Journal of Digital Education and Learning Engineering ปีที่ 1 ฉบับที่ 1 (เดือน มกราคม – มีนาคม 2568) หน้า 119-128
Journal of Digital Education and Learning Engineering, Volume 1 No. 1 (January – March 2025) Page 119-128

บทความเทคนิค (Technical Article)

Journal of Digital Education and Learning Engineering
ดำเนินการวารสารโดย สมาคมการศึกษาดิจิทัลและวิศวกรรมการเรียนรู้

Development of a Flexible Reporting System Using Design Patterns

Hiranya Boonchoo & Jirapipat Thanyaphongphat*

College of Arts, Media and Technology, Chiang Mai University, Chiangmai, Thailand

Received: March 15, 2025 Revised: March 25, 2025 Accepted: March 29, 2025

Abstract

Organizations managing large datasets require adaptable and customizable reporting systems to generate precise reports and exports, supporting informed decision-making. This research investigates the implementation of reporting systems using Design Patterns, a software engineering methodology that offers reusable solutions to common challenges. The study examines how Design Patterns enhance system adaptability and provide a framework for software development best practices, thereby improving the quality of work in the digital sector. The research outlines essential stages in developing a reporting system, including Design Pattern analysis, system architecture, and testing. A comparative analysis shows that systems developed with Design Patterns achieve reduced development time compared to those without them. The study also explores applications across business, educational, and research domains. Challenges in designing scalable systems, such as complexity and maintenance issues, are addressed through solutions like the Builder Pattern for managing intricate objects and the Strategy Pattern for handling various report generation approaches. These methods not only address complex requirements but also foster skill development by allowing developers to gain hands-on experience in system customization and software architecture. The research concludes that Design Patterns enhance system flexibility, decrease development time, and facilitate quicker error identification. They assist organizations and educational institutions in efficient resource allocation, allowing a focus on innovation rather than process management. The study demonstrates that Design Patterns improve system scalability, maintainability, and learning opportunities, making them valuable in both business and educational contexts.

Keywords: Builder Pattern, Design Patterns, Expansion of reporting system, Strategy Pattern, Learning Engineering

Introduction

The development of a reporting system is essential for data management and decision-making at the organizational level. In environments with significant public and private sector investments, flexible and customizable reports are crucial tools for tracking and analyzing various data, particularly those related to different account types and companies. These reports must be presented in both detailed and timely

*Corresponding author.
Email address: jirapipat.than@cmu.ac.th

formats. Developing such a system will facilitate access to essential data, allowing users to select time periods and account types as needed. This capability supports quick and accurate decision-making (Michaud & Walker, 2019).

Currently, many organizations' reporting systems lack flexibility and fail to fully meet users' specific needs, particularly in organizations that operate across multiple sectors and handle large amounts of data. Emphasizing the importance of using Design Patterns in software system development helps reduce complexity and development time (Buddhavarapu & Shaik, 2017). Systems designed with Design Patterns simplify development and can be adapted to rapidly changing conditions. Therefore, incorporating Design Patterns into the development of reporting systems enhances flexibility and ensures adaptability to evolving requirements.

Traditional reporting systems suffer from high maintenance costs, lack of flexibility, and limited scalability. Several studies highlight these challenges and the need for adaptable architecture. Rigid reporting structures lead to inefficiencies in data retrieval and modification, increasing operational costs (Michaud & Walker, 2019). Applying Design Patterns improves software reusability and reduces the complexity of enterprise reporting systems (Buddhavarapu & Shaik, 2017). These references reinforce the motivation for adopting a structured, pattern-based approach in this research, ensuring the system meets modern development and usability standards.

One of the key challenges in developing a reporting system is the long development time and the complexity of data management, especially when dealing with large datasets from multiple sources and generating complex reports. These challenges can lead to errors in data reporting (Sommerville, 2011). The software being developed aims to address these issues by implementing Design Patterns that improve development processes and create a highly adaptable reporting system. This system will support expansion to meet user needs without requiring frequent updates that consume time and resources.

The developed reporting system is designed to be applicable in various domains, including educational institutions, business enterprises, and public sector reporting. In educational institutions, the system can generate academic reports, student performance analyses, and course-related statistics. Business enterprises can utilize the system for financial and inventory reporting, optimizing decision-making processes. Government and public organizations can leverage the system for generating transparent public data reports, ensuring regulatory compliance. By allowing users to customize reporting conditions dynamically, the system enhances operational flexibility, decision support, and data-driven insights, making it a valuable tool in diverse organizational settings.

The primary objective of this system is to provide a tool that enables flexible report design, allowing users to define report parameters, such as account types and time periods, while exporting data in appropriate formats, including PDF and Excel. This functionality supports decision-making and strategic planning effectively (Buddhavarapu & Shaik, 2017). The goal is to develop a system that is highly flexible, user-friendly, and capable of promptly meeting organizational needs, with a focus on enhancing operational effectiveness and reducing development time.

Objectives

1) To develop a report generation system that allows users to define report generation conditions.
2) To apply Design Pattern techniques to increase flexibility and reduce system complexity.
3) To reduce the time required to develop, maintain, and expand the system.

Methodology

The development of a customizable report generation system follows the Agile Model, as shown in Figure 1.

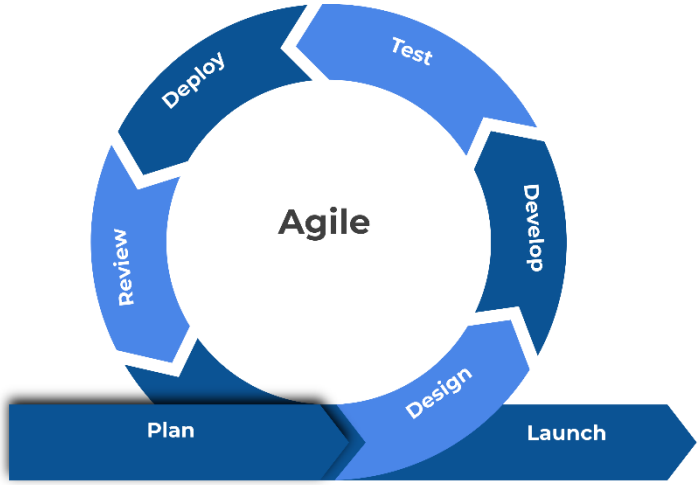

Figure 1. Agile Model.

1) Plan

This step involves collecting and managing data required for the system through meetings and interviews with potential users. The system must define report formats and filtering conditions, such as categories of financial data, entity names, and reporting periods to ensure flexibility in report customization.

2) Design system

In this step, the systems analyst (SA) will analyze the data obtained from the previous step to create a document specifying customer needs and technical requirements, including designing the system to support flexible condition selection using the principles of appropriate Design Patterns, such as the Builder Pattern used to manage the process of creating report components, or the Strategy Pattern used to select the method of displaying the report in various formats.

3) System development process

The development of the report generation system will begin with the design of the database and data structure that will support the selection of data in each field. Including filtering data according to specified conditions, such as financial categories, entity names based on user-defined parameters, company name

and time periods. The system supports multiple export formats, including structured document files and spreadsheet-based reports for enhanced usability.

3.1) Database design

In this development, the database management tool, PostgreSQL, is used. First, the database must be designed to support the storage of data related to report generation, which will include necessary data such as account type, quantity, company name and data related to the time periods selected by the user for correct data filtering. This design will use the principle of Normalization to ensure that the data has the correct relationship and can be searched quickly.

3.2) Backend development with .Net Core

In the Backend section, .NET Core will be used as the main development technology because it is flexible and highly efficient. An API will be created that supports receiving data from the Frontend and exporting data in various formats, such as PDF and Excel, which will use data management techniques such as LINQ and creating PDF or Excel files with appropriate libraries, including Fast Report for PDF report generation and Mini Excel for Excel report generation.

3.4) Frontend development with Next.js

Next.js helps develop web applications efficiently and quickly by using Redux to manage the status of data selected by the user, such as data filtering conditions. And send those data to the Backend. Using Redux allows data in the application to be shared between components without having to send data through props every time, which makes data development and management easier.

3.5) Using design patterns in development

While Design Patterns are widely used in software development, this research uniquely integrates the Builder Pattern and Strategy Pattern to enhance reporting system flexibility. Traditional reporting tools often rely on rigid architecture, making customization and expansion challenging.

The Builder Pattern enables modular report component creation, allowing flexible structuring of reports, including headers, tables, and summaries.

The Strategy Pattern supports multiple export formats (PDF, Excel) without modifying the core system architecture.

Conventional reporting systems typically require hard-coded templates, making modifications complex and time-consuming. Our approach introduces a scalable, maintainable, and adaptable framework, ensuring easy integration of new features without significant refactoring.

4) Testing

Testing will be performed at both Unit Testing, Integration Testing, and User Acceptance Testing (UAT) levels to ensure that the system can work as specified. The testing will cover data filtering, report generation in PDF and Excel formats, and system response to user needs.

5) Deployment

After development and testing, the system will be installed on the Cloud Server with the PostgreSQL database and API set up for use.

6) System review

When the system is used, feedback from users will be collected to analyze problems and provide suggestions. Further development plans will be planned in the future to make the system more flexible and expandable.

7) Official launch

After the system has been improved based on feedback, the system will be officially launched with notification to users and a user manual and video tutorial to reduce usage problems.

Result

From the implementation of various steps in developing a flexible and efficient reporting system, the selection of appropriate technology includes API development with .NET Core and Frontend development with Next.js. For the system to be completed according to the objectives, it is necessary to have a design that supports easy system expansion, using Design Pattern as the main guideline for development. The system structure must be able to handle a variety of data and user-specified conditions, including the use of Builder Pattern to separate report elements and Strategy Pattern to define how to export reports in various formats, such as PDF and Excel.

The Builder Pattern is used to create separate reports, dividing the report structure into elements such as report headings, data lists, results summary, and settings, which allows the report format to be defined as desired. The system will have a builder that can assemble reports as specified, and a director that defines the creation sequence, as shown in Figure 2.

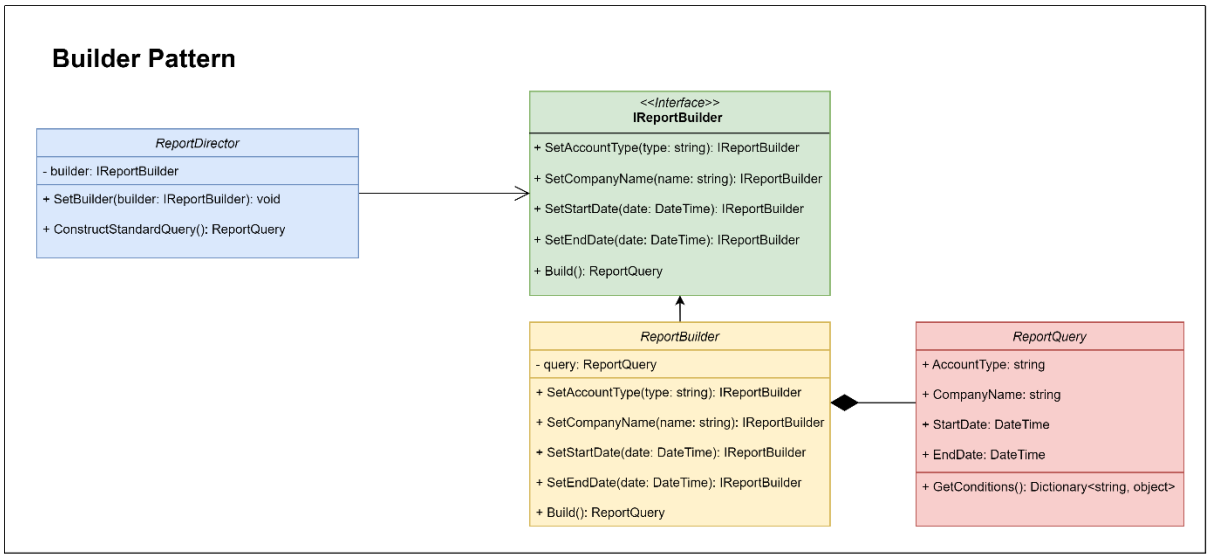

Figure 2. Design Pattern structure diagram of the report generation system, showing the use of the Builder Pattern to create report elements.

The Strategy Pattern is used to select how to export reports in various formats, such as PDF and Excel. The system can change the strategy according to the user's needs without having to change the main

structure. For example, if the report is to be output as PDF, the system will use the PDF Export Strategy that relies on FastReport, and if the report is to be output as Excel, the Excel Export Strategy that relies on MiniExcel will be used, as shown in Figure 3.

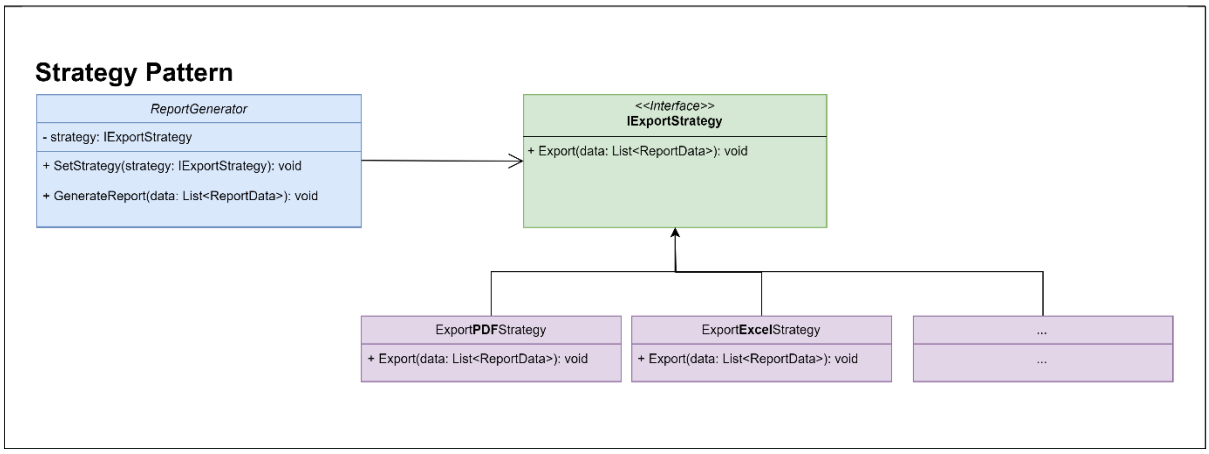

Figure 3. Design Pattern structure diagram of the report generation system, showing the use of the Builder Pattern to create report elements.

When these two Design Patterns work together, the Builder Pattern helps to structure the report in a flexible way, while the Strategy Pattern helps to select the appropriate export method, allowing the system to support customizable report output and add new functions without affecting the original structure.

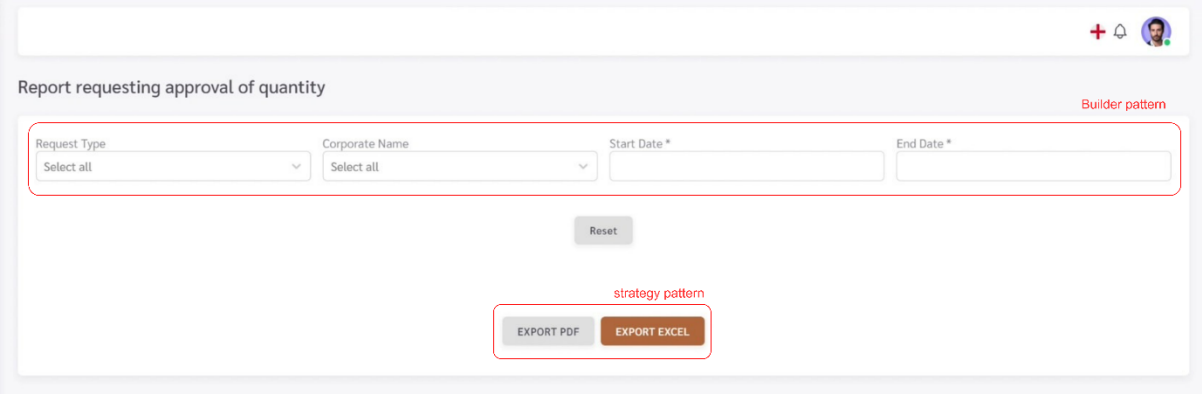

Figure 4. Design Pattern structure diagram of the report generation system, showing the use of the Builder Pattern to create report elements.

There is an example of a report output web page format that allows users to define report output conditions according to their needs, such as users can select the type of account quantity (e.g. machinery, spare parts, molds, raw materials), including the name of the company, the start period, and the end period for which they want to generate the report. In this section, the system will retrieve data from the PostgreSQL database according to the specified conditions, with data filtering to meet the user's needs. It also uses the Builder Pattern to structure the report systematically, such as adding topics, data tables, and summaries,

and uses the Strategy Pattern to help select the report format, such as PDF or Excel, as shown in Figure 4. When the report export button is pressed, the results will be in PDF or Excel format, as shown in Figure 5.

Bill of quantity Type	Corporate name	Juristic Number	Bill of quantity Name	Request	Unit	Approved Date
Machine	A company	3581567018173	Vertical Machining Center - BMV850	10	PA	1/11/2024
Spare	A company	3581567018173	MILLING AND BORING MACHINES	100	PA	1/11/2024
Mold	Entreprenuer 1	6948503872669	Preform Mould & Hot Runner System	20	LTN	5/11/2024
Raw material	Entreprenuer 2	9321266791397	GOCHUKANG / RED PEPPER PASTE	2000	KGM	5/11/2024
Machine	B company	0715491990663	Heavy duty milling machine	2700	KGM	5/11/2024

Figure 5. Example of report output results in PDF format.

The system testing provides accurate and stable results in all cases where data filtering conditions are selected, through testing at both Unit Testing, Integration Testing, and User Acceptance Testing (UAT). The results of the testing ensure that the system can operate as specified and fully meet user needs. The results are in line with the intended objectives. The system can help improve the efficiency of EEC operations and make business decision-making processes faster and more accurate.

The experimental results confirm that applying Design Patterns significantly enhances development efficiency, maintainability, and system scalability. Development time was reduced from 40 to 30 days (25% improvement), maintenance efforts decreased from 15 to 5 hours (66.67% improvement), extensibility improved as new features could be added within 5 days instead of 20 (75% improvement), and testing time was reduced from 8 to 3 hours (62.5% improvement). These findings align with the research objectives by demonstrating that a pattern-based approach reduces complexity, improves adaptability, and supports long-term sustainability, making the system more efficient and easier to maintain. The details are shown in the following table.

Table 1.

Abstract Report Output Comparison Table

	Simple Reporting	Design Pattern Reporting
Development Time	Slower development and may require multiple iterations.	Faster development and better code management.
Maintainability	Code is not modularized, making it complex.	Changes can be made without affecting the existing code.
Extensibility	Multiple changes are required to support new features.	New features can be added without affecting the existing structure.

Testing	Difficult to detect bugs.	Each module can be tested separately.

From Table 1, the use of Design Patterns increases the flexibility of the system, making it easier to add new features, reduce code complexity, and better separate modules, resulting in easier maintenance and expansion of the system. In addition, testing can be done in sections, making it easier to find errors, unlike the traditional method that requires testing the entire system, which shows the improved efficiency of using Design Patterns in developing reporting systems.

Table 2.

Comparison table of concrete report output

	Simple Reporting	Design Pattern Reporting
Development Time	40 Working Days	30 Working Days
Maintainability	15 Hours per Time	5 Hours per Time
Extensibility	20 Working Days	5 Working Days
Testing	8 Hours per Test	3 Hours per Test

To calculate the result in percentage form, we will calculate the percentage reduction from Normal Reporting to Design Pattern Reporting using the formula: Find the difference between Time (Normal) and Time (Design Pattern), divide by Time (Normal), and multiply by 100.

Table 3.

Development statistics calculation table

	Simple Reporting	Design Pattern Reporting	Percentage reduction from normal
Development Time	320 hours	240 hours	25%
Maintainability	15 hours	5 hours	66.67%
Extensibility	160 hours	40 hours	75%
Testing	8 hours	3 hours	62.5%
Total	503 hours	288 hours	42.74%

From Table 3, it is found that using Design Pattern in reporting can effectively reduce development and maintenance time. It can reduce development time by 25%, maintenance by 66.67%, system expansion time by 75%, and testing by 62.5%. In summary, using Design Pattern can reduce the total time by 42.74%, making system development and maintenance faster and more flexible to expand the system in the future.

Discussion and Conclusion

The research in this project aims to develop and test a customizable report system based on user-defined conditions, using Design Pattern to increase flexibility and ease of modification. The results of the operation found that this approach significantly increases the flexibility of development and reduces the complexity of the code. Compared to traditional report generation methods, the system using Design Pattern can reduce development time by 25%, reduce maintenance time by 66.67%, and speed up system testing by 62.5%, reflecting that the use of Builder Pattern and Strategy Pattern is highly effective in structuring and managing report generation.

Compared to similar research or projects in the past, the developed system demonstrates the ability to use Design Pattern to reduce the complexity of system development and make it easy to modify and extend functions without affecting the main structure of the system. These results are consistent with the concept of using Design Pattern to develop a highly flexible software system that can support future changes. Using Strategy Pattern and Builder Pattern to develop the system allows users to customize the display as desired and increases the convenience of changing the report format in the future, which is an advantage that helps the system development to be flexible and easy to maintain.

Recommendations for future research include adding new features or functions. That can support more complex calculations or data processing, such as creating reports that can link data from multiple sources or developing APIs that can be used to connect to other systems. However, using design patterns can help the system be more flexible in some cases.

However, excessive use can make it more difficult to change or expand the system in the future due to reliance on systems or components that are already designed in the pattern. Therefore, the use of design patterns should be considered appropriate for the problem to be solved. Patterns that meet the needs and are appropriate for the size and complexity of the project should be selected to maximize the benefits of the design.

References

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.

Sommerville, I. (2011). Software engineering (9th ed.). Addison-Wesley.

Michaud, R., & Walker, G. (2019). Enhancing decision-making in organizations with tailored reporting systems. Journal of Business Analytics, 12(3), 22-35.

Buddhavarapu, S., & Shaik, M. (2017). Design patterns for effective reporting in enterprise systems. International Journal of Computer Applications, 159(7), 11-16.