

ANÁLISIS DE LA CALIDAD DE SOFTWARE

Daniel Parra – 20181020110

Diseño arquitectural de software y patrones

2023

Se realizó la comprobación de una aplicación de escritorio CRUD, que permite añadir instructores y monitores a una serie determinada de cursos, para ello se utilizó el programa de análisis SourceMonitor.

Los campos que se muestran en el programa son los siguientes:

- File Name: El nombre del archivo de código fuente analizado.
- Lines: La cantidad total de líneas en el archivo.
- Statements: La cantidad total de declaraciones de código en el archivo.
- % Branches: El porcentaje de ramificaciones condicionales en el código. Las ramificaciones condicionales incluyen estructuras como if, else, switch, etc.
- Calls: La cantidad total de llamadas a funciones o métodos en el código.
- % Comments: El porcentaje de líneas de comentarios en el código.
- Classes: La cantidad de clases definidas en el archivo.
- Methods/Class: El promedio de métodos por clase en el archivo.
- Avg Stmts/Method: El promedio de declaraciones de código por método en el archivo.
- Max Complexity: La complejidad máxima de McCabe en el archivo. La complejidad de McCabe mide la complejidad del flujo de control en un programa.
- Max Depth: La profundidad máxima de anidamiento en el archivo.
- Avg Depth: El promedio de profundidad de anidamiento en el archivo.
- Avg Complexity: El promedio de complejidad de McCabe en el archivo.

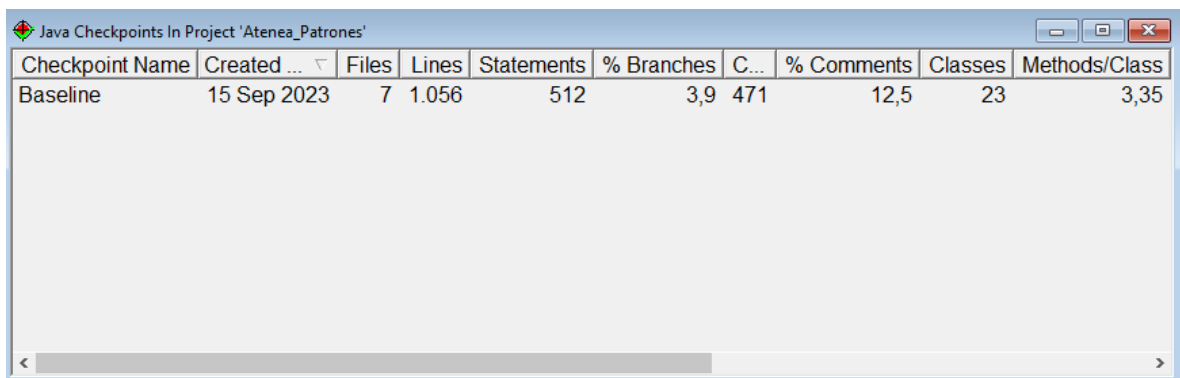
El programa muestra dos tablas, una general del proyecto:

Checkpoint Name	Created On	Files	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmts/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
Baseline	15 Sep 2023	6	526	306	6,5	147	11,4	6	7,00	4,50	7	4	1,78	1,50

Y otra específica para cada archivo:

File Name	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmts/Method	Max Complexity	Max Depth	
	Avg Depth	Avg Complexity									
controlador\Controlador.java	181	123	12,2	91	6,1	1	6,00	15,67	7		
	4	2,06	3,67								
DAO\CursoDao.java	69	27	3,7	16	43,5	1	1,00	13,00	2	3	
	1,48	2,00									
DAO\PersonaDao.java		66	46	6,5	37	7,6	1	2,00	15,50	3	
	4	2,15	2,50								
modelo\Conexion.java		27	15	6,7	3	18,5	1	1,00	9,00	2	
	3	1,53	2,00								
modelo\Curso.java	90	48	0,0	0	0,0	1	16,00	1,31	1	2	
	1,35	1,00									
modelo\Persona.java	93	47	0,0	0	9,7	1	16,00	1,31	1	2	
	1,38	1,00									

Muestra de las tablas proporcionada por SourceMonitor:



Checkpoint Name	Created ...	Files	Lines	Statements	% Branches	C...	% Comments	Classes	Methods/Class
Baseline	15 Sep 2023	7	1.056	512	3,9	471	12,5	23	3,35

Java Checkpoints In Project 'Atenea_Patrones'								
	% Comments	Classes	Methods/Class	Avg Stmt/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity
1	12,5	23	3,35	4,36	7	4	1,82	1,27

Java Checkpoints In Project 'Atenea_Patrones'									
Checkpoint Name	Created ... ▾	Files	Lines	Statements	% Branches	C...	% Comments	Classes	Methods/Class
Baseline	15 Sep 2023	6	526	306	6,5	147	11,4	6	7,00

Files in Java Project 'Atenea_Patrones', Checkpoint 'Baseline' [Base Directory: 'F:\Documentos\NetBeansProjects\Atenea_Patrones\src\']									
File Name	Lines	Statements	% Branches	Calls	% Comments	Classes	Methods/Class	Avg Stmt/Method	
controlador\Controlador.java	181	123	12,2	91	6,1	1	6,00		
DAO\CursoDao.java	69	27	3,7	16	43,5	1	1,00		
DAO\PersonaDao.java	66	46	6,5	37	7,6	1	2,00		
modelo\Conexion.java	27	15	6,7	3	18,5	1	1,00		
modelo\Curso.java	90	48	0,0	0	0,0	1	16,00		
modelo\Persona.java	93	47	0,0	0	9,7	1	16,00		

Files in Java Project 'Atenea_Patrones', Checkpoint 'Baseline' [Base Directory: 'F:\Documentos\NetBeansProjects\Atenea_Patrones\src\']									
	% Comments	Classes	Methods/Class	Avg Stmts/Method	Max Complexity	Max Depth	Avg Depth	Avg Complexity	
5	6,1	1	6,00	15,67	7	4	2,06	3,67	
6	43,5	1	1,00	13,00	2	3	1,48	2,00	
7	7,6	1	2,00	15,50	3	4	2,15	2,50	
8	18,5	1	1,00	9,00	2	3	1,53	2,00	
0	0,0	1	16,00	1,31	1	2	1,35	1,00	
0	9,7	1	16,00	1,31	1	2	1,38	1,00	

Conclusiones sobre el Análisis de Código con SourceMonitor

El análisis de código con la herramienta SourceMonitor proporciona información valiosa sobre la calidad y la estructura del código fuente de una aplicación. A través de la evaluación de diferentes métricas, como líneas de código, declaraciones, complejidad y comentarios, es posible obtener una visión general de la salud del proyecto y detectar áreas de mejora. En este informe, se resumen los hallazgos clave en base a los datos proporcionados por SourceMonitor.

Calidad del Código:

Uno de los aspectos más destacados es la calidad general del código. La mayoría de los archivos evaluados muestran una calidad aceptable, con una proporción significativa de comentarios que facilitan la comprensión del código. Sin embargo, también se identificaron áreas de mejora. Algunos archivos tienen métodos con complejidad relativamente alta, lo que puede dificultar el mantenimiento y la depuración. Es esencial abordar estas áreas para simplificar el código y reducir su complejidad.

Estructura y Organización:

La estructura y organización del proyecto son elementos críticos en el desarrollo de software. En este análisis, se observa que la mayoría de los archivos están organizados en clases y métodos, lo que indica una buena modularidad. Sin embargo, se detectó un problema de anidamiento excesivo en algunos lugares, lo que puede dificultar la legibilidad y el mantenimiento del código. Se recomienda revisar y refactorizar estas secciones para reducir la profundidad de anidamiento.

Comentarios y Documentación:

Los comentarios desempeñan un papel fundamental en la comprensión del código. Si bien la mayoría de los archivos cuentan con comentarios adecuados, hay áreas donde la documentación es insuficiente o incluso inexistente. Es fundamental mantener una práctica consistente de documentación para ayudar a los desarrolladores futuros a entender rápidamente el propósito y el funcionamiento de cada parte del código.

Áreas de Mejora:

Con base en los resultados del análisis, se identifican las siguientes áreas de mejora:

- **Simplificación del Código:** Abordar los métodos con alta complejidad para simplificarlos y hacer que sean más comprensibles.
- **Reducción del Anidamiento:** Refactorizar las áreas con anidamiento excesivo para mejorar la legibilidad.
- **Mayor Documentación:** Añadir comentarios adicionales en áreas donde la documentación es escasa.
- **Pruebas y Validaciones:** Implementar pruebas y validaciones para garantizar que el código funcione correctamente y evitar errores.