and decrease operating cost

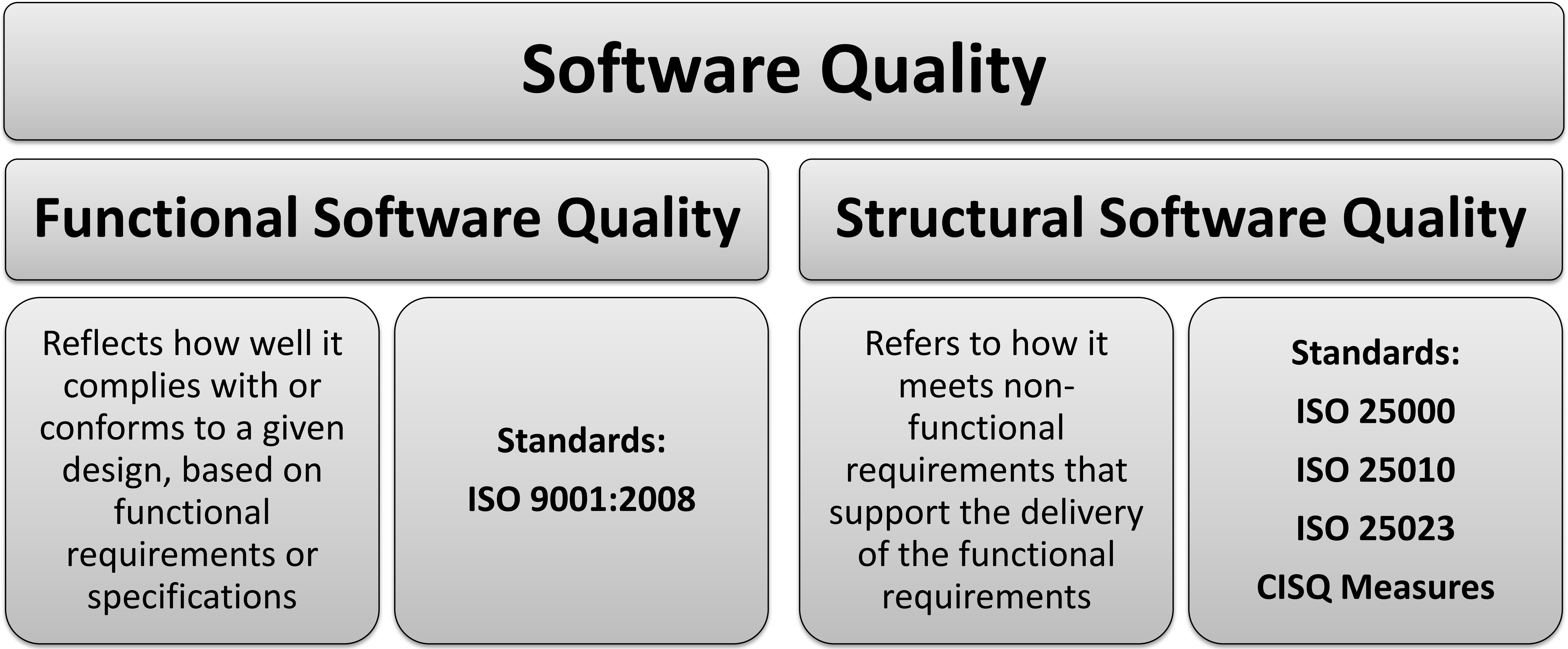
Support: Decrease support time and costs, ownership costs, time to market

Increase clients satisfaction and Win before competitors

=

Maximized Revenue and Profit

Software Quality Definitions



Software Quality

Functional Software Quality

Reflects how well it complies with or conforms to a given design, based on functional requirements or specifications

Standards:

ISO 9001:2008

Structural Software Quality

Refers to how it meets non-functional requirements that support the delivery of the functional requirements

Standards:

ISO 25000

ISO 25010

ISO 25023

CISQ Measures

SEND YOUR QUESTIONS TO JAVAONE@SOFTAREX.COM

Definitions (ISO/IEC 25010)

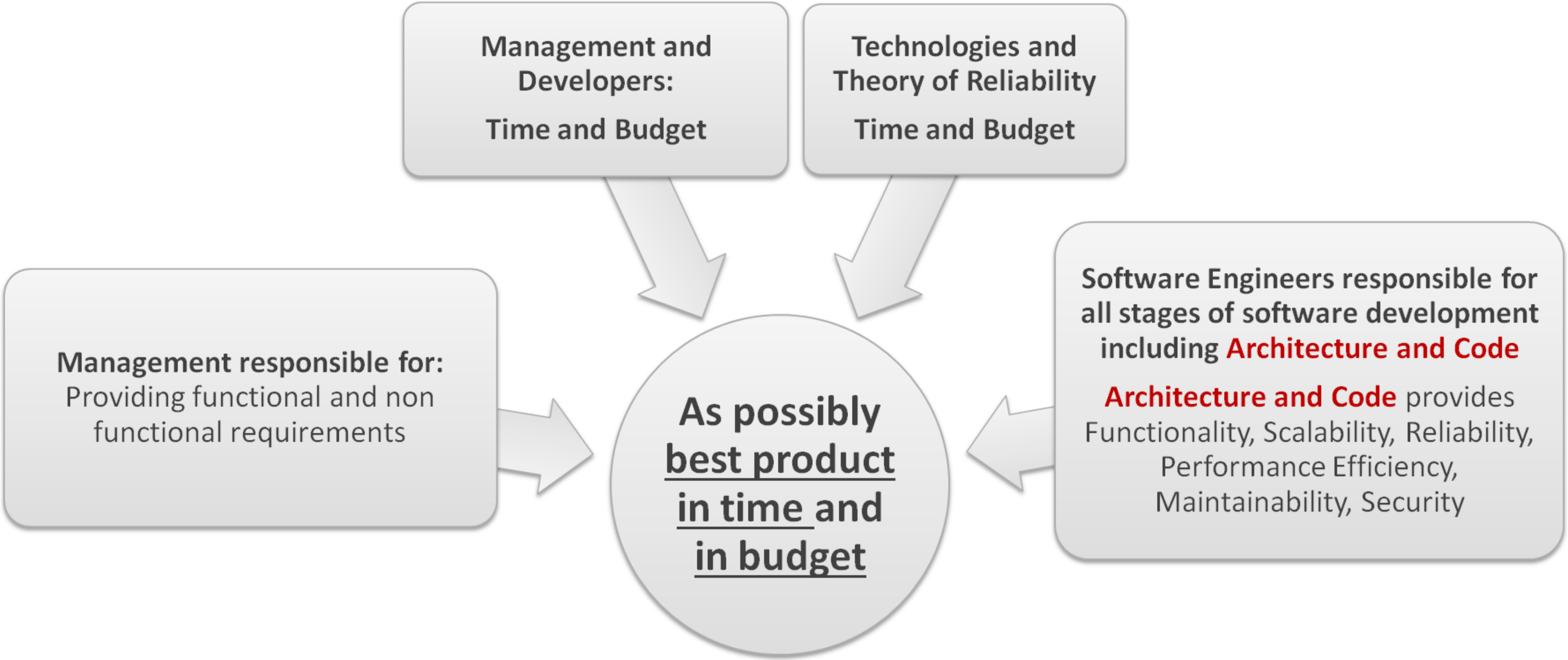
Software Product Quality Model—a model that categorizes product quality properties into eight characteristics (functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability). Each characteristic is composed of a set of related sub-characteristics

Software Quality degree to which a software product satisfies stated and implied needs when used under specified conditions

Structural Quality the degree to which a set of static attributes of a software product satisfy stated and implied needs for the software product to be used under specified conditions—a component of software quality. This concept is referred to as internal software quality

Software Quality Model a defined set of software characteristics, and of relationships between them, which provides a framework for specifying software quality requirements and evaluating the quality of a software product

Who Influencing to Software Quality



Management and Developers:

Time and Budget

Technologies and Theory of Reliability

Time and Budget

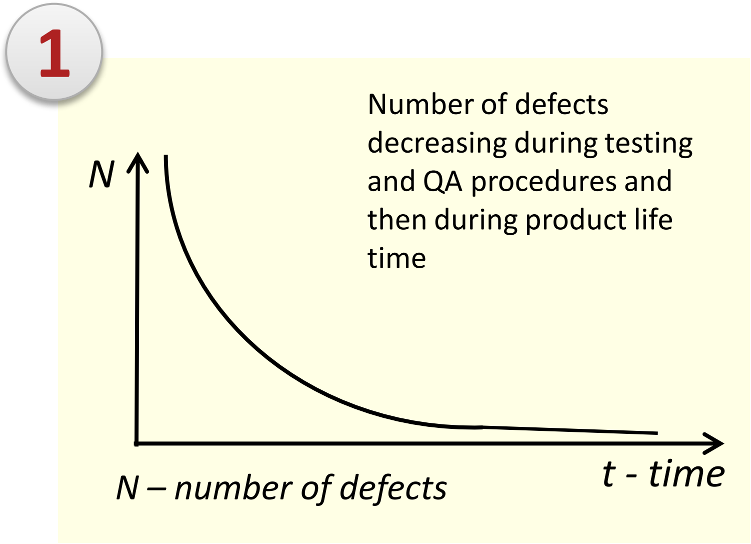
Management responsible for: Providing functional and non functional requirements

As possibly **best product in time and in budget**

Software Engineers responsible for all stages of software development including Architecture and Code

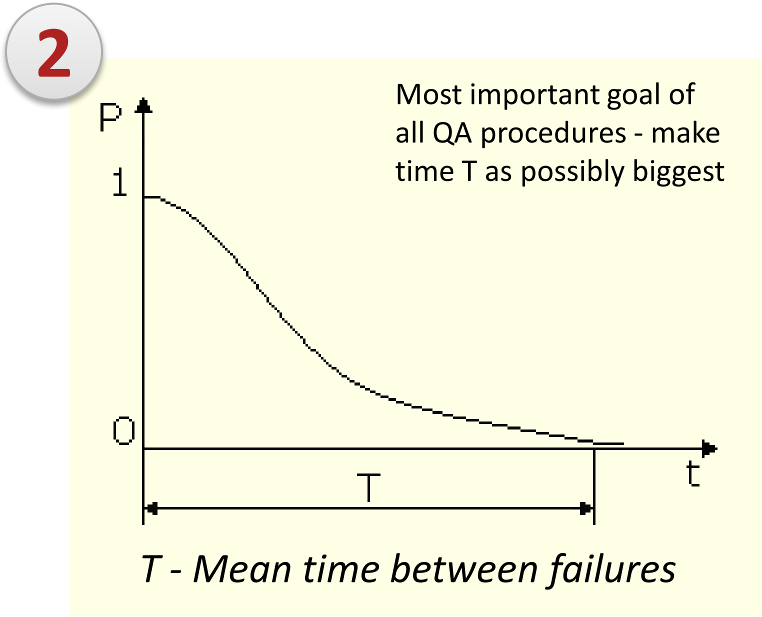
Architecture and Code provides Functionality, Scalability, Reliability, Performance Efficiency, Maintainability, Security

Remember about Theory of Reliability



Number of defects decreasing during testing and QA procedures and then during product life time

N – number of defects



Most important goal of all QA procedures - make time T as possibly biggest

T - Mean time between failures

- Than more time you invest into testing then more defects you can find. Define point when you can stop testing and move to production
- Than more time you spend for testing than biggest budget is necessary. Keep balance between time and budget

- Depending from product specifics it will have different requirements for reliability and different requirements for Mean time between failures. Define particular requirements for your product and realize them
- All your team should consider balance between time for quality procedures and budget

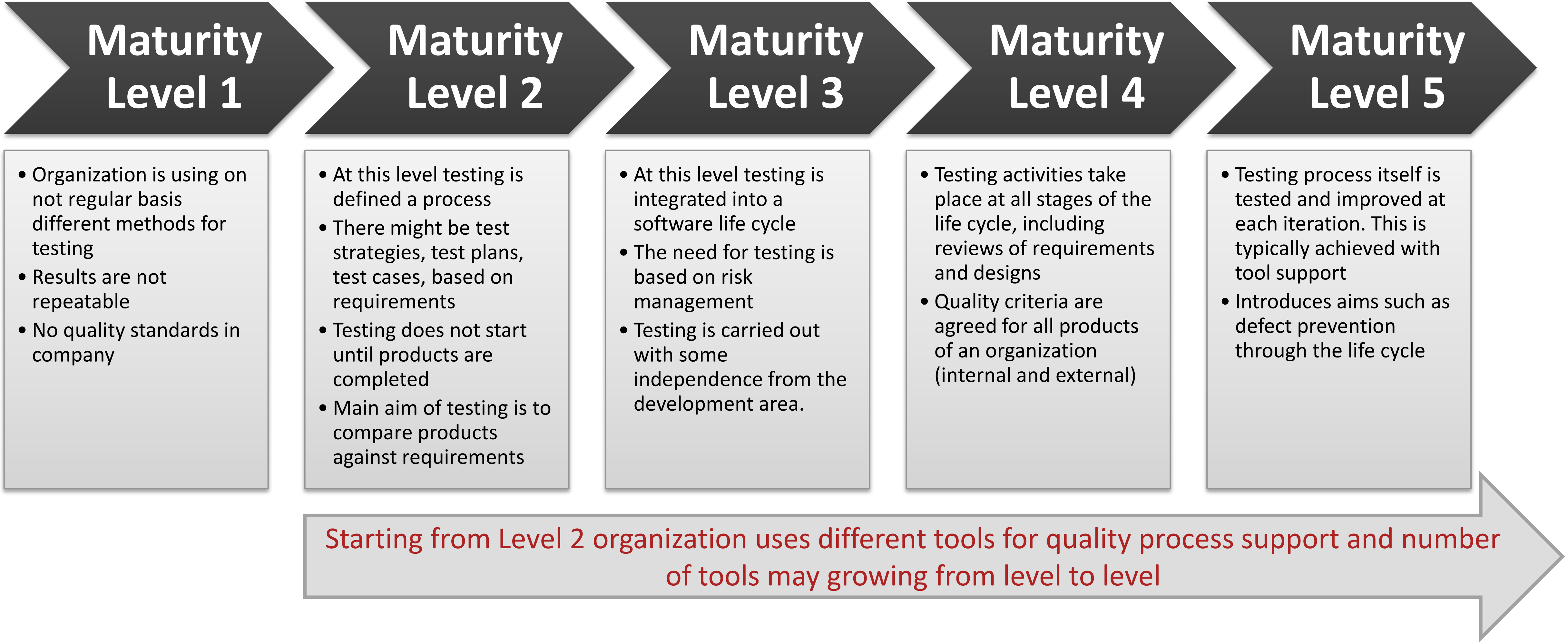
Software Quality and Testing Maturity Model

Capability Maturity Model Integration (CMMI) - models are collections of best practices that help organizations to improve their processes. These models are developed by product teams with members from industry, government, and the Carnegie Mellon® Software Engineering Institute (SEI). This model, called CMMI for Development (CMMI-DEV), provides a comprehensive integrated set of guidelines for developing products and services.

Test Maturity Model Integration (TMMI)- was based on the Capability Maturity Model Integration. Its aim is to provide a framework for assessing the maturity of the test processes in an organization, and so providing targets on improving maturity. TMMI now managed by the TMMI Foundation.

Software Quality and Testing Maturity Model

Each more higher level in model encapsulate and improve all properties and characteristics of lower level



Maturity Level 1

- Organization is using on not regular basis different methods for testing
- Results are not repeatable
- No quality standards in company

Maturity Level 2

- At this level testing is defined a process
- There might be test strategies, test plans, test cases, based on requirements
- Testing does not start until products are completed
- Main aim of testing is to compare products against requirements

Maturity Level 3

- At this level testing is integrated into a software life cycle
- The need for testing is based on risk management
- Testing is carried out with some independence from the development area.

Maturity Level 4

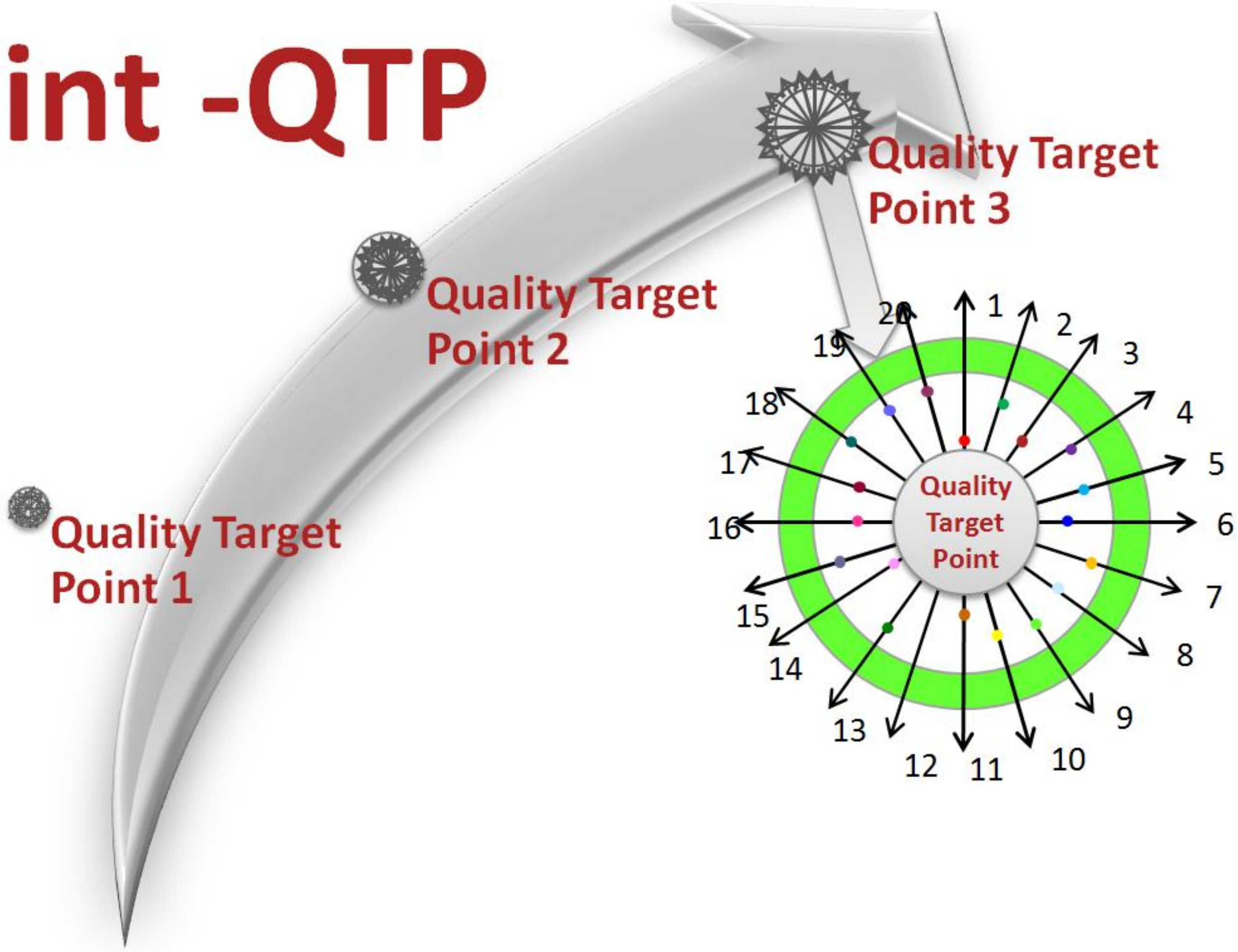
- Testing activities take place at all stages of the life cycle, including reviews of requirements and designs
- Quality criteria are agreed for all products of an organization (internal and external)

Maturity Level 5

- Testing process itself is tested and improved at each iteration. This is typically achieved with tool support
- Introduces aims such as defect prevention through the life cycle

Starting from Level 2 organization uses different tools for quality process support and number of tools may growing from level to level

Quality Target Point -QTP



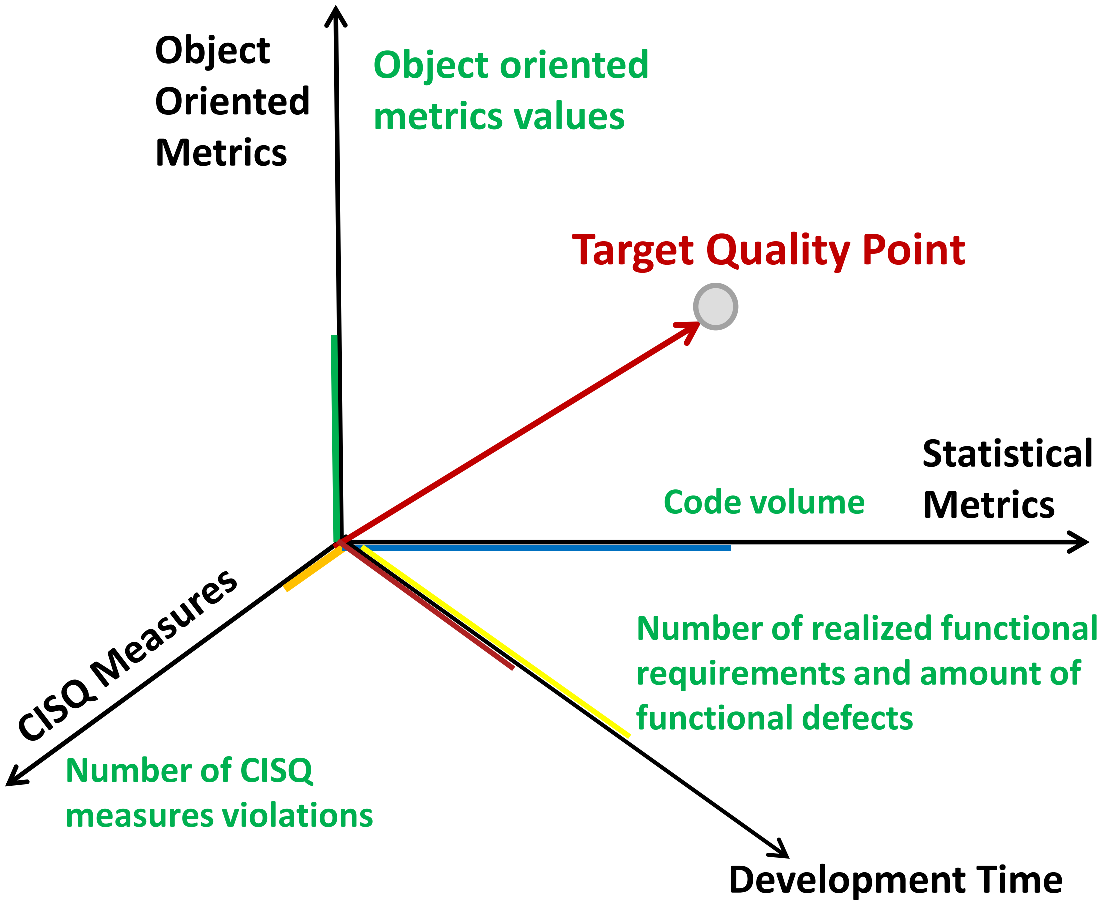
Base Definition of Quality Target Point (QTP)

Key factors influencing to Software Quality:

- Correspondence to Functional Requirements
- Code Architecture and organization
- Non functional requirements - Reliability, Performance Efficiency, Maintainability, Security
- Size of source code

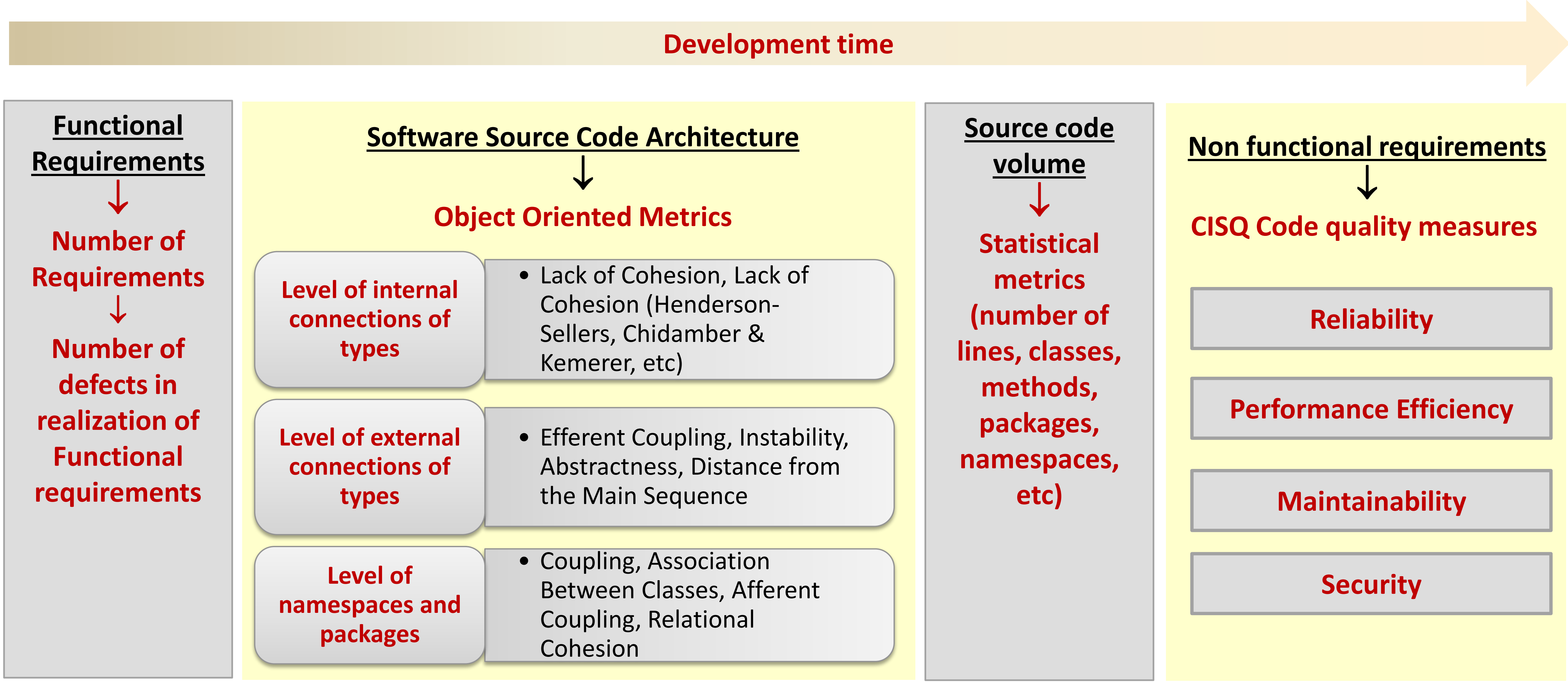
Quality Target Point – abstract point where realized necessary **Functional Requirements** and achieved following conditions:

1. Values of **Object Oriented Metrics** ∈ [*intervals of recommended values*]
2. Number **of CISQ Measures violations → 0**
3. **Source code volume have minimal size** and provide all necessary functionality. At least source code not have duplications
4. Number of defects in functionality meet to planned values which allow use system by users
5. Quality Target Point has exact defined date



Object Oriented Metrics
Object oriented metrics values
Target Quality Point
Statistical Metrics
Code volume
CISQ Measures
Number of realized functional requirements and amount of functional defects
Number of CISQ measures violations
Development Time

Base Parameters of QTP and its Measures



Development time

Functional Requirements

↓

Number of Requirements

↓

Number of defects in realization of Functional requirements

Software Source Code Architecture

↓

Object Oriented Metrics

Level of internal connections of types	• Lack of Cohesion, Lack of Cohesion (Henderson-Sellers, Chidamber & Kemerer, etc)
Level of external connections of types	• Efferent Coupling, Instability, Abstractness, Distance from the Main Sequence
Level of namespaces and packages	• Coupling, Association Between Classes, Afferent Coupling, Relational Cohesion

Source code volume

↓

Statistical metrics (number of lines, classes, methods, packages, namespaces, etc)

Non functional requirements

↓

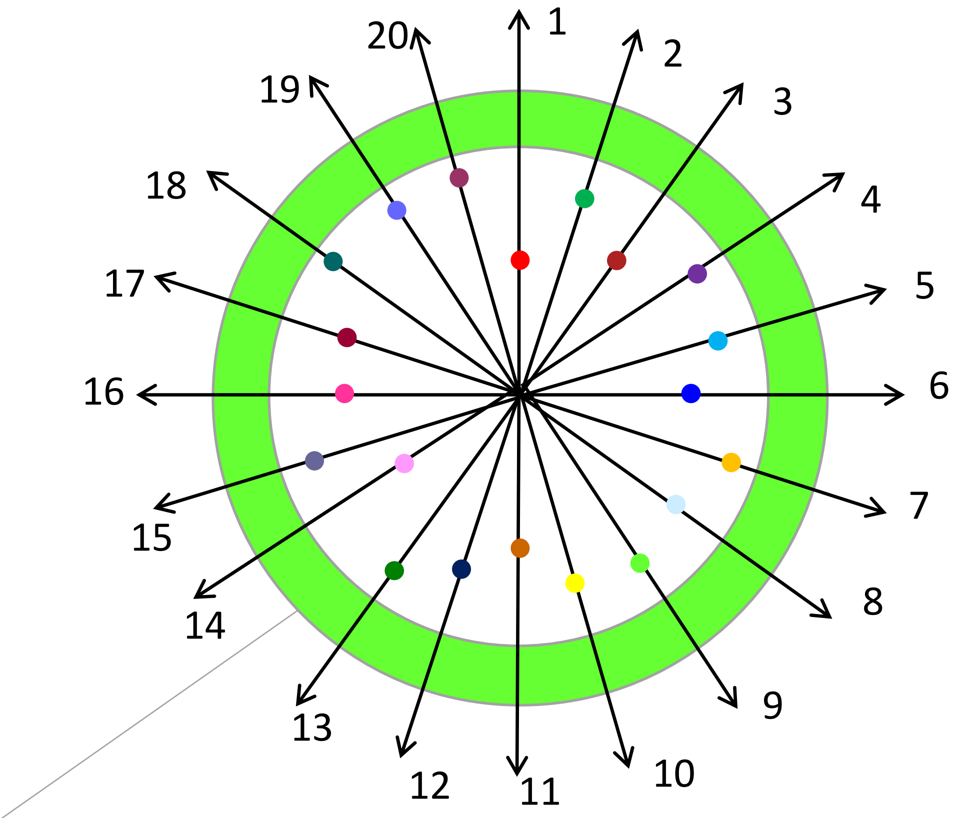
CISQ Code quality measures

- Reliability
- Performance Efficiency
- Maintainability
- Security

N-Dimensional Representation of Quality Target Point

Quality Target Point in N-Dimensional Representation.

Dots on chart is current values of parameters



Green area – area where values of QTP parameters are acceptable or just best if its value equal Radius of Green Circle

Possible list of parameters for Quality Target Point

Object Oriented Metrics

Level of internal connections of types

1. Lack of Cohesion
2. Lack of Cohesion (Henderson-Sellers, Chidamber & Kemerer, etc)

Level of external connections of types

3. Efferent Coupling
4. Instability
5. Abstractness
6. Distance from the Main Sequence

Level of namespaces and packages

7. Coupling
8. Association Between Classes
9. Afferent Coupling
10. Relational Cohesion

CISQ Measures

11. Reliability
12. Performance Efficiency
13. Maintainability
14. Security

Statistical Metrics

15. Number of lines of code
16. Number of methods
17. Number of realized functional requirements
18. Amount of functional defects
19. Mean time between failures
20. …….

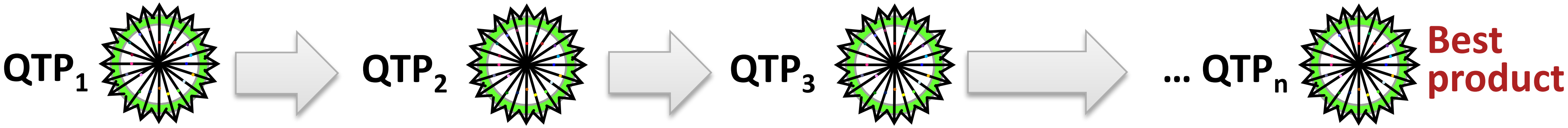
Best Software Product is a Quality Target Point with best values

Best software product is a software product which have final (last) **Quality Target Points** where all parameters equal to their planned values or have acceptable deviations

$$\textbf{Best product} = QTP_n(p_i=v_i,\ p_{i+1}=v_{i+1},\ldots,\ p_j=v_j)$$

Where $\mathbf{p_i}$ – parameter in QTP, $\mathbf{v_i}$-planned value of parameter $\mathbf{p_i}$ on QTP_n, $i \in [1,j]$, j – number of parameters in QTP

Number of parameters not achieved on final QTP and its deviations from planned values show general quality of your product and how far it from ideal/best/good/acceptable state

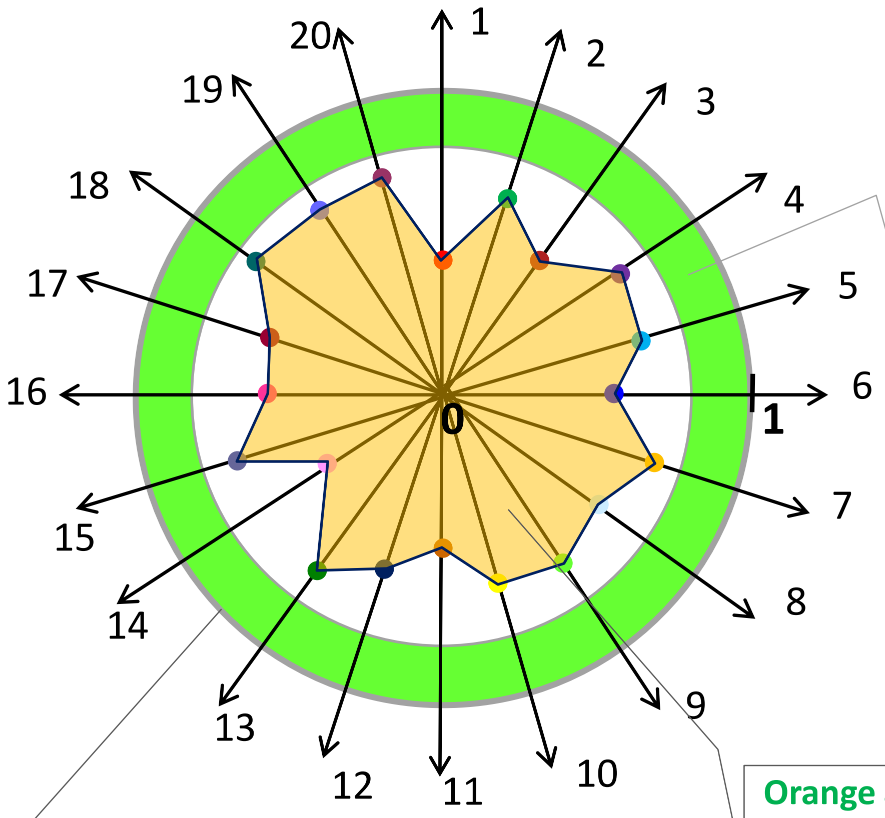


$\mathbf{QTP_1}$ → $\mathbf{QTP_2}$ → $\mathbf{QTP_3}$ → **…** $\mathbf{QTP_n}$ **Best product**

Product Quality Coefficient based on QTP

Quality Target Point in N-Dimensional Representation.

Dots on chart is current values of parameters



Green area – area where values of QTP parameters are acceptable or just best if its value equal to Radius of Green Circle (In our case Radius = 1 basing on our measure system). **S- Square of QTP circle when all parameters have best values. S=3.14 in case if Radius =1**

Grey Circle–QTP circle, maximal value of parameters

Orange area – area covered by current values of QTP parameters. **Square of this area is S_{QTP}**

Important conclusions

1. Lets define ***k*** as Coefficient of Product Quality:

$$k = 1 - \frac{S_{QTP}}{S}$$

2. Value of ***k*** can be calculated basing on square of area defined by values of QTP parameters
3. If ***k=0*** then we have situation:

Best product = $QTP_n(p_i=1, p_{i+1}=1,..., p_j=1)$

4. Than smallest value have ***k*** than better product quality we have and vise versa
5. With this approach basing on time spent for transition from QTP_n to QTP_{n+1} it possible estimate amount of man-hours and cost for this transition. Information about time can be taken from project management system.

Quality Target Point and Testing Maturity Model

- Approaches for software quality depends:
  - From CMMI level because on higher levels company usually works on more difficult projects with appropriate requirements
  - TMMI level depends from CMMI level and define appropriate approaches and requirements to software quality
- Company define own approaches for creation Quality Target Points depending from:
  - CMMI and TMMI levels
  - Requirements for particular project
  - Available time and budget
  - Available tools for measuring parameters for Quality Target Points
  - Available skills in team
  - Parameters in Quality Target Points should varies depending from particular project for providing best quality of particular project

Defining Quality Target Point for project

General rules for using Quality Target Point

- Define **exact list of measurable parameters** which you can measure during software development and testing processes
- Define **OPTIMAL list** of parameters for Quality Target Point
- Define exact goals and values of parameters in every Quality Target Point
- Define how you will measure each parameter in Quality Target Point. If you **can't measure** some parameter **DON'T include it into QTP**
- Define how you will track values of parameters in each Quality Target Point
- Define schedule when you can achieve every Quality Target Point
- Explain to all team members each parameter in your Quality Target Point
- Explain to all team members how to work with tools used for QTP parameters measures and tracking

Base parameters for QTP

1. Statistical Metrics: Number of (application level): packages, namespaces, types, global types, classes, global classes, interfaces, global interfaces, structures, global structures, methods, properties, fields, lines of code, comments, comments density, levels, public data percentage, Halsted complexity, number of parameters in methods, number of functions, overloading

2. Object Oriented Metrics: Coupling, Afferent Coupling, Efferent Coupling, Instability, Relational Cohesion, Distance from the Main Sequence, Abstractness, Association Between Classes, Cohesion of LCOM , Cohesion of LCOM HS , Cyclomatic complexity, Depth Of Inheritance Tree

3. CISQ Measures of Reliability, Performance Efficiency, Maintainability, Security. In total it is defined 86 CISQ Quality Characteristic Measures particularly for Reliability – 29, Performance Efficiency- 15, Maintainability – 20 and Security -22

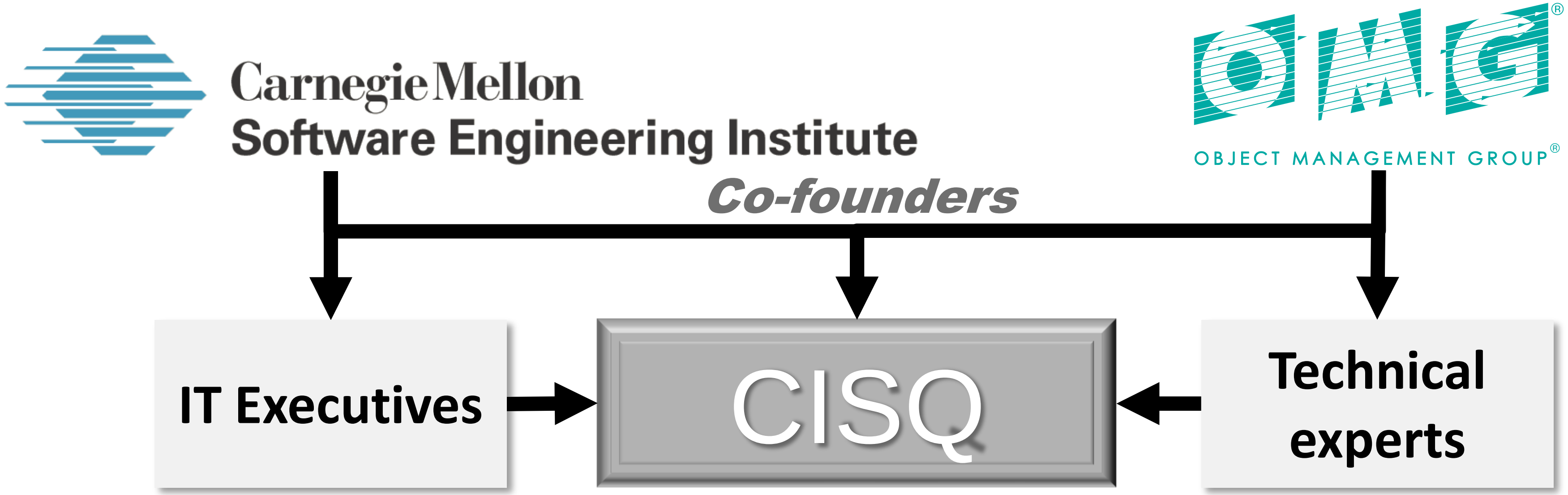
4. Parameters which reflect Functional Software Quality: number of realized features against to total number of features, number of functional defects, number of simultaneously served users, etc.

Object Oriented Metrics

Metrics	Level of software hierarchy organization							Metrics Ranges
	Namespaces and packages	Types	Classes	Methods	Interfaces	Structures	Enumerations	
Coupling	+		+		+	+		**[0,67-1]**
Afferent Coupling	+		+		+	+		**[0,N]**, where N-number of all types in package or namespace
Efferent Coupling	+		+		+	+		**[0,N]**, where N-number of all types outside of package or namespace
Instability	+							**[0,1]**, 0- maximally unstable class/package/namespace, 1- maximally stable class/package/namespace
Relational cohesion	+							**[1.5, 4]** – best value of this metric in this rages
Distance from the Main Sequence	+							**[0,1] ,** 0 – means that package match to main sequence, 1- package as possibly far from main sequence
Abstractness	+	+	+		+	+	+	**[0,1]**, 0 – absolutely concrete package/namespace; 1 – absolutely abstract package/namespace
Cohesion LCOM		+	+	+				**[0, 1]**, LCOM = 0 – absolute cohesion of class, LCOM = 1-no cohesion in class
Cohesion LCOMHS		+	+	+				**[0, 2]**, LCOM HS>1 – bad cohesion
Association between classes		+	+		+	+	+	**[0, ∞]** – than bigger value of this metric than better. This means that class actively used in code

CISQ - Consortium for IT Software Quality

CISQ - Consortium for IT Software Quality (www.it-cisq.org). An IT industry leadership group comprised of IT executives from the Global 2000, system integrators, outsourced service providers, and software technology vendors committed to introducing a computable metrics standard for measuring software quality & size.

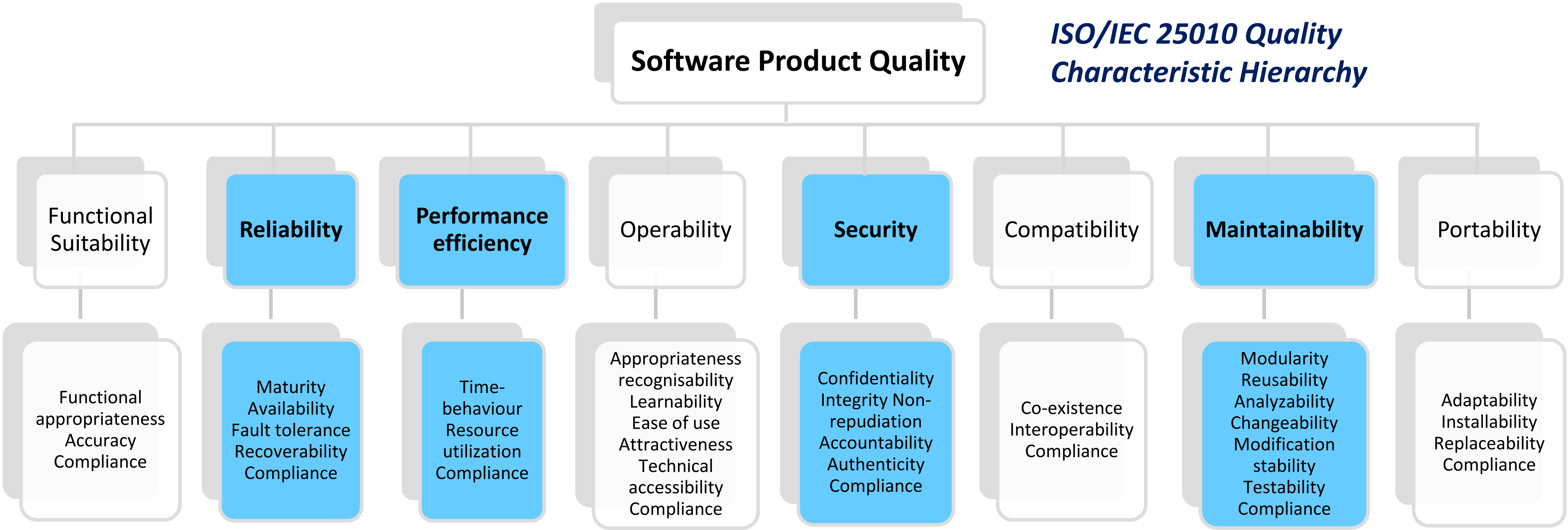


OMG Special Interest Group

CISQ is chartered to define automatable measures of software size and quality that can be measured in the source code, and promote them to become Approved Specifications of the OMG

CISQ defined measures for quality characteristics

ISO/IEC 25010 Quality Characteristic Hierarchy

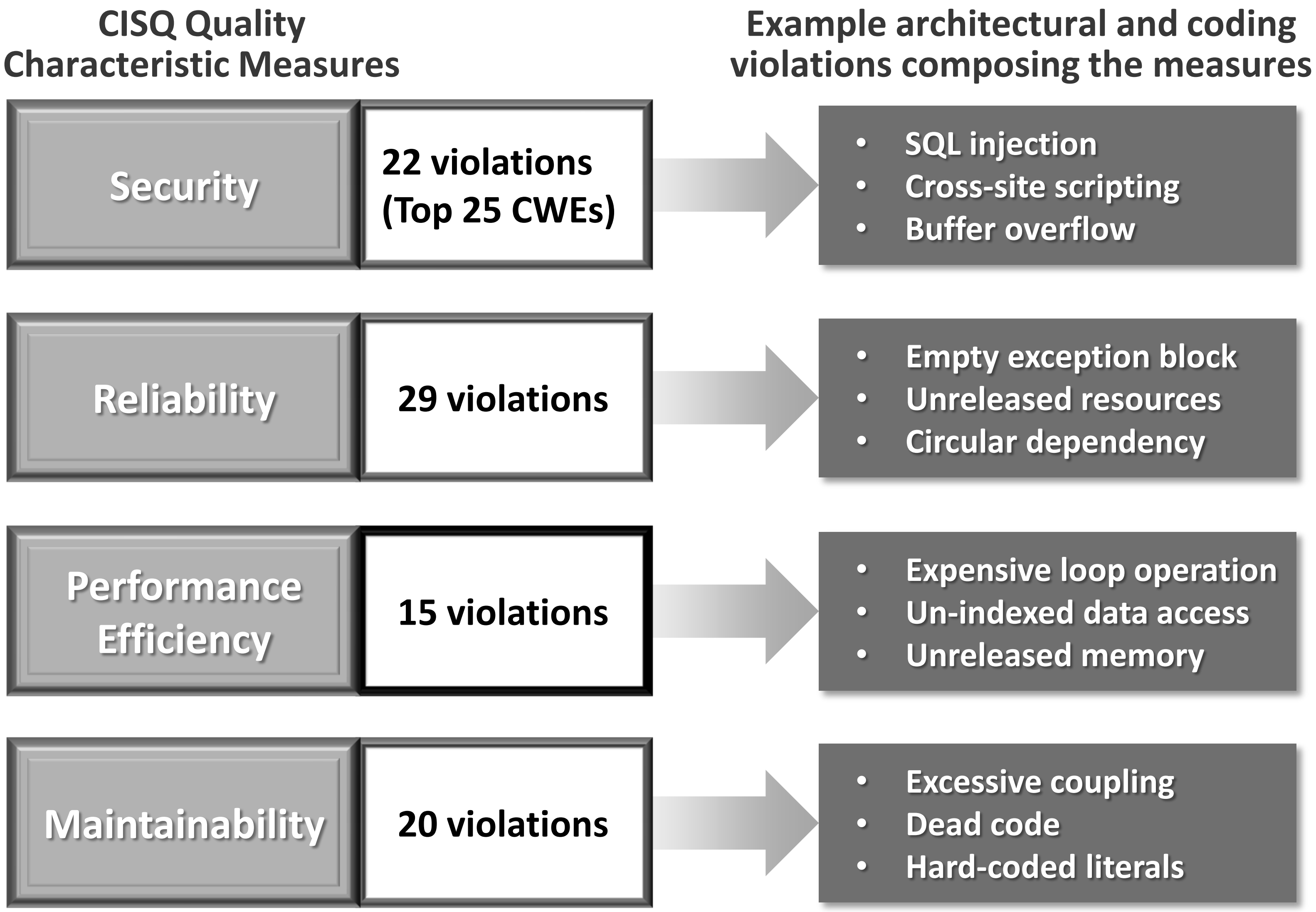


Software Product Quality

- Functional Suitability
  - Functional appropriateness
  - Accuracy
  - Compliance
- **Reliability**
  - Maturity
  - Availability
  - Fault tolerance
  - Recoverability
  - Compliance
- **Performance efficiency**
  - Time-behaviour
  - Resource utilization
  - Compliance
- Operability
  - Appropriateness recognisability
  - Learnability
  - Ease of use
  - Attractiveness
  - Technical accessibility
  - Compliance
- **Security**
  - Confidentiality
  - Integrity
  - Non-repudiation
  - Accountability
  - Authenticity
  - Compliance
- Compatibility
  - Co-existence
  - Interoperability
  - Compliance
- **Maintainability**
  - Modularity
  - Reusability
  - Analyzability
  - Changeability
  - Modification stability
  - Testability
  - Compliance
- Portability
  - Adaptability
  - Installability
  - Replaceability
  - Compliance

CISQ defined automatable measures for quality characteristics highlighted in light blue

CISQ Measures Violations and its relation to ISO



CISQ Quality Characteristic Measures		Example architectural and coding violations composing the measures
Security	22 violations (Top 25 CWEs)	SQL injection Cross-site scripting Buffer overflow
Reliability	29 violations	Empty exception block Unreleased resources Circular dependency
Performance Efficiency	15 violations	Expensive loop operation Un-indexed data access Unreleased memory
Maintainability	20 violations	Excessive coupling Dead code Hard-coded literals

CISQ Measures and its relation to ISO

- ISO 25000 series replaces ISO/IEC 9126 (Parts 1-4)
- ISO 25010 defines quality characteristics and sub-characteristics
- CISQ conforms to ISO 25010 quality characteristic definitions
- ISO 25023 defines measures, but not at the source code level
- CISQ supplements ISO 25023 with source code level measures

Example of CISQ Performance Efficiency Measure Elements

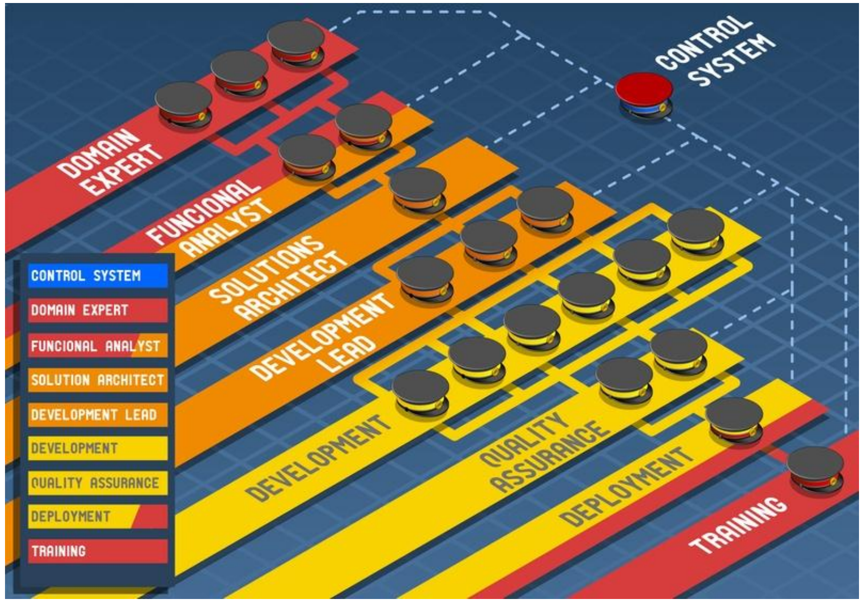
Performance Efficiency Pattern	Consequence	Objective	Measure Element
ASCCPEM-PRF-1: Static Block Element containing Class Instance Creation Control Element	Software that is coded so as to execute expensive computations repeatedly (such as in loops) requires excessive computational resources when the usage and data volume grow	Avoid upfront initialization of software data elements	**Number of instances** where a storable data element or member data element is initialized with a value in the 'Write' action and is located in a block of code which is declared as static
ASCCPEM-PRF-2: Immutable Storable and Member Data Element Creation	Software featuring known underefficient coding practices requires excessive computational resources	Avoid unnecessary usage of additional immutable data elements	**Number of instances** where a named callable control element or method control element creates immutable text data elements via the string concatenation statement (which could be avoided by using text buffer data elements)

Functional Software Quality measures which can be included into QTP

Parameters which reflect Functional Software Quality:

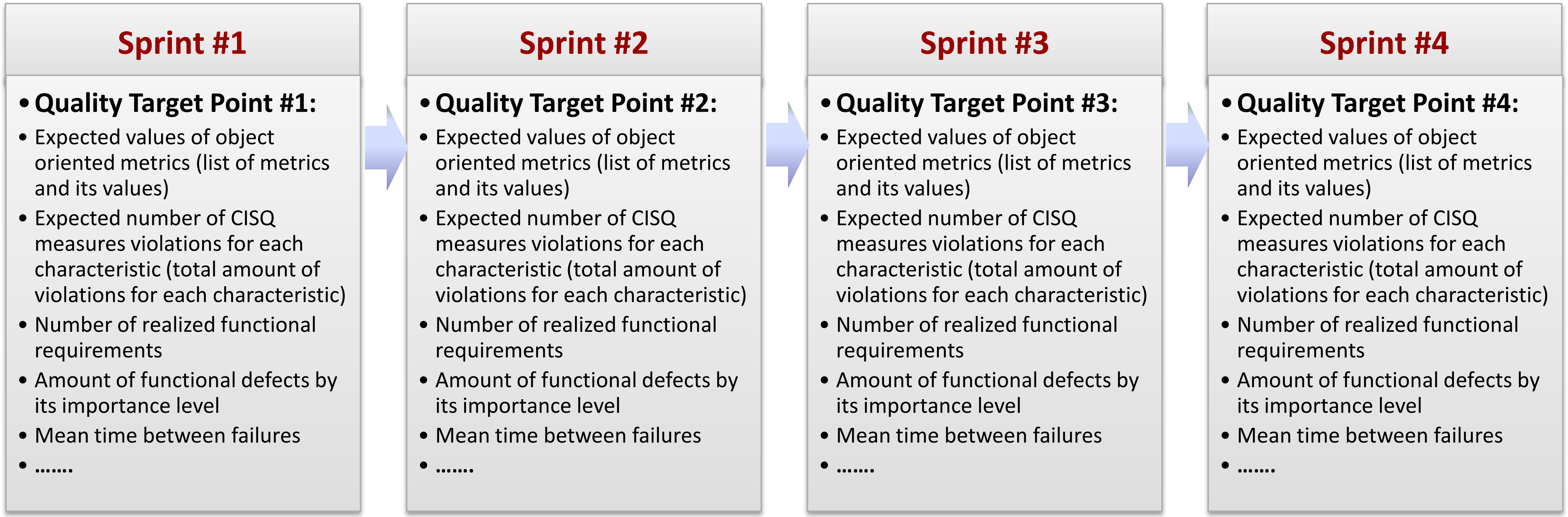
- Number of realized features against to total number of necessary features
- Number of defects which can find team from N testers during some strongly defined time
- Time characteristics for execution of some particular operations/functions
- Number of users served simultaneously in some defined conditions
- Maximal amount of users which can be served simultaneously with appropriate level of quality and user’s experience/expectations
- Mean time between failures
- Some specific parameters with its values from particular standards
- Some specific parameters with its values which should be achieved basing on particular requirements

Software development process with Quality Target Point



Quality Target Point in SCRUM process

- Define for each Sprint its own **Quality Target Point** with exact set of measurable parameters, source code checking schedule and goals which you need to achieve
- Measure and control how you achieving your goals in each **Quality Target Point**
- Depending from results on every **Quality Target Point** which you achieve in reality make changes in next **Quality Target Point**



Sprint #1

- **Quality Target Point #1:**
- Expected values of object oriented metrics (list of metrics and its values)
- Expected number of CISQ measures violations for each characteristic (total amount of violations for each characteristic)
- Number of realized functional requirements
- Amount of functional defects by its importance level
- Mean time between failures
- …….

Sprint #2

- **Quality Target Point #2:**
- Expected values of object oriented metrics (list of metrics and its values)
- Expected number of CISQ measures violations for each characteristic (total amount of violations for each characteristic)
- Number of realized functional requirements
- Amount of functional defects by its importance level
- Mean time between failures
- …….

Sprint #3

- **Quality Target Point #3:**
- Expected values of object oriented metrics (list of metrics and its values)
- Expected number of CISQ measures violations for each characteristic (total amount of violations for each characteristic)
- Number of realized functional requirements
- Amount of functional defects by its importance level
- Mean time between failures
- …….

Sprint #4

- **Quality Target Point #4:**
- Expected values of object oriented metrics (list of metrics and its values)
- Expected number of CISQ measures violations for each characteristic (total amount of violations for each characteristic)
- Number of realized functional requirements
- Amount of functional defects by its importance level
- Mean time between failures
- …….

Integration of Quality Target Point approach into SCRUM process

Before project development start it need to do:

- Prepare your Quality Target Points and define them for every SPRINT
- Create plan for source code quality improvements approaches basing on your needs, abilities and which meet to your requirements
- Prepare optimal set of source code analysis tools: Profilers, Code review tools, Code style checking tools, Code verification tools, Security and vulnerability checking tools, Architecture checking tools OR select some universal tools for code checking, analysis and improvements
- Create implementation plan for using in development of selected tools
- Teach your engineers for using of all necessary tools
- Use all tools constantly during development
- Constantly control and measure parameters in your QTPs
- Introduce into process procedures for code improvements basing on results of source code analysis

Software tools for source code analysis

Select MINIMAL set of tools which will cover measures of all parameters in your QTPs for as possible widest list of your projects

Classes of source code analysis, bugs tracking and quality control tools

- Bug trackers
- Profilers (code optimization)
- Code review tools
- Code checking tools
- Verification tools
- Security and vulnerability
- Architecture
- Tools for calculation source code metrics
- Universal tools for providing large set of tools and features

Practical aspects of Quality Target Point usage

Starting usage approach with Quality Target Point

Conclude all steps for implementation of QTP approach as discussed above. Here most important steps:

1. Define Quality Target Points, its parameters and its values for every Sprint
2. Select tool for parameters measuring and tracking
3. Teach your team for understanding approach with Quality Target Point, tools which will be used, explain to your team QTP measures and parameters

Possible list of parameters for Quality Target Point

Object Oriented Metrics	**Level of internal connections of types**
	1. Lack of Cohesion
	2. Lack of Cohesion (Henderson-Sellers, Chidamber & Kemerer, etc)
	Level of external connections of types
	3. Efferent Coupling
	4. Instability
	5. Abstractness
	6. Distance from the Main Sequence
	Level of namespaces and packages
	7. Coupling
	8. Association Between Classes
	9. Afferent Coupling
	10. Relational Cohesion
CISQ Measures	11. Reliability
	12. Performance Efficiency
	13. Maintainability
	14. Security
Statistical Metrics	15. Number of lines of code
	16. Number of methods
	17. Number of realized functional requirements
	18. Amount of functional defects
	19. Mean time between failures
	20. …….

softarex technologies

codeNforcer is a tool for measuring parameters for QTP

Web based code analysis and code improvements system. System accessible in public or corporate cloud

Source code analysis and improvements

- Supported programming languages: Java. Support for C++, C#, PHP and Objective C coming soon
- Source code checking basing on schedule
- Object Oriented Metrics calculation
- Code convention checking
- Code checking basing on user’s rules
- Source code statistics collection
- Code validation for Reliability, Efficiency, Maintainability, Security improvements basing on CISQ measures and other rules
- Recommendations for source code improvements basing on analysis of Object Oriented Metrics
- Recommendations for source code improvements basing on CISQ measures for Reliability, Efficiency, Maintainability, Security

Team work and integrations

- Creating project’s groups
- Users and Projects management
- Integration with JIRA for interaction on level of users, projects and SCRUM dashboards
- Loading projects, users and statistics necessary for QTP from JIRA
- Web based tools for source code review
- Integration with SVN, GIT and TFS
- Team notifications by email
- Assigning tasks for developers in JIRA for source code improvements basing on generated recommendations

Measurements and Reports

- Source code statistic including weekly and monthly analysis
- Creation, management and tracking of QTPs for projects
- Metrics calculations and their changes dynamics (reflected in QTP)
- Tracking of improvements progress for Performance, Reliability, Maintainability, Security violations

SEND YOUR QUESTIONS TO JAVAONE@SOFTAREX.COM

QTP parameters which can be measured by codeNforcer

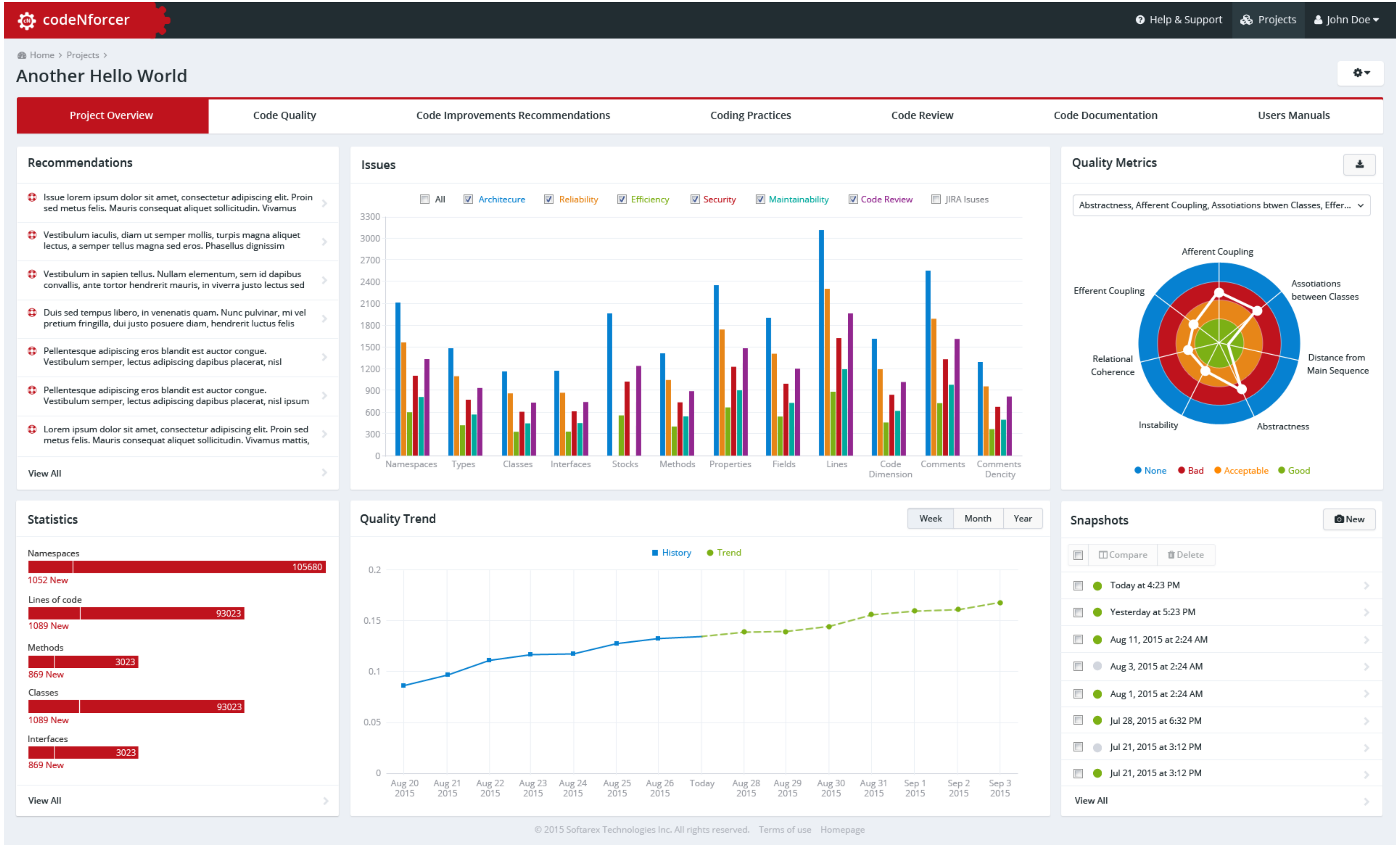
1. Statistical metrics and information about source code

2. Object Oriented Metrics: Coupling, Afferent Coupling, Efferent Coupling, Instability, Relational Cohesion, Distance from the Main Sequence, Abstractness, Association Between Classes, Cohesion of LCOM , Cohesion of LCOM HS, Depth Of Inheritance Tree

3. CISQ Measures of Reliability, Performance Efficiency, Maintainability and Security

CISQ Quality Characteristic Measures	Number of measures defined by CISQ	CISQ measures available in codeNforcer
Security	22	6
Reliability	29	18
Performance Efficiency	15	6
Maintainability	20	14
Total:	**86**	**44**

Project overview in codeNforcer



Another Hello World

Recommendations

- Issue lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sed metus felis. Mauris consequat aliquet sollicitudin. Vivamus
- Vestibulum iaculis, diam ut semper mollis, turpis magna aliquet lectus, a semper tellus magna sed eros. Phasellus dignissim
- Vestibulum in sapien tellus. Nullam elementum, sem id dapibus convallis, ante tortor hendrerit mauris, in viverra justo lectus sed
- Duis sed tempus libero, in venenatis quam. Nunc pulvinar, mi vel pretium fringilla, dui justo posuere diam, hendrerit luctus felis
- Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl
- Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl ipsum
- Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sed metus felis. Mauris consequat aliquet sollicitudin. Vivamus mattis,

View All

Issues

Quality Metrics

Abstractness, Afferent Coupling, Assotiations btwen Classes, Effer...

Statistics

Namespaces
105680
1052 New

Lines of code
93023
1089 New

Methods
3023
869 New

Classes
93023
1089 New

Interfaces
3023
869 New

View All

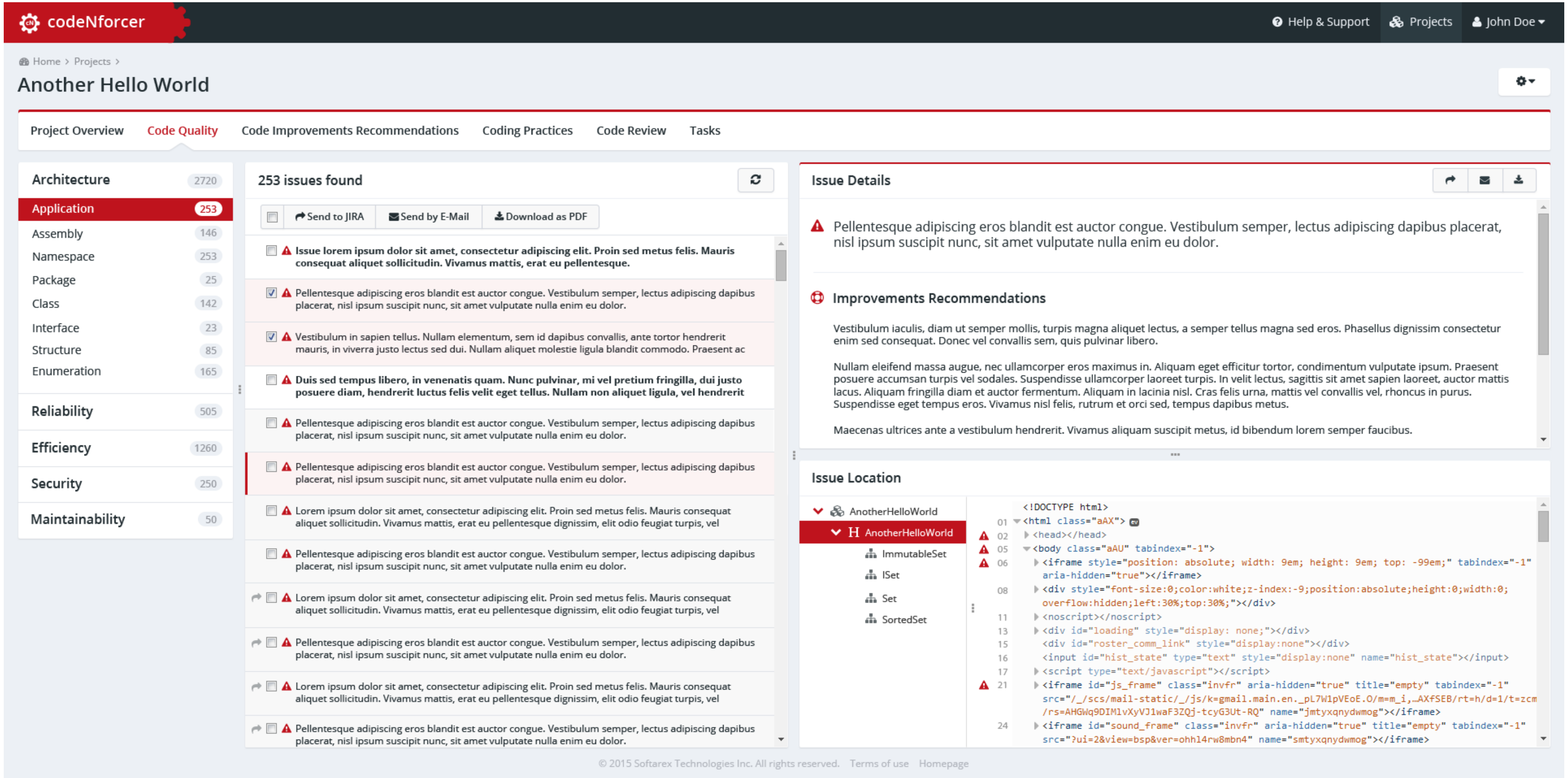
Quality Trend

Snapshots

- □ ● Today at 4:23 PM
- □ ● Yesterday at 5:23 PM
- □ ● Aug 11, 2015 at 2:24 AM
- □ ● Aug 3, 2015 at 2:24 AM
- □ ● Aug 1, 2015 at 2:24 AM
- □ ● Jul 28, 2015 at 6:32 PM
- □ ● Jul 21, 2015 at 3:12 PM
- □ ● Jul 21, 2015 at 3:12 PM

View All

Send tasks into JIRA and by email for improvements of your code



codeNforcer

Help & Support | Projects | John Doe

Home › Projects ›

Another Hello World

Project Overview | Code Quality | Code Improvements Recommendations | Coding Practices | Code Review | Tasks

Architecture	2720
Application	253
Assembly	146
Namespace	253
Package	25
Class	142
Interface	23
Structure	85
Enumeration	165
Reliability	505
Efficiency	1260
Security	250
Maintainability	50

253 issues found

□ Send to JIRA | Send by E-Mail | Download as PDF

- □ **Issue lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sed metus felis. Mauris consequat aliquet sollicitudin. Vivamus mattis, erat eu pellentesque.**
- ☑ Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl ipsum suscipit nunc, sit amet vulputate nulla enim eu dolor.
- ☑ Vestibulum in sapien tellus. Nullam elementum, sem id dapibus convallis, ante tortor hendrerit mauris, in viverra justo lectus sed dui. Nullam aliquet molestie ligula blandit commodo. Praesent ac
- □ **Duis sed tempus libero, in venenatis quam. Nunc pulvinar, mi vel pretium fringilla, dui justo posuere diam, hendrerit luctus felis velit eget tellus. Nullam non aliquet ligula, vel hendrerit**
- □ Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl ipsum suscipit nunc, sit amet vulputate nulla enim eu dolor.
- □ Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl ipsum suscipit nunc, sit amet vulputate nulla enim eu dolor.
- □ Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sed metus felis. Mauris consequat aliquet sollicitudin. Vivamus mattis, erat eu pellentesque dignissim, elit odio feugiat turpis, vel
- □ Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl ipsum suscipit nunc, sit amet vulputate nulla enim eu dolor.
- □ Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sed metus felis. Mauris consequat aliquet sollicitudin. Vivamus mattis, erat eu pellentesque dignissim, elit odio feugiat turpis, vel
- □ Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl ipsum suscipit nunc, sit amet vulputate nulla enim eu dolor.
- □ Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin sed metus felis. Mauris consequat aliquet sollicitudin. Vivamus mattis, erat eu pellentesque dignissim, elit odio feugiat turpis, vel
- □ Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl ipsum suscipit nunc, sit amet vulputate nulla enim eu dolor.

Issue Details

Pellentesque adipiscing eros blandit est auctor congue. Vestibulum semper, lectus adipiscing dapibus placerat, nisl ipsum suscipit nunc, sit amet vulputate nulla enim eu dolor.

Improvements Recommendations

Vestibulum iaculis, diam ut semper mollis, turpis magna aliquet lectus, a semper tellus magna sed eros. Phasellus dignissim consectetur enim sed consequat. Donec vel convallis sem, quis pulvinar libero.

Nullam eleifend massa augue, nec ullamcorper eros maximus in. Aliquam eget efficitur tortor, condimentum vulputate ipsum. Praesent posuere accumsan turpis vel sodales. Suspendisse ullamcorper laoreet turpis. In velit lectus, sagittis sit amet sapien laoreet, auctor mattis lacus. Aliquam fringilla diam et auctor fermentum. Aliquam in lacinia nisl. Cras felis urna, mattis vel convallis vel, rhoncus in purus. Suspendisse eget tempus eros. Vivamus nisl felis, rutrum et orci sed, tempus dapibus metus.

Maecenas ultrices ante a vestibulum hendrerit. Vivamus aliquam suscipit metus, id bibendum lorem semper faucibus.

Issue Location

- AnotherHelloWorld
  - AnotherHelloWorld
    - ImmutableSet
    - ISet
    - Set
    - SortedSet

```
<!DOCTYPE html>
<html class="aAX">
 <head></head>
 <body class="aAU" tabindex="-1">
  <iframe style="position: absolute; width: 9em; height: 9em; top: -99em;" tabindex="-1"
  aria-hidden="true"></iframe>
  <div style="font-size:0;color:white;z-index:-9;position:absolute;height:0;width:0;
  overflow:hidden;left:30%;top:30%;"></div>
  <noscript></noscript>
  <div id="loading" style="display: none;"></div>
  <div id="roster_comm_link" style="display:none"></div>
  <input id="hist_state" type="text" style="display:none" name="hist_state"></input>
  <script type="text/javascript"></script>
  <iframe id="js_frame" class="invfr" aria-hidden="true" title="empty" tabindex="-1"
  src="/_/scs/mail-static/_/js/k=gmail.main.en._pL7W1pVEoE.O/m=m_i,…AXfSEB/rt=h/d=1/t=zcm
  /rs=AHGWq9DIMlvXyVJ1waF3ZQj-tcyG3Ut-RQ" name="jmtyxqnydwmog"></iframe>
  <iframe id="sound_frame" class="invfr" aria-hidden="true" title="empty" tabindex="-1"
  src="?ui=2&view=bsp&ver=ohhl4rw8mbn4" name="smtyxqnydwmog"></iframe>
```

Terms of use Homepage

softarex technologies

How to start with codeNforcer

Try and use online www.codenforcer.com

Live demo and Test Drive at HOL | Session ID: HOL10289

Session Title: Using Automation Tools for Code Quality Improvements for Java Applications

Venue / Room: Hilton—Franciscan Room A

Date and Time: 10/27/15, 16:00 - 17:00, 17:30 - 18:30

cN codeNforcer.com

Conclusion

Conclusion

- **Achieving of Quality Target Point means that your software have as possibly best source code quality and provide all required functionality with all necessary non - functional requirements**
- **Best software product** is a software product which have final (last) **Quality Target Points** where all parameters equal to their planned values or have acceptable deviations
- **Product Quality Coefficient** based on QTP allow measure quality of your software product by one number. If this coefficient **equal to 0 or close to 0** then this means that system have very small amount of defects on all levels
- **Plan your QTPs before project's start and explain its to your team**
- **Select MINIMAL set of tools which will cover measures of all parameters in your QTPs on as possible widest list of your projects**

Appreciations

My family which inspire me to new achievements

All team of Softarex Technologies Inc which help me every day to achieve our aims

Thank You for Your Attention

Softarex Technologies, Inc.

Headquarters

901 N. Pitt Street, Suite 320
Alexandria, VA 22314, USA
Tel: +1 (703) 836 18 60
E-mail: info@softarex.com

SEND YOUR QUESTIONS TO JAVAONE@SOFTAREX.COM