Software Automated Testing: A Solution to Cover Maximum Test Plan and to Increase Software Quality

G.Srikanth R.Venkata Ramana Chary

ABSTRACT
Software plays an important role in complex systems, especially for complex applications such as Video surveillance application, transportation, financial management, communication, biomedical applications and so on. For these systems, tester needs to concentrate on performances such as efficient operation, fault tolerance, safety and security. The basic problem is the complexity of the task, which has to be performed on a software application. Unlike hardware, software cannot break or wear out, but can fail during its life cycle [1].Software problems, essentially, have to be solved with quality assurance tools such as testing procedures, quality data reporting.

In this context, the paper proposes a new approach to automate software testing process to cover maximum test plan time and also to increase software quality. In this paper a method explained to cover each module without missing any test scenario and guarantees software with high quality and decrease testing life cycle time by establishing automation life cycle to develop scripts.

Keywords
PTZ: Pan Tilde Zoom, NTP: Network Time protocol, DDNS: Dynamic DNS, DNS: Domain Name System

1. INTRODUCTION
A software system is often subjected to conflicting requirements; in fact it has to be reliable in its application and, at the same time, has to follow the needs of the market with competitive costs [1, 2]. In this context the test process, through quantitative planning, tracking and automation, covers a fundamental role. Software reliability testing Combines the use of quantitative reliability aims with operational profiles (profiles of system use), that guide developers in the testing implementation. An important innovation would be the introduction of the automated test in order to reduce both time and cost of development without inducing, in the system, different Failures from the ones which we want to analyze. Inadequate and ineffective testing is responsible for many problems regarding software reliability faced by computer users. On the other hand, the complexity of modern software packages makes exhaustive testing difficult. Nevertheless, automated testing can help to improve efficiency of the testing process in order to identify areas of a program that are prone to failure. Automated testing can be applied in large portions of many applications, with reduction of the workload on overburdened testers. Until a few years ago, developers considered software testing as a secondary activity, if compared with the development phase, Nowadays the test represents, in many fields of application, the starting point for the development of the product and its cost is often comparable with the cost of the product development. In fact, it has been estimated that software testing, able to detect errors in source code, involves more than 50% of software development [4]. Time and cost can be significantly reduced through the use of automated test generators [4].Among the activities that allow the detection of nonconformity and potential failures in different phases of the software product, implementation software Verification and Validation (V&V) plays a fundamental role. To this aim, some standards and guidelines have been issued about software as a key component which contributes to system behavior and performance; some examples are represented by the International Standard ISO/IEC 9126 [5], which defines a quality model for a software product, in order to satisfy the customer optimizing the product and IEEE Standard 1012 [6] for Software Verification and Validation, which attempts to establish a common framework for all activities and tasks in support of software life cycle steps. In particular, amongst the different steps, the efforts on improving quality are concerned with V&V phases; these activities determine whether products of a given activity conform to the requirements and whether the software satisfies its intended use and customer needs. Verification and Validation is the activity in software production that allows production costs to be decreased and, at the same time, to increase software reliability [3, 7]. Verification is the process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. Validation is the process of checking in order to ensure compliance with software requirements [6]. V&V activities carry out the integration test that exposes defects in the interfaces and interaction between integrated components (modules); progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software will work as a system. Consider an example of complex application. Video surveillance is an electronic security system which is used to monitor, capture and store video data to be used for analysis. The system consists of analog and IP cameras, Recorders. Video surveillance software application enables customers to remotely manage the full suite Digital Video Recorders under one seamless graphical user interface. It supports more Digital video records. This software application supports to view live video, Playback video from device hard disk, record notifications from different devices, perform PTZ operations, configure devices, export video etc.

Configuration module of this application enables the user to specify

Camera settings like display title, brightness, contrast, camera tampering, camera title.

System settings like system information, RS485, RS232.

Network settings like enable NTP, DDNS, DNS.

Notification settings like notify IP address, notify port, notification type.

Recording settings like overwrite, camera number, record settings, compression settings.

Alarms settings like alarm title, sensor type, alarm events, full screen, send Email.

As video surveillance software supports many devices, manual validation of configuration module for all these devices is very cumbersome and time consuming task. It takes more time for validating Configuration module. By automating this module the validation time can be reduced

2. PROPOSED TEST METHODOLOGY

Several well-known methodologies for testing general purpose software exist in literature [8, 9]. Such techniques follow a structural or a functional approach, also identified as the white-box and the black-box approach respectively. The proposed methodology can be classified as a black-box approach, where the software under test has to be verified with a suitable studied set of inputs whose expected outputs are known only on the basis of the functional specifications. In addition to the black-box approach, we propose a test method with test sets well representative of the field behavior of the system, according to the block diagram shown in Fig. 1.

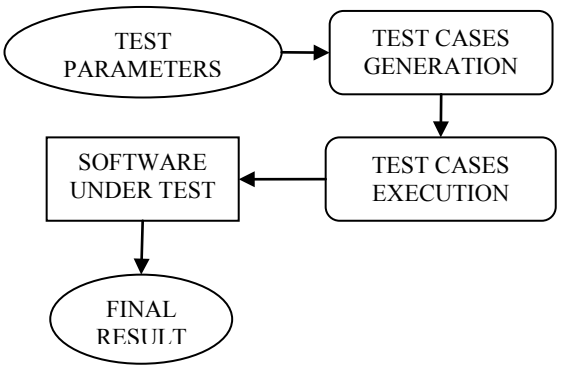

Fig1. Block diagram of the test method.

One can observe that “test parameters” are inputs, “final results” are outputs. The “software under test” is the object of the analysis and “test cases generation”, “test cases execution” is activities. The proposed methodology takes advantage of pseudo randomization voted to increase the number of test sequences and, at the same time, to simulate better the possible real conditions; it allows the test sequences to be modified without changing the programming code, for its high flexibility performances [10–11]. Test case generator creates quite unlimited sequences that can be iterated on the software under test. Both the inputs are Excel file where we give different inputs and the outputs is txt file where the result will be save. The test results, represented by the output of the software under test, are compared with the expected results obtained from the test cases, at the end of the testing phase; final results allow information to be deduced about the quality in use of the software under test and new data for dynamic test case generation [12–14]. Detailed steps concerning the comparison between obtained and expected results are shown in Fig. 2. If no differences between obtained and expected values are present, the test is stopped; vice versa, a fault has to be tracked and the software has to come back to the development team in order to be analyses and correct the bug; successively the test has to be carried out again. There is also the possibility of locating unclear values, such as ambiguities that have to be solved manually by repeating ambiguous sequences as video surveillance software supports many devices, manual validation of configuration module for all these devices is very cumbersome and time consuming task. It takes more time for validating Configuration module.

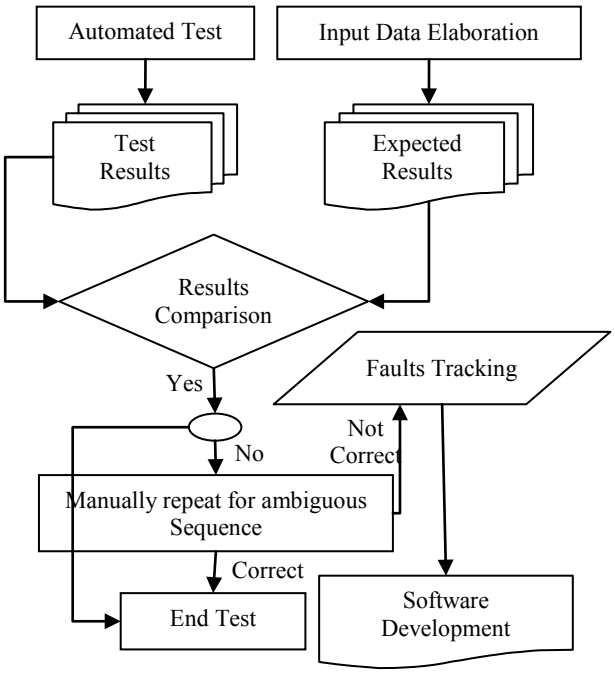

Fig 2. Automated software testing process

By automating this module the validation time can be reduced .To automate the Configuration module, a framework designed which gave more flexibility for further enhancement of scripts. Scripts were developed which run in un-attended mode. Automation life cycle defined to achieve good result.

3. AUTOMATION DEVELOPMENT LIFECYCLE

Most of the companies today were concentrating to automate everything for that companies were purchasing tools and providing training to automation team but by the end they were not achieving the goals reason is nothing but lack of planning to solve this problem before starting automation team should develop one lifecycle . Example here I developed one lifecycle to automate video application by this I completed the automation within the

time and achieved the goal. Fig3 explain the automation development lifecycle.

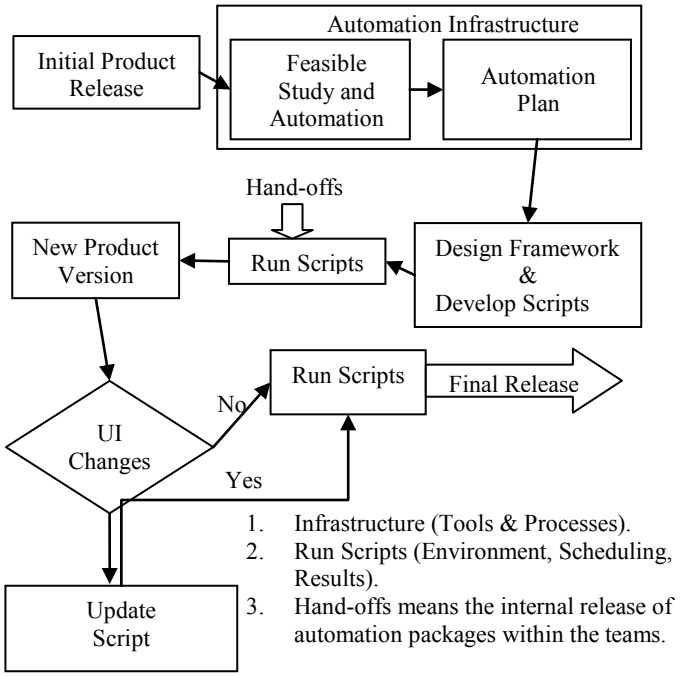

Fig3. Automation development lifecycle

In this initial product is release to automation team. Then automation team will do feasible study that which tool is suitable and how many days were need to automate , what is business benefit by automating ,How many automatable test cases are there , number of resources needed to automate . All these come under automation infrastructure and then team will develop automation framework which is reusable structure then automation team develop scripts with in time and these test scripts will give to Hand-offs person who don't have knowledge on scripting will execute the scripts and when next build release to QA team they will run the developed scripts on the application if any changes were in UI they will change the script else continue executing the scripts by following above lifecycle we can achieve good results.

4. USING FREEWARE TO AUTOMATE APPLICATIONS

Most of the companies were concentrating to improve the quality of the product and at the same time trying to reduce the cost of the testing efforts.

By using commercial tools like QTP , Load Runner they can automate the application but it is more cost .To reduce the cost and improve the quality of the product best solution to use open sources or freeware like Auto It ,WATIR etc. By that quality of the product increases and cost of the test life cycle decreases.

5. VALIDATION OF THE METHODOLOGY

The validity of the proposed approach is verified by considering an industrial application, constituted by a multifunction video surveillance system application as represented in Fig. 4. For this application the software suite covers a fundamental role in all the activities involved in the managing of different type of DVR's

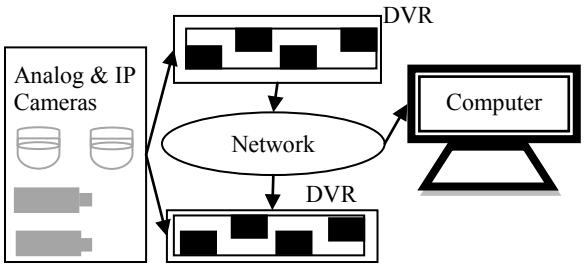

Fig 4.Typical components of a video surveillance application

6. RESULTS BY DOING AUTOMATION VIDEO APPLICATION

Below formula explain how much time we reduced by doing automation in terms of days

Say No: of devices = 27

Manual time taken for each device = 4 hrs

Automation time taken for each device = 1 hr

Total time taken to test manually for all devices TM = (No. of devices) *(Manual time taken for each device) hrs

Percentage of time saved = (1- (No. of devices / TM))*(100)

Percentage of time saved = (1-(27/ (27*4)))*(100) = 75%

Total time taken to test manually for all devices TM = (No. of devices) *(Manual time taken for each device) hrs

Total time taken to test all the devices using automation scripts TA = (No. of devices) *(automation time taken for each device) hrs

In days saved = (TM/24)-(TA/24) Days

= 4.5 - 1.125 days

= 3.375 days

By automating configuration part of video surveillance software application .The quality of the product increases and manual test effort were decreases .Fig 5 explains how quality of the product increases and days of effort decreases by doing automation

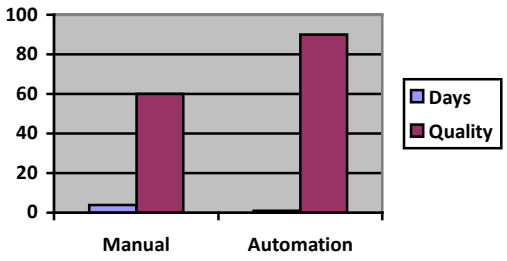

Fig 5. No. of Days Vs Quality

7. CONCLUSIONS

Paper proposes a new approach to automate software testing process to cover maximum test plan time and also to increase software quality. In this paper a method explained to cover each module without missing any test scenario and guarantee software with high quality level, decrease of the testing time. Establishing automation life cycle to develop scripts with in time and benefits of using freeware language like Auto IT to developing automation scripts. Application will be tested by using test scripts and produce results with in less time .Moreover the research wishes to define some parameters of the software life cycle to show the generality of the proposed technique.

8. REFERENCES

[1] G. Betta, D. Capriglione, A. Pietrosanto, P. Sommella, A statistical approach for improving the performance of a testing methodology for measurement software, IEEE Transactions on Instrumentation and Measurement 57 (6) (June 2008) 1118–1126.

[2] A.Birolini, Reliability Engineering — Theory and Practice, Springer-Verlag3-540-40287-X, 2004.

[3] D. Galin, Software Quality Assurance: From Theory to Implementation, Pearson Addison Wesley, Harlow, England, 2004.

[4] E. Diaz, J. Tuya, R. Blanco, Automated software testing using a metaheuristic technique based on tabu search, Proceedings of 18th IEEE International Conference on Automated Software Engineering, 2003, pp. 310–313.

[5] Burnstein, Practical software testing: a process-oriented approach, Springer- Verlag, New York, 2003, ISBN:0-387-95131-8.

[6] H. Freeman, Software testing, IEEE Instrumentation & Measurement Magazine 5 (3) (September 2002) 48–50.

[7] ISO/IEC 9126: Information technology — software product evaluation — quality characteristics and guidelines for their use, 2001.

[8] S.C. Ntafos, on comparisons of random, partition, and proportional partition testing, IEEE Transactions on Software Engineering 27 (10) (October 2001) 949–960.

[9] S. Stoica, Robust test methods applied to functional design verification, Proceedings of IEEE International Test Conference, September 1999, pp. 848–857.

[10] ANSI / IEEE St. 829, Standard for software test documentation, , 1998.

[11] J.D.Musa, Introduction to software reliability engineering and testing, Proceedings of the 8th nternational Symposium on Software Reliability Engineering, 1997.

[12] Reliability Analysis Center, Introduction to Software Reliability: A State of the Art Review, Reliability Analysis Center (RAC), 1996.

[13] M.A. Bailey, T.E. Moyers, S. Ntafos, An application of random software testing, IEEE MILCOM, Conf. Rec., vol. 3, November 1995, pp. 1098–1202.

[14] S.M. Phadke, Quality Engineering Using Robust Design, Prentice-Hall, Englewood Cliffs, NJ0137451679, 1989.

[15] ANSI / IEEE Std. 1012, IEEE Standard for Software Verification and Validation Plans, 1986.

9. AUTHORS PROFILE

G.Srikanth received Masters Degree in Software Engineering from the University of JNTU, India and Bachelor degree in Computer Science & Engineering from the University of JNTU, India. From 2008 to 2009, he has been with the Department of Computer Science & Engineering; at VITS Engineering College as an Assistant Professor .His current research interests include Automation of new applications and reduce test life cycle and manual efforts, reliability evaluation test, fault detection and diagnosis, quality control, where his publications are focused.

R.Venkata Ramana Chary received Masters Degree in information technology; presently pursuing his PhD from GITAM University AP, India. And working as an Associate Professor, at Padmasri Dr.BV Raju Institute of Technology AP, India. His current research interests include fault detection and diagnosis, quality control, performance measurements, algorithms analysis and programming Techniques.