UNIVERSIDAD ANDRÉS BELLO
FACULTAD DE INGENIERÍA
ESCUELA DE INFORMÁTICA
INGENIERÍA EN COMPUTACIÓN E INFORMÁTICA

UNIVERSIDAD
ANDRES BELLO

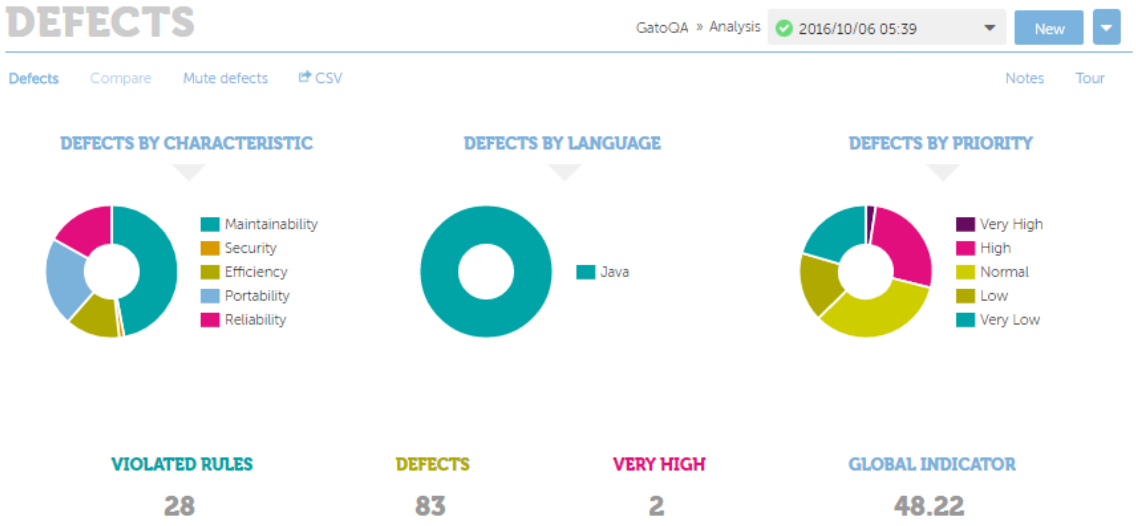
Taller n°2 – Análisis de juego “GATO”

EMILIO VALDIVIA IGLESIAS

Herr.	Error/Defecto	Descripción	Responsable	Importancia	Esfuerzo
Kiuwan	Provide Javadoc comments for public methods.	Defecto alude a todos los métodos Java públicos sin comentar	Emilio Valdivia	Alta	6 min
Kiuwan	Avoid declaring multiple variables in one statement.	Evitar declarar muchas variables en una sola declaración	Emilio Valdivia	Muy Baja	24 min
Kiuwan	Provide a by default private constructor in utility classes.	Proveer un constructor por defecto en clases de utilidades	Emilio Valdivia	Normal	4 h
Kiuwan	Avoid the incorrect naming of non-static methods.	Evitar el nombramiento incorrecto de métodos que no son estáticos	Emilio Valdivia	Baja	3 min
Kiuwan	Avoid creating asignment chains.	Evitar crear asignación de variables en cadena	Emilio Valdivia	Muy Baja	4h
Sonar	Move the "OOO" string literal on the left side of this string comparison	Mover el string “OOO” al lado izquierdo de la comparación	Emilio Valdivia	Alta	2 min
Sonar	Move the "XXX" string literal on the left side of this string comparison.	Mover el string “XXX” al lado izquierdo de la comparación	Emilio Valdivia	Alta	2 min
Sonar	Remove those useless parenthesis	Remover paréntesis sin usar	Emilio Valdivia	Alta	2 min
Sonar	Declare "acumY3" on a separate line	Declarar variable “acumY3” en una linea separada	Emilio Valdivia	Alta	2 min

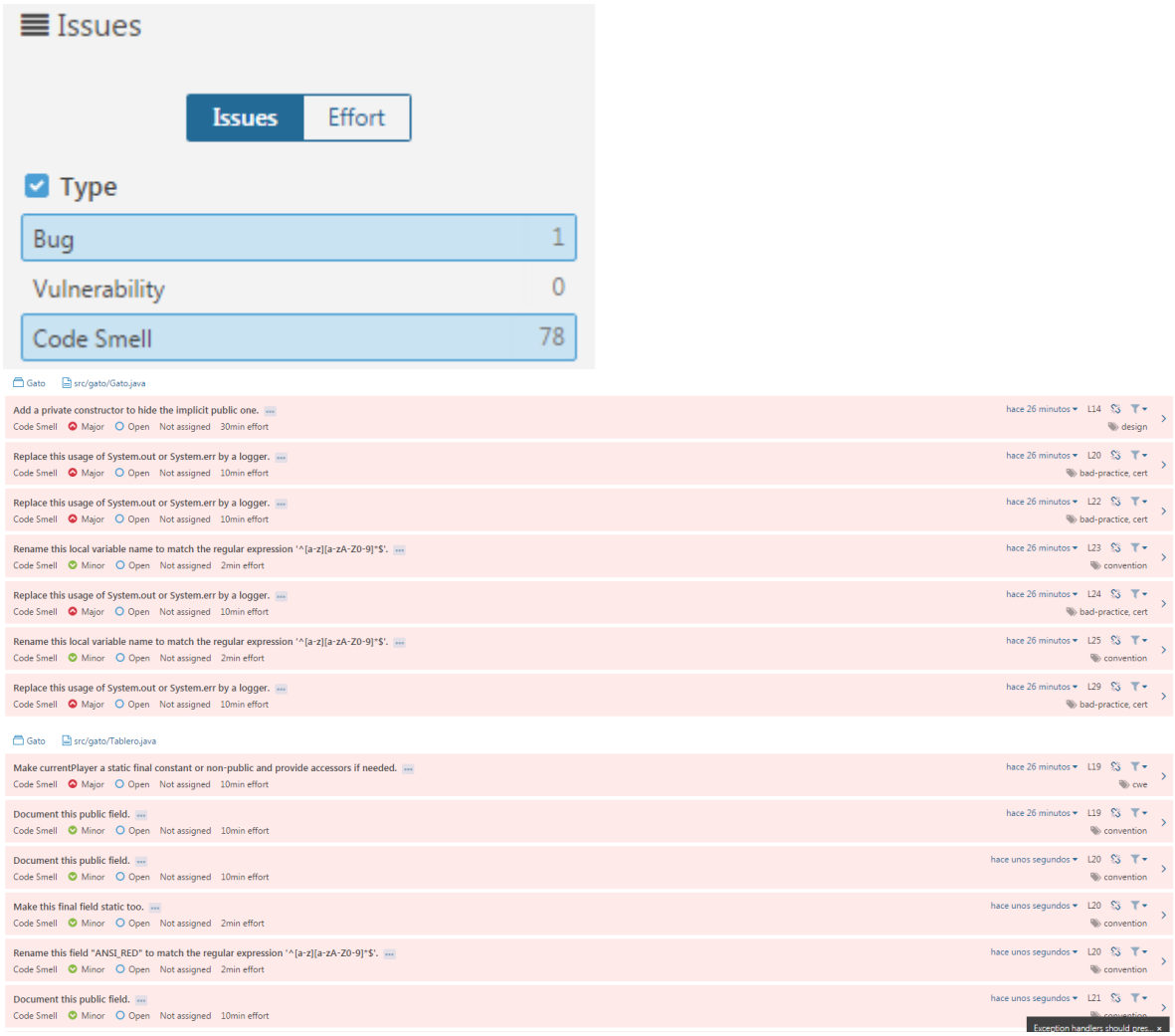
Sonar	Document this public method	Documentar este método público	Emilio Valdivia	Menor	2 min

Capturas de pantalla
Kiuwan:



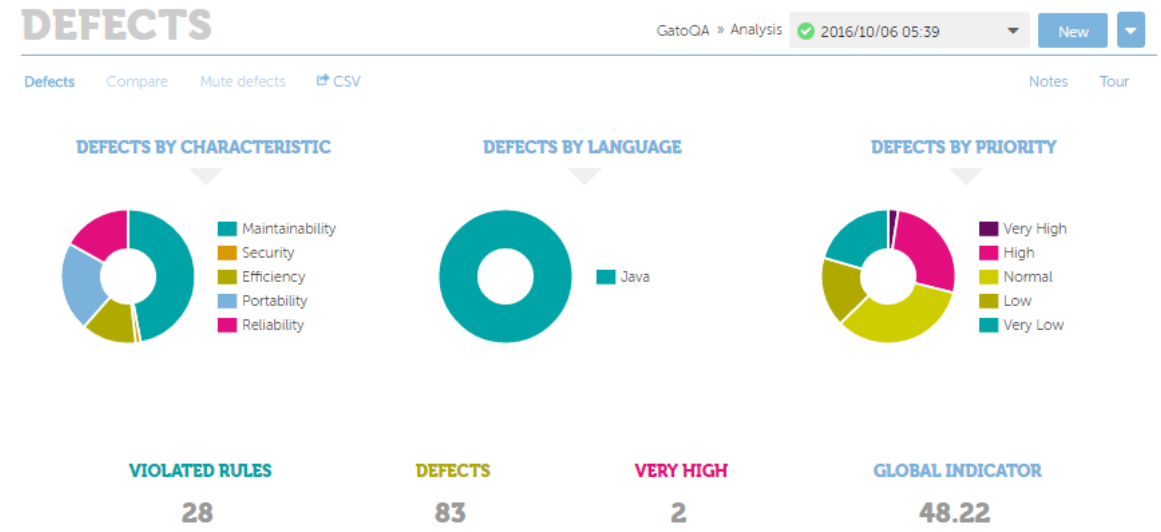
Σ		83						67h 20
	1	2	Cyclomatic complexity.		Maintainability	Java	8h 00
	2	18	Do not use Runtime and System classes.		Portability	Java	9h 00
	1	1	Avoid using method calls in a loop.		Efficiency	Java	30m
	1	1	Provide Javadoc comments for public fields.		Maintainability	Java	06m
	1	1	Avoid using try statements in loops.		Efficiency	Java	30m
	1	1	Do not instantiate temporal Objects in loops bodies.		Efficiency	Java	06m
	1	10	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal').		Reliability	Java	30m
	2	6	Follow the limit for number of statements in a method.		Maintainability	Java	24h 00
	1	4	Initialize all local variables at the declaration statement.		Reliability	Java	24m
	1	4	Avoid declaring multiple variables in one statement.		Maintainability	Java	24m
	1	2	Avoid duplicate literals.		Maintainability	Java	1h 00
	1	1	Avoid using components calling too many other components.		Maintainability	Java	30m
	1	1	Avoid invoking a static method of java.lang.Math invoke on a constant.		Efficiency	Java	06m
	1	6	Avoid creating or assigning a variable within a loop.		Efficiency	Java	36m
	1	2	Follow the limit for number of return statements.		Maintainability	Java	8h 00
	1	1	Avoid unused parameters.		Efficiency	Java	4h 00
	1	1	Provide Javadoc comments for public methods.		Maintainability	Java	06m
	1	1	Define every field private or protected		Maintainability	Java	03m
	1	1	Provide a by default private constructor in utility classes.		Maintainability	Java	4h 00
	1	1	Provide a branch block for 'if' statements.		Maintainability	Java	30m
	1	1	Avoid the incorrect naming of non-static methods.		Maintainability	Java	03m
	1	8	Avoid creating asignment chains.		Maintainability	Java	4h 00
	1	3	Avoid incorrect name format in the final fields.		Maintainability	Java	09m
	1	2	Avoid incorrect name format in local variables.		Maintainability	Java	06m
	1	1	Avoid class names that are less than 5 characters.		Maintainability	Java	03m
	1	1	Avoid dangerous J2EE API, use replacements from security-focused libraries (like OWASP ESAPI)		Security	Java	06m
	1	1	Avoid using do-while statements.		Maintainability	Java	30m
	1	1	Always follow the java method naming conventions.		Maintainability	Java	03m

Sonar:


Issues
Issues
Effort
Type
Bug 1
Vulnerability 0
Code Smell 78
Gato
src/gato/Gato.java
Add a private constructor to hide the implicit public one.
Code Smell Major Open Not assigned 30min effort
hace 26 minutos L14
design
Replace this usage of System.out or System.err by a logger.
Code Smell Major Open Not assigned 10min effort
hace 26 minutos L20
bad-practice, cert
Replace this usage of System.out or System.err by a logger.
Code Smell Major Open Not assigned 10min effort
hace 26 minutos L22
bad-practice, cert
Rename this local variable name to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
Code Smell Minor Open Not assigned 2min effort
hace 26 minutos L23
convention
Replace this usage of System.out or System.err by a logger.
Code Smell Major Open Not assigned 10min effort
hace 26 minutos L24
bad-practice, cert
Rename this local variable name to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
Code Smell Minor Open Not assigned 2min effort
hace 26 minutos L25
convention
Replace this usage of System.out or System.err by a logger.
Code Smell Major Open Not assigned 10min effort
hace 26 minutos L29
bad-practice, cert
Gato
src/gato/Tablero.java
Make currentPlayer a static final constant or non-public and provide accessors if needed.
Code Smell Major Open Not assigned 10min effort
hace 26 minutos L19
cwe
Document this public field.
Code Smell Minor Open Not assigned 10min effort
hace 26 minutos L19
convention
Document this public field.
Code Smell Minor Open Not assigned 10min effort
hace unos segundos L20
convention
Make this final field static too.
Code Smell Minor Open Not assigned 2min effort
hace unos segundos L20
convention
Rename this field "ANSI_RED" to match the regular expression '^[a-z][a-zA-Z0-9]*$'.
Code Smell Minor Open Not assigned 2min effort
hace unos segundos L20
convention
Document this public field.
Code Smell Minor Open Not assigned 10min effort
hace unos segundos L21
Exception handlers should pres...

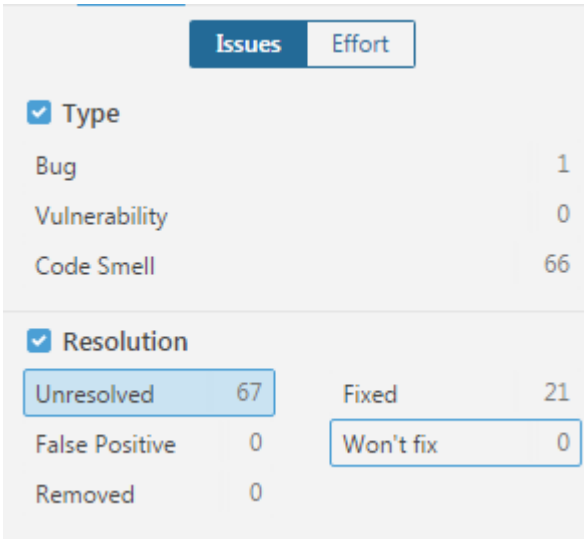
Capturas de pantalla con correcciones:

Kiuwan:



	Files	Defects	Rule	Priority	Characteristic	Language	Effort
Σ		83					67h 20
	1	2	Cyclomatic complexity.		Maintainability	Java	8h 00
	2	18	Do not use Runtime and System classes.		Portability	Java	9h 00
	1	1	Avoid using method calls in a loop.		Efficiency	Java	30m
	1	1	Provide Javadoc comments for public fields.		Maintainability	Java	06m
	1	1	Avoid using try statements in loops.		Efficiency	Java	30m
	1	1	Do not instantiate temporal Objects in loops bodies.		Efficiency	Java	06m
	1	10	Avoid calling varString.equals('literal') or varString.equalsIgnoreCase('literal').		Reliability	Java	30m
	2	6	Follow the limit for number of statements in a method.		Maintainability	Java	24h 00
	1	4	Initialize all local variables at the declaration statement.		Reliability	Java	24m
	1	4	Avoid declaring multiple variables in one statement.		Maintainability	Java	24m
	1	2	Avoid duplicate literals.		Maintainability	Java	1h 00
	1	1	Avoid using components calling too many other components.		Maintainability	Java	30m
	1	1	Avoid invoking a static method of java.lang.Math invoke on a constant.		Efficiency	Java	06m
	1	6	Avoid creating or assigning a variable within a loop.		Efficiency	Java	36m
	1	2	Follow the limit for number of return statements.		Maintainability	Java	8h 00
	1	1	Avoid unused parameters.		Efficiency	Java	4h 00
	1	1	Provide Javadoc comments for public methods.		Maintainability	Java	06m
	1	1	Define every field private or protected		Maintainability	Java	03m
	1	1	Provide a by default private constructor in utility classes.		Maintainability	Java	4h 00
	1	1	Provide a branch block for 'if' statements.		Maintainability	Java	30m
	1	1	Avoid the incorrect naming of non-static methods.		Maintainability	Java	03m
	1	8	Avoid creating asignment chains.		Maintainability	Java	4h 00
	1	3	Avoid incorrect name format in the final fields.		Maintainability	Java	09m
	1	2	Avoid incorrect name format in local variables.		Maintainability	Java	06m
	1	1	Avoid class names that are less than 5 characters.		Maintainability	Java	03m
	1	1	Avoid dangerous J2EE API, use replacements from security-focused libraries (like OWASP ESAPI)		Security	Java	06m
	1	1	Avoid using do-while statements.		Maintainability	Java	30m
	1	1	Always follow the java method naming conventions.		Maintainability	Java	03m

Sonar:



Preguntas:

- ¿En qué lenguaje se programó la aplicación/código?

RE: Java

- ¿Cuántos errores/defectos arrojó cada herramienta?

RE: 78 Sonar y 83 Kiuwan.

- Hacer una tabla comparativa con los errores/defectos que se repiten en ambas

Error/ Defecto	SonarQube	Kiuwan
Documentar métodos públicos	X	X
Eliminar loop dentro de try catch		X
Eliminar paréntesis	X	X
Complejidad ciclomática		X
Evitar declaraciones en cadena	X	X
Evitar mal uso de métodos estáticos	X	

- ¿Poseen errores distintos?, ¿Por qué?
RE: No en gran mayoría, son solo 5 errores de diferencia y puede deberse a los alcances que contemplen analizar cada una de las aplicaciones.

- Luego de reparados los 5 (mínimo) errores/defectos, ¿Qué errores se mantienen?
RE: En SONAR disminuyeron los errores a 67, mientras que Kiuwo se mantuvo la misma cantidad.

-¿Cuáles son las principales diferencias entre SonarQube y Kiuwan?
Sonar es de código abierto, mientras que Kiuwan no.
Kiuwan al ser de pago tiene mayor detalle en los reportes.
Kiuwan muestra atributos de calidad en KPI's.
Kiuwan entre mucho más detalle en problemas y conceptos más complejos.
- Conclusiones

Las herramientas presentadas van a encontrar errores en función de las buenas/malas prácticas del programador.

Kiuwan presenta mayor solidez y robustez en cuanto a los reportes generados.

Ambas aplicaciones pueden entregar buenos resultados, pero Kiuwan es lejos más fácil de utilizar, más completa y da un mejor detalle respecto a los errores encontrados.