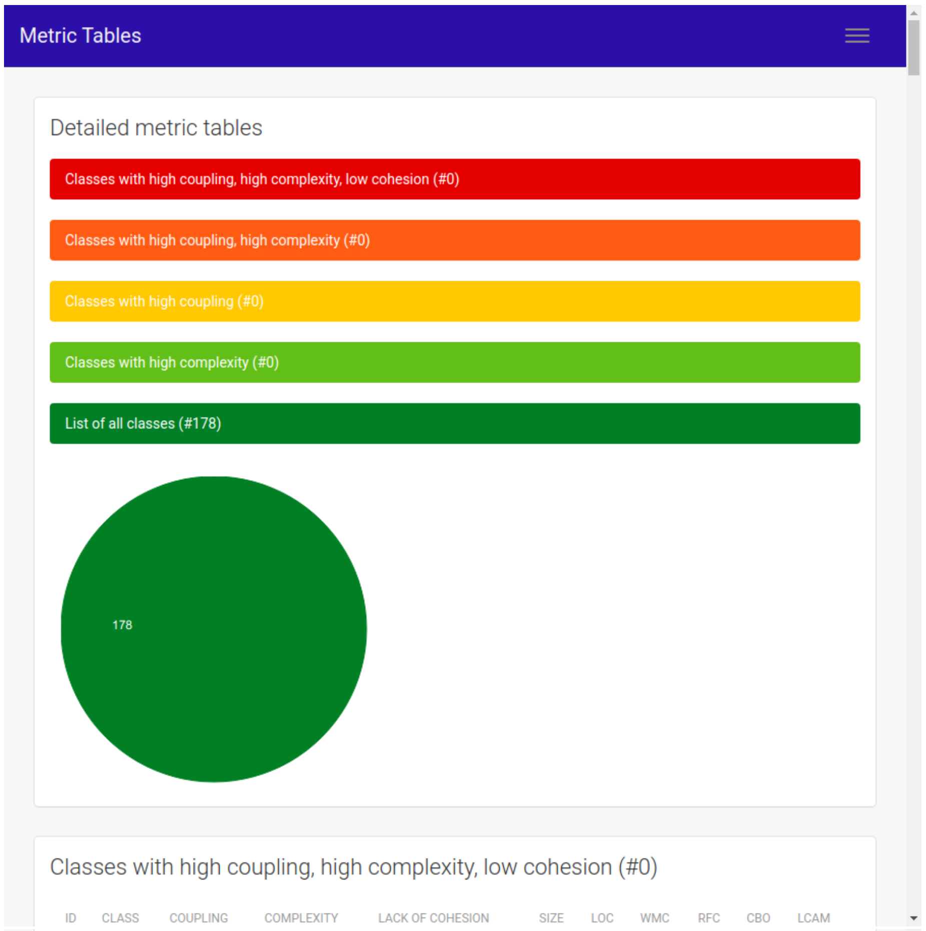
We used CodeMR and designite tools for the code metrics

Information

This project contains

- 6247 lines of code
- 135 classes
- 28 packages

Analysis


Metric Tables

Detailed metric tables

Classes with high coupling, high complexity, low cohesion (#0)

Classes with high coupling, high complexity (#0)

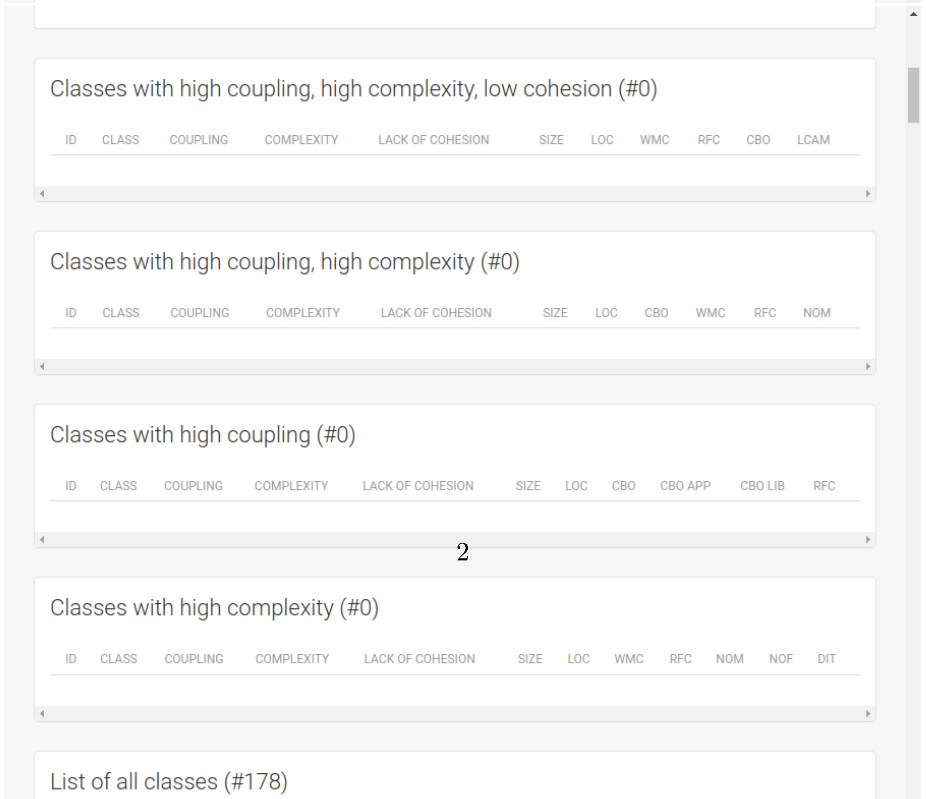
Classes with high coupling (#0)

Classes with high complexity (#0)

List of all classes (#178)

Classes with high coupling, high complexity, low cohesion (#0)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	WMC	RFC	CBO	LCAM


Classes with high coupling, high complexity, low cohesion (#0)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	WMC	RFC	CBO	LCAM

Classes with high coupling, high complexity (#0)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	CBO	WMC	RFC	NOM

Classes with high coupling (#0)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	CBO	CBO APP	CBO LIB	RFC

Classes with high complexity (#0)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	WMC	RFC	NOM	NOF	DIT

List of all classes (#178)

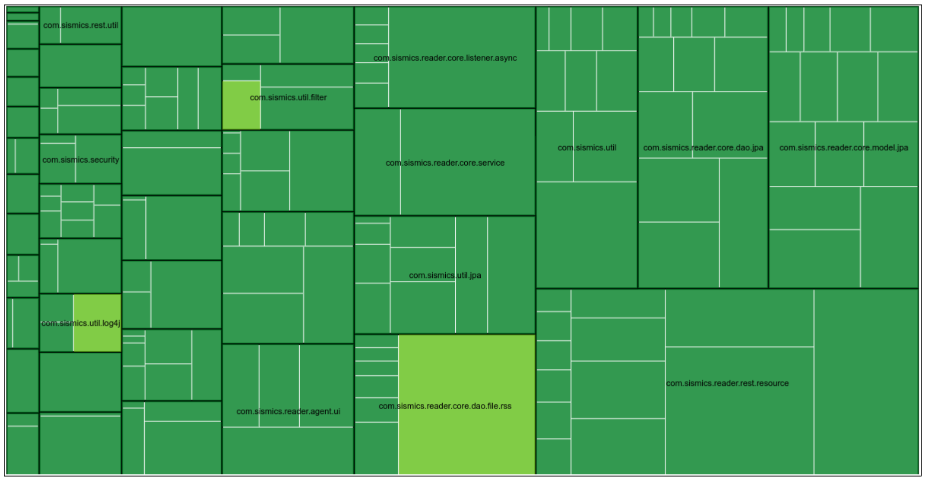
Package-wise analysis

After refactoring, the coupling and complexity reduced.

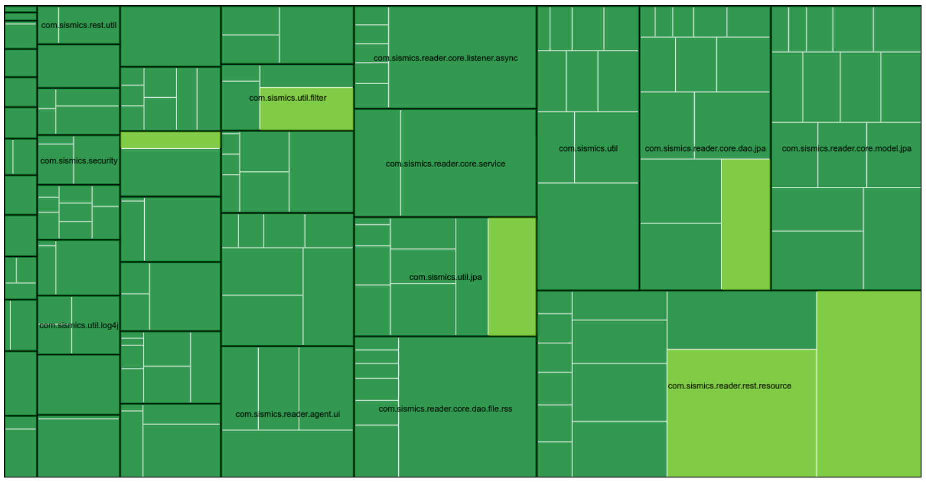
Name	Complexity	Coupling	Size	Lack of Cohesion
com.sismics.reader.core.dao.file.rss	low-medium	low	low-medium	low
com.sismics.reader.core.dao.jpa	low	**medium**	medium-high	low
com.sismics.reader.core.dao.jpa.mapper	low	low-medium	low-medium	low
com.sismics.reader.core.event	low	low-medium	low-medium	low
com.sismics.reader.core.listener.async	low	low-medium	low-medium	low
com.sismics.reader.core.model.jpa	low-medium	**medium**	medium-high	low
com.sismics.reader.core.service	low	low	low-medium	low
com.sismics.reader.core.util	low	low-medium	low-medium	low
com.sismics.util	low	low-medium	low-medium	low
com.sismics.util.jpa	low	low-medium	low-medium	low

Treemaps

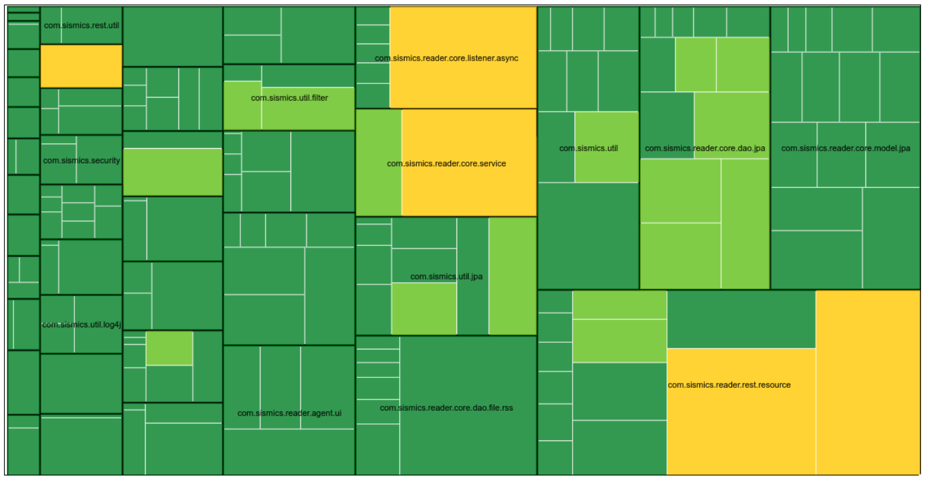
Cohesion



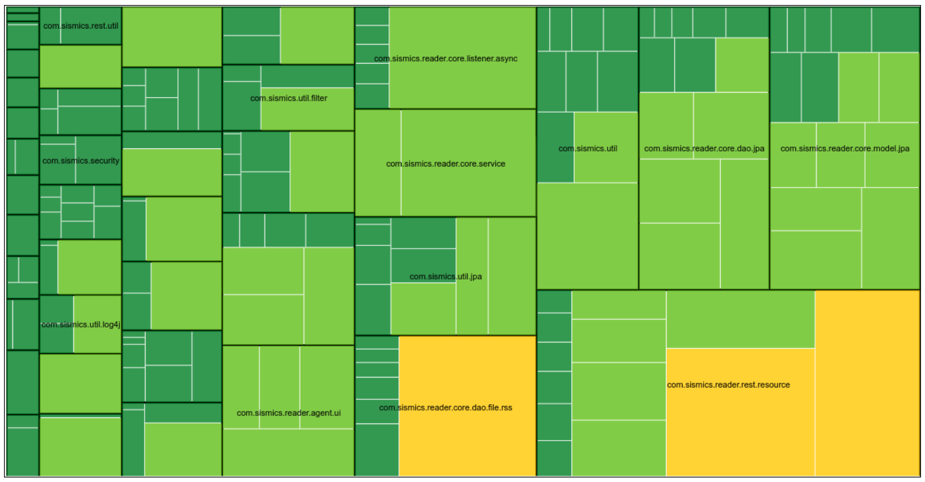
Complexity



Coupling



Size



CodeMR on the files modified heavily

Name	CBO	DIT	NOC	WMC	LOC	LCOM	LTCC
RssReader	5	0	0	16	362	1.075	1.0
ArticleDao	7	0	0	1	88	0.0	0.0
SubscriptionImportAsyncListener	16	0	0	16	277	0.0	0.0
AppContext	13	0	0	17	66	0.893	1.0
FeedService	12	1	0	13	268	0.0	0.0

- CBO has decreased in most classes. This indicates that the classes are now less dependent on other classes.
- The lack of cohesion in RssReader has increased as indicated by both LCOM and LTCC.
- WMCC has decreased significantly in all classes. This indicates that the classes are now more manageable and easier to maintain.
- The DIT has also decreased in all classes.
- LOC however has increased in all classes. This is due to the addition of new methods and refactoring of the existing code.

Codalyze analysis on these files

RssReader

Function Name	Start Line	End Line	Cyclomatic Complexity (Threshold: 10)
RssReader::RssReader	198	202	1
RssReader::readRssFeed	209	246	6
RssReader::initializeElementHandlers	249	284	1
RssReader::startElement	286	298	4

Function Name	Start Line	End Line	Cyclomatic Complexity (Threshold: 10)
RssReader::isRootElement	300	302	3
RssReader::handleRss	304	308	1
RssReader::handleFeed	310	316	1
RssReader::handleRdf	318	322	1
RssReader::handleItem	324	335	3
RssReader::handleEntry	337	348	2
RssReader::handleLink	350	359	2
RssReader::handleComments	361	367	2
RssReader::handleCreator	369	373	2
RssReader::handleDate	375	379	3
RssReader::handleContentEncoded	381	385	2
RssReader::handleContent	387	393	2
RssReader::handleEnclosure	395	412	4
RssReader::initializeEndElementHandlers	416	435	1
RssReader::endElement	440	451	2
RssReader::handleTitle	452	459	3
RssReader::handleLink	461	468	3
RssReader::handleDescription	470	475	2
RssReader::handleLanguage	477	482	2
RssReader::handleGuid	484	489	2
RssReader::handleComments	491	502	6
RssReader::handleItemDescription	504	509	3
RssReader::handleCreator	511	516	3
RssReader::handleDate	518	523	3
RssReader::handlePubDate	525	530	2
RssReader::handleContentEncoded	532	537	3
RssReader::handleSummary	539	544	3
RssReader::handleContent	546	551	2
RssReader::handleAuthorName	553	558	2
RssReader::initFeed	563	567	1
RssReader::pushElement	574	582	3
RssReader::popElement	588	597	3
RssReader::validateFeed	603	607	2
RssReader::fixGuid	612	618	3
RssReader::characters	621	628	2
RssReader::fatalError	631	637	2
RssReader::getContent	644	648	1
RssReader::getFeed	655	657	1
RssReader::getArticleList	664	666	1

IndexingService

Function Name	Start Line	End Line	Cyclomatic Complexity (Threshold: 10)	Lin
IndexingService::IndexingService	59	61	1	3
IndexingService::startUp	64	78	5	14
IndexingService::shutDown	81	96	5	16
IndexingService::runOneIteration	99	103	1	4
IndexingService::scheduler	106	108	1	3
IndexingService::searchArticles	119	163	6	37
IndexingService::rebuildIndex	169	172	1	4
IndexingService::getDirectory	179	181	1	3
IndexingService::getDirectoryReader	190	213	6	24

FeedService

Function Name	Start Line	End Line	Cyclomatic Complexity (Threshol
FeedService::startUp	63	64	1
FeedService::shutDown	67	68	1
FeedService::runOneIteration	71	79	2
FeedService::scheduler	82	86	1
FeedService::synchronizeAllFeeds	91	134	7
FeedService::synchronize	141	165	2
FeedService::parseFeedOrPage	261	303	8
FeedService::logParsingError	305	315	5
FeedService::createInitialUserArticle	325	349	4
FeedService::ArticleService::ArticleService	361	364	1
FeedService::ArticleService::getArticleToRemove	369	405	7
FeedService::ArticleService::getNewerArticleList	407	415	3
FeedService::ArticleService::getOldestArticle	417	426	4
FeedService::ArticleService::completeArticleList	431	438	4
FeedService::ArticleService::removeOldArticles	443	477	6
FeedService::ArticleService::call	479	482	1
FeedService::CreateFeed::createFeed	487	535	7
FeedService::CreateFeed::isFaviconUpdated	543	549	3
FeedService::ManageArticles::ManageArticles	560	574	3
FeedService::ManageArticles::updateArticles	576	620	6
FeedService::ManageArticles::createArticles	622	662	4
FeedService::ManageArticles::call	664	667	1
FeedService::ManageArticles::getArticleMap	669	671	1

SubscriptionImportAsyncListener

Function Name	Start Line	End Line	Cyclomatic Complexity (
SubscriptionImportAsyncListener::onSubscriptionImport	75	89	3

Function Name	Start Line	End Line	Cyclomatic Complexity (
SubscriptionImportAsyncListener::createJob	100	145	4
SubscriptionImportAsyncListener::getOutlineCount	147	187	6
SubscriptionImportAsyncListener::getFeedCount	195	206	2
SubscriptionImportAsyncListener::processImportFile	215	253	8
SubscriptionImportAsyncListener::getOutlineList	255	293	6
SubscriptionImportAsyncListener::importOutline	302	425	***15***

We can observe that the methods are now well within the thresholds for cyclomatic complexity, lines of code and parameter count.

Designite analysis

- **MainActivity:** Decreased LOC (321→0) and FANIN (1→0), with a slight decrease in FANOUT (13→12).
  Positive: Clears out extraneous code and reduces coupling.
  Tradeoff: Shifts responsibilities that must be managed elsewhere.
- **UserResource:** Increased FANOUT (18→24) while LOC remains unchanged (445).
  Positive: Likely adds functionality or delegates tasks.
  Tradeoff: Higher external calls may increase coupling and dependency complexity.
- **BaseResource (Variant 1):** All metrics unchanged (NOF: 3, NOM: 3, LOC: 53, FANOUT: 2).
  Positive: Stable, optimal design.
  Tradeoff: No further optimization opportunities applied.
- **BaseResource (Variant 2):** Increased NOM (3→4) and FANOUT (2→3) with decreased LOC (53→39).
  Positive: Successful decomposition into smaller, focused methods.
  Tradeoff: Slight increase in coupling (FANOUT) could pose minor dependency risks.
- **SecurityFilter:** Decreased FANIN (2→1) and FANOUT (7→6).
  Positive: Reduced coupling leads to a clearer separation of concerns.
  Tradeoff: Fewer incoming calls might limit its reusability by other components.
- **RequestContextFilter:** All metrics remain essentially unchanged.
  Positive: Indicates a well-designed, stable class.
  Tradeoff: No direct improvements, which is acceptable given its optimal state.
- **LocaleUtil:** All metrics remain unchanged.
  Positive: Demonstrates robust utility design.

Tradeoff: No refactoring benefits applied, though functionality remains clear.

- **IndexingService:** All metrics remain unchanged.
Positive: Stability confirms its design serves its purpose effectively.
Tradeoff: No enhancements, but performance is maintained.

- **FeedService:** Increased NOM (16→32), LOC (350→371), WMC (56→69), and FANOUT (20→28); decreased LCOM (0.4375→0.21875).
Positive: Method decomposition improves cohesion and maintainability.
Tradeoff: More methods and external interactions may raise overall complexity and coupling.

- **SubscriptionImportAsyncListener:** Decreased LOC (391→370) and WMC (42→40) with increased FANOUT (18→32).
Positive: A leaner design with lower overall code size.
Tradeoff: Increased external dependencies might complicate future maintenance.

- **UserDao:** Increased FANIN (1→3) and FANOUT (5→6).
Positive: Becoming a more central, reusable component.
Tradeoff: Elevated coupling risks turning it into a "god class" without proper modularization.

- **RssReader:** Increased NOF (22→25), NOM (14→43), and FANOUT (7→8); decreased LOC (447→425) and WMC (97→86).
Positive: Breaking down methods improves cohesion and maintainability.
Tradeoff: A higher number of methods can make navigation more challenging despite reduced per-method complexity.

- **RssReader.FeedType & RssReader.Element:** New helper/inner classes introduced with minimal LOC and WMC.
Positive: Clarifies design by isolating helper functionality.
Tradeoff: Additional components may slightly complicate the overall project structure.

- **AtomUrlGuesser (Merged Class):** Increased NOM (merged to 4), LOC (increased to 84), and WMC (increased to 22).
Positive: Merging similar strategies reduces redundancy and centralizes functionality.
Tradeoff: Consolidation leads to higher complexity within a single class that requires careful management.

- **Constants:** Decreased NOF (14→1), LOC (62→12), and FANIN (7→6).
Positive: Replacing a bloated class with an enum greatly simplifies design and improves encapsulation.
Tradeoff: Changes may necessitate updates in areas that previously depended on the old structure.

- **StarredReader:** Increased NOF (2→3), NOM (2→6), NOPM (2→6),

LOC (91→112), and WMC (13→16) with decreased FANOUT (5→3).
Positive: Decomposing functionality into more methods enhances modularity and lowers coupling.
Tradeoff: An increase in size and complexity could make the class bulkier if not carefully managed.

- **StarredArticleImportedListener:** Class completely removed.
Positive: Simplifies overall architecture by eliminating redundancy.
Tradeoff: Its functionality must be redistributed to avoid gaps.

Conclusion

The refactoring has resulted in a more maintainable codebase. The classes are now less dependent on each other and the complexity has reduced. The cohesion has improved (as seen by increased NOM, reduced WMC per method). The clarity of the code has improved with the removal of redundant code and the introduction of helper classes. (e.g., merging strategies and simplifying constants).
