**FINAL REPORT**

**Objectives:**

1. **Assess the overall health and maintenance of the software projects.**

**Questions:**

1. What is the current size of the software project under analysis
2. What is the maximum number of open issues in each project?

**Objectives:**

**2. Identifying projects with powerful maintenance processes.**

**Questions:**

1. How does the size of a project relate with its ability to manage and resolve open issues?
2. What is the relation between project size and the frequency of the commits?

**Objectives:**

1. **Evaluating the scalability and complexity of software projects based on their size and development activity.**

**Questions:**

1. How does the size of a project impact its long-term maintainability and scalability?
2. Are projects with a larger number of commits more likely to encounter scalability challenges?

**Metrics:**

1. **Size:**

* **Understanding the size of the project is crucial for assessing its scalability, maintainability and potential risks.**

1. **Maximum Number of Open issues:**

* **Monitoring the maximum number of open issues helps assess the projects maintenance backlog and overall health of the software.**

1. **Maximum number of commits:**

* **Tracking the maximum number of commits provides insights into the project’s development.**

**Section 2**

**Geometric Weather**

**(https://github.com/WangDaYeeeeee/GeometricWeather)**

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| Size | Medium to large codebase with multiple files and directories. |
| Complexity | Moderate to high complexity due to various features and functionalities |
| Activity level | Actively maintained with regular updates and contributions |
| Programming Language | Primarily written in Java |
| Community | Geometric Weather has a growing community of users and contributors who provide feedback, report issues, and contribute code enhancements. The community engagement includes discussions on GitHub, feature requests, bug reports, and collaboration on improving the app's functionality and user experience. |

Geometric Weather is a weather app for Android devices that provides detailed weather information based on location. It offers features such as current weather conditions, forecasts, air quality index, UV index, and weather alerts. The app utilizes APIs to fetch weather data and displays it through a user-friendly interface with customizable themes and widgets. Users can track weather conditions for different locations, receive notifications for severe weather events, and access.

**Elastic Search**

(**https://github.com/elastic/elasticsearch**)

|  |  |
| --- | --- |
| Attributes | Description |
| Size | The Elastic search codebase is substantial, consisting of thousands of files and significant lines of Java code. |
| Complexity | Elastic search is known for its complex architecture, including components such as index management, distributed search, data ingestion, and RESTful API interfaces. |
| Activity level | The project is actively maintained by the Elastic search development team and has a vibrant community of contributors. It receives regular updates and bug fixes. |
| Programming language | Elastic search is primarily written in Java, leveraging the Java Virtual Machine (JVM) for cross-platform compatibility and performance. |
| Community | The Elastic search project has a strong community of users and contributors who actively participate in discussions, issue tracking, and code contributions. |

Elastic search is a distributed search and analytics engine designed to store, search, and analyze large volumes of structured and unstructured data in real-time.

**BYTE BUDDY**

**(https://github.com/raphw/byte-buddy)**

|  |  |
| --- | --- |
| Attribute | Description |
| Size | Medium-sized codebase with multiple files and directories. |
| Complexity | Moderate complexity due to advanced byte code manipulation. |
| Activity level | Actively maintained with regular updates and contributions. |
| Programming Language | Primarily written in Java. |
| Community | Byte Buddy has a supportive community of users and contributors who actively participate in discussions, issue tracking, and code contributions. The project is well-documented with tutorials, guides, and examples available for developers. |

Byte Buddy is a Java library for creating and modifying Java classes at runtime. It allows developers to generate new classes, modify existing classes, or create proxies for existing classes. This makes it useful for various tasks such as bytecode manipulation, AOP (Aspect-Oriented Programming), mocking, and dynamic code generation. Byte Buddy simplifies tasks that involve dynamically creating or modifying Java classes, offering a high-level API for bytecode manipulation. It is commonly used in frameworks and libraries that require dynamic class generation or manipulation, such as testing frameworks, ORM (Object-Relational Mapping) tools, and dependency injection frameworks.

**Ant Media Server**

**(https://github.com/ant-media/Ant-Media-Server)**

|  |  |
| --- | --- |
| Attribute | Description |
| Size | Large |
| Complexity | Medium to high |
| Activity Level | Active |
| Programming language | Java |
| Community | Strong community support with active development and user engagement |

The Ant Media Server is a powerful Java-based software solution designed for real-time video streaming applications. It provides features such as adaptive bitrate streaming, WebRTC support, and both live and on-demand video streaming capabilities. The program allows users to deploy their own video streaming infrastructure, enabling applications ranging from live event broadcasting to online video conferencing and interactive multimedia applications. The Ant Media Server project boasts an active community of developers and users, contributing to its ongoing development and support.

**Bazel**

**(https://github.com/bazelbuild/bazel)**

|  |  |
| --- | --- |
| Attribute | Description |
| Size | Large codebase with multiple directories and files |
| Complexity | Moderate to high due to its build system functionalities |
| Activity Level | Actively maintained with frequent updates and contributions |
| Programming Language | Primarily written in Java, with some components in other languages |
| Community | Bazel has a vibrant community of users and contributors who actively participate in discussions, issue tracking, and code contributions. The community provides support through forums, mailing lists, and online documentation. |

Bazel is a build system that allows for fast and reliable building, testing, and deployment of software projects. It provides a language-agnostic, scalable, and reproducible build infrastructure. Bazel is particularly known for its support for large, multi-language projects and its use of dependency graphs to optimize build performance.

**Section 3**

The "CK" tool, available at the GitHub repository maintained by Maurício Aniche under the URL mauricioaniche/ck, is an open-source software utility designed to analyze and measure the structural and maintainability characteristics of software projects. CK, short for "CodeKeeper," provides developers and project maintainers with valuable insights into various aspects of their codebases, facilitating informed decision-making to enhance software quality. It offers functionality such as calculating software metrics related to code complexity, size, coupling, and cohesion, along with visualizations of these metrics and trend analysis capabilities. CK's ease of use and integration into continuous integration pipelines make it a practical tool for automating code analysis and benchmarking. Supported by an active community of users and contributors, CK continues to evolve, improving its reliability, usability, and functionality over time.

**CITATION**

**@manual{aniche-ck,**

**title={Java code metrics calculator (CK)},**

**author={Maurício Aniche},**

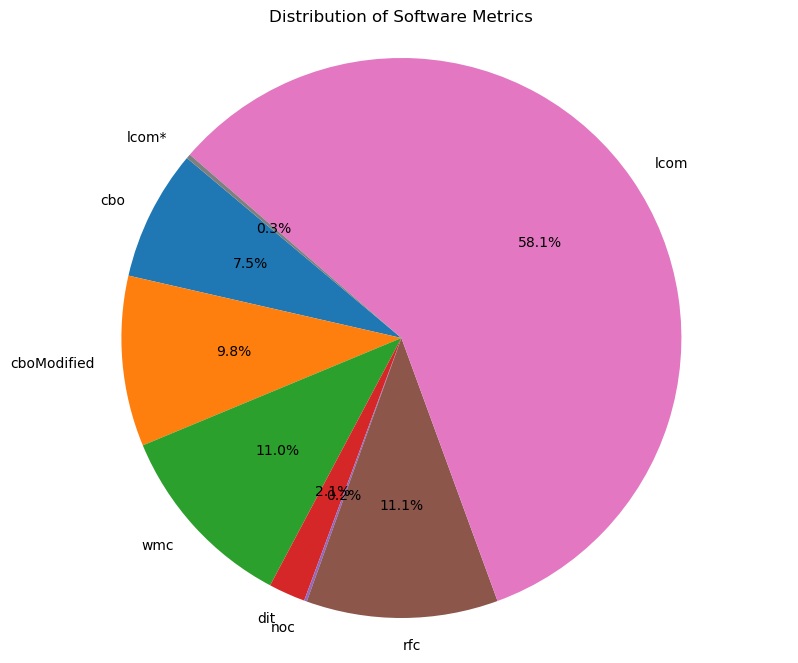
**year={2015},**

**note={Available in https://github.com/mauricioaniche/ck/}**

**}**

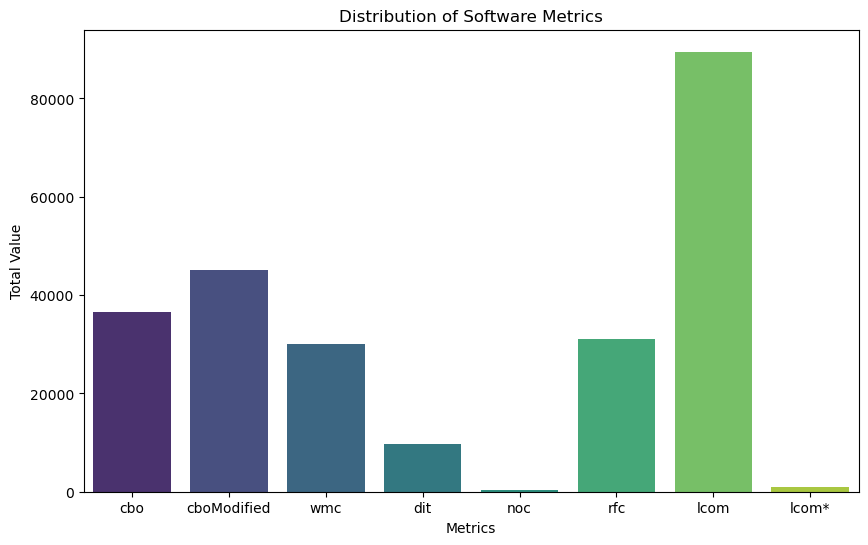
**SECTION 4**

1. **GeoMetric Weather**



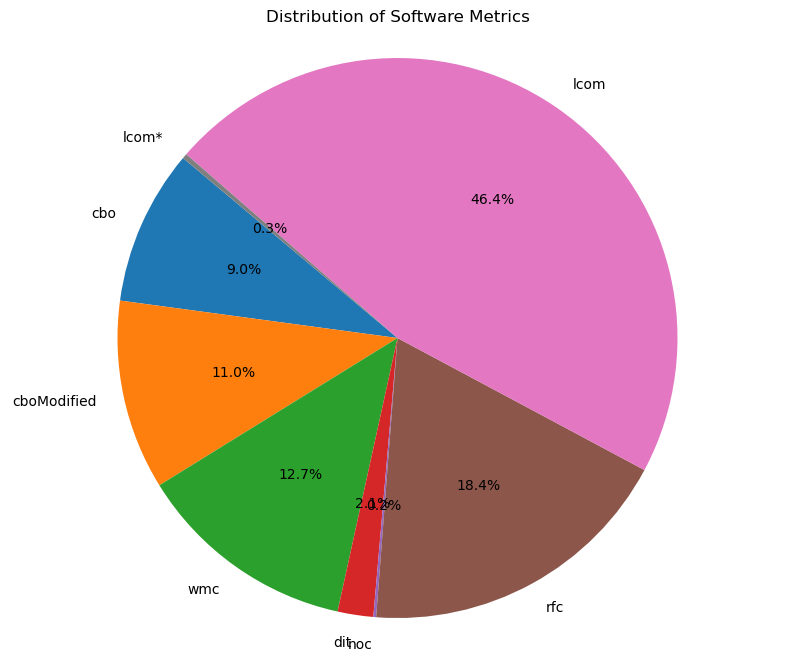
Overall, the codebase appears to have a relatively well-structured and maintainable design with a focus on method cohesion (high LCOM and low LCOM\*). The inheritance hierarchy is shallow (low DIT), and there's minimal coupling between classes (low CBO and NOC). However, there are moderate complexities in terms of the number of methods per class (moderate WMC) and method modifications affecting coupling (moderate CBO Modified). Keeping an eye on these moderate complexities can help in ensuring the codebase remains maintainable and scalable as it evolves.

1. Byte Buddy



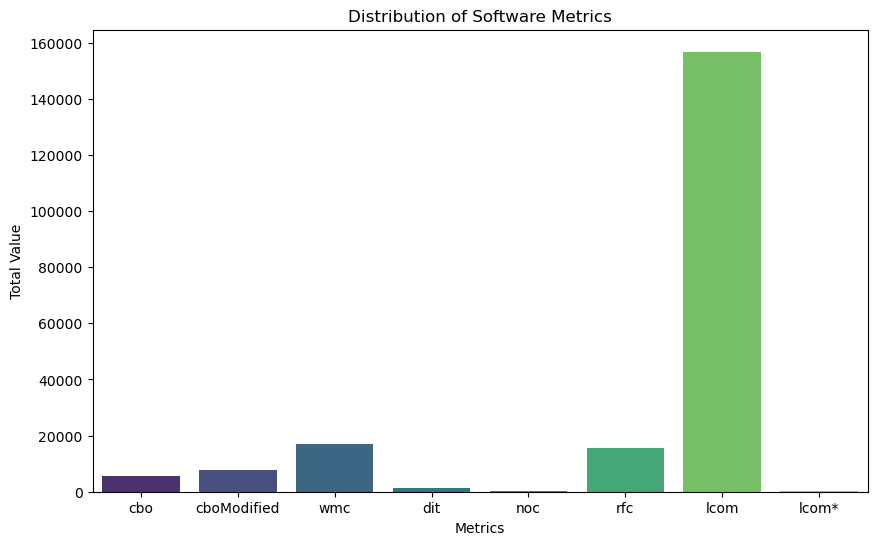
The codebase exhibits a high degree of method cohesion (high LCOM) and a high number of methods that can be called in response to messages (high RFC). However, there are concerns related to the high level of interdependencies between classes (high CBO and CBO Modified), high complexity in terms of the number of methods per class (high WMC), and a moderate inheritance hierarchy (moderate DIT).These metrics suggest that while the codebase has strong method cohesion and potentially powerful class functionalities, there may be challenges related to maintainability, complexity, and class interdependencies.

1. Elastic Search



The codebase exhibits moderate method cohesion (moderate LCOM) and a significant number of methods that can be called in response to messages (high RFC). However, there are concerns related to the moderate level of interdependencies between classes (moderate CBO and CBO Modified), moderate complexity in terms of the number of methods per class (moderate WMC), and a shallow inheritance hierarchy (low DIT).These metrics suggest that while the codebase has some well-structured elements and method cohesion, there may be areas to address in terms of reducing class interdependencies, managing method complexity, and balancing inheritance relationships. A review and potential refactoring of the codebase could help in improving maintainability, scalability, and overall code quality.

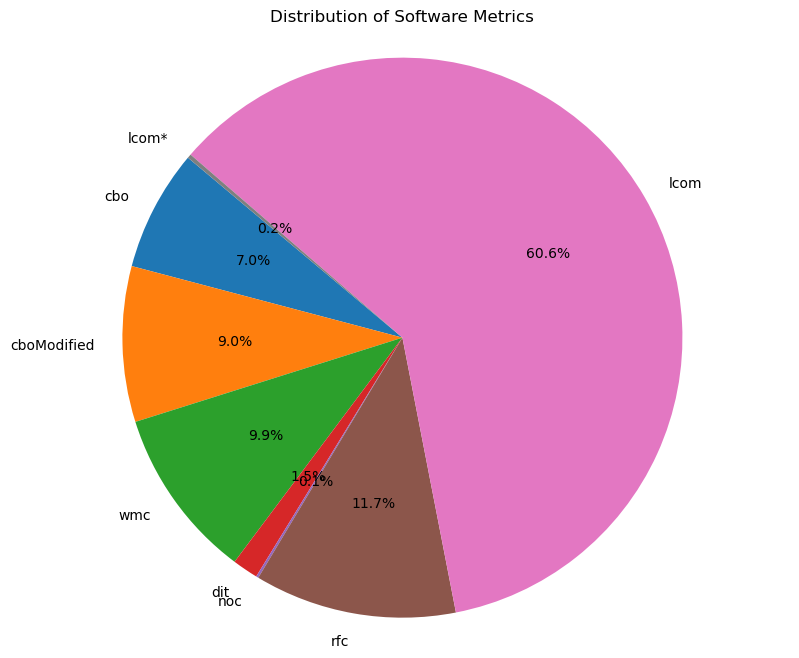
1. AntMedia



The codebase exhibits high method cohesion (high LCOM) and a high number of methods that can be called in response to messages (high RFC). However, there are concerns related to the moderate to high level of interdependencies between classes (moderate CBO and CBO Modified), high complexity in terms of the number of methods per class (high WMC), and a shallow inheritance hierarchy (low DIT).

These metrics suggest that while the codebase has strong method cohesion and potentially powerful class functionalities, there may be challenges related to maintainability, complexity, and class interdependencies. It would be beneficial to review and refactor the codebase to reduce coupling, simplify class hierarchies, and manage method complexity to ensure long-term maintainability and scalability.

1. Bazel



The codebase exhibits strong method cohesion (high LCOM) and a moderate number of methods that can be called in response to messages (moderate RFC). The inheritance hierarchy is shallow (low DIT), and there's minimal coupling between classes (low CBO and CBO Modified), along with a flat class hierarchy (low NOC). However, there are concerns related to the moderate complexity in terms of the number of methods per class (moderate WMC).Overall, the codebase appears to have a well-structured design with a focus on method cohesion and a balanced inheritance hierarchy. While there are some areas to address in terms of method complexity and class interdependencies, the codebase seems to be in a relatively good state with potential for further optimization and improvement.

**References**

1. [**https://github.com/mauricioaniche/ck.git**](https://github.com/mauricioaniche/ck.git)
2. **Arar ÖF, Ayan K. Deriving thresholds of software metrics to predict faults on open source software: Replicated case studies. Expert Systems with Applications. 2016 Nov 1;61:106-21.**
3. [**https://www.researchgate.net/figure/Data-Collected-to-Calculate-CK-Metrics\_tbl1\_224812604**](https://www.researchgate.net/figure/Data-Collected-to-Calculate-CK-Metrics_tbl1_224812604)
4. [**https://www.researchgate.net/publication/318134638\_Effects\_of\_Mean\_Metric\_Value\_Over\_CK\_Metrics\_Distribution\_Towards\_Improved\_Software\_Fault\_Predictions**](https://www.researchgate.net/publication/318134638_Effects_of_Mean_Metric_Value_Over_CK_Metrics_Distribution_Towards_Improved_Software_Fault_Predictions)