Sviluppatore 1

Label Scansione	08/05/2018 10:31		08/05/2018 11:42	
Data scansione	08/05/2018 10:31		08/05/2018 11:42	
LOC	693		699	
Risk index	**81,09**		**0**	
Global indicator	**69,96**		**95,61**	
Efficiency	97,55		97,63	
Maintainability	0,11		91,5	
Portability	100		100	
Reliability	72,67		91,51	
Security	100		100	
Total defects	**119**	**0,172**	**102**	**0,146**
Very high	4		0	
High	19		10	
Normal	45		43	
Low	45		43	
Very low	6		6	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Provide a branch block for 'if' statements	23	Provide a branch block for 'if' statements	21
n°2	Avoid assigning a value to a variable that has already been assigned	10	Avoid assigning a value to a variable that has already been assigned	10
n°3	Avoid catch blocks with empty bodies	5	Do not check the length of a series by invoking the String.equals('')	5
n°4	Do not check the length of a series by invoking the String.equals('')	5	Avoid using components calling too many other components	4
n°5	Cyclomatic complexity	4	Avoid capturing java.lang.Exception exceptions	4
n°6	Avoid using components calling too many other components	4	Avoid Exception, RuntimeException o Throwable in catch or throw statements	4
n°7	Avoid capturing java.lang.Exception exceptions	4	Classes internally strongly coupled must be avoided	3
n°8	Avoid Exception, RuntimeException o Throwable in catch or throw statements	4	Instantiate StringBuffer/StringBuilder with explicit initial size	3
n°9	Classes internally strongly coupled must be avoided	3	Initialize all static fields	2
n°10	Instantiate StringBuffer/StringBuilder with explicit initial size	3	Avoid non-final public static fields	2

Label Scansione	01/06/2018		01/06/2018	
Data scansione	01/06/2018 11:44		01/06/2018 11:58	
LOC	1271		720	
Risk index	**6,75**		**2,33**	
Global indicator	**64,31**		**62,52**	
Efficiency	96,99		99,67	
Maintainability	77,97		93,12	
Portability	92,04		75,8	
Reliability	74,96		58,06	
Security	0		0	
Total defects	**341**	**0,268**	**188**	**0,261**
Very high	9		7	
High	23		12	
Normal	133		54	
Low	138		95	
Very low	38		20	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Provide a branch block for 'if' statements	27	Provide a branch block for 'if' statements	22
n°2	Avoid Exception, RuntimeException o Throwable in catch or throw statements	24	Avoid Exception, RuntimeException o Throwable in catch or throw statements	22
n°3	Avoid capturing java.lang.Exception exceptions	16	Avoid capturing java.lang.Exception exceptions	14
n°4	Initialize all local variables at the declaration statement	15	Do not access or modify java.security configuration objects (Policy, Security, Provider, Principal, KeyStore)	12
n°5	Always check object type before cast	14	Initialize all local variables at the declaration statement	8
n°6	Avoid using numeric literals	14	Provide a by default private constructor in utility classes	8
n°7	Avoid using bitwise operators to make comparisons	12	Avoid using components calling too many other components	7
n°8	Do not access or modify java.security configuration objects (Policy, Security, Provider, Principal, KeyStore)	12	Always use specific exceptions in the throws clause	7
n°9	Avoid using components calling too many other components	11	Avoid cyclic dependencies between packages	7
n°10	Follow the limit for number of return statements	10	Avoid exposing sensible information through log	7

Label Scansione	24/10/2018 15:07		24/10/2018 16:06	
Data scansione	24/10/2018		24/10/2018	
LOC	878		899	
Risk index	**1,14**		**0**	
Global indicator	**77,75**		**99,22**	
Efficiency	99,98		99,98	0,1%
Maintainability	37,83		96,78	99,88%
Portability	100		100	0,0%
Reliability	89,92		99,87	20,6%
Security	76,59		100	0,0%
Total defects	**119**		**90**	**0,100**
Very high	0		0	
High	29		2	
Normal	37		33	
Low	50		52	
Very low	3		3	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Provide Javadoc comments for public classes and interfaces	12	Provide a branch block for 'if' statements	13
n°2	Avoid Exception, RuntimeException o Throwable in catch or throw statements	12	Avoid Exception, RuntimeException o Throwable in catch or throw statements	11
n°3	Provide a branch block for 'if' statements	9	Declare constant fields private and final	8
n°4	Declare constant fields private and final	8	Avoid using components calling too many other components	7
n°5	Avoid catch blocks with empty bodies	7	Provide Javadoc comments for protected fields	5
n°6	Avoid using components calling too many other components	7	Avoid capturing java.lang.Exception exceptions	5
n°7	Avoid giving method local variables and parameters the same name as class fields	6	Always check object type before cast	4
n°8	Avoid parameter method names that provoke conflicts with class members names	5	Use constructors than initialize all the fields in a class	4
n°9	Provide Javadoc comments for protected fields	5	Classes internally strongly coupled must be avoided	3
n°10	Avoid capturing java.lang.Exception exceptions	5	Provide Javadoc comments for public methods in Java classes (non-interface)	3

Label Scansione	05/11/2018 16:51		05/11/2018 17:04	
Data scansione	05/11/2018 16:51		05/11/2018	
LOC	455		453	
Risk index	**0**		**0**	
Global indicator	**97,67**		**98,85**	
Efficiency	99,69		99,69	
Maintainability	95,25		95,59	
Portability	100		100	
Reliability	94,96		99,73	
Security	100		100	
Total defects	**31**	**0,068**	**26**	**0,057**
Very high	0		0	
High	4		2	
Normal	17		16	
Low	10		8	
Very low	0		0	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Instantiate StringBuffer/StringBuilder with explicit initial size	6	Instantiate StringBuffer/StringBuilder with explicit initial size	6
n°2	Avoid modifications on method or constructor parameters	4	Avoid modifications on method or constructor parameters	4
n°3	Avoid creating or assigning a variable within a loop	3	Avoid creating or assigning a variable within a loop	3
n°4	Avoid if statements with empty bodies	2	Initialize all local variables at the declaration statement	2
n°5	Initialize all local variables at the declaration statement	2	Maximum allowed number of methods	1
n°6	Use a branches block for 'else' statements	2	Avoid excesive import lines	1
n°7	Maximum allowed number of methods	1	Avoid declaring multiple variables in one statement	1
n°8	Avoid excesive import lines	1	Avoid using components calling too many other components	1
n°9	Avoid declaring multiple variables in one statement	1	Classes internally strongly coupled must be avoided	1
n°10	Avoid using components calling too many other components	1	Counts the number of method calls	1

Label Scansione	14/02/2019 17:38		15/02/2019 10:35	
Data scansione	**14/02/2019 17:38**		15/02/2019 10:35	
LOC	254		256	
Risk index	41,33		**0**	
Global indicator	77,98		**99,1**	
Efficiency	99,77		99,78	
Maintainability	5,55		96,32	
Portability	100		100	
Reliability	99,97		100	
Security	100		100	
Total defects	19	**0,075**	**18**	**0,070**
Very high	1		0	
High	1		1	
Normal	7		6	
Low	6		6	
Very low	4		5	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Minimize using syncrohized collections: detect use of synchronized collectiorns (Vector, Hashtable) and indicate it	3	Minimize using syncrohized collections: detect use of synchronized collectiorns (Vector, Hashtable) and indicate it	3
n°2	Do not place statements on the same line as {opening brace	2	Leave a white space between arguments	2
n°3	Cyclomatic complexity	1	Do not place statements on the same line as {opening brace	2
n°4	Avoid unused imports	1	Avoid unused imports	1
n°5	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°6	Classes internally strongly coupled must be avoided	1	Classes internally strongly coupled must be avoided	1
n°7	Initialize all local variables at the declaration statement	1	Avoid duplicate literals	1
n°8	Avoid duplicate literals	1	Provide Javadoc comments for public methods in Java classes (non-interface)	1
n°9	Provide Javadoc comments for public methods in Java classes (non-interface)	1	Avoid capturing java.lang.Exception exceptions	1
n°10	Avoid capturing java.lang.Exception exceptions	1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1

Label Scansione	22/02/2019 16:41		22/02/2019 17:17	
Data scansione	22/02/2019 16:41		22/02/2019 17:17	
LOC	2.188		2.110	
Risk index	**1,34**		**0**	
Global indicator	**71,66**		**98,67**	
Efficiency	96,33		96,75	0,0%
Maintainability	82,72		98,63	0,36%
Portability	97,74		97,58	0,0%
Reliability	99,47		99,45	4,8%
Security	0,01		100	0,0%
Total defects	**189**	**0,086**	**166**	**0,079**
Very high	7		1	
High	20		18	
Normal	58		52	
Low	78		68	
Very low	26		27	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	20	Avoid Exception, RuntimeException o Throwable in catch or throw statements	18
n°2	Avoid capturing java.lang.Exception exceptions	17	Avoid capturing java.lang.Exception exceptions	15
n°3	Avoid unused parameters	14	Avoid unused parameters	12
n°4	Avoid using components calling too many other components	12	Avoid using components calling too many other components	11
n°5	Avoid methods that invoke only overwritten supermethods	10	Provide Javadoc comments for public methods in Java classes (non-interface)	10
n°6	Provide Javadoc comments for public methods in Java classes (non-interface)	10	Avoid methods that invoke only overwritten supermethods	9
n°7	Do not place statements on the same line as {opening brace	9	Do not place statements on the same line as {opening brace	9
n°8	Classes internally strongly coupled must be avoided	7	Classes internally strongly coupled must be avoided	6
n°9	Avoid incorrect name format in the final static fields	5	Avoid incorrect name format in the final static fields	5
n°10	Always check object type before cast	4	Always check object type before cast	4

Fig.3.17. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 2

Label Scansione	VBApp-GiuseppeVischi VB06.19.00-TFS_76646_30		VBApp-GiuseppeVischi VB06.19.00-TFS_76646_32	
Data scansione	05/01/2018		05/01/2018	
LOC	455		536	
Risk index	**0**		**0**	
Global indicator	**98,03**		**99,54**	
Efficiency	99,94		99,98	
Maintainability	95,16		98,06	
Portability	100		100	
Reliability	96,4		99,98	
Security	100		100	
Total defects	**30**	**0,066**	**36**	**0,067**
Very high	0		0	
High	6		0	
Normal	13		24	
Low	1		8	
Very low	10		4	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	6	Old commented code	5
n°2	Avoid using components calling too many other components	3	Avoid class or interface names too long	4
n°3	Avoid capturing NullPointerExceptions	2	Avoid using components calling too many other components	4
n°4	Avoid using a public static final field of type Matrix	2	Follow the limit for number of statements in a method	4
n°5	Avoid using public static final array fields	2	Provide Javadoc comments for public methods in Java classes (non-interface)	4
n°6	Instantiate StringBuffer/StringBuilder with explicit initial si	2	Always check object type before cast	2
n°7	Avoid putting non-Javadoc comments between Javadoc and class / method de2clarations		Classes internally strongly coupled must be avoided	2
n°8	Avoid unused imports	1	Avoid using numeric literals	2
n°9	Maximum allowed number of methods	1	Avoid creating an instance of a number using new	2
n°10	Avoid calling non-final, non-static and non-private methods from constructors	1	Duplicated code: small blocks	2

Label Scansione	08/05/2018 17:04		11/05/2018 11:24	
Data scansione	08/05/2018		11/05/2018 11:24	
LOC	1676		1682	
Risk index	**1,64**		**0**	
Global indicator	**89,54**		**98,58**	
Efficiency	99,89		99,95	
Maintainability	63,93		94,34	
Portability	100		100	
Reliability	91,22		99,62	
Security	99,97		99,98	
Total defects	**121**	**0,072**	**109**	**0,065**
Very high	0		0	
High	29		12	
Normal	43		39	
Low	39		34	
Very low	10		24	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	13	Avoid putting non-Javadoc comments between Javadoc and class / method declarations	14
n°2	Avoid capturing java.lang.Exception exceptions	13	Avoid Exception, RuntimeException o Throwable in catch or throw statements	13
n°3	Avoid if statements with empty bodies	7	Avoid capturing java.lang.Exception exceptions	13
n°4	Follow the limit for number of statements in a method	7	Follow the limit for number of statements in a method	7
n°5	Always check object type before cast	5	Always check object type before cast	5
n°6	NULL Pointer Dereference	5	NULL Pointer Dereference	5
n°7	Cyclomatic complexity	4	Avoid using components calling too many other components	4
n°8	Avoid using components calling too many other components	4	Avoid using numeric literals	4
n°9	Avoid using numeric literals	4	Do not declareparivate fields that are used in only one method	4
n°10	Do not declareparivate fields that are used in only one method	4	Provide a branch block for 'if' statements	4

Label Scansione	01/06/2018 09:42		01/06/2018 11:12	
Data scansione	01/06/2018		01/06/2018	
LOC	649		653	
Risk index	**75,09**		**47,46**	
Global indicator	**70,48**		**92,03**	
Efficiency	65,3		99,96	
Maintainability	2,47		65,91	
Portability	100		100	
Reliability	98,42		99,87	
Security	99,94		99,98	
Total defects	**79**	**0,122**	**53**	**0,081**
Very high	2		0	
High	20		9	
Normal	33		23	
Low	14		10	
Very low	10		11	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Do not instantiate temporal Objects in loops bodies	6	Follow the limit for number of statements in a method	5
n°2	Follow the limit for number of statements in a method	5	Avoid incorrect name format in local variables	5
n°3	Initialize all local variables at the declaration statement	5	Cyclomatic complexity	4
n°4	Avoid capturing java.lang.Exception exceptions	5	Avoid capturing java.lang.Exception exceptions	4
n°5	Cyclomatic complexity	4	Avoid Exception, RuntimeException o Throwable in catch or throw statements	4
n°6	Always check object type before cast	4	Always check object type before cast	4
n°7	Instantiate StringBuffer/StringBuilder with explicit initial size	4	Instantiate StringBuffer/StringBuilder with explicit initial size	3
n°8	Unchecked input in loop condition	4	Avoid nested IF sentences with too many levels	3
n°9	Avoid using method calls in a loop	3	Unchecked input in loop condition	2
n°10	Avoid nested IF sentences with too many levels	3	Provide a branch block for 'if' statements	2

Label Scansione	29/08/2018 16:04		2018/08/29 16:16	
Data scansione	29/08/2018		29/08/2018 16:16	
LOC	193		197	
Risk index	**4,26**		**0**	
Global indicator	**79,85**		**98,7**	
Efficiency	100		100	
Maintainability	13,43		94,61	
Portability	100		100	
Reliability	100		99,85	
Security	99,93		99,97	
Total defects	**17**	**0,088**	**13**	**0,066**
Very high	0		0	
High	5		1	
Normal	5		6	
Low	5		5	
Very low	2		1	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid unused parameters	2	Avoid modifications on method or constructor parameters	2
n°2	Provide Javadoc comments for public classes and interfaces	1	Provide Javadoc comments for public fields	1
n°3	Avoid nested IF sentences with too many levels	1	Avoid using components calling too many other components	1
n°4	Provide Javadoc comments for public fields	1	Do not declareparivate fields that are used in only one method	1
n°5	Avoid using components calling too many other components	1	Too many uses of the negation operator (!) in a method	1
n°6	Do not declareparivate fields that are used in only one method	1	NULL Pointer Dereference	1
n°7	Follow the limit for number of statements in a method	1	Avoid class or interface names too long	1
n°8	Too many uses of the negation operator (!) in a method	1	Provide Javadoc comments for public methods in Java classes (non-interface)	1
n°9	NULL Pointer Dereference	1	Avoid capturing java.lang.Exception exceptions	1
n°10	Avoid class or interface names too long	1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1

Fig.3.18. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 3

Label Scansione	V06.19.00_TFS-76640_1		V06.19.00_TFS-76640_4	
Data scansione	05/01/2018		05/01/2018 11:32	
LOC	1.265		1.244	
Risk index	**1,53**		**0**	
Global indicator	**94,46**		**99,3**	
Efficiency	98,85		100	
Maintainability	79,26		98,78	
Portability	100		100	
Reliability	97,83		98,22	
Security	100		100	
Total defects	**117**	**0,092**	**96**	**0,077**
Very high	1		0	
High	10		1	
Normal	84		81	
Low	16		8	
Very low	6		6	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Always check object type before cast	20	Always check object type before cast	20
n°2	Avoid using numeric literals	12	Avoid using numeric literals	12
n°3	Declare the List and Set variables with their interface type	9	Initialize all local variables at the declaration statement	11
n°4	Initialize all local variables at the declaration statement	8	Declare the List and Set variables with their interface type	9
n°5	Avoid using components calling too many other components	6	Avoid modifications on method or constructor parameters	7
n°6	Avoid modifications on method or constructor parameters	6	Avoid using components calling too many other components	6
n°7	Do not declare private fields that are used in only one method	4	Do not declare private fields that are used in only one method	4
n°8	Avoid returning null in a method declaration	4	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	4
n°9	Follow the limit for number of return statements	4	Classes internally strongly coupled must be avoided	3
n°10	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	4	Declare constant fields private and final	3

Label Scansione	V06.19.00_TFS-76640_1_javascript		V06.19.00_TFS-76640_2_javascript	
Data scansione	05/01/2018		05/01/2018 13:54	
LOC	448		443	
Risk index	**0,21**		**0**	
Global indicator	**87,75**		**99,59**	
Efficiency	100		100	
Maintainability	98,56		98,35	
Portability	100		100	
Reliability	48,75		99,88	
Security	100		100	
Total defects	**53**	**0,118**	**42**	**0,095**
Very high	0		0	
High	7		1	
Normal	11		0	
Low	35		41	
Very low	0		0	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid using logical comparators == and !=	29	Avoid using logical comparators == and !=	29
n°2	Avoid unused local variable	11	Avoid using Array and Object constructors	5
n°3	Define variables with var	5	Place whitespaces between logical operators and its operands	4
n°4	Place whitespaces between logical operators and its operands	4	Merge nested if statements using a short-circuit operator	2
n°5	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°6	Avoid declaring a variable with a name that is already used	1	Else if statements should finish with an else clause	1
n°7	Merge nested if statements using a short-circuit operator	1		
n°8	Else if statements should finish with an else clause	1		
n°9				
n°10				

Label Scansione	V06.19.00_TFS-76640_post_modifiche_tracciato_V1_javascript		V06.19.00_TFS-76640_post_modifiche_tracciato_V5_javascript	
Data scansione	22/01/2018		22/01/2018	
LOC	727		611	
Risk index	**96,93**		**0**	
Global indicator	**56,32**		**99,62**	
Efficiency	59,5		100	1,2%
Maintainability	0,21		98,42	19,76%
Portability	100		100	0,0%
Reliability	44,37		99,94	0,4%
Security	100		100	0,0%
Total defects	**76**	**0,105**	**48**	**0,079**
Very high	4		0	
High	16		1	
Normal	4		3	
Low	52		44	
Very low	0		0	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using logical comparators == and !=	40	Avoid using logical comparators == and !=	27
n°2	Define variables with var	9	Place whitespaces between logical operators and its operands	6
n°3	Place whitespaces between logical operators and its operands	6	Merge nested if statements using a short-circuit operator	5
n°4	Merge nested if statements using a short-circuit operator	5	Avoid using Array and Object constructors	5
n°5	Avoid functions with excessive number of lines	3	Avoid using functions with too many paramters	3
n°6	Avoid declaring a variable with a name that is already used	3	Avoid using components calling too many other components	1
n°7	Do not use ? ternary operator to evaluate conditions	2	Else if statements should finish with an else clause	1
n°8	Duplicated code: small block	2		
n°9	Avoid too big JavaScript files	1		
n°10	Avoid using components calling too many other components	1		

Label Scansione	V06.20.00_TFS-81174_V1		V06.20.00_TFS-81174_V2	
Data scansione	08/02/2018		08/02/2018	
LOC	1.293		1.292	
Risk index	**0**		**0**	
Global indicator	**98,51**		**99,1**	
Efficiency	99,83		99,83	
Maintainability	93,87		96,38	
Portability	100		100	
Reliability	99,87		99,87	
Security	100		100	
Total defects	**92**	**0,071**	**90**	**0,070**
Very high	0		0	
High	6		4	
Normal	48		48	
Low	22		22	
Very low	16		16	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid using numeric literals	16	Avoid using numeric literals	16
n°2	Avoid creating or assigning a variable within a loop	15	Avoid creating or assigning a variable within a loop	15
n°3	Always check object type before cast	10	Always check object type before cast	10
n°4	Leave a white space between arguments	10	Leave a white space between arguments	10
n°5	Follow the limit for number of statements in a method	7	Follow the limit for number of statements in a method	7
n°6	Make methods static if they do not use instance class members	4	Make methods static if they do not use instance class members	4
n°7	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	2	Declare the List and Set variables with their interface type	2
n°8	Initialize all local variables at the declaration statement	2	Initialize all local variables at the declaration statement	2
n°9	Declare the List and Set variables with their interface type	2	Avoid fields and methods with the same name	2
n°10	Avoid incorrect name format in local variables	2	Avoid incorrect name format in local variables	2

Label Scansione	V06.20.00_TFS-81175_javascript_V1		V06.20.00_TFS-81175_javascript_V2	
Data scansione	**08/02/2018 18:00**		08/02/2018	
LOC	2490		2.486	
Risk index	94,54		**82,78**	
Global indicator	47,62		**48,49**	
Efficiency	83,39		95,19	
Maintainability	0,01		0,07	
Portability	100		100	
Reliability	55,89		53,15	
Security	32,13		29,12	
Total defects	646	**0,259**	**643**	**0,259**
Very high	7		4	
High	122		123	
Normal	97		95	
Low	420		421	
Very low	0		0	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using logical comparators == and !=	260	Avoid using logical comparators == and !=	260
n°2	Place whitespaces between logical operators and its operands	104	Place whitespaces between logical operators and its operands	104
n°3	Place body of if statements between braces	83	Place body of if statements between braces	83
n°4	Merge nested if statements using a short-circuit operator	40	Merge nested if statements using a short-circuit operator	41
n°5	Avoid a high number of nested ifs	40	Avoid a high number of nested ifs	40
n°6	Define variables with var	31	Define variables with var	31
n°7	Avoid statements without semicolon	25	Avoid statements without semicolon	26
n°8	Avoid unused local variable	10	Avoid unused local variable	10
n°9	Do not use ? ternary operator to evaluate conditions	9	Do not use ? ternary operator to evaluate conditions	9
n°10	Avoid using methods with high cyclomatic complexity values	6	Avoid using methods with high cyclomatic complexity values	6

Label Scansione	V06.21.00_TFS-86677_V1_javascript		V06.21.00_TFS-86677_V2_javascript	
Data scansione	07/05/2018		07/05/2018 16:55	
LOC	150		147	
Risk index	**0,17**		**0,17**	
Global indicator	**96,65**		**96,89**	
Efficiency	100		100	0,0%
Maintainability	85,64		86,62	2,60%
Portability	100		100	0,0%
Reliability	99,95		100	0,0%
Security	100		100	0,0%
Total defects	**14**	**0,093**	**4**	**0,027**
Very high	0		0	
High	1		1	
Normal	0		0	
Low	13		3	
Very low	0		0	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid using logical comparators == and !=	6	Merge nested if statements using a short-circuit operator	2
n°2	Place whitespaces between logical operators and its operands	4	Avoid using components calling too many other components	1
n°3	Merge nested if statements using a short-circuit operator	2	Avoid using Array and Object constructors	1
n°4	Avoid using components calling too many other components	1		
n°5	Avoid using Array and Object constructors	1		
n°6				
n°7				
n°8				
n°9				
n°10				

Label Scansione	V06.21.00_TFS-91975_V1		V06.21.00_TFS-91975_V3	
Data scansione	22/06/2018		22/06/2018 17:20	
LOC	1.753		1.757	
Risk index	**4,8**		**0**	
Global indicator	**76,55**		**99,43**	
Efficiency	99,95		99,95	
Maintainability	59,3		97,8	
Portability	100		100	
Reliability	99,77		99,77	
Security	40,14		100	
Total defects	**51**	**0,029**	**45**	**0,026**
Very high	4		0	
High	9		9	
Normal	28		27	
Low	10		7	
Very low	0		2	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid using components calling too many other components	7	Avoid using components calling too many other components	7
n°2	Classes internally strongly coupled must be avoided	5	Classes internally strongly coupled must be avoided	5
n°3	Follow the limit for number of statements in a method	5	Follow the limit for number of statements in a method	5
n°4	Always check object type before cast	5	Always check object type before cast	5
n°5	Avoid the @return Javadoc tag in void methods	5	Avoid the @return Javadoc tag in void methods	5
n°6	Cyclomatic complexity	3	Initialize all static fields	2
n°7	Provide a branch block for 'if' statements	3	Maximum allowed number of methods	2
n°8	Initialize all static fields	2	Maximum allowed number of fields	2
n°9	Maximum allowed number of methods	2	Avoid using java.util.Date or java.util.GregorianCalendar	2
n°10	Maximum allowed number of fields	2	Unchecked input in loop condition	2

Label Scansione	V06.22.00_TFS-96537_V1		V06.22.00_TFS-96537_V7	
Data scansione	**28/08/2018 16:29**		31/08/2018	
LOC	521		666	
Risk index	0,74		**0**	
Global indicator	91,57		**97,76**	
Efficiency	79,39		99,66	
Maintainability	82,27		90,71	
Portability	100		100	
Reliability	97,96		99,95	
Security	100		100	
Total defects	97	**0,186**	**79**	**0,119**
Very high	0		0	
High	9		0	
Normal	66		74	
Low	14		5	
Very low	8		0	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using numeric literals	49	Avoid using numeric literals	50
n°2	Avoid unused parameters	9	Close JDBC resources when finishing using	8
n°3	Close JDBC resources when finishing using	8	Follow the limit for number of statements in a method	4
n°4	Leave a white space between arguments	7	Avoid using components calling too many other components	3
n°5	Do not use the printStackTrace method	5	Instantiate StringBuffer/StringBuilder with explicit initial size	2
n°6	Follow the limit for number of statements in a method	3	Avoid capturing java.lang.Exception exceptions	2
n°7	Avoid if statements with empty bodies	2	Avoid Exception, RuntimeException o Throwable in catch or throw statements	2
n°8	Avoid using components calling too many other components	2	Classes internally strongly coupled must be avoided	1
n°9	Avoid the @return Javadoc tag in void methods	2	Only use Hibernate/JPA for database access	1
n°10	Avoid unused private methods and constructors	1	Initialize all local variables at the declaration statement	1

Label Scansione	V06.22.00_TFS-98079_V1		V06.22.00_TFS-98079_V3	
Data scansione	18/09/2018		18/09/2018	
LOC	322		319	
Risk index	**0**		**0**	
Global indicator	**97,4**		**99,87**	
Efficiency	100		100	0,0%
Maintainability	92,19		99,75	39,37%
Portability	98,77		100	0,0%
Reliability	97,25		99,68	0,0%
Security	100		100	59,9%
Total defects	**22**	**0,068**	**15**	**0,047**
Very high	0		0	
High	4		0	
Normal	10		9	
Low	7		5	
Very low	1		1	

	#23		#26	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Provide a branch block for 'if' statements	5	Provide a branch block for 'if' statements	3
n°2	Avoid using components calling too many other components	2	Avoid using components calling too many other components	2
n°3	Classes internally strongly coupled must be avoided	2	Classes internally strongly coupled must be avoided	2
n°4	Always check object type before cast	2	Always check object type before cast	2
n°5	Do not use Runtime and System classes	1	Initialize all local variables at the declaration statement	1
n°6	Avoid parameter method names that provoke conflicts with class members names	1	Avoid modifications on method or constructor parameters	1
n°7	Avoid nested IF sentences with too many levels	1	Avoid TODO comments in production code	1
n°8	Avoid comparing floating point types	1	Avoid creating or assigning a variable within a loop	1
n°9	Initialize all local variables at the declaration statement	1	Provide Javadoc comments for public methods in Java interfaces	1
n°10	Avoid giving method local variables and parameters the same name as class fields	1	Provide Javadoc comments for private methods	1

Fig.3.19 Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 4

Label Scansione	wws-console-framework		wws-console-framework	
Data scansione	06/12/2018		21/12/2018	
LOC	3.709		3.650	
Risk index	**0,49**		**0**	
Global indicator	**66,16**		**96,92**	
Efficiency	97,99		99,11	
Maintainability	59,92		90,02	
Portability	99,53		99,78	
Reliability	96,27		98,72	
Security	0,15		98,82	
Total defects	**641**	**0,173**	**541**	**0,148**
Very high	6		0	
High	83		53	
Normal	229		227	
Low	248		198	
Very low	75		63	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Always check object type before cast	59	Always check object type before cast	53
n°2	Avoid Exception, RuntimeException o Throwable in catch or throw statements	35	Do not define 'Object' variables	33
n°3	Provide Javadoc comments for public methods in Java classes (non-interface	35	Avoid Exception, RuntimeException o Throwable in catch or throw statements	26
n°4	Avoid capturing java.lang.Exception exceptions	34	Avoid using components calling too many other components	25
n°5	Do not access or modify java.security configuration objects (Policy, Security, Provider, Principal, KeyStore)	26	Avoid capturing java.lang.Exception exceptions	25
n°6	Avoid using components calling too many other components	24	Use constructors than initialize all the fields in a class	23
n°7	Follow the limit for number of return statements	22	Follow the limit for number of return statements	21
n°8	Use constructors than initialize all the fields in a class	22	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	21
n°9	Provide a branch block for 'if' statements	21	Do not access or modify java.security configuration objects (Policy, Security, Provider, Principal, KeyStore)	19
n°10	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	19	Avoid using numeric literals	17

Label Scansione	wws-console-user-service		wws-console-user-service	
Data scansione	11/12/2018		21/12/2018	
LOC	21586		4659	
Risk index	**15,17**		**0.28**	
Global indicator	**60,82**		**68,92**	
Efficiency	97,35		98,13	
Maintainability	18,02		78	
Portability	97,85		95,25	
Reliability	88,44		91,01	
Security	28,28		1,21	
Total defects	**2564**	**0,119**	**553**	**0,119**
Very high	56		13	
High	453		94	
Normal	1174		172	
Low	629		183	
Very low	252		91	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Duplicated code: small block	527	Avoid using numeric literals	37
n°2	Duplicated code: medium block	264	Do not use Runtime and System classes	34
n°3	Indicate the character encoding used	222	Avoid hard-coded or in-comment passwords in code	34
n°4	Always include a doctype declaration	222	Provide Javadoc comments for public methods in Java classes (non-interface)	27
n°5	Use descriptive comments in the pages	222	Follow the limit for number of return statements	24
n°6	Use a header comment for every page	88	Serializable Class Containing Sensitive Data	22
n°7	Tittle' attribute of a link not found	88	Use constructors than initialize all the fields in a class	21
n°8	Do not use function declarations within blocks	70	Avoid using components calling too many other components	19
n°9	Avoid returning null in a method declaration	52	Include a proper response code along with an appropriate return type	17
n°10	Avoid returning null in a method declaration	49	Provide Javadoc comments for public classes and interfaces	16

Label Scansione				
Data scansione	12/02/2019		12/02/2019	
LOC	11.590		9.199	
Risk index	**2,89**		**3,45**	
Global indicator	**88,23**		**91,93**	
Efficiency	99,61		99,96	
Maintainability	62,98		65,32	
Portability	99,91		100	
Reliability	99,6		99,99	
Security	87,18		100	
Total defects	**845**	**0,073**	**332**	**0,036**
Very high	22		19	
High	102		21	
Normal	455		148	
Low	235		132	
Very low	31		12	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	64	Follow the limit for number of return statements	58
n°2	Avoid using numeric literals	62	Avoid using components calling too many other components	30
n°3	Always use specific exceptions in the throws clause	61	Avoid returning null in a method declaration	29
n°4	Avoid using components calling too many other components	46	Avoid returning null in a method declaration	24
n°5	Follow the limit for number of return statements	45	Cyclomatic complexity	19
n°6	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	35	Avoid using numeric literals	15
n°7	Make methods static if they do not use instance class members	35	Always check object type before cast	10
n°8	Follow the limit for number of statements in a method	25	Include a proper response code along with an appropriate return type	10
n°9	Avoid duplicate literals	24	Classes internally strongly coupled must be avoided	9
n°10	Avoid comparing a variable with itself	24	Avoid TODO comments in production code	8

Label Scansione			30/01/2019 21:26	
Data scansione	**30/01/2019**		30/01/2019	
LOC	3852		5.346	
Risk index	0		**0**	
Global indicator	98,6		**80,41**	
Efficiency	99,46		99,57	
Maintainability	96,2		97,61	
Portability	99,82		100	
Reliability	99,31		99,97	
Security	98,98		18,39	
Total defects	459	**0,119**	**320**	**0,060**
Very high	1		5	
High	29		23	
Normal	169		135	
Low	203		135	
Very low	57		22	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Follow the limit for number of return statements	33	Follow the limit for number of return statements	38
n°2	Avoid Exception, RuntimeException o Throwable in catch or throw statements	26	Serializable Class Containing Sensitive Data	29
n°3	Avoid using components calling too many other components	25	Avoid using numeric literals	27
n°4	Avoid capturing java.lang.Exception exceptions	25	Include a proper response code along with an appropriate return type	24
n°5	Use constructors than initialize all the fields in a class	23	Avoid using components calling too many other component	23
n°6	Use constructors than initialize all the fields in a class	21	Use constructors than initialize all the fields in a class	22
n°7	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	21	Avoid creating or assigning a variable within a loop	13
n°8	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	20	Avoid creating or assigning a variable within a loop	12
n°9	Do not access or modify java.security configuration objects (Policy, Security, Provider, Principal, KeyStore)	19	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	12
n°10	Avoid using numeric literals	17	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	8

Fig.3.20. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 5

Label Scansione	abas_ubi		abas_ubi	
Data scansione	10/01/2018		11/01/2018	
LOC	880		876	
Risk index	**99,67**		**99,94**	
Global indicator	**74,31**		**74,29**	
Efficiency	87,7		87,61	
Maintainability	0		0	
Portability	100		100	
Reliability	99,37		99,37	
Security	100		100	
Total defects	**104**	**0,118**	**103**	**0,118**
Very high	8		8	
High	16		15	
Normal	74		74	
Low	5		5	
Very low	1		1	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	cyclomatic complexity	8	cyclomatic complexity	8
n°2	Avoid nested IF sentences with too many levels	7	Avoid nested IF sentences with too many levels	6
n°3	Avoid static collections; they can grow without bounds	5	Avoid static collections; they can grow without bounds	5
n°4	Provide Javadoc comments for public fields	1	Provide Javadoc comments for public fields	1
n°5	Avoid using method calls in a loop	1	Avoid using method calls in a loop	1
n°6	Avoid classes / interfaces with too many lines of code	1	Avoid classes / interfaces with too many lines of code	1
n°7	Avoid excesive import lines	1	Avoid excesive import lines	1
n°8	Declare the List and Set variables with their interface type	19	Declare the List and Set variables with their interface type	19
n°9	Follow the limit for number of statements in a method	11	Follow the limit for number of statements in a method	11
n°10	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	11	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	11

Label Scansione	08/02/2018 17:30		08/02/2018 18:11	
Data scansione	08/02/2018		08/02/2018	
LOC	506		504	
Risk index	**0,69**		**0**	
Global indicator	**93,58**		**97,84**	
Efficiency	89,93		99,23	
Maintainability	80,59		91,48	
Portability	100		100	
Reliability	99,84		99,84	
Security	100		100	
Total defects	**40**	**0,079**	**34**	**0,067**
Very high	0		0	
High	8		3	
Normal	24		23	
Low	4		4	
Very low	4		4	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Do not instantiate temporal Objects in loops bodies	4	Do not instantiate temporal Objects in loops bodies	1
n°2	Avoid nested IF sentences with too many levels	3	Avoid nested IF sentences with too many levels	1
n°3	Avoid excesive import lines	1	Avoid excesive import lines	1
n°4	Follow the limit for number of statements in a method	7	Follow the limit for number of statements in a method	6
n°5	Declare the List and Set variables with their interface type	5	Declare the List and Set variables with their interface type	5
n°6	Avoid using numeric literals	3	Avoid using numeric literals	3
n°7	Always check object type before castessential	3	Always check object type before castessential	3
n°8	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	2	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	2
n°9	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°10	Classes internally strongly coupled must be avoided	1	Classes internally strongly coupled must be avoided	1

Label Scansione	12/04/2018 09:51		12/04/2018 09:55	
Data scansione	12/04/2018		12/04/2018	
LOC	2478		2495	
Risk index	**93,79**		**92,55**	
Global indicator	**73,62**		**74,26**	
Efficiency	83,29		86,67	
Maintainability	0,08		0,09	
Portability	100		100	
Reliability	99,88		99,89	
Security	99,98		99,98	
Total defects	**385**		**353**	**0,141**
Very high	12		12	
High	39		39	
Normal	284		254	
Low	31		29	
Very low	19		19	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	12	Cyclomatic complexity	12
n°2	Avoid nested IF sentences with too many levels	11	Avoid nested IF sentences with too many levels	11
n°3	Avoid using method calls in a loop	9	Avoid using method calls in a loop	9
n°4	Avoid excesive import lines	3	Avoid excesive import lines	3
n°5	Avoid returning Java.lang.Object, instead convert it to a specific type	3	Avoid returning Java.lang.Object, instead convert it to a specific type	3
n°6	Provide Javadoc comments for public fields	2	Avoid unused imports	2
n°7	Do not instantiate temporal Objects in loops bodies	2	Do not instantiate temporal Objects in loops bodies	2
n°8	Avoid the use of reflection	2	Avoid the use of reflection	2
n°9	Maximum allowed number of methods	1	Maximum allowed number of methods	1
n°10	Provide Javadoc comments for public classes and interfaces	1	Provide Javadoc comments for public fields	1

Label Scansione	89459_ISP - PRJ Gestione TAG 9F34 - STEP 2_1		89459_ISP - PRJ Gestione TAG 9F34 - STEP 2_1	
Data scansione	14/05/2018		14/05/2018	
LOC	1.346		1.339	
Risk index	**7,19**		**6,8**	
Global indicator	**94,23**		**95,45**	
Efficiency	99,33		99,75	
Maintainability	75,75		80,66	
Portability	100		100	
Reliability	99,96		99,96	
Security	100		100	
Total defects	**503**	**0,374**	**75**	**0,056**
Very high	0		0	
High	14		11	
Normal	52		52	
Low	6		6	
Very low	6		6	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	3	Cyclomatic complexity	3
n°2	Avoid nested IF sentences with too many levels	3	Avoid nested IF sentences with too many levels	3
n°3	Avoid excesive import lines	2	Avoid excesive import lines	2
n°4	Maximum allowed number of methods	1	Maximum allowed number of methods	1
n°5	Date format class java.text.SimpleDateFormat spends many resources	1	Date format class java.text.SimpleDateFormat spends many resources	1
n°6	Avoid classes / interfaces with too many lines of code	1	Avoid classes / interfaces with too many lines of code	1
n°7	Do not instantiate temporal Objects in loops bodies	1	Avoid using numeric literals	20
n°8	Avoid unused parameters	1	Follow the limit for number of statements in a method	12
n°9	Avoid unused local variables	1	lways check object type before cast	5
n°10	Avoid using numeric literals	20	Instantiate StringBuffer/StringBuilder with explicit initial size	3

Label Scansione	89459_ISP - PRJ Gestione TAG 9F34 - STEP 2_2		89459_ISP - PRJ Gestione TAG 9F34 - STEP 2_2	
Data scansione	14/05/2018		14/05/2018	
LOC	2967		2.964	
Risk index	52,23		**51,87**	
Global indicator	83,47		**83,84**	
Efficiency	98,6		99,24	
Maintainability	30,22		31,25	
Portability	100		100	
Reliability	99,85		99,89	
Security	99,99		99,99	
Total defects	325	**0,110**	**281**	**0,095**
Very high	2		2	
High	33		34	
Normal	233		191	
Low	49		46	
Very low	8		8	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Duplicated code: big block	2	Duplicated code: big block	2
n°2	Cyclomatic complexity	11	Cyclomatic complexity	11
n°3	Avoid nested IF sentences with too many levels	11	Avoid nested IF sentences with too many levels	11
n°4	Duplicated code: medium block	4	Duplicated code: medium block	4
n°5	Avoid classes / interfaces with too many lines of code	2	Do not instantiate temporal Objects in loops bodies	3
n°6	Avoid excesive import lines	2	Avoid classes / interfaces with too many lines of code	2
n°7	Do not instantiate temporal Objects in loops bodies	1	Avoid excesive import lines	2
n°8	Avoid unused local variables	1	Prevents the use of some ones of the String class constructors	1
n°9	Prevents the use of some ones of the String class constructors	1	Avoid using numeric literals	58
n°10	Avoid using numeric literals	58	Avoid duplicate literals	28

Label Scansione	tag 9F34 Fastbank isssbe		tag 9F34 Fastbank isssbe	
Data scansione	17/07/2018		17/07/2018	
LOC	2.109		2.148	
Risk index	**93,91**		**97,88**	
Global indicator	**77,5**		**76,78**	
Efficiency	**99,96**		99,96	
Maintainability	3,29		0,21	
Portability	100		100	
Reliability	99,99		99,99	
Security	100		100	
Total defects	**311**	**0,147**	**318**	**0,148**
Very high	5		9	
High	27		23	
Normal	189		194	
Low	78		80	
Very low	12		12	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Duplicated code: big block	5	Duplicated code: big block	9
n°2	Avoid nested IF sentences with too many levels	7	Avoid nested IF sentences with too many levels	7
n°3	Cyclomatic complexity	6	Cyclomatic complexity	6
n°4	Duplicated code: medium block	4	Maximum allowed number of methods	3
n°5	Maximum allowed number of methods	3	Provide Javadoc comments for public fields	3
n°6	Provide Javadoc comments for public fields	3	Maximum allowed number of fields	2
n°7	Maximum allowed number of fields	2	Date format class java.text.SimpleDateFormat spends many resources	1
n°8	Date format class java.text.SimpleDateFormat spends many resources	1	Avoid classes / interfaces with too many lines of code	1
n°9	Avoid classes / interfaces with too many lines of code	1	Avoid using numeric literals	106
n°10	Avoid using numeric literals	103	Avoid field names that differ only in case	53

Label Scansione	11/07/2018 08:56		11/07/2018 09:24	
Data scansione	11/07/2018		11/07/2018	
LOC	2.155		2.158	
Risk index	**78,5**		**78,25**	
Global indicator	**66,3**		**66,46**	
Efficiency	65,82		65,96	
Maintainability	70,78		71,03	
Portability	100		100	
Reliability	56,51		56,57	
Security	55,13		55,3	
Total defects	**503**	**0,233**	**502**	**0,233**
Very high	1		1	
High	98		98	
Normal	308		307	
Low	46		46	
Very low	50		50	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Use of Hard-coded Credentials	1	Use of Hard-coded Credentials	1
n°2	Initialize all static fields	22	Initialize all static fields	22
n°3	Avoid catch blocks with empty bodies	17	Avoid catch blocks with empty bodies	17
n°4	Avoid static collections; they can grow without bounds	11	Avoid static collections; they can grow without bounds	11
n°5	Avoid using method calls in a loop	7	Avoid using method calls in a loop	7
n°6	Avoid unused private methods and constructors	5	Avoid unused private methods and constructors	5
n°7	Avoid using try statements in loops	4	Avoid using try statements in loops	4
n°8	Do not instantiate temporal Objects in loops bodies	3	Do not instantiate temporal Objects in loops bodies	3
n°9	Avoid using toString method in a String	3	Avoid using toString method in a String	3
n°10	Date format class java.text.SimpleDateFormat spends many resources	3	Date format class java.text.SimpleDateFormat spends many resources	3

Fig.3.21. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Label Scansione	Abas 2.16		Abas 2.16	
Data scansione	02/07/2018		03/07/2018	
LOC	378		356	
Risk index	**53,12**		**0,27**	
Global indicator	**93,36**		**96,61**	
Efficiency	95,82		98,94	
Maintainability	78,74		86,3	
Portability	100		100	
Reliability	100		99,99	
Security	100		100	
Total defects	**30**	**0,079**	**22**	**0,062**
Very high	0		0	
High	5		3	
Normal	22		18	
Low	2		1	
Very low	1		0	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid using numeric literals	16	Avoid using numeric literals	11
n°2	Follow the limit for number of statements in a metho	4	Follow the limit for number of statements in a method	4
n°3	Avoid excesive import lines	1	Classes internally strongly coupled must be avoided	1
n°4	Do not instantiate temporal Objects in loops bodies	1	Avoid using components calling too many other components	1
n°5	Date format class java.text.SimpleDateFormat spends many resources	1	Avoid nested IF sentences with too many levels	1
n°6	Avoid nested IF sentences with too many levels	1	Avoid excesive import lines	1
n°7	Avoid using components calling too many other components	1	Avoid class or interface names too long	
n°8	Classes internally strongly coupled must be avoided	1	Avoid TODO comments in production code	1
n°9	Provide Javadoc comments for public methods in Java classes (non-interface)	1		
n°10	Provide Javadoc comments for private methods	1		

Sviluppatore 6

Label Scansione	ABAS bug fixing DisposalResenderThread		ABAS bug fixing DisposalResenderThread	
Data scansione	20/06/2018		20/06/2018	
LOC	308		325	
Risk index	**99,98**		**71,09**	
Global indicator	**72,35**		**88,78**	
Efficiency	23,08		98,59	
Maintainability	42,86		52,87	
Portability	100		100	
Reliability	99,9		100	
Security	99,93		100	
Total defects	**52**	**0,169**	**25**	**0,077**
Very high	0		0	
High	14		6	
Normal	29		17	
Low	8		1	
Very low	1		1	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using numeric literals	11	Avoid using numeric literals	11
n°2	Do not instantiate temporal Objects in loops bodies	6	Cyclomatic complexity	2
n°3	Declare the List and Set variables with their interface	4	Avoid nested IF sentences with too many levels	2
n°4	Cyclomatic complexity	2	Follow the limit for number of statements in a metho	2
n°5	Avoid using method calls in a loop	2	Follow the limit for number of lines in a method	2
n°6	Initialize all local variables at the declaration statement	2	Avoid excesive import lines	1
n°7	Follow the limit for number of statements in a method	2	Date format class java.text.SimpleDateFormat spends many resources	1
n°8	Follow the limit for number of lines in a method	2	Date format class java.text.SimpleDateFormat spends many resources	1
n°9	Loop with Unreachable Exit Condition ('Infinite Loop'	2	Classes internally strongly coupled must be avoided	1
n°10	Do not check the length of a series by invoking the St	2	Avoid class or interface names too long	1

Label Scansione	Active_tpay		Active_tpay	
Data scansione	15/05/2018		15/05/2018	
LOC	251		252	
Risk index	**0,85**		**0,83**	
Global indicator	**92,16**		**93,37**	
Efficiency	99,72		99,76	76,6%
Maintainability	66,53		71,7	18,93%
Portability	100		100	0,0%
Reliability	100		99,97	0,1%
Security	100		100	0,1%
Total defects	**16**		**15**	**0,060**
Very high	0		0	
High	3		3	
Normal	9		9	
Low	3		3	
Very low	1		0	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Follow the limit for number of statements in a method	3	Follow the limit for number of statements in a method	3
n°2	Only use Hibernate/JPA for database access	3	Only use Hibernate/JPA for database access	3
n°3	Avoid class or interface names too long	3	Avoid class or interface names too long	3
n°4	Provide Javadoc comments for public classes and interfaces	1	Provide Javadoc comments for public classes and inte	1
n°5	Cyclomatic complexity	1	Cyclomatic complexity	1
n°6	Avoid nested IF sentences with too many levels	1	Avoid nested IF sentences with too many levels	1
n°7	Declare the List and Set variables with their interface type	1	Avoid using components calling too many other components	1
n°8	Avoid using components calling too many other components	1	Classes internally strongly coupled must be avoided	1
n°9	Avoid using components calling too many other components	1	Always check object type before cast	1
n°10	Classes internally strongly coupled must be avoided	1		

Label Scansione	HBAS pin services Credem		HBAS pin services Credem	
Data scansione	29/10/2018		29/10/2018	
LOC	380		422	
Risk index	**0,01**		**0**	
Global indicator	**94,84**		**98,46**	
Efficiency	99,73		99,73	
Maintainability	84,17		93,76	
Portability	100		100	
Reliability	93,85		99,85	
Security	100		100	
Total defects	**39**	**0,103**	**35**	**0,083**
Very high	0		0	
High	4		1	
Normal	29		29	
Low	6		5	
Very low	0		0	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid using numeric literals	10	Avoid using numeric literals	10
n°2	Only use Hibernate/JPA for database access	5	Only use Hibernate/JPA for database access	6
n°3	Avoid using components calling too many other comp	4	Avoid using components calling too many other comp	6
n°4	Always use specific exceptions in the throws clause	4	Always use specific exceptions in the throws clause	6
n°5	Avoid Exception, RuntimeException o Throwable in catch or throw statements	4	Avoid Exception, RuntimeException o Throwable in catch or throw statements	4
n°6	Avoid Exception, RuntimeException o Throwable in catch or throw statements	3	Always use specific exceptions in the throws clause	4
n°7	Provide the @throws tag in Javadoc comments for public and protected methods	3	Follow the limit for number of statements in a metho	3
n°8	Avoid catch blocks with empty bodies	2	Cyclomatic complexity	1
n°9	Cyclomatic complexity	1	Avoid class or interface names too long	1
n°10	Provide Javadoc comments for public methods in Jav	1		

Label Scansione	PPW_2.5.2		PPW_2.5.2	
Data scansione	**26/02/2018**		26/02/2018	
LOC	9437		9.504	
Risk index	96,92		**96,78**	
Global indicator	44,88		**44,87**	
Efficiency	41,4		41,13	
Maintainability	0,31		0,32	
Portability	100		100	
Reliability	89,73		89,56	
Security	19,85		20,17	
Total defects	1394	**0,148**	**1.407**	**0,148**
Very high	65		65	
High	210		214	
Normal	761		764	
Low	270		276	
Very low	88		88	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid using numeric literals	132	Avoid using numeric literals	134
n°2	Follow the limit for number of statements in a metho	75	Follow the limit for number of statements in a method	75
n°3	Initialize all local variables at the declaration statement	57	Initialize all local variables at the declaration stateme	57
n°4	Provide Javadoc comments for public methods in Java classes (non-interface)	52	Provide Javadoc comments for public methods in Java classes (non-interface)	53
n°5	Avoid Exception, RuntimeException o Throwable in catch or throw statements	51	Avoid Exception, RuntimeException o Throwable in catch or throw statements	51
n°6	Avoid using components calling too many other components	47	Avoid using components calling too many other comp	47
n°7	Cyclomatic complexity	44	Cyclomatic complexity	44
n°8	Always use specific exceptions in the throws claus	44	Always use specific exceptions in the throws clause	44
n°9	Avoid methods with too many parameters	43	Avoid methods with too many parameters	43
n°10	Avoid field names that differ only in case	37	Avoid field names that differ only in case	37

Label Scansione	Wire BPB		Wire BPB	
Data scansione	03/10/2018		06/11/2018	
LOC	646		1.221	
Risk index	**2,39**		**0,03**	
Global indicator	**71,07**		**78,77**	
Efficiency	99,15		99,6	0,0%
Maintainability	93,31		96,87	10,23%
Portability	100		100	0,0%
Reliability	82,96		99,85	6,0%
Security	0		12,25	0,0%
Total defects	**57**	**0,088**	**59**	**0,048**
Very high	2		1	
High	11		5	
Normal	34		44	
Low	9		9	
Very low	1		0	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Initialize all static fields	7	Always check object type before cast	12
n°2	Do not define 'Object' variables	7	Avoid using components calling too many other components	11
n°3	Avoid using components calling too many other components	6	Only use Hibernate/JPA for database acces	10
n°4	Always check object type before cast	6	Do not define 'Object' variables	5
n°5	Instantiate StringBuffer/StringBuilder with explicit initial size	4	Avoid class or interface names too long	4
n°6	Avoid returning Java.lang.Object, instead convert it to a specific typ	3	Avoid returning Java.lang.Object, instead convert it to a specific type	2
n°7	Only use Hibernate/JPA for database access	3	XML entity injection	1
n°8	Provide the @throws tag in Javadoc comments for public and protected methods	3	Do not cast a variable to an interface implemented by that variable or to the base class that extends it	1
n°9	XML entity injection	2	Cyclomatic complexity	1
n°10	Use constructors than initialize all the fields in a class	2	Avoid excesive import lines	1

Label Scansione	SERSI - 1.0.0		SERSI - 1.0.0	
Data scansione	18/01/2019		04/02/2019	
LOC	1.716		1.912	
Risk index	**0**		**0**	
Global indicator	**98,8**		**99,95**	
Efficiency	99,8		99,9	
Maintainability	94,99		99,88	
Portability	100		100	
Reliability	99,99		99,99	
Security	99,8		99,97	
Total defects	**24**	**0,014**	**23**	**0,012**
Very high	1		0	
High	5		4	
Normal	18		18	
Low	0		1	
Very low	0		0	

Top 10 Defect			**Rule**	**Defects**
n°1	Always check object type before cast	6	Always check object type before cast	5
n°2	Always normalize system inputs	4	Always normalize system inputs	4
n°3	Serializable Class Containing Sensitive Data	3	Serializable Class Containing Sensitive Data	3
n°4	A package does not depend on less stable packages	3	A package does not depend on less stable packages	2
n°5	A package does not depend on less stable packages	2	Incorrect Behavior Order: Validate Before Canonicalize	2
n°6	Incorrect Behavior Order: Validate Before Canonicaliz	2	EJB Bad Practices: Use of Java I/O	2
n°7	Avoid inaccessible classes	1	Avoid methods that invoke only overwritten supermethods	1
n°8	Avoid unused private methods and constructors	1	Avoid unused private methods and constructors	1
n°9	Avoid static collections; they can grow without bounds	1	Cyclomatic complexity	1
n°10			Avoid static collections; they can grow without bounds	1

Label Scansione	SERSI - sviluppo bollettini postali		SERSI - sviluppo bollettini postali	
Data scansione	**17/01/2019**		17/01/2019	
LOC	2021		2.097	
Risk index	0		**0**	
Global indicator	99,12		**99,34**	
Efficiency	99,93		99,93	
Maintainability	96,74		97,25	
Portability	100		100	
Reliability	99,57		99,99	
Security	99,98		99,99	
Total defects	45	**0,022**	**27**	**0,013**
Very high	1		1	
High	8		4	
Normal	36		22	
Low	0		0	
Very low	0		0	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Always check object type before cast	19	Always check object type before cast	6
n°2	Always normalize system inputs	4	Always normalize system inputs	4
n°3	Serializable Class Containing Sensitive Data	3	Counts the number of method calls	3
n°4	Count the numbers of method call	3	A package does not depend on less stable packages	3
n°5	Provide a break or return statement for the default label in a switch	2	Serializable Class Containing Sensitive Data	3
n°6	A package does not depend on less stable packages	2	Incorrect Behavior Order: Validate Before Canonicalize	2
n°7	Incorrect Behavior Order: Validate Before Canonicali	2	Incorrect Behavior Order: Validate Before Canonicalize	1
n°8	EJB Bad Practices: Use of Java I/O	2	Avoid static collections; they can grow without bounds	1
n°9	Avoid inaccessible classes	1	Avoid static collections; they can grow without boun	1
n°10	Avoid unused imports	1	Avoid static collections; they can grow without boun	1

Fig.3.22. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 7

Label Scansione	NAG		NAG4	
Data scansione	15/03/2018		15/03/2018	
LOC	807		924	
Risk index	**21,95**		**33,82**	
Global indicator	**80,02**		**78,02**	
Efficiency	99,98		99,98	
Maintainability	14,63		6,76	
Portability	100		100	
Reliability	99,47		99,52	
Security	100		100	
Total defects	**62**	**0,077**	**75**	**0,081**
Very high	2		3	
High	6		6	
Normal	35		41	
Low	14		6	
Very low	5		19	

	NAG		NAG4	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Follow the limit for number of statements in a method	11	Avoid putting non-Javadoc comments between Javadoc and class / method declarations	14
n°2	Avoid using numeric literals	9	Follow the limit for number of statements in a method	12
n°3	Always check object type before cast	4	Avoid using numeric literals	12
n°4	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	4	Always check object type before cast	5
n°5	Provide a branch block for 'if' statements	3	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	4
n°6	Cyclomatic complexity	2	Cyclomatic complexity	3
n°7	Avoid nested IF sentences with too many levels	2	Avoid nested IF sentences with too many levels	2
n°8	Avoid using components calling too many other components	2	Avoid using components calling too many other components	2
n°9	Classes internally strongly coupled must be avoided	2	Classes internally strongly coupled must be avoided	2
n°10	Only use Hibernate/JPA for database access	2	Only use Hibernate/JPA for database access	2

Label Scansione	Wire core Soap XML Personal data		Wire core Soap XML Personal data 2	
Data scansione	13/06/2018		13/06/2018	
LOC	155		159	
Risk index	**98,25**		**58,75**	
Global indicator	**91,67**		**95,48**	
Efficiency	96,04		96,14	
Maintainability	67,66		83,92	
Portability	100		100	
Reliability	99,71		99,71	
Security	100		100	
Total defects	**14**	**0,090**	**13**	**0,082**
Very high	0		0	
High	4		3	
Normal	8		7	
Low	2		2	
Very low	0		1	

	Wire core Soap XML Personal data		Wire core Soap XML Personal data 2	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	2	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	2
n°2	Follow the limit for number of statements in a method	2	Cyclomatic complexity	1
n°3	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	2	Avoid using method calls in a loop	1
n°4	Avoid using method calls in a loop	1	Redeclare a class with only abstract methods and static final fields as an interface	1
n°5	Redeclare a class with only abstract methods and static final fields as an interface	1	Avoid using components calling too many other components	1
n°6	Avoid using components calling too many other components	1	Classes internally strongly coupled must be avoided	1
n°7	Classes internally strongly coupled must be avoided	1	Follow the limit for number of statements in a method	1
n°8	Declare classes where all constructors are private as 'final'	1	Declare classes where all constructors are private as 'final'	1
n°9	Do not check the length of a series by invoking the String.equals('')	1	Do not check the length of a series by invoking the String.equals('')	1
n°10	Provide a by default private constructor in utility classes	1	Provide a by default private constructor in utility classes	1

Label Scansione	ISP Statistiche 1		ISP Statistiche 3	
Data scansione	07/09/2018			
LOC	217		229	
Risk index	**0,3**		**0**	
Global indicator	**90,61**		**99,94**	
Efficiency	98,03		100	0,0%
Maintainability	69,61		99,76	−116,42%
Portability	100		100	0,0%
Reliability	91,61		100	0,1%
Security	100		100	0,0%
Total defects	**40**		**14**	**0,061**
Very high	0		0	
High	7		0	
Normal	11		4	
Low	14		6	
Very low	8		4	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Provide a branch block for 'if' statements	4	Use constructors than initialize all the fields in a class	3
n°2	Use constructors than initialize all the fields in a class	3	Make methods static if they do not use instance class members	2
n°3	Avoid unused imports	2	Always follow the java method naming conventions	2
n°4	Avoid calling non-final, non-static and non-private methods from constructors	2	Avoid incorrect name format in local variables	2
n°5	Define class attributes at the beginning of the class	2	Avoid using components calling too many other components	1
n°6	Make methods static if they do not use instance class members	2	Provide the @throws tag in Javadoc comments for public and protected methods	1
n°7	Follow the limit for number of return statements	2	Follow the limit for number of statements in a method	1
n°8	Use a branches block for 'else' statements	2	Control number of exceptions in a clause	1
n°9	Always follow the java method naming conventions	2	Provide a by default private constructor in utility classes	1
n°10	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	2		

Fig.3.23. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 8

Label Scansione	V06.19.00-TFS-79294_01		V06.19.00-TFS-79294_03	
Data scansione	19/01/2018		19/01/2018	
LOC	1.168		1.203	
Risk index	**10,48**		**0**	
Global indicator	**56,37**		**99,46**	
Efficiency	98,57		99,47	
Maintainability	15,8		99,06	
Portability	96,44		98,67	
Reliability	99,18		99,7	
Security	0,35		100	
Total defects	**100**	**0,086**	**65**	**0,054**
Very high	2		0	
High	23		8	
Normal	37		36	
Low	28		19	
Very low	10		2	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid creating or assigning a variable within a loop	12	Avoid creating or assigning a variable within a loop	12
n°2	Always check object type before cast	9	Always check object type before cast	9
n°3	Provide Javadoc comments for public classes and interfaces	7	Avoid using components calling too many other components	6
n°4	Provide Javadoc comments for public fields	7	Classes internally strongly coupled must be avoided	5
n°5	Provide Javadoc comments for public methods in Java classes (non-interface)	7	Do not use Runtime and System classes	4
n°6	Avoid using components calling too many other components	6	Old commented code	4
n°7	Classes internally strongly coupled must be avoided	5	Follow the limit for number of statements in a method	3
n°8	Provide Javadoc comments for protected fields	5	Provide Javadoc comments for public methods in Java classes (non-interface)	3
n°9	Provide Javadoc comments for private methods	5	Date format class java.text.SimpleDateFormat spends many resources	2
n°10	Do not use Runtime and System classes	4	Declare the List and Set variables with their interface type	2

Label Scansione	V06.22.00-TFS-96538_01		V06.22.00-TFS-96538_03	
Data scansione	28/08/2018		28/08/2018 17:15	
LOC	821		830	
Risk index	**5,09**		**0,01**	
Global indicator	**86,6**		**96,34**	
Efficiency	99,75		99,76	
Maintainability	43,16		84,51	
Portability	100		100	
Reliability	99,44		99,94	
Security	100		100	
Total defects	**48**	**0,058**	**45**	**0,054**
Very high	1		0	
High	8		7	
Normal	31		30	
Low	5		5	
Very low	3		3	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Do not declare private fields that are used in only one method	10	Do not declare private fields that are used in only one method	10
n°2	Follow the limit for number of statements in a method	9	Follow the limit for number of statements in a method	9
n°3	Avoid nested IF sentences with too many levels	3	Avoid nested IF sentences with too many levels	3
n°4	Always check object type before cast	2	Always check object type before cast	3
n°5	Avoid using components calling too many other components	2	Avoid using components calling too many other components	2
n°6	Classes internally strongly coupled must be avoided	2	Classes internally strongly coupled must be avoided	2
n°7	Avoid creating or assigning a variable within a loop	2	Avoid creating or assigning a variable within a loop	2
n°8	Cyclomatic complexity	1	Avoid using too many 'non-final static' fields	1
n°9	Avoid using too many 'non-final static' fields	1	Avoid unused private methods and constructors	1
n°10	Avoid unused private methods and constructors	1	Maximum allowed number of fields	1

Fig.3.24. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 9

Label Scansione	V06.19.00-TFS-79115_01		V06.19.00-TFS-79115_05	
Data scansione	10/01/2018		19/01/2018	
LOC	1.135		1.242	
Risk index	**67,03**		**0**	
Global indicator	**69,06**		**97,38**	
Efficiency	66,25		93,16	
Maintainability	0		97,82	
Portability	99,99		100	
Reliability	93,95		96,39	
Security	100		100	
Total defects	**214**	**0,189**	**169**	**0,136**
Very high	5		0	
High	23		11	
Normal	120		107	
Low	54		41	
Very low	12		10	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Avoid using numeric literals	32	Always check object type before cast	29
n°2	Always check object type before cast	27	Avoid modifications on method or constructor parameters	16
n°3	Do not check the length of a series by invoking the String.equals('')	13	Do not check the length of a series by invoking the String.equals('')	12
n°4	Avoid creating or assigning a variable within a loop	12	Avoid creating or assigning a variable within a loop	9
n°5	Avoid field names that differ only in case	11	Provide Javadoc comments for public methods in Java classes (non-interface)	6
n°6	Avoid using variables with the same name	11	Follow the limit for number of statements in a method	6
n°7	Provide a branch block for 'if' statements	7	Avoid using variables with the same name	6
n°8	Provide Javadoc comments for public classes and interfaces	6	Avoid using numeric literals	6
n°9	Provide Javadoc comments for public fields	6	Avoid using components calling too many other components	6
n°10	Avoid using components calling too many other components	6	Avoid field names that differ only in case	6

Label Scansione	VB06.20.00_TFS-80395_01		VB06.20.00_TFS-80395_02	
Data scansione	31/01/2018		06/02/2018	
LOC	443		468	
Risk index	**0**		**0**	
Global indicator	**97,63**		**98,82**	
Efficiency	99,9		99,91	
Maintainability	93,73		98,36	
Portability	100		100	
Reliability	96,14		96,65	
Security	100		100	
Total defects	**63**	**0,142**	**65**	**0,139**
Very high	0		0	
High	2		0	
Normal	41		46	
Low	15		15	
Very low	5		4	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Always check object type before cast	11	Always check object type before cast	11
n°2	Avoid modifications on method or constructor parameters	10	Avoid modifications on method or constructor parameters	10
n°3	Avoid using components calling too many other components	3	Old commented code	4
n°4	Avoid using numeric literals	3	Avoid using components calling too many other components	3
n°5	Avoid field names that differ only in case	3	Avoid using numeric literals	3
n°6	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	3	Avoid field names that differ only in case	3
n°7	Do not check the length of a series by invoking the String.equals('')	3	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	3
n°8	Avoid using variables with the same name	3	Do not check the length of a series by invoking the String.equals('')	3
n°9	Provide Javadoc comments for public methods in Java classes (non-interface)	3	Avoid using variables with the same name	3
n°10	Serialization of fields of a 'Bean' class	3	Provide Javadoc comments for public methods in Java classes (non-interface)	3

Label Scansione	V06.20.00-TFS-83000_01		V06.20.00-TFS-83000_02	
Data scansione	27/02/2018		27/02/2018	
LOC	798		774	
Risk index	**19,33**		**0**	
Global indicator	**79,17**		**98,32**	
Efficiency	99,54		99,91	28,9%
Maintainability	19,4		98,61	100,00%
Portability	99,99		99,99	0,0%
Reliability	91,4		94,23	2,5%
Security	100		100	0,0%
Total defects	**177**		**112**	**0,145**
Very high	2		0	
High	4		2	
Normal	135		75	
Low	24		23	
Very low	12		12	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using numeric literals	33	Always check object type before cast	24
n°2	Always check object type before cast	24	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	7
n°3	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	15	Avoid using components calling too many other components	6
n°4	Do not check the length of a series by invoking the String.equals('')	13	Avoid modifications on method or constructor parameters	6
n°5	Old commented code	12	Provide Javadoc comments for public methods in Java classes (non-interface)	6
n°6	Avoid using components calling too many other components	6	Follow the limit for number of return statements	6
n°7	Avoid modifications on method or constructor parameters	6	Justify the opening and closing braces in code blocks	6
n°8	Provide Javadoc comments for public methods in Java classes (non-interface)	6	Avoid field names that differ only in case	5
n°9	Justify the opening and closing braces in code blocks	6	Do not check the length of a series by invoking the String.equals('')	5
n°10	Avoid field names that differ only in case	5	Avoid using variables with the same name	5

Label Scansione	V06.21.00-TFS-87989-JS_01		V06.21.00-TFS-87989-JS_02	
Data scansione	20/04/2018		20/04/2018	
LOC	110		110	
Risk index	**0,33**		**0,33**	
Global indicator	**94,26**		**94,35**	
Efficiency	100		100	
Maintainability	75,89		76,29	
Portability	100		100	
Reliability	99,43		99,43	
Security	100		100	
Total defects	**11**	**0,100**	**10**	**0,091**
Very high	0		0	
High	1		1	
Normal	1		1	
Low	9		8	
Very low	0		0	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using logical comparators == and !=	5	Avoid using logical comparators == and !=	5
n°2	Merge nested if statements using a short-circuit operator	2	Merge nested if statements using a short-circuit operator	2
n°3	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°4	Always specify a radix when using parseInt	1	Always specify a radix when using parseInt	1
n°5	lace whitespaces between logical operators and its operands	1	Avoid using Array and Object constructors	1
n°6	Avoid using Array and Object constructors	1		
n°7				
n°8				
n°9				
n°10				

Fig.3.25. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 10

Label Scansione	V06.19.00_TFS-77678_1		V06.19.00_TFS-77678_2	
Data scansione	19/01/2018		19/01/2018	
LOC	145		148	
Risk index	**80,58**		**0**	
Global indicator	**76,95**		**99,79**	
Efficiency	99,84		99,85	
Maintainability	1,2		99,36	
Portability	100		100	
Reliability	99,84		99,85	
Security	100		100	
Total defects	**12**	**0,083**	**11**	**0,074**
Very high	1		0	
High	0		0	
Normal	8		8	
Low	1		1	
Very low	2		2	

	V06.19.00_TFS-77678_1		V06.19.00_TFS-77678_2	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	1	Avoid using components calling too many other components	1
n°2	Avoid using components calling too many other components	1	Only use Hibernate/JPA for database access	1
n°3	Only use Hibernate/JPA for database access	1	Close JDBC resources when finishing using	1
n°4	Close JDBC resources when finishing using	1	Follow the limit for number of statements in a method	1
n°5	Follow the limit for number of statements in a method	1	Avoid field names that differ only in case	1
n°6	Avoid field names that differ only in case	1	Avoid unused fields	1
n°7	Avoid unused fields	1	Avoid duplicate literals	1
n°8	Avoid duplicate literals	1	Avoid modifications on method or constructor parameters	1
n°9	Avoid modifications on method or constructor parameters	1	Avoid using variables with the same name	1
n°10	Avoid using variables with the same name	1	Avoid incorrect name format in the final static fields	1

Label Scansione	V06.20.00_TFS-83074		V06.20.00_TFS-83074_2	
Data scansione	19/02/2018		19/02/2018	
LOC	1854		1855	
Risk index	**0,9**		**0**	
Global indicator	**92,15**		**98,22**	
Efficiency	95,9		95,99	
Maintainability	71,8		97,89	
Portability	99,96		99,96	
Reliability	97,74		97,69	
Security	100		100	
Total defects	**179**	**0,097**	**177**	**0,095**
Very high	2		0	
High	17		15	
Normal	113		113	
Low	41		43	
Very low	6		6	

	V06.20.00_TFS-83074		V06.20.00_TFS-83074_2	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Always check object type before cast	30	Always check object type before cast	30
n°2	Avoid modifications on method or constructor parameters	16	Avoid modifications on method or constructor parameters	16
n°3	Avoid using components calling too many other components	9	Avoid using components calling too many other components	9
n°4	Provide Javadoc comments for public methods in Java classes (non-interface)	8	Provide Javadoc comments for public methods in Java classes (non-interface)	8
n°5	Classes internally strongly coupled must be avoided	7	Classes internally strongly coupled must be avoided	7
n°6	Follow the limit for number of statements in a method	7	Follow the limit for number of statements in a method	7
n°7	Do not check the length of a series by invoking the String.equals('')	7	Old commented code	6
n°8	Old commented code	6	Do not check the length of a series by invoking the String.equals('')	6
n°9	Date format class java.text.SimpleDateFormat spends many resources	5	Date format class java.text.SimpleDateFormat spends many resources	5
n°10	Avoid creating or assigning a variable within a loop	5	Avoid creating or assigning a variable within a loop	5

Label Scansione	V06.20.00_TFS-83720		V06.20.00_TFS-83720_1	
Data scansione	12/03/2018		12/03/2018	
LOC	1667		1722	
Risk index	**0,04**		**0**	
Global indicator	**75,7**		**99,73**	
Efficiency	99,86		99,98	0,0%
Maintainability	95,71		98,86	98,79%
Portability	100		100	0,0%
Reliability	99,91		99,99	0,0%
Security	0		100	0,0%
Total defects	**194**		**180**	**0,105**
Very high	4		0	
High	7		0	
Normal	68		65	
Low	77		77	
Very low	38		38	

	V06.20.00_TFS-83720		V06.20.00_TFS-83720_1	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using variables with the same name	24	Avoid field names that differ only in case	24
n°2	Avoid field names that differ only in case	24	Avoid unused parameters	24
n°3	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	24	Avoid using variables with the same name	24
n°4	Avoid incorrect name format in the final static fields	14	Avoid incorrect name format in the final static fields	14
n°5	Only declare 'protected' constructors for abstract classes	14	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	14
n°6	Avoid using components calling too many other components	12	Avoid using components calling too many other components	12
n°7	Justify the opening and closing braces in code blocks	12	Only declare 'protected' constructors for abstract classes	12
n°8	Avoid using numeric literals	8	Justify the opening and closing braces in code blocks	8
n°9	Instantiate StringBuffer/StringBuilder with explicit initial size	7	Avoid using numeric literals	7
n°10	Use of Hard-coded Credentials	5	Instantiate StringBuffer/StringBuilder with explicit initial size	5

Label Scansione	WSNOTIFYBACKEND		WSNOTIFYBACKEND_7	
Data scansione	27/04/2018		27/04/2018	
LOC	3.634		3.646	
Risk index	**2,79**		**0**	
Global indicator	**59,06**		**99,55**	
Efficiency	98,09		99,76	
Maintainability	52,22		98,48	
Portability	9,56		100	
Reliability	72,91		99,77	
Security	0,56		100	
Total defects	**361**	**0,099**	**263**	**0,072**
Very high	6		0	
High	62		12	
Normal	105		107	
Low	114		79	
Very low	74		65	

	WSNOTIFYBACKEND		WSNOTIFYBACKEND_7	
Top 10 Defect	**Rule**	**Defects**	**Rule**	Defects
n°1	Provide Javadoc comments for public classes and interfaces	30	Avoid incorrect name format in the final static fields	25
n°2	Provide Javadoc comments for public methods in Java classes (non-interface)	30	Avoid using components calling too many other components	23
n°3	Avoid incorrect name format in the final static fields	25	Close JDBC resources when finishing using	17
n°4	Avoid using components calling too many other components	23	Avoid incorrect name format in local variables	16
n°5	Close JDBC resources when finishing using	17	Follow the limit for number of return statements	14
n°6	Avoid incorrect name format in local variables	16	Do not use return statements inside catch blocks	14
n°7	Follow the limit for number of return statements	14	Avoid unnecessary constructors	13
n°8	Do not use return statements inside catch blocks	14	Follow the limit for number of statements in a method	11
n°9	Avoid unnecessary constructors	13	Classes internally strongly coupled must be avoided	9
n°10	Follow the limit for number of statements in a method	12	Avoid cyclic dependencies between packages	9

Label Scansione	WSNOTIFYCOMMON		WSNOTIFYCOMMON_2	
Data scansione	**27/04/2018**		11/05/2018	
LOC	1305		1.360	
Risk index	0		**0**	
Global indicator	**100**		**100**	
Efficiency	100		100	
Maintainability	99,99		99,99	
Portability	100		100	
Reliability	99,99		99,99	
Security	100		100	
Total defects	104	**0,080**	**105**	**0,077**
Very high	0		0	
High	0		0	
Normal	8		8	
Low	73		74	
Very low	23		23	

	WSNOTIFYCOMMON		WSNOTIFYCOMMON_2	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Make methods static if they do not use instance class members	41	Make methods static if they do not use instance class members	41
n°2	Provide Javadoc comments for protected fields	21	Provide Javadoc comments for protected fields	21
n°3	Avoid incorrect name format in the final static fields	14	Avoid incorrect name format in the final static fields	14
n°4	Avoid unnecessary constructors	9	Avoid unnecessary constructors	9
n°5	Declare classes where all constructors are private as 'final'	6	Declare classes where all constructors are private as 'final'	6
n°6	Avoid empty classes or intefaces	6	Avoid cyclic dependencies between packages	6
n°7	Avoid cyclic dependencies between packages	5	Avoid empty classes or intefaces	6
n°8	A package does not depend on less stable packages	2	A package does not depend on less stable packages	2
n°9				
n°10				

Label Scansione	V06.22.00_TFS-75587		V06.22.00_TFS-75587_10	
Data scansione	17/07/2018		17/07/2018	
LOC	439		334	
Risk index	**81,54**		**0**	
Global indicator	**66,55**		**98,76**	
Efficiency	99,7		99,77	1,7%
Maintainability	75,46		94,87	46,97%
Portability	63,08		100	90,4%
Reliability	99,39		100	26,9%
Security	0		100	99,4%
Total defects	**131**	**0,298**	**31**	**0,093**
Very high	1		0	
High	6		0	
Normal	111		29	
Low	12		1	
Very low	1		1	

	V06.22.00_TFS-75587		V06.22.00_TFS-75587_10	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Do not hard code character for line separator	47	Avoid using numeric literals	20
n°2	Avoid using numeric literals	35	Close JDBC resources when finishing using	4
n°3	Avoid field names that differ only in case	8	Follow the limit for number of statements in a method	2
n°4	Avoid using variables with the same name	8	Avoid using components calling too many other components	1
n°5	Follow the limit for number of statements in a method	5	Classes internally strongly coupled must be avoided	1
n°6	Do not use Runtime and System classes	4	Only use Hibernate/JPA for database access	1
n°7	Avoid duplicate literals	4	Serializable Class Containing Sensitive Data	1
n°8	Close JDBC resources when finishing using	4	Justify the opening and closing braces in code blocks	1
n°9	Avoid using components calling too many other components	2		
n°10	Instantiate StringBuffer/StringBuilder with explicit initial size	2		

Label Scansione	V06.22.00_TFS-85877		V06.22.00_TFS-85877_2	
Data scansione	20/07/2018		20/07/2018	
LOC	462		462	
Risk index	**0,03**		**0**	
Global indicator	**75,62**		**98,87**	
Efficiency	99,94		99,94	
Maintainability	96,83		96,83	
Portability	100		100	
Reliability	98,37		98,37	
Security	0		100	
Total defects	**65**	**0,141**	**64**	**0,139**
Very high	1		0	
High	0		0	
Normal	41		41	
Low	18		18	
Very low	5		5	

	V06.22.00_TFS-85877		V06.22.00_TFS-85877_2	
Top 10 Defect	**Rule**	**Defects**	**Rule**	Defects
n°1	Always check object type before cast	13	Always check object type before cast	13
n°2	Avoid using numeric literals	6	Avoid using numeric literals	6
n°3	Avoid using components calling too many other components	5	Avoid using components calling too many other components	5
n°4	Avoid field names that differ only in case	5	Avoid field names that differ only in case	5
n°5	Provide a branch block for 'if' statements	5	Provide a branch block for 'if' statements	5
n°6	Avoid using variables with the same name	5	Avoid using variables with the same name	5
n°7	Avoid modifications on method or constructor parameters	3	Avoid modifications on method or constructor parameters	3
n°8	Old commented code	3	Old commented code	3
n°9	Serialization of fields of a 'Bean' class	3	Serialization of fields of a 'Bean' class	3
n°10	Classes internally strongly coupled must be avoided	2	Classes internally strongly coupled must be avoided	2

Label Scansione	V01.13.00_TFS-98660		V01.13.00_TFS-98660_11	
Data scansione	26/09/2018		27/09/2018	
LOC	150		157	
Risk index	**21,4**		**0**	
Global indicator	**77,64**		**98**	
Efficiency	32,29		100	
Maintainability	64,28		93,62	
Portability	93,18		100	
Reliability	97,15		97,79	
Security	100		100	
Total defects	**41**	**0,273**	**21**	**0,134**
Very high	0		0	
High	5		0	
Normal	31		17	
Low	4		4	
Very low	1		0	

	V01.13.00_TFS-98660		V01.13.00_TFS-98660_11	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using numeric literals	19	Avoid using numeric literals	8
n°2	Always check object type before cast	3	Avoid Exception, RuntimeException o Throwable in catch or throw statements	3
n°3	Avoid Exception, RuntimeException o Throwable in catch or throw statements	3	Always check object type before cast	3
n°4	Prevents the use of some ones of the String class constructors	2	Avoid using components calling too many other components	2
n°5	Avoid using components calling too many other components	2	Always use specific exceptions in the throws clause	2
n°6	Follow the limit for number of statements in a method	2	Classes internally strongly coupled must be avoided	1
n°7	Always use specific exceptions in the throws clause	2	Avoid modifications on method or constructor parameters	1
n°8	Do not use the printStackTrace method	1	Avoid capturing java.lang.Exception exceptions	1
n°9	Do not use Runtime and System classes	1		
n°10	Avoid methods that invoke only overwritten supermethods	1		

Fig.3.26. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 11

Label Scansione	AMS - Standardizzazione query		AMS - Standardizzazione query	
Data scansione	05/01/2018		08/01/2018	
LOC	107		69	
Risk index	**100**		**100**	
Global indicator	**11,08**		**25,91**	
Efficiency	0,02		7,48	
Maintainability	0		5,43	
Portability	57,3		0	
Reliability	18,97		100	
Security	0		0	
Total defects	**43**	**0,402**	**10**	**0,145**
Very high	4		1	
High	13		6	
Normal	16		1	
Low	9		2	
Very low	1		0	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Do not hard code character for line separator	8	Do not use the printStackTrace method	2
n°2	Avoid using try statements in loops	3	Avoid sensitive information exposure through error messages	2
n°3	Avoid the use of reflection	3	Avoid absolute paths	1
n°4	Avoid the @return Javadoc tag in void methods	3	Provide Javadoc comments for public fields	1
n°5	Cyclomatic complexity	2	Provide Javadoc comments for public classes and interfaces	1
n°6	Java access restriction subverted (Reflection)	2	Avoid using components calling too many other components	1
n°7	Use equals() when comparing Strings	2	Provide Javadoc comments for public methods in Java classes (non-interface)	1
n°8	Avoid using numeric literals	2	Provide a by default private constructor in utility classes	1
n°9	Provide a branch block for 'if' statements	2		
n°10	Provide Javadoc comments for public fields	1		

Label Scansione	HBASv4_REL_2018_10_22		HBASv4_REL_2018_10_22	
Data scansione	26/10/2018		26/10/2018	
LOC	113		127	
Risk index	**93,21**		**8,42**	
Global indicator	**86,73**		**66,36**	
Efficiency	99,89		94,78	
Maintainability	45,43		59,73	
Portability	100		100	
Reliability	97,7		99,79	
Security	99,89		0	
Total defects	**23**	**0,204**	**9**	**0,071**
Very high	0		1	
High	2		4	
Normal	12		3	
Low	7		0	
Very low	2		1	

Top 10 Defect	**Rule**	**Defects**	**Rule**	Defects
n°1	Always use specific exceptions in the throws clause	3	Use of Hard-coded Credentials	1
n°2	Avoid Exception, RuntimeException o Throwable in catch or throw statements	3	Maximum allowed number of fields	1
n°3	Declare the List and Set variables with their interface type	2	Avoid static collections; they can grow without bounds	1
n°4	Follow the limit for number of statements in a method	2	Provide Javadoc comments for public fields	1
n°5	Use constructors than initialize all the fields in a class	2	Provide Javadoc comments for public classes and interfaces	1
n°6	With multiple (overloaded) constructors, use 'this' to call more generic constructors inside constructors	2	Declare the List and Set variables with their interface type	1
n°7	Cyclomatic complexity	1	Declare classes where all constructors are private as 'final'	1
n°8	Avoid if/else-if chains performing type testing	1	Avoid TODO comments in production code	1
n°9	Avoid using components calling too many other components	1	Provide Javadoc comments for private methods	1
n°10	Always check object type before cast	1		

Label Scansione	BatchReportTicketMonitoring		BatchReportTicketMonitoring	
Data scansione	17/01/2019		17/01/2019	
LOC	269		278	
Risk index	**3,24**		**0,72**	
Global indicator	**73,1**		**75,54**	
Efficiency	96,77		99,97	99,7%
Maintainability	97,32		97,45	100,00%
Portability	100		100	#ΔΙς/0!
Reliability	89,6		97,4	81,0%
Security	0		0	#ΔΙς/0!
Total defects	**7**		**5**	**0,018**
Very high	1		0	
High	4		1	
Normal	1		2	
Low	1		1	
Very low	0		1	
			0	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Use of Hard-coded Credentials	1	Use of Hard-coded Credentials	1
n°2	Avoid capturing NullPointerExceptions	1	Avoid capturing NullPointerExceptions	1
n°3	Do not instantiate temporal Objects in loops bodies	1	Cyclomatic complexity	1
n°4	Cyclomatic complexity	1	Only use Hibernate/JPA for database access	1
n°5	Close input and output resources in finally blocks	1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1
n°6	Only use Hibernate/JPA for database access	1		
n°7	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1		
n°8		1		
n°9				
n°10				

Fig.3.27. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 12

Label Scansione	Santadner - Advanced Navigation Menu [4]		Santadner - Advanced Navigation Menu [4]	
Data scansione	12/07/2018		12/07/2018	
LOC	78		77	
Risk index	**4,58**		**0,25**	
Global indicator	**77,53**		**92,53**	
Efficiency	58,9		69,34	
Maintainability	36,89		92,43	
Portability	100		100	
Reliability	99,37		99,99	
Security	100		100	
Total defects	**12**	**0,154**	**10**	**0,130**
Very high	0		0	
High	2		1	
Normal	7		6	
Low	3		3	
Very low	0		0	

	Santadner - Advanced Navigation Menu [4]		Santadner - Advanced Navigation Menu [4]	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Provide Javadoc comments for public classes and interfaces	1	Avoid static collections; they can grow without bounds	1
n°2	Avoid static collections; they can grow without bounds	1	Avoid using components calling too many other components	1
n°3	Avoid using components calling too many other components	1	Classes internally strongly coupled must be avoided	1
n°4	Classes internally strongly coupled must be avoided	1	Follow the limit for number of statements in a method	1
n°5	Follow the limit for number of statements in a method	1	Do not declare private fields that are used in only one method	1
n°6	Do not declare private fields that are used in only one method	1	Declare the List and Set variables with their interface type	1
n°7	Initialize all local variables at the declaration statement	1	Only use Hibernate/JPA for database access	1
n°8	Declare the List and Set variables with their interface type	1	Provide Javadoc comments for public methods in Java classes (non-interface)	1
n°9	Only use Hibernate/JPA for database access	1	Avoid capturing java.lang.Exception exceptions	1
n°10	Provide Javadoc comments for public methods in Java classes (non-interface)	1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1

Label Scansione	[0] Marketing point - Banner		[8] Marketing point - Banner	
Data scansione	23/07/2018		23/07/2018	
LOC	95		111	
Risk index	**98,91**		**0,11**	
Global indicator	**72,49**		**96,69**	
Efficiency	77,16		87,33	
Maintainability	0		95,92	
Portability	100		100	
Reliability	99,99		100	
Security	100		100	
Total defects	**12**	**0,126**	**11**	**0,099**
Very high	1		0	
High	1		1	
Normal	7		7	
Low	3		3	
Very low	0		0	

	[0] Marketing point - Banner		[8] Marketing point - Banner	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Declare the List and Set variables with their interface type	2	Declare the List and Set variables with their interface type	2
n°2	Cyclomatic complexity	1	Avoid static collections; they can grow without bounds	1
n°3	Avoid static collections; they can grow without bounds	1	Avoid using components calling too many other components	1
n°4	Avoid using components calling too many other components	1	Classes internally strongly coupled must be avoided	1
n°5	Classes internally strongly coupled must be avoided	1	Follow the limit for number of statements in a method	1
n°6	Follow the limit for number of statements in a method	1	Do not declare private fields that are used in only one method	1
n°7	Do not declare private fields that are used in only one method	1	Only use Hibernate/JPA for database access	1
n°8	Only use Hibernate/JPA for database access	1	Provide Javadoc comments for public methods in Java classes (non-interface)	1
n°9	Provide Javadoc comments for public methods in Java classes (non-interface)	1	Avoid capturing java.lang.Exception exceptions	1
n°10	Avoid capturing java.lang.Exception exceptions	1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1

Label Scansione	[0] Marketing point - ManagerInserimentoPuntoMarketing.java		[2] Marketing point - ManagerInserimentoPuntoMarketing.java	
Data scansione	23/07/2018		23/07/2018	
LOC	515		512	
Risk index	**87,37**		**77,5**	
Global indicator	**65,57**		**72,54**	
Efficiency	37,94		63,54	15,1%
Maintainability	5,08		12,45	60,09%
Portability	98,87		98,97	0,0%
Reliability	97,07		99,13	0,6%
Security	100		100	0,0%
Total defects	**100**		**63**	**0,123**
Very high	1		1	
High	11		9	
Normal	69		36	
Low	19		16	
Very low	0		1	

	[0] Marketing point - ManagerInserimentoPuntoMarketing.java		[2] Marketing point - ManagerInserimentoPuntoMarketing.java	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Do not check the length of a series by invoking the String.equals('')	26	Avoid creating or assigning a variable within a loop	13
n°2	Avoid creating or assigning a variable within a loop	13	Avoid using numeric literals	11
n°3	Avoid using numeric literals	11	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	9
n°4	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	9	Avoid using method calls in a loop	4
n°5	Always check object type before cast	7	Declare the List and Set variables with their interface type	3
n°6	Avoid using method calls in a loop	4	Avoid duplicate literals	2
n°7	Declare the List and Set variables with their interface type	3	Cyclomatic complexity	1
n°8	Avoid duplicate literals	2	Avoid nested IF sentences with too many levels	1
n°9	Avoid creating an instance of a number using new	2	Avoid excesive import lines	1
n°10	Avoid unused parameters	2	Do not use Runtime and System classes	1

Label Scansione	ManagerNavigationMenuNew.java - [0]		ManagerNavigationMenuNew.java - [2]	
Data scansione	24/03/2018		24/03/2018	
LOC	212		223	
Risk index	**55,17**		**51,23**	
Global indicator	**75,23**		**75,6**	
Efficiency	91,76		93,49	
Maintainability	0,08		0,03	
Portability	100		100	
Reliability	100		100	
Security	100		100	
Total defects	**33**	**0,156**	**31**	**0,139**
Very high	1		1	
High	1		1	
Normal	13		13	
Low	17		15	
Very low	1		1	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid creating or assigning a variable within a loop	13	Avoid creating or assigning a variable within a loop	11
n°2	Declare the List and Set variables with their interface type	7	Declare the List and Set variables with their interface type	7
n°3	Cyclomatic complexity	1	Cyclomatic complexity	1
n°4	Avoid static collections; they can grow without bounds	1	Avoid static collections; they can grow without bounds	1
n°5	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°6	Classes internally strongly coupled must be avoided	1	Classes internally strongly coupled must be avoided	1
n°7	Follow the limit for number of statements in a method	1	Follow the limit for number of statements in a method	1
n°8	Do not declare private fields that are used in only one method	1	Do not declare private fields that are used in only one method	1
n°9	Only use Hibernate/JPA for database access	1	Only use Hibernate/JPA for database access	1
n°10	Follow the limit for number of lines in a method	1	Follow the limit for number of lines in a method	1

Label Scansione	ManagerNavigationMenuUpdate - [0]		ManagerNavigationMenuUpdate - [1]	
Data scansione	27/07/2018		27/07/2018	
LOC	106		112	
Risk index	97,13		**95,65**	
Global indicator	73,3		**73,83**	
Efficiency	81,47		84,32	
Maintainability	0		0	
Portability	100		100	
Reliability	99,99		99,99	
Security	100		100	
Total defects	13	**0,123**	**12**	**0,107**
Very high	1		1	
High	1		1	
Normal	8		7	
Low	3		3	
Very low	0		0	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	1	Cyclomatic complexity	1
n°2	Avoid static collections; they can grow without bounds	1	Avoid static collections; they can grow without bounds	1
n°3	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°4	Classes internally strongly coupled must be avoided	1	Classes internally strongly coupled must be avoided	1
n°5	Follow the limit for number of statements in a method	1	Follow the limit for number of statements in a method	1
n°6	Avoid using numeric literals	1	Avoid using numeric literals	1
n°7	Do not declare private fields that are used in only one method	1	Do not declare private fields that are used in only one method	1
n°8	Declare the List and Set variables with their interface type	1	Declare the List and Set variables with their interface type	1
n°9	Only use Hibernate/JPA for database access	1	Only use Hibernate/JPA for database access	1
n°10	Too many uses of the negation operator (!) in a method	1	Provide Javadoc comments for public methods in Java classes (non-interface)	1

Fig.3.28. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 13

Label Scansione	HBas - Cabel - G20		HBas - Cabel - G20	
Data scansione	14/05/2018		14/05/2018	
LOC	122		122	
Risk index	**93,31**		**93,08**	
Global indicator	**82,77**		**83,29**	
Efficiency	98,63		99,77	
Maintainability	31,61		31,61	
Portability	100		100	
Reliability	95,38		96,71	
Security	100		100	
Total defects	**30**	**0,246**	**27**	**0,221**
Very high	0		0	
High	2		2	
Normal	20		17	
Low	8		8	
Very low	0		0	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Cyclomatic complexity	1	Cyclomatic complexity	1
n°2	Avoid nested IF sentences with too many levels	1	Avoid nested IF sentences with too many levels	1
n°3	Avoid using numeric literals	5	Avoid using numeric literals	5
n°4	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	5	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	4
n°5	Follow the limit for number of statements in a method	2	Follow the limit for number of statements in a method	2
n°6	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°7	Always use specific exceptions in the throws clause	1	Always use specific exceptions in the throws clause	1
n°8	Classes internally strongly coupled must be avoided	1	Classes internally strongly coupled must be avoided	1
n°9	Too many uses of the negation operator (!) in a method	1	Too many uses of the negation operator (!) in a method	1
n°10	Do not check the length of a series by invoking the String.equals('')	1	Only use Hibernate/JPA for database access	1

Label Scansione	HBas - Cabel - G20		HBas - Cabel - G20	
Data scansione	14/05/2018		14/05/2018	
LOC	104		104	
Risk index	**97,08**		**96,73**	
Global indicator	**82,22**		**83,55**	
Efficiency	96,68		99,79	
Maintainability	29,69		29,69	
Portability	100		100	
Reliability	96,52		99,75	
Security	100		100	
Total defects	**15**	**0,144**	**9**	**0,087**
Very high	0		0	
High	2		2	
Normal	12		6	
Low	1		1	
Very low	0		0	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	1	Cyclomatic complexity	1
n°2	Avoid nested IF sentences with too many levels	1	Avoid nested IF sentences with too many levels	1
n°3	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	3	Avoid using components calling too many other components	1
n°4	Do not check the length of a series by invoking the String.equals('')	3	Follow the limit for number of statements in a method	1
n°5	Avoid using components calling too many other components	1	Always use specific exceptions in the throws clause	1
n°6	Follow the limit for number of statements in a method	1	Classes internally strongly coupled must be avoided	1
n°7	Always use specific exceptions in the throws clause	1	Only use Hibernate/JPA for database access	1
n°8	Classes internally strongly coupled must be avoided	1	Provide the @throws tag in Javadoc comments for public and protected methods	1
n°9	Only use Hibernate/JPA for database access	1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1
n°10	Provide the @throws tag in Javadoc comments for public and protected methods	1		

Label Scansione	Iccrea Atm - Estrazione codici ECR - Hbas		Iccrea Atm - Codici ECR - Hbas	
Data scansione	24/07/2018		24/07/2018	
LOC	91		91	
Risk index	**0,76**		**0,16**	
Global indicator	**89,81**		**97,61**	
Efficiency	99,82		99,83	1,1%
Maintainability	56,33		89,87	0,00%
Portability	100		100	0,0%
Reliability	100		100	1,4%
Security	100		100	0,0%
Total defects	**10**		**9**	**0,099**
Very high	0		0	
High	1		0	
Normal	9		9	
Low	0		0	
Very low	0		0	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Provide Javadoc comments for public classes and interfaces	1	Avoid using numeric literals	5
n°2	Avoid using numeric literals	5	Follow the limit for number of statements in a method	2
n°3	Follow the limit for number of statements in a method	2	Avoid using components calling too many other components	1
n°4	Avoid using components calling too many other components	1	Only use Hibernate/JPA for database access	1
n°5	Only use Hibernate/JPA for database access	1		
n°6				
n°7				
n°8				
n°9				
n°10				

Label Scansione	Iccrea Atm - Codici ECR - Hbas		Iccrea Atm - Codici Ecr - Hbas	
Data scansione	24/07/2018		26/07/2018	
LOC	143		157	
Risk index	**97,64**		**95,31**	
Global indicator	**77,41**		**78,02**	
Efficiency	99,58		99,91	
Maintainability	3,75		6,03	
Portability	100		100	
Reliability	99,45		99,53	
Security	100		100	
Total defects	**19**	**0,133**	**18**	**0,115**
Very high	0		0	
High	5		5	
Normal	10		9	
Low	4		4	
Very low	0		0	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid nested IF sentences with too many levels	4	Avoid nested IF sentences with too many levels	4
n°2	Cyclomatic complexity	1	Cyclomatic complexity	1
n°3	Always use specific exceptions in the throws clause	2	Always use specific exceptions in the throws clause	2
n°4	Provide the @throws tag in Javadoc comments for public and protected methods	2	Provide the @throws tag in Javadoc comments for public and protected methods	2
n°5	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°6	Follow the limit for number of statements in a method	1	Follow the limit for number of statements in a method	1
n°7	Only use Hibernate/JPA for database access	1	Only use Hibernate/JPA for database access	1
n°8	Too many uses of the negation operator (!) in a method	1	Follow the limit for number of lines in a method	1
n°9	Too many uses of the negation operator (!) in a method	1	Too many uses of the negation operator (!) in a method	1
n°10	Avoid creating an instance of a number using new	1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	2

Label Scansione	T-Pay - Scheduler - Statistiche versamenti		T-Pay - Scheduler - Statistiche versamenti	
Data scansione	05/09/2018 10:52		05/09/2018	
LOC	95		95	
Risk index	**95,76**		**92,07**	
Global indicator	**86,72**		**94,45**	
Efficiency	99,91		100	
Maintainability	100		76,14	
Portability	99,83		100	
Reliability	99,99		99,99	
Security	100		100	
Total defects	12	**0,126**	**10**	**0,105**
Very high	0		0	
High	2		1	
Normal	3		2	
Low	7		7	
Very low	0		0	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	1	Cyclomatic complexity	1
n°2	Provide Javadoc comments for public fields	1	Avoid using components calling too many other components	1
n°3	Avoid using components calling too many other components	1	Follow the limit for number of statements in a method	1
n°4	Follow the limit for number of statements in a method	1	Provide a branch block for 'if' statements	3
n°5	Do not hard code character for line separator	1	Avoid class or interface names too long	1
n°6	Provide a branch block for 'if' statements	3	Provide Javadoc comments for public methods in Java classes (non-interface)	1
n°7	Avoid class or interface names too long	1	Avoid capturing java.lang.Exception exceptions	1
n°8	Provide Javadoc comments for public methods in Java classes (non-interface)	1	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1
n°9	Avoid capturing java.lang.Exception exceptions	1		
n°10	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1		

Label Scansione	CSE - #PRJ TCR - autenticazione tramite smartphone - sviluppo ppw		CSE - #PRJ TCR - autenticazione tramite smartphone - sviluppo ppw	
Data scansione	15/11/2018		15/11/2018	
LOC	249		186	
Risk index	**100**		**99,81**	
Global indicator	**47,15**		**69,17**	
Efficiency	0,19		0,27	0,3%
Maintainability	55,68		47,24	37,81%
Portability	100		100	0,0%
Reliability	96,9		99,99	0,1%
Security	0		100	0,0%
Total defects	**19**	**0,076**	**15**	**0,081**
Very high	2		1	
High	6		3	
Normal	9		7	
Low	1		3	
Very low	1		1	

	#23		#26	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Do not create Datasource variables in local methods	1	Do not create Datasource variables in local methods	1
n°2	Weak symmetric encryption algorithm	1	Cyclomatic complexity	2
n°3	Avoid using StringBuffer to store a String that is never modified	3	Avoid nested IF sentences with too many levels	1
n°4	Cyclomatic complexity	2	Follow the limit for number of statements in a method	2
n°5	Inadequate padding	2	Avoid using components calling too many other components	1
n°6	Avoid using method calls in a loop	1	Avoid duplicate literals	1
n°7	Avoid nested IF sentences with too many levels	1	Follow the limit for number of lines in a method	1
n°8	Avoid throwing RuntimeExceptions	1	Only use Hibernate/JPA for database access	1
n°9	Avoid using numeric literals	6	Avoid nesting more than 1 try / catch	1
n°10	Follow the limit for number of statements in a method	2	Use constructors than initialize all the fields in a class	1

Label Scansione	Phoenix ATM - Messaggio ATA di Ricircolo - sviluppo hbas		Phoenix ATM - Messaggio ATA di Ricircolo - sviluppo hbas	
Data scansione	21/02/2019		21/02/2019	
LOC	729		257	
Risk index	**0**		**0**	
Global indicator	**99,56**		**100**	
Efficiency	100		100	
Maintainability	98,49		100	
Portability	100		100	
Reliability	99,61		100	
Security	100		100	
Total defects	**5**	**0,007**	**0**	**0,000**
Very high	0		0	
High	3		0	
Normal	1		0	
Low	1		0	
Very low	0		0	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	2		
n°2	Provide 'default' label for each switch statement	1		
n°3	Only use Hibernate/JPA for database access	1		
n°4	Avoid Exception, RuntimeException o Throwable in catch or throw statements	1		
n°5				
n°6				
n°7				
n°8				
n°9				
n°10				

Label Scansione	Phoenix - #PRJ Bollo Auto CBILL Phoenix - sviluppi wire		Phoenix - #PRJ Bollo Auto CBILL Phoenix - sviluppi wire	
Data scansione	18/03/2019 12:18		18/03/2019	
LOC	178		178	
Risk index	**0**		**0**	
Global indicator	**98,31**		**99,13**	
Efficiency	95,54		99,96	
Maintainability	96,3		96,3	
Portability	100		100	
Reliability	100		100	
Security	100		100	
Total defects	3	**0,017**	**2**	**0,011**
Very high	0		0	
High	2		1	
Normal	1		1	
Low	0		0	
Very low	0		0	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Do not instantiate temporal Objects in loops bodies	1	Cyclomatic complexity	1
n°2	Cyclomatic complexity	1	Only use Hibernate/JPA for database access	1
n°3	Only use Hibernate/JPA for database access	1		
n°4				
n°5				
n°6				
n°7				
n°8				
n°9				
n°10				

Fig.3.29. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 14

Label Scansione	01/06/2018 09:33		01/06/2018 17:13	
Data scansione	01/06/2018		01/06/2018	
LOC	529		517	
Risk index	**1,55**		**2,65**	
Global indicator	**75,51**		**71,67**	
Efficiency	98,15		99,97	
Maintainability	79,37		80,05	
Portability	24,07		100	
Reliability	97,95		98,15	
Security	56,82		0	
Total defects	**96**	**0,181**	**108**	**0,209**
Very high	2		1	
High	10		8	
Normal	35		32	
Low	34		34	
Very low	15		33	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid absolute paths	2	Weak cryptographic hash	1
n°2	Avoid unused parameters	2	Cyclomatic complexity	1
n°3	Cyclomatic complexity	1	Avoid unused imports	1
n°4	Do not cast a variable to an interface implemented by that variable or to the base class that extends it	1	Avoid excesive import lines	1
n°5	Avoid excesive import lines	1	Avoid unused parameters	1
n°6	Avoid sensitive information exposure through error messages	1	Avoid sensitive information exposure through error messages	1
n°7	Avoid using try statements in loops	1	Close input and output resources in finally blocks	1
n°8	Close input and output resources in finally blocks	1	Avoid nested IF sentences with too many levels	1
n°9	Avoid nested IF sentences with too many levels	1	Avoid throwing RuntimeExceptions	1
n°10	Avoid throwing RuntimeExceptions	1	Do not declare private fields that are used in only one method	7

Label Scansione	05/06/2018 10:39		05/06/2018 10:52	
Data scansione	05/06/2018		05/06/2018	
LOC	394		392	
Risk index	**6,32**		**3,37**	
Global indicator	**63,84**		**68,86**	
Efficiency	99,99		99,99	
Maintainability	49,02		41,82	
Portability	100		100	
Reliability	95,51		95,51	
Security	0		28,75	%
Total defects	**75**	**0,190**	**75**	**0,191**
Very high	1		0	
High	11		12	
Normal	18		18	
Low	12		12	
Very low	33		33	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Weak cryptographic hash	1	Avoid duplicated imports	4
n°2	Avoid duplicated imports	4	Cyclomatic complexity	1
n°3	Cyclomatic complexity	1	Avoid unused imports	1
n°4	Avoid excesive import lines	1	Avoid excesive import lines	1
n°5	Avoid unused parameters	1	Avoid unused parameters	1
n°6	Avoid sensitive information exposure through error messages	1	Avoid sensitive information exposure through error messages	1
n°7	Close input and output resources in finally blocks	1	Close input and output resources in finally blocks	1
n°8	Avoid nested IF sentences with too many levels	1	Avoid nested IF sentences with too many levels	1
n°9	Avoid throwing RuntimeExceptions	1	Avoid throwing RuntimeExceptions	1
n°10	Avoid using components calling too many other components	2	Avoid using components calling too many other components	2

Label Scansione	29/06/2018 09:59		29/06/2018 10:47	
Data scansione	29/06/2018		29/06/2018	
LOC	46		54	
Risk index	**6,22**		**13,6**	
Global indicator	**76,58**		**86,48**	
Efficiency	100		100	1,8%
Maintainability	0,34		41,87	0,85%
Portability	100		100	75,9%
Reliability	98,95		100	0,2%
Security	100		100	#ΔIç/0!
Total defects	**8**		**5**	**0,093**
Very high	0		0	
High	2		1	
Normal	4		3	
Low	2		1	
Very low	0		0	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid unused parameters	1	Avoid nested IF sentences with too many levels	1
n°2	Provide Javadoc comments for public classes and interfaces	1	Avoid using components calling too many other components	1
n°3	Avoid using components calling too many other components	1	Classes internally strongly coupled must be avoided	1
n°4	Classes internally strongly coupled must be avoided	1	Follow the limit for number of statements in a method	1
n°5	Follow the limit for number of statements in a method	1	Avoid class or interface names too long	1
n°6	Always check object type before cast	1		
n°7	Avoid class or interface names too long	1		
n°8	Provide Javadoc comments for public methods in Java classes (non-interface)	1		
n°9				
n°10				

Label Scansione	06/07/2018 11:33		06/07/2018 17:47	
Data scansione	06/07/2018		06/07/2018	
LOC	2.256		587	
Risk index	**73,65**		**10,48**	
Global indicator	**84,17**		**93,26**	
Efficiency	94,73		99,98	
Maintainability	37,59		71,87	
Portability	100		100	
Reliability	98,59		99,2	
Security	99,96		99,98	
Total defects	**398**	**0,176**	**107**	**0,182**
Very high	0		0	
High	36		55	
Normal	327		44	
Low	32		6	
Very low	3		2	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	9	Avoid unused imports	3
n°2	Avoid nested IF sentences with too many levels	9	Avoid excesive import lines	2
n°3	Avoid using try statements in loops	4	Avoid nested IF sentences with too many levels	2
n°4	Date format class java.text.SimpleDateFormat spends many resources	3	Avoid unused fields	2
n°5	Avoid excesive import lines	2	Cyclomatic complexity	1
n°6	Do not instantiate temporal Objects in loops bodies	2	Provide a break or return statement for the default label in a switch	1
n°7	Avoid catch blocks with empty bodies	2	Provide Javadoc comments for public fields	1
n°8	Avoid unused imports	1	Avoid using numeric literals	20
n°9	Provide a break or return statement for the default label in a switch	1	Avoid using components calling too many other components	6
n°10	Provide Javadoc comments for public fields	1	Follow the limit for number of statements in a method	6

Label Scansione	13/07/2018 09:30		13/07/2018 09:34	
Data scansione	**13/07/2018 09:30**		13/07/2018	
LOC	152		152	
Risk index	**0**		**0**	
Global indicator	**99,59**		**99,59**	
Efficiency	99,94		99,94	
Maintainability	98,53		98,53	
Portability	100		100	
Reliability	100		100	
Security	99,76		99,76	
Total defects	10	**0,066**	**10**	**0,066**
Very high	0		0	
High	0		0	
Normal	8		8	
Low	0		0	
Very low	2		2	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Follow the limit for number of statements in a method	3	Follow the limit for number of statements in a method	3
n°2	Loop with Unreachable Exit Condition ('Infinite Loop')	2	Loop with Unreachable Exit Condition ('Infinite Loop')	2
n°3	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°4	Avoid using numeric literals	1	Avoid using numeric literals	1
n°5	Only use Hibernate/JPA for database access	1	Only use Hibernate/JPA for database access	1
n°6	Avoid using do-while statements	2	Avoid using do-while statements	2
n°7				
n°8				
n°9				
n°10				

Label Scansione	19/07/2018 10:28		19/07/2018 12:45	
Data scansione	19/07/2018		19/07/2018	
LOC	445		458	
Risk index	**0,17**		**0,16**	
Global indicator	**96,13**		**97,27**	
Efficiency	95,48		98,83	5,3%
Maintainability	88,87		89,29	47,70%
Portability	100		100	0,0%
Reliability	98,21		99,98	0,6%
Security	99,91		99,91	0,0%
Total defects	**41**	**0,092**	**26**	**0,057**
Very high	0		0	
High	4		3	
Normal	31		19	
Low	6		4	
Very low	0		0	

	#23		#26	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid excesive import lines	1	Avoid excesive import lines	1
n°2	Cyclomatic complexity	1	Cyclomatic complexity	1
n°3	Avoid nested IF sentences with too many levels	1	Avoid nested IF sentences with too many levels	1
n°4	Do not instantiate temporal Objects in loops bodies	1	Do not instantiate temporal Objects in loops bodies	1
n°5	Avoid using repeated cast over the same object or variable (Max 3 cast of every type)	1	Declare the List and Set variables with their interface type	10
n°6	Declare the List and Set variables with their interface type	10	Loop with Unreachable Exit Condition ('Infinite Loop')	3
n°7	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	9	Avoid using components calling too many other components	1
n°8	Always check object type before cast	4	Always check object type before cast	1
n°9	Loop with Unreachable Exit Condition ('Infinite Loop')	3	Follow the limit for number of statements in a method	1
n°10	Avoid using components calling too many other components	1	Only use Hibernate/JPA for database access	1

Label Scansione	01/08/2018 14:24		01/08/2018 16:11	
Data scansione	01/08/2018		01/08/2018	
LOC	210		214	
Risk index	**85,44**		**47,65**	
Global indicator	**72,28**		**93,34**	
Efficiency	30,44		94,71	
Maintainability	37,55		75,8	
Portability	100		100	
Reliability	98,95		99,96	
Security	99,96		99,85	
Total defects	**27**	**0,129**	**16**	**0,075**
Very high	0		0	
High	9		3	
Normal	17		12	
Low	1		1	
Very low	0		0	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Do not instantiate temporal Objects in loops bodies	2	Avoid unused imports	1
n°2	Avoid using try statements in loops	2	Cyclomatic complexity	1
n°3	Provide Javadoc comments for public classes and interfaces	1	Avoid nested IF sentences with too many levels	1
n°4	Avoid unused parameters	1	Do not instantiate temporal Objects in loops bodies	1
n°5	Cyclomatic complexity	1	Avoid using components calling too many other components	2
n°6	Avoid nested IF sentences with too many levels	1	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	1
n°7	All input/output resources must have a flow with buffer	1	Do not define 'Object' variables	1
n°8	Avoid modifications on method or constructor parameters	3	Follow the limit for number of statements in a method	1
n°9	Avoid using components calling too many other components	2	Only use Hibernate/JPA for database access	1
n°10	Only use Hibernate/JPA for database access	2	Follow the limit for number of lines in a method	1

Label Scansione	02/08/2018 09:37		02/08/2018 10:13	
Data scansione	02/08/2018 09:37		02/08/2018	
LOC	217		226	
Risk index	**46,6**		**0**	
Global indicator	**93,51**		**97,66**	
Efficiency	94,85		99,97	
Maintainability	76,37		90,01	
Portability	100		100	
Reliability	99,85		100	
Security	99,96		99,97	
Total defects	16	**0,074**	**12**	**0,053**
Very high	0		0	
High	3		1	
Normal	12		10	
Low	1		1	
Very low	0		0	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	1	Cyclomatic complexity	1
n°2	Avoid nested IF sentences with too many levels	1	Avoid using components calling too many other components	2
n°3	Do not instantiate temporal Objects in loops bodies	1	Do not define 'Object' variables	2
n°4	Avoid using components calling too many other components	2	Follow the limit for number of statements in a method	1
n°5	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal')	1	Only use Hibernate/JPA for database access	1
n°6	Do not define 'Object' variables	1	Follow the limit for number of lines in a method	1
n°7	Follow the limit for number of statements in a method	1	Too many uses of the negation operator (!) in a method	1
n°8	Only use Hibernate/JPA for database access	1	Loop with Unreachable Exit Condition ('Infinite Loop')	1
n°9	Follow the limit for number of lines in a method	1	Declare the List and Set variables with their interface type	1
n°10	Too many uses of the negation operator (!) in a method	1	Follow the limit for number of return statements	1

Fig.3.30. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.

Sviluppatore 15

Label Scansione	AdeguamentoPCI		AdeguamentoPCI	
Data scansione	24/09/2018		25/09/2018	
LOC	1.967		1.967	
Risk index	**70,38**		**70,38**	
Global indicator	**71,87**		**71,87**	
Efficiency	89,18		89,18	
Maintainability	0,63		0,63	
Portability	74,16		74,16	
Reliability	100		100	
Security	100		100	
Total defects	**320**	**0,163**	**320**	**0,163**
Very high	6		6	
High	43		43	
Normal	165		165	
Low	104		104	
Very low	2		2	

Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	6	Cyclomatic complexity	6
n°2	Do not use Runtime and System classes	21	Do not use Runtime and System classes	21
n°3	Duplicated code: medium block	8	Duplicated code: medium block	8
n°4	Avoid using try statements in loops	8	Avoid using try statements in loops	8
n°5	Avoid nested IF sentences with too many levels	3	Avoid nested IF sentences with too many levels	3
n°6	Provide Javadoc comments for public classes and interfaces	1	Provide Javadoc comments for public classes and interfaces	1
n°7	Avoid excesive import lines	1	Avoid excesive import lines	1
n°8	Avoid classes / interfaces with too many lines of code	1	Avoid classes / interfaces with too many lines of code	1
n°9	Avoid using numeric literals	76	Avoid using numeric literals	76
n°10	Avoid using java.util.Date or java.util.GregorianCalendar	21	Avoid using java.util.Date or java.util.GregorianCalendar	21

Label Scansione	servlet sws		servlet sws	
Data scansione	28/11/2018		28/11/2018	
LOC	301		308	
Risk index	**100**		**100**	
Global indicator	**38,35**		**35,03**	
Efficiency	68,76		71,83	
Maintainability	3,14		2,9	
Portability	73,77		59,93	
Reliability	69,88		60,3	
Security	0		0	
Total defects	**91**	**0,302**	**99**	**0,321**
Very high	1		6	
High	44		59	
Normal	28		22	
Low	13		8	
Very low	5		4	

Top 10 Defect	Rule	Defects	Rule	Defects
n°1	Cyclomatic complexity	1	Avoid non-neutralized user-controlled input composed in a pathname to a resource	3
n°2	Improper Output Neutralization for Logs	30	Improper Neutralization of CRLF Sequences in HTTP Headers ('HTTP Response Splitting')	2
n°3	Do not use Runtime and System classes	4	Cyclomatic complexity	1
n°4	Initialize all static fields	4	Improper Output Neutralization for Logs	43
n°5	Avoid using method calls in a loop	1	Do not use Runtime and System classes	5
n°6	Avoid using too many 'non-final static' fields	1	Initialize all static fields	4
n°7	Avoid using synchronized modifier in the method declaration	1	Avoid using method calls in a loop	1
n°8	Avoid nested IF sentences with too many levels	1	Avoid using too many 'non-final static' fields	1
n°9	Avoid getQueryString(), using getParameter()	1	Avoid using synchronized modifier in the method declaration	1
n°10	All input/output resources must have a flow with buffer	1	Avoid nested IF sentences with too many levels	1

Label Scansione	SSO		SSO	
Data scansione	12/12/2018		13/12/2018	
LOC	289		300	
Risk index	**100**		**97,33**	
Global indicator	**53,11**		**53,15**	
Efficiency	99,77		99,9	0,0%
Maintainability	0		0	0,00%
Portability	97,47		97,57	0,0%
Reliability	99,83		99,83	0,0%
Security	0		0	0,0%
Total defects	**96**		**19**	**0,063**
Very high	3		3	
High	4		3	
Normal	44		3	
Low	15		9	
Very low	30		1	

	#15		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	2	Cyclomatic complexity	2
n°2	Use of Hard-coded Credentials	1	Use of Hard-coded Credentials	1
n°3	Do not use Runtime and System classes	1	Do not use Runtime and System classes	1
n°4	Avoid using too many 'non-final static' fields	1	Provide Javadoc comments for public classes and interfaces	1
n°5	Provide Javadoc comments for public classes and interfaces	1	Avoid nested IF sentences with too many levels	1
n°6	Avoid nested IF sentences with too many levels	1	Follow the limit for number of statements in a method	4
n°7	Declare constant fields private and final	16	Avoid unused fields	2
n°8	Do not declare private fields that are used in only one method	13	Avoid using components calling too many other components	1
n°9	Follow the limit for number of statements in a method	4	Classes internally strongly coupled must be avoided	1
n°10	Avoid unused fields	2	Only use Hibernate/JPA for database access	1

Label Scansione	17/01/2019 11:33		17/01/2019 14:04	
Data scansione	17/01/2019		17/01/2019	
LOC	217		226	
Risk index	**20,12**		**15,73**	
Global indicator	**86,8**		**87,85**	
Efficiency	99,58		99,61	
Maintainability	48,64		52,8	
Portability	100		100	
Reliability	94,94		95,27	
Security	100		100	
Total defects	**51**	**0,235**	**48**	**0,212**
Very high	0		0	
High	1		1	
Normal	42		41	
Low	6		6	
Very low	2		2	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Close input and output resources in finally blocks	1	Close input and output resources in finally blocks	1
n°2	Avoid using numeric literals	35	Avoid using numeric literals	35
n°3	Avoid using java.util.Date or java.util.GregorianCalendar	2	Avoid using java.util.Date or java.util.GregorianCalendar	2
n°4	Follow the limit for number of statements in a method	2	Follow the limit for number of statements in a method	1
n°5	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°6	Classes internally strongly coupled must be avoided	1	Classes internally strongly coupled must be avoided	1
n°7	Only use Hibernate/JPA for database access	1	Only use Hibernate/JPA for database access	1
n°8	Avoid capturing java.lang.Exception exceptions	2	Avoid capturing java.lang.Exception exceptions	2
n°9	Avoid Exception, RuntimeException o Throwable in catch or throw statements	2	Avoid Exception, RuntimeException o Throwable in catch or throw statements	2
n°10	Use constructors than initialize all the fields in a class	1	Use constructors than initialize all the fields in a class	1

Label Scansione	InquiryStock		InquiryStock	
Data scansione	**20/02/2019 10:38**		20/02/2019	
LOC	322		323	
Risk index	0,04	%	**0**	
Global indicator	**96,05**		**99,82**	
Efficiency	99,65		99,76	
Maintainability	86,06		99,42	
Portability	100		100	
Reliability	97,25		100100	
Security	100			
Total defects	30	**0,093**	**13**	**0,040**
Very high	0		0	
High	2		0	
Normal	21		10	
Low	5		3	
Very low	2		0	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Provide Javadoc comments for public classes and interfaces	2	Avoid using components calling too many other components	2
n°2	Initialize all local variables at the declaration statement	11	Classes internally strongly coupled must be avoided	2
n°3	Avoid using components calling too many other components	2	Follow the limit for number of statements in a method	2
n°4	Classes internally strongly coupled must be avoided	2	Only use Hibernate/JPA for database access	2
n°5	Follow the limit for number of statements in a method	2	Close JDBC resources when finishing using	1
n°6	Only use Hibernate/JPA for database access	2	Use PreparedStatement instead of Statement to similar requests	1
n°7	Close JDBC resources when finishing using	1	Use constructors than initialize all the fields in a class	1
n°8	Use PreparedStatement instead of Statement to similar requests	1	Provide Javadoc comments for public methods in Java classes (non-interface)	1
n°9	Provide Javadoc comments for public methods in Java classes (non-interface)	2	Write one statement per line	1
n°10	Avoid returning null in a method declaration	2		

Label Scansione	22/02/2019 12:56		22/02/2019 16:37	
Data scansione	22/02/2019		22/02/2019	
LOC	427		427	
Risk index	**53,2**		**53,01**	
Global indicator	**76,79**		**76,99**	
Efficiency	99,76		99,85	0,0%
Maintainability	0,9		1,2	7,88%
Portability	99,03		100	0,0%
Reliability	99,99		100	0,3%
Security	100		100	0,0%
Total defects	**33**	**0,077**	**27**	**0,063**
Very high	2		2	
High	1		0	
Normal	26		22	
Low	3		2	
Very low	1		1	

	#23		#26	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	2	Cyclomatic complexity	2
n°2	Do not use Runtime and System classes	1	Avoid using numeric literals	11
n°3	Avoid using numeric literals	12	Follow the limit for number of statements in a method	3
n°4	Follow the limit for number of statements in a method	4	Avoid using components calling too many other components	2
n°5	Avoid using components calling too many other components	2	Classes internally strongly coupled must be avoided	2
n°6	Classes internally strongly coupled must be avoided	2	Only use Hibernate/JPA for database access	2
n°7	Only use Hibernate/JPA for database access	2	Close JDBC resources when finishing using	1
n°8	Use PreparedStatement instead of Statement to similar requests	2	Use PreparedStatement instead of Statement to similar requests	1
n°9	Initialize all local variables at the declaration statement	1	Use constructors than initialize all the fields in a class	1
n°10	Close JDBC resources when finishing using	1	Provide Javadoc comments for public methods in Java classes (non-interface)	1

Label Scansione	18/03/2019 10:26		18/03/2019 15:27	
Data scansione	18/03/2019		18/03/2019	
LOC	499		490	
Risk index	**15,66**		**0**	
Global indicator	**79,78**		**98,37**	
Efficiency	99,79		99,81	
Maintainability	13,29		93,19	
Portability	100		100	
Reliability	99,92		99,96	
Security	100		100	
Total defects	**37**	**0,074**	**30**	**0,061**
Very high	1		0	
High	5		2	
Normal	21		18	
Low	10		9	
Very low	0		1	

	#21		#22	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Cyclomatic complexity	1	Provide Javadoc comments for public classes and interfaces	1
n°2	Provide Javadoc comments for public classes and interfaces	3	Avoid unused imports	1
n°3	Avoid unused imports	2	Only use Hibernate/JPA for database access	5
n°4	Follow the limit for number of statements in a method	5	Avoid using components calling too many other components	4
n°5	Only use Hibernate/JPA for database access	5	Classes internally strongly coupled must be avoided	4
n°6	Avoid using components calling too many other components	4	Follow the limit for number of statements in a method	3
n°7	Classes internally strongly coupled must be avoided	4	Always check object type before cast	2
n°8	Always check object type before cast	2	Provide Javadoc comments for public methods in Java classes (non-interface)	3
n°9	Initialize all local variables at the declaration statement	1	Use constructors than initialize all the fields in a class	2
n°10	Provide Javadoc comments for public methods in Java classes (non-interface)	3	Avoid creating or assigning a variable within a loop	2

Label Scansione	15/04/2019 12:19		15/04/2019 12:55	
Data scansione	15/04/2019 12:19		15/04/2019	
LOC	89		91	
Risk index	0	%	**0**	
Global indicator	**99,7**		**99,68**	
Efficiency	99,86		99,85	
Maintainability	98,81		98,74	
Portability	100		100	
Reliability	100		100	
Security	100		100	
Total defects	8	**0,090**	**7**	**0,077**
Very high	0		0	
High	0		0	
Normal	4		4	
Low	3		3	
Very low	1		0	

	#25		#18	
Top 10 Defect	**Rule**	**Defects**	**Rule**	**Defects**
n°1	Avoid using components calling too many other components	1	Avoid using components calling too many other components	1
n°2	Classes internally strongly coupled must be avoided	1	Classes internally strongly coupled must be avoided	1
n°3	Follow the limit for number of statements in a method	1	Follow the limit for number of statements in a method	1
n°4	Only use Hibernate/JPA for database access	1	Only use Hibernate/JPA for database access	1
n°5	Provide Javadoc comments for public methods in Java classes (non-interface)	1	Provide Javadoc comments for public methods in Java classes (non-interface)	1
n°6	Avoid creating or assigning a variable within a loop	1	Avoid creating or assigning a variable within a loop	1
n°7	Provide Javadoc comments for protected fields	1	Provide Javadoc comments for protected fields	1
n°8	Provide Javadoc comments for private methods	1		
n°9				
n°10				

Fig.3.31. Modifiche della qualità su prodotti diversi assegnati allo sviluppatore e della consistenza dei 10 difetti più numerosi.