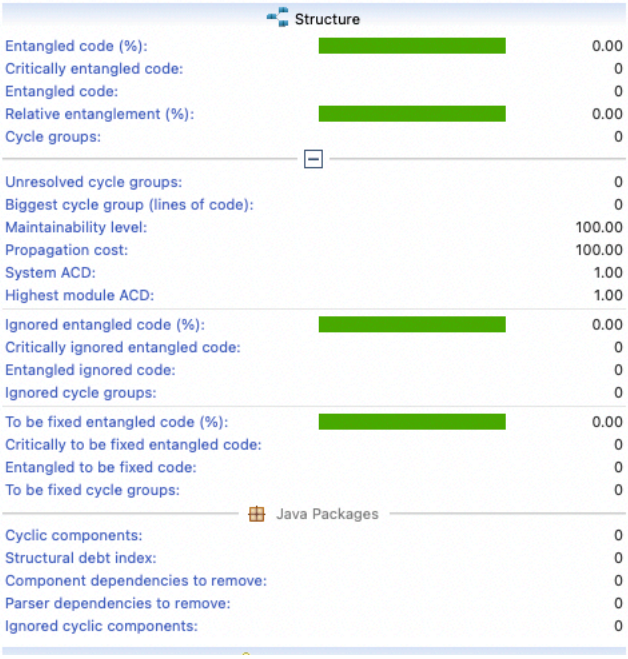
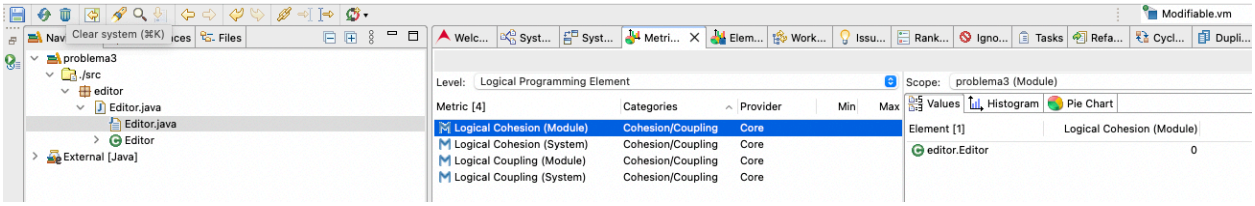
Acoplamiento

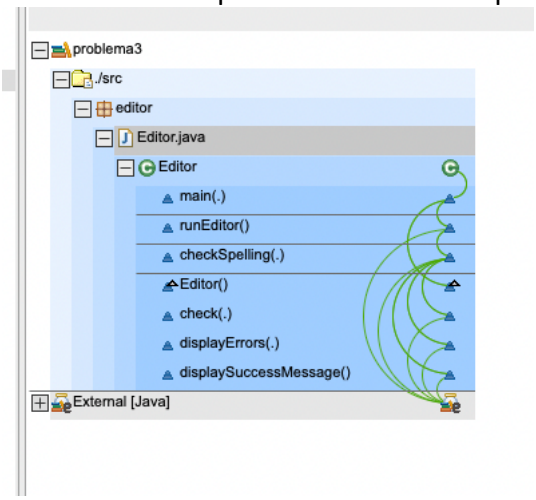
Problema 3



Según el análisis estructural de este problema podemos decir que el average component dependency es de 1, lo cual nos indica en este caso que, para nuestro pequeño proyecto, solo se debe administrar este archivo.
La cohesión inicial es de 0 al tener solamente un archivo en el proyecto



Las referencias apuntan hacía la clase principal



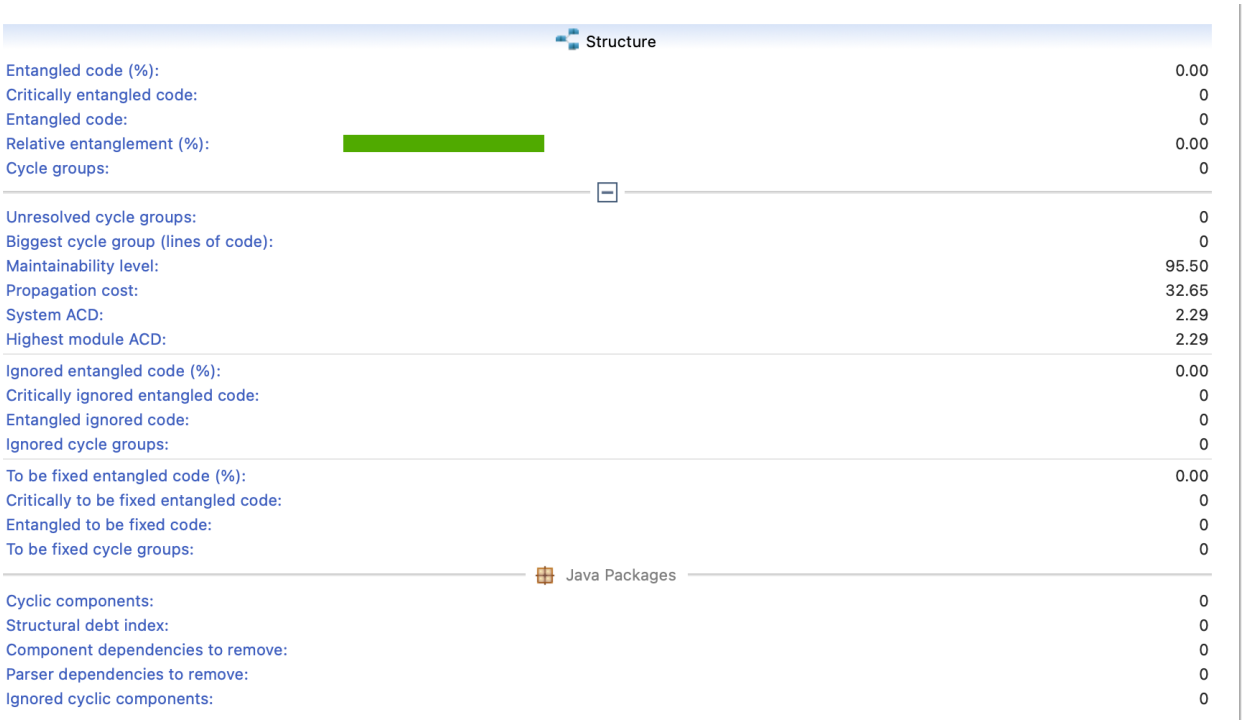
Refactoring propuesto

En este momento el código se encuentra condensado en un solo archivo lo que nos lleva al inconveniente de que a pesar de no tenemos métricas que nos muestren algún tipo de inconveniente en cuanto al funcionamiento de la aplicación, la clase está asumiendo responsabilidades que no le competen o que se podrían delegar a otras clases.

Resultados de refactoring sobre la clase editor

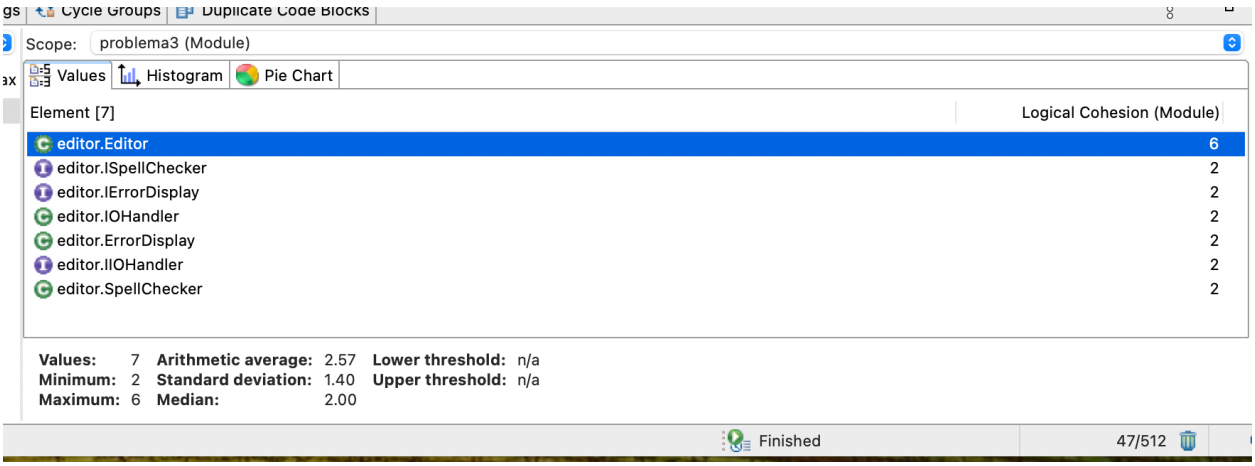
Structure	
Entangled code (%):	0.00
Critically entangled code:	0
Entangled code:	0
Relative entanglement (%):	0.00
Cycle groups:	0
Unresolved cycle groups:	0
Biggest cycle group (lines of code):	0
Maintainability level:	96.67
Propagation cost:	44.00
System ACD:	2.20
Highest module ACD:	2.20
Ignored entangled code (%):	0.00
Critically ignored entangled code:	0
Entangled ignored code:	0
Ignored cycle groups:	0
To be fixed entangled code (%):	0.00
Critically to be fixed entangled code:	0
Entangled to be fixed code:	0
To be fixed cycle groups:	0
Java Packages	
Cyclic components:	0
Structural debt index:	0
Component dependencies to remove:	0
Parser dependencies to remove:	0
Ignored cyclic components:	0

Vale la pena mencionar que, al tratar de desacoplar este tipo de código, según el tamaño del proyecto, se puede estar incurriendo en la adición de ACD, según el tamaño del proyecto se puede estar subiendo la complejidad de la arquitectura al intentar desacoplar las clases polifacéticas, así como el costo de propagación.
La ganancia se ve reflejada del lado de la legibilidad y facilidad de modificación al tener responsabilidades mejor definidas.

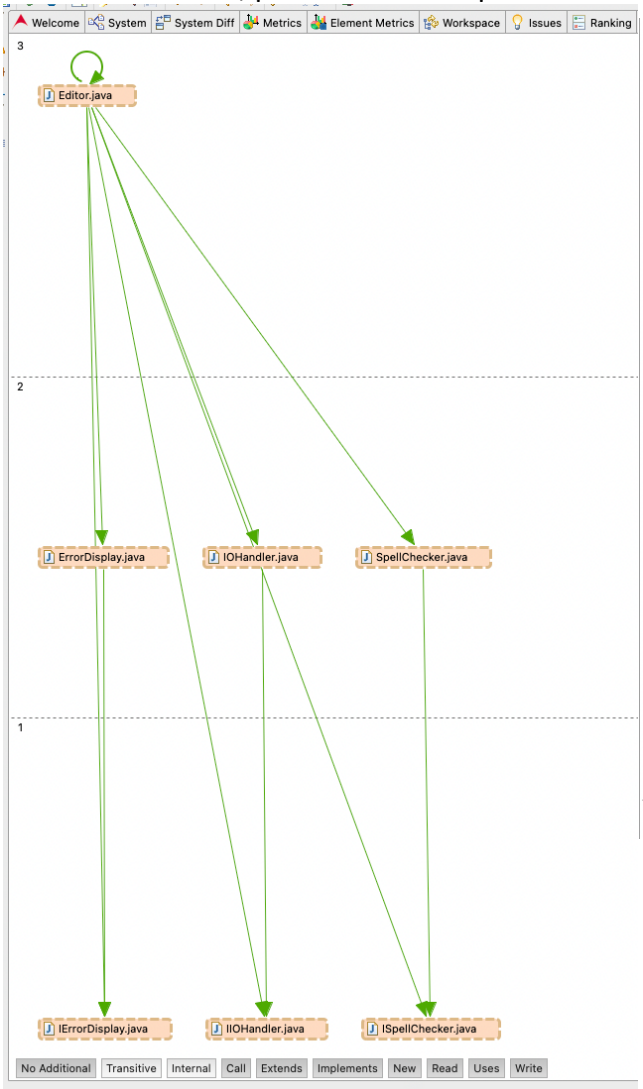


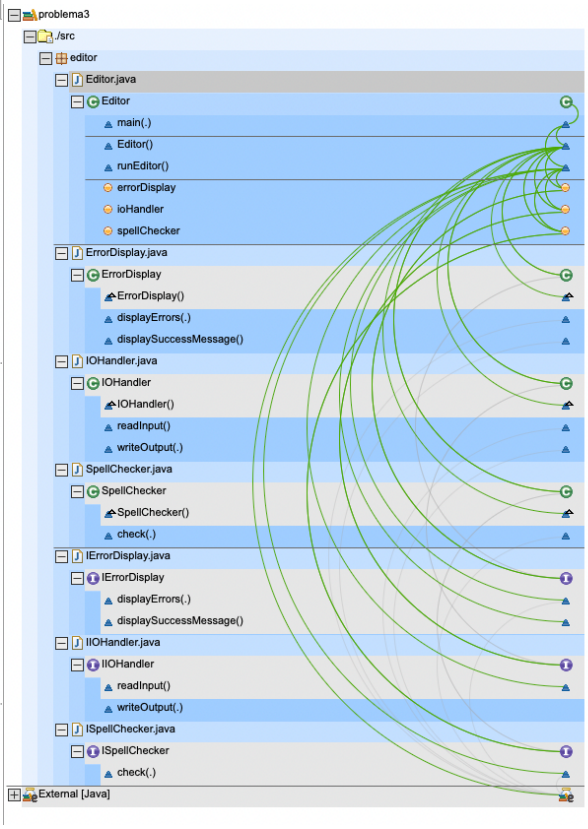
Así como lo demuestra este último análisis, se pueden agregar nuevas clases o separar el código en responsabilidades más definidas. Se comienza a comprometer el ACD del proyecto, pero se gana en propagación de los cambios. Para este último se realizó principalmente, la separación de la clase editor en clases, haciendo uso de interfases, para poder especializar según responsabilidades. Añadiendo a la clase Editor las instancias de:

- SpellChecker
- IOHandler
- ErrorDisplay



Luego de los cambios realizados podemos observar que la cohesión de la clase Editor es relativamente alta, para evitar el acoplamiento con las clases recién agregadas.





La dependencias creadas desde la clase Editor se presenta a las nuevas interfases más no a las clases directamente evitando el acoplamiento