Design Refactoring with Acellere Gamma Partitioning Tool

A Case Study

Key Terms

- CBO – Coupling Between Objects

https://help.mygamma.io/documentation/metrics/#coupling-between-objects

- RFC – Response for Class

https://help.mygamma.io/documentation/metrics/#response-for-class

- NOM – Number of Methods in a Class

https://help.mygamma.io/documentation/metrics/#number-of-methods

- cdisp – Coupling Dispersion, calculated as CBO / RFC
- ExecLOC – Executable Lines of Code

https://help.mygamma.io/documentation/metrics/#number-of-statements

- LCOM – Lack of Cohesion among Methods

https://help.mygamma.io/documentation/metrics/#lack-of-cohesion-of-methods

- God Class – Structural Design Anti-Pattern

https://help.mygamma.io/documentation/god-class/#anti-pattern-god-class

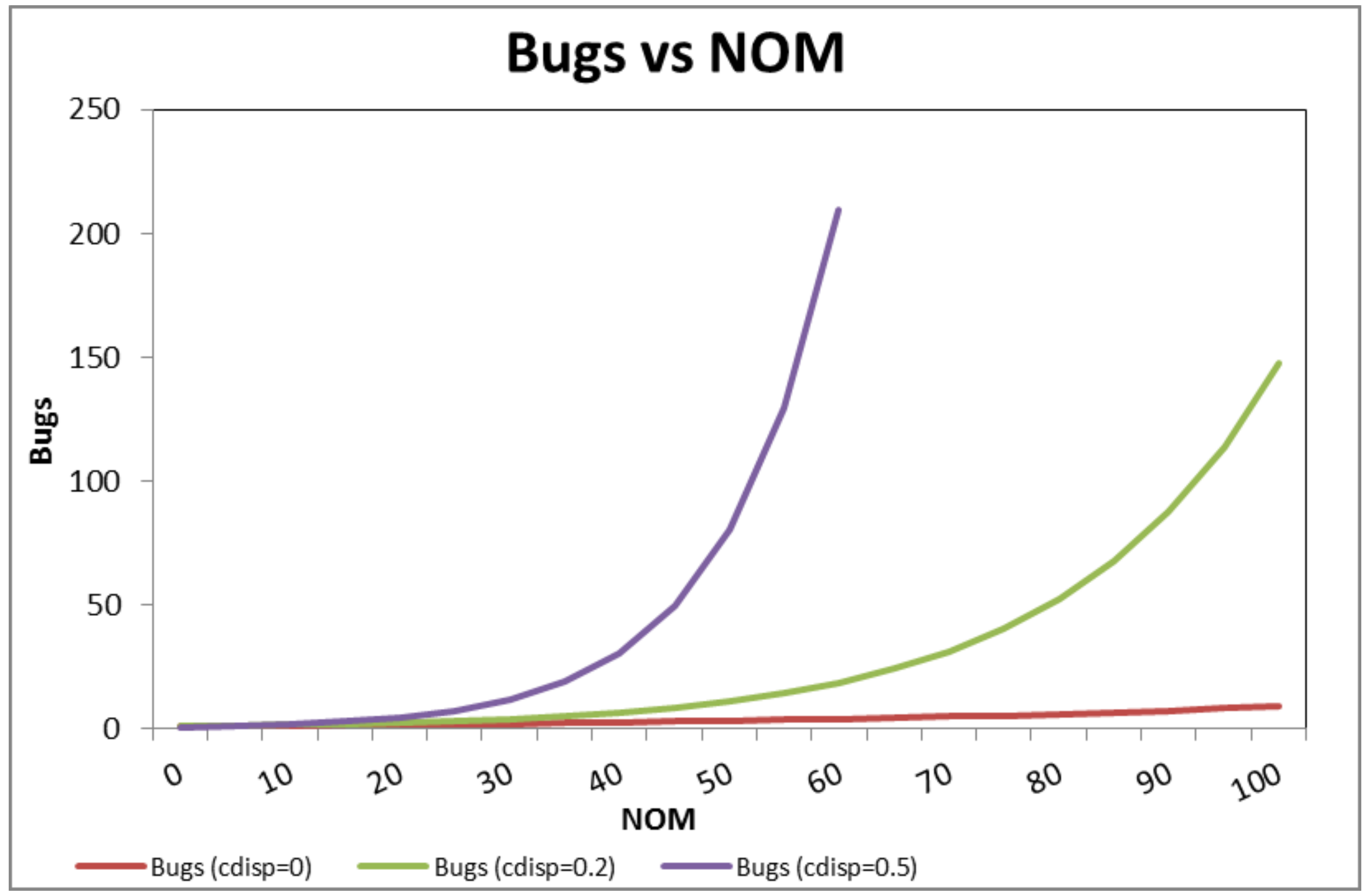
- Overall Rating – Quality Score of a code component as calculated by Gamma

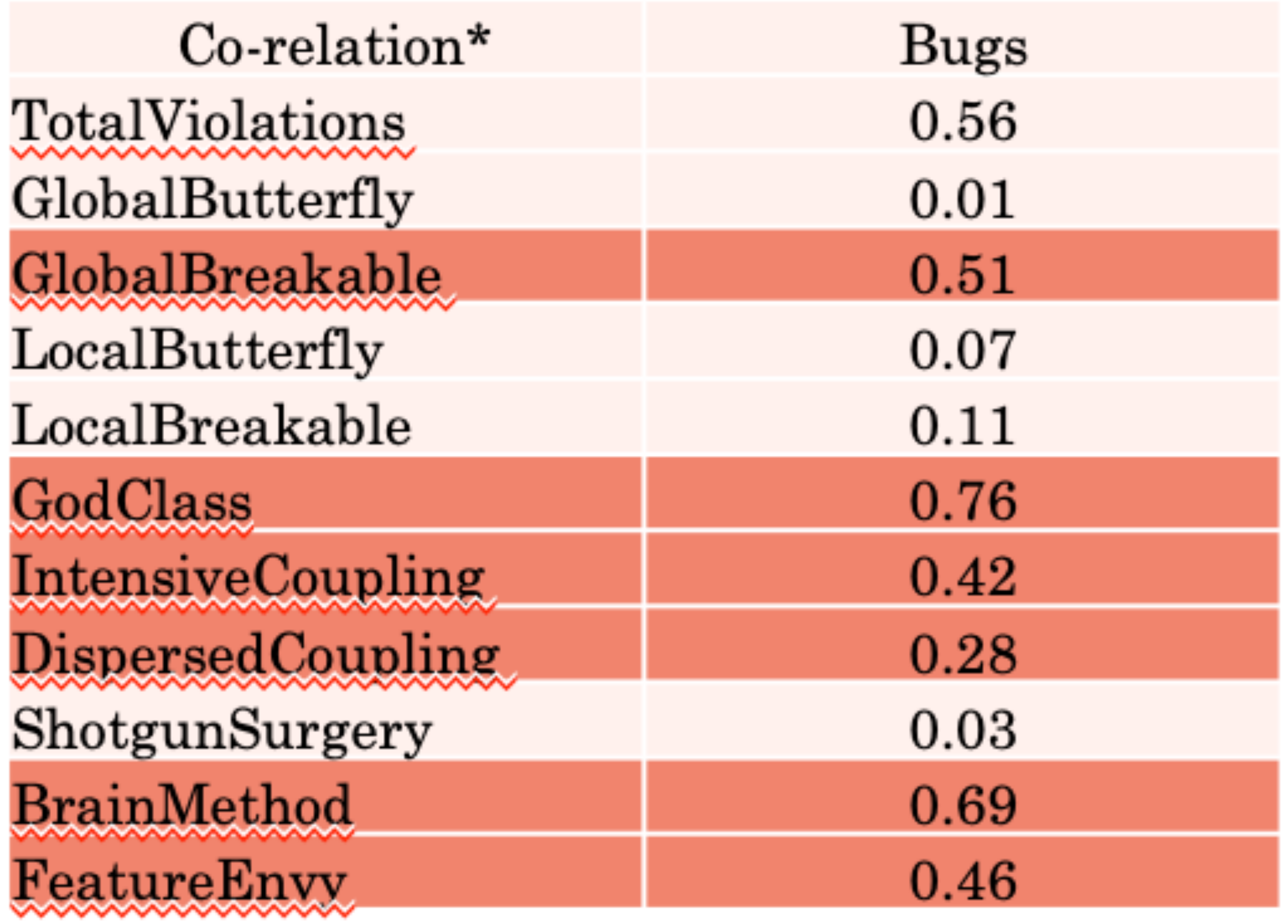
https://help.mygamma.io/guides/gamma-score/#the-gamma-score

- For other terms, refer: https://help.mygamma.io/documentation/

Motivation

- Certain metrics and design anti-patterns have a high correlation with bugs
- Example: bug counts increase with high NOM and a high coupling dispersion
- Example: A God Class has a 76% correlation with high number of bugs (i.e. chances of high bug counts due to bad design)
- Other design anti-patterns also have a fairly high correlation with bugs
- High values of these metrics/design issues also result in high amount of code churn when a feature is to be added or a bug is to be fixed

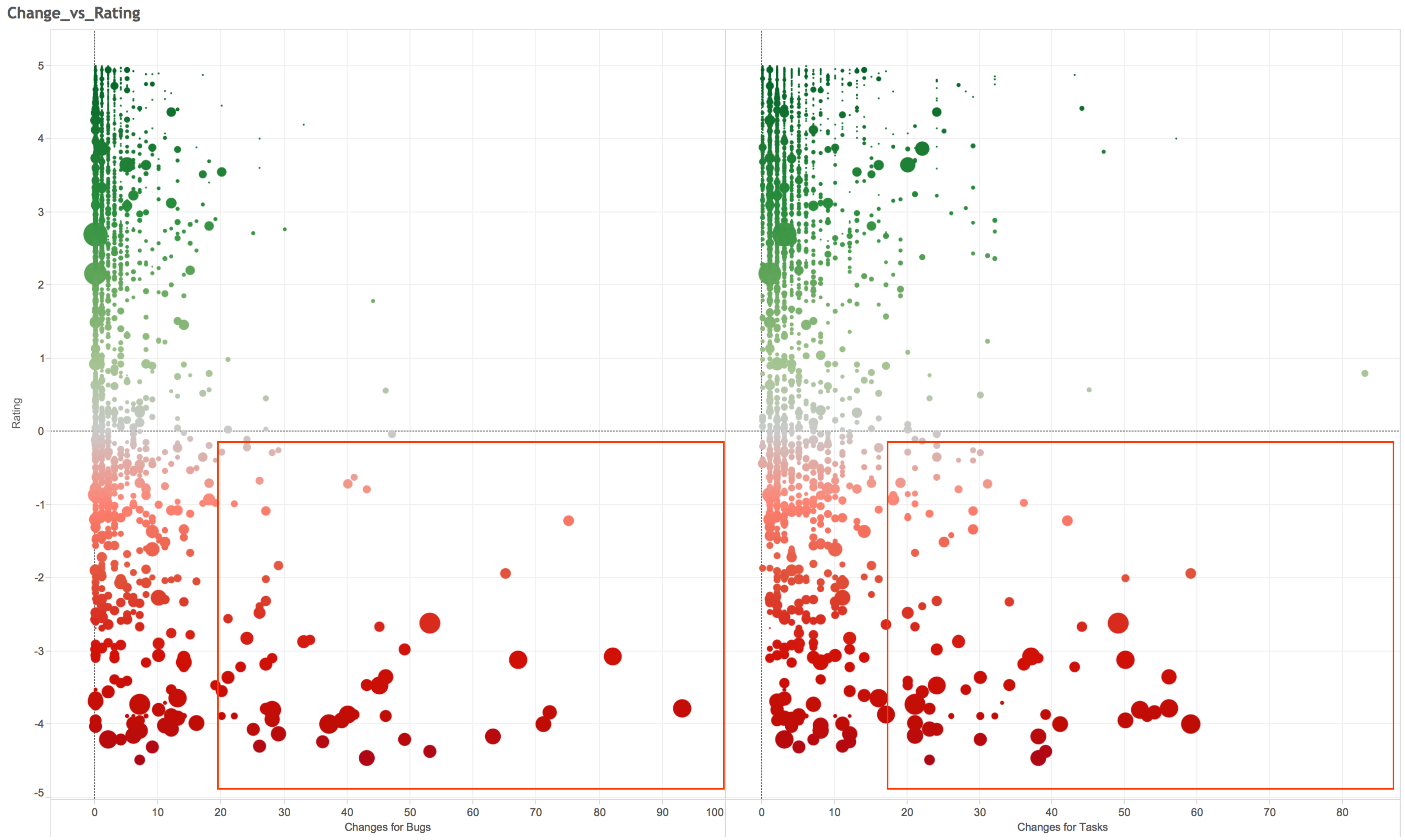




Co-relation*	Bugs
TotalViolations	0.56
GlobalButterfly	0.01
GlobalBreakable	0.51
LocalButterfly	0.07
LocalBreakable	0.11
GodClass	0.76
IntensiveCoupling	0.42
DispersedCoupling	0.28
ShotgunSurgery	0.03
BrainMethod	0.69
FeatureEnvy	0.46

Motivation

- The adjacent picture shows code components with a low design rating are frequently involved in bugs and features (tasks)
- This means they go through multiple, frequent changes, are difficult to maintain, and if not refactored, can lead to an increased risk of bugs and maintainability issues over time



It follows that design issues are contributors to bugs, and improving design will reduce bugs and improve long-term maintainability

Refactoring support in Gamma to improve design

- Typical design attributes related to high bugs in a component and frequent churn are: lack of separation of concerns, lack of encapsulation and loss of abstraction
- This results in monolithic components which are usually changed frequently as they aggregate multiple disparate functionalities and are deeply coupled with other parts of the system
- These design issues emerge over time when new functionality is added without evaluating if it belongs to the right component, and hence results in unwanted dependencies, high coupling, exposure of data, and loss of abstraction
- Gamma’s Partitioning Tool helps developers fix such issues in existing code by identifying abstractions and suggesting new components which will result in a cleaner, more maintainable and cohesive structure
- It helps fix design anti-patterns such as God Class, which is responsible for bugs from design perspective, and in that process, also improve metrics such as coupling, LCOM, Number of Methods, etc.
- The following slides illustrate an actual example class refactored with the help of Partitioning tool, and shows how it helps fix design issues

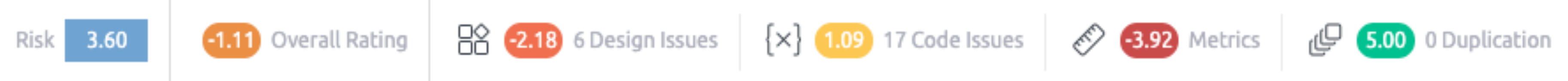
Refactoring Process and Example Source

- Apache Kafka: https://github.com/apache/kafka.git
- Java Class: org.apache.kafka.streams.processor.internals.InternalTopologyBuilder
- This class was chosen because it is a hotspot (Gamma score < 0), changed frequently, and has several design issues
- In this exercise, multiple iterations of refactoring were performed, guided by the Gamma Partitioning Tool, and at each logical step, a Gamma scan was done to measure improvements

Before – Class InternalTopologyBuilder

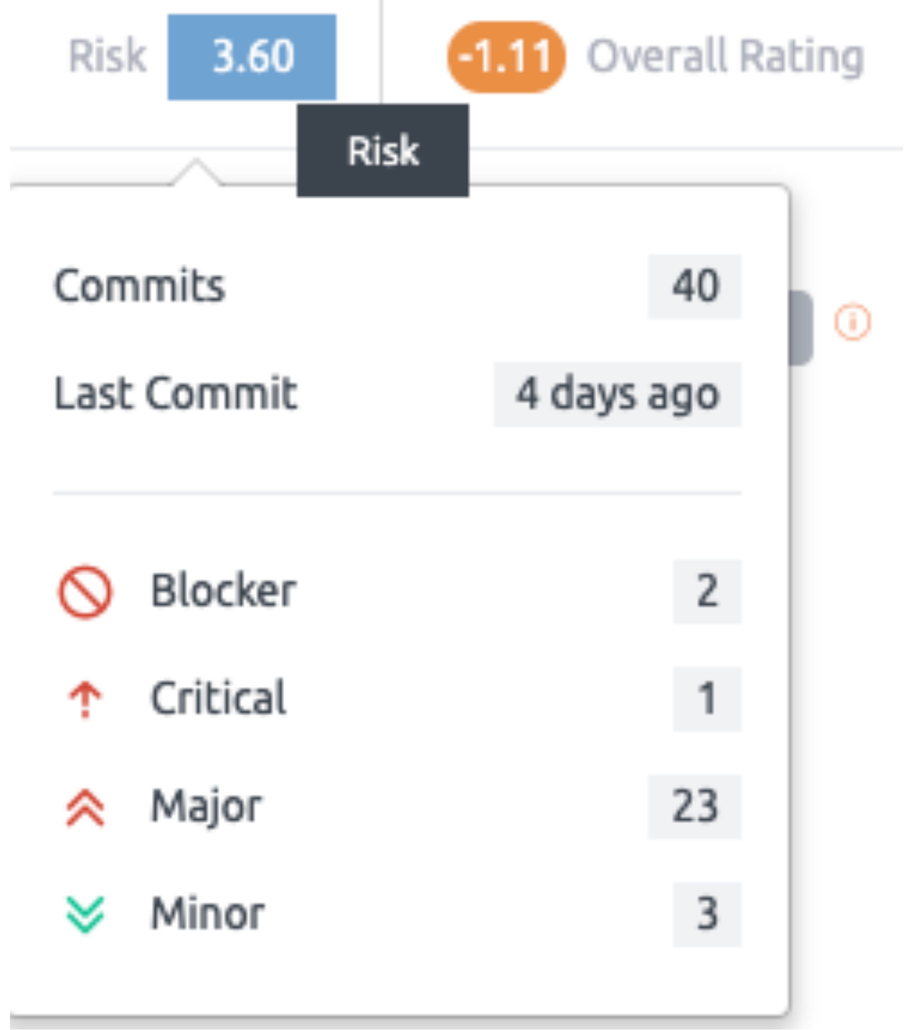
- Characteristics:
- Frequently changed and participating in bugs (extracted from Apache Jira: https://issues.apache.org/jira/)
- God Class and other design issues, many metrics violations
- No duplication (good), some code issues

Hotspot: Overall Rating < 0



Risk 3.60 | -1.11 Overall Rating | -2.18 6 Design Issues | {x} 1.09 17 Code Issues | -3.92 Metrics | 5.00 0 Duplication

Frequently Changed, involved in bugs

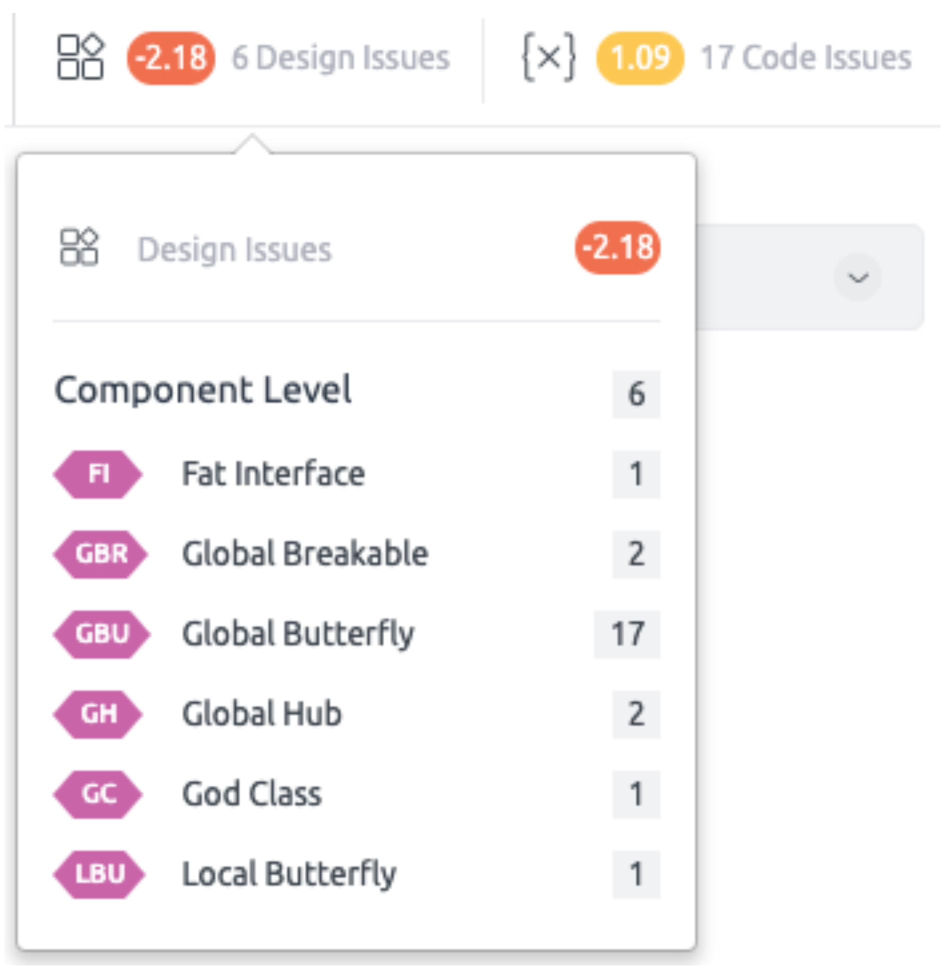


Risk 3.60 | -1.11 Overall Rating

Risk

Commits	40
Last Commit	4 days ago
Blocker	2
Critical	1
Major	23
Minor	3

Several Design Issues – esp. God Class

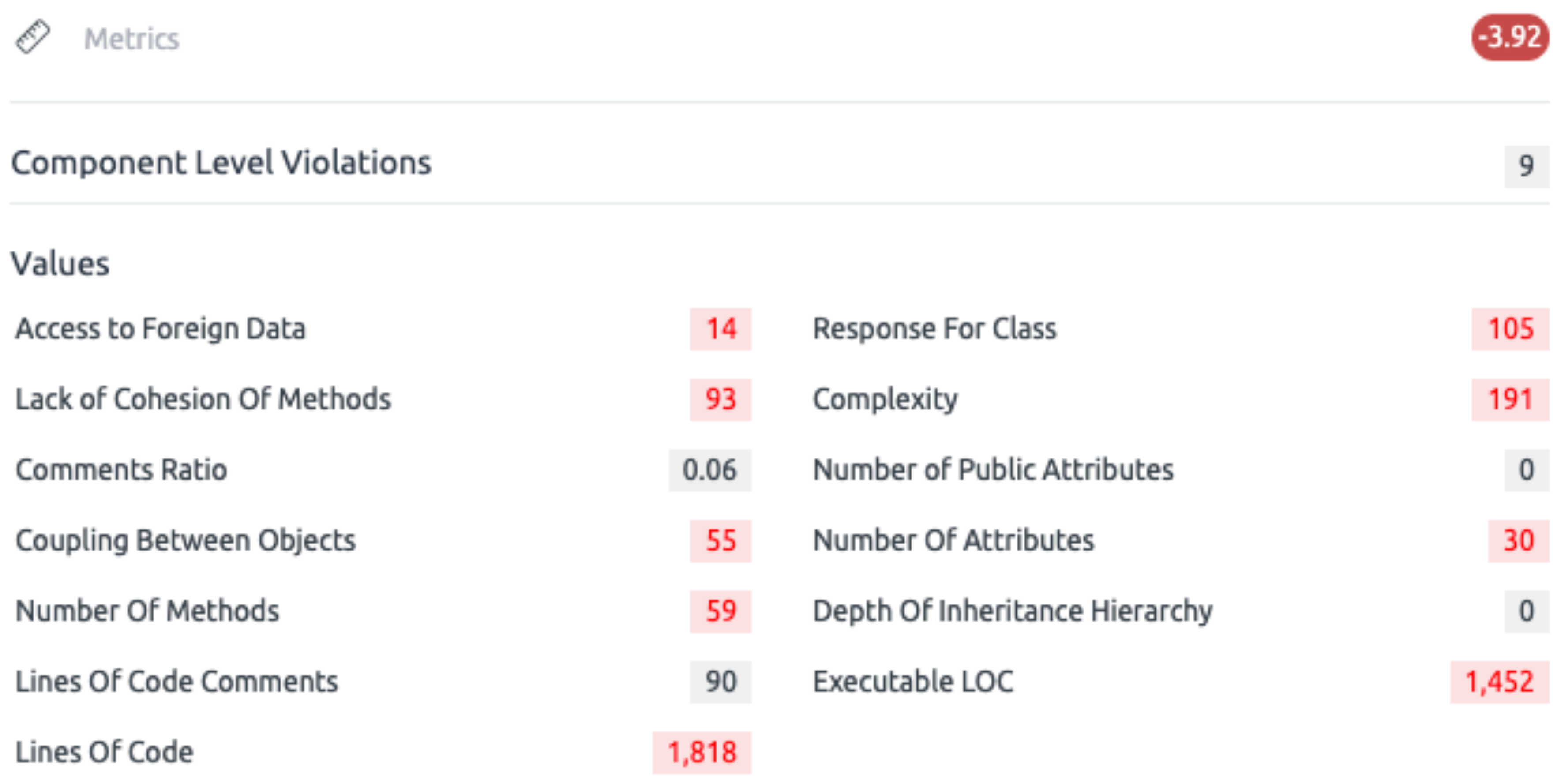


-2.18 6 Design Issues | {x} 1.09 17 Code Issues

Design Issues -2.18

	Component Level	6
FI	Fat Interface	1
GBR	Global Breakable	2
GBU	Global Butterfly	17
GH	Global Hub	2
GC	God Class	1
LBU	Local Butterfly	1

Many metrics violations – high coupling, lack of cohesion, too many methods, high lines of code



Metrics -3.92

Component Level Violations 9

Values

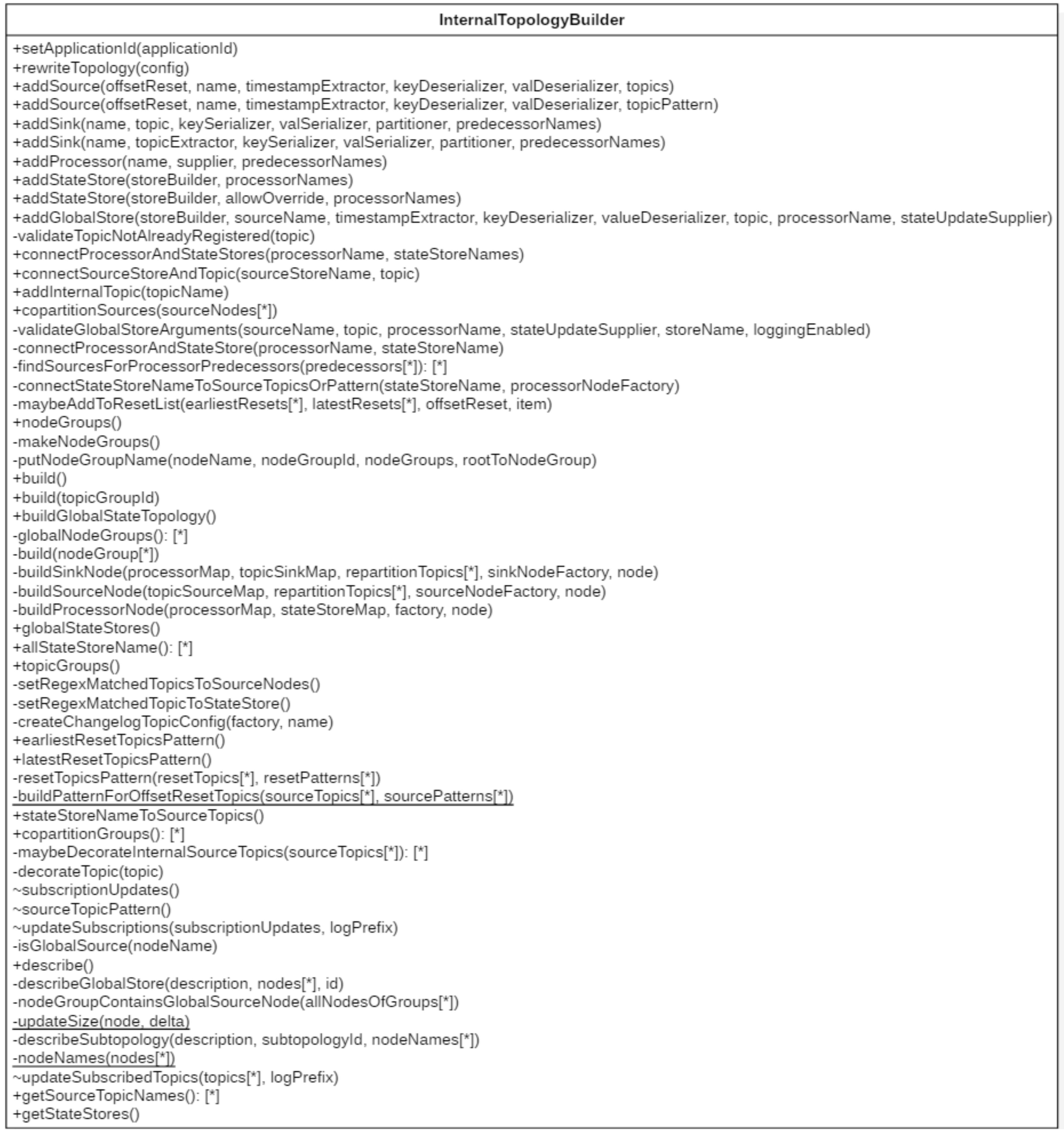
Access to Foreign Data	14	Response For Class	105
Lack of Cohesion Of Methods	93	Complexity	191
Comments Ratio	0.06	Number of Public Attributes	0
Coupling Between Objects	55	Number Of Attributes	30
Number Of Methods	59	Depth Of Inheritance Hierarchy	0
Lines Of Code Comments	90	Executable LOC	1,452
Lines Of Code	1,818		

Analysis - Class InternalTopologyBuilder

Refactoring conditions

- Public interface should remain unchanged (as we don't want the client-side code to change)
- As a result, some dependency-related design issues (e.g. Global Butterfly) will not be addressed, because we are not changing the public interface
- Existing class should not be a hotspot anymore (overall rating > 0)
- God Class design issue should be fixed
- Class size, number of methods, coupling should reduce
- Resulting additional classes should not be hotspots or God Classes

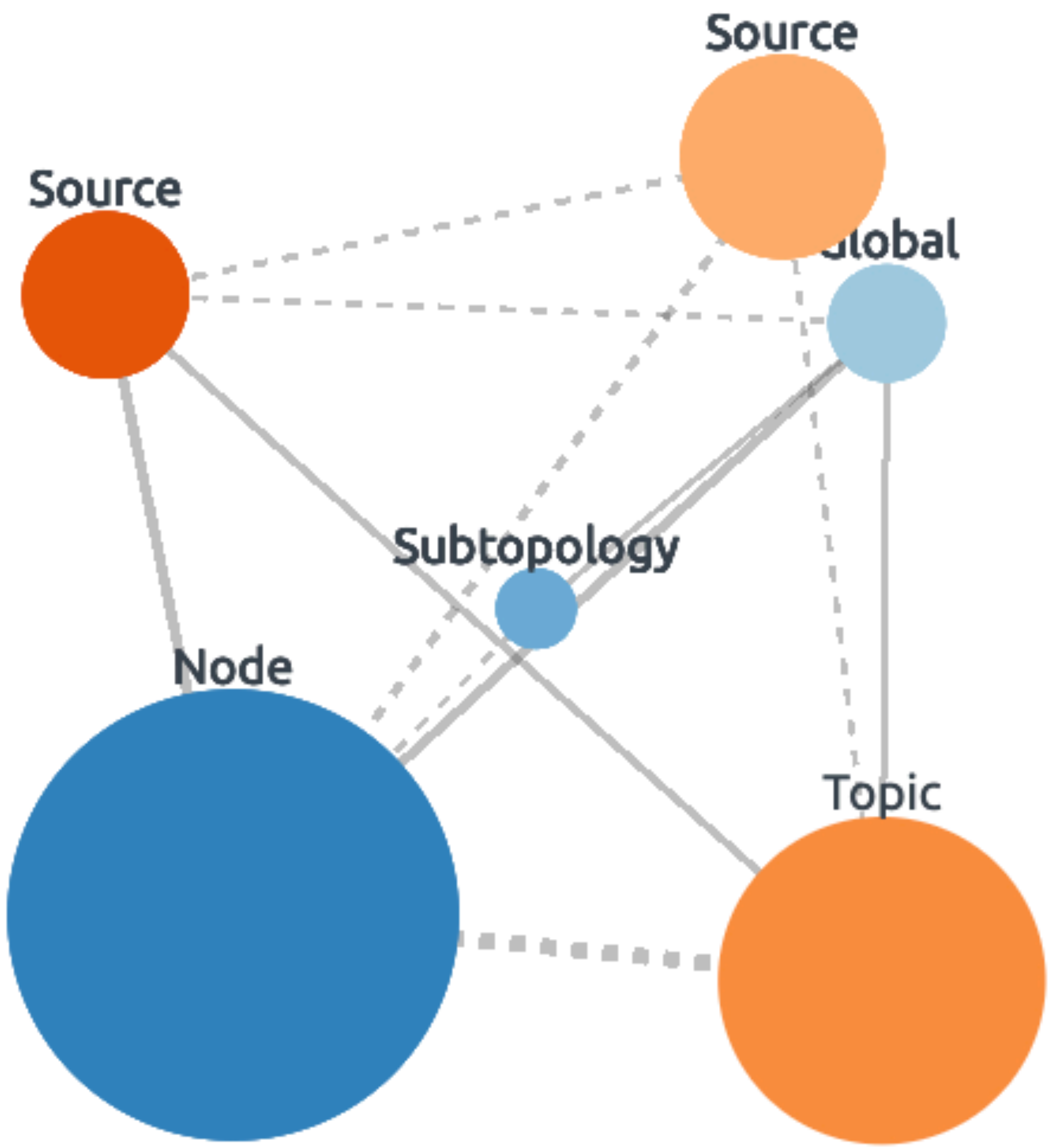
Large public interface



```
InternalTopologyBuilder
+setApplicationId(applicationId)
+rewriteTopology(config)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topics)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topicPattern)
+addSink(name, topic, keySerializer, valSerializer, partitioner, predecessorNames)
+addSink(name, topicExtractor, keySerializer, valSerializer, partitioner, predecessorNames)
+addProcessor(name, supplier, predecessorNames)
+addStateStore(storeBuilder, processorNames)
+addStateStore(storeBuilder, allowOverride, processorNames)
+addGlobalStore(storeBuilder, sourceName, timestampExtractor, keyDeserializer, valueDeserializer, topic, processorName, stateUpdateSupplier)
-validateTopicNotAlreadyRegistered(topic)
+connectProcessorAndStateStores(processorName, stateStoreNames)
+connectSourceStoreAndTopic(sourceStoreName, topic)
+addInternalTopic(topicName)
+copartitionSources(sourceNodes[*])
-validateGlobalStoreArguments(sourceName, topic, processorName, stateUpdateSupplier, storeName, loggingEnabled)
-connectProcessorAndStateStore(processorName, stateStoreName)
-findSourcesForProcessorPredecessors(predecessors[*]): [*]
-connectStateStoreNameToSourceTopicsOrPattern(stateStoreName, processorNodeFactory)
-maybeAddToResetList(earliestResets[*], latestResets[*], offsetReset, item)
+nodeGroups()
-makeNodeGroups()
-putNodeGroupName(nodeName, nodeGroupId, nodeGroups, rootToNodeGroup)
+build()
+build(topicGroupId)
+buildGlobalStateTopology()
-globalNodeGroups(): [*]
-build(nodeGroup[*])
-buildSinkNode(processorMap, topicSinkMap, repartitionTopics[*], sinkNodeFactory, node)
-buildSourceNode(topicSourceMap, repartitionTopics[*], sourceNodeFactory, node)
-buildProcessorNode(processorMap, stateStoreMap, factory, node)
+globalStateStores()
+allStateStoreName(): [*]
+topicGroups()
-setRegexMatchedTopicsToSourceNodes()
-setRegexMatchedTopicToStateStore()
-createChangelogTopicConfig(factory, name)
+earliestResetTopicsPattern()
+latestResetTopicsPattern()
-resetTopicsPattern(resetTopics[*], resetPatterns[*])
-buildPatternForOffsetResetTopics(sourceTopics[*], sourcePatterns[*])
+stateStoreNameToSourceTopics()
+copartitionGroups(): [*]
-maybeDecorateInternalSourceTopics(sourceTopics[*]): [*]
-decorateTopic(topic)
~subscriptionUpdates()
~sourceTopicPattern()
~updateSubscriptions(subscriptionUpdates, logPrefix)
-isGlobalSource(nodeName)
+describe()
-describeGlobalStore(description, nodes[*], id)
-nodeGroupContainsGlobalSourceNode(allNodesOfGroups[*])
-updateSize(node, delta)
-describeSubtopology(description, subtopologyId, nodeNames[*])
-nodeNames(nodes[*])
~updateSubscribedTopics(topics[*], logPrefix)
+getSourceTopicNames(): [*]
+getStateStores()
```

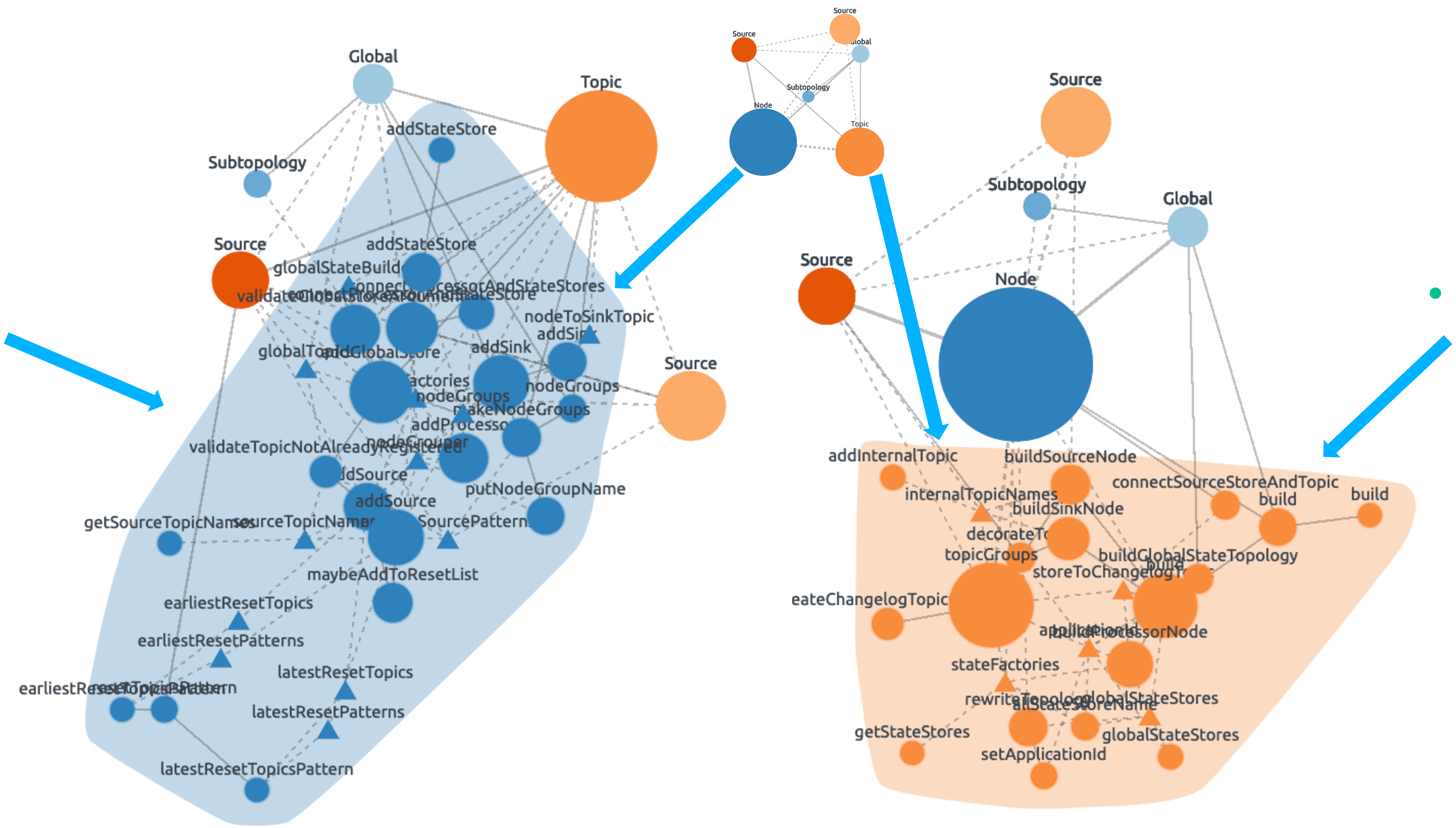
Before State – Identified Partitions

- Partitions identified by the Gamma Partitioning Tool suggest 3 separate abstractions: Source, Topic, Node
- An ideally designed class will have fewer (or just single) abstractions as it represents a single concern
- As a first step, we will extract the Source, Topic and Global abstractions



Before State – Identified Partitions Drilldown

- Node,
Store,
Pattern

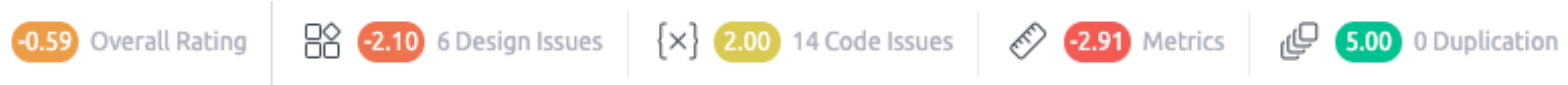


- Topic,
State, Store

Iteration 1 – Refactor Action

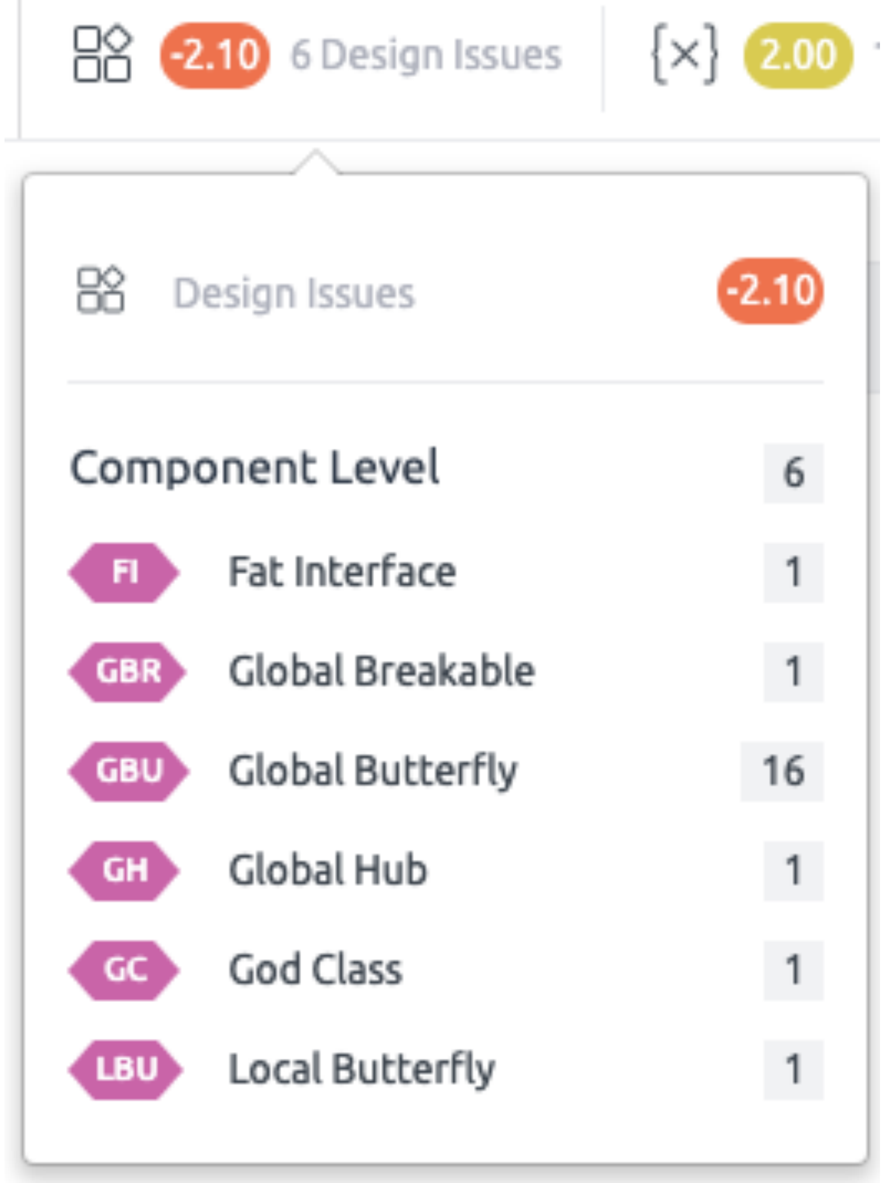
- Extract new class Refac_Topic to represent Topic, Store and Node builder related functionality, which is fairly cohesive
- Extract new class Refac_SourceSink to represent logic related to managing sources and sinks connected with nodes in a topology
- Also create a new class Refac_GlobalTopics to represent the global topics (“Global” partition in the previous picture)

Iteration 1 result – Simplified Class InternalTopologyBuilder



-0.59 Overall Rating | -2.10 6 Design Issues | {x} 2.00 14 Code Issues | -2.91 Metrics | 5.00 0 Duplication

Improved Design Rating

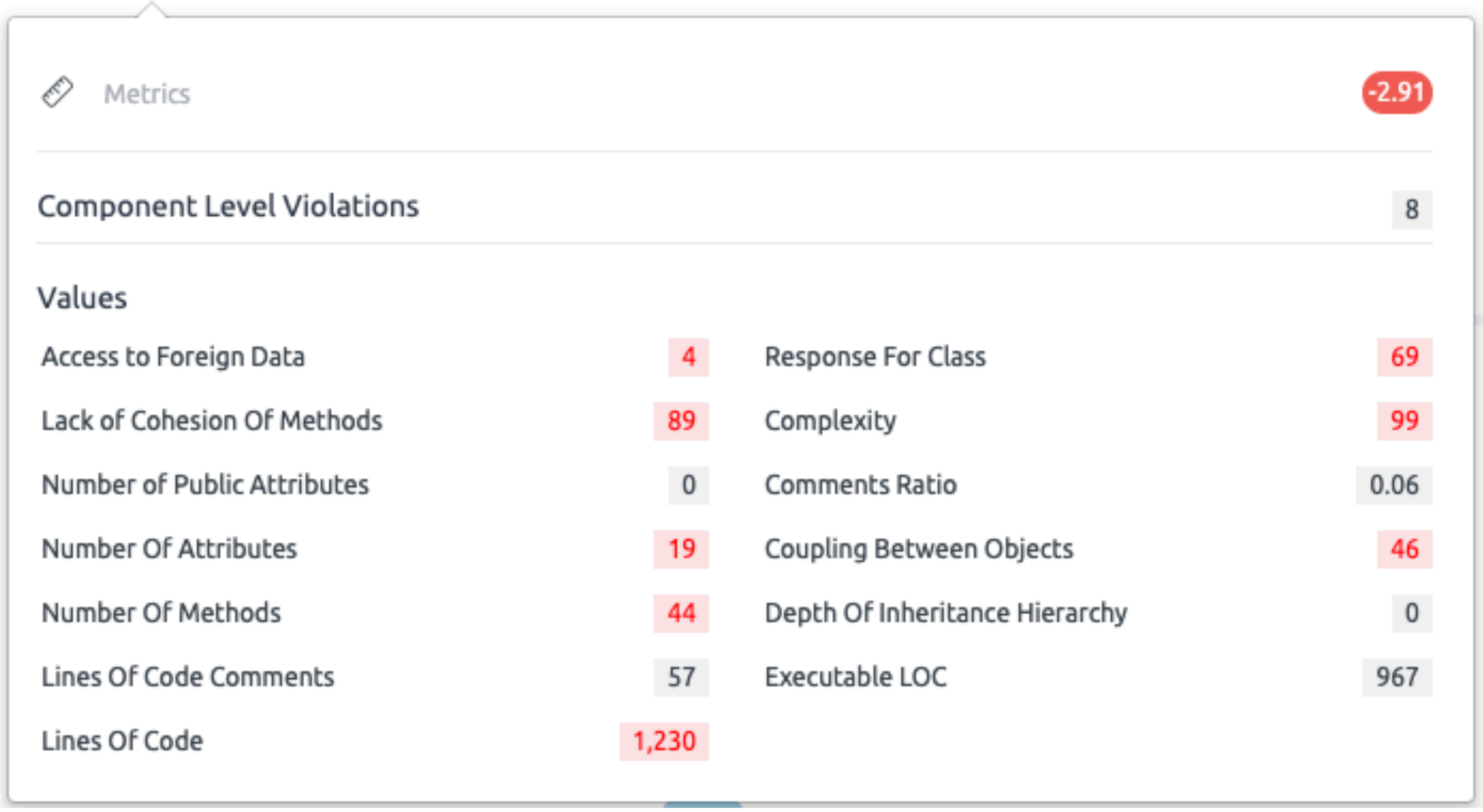


-2.10 6 Design Issues {x} 2.00

Design Issues -2.10

Component Level		6
FI	Fat Interface	1
GBR	Global Breakable	1
GBU	Global Butterfly	16
GH	Global Hub	1
GC	God Class	1
LBU	Local Butterfly	1

Improved Metrics



Metrics -2.91

Component Level Violations 8

Values

Access to Foreign Data	4	Response For Class	69
Lack of Cohesion Of Methods	89	Complexity	99
Number of Public Attributes	0	Comments Ratio	0.06
Number Of Attributes	19	Coupling Between Objects	46
Number Of Methods	44	Depth Of Inheritance Hierarchy	0
Lines Of Code Comments	57	Executable LOC	967
Lines Of Code	1,230		

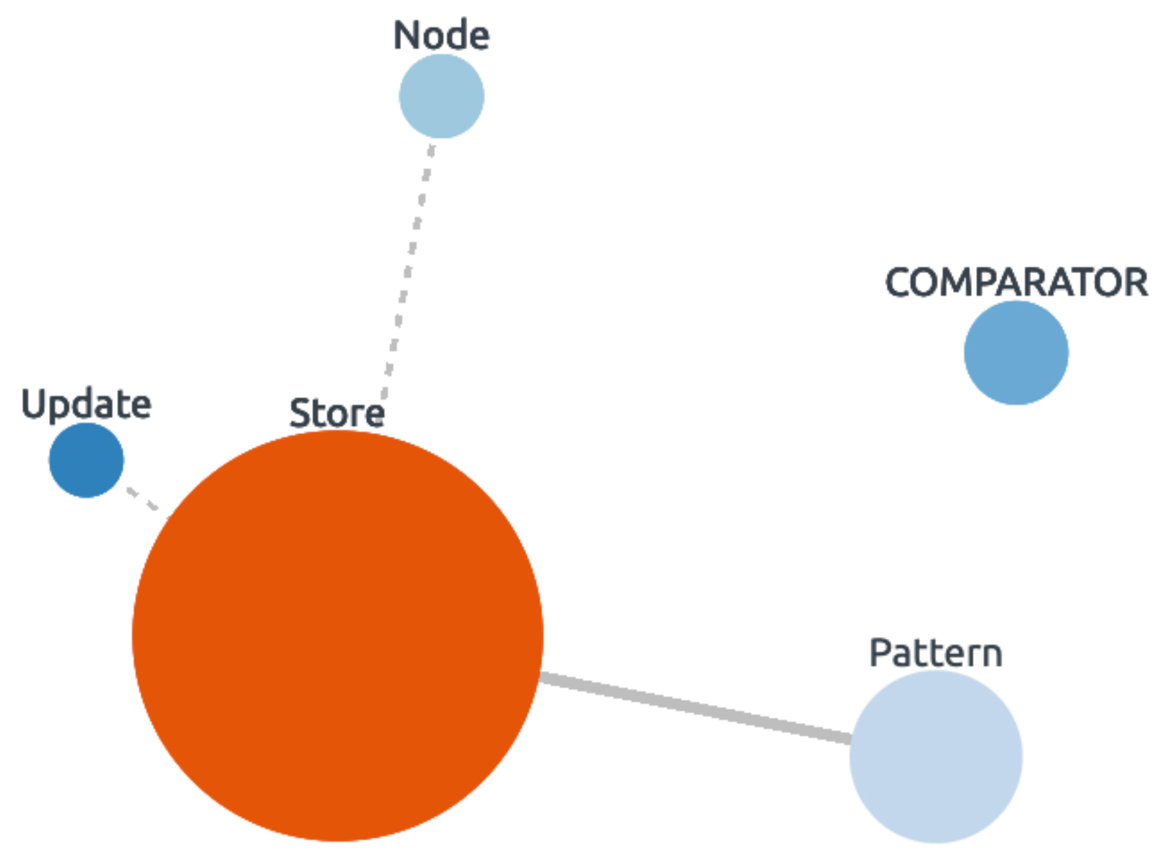
Result

- Improved rating: -1.11 to -0.59
- Improved design rating: -2.18 to -2.10
- Improved cohesion (93 to 89)
- Fewer Methods, Reduced Coupling
- Still a hotspot (overall rating < 0), still a God class, although less severe, improved overall metrics
- More improvement needed!

Iteration 1 result – Simpler Partitions

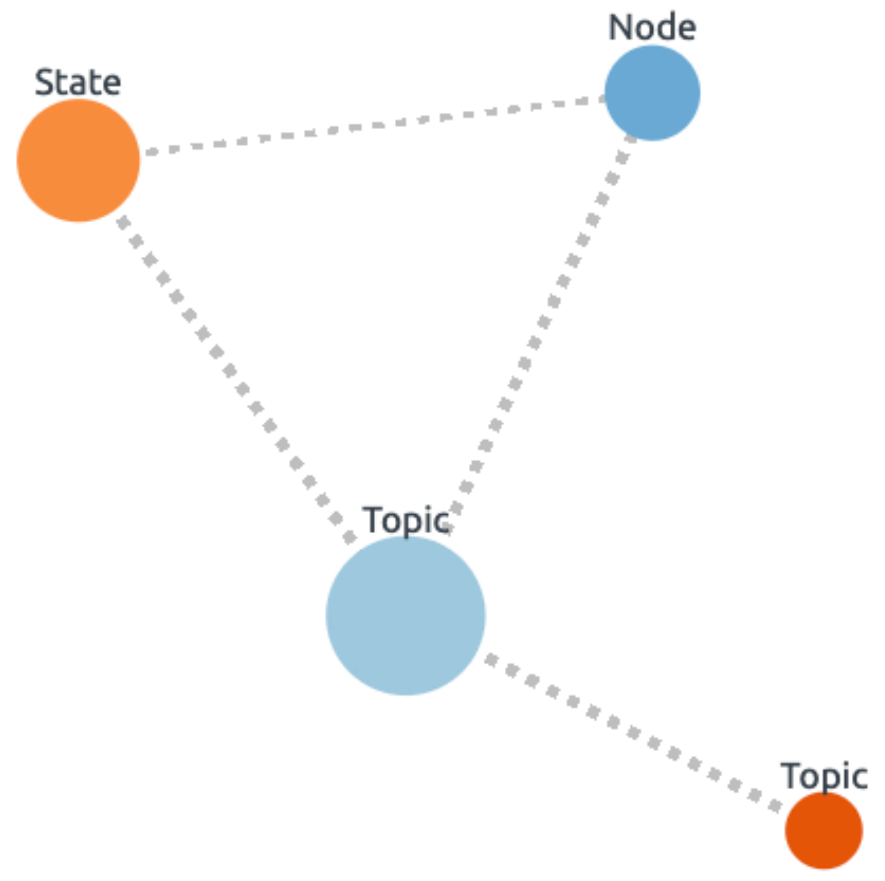
-0.59 2.10 2.00 -2.91 5.00

InternalTopologyBuilder



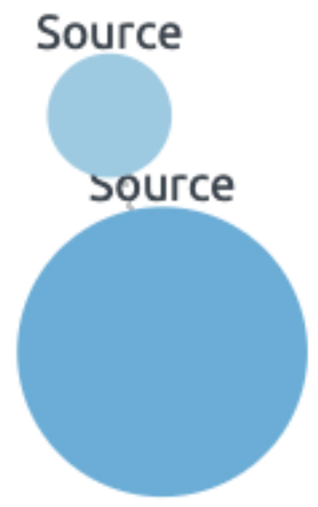
0.54 0.08 0.13 -1.23 5.00

Refac_Topic



2.58 3.48 2.50 0.70 5.00

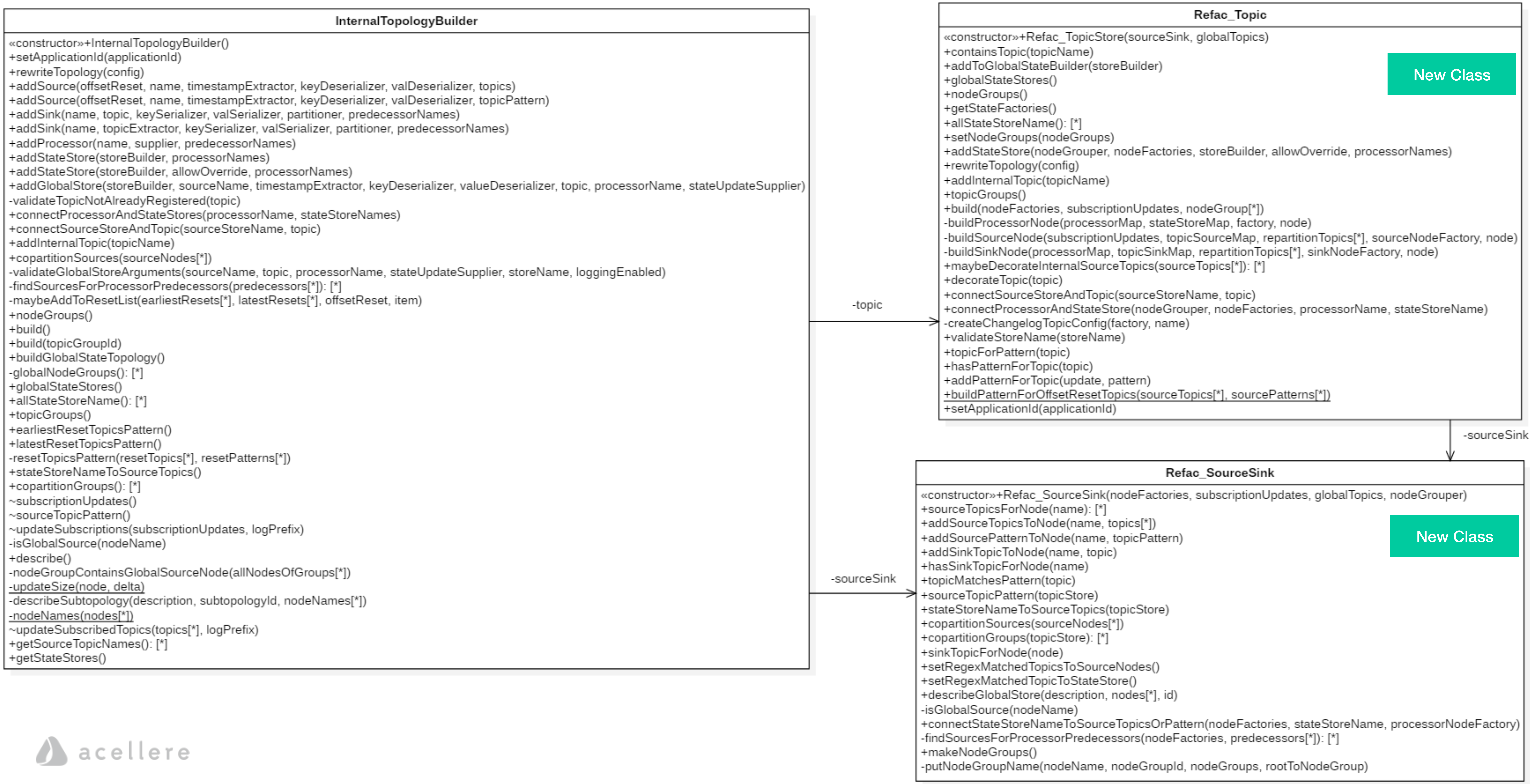
Refac_SourceSink



Result

- In Iteration 1 we reduced 2 of the large partitions of InternalTopologyBuilder by creating the Refac_Topic and RefacSourceSink classes which represent those abstractions more cohesively, rather than aggregating everything in InternalTopologyBuilder
- This resulted in simplified partitions for the original class, as well as the new classes, which have fairly cohesive, and not large, partitions
- Iteration 2 improves this further by additional partitioning of InternalTopologyBuilder

Iteration 1 result – Simpler Partitions



InternalTopologyBuilder

```
«constructor»+InternalTopologyBuilder()
+setApplicationId(applicationId)
+rewriteTopology(config)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topics)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topicPattern)
+addSink(name, topic, keySerializer, valSerializer, partitioner, predecessorNames)
+addSink(name, topicExtractor, keySerializer, valSerializer, partitioner, predecessorNames)
+addProcessor(name, supplier, predecessorNames)
+addStateStore(storeBuilder, processorNames)
+addStateStore(storeBuilder, allowOverride, processorNames)
+addGlobalStore(storeBuilder, sourceName, timestampExtractor, keyDeserializer, valueDeserializer, topic, processorName, stateUpdateSupplier)
-validateTopicNotAlreadyRegistered(topic)
+connectProcessorAndStateStores(processorName, stateStoreNames)
+connectSourceStoreAndTopic(sourceStoreName, topic)
+addInternalTopic(topicName)
+copartitionSources(sourceNodes[*])
-validateGlobalStoreArguments(sourceName, topic, processorName, stateUpdateSupplier, storeName, loggingEnabled)
-findSourcesForProcessorPredecessors(predecessors[*]): [*]
-maybeAddToResetList(earliestResets[*], latestResets[*], offsetReset, item)
+nodeGroups()
+build()
+build(topicGroupId)
+buildGlobalStateTopology()
-globalNodeGroups(): [*]
+globalStateStores()
+allStateStoreName(): [*]
+topicGroups()
+earliestResetTopicsPattern()
+latestResetTopicsPattern()
-resetTopicsPattern(resetTopics[*], resetPatterns[*])
+stateStoreNameToSourceTopics()
+copartitionGroups(): [*]
~subscriptionUpdates()
~sourceTopicPattern()
~updateSubscriptions(subscriptionUpdates, logPrefix)
-isGlobalSource(nodeName)
+describe()
-nodeGroupContainsGlobalSourceNode(allNodesOfGroups[*])
-updateSize(node, delta)
-describeSubtopology(description, subtopologyId, nodeNames[*])
-nodeNames(nodes[*])
~updateSubscribedTopics(topics[*], logPrefix)
+getSourceTopicNames(): [*]
+getStateStores()
```

Refac_Topic

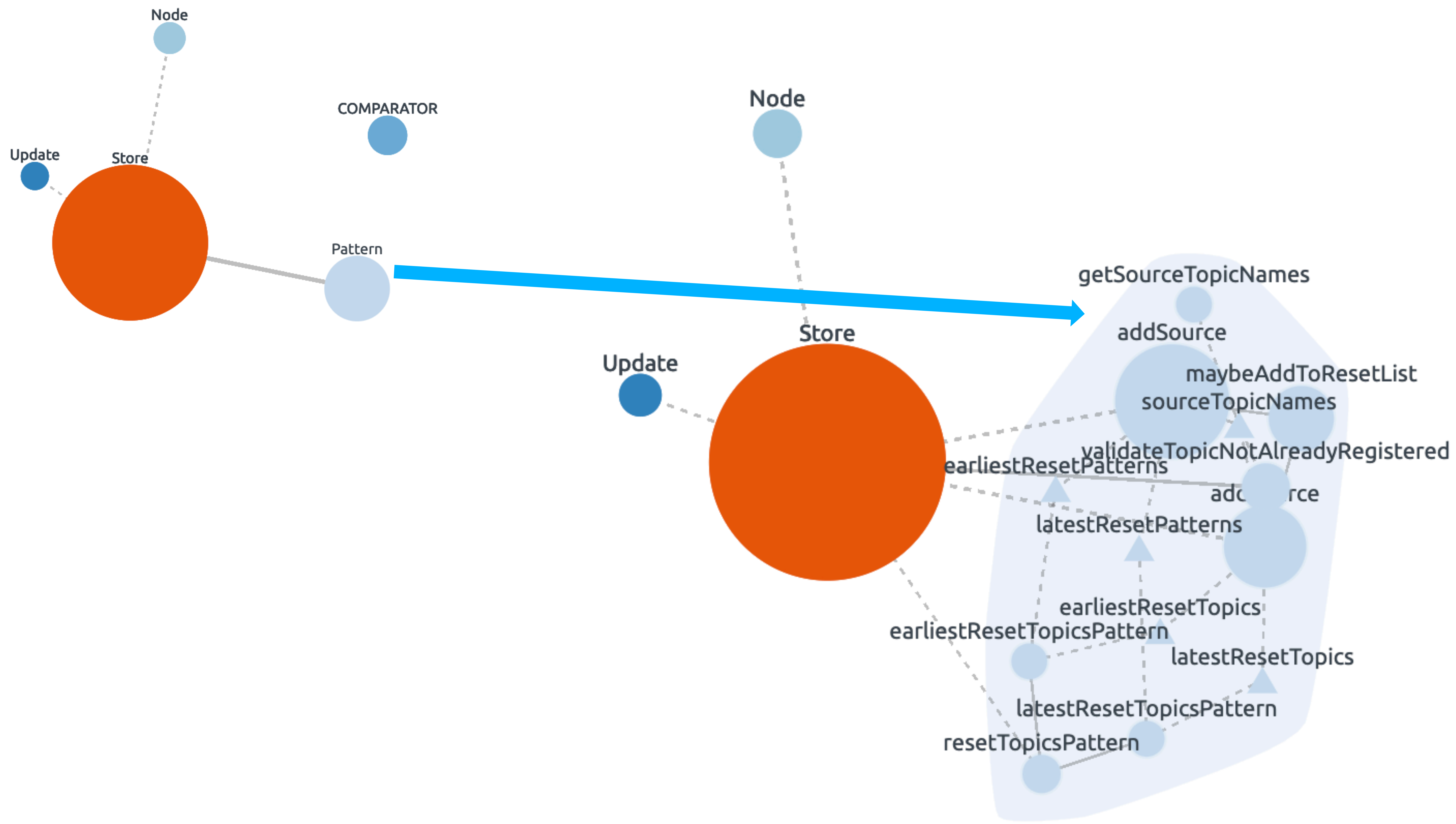
```
«constructor»+Refac_TopicStore(sourceSink, globalTopics)
+containsTopic(topicName)
+addToGlobalStateBuilder(storeBuilder)
+globalStateStores()
+nodeGroups()
+getStateFactories()
+allStateStoreName(): [*]
+setNodeGroups(nodeGroups)
+addStateStore(nodeGrouper, nodeFactories, storeBuilder, allowOverride, processorNames)
+rewriteTopology(config)
+addInternalTopic(topicName)
+topicGroups()
+build(nodeFactories, subscriptionUpdates, nodeGroup[*])
-buildProcessorNode(processorMap, stateStoreMap, factory, node)
-buildSourceNode(subscriptionUpdates, topicSourceMap, repartitionTopics[*], sourceNodeFactory, node)
-buildSinkNode(processorMap, topicSinkMap, repartitionTopics[*], sinkNodeFactory, node)
+maybeDecorateInternalSourceTopics(sourceTopics[*]): [*]
+decorateTopic(topic)
+connectSourceStoreAndTopic(sourceStoreName, topic)
+connectProcessorAndStateStore(nodeGrouper, nodeFactories, processorName, stateStoreName)
-createChangelogTopicConfig(factory, name)
+validateStoreName(storeName)
+topicForPattern(topic)
+hasPatternForTopic(topic)
+addPatternForTopic(update, pattern)
+buildPatternForOffsetResetTopics(sourceTopics[*], sourcePatterns[*])
+setApplicationId(applicationId)
```

Refac_SourceSink

```
«constructor»+Refac_SourceSink(nodeFactories, subscriptionUpdates, globalTopics, nodeGrouper)
+sourceTopicsForNode(name): [*]
+addSourceTopicsToNode(name, topics[*])
+addSourcePatternToNode(name, topicPattern)
+addSinkTopicToNode(name, topic)
+hasSinkTopicForNode(name)
+topicMatchesPattern(topic)
+sourceTopicPattern(topicStore)
+stateStoreNameToSourceTopics(topicStore)
+copartitionSources(sourceNodes[*])
+copartitionGroups(topicStore): [*]
+sinkTopicForNode(node)
+setRegexMatchedTopicsToSourceNodes()
+setRegexMatchedTopicToStateStore()
+describeGlobalStore(description, nodes[*], id)
-isGlobalSource(nodeName)
+connectStateStoreNameToSourceTopicsOrPattern(nodeFactories, stateStoreName, processorNodeFactory)
-findSourcesForProcessorPredecessors(nodeFactories, predecessors[*]): [*]
+makeNodeGroups()
-putNodeGroupName(nodeName, nodeGroupId, nodeGroups, rootToNodeGroup)
```

Iteration 2 – Identified Partitions – InternalTopologyBuilder

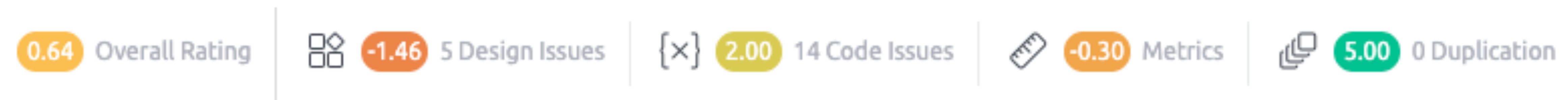
- Extract Pattern (on Topic) abstraction to its own class



Iteration 2 – Refactor Action

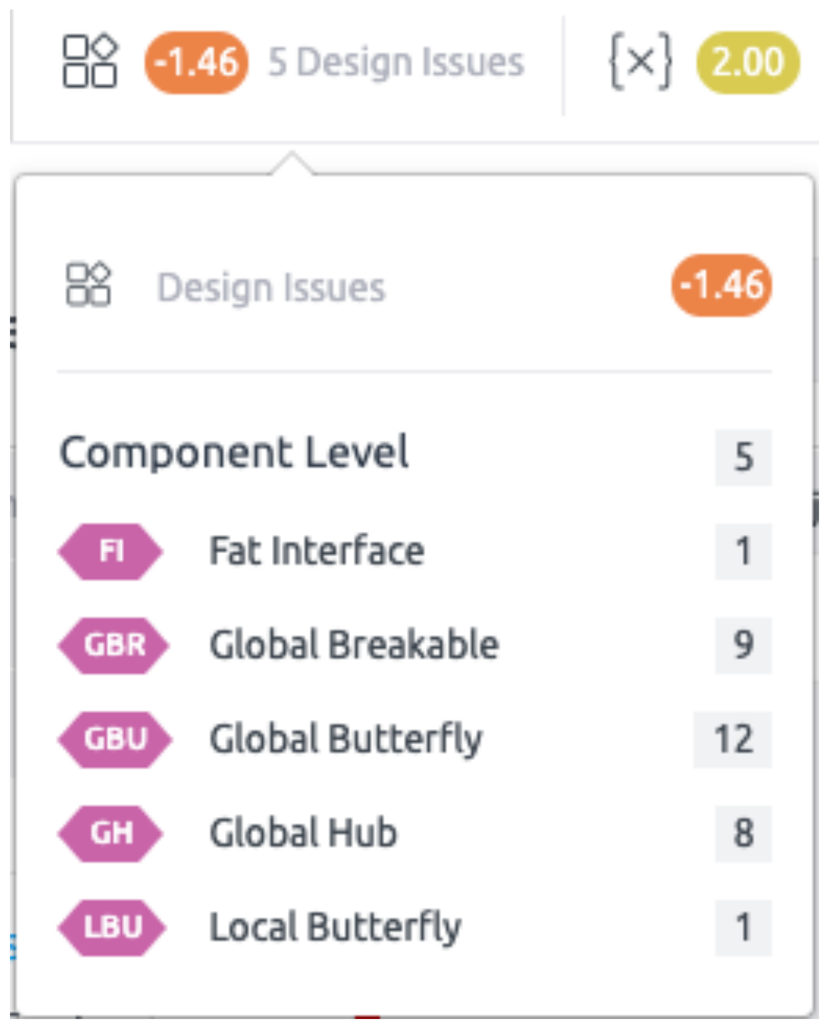
- Extract out the “Pattern” abstraction from InternalTopologyBuilder to a new Refac_TopicPatterns class
- TopicPatterns is a fairly isolated abstraction ideally represented in its own class, and is not really the concern of InternalTopologyBuilder

Iteration 2 result – Class InternalTopologyBuilder – Hotspot Removal



0.64 Overall Rating | -1.46 5 Design Issues | 2.00 14 Code Issues | -0.30 Metrics | 5.00 0 Duplication

Design - God Class Fixed!

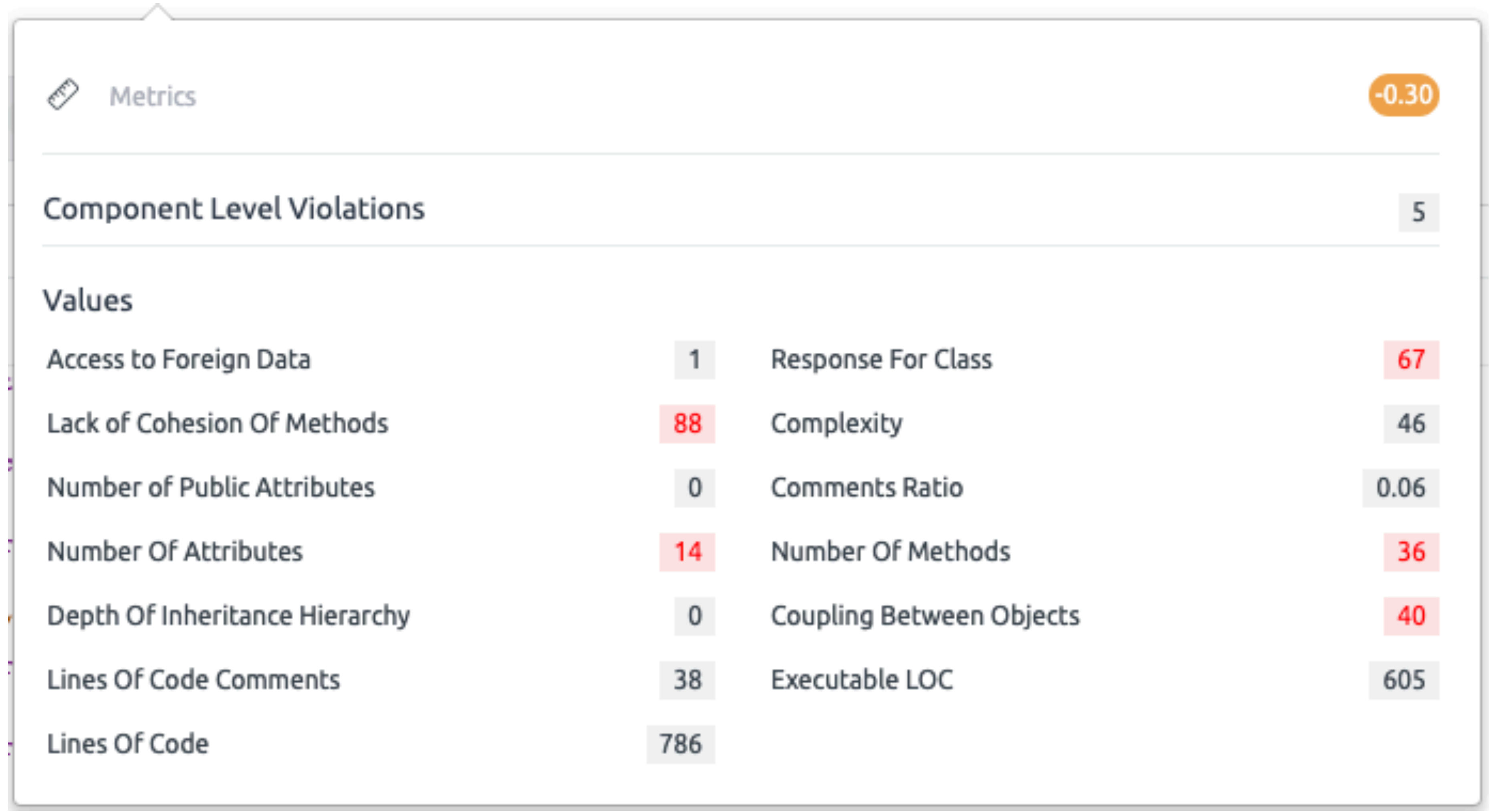


-1.46 5 Design Issues | 2.00

Design Issues -1.46

Component Level	5
FI Fat Interface	1
GBR Global Breakable	9
GBU Global Butterfly	12
GH Global Hub	8
LBU Local Butterfly	1

Further Improvement in Metrics



Metrics -0.30

Component Level Violations 5

Values

Access to Foreign Data	1	Response For Class	67
Lack of Cohesion Of Methods	88	Complexity	46
Number of Public Attributes	0	Comments Ratio	0.06
Number Of Attributes	14	Number Of Methods	36
Depth Of Inheritance Hierarchy	0	Coupling Between Objects	40
Lines Of Code Comments	38	Executable LOC	605
Lines Of Code	786		

Result

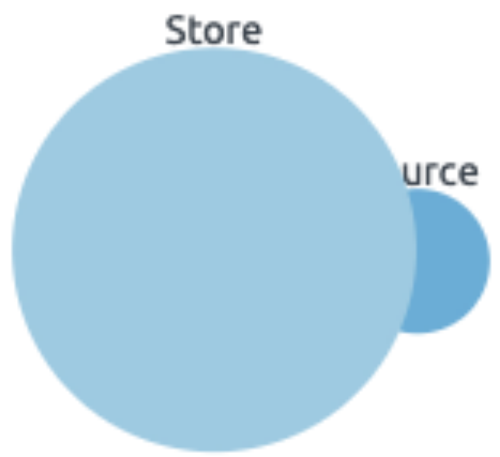
- Not a hotspot anymore: Rating changed from -0.59 to 0.64
- God Class design issue fixed
- Complexity under threshold (50)
- Improvement in other metrics (reduced coupling, number of methods, cohesion, lines of code, etc.)
- Although other design issues and metrics violation exist, the primary conditions for refactoring are met

In this Iteration, we successfully addressed the hotspot and God Class issues by fixing them via partitioning and introducing additional classes with cleaner abstractions

Iteration 2 result – Cleaner Partitions

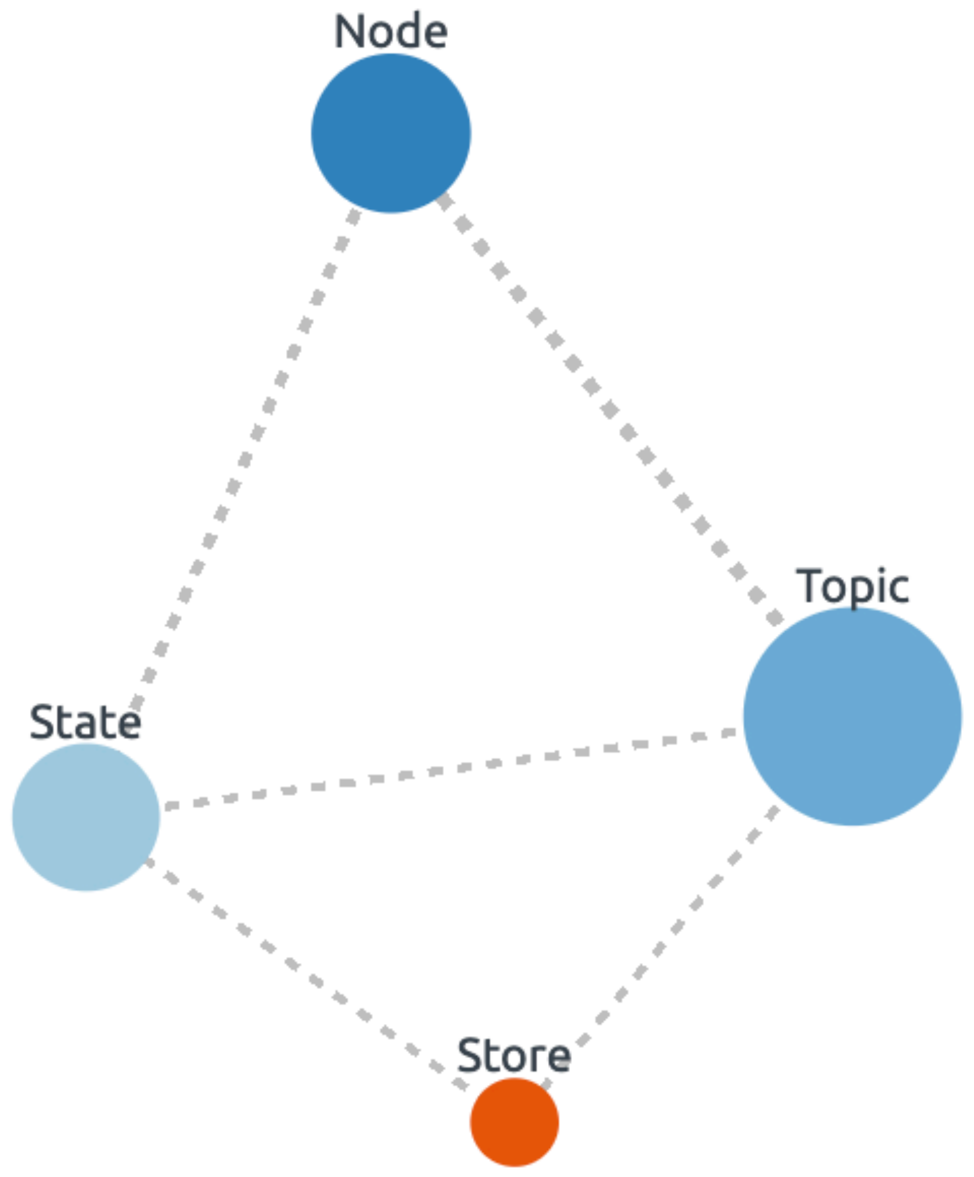
0.64 | -1.46 | 2.00 | -0.30 | 5.00

InternalTopologyBuilder



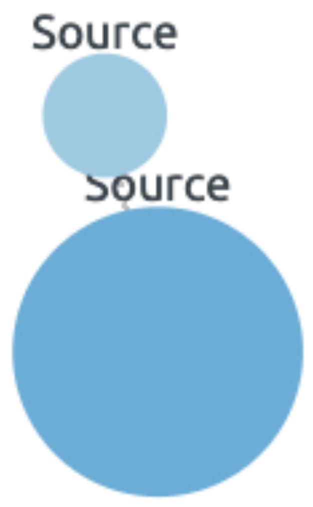
0.80 | 0.06 | 2.00 | -1.04 | 5.00

Refac_Topic



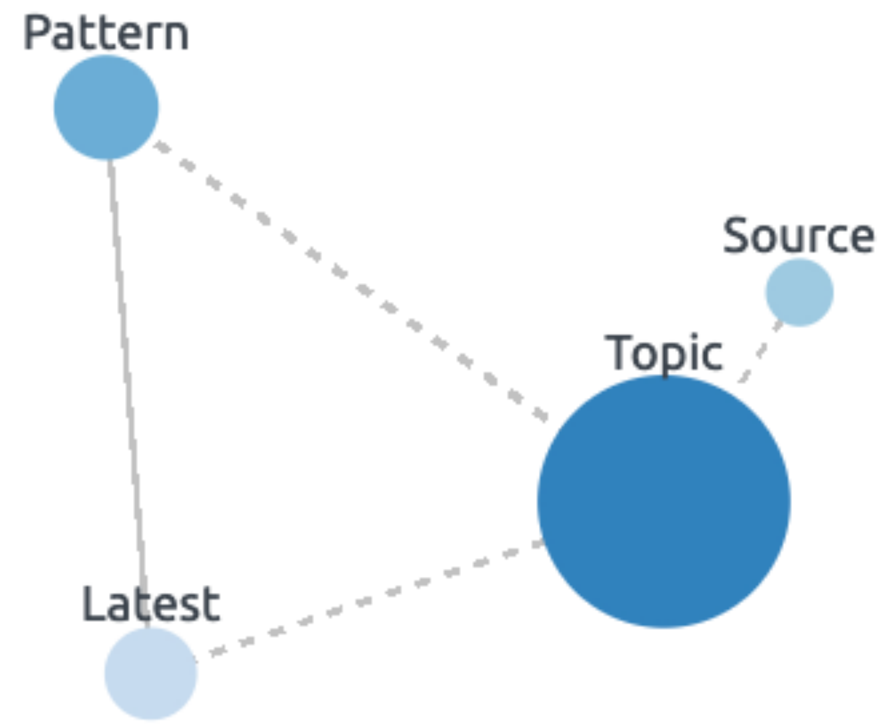
2.56 | 3.48 | 2.50 | 0.66 | 5.00

Refac_SourceSink



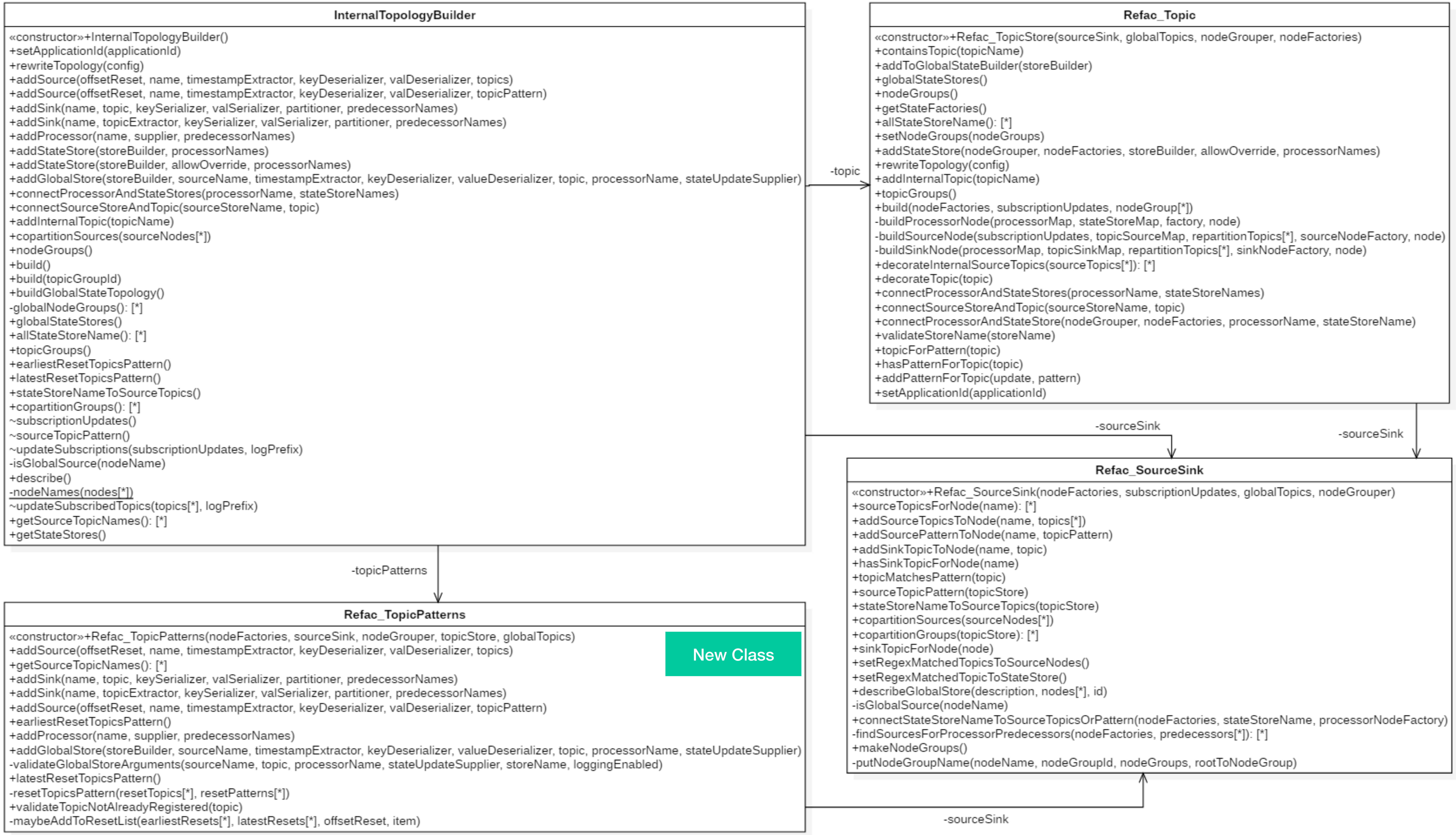
2.69 | 3.48 | 2.50 | 0.99 | 5.00

Refac_TopicPatterns



Refactoring resulted in cleaner partitions. However, in the process, we introduced another God Class: Refac_Topic, which is the subject of Iteration 3

Iteration 2 result – Cleaner Partitions



InternalTopologyBuilder

```
«constructor»+InternalTopologyBuilder()
+setApplicationId(applicationId)
+rewriteTopology(config)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topics)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topicPattern)
+addSink(name, topic, keySerializer, valSerializer, partitioner, predecessorNames)
+addSink(name, topicExtractor, keySerializer, valSerializer, partitioner, predecessorNames)
+addProcessor(name, supplier, predecessorNames)
+addStateStore(storeBuilder, processorNames)
+addStateStore(storeBuilder, allowOverride, processorNames)
+addGlobalStore(storeBuilder, sourceName, timestampExtractor, keyDeserializer, valueDeserializer, topic, processorName, stateUpdateSupplier)
+connectProcessorAndStateStores(processorName, stateStoreNames)
+connectSourceStoreAndTopic(sourceStoreName, topic)
+addInternalTopic(topicName)
+copartitionSources(sourceNodes[*])
+nodeGroups()
+build()
+build(topicGroupId)
+buildGlobalStateTopology()
-globalNodeGroups(): [*]
+globalStateStores()
+allStateStoreName(): [*]
+topicGroups()
+earliestResetTopicsPattern()
+latestResetTopicsPattern()
+stateStoreNameToSourceTopics()
+copartitionGroups(): [*]
~subscriptionUpdates()
~sourceTopicPattern()
~updateSubscriptions(subscriptionUpdates, logPrefix)
-isGlobalSource(nodeName)
+describe()
-nodeNames(nodes[*])
~updateSubscribedTopics(topics[*], logPrefix)
+getSourceTopicNames(): [*]
+getStateStores()
```

Refac_Topic

```
«constructor»+Refac_TopicStore(sourceSink, globalTopics, nodeGrouper, nodeFactories)
+containsTopic(topicName)
+addToGlobalStateBuilder(storeBuilder)
+globalStateStores()
+nodeGroups()
+getStateFactories()
+allStateStoreName(): [*]
+setNodeGroups(nodeGroups)
+addStateStore(nodeGrouper, nodeFactories, storeBuilder, allowOverride, processorNames)
+rewriteTopology(config)
+addInternalTopic(topicName)
+topicGroups()
+build(nodeFactories, subscriptionUpdates, nodeGroup[*])
-buildProcessorNode(processorMap, stateStoreMap, factory, node)
-buildSourceNode(subscriptionUpdates, topicSourceMap, repartitionTopics[*], sourceNodeFactory, node)
-buildSinkNode(processorMap, topicSinkMap, repartitionTopics[*], sinkNodeFactory, node)
+decorateInternalSourceTopics(sourceTopics[*]): [*]
+decorateTopic(topic)
+connectProcessorAndStateStores(processorName, stateStoreNames)
+connectSourceStoreAndTopic(sourceStoreName, topic)
+connectProcessorAndStateStore(nodeGrouper, nodeFactories, processorName, stateStoreName)
+validateStoreName(storeName)
+topicForPattern(topic)
+hasPatternForTopic(topic)
+addPatternForTopic(update, pattern)
+setApplicationId(applicationId)
```

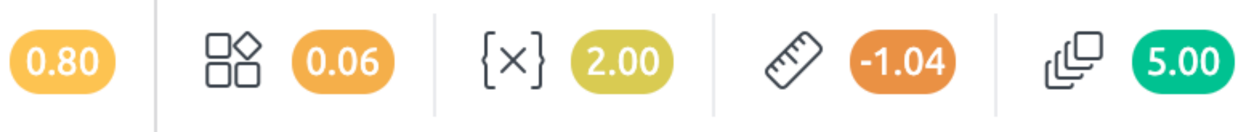
Refac_SourceSink

```
«constructor»+Refac_SourceSink(nodeFactories, subscriptionUpdates, globalTopics, nodeGrouper)
+sourceTopicsForNode(name): [*]
+addSourceTopicsToNode(name, topics[*])
+addSourcePatternToNode(name, topicPattern)
+addSinkTopicToNode(name, topic)
+hasSinkTopicForNode(name)
+topicMatchesPattern(topic)
+sourceTopicPattern(topicStore)
+stateStoreNameToSourceTopics(topicStore)
+copartitionSources(sourceNodes[*])
+copartitionGroups(topicStore): [*]
+sinkTopicForNode(node)
+setRegexMatchedTopicsToSourceNodes()
+setRegexMatchedTopicToStateStore()
+describeGlobalStore(description, nodes[*], id)
-isGlobalSource(nodeName)
+connectStateStoreNameToSourceTopicsOrPattern(nodeFactories, stateStoreName, processorNodeFactory)
-findSourcesForProcessorPredecessors(nodeFactories, predecessors[*]): [*]
+makeNodeGroups()
-putNodeGroupName(nodeName, nodeGroupId, nodeGroups, rootToNodeGroup)
```

Refac_TopicPatterns

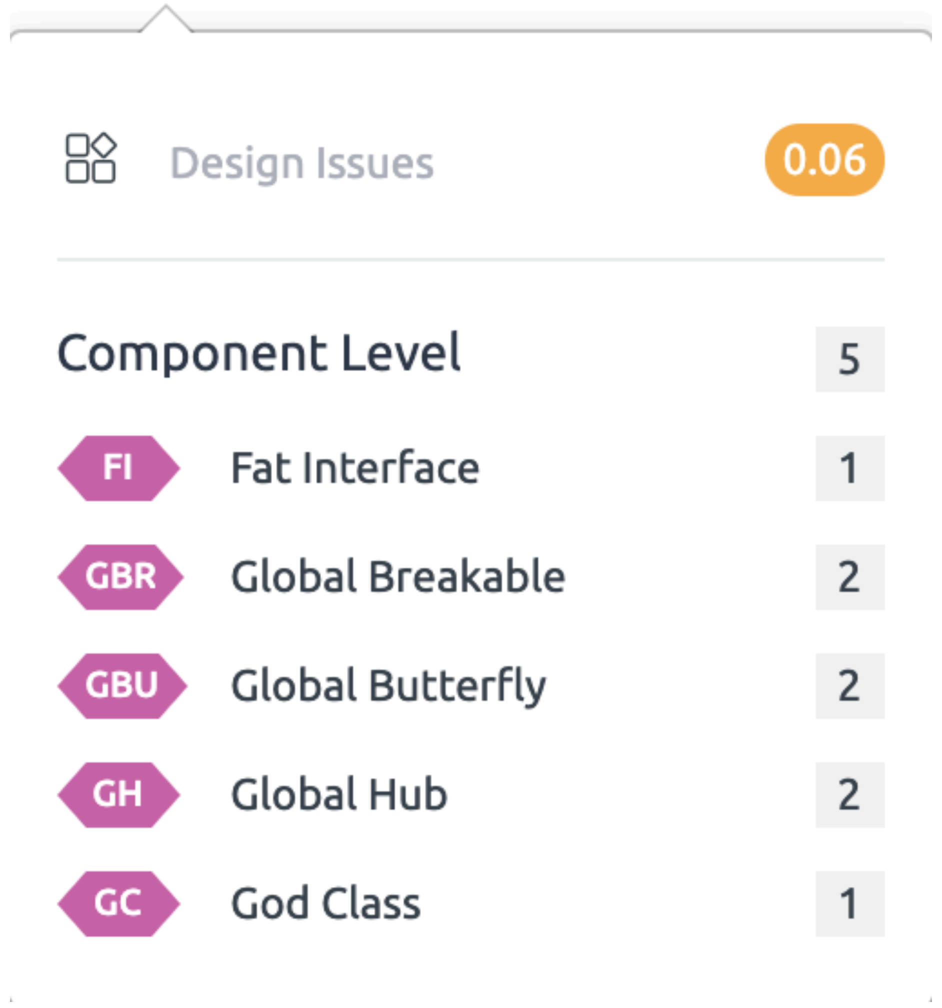
```
«constructor»+Refac_TopicPatterns(nodeFactories, sourceSink, nodeGrouper, topicStore, globalTopics)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topics)
+getSourceTopicNames(): [*]
+addSink(name, topic, keySerializer, valSerializer, partitioner, predecessorNames)
+addSink(name, topicExtractor, keySerializer, valSerializer, partitioner, predecessorNames)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topicPattern)
+earliestResetTopicsPattern()
+addProcessor(name, supplier, predecessorNames)
+addGlobalStore(storeBuilder, sourceName, timestampExtractor, keyDeserializer, valueDeserializer, topic, processorName, stateUpdateSupplier)
-validateGlobalStoreArguments(sourceName, topic, processorName, stateUpdateSupplier, storeName, loggingEnabled)
+latestResetTopicsPattern()
-resetTopicsPattern(resetTopics[*], resetPatterns[*])
+validateTopicNotAlreadyRegistered(topic)
-maybeAddToResetList(earliestResets[*], latestResets[*], offsetReset, item)
```

Iteration 2 result – New Class Refac_Topic



0.80	0.06	2.00	-1.04	5.00

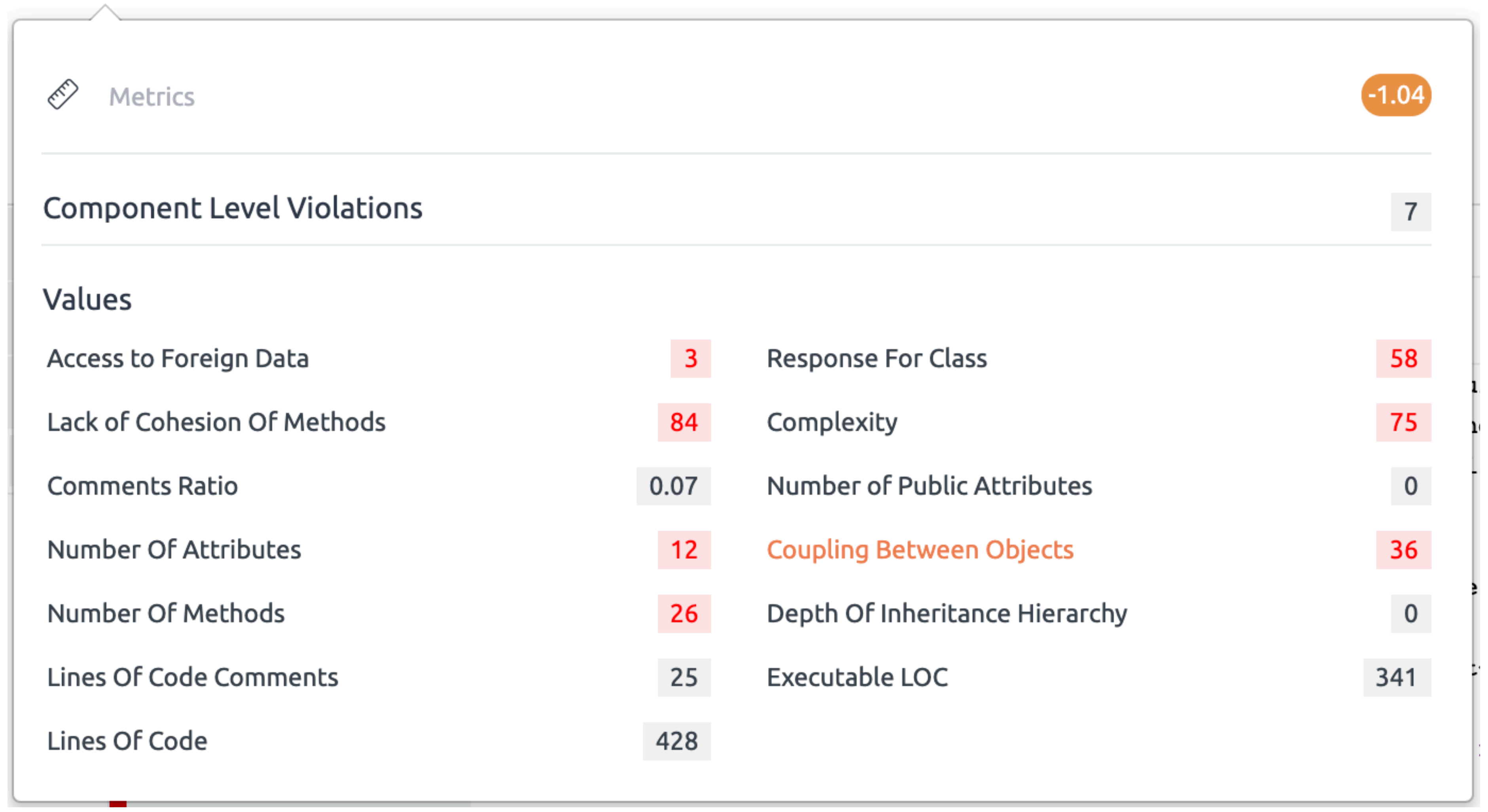
New God Class Introduced – Needs to be Fixed!



Design Issues 0.06

Component Level	5
FI Fat Interface	1
GBR Global Breakable	2
GBU Global Butterfly	2
GH Global Hub	2
GC God Class	1

Some metrics violations – needs improvement



Metrics -1.04

Component Level Violations 7

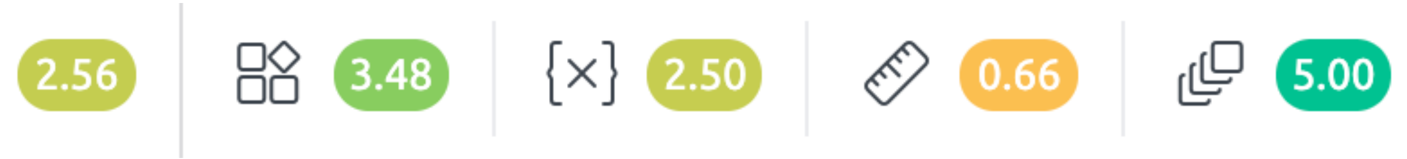
Values

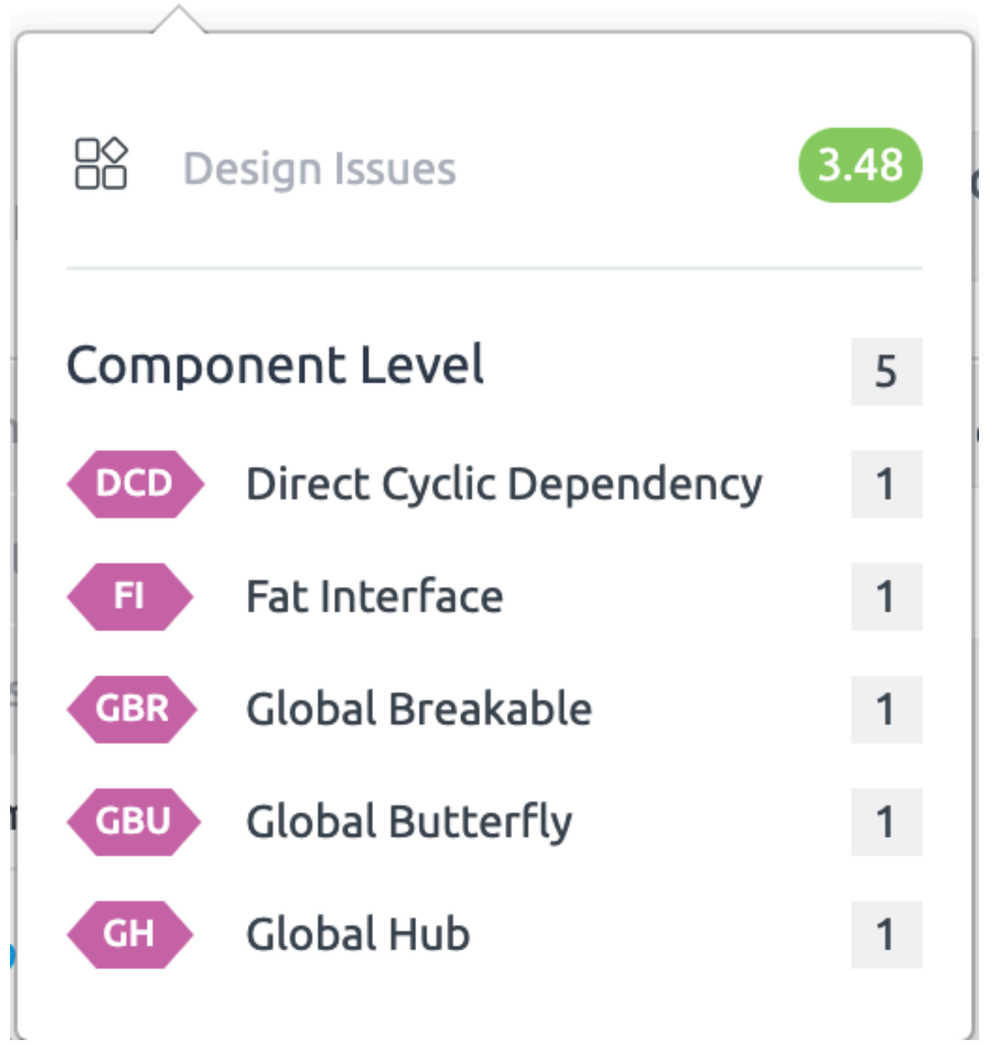
Access to Foreign Data	3	Response For Class	58
Lack of Cohesion Of Methods	84	Complexity	75
Comments Ratio	0.07	Number of Public Attributes	0
Number Of Attributes	12	Coupling Between Objects	36
Number Of Methods	26	Depth Of Inheritance Hierarchy	0
Lines Of Code Comments	25	Executable LOC	341
Lines Of Code	428		

- Detected as God Class, although not a hotspot (overall rating > 0)
- Some metrics violated (CBO, LCOM, Complexity)

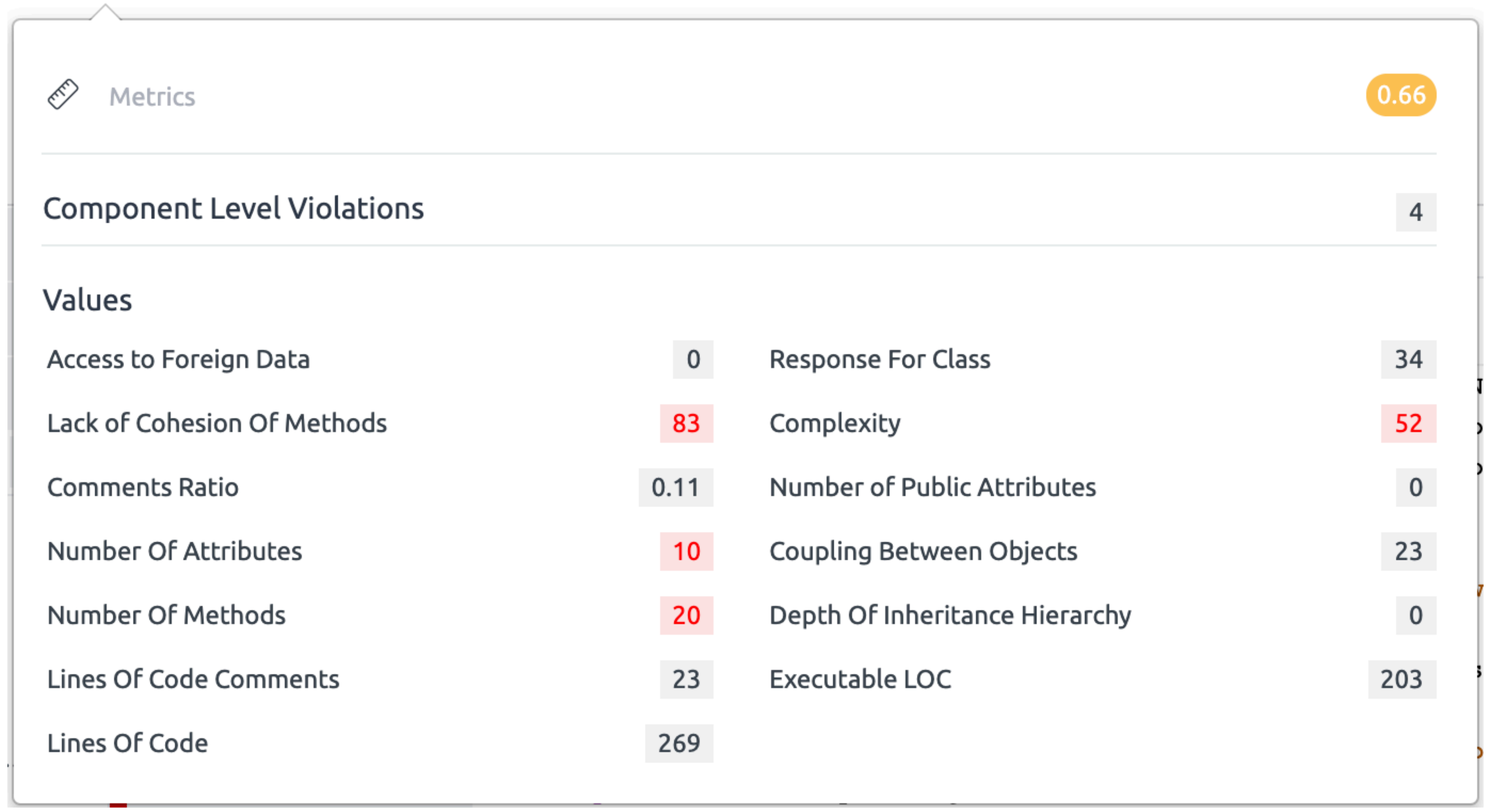
New God Class introduced – needs to be fixed, so run Partitioning on this class

Iteration 2 result – New Class Refac_SourceSink




Design Issues 3.48

Component Level	5
DCD Direct Cyclic Dependency	1
FI Fat Interface	1
GBR Global Breakable	1
GBU Global Butterfly	1
GH Global Hub	1


Metrics 0.66

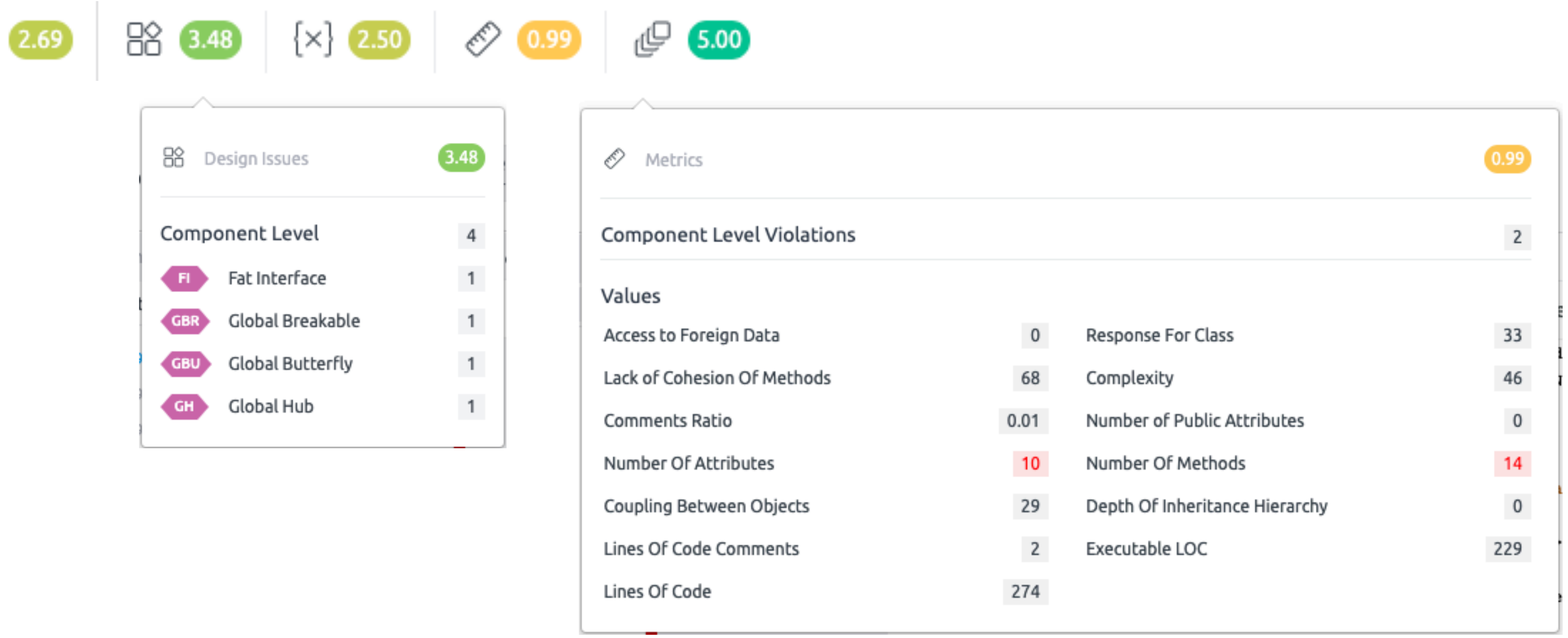
Component Level Violations 4

Values

Access to Foreign Data	0	Response For Class	34
Lack of Cohesion Of Methods	83	Complexity	52
Comments Ratio	0.11	Number of Public Attributes	0
Number Of Attributes	10	Coupling Between Objects	23
Number Of Methods	20	Depth Of Inheritance Hierarchy	0
Lines Of Code Comments	23	Executable LOC	203
Lines Of Code	269		

- New class looks ok, although still has some lack of cohesion, but under threshold (77)
- Cyclic dependency should be removed (part of next refactoring – Iteration 3)

Iteration 2 result – New Class Refac_TopicPatterns



Design Issues 3.48

Component Level	4
FI Fat Interface	1
GBR Global Breakable	1
GBU Global Butterfly	1
GH Global Hub	1

Metrics 0.99

Component Level Violations 2

Values

Access to Foreign Data	0	Response For Class	33
Lack of Cohesion Of Methods	68	Complexity	46
Comments Ratio	0.01	Number of Public Attributes	0
Number Of Attributes	10	Number Of Methods	14
Coupling Between Objects	29	Depth Of Inheritance Hierarchy	0
Lines Of Code Comments	2	Executable LOC	229
Lines Of Code	274		

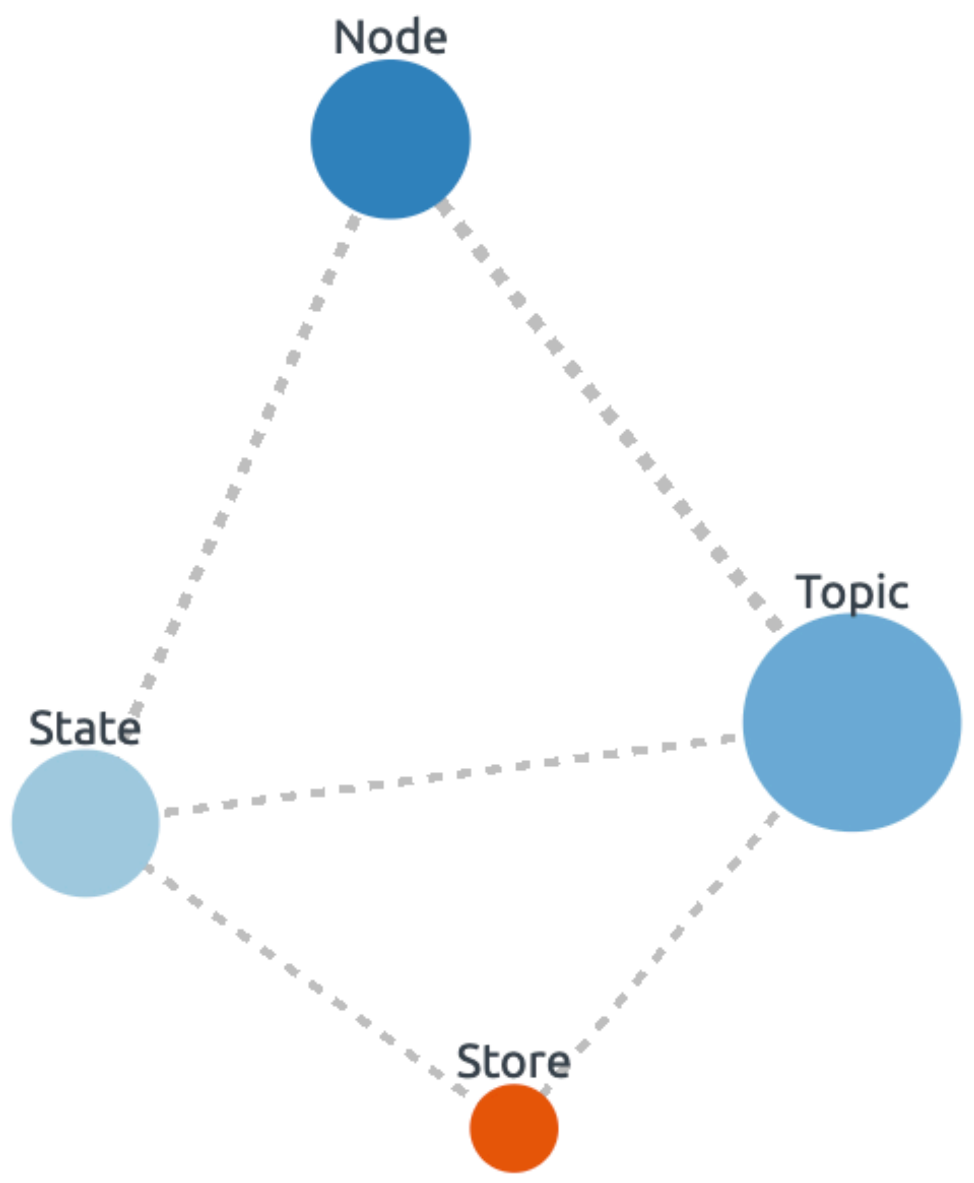
- Design issues are related to dependencies (we are not changing the public interface of the original class, so dependency-related design issues will not be refactored)
- Looks good with very few metrics violations

Iteration 3 – Action :: Fix new God Class: Refac_Topic

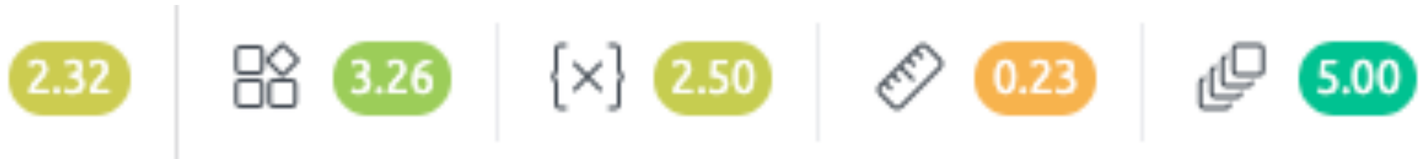
Strategy

- Class InternalTopologyBuilder (the original target) has reached the expected outcome – No hotspot, No God Class, Reasonable metrics values
- The newly introduced class Refac_Topic, though, was detected as a God Class, although not a hotspot
- Next step is to introduce additional refactoring of Refac_Topic to better represent its abstractions and remove the God Class design issue
- Refactor Action: Extract node builder functionality from Refac_Topic to a separate Refac_NodeBuilder class, as this abstraction is not related directly to Topic

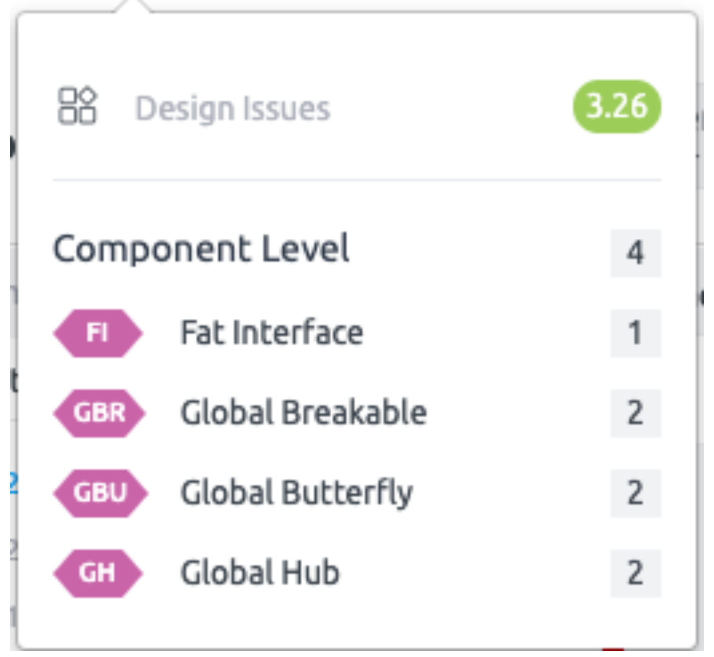
Partitions: Refac_Topic



Iteration 3 result – Refactored Class Refac_Topic



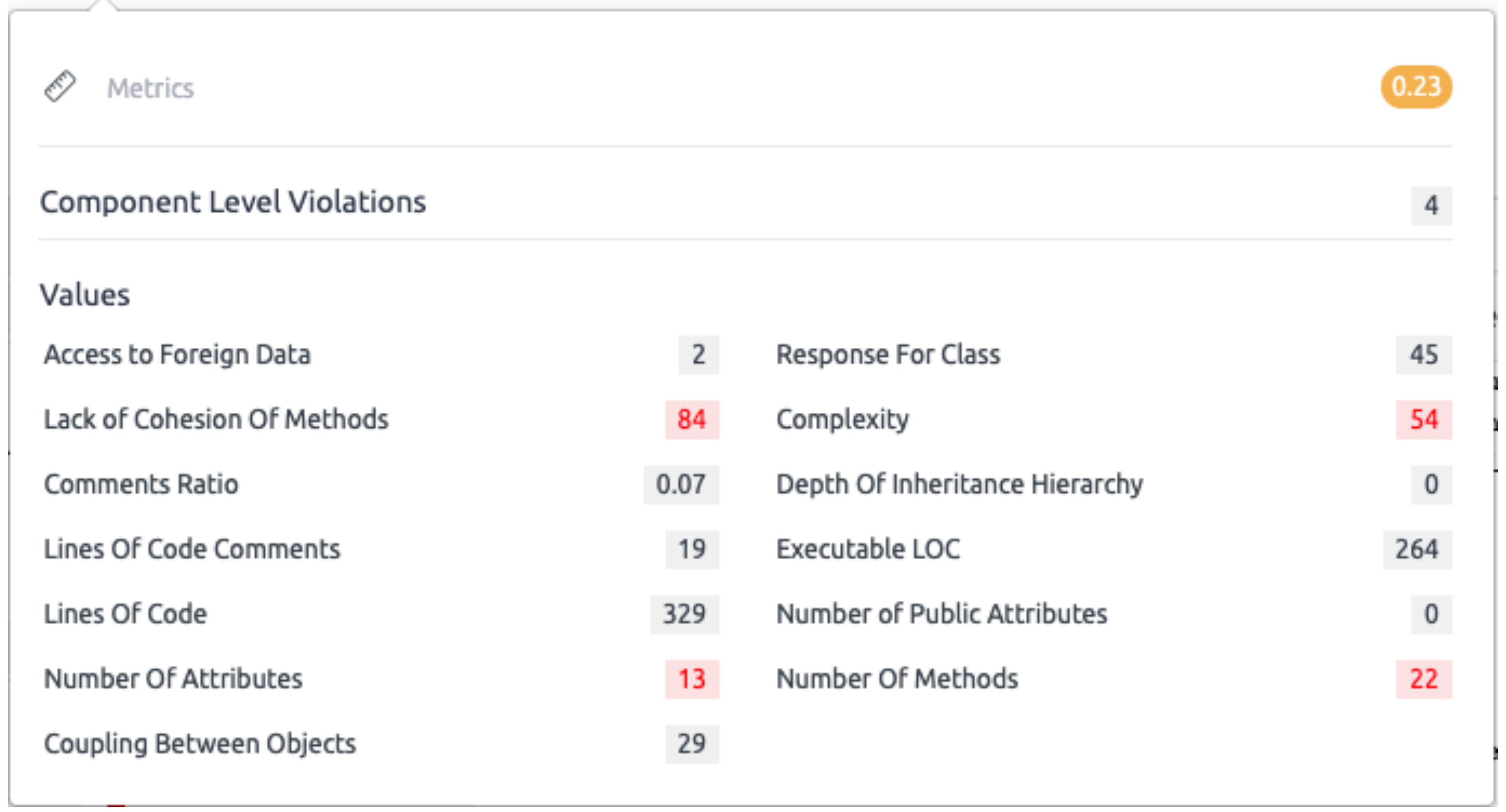
Design - God Class Fixed!



Design Issues 3.26

Component Level	4
FI Fat Interface	1
GBR Global Breakable	2
GBU Global Butterfly	2
GH Global Hub	2

Improvement in Metrics



Metrics 0.23

Component Level Violations 4

Values

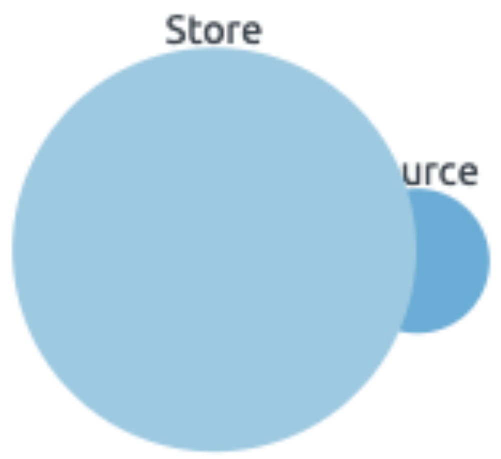
Access to Foreign Data	2	Response For Class	45
Lack of Cohesion Of Methods	84	Complexity	54
Comments Ratio	0.07	Depth Of Inheritance Hierarchy	0
Lines Of Code Comments	19	Executable LOC	264
Lines Of Code	329	Number of Public Attributes	0
Number Of Attributes	13	Number Of Methods	22
Coupling Between Objects	29		

- Class looks good, although still has some lack of cohesion (threshold is 77)
- No further refactoring needed

Iteration 3 result – Cleaner Partitions

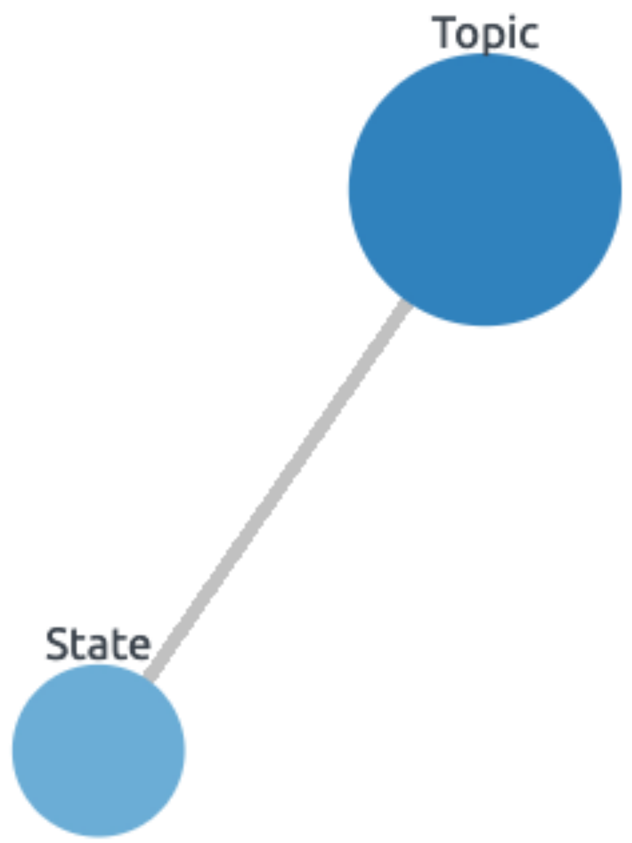
0.63 | -1.46 | {x} 2.00 | -0.34 | 5.00

InternalTopologyBuilder



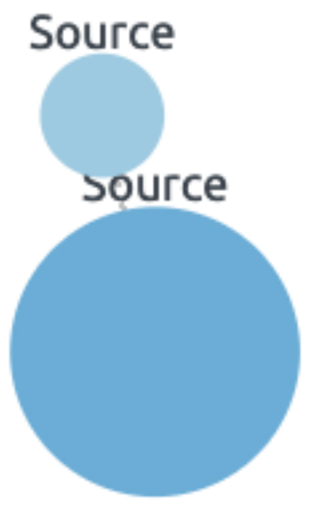
2.32 | 3.26 | {x} 2.50 | 0.23 | 5.00

Refac_Topic



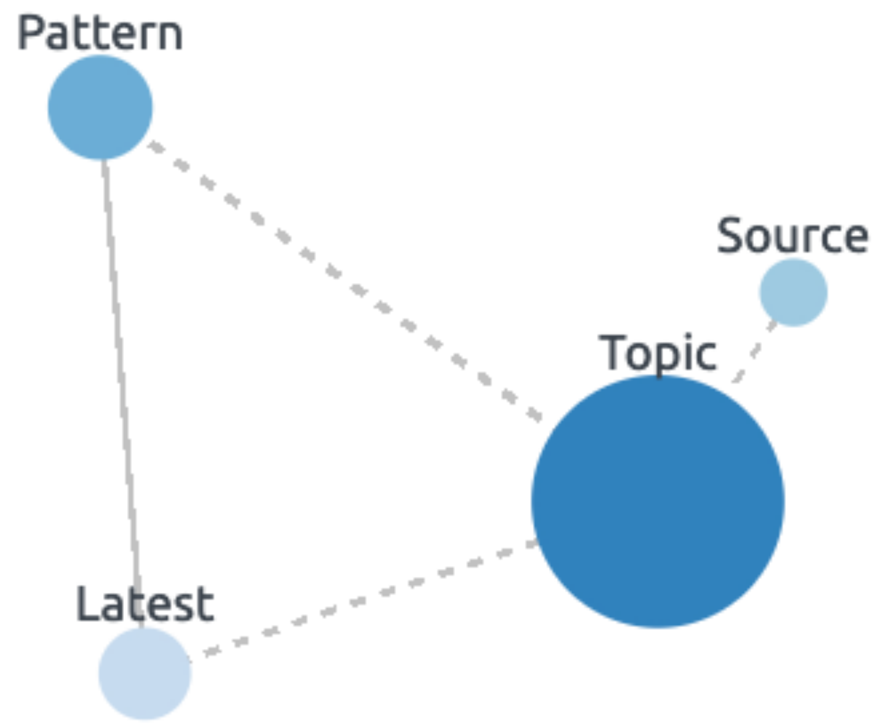
2.81 | 3.48 | {x} 5.00 | 0.66 | 5.00

Refac_SourceSink



2.91 | 3.42 | {x} 5.00 | 0.95 | 5.00

Refac_TopicPatterns

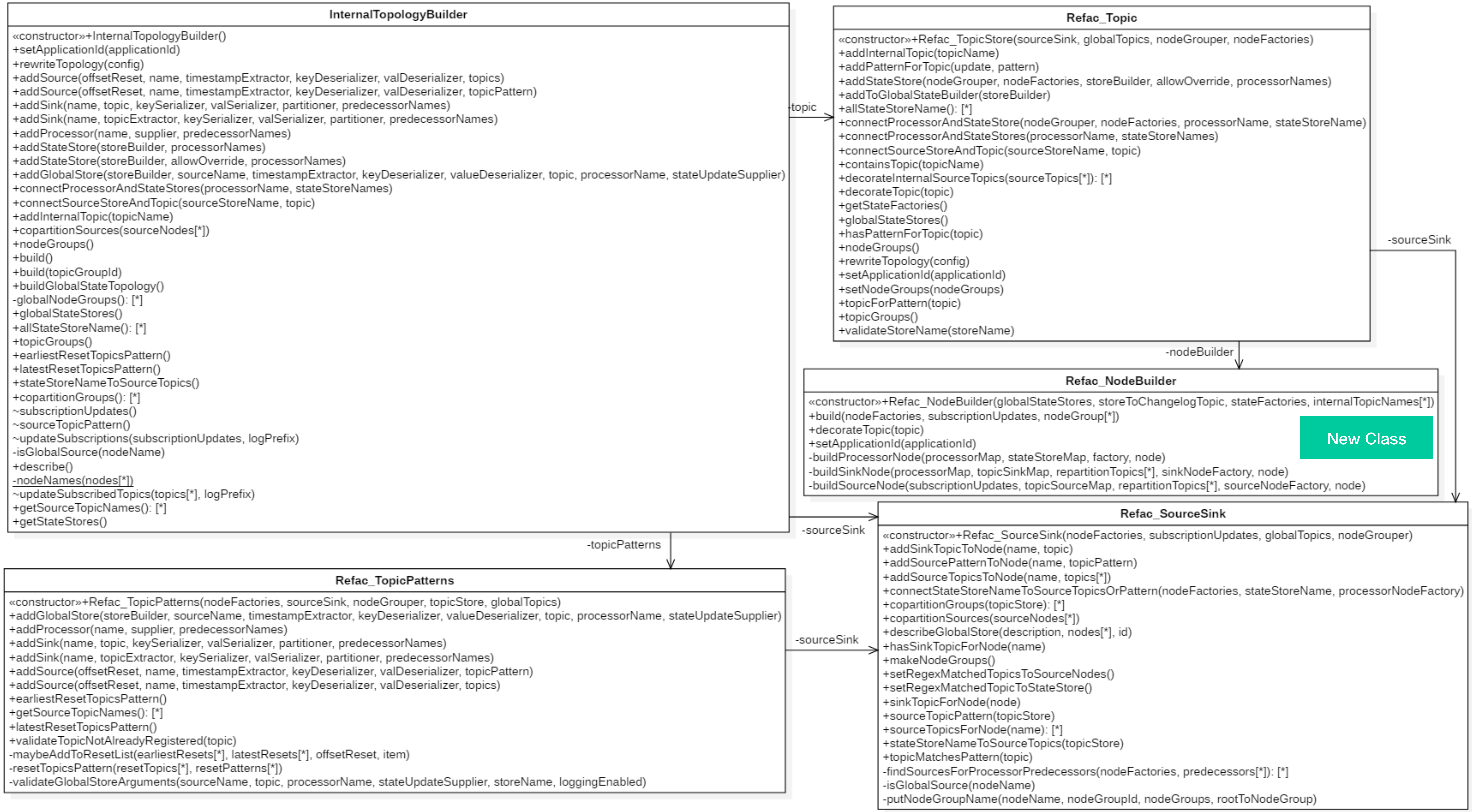


- In this final iteration, we refactored Refac_Topic to extract Refac_NodeBuilder out of it
- With this we successfully refactored the original InternalTopologyBuilder to smaller abstractions where each of the new abstractions are not hotspots, not God classes and represent meaningful abstractions
- The original class is also simplified, not a hotspot anymore and not a God Class anymore

3.66 | 3.64 | {x} 5.00 | 2.66 | 5.00

Refac_NodeBuilder

Iteration 3 result – Cleaner Partitions



InternalTopologyBuilder

«constructor»+InternalTopologyBuilder()
+setApplicationId(applicationId)
+rewriteTopology(config)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topics)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topicPattern)
+addSink(name, topic, keySerializer, valSerializer, partitioner, predecessorNames)
+addSink(name, topicExtractor, keySerializer, valSerializer, partitioner, predecessorNames)
+addProcessor(name, supplier, predecessorNames)
+addStateStore(storeBuilder, processorNames)
+addStateStore(storeBuilder, allowOverride, processorNames)
+addGlobalStore(storeBuilder, sourceName, timestampExtractor, keyDeserializer, valueDeserializer, topic, processorName, stateUpdateSupplier)
+connectProcessorAndStateStores(processorName, stateStoreNames)
+connectSourceStoreAndTopic(sourceStoreName, topic)
+addInternalTopic(topicName)
+copartitionSources(sourceNodes[*])
+nodeGroups()
+build()
+build(topicGroupId)
+buildGlobalStateTopology()
-globalNodeGroups(): [*]
+globalStateStores()
+allStateStoreName(): [*]
+topicGroups()
+earliestResetTopicsPattern()
+latestResetTopicsPattern()
+stateStoreNameToSourceTopics()
+copartitionGroups(): [*]
~subscriptionUpdates()
~sourceTopicPattern()
~updateSubscriptions(subscriptionUpdates, logPrefix)
-isGlobalSource(nodeName)
+describe()
-nodeNames(nodes[*])
~updateSubscribedTopics(topics[*], logPrefix)
+getSourceTopicNames(): [*]
+getStateStores()

-topic

Refac_Topic

«constructor»+Refac_TopicStore(sourceSink, globalTopics, nodeGrouper, nodeFactories)
+addInternalTopic(topicName)
+addPatternForTopic(update, pattern)
+addStateStore(nodeGrouper, nodeFactories, storeBuilder, allowOverride, processorNames)
+addToGlobalStateBuilder(storeBuilder)
+allStateStoreName(): [*]
+connectProcessorAndStateStore(nodeGrouper, nodeFactories, processorName, stateStoreName)
+connectProcessorAndStateStores(processorName, stateStoreNames)
+connectSourceStoreAndTopic(sourceStoreName, topic)
+containsTopic(topicName)
+decorateInternalSourceTopics(sourceTopics[*]): [*]
+decorateTopic(topic)
+getStateFactories()
+globalStateStores()
+hasPatternForTopic(topic)
+nodeGroups()
+rewriteTopology(config)
+setApplicationId(applicationId)
+setNodeGroups(nodeGroups)
+topicForPattern(topic)
+topicGroups()
+validateStoreName(storeName)

-sourceSink

-nodeBuilder

Refac_NodeBuilder

«constructor»+Refac_NodeBuilder(globalStateStores, storeToChangelogTopic, stateFactories, internalTopicNames[*])
+build(nodeFactories, subscriptionUpdates, nodeGroup[*])
+decorateTopic(topic)
+setApplicationId(applicationId)
-buildProcessorNode(processorMap, stateStoreMap, factory, node)
-buildSinkNode(processorMap, topicSinkMap, repartitionTopics[*], sinkNodeFactory, node)
-buildSourceNode(subscriptionUpdates, topicSourceMap, repartitionTopics[*], sourceNodeFactory, node)

New Class

-sourceSink

-topicPatterns

Refac_SourceSink

«constructor»+Refac_SourceSink(nodeFactories, subscriptionUpdates, globalTopics, nodeGrouper)
+addSinkTopicToNode(name, topic)
+addSourcePatternToNode(name, topicPattern)
+addSourceTopicsToNode(name, topics[*])
+connectStateStoreNameToSourceTopicsOrPattern(nodeFactories, stateStoreName, processorNodeFactory)
+copartitionGroups(topicStore): [*]
+copartitionSources(sourceNodes[*])
+describeGlobalStore(description, nodes[*], id)
+hasSinkTopicForNode(name)
+makeNodeGroups()
+setRegexMatchedTopicsToSourceNodes()
+setRegexMatchedTopicToStateStore()
+sinkTopicForNode(node)
+sourceTopicPattern(topicStore)
+sourceTopicsForNode(name): [*]
+stateStoreNameToSourceTopics(topicStore)
+topicMatchesPattern(topic)
-findSourcesForProcessorPredecessors(nodeFactories, predecessors[*]): [*]
-isGlobalSource(nodeName)
-putNodeGroupName(nodeName, nodeGroupId, nodeGroups, rootToNodeGroup)

Refac_TopicPatterns

«constructor»+Refac_TopicPatterns(nodeFactories, sourceSink, nodeGrouper, topicStore, globalTopics)
+addGlobalStore(storeBuilder, sourceName, timestampExtractor, keyDeserializer, valueDeserializer, topic, processorName, stateUpdateSupplier)
+addProcessor(name, supplier, predecessorNames)
+addSink(name, topic, keySerializer, valSerializer, partitioner, predecessorNames)
+addSink(name, topicExtractor, keySerializer, valSerializer, partitioner, predecessorNames)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topicPattern)
+addSource(offsetReset, name, timestampExtractor, keyDeserializer, valDeserializer, topics)
+earliestResetTopicsPattern()
+getSourceTopicNames(): [*]
+latestResetTopicsPattern()
+validateTopicNotAlreadyRegistered(topic)
-maybeAddToResetList(earliestResets[*], latestResets[*], offsetReset, item)
-resetTopicsPattern(resetTopics[*], resetPatterns[*])
-validateGlobalStoreArguments(sourceName, topic, processorName, stateUpdateSupplier, storeName, loggingEnabled)

-sourceSink

Summary

Before

Original Class	Overall Rating	Design Rating	Metrics Rating	Code Quality Rating	ELOC	NOM	Complexity	CBO	RFC
InternalTopologyBuilder	-1.11	-2.18	-3.92	1.09	1452	59	191	55	105

After

Refactored Classes	Overall Rating	Design Rating	Metrics Rating	Code Quality Rating	ELOC	NOM	Complexity	CBO	RFC
InternalTopologyBuilder	0.63	-1.46	-0.34	2.0	605	36	46	40	71
Refac_Topic	2.32	3.26	0.23	2.50	264	22	54	29	45
Refac_SourceSink	2.81	3.48	0.66	5.0	203	20	52	23	34
Refac_TopicPatterns	2.91	3.42	0.95	5.0	229	14	46	29	37
Refac_NodeBuilder	3.66	3.64	2.66	5.0	105	7	24	22	22
Refac_GlobalTopics	4.78	4.76	4.63	5.0	9	3	3	3	3
Refac_TopicHelper	4.28	3.62	4.23	5.0	27	3	6	8	9
Refac_TopologyDescriptionGen	3.77	3.64	3.36	5.0	64	6	15	16	16

- We eliminated the hotspot InternalTopologyBuilder through successive refactoring with the help of Gamma's Partitioning Tool
- The resulting classes have no hotspots or God Classes, which are strongly correlated with bugs
- In the process we also created more meaningful abstractions which represent a single concept, and are hence easier to understand and maintain for new developers
- Future change is now more localized
- The resulting classes have lower complexity, lines of code, coupling and RFC, and are overall more robust towards change

Summary

- In this example we saw how Gamma’s Partitioning Tool is useful in design refactoring to eliminate anti-patterns which correlate with bugs (e.g. God Class)
- The refactoring exercise was targeted towards improving the internal structure of InternalTopologyBuilder by creating meaningful abstractions, guided by the Partitioning tool
- Further improvement is possible (beyond the scope of this exercise) by addressing the large public interface of the original class – this is a fat interface, and hence has many incoming dependencies due to multiple represented concerns (design issues: Global Butterfly and Local Butterfly)
- Refactoring the public interface will result in distribution of incoming dependencies to other (more relevant) classes, and avoid frequent changes to InternalTopologyBuilder

acellere