Design, development, and evaluation of a Java-centric software solution for advanced enterprise financial reporting

Hrybniuk Volodymyr¹, Kopp Andrii²

¹ Second year Master’s Student; National Technical University «Kharkiv Polytechnic Institute»; Ukraine
² Ph.D. in Computer Science, Associate Professor, Associate Professor of the Department of Software Engineering and Management Intelligent Technology; National Technical University «Kharkiv Polytechnic Institute»; Ukraine

Abstract. This research delves into the intricacies of crafting an advanced software solution, with Java at its core, aimed at improving financial reporting for modern enterprises. Through a systematic approach encompassing design, development, and evaluation phases, the study emphasizes the unparalleled capabilities of Java in ensuring scalability, user experience, data integrity, and security. The outcome reveals a Java-driven software that not only meets contemporary reporting demands but also showcases potential for future integrations and innovations in the dynamic financial landscape.

Keywords: Java, Financial Reporting, Enterprise Software, ReactJS, Software Development, Data Integrity, Scalability, User Experience, Security
**Introduction.** Financial reporting serves as the backbone of corporate transparency, facilitating stakeholders – from investors to regulatory bodies – in making informed decisions based on a company’s fiscal health. At the heart of business ecosystems, these reports often determine a company’s attractiveness for investments, its creditworthiness, and even its strategic directions. Consequently, the accuracy, timeliness, and clarity of these reports are of paramount importance. Since the dawn of commerce, financial reporting has served as the bedrock for enterprise decision-making. By offering a clear snapshot of an organization’s financial health, these reports facilitate strategic planning, risk management, and stakeholder communication. Within larger enterprises, the complexity of this reporting multiplies, encompassing numerous transactions, diverse financial instruments, and intricate regulatory compliance requirements.

The sheer intricacy of modern financial operations underscores the need for automation. The manual approach, prevalent in the earlier days of business, not only consumed an inordinate amount of time but was also riddled with potential inaccuracies. The evolution of technology ushered in the era of automation, transforming the landscape of financial reporting. From basic spreadsheet automations to sophisticated software suites, the journey has been remarkable. Within this technological renaissance, Java has positioned itself as an indispensable ally [1]. Its platform-agnostic nature, coupled with its robust architecture, makes it an ideal candidate for crafting financial software solutions. Java’s inherent security features ensure data integrity, while its vast library ecosystem facilitates seamless integrations and feature enhancements [2].

A survey of the current market reveals an array of software solutions, each with its own merits and demerits. While some have adeptly harnessed Java’s capabilities, others have only skimmed the surface, leading to solutions that might falter in scalability, integration, or user experience. These gaps underscore the need for a more holistic, Java-centric approach. Given this backdrop, our study embarks on a journey to weave Java’s strengths into a financial reporting software,
aspiring to bridge existing gaps and cater to the nuanced needs of modern enterprises. We explore design considerations, development methods and evaluation frameworks, with Java as our guiding principle.

**Problem Statement.** In embarking on this research journey, software constraints are both clear and ambitious, centered around the Java capabilities:

1. Design Efficiency: To harness Java’s versatile architecture and libraries, aiming to craft a software solution that is both robust and adaptable to the dynamic landscape of financial reporting.

2. User Experience: Utilizing ReactJS [3], our goal was to curate an intuitive and responsive user interface, ensuring that users can effortlessly navigate and leverage the software’s features.

3. Data Integrity: With Java’s renowned data processing prowess, our objective was to guarantee swift and accurate financial data management, thereby producing reliable reports in real-time.

4. Scalability: Leveraging Java’s vast ecosystem, we aimed to develop software that can seamlessly integrate with various enterprise systems, while also being scalable to accommodate future growth and enhancements.

5. Security: Given the sensitive nature of financial data, employing Java’s robust security frameworks was a priority, ensuring that data remains protected against potential threats.

In essence, these constraints, underpinned by Java, shaped our research direction, guiding our efforts towards developing a software solution aptly equipped for modern enterprise financial reporting challenges.

Hence, this study aims to answer the following two research questions:

- **RQ₁**: How the popularity of different programming languages suitable for the finance domain is connected with their performance rather than design efficiency, user experience, data integrity, scalability, and security?

- **RQ₂**: How to evaluate a designed and developed Java-centric software solution for advanced enterprise financial reporting in terms of its modularity, scalability, and
maintainability?

State-of-the-Art. The realm of financial software development is vast, with numerous tools and platforms vying for dominance. Through a review of the literature, certain trends and notable observations emerge, central among them being the prominence of Java. Based on structure and content, financial reporting can be segmented into various types [4]:

- By Structure: Reporting that presents information as of a specific date (containing point-in-time indicators) and that encompasses information over a specific (reporting) period (comprising interval indicators).
- By Content and Sources: Statistical, Financial, Tax, Specialized Reporting, and Internal Business (Managerial) Reporting.

Historical insights reveal a trend of software gravitating towards complexity. However, the universality and robustness of Java has consistently been recognized as a facilitator in managing this complexity, especially in the finance sector where precision is paramount. Numerous studies [4, 5] have emphasized Java’s scalability and security features as assets in developing financial systems.

Methodology. In any research endeavor, the chosen methodology dictates the path of inquiry, shaping the tools, techniques, and processes employed to derive results. This section delineates the systematic approach adopted for designing, developing, and evaluating the proposed financial software solution. By detailing the underpinning methods and tools, it provides a blueprint of the journey from conceptualization to realization, ensuring the study’s rigor and reproducibility.

1. Development Tools:
   - Platforms: Utilized platforms include IntelliJ IDEA for Java development [6] and AWS (Amazon Web Services) for backend infrastructure planning [7].
   - Programing Language: Java, with its robust libraries and extensive support, was the primary language selected for this project due to its adaptability, performance, and security capabilities.
   - Constraints: Emphasis was placed on creating an intuitive user interface, ensuring data integrity,
scalability, and maintaining a high-security standard.

2. Delivery Tools and Approaches:
   - Tools: The Maven build tool [8] was integrated for dependency management, while Git was employed for version control [9]. Jenkins, paired with Java, ensured continuous integration and delivery [10].
   - Approaches: An Agile software engineering methodology [11], tailored for Java development, was utilized, allowing for iterative progress, feedback incorporation, and flexibility to changing requirements.

3. Evaluation Criteria:
   - User satisfaction, measured via surveys and feedback sessions.
   - Data processing accuracy, ensuring Java’s inherent strengths are leveraged for error-free computations.
   - Speed of execution, with benchmarks set to industry standards.

4. Evaluation Methods:
   - User Testing: Sessions conducted to glean real-time user feedback.
   - Automated Testing: JUnit unit-testing framework [12] was employed to validate data processing accuracy in Java.
   - Performance Benchmarking: Comparing our software’s execution speed against prevailing industry solutions.

The proposed solution considers data-centric architecture instead of product-centric architecture. Enterprise Architecture development process emphasizes collecting accurate data with qualified structure and recognizing decision makers’ and stakeholders’ needs and purpose of architecture that helps to create “fit for purpose” views and analyses as architecture results [3].

Therefore, focusing on the technical side of the developed software solution, we need to test the following null and alternative hypotheses in this study:

- \( H_0 \): There is no correlation between the popularity of programming languages used for financial applications and their performance.
- \( H_1 \): There is a significant correlation between the popularity of programming languages used for financial applications and their performance.
Let us rank the most popular programming languages according to their popularity for finance and fintech [13] and performance [14]. Their rankings are shown in Table 1.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Python</th>
<th>Java</th>
<th>JavaScript</th>
<th>Scala</th>
<th>C++</th>
<th>C#</th>
<th>Ruby</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popularity</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Performance</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

By applying the regression analysis [15], we have obtained the high $p$-value of 0.70 ($\geq 0.05$), which indicates we can accept the null hypothesis $H_0$ and assume that there is no significant correlation between the popularity of programming languages used for financial applications and their performance. Hence, appropriate tools can be employed for the designated domain to ensure efficient design, optimal user experience, reliable data integrity, scalability, and crucial security measures for financial reporting.

**Software Design and Features.** In the realm of financial reporting, the software’s design and feature set dictate its efficacy, user acceptance, and overall impact. While the backend operations ensure data integrity and accuracy, it’s the interface and functionalities that users directly interact with. Furthermore, the security architecture underpinning these features is paramount, given the sensitive nature of financial data. This section looks at the design considerations and features of our software, emphasizing the central role of Java in shaping its design and ensuring its ability to meet business needs.

The UI (User Interface), developed primarily using ReactJS library [3], stands as the gateway between the software and its users. Given that financial reporting tools are often navigated by both tech-savvy and non-technical users, an intuitive UI is vital.

The software is laden with features tailored for its target audience:

- **Dynamic Reporting:** Built on Java’s efficient data processing capabilities, users can generate diverse financial reports in real-time.
INFORMATION AND WEB TECHNOLOGIES

- Customizable Dashboards: Using ReactJS [3], dashboards can be customized, enabling users to view metrics most pertinent to their roles and responsibilities.
- Data Analytics: The software integrates Java-based analytics tools, aiding businesses in drawing meaningful insights from raw data.
- Integration Capabilities: Built to be interoperable, it can seamlessly connect with other enterprise systems, maximizing data utility.

For the enterprise financial sector, where accuracy and timeliness are paramount, these Java-centric features ensure that the software isn’t just a reporting tool, but a comprehensive financial assistant.

Development and Implementation. For addressing the task at hand, a client-server architecture was chosen. The client-server architecture is one of the software architectural patterns and is the dominant concept in creating distributed network applications, anticipating interaction and data exchange between them.

It consists of the following main components:
- A set of servers that provide information or other services to the clients.
- A set of clients that use services provided by servers.
- A network that facilitates interaction between clients and servers.

The chosen architecture (Fig. 1) embodies a well-thought-out design strategy that emphasizes modularity, scalability, and maintainability.

The general structure of a Java-based software solution for generating enterprise financial reports includes:
- Controllers: Process HTTP requests and interact with the user.
- Services: contain business logic for processing financial data and generating reports.
- Repository: provides access to the database and perform data operations.
- Models: represent financial objects and reports.
- Utilities: contain helper classes and utilities for connecting to the database.
- Configuration: includes the main application class
(e.g., “Application”) and the configuration file (“application.properties” or “application.yml”), which specifies the database connection settings and other settings.

![General backend architecture](image)

- Audit: audits financial transactions and logs events.
- RegularDataUpdate: provides automatic updating of financial data from other sources or systems, includes methods for scheduling updates, retrieving data, updating the database, and handling errors.

By segregating the system into distinct components, each dedicated to a specific function, we ensure both flexibility and robustness. This architecture, designed with future expansion and adaptability in mind, positions the project to seamlessly accommodate evolving financial reporting needs. Furthermore, the clear delineation of roles between components simplifies troubleshooting, encourages iterative development, and fosters a sustainable framework for long-term system health. In a domain where precision, reliability, and adaptability are paramount, this architectural choice provides a strong foundation for a dependable and resilient financial reporting platform.

Servers operate independently of each other. Clients also operate concurrently and independently of each other. There is no strict binding of clients to servers. More typically, a single server processes requests from different clients.
simultaneously; on the other hand, a client might address one server at one moment and another server the next. Clients need to be aware of the available servers but may have no knowledge of other clients’ existence.

For developing the server-side, the MVC (Model-View-Controller) architectural pattern was chosen (Fig. 2) [16]. This pattern divides the system into three interconnected parts: data model, view (user interface), and control module. It is applied to separate data (model) from the user interface (view) in such a way that changes to the user interface have minimal impact on data processing, and data model changes can be made without altering the user interface. The aim of the pattern is flexible software design, which should facilitate future changes or extensions to programs, and provide the possibility of reusing individual program components. Furthermore, using MVC in large systems contributes to their orderly structure, making them more understandable by reducing complexity.

**Figure 2**

General scheme of the system’s architecture
Results and Discussion. Translating a software concept into a tangible product is one journey; assessing its real-world performance and drawing insights is another. In this phase, the fruits of our developmental labor were put to the test, revealing not only the software’s capabilities but also the overarching influence of Java in shaping its success.

This section delves into the main findings from our software deployment, discusses the broader implications of these results, and contemplates potential avenues for enhancement. As we navigate through the outcomes, the pivotal role of Java remains central to our discourse, reinforcing its significance in the project’s achievements.

Let us use a coupling metric to evaluate the designed and developed Java-centric software solution for advanced enterprise financial reporting in terms of its modularity, scalability, and maintainability. The software coupling metric that can be used to estimate the whole architectural solution is given below [17]:

$$C = \frac{1}{N} \sum_{i=1}^{N} C_i,$$

where $N$ is the number of interacting software components; $C_i$ is the coupling of $i$-th software component.

To calculate the coupling of a particular software component, the following formula should be used. It takes into account both incoming and outgoing connections to indicate the degree of interaction between modules [17]:

$$C_i = 1 - \frac{1}{1 + in\_deg(C_i) + out\_deg(C_i)},$$

where $in\_deg(C_i)$ is the number of software components dependent on the $i$-th software component, $ou\_deg(C_i)$ is the number of software components on which the $i$-th software component depends.

Therefore, let us apply the coupling metrics to evaluate
the designed and developed Java-centric software solution for advanced enterprise financial reporting in terms of its modularity, scalability, and maintainability (Table 2).

<table>
<thead>
<tr>
<th>Component</th>
<th>( i )</th>
<th>( \text{in}_\text{deg}(C_i) )</th>
<th>( \text{out}_\text{deg}(C_i) )</th>
<th>( C_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>Controllers</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>Utilities</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>Audit</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>RegularDataUpdate</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>Models</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0.50</td>
</tr>
<tr>
<td>Services</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>0.86</td>
</tr>
<tr>
<td>Repository</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0.67</td>
</tr>
</tbody>
</table>

The highest coupling values were obtained for “Services” (0.86) and “Repository” (0.67) components (Table 2), which signalize their low scalability and maintainability (Fig. 3). Hence, one of the possible ways to overcome this issue, is using microservices architecture to achieve higher modularity and single responsibility for each service [18].

![Figure 3](evaluation.png)
However, the overall coupling is moderate ($C = 0.57$), which allows us to assume the appropriate level of modularity, scalability, and maintainability of the proposed Java-centric software solution for advanced enterprise financial reporting (Fig. 1).

Post-development, the software demonstrated robust performance, with Java’s inherent efficiencies leading to faster data processing and real-time financial report generation. The positive reception and software performance underscore the potential of Java-driven solutions in the enterprise financial domain. This not only reiterates Java’s position as a leading programming tool but also emphasizes its adaptability and efficiency in addressing specific industry challenges. In the broader context, the findings advocate for a deeper integration of Java in future financial software endeavors.

While the software excelled in many areas, there’s always room for growth. Some users suggested more advanced analytics tools, hinting at leveraging Java’s machine learning capabilities. Additionally, as the world of finance evolves, integrating emerging technologies like blockchain, in tandem with Java, could be the next frontier.

In essence, the results reaffirm Java’s pivotal role in developing reliable, efficient, and user-friendly financial software. The discussions arising from these findings chart a promising direction for future refinements and innovations in the domain.

Conclusion. Throughout this research journey, we unveiled the profound potential of harnessing Java for designing, developing, and implementing a financial reporting software tailored to modern enterprise needs.

As the result, the study answered the following research questions:
- The popularity of different programming languages used in the finance domain is not connected with their performance - the $p$-value of 0.70 ($\geq 0.05$) obtained as the result of regression analysis [15] confirms the acceptance of null hypothesis and allows us to consider design efficiency, user experience, data integrity, scalability, and security rather than performance when developing the Java-based enterprise
The well-known and widely-used software coupling metric \[C_7\] can be applied to evaluate a designed and developed Java-centric software solution for advanced enterprise financial reporting in terms of modularity, scalability, and maintainability – the obtained results demonstrate high coupling of service-centric \((C_7 = 0.86)\) and data-centric \((C_7 = 0.67)\) components, however the overall coupling of the proposed solution’s architecture is estimated as moderate \((C = 0.57)\).

As for the software’s prospects, the adaptable nature of Java positions it greatly for iterative improvements and integrations. With the continuous evolution of financial systems and emerging technologies, Java’s versatility promises a roadmap for integrating advanced features, analytics, and possibly new paradigms like blockchain. Moreover, a microservices approach \([18]\) can be further applied to improve modularity, scalability, and maintainability of the proposed solution.

In summation, this exploration reinforced Java’s stature as a cornerstone in modern software development, particularly for applications demanding precision, efficiency, and adaptability. The horizon looks promising for the Java-based financial software.

References:


INFORMATION AND WEB TECHNOLOGIES


