ISSN No:-2456-2165

Version Control using Layered Architecture and Design Patterns for GITHUB

Ayisha Siddiqua L[1]
[1]M.tech Software Engineering, VIT University, Vellore-632014, Tamil Nadu, India

Dr.A.Gayathri[2]
[2]Assistant Professor (Senior), Dept. of Software Engineering, VIT University
Vellore-632014,Tamil Nadu, India

Abstract:- Using distributed version controls, most of the software projects can be easily shared and managed through online websites which lets developers and also projects leaders, supporters and students who develop so many codes and who want to resolve bugs and errors to move forward in their work [1]. Github is one such example where version control is used as there are many projects that are being shared in that website by many software developers. In the existing system MVC (Model View Controller) was used but in the proposed system we use Layered Architecture and one design pattern to resolve the complexity in version control system for Github [1].

Keywords:- *Version Control, MVC, Github, Layered Architecture, Design Patterns.*

I. INTRODUCTION

Design patterns can be seen as a way to accomplish enormous scale reuse by catching effective software development design practice inside a specific setting. Patterns may not be restricted in what they can depict and can be utilized to typify great design rehearses at both the determination and usage levels. In this way, design patterns can be applied at a wide range of levels of reflection in the software improvement life-cycle, and can concentrate on reuse inside architectural design just as point by point design and execution. We build up layered quantum-computer architecture, which is a deliberate system for handling the individual difficulties of building up a quantum computer while developing a durable gadget design. In the existing system MVC (Model View Controller) was used but it is not that reliable and complexity increases. As Github is a modern user interface it becomes difficult to use MVC. Also, MVC architecture concentrates only on presentation so we use Layered Architecture because it focuses on entire system and it divides the entire application into meaningful groups that is presentation, business logic and data storage in database and one design pattern to resolve the complexity in version control system for Github and also to reuse some functionality. We have proposed the use of linked list in place of trees for version control for better memory utilization [2].

- *MVC (Model View Controller)*

In the existing system where MVC is used they mainly focused on the characteristics of developers like “characteristics”, “initiative” and “leadership” rather than the application that is Github. Github is the platform where the projects are uploaded by many software developers and students. Since Github is transparent many users can access this and upload their files. If there are any errors or any bugs they can be easily fixed by others also. In this way it helps developers. In Github we can see the person who developed the project, name of the project, name of the project owner, overview for the project, URL and it also contains the programming languages being used by the developers and students which will give us the result which programming language has more demand and which programming language have more scope. As these functionalities increase day by day it becomes complex to use MVC so in this paper we have proposed layered architecture.

II. SYSTEM ARCHITECTURE

In the existing system Layered architecture is used and it describes mainly about the system and not about the characteristics of the developer. Layered Architecture focuses on entire system and it divides the entire application into meaningful groups that is presentation, business logic and data storage in database. In figure 1, these three layers contain many components. Presentation layer is the one layer that is visible to be the users, that shows UI designs like the home page interface and it is also used to show on which language the interfaces are developed such as HTML, CSS, JavaScript. In business layer, business workflows, business entities, business, components are shown and authentication services can be done for login pages to encrypt and decrypt passwords for security purposes [7]. Third layer (Figure 1) is database layer where all data are stored for future purposes. All login and sign up details will be stored in the database. In order to provide security in database we use many Algorithms [7].

ISSN No:-2456-2165

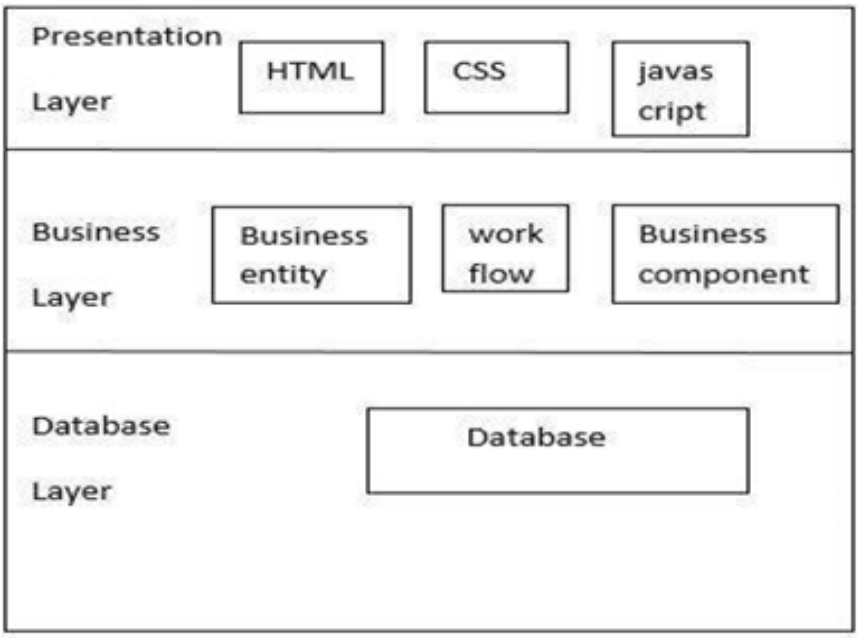

Fig 1:- Layered Architecture

III. PROPOSED DESIGN PATTERN

The purpose of design pattern is to decrease the complexity and it can be reused so that resources will be saved. There are totally 23 design patters which are used based on the application scenario and for Github we use design pattern that is decorator pattern [5].

➢ *Justification for the Design Pattern*

As Github is a platform where functionalities need to be added day by day so it is better to use decorator design pattern as its main aim is to add functionalities without changing the existing system and also, we can retrieve the old system even after removing some functionalities that are unnecessary. For example: the user may want to get SMS on latest projects and not just mail so we can add SMS function along with mail function that is already there. Adding SMS function to the existing system becomes easy when we use decorator pattern [3].

➢ *Intent of Decorator Pattern:*

Attach additional responsibilities to an object dynamically. Decorator provides flexible alternative for sub classing to extend functionality. It is also known as wrapper.

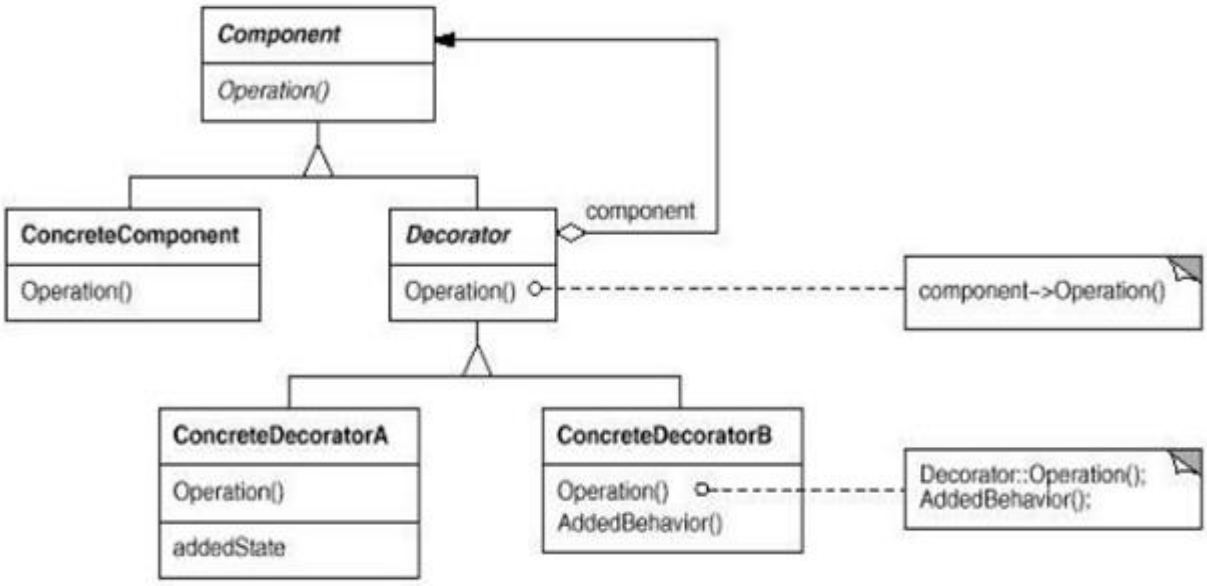

Fig 2:- Structure of Decorator Pattern

ISSN No:-2456-2165

➢ *Participants:*

- *Component (Visual Component)*

It defines an interface for objects which have responsibilities that are added dynamically to them (Figure 2).

- *Concrete Component (TextView)*

It defines object on which some additional responsibilities may be easily attached (Figure 2).

- *Decorator*

Maintains a reference to a Component objects and also it defines the interface which conforms to Component's interface (Figure 2).

IV. REVISION CONTROL FOR

➢ *Versions*

It helps to add new responsibilities to the component. In the existing system they used one software tool that is Revision Control System which will assess the task[2]. It was developed for programming environment and was used to store source programs, specifications, documentation and also test data. It is also used to manage graphics in computers, VLSI layouts, form letters, book chapters, papers etc.

Versions are depicted in the form RCS revision tree. Tree consists of nodes and branches. In these nodes are version and they are connected to other versions through branches [2].

There are three types, the first type (Figure 3) is young slender which doesn't have any further branches [2].

Fig 3:- Young Slender

The second type (Figure 4) is a revision tree with one side branch.

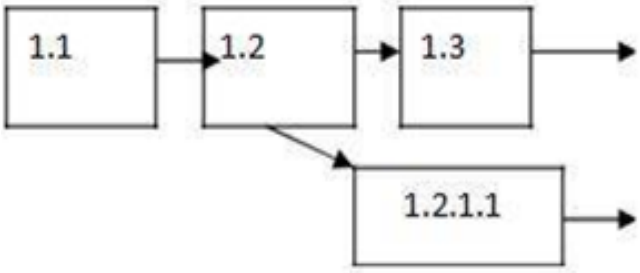

Fig 4:- Revision Tree with One Branch

The third type (Figure 5) is a revision tree with local modification

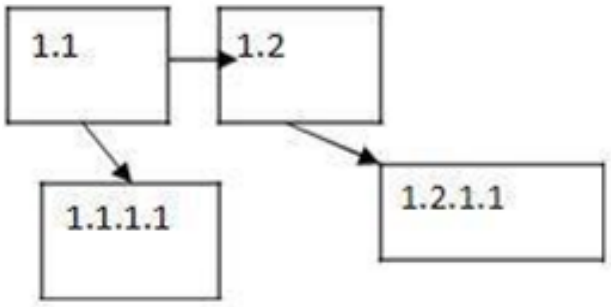

Fig 5:- Revision Tree with Local Modification

V. PROPOSED SYSTEM FOR VERSION CONTROL

In this paper the proposed system suggests that we used linked list instead of trees with nodes and branches [7]. As linked list has two fields that is data field and address field it will store the address of the next field which will help to access the next data details easily. Insertion and deletion of nodes can be done easily by just changing the address filed. It is dynamic so anything can be changed during the run time. Memory is also utilized in an effective manner by just using struct and malloc function in the program [6].

VI. CONCLUSION AND FUTURE WORK

As Github is a modern user interface it becomes difficult to use MVC. Also, MVC architecture concentrates only on presentation so we propose Layered Architecture because it focuses on entire system and it divides the entire application into meaningful groups that is presentation, business logic and data storage in database and one design pattern to resolve the complexity in version control system for Github and also to reuse some functionality. We have also proposed the use of linked list in place of trees for version control for better memory utilization. We intend to build up a suggestion system to pick reasonable designers to cooperate for specific works and undertakings in GitHub. We could then re-measure the properties after our proposal system is applied to a subset of the designer and undertaking quantity and examine if there is any particular change in the properties of the sub-organize. It is important to comprehend GitHub and other online web based environments that give a basic and incorporated approach to repositories permitting people and undertakings of any size to work together in innovative ways.

REFERENCES

[1]. Li, S., Tsukiji, H., & Takano, K. (2016, March). Analysis of software developer activity on a distributed version control system. In 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA) (pp. 701-707). IEEE.

[2]. Tichy, W. F. (1985). RCS—a system for version control. *Software: Practice and Experience*, *15*(7), 637-654.

[3]. Agrawal, K., Aschauer, M., Thonhofer, T., Bala, S., Rogge-Solti, A., & Tomsich, N. (2016, September). Resource classification from version control system logs. In *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)* (pp. 1-10). IEEE.

[4]. de Moura, M. H. D., do Nascimento, H. A. D., & Rosa, T. C. (2014, September). Extracting new metrics from Version Control System for the comparison of software developers. In *2014 Brazilian Symposium on Software Engineering* (pp. 41-50). IEEE.

[5]. Gioachin, F., Liang, Q., Yao, Y., & Lee, B. S. (2012, December). Protego: In-Memory Version Control System in the Cloud. In *2012 19th Asia-Pacific Software Engineering Conference* (Vol. 1, pp. 232-239). IEEE.

[6]. Aslan, K., Skaf-Molli, H., & Molli, P. (2012, May). Connecting Distributed Version Control Systems communities to linked open data. In *2012 International Conference on Collaboration Technologies and Systems (CTS)* (pp. 242-250). IEEE.

[7]. Lass, M., Leibenger, D., & Sorge, C. (2016, November). Confidentiality and authenticity for distributed version control systems-A mercurial extension. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)* (pp. 1-9). IEEE.

[8]. De Nies, T., Magliacane, S., Verborgh, R., Coppens, S., Groth, P. T., Mannens, E., & Van De Walle, R. (2013, October). Git2PROV: Exposing Version Control System Content as W3C PROV. In *International Semantic Web Conference (Posters & Demos)* (pp. 125-128).

[9]. Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development*. " O'Reilly Media, Inc.".

[10]. Blischak, J. D., Davenport, E. R., & Wilson, G. (2016). A quick introduction to version control with Git and GitHub. *PLoS computational biology*, *12*(1), e1004668.