International Journal of Scientific Research & Engineering Trends
Volume 4, Issue 4, July-Aug-2018, ISSN (Online): 2395-566X

Test Suite Minimization and Parallel Scheduling

M.Tech.Scholar Shammi
Dept.of CSE DIET Karnal
University KUK
Haryana, India
E-mail- ku08sham@gmail.com

Asst.Prof.Nitin Bansal
Dept.of CSE DIET Karnal
University KUK
Haryana, India
E-mail-nitinbansal003@gmail.com

***Abstract* – Testing is a vital but luxurious task vital for the assembly of elevated quality software. As such, there is outstanding possible for each useful method that enables the detection of extra faults alongside manipulated software assessing funds. One assessing strategy is to orient the assessing regimen concerning concrete, attainable criteria. Regression assessing is one of the vital maintenance hobbies, but it needs a outstanding deal of period and effort. Often, software firms have pressures alongside period and budget, so luxurious and time-consuming regression assessing might be a main burden for them. To vanquish these design and cost-related concerns alongside regression assessing, countless researchers have counseled assorted price competent regression assessing methods in particular, examination case prioritization methods have been actively learned because they furnish appealing benefits, such as flexibility for testers who demand to adjust their assessing efforts for the manipulated period and budget. The Counseled Examination case Minimization methods endeavor to design examination cases in an killing order according to given criterion. The main intention of this Minimization is to rise the likelihood that if the examination cases are utilized for regression assessing in the given order, they will extra closely encounter the goal than they should if they were gave in a little supplementary order.**

***Keywords*- Test Suites, Component Based Software, Time-Consuming, Software Firms etc.**

I. INTRODUCTION

1. Software component

Software constituent is a software constructing block that is confirms to a constituent ideal and it could be independently composed and used missing change in step with the constitution common [1].

2.Component-based software

Component-primarily based software is a set of selfcontained and loosely coupled materials that permit plug-and play.The components might have been composed in disparate software program design tongues and gave on disparate operational durations, and distributed across geographic distances. A little materials could be industrialized inresidence, as others could be 0.33-celebration or business-offthe-shelf (COTS) components, alongside the idea software unavailable.

II. COMPONENT-BASED SOFTWARE VERSUS TRADITIONAL PROGRAMS

The goal of software development is to ensure the high quality of a software product, like high reliability, reusability and maintainability. To achieve these goals in component-based engineering (CBE), first of all we need to understand the main characteristics of component-based software.

1. Properties of Component-Based Software

Differing from old and traditional software systems, component-based software usually shows the following unique characteristics:

2. Source code availability

When growing factor-primarily based software, developers favor, rather than asking for the program, to accept COTS elements every time appropriate materials are available.

3. Distribution

With the progress of Internet, greater and further issue-based totally software is distributed across networks. Disparate elements will be projected, industrialized and consolidated in distributed nature.

4. Reusability

One of the principle major goals of component-based software engineering is to develop software reusability, in order that it may enhance the exceptional of upcoming produce and at the alike duration, reduce project cost.

III.COMPONENT-BASED SOFTWARE INFRASTRUCTURE

1. Component Model

A constituent best is the spine of a aspect-based totally System. It affords essential prop for constituent progress,charter,contact,placement,andevolvement.Currently,.NET/COM/COM+, CORBA, and EJB.

1.1 Enterprise Java Bean (EJB)

- Neutral
- Mechanical push
- Mechanical pull

EJB industrialized with the aid of Sun Microsystems, is a constituent ideal for server-based factor-primarily based

software production. The Java bean is the more reusable constituent in EJB. The key mechanism that manages beans and their touch is the EJB field or EJB server.

1.2 NET Framework

.NET is the ultra-modern Framework from Microsoft that supports element-based totally software improvement. The adoption of the .NET framework as a computing length for disparate software layout tongues and the adoption of XML as the manner for contact amid disparate components notably decorate the interoperability trouble. On the customer facet,.NET helps all types of mechanisms, from Windows CE to Windows XP. Over the server side, all forms of employer services have been consolidated i.e. SQL server and transactions server.

1.3Component Object Model (COM)

COM is an older architecture for component-based software. The first COM model included only basic features for component integration, packing, and a binary standard to ensure the interoperability among different components. DCOM, an extension of COM, provides the ability to integrate components in a distributed environment.

1.4 Common Object Request Broker Architecture (CORBA)

CORBA as defined by OMG provides a common framework to integrate components regardless their different locations, programming languages and platforms. A critical part of CORBA is the object request broker (ORB), which provides the basic mechanism to achieve the transparency characteristic.

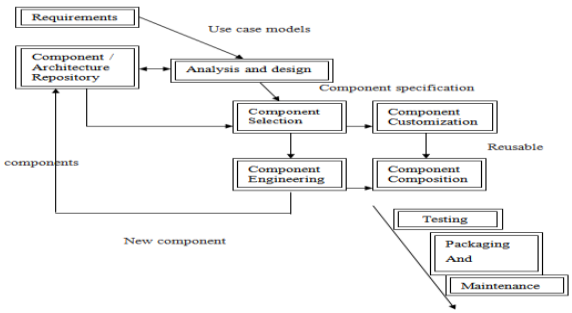

Fig.1 Engineering process for Component-Based Software

IV. ENGINEERING PROCESS FOR COMPONENT-BASED SOFTWARE

Combining a cluster of dependable software constituents couldn't yield a fairly reliable aspect-primarily based software system. The key to accomplishment is the software system, "a framework for responsibilities which might be demanded to craft the fantastic software" [5].

1. A process model for component-based software

Component-based totally software will speedy evolve above its lifetime, so a processing perfect for aspect-based preparations needs to be an evolutionary version. The finished engineering manner for component-based totally software can be tear into the pursuing intervals, as shown in Figure 1.1[6]: necessities, scutiny, design and implementation. These periods incorporate constituent choice, customization and constitution, assessing, placement, and protection.

2.Requirements

Requirements analysis will derive specifications for system under development.During this phase, we follow conventionally required engineering approaches; and the UML use-case model is a typical document of this phase

3. Analysis and design

In this phase, software components and their specifications will be determined. The interactions, dependencies, and structural relationships among these software components will be decided as well.

4.Component selection, development and customization

Component choice is the process of discovering matching elements from the repository of reusable additives. The constituent choice method might be able to discover a cluster of candidate components, a manipulate adoption frequently desires a flawless fit of the candidate elements alongside constituent specification. Each mismatch within the above spans desires customization of additives. If, later on the manner of constituent choice, no eligible candidate is diagnosed, a brand new constituent will be industrialized in keeping with the constituent specifications.

5. Composition of Components

Component constitution is the method of incorporating all obtainable elements jointly, that includes all presently industrialized elements, customized reusable and COTS additives. To properly incorporate these components, constituent design and a selected constituent best needs to be seized into concept.

6. Testing

Testing of factor-based totally software have to be believed in elements: assessing of person ingredients and assessing of element-primarily based software. In finish, after each single individual constituent has been safely tested, assessing of issue-primarily based software will generally attention on contact amid disparate ingredients.

7. Deployment

Component-based software can be used as a component for another system. Therefore, it is necessary to provide packaging and deployment capacity.

8. Maintenance.

The maintenance for CBS focuses on how to adequately the model various modification activities and how to determine how much effort is adequate.

V. COMPONENT–BASED SOFTWARE TESTING

To ensure the delivery of quality software, effective and efficient testing is a key part of software development. Test methodologies [8] are generally divided into two types: black box and white box. Black-

box approach, such as random testing and functional testing, in which do not require the knowledge of implementation details, but when applied to CBSE systems they may encounter a problems similar to those that found in testing of the traditional programs. In white-box approach, internal structure of the program is tested, so in this testing, firstly the program structure is examined and then various test cases are derived based on the program logic.

VI.ISSUES AND CHALLENGES OF COMPONENT-BASED SOFTWARE TESTING

1.Why is ok checking out for component -based totally software program necessary?

Why does an consolidated association of dependable Software ingredients call for to be more examined? A little blame may be attributed to the lack of ok components that want a whole lot of reconfiguration or maybe software modifications of the early additives. Nevertheless, even after "flawlessly matched" elements are consolidated, the pursuing topics ought to yet be encountered, that over again want more assessing efforts.

- Inconsistent infrastructure and environment
- Inconsistent interaction model

2. Difficulties in adequate testing and maintenance for component-based software

There are a few fundamental factors that affect our testing and protection activities.

2.1 Code availability- For COTS components, basis Application is extraordinarily regularly not obtainable, and the contact amid parts will have to pass throughout Pre defined constituent interfaces. The lack of basis Application motives quite a few setbacks.

2.2Performance and reliability evaluation-Performance is one of the best features that are mainly affected by component-based software features. Reliability and usability also need to be re-examined. To analyze these quality features of component-based software, a key issue is how to reuse the results provided by the software components.

2.3 Adequacy- Test adequacy is one of the toughest topics in assessing factor-based totally software program. On one hand, no longer all mounted exam adequacy standards can be utilized, highly for the ones basis code established criteria. On the supplementary hand, even nevertheless a little black-box established criteria can be followed, the concern is: Do we yet choice to apply the ones criteria?

2.4 Maintenance

When a constituent in factor-primarily based software is adjusted or upgraded, a protection interest happens [9, 10]. Due to infinite of the characteristics of aspect-based software, difficulties may be encountered after set up upkeep ways are carried out. The charge of protection length for trendy software as two-thirds of the finished charge and it could be but greater for retaining CBS.

VII. COMPONENT-BASED SOFTWARE'S REGRESSION TESTING

Regression testing, which is aimed to conforms that the altered or modifird software still now fullfills the specifications validated before the modification activities, is much important for the success of a component-based system. Regression testing involves many different issues, for instance, test-suite management and regression-testing automation.

Regression testing for component-based software is significantly different from regression testing for traditional software. This is mainly because traditional software maintenance is usually carried out by the same team; therefore, the team has a full knowledge of the software. For component-based software, maintenance is often carried out by component providers. After that, users have to maintain their systems according to the changes that the component providers have made. Component providers have full control of their components, and can therefore use traditional approaches to maintain their components.

VIII. PREVIOUS RELATED WORK

Lionel C. Briand et al., 2004 [17] This paper describes an empirical research of the value effectiveness of famous country-based trying out techniques for training or clusters of instructions that showcase a kingdom-based behavior. This is nearly relevant as many item-oriented methodologies recommend modeling such components with state charts that could then be used as a basis for checking out. Their outcomes, based on a series of three experiments, show that in maximum instances kingdom-based totally strategies aren't likely to be enough with the aid of them to trap most of The faults present inside the code.

Though beneficial, they need to be complemented with black-box, useful testing. They consciousness here on a specific technique, Category Partition, as this is the most generally used and referenced black-field, practical testing method. Two exclusive oracle strategies had been implemented for checking the success of check cases. One is a very precise oracle checking the concrete nation of gadgets while the alternative one is primarily based at the notion of nation invariant (abstract states).

Rountev Atanas et al., 2004 [18] Testing of polymorphism in item-oriented software program might also require coverage of all feasible bindings of receiver training and target strategies at name web sites. Tools that degree this coverage need to apply class evaluation to compute the coverage necessities. However, conventional

whole-application class evaluation can't be used when trying out incomplete packages.

To remedy this hassle, they present a general approach for adapting whole-application elegance analyses to operate on application fragments. Furthermore, seeing that analysis precision is crucial for coverage tools, they offer precision measurements for several analyses by using figuring out which of the computed insurance necessities are simply possible for a set of problem additives. Their paintings permits the usage of entire-program magnificence analyses for checking out of polymorphism in partial packages, and identifies analyses that probably are correct applicants to be used in coverage equipment.

Li, Keqin et al., 2006 [19] In this paper the design of complicated structures, e.g., telecom services, is these days generally based on the combination of additives (COTS), loosely coupled in dispensed architectures. When components come from third party resources, their inner structure is usually unknown and the documentation is inadequate.

Therefore, the device integrator faces the hassle of presenting a required gadget assembling COTS whose behaviour is barely detailed and for which no version is usually available.

Guo Fuliang et al., 2012 [20] In this paper element-based software increase has grow to be the main expand method.Meanwhile the way to guarantee its great is an important studies content, such as component-based totally software program integration checking out. This paper introduces a metadata configuration version describing indoors facts of element to boom testability.

The metadata configuration version(MCM) consists of three components: simple attributes, additive attributes and incorporated attributes. And a wellknown framework of the version is further given, that's consisted of several training. Each magnificence consists of numerous attributes, and their meanings are defined. Finally the case have a look at based on previous model is finished, and the corresponding effects are given. All those display successfully that the models they supplied are valid and helpful for issue-primarily based software program integration trying out.

Patrícia DL Machado et al., 2007 [21] In this paper an integration trying out approach for componentbased software program is presented. The technique, based totally at the extensively used UML (Unified Modelling Language) notation, covers a complete integration checking out method at a contractual element degree and it's far supported by the usage of gear.

Components and their interfaces are precise by using UML diagrams and OCL (Object Constraint Language) constraints. Software underneath test is constructed from composition of components in a standard thing-based totally software program improvement process. A case examine that is carried out inside the Java language is presented to demonstrate the software of the technique.

IX. PROBLEM DEFINITION

Component hooked up Assessing can be gave employing Assorted instruments obtainable in Assorted Languages, For Example JUNIT for Java beneath Net beans Environment.

- The Main task of this task is to Examination all the parts in a particular Arrangement as a finished or in supplementary phrases the aim is to examination the
- Integration of the materials within the system.
- The early duration is to craft Constituents of the Association that may be assessing through Instruments inclusive of J unit.
- The important undertaking within the next period is to design and bring constituent exam cases for the crafted examination fashions. An automatic tool/software may be applied for generating of examination instances.
- Finally those constituents can be tested for Integration afterward asking for Examination Case minimization algorithms.
- Comparison of the Previously Generated Test Cases and Minimized Integrated Test cases will be Performed for Detection of Fault Tolerance and hence era of Optimal Test cases.

X. OBJECTIVES

- To Study Component Testing and its Environment (Junit and Netbeans).
- To Create Large Component Based System Model(Must support Component Integration).
- To Create Unit test for Each Component.
- To Apply Unit Tests Separately Without minimization Algorithm.
- Apply minimization Algorithm for each component and create and Integrated Unit Test.
- To Evaluate the Performance and Fault Toloerance of Minimized Integrated Test cases.

XI. METHODOLOGY

When developing a component-based software system, we have to test each component individually. Components may have been developed by different people, written in different programming languages, and executed in different operating platforms. Therefore, the interfacing among components needs to be tested. Integration testing is necessary to ensure communication between components is correct.IEEE defines Integration Testing as: "Testing in which hardware components, software components or both are combined and then tested to evaluate interaction between them".To carry out our study we need to study extensively on component Testing and its

Environmen(Junit and Netbeans),what we first want to create is a large Component Based System Model (That must support Component Integration).

Unit test cases for Each Component are automatically crated using JUnit in Netbeans.Unit Tests are applied Separately and Sequentially Without minimization Algorithm, and total time is captured for such test suite.

XII. WORKFLOW OF TEST CASE PRIORITIZATION

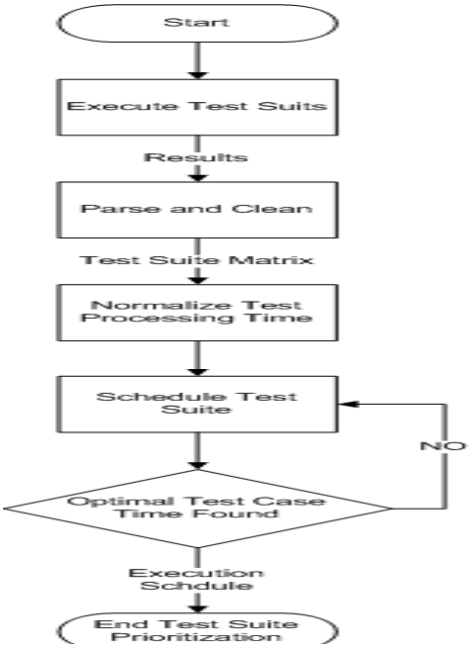

Fig.2 Working model.

A arranging mission set is crafted for request of arranging set of rules reliant on processing duration and weights.The examination fits are projected for each solitary processor and multiprocessor settings to locate the most suitable aftermath and arranging length.

First, we gift the examination cases in Weka assignment underneath a solitary examination suite.The exam suites are generated employing computerized exam suite creation employing JUnit. The aftermath produced are within the shape of examination case,quantity of examinations run, its ruin and accomplishment Later that exam match aftermath are parsed and wiped clean in order that they may be imported in MATLab Workspace.

The Examination Suite Matrix so received is subsequent normalized to millisecond advantages for common scheduling.The normalized matrix encompasses processing length of each single exam and completed no of examinations gave in that alike exam case.

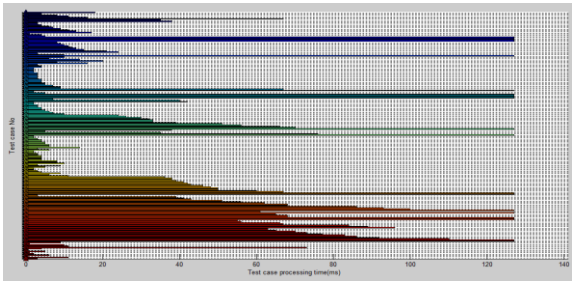

Fig.3 plot showing All 200 test and the Max Processing time.

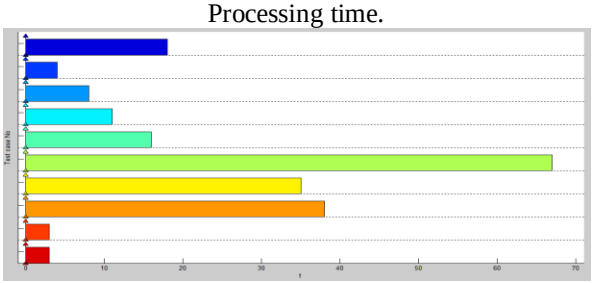

Fig.4 processing time First 10 Test Cases in The Test Suite.

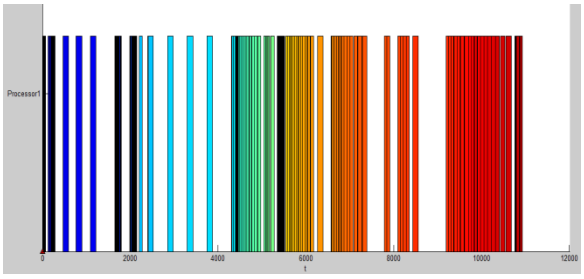

Fig.5 Single Processor Scheduling of Test Suite.

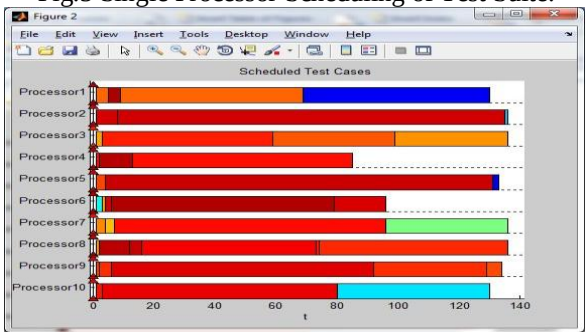

Fig.6 Scheduled Test Cases over 10 Processors.

XIII. CONCLUSION AND FUTURE WORK

Test case Minimization methods undertaking to layout examination cases in an killing order in keeping with given criterion. The main aim of this Minimization is to rise the probability that if the examination instances are utilized for regression assessing within the given order, they will extra intently encounter the purpose than they need to if they have been gave in a touch supplementary order. For example, testers may want to layout exam cases in an order that achieves application coverage on the fastest rate probably,exerts functions in order of

predicted frequency of use, or increases the chance of noticing faults foremost in checking out. Various Trials in Constituent established integration assessing are found out in this file and the paintings could grant a bit visions and treatments to understand such trials in constituent hooked up consolidated trying out. Examination case minimization and prioritizations are an important element in constituent mounted integration trying out. Examination case prioritization methods can be industrialized to design examination instances for killing in an order that endeavors to upward thrust their effectiveness at stumble upon a little presentation intention.

The major intention of Examination case Minimization is to upward push the likelihood that if the examination cases are applied for regression assessing inside the given order, they'll more intently come upon the goal than they need to if they have been gave in a little supplementary order. For instance, testers may want to layout examination instances in an order that achieves application insurance at the quickest price likely,exerts features in order of expected frequency of use, or will increase the chance of noticing faults main in checking out.

Regression assessing is a critical technique transpiring as the maintenance period of the software lifecycle. It desires massive numbers of examination cases to vow the attainment of a unique diploma of best. Making, examination case could produce exponentially. This scutiny paper recommended examination case minimizing employing the advised set of rules.

The Examination case minimization may want to cause most appropriate duty detection efficacy. The after math make clear the effectiveness of the counseled set of rules by completed momentous exam case length intricacy reduction, Reliant at the no of the processors applied the set of rules can limit the examination instances enhancing performance by way of 2-three instances

ACKNOWLEDGMENT

With great pleasure and honour I convey my sincear gratitude to Hon. Ass.Prof.Nititn Bansal HOD of CSE department of DIET,karnal for his professional guide ,keen intrest and constant support to the my thesis work.
I highly thankful to my guide and all faculty member of DIET karnal whose untiring efforts and hard work helps me in successful implementation of the thesis work Technical support is utmost importance in completing research work on test case minimization .I acknowledge the help came from Nitin sir and my cse department and my parents also

REFERENCES

[1]. David Z. Pan, Senior Member, IEEE, Bei Yu, And Jhih-Rong Gao “Design For Manufacturing With Emerging Nanolithography” IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 32, No. 10, October 2013 (9, Regular)
[2]. M. Lu, Et Al., “Novel Customized Manufacturable DFM Solutions,” Proc. SPIE Photo Mask Technology 2012, Vol. 8522, Pp. 852223, December 2012.
[3]. Sergio Gomez And Francesc Moll. “Lithography Aware Regular Cell Design Based On A Predictive Technology Model.” J. Low Power Electronics, 6(4):1–14, 2010
[4]. B. Le Gratiet, F. Sundermann, J. Massin, Et Al., “Improved CD Control For 45-40 Nm CMOS Logic Patterning: Anticipation For 32-28 Nm”, In Proceedings Of SPIE Vol. 7638,76380A (2010)
[5]. Shi-Hao Chen, Ke-Cheng Chu, Jiing-Yuan Lin And Cheng-Hong Tsai “DFM/DFY Practices During Physical Designs For Timing, Signal Integrity, And Power” 2007 IEEE Conference.
[6]. Wing Chiu Tam And Shawn Blanton “To DFM Or Not To DFM” IEEE Asia Pacific Conference On Circuits And Systems, 2006.
[7]. Raina Rajesh “What Is DFM & DFY And Why Should I Care?” INTERNATIONAL TEST CONFERENCE 2009
[8]. Garg Manish, Kumar Aatish “Litho-Driven Layouts For Reducing Performance Variability” IEEE 200
[9]. Daehyun Jang, Naya Ha, Joo-Hyun Park, Seung-Weon Paek “DFM Optimization Of Standard Cells Considering Random And Systematic Defect” International Soc Design Conference 2008.
[10]. Sergio Gomez, Francesc Moll, Antonio Rubio “Design Guidelines Towards Compact Litho-Friendly Regular Cells” SPIE Photomask Technology 2012
[11]. "Design For Manufacturability" Http://Www.Mentor.Com/Blogs/
[12]. “Litho Friendly Design Kit, A Tool Of DFM Strategy”, (Http://Www.Eetimes.Com/Electrical-Engineers/Education-Training/Tech-Papers/4130133/Litho-Friendly-Design-Kit-A-Tool-Of-DFM-Strategy).
[13]. Y. Borodovsky, “Lithography 2009 Overview Of Opportunities,” In Proc.Semicon West, 2009.
[14]. J. A. Torres, “Layout Verification In The Era Of Process Uncertainty: Target Process Variability Bands Versus Actual Process Variability Bands,” In

Proc. SPIE Design Manufacturability Through Design-Process Integration II, Vol. 6925. 2008, Pp. 692509-1–692509-8.

[15]. A. Carlson and T.-J. Liu, "Negative and Iterated Spacer Lithography Processes for Low Variability And Ultra Dense Integration," In Proc. SPIE Optical Microlithography XXI, Vol. 6924. 2008, Pp. 69240B-1–69240B-9.

AUTHOR PROFILE

M.Tech Scholar Shammi
Dept.of CSE DIET KARNAL
Address- Karnal, India
E-mail-ku08sham@gmail.com