Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 9 No. 6 (2025) 1375 - 1384 e-ISSN: 2580-0760

Correlation Analysis of ISO 25010 Modularity, CK Metrics, and Architecture Smells

Maulana Alirridlo[1], Enrico Almer Tahara[2], Siti Rochimah[3]
[1,2,3]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

[1]6025241006@student.its.ac.id, [2]6025241062@student.its.ac.id, [3]siti@if.its.ac.id

Abstract

Open-source software projects face increasing challenges in maintaining design quality as they evolve, often resulting in technical debt accumulation and reduced maintainability. This study explores the relationship between software modularity, measured using ISO/IEC 25010 quality attributes, Chidamber and Kemerer (CK) object-oriented metrics, and architectural smells (AS) in Java-based open-source software. Six Java-based open-source projects were strategically selected based on varying complexity levels (ranging from 6-994 classes) and different application domains to ensure comprehensive analysis coverage using DesigniteJava to extract AS, CK metrics, and modularity indicators. Correlation analyses showed that architectural smells such as Cyclic Dependency, Ambiguous Interface, and God Component are strongly correlated with CK metrics like Weighted Methods per Class, Depth of Inheritance Tree, and Number of Children. These CK metrics also exhibited strong positive correlations with Cyclomatic Complexity, indicating that structurally complex components also tend to have more complex control logic. Dense Structure was found to negatively correlate with Coupling of Components Conformance, suggesting its effect on modularity compliance. On the other hand, smells like Feature Concentration and Scattered Functionality showed weak or inconsistent correlations with these metrics. The findings highlight the importance of addressing specific architectural smells to improve modularity and software quality.

Keywords: Architecture Smells; CK Metrics; ISO 25010; Pearson; Spearman

How to Cite: S. M. A. Mujahidin, M. A.Q. Riyadi, A. M. Dewi, and M. Kamal, "Correlation Analysis of ISO 25010 Modularity, CK Metrics, and Architecture Smells", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 9, no. 6, pp. 1375 - 1384, Dec. 2025.
Permalink/DOI: https://doi.org/10.29207/resti.v9i6.6825

Received: June 24, 2025
Accepted: October 18, 2025
Available Online: December 10, 2025

This is an open-access article under the CC BY 4.0 License
Published by Ikatan Ahli Informatika Indonesia

1. Introduction

In modern software development, failure to maintain design quality can have a serious impact on maintenance costs, release delays, and accumulation of technical debt. Therefore, design quality assurance is a fundamental aspect to ensure systems that are modular, maintainable, and adaptable to changing needs. Standards such as ISO/IEC 25010 [1] provide an evaluation framework that covers characteristics such as maintainability and modularity, but their practical application requires the support of quantitative metrics. Chidamber and Kemerer (CK) Metrics provide technical indicators of system complexity and cohesion [2], while the phenomenon of architectural smells indicates architectural patterns that deviate from good design principles [3]. The combination of quality metrics and architectural smells detection is a promising approach to maintain software design sustainability in the long term [4].

ISO/IEC 25010 defines eight main characteristics of software product quality, namely: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. Each of these characteristics has more specific sub-characteristics, for example maintainability includes modularity, reusability, analysability, modifiability, and testability. Using this standard, organisations can conduct a systematic assessment of whether the software has met functional requirements, performance efficiency, interoperability, ease of use, reliability, security, maintainability, and portability. ISO/IEC 25010-based software quality measurement is essential to ensure that software not only meets technical specifications, but also provides a good user experience and is easy to modify in the future [1], [5]. Structured quality evaluation helps organisations identify areas for improvement and reduces the risk of software failure [6]-[9].

Code quality measurement using metrics such as Chidamber and Kemerer Metrics (CK Metrics) is very important in object-oriented software development because these metrics provide quantitative indicators that can predict system maintainability and reliability [10], [11]. CK Metrics, which include metrics such as Weighted Methods per Class (WMC), Coupling Between Objects (CBO), Depth of Inheritance Tree (DIT), and Lack of Cohesion in Methods (LCOM), have been empirically proven to contribute to software design quality [4], [12]-[14]. For example, high WMC values can increase the risk of code errors, while excessive CBO indicates dependencies between classes that can complicate maintenance and testing. Previous studies show that keeping the values of these metrics within recommended limits can improve maintainability indices and reduce debug time, thus easing the overall software development and maintenance process [2], [15]-[17].

On the other hand, the phenomenon of architectural smells, i.e. undesirable architectural design patterns, negatively impacts software quality by increasing technical debt and complicating the system evolution process. Architectural smells operate at the architectural level, such as components and connectors, in contrast to code smells that focus on the implementation level. The presence of architectural smells such as cyclic dependencies and hub-like modularization has been shown to decrease modularity, reduce module cohesion, and increase testing complexity. Empirical research on several open-source and enterprise projects shows that removing architectural smells can reduce bug rates, speed up the refactoring process, and reduce maintenance costs [3], [18]-[20]. Moreover, there is a significant relationship between CK metrics and the occurrence of architectural smells, where non-ideal metric values can trigger harmful architectural anomalies [21]. Therefore, the integration of CK Metrics measurement with architectural smells detection is essential to improve design quality and software sustainability [4].

High-quality software development requires a deep understanding of the factors that influence design quality [22]. Modularity, CK metrics, and architectural smells are important aspects that affect software quality, but previous research has not comprehensively examined the relationship between these three aspects together [23]-[25]. Therefore, there is still a lack of clarity in understanding the dominant factors that influence software design. It is not yet known whether modularity is more influenced by CK Metrics or architectural smells, making it difficult to determine the main factors that should be prioritised in improving software quality.

To address this gap, a multi-correlation approach using methods such as Pearson and Spearman is needed [5]. This approach provides evidence-based validation of the relationships between architectural smells, ISO/IEC 25010-based modularity, and CK Metrics, offering concrete empirical findings that can guide software architecture decisions. By systematically analyzing these correlations, this research establishes a dominance hierarchy among quality factors, enabling organizations to prioritize improvement efforts based on data-driven insights rather than subjective assessments. The empirical evidence generated serves as a practical foundation for software architects and quality managers, while also providing baseline data for future advanced analytics and tool development in software engineering. Thus, this study contributes to the consolidation of software quality knowledge and supports evidence-based decision-making in high-quality software development practices.

2. Methods

This chapter describes the methodological approach used in the research to analyse the relationship between ISO 25010-based modularity, the CK (Chidamber and Kemerer) metric, and Architecture Smells (AS) in object-oriented software. This research aims to understand the extent to which the three elements are interrelated in the context of software quality, particularly in open source software projects developed using the Java programming language.

The research flowchart shown in Figure 1 illustrates the main stages carried out in this study, starting from the data input process, metrics extraction, to the calculation of correlations between the analysed variables.

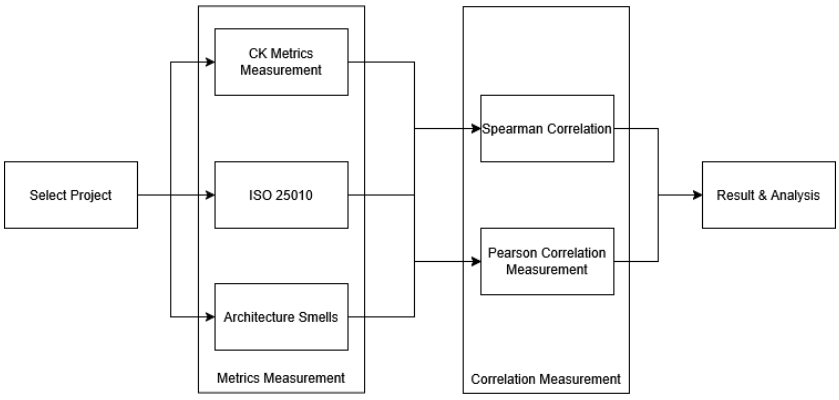


Figure 1. Research Flow

The input data was obtained from open-source software projects that were selected based on certain criteria, such as project size and class structure, to ensure that the complexity and scale of the projects matched the measurement needs. The projects were then analysed for modularity using an approach based on the ISO 25010 standard, the internal quality of the software was measured using six CK metrics (i.e. WMC, DIT, NOC, CBO, RFC, and LCOM), and the presence of Architecture Smells was analysed using Sharma et al.'s approach, which includes categories such as Cyclic Dependency, Unstable Dependency, Ambiguous Interface, God Component, Feature Concentration, Scattered Functionality, and Dense Structure.

The entire analysis process was carried out systematically with the support of tools such as DesigniteJava to detect Architecture Smells, as well as manual and semi-automated calculations to obtain ISO 25010 CK and modularity metric values. The results of each stage were used as the basis for correlation calculations using Pearson and Spearman statistical methods, to determine the strength and direction of the relationship between variables in the context of software quality.

2.1 System Dataset Selection

The first step in this research started with the selection of open-source projects publicly available on GitHub as objects of observation to analyse the correlation between Architecture Smells (AS), CK metrics, and ISO 25010-based modularity. All projects used were developed using the Java programming language and were selected based on their diversity in size and complexity to provide a broader scope for analysis.

There were six projects evaluated, each labelled from S1 to S6. Details of the projects can be seen in Table 1. Project S1 is a school management system, while S2 is a hospital management application. Project S3 is a pet hotel management system. S4 is a hotel booking application titled ‘I Want to Travel’. S5 focuses on an appointment booking system, while S6 is a private tutoring booking application called Huna. These projects were chosen because they have variations in the number of lines of code and the number of classes, which is expected to provide more representative correlation results.

Table 1. Selected Project and Parameter Overview

Project Label	Project Name	Line of Code	Number of Classes
S1	School Management	156	6
S2	Hospital Management	2291	50
S3	Pet Hotel	5662	105
S4	I want to travel (hotel booking app)	3801	138
S5	Appointment Booking	28310	524
S6	Huna (tutoring booking app)	38224	994

Once the projects were selected, the Architecture Smells detection process was performed using DesigniteJava software, which was able to identify various indicators of problematic architecture in Java-based software projects.

2.2 Metrics Measurement

Once a software project has been selected, the next step in this research is to measure three main elements: Architecture Smells (AS), CK metrics, and modularity based on ISO 25010. The entire measurement process was conducted using the DesigniteJava tool, with adjustments to the calculations to ensure all values were at the same level, i.e. per class, to facilitate valid correlation analysis [26].

The Architecture Smells measurement was conducted using DesigniteJava to analyse the structure of Java-based projects. The tool automatically detects various types of AS that reflect violations of architectural design principles, namely God Component, Cyclic Dependency, Feature Concentration, Unstable Dependency, Ambiguous Interface, Scattered Functionality, and Dense Structure. The output result is the number of smells per category in the entire project. To obtain the AS density value per class, the total number of detected ASes is divided by the total number of classes in the project.

CK Metrics measurements were also performed using DesigniteJava, which resulted in WMC, DIT, NOC, CBO, RFC, and LCOM values for each class in the project. The values of each of these metrics were then summed over all classes, and then divided by the total number of classes to produce the average CK metrics per class.

The measurement of modularity based on the ISO 25010 standard in this study focuses on two parameters: Coupling of Components Conformance (CCC) and Cyclomatic Complexity (CC). CCC is calculated based on the ratio between the number of classes with fan-out = 0 to the total number of classes in the project. For CC, the cyclomatic complexity value is taken from each method detected by DesigniteJava, then summed and divided by the total number of methods. The result is then divided by the number of classes to obtain the average CC per class. This approach is done so that all metric values can be compared equally at the same unit of analysis level, which is the class.

With this normalisation, the AS, CK, and ISO 25010 modularity metrics can be analysed in parallel to

identify relationships and influences between variables in the context of software quality.

2.3. Correlation Measurement

In this stage, correlation analyses were conducted between ISO 25010-based modularity metrics, CK metrics, and smell architecture to see the relationship between each pair of metrics. The two correlation approaches used are Pearson and Spearman. Pearson analysis is used to measure linear relationships between metrics assuming normally distributed data, while Spearman is used to look at monotonic relationships without assuming a particular data distribution [27]. Both approaches were applied in parallel to each pair of metrics to obtain a more comprehensive picture of the relationship. Correlation analyses were performed in pairs with schemes: smell architecture against CK metrics, CK metrics against ISO 25010 modularity metrics, and modularity metrics against smell architecture. The results of these correlation measurements were then analysed to identify patterns of interrelationships between metrics and evaluate the extent to which modularity and architectural quality can be mapped through the metrics used.

3. Results and Discussions

In this section, the experiment was conducted to analyze the correlations between CK Metrics, ISO Metrics, and Architecture Smells across six open-source software projects. The primary goal was to determine how these metrics relate to one another and identify any statistically significant correlations that might inform better development and quality-assurance practices.

The study involved selecting six open-source software projects and gathering relevant data metrics for analysis. CK Metrics, which assess structural properties of the code, such as coupling, cohesion, and complexity. ISO Metrics, on the other hand, provided a standardized evaluation of software quality. Additionally, Architecture Smells served as indicators of potential issues within the software architecture.

To analyze the relationships among these metrics, both Pearson and Spearman correlation analyses were employed. The Pearson correlation analysis focused on measuring linear relationships, providing insights into the strength and direction of correlations. In contrast, the Spearman correlation analysis was utilized to assess monotonic relationships, rendering it suitable for non-parametric data. The employed combination of both methods enabled a comprehensive examination of both linear and non-linear dependencies among the metrics.

3.1 Metrics Measurement

A total of 13 metrics were used in this study to evaluate software quality. The metrics include three groups: 7 architecture smell metrics, 4 metrics from CK metrics, and 2 metrics based on ISO/IEC 25010. The values of each metric have been averaged based on the number of classes in each sample (S1-S6) to enable balanced and relevant comparisons between samples. The measurement results are shown in Table 2.

Table 2. Metrics Measurement Result

Metrics	S1	S2	S3	S4	S5	S6
AS						
CD	0	0	0.00952	0	0.04962	0
UD	0	0	0.02857	0	0.02099	0
AI	0	0	0	0	0.00191	0
GC	0	0	0	0	0.00954	0
FC	0	0.0566	0.0381	0.03623	0.02863	0.04125
SF	0	0	0	0.02899	0.01336	0
DS	0	0.01887	0.00952	0	0.00191	0
CK						
WMC	4.16667	4.60377	6.30476	5.06522	15.063	0.16097
DIT	0	0	0.07619	0.00725	0.8626	0
LCOM	1.11E+15	7.46E+14	8.16E+13	1.03E+15	2.39E+15	-9.175
NOC	0	0	0.07619	0.00725	0.45802	0
ISO 25010						
CCC	1	0.13208	0.3619	0.49275	0.20611	1
CC	4.16667	4.60377	6.30476	5.06522	15.063	0.16097

The architecture smell metric showed that S3 and S5 contained the most indications of architectural problems compared to the other samples, especially in terms of Cyclic Dependency (CD), Unstable Dependency (UD), and Feature Concentration (FC). S5 in particular has the highest value for CD, which indicates communication between components that should be independent. In addition, the presence of Scattered Functionality (SF) and Dense Structure (DS) in some samples also indicates a non-optimal component structure.

On the CK metric, Weighted Methods per Class (WMC) in S5 indicates high method complexity, while the Depth of Inheritance Tree (DIT) and Number of Children (NOC) values indicate the level of inheritance usage in the system, where S5 also has the highest value, indicating the potential utilization of reuse through inheritance. Lack of Cohesion in Methods (LCOM) shows extreme and unnatural values in some samples, particularly S1 and S5, which most likely indicates calculations that need to be reviewed or the presence of very large structural anomalies in class cohesion.

In terms of ISO 25010, Coupling of Components Conformance (CCC) tends to be high in S1 and S6, indicating good modularity compliance. In contrast, the Cyclomatic Complexity (CC) values are negative in all samples except S6, which needs to be reconfirmed - it could be a result of normalization against a baseline or an indicator of deviation from ideal complexity.

Overall, it appears that S5 has the most significant level of complexity and architectural smell, followed by S3, while S1 and S6 tend to be cleaner of design problem indicators. With the data averaged per class, further correlation analysis can provide insight into the relationship between these metrics and their impact on software quality in general.

3.2 Correlation Measurement

To understand the relationship between the metrics used, Pearson correlation analysis was performed by comparing three groups of metrics: Architecture Smell (AS) with CK Metrics, CK Metrics with ISO 25010, and Architecture Smell with ISO 25010.

Figure 2 presents a heatmap showing the Pearson correlation between Architecture Smells (AS) and CK metrics, revealing that certain smells have strong, statistically significant relationships with class complexity measures. Cyclic Dependency (CD), for instance, exhibits very high positive correlations with WMC (r = 0.934), DIT (r = 0.994), and NOC (r = 0.999), suggesting a strong link between cyclical dependencies and complex inheritance structures. Similarly, Ambiguous Interface (AI) and God Component (GC) show nearly identical patterns, with strong correlations to DIT (r = 0.996), NOC (r = 0.987), WMC (r = 0.908), and LCOM (r = 0.843), indicating that systems with these smells often feature deep inheritance hierarchies and complex class designs. In contrast, Feature Concentration (FC), Scattered Functionality (SF), and Dense Structure (DS) display weak correlations and high p-values, implying they are likely influenced by factors beyond the scope of CK metrics.

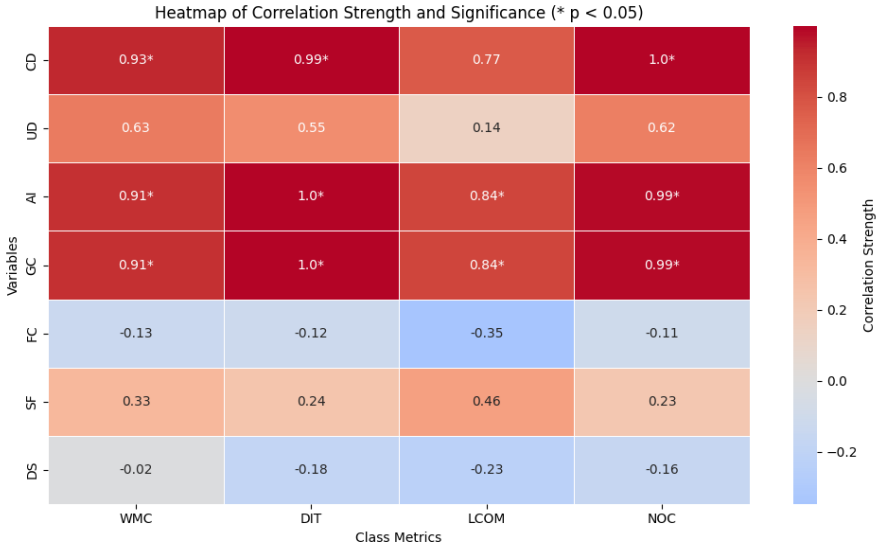


Figure 2. Pearson Correlation between AS and CK Metrics

Figure 3 shows the Pearson correlation between CK Metrics and ISO 25010, highlighting a very strong and significant positive correlation between WMC and Cyclomatic Complexity (CC) (r = 1.000, p = 0.000), suggesting that classes with more or more complex methods significantly increase system complexity. Similar strong correlations are observed for DIT (r = 0.927) and NOC (r = 0.937), indicating that deeper inheritance and more child classes contribute to higher logical complexity. LCOM also correlates positively with CC (r = 0.843), reinforcing the link between low cohesion and increased complexity. In contrast, CBO shows a weak, statistically insignificant negative correlation with CC (r = -0.633, p = 0.178), suggesting it may be less useful for assessing complexity in this context.

Figure 4 presents the Pearson correlation between Architecture Smells (AS) and ISO 25010, revealing that certain smells, Cyclic Dependency (CD), Architectural Inconsistency (AI), and God Class (GC), have strong, statistically significant positive correlations with Cyclomatic Complexity (CC) ($r \geq 0.910$, $p < 0.05$). This indicates that these smells are closely associated with increased logical complexity, reduced maintainability, and higher error risk, aligning with ISO 25010's complexity-related quality concerns. In contrast, correlations between smells and Coupling of Components Conformance (CCC) are generally weak and not statistically significant; for example, Dense Structure (DS) shows a moderate negative correlation with CCC (r = -0.663, p = 0.152), while smells like Feature Concentration (FC) and Scattered Functionality (SF) show minimal or inconsistent relationships. These results suggest that while some architecture smells strongly impact system complexity, others may have negligible or unrelated effects.

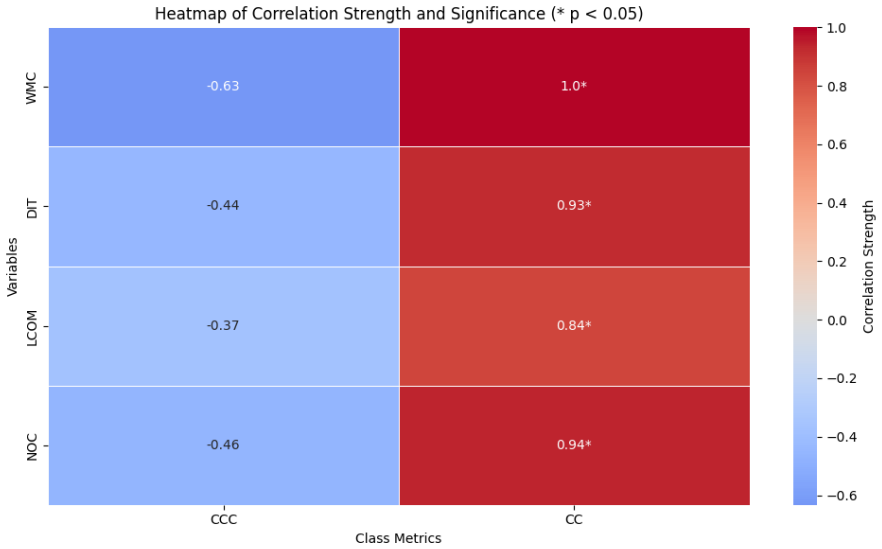


Figure 3. Pearson Correlation between CK Metrics and ISO 25010

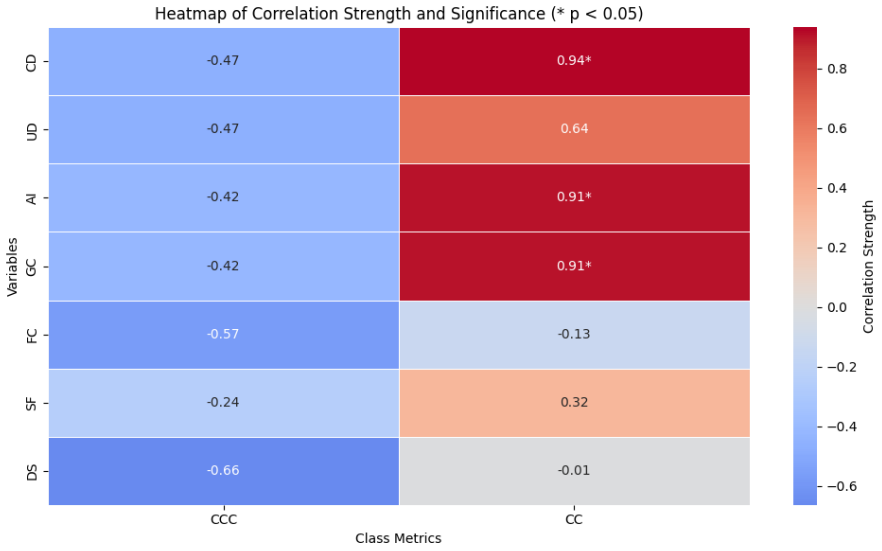


Figure 4. Pearson Correlation between AS and ISO 25010

Figure 5 shows the Spearman correlation between Architecture Smells (AS) and CK Metrics, revealing several significant relationships. Cyclic Dependency (CD) strongly correlates with WMC (r = 0.845, p = 0.034), DIT (r = 0.898, p = 0.015), and NOC (r = 0.898, p = 0.015), suggesting that complex inheritance structures and higher method counts increase the likelihood of dependency cycles. Unstable Dependency (UD) also shows significant correlations with DIT and NOC (r = 0.826, p = 0.043), indicating that instability in dependencies is linked to deep inheritance and class reuse. Ambiguous Interface (AI) and God Component (GC) show moderate but statistically insignificant correlations with CK metrics. Feature Concentration (FC) has a high negative correlation with LCOM (r = -0.771, p = 0.072), suggesting a potential link to increased cohesion. Other smells, such as Scattered Functionality (SF) and Dense Structure (DS), show no significant correlations with CK metrics.

Figure 6 shows the Spearman correlation between CK Metrics and ISO 25010, revealing strong, statistically significant positive correlations between Cyclomatic Complexity (CC) and WMC (r = 1.000), DIT (r = 0.941, p = 0.005), and NOC (r = 0.941, p = 0.005). This suggests that increased method count, inheritance depth, and subclassing contribute to higher logical complexity, reflecting more control paths and decision points in the system. In contrast, the correlation between CK metrics and Coupling of Components Conformance (CCC) is weak and statistically insignificant (e.g., WMC: r = -0.638, p = 0.173), indicating that CK metrics may not effectively capture modularity adherence, which may require more specialized architectural measures.

Figure 7 presents the Spearman correlation between Architecture Smells (AS) and ISO 25010, highlighting that Cyclic Dependency (CD) has a significant positive correlation with Cyclomatic Complexity (CC) (r = 0.845, p = 0.034), suggesting that architectural cycles

contribute to increased logical complexity. Additionally, Dense Structure (DS) shows a strong negative correlation with Coupling of Components Conformance (CCC) (r = -0.893, p = 0.016), indicating that dense interconnections undermine modularity and cohesion. These findings reinforce earlier Pearson-based results, emphasizing CD and DS as key indicators of complexity and modularity issues, while other smells like Scattered Functionality (SF) and Feature Concentration (FC) show weaker or negligible correlations, and may be less critical for assessing overall software quality.

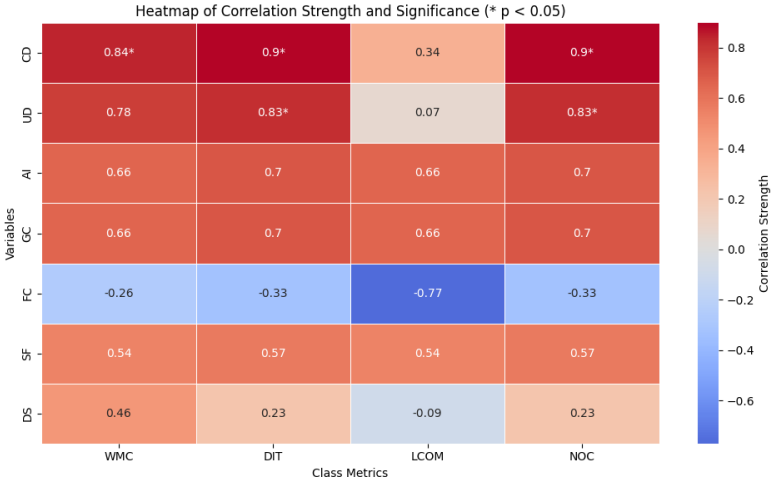


Figure 5. Spearman Correlation between AS and CK Metrics

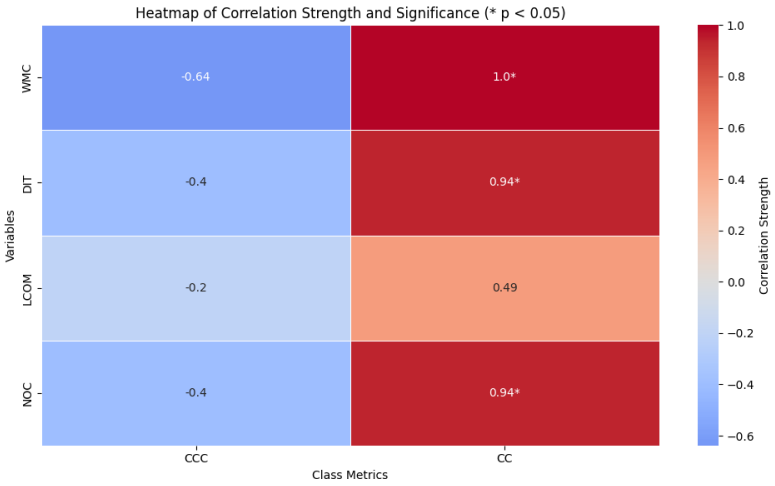


Figure 6. Spearman Correlation between CK Metrics and ISO 25010

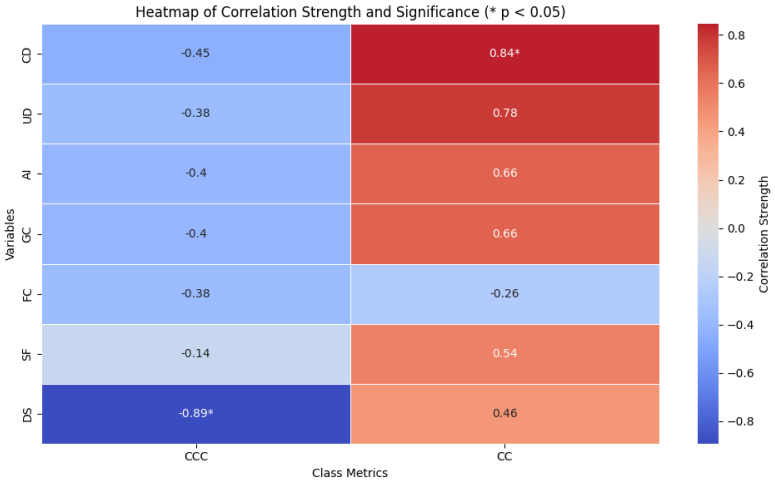


Figure 7. Spearman Correlation between AS and ISO 25010

3.3 Relation Classification Discussion

Figures 2-7 summarize the Pearson and Spearman correlation results between Architecture Smell (AS), CK metrics, and ISO 25010 modularity aspects (CCC and CC). These correlations help identify which metrics have strong, weak, direct, or inverse relationships and validate the consistency between both correlation methods. The correlation analyses between Architecture Smells (AS), CK metrics, and ISO 25010 modularity aspects reveal consistent patterns across both Pearson and Spearman methods, which can be categorized into distinct relationship types based on their strength and consistency.

Strong Structure-Complexity Relationships emerge as the most significant finding, particularly involving three high-impact architectural smells that consistently demonstrate robust correlations with structural complexity metrics. Cyclic Dependency (CD) shows exceptionally strong correlations with Weighted Methods per Class (WMC), Depth of Inheritance Tree (DIT), and Number of Children (NOC) across both correlation methods, with Pearson values ranging from 0.934 to 0.999 and Spearman values from 0.845 to 0.898. This consistency confirms that systems with dependency cycles tend to exhibit higher structural complexity through deeper inheritance hierarchies and more complex class designs. Similarly, Ambiguous Interface (AI) and God Component (GC) display nearly identical correlation patterns with structural metrics, particularly showing strong relationships with DIT and NOC, indicating that these smells commonly emerge in architectures with complex inheritance structures and oversized components.

The relationship between CK metrics and logical complexity presents another crucial finding, where three key structural metrics—WMC, DIT, and NOC—consistently predict increases in Cyclomatic Complexity (CC). These metrics show remarkably strong positive correlations with CC in both Pearson (ranging from 0.927 to 1.000) and Spearman (0.941 to 1.000) analyses, demonstrating that structural complexity and logical complexity tend to increase together rather than inversely. This finding contradicts the initial assumption of an inverse relationship and instead supports the principle that architecturally complex systems also tend to have more complex control logic, creating a compounding effect on overall system complexity.

Modularity-specific relationships reveal that Dense Structure (DS) operates differently from other architectural smells, primarily affecting component coupling rather than logical complexity. DS shows a significant negative correlation with Coupling of Components Conformance (CCC) in Spearman analysis ($r = -0.893$, $p = 0.016$), while exhibiting weak correlations with CC. This pattern suggests that dense interconnections between components primarily undermine modularity compliance and system cohesion, making DS a specialized indicator for architectural modularity issues rather than a general complexity predictor.

Weak or inconsistent relationships characterize several metrics that show limited correlation strength or conflicting results between correlation methods. Feature Concentration (FC) and Scattered Functionality (SF) consistently exhibit weak and statistically insignificant correlations across all analyses, suggesting these smells may reflect architectural design concerns that cannot be effectively captured through class-level CK metrics or ISO 25010 modularity criteria. Additionally, Lack of Cohesion in Methods (LCOM) demonstrates inconsistent behavior, showing strong positive correlation with CC in Pearson analysis ($r = 0.843$, $p = 0.035$) but much weaker correlation in Spearman analysis ($r = 0.486$, $p = 0.329$), indicating potential non-linear relationships or context-dependent interactions between class cohesion and control complexity.

Method consistency analysis reveals high agreement between Pearson and Spearman approaches for the most critical relationships, particularly those involving CD, AI, GC with structural metrics, and the relationship between key CK metrics and CC. This consistency validates the robustness of these findings across different statistical assumptions. However, discrepancies observed in LCOM correlations and some DS relationships highlight the importance of considering both linear and non-linear dependencies when interpreting software quality metrics.

In summary, this study followed a structured workflow that began with selecting six diverse open-source Java projects, extracting CK metrics, ISO 25010 modularity indicators, and architectural smells, and then performing correlation analysis using both Pearson and Spearman methods. The intended outcome of this process was to clarify the interplay between structural metrics, modularity, and design anomalies. The results confirmed that certain architectural smells, particularly Cyclic Dependency, Ambiguous Interface, and God Component, are strongly correlated with higher structural and logical complexity, while CK metrics such as WMC, DIT, and NOC consistently predict increases in cyclomatic complexity. At the same time, Dense Structure was found to undermine modularity compliance, whereas Feature Concentration and Scattered Functionality showed limited measurable effects. These findings establish practical priorities for software quality management, where development teams should focus on detecting and addressing the most critical smells while using key CK metrics as early warning indicators. Preventive measures that target structural design issues can therefore reduce overall system complexity, improve maintainability, and help manage technical debt more effectively.

4. Conclusions

Overall, the discussion highlighted that software modularity, structural complexity, and architectural

smells are deeply interconnected. The correlations observed across multiple projects reveal consistent patterns where certain metrics and smells dominate in shaping system quality. This provides the foundation for a comprehensive understanding of the relationship between software quality metrics and architectural smells in object-oriented systems, with significant practical implications for software development.

Analysis of six Java open-source projects revealed that structural complexity metrics such as Weighted Methods per Class (WMC), Depth of Inheritance Tree (DIT), and Number of Children (NOC) can serve as reliable early indicators for predicting increases in system logical complexity. These findings enable development teams to identify potential architectural issues before they become critical, allowing for more effective and efficient preventive measures.

More importantly, this study identified three architectural smells that have the most destructive impact on system quality: Cyclic Dependency (CD), Ambiguous Interface (AI), and God Component (GC). These three smells are not only strongly correlated with a decrease in modularity, but also have a cascading effect that increases the overall complexity of the system. Conversely, smells such as Feature Concentration (FC), Scattered Functionality (SF), and Dense Structure (DS) exhibit more limited and inconsistent impacts. Understanding this hierarchy of priorities enables developers and software architects to allocate code improvement resources more strategically, focusing on issues that provide the highest return on investment in terms of improved maintainability and system quality.

The practical contribution of this research lies in providing a framework that can be directly applied in everyday software development practices. Development teams can use a combination of CK metrics and architectural smell detection as an early warning system, enabling them to make more informed architectural decisions and perform more measurable maintenance. This not only improves short-term software quality but also reduces technical debt and extends the lifecycle of software systems.

For future research, it is necessary to expand the scope of analysis to more diverse programming languages and application domains to enhance the generalizability of the findings. Longitudinal studies observing the evolution of metrics and architectural smells over time will provide valuable insights into patterns of architectural decay and software aging. Additionally, integration with runtime attributes such as performance and fault-proneness, along with the development of predictive models based on machine learning, will open opportunities for more proactive and automated systems for detecting and preventing architectural smells.

Acknowledgements

The author declares no conflict of interest and received no funding. Thanks to Institut Teknologi Sepuluh Nopember (ITS) for facilities and supportive atmosphere.

References

[1] S. Wagner, *ISO/IEC 25010*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2023. doi: https://doi.org/10.1007/978-3-642-38571-1.

[2] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, Jun. 1994, doi: 10.1109/32.295895.

[3] F. A. Fontanaa, V. Lenarduzzi, R. Roveda, and D. Taibi, “Are Architectural Smells Independent from Code Smells? An Empirical Study,” Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.11755

[4] I. M. A. Wikantyasa, A. P. Kurniawan, and S. Rochimah, “C K Metric and Architecture Smells Relations: Towards Software Quality Assurance,” in *2023 14th International Conference on Information and Communication Technology and System, ICTS 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 13–17. doi: 10.1109/ICTS58770.2023.10330874.

[5] D. Yuniasri, S. Rochimah, and A. B. Raharjo, “A Correlation Analysis between ISO 25010 based Modularity and CK Metrics in Object-Oriented Software,” in *3rd International Conference on Advanced Science and Engineering, ICOASE 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 80–85. doi: 10.1109/ICOASE51841.2020.9436617.

[6] M. I. F. Nuzula and S. Rochimah, “Evaluation of Service Quality in Human Resource Information Systems Using the ISO/IEC 25010,” in *2023 International Seminar on Application for Technology of Information and Communication: Smart Technology Based on Industry 4.0: A New Way of Recovery from Global Pandemic and Global Economic Crisis, iSemantic 2023*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 215–220. doi: 10.1109/iSemantic59612.2023.10295365.

[7] F. Zahid, X. Chen, S. Sohail, B. Li, and M. Po-Leen Ooi, “Systematic Mapping Study to Assess Security Landscape for IoT-based Smart Farming Systems,” *Comput Secur*, p. 104790, Dec. 2025, doi: 10.1016/j.cose.2025.104790.

[8] J. Oviedo, M. Rodriguez, A. Trenta, D. Cannas, D. Natale, and M. Piattini, “ISO/IEC quality standards for AI engineering,” Nov. 01, 2024, *Elsevier Ireland Ltd*. doi: 10.1016/j.cosrev.2024.100681.

[9] C. H. Cai, J. Sun, and G. Dobbie, “B model quality assessments on automated reachability repair with ISO/IEC 25010,” *Sci Comput Program*, vol. 214, Feb. 2022, doi: 10.1016/j.scico.2021.102732.

[10] Y. A. Khan, M. O. Elish, and M. El-Attar, “A systematic review on the impact of CK metrics on the functional correctness of object-oriented classes,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, pp. 258–273. doi: 10.1007/978-3-642-31128-4_19.

[11] G. Singh, “Metrics for measuring the quality of object-oriented software,” *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 5, pp. 1–5, Aug. 2013, doi: 10.1145/2507288.2507311.

[12] L. A. Tremasova, V. M. Korovushkina, K. A. Panteleeva, and D. V. Gadasin, “Distributed Information System Software Code Reliability Evaluation,” in *Wave Electronics and its Application in Information and Telecommunication Systems, WECONF - Conference Proceedings*, Institute of Electrical and Electronics Engineers, 2024. doi: 10.1109/WECONF61770.2024.10564614.

[13] R. Jangra, O. P. Sangwan, and D. Nandal, “A holistic review on optimization strategies for C & K metrics in object oriented testing,” in *Proceedings - 2021 4th*

International Conference on Computational Intelligence and Communication Technologies, CCICT 2021, Institute of Electrical and Electronics Engineers Inc., Jul. 2021, pp. 362–371. doi: 10.1109/CCICT53244.2021.00073.

[14] R. Jangra, O. P. Sangwan, and D. Nandal, "A Novel Approach for Software Effort Estimation using Optimized C&K Metrics," in *Proceedings - 2022 5th International Conference on Computational Intelligence and Communication Technologies, CCICT 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 505–513. doi: 10.1109/CCiCT56684.2022.00095.

[15] A. Kurmangali, M. E. Rana, and W. N. W. Ab Rahman, "Impact of Abstract Factory and Decorator Design Patterns on Software Maintainability: Empirical Evaluation using CK Metrics," in *2022 International Conference on Decision Aid Sciences and Applications (DASA)*, IEEE, Mar. 2022, pp. 517–522. doi: 10.1109/DASA54658.2022.9765083.

[16] S. Prykhodko, "Evaluating Quality of Software Systems by the Confidence and Prediction Intervals of Regressions for RFC, CBO and WMC Metrics," *WSEAS Transactions on Systems*, vol. 23, pp. 322–330, 2024, doi: 10.37394/23202.2024.23.36.

[17] L. Ardito, R. Coppola, L. Barbato, and D. Verga, "A Tool-Based Perspective on Software Code Maintainability Metrics: A Systematic Literature Review," 2020, *Hindawi Limited*. doi: 10.1155/2020/8840389.

[18] P. Bacchiega, D. Rusconi, P. Mereghetti, and F. A. Fontana, "Refactoring of a Microservices Project Driven by Architectural Smell Detection," in *Proceedings - IEEE 21st International Conference on Software Architecture Companion, ICSA-C 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 281–288. doi: 10.1109/ICSA-C63560.2024.00040.

[19] M. Benidris, E. A. O. Elfallah, M. A. Cheema, M. S. A. Alsma, T. Z. Kadiki, and A. Elbarsha, "The Correlation Between Architecture Smells, Package Stability and Reusability: A Case Study," in *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering, MI-STA 2024 - Proceeding*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 1–6. doi: 10.1109/MI-STA61267.2024.10599745.

[20] P. Gnoyke, S. Schulze, and J. Kruger, "An Evolutionary Analysis of Software-Architecture Smells," in *Proceedings - 2021 IEEE International Conference on Software Maintenance and Evolution, ICSME 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 413–424. doi: 10.1109/ICSME52107.2021.00043.

[21] M. Esposito, M. Robredo, F. A. Fontana, and V. Lenarduzzi, "On the correlation between Architectural Smells and Static Analysis Warnings," Sep. 2025, doi: 10.1007/s11219-025-09730-7.

[22] U. Iftikhar, N. Bin Ali, J. Börstler, and M. Usman, "A tertiary study on links between source code metrics and external quality attributes," Jan. 01, 2024, *Elsevier B.V.* doi: 10.1016/j.infsof.2023.107348.

[23] J. Lenhard, M. Blom, and S. Herold, "Exploring the suitability of source code metrics for indicating architectural inconsistencies," *Software Quality Journal*, vol. 27, no. 1, pp. 241–274, Mar. 2019, doi: 10.1007/s11219-018-9404-z.

[24] K. R. Sampath and D. I. De Silva, "Cognitive-Integrated Complexity Analysis: A Comprehensive Software Visualization Tool," in *Proceedings of ICITR 2024 - 9th International Conference on Information Technology Research: Harnessing the Potential of Information Technology*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ICITR64794.2024.10857757.

[25] R. Jolak, S. Karlsson, and F. Dobslaw, "An empirical investigation of the impact of architectural smells on software maintainability," *Journal of Systems and Software*, vol. 225, Jul. 2025, doi: 10.1016/j.jss.2025.112382.

[26] H. Nandani, M. Saad, and T. Sharma, "DACOS—A Manually Annotated Dataset of Code Smells," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, IEEE, May 2023, pp. 446–450. doi: 10.1109/MSR59073.2023.00067.

[27] S. Pires and C. Mascarenhas, "Cyber Threat Analysis Using Pearson and Spearman Correlation Via Exploratory Data Analysis," in *ICSCCC 2023 - 3rd International Conference on Secure Cyber Computing and Communications*, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 257–262. doi: 10.1109/ICSCCC58608.2023.10176973.