Task 2B: Identification and Analysis of High Coupling and High Complexity Classes

Objective

The objective of Task 2B is to identify classes with **high coupling and/or high complexity** using static code analysis metrics, analyze their impact on software quality, and highlight potential refactoring candidates.

Overview of Analysis

A total of **532 classes** were analyzed using software quality metrics such as:

- Coupling Between Objects (CBO)
- Weighted Methods per Class (WMC)
- Response for a Class (RFC)
- Lines of Code (LOC)
- Lack of Cohesion (LCOM)

The analysis focuses on identifying:

- Classes with **high coupling**
- Classes with **high coupling and high complexity**

Metric Summary

Based on the detailed metric tables and charts:

Category	Number of Classes
Classes with high coupling & high complexity	1

Classes with high coupling	8
Classes with high complexity only	0
Total classes analyzed	532

The majority of classes fall within acceptable thresholds, indicating overall good modularity, with a small set of critical classes requiring attention.

Classes with High Coupling

The following **8 classes** exhibit high coupling, indicating strong dependencies on multiple other classes, which can reduce maintainability and flexibility:

1. **JPAWeblogEntryManager**
    - LOC: 754
    - CBO: 20
    - RFC: 50
    - Observation: Very large class with multiple responsibilities and high external dependencies.
2. **JPAWeblogManagerImpl**
    - LOC: 389
    - CBO: 25
    - RFC: 43
    - Observation: High dependency on application-level components.
3. **MediaCollection**
    - LOC: 357
    - CBO: 21
    - RFC: 23
4. **EntryCollection**
    - LOC: 274
    - CBO: 26
    - RFC: 43
5. **CommentServlet**
    - LOC: 210
    - CBO: 23
    - RFC: 42
6. **ThemeManagerImpl**
    - LOC: 208
    - CBO: 21
    - RFC: 38
7. **WebloggerImpl**

    - LOC: 199
    - CBO: 26
    - RFC: 54
    - Observation: Low cohesion combined with high coupling makes this class difficult to maintain.
8. **RollerContext**
    - LOC: 155
    - CBO: 24
    - RFC: 35

Class with High Coupling and High Complexity

Only **one class** falls under this critical category:

PageServlet

- LOC: 343
- CBO: 26
- WMC: 107
- RFC: 52

Observation:

- This class has a very high number of methods and responsibilities.
- Strong indicator of a **God Class**.
- Difficult to test, maintain, and extend.

Impact on Software Quality

High coupling and complexity can lead to:

- Reduced maintainability
- Increased risk of defects
- Difficulties in testing and debugging
- Lower code reusability

Classes like **JPAWeblogEntryManager** and **PageServlet** pose the highest risk due to their size and dependency density.

Recommendations

1. **Refactor large classes** by applying:
    - Single Responsibility Principle (SRP)
    - Separation of concerns
2. **Introduce interfaces** to reduce direct dependencies.
3. **Extract helper or service classes** from God classes.
4. **Reduce method count** by breaking complex logic into smaller units.

Conclusion

Task 2B successfully identified a small subset of classes with high coupling and complexity. While the overall system shows good structural quality, targeted refactoring of the identified classes will significantly improve maintainability, scalability, and long-term code health.