IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

BUG ADMINISTRATIVE SYSTEM USING DISTRIBUTED CLIENT SERVER COMPUTING TECHNOLOGY

C. Rukmani[1] M.Sc.,M.Phil., V.SATHIYA PRIYA[2] M.Sc.,M.Phil.,G.Archana[3] M.Sc.,M.Phil.,B.Ed.,
Assistant Professor
Department of Computer Science
Adthiyaman Arts and Science College for Women
Uthangarai, Krishnagiri, India

ABSTRACT

Bug Administrative System useful for application development in an organization. The Bug Administrative System is a web based application that can be accessed throughout the organization. This system can be used for logging bug against an application/module, assigning bug to individuals and tracking the bugs to resolution. There are features like email warning, user protection, user admission control, report maker etc in this system. It has been designed to be having the view of distributed architecture, with central storage of the database. The application for the storage of the data has been planned. Using the build of MS-SQL Server and all the user interfaces has been planned using the ASP.Net technologies. The database connectivity is planned using the "SQL Connection" methodology. The set of security and data protective mechanism have been given a big choice for proper usage. The application takes care of different modules and their associated reports, which are produced as per the relevant strategies and standards that are put send out by the administrative staff.

Keywords: Administrative system, warming, protection, distributed architecture etc.

1. INTRODUCTION

Bug Administrative System is the framework which empowers to decide the bugs. It is not just decides the bugs but rather gives the entire data with respect to bugs detected. Bug Administrative System guarantees the client of it who has to think about a give data in regards to the recognized bug. Utilizing this no bug will be unfixed in the created application. The designer builds up the task according to client prerequisites. In the testing stage the analyzer will distinguish the bugs. At whatever point the analyzer experience period number of bugs he includes the bug id and data in the database. The analyzer reports to both task chief and engineer. The bug points of interest in the database table are open to both task director and developer. When a client puts demand or requests for an item to be produced. The task administrator is in charge of adding clients to Bug Administrative System and appointing project to the clients.

1.1 PURPOSE AND SCOPE

The purpose of Bug Administrative System for improving software reliability is to provide better service to the administrator or useful for applications developed in an organization. The "Bug Administrative System for Improving Software Reliability" is a web based application that can be accessed throughout the organization. This system can be used for logging bugs against an application/module, assigning them to team members and tracking them for resolution. There are features like email notifications, user maintenance, user access control, report generators etc.

The principle targets of the Bug Administrative System are:

- Identifying the bugs in the created application.
- No bug will be unfixed in the created application.
- Not just distinguishing the bugs yet in addition giving the bug data. When the bugs are distinguished. They are accounted for to the task director and designer. To guarantee that that has to think about the bug can learn not long after it is reported.

The Bug Administrative System begin in this chapter is placed exact in between the close notes on your monitor and an advanced .The Bug Administrative System allows you and your team members to collaboratively file, change, and report on bug through a web interface. The Bug Administrative can be set up on server connected to the local network or the Internet, so it can be achieve from any suitable location. It uses a role-based security mechanism, allowing you to decide who in your team can perform which operation.

The Bug Administrative System allows you to bring out four important tasks:

- finding bugs,
- changing bugs reporting,
- about bugs,
- Application maintenance as using components.

The role-based security mechanism implemented in the Bug Administrative System permit access to each of these features to the roles defined in the system. Each user should be hand over to at least one role so they can log in and perform one of these tasks.Bug Administrative System is a web-based application designed to help a workgroup keep track of bug and tasks using a shared central resource. The system was planned specifically with the IT department in mind, where quick access to shared data and history is a requirement, both from an internal organizational side, as well as to complete the needs of the clients. It supplies one top solution for all the bug issues in the software growth.

1.2 PROBLEM STATEMENT

Bug Administrative System we are facing so many problems apart from of which system is used. In this section we are discussing some of the annoyances. In the problem system, of my thesis the task supervisor assigns the tasks to the builders. The builder develops the projects as per purchaser requirements. The assignment supervisor itself assigns the evolved programs to the tester for testing. Inside the checking out section, when the tester encounters no. of bugs then he/she reviews to the challenge supervisor and developer about the computer bug information.

While developing a web-based application of a Bug Administrative System designed to help a workgroup keep track of bug and tasks way of shared middle resource. The system was planned specifically with the IT department in mind, where fast access to shared data and history is a requirement, both from an inside of organizational viewpoint, as well as to fulfill the needs of the client. The current system is mostly concerned with the storing bug onto the file system with little to no traceability, making the tracking of the bug difficult at a later time. Most of the work of inserting the bug and tracking them back at some later point of time requires human intervention and is done manually. This makes the system limited and hence results in degraded performance.

In this research having information retrieval is a very big process and it becomes very difficult to handle huge databases manually with same efficiency and at the same time with the increase in the database the time to retrieve the concerned information also increases manifolds.

Lack of organization of the files makes it flat to information loss due to accidental deletion of files. No security because the files are visible to the users. More over every user has the same level of access to the files. Report generation is a big task and precision is as much important as output is most of the work is done by humans with minimum to no intervention by machines.

Humans are subjected to other factors like stress, emotions etc. that may reduce their work efficiency which is not the case with the machines, hence prolonged and maintained efficiency.

1.2.1 Bottlenecks of the Problem System

The tester report which is called “bug report” file is within the form of physical record .If the reports damaged then the whole data approximately the malicious program may be lost. The computer bug records aren’t stored inside the database for previous system.

1.3 PROBLEM SOLUTION

Find the solution of my thesis for the Bug Administrative System is to test the application for the bug and report it to the task director and designer. The principle expectation behind the Bug Administrative System is that to track bug and report them. Store the bug data with a one of a kind id in the database for future reference. In this way, this makes the activity of taking care of the bug simple.

The primary destinations of the Bug Administrative System are:

- Identifying the bug in the created application.
- No bug will be unfixed in the created application.
- Not just recognizing the bug yet in addition giving the bug data.
- As soon as the bug are distinguished. They are accounted for to the undertaking chief and engineer.
- To guarantee that has to think about the bug can learn not long after it is reported.

. Other terminology frequently used to describe this process includes:

- Problem Tracking
- Change Management
- Fault Management

2. USES OF BUG ADMINISTRATION

An effective bug tracking system is important for quick design of complex blocks and systems. A central database which collects all the known bugs and desired enhancements allows the whole team to know the state of the design and prevents designers from debugging the known problems several times. It also makes sure that known problems are not forgotten.

Bug rate administration is another main use for bug tracking. The most effective testing and debug strategy for any phase of the project can be defined by the current bug rate and position on the curve. When integration begins, then usually formal bug administration also begins, that is when the work of two or more designers is combined in to a larger block. At all stages of design, some form of bug tracking is required (**Pierre Bricaud, 2002**).

The comparison to elections and publicity is truly amazing given that campaign reform and an attempt to end undue influence returning to a "one person, one vote" ideal has been at the forefront of politics for years **(Laffoon 2003)**. **Hooimeijer and Weimer** observed that bug reports with more comments get fixed sooner. They also noted that bug reports that are easier to read also have shorter life times more recently, **Aranda** and **Venolia** have examined communication that takes place between developers outside the Bug Administrative System.

In our previous work, we discussed shortcomings of obtainable Bug Administrative Systems, which led to the four areas of enhancement presented in this paper. Asking the right questions, is a crucial part of debugging and several techniques such as algorithmic debugging and the Why Line tool support developers in doing so. **Perry et al.,** for example, studied individual developers‟ superficial and actual time allocation for various activities throughout their day-to-day work. Their studies establish that over half of each developer's time was tired act together with co-workers. Her work identified not only the technical axis of knowledge relationship undertaken by software developers, but also its social axis. Their study found that developers at Microsoft "reported spending nearly half their time fixing bugs." Although much analysis has been done on software development in general, little if any of this attention has been directed toward the in-depth study of how issue or Bug Administrative Systems fit into the role of facilitating communication and collaboration, especially within the context of small, collocated teams that are still the norm in many organizations.

2.1 MULTIPLYING PRODUCTS

Matthew B. Doar, 2005 was proposed very simple values for Bug's information. They are a string of text is used to describe to bug or a problem, the name of the person assigned to a bug. In this the field holds only one value at a time, and those things are very simple. In this when one field can have various values at the same time every one becomes more complex. In this we have to possible we are avoiding the fields with multiple values .By avoiding these multiple values for a fields they tend to make writing useful information much harder. We are imagining a Bug Tracking Tool which contains only three fields in its bugs. They are Owner, Description, and Product. In this Owner is a single value field. That means only one person owns a bug at a time. Description is a text string. And the Product is the multi valued fields; it represents special products that are exaggerated by a bug **(Matthew B. Doar, 2005).**

2.2 ONE BUG, MULTIPLE RELEASES

Matthew B. Doar, 2005 was proposed another approach, for bug is a group of various Releases that the bug exists in. In this Bug Tracking System it is the simplest way to deal the bug is abscond the information about the affected releases. We are maintaining a Spreadsheet for each and every Release and it is easy to organize with the development team for identifying where the bug is fixed. This approach is tiresome and make flat to error. But it is common on smaller projects.

Another approach for handle a bug is to prepare two copies of an original bug and we are changing the value for the release found in each of the two bugs. In this Bug Tracking System each of the copies will have its own unique bug identifier. This approach is very useful we know the bug count for each and every release. And some of the Bug Tracking Systems support duplicating bugs mostly. In this Bug Tracking System disadvantage is information will regularly be added to just one copy and not the others. The main disadvantage is that developers, customers and product managers locate it difficult to maintain which bug is fixed in which release. In some of the Bug Tracking Tools maintain to support adding multiple releases for a bug. But their reports are not robust as might be expected when we are using the multiple release values. Keeping track of Bugs in multiple releases of a product it is hard to automatically and not suit in existing Bug Tracking Tools **(Matthew B. Doar, 2005).**

2.3 CUSTOMIZING THE BUG ADMINISTRATIVE SYSTEM

One common customization is changing the state of a bug in order to make them better to fit in the projects presented workflow. The administrator of the system changes the name of the name of the system for each field **(Matthew B. Doar, 2005).**

The locations of the bugs are determined by, in which stage the bug appeared or discovered; and in which place in the system it appeared. **Fry** and **Weimer (2010)** defined fault localization as: "The task of determining if a program or code fragment contains a bug, and if so, locating exactly where that defect resides". As we attempt to enhance the quality control in the software, we have to recognize different phases of the software development such as: Analysis, Design, Testing and Maintenance. At the research context, we will concentrate only on the phases: (Maintenance and Testing).

3. DISTRIBUTED CLIENT SERVER COMPUTING TECHNOLOGY

3.1 THE CLIENT-SERVER MODEL IN A DISTRIBUTED COMPUTING SYSTEM

Challenging to selecting a client server model in my thesis. Because distributed computing system to perform request and response concept only. In this concept not a simple inside of system using number of protocols. A protocol is nothing set of rules and regulation. Here I am using HTTP protocols. A distributed computing system is a set of application and system programs, and data dispersed across a number of independent personal computers connected by a communication network. In order to provide requested services to users the system and relevant application programs must be executed. Because services are provided as a result of executing programs on a number of computers with data stored on one or more locations, the whole computing activity is called distributed computing[1],[2].

3.1.1 Basic Concepts

The problem is how to formalize the development of distributed computing. The above shows that the main issue of distributed computing is programs in execution, which are called processes. The second issue is that these processes cooperate or compete in order to provide the requested services. This means that these processes are synchronized. A natural model of distributed computing is the client-server model, which is able to deal with the problems generated by distribution, could be used to describe computation processes and their behavior when providing services to users, and allows design of system and application software for distributed computing systems.

According to this model there are two processes, the **client**, which requests a service from the user's point of view a distributed computing system can provide the following services: printing, electronic mail, file service, authentication, naming, database service and computing service. These services are provided by appropriate servers. Because of the restricted number of servers (implied by a restricted number of resources on which these servers were implemented), clients compete for these servers [1].

An association between this abstract model and its physical implementation is shown in Figure: 3.1. In particular the basic items of the model: the client and server, and request and response are shown. In this case, the client and server processes execute on two different computers. They communicate at the virtual (logical) level by exchanging requests and responses.

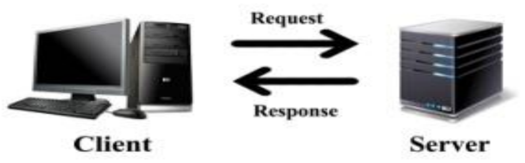

Figure 3.1: The basic client-server model

3.1.2 Environment

The item condition comprises of the accompanying things:

Users

People who utilize the system.

Network

Organization intranet for secure communications.

Database

Main persistent storage for storing system data, including projects, user and group information, bug reports, etc. It also handles user queries for locating database records.

File system

Additional storage format for keeping the exported queries and templates.

Web server

Server for manipulating HTTP requests and generating HTML pages to the requesting clients.

3.2 OVERALL DESCRIPTION

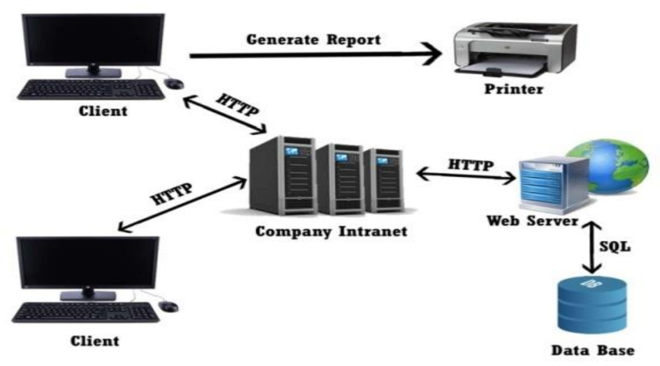

Fig:3.2 The Relationship

3.2.1 System Description

The following diagram describes the system and its environment in general. The bug administrative application will execute inside the web server and the users communicate to the system by using their web browsers. The system stores data in the database retrieves and manipulates the data according to the user's request, then displays the results in HTML on user's browser [1].

In order to allow a client and a server to exchange requests and responses, there is a need to employ a communication network that links computers on which these processes run. We also established in order to locate a server; the operating systems must be concerned. The question is what would be the architecture of a distributed computing system that supports a distributed application developed on the basis of client-server model. Figure 3.2 illustrates the relationship between such a distributed application, the operating system supporting it and communication ability of a distributed computing system [1].

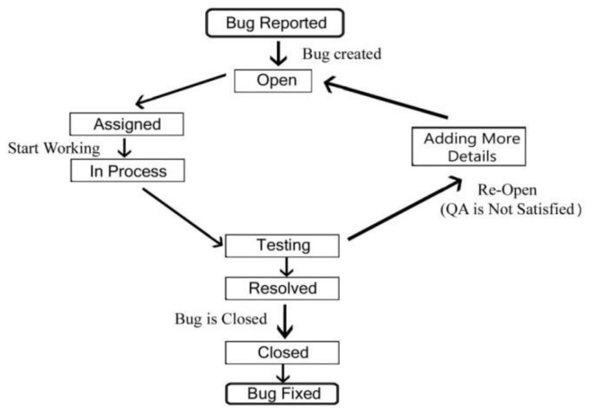

Fig:3.3 Bug Life Cycle

4. CONCLUSION

It has been a great pleasure for me to work on this exciting and challenging research. This research proved good for me as it provided practical knowledge of not only programming in ASP.NET and VB.NET web based application and no some extent Windows Application and SQL Server, but also about all handling procedure related with **Bug Administrative System using distributed client server computing technology.** It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

APPENDIX III

REFERENCES

[1] “Modelling for Distributed Network Systems: The Client-Server Model”. Springer, Boston, MA In: Distributed Network Systems. Network Theory and Applications, vol -15. (2005)

[2] “Distributed Network System From Concept to implementations”, Jia,W:Zhou,W. 2005, XXVII,513 p,,Hardcover ISBN:978-0-387-23839-5

[3] “Architecture of Network and Client-Server mode” Zhang,H.(2013).l. *arXiv preprint arXiv:1307.6665*.

[4]” Effective Bug Tracking Systems: Theories and Implementation**”** Akhilesh Babu Kolluri, K. Tameezuddin, Kalpana Gudikandula [1]*Department of CS, DRK Institute of Science & Technology, Ranga Reddy, Andhra Pradesh, India*

[5] Programming C#: Building .NET Applications with C# By Jesse Liberty

[6] Learning C# 2005 by Jesse Liberty

[7] http://csharp.net-informations.com/data-providers/csharp-sqldataadapter.html

[8] Microsoft MSDN

[9]. http://netservicesindia.com/blog/?p=35

[10]http://www.zeali.net/mirrors/w3cshool/aspnet/aspnet_newfeatures.asp.html