# Section 1: Objectives

Goal: Does class size (LOC) correlate with maintainability risks as proxied by complexity (WMC) and coupling (CBO)?

Metrics used.

LOC (per class) – size proxy

WMC (Weighted Methods per Class) – complexity proxy

CBO (Coupling Between Objects) – coupling proxy

Question: For large java projects that were in production for a while. Is there a connection between Lines of Code (LOC) and its Weighted Methods per Class(WMC) or Coupling Between Objects(CBO)?

# Section 2: Description

Age: At least 3 years old

Size: 10,00 LOC

Amount of developers: More than 3

We go through 5 projects, Keycloak, Jfreechart, Spring-boot, Rxjava, and Commons-Lang

We generate a summary of the stats for **LOC, WMC, CBO.** And we compare the programs and observer patterns throughout the data.

###### Keycloak

Open Source Identity and Access Management For Modern Applications and Services

###### Jfreechart

JFreeChart is a comprehensive free chart library for the Java™ platform that can be used on the client-side (JavaFX and Swing) or the server side, with export to multiple formats including SVG, PNG and PDF.

###### Spring-boot

Spring Boot is a Java-based framework for creating stand-alone, it helps to create Spring-powered, production-grade applications and services with absolute minimum fuss. It takes an opinionated view of the Spring platform so that new and existing users can quickly get to the bits they need.

###### Rxjava

open-source library for building reactive, event-driven, and asynchronous programs in Java and Android, using the ReactiveX pattern

###### Commons-lang

Java library offering helper utilities and enhancements for core Java classes, focusing on areas like String manipulation, numerical methods, object reflection, concurrency, and help with generating equals(), hashCode(), and toString() methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | URL | Age  (Years) | Size (LoC) | # of Classes |
| KeyCloak | https://github.com/keycloak/keycloak | 11 | 178,107 | 43 |
| jfreechart | https://github.com/jfree/jfreechart | 25 | 49,471 | 507 |
| Spring-Boot | https://github.com/spring-projects/spring-boot | 11 | 272,997 | 10,420 |
| RxJava | https://github.com/ReactiveX/RxJava | 12 | 34,335 | 2,899 |
| Common-Langs | https://github.com/apache/commons-lang | 23 | 73,663 | 742 |

# Section 3: Tools

We used a tool call CK, the tool measures Chidamber and Kemerer (C&K) metrics for Java source code.

Built CK (v0.7.0) and wrote a runner that emits class.csv, method.csv, field.csv, variable.csv.

Ran CK on java programs and filtered to production files for analysis.

Aniche, M. (2023). CK - Chidamber and Kemerer metrics for Java code

# Section 4: Data

KeyCloak (Security Domain)

* **Size**: Smallest project (43 classes)
* **Characteristics**: High coupling density, moderate complexity
* **Maintainability**: Good overall, but security testing classes show high complexity
* **Key Issues**: Security verification classes have high LCOM values, indicating poor cohesion

Spring-Boot (Web Framework Domain)

* **Size**: Largest project (10,420 classes)
* **Characteristics**: Lowest WMC, moderate coupling, very low LOC per class
* **Maintainability**: Excellent for its size - demonstrates that large projects can maintain good quality
* **Key Issues**: High coupling density due to auto-configuration dependencies
* **Notable**: Despite being the largest project, it has the lowest average WMC (6.1)

JFreeChart (Visualization Domain)

* **Size**: Medium project (507 classes)
* **Characteristics**: Highest complexity density, moderate coupling
* **Maintainability**: Challenging due to high WMC and LCOM values
* **Key Issues**: Chart rendering classes are highly complex with poor cohesion

Commons-Lang (Utilities Domain)

* **Size**: Large project (742 classes)
* **Characteristics**: Highest LCOM values, lowest coupling density
* **Maintainability**: Mixed - good coupling but poor cohesion in utility classes
* **Key Issues**: Utility classes with many static methods show poor cohesion

RxJava (Reactive Programming Domain)

* **Size**: Very large project (2,899 classes)
* **Characteristics**: Lowest complexity density, moderate coupling
* **Maintainability**: Surprisingly good for its size
* **Key Issues**: Test classes dominate the outliers, indicating testing complexity

1. Size Distribution and Complexity Patterns

Project Size Ranking:

* Spring-Boot: 10,420 classes (largest)
* RxJava: 2,899 classes
* Commons-Lang: 742 classes
* JFreeChart: 507 classes
* KeyCloak: 43 classes (smallest)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Project | WMC | CBO | LCOM | LOC | RFC |
| KeyCloak | 8.7 | 6.6 | 41.0 | 61.8 | 19.5 |
| Spring-Boot | 6.1 | 6.1 | 37.0 | 31.3 | 12.0 |
| JFreeChart | 36.7 | 8.6 | 392.1 | 143.1 | 29.0 |
| Commons-Lang | 21.2 | 4.7 | 647.1 | 106.6 | 18.1 |
| RxJava | 12.6 | 9.1 | 403.1 | 104.7 | 16.0 |

Complexity per Line of Code:

* JFreeChart: 0.2566 (highest complexity density)
* Commons-Lang: 0.1987
* Spring-Boot: 0.1956
* KeyCloak: 0.1411
* RxJava: 0.1204 (lowest complexity density)

Coupling per Line of Code:

* Spring-Boot: 0.1941 (highest coupling density)
* KeyCloak: 0.1061
* RxJava: 0.0873
* JFreeChart: 0.0600
* Commons-Lang: 0.0439 (lowest coupling density)

# Section 5: Conclusions

From the data, we can observe that Size does NOT directly correlate with poor Maintainability.

**Best to Worst (by complexity per LOC):**

1. **RxJava** (2,899 classes) - 0.1204
2. **KeyCloak** (43 classes) - 0.1411
3. **Spring-Boot** (10,420 classes) - 0.1956
4. **Commons-Lang** (742 classes) - 0.1987
5. **JFreeChart** (507 classes) - 0.2566

From the list above that the complexity per LOC does not relate to the total amount of classes that the programs have. JFreeChart has the highest complexity per LOC but its only the 3 biggest program we observed. Meanwhile RxJava the second biggest project has the best complexity per LOC.

# Section 6: References

Aniche, M. (2023). *CK - Chidamber and Kemerer metrics for Java code*