DESIGN OF SOFTWARE PATTERN FRAMEWORK BASED ON AUTOMATIC TEST SYSTEM

[1]Kunthumalla Latha, [2]Dr.Dhanraj Verma
[1]Research Scholar, Dept. of C.S.E, Dr.A.P.J. Abdul Kalam University, Indore- Dewas Bypass Road, Indore, M.P, India.
[2]Research Guide, Dept. of C.S.E, Dr.A.P.J. Abdul Kalam University, Indore- Dewas Bypass Road, Indore, M.P, India.

ABSTRACT: In this paper the design of software pattern framework based on automatic test system is introduced. Basically in software engineering, pattern framework is considered as most re-useful design. Complexity is improved in the testing software applications. Hence to improve the performance in software development automatic test technology is introduced. This software pattern framework is independent of hardware system. Software pattern framework works well when it is implemented with automatic test system. This software pattern framework reduces the development cost and improves the accuracy of the system in effective way.

KEYWORDS: Software Pattern Framework, Automatic Test System, Software Engineering, Common Technology Analysis, Function models.

I. INTRODUCTION

Software testing is a fundamental and fundamental quality of programming that is used to confirm the desired results of the software program / application. A successful test should show that a program contains errors instead of working properly. Testing is inherent every time the SDLC is run. When the item is manufactured and added to the build structure, tests are performed to ensure that it works accurately and properly [1]. The deception information during the test method is reduced to quantify the unshakable quality of the software.

As the profits of the PC business gradually become fundamental and entangled in the high-tech society, the unshakable quality and quality of PC software become more and more important and fundamental. In fact, the software should become a real source of power outages that were revealed in various environments. With the growing interest in software, so does its size, code complexity, and criticality. The size of the expanded code leads to more bugs in the software. A software module is said to be error prone if it contains an immense number of errors that fundamentally impair its usefulness.

Predicting the future has always fascinated people around the world. Various advances by individuals, such as glass observers, meteorologists, stock market researchers, specialists, computer scientists and architects, seek to improve the viability of waiting for the approach of perfect situations.

Normally, prediction should be possible in two different ways. The first forecast method is guesswork and the second method is based on available information or recorded information [2]. Guess-based predictions generally do not produce good results and often lead to poor decisions that lead to bad luck. The most popular way to create expectations depends on information. For example, information in visionary terms includes information about an individual's birth, the location of the stars, and the past. After a long study of people with comparable birth information, the heavenly prophets established some guidelines and purposes that are subject to rigorous review, and rules are derived for future expectations. Therefore, forecasting is a logical endeavor.

Requirements engineering is one of the main periods of the software development life cycle. This phase begins the development life cycle with a social event that meets the customer's requirements. Clients can generally be of different types, so the types of activities also change depending on the client. In general, there are three types of activities. These are 1) small or custom companies, 2) large or market-driven tasks, and 3) very large companies. Regardless of the size of the company, ensuring the quality of the end result is a major concern. It is important then to test the item against these requirements [3]. There are two types of requirements in a milling business. You are:

Functional requirements: you determine what the manager has to do. This type of requirement manages the iconic substance of the software. They indicate the behavior or functionality of the software item.

Non-functional requirements: In software development, requirements that describe how the framework should function or perform functions under different conditions are called non-functional requirements [4]. These types of requirements are intended to complement the functional properties of the framework. They are difficult to test and are periodically checked in the abstract.

Different types of requirements, such as business requirements, market requirements, and user interface requirements, structure the subset of the two main types of software requirements. The main concern of this investigation is the properties of the malfunctioning frame. These properties are difficult to decipher and destroy [5]. These traits are exceptionally emotional and, when it comes to the real world, they continue in unusual ways, making them difficult to remove.

II. SOFTWARE DEVELOPMENT LIFE CYCLE

An SDLC has different activities together to achieve the ideal results. SDLC is a software procedure model that talks about the life model of software development. Different software process models are anticipated with different patterns like waterfall model, spiral model, incremental model, etc. Each of these models is based on related software engineering exercises to shape the overall process system. Fig.1 shows the time periods for the procedural structure exercises. These SDLC periods are invaluable for IT professionals to achieve ideal task goals. These SDLC periods are given below:

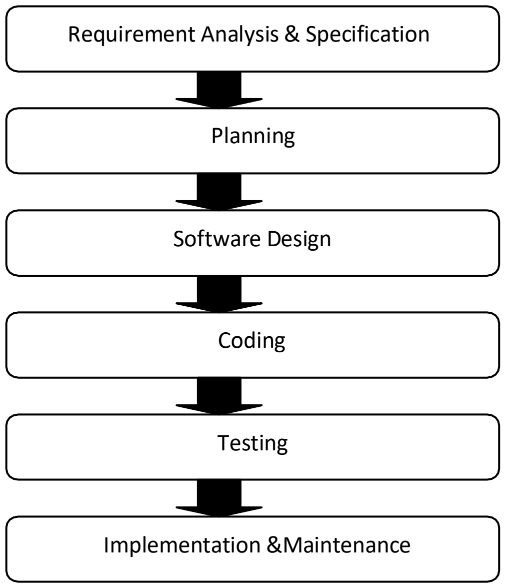
Requirement Analysis & Specification
Planning
Software Design
Coding
Testing
Implementation &Maintenance

Fig. 1: SOFTWARE DEVELOPMENT LIFE CYCLE

(a) Analysis and specification of needs

In the software development process, requirements research and specification is a notable process among the core activities. At this point, all the framework requirements can be restricted. The prerequisite survey integrates the strengths, for example, accessibility of need, unwavering quality of the item, execution, constraints, and goals that the customer expects from the framework (Pressman 2010). Requirements are compiled by customers and an SRS (Software Requirements Specification) dataset is created, leading to the transition to the next procedural model period.

(b) Planning

Planning provides a series of exercises that characterize the scope of the business, with each special run directed by the designer and the risk factor. These exercises organize the resources needed to create the job on time. Finally, planning manages and supervises the business activities that allow the association to deliver the business on the scheduled time.

(c) Systems and software design

This movement has to do with a structuring process to solve the problems identified with the software. Specifications and requirements planning are modified in a format appropriate for use in certain programming languages. Planning requires the general grouping of the designer to characterize software engineering. All the special features of the peer-to-peer setup can become too expensive at a later time and also affect the presentation of the software.

(d) Encoding

In this step, the organized structure is converted to a programming language such as C, C ++, Pascal, and Java to update the planned design. The coding step affects the testing and execution steps. Due to the cost of testing and maintenance, the software is much higher than the cost of coding. In this way, the main concern of the coding phase is to develop simple and clear projects to reduce testing and maintenance costs.

(e) System test

Testing is an important step in software development. The main purpose of testing is to find bugs in the software. The software bug can be displayed during the prerequisite, structure, and coding phase. There can be many reasons for software failure. The primary testing period is unit testing, which is performed by designers at the time of development. After the unit tests are performed, coordination tests are performed that focus on testing the interconnected modules. After the frame is assembled, frame tests are performed to verify that all requirements are met. Finally, commit tests are performed to show the client that the framework is active. This is the key strategy for dynamically reviewing and approving a framework. There are various test devices and strategies available for testing.

(f) Implementation and maintenance

When the SDLC is exhausted, the software is passed for operational use and shipped to the customer. Some problems identified with the framework created can be discovered immediately after it begins to be used in practice and are understood from then on. Some issues continue to occur inconsistently and will be addressed as needed. Now this procedure applies to maintenance.

III. TYPES OF SOFTWARE TESTS

There are a number of testing strategies and procedures to meet different requirements at different stages of the life cycle. Software testing can be divided into complementary classes: black box testing, white box testing, gray box program testing, quality assurance (QA) testing, functional testing, performance testing, performance testing scalability, manual tests, automatic tests. The description of each type of test is given below:

(a) Black box test

The screening test is not identified with the company's internal code region. Take an outside perspective on the test questions. These tests may or may not be helpful, but they are generally helpful. The tester selects essential and invalid information and selects the error in the correct place (Bell et al. 2006). There is no information about the internal structure of the exam question. As a discovery tester, you do not need to understand program code.

(b) White box test

White box testing uses an internal perspective of the framework to design test cases. It requires programming skills to perceive all routes through the element. The evaluator selects the test jobs to practice routes through the code and selects the appropriate performance (Bell et al. 2006).

(c) Gray box test

This type of programming test is a combination of black box tests and white box tests. An attempt is made to change the quality of each type and combine them into a "complete" test more unmistakable than the absolute of its parts. Dim Box can use the simple and easy-to-use method of screening versus end-to-end testing, the code of which is focused on white box testing (Bell et al. 2006).

(d) Test run

Functional tests are carried out to verify the best possible handling of the element. It can be combined with tests of counting and numerical precision for intelligent and monetary programming and practical tests for graphical user interfaces (Bell et al. 2006).

(e) Performance test

Performance tests select item performance through the speed of estimates or response to the customer (Bell et al. 2006).

(f) Manual test

With manual testing, testers physically execute the code piece by piece. At the end of the day, this type of test takes the tests gradually. Human impedance is an important part of these tests. Using manual tests to test the code, the tester / designer presses the socket, selects the connection, confirms the benefits, and physically performs the entire test task. In this way, the term “manual” is legitimized.

(g) Automated tests

Mechanized tests try to run the code in auto run mode. The tests are set up with the fully automated route of a QA tester / QA technician using the equipment so that a collection of tests can be limited by pressing an isolated output or entering a single command without limits.

IV. SOFTWARE PATTERN FRAMEWORK BASED ON AUTOMATIC TEST SYSTEM

The below Fig. 2 shows the flow chart of software pattern framework based on automatic test system. In this initially, input data is analyzed using common technology. Implementation is done based on the function models. Framework is designed and built. Now testing procedure is implemented for the designed framework. After testing exact outcome is obtained then data is moved to the block of specific application. If the exact outcome is not obtained then feedback is given to framework block so that it will check and give update. After that process will be stopped.

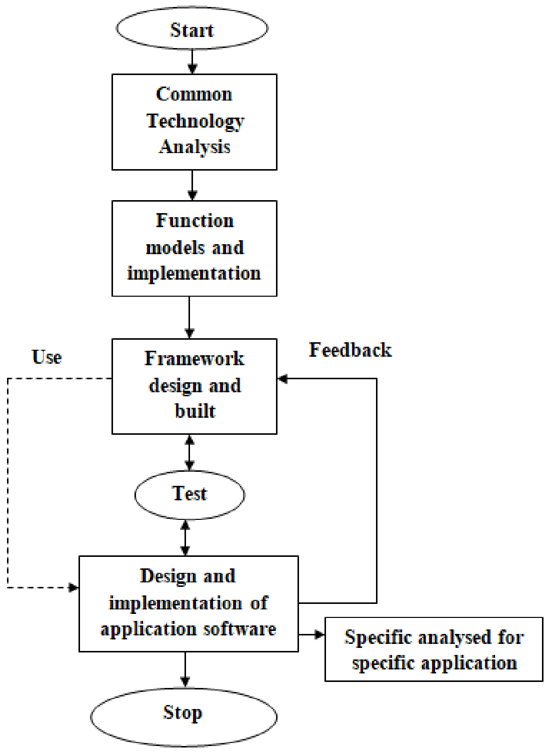
Start
Common Technology Analysis
Function models and implementation
Use
Framework design and built
Feedback
Test
Design and implementation of application software
Specific analysed for specific application
Stop

Fig. 2: FLOW CHART OF SOFTWARE PATTERN FRAMEWORK BASED ON AUTOMATIC TEST SYSTEM

ALGORITHM:

Step-1: In this initially, input data is analyzed using common technology.

Step-2: Implementation is done based on the function models.

Step-3: Framework is designed and built.

Step-4: Now testing procedure is implemented for the designed framework.

Step-5: After testing exact outcome is obtained then data is moved to the block of specific application.

Step-6: If the exact outcome is not obtained then feedback is given to framework block so that it will check and give update.

Step-7: After that process will be stopped.

Software testing provides a technique for reducing maintenance costs, limiting errors, and overhead programming costs. Due to the life cycle of the program change, it turned out to be the most extraordinary parameter. With the help of IT test tools, developers and analyzers can easily automate the entire test move by changing programming. Change its appearance and source code. Very impressive programming testing through efficient testing

In programming automation, tests are similar to an item progression method. Experience a life cycle similar to that of the programming difference. The remarkable feeling that it have to deal with who invents it. There is constant research on who creates the content, be it a fashion designer or a trial partner. It is certainly a wise task, and several affiliates hope that the effort will be a joint effort between the analyzer and the architect. The automation system is associated with a considerable amount of effort, which requires joint work, considering how much additional time and budget requirements are spent

Table 1: PARAMETERS OF SOFTWARE PATTERN FRAMEWORK BASED ON AUTOMATIC TEST SYSTEM

S.No	Parameter	Software Pattern Framework Based On Automatic Test System
1	Cost	Low
2	Accuracy	High
3	Time	Less
4	Scalability	High

The below Fig.3 shows the Accuracy and Scalability of software pattern framework based on automatic test system. Accuracy is improved and less time is taken while testing the software pattern framework based on automatic test system. Therefore scalability of the system is also improved.

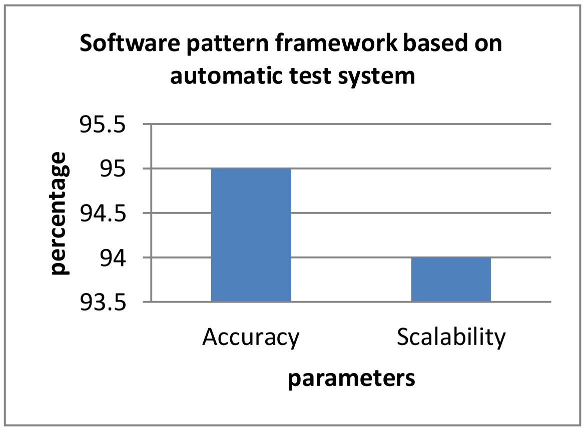

Fig. 3: ACCURACY AND SCALABILITY

Table 2: COST ANALYSIS

Method	Cost with respect to average value
Interactive Design Pattern Recommendation	64%

Software pattern framework based on Automatic Test System	15%

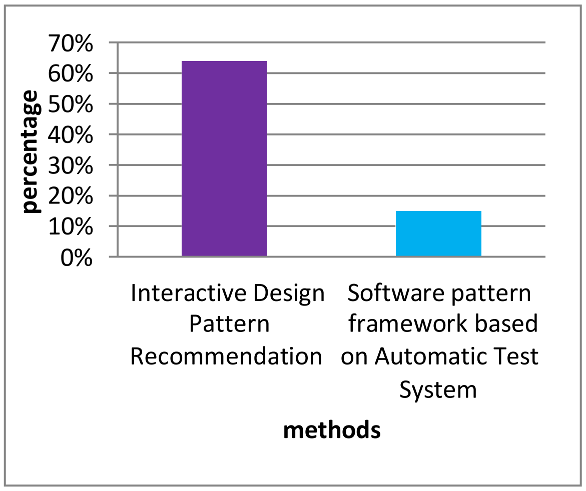

Fig. 4: COST ANALYSIS

The above Fig. 4 shows the cost analysis of two methods as 'Interactive Design Pattern Recommendation' and 'software pattern framework based on automatic test system'. Cost is reduced in software pattern framework based on automatic test system.
The effectiveness is the impact of the solution that provided by design pattern. Most of the developers using design patterns to solve problems. This shows how the patterns effectively in software developments. This methodology will formalize depend on the questionnaire that will spread to the developers and students in the college of computer science, and gather the answers to validates our goals. The questions are answered using the likert scale that is ranging from 1 to 5.

- Very low effect indicating 1
- Low effect indicating 2
- Nominal/Average effect indicating 3
- High effect indicating 4
- Very high effect indicating 5

Table 3 showed that 47.2% of the responses agreed with effectiveness of design pattern in which 6.4% of the software engineers strongly agreed and 40.8% of the professionals agreed with it. 40.8% of the responses remained neutral for effectiveness of design pattern. 11.8% of the responses were not in favor of the effectiveness of design pattern.

Table 3: ANALYSIS OF EFFECTIVENESS OF PATTERN

Q.No.	**Low**	**Nominal**	**High**	**Very high**
Q1	3	11	17	0
Q2	1	20	7	3

Q3	7	7	14	3
Total	11	38	38	6
Avg.	11.8%	40.8%	40.8%	6.4%

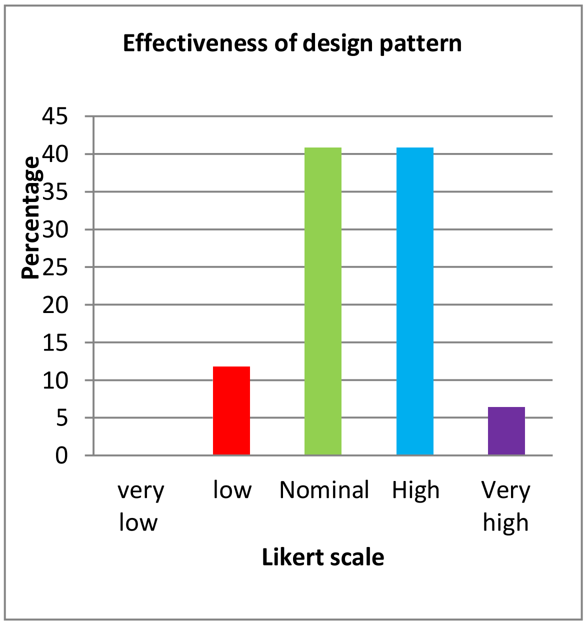

Fig. 5: EFFECTIVENESS OF PATTERN

V.CONCLUSION

Hence in this paper the design of software pattern framework based on automatic test system was introduced. Efficiency and quality of system is improved based on the applications of development. The pattern which is developed in this paper is convenient at ATS software framework. The
Software framework is used in the every stage from design. The result of this paper is validated by questionnaire which estimates the efficiency of impact of the solution that provided by design pattern. States that 47.2% of the responses agreed with effectiveness of design pattern in which 6.4% of the software engineers strongly agreed and 40.8% of the professionals agreed with it. Therefore from results it can observe that it reduces the cost & time, increases the accuracy and scalability with great efficiency.

VI. REFERENCES

[1] Guo RongBin, Zhao XiuCai. The trend of automatic test system [J]. Foreign Electronic Measurement Technology, 2014, 33(6)(In Chinese)
[2] YANG Aibing, SUN Ye, PENG Wei. Research on the software development model of automatic test system [J]. Industrial instrumentation & automation, 2010(6), 16-18
[3] Liu Q. Design of universal ATS software framework based on signal[J]. Electronic Measurement Technology, 2012
[4] Wang Y P, Wang D G, Zhong-De W U. Signal-Oriented ATS Test Software Structure and the Model[J]. Journal of Naval Aeronautical & Astronautical University, 2013.

[5] Liu Qi, He Yuzhu. Design of universal ATS software framework based on signal[J]. Electronic Measurement Technology, 2012, (12):46-49.
[6] Sun Xiaojin, Guo Enquan. Design of software architecture for fault diagnosis system based on IEEE1232[J].Journal of Electronic Measurement and Instrumentation, 2014, 28(1):36-42.
[7] Li Chuan-Huang, WANG Wei-Ming, SHI Yin-Yan. Performance Prediction Method for UML Software Architecture and Its Automation[J]. Journal of Software, 2013, (7):1512-1528.
[8] LIU Feng, SUN Yong. Application of Design Patterns and Component Technology to Business Logic Layer[J]. Computer Systems & Applications, 2011, 20(10):154-159.
[9] ZHANG Yuan, ZHANG Zhao, LIU Rui. Modular design of virtual experiment system based on MVC design model[J]. Computer Engineering & Science, 2013, 35(8):125-129.
[10] Guo Shungzhou, Liang Jinlan. Design and Implement action of Automated Testing Framework[C].Computer Measurement & Control. 2009:224-227.
[11] LIU Jin Ning, MENG Chen, HOU Yan. Research of ATS software model[J]. Measurement Control Technology and Instruments, 2007, 33(9): 81-84.
[12] WANG Ying-qiang, CHEN Sui-yang, WANG Zheng-feng. Design and Implementation of Reusable Software Development Framework Based on . Net[J]. Computer Technology and Development, 2014, (6):122-126.