

# AD Dashboard

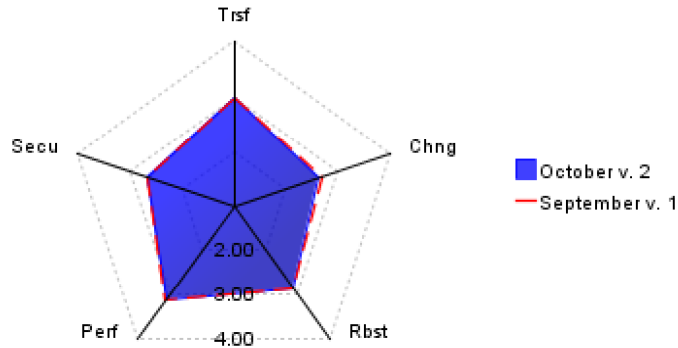
## Report on Application: Trading

This document reports on the selected context - including focus on its constituent contexts and technologies - and is designed to be printed as a summary report.

## Health Factors for Trading in October v. 2 [2]snapshot

Assessment of the quality of the selected component. Click on the hyperlinks below to see the application risk factors for the current context - component and snapshot -.

Transferability	<b>2.96</b>
Changeability	<b>2.63</b>
Robustness	<b>2.85</b>
Performance	<b>3.1</b>
Security	<b>2.64</b>

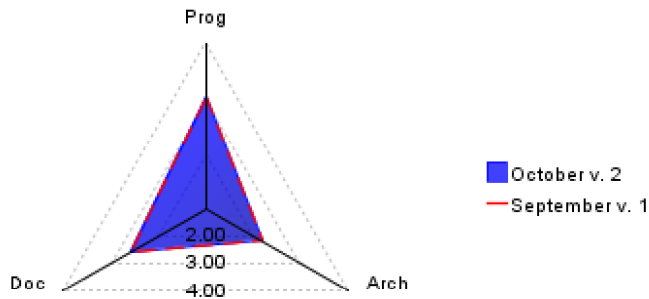


SEI Maintainability	<b>3.23</b>	3.23
Technical Quality Index	<b>2.85</b>	2.85

Technical Quality and SEI Maintainability assess the cost and difficulty/ease to maintain an application in the future. Technical Quality is based on 100s of metrics about the source code. SEI Maintainability is based on statistics about 100s of development projects.

## Rule Compliance for Trading in October v. 2 [2]snapshot

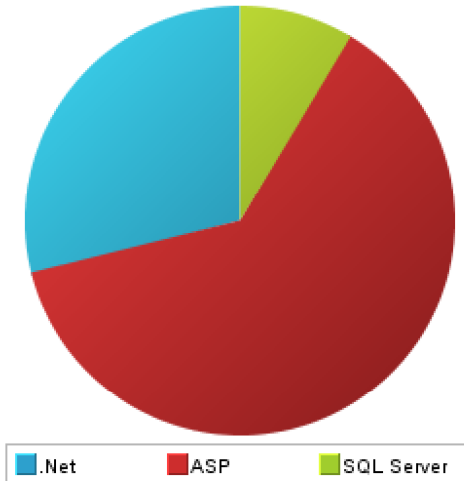
Assessment of the compliance to rules for the selected component. Click on the hyperlinks below to drilldown on rule compliance information for the current context - component and snapshot -.



Programming Practices	<b>3.02</b>
Architectural Design	<b>2.2</b>
Documentation	<b>2.56</b>

Quantity of TradingInOctober v. 2 [2]snapshot

Assessment of the technical structure and size of the selected component. Graphical display of the contribution (to the Back-fired Function Points total) of the 20 largest sub-components. Click on the hyperlinks below for more detailed assessment.



Quantity Metrics Summary	Metric Value
Technical Size	
Number of Code Lines	<b>147,422</b>
Number of Files	<b>422</b>
Number of Classes	<b>398</b>
Number of Programs	<b>0</b>
Number of Forms	<b>1</b>
Number of SQL Artifacts	<b>614</b>
Number of Tables	<b>135</b>
Functional Weight	
Backfired IFPUG Function Points	<b>2,361.24</b>
Automated IFPUG Function Points Estimation	<b>1,013</b>
Number of Decision Points	<b>26,500</b>

Robustness focus

Robustness measures the level of risk and the likelihood of having application failures and application defects due to modifications. Robustness measures as well the level of effort necessary to test the application

Child Metric Name	Child Metric Status	Child Metric Grade
Programming Practices - Error and Exception Handling	<b>High Risk</b>	2.87
Architecture - Object-level Dependencies	<b>Moderate Risk</b>	3.25
Complexity - Algorithmic and Control Structure Complexity	<b>Moderate Risk</b>	3.39
Architecture - Multi-Layers and Data Access	<b>Very High Risk</b>	1.35
Complexity - SQL Queries	<b>Moderate Risk</b>	3.35
Complexity - Technical Complexity	<b>High Risk</b>	2.93
Architecture - Reuse	<b>Very High Risk</b>	1.77
Complexity - OO Inheritance and Polymorphism	<b>Moderate Risk</b>	3.72
Architecture - OS and Platform Independence	<b>Moderate Risk</b>	3.98
Programming Practices - OO Inheritance and Polymorphism	<b>Low Risk</b>	4
Programming Practices - Structuredness	<b>Very High Risk</b>	1.82
Dead code (static)	<b>High Risk</b>	2.53
Volume - Number of Components	<b>High Risk</b>	2.96

⌵ Performance focus

---

Performance measures the likelihood of potential performance bottlenecks and the potential future scalability issues linked to coding practices.

Child Metric Name	Child Metric Status	Child Metric Grade
Efficiency - Expensive Calls in Loops	<b>High Risk</b>	2.66
Efficiency - Memory, Network and Disk Space Management	<b>Moderate Risk</b>	3.58
Efficiency - SQL and Data Handling Performance	<b>High Risk</b>	2.98
Complexity - SQL Queries	<b>Moderate Risk</b>	3.35

⌵ Security focus

---

Security measures the likelihood of potential security breaches linked to coding practices and application source code.

Child Metric Name	Child Metric Status	Child Metric Grade
Programming Practices - Error and Exception Handling	<b>High Risk</b>	2.87
Secure Coding - Input Validation	<b>Low Risk</b>	4
Architecture - Multi-Layers and Data Access	<b>Very High Risk</b>	1.35
Secure Coding - Encapsulation	<b>Moderate Risk</b>	3.25
Efficiency - Memory, Network and Disk Space Management	<b>Moderate Risk</b>	3.58
Architecture - OS and Platform Independence	<b>Moderate Risk</b>	3.98

Transferability focus

Transferability measures how easily applications can be moved across teams or team members including in-house and outsourced development teams.

Child Metric Name	Child Metric Status	Child Metric Grade
Complexity - Algorithmic and Control Structure Complexity	Moderate Risk	3.39
Documentation - Volume of Comments	High Risk	2.08
Complexity - SQL Queries	Moderate Risk	3.35
Documentation - Naming Convention Conformity	Moderate Risk	3.37
Documentation - Bad Comments	Moderate Risk	3.55
Documentation - Style Conformity	Very High Risk	1.33
Complexity - OO Inheritance and Polymorphism	Moderate Risk	3.72
Programming Practices - Structuredness	Very High Risk	1.82
Dead code (static)	High Risk	2.53
Volume - Number of Components	High Risk	2.96
Architecture - Object-level Dependencies	Moderate Risk	3.25
Volume - Number of LOC	Moderate Risk	3.4
Programming Practices - OO Inheritance and Polymorphism	Low Risk	4
Programming Practices - File Organization Conformity	Moderate Risk	3.98

☑ Changeability focus

Changeability measures how easily applications can be modified in order to implement new features, correct errors, or change the applications environment.

Child Metric Name	Child Metric Status	Child Metric Grade
Architecture - Multi-Layers and Data Access	Very High Risk	1.35
Programming Practices - Modularity and OO Encapsulation Conformity	Very High Risk	1.98
Programming Practices - Structuredness	Very High Risk	1.82
Architecture - Object-level Dependencies	Moderate Risk	3.25
Complexity - Functional Evolvability	Moderate Risk	3.79
Complexity - SQL Queries	Moderate Risk	3.35
Documentation - Volume of Comments	High Risk	2.08
Documentation - Naming Convention Conformity	Moderate Risk	3.37
Complexity - Algorithmic and Control Structure Complexity	Moderate Risk	3.39
Complexity - OO Inheritance and Polymorphism	Moderate Risk	3.72
Architecture - OS and Platform Independence	Moderate Risk	3.98
Dead code (static)	High Risk	2.53

📄 Programming Practices focus

Programming Practices

Child Metric Name	Child Metric Status	Child Metric Grade
Programming Practices - Structuredness	Very High Risk	1.82
Programming Practices - Modularity and OO Encapsulation Conformity	Very High Risk	1.98
Dead code (static)	High Risk	2.53
Efficiency - Expensive Calls in Loops	High Risk	2.66
Programming Practices - Error and Exception Handling	High Risk	2.87
Complexity - Technical Complexity	High Risk	2.93
Volume - Number of Components	High Risk	2.96
Efficiency - SQL and Data Handling Performance	High Risk	2.98
Secure Coding - Encapsulation	Moderate Risk	3.25
Complexity - SQL Queries	Moderate Risk	3.35
Complexity - Algorithmic and Control Structure Complexity	Moderate Risk	3.39
Volume - Number of LOC	Moderate Risk	3.4
Efficiency - Memory, Network and Disk Space Management	Moderate Risk	3.58
Complexity - OO Inheritance and Polymorphism	Moderate Risk	3.72
Programming Practices - File Organization Conformity	Moderate Risk	3.98
Programming Practices - OO Inheritance and Polymorphism	Low Risk	4
Secure Coding - Input Validation	Low Risk	4

⌵ Architectural Design focus

---

Architectural Design

Child Metric Name	Child Metric Status	Child Metric Grade
Architecture - Multi-Layers and Data Access	Very High Risk	1.35
Architecture - Reuse	Very High Risk	1.77
Architecture - Object-level Dependencies	Moderate Risk	3.25
Architecture - OS and Platform Independence	Moderate Risk	3.98

⌵ Documentation focus

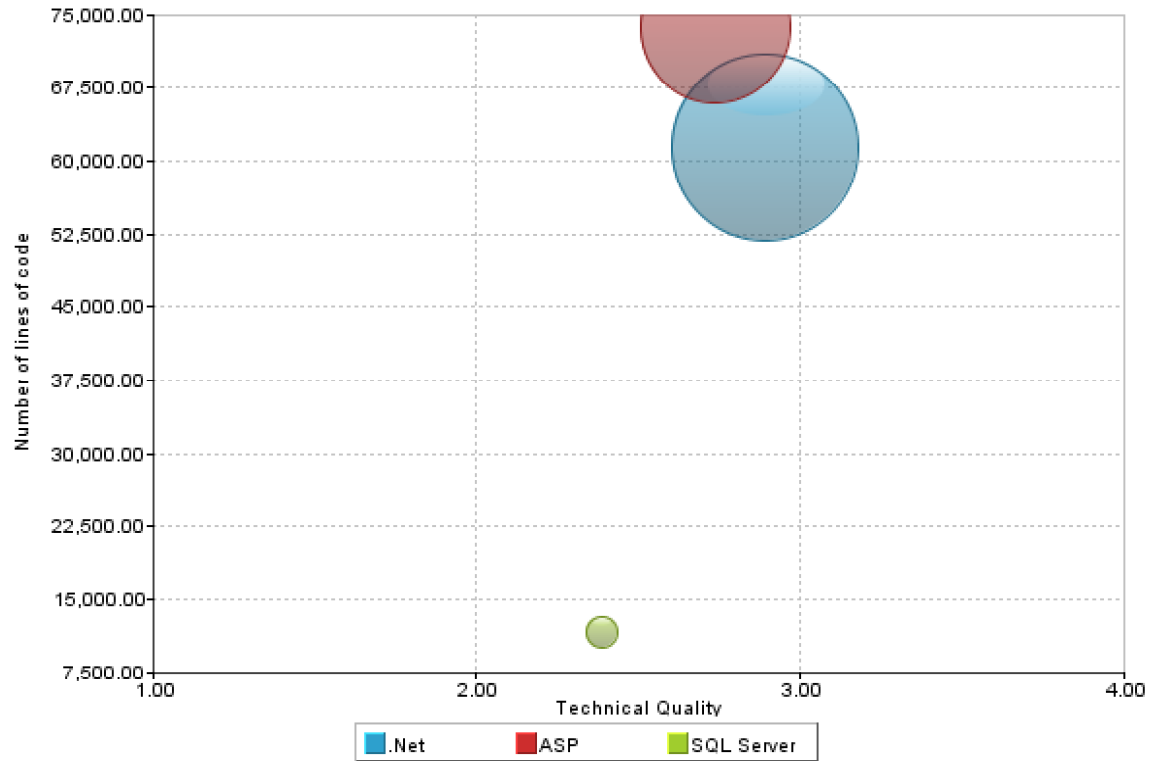
---

Documentation

Child Metric Name	Child Metric Status	Child Metric Grade
Documentation - Style Conformity	Very High Risk	1.33
Documentation - Volume of Comments	High Risk	2.08
Documentation - Naming Convention Conformity	Moderate Risk	3.37
Documentation - Bad Comments	Moderate Risk	3.55



Mapping of 'Technical Quality'x'Number of lines of code'x'Number of artifacts'



Object/Context	Technical Quality	Variation	
.Net	High Risk	➡	2.89
ASP	High Risk	➡	2.75
SQL Server	High Risk	➡	2.39

⌵ This section lists all violated rules in the current context, along with the number of non-compliant objects.

Violated Rules


	Diagnostic-based Metric Name	Number of Defective Objects	Metric Description
⌵ ASP	Diagnostic-based Metric relative to ASP language	7,024	7,024ASP Objects are considered as defects
	ASP: Avoid JavaScript Functions having a very low Comment/Code ratio	1,954	JavaScript Functions should have at least a ratio comment/code > X % The threshold is a parameter and can be changed at will.
	ASP: Avoid undocumented JavaScript Functions	1,878	JavaScript Functions should have comments
	ASP: Avoid unreferenced JavaScript Functions	1,298	All JavaScript Functions should be referenced. An unreferenced Artifact is an Artifact that is not explicitly called from within the analyzed code.
	ASP: Avoid Artifacts with lines longer than 80 characters	946	Avoid Artifacts with lines longer than 80 characters
	ASP: Avoid Artifacts with High Fan-Out	120	Avoid Artifacts with High Fan-Out (Fan-Out > X). The Fan-out of an Artifact is the number of other Artifacts that are referenced in it. When computing the Fan-Out of an Artifact, multiple accesses to the same component of an Artifact are counted as one access. The threshold is a parameter and can be changed at will.
	ASP: Avoid unreferenced code	114	All code should be referenced. An unreferenced Artifact is an Artifact that is not explicitly called from within the analyzed code.
	ASP: Avoid Artifacts with High Cyclomatic Complexity	97	Avoid Artifacts with High Cyclomatic Complexity (CC greater than X) Cyclomatic Complexity is a measure of the complexity of the control structure of an Artifact. It is the number of linearly independent paths and therefore, the minimum number of independent paths when executing the software. The threshold is a parameter and can be changed at will.
	ASP: Avoid Artifacts with High Fan-In	96	Avoid Artifacts with High Fan-In (Fan-In > X). The Fan-In of an Artifact is the number of other Artifacts that are referencing it. When computing the Fan-In of an Artifact, multiple accesses to it from the same Artifact are counted as one access. The threshold level is a parameter that can be changed at will.
	ASP: Use of style sheets	89	Style sheet (*.css) should be used (instead of no css or hardcoded style definition like style= )
	ASP: Use of error handling page	88	Pages should use error handling page
	ASP: Avoid Artifacts with High Integration Complexity	70	Avoid Artifacts with High Integration Complexity (IC greater than X). Integration Complexity measures the number of independent integration paths. Integration paths are paths of the control flow graph in which another object is invoked. The threshold is a parameter and can be changed at will.
	ASP: Avoid Artifacts with High Depth of Code	64	Depth of Code is measured as the maximum number of nested control statements in an artifact. For example, an artifact that contains an IF statement which contains a While loop which itself contains another IF statement will have a Depth of Code of 3 (at least). Avoid Artifacts with Depth of Code (DoC) greater than X. The threshold level is a parameter that can be changed at will.
	ASP: Avoid ASP pages having a very low Comment/Code ratio	64	Pages should have at least a ratio comment/code > X % The threshold is a parameter and can be changed at will.
	ASP: Avoid Artifacts with High Essential Complexity	55	Avoid Artifacts with High Essential Complexity (EC greater than X). Essential Complexity measures the number of non-structured independent paths. Non-structured paths are paths of the control flow graph in which an instruction that interrupt the flow is present.

			The threshold is a parameter and can be changed at will.
	ASP: Avoid direct access to database Tables	43	Pages should not access directly database Tables
	ASP: Avoid direct use of database objects	25	Pages should not directly use Database Objects
	ASP: Avoid undocumented Pages	15	Pages should have comments
	ASP: Index pages and global.asa location in the root directory	4	Index pages and global.asa must be located in the root directory
	ASP: Avoid non standard file extensions	3	All files should have a standard file extensions . The list of standard, authorized extensions is a parameter that can be changed at will.
	ASP: Avoid large Page files	1	Pages should not have more than X lines of code. The threshold is a parameter and can be changed at will.
SQL Server	Diagnostic-based Metric relative to SQL Server language	3,085	3,085 SQL Server Objects are considered as defects
	T-SQL: Stored Procedure naming convention - prefix control	456	Names of Procedures should start with X. The prefix value is a parameter that can be changed at will
	T-SQL: Avoid Stored Procedures not returning a status value	412	Avoid Stored Procedures not returning a status value (RETURN)
	T-SQL: Avoid Artifacts with lines longer than 80 characters	298	Avoid Artifacts with lines longer than 80 characters
	T-SQL: Avoid Functions and Procedures doing an Insert, Update or Delete without including error management	294	Avoid stored procedures or functions doing Insert, Update, Delete, Select or Create Table and not including error management (check @@error variable),
	T-SQL: Avoid functions and procedures with a very low comment/code ratio	268	Functions and Procedures should have at least a ratio comment / code >= X The threshold is a parameter and can be changed at will.
	T-SQL: Avoid undocumented functions and procedures	240	Functions and procedures should have comments
	T-SQL: Avoid Functions and Procedures doing an Insert, Update or Delete without managing a transaction	143	Avoid Stored Procedures doing an Insert, Update or Delete and not managing a transaction (execute "begin tran" if @@trancount=0)
	T-SQL: Table naming convention - prefix control	135	Names of Tables should start with X. The prefix value is a parameter that can be changed at will
	T-SQL: Avoid tables not involved in a Foreign Key	111	List of all tables not involved in a Foreign Key (FK)
	T-SQL: Avoid tables without a clustered index	94	List all tables that do not have a clustered index
	T-SQL: Avoid long Table names	90	List of database tables that have more than X characters. The length is a parameter that can be changed at will.
	T-SQL: Avoid Functions / Procedures with a complex SELECT clause	90	Avoid Functions / Procedures with a SELECT clause returning more than 9 columns or with a 'SELECT *' query. Such queries are considered complex. Changing threshold value requires a Metric Assistant configuration update.
	T-SQL: Avoid Tables access directly from client-side SQL queries	69	Avoid Tables accessed directly from client-side SQL queries
	T-SQL: Avoid Artifacts with too many parameters	65	Avoid artifacts with more than X parameters. The threshold X is a parameter and it can be changed at will.
	T-SQL: Avoid Functions / Procedures with SQL statement including Subqueries	57	Functions / Procedures should not use SQL statement including Subqueries (at least subqueries should be avoided).
	T-SQL: Avoid Artifacts with High Fan-In	51	Avoid Artifacts with High Fan-In (Fan-In > X) The Fan-In of an Artifact is the number of other Artifacts that are referencing it. When computing the Fan-In of an Artifact, multiple accesses to it from the same Artifact are counted as one access. The threshold level is a parameter that can be changed at will.
	T-SQL: Avoid Functions/Procedures with SQL statement using Group By clause	39	Functions / Procedures should not use Group By clause in SQL statement

	T-SQL: View naming convention - prefix control	23	Names of Views should start with X. The prefix value is a parameter that can be changed at will
	T-SQL: Avoid using Cursors	22	Avoid using Cursors
	T-SQL: Avoid long View names	20	List of database views that have more than X characters. The length is a parameter that can be changed at will.
	T-SQL: Avoid use of "truncate table"	19	Avoid using "truncate table",
	T-SQL: Avoid Artifacts with High Cyclomatic Complexity	17	Avoid Artifacts with High Cyclomatic Complexity (CC greater than X) ) Cyclomatic Complexity is a measure of the complexity of the control structure of an Artifact. It is the number of linearly independent paths and therefore, the minimum number of independent paths when executing the software. The threshold is a parameter and can be changed at will.
	T-SQL: Avoid Functions / Procedures with queries on too many Tables	16	Avoid Functions / Procedures with queries on more than 4 Tables. Queries with more than 4 Tables is considered complex. Changing the threshold value requires Metric Assistant configuration update.
	T-SQL: Avoid Tables with more than 20 columns on an OLTP system	15	Avoid Tables with more than 20 columns on an OLTP system
	T-SQL: Avoid Artifacts with High Fan-Out	11	Avoid Artifacts with High Fan-Out (Fan-Out > X). The Fan-out of an Artifact is the number of other Artifacts that are referenced in it. When computing the Fan-Out of an Artifact, multiple accesses to the same component of an Artifact are counted as one access. The threshold is a parameter and can be changed at will.
	T-SQL: Avoid using temporary Objects	9	Triggers, Views, Functions and Procedures should not use temporary Objects (except temporary Tables)
	T-SQL: Avoid Artifacts with High Integration Complexity	8	Avoid Artifacts with High Integration Complexity (IC greater than X) ) Integration Complexity measures the number of independent integration paths. Integration paths are paths of the control flow graph in which another object is invoked. The threshold is a parameter and can be changed at will. Artifacts are
	T-SQL: Avoid Artifacts with High Depth of Code	5	Depth of Code is measured as the maximum number of nested control statements in an artifact. For example, an artifact that contains an IF statement which contains a While loop which itself contains another IF statement will have a Depth of Code of 3 (at least). Avoid Artifacts with Depth of Code (DoC) greater than X. The threshold level is a parameter that can be changed at will.
	T-SQL: Avoid Functions / Procedures with High RAW SQL Complexity	5	Avoid Functions / Procedures with High RAW SQL Complexity (SQL Complexity greater than X). RAW SQL Complexity measures the total number of tables used in FROM Clauses (Note: a single table used X times counts X). The threshold is a parameter and can be changed at will.
	T-SQL: Avoid nested Stored Procedures using temporary Tables	2	Nested Stored Procedures should not use temporary Tables
	T-SQL: Avoid Artifacts with High Essential Complexity	1	Avoid Artifacts with High Essential Complexity (EC greater than X) ) Essential Complexity measures the number of non-structured independent paths. Non-structured paths are paths of the control flow graph in which an instruction that interrupt the flow is present. The threshold is a parameter and can be changed at will.

▼ .Net	Diagnostic-based Metric relative to .Net language	14,158	14,158.Net Objects are considered as defects
	.NET: Avoid Methods with a very low comment/code ratio	2,431	Methods should have at least a ratio comment/code > X % The threshold is a parameter and can be changed at will.
	.NET: Avoid uncommented Methods	2,323	Methods should have comments
	.NET: Avoid Artifacts with lines longer than 80 characters	1,516	Avoid Artifacts with lines longer than 80 characters
	.NET: Avoid Artifacts with High Fan-Out	1,246	Avoid Artifacts with High Fan-Out (Fan-Out > X). The Fan-out of an Artifact is the number of other Artifacts that are referenced in it. When computing the Fan-Out of an Artifact, multiple accesses to the same component of an Artifact are counted as one access. The threshold is a parameter and can be changed at will.
	.NET: Avoid unreferenced Methods	1,043	All Methods should be referenced. An unreferenced Artifact is an Artifact that is not explicitly called from within the analyzed code.
	.NET: Avoid catching an exception of type Exception	925	This metric reports all methods catching an exception of type Exception. The exception Exception should never be caught in a catch statement.
	.NET: Avoid empty finally blocks	545	This metric reports all methods having at least one empty finally block (empty or have only comments). In a try and catch/finally statement, finally blocks should contain code to handle the thrown exception.
	ADO.NET: Avoid using untyped DataSet	489	Avoid using Untyped DataSet. This also apply to DataTables and DataRows.
	.NET: Avoid empty catch blocks	429	This metric reports all methods having at least one empty catch block (empty or have only comments). In a try and Catch statement, Catch blocks should have code to handle the thrown exception. If they are empty or have only comments, the Exception will not be handled.
	.NET: Avoid Artifacts with High Essential Complexity	407	Avoid Artifacts with High Essential Complexity (EC greater than X ). Essential Complexity measures the number of non-structured independent paths. Non-structured paths are paths of the control flow graph in which an instruction that interrupt the flow is present. The threshold is a parameter and can be changed at will.
	.NET: Avoid Artifacts with High Fan-In	286	Avoid Artifacts with High Fan-In (Fan-In > X) The Fan-In of an Artifact is the number of other Artifacts that are referencing it. When computing the Fan-In of an Artifact, multiple accesses to it from the same Artifact are counted as one access. The threshold level is a parameter that can be changed at will. Artifacts are ...
	.NET: Methods naming convention - case and character set control	219	Names of Methods should start with an uppercase character (except set/get/op) and should not include any underscore
	.NET: Avoid Classes with a High Lack of Cohesion	214	Avoid Classes with High Lack of Cohesion in Methods (LCOM > X) LCOM is an indicator of a Class whose methods access few and different of its fields. The threshold is a parameter and can be changed at will.
	.NET: Avoid direct access to database Tables	211	Applications should not access directly database Tables
	.NET: Data Access must be based on Stored Procedure Calls	211	Data Access must be based on Stored Procedure Calls.
	.NET: Avoid Classes with a High Lack of Cohesion 2	201	Avoid Classes with High Lack of Cohesion in Methods 2 (LCOM2 > X) LCOM is an indication that Methods of a Class accesses few and different fields of the Class. LCOM2 does not take into account methods that do not access any field. The threshold is a parameter and can be changed at will.
	.NET: Private Fields naming convention -	172	Names of Private Fields should not start with an

	case and character set control		uppercase character and should not include any underscore
	.NET: Avoid unreferenced Classes	121	All Classes should be referenced. An unreferenced Artifact is an Artifact that is not explicitly called from within the analyzed code.
	.NET: Avoid Classes with High Coupling Between Objects	113	Avoid Classes with High Coupling Between Object (CBO > X) The Coupling Between Object (CBO) is equal to the fan-out of a Class, that is, the number of other Classes that are referenced through one of its methods or one of its fields. The threshold is a parameter and can be changed at will.
	.NET: Class naming convention - case and character set control	107	Names of Classes should start with an uppercase character and should not include any underscore
	.NET: Avoid large Methods - too many Lines of Code	82	Methods should not have more than X lines of code. The threshold is a parameter and can be changed at will.
	.NET: Controls naming convention - prefix, case and character set control	80	Names of Controls should respect naming conventions (btn* for buttons, chk* for check boxes, cmb* for combo boxes,) and should not include any underscore. The list of Controls can be updated by editing the diagnostic Stored Procedure.
	.NET: Avoid Classes with a very low comment/code ratio	74	Classes should have at least a ratio comment/code > X % The threshold is a parameter and can be changed at will.
	.NET: Avoid declaring Public Class Fields	67	Public Class Fields should not be used
	.NET: Avoid Artifacts with High Depth of Code	67	Depth of Code is measured as the maximum number of nested control statements in an artifact. For example, an artifact that contains an IF statement which contains a While loop which itself contains another IF statement will have a Depth of Code of 3 (at least). Avoid Artifacts with Depth of Code (DoC) greater than X. The threshold level is a parameter that can be changed at will.
	.NET: Avoid Artifacts with High Cyclomatic Complexity	61	Avoid Artifacts with High Cyclomatic Complexity (CC greater than X ) Cyclomatic Complexity is a measure of the complexity of the control structure of an Artifact. It is the number of linearly independent paths and therefore, the minimum number of independent paths when executing the software. The threshold is a parameter and can be changed at will.
	.NET: Avoid artifacts with too many parameters	57	Avoid artifacts with more than X parameters. The threshold X is a parameter and it can be changed at will.
	.NET: Avoid String concatenation in loops	57	All String objects that use concatenation in loops (for, while, do while) will be reported.
	.NET: Avoid unreferenced Data Members	53	All Data Members should be referenced (this exclude getter and setter). An unreferenced Artifact is an Artifact that is not explicitly called from within the analyzed code.
	.NET: Avoid large number of String concatenation	51	All String Classes should not call more than X times the + Method
	.NET: Avoid instantiations inside loops	46	Reports all artifacts with loops (for, while, do while) that contain object instantiations (object creation).. .NET artifacts include all methods and constructors.
	.NET: Avoid uncommented Classes	45	Classes should have comments
	.NET: Public Fields naming convention - case and character set control	40	Names of Public Fields should start with an uppercase character and should not include any underscore
	.NET: Avoid High Response for a Class	30	Avoid High Response for a Class (RFC > X) RFC is the total number of local methods and remote methods called by methods in the Class. The threshold is a parameter and can be changed at will.
	.NET: Avoid Artifacts with High Integration Complexity	23	Avoid Artifacts with High Integration Complexity (IC greater than X). Integration Complexity measures the number of independent integration paths. Integration

			paths are paths of the control flow graph in which another object is invoked. The threshold is a parameter and can be changed at will.
	.NET: Avoid Classes with High Weighted Methods per Class	20	Avoid Classes with High Weighted Methods per Class (WMC > X). The Weighted Methods per Class metric is defined as the sum of all the Classes method?s cyclomatic complexity. The threshold is a parameter and can be changed at will.
	ADO.NET: Disable constraints before merging DataSet	20	Call to DataSet.Merge method should be done with disabled constraints.
	.NET: Avoid Classes with a High Public Data Ratio	11	Avoid Classes with a High Public Data Ratio greater than X % Public Data Ratio is the percentage of public fields among all fields. The threshold is a parameter and can be changed at will.
	.NET: Enumerations naming convention - case and character set control	9	Names of Enumerations should start with an uppercase character and should not include any underscore
	.NET: Avoid large Classes - too many Methods	9	Classes should have less than X Methods. The threshold is a parameter and can be changed at will.
	.NET: DataReader must be called using CommandBehavior.CloseConnection enumeration	8	The purpose of this diagnostic is to detect calls to the DataReader´s ExecuteReader method made without the use the CommandBehavior.CloseConnection style.  This diagnostic is valid for any class that the inherit directly or indirectly from IDataReader including SqlDataReader, OleDbDataReader, OracleDataReader, OdbcDataReader.
	.NET: Provide a private default constructor for utility classes	7	Utility classes must have a private default constructor. They also must not have other constructors. Default constructors are constructors without any parameter. Utility classes are static classes: all their fields and methods are static.
	.NET: Avoid using String.Empty	6	String.Empty should not be used
	.NET: Properties naming convention - case and character set control	6	Names of Properties should start with an uppercase character and should not include any underscore
	.NET: Avoid using exception handling inside loops	5	Reports all methods with loops (for, while, do while) that contains a try/catch block.
	.NET: Avoid using Keywords as names	4	Keywords should not be used as names
	.NET: Call ´base.Dispose()´ in the "finally" block of ´Dispose(bool)´ methods	3	This diagnostic lists all Dispose methods with no try finally block or with no call to .finalize() within the finally block. This applies only for Dispose(bool) method defined in classes which implement the IDisposable interface.
	.NET: Close SQL connection ASAP	3	SQL connection should be closed within the Method / Function / Sub that opened it
	.NET: Avoid throwing an exception of type Exception	2	This metric reports all methods throwing an exception of type Exception. The exception Exception should never been thrown.
	.NET: Avoid the use of is inside loops	2	All methods that use "is" operator inside a loop will be reported.
	.NET: Avoid Classes with a High Number Of Children	1	Avoid Classes with a High Number Of Children (NOC > X) NOC is the number of immediate <Sub-Classes> of the Class. The threshold is a parameter and can be changed at will.
 All	Diagnostic-based Metric relative to All language	2,171	2,171 Objects are considered as defects
	Avoid Too Many Copy Pasted Artifacts	1,496	This metric measures the ratio between the number of duplicated, copy/pasted artifacts and the total number of artifacts.
	Avoid Artifact with high Commented-out Code Lines/Code Lines ratio	264	Commented-out Code Lines/Code Lines ratio measures the amount of code left in comments versus the size of the source code (excluding comments and blank lines)

			this is done for a given artifact.
	Avoid SQL queries that no index can support	147	<p>This metric retrieves the artifacts containing at least one SQL Query not using a table's indexes. An SQL Query is using table's indexes if all the following conditions are true:</p> <ul style="list-style-type: none"> <li>- At least one index is defined for each table participating in the from list</li> <li>- Queries must reference left-most columns of the index key</li> <li>- Data matching where clause criterias are highly selective</li> </ul> <p>where selectivity = 1/density</p> <p>low number = high selectivity = low density high number = low selectivity = high density</p> <p>selectivity specify percentage of rows that match where clause criterias.</p> <p>Low number is linked to a high selectivity. If selectivity is 100%, all values of the index key are different and index is useful (best case) high number is linked to low selectivity: if density is 100%, all values are identical and index is inefficient.</p>
	Avoid artifacts having recursive calls	101	This metric retrieves all artifacts that are explicitly calling themselves (recursive call). Only executable artifacts are considered.
	Avoid SQL queries with implicit conversions in the WHERE clause	36	This metric retrieves all SQL artifacts with a query using implicit type conversion.
	Avoid using SQL queries inside a loop	32	This metric retrieves all artifacts using at least one SQL query inside a loop statement.
	Avoid "SELECT *" queries	24	This metric retrieves the artifacts containing queries using the "SELECT *" statement.
	Avoid having multiple artifacts inserting data on the same SQL table	14	This metric detects tables having too many ways to insert data into them. It retrieves table having more than X artifacts inserting these tables, where X a configurable parameter.
	Avoid having multiple artifacts updating data on the same SQL table	14	This metric detects tables having too many ways to update them. It retrieves tables having more than X artifacts updating these tables, where X a configurable parameter.
	Never use SQL queries with a cartesian product	13	<p>List all SQL Artifacts except tables having a query that has a cartesian join, i.e; that does not explicitly state a join condition among the tables</p> <p>A cartesian join is found if, for any of the table in the from clause, no column of the table is found in a join (either in the from or where clause).</p>
	Avoid having multiple artifacts deleting data on the same SQL table	11	This metric detects tables having too many ways to insert data into them. It retrieves table having more than X artifacts inserting these tables, where X a configurable parameter.
	SQL: Prefer UNION ALL to UNION	9	List all objects using UNION (without ALL).
	Avoid method invocation in a loop termination expression	8	Finds all loops (for, while, do while) termination that call a method.
	Avoid SQL queries not using the first column of a composite index in the WHERE clause	2	<p>This metric retrieves all client-server artifacts that contain an SQL query with a WHERE clause not using the first column of composite index (multiple column base index).</p> <p>The value of the metric is number of client-server artifacts that contain an SQL query not using the first column of a composite index divided by the number of client-server artifacts using a composite index.</p> <p>An artifacts using a composite index means here artifacts containing a query that uses at least one column of a composite index.</p>

