

## 1. JPAWeblogManagerImpl

### Design Smell - Feature Envy

A method has **Feature Envy** when it uses another class's data and methods more than its own class's data.

### SonarQube Reference:

The screenshot shows a SonarQube analysis results page for a Java file named `JPAWeblogManagerImpl.java`. The top navigation bar includes tabs for 'Adaptability' (highlighted) and 'Not focused'. A message at the top says 'Refactor this method to reduce its Cognitive Complexity from 19 to the 15 allowed.' Below this, a note states 'Cognitive Complexity of methods should not be too high [java:S3776](#)'. The 'Software qualities impacted' section shows 'Maintainability' with a 'High' rating. The bottom navigation bar includes tabs for 'Where is the issue?' (highlighted), 'Why is this an issue?', 'How can I fix it?', 'Activity', and 'More info'. The main content area displays the Java code with line numbers 268 through 280. Line 273 contains the method `private void addWeblogContents(Weblog newWeblog)`, which is underlined with a red wavy line, indicating it is the source of the issue. A callout box above this line says 'Refactor this method to reduce its Cognitive Complexity from 19 to the 15 allowed.' The code itself includes logic for storing and flushing the weblog, and granting ADMIN permission to the creator.

```
app/.../roller/weblogger/business/jpa/JPAWeblogManagerImpl.java
```

```
268 rishik...      this.strategy.store(newWeblog);
269          this.strategy.flush();
270          this.addWeblogContents(newWeblog);
271      }
272
273  private void addWeblogContents(Weblog newWeblog)
274
275      throws WebloggerException {
276
277      // grant weblog creator ADMIN permission
278      List<String> actions = new ArrayList<>();
279      actions.add(WeblogPermission.ADMIN);
280      roller.getUserManager().grantWeblogPermission(
281          newWeblog, newWeblog.getCreator(), actions);
```

High Cognitive Complexity means

- Handles too many cases
- Coordinates many external objects
- Knows too much about another class

### Designite Java Reference:

The tool detected an instance of this smell because `addWeblogContents` is more interested in members of the type: `Weblogger`.

## 2. JPAWeblogEntryManagerImpl

### Design Smell - Insufficient Modularization

#### SonarQube Reference

Adaptability | Not focused

Refactor this method to reduce its Cognitive Complexity from 27 to the 15 allowed. ↗

Cognitive Complexity of methods should not be too high [Java:S3776](#)

Software qualities impacted: Maintainability ⚡ High

Open by KRishika01 · Code Smell · Critical

Where Is the Issue? Why is this an issue? How can I fix it? Activity More info

... Show 176 more lines

```
177 rishik...    /**
178      * @inheritDoc
179      */
180     // T000: perhaps the createAnchor() and queuePings() items should go outside this method?
181     @Override
182     public void saveWeblogEntry(WeblogEntry entry) throws WebloggerException {
183
184         if (entry.getCategory() == null) {
185             // Entry is invalid without category, so use weblog client cat
186             WeblogCategory cat = entry.getWebsite().getBloggerCategory();
187             if (cat == null) {
188                 // Still no category, so use first one found
189                 cat = entry.getWebsite().getWeblogCategories().iterator().next();
190             }
191             entry.setCategory(cat);
192         }
193     }
194 }
```

Refactor this method to reduce its Cognitive Complexity from 27 to the 15 allowed.

Tags  
brain-overload

Line affected  
L182

Effort  
17 min

Introduced  
2 days ago

Open in I

- The method has too much logic in one place
- Many branches, loops, or nested conditions
- The method is handling multiple responsibilities

#### Designite Java Reference:

The tool detected the smell in this class because the class has a bloated interface (a large number of public methods). Total public methods in the class: 44 public methods

## 3. WeblogEntry

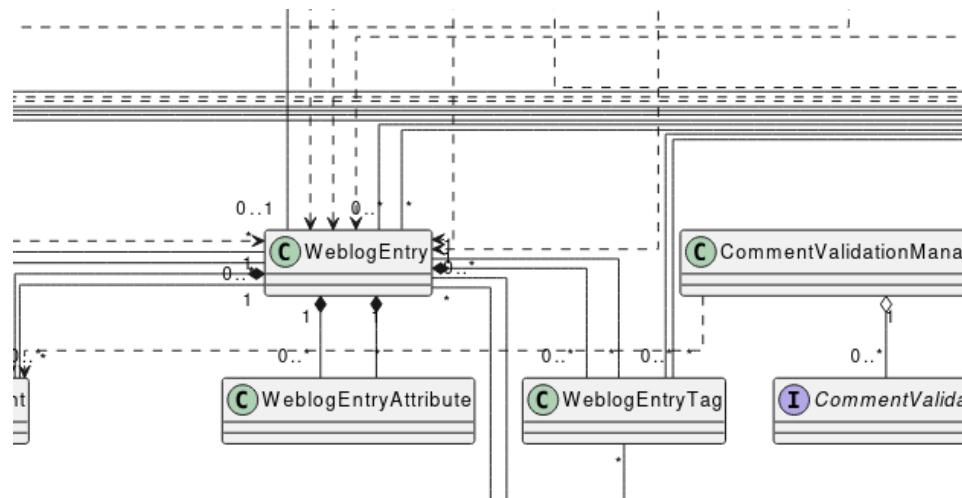
### Design smell - hub-like modularisation

#### Designite reference:

The tool detected the smell in this class because this class has high number of incoming as well as outgoing dependencies. Incoming dependecies: EntryEdit; Entries; EntryRemove; Comments; EntryBean; WeblogEntriesPermalinkPager; WeblogEntriesListPager;

WeblogEntriesPreviewPager; WeblogEntriesLatestPager;  
WeblogEntriesMonthPager; WeblogEntriesDayPager;  
WeblogTrackbackRequest; WeblogCommentRequest;  
WeblogPreviewRequest; WeblogPageRequest; SiteWideCache;  
URLModel; TrackbackServlet; PageServlet; CommentServlet;  
WeblogCalendarModel; BigWeblogCalendarModel;  
WeblogEntryManager; PluginManagerImpl; PluginManager;  
EncodePreTagsPlugin; WeblogEntryPlugin; ObfuscateEmailPlugin;  
ConvertLineBreaksPlugin; SmileysPlugin; IndexOperation;  
AddEntryOperation; LuceneIndexManager; RemoveEntryOperation;  
ReIndexEntryOperation; IndexManager; AutoPingManager;  
JPAWeblogEntryManagerImpl; JPAAutoPingManagerImpl; Weblog;  
WeblogEntryComment; WeblogEntryTag; CommentSearchCriteria;  
WeblogEntryAttribute; WeblogEntryWrapper; CacheManager;  
CacheHandler; MailUtil; Trackback; RollerAtomHandler; EntryCollection;  
BloggerAPIHandler; MetaWeblogAPIHandler. Outgoing dependencies:  
WeblogEntry.PubStatus; Weblog; WeblogCategory; User;  
WebloggerFactory; WeblogEntryAttribute; WebloggerException;  
WeblogEntryTag; Utilities; WebloggerRuntimeConfig;  
WeblogEntryManager; CommentSearchCriteria; DateUtil;  
GlobalPermission; WeblogPermission; UserManager;  
WeblogEntryPlugin; HTMLSanitizer; WeblogEntryComment;  
RollerConstants

## Uml reference:



- Acts as a **dependency hub** with very high fan-in and fan-out
  - **Tightly coupled** to many unrelated modules

- Violates **single responsibility principle**
- Changes in this class can **impact many other classes**
- Difficult to **test, maintain, and evolve** independently
- Reduces overall **modularity and system stability**

## 4. LuceneIndexManager

### Design smell: Spaghetti Code

Spaghetti code refers to source code with a convoluted, tangled, and unstructured control flow, making it difficult to understand, maintain, or extend.

#### SonarQube Reference:

The screenshot shows the SonarQube interface for the file `LuceneIndexManager.java`. On the left, there's a sidebar with metrics: New Lines (0), Lines of Code (342), Lines (486), Statements (146), Functions (25), Classes (1), Files (1), Comment Lines (39), Comments (%) 10.2%, Generated Lines (0), Generated Lines of Code (0), and Complexity (50). The Complexity section is expanded to show Cyclomatic Complexity (50) and Cognitive Complexity (59). The main panel shows the Java code with SonarQube annotations. The annotations are mostly blank, except for the first few lines which contain a copyright notice and license information.

```

1  rtshk...   /*
2   * Licensed to the Apache Software Foundation (ASF) under one or more
3   * contributor license agreements. The ASF licenses this file to You
4   * under the Apache License, version 2.0 (the "License");
5   * you may not
6   * use this file except in compliance with the License.
7   *
8   * You may obtain a copy of the License at
9   *
10  *      http://www.apache.org/licenses/LICENSE-2.0
11  *
12  * Unless required by applicable law or agreed to in writing, software
13  * distributed under the License is distributed on an "AS IS" BASIS,
14  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  * See the License for the specific language governing permissions and
16  * limitations under the License. For additional information regarding
17  * copyright in this work, please see the NOTICE file in the top level
18  * directory of this distribution.
19  */
20
21 package org.apache.roller.weblogger.business.search.lucene;
22
23 import java.io.File;
24 import java.io.IOException;
25 import java.lang.reflect.InvocationTargetException;
26 import java.nio.file.Files;
27 import java.nio.file.Path;
28 import java.util.ArrayList;
29 import java.util.List;
30 import java.util.Set;
31 import java.util.TreeSet;
32 import java.util.concurrent.locks.ReadWriteLock;
33 import java.util.concurrent.locks.ReentrantReadWriteLock;
34 import org.apache.commons.beanutils.ConstructorUtils;
35 import org.apache.commons.logging.Log;

```

From the picture we can see that,

LOC: 342

Cognitive complexity: 59

Cyclomatic complexity: 50

#### Indicators of Spaghetti code:

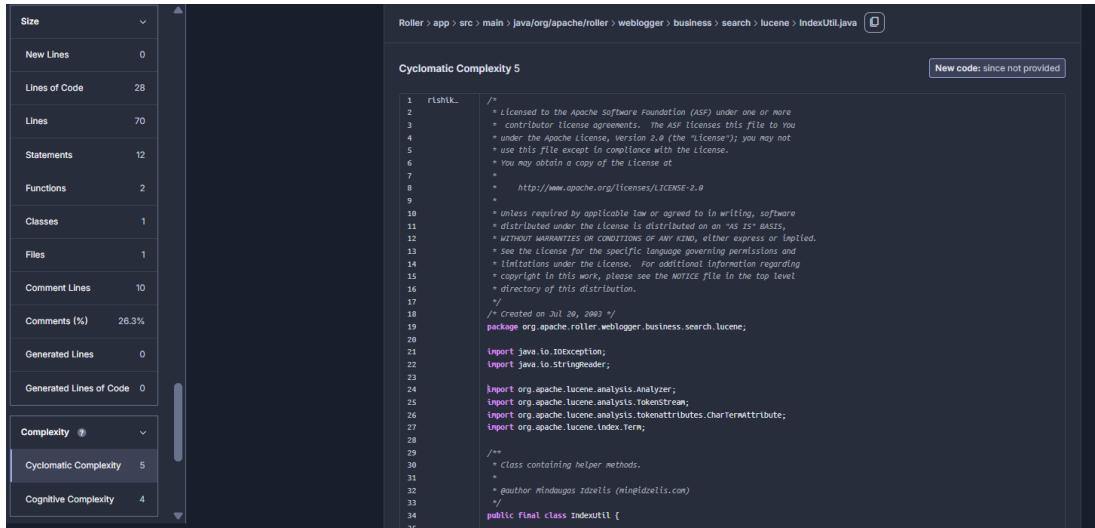
1. **Very high cyclomatic complexity (50)**
  - Many branching paths hence tangled flow
2. **Very high cognitive complexity (59)**
  - Nested conditionals mixed concerns, jumps in logic
3. **High LOC in a single class/method**
  - 342 LOC → suggests long, monolithic logic blocks

## 5. IndexUtil

## Design smell: Lazy class

It only has one static method that wraps Lucene's Term creation, which is minimal functionality that doesn't justify a separate class

### SonarQube reference:



The screenshot shows the SonarQube interface for the file `IndexUtil.java`. On the left, there is a sidebar with metrics: Size (New Lines: 0, Lines of Code: 28, Lines: 70, Statements: 12, Functions: 2, Classes: 1, Files: 1, Comment Lines: 10, Comments (%): 26.3%, Generated Lines: 0, Generated Lines of Code: 0), and Complexity (Cyclomatic Complexity: 5, Cognitive Complexity: 4). The main pane displays the Java code for `IndexUtil`, which contains a single static method `getTerm` that wraps Lucene's token analysis into Term creation. The cyclomatic complexity is annotated as 5.

```
1  rtshtk_ /*  
2   * Licensed to the Apache Software Foundation (ASF) under one or more  
3   * contributor license agreements. The ASF licenses this file to you  
4   * under the Apache License, Version 2.0 (the "License"); you may not  
5   * use this file except in compliance with the License.  
6   * You may obtain a copy of the License at  
7   *  
8   *      http://www.apache.org/licenses/LICENSE-2.0  
9   *  
10  * Unless required by applicable law or agreed to in writing, software  
11  * distributed under the License is distributed on an "AS IS" BASIS,  
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
13  * See the License for the specific language governing permissions and  
14  * limitations under the License. For additional information regarding  
15  * copyright in this work, please see the NOTICE file in the top level  
16  * directory of this distribution.  
17  */  
18  /* Created on Jul 29, 2003 */  
19  package org.apache.roller.weblogger.business.search.lucene;  
20  
21  import java.io.IOException;  
22  import java.io.StringReader;  
23  
24  import org.apache.lucene.analysis.Analyzer;  
25  import org.apache.lucene.analysis.TokenStream;  
26  import org.apache.lucene.analysis.TokenAttributes.CharTermAttribute;  
27  import org.apache.lucene.IndexTerm;  
28  
29  /**  
30  * Class containing helper methods.  
31  *  
32  * @author Mindaugas Tzelis (mtingzells.com)  
33  */  
34  public final class IndexUtil {
```

### Justification:

- Single method** - It only has one public static method `getTerm()`
- Minimal responsibility** - Just wraps Lucene token analysis into Term creation
- No meaningful abstraction** - It's just a "helper" class with no real identity
- Unnecessary indirection** - The ~20 lines of logic could easily live in the classes that actually use it

### UML justification:

