Business Intelligence Algorithmic Trading Analysis Tool Project

Date: December 5, 2023
School: University of the Fraser Valley
Department: CIS
Course: Comp 371 - Object Oriented Modeling and Design
Project: BI Tool Final Submission
Project Team: Team Super Group
Student Names: Aimen Arif - Scrum Master, Shifat and Akarshi - Code Team 1, Tejveer and Jesper - Code Team 2
Product Owner: Dr. Youry Khmelevsky

1 Current Super Group Phase

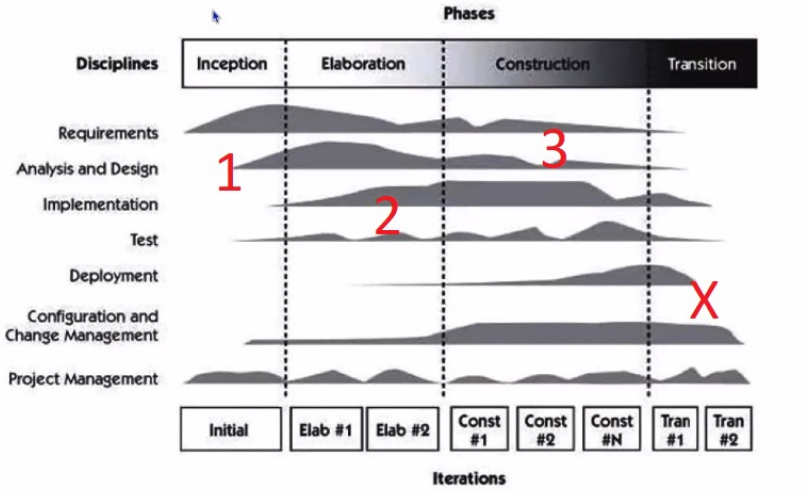

Figure 1: Phases [1]

2 Introduction

The essence of our algorithmic trading analysis is a fusion of business strategy (cost benefit) and OOP coding design with a focus on patterns. The business use case emerges as a guiding force, steering the team towards the goals of heightened returns and elevated accuracy within our trading algorithm. Against the backdrop of a five percent accuracy algorithm, the team undertakes the task of parsing through the 95 percent failures [1], methodically identifying errors with the most substantial impact [2]. With the principles of low coupling, high cohesion, Model-View-Controller and singleton taking precedence we avoid entanglements with specific companies or software solutions to preempt potential future costs and aim for low overhead and agile complainant interchangeability. Furthermore, by targeting minimal overhead and promoting flexible complainant interchangeability, we not only improve the adaptability of our trading algorithm but also strengthen our ability to seize on new opportunities in the dynamic financial markets [1].

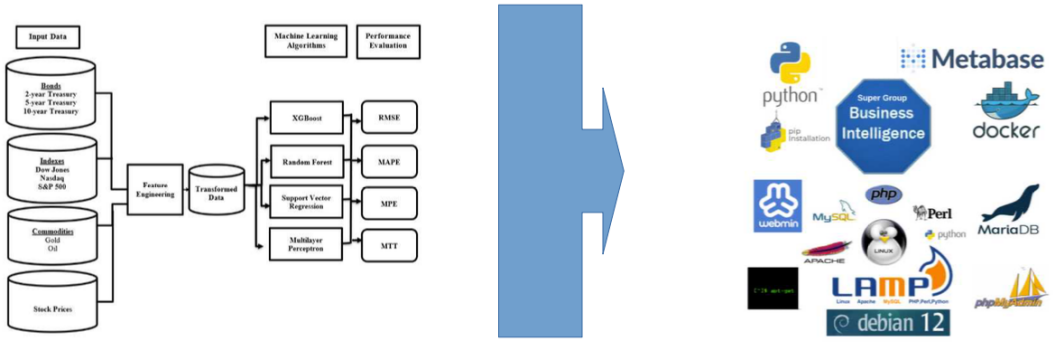

Figure 2: Tools and Architecture [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [21]

3 Pattern Development

By diagramming, analyzing, and examining our errors and successes, we continuously honed our application to be efficient and functionally meet the stakeholders' needs. From top-level analysis of use cases to middle-tier diagramming and finally low-level class design, we found that the most appropriate patterns applied to our project are: GRASP (General Responsibility Assignment Software Patterns) with high cohesion and low coupling [1], the Singleton pattern from the Gang of Four [1], and the classic Model-View-Controller (MVC) pattern developed at XEROX Palo Alto [20]. With a LAMP architecture for web infrastructure accessing large database data the MVC pattern was most appropriate. Trygve Reenskaug developed the MVC pattern at Xerox Palo Alto - He wanted a pattern for users to interact with a large, convoluted data set. This substantially improves the robustness and maintainability of our codebase. These design patterns, based on our extensive investigation, serve as a guiding framework, ensuring that our application not only runs effortlessly but also evolves effectively to fulfill the demands of stakeholders on an ongoing basis [1].

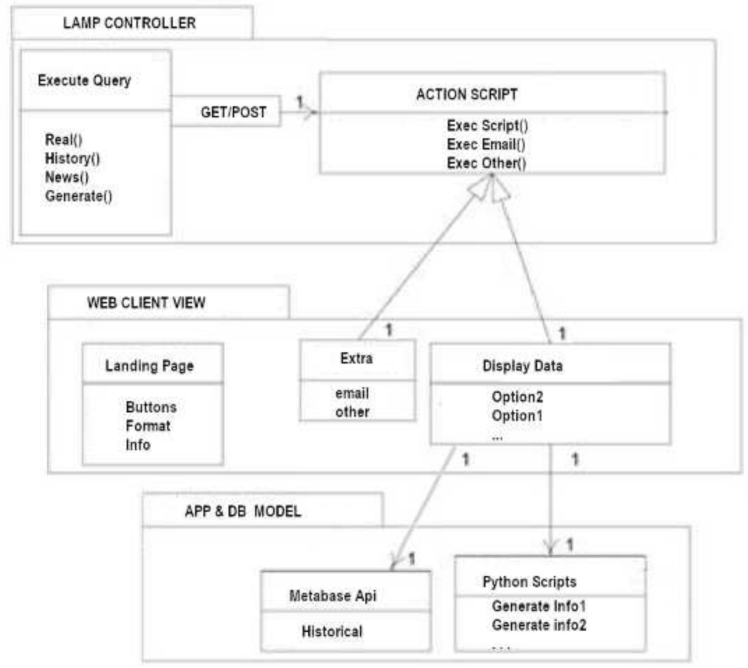

Figure 3: MVC Diagram [4]

We soon realized that when using the third-party METABASE API to generate a wealth of financial data charts, the heavy resources used, especially in a multi-user environment, would prove exceedingly expensive to initialize every time the tool was accessed. The Lazy Singleton pattern solved the resource burden by deferring instantiation to only when needed [1].

GRASP Polymorphism plays a pivotal role in fostering adaptability. By polymorphic pattern for the ticker data API the tool can seamlessly process diverse formats for analysis, ranging from ticker data to graphs and, in future, fundamental financial statements and technical analysis charts. This flexibility ensures that the BI tool can evolve without requiring extensive modifications [1].

Coupled with Polymorphism, are the principles of High Cohesion and Low Coupling contribute to the tool's structural integrity. We stay with a traditional UNIX notion of the daemon where we small programs / classes do one thing well [1]. We aim for scalability and adaptability with lowest cost in time, money and effort.

4 Software Licensing

Debian GNU License, Apache2 Apache2 License, Mariadb GNU Lesser General Public License, PHP Creative Commons Attribution 3.0 License, Python The Python Software Foundation License 2.0, Metabase GNU Affero General Public License. (Unmodified open-source system software operates without legal constraints). Project DNS Hosting provided by FreeDNS [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [21].

5 Glossary

- **Use Case:** A written description of how a system interacts with users or actors to achieve specific goals and tasks [1].
- **Objects:** An object refers to a variable, a data structure, a function, or a method. It is an instance of a class and what actually runs in the computer [1].

- **Controller:** A hardware or software program that manages and directs the flow of data between two entities [1].
- **Low Coupling:** Modules are independent and change in one module will have little or no impact on other modules. Least possible dependencies [1].
- **High Cohesion:** Elements are closely related and focused on a single purpose, a module with high cohesion will contain elements that are tightly related and united in their purpose [1].
- **GRASP (General Responsibility Assignment Software Patterns):** A set of nine fundamental principles in object design and analysis [1].
- **Pure Fabrication:** A class that does not represent a concept, not real objects, in the problem domain, specially made up to achieve low coupling, high cohesion, and the reuse potential [1].
- **Information Expert:** It is a principle that determines where and how to delegate responsibilities such as methods, computer fields, and so on [1].
- **Indirection:** This pattern allows for low coupling and reuses the potential between two elements by assigning the responsibility of mediation between them to an intermediate object [1].
- **Polymorphism:** A core concept of object-oriented programming, you are able to access objects of different types through the same interface [1].
- **Protected Variation:** This pattern uses polymorphism to create multiple implementations of this interface and protects the elements from variation on other elements (objects, systems, and subsystems) by wrapping the focus of instability with an interface [1].

6 Conclusion

Finally, our journey towards developing a customizable Business Intelligence tool as been a systematic evolution. We strengthened the basis of our project with the concepts of High Cohesion and Low Coupling, the GRASP Polymorph pattern, the Group of Four Singleton pattern, and the Xerox MVC pattern, starting with thorough planning and proceeding through extensive testing. Then we approached the end of the construction process. Code completion, testing, performance optimization, and regulatory compliance were checked. A security audit continued, user training materials were developed, and we developed a deployment strategy with an emphasis on resolving any issues in the early phases. Throughout, the use of GRASP patterns has been critical in developing a strong system architecture. Our agile methodology supported a smooth transition by emphasizing ongoing improvement and refinement. Now we are preparing our system for dynamic adaptability by focusing on configuration and change management throughout the transition period. In the future, our strategic plan will incorporate both fundamental and technical analysis. This forward-thinking approach demonstrates our dedication to innovation, ensuring that our system remains at the forefront of innovations in the dynamic world of algorithmic trading [1].

7 References

[1] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd ed. Addison, October 20, 2004.

[2] Albert Wong, Albert, Steven Whang, Emilio Sagre, Niha Sachin, Gustavo Dutra, Yew-Wei Lim, Gaetan Hains, Youry Khmelevsky, and Frank Zhang. *Short-Term Stock Price Forecasting Using Exogenous Variables and Machine Learning Algorithms*. arXiv [q-Fin.TR], 2023. `http://arxiv.org/abs/2309.00618`.

[3] "LAMP (Linux, Apache, MySQL, PHP)," TechTarget. `https://whatis.techtarget.com/definition/LAMP-Linux-Apache-MySQL-PHP` (accessed October 24, 2023).

[4] "Atlassian." 2020. *Atlassian.com. Atlassian — Start page*. `https://start.atlassian.com/`.

[5] "Open-Source Tool That Uses Simple Textual Descriptions to Draw Beautiful UML Diagrams." n.d. *PlantUML.com*. `https://plantuml.com/`.

[6] News API. "News API – Search News and Blog Articles on the Web," n.d. `https://newsapi.org/`.

[7] "Model–View–Controller." Wikipedia, Wikimedia Foundation, 21 Oct. 2023, en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. Accessed 21 Nov. 2023.
[8] "Debian – The Universal Operating System," October 7, 2023. https://www.debian.org/.
[9] Metabase — Business Intelligence, Dashboards, and Data Visualization. "Metabase — Business Intelligence, Dashboards, and Data Visualization," n.d. https://www.metabase.com/.
[10] Python.org. "Welcome to Python.Org," November 15, 2023. https://www.python.org/.
[11] PyPI. "PyPI · The Python Package Index," n.d. https://pypi.org/.
[12] Cameron, Jamie. "Webmin." Webmin, n.d. https://webmin.com/.
[13] Docker. "Docker: Accelerated Container Application Development," October 18, 2023. https://www.docker.com/.
[14] MariaDB.org. "MariaDB Foundation - MariaDB.Org," November 13, 2019. https://mariadb.org/.
[15] Contributors, phpMyAdmin. "phpMyAdmin." phpMyAdmin, n.d. https://www.phpmyadmin.net/.
[16] "PHP: Hypertext Preprocessor," November 9, 2023. https://www.php.net/.
[17] "MySQL," n.d. https://www.mysql.com/.
[18] "The Perl Programming Language - Www.Perl.Org," n.d. https://www.perl.org/.
[19] "Welcome to The Apache Software Foundation!," n.d. https://www.apache.org/.
[20] Trygve Reenskaug. 1979. *Model View Controller (MVC)*. https://medium.com/bumble-tech/do-mvc-like-its-1979-da62304f6568.
[21] "FreeDNS - Free DNS - Dynamic DNS - Static DNS Subdomain and Domain Hosting," n.d. https://freedns.afraid.org/.