computers

MDPI

Article

The Impact of the Web Data Access Object (WebDAO) Design Pattern on Productivity

Zoltán Richárd Jánki * and Vilmos Bilicki

Department of Software Engineering, Institute of Informatics, University of Szeged, 6720 Szeged, Hungary; bilickiv@inf.u-szeged.hu
* Correspondence: jankiz@inf.u-szeged.hu

Abstract: In contemporary software development, it is crucial to adhere to design patterns because well-organized and readily maintainable source code facilitates bug fixes and the development of new features. A carefully selected set of design patterns can have a significant impact on the productivity of software development. Data Access Object (DAO) is a frequently used design pattern that provides an abstraction layer between the application and the database and is present in the back-end. As serverless development arises, more and more applications are using the DAO design pattern, but it has been moved to the front-end. We refer to this pattern as WebDAO. It is evident that the DAO pattern improves development productivity, but it has never been demonstrated for WebDAO. Here, we evaluated the open source Angular projects to determine whether they use WebDAO. For automatic evaluation, we trained a Natural Language Processing (NLP) model that can recognize the WebDAO design pattern with 92% accuracy. On the basis of the results, we analyzed the entire history of the projects and presented how the WebDAO design pattern impacts productivity, taking into account the number of commits, changes, and issues.

Keywords: WebDAO; design pattern; productivity; software development; open source

check for updates

Citation: Jánki, Z.R.; Bilicki, V. The Impact of the Web Data Access Object (WebDAO) Design Pattern on Productivity. *Computers* **2023**, *12*, 149. https://doi.org/10.3390/computers12080149

Academic Editor:Paolo Bellavista

Received: 13 June 2023
Revised: 18 July 2023
Accepted: 24 July 2023
Published: 27 July 2023

1. Introduction

Early software development was performed in machine language, which was a time-consuming and error-prone process. With the development of higher-level programming languages, programming became more accessible to a broader audience. “The discipline of software engineering required development methodologies such as Waterfall and Agile that increase the efficacy and quality of software development, but the quality of the source code must also have been enhanced [1]”. These methodologies provide a structured development framework that produces well-organized source code, thereby having a positive influence on source code quality. The planning and design phases must play significant roles in the development process because these duties will be reflected in the source code. The appearance of object-oriented programming resulted in significant advancements in software development, which still have a great deal of potential. Best practices in novel methodologies and techniques have always guided developers, but they are sometimes language- or framework-specific. To be readily adaptable, it is necessary to collect the best practices and specify so-called design patterns from them. The patterns are time-tested solutions to common problems; consequently, software developers can use them as templates to resolve common issues, resulting in more reusable and maintainable code. In addition, design patterns provide a universal language for software professionals, allowing for clear communication regarding solutions and architecture. This improves collaboration and increases the effectiveness of debugging and code evaluations. Maintaining and modifying systems with inadequate documentation necessitates the detection of design patterns [2]. Despite these advantages, the application and significance of design patterns are frequently undervalued or neglected. Numerous developers prioritize functionality over the long-term advantages of using design patterns.

Occasionally, however, it is necessary to view our software as a system. The low-level details and the high-level structure comprise a single entity. One cannot exist without the other, and there is no distinct line separating them. Architecture functions as a blueprint for a system, providing an abstraction that manages system complexity and establishes communication and coordination among elements. It provides a structured approach for meeting technical and operational requirements while optimizing crucial quality attributes such as performance and security. In addition, it involves making crucial decisions regarding the organization of software development, and each of these decisions has a significant impact on the quality, maintainability, performance, and overall success of the final product [3,4].

The layered architecture, also known as the n-tier architecture, is one of the most widely used architectural patterns, but is especially prevalent in contemporary Web development. The layered architecture presents the traditional IT communication and organizational structure, making it an appropriate design choice in many situations. It has become the de facto standard for Java 2 Platform and Enterprise Edition (J2EE) applications, and the layered architecture pattern is currently utilized by the most prominent Web frameworks as well [5].

Despite the fact that these front-end frameworks were designed to construct applications with layered architecture, it is the developer's responsibility to thoroughly implement the architecture or simplify the structure. Often, software developers implicitly employ well-proven architectural solutions resulting from the implementation and best practices of particular frameworks. In a system, it is always crucial how the front-end and the back-end are communicating and how the data sources can be accessed. It is the developer's responsibility to not only concentrate on creating a user-friendly interface but also to ensure a robust and efficient back-end infrastructure capable of handling requests and delivering responses quickly and accurately. When front-end and back-end technologies are considered, there are numerous development platforms that can implement a potent combination of frameworks. Technically, there are no inherent limitations dictating specific combinations. Nevertheless, certain pairings are typically recommended due to their complementary characteristics or proven efficacy in specific scenarios. In Web development, two prominent stacks are the MEAN and MERN stacks, which provide so-called full-stack development. Not only do these frameworks work well together, but they also offer a high level of productivity because an entire system can be implemented with knowledge of a single programming language [6].

According to a well-known principle [7], "Every software problem can be solved with another layer of indirection". A classic layered architecture consists of four layers: a presentation layer, a business layer, a persistence layer, and a database layer [8]. In practice, however, there are different shared services that should be used at various locations of the application and must be accessed via the business layer. Since these shared services are not explicitly a part of the business layer, it makes sense to delegate them as a so-called service layer. However, in many instances it is not necessary to traverse the service layer, so it is left as an open layer that can be disregarded if necessary. The link between the data source and the client is the persistence layer. Data Access Object (DAO) incorporates this isolated layer into a system and provides elegant abstract access to the data source. DAO is a well-proven design pattern that is essential in J2EE application development. Today, the DAO design pattern is also extensively utilized in Web development. Traditional DAO is placed at the back-end of a system, but as serverless development becomes more prevalent, clients should access the data source directly or through an unknown system. Since modern front-end frameworks such as Angular, React, and Vue.js follow the concepts of layered architecture [9], DAO must take place in the front-end. Technically, it can represent an open layer in the architecture, so it can function as a service layer in a contemporary five-tier architecture.

DAO is considered a design pattern that, due to its simplicity, may increase software development productivity. Productivity is one of the most important subjects that ties together technical, social, and economic factors [10]. It can be measured from a variety of perspectives [11,12], but focused statistics can be obtained by analyzing source codes and project metrics. Organizations are constantly on the lookout for new methods to enhance development productivity [13]. It can be substantially improved if the following six primary options are considered: getting the most out of people, making steps more efficient, eliminating steps, eliminating rework, creating simplified products, and reusing components [14]. DAO keeps the source code simple and reusable, allowing at least two of the six requirements to be met.

Considering the modern DevOps methodologies, software releases have become faster and more frequent than ever before by utilizing Continuous Integration, Continuous Delivery (CI/CD); however, this technology requires a high level of productivity [15]. DAO is an end-to-end design pattern because it affects all phases of the software development lifecycle. It is believed that writing source code with explicit design patterns and guaranteeing quality with build pipelines and review procedures results in software that is more maintainable [16].

After several years of Web development experience, we discovered that developers frequently employ the DAO design pattern or DAO-like structures in the front-end. Today, Angular, React, and Vue.js are the three most prominent front-end frameworks in the world, but Angular was the first to offer TypeScript as its primary implementation language, thereby making the source code type safe [17]. DAO is an object-oriented design pattern, so Angular can serve as a firm basis for gathering sufficient samples for analysis. To provide global statistics on the use of DAO in Angular, either extensive manual labor or dependable automation is required.

In this study, we present a 92% accurate machine-learning-based technique for detecting the DAO design pattern in Angular applications. Using our trained model, we demonstrate that WebDAO is present in over 25% of open-source Angular projects and that it enhances productivity by requiring over two times fewer source code modifications than projects that do not use WebDAO. In addition, the lengths of the issues are nearly 60 h shorter, and the standard deviation of the issue lengths is substantially lower than in projects that do not use WebDAO.

The main contributions of this paper are as follows:

- Introducing the WebDAO design pattern.
- A machine learning (ML) model for detecting the WebDAO design pattern in Angular applications.
- A dataset containing 19,116 Angular projects downloaded from GitHub.
- The analysis of a retrieved dataset using a self-trained machine learning model.
- Comparison of project and productivity metrics with the WebDAO design pattern in mind.

Our study is guided by the three research questions listed below:

RQ1: How accurately can we detect the WebDAO design pattern?

RQ2: In how many projects is the WebDAO design pattern dominant?

RQ3: How does the WebDAO design pattern influence the productivity and the project timeline?

The remaining sections are organized as follows: In Section 2, we provide an overview of the focused design pattern. In Section 3, we discuss the existing literature review in this field. Section 4 describes the experimental dataset, ML algorithms, and evaluation methods. Section 5 presents the experimental results and Section 6 presents a short discussion. In Section 7, we present the potential threats to validity. Section 8 concludes the study by presenting the key findings, limitations, and future research.

2. Background

This section provides context for the software structural patterns analyzed in the present paper.

2.1. Layered Architecture in Front-End Applications

The layered architecture pattern emphasizes the concept of closed layers, meaning that requests must traverse through the layer directly below them as they move through the architecture. For example, a request from the presentation layer must first go through the business layer and then the persistence layer before reaching the database layer. This closed-layer approach is essential for maintaining layers of isolation [18,19]. Layers of isolation ensure that changes made within one layer have minimal impact on components in other layers. If the presentation layer were allowed direct access to the persistence layer or database layer, changes to the database structure would affect both the business layer and the presentation layer, resulting in a tightly coupled and interdependent application. The layers-of-isolation concept enables each layer to operate independently, making it easier to modify and refactor specific layers without disrupting the entire architecture [20]. While closed layers promote isolation, there are cases where certain layers may be open. For example, introducing a shared-services layer restricts access to common service components from the business layer but not the presentation layer. By creating an open layer, access restrictions can be governed more effectively. However, open layers can introduce challenges, such as the business layer needing to go through the services layer to reach the persistence layer. To address this, open layers are created within the architecture to allow specific layers to be bypassed when necessary.

Modern Web frameworks have begun to employ similar front-end concepts. Google's Angular framework perfectly matches the concepts of layered architecture [21–23]. As presented in Figure 1, Angular's component-based architecture segregates the presentation, business, and service layers. Occasionally, the distinctions between the business and service layers become blurry. Angular templates are responsible for the presentation; components and services contain the business logic; and services serve as a link between the front-end and the back-end. In serverless development, the entire back-end is hidden from the developer, allowing the business logic and service layers to communicate directly with the back-end. This communication can be managed through an Application Programming Interface (API) or a system-specific Software Development Kit (SDK).

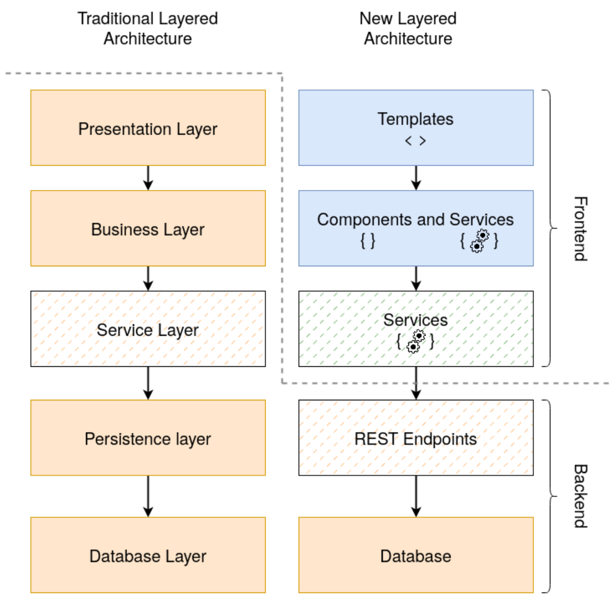


Figure 1. Comparison of traditional and modern layered architecture.

2.2. The WebDAO Design Pattern

Data Access Object (DAO) is a structural design pattern that is frequently combined with the layered architecture pattern. DAO functions as a bridge between the business layer and the database layer in a layered architecture. It provides an abstraction for accessing and manipulating data from diverse data sources, including databases, APIs, and files. This allows the business logic layer to interact with data in a consistent and standardized manner. Typically, the DAO comprises interfaces and concrete implementations. Interfaces define the contract for data access operations, while implementations provide the actual implementation details for interacting with the data source [24]. Utilizing DAO within the layered architecture improves code reuse, maintainability, and productivity. The separation of concerns facilitates testing and modifications in the future. Changes to the data access implementation, such as transferring databases or integrating new data sources, can be limited to the DAO layer, thereby mitigating the impact on other layers. To maximize data access efficacy, DAO can also integrate additional functionality, such as caching or connection pooling. It functions as a centralized access point for data operations, ensuring consistency and encapsulating the complexities of data access. However, in contemporary Web development, a similar layer with the same responsibility exists on the front-end. Because it functions as a DAO with Web capabilities, we refer to it as WebDAO. WebDAO is typically placed in the service layer, where an interface or class of the domain entity is defined, and provides all four fundamental database operations: create, read, update, and delete (CRUD). As shown in Figure 2, DAO can still play a significant role in the back-end, but WebDAO is the link between the front-end and back-end.

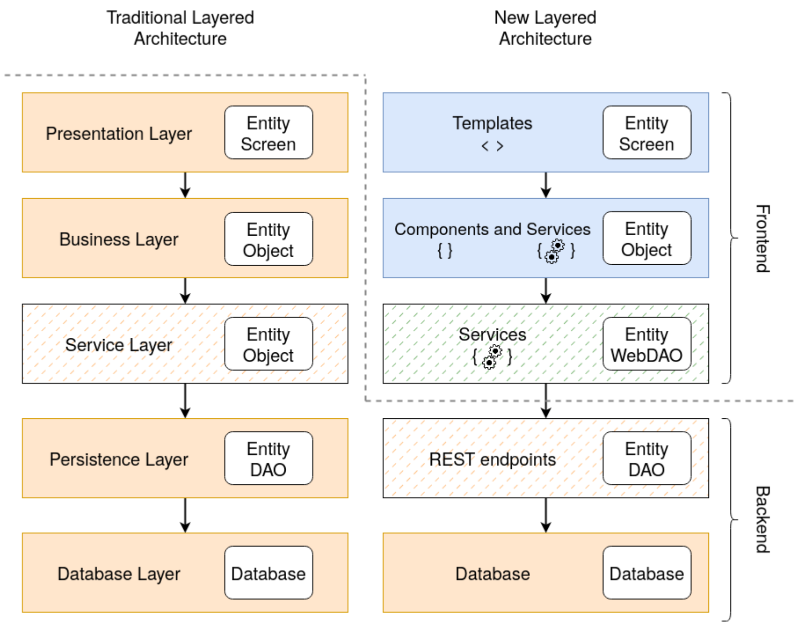


Figure 2. Presence of WebDAO in modern layered architecture.

With the contribution of Inclouded [25], we have already implemented such a solution in the form of an installable package [26] that is publicly available for developers who use the NodeJS runtime environment. This package supports standardized telemedicine application development and provides tools for Not-only Structured Query Language (NoSQL).

3. Related Work

To recognize design patterns within a system, it is necessary to comprehend how the employed technologies and frameworks function. S. Rathinam conducted a comprehensive comparison of Angular, React, and Vue.js [27]. Angular has the steepest learning curve, whereas Vue.js has the shallowest learning curve. It is claimed that Angular has the worst performance due to its complex structure, which can result in delayed initial load times. Angular and React have the largest communities, but React has stronger support due to its simplicity. Scalability is a strength of all frameworks, but Angular is recommended for very large applications. React and Vue.js are highly adaptable and allow developers to use arbitrary libraries for state management and routing problems. Although the three frameworks were developed using distinct concepts, they share similar capabilities and solutions. Object-oriented programming is necessary for DAO and WebDAO design patterns. WebDAO could provide an abstraction layer to manage data access and persistence in the context of a Web application, enabling the isolation of the application's business logic from the underlying data storage or APIs. Angular is the most typed framework because TypeScript is its principal language, so this requirement is met by this framework. F. Al-Hawari examined six projects and outlined five client-side design patterns [28]. However, the aforementioned design patterns exist primarily in the form of User Interface (UI) components, and by implementing them, the source code's reusability is achieved and the required development time is reduced. Manually finding these design patterns in a large codebase is tedious and error-prone. It is always better to automate a process if it is possible. Automating a process whenever practicable is always preferable. Some design patterns can also be predicted using code metrics, but a machine learning approach can be more effective because it can make decisions based not only on the facts but also on the contexts. ML and NLP revolutionize design pattern identification and implementation. ML systems can learn to recognize design trends using millions of code samples. NLP techniques not only accelerate the discovery of design patterns but also reduce human error. Using ML models, S. Komolov et al. [29] developed a method for detecting Model–View–Presenter (MVP) and Model–View–ViewModel (MVVM) architectural design patterns. They gathered 5973 Java-based Android projects from GitHub and compared the performance of their approach to other published ones. Their contributions are accurate to the extent of 83%. In their study, the prediction of design patterns is a metric-based implementation utilizing the Java code metrics calculator (CK) [30] to extract metrics. It is difficult and potentially misleading to evaluate a project based on code metrics. In addition, the findings yield no additional statistical results. However, code metrics could be used to estimate how a desired pattern impacts a project's source code quality and development processes. S. Wagner and M. Ruhe [31] collected the productivity factors in software development. We found that many of our metrics for measuring productivity can be found in their list as well. However, our study focuses on technical factors mostly. S. Choudhary et al. [32] evaluated the efficacy of open-source projects. They downloaded 6401 mature projects with a minimum of 10 stars and 3 contributors. They examined the active phases of the project timeline and demonstrated that coordination is associated with an effect during these phases. This measurement technique gave us a solid notion for efficiently locating peak periods in the project timeline. Due to the layered architecture pattern's well-separated components and low development complexity, testability and development productivity are expected to increase with the new version of the layered architecture as well. We measure a large number of projects in order to gain a broader understanding of how WebDAO is utilized by developers around the globe, as well as how it affects productivity and the project lifecycle. In this paper, we present an NLP-based technique for detecting the WebDAO design pattern in Angular-framework-based applications. Using our accurate model, we count the number of projects that contain the design pattern and compare the productivity of WebDAO and NO_WebDAO cases.

4. Materials and Methods

In this section, we present our methodology for accurate design pattern detection and analysis technique for measuring productivity in open-source projects.

4.1. Dataset

We collected projects from GitHub, the largest open-source repository space, in order to gather an enormous quantity of data. Regarding WebDAO, it is essential to examine projects that employ object-oriented concepts and have a back-end. Filtering only repositories with a back-end is not a simple task, but if the query is limited to serverless development, we can guarantee that the projects have a background system with which the client is communicating. We opted to search for Angular projects in order to discover a large number of projects that use a modern, object-oriented, and strongly typed front-end framework. Since Angular 2+ is a Google product and Google Cloud Firebase is a Google platform, the framework's compatibility with the cloud is guaranteed. There are over 1 million Angular projects on GitHub, and over 19,000 of them are associated with a Firebase project. We downloaded all Angular 2+ applications with a Firebase-hosted back-end. Finally, we had an 836 GigaBytes (GB) dataset containing 19,116 open-source projects using Angular 2+ with Firebase. Using the GitHub API, we have collated, in addition to repositories, all issues, pull requests, and commits associated with the projects. Our dataset comprises information through 3 March 2022.

4.2. WebDAO Detection

To detect WebDAO, seven crucial conditions must be met.

- Abstraction: a WebDAO class should provide a well-defined interface or abstraction that encapsulates the underlying data source.
- Data persistence operations: WebDAO should encapsulate the operations necessary to interact with the data source. It must offer CRUD methods for creating, reading, modifying, and removing data entities.
- Data source independence: a WebDAO should insulate the application from the specifics of the data source being used, regardless of whether it is a relational database, a document store, or another type of persistence mechanism.
- Distinct layer: a WebDAO must provide a separate layer from the business logic and the layer of persistence.
- Encapsulation: a WebDAO class must encapsulate the logic and specifics of data source access. It conceals the underlying implementation details, such as SQL queries or specific data access APIs, and provides the application with a clean and consistent API.
- Transactions and error management: A WebDAO typically manages error conditions and provides mechanisms for managing exceptions that may occur during data access. In addition, it may support transactions to guarantee atomicity and consistency across multiple data operations.
- Testability: DAOs should be architected to be readily testable in isolation from the remainder of the application. They can be mimicked or stubbed to write unit tests that ensure the correct operation of the data access logic.

Table 1 provides a summary of techniques for implementing the aforementioned criteria in an Angular project.

Table 1. Criteria and implementation techniques of WebDAO in Angular.

Criteria	Implementation Technique
Abstraction	Must have an interface or class that describes the entity's properties and is utilized for data manipulation.
Data persistence operations	At least the four CRUD operations are implemented for a given entity.
Data source independence	If the data source is modified, the returned values can be processed by the business logic without any source code modifications.

Table 1. *Cont.*

Criteria	Implementation Technique
Distinct layer	WebDAO is implemented in a distinct file, similar to how a service is, and is therefore isolated.
Encapsulation	Data access logic is implemented in an importable or injectable component.
Transactions and error management	Exception handling must be error-specific. In serverless development, SDKs are responsible for this.
Testability	Testability can be controlled if WebDAO is partitioned at the file level.

In Appendix A, Code A1 provides an example of an Angular source code that satisfies the requirements of the WebDAO design pattern. In the example, the entity being described is called Item, and Angular's Http library is used to manage API connection requests. All CRUD operations are implemented, and the entire WebDAO is contained in a distinct service file, allowing for independent testing. Error management is also a component of the Item WebDAO.

Due to the fact that WebDAO is a well-defined design pattern, its presence in a project can be readily identified. To generate usage statistics for this design pattern, an automated solution is required. WebDAO is too complex for string parsing or regular expressions to locate. Although some criteria can be verified, others cannot. NLP techniques can not only recognize texts but also comprehend texts and source codes in context. Word2Vec is a popular method for representing words with vectors and analyzing context by discovering word similarities [33]. It was a significant development in NLP because words are no longer considered atomic units. It is often combined with a decision-tree-based Random Forest classifier, which can produce high accuracy [34]. Google researchers introduced Bidirectional Encoder Representations from Transformers (BERT) for the first time in 2018. It is a more complex machine learning technique using a neural network that manages word embeddings based on the surrounding words' context [35]. We trained both Random Forest classifier and BERT and compared the outcomes.

4.3. Data Preprocessing for Training

To claim that WebDAO is present in a project, the project must satisfy the prerequisites. The distinct layer criterion can only be met if the WebDAO resides in a separate file from the business logic. WebDAO must be implemented as a service when utilizing the Angular framework. If there is no service in the undertaking, WebDAO requirements cannot be satisfied. To determine whether WebDAO is present or not, we organize and analyze only the service files of a project. If all other conditions are met, the sorted service files must be analyzed. The size of a service file is modest, but the total size of services for 19,166 applications is much larger. Since the context is not altered by reformatting the source code and TypeScript and JavaScript codes can be written on a single line, it is reasonable to minify the source codes by removing white spaces and new lines. In a single-line implementation, it is essential to remember that a single-line comment can disable the source code after it. Thus, we also removed single-line comments. In its minified form, the size of the complete service dataset is 248 MegaBytes (MB).

To conduct an exhaustive study, we deemed it necessary to examine more than one ML technique for solving our classification problem. If WebDAO is present and the majority of entities are managed with WebDAO, the project is categorized as employing the WebDAO design pattern; otherwise, it is not. To select a valuable training dataset, we deemed it essential to designate mature projects. We determined the level of maturation based on the number of forks. We chose the 112 GitHub projects with the most forks among those that were downloaded. We manually selected 504 Angular services based on the presence or absence of WebDAO. We discovered that we cannot classify a project based on a single service file discovered at random, so we analyzed the projects on a file-by-file basis. We labeled 247 services as WebDAO and 257 services as not WebDAO in the training and validation datasets, respectively. On the basis of the designated service files, 35 projects were identified as using WebDAO, whereas in 77 projects, WebDAO was not dominant or not present at all. There were 378 service codes added to the training set and 126 files added to the validation set. The dataset was a Comma-Separated Values (CSV) file with 3 columns containing, in order, the label, the project name, and the minified source code.

4.4. Burst Detection and Productivity Measurements

Using DAO is claimed to increase development efficiency; therefore, we intended to find whether WebDAO could have similar effects. In addition, the presence of a design pattern may influence the activity in the project, so we decided to analyze not only the number of lines of code modified but also the number of issues and pull requests and their length. In order to comprehend the development processes and validate the downloaded data, we first investigated the developers' activity in the project. According to the commits and their timestamps, the activity appears to be sporadic rather than constant. In software development, a burst is a time period in which the activity is higher than a calculated average. Lappas et al. [36] developed an algorithm that requires linear time to solve the Maximum Sum Segment problem for a burst sequence. In their study, the burstiness of events in an interval I within a larger sequence S is defined as the difference between the ratios of the frequency of events in I and S and the ratios of the durations of I and S. The formula can be seen in Equation (1). In the context of a project, m represents the age of the project in hours. The duration of every period beginning with l and ending with r is a day, so it takes 24 h. Every tth period, the number of commits between l and r (within 24 h) is calculated and compared to the average number of commits over the lifetime of the project. A burstiness value of 0 indicates that the level of activity is ordinary. Positive burstiness values mean that the activity is above-average and therefore bursty, whereas negative values show that there is no or less activity on the project.

$$B(t,[l:r]) = \left(\frac{\sum_{i=l}^{r} y_{ti}}{\sum_{j=1}^{m} y_{tj}} - \frac{len(Y_t[l:r])}{m} \right) \tag{1}$$

We calculated and displayed the project's timeline using the Maximum Sum Segment algorithm. We also plotted the issues, their period, the commits that belonged to the issues, and the commits that contained modifications to Angular services to validate how developer activity changes. Long active periods are depicted in Figure 3 for a project utilizing WebDAO and a project not utilizing WebDAO, respectively. The bursty periods are depicted by vertical lines colored gray. The open periods of the issues and pull requests are represented by horizontal black lines. The blue bars represent commits that are unrelated to service files, while the green bars represent commits that contain modifications to services. The height of the bars indicates the total number of modified lines during the commit. It is recognized that bursty periods might bring in new bugs or feature requests or fix existing issues. In this instance, the largest commits necessitated modifications to the services.

Regarding development productivity, developer time spent on a task is the most apparent metric. However, we can readily obtain additional metrics by utilizing a version control system. In terms of productivity, the number of modifications, the frequency of commits, the number of issues created, and the percentage of new lines added and deleted can still be decisive factors. Here, we provide data on how developers use WebDAO in Angular and how the design pattern affects the previously mentioned metrics.

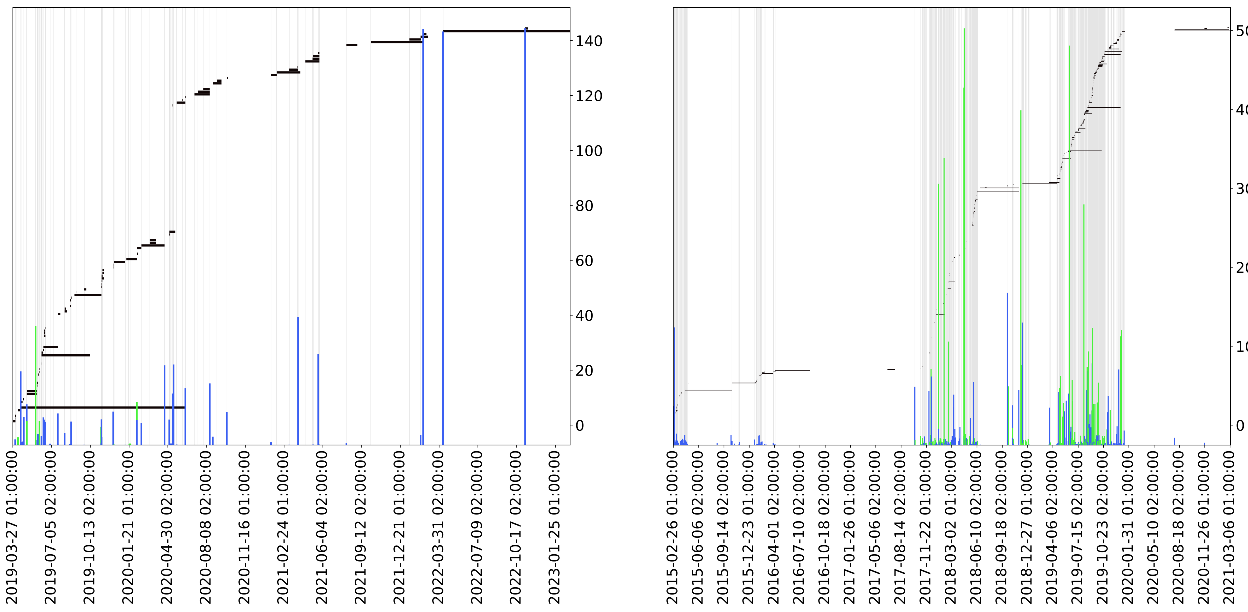


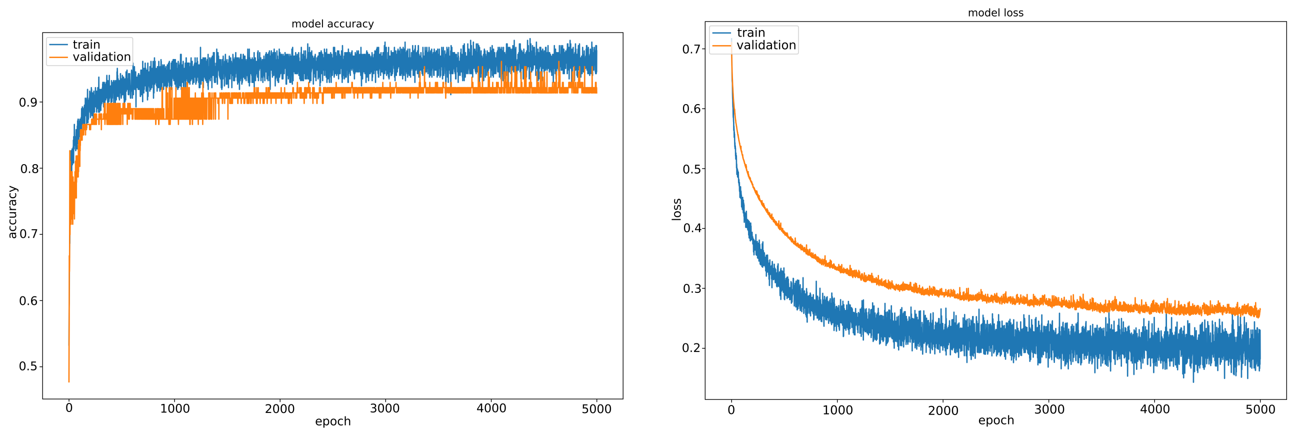
Figure 3. Project commit history and bursty periods for projects using and not using WebDAO.

5. Results

This section presents the achieved results in the classification of Angular projects based on applying the WebDAO design pattern using machine learning and the impact of WebDAO on productivity.

5.1. RQ1: WebDAO Classifier Solutions

Here, we contrast two NLP classification strategies for WebDAO-featured files and projects. In our initial solution, we combined Word2Vec and a Random Forest classifier. Word2Vec facilitates the transformation of words into vectors, and 100 estimators were used to train a Random Forest classifier. Surprisingly, we achieved an F1-score of 89.8%, along with a precision of 86.3% and a recall of 93.6%. The text processing with Word2Vec was managed with the Gensim library [37] and setting the vector size argument to 100 and the window to 5. We used the Random Forest algorithm that is implemented in the scikit-learn package [38] with the default parameters. However, the performance of a neural network on the problem blew us away. We created a BERT model with five layers: one input layer, two Keras layers [39,40], one dropout layer, and one dense layer. In total, 769 out of 109,483,010 parameters are trainable in our BERT model. The training was conducted on an NVIDIA GeForce RTX3060 12 GB GPU (Nvidia Corporation, Santa Clara, CA, USA) with 5000 epochs and 64-epoch batches. The model and text processing component were implemented using the tensorflow, tensorflow_text, and tensorflow_hub packages [41]. As shown by the accuracy and loss curves in Figure 4a,b, this many epochs were required for training. In Table 2, the attained precision, recall, and F1-Score values for both classes are presented. According to the values of the confusion matrix listed in Figure 5, the number of false positives for both classes is incredibly low, and the F1-score for both classes was 92%. Overall, we have a trustworthy BERT model that can designate an Angular class as WebDAO or not.



(**a**) Model accuracy (**b**) Model loss

Figure 4. Learning curves of the BERT model.

Table 2. Precision, recall, and F1-score values of our WebDAO classifier.

Class	Precision	Recall	F1-Score
NO_DAO	91%	94%	92%
DAO	93%	90%	92%

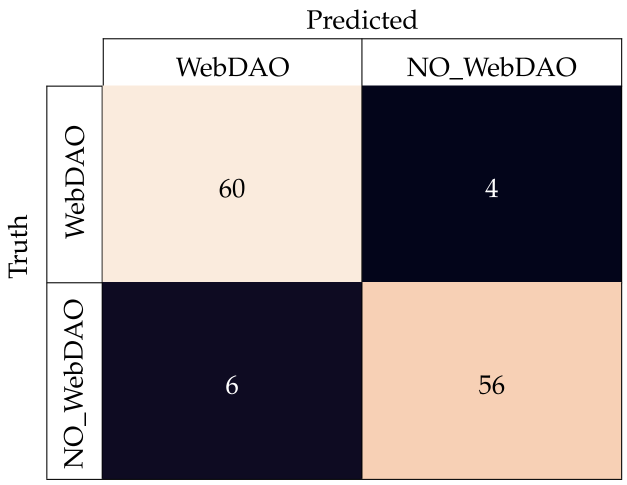


Figure 5. Confusion matrix of BERT training.

5.2. RQ2: Utilization of The WebDAO Design Pattern in Open-Source Projects

During the course of our research, we amassed thousands of open-source projects for the purpose of determining how Angular developers utilize best practices and training a neural network to detect the WebDAO design pattern. Using the trained BERT model, we analyzed the downloaded Angular applications to compile WebDAO design pattern usage statistics. We evaluated all downloaded projects at the file level, and based on this evaluation, we classified the projects as either WebDAO-dominant or not. The total number of projects was 19,116. As shown in Figure 6, 28.09% of them use WebDAO in their services, while 71.91% do not use WebDAO or it is not the dominant technique. It is essential to note, however, that in 27% of projects classified as NO_WebDAO, there are no services at all. Only 34.81% of the service files are designated WebDAO, while 65.19% are not (see Figure 7a). Regarding the projects, a similar ratio can be observed if the 3746 projects in the dataset that do not contain services are removed. In total, 34.94% of the projects that use services employ the WebDAO design pattern, while 65.06% do not (see Figure 7b). Globally, WebDAO is prevalent in one-third of all projects. However, the majority of developers simply adhere to best practices and do not use them explicitly.

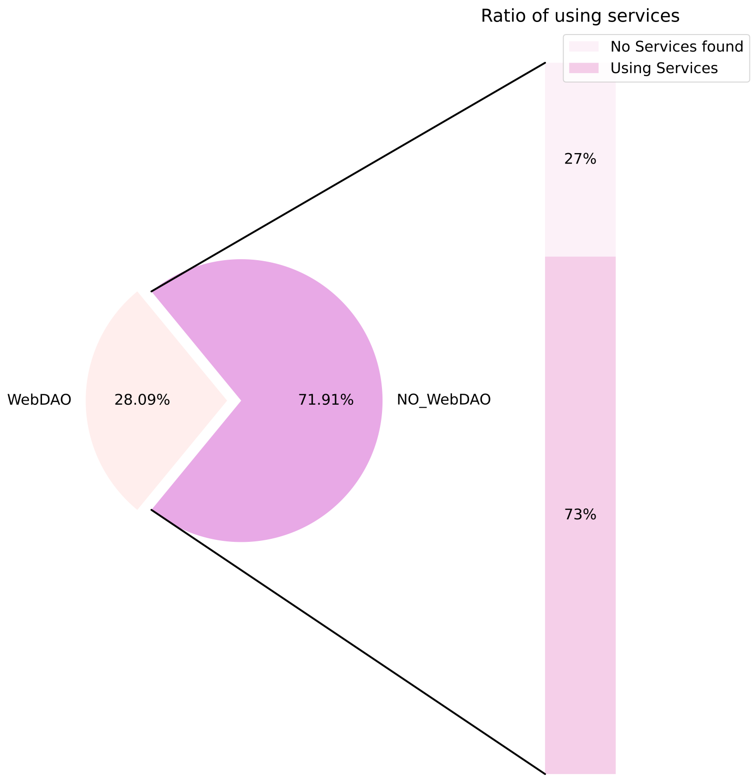
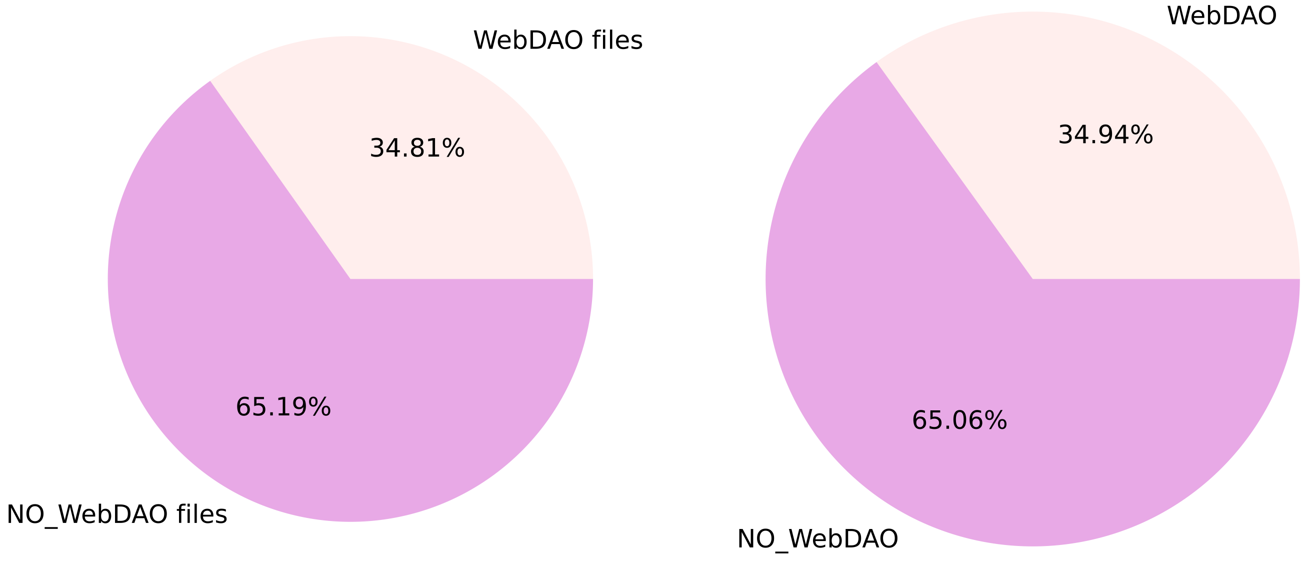


Figure 6. Ratio of using WebDAO in open-source Angular projects.



(**a**) File-level ratio (**b**) Ratio including only services

Figure 7. Ratio of projects utilizing WebDAO including only Angular services and ratio of Angular services that are WebDAO.

5.3. RQ3: Impact of WebDAO on Development Productivity

WebDAO is anticipated to have comparable effects on software development due to its functional and architectural similarities with DAO. Here, we downloaded the entire project history, including commits, commit information, issues, and pull requests, so that we could generate statistics from multiple perspectives. First, we selected projects with comparable metrics in order to conduct a valid comparison. Since productivity cannot be measured on projects with one or two commits or a brief duration, we filtered out mature projects. To be considered mature, a project must have been active for more than six months since its inception. Additionally, there is only one project in the dataset with more than 10,000 commits; therefore, it was eliminated so as not to skew the statistics. It is necessary to have at least one issue or pull request in order to evaluate productivity in terms of bugs and features, so we removed repositories that lacked at least one issue or pull request. Finally, the GitHub bot generates issues for numerous projects due to deprecated package versions. These issues remained open for extended periods of time, so we removed them as well. On this thoroughly cleansed dataset, we made the following observations. It can be seen in Figure 8 that in both WebDAO and NO_WebDAO cases, the number of commits and the number of issues are similar. Using WebDAO reduces the number of code modifications more than twofold, and this is true for both new line insertions and deletions. It indicates that maintainability and code reuse are significantly enhanced. Also, the average duration of each issue is approximately 60 h shorter, which is equivalent to more than an entire week of workdays. In addition, the standard deviation of the length of issues is 71 h less in the WebDAO case, allowing for more accurate estimates of the time required to complete a task or resolve an issue. In the case of WebDAO, there are on average more commits containing modifications to the service code.

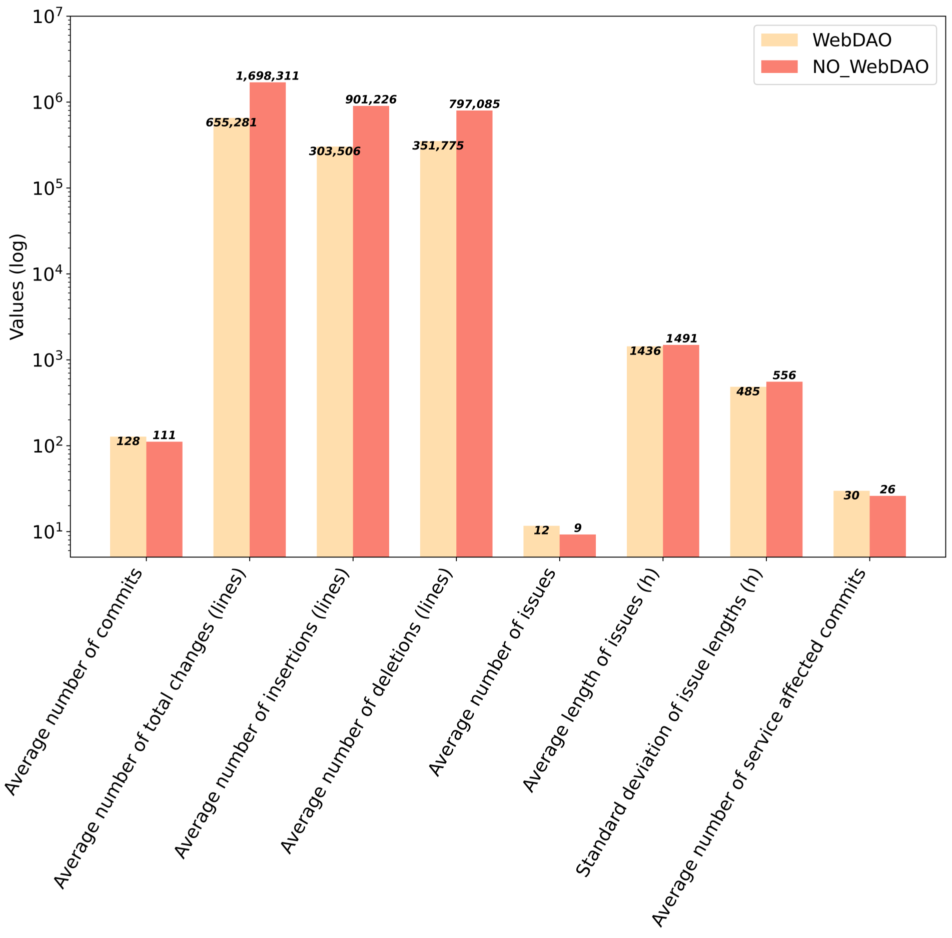
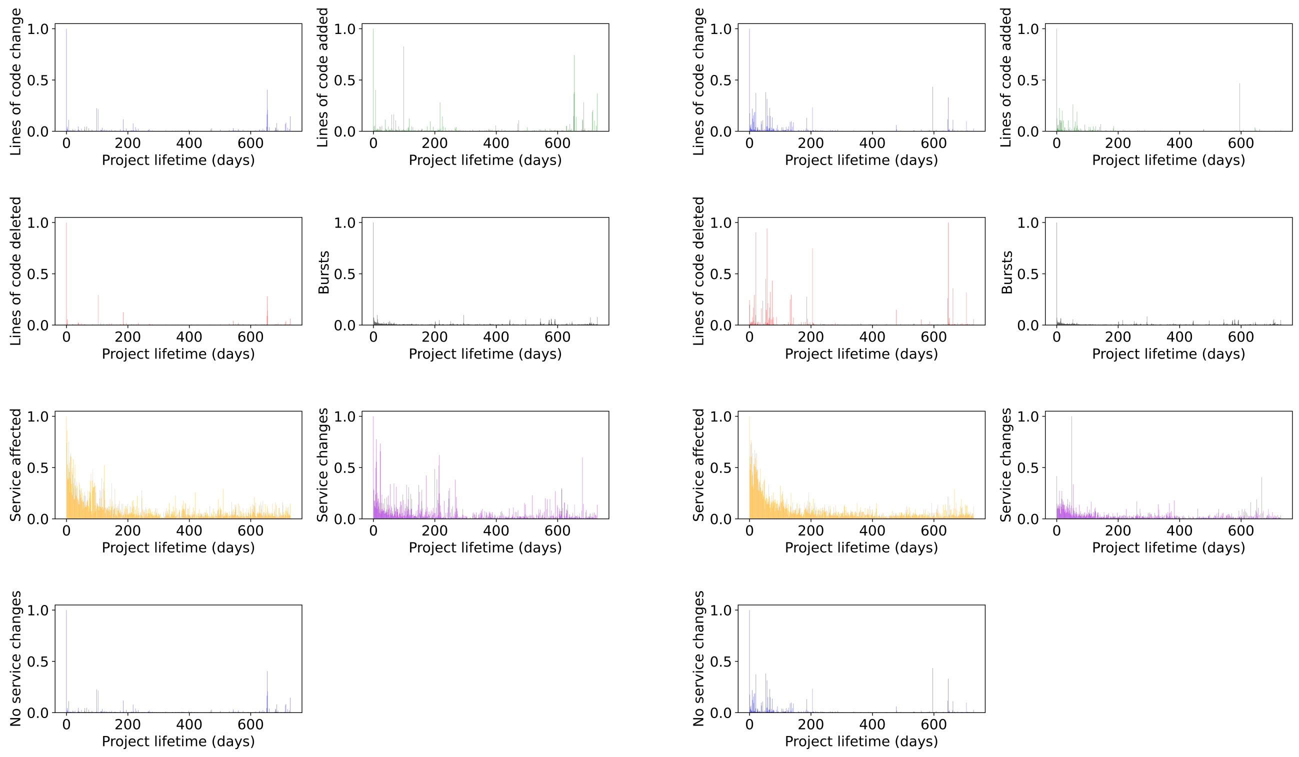


Figure 8. Project metrics dependent on whether or not WebDAO is used.

In Figure 9a,b, we contrasted the WebDAO and NO_WebDAO projects with regard to bursty periods. In these results, we maintained the maturity criterion and displayed the first two years (720 days) of projects with at least six months of activity. Clearly, the greatest amount of activity can be observed at the commencement of the projects. Nonetheless, if the

WebDAO design pattern is not used in the project, this initial active period is significantly longer and more bursty. This applies both to insertions and deletions. The previous statistics regarding service-affected commits have similar results to the fifth chart regarding bursty periods containing service-affected commits, but the WebDAO design pattern improves the maintainability of the services due to fewer lines of code changes. Overall, the WebDAO design pattern introduces the capabilities and benefits of the original DAO design pattern to contemporary front-end development.



(**a**) Project metrics in bursty periods for projects using WebDAO

(**b**) Project metrics in bursty periods for projects not using WebDAO

Figure 9. The first two years of the WebDAO and NO_WebDAO projects with regard to bursty periods.

6. Discussion

As is stated in [31], there are a huge number of technical and soft factors that can significantly influence development productivity. Our study focused on the productivity measurement based on repository metrics and the application of the WebDAO design pattern. Taking into account the technical factors, our methodology measures the productivity in the sense of reusability, platform volatility, and project duration, which are less-researched areas. Our evaluations do not meet the soft factors; however, they are more well-travelled fields.

After overviewing the related works, we found that our trained BERT model is very promising with its 92% accuracy. With this NLP technique, large datasets can be easily analyzed with a relatively small error, so the results are representative.

WebDAO is often implemented as an installable application, such as an SDK. Typically, open-source libraries that employ WebDAO implement broadly adopted open industry standards. Fast Healthcare Interoperable Resources (FHIR) [42] and TeleManagement Forum (TMForum) [43] are excellent examples of how resources and entities are recommended to be described in the healthcare and telecommunications industries. It is common for proprietary projects to implement their own private solutions using WebDAO, making the systems simple to integrate. If proprietary systems are only freely available for integration, it may be possible to provide an open-source SDK while the remaining system components are closed. In general, proprietary projects are typically more standardized, so WebDAO usage can be more prevalent there.

Considering the modern DevOps methodologies, WebDAO supports clean code and thereby offers maintainable software development processes. It is a basic requirement today, since software releases are more frequent.

7. Threats to Validity

There are a few threats to the validity of the statistics used in this study. The threats are listed in four categories below: threats to internal validity, external validity, construct validity, and conclusion validity [44].

Threats to internal validity: We did not identify any threats to internal validity.

Threats to external validity: The fact that team-level observations of the WebDAO design pattern's effects were not made represents one of the most significant threats. This restricts our ability to generalize our findings to broader contexts, such as how this design pattern might affect the productivity or efficacy of entire teams or even companies. Also, developers with more experience may utilize the design pattern more frequently.

Threats to construct validity: It is possible that the design pattern is implemented as a library that is hard to analyze even manually. Lastly, there is the possibility of distortion caused by bot-generated issues. While we have attempted to identify and remedy such issues, we cannot guarantee that all of them have been identified and addressed. The presence of these bot-generated issues may have artificially inflated or deflated our statistics, jeopardizing the reliability of our data and conclusions.

Threats to conclusion validity: We did not identify any threats to conclusion validity.

8. Conclusions

The evolution of software development practices has introduced new dimensions to the application of design patterns, particularly within the domain of Data Access Objects (DAO). Our research delved into this changing landscape and focused on the implementation of WebDAO, a DAO pattern variant relocated to the front-end, in serverless development. By analyzing open-source Angular projects, we aimed to quantify the usage and influence of the WebDAO design pattern on development efficiency. Using a Natural Language Processing (NLP) model expressly trained for this task, we were able to identify the WebDAO design pattern with a remarkable 92% accuracy. This potent tool allowed us to explore the entire history of these projects and obtain invaluable insights. Although WebDAO has never been formally introduced, it has been observed that Angular developers frequently apply it implicitly as a design pattern. Our analysis included a broad variety of metrics, such as the number of commits, changes, and issues, in more than 19,116 Angular projects, providing a comprehensive view of the WebDAO design pattern's influence. The concept that the DAO pattern improves productivity is well established, but its WebDAO counterpart has not been investigated until now. Our findings not only validate the presence of WebDAO in contemporary software development but also shed light on its potential productivity-boosting effects. We demonstrated that WebDAO enhances code reusability and maintainability, allowing for more accurate estimations of the time required to fix bugs and implement new features. We demonstrated that WebDAO reduces the number of source code modifications by more than two times compared to development without this design pattern. Taking into account the duration of development, developers expend approximately 60 fewer hours on implementation if WebDAO is present. Lastly, the smaller standard deviation of issue lengths indicates that developments can be better planned and deadlines can be met with greater ease. Future plans call for extending our analysis to include additional front-end frameworks. In addition, we intend to employ generative language models to enhance software development efficiency and identify additional design patterns in contemporary front-end applications.

Author Contributions: Conceptualization, Z.R.J. and V.B.; methodology, Z.R.J.; validation, Z.R.J.; writing—original draft preparation, Z.R.J.; writing—review and editing, Z.R.J. and V.B.; visualization, Z.R.J.; supervision, V.B.; All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program (RRF-2.3.1-21-2022-00004). Project no. TKP2021-NVA-09 has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

Data Availability Statement: Data used in this research can be found here: https://drive.google.com/drive/folders/1e7a_-iDOUndUMuKoAEWJ5PwcWP3plfxB, accessed on 13 June 2023.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Code A1 is an example of WebDAO implementation in Angular service.

Code A1. WebDAO implementation in Angular service.

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { catchError, map, tap } from 'rxjs/operators';

export interface Item {
  id: number;
  name: string;
  // other properties as needed
}

@Injectable({
  providedIn: 'root',
})
export class ItemDaoService {
  private itemsUrl = 'api/items';  // URL to the web api

  httpOptions = {
        headers: new HttpHeaders({ 'Content-Type': 'application/json' })
  };

  constructor(private http: HttpClient) { }

  /** GET all items from the server */
  getItems(): Observable<Item[]> {
        return this.http.get<Item[]>(this.itemsUrl)
        .pipe(
        tap(_ => console.log('fetched items')),
        catchError(this.handleError<Item[]>('getItems', []))
        );
  }

  /** GET item by id */
  getItem(id: number): Observable<Item> {
        const url = `${this.itemsUrl}/${id}`;
        return this.http.get<Item>(url).pipe(
        tap(_ => console.log(`fetched item id=${id}`)),
        catchError(this.handleError<Item>(`getItem id=${id}`))
        );
  }

  /** POST: add a new item to the server */
  addItem (item: Item): Observable<Item> {
        return this.http.post<Item>(this.itemsUrl, item, this.httpOptions).pipe(
        tap((newItem: Item) => console.log(`added item w/ id=${newItem.id}`)),
        catchError(this.handleError<Item>('addItem'))
        );
  }

  /** DELETE: delete the item from the server */
  deleteItem (item: Item | number): Observable<Item> {
        const id = typeof item === 'number' ? item : item.id;
        const url = `${this.itemsUrl}/${id}`;

        return this.http.delete<Item>(url, this.httpOptions).pipe(
        tap(_ => console.log(`deleted item id=${id}`)),
        catchError(this.handleError<Item>('deleteItem'))
        );
  }

  /** PUT: update the item on the server */
  updateItem (item: Item): Observable<any> {
        return this.http.put(this.itemsUrl, item, this.httpOptions).pipe(
        tap(_ => console.log(`updated item id=${item.id}`)),
        catchError(this.handleError<any>('updateItem'))
        );
  }

  /**
   * Handle Http operation that failed.
   * Let the app continue.
   * @param operation - name of the operation that failed
   * @param result - optional value to return as the observable result
   */
  private handleError<T> (operation = 'operation', result?: T) {
        return (error: any): Observable<T> => {
        console.error(error); // log to console instead
        console.log(`${operation} failed: ${error.message}`);
        return of(result as T);
        };
  }
}
```

References

1. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.M. *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed.; Addison-Wesley Professional: Boston, MA, USA , 1994.
2. Kouli, M.; Rasoolzadegan, A. A Feature-Based Method for Detecting Design Patterns in Source Code. *Symmetry* **2022**, *14*, 1491. [CrossRef]

3. Martin, R.C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*, 1st ed.; Prentice Hall Press: Hoboken, NJ, USA, 2017; pp. 304–306.
4. Jaiswal, M. Software Architecture and Software Design. *Int. Res. J. Eng. Technol.* **2019**, *6*, 2452–2454. [CrossRef]
5. Understand the Most Reliable Frontend Architecture | Bits and Pieces. Available online: https://blog.bitsrc.io/understand-the-most-reliable-frontend-architecture-c8578e3166b (accessed on 13 June 2023).
6. Aggarwal, S.; Verma, J. Comparative analysis of MEAN stack and MERN stack. *Int. J. Recent Res. Asp.* **2018**, *5*, 127–132.
7. Larman, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd ed.; Pearson: London, UK, 2004.
8. Richards, M. *Software Architecture Patterns*, 1st ed.; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015; pp. 1–9.
9. Top JavaScript Trends in 2023: Frameworks & Libraries | Codica. Available online: https://www.codica.com/blog/top-javascript-trends/ (accessed on 13 June 2023).
10. Duarte, C.H.C. Software Productivity in Practice: A Systematic Mapping Study. *Software* **2022**, *1*, 164–214. [CrossRef]
11. Krishnan, M.S.; Kriebel, C.H.; Kekre, S.; Mukhopadhyay, T. An Empirical Analysis of Productivity and Quality in Software Products. *Manag. Sci.* **2000**, *46*, 745–759. [CrossRef]
12. Mills, H.D. *Software Productivity*; Dorset House Publishing: New York, NY, USA, 1988.
13. Trendowicz, A.; Münch, J. Chapter 6: Factors Influencing Software Development Productivity—State-of-the-Art and Industrial Experiences. *Adv. Comput.* **2009**, *77*, 185–241.
14. Boehm, B.W. Improving Software Productivity. *Computer* **1988**, *20*, 43–57. [CrossRef]
15. Jeremy, D.K.; Sfenrianto, S. Analysis Software Developer Productivity Based on Work Schedule Scheme with Git Commit Metric and Deployment with CI/CD. *J. Pendidik. Konseling* **2022**, *4*, 966–977. [CrossRef]
16. Latte, B.; Henning, S.; Wojcieszak, M. Clean Code: On the Use of Practices and Tools to Produce Maintainable Code for Long-Living Software. In Proceedings of the 6th Collaborative Workshop on Evolution and Maintenance of Long-Living Systems, Stuttgart, Germany, 18–22 February 2019.
17. Xing, Y.; Huang, J.; Lai, Y. Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development. In Proceedings of the 2019 11th International Conference on Computer and Automation Engineering, Perth, Australia, 23–25 February 2019. [CrossRef]
18. Savolainen, J.E.; Myllärniemi, V. Layered Architecture Revisited—Comparison of Research and Practice. In Proceedings of the 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, Cambridge, UK, 14–17 September 2009. [CrossRef]
19. Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*; John Wiley & Sons: Chichester, UK, 1996.
20. Shklar, L.; Rosen, R. *Web Application Architecture: Principles, Protocols and Practices*; John Wiley & Sons Ltd.: Chichester, UK, 2003.
21. Cincovic, J.; Delcev, S.; Draskovic, D. Architecture of web applications based on Angular Framework: A Case Study. In Proceedings of the 9th International Conference on Information Systems and Technologies (ICIST 2019), Cairo, Egypt, 24–26 March 2019; pp. 254–259.
22. Angular Architecture Patterns—High Level Project Architecture * NETMedia. Available online: https://netmedia.agency/dev/angular-architecture-patterns-high-level-project-architecture_5589 (accessed on 13 June 2023).
23. Geetha, G.; Mittal, M.; Mohana, P.K.; Ponsam, G. Interpretation and Analysis of Angular Framework. In Proceedings of the International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 8–9 December 2022.
24. Matic, D.; Butorac, D.; Kegalj, H. Data Access Architecture in Object-Oriented Applications Using Design Patterns. In Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, Dubrovnik, Croatia, 12–15 May 2004. [CrossRef]
25. Inclouded. Available online: http://inclouded.hu/ (accessed on 13 June 2023).
26. @inclouded/fhirapi—npm. Available online: https://www.npmjs.com/package/@inclouded/fhirapi (accessed on 13 June 2023)
27. Rathinam, S. Analysis and Comparison of Different Frontend Frameworks. In Proceedings of the Applications and Techniques in Information Security, Manipal, India, 30–31 December 2022. [CrossRef]
28. Al-Hawari, F. Software Design Patterns for Data Management Features in Web-Based Information Systems. *J. King Saud Univ.—Comput. Inf. Sci.* **2022**, *34*, 10028–10043. [CrossRef]
29. Komolov, S.; Dlamini, G.; Megha, S.; Mazzara, M. Towards Predicting Architectural Design Patterns: A Machine Learning Approach. *Computers* **2022**, *11*, 151–170. [CrossRef]
30. Aniche, M. Java Code Metrics Calculator (CK). 2015. Available online: https://github.com/mauricioaniche/ck/ (accessed on 13 June 2023).
31. Wagner, S.; Ruhe, M. A Systematic Review of Productivity Factors in Software Development. *arXiv* **2018**, arXiv:1801.06475. [CrossRef]
32. Choudhary S.; Bogart, C.; Rose, C.; Herbsleb, J. Using Productive Collaboration Bursts to Analyze Open Source Collaboration Effectiveness. In Proceedings of the 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), London, ON, Canada, 18–21 February 2020. [CrossRef]
33. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the International Conference on Learning Representations, Scottsdale, AZ, USA, 2–4 May 2013. [CrossRef]

34. Vora, P.; Khara, M.; Kelkar, K., Classification of Tweets based on Emotions using Word Embedding and Random Forest Classifiers. *Int. J. Comput. Appl.* **2017**, *178*, 1–7. [CrossRef]
35. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019. [CrossRef]
36. Lappas, T.; Arai, B.; Platakis, M.; Kotsakos, D.; Gunopulos, D. On burstiness-aware search for document sequences. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009. [CrossRef]
37. Rehurek, R.; Sojka, P. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, Valletta, Malta, 22 May 2010; pp. 45–50.
38. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
39. bert_en_uncased_preprocess. Available online: https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3 (accessed on 13 June 2023).
40. bert_en_uncased_L-12_H-768_A-12. Available online: https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4 (accessed on 13 June 2023).
41. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf (accessed on 18 July 2023).
42. Index—FHIR v5.0.0. Available online: https://www.hl7.org/fhir/ (accessed on 17 July 2023).
43. TM Forum—How to Manage Digital Transformation, Agile Business Operations & Connected Digital Ecosystems. Available online: https://www.tmforum.org/ (accessed on 17 July 2023).
44. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering: An Introduction*; The Kluwer International Series in Software Engineering; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2000.