ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

SOFTWARE QUALITY ASSURANCE USING TIME STAMP TEMPORAL QUALITY ASSURANCE ALGORITHM

M.Gowthami[1], P. Thenmozhi[2]

M.phil (Full Time) Research Scholars, Dept., of Computer Science, Kongu Arts and Science College, Erode, India[1]

Assistant Professor, Department Of Computer Science, Kongu Arts and Science College, Erode, India[2]

Abstract : Time stamp Temporal Quality Assurance algorithm is also a planning system for manufacturing processes that helps in analysis the productivity technique. It was developed with four kinds of clients. The clients are Manufacturer, Agent, Suppliers and Customer. Using this system the manufacturer will know about the performance of the agents and similarly the agent will know about the performance of the suppliers. Using this system the supplier will be able to know about the customer performance towards the products. The Software Quality Assurance technique implemented in this system is "Time stamp Temporal Quality Assurance". Using the above technique the system can mine the useful information from the large amount of data stored in the database. The proposed method QA is used to eliminate the waste provable quality assurance. It increases the accuracy of 68% and it improves average recall of 76%. Effort needed to find defect inducing changes only the user, who has rights to access, can get the required information. It provides employees involvement in decision making supplier participation to improve total quality control using TSTQA software quality assurance algorithm.

Keywords: SQA, Change management, dynamic reconfiguration, QoS assurance process, System evolution.

I.INTRODUCTION

Software system is playing an increasingly important role in our society. Software systems are growing up in both size and complexity, developing high quality and reliable systems is becoming more challenging and more expensive. Building software products on time, within budget and with highest quality is the demand in the area of software quality assurance [1]. Software quality assurance encompasses the entire software development process, which includes processes such as requirements definition, software design, coding, source code control, code reviews, change management, configuration management, testing, release management, and product integration. Software quality assurance is organized into goals, commitments, abilities, activities, measurements, and verifications.

There are many techniques used to improve testing efficiency in software quality assurance. Quality assurance techniques are divided into 2 types: static and dynamic testing techniques. Static testing techniques are based on manual examination (review) and automatic analysis (static analysis) of the code or other project documentation without the execution of the code. to obtain the validation it must also satisfy the requirements identified in the analysis. The Test Factory verifies the correct functionality of the software product implemented and responsiveness to functional and technical specifications. [7,8]. However, most of existing approaches is often performed static testing techniques separately from dynamic testing techniques that often lead to the waste of effort in software quality assurance. Combining static and dynamic testing techniques is one of the approaches that can help tackle this problem [8].

The traditional approach to Quality assurance is at the root of many problems that business faces today. The high inventory levels, soaring costs, adverse relationships with suppliers and quality issues which either stop production or results in poor products are just a few of the problems with the current way of Quality assurance.

During the last two decades, the Quality assurance environment has become one of the most crucial elements in establishing the value added contents for the products and services and hence has become the vital factor in the dynamic international market. Shortages of raw material, shorter lead time, high quality, increasing the variety of products with smaller runs, inflation, productivity and introduction of a JIT Quality assurance system etc. prompted the realization of the importance of Quality assurance Software Quality Assurance helps ensure the development of high-quality software. Software Quality Assurance practices are implemented in most types of software development, regardless of the underlying software development model being used. Software Quality Assurance incorporates and implements software testing methodologies to test software. Rather than checking for quality after completion, Software Quality Assurance processes tests for quality in each phase of development until the software is complete. With Software Quality Assurance, the software development process moves into the next phase only once the current/previous phase complies with the required quality standards. Software Quality Assurance generally works on

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

one or more industry standards that help in building software quality guidelines and implementation strategies. Quality Assurance has gained wide acceptance as an integral part of any, good Software Development Life Cycle. An efficient QA organization must be designed so that it facilitates the activities that are necessary to measure the characteristics of software quality that are pertinent to the success of the business. Such activities may include code inspections, requirements gathering and system monitoring. The inability to measure quality can cause a business to lose valuable customers and struggle to compete in the market place. During the ever changing market landscape, firms must react to external forces to ensure that their activity configurations are internally consistent and appropriate for the current environment of the firm. This challenge is more acute when the competitive landscape changes, such that a new set of high-performing activities are required.

It has been noted that adaptive firms must maintain a balance of exploitation and exploration to stay ahead of the curve in the market place. To accomplish this, some firms may rely on centralized organizations while others rely on decentralized. Using the internet as an event that changed the competitive landscape for many companies we can illustrate the new activity choices that became available to companies.

Similar challenges will be faced by the QA organization. Again, using the same event as above, we can illustrate the new activity choices that would become available to the QA organization. One option could be to form a QA organization that is dedicated to any business done via the internet. This option may require developing new processes, acquiring new skill sets, relocating and retraining existing personnel or creating a new QA methodology that supports the more sophisticated development processes that are often associated with newer technologies. Furthermore, it may be more conducive to measuring specific quality metrics that are more relevant to that particular business such as efficiency and reliability, which are both included in the quality model.

Another option that may become available to the QA organization can be seen when a firm merges with another firm due to an acquisition. This is a very common occurrence as many firms rely on acquisitions to grow their business and in most cases will require the integration of multiple systems. This may be best accomplished by maintaining a centralized QA organization that applies a rigid and consistent set of testing standards in order to properly measure these characteristics. A centralized model is also conducive to knowledge transfer from the acquired company and allows for the creation of a consistent set of documents that will aid the learning process. This enables the measurement of yet another quality characteristic, that of learn ability.

Now consider a decentralized group in this same scenario. Multiple QA teams are now charged with testing various integration points of numerous applications. This model poses many measurement challenges for the acquired firm including:

- Review of documentation and the approval process may vary from team to team. This may make it challenging to measure software learn ability.
- Testing artifacts are created with a wide variety of formats and templates, which may make it difficult to measure software functionality.
- Multiple teams must be briefed on conformance and compliance details, which leave measurement open to interpretation.
- Multiple dependencies are created among test teams and development teams, which may cause measurements to be incomplete and fragmented.
- Testing can not start until all prerequisites are in place, which may extend the time that it takes to gather key quality metrics.

The Contemporary Quality Assurance professional can refer to this study as their starting point to understand why certain applications in their portfolio may be suffering from certain software deficiencies. The understanding gained from this research can also be used by Quality Assurance professionals and corporate managers alike to suggest areas where subtle changes to existing processes may make measuring the software quality characteristics that are critical to their business easier without disrupting normal operating procedures. More specifically, the findings from this study can be analyzed to determine if decentralized and centralized Quality Assurance organizations elicit certain human behaviors that are essential for measuring a certain characteristic of software quality.

The quality of products is dependent upon that of the participating constituents. Some of which are sustainable and effectively controlled while others are not. The processes which are managed with Quality Assurance pertain to Software Quality Management. If the specification does not reflect the true quality requirements, the product's quality cannot be guaranteed.

In the previous methodology, the user can able to store only small amount of data and data retrieval operation is not much efficient. The previous methodology is not reliable for complex storage of data. The user cannot get the required information from the data using the existing system. The operations on data are not possible by using the previous methodology.

Also, the previous methodology serves as only storage medium. It cannot help the user to make knowledge driven decisions. It does not serve as prediction tool for the user. The information hidden in the data are not extracted by using the previous methodology. However the retrieval of data from the database is not much easier. The existing system does not answer the business questions. Extracting information from the database is time consuming and cumbersome job in the existing system.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

The problem statement addressed was: The role, if any, that organizational structure plays on measuring software quality. Measuring software quality often involves activities that require collaborating with multiple groups within the organization. In some cases, the groups may be internal to the enterprise, while others may be external, such as a customer.

Today's companies must design QA organizations that strengthen their presence in the marketplace. This can only be accomplished by ensuring that each department, including the QA organization has access to the information and is able to perform the activities that are required to measure the characteristics of software quality that are most relevant to their business. The QA organization is not immune to this problem and is the focus of this research.

1.1. Objective

The system was developed for four kinds of clients. The clients are Manufacturer, Agent, Suppliers, and Customer. This system is a client-server project in which the user can request the system for some operation and system will response for those request by the user. The user can enter into the system by entering the name, password, and client level and user code. All the data retrieval operations are done by using the user code. This system provides access rights to the user. So only the authorized clients will able to use this system. This system provides the data base security by allowing only the authorized clients to view the data stored in the database. This system serves as the prediction tool for the user.

Using this system the manufacturer will know about the performance of his agents and similarly the agent will know about the performance of his suppliers. Using this system the supplier will be able to know about the customer preference towards the products. The users will able view the details belong to his lowest level user only. Thus system provides the client security. This system also maintains the orders and transactions of the user. Software Quality Assurance is done by using the Time stamp Temporal Quality Assurance. Using Time stamp Temporal Quality Assurance method, the user can predict the behavior of purchase and sales of the product.

The sales and purchase details of the product are taken from the appropriate database of the user. Both classification and prediction are made according to certain assumptions. For decision trees, the user will give the certain threshold and the calculations are done based on the threshold and the result will be shown to the user.

There are a number of objectives for this paper, the first is to define the principles of software testing, describe the numerous testing methodologies and how to effectively conduct this testing on projects in industry. The second objective is to evaluate what constitutes software quality and what factors affect this quality and how, when and where QA can be used in the project life-cycle for improving product quality. The third objective is to outline the test and QA effort during a project in a particular company and to evaluate the adoption of improved practices during subsequent projects in the same company. The fourth objective is to develop the improved practices into a framework for evaluation in other companies.

1.2. Problem Affiliations

Unlikely conventional development methodologies software development integrates QA practices in development activities, rather than practicing them independently and separately. Sometimes due to managerial or/and organizational issues, customer or project requires some standards to be followed [21].

According to this report one of the important reasons which makes methodologies incompatible is that the organizations adopted, were large scale with high level of management, organizational hierarchy and governance; where the organizations which adopted TSTQA methodologies were focusing on small or single team projects. This case study places a claim "Currently, the existing just in time project learning techniques seem to lack means to perceive the organizational. For example, they do not address the important aspects of systematically defining, validating, packaging and storing the results of JIT projects."[23]. It can be observed that all the aspects, this claim addressed, are SQA responsibilities. Any kind of improvements within the organization, project or development are to achieve higher level of software products.

1.3. Proposed Solution

TSTQ has the following flavors of influence on our study:

- Claims of higher quality through TSTQA development (guidelines, survey results, favors and critics)
- Integrated QA activities in TSTQA methodologies (guidelines, favors and critics)
- Claims of lacking in defined standards in TSTQA development (critics, surveys)
- Research done to enhance throughput over TSTQA development projects (research results and suggestions)

QA must be integrated rather than shifted, here shifting implies as most of the QA activities in agile projects are performed by the developer or QA personnel are supposed to perform as developer. There is no doubt that TSTQ methodologies came up with higher quality due their incremental and test-driven nature. Production of higher quality, and absence of systematic, organized and well defined procedures and standards, show that there is much space of improvement in agile SQA activities. Baseline of our suggested solution is "to redefine SQA activities rather than shifting".

II RELATED WORK

2.1. User View

Whereas the transcendental view is ethereal, the user view is more concrete, grounded in product characteristics that meet the user's needs. This view of quality evaluates the product in a task context and can thus be a highly personalized view. In reliability and performance modeling, the user view is inherent, since the methods assess product behavior with respect to operational

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

profiles (that is, to expected functionality and usage patterns). Product usability is also related to the user view: in usability laboratories, researchers observe how users interact with software products. [13]

2.2. Manufacturing View

The Manufacturing view focuses on product quality during production and after delivery. This view examines whether or not the product was constructed "right the first time," in an effort to avoid the costs associated with rework during development and after delivery. This process focus can lead to quality assessment that is virtually independent of the product itself. The manufacturing approach was adopted by ISO 90013 and the Capability Maturity Model advocates conformance to process rather than to specification. There is little evidence that conformance to process standards guarantees good products. In fact, critics of this view suggest that process standards guarantee only uniformity of output and can thus institutionalize the production of mediocre or bad products. However, this criticism may be unfair.

2.3. Product View

Whereas the user and manufacturing views examine the product from without, a product view of quality looks inside, considering the product's inherent characteristics. This approach is frequently adopted by software-metrics advocates, who assume that measuring and controlling internal product properties (internal quality indicators) will result in improved external product behavior (quality in use). Assessing quality by measuring internal properties is attractive because it offers an objective and context independent view of quality. However, more research is needed to confirm that internal quality assures external quality and to determine which aspects of internal quality affect the product's use.

2.4. Value-Based View

Different views can be held by different groups involved in software development. Customers or marketing groups typically have a user view, researchers a product view, and the production department a manufacturing view. If the difference in viewpoints is not made explicit, misunderstandings about quality created during project initiation are likely to resurface as (potentially) major problems during product acceptance.

If the user's view is stated explicitly during requirements specification, the technical specification that drives the production process can be derived directly from it as can product functionality and features. However, problems can arise when changes to the requirements occur. At this point, the user's requirement for a useful product may be in conflict with the manufacturer's goal of minimizing rework.

The value-based view of quality becomes important. Equating quality to what the customer is willing to pay for encourages everyone to consider the trade-offs between cost and quality. A value-based perception can involve techniques to manage conflicts when requirements change. Among them are "design to cost," in which design possibilities are constrained by available resources and "requirements scrubbing," in which requirements are assessed and revised in light of costs and benefits. Product purchasers take a rather different value-based view. The internal software measures are irrelevant. Purchasers compare the product cost with the potential benefits.

2.5. Jit

The Just-In-Time (JIT) is a planning system for manufacturing processes that helps in achieving high-volume production using the minimal inventories. The system eliminates the inventory of raw materials, work in progress, and finished goods by making them available as and when required. The items are picked up by the worker and fed directly into the production process. The finished goods are produced only at the time they are required for sale. The implementation of the JIT system requires complete transformation of methods of designing products and services, assigning responsibilities to workers, and organizing work.
JIT system, the finished goods are assembled just before they are sold, the sub assemblies are made just before the products are assembled and the components are fabricated just before the sub-assemblies are made. The work-in-progress inventory is always kept at a low level, thus reducing the production lead times. The firms should achieve and maintain high performance levels in all their operational areas to facilitate the smooth flow of materials in the JIT Systems.

The JIT manufacturing system is based on the concept of continuous improvement, which includes the two mutually supporting components of people involvement and total quality control.
People Involvement is a Human Resources Management component plays a vital role in the implementation of the JIT manufacturing system. The successful implementation of a JIT program requires teamwork, discipline, and supplier involvement.

Total Quality Control is a firm can produce high quality products only through the combined efforts of all the departments including the purchase department, quality control department, and personnel department. The concept of 'immediate customer' helps the firms to achieve the required quality levels.

Team Work involves activities like suggestion programs, and quality circle programs which enable employees to actively participate. Suggestion programs are conducted to encourage the employees to their ideas on how to improve a process. In quality circles, people working in similar types of operations meet at regular intervals and discuss ways to improving the quality of their processes.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

In the supplier involvement firms can allow suppliers to participate in design review and to suggest new designs and methods for improving product quality or productivity. JIT firms enter into contracts with their suppliers instead of inviting competitive bids from a set of suppliers. The JIT firm can share its production plans and schedule with its suppliers so that they can plan their business and capacity requirements. Linear production schedules relate to the development of production schedules with uniform workloads. The maintenance of linear production schedules requires the identification and elimination of production bottlenecks, a balance in the production system, and a reduction in set-up time.
A JIT uses the concept of 'immediate customer' where each worker in the firm considers the next worker who continues the production process as the customer. Therefore, it is the responsibility of the worker to ensure that the product is processed to meet specifications and quality requirements before passing it on to the next worker. Only items of acceptable quality are delivered to the immediate customer. In case a worker delivers a defective item or an improperly finished item to his/her immediate customer, the worker who identifies the defect is authorized to stop the process and take necessary actions

In a JIT Manufacturing system, firms maintain inventory in the smallest possible lot sizes. This is done to reduce the cycle inventory, cut lead times, and achieve a uniform work load. JIT firms should maintain long-term relationships with their suppliers as they are responsible for providing the timely delivery of good quality inventory.

In the JIT system, quality control begins from the source where the workers are encouraged to maintain the quality of work. The production process is stopped immediately when a quality problem is identified and is continued only after the problem has been sorted out.
JIT system is possible only for minimal stock in the small organization and is kept for re-working faulty product. Production is very reliant on suppliers and if stock is not delivered on time, the whole production schedule can be delayed. There is no spare finished product available to meet unexpected orders, because all products are made to meet actual orders. However, JIT is a very responsive method of production.

2.6. Timestamp

A timestamp is a sequence of characters or encoded information identifying when a certain event occurred, usually giving date and time of day, sometimes accurate to a small fraction of a second.

III. THE QA ORGANIZATIONAL MODELS

The key distinction between a centralized and decentralized QA organization, as with most other organizations is the general degree to which delegation exists [17]. In a centralized QA organization, decisions are made by a single governing body and delegation is minimized such that subunits have no autonomy or decision making authority [17]. Consider a centralized QA organization that may have separate groups dedicated to the various LOBs; all decisions are still made by the same governing body. However, in a decentralized QA organization, the degree of delegation is greater so that each LOB has the ability to establish its own processes, standards, and procedures.

Fig1: Centralized QA Organization

In a centralized QA model, a single QA group services all JITs in the enterprise. The centralized model is sometimes referred to as a QA center of excellence and has to be prepared to address the demand for services across the enterprise so that QA resources can be provisioned and allocated to JIT projects as they are initiated. In a decentralized model, each JIT is serviced by its own, dedicated QA group. Each QA group may or may not adhere to the same QA practices and may be governed by different processes.

IV. SYSTEM METHODOLOGY

4.1 Structure of Time Stamp Temporal Quality Assurance

The fundamentals of TSQA deal with a planned activity to evaluate the development process during its progress. The plan or architecture must be placed around the entry to and the output from each stage of the development effort.

If location and cause of the software defects or errors are taken into consideration during the software development, then there is a starting point for assuring the quality of each stage. These defects can also be considered in relation to the factors that affect the software quality. The classification of the causes of the defects can be addressed by TSQA.

TSQA is a continuously evolving entity with an emphasis on improving. The architecture of TSTQ is planning the project initiation by creating the QA plan. In the Management of the Project life-cycle they create defect removal and defect injection prevention activities. Investigate Software Quality improvement.

4.1.1 Planning From the Project Initiation and Project Planning Stage

Projects that are carried out, in house are more susceptible to failure than projects which go under the more formal external contract route. For this reason the schedule and budget failures are accompanied by lower than acceptable software quality, this is largely due to a more casual attitude to meet deadlines. Contract review can alleviate this by ensuring that the correct measures are put in place for the project. Following the contract review,

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

the project plans and schedule should be documented. Any risks that are envisaged at this stage should also be documented with a probability of occurrence and a mitigation plan identified should the risk occur.

4.1.2 Management of the Project Life-Cycle Activities and Components

Software Quality Assurance Defect Removal

Considering that there are several factors that affect software quality there are a number of activities that can be followed to improve the development stages in terms of software quality. The activities are discussed below.

- Reviews
- Inspections
- Walk through
- Testing
- Configuration management

An inspection and walkthrough is an improvement over the desk-checking process (the process of a programmer reading the own program before testing it). Inspections and walkthroughs are more effective, again because people other than the programs author are involved in the process. These methods generally are effective in finding from 30 to 70% of the logic-design and coding errors in typical programs.

Procedural order and teamwork lie at the heart of formal design reviews, inspections or walk-through. Each participant is expected to emphasize his or her area of expertise. The knowledge that the work item will be reviewed stimulates the team to work to their upper end of productivity.

For different stages of the development process, there are different defects that get injected into the software. The rate of defect injection differs for each stage of development. The QA activities must match the defect injection rate and type to be effective at their removal.

4.2 Complications with Software and Its Quality Assurance

The software quality assurance is consists of people's expectation and experiences of the system. People have own their options on how a product should work, how fast it responds their commands and so on. Quality Assurance has its roots in assuring the quality of a manufactured physical product; this is achieved by inspecting the product and evaluating its quality near its completion or at various stages of production. Software however is not as tangible as products that are more physical. The "invisible" nature of software adds to the complications of assessing its quality. "Industrial products are visible, software products are invisible. Most of the defects in an industrial product can be detected during the manufacturing process, however defects in software products are invisible, as in the fact that parts of a software package may be absent from the beginning".

There are further complications with assessing software quality; this is attributed to its inherent complexity. Software systems have grown from standalone systems on a single server to globally networked servers spanning multiple countries and multiple servers. Software may be developed by a team of people who carry out specific roles; the roles are played out during different stages of development. The teamwork driven development life-cycle is open to a multitude of problems, particularly because of the inter-dependence of people in the life-cycle.

Poor relationships between individual team members affect the productivity and creativity of the team. The experience of the team can also have implications where experienced members are supporting inexperienced members. If a project team member departs during the middle of the life-cycle, the consequences of this departure can impact on the success of the project.

The software development team is affected by external factors such as the customer's documented requirements and how detailed and accurate they represent the actual requirements. The schedule and budget allocated to the project will also have an effect on the quality of the software. After a project has been completed and installed in its target environment, the system must then be maintained for the duration of its lifespan, the ease with which these changes are conducted successfully can affect the quality of the system.

4.3 Time stamp Temporal Quality Assurance (TSTQA)

The basic hypothesis required in order to make prediction using this algorithm is that that clients of same type will show the same behavior. The basic philosophy of this algorithm is 'do as your neighbor do'. In order to predict the behavior of the product, start to look at the behavior of clients using those products and from that make further prediction for that product. The data are taken from appropriate databases. The user can view the result graphically. This algorithm is a purest searching technique because the data set itself is used for the reference.

ALGORTHIM: TIMESTAMP TEMPORAL QUALITY ASSURANCE

Input: T,s

Output: σ

0.$\sigma = \in$

1.$node_{cur}$ = root of T

2.$node_{next}$ = NULL

3. repeat

4. remove tail symbol σ' from s

5. $node_{next}$ = child of $node_{cur}$ with respect to σ'

6. if $node_{next}$ exists then

7. $node_{cur} = .node_{next}$

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

8. else

9. break

10. until (s==∈)

11. $\sigma =$ the symbol with the highest conditional probability in $node_{cur}$

12. return σ

Step 1: collect the estimates

Assume that a neighbor node is available at time T

Item i

Similarity between same rating patterns S

Step 2: Takes group estimation

Neighbor = similar users

Generate a prediction for an item i by analyzing ratings for i from users in S neighborhood

$$\text{pred (s,i)} = \bar{r} + \frac{\sum_{n\ c\ neighbours\ (s)} \text{sim}(s,n).\ (r_{ni} - r_n)}{\sum_{n\ c\ neighbours\ (s)} \text{sim}(s,n)}$$

Step 3: Item-Based Nearest Neighbor

Generate predictions based on similarities between items.

Prediction for a user s and item i us composed of a weighted sum of the user s ratings for items most similar to i.

$$\text{pred (s,i)} = \bar{r} + \frac{\sum_{j \in ratedItems\ (s)} \text{sim}(i,j).\ r_{si}}{\sum_{j \in ratedItems\ (s)} \text{sim}(i,j)}$$

Step 4: Reduce domain complexity by mapping the item space to a smaller number of underlying dimensions.

V. SYSTEM IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving to the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods. Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

Every developed system must be implemented to fulfill the mode of development. There are many software implementation methods. In this system, direct change over from existing system to computer system is being carried out. It is the process of converting a new or revised system designed into a working system. It is the essential stage in achieving a successful new system, because usually it involves a lot of upheaval in the user. It must therefore be carefully planned and controlled to avoid problems in the implementation process.

5.1 Modules

- Client Module
- Data Input Module
- Client View Module
- Client Authentication Module
- Product Transaction Module
- Database Module
- Time Stamp Temporal Quality Assurance Algorithm Module
- Graphical Output Module

Most Organizations have large databases that contain a wealth of potentially accessible information. However, it is usually very difficult to access this information. The unbridled growth of data will inevitably lead to a situation in which it is increasingly difficult to access the desired information. Hence there is a need to build a system for efficient storage and retrieval of data that are stored in the databases.

However, in order to assess the knowledge hidden in the database, the data Software Quality Assurance concepts and techniques should be implemented in the system. Using Software Quality Assurance, it is possible to find the optimal segmentation in the large database. The user will predict various aspects of the product by using the data Software Quality Assurance techniques. The main purpose of this system is to store large amount of data in the database and to retrieve the data in an efficient way and to serve as a prediction tool for the user.

The "Data Software Quality Assurance Techniques "is a client server project serves as a prediction tools for various activities of the supply chain

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

management in an industry. Timestamp temporal Quality Assurance methodology is used. The results of these algorithms are shown to the user graphically.

- Every company process store large quantities of highly detailed information about customers, markets, products and manufacturing. Using data Software Quality Assurance organizations are able to resolve a diverse range of business problems and create new opportunities for themselves.
- Finding the correlation or patterns among dozens of fields in large database and interpreting the mined results as a graphical outputs requires parallel database management and advanced data Software Quality Assurance tools
- Graphical representation provides on easier way of understanding the mined results than any other way of representation.
- There is a situation that more than one client can order for the product at the same time. Hence the product stock levels are updated immediately after the delivery of the product. So the user will able to know the quantity of the product.
- Database security and client authentication are used. Only the authorized client can access the database. Clients are at different levels like manufacturer, agent, supplier and customer. Restriction of client at one level to access database through other client levels maintains the client level security and enhances database security

5.1.1 Client Module

A client module is a network module that supports and implements the client side of a Network Programming Interface (NPI).The main objective of this module is to develop an interface that provides all types of request to the server. The client module is to give client request to the server.

5.1.2 Data Input Module

This module is used to develop interfaces for providing input data for various processes. Input data are entered in appropriate screens. They are checked for validity and stored in appropriate database.

5.1.3 Client View Module

The purpose of this module is to develop the client interfaces to view the purchase potential of clients. This module contains the details about client transaction, payment, ordering.

5.1.4 Client Authentication Module

Client Authentication Module reduces costs and frees up server cycles by eliminating the need to manage and enforce authentication individually across applications. The main objective of this module is to restrict the unauthorized client from accessing the database.

5.1.5 Product Transaction Module

This module performs the operations of product order checking and the cash transaction of the client. This module also produces reports regarding the payments of the various clients under the user. This module is also responsible to indicate the user about the stocks available of a particular product. This module has done the calculation regarding the payments. If the transaction is credit, then this module is responsible to calculate the balance amount at all times. When the payment is made, then this module updates the current balance of the particular user.

5.1.6 Database Module

This module deals with all types of transactions within database and maintains large amount of data stored in database. The transactions like store, update, delete etc.

5.1.7 Time Stamp Temporal Quality Assurance Algorithm Module

Software Quality Assurance of information from the database is done in this module. The algorithms used in this system are TSTQA Rules. The user can able to analysis the data in years, months, and days. After entering the user needs, the system will display the result for the algorithm in the graphical format. Using the query tool the user can get the statistics of the particular product.

5.1.8 Graphical Output Module

This module deals with retrieving the required data from the database and displays the result in the graphical format. The values for the graph are calculated at the run time depending upon the needs of the user. The values stored for the graph are deleted, when the next user request is made to the system and the graph is displayed with new computed values.

VI. FINDINGS AND RESULT

6.1 SQA Findings

The data collected from the surveys on the quality sub-characteristics suggested that type of QA organization, centralized or decentralized. The only exception was the suitability sub-characteristic, which is part of the functionality characteristic. The results suggested that a decentralized QA organization has better to measure software suitability. Therefore, there was only one instance where there was sufficient evidence not to accept the null hypothesis. The implication to quality conscious enterprises is that they have a multitude of solutions available to them in deciding how to build a QA organization that is aligned with their overall mission.

6.2 Results

In this project involve the four kinds of activity Manufacturing, Agent, Supplier and Customer. In this level the implement SQA and JIT method data security only 28%, 38% and communication complexity is 46%, 49% and verifier storage complexity is 26%, 28%. Finally the level of average in this project reached between 33.33% and 38.33%.

The focus on data security by implementing TSTQA method have been improved upto 68% and communication complexity maintained upto 78% . The storage verification complexity is 62%. Finally the level of average in this project to reached 69.3%.

Compare with the SQA the JIT and the TSTQA to improve in data security, communication complexity, verification storage complexity and average is having reached 69% in all level.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 1, January 2014

The table shows the accuracy of different methods JIT, TSTQ which is generates by using the column analysis

Metric \ Scheme	SQA	JIT	TSTQA
Data Dynamics	No		
Public audit ability	Yes	Yes	No
Data Security	28%	38%	68%
Communication Complexity	46%	49%	78%
Verifier Storage Complexity	26%	28%	62%
Average	33.33%	38.33%	69.3%

chart. The proposed approach has the better accuracy than the other.

Comparisons of Different Remote Data Integrity Checking Schemes

VII. CONCLUSION

The data from this research suggested that difficulty in measuring any of the software quality sub-characteristics is outlined in the software quality model is essentially the in either a decentralized or centralized QA organization. The results showed that a decentralized QA organization would have better TSTQA to measure software suitability. The software testing was insightful and of benefit for testing multiple products in different companies. Testing is difficult and requires detailed test plans. These plans must tie the testing approach to the software design and development schedule. This requires careful consideration of the product and demands that resources are prepared in advance of testing. The test plan ideally should be risk based so that it can yield better test benefits where test execution time is limited. Software testing is not sufficient in its own right to ensure that a quality product is realized. There are other quality factors that have to be considered and planned into the project lifecycle.

The QA process consists of a combined development and testing process, it is more beneficial in improving the quality of each project phase. With the emphasis of quality in this process, the experience of the QA team can strengthen the project team as a whole in the mindset of Quality Assurance. While the QA process is a combined effort, if the QA team can report independently of the development team, it can be more effective than a dependent team. In addition to an independent QA team, the inclusion of customers in the QA aspect of the project can also have a contribution to improved quality and reduced defects. It is also more effective to have the customers assess quality during different stages of the development cycle. The customers themselves may be included or a body of representatives which can assist with determining the quality assessment of the software.

Software quality metrics are required to track the defects and quality improvements at each stage of the project lifecycle. Graphs of the metrics can be used to plot trends over time of these software quality improvements to assist with the management of the test execution and quality initiative.

7.1 Future Work

This computerized method is a well-suited application for the real time business activities. It posses many robust features, still it can be expanded for additional features. The Timestamp Temporal Quality algorithm can be extended to analyze the products selling by month, year wise.

For further development of this system, any other data Software Quality Assurance techniques can be implemented to the organization concern. This system is capable of incorporated in any organization, which needs data Software Quality Assurance operations.

REFERENCES

1. Aaen, L. Mathiassen, “SPI: How to Navigate Improvement Projects”, John Wiley & Sons, Ltd, 2006.
2. C. Andersson, T. Thelin, P. Runeson, and N. Dzamashvili, An experimental evaluation of inspection and testing for detection of design faults, 2003 International Symposium on Source.
3. P. Anderson, The use and limitations of static-analysis tools to improve software quality, Jun 2008.
4. A.Aggarwal and P.Jalote, Integrating static and dynamic analysis for detecting vulnerabilities, Jun 2008 (PDNS2008).
5. C. Artho and A. Biere, Combined static and dynamic analysis, (IPSN2005), April, 2005, pp13–19.
6. E. Baker, M. Fisher, Schulmeyer and Mc-Manus, eds., Prentice Hall, Upper Saddle River, “Software Quality Program Organization,” Handbook of Software Quality Assurance, N.J., 1999, pp. 115–145.
7. P. Centonze, R.J. Flynn, and M. Pistoia, Combining static and dynamic analysis for automatic identification of precise access-control policies, Jan, 2000.
8. T. Chang, A. Danylyzsn, S. Norimatsu,J. Rivera, D. Shepard, A. Lattanze, and J. Tomayko, Continuous verification' in mission critical software development vol. 5, pp. 273-284 vol.5, 1997.
9. S.D. Cha, T.J. Shimeall, and Y.R. Kwon and S.S. So, An empirical evaluation of six methods to detect faults in software, 2002
10. R.A.Demillo, M.W.McCracken, R.J. Martin, J.F.Passafiume, Addison-Wesley Software Testing and Evaluation, 1987.

BIOGRAPHY