An efficient technique for test case prioritization and generation using ADDTL_TCP and genetic algorithm approach

A Dissertation proposal

Submitted

By

Pooja Randhawa

To

Department of computer Science Engineering

In partial fulfillment of the requirement for the

Award of degree of

Master of technology in computer Science Engineering

Under the guidance of

Mr. Mohit Arora

(Assistant Professor)

(May 2015)

ABSTRACT

Testing is very important part for any successful software development. Regression testing is the process of testing which is used to modify new system by using previous test suites. Due to the large test suites, retesting take more time to execute. This is the main issue of retesting, can be handled with test case prioritization techniques. In Test case prioritization technique test cases have higher priority are executed before lower priority. The main objective of the test case prioritization is to detect the faults as soon as possible so that the work can begin earlier .In test case generation we produce the test cases through manual or automatic.

In this research, a model based test case prioritization has been used and various techniques has been employed for the creation of test cases, after generation of test cases these test cases has to be implemented on the different types of components present in software system. To detect early bugs in the software these cases has to be implemented on the basis of priority. In this research the ADDTL_TCP approach is used for the purpose of test case prioritization. The priority table generated by this approach is used for optimizing these cases by implementing genetic algorithm.

CERTIFICATE

This is to certify that Pooja Randhawa has completed M.TECH dissertation proposal titled "**An efficient technique for test case prioritization and generation using ADDTL_TCP and genetic algorithm**" under my guidance of her original investigation and study. No part of the dissertation has ever been submitted for any other degree or diploma.

The dissertation is fit for the submission and the partial fulfilment of the conditions for the award of M.Tech Computer Science and Engineering.

Date: 4 May 2015

Signature of advisor

(Mr. Mohit Arora)

ACKNOWLEGMENT

I would like to thanks my mentor **Mr. Mohit Arora** for their deepest appreciation. I would like to thank him for encouraging me. His advice on research have been priceless. He has shown the substance of genius and attitude. He continually and persuasively conveyed of spirit adventure in regard to research and scholarship, and excitement in regard to teaching. Without his supervision and constant help this research will not possible

DECLARATION

I hereby declare that the dissertation proposal entitled, **“An efficient technique for test case prioritization and generation using ADDT_TCP and genetic algorithm”** submitted for the M.Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date: 4 May 2015 Reg. No.

10812657

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Software engineering is an approach that deals with design, development and maintenance of software. Software project management has wider scope than software engineering process as it also involves communication, pre and post-delivery support etc.

Testing is the process to identifying the difference between the expected the actual result. Testing is an important phase of software development which aims at producing high reliable system and maintaining quality [17]. Reliability means probability of the failure free operation in a given environment in given interval of time. Maintaining means the system is able to work in when system is not working properly.

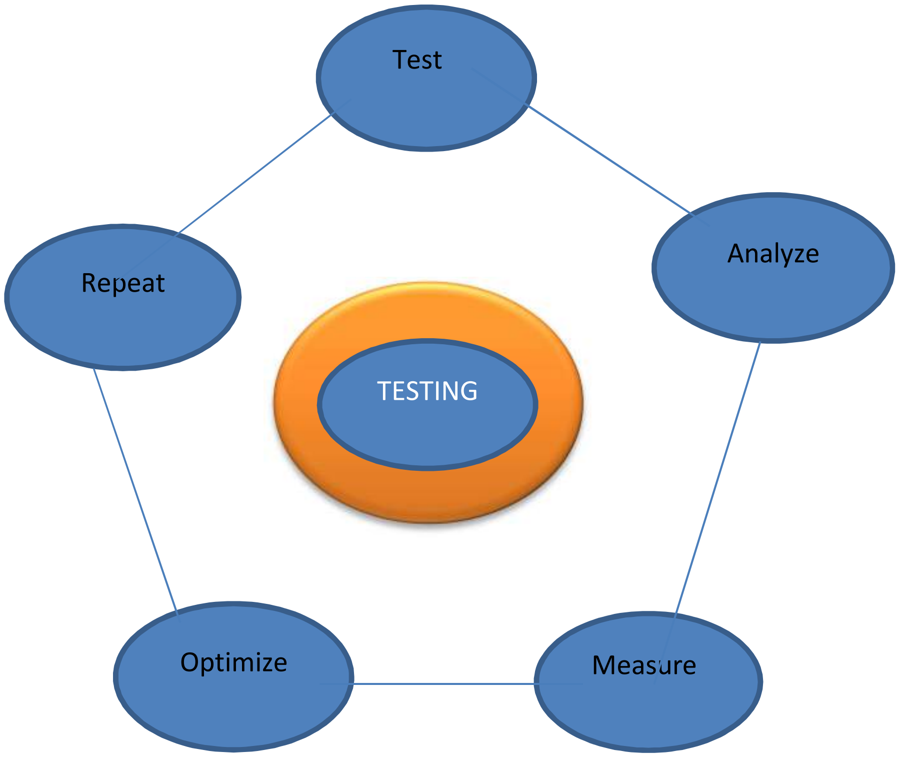


Fig 1.1 Testing process

1.1 Software Development Life Cycle

A Software Development Life Cycle is essentially a series of phases through which the software undergoes during its development. Various phases are: feasibility study, requirement gathering and analysis, design, coding and unit testing, integration and testing, maintenance [18].

- **Feasibility study**

The goal of this phase is to check whether it is feasible to develop software or not.

- **Requirement Gathering and Analysis**

In this phase, all the requirements of the customers are gathered to develop software. Requirements analysis sometimes requires individuals/teams from client as well as service provider sides to get detailed and accurate requirements; often there has to be a lot of communication to and from to understand these requirements.

- **Design**

In systems design, the design functions and operations are described in detail, including screen layouts, business rules, process diagrams and other documentation [4].

- **Coding and Unit Testing**

In this phase code is accomplished in modules. Then unit testing is done of every individual module.

- **Integration and Testing**

All the modules are integrated in a planned manner. Once all the modules are integrated, then integration testing will be done.

• **Maintenance**

The placement of the system includes deviations and developments. It is the most important phase of SDLC which takes more effort than other phases.

1.2Classification of testing techniques

- Static and dynamic
- White box testing and Black box testing
- Manual and automated

1.2.1Static and dynamic testing

• **Static testing**

In static testing verification though many approaches are available these are reviews, walkthroughs, or inspections. This type of testing used by developers who wrote the code .It does not involve actual program execution[19].

• **Dynamic testing**

In dynamic testing validation technique is used to executing the program code with a given set of test cases .Examining the result to check whether it operated as expected .In dynamic testing confirmation of the software function according to the specification.

1.2.2Manual and automated testing

In manual software testing is the process of testing the software that is carried out by group or an individual. Manual testing takes more time as compared to automatic testing .Automatic software testing is the process of creating test strips, which can be run automatically, repetitively through several iterations[21]. As compare to manual and automated testing automated testing is more efficient for time.

1.2.3White box and black box testing

- **white box testing**

White box testing is also known as clear box testing, glass box testing, transparent box testing. White box testing used to test the internal structure and the working of any program or application, as opposed the functionality of black box testing. In white box testing programing skills and internal structure of the system are used to design the test cases.In white box testing tester pick the input through the coding and then determine output [6].

- **Black box testing**

Black box testing delights the software as a "black box", scrutinizing the functionality without any information of inner implementation .Test cases derived from the specification of the program [7] .Two advantages of functional testing that are independent how the program implemented, so that there is no effect on the test cases during the implementation .Second is, the test case occur in development stage are occur in parallel with implementation[22] .This can reduce the development time for the project To progress the cost and efficiency of regression testing different approaches used:

1.3 Four regression testing technique

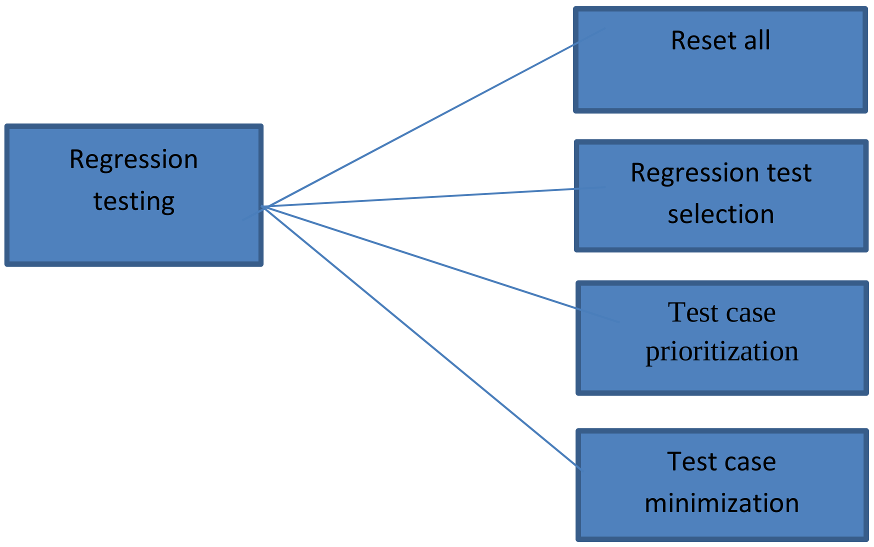


Figure 1.2Regression testing techniques

1.3.1 Reset all

This method is one of the conventional method. In regression testing there areall the test from existing test suites are returned. This is the most expensive technique as compared to other regression testing techniques because regression test suites are very costly to execute in full as it require more time and budget[23]. Reset all is one of the method for regression testing in which all the tests in the existing test bucket or suit should be re-executed .This is very expensive as it require huge time.

1.3.2 Regression testing test selection

The main advantage of this technique is that this technique is less expensive as compared to other testing techniques. This technique reduces the cost of regression testing by the selection of subset of an existing test suite to use in retesting a modified program. Test selection techniques broadly classified in to two categories.

Coverage techniques: coverage technique depends on the coverage criteria. This is used to find coverable program component these component have been modified and after that select the test cases those test cases work on these components.

Minimization techniques: this technique also similar to coverage the difference is only that there is the selection of minimum number of test cases.

1.3.4 Test case prioritization

The main advantage of test case prioritization to detect the fault as soon as possible and to rank the test cases execution in order and in test case prioritization there are some benefits of this technique with the help of this it provide a way to identify or find the more bugs and this is under resource constrain condition[24]. In simple language prioritization meaning is: testing in order wise. Order also can be increasing or decreasing but mostly in test case prioritization increasing order used. In test case prioritization firstly check the highest priority after that lowest and at the end minimum lowest priority.

1.3.5 Regression test case minimization

In the test case minimization we select the test cases that is used to the execution of the program then the test cases we select are in huge numbers then we can reduce those with the help of test case minimization technique and this technique will help to identify the redundant test cases.

1.4Component-based software engineering

Component-based software engineering (CBSE) is a process that accentuates the outline and development of PC based frameworks utilizing reusable programming "segments." Clements *CLE95+ portrays CBSE in the accompanying way: CBSE is changing the way vast programming frameworks are created [11]. CBSE encapsulates the "purchase, don't fabricate" reasoning upheld by Fred Brooks and others. In the same way that early subroutines freed the software engineer from contemplating points of interest, CBSE shifts the accentuation from programming to making programming frameworks. Usage has offered approach to coordination as the core interest. At its establishment is the suspicion that there is sufficient shared characteristic in numerous vast programming frameworks to advocate creating reusable segments to adventure and fulfill that shared trait [25]. Segment based programming improvement methodology is taking into account the thought to create programming frameworks by selecting fitting off-the-rack segments and after that to collect them with a decently characterized programming structural engineering.

Component-based software engineering is a member of programming designing which underscores the partition of concerns in appreciation of the far reaching usefulness accessible all through a given programming framework. This practice expects to realize a just as colossal level of profits in both the transient and the long haul for the product itself and for associations that support such programming. Programming specialists see segments as a component of the beginning stage for service orientation. Parts assume this part, for instance, in Web Services, and the sky is the limit from there as of late, in Service-Oriented Architecture whereby a part is changed over into an administration and in this manner acquires further attributes past that of a standard part. Parts can create occasions or expend occasions and can be utilized for event driven architecture[26].

1.4.1 Components

Components give an administration without respect to where the segment is executing then again its customizing dialect [12]

• A segment is a free executable substance that can be comprised of one or more executable articles;

• The segment interface is distributed and what not cooperation's are through the distributed interface Segments communicate through interfaces.

An interface is the association with the client that will communicate with a segment .In the event that an interface is changed the client necessities to realize that it has changed and how to utilize the new form of it [13]. Capacities that are presented to the client are normally called Application Programmable Interface (API). In the event that there is a change to the API, the client needs to recompile his code as well. This is not the situation in interpretative dialects like Smalltalk or Java, yet for assembled dialects, for example, C/C++. In an item situated world, an interface is a situated of the open routines characterized for an article. Typically the article can be controlled just through its interface. In C++ the client needs to recompile the code just at the point when an interface, eluded from the code, is changed. There is additionally a downside that the client of the class must utilize the same programming dialect all through the entirety advancement [14]. Dividing the interface from the execution is a approach to evade this tight coupling. This sort of partition is made with parallel interfaces as done in CORBA and COM, the part models portrayed in the following area. Parallel interfaces are characterized in an interface definition language (IDL) and an IDL compiler, which creates stubs and intermediaries, makes the applications area transparent.

1.4.2 Component Modals

The component characterizes the standards structures and standard interfaces between the parts [15]. They make it conceivable to parts to being conveyed and to impart. The correspondence can be built between segments on the same hub (PC) or between diverse hubs. For the later we are talkies about segment conveyance. Segment models are the most essential venture to lift the segment programming off the ground. In the event that parts are created free of one another then it is profoundly far-fetched that parts grew under such conditions have the capacity to chip in helpfully [16]. The essential objective of segment innovation, autonomous sending and get together of parts is not attained to a segment model backings parts by driving them to fit in with specific benchmarks and permits occasions of these segments to participate with different parts in this model. The three noteworthy segment models are utilized today with victory. These three are COM, JavaBeans, and CORBA what's more, every one of them have diverse levels of administration for the application engineer.

1.4.3 Quality Characteristics of Components

As much work is yet to be finished segment based programming improvement, QA advancement for componentbased programming improvement needs to address the two securely attached parts: 1) How to guarantee nature of a part? 2) How to guarantee nature of programming frameworks taking into account parts? To answer the inquiries, models ought to be elevated to characterize the general quality control of parts and frameworks; measurements ought to be found to gauge the size, multifaceted nature, reusability and dependability of parts and frameworks; and devices ought to be chosen to test the current parts and frameworks [17]. To assess a part, we must decide how to ensure the nature of the segment. The quality attributes of segments are the establishment to ensure the nature of the parts, and hence the establishment to ensure the nature of the entirety part based programming frameworks. Here we propose a rundown of suggested qualities for the nature of parts:

- Functionality
- Interface

- Usability
- Testability
- Maintainability
- Reliability

1.4.4 Examples of Component Based Approach

To begin with, let us give an design of straightforward stereo frameworks which contains of woofer, sub-woofer, sound box and so forth in the event that somebody needs to assemble the stereo frameworks by collecting the off the rack parts like sound box and so forth then he may be encountering some principalthan the individuals who construct the framework from the essentialcourse. But now all the straightforward and muddled frameworks are assembled utilizing the segment approach where a few parts are now grown by some engineer and they are put away in the library for their re-utilization. The primary idea is that those off the rack parts require not be changed for their individual purposes. In the event that we need alter a percentage of the parts to fit as per their materialness, we need to remake it and store them in the library. Case of some unpredictable framework, where every segments itself can be seen as a framework, is Naval Combat framework. The framework comprises of some radar for recognition, Helicopter, submarine, Rocket Launcher, some military aircraft and so on. Every segment here is some enormous frameworks and affiliation is additionally extremely convoluted in such cases.

1.5 Test Cases

A test case, in software engineering, is a situated of conditions under which an analyzer will figure out if an application, programming framework or one of its highlights is functioning as it was initially settled for it to do. The system for figuring out if a product project or framework has finished or fizzledsuch a test is known as a test prophet. In a few settings, a prophet could be a prerequisite or utilization case, while in others it could be a heuristic. It may take numerous experiments to confirm that a product system or framework is considered sufficiently investigated to be discharged. Experiments are frequently alluded to as test scripts, especially when composed - when they are generally gathered into test suites.

- **Formal Test Cases**

To completely test that all the necessities of an application are met, there must be no less than two experiments for every necessity: one positive test and one negative test. On the off chance that a necessity has sub-prerequisites, every sub-prerequisite must have no less than two experiments. Staying informed regarding the connection between the necessity and the test is often done utilizing a traceability network[27]. Composed experiments ought to incorporate a portrayal of the usefulness to be tried, and the arrangement needed to guarantee that the test can be directed. A formal composed experiment is portrayed by a known data and by a normal yield, which is worked out before the test is executed. The known info ought to test a precondition and the normal yield ought to test a postcondition.

- **Informal Test Cases**

For applications or frameworks without formal necessities, experiments can be composed taking into account the acknowledged ordinary operation of projects of a comparative class. In a few schools of testing, experiments are not composed at everything except the exercises and results are accounted for after the tests have been run.

In situation testing, speculative stories are utilized to help the analyzer thoroughly consider a complex issue or framework. These situations are generally not recorded in any point of interest. They can be as basic as a chart for a testing domain or they could be a depiction written in writing. The perfect situation test is a story that is persuading, solid, complex, and simple to assess. They are normally not quite the same as experiments in that experiments are single steps while situations cover various ventures of the key.

Comparison between algorithms

Previous work (prim'salgorithm)

```
1. Start

2. Set T as Empty

3. T1 [intracount] = 0; T2 [intercount] = 0;

4. If CK (Si) -> CK (Sj) then w = 0 // if states are interacting in same components.
```

```
5. If CK (Si) ->Cl (Sj) then w = 1 // if states are interacting in different components.
6. Select any state of any component, set S = {s}
7. T1 [intracount] = S(T) // Add Test case suite of state into T1.
8. Intracount ++;
9. Find lightest weight interactive state such that one endpoint is in S and other is in V\S.
10. If (w=0) then
11. T1 [intracount] = S(T) // Add Test case suite of state into T1.
12. Intracount ++;
13. Else
14. T2 [intracount] = S(T) // Add Test case suite of state into T2.
15. Intracount ++;
16. End if
17. If (V\S! = Ǿ) then step 9 // if no more state connected
18. T = T + T1 // Add T1 into T
19. T = T + T2 // Add T2 into T
20. Set T' = Reverse of T
21. Output T'
```

CHAPTER 2

LITERATURE REVIEW

ShwetaA.Joshi et al [2014] “ effective use of Prim’s algorithm for model based test case prioritization “ software testing is verification and validation for any software product so every time that’s not possible to perform every test case so, its important to decide test case prioritization. CBSD using the concept of prims’s algorithm and then observe the model based prioritization algorithm which makes use of CIG as the input for large size CBSD and then testing real time system as an example and to generate test cases [4].

Dan Hao et al [2014] “Adaptive Test-Case Prioritization Guided by Output Inspection” Experiment prioritization is to calendar the execution request of experiments in order to amplify some goal (e.g., uncovering blames early). The current experiment prioritization methodologies separate the methodology of experiment prioritization and the procedure of experiment execution by exhibiting the execution request of all experiments before software engineers begin running experiments. As the execution data of the altered project is not accessible for the current experiment prioritization approaches, these methodologies primarily depend on just the execution data of the past system before adjustment. To address this issue, we show a versatile experiment prioritization approach, which decides the execution request of experiments all the while amid the execution of experiments. Specifically, the versatile methodology chooses experiments in light of their shortcoming discovery ability, which is ascertained in view of the yield of chose experiments. When an experiment is chosen and runs, the deficiency location capacity of every unselected experiment is adjusted by yield of the most recent chose experiment. To assess the viability of the proposed versatile methodology, we directed a test think about on eight C projects and four Java programs. The trial results demonstrate that the versatile methodology is typically fundamentally better than the aggregate experiment prioritization methodology and aggressive to the extra experiment prioritization approach. Additionally, the versatile methodology is superior to the extra approach on a few subjects (e.g, supplant and plan)[5].

Zheng Yan et al [2014] “Autonomic Trust Management for a Component-Based Software System” Trust assumes an essential part in programming frameworks, particularly segment based frameworks in which parts or their situations fluctuate [20]. This paper presents an autonomic trust administration answer for a segment based programming framework. We propose a versatile trust control model to indicate, assess, create, and guarantee the trust connections among framework elements. This model concerns the quality properties of the substance and various trust control modes upheld by the framework. Specifically, its parameters can be adaptively balanced in view of runtime trust appraisal keeping in mind the end goal to reflect genuine framework setting and circumstance. Based on this model, we further add to various calculations that can be received by a trust administration system for autonomic administration of trust amid segment execution. We confirm the calculations' practicality through reenactments and show the viability and profits of our answer. We likewise talk about the issues for fruitful organization of our answer in a component software platform.

Xinyu Zhang et al [2014] “The Research of the Component-Based Software Engineering” Utilizing component-based software engineering (CBSE), applications are made out of reusable segments with decently characterized interfaces and conduct [] . Very dependable programming generation instruments and chipping in environment turns into the discriminating piece of the part based programming building; it’s not just one of the cutting edge key focuses for programming innovation advancement, additionally basic establishment for the improvement of programming businesses, having imperative useful importance and long haul key significance.

Ning, J.Q. [2013] “Component-based software engineering (CBSE)” Component-based software engineering or CBSE speaks to another advancement standard: amassing programming frameworks from parts. This exploration territory has raised a gigantic measure of hobbies both in the examination group and in the product business an uncommon marvel in the field of programming building. The paper talks about the innovation framework important to bolster CBSE. Specifically, the creators exhibit the outcomes delivered by the CBSE examination task directed at Andersen Consulting in the connection of how their methodology and instruments robotize a part based advancement standard [22].

Shete, N. et al [2013] “An empirical study of test cases in software testing” Software is a situated of directions executed by a PC which are intended to perform a specific undertaking [7] . Programming Development Life Cycle is utilized to add to the product. Programming Testing is the essential period of software development life cycle (SDLC). Programming testing is a piece of SDLC. Testing satisfies the client's prerequisite. Notwithstanding that testing procedure discovers and uproots the bugs of the product. Creating association tries to demonstrate that the product they creating is quality programming and procedure used to grow and to test programming are quality methods. To test any product, analyzer composes experiments in view of Software Requirement Specification (SRS). SRS contain all the useful and non-practical necessities of the product. Singular segment (Unit) prerequisite determinations are composed in point of interest. Test engineer and/ or Tester utilized SRS to compose experiments. Experiments are utilized to test the product altogether in manual testing. All little provisos of the product could be recognized by experiments. This paper concentrates on the noteworthiness of experiments and their part to test programming utilized as a part of IT commercial ventures. The specialist has reasoned that without experiments testing would not be conceivable.

Georgiana Macariu et al [2013] “Enabling Parallelism and Resource Sharing in Multi-core Component-based Systems” Complex ongoing implanted frameworks oblige ensures as to certification of their timing prerequisites. Such ensures can be inferred utilizing propelled outline and examination strategies. Numerous configuration arrangements address the multifaceted nature of these frameworks utilizing part based methods. In this paper we concentrate on asset partaking in part based frameworks with a few parts executing on a multi-center processor. We consider that the errands of every segment can be planned on any center with the likelihood of two errands having a place with the same part executing in parallel. We propose the Parallel Various leveled Resource Policy, a novel asset imparting approach for multi-center part based frameworks. We likewise build up a point by point reaction time based schedule ability examination for the single person parts and for the made framework, accepting that intraand between part asset imparting happens [11].

Guoyou Shi et al [26] "Object-oriented design and implementation of communication software based on TCP/IP and multithread" As a vital segment of boat route and checking framework, correspondence software integrates the capacities of information procurement, correspondence and presentation into an entire, and is the key connections of canny sea transportation. In view of TCP/IP (Transmission Control Protocol/ Internet Protocol) convention, correspondence strategy in the middle of server and customers is presented with grouping outline and movement chart of article situated configuration portrayals. Server side of the product uses multithread innovation to accomplish system correspondence with exchange focal point of the correspondences administrator and customer, and completions the capacities of information obtaining and parsing. In the meantime, customer side of the product builds associations with server and parses getting ongoing paired information to show in information matrix. During the time spent establishment and troubleshooting, issues of customer breakdown and offbeat information in the middle of server and customers and less information in the customers are illuminated [12].

SajjadMahmood et al [2012] “A Degree Centrality-Based Approach to Prioritize Interactions of Component-Based Systems” Component Based System (CBS) advancement centers on coordinating programming parts, regularly grew by diverse gatherings, to assemble an application. Part reconciliation plays a discriminating part in general CBS advancement. As the quantity of communications increments in a CBS, there is a requirement for a superior administration of the part mix process. In a huge also, complex CBS, a framework integrator is keen on breaking down distinctive communications to recognize which collaborations are more critical than others. In this paper, we introduce a communication prioritization process which utilizes the idea of 'degree centrality' to investigate part connections and along these lines organize the utilization case situations of the framework taking into account the centrality of singular associations [6].

Saddek Bensalem et al [2012] “Synthesizing Distributed Scheduling Implementation for Probabilistic Component-based Systems” Creating simultaneous frameworks ordinarily include an extensive troubleshooting period, because of the gigantic number of conceivable many-sided practices [7]. Utilizing abnormal state portrayal formalism at the halfway level between the determinations what's more, the code can tremendously help diminish the expense of this procedure, what's more, the presence of remaining bugs in the conveyed code. Check is considerably more moderate at this level. A programmed interpretation of segment based frameworks into running code, which protects the worldly properties of the outline, makes a difference incorporating solid code. We give her a change from an abnormal state portrayal formalism of segment based framework with probabilistic decisions into running code. This change includes synchronization utilizing imparted variables. This synchronization is segment based instead of interactionbased, due to the need to ensure a stable perspective for a segment that performs probabilistic decision. We give the synchronization calculation and give an account of the usage.

Benharref, A et al [2011] "A New Approach for Quality Enforcement in Communities of Web Services"These days, Web Services are considered as accepted and pulling in disseminated methodology of use/administrations incorporation over the Internet. Web Services can likewise work inside groups to enhance their perceivability and piece of the overall industry. In a group, Web Services normally offer contending and/or supplementing administrations. In this paper, we expand the group approach by characterizing a particular reason group to screen Web Services working in any Web Services group. This observing group comprises of a set of Web Services equipped for watching other Web Services. Customers, suppliers, and additionally directors of groups can make utilization of the observing group to check if a Web Service is working obviously. This paper characterizes the general structural planning of the checking group, the plan of action behind, diverse tenets and terms to be regarded by its parts, administrations it offers to its different classes of clients. The paper likewise introduces guaranteeing test results utilizing the checking group [2].

Yukyong Kim et al [2011] "Adaptable Web Services Modeling Using Variability Analysis" Web administrations have as of late come into concentrate as they find themselves able to make programming frameworks adaptable, reusable, and financially savvy. In dispersed Web administrations situations, Web administrations ought to be extremely versatile on the grounds that the potential number of administrations can be amazingly vast and target shoppers and administration suppliers need to face different circumstances. To advance the flexibility of administrations, the variability on Web administrations ought to judiciously be considered and displayed. In this paper, we display a strategy for displaying exceptionally versatile Web benefits by breaking down the variability regarding engineering and behavioral peculiarities. We characterize displaying parts of Web administrations to concentrate the variability and variety focuses. We concentrate on the displaying of variability of Web administrations from variety purposes of structural peculiarities and behavioral gimmicks. In this work, we attempt to portray how versatile Web administrations are produced with a suitable portrayal of the variability as a system for altering frameworks focused around Web administrations [14].

Al-Masri, E et al [2011] "Discovering the best web service: A neural network-based solution" Separating between Web benefits that have comparative functionalities is turning into a significant test into the disclosure of Web administrations. In this paper we propose a system for empowering the productive disclosure of Web administrations utilizing artificial neural systems (ANN) best known for their speculation capacities. The center of this system is applying a novel neural system model to Web administrations to focus suitable Web administrations focused around the thought of the quality of Web service (QWS). The principle idea of QWS is to evaluate a Web administration's conduct and capacity to convey the asked for usefulness. Through the total of QWS for Web benefits, the neural system is equipped for distinguishing those administrations that fit in with a mixed bag of class items. The general execution of the proposed strategy demonstrates a 95% achievement rate for finding Web administrations of investment. To test the power and viability of the neural system calculation, a percentage of the QWS features were rejected from the preparation set and results demonstrated a noteworthy effect in the general execution of the framework. Consequently, finding Web benefits through a wide choice of value traits can significantly be impacted with the determination of QWS peculiarities used to give a general evaluation of Web administrations [1].

Pour, G. [2011] "Integrating component-based and reuse-driven software engineering business into software and information engineering curriculum" Component-based enterprise software engineering (CBESE) is a quickly developing pattern in the product building territory. In the CBESE approach, programming frameworks are no more manufactured without any preparation. Rather, reusable programming segments, manufactured by in-house designers or business sellers, are utilized as the building squares of new segment based endeavor programming frameworks. The move from conventional programming building to segment based venture programming designing is regularly blocked or thwarted by a mixture of building, procedure related, business-situated, framework, authoritative and administration issues. Determining those issues obliges an efficient way to building a part based and reuse-driven programming designing business. That is the reason programming specialists need to procure another arrangement of aptitudes. This has presented the requirement for another course succession that coordinates part based endeavor programming building into the product and data designing educational module [12]. Another real course in this arrangement is centered on part based and reuse-driven programming designing organizations. In this paper, we impart our experience of growing such another course. The course is expected to give a strong establishment to coordinating of exploration into training in the territory of part based undertaking programming designing. In this paper, we introduce the course's association, its parts, and our feasible arrangements for the course.

Vu Tran et al [2011] "A procurement-centric model for engineering component-based software systems" The late surge of enthusiasm inside the product business in building more perplexing, dependable, and viable, yet practical programming arrangements through the mix of financially accessible programming items, has brought about a huge move far from the improvement driven toward a more obtainment driven way to vast scale framework development. This methodology, known as component-based software engineering (CBSE), concentrates on the distinguishing proof, determination, assessment, acquirement, incorporation, and advancement of reusable programming segments to give complex coordinated arrangements at least improvement cost. These parts are frequently commercial off-the-shelf (COTS) items [13]. Contrasted with customary improvement driven programming building methodologies, CBSE guarantees a more productive and powerful way to the conveyance of programming answers for the business. In any case, thinking little of the specialized dangers connected with the choice, assessment, and combination of programming parts has regularly brought about timetable postpone and expanded improvement/support cost in coordinated framework advancement ventures. The paper presents an acquirement driven model for segment based coordinated framework usage. The model, called COTS-based Integrated Systems Development, portrays an orderly way to the determination, assessment and coordination of reusable programming segments. In particular, the model distinguishes key designing stages and their sub-stages that are frequently disregarded, or just implied, in present advancement driven models. At last, the paper portrays a coordination of improvement driven and acquirement driven models used to backing the advancement of incorporated programming frameworks at the Mitsubishi Consumer Electronics Engineering Center.

Thomas, J.P et al[2009] "Modeling of Web services flow" Administrations, for example, programmed buying, programmed redesigning of costs, or getting most recent data and so forth, can be given on the Internet utilizing Web administrations innovation. A customer can get to these administrations utilizing the Internet. Web administrations framework incorporates a few principles, for example, straightforward article access convention (SOAP), Web administrations portrayal dialect (WSDL) and all inclusive depiction, revelation and incorporation (UDDI). In this paper we speak to appropriated Web benefits by demonstrating the stream of messages and techniques in a Web administration exchange. Such a model helps the Web administrations creator to guarantee the accuracy of Web streams as far as gridlock and right end of the Web administrations exchange. WSDL and techniques are displayed utilizing Petri Nets. A product device is actualized for separating the model from the WSDL portrayal of the Web administrations stream.

Shashank, S.P et al [2009] "A systematic literature survey of integration testing in component-based software engineering" Component-based software engineering (CBSE) has developed as a methodology that offers quick improvement of framework utilizing less assets and exertion. The center thought of reuse and chopping down the improvement expenses can be accomplished if the segments offer dependable administrations. Therefore, joining parts and testing turn into an imperative stage in CBSE. Mix of segments is an imperative movement. This includes understanding correspondence and coordination between the parts. Engineers are not furnished with sufficient data on these parts. As an aftereffect of this, comprehension information stream while coordinating these segments is a test. Part based programming encourages improvement of complex frameworks by permitting combination of reusable segments. Testing parts is a testing zone of exploration. There have been inconveniences coordinating the segments. This thus influences the quality and dependability of the product. Our examination goes for discovering the current combination testing and difficulties in CBSE. The precise writing study is in view of 49 articles gathered from various stage choice methodologies. These articles have been distributed inside the time compass of 1995-2009.[8]

Singh, S. et al [2009] “Efficient Component Based Software Engineering using the TCEM Methodology and the TCET Tool” As of now there are no exhaustive devices that can be utilized to outfit the maximum capacity of component based software engineering CBSE). In this paper, we portray a novel segment based programming advancement system; called the "total component engineering methodology" (TCEM) that we have considered, formed and tried that can be effectively and successfully connected to each period of the CBSE process. We likewise portray a novel and far reaching visual instrument called the "total component engineering tool" (TCET) which uses upgraded visual documentations and peculiarities to proficiently bolster our new advancement procedure. This novel instrument and system supplement one another and can be utilized to create excellent part based programming that is profoundly reasonable, versatile, viable and reusable[9].

CHAPTER 3
PRESENTWORK

3.1 PROBLEM FORMULATION

In the test case prioritization various techniques has been used for development of test cases so that in minimum computation of time maximum errors can be extracted from the software product. To check prioritization of different test cases generated, various algorithms have been purposed. In this work the ADDTL_TCP approach is used for the purpose of test case prioritization. The priority table generated by this approach is used for optimizing these cases by implementing genetic algorithm. Genetic algorithm is an approach which generates an output from two inputs from parent class. This develops cross-over and mutation off springs according to cross-over and mutation probability. This approach provides best test case priority which is dependent on functionality and components interrelations.

3.2 OBJECTIVES

a)To study various approaches for test case generation in software testing.

b)To implement prioritization algorithm to set test case priority on the basis of usage functionality of component.

c) To optimize test case prioritization on the basis genetic algorithm.

d) Genetic algorithm develops cross-over and mutation off spring.

d) To execute test cases and provide results in less time

3.3 RESEARCHMETHODOLOGY

In this work various techniques has been used for the creation of test cases, after generation of test cases these test cases has to be implemented on the different types of components present in software system. To detect early bugs in the software these cases has to be implemented on the basis of priority.

To check prioritization of different test cases generated, various algorithms have been purposed. In this work the ADDTL_TCP approach is used for the purpose of test case prioritization. The priority table generated by this approach is used for optimizing these cases by implementing genetic algorithm. Genetic algorithm is an approach which generates an output from two inputs from parent class. This develops cross-over and mutation off springs according to cross-over and mutation probability. This approach provides best test case priority which is dependent on functionality and components interrelations.

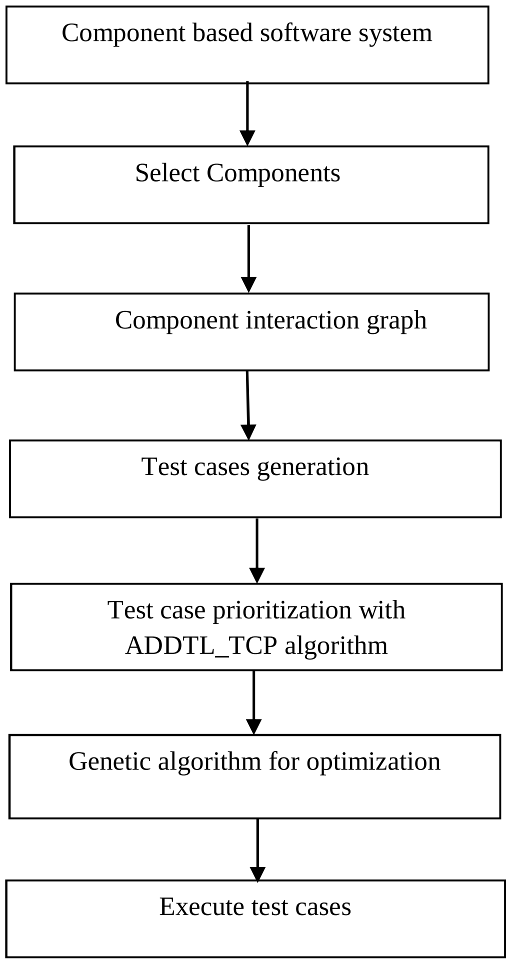
Algorithm

```
Input Test Suit, Test cases, Components (C), Web service (W)

Output Prioritize test cases

Genetic Algorithm Initialization

1. Population Size = number of test suites

For i =1 to population

Initialize all the test suits T(i)

Select all the test suits on the basis of selection

2. GA Crossover

For i = 1 to   population

(Parent1, Parent2) = Random (Test suits)

If Test suit = w
```

```
Generate child (c)w
Else
Test case for component
End if
(newchild1, newchild2) GA crossover (parent1, parent2)
Add child to generation of test suits
End crossover
3. GA Mutation
For i = 1 to population
If new child = parents
New child = GA Mutation (parent)
End if
Add new child T(i) to T
End Mutation
4. for i = 1 to generation
Compute fitness
Fitness = if  T(i) <  T(i+1) coverage area of component
Do while
Replace T(i+1) do positive of T(i)
End do while
End fitness computation
```

5. select test suit that having maximum fitness

Flow chart


Component based software system
Select Components
Component interaction graph
Test cases generation
Test case prioritization with ADDTL_TCP algorithm
Genetic algorithm for optimization
Execute test cases

CHAPTER4

RESULTS AND DISCUSSIONS

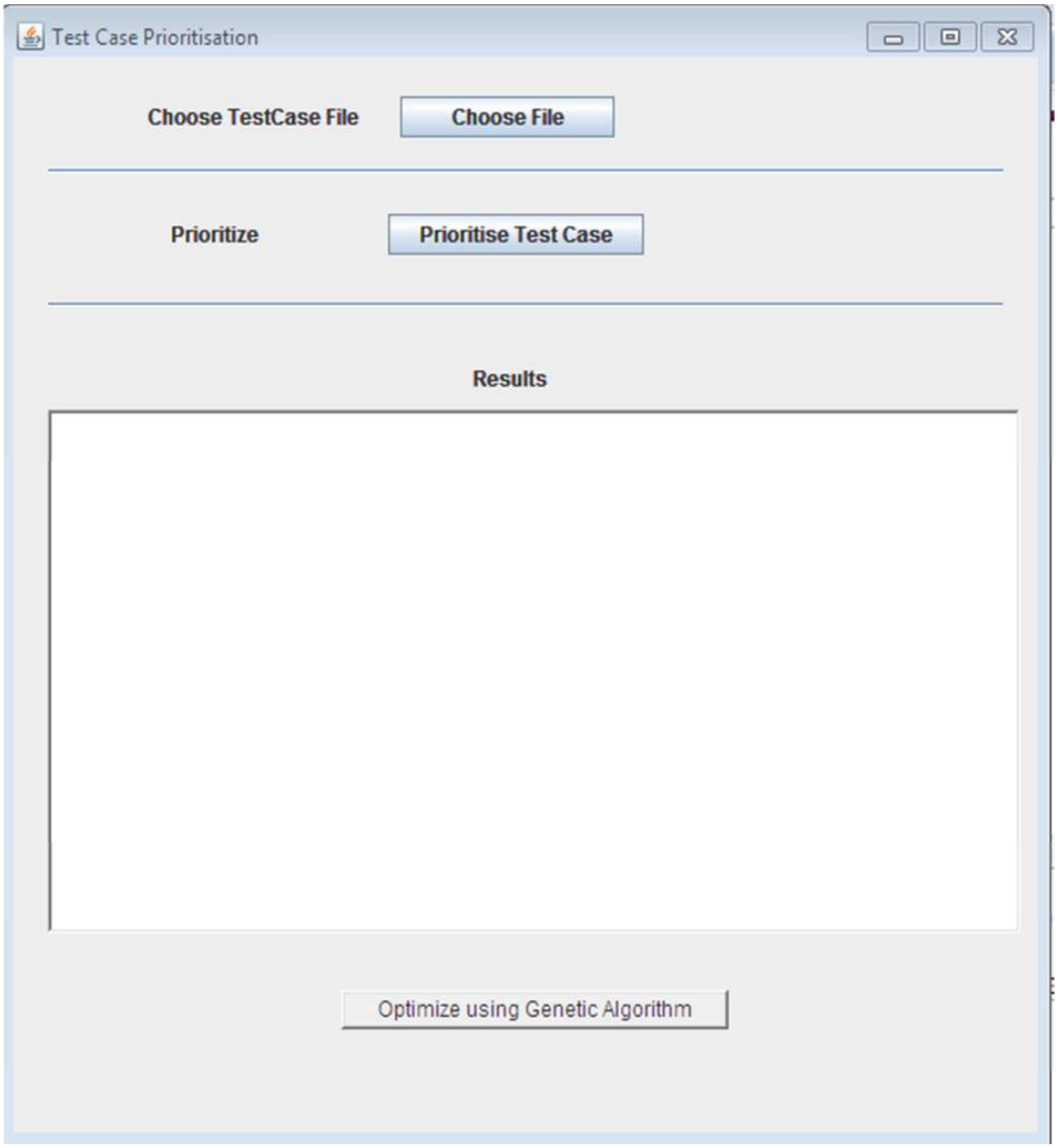
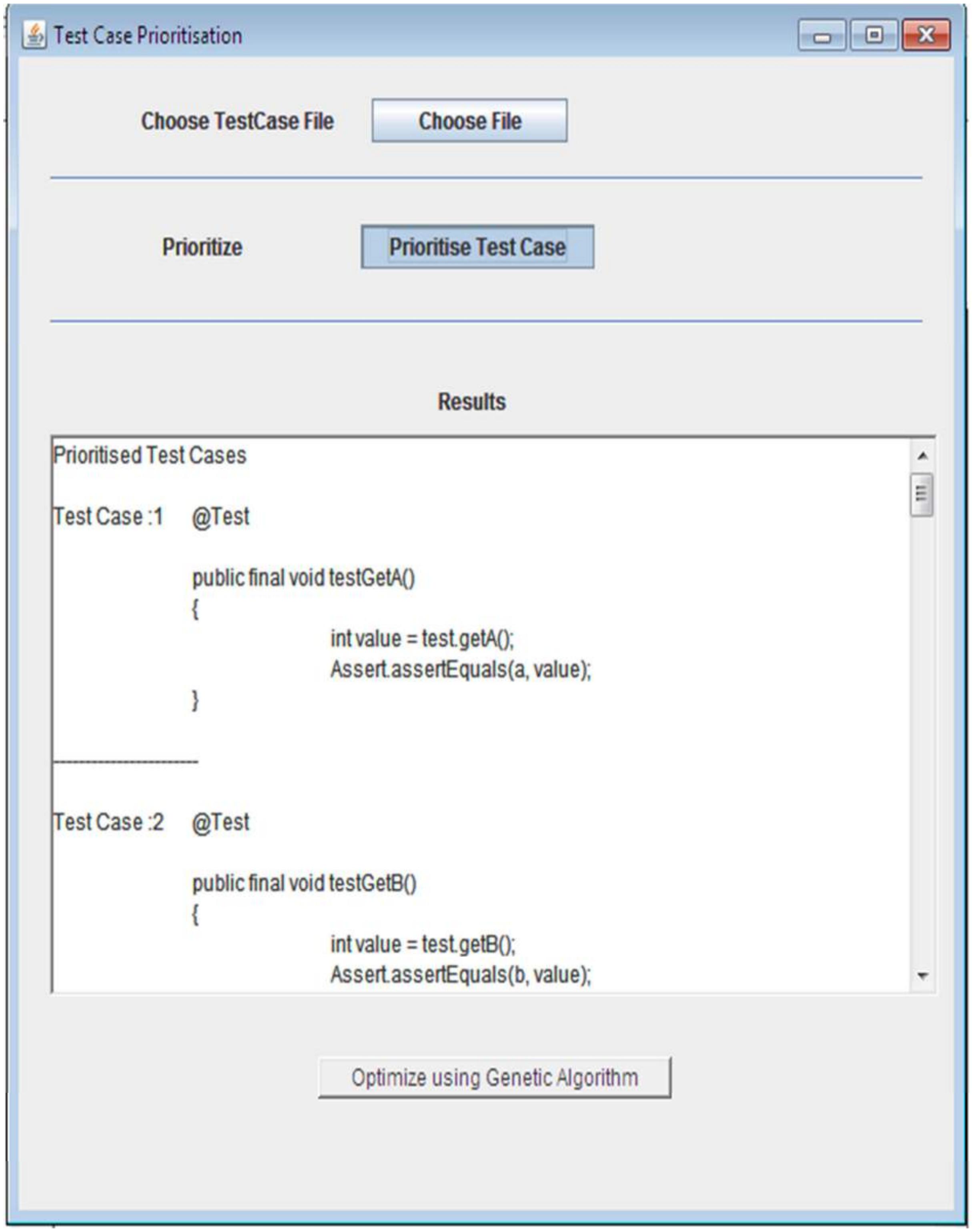
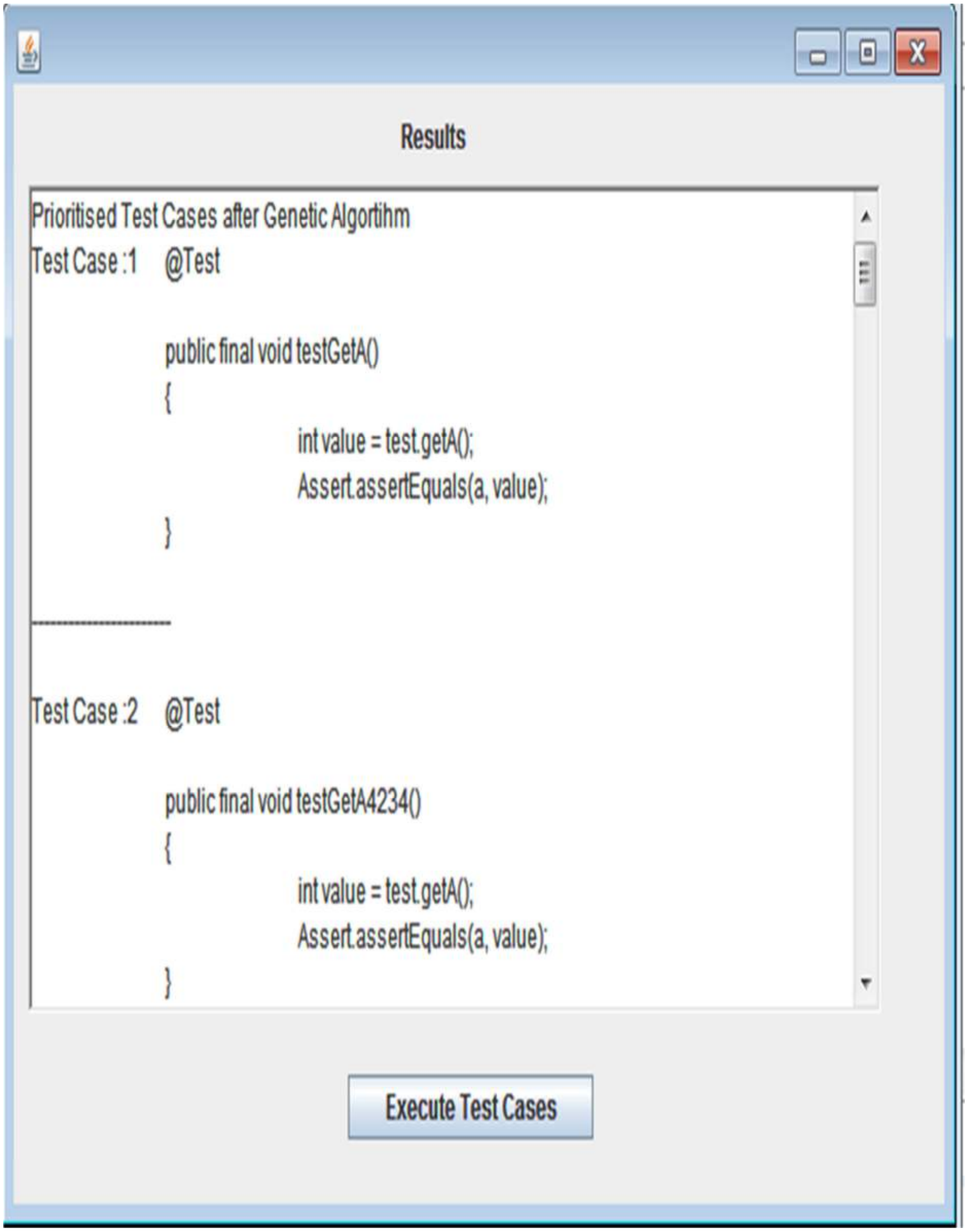


Fig 4.1 Test Cases for Prioritization



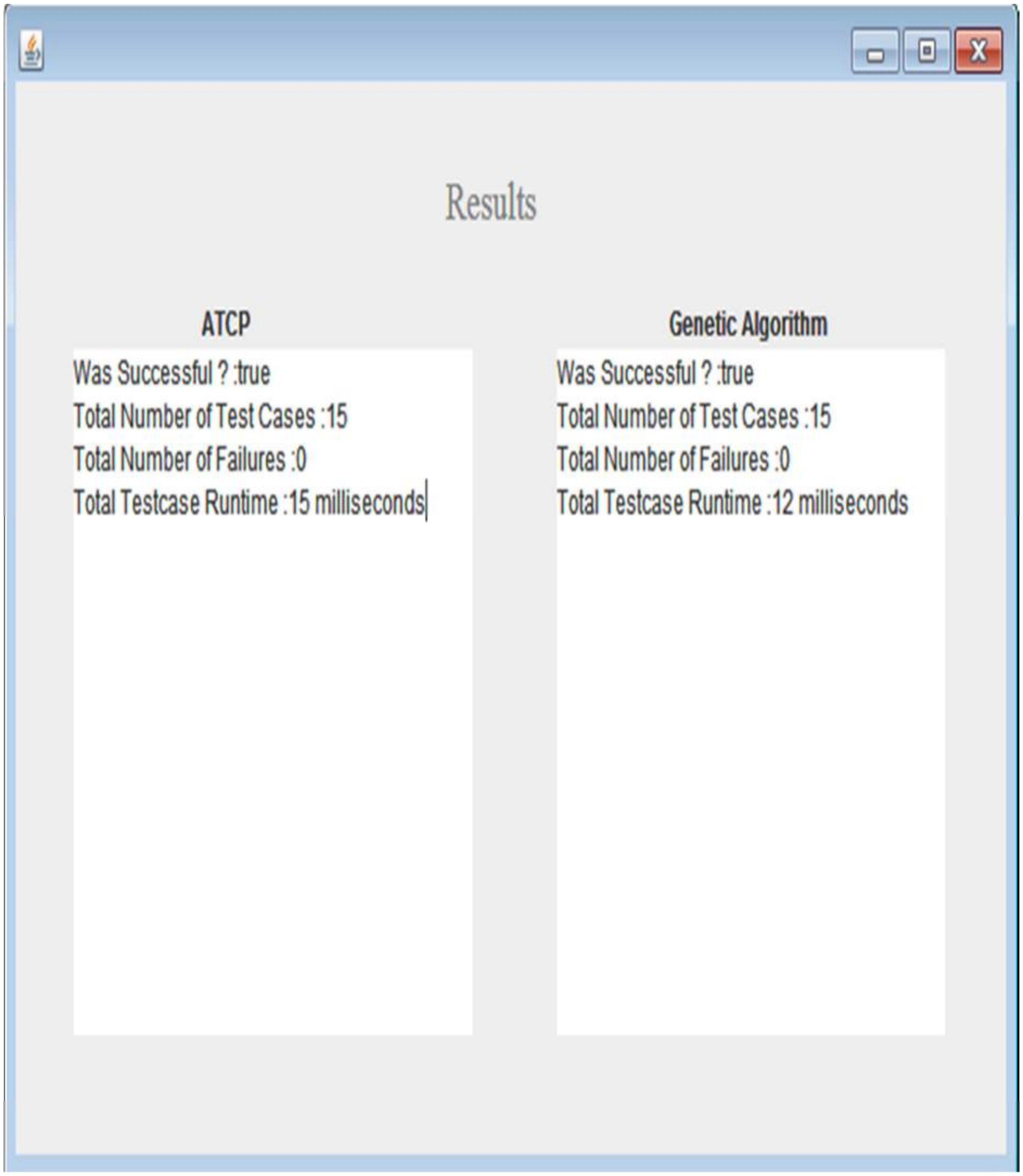
```
Prioritised Test Cases

Test Case :1    @Test

                public final void testGetA()
                {
                                int value = test.getA();
                                Assert.assertEquals(a, value);
                }

-----------------------

Test Case :2    @Test

                public final void testGetB()
                {
                                int value = test.getB();
                                Assert.assertEquals(b, value);
```

Fig 4.2 Apply Addtl_Tcp Algorithm



```
Prioritised Test Cases after Genetic Algortihm
Test Case :1    @Test

                public final void testGetA()
                {
                                int value = test.getA();
                                Assert.assertEquals(a, value);
                }

------------------------

Test Case :2    @Test

                public final void testGetA4234()
                {
                                int value = test.getA();
                                Assert.assertEquals(a, value);
                }
```

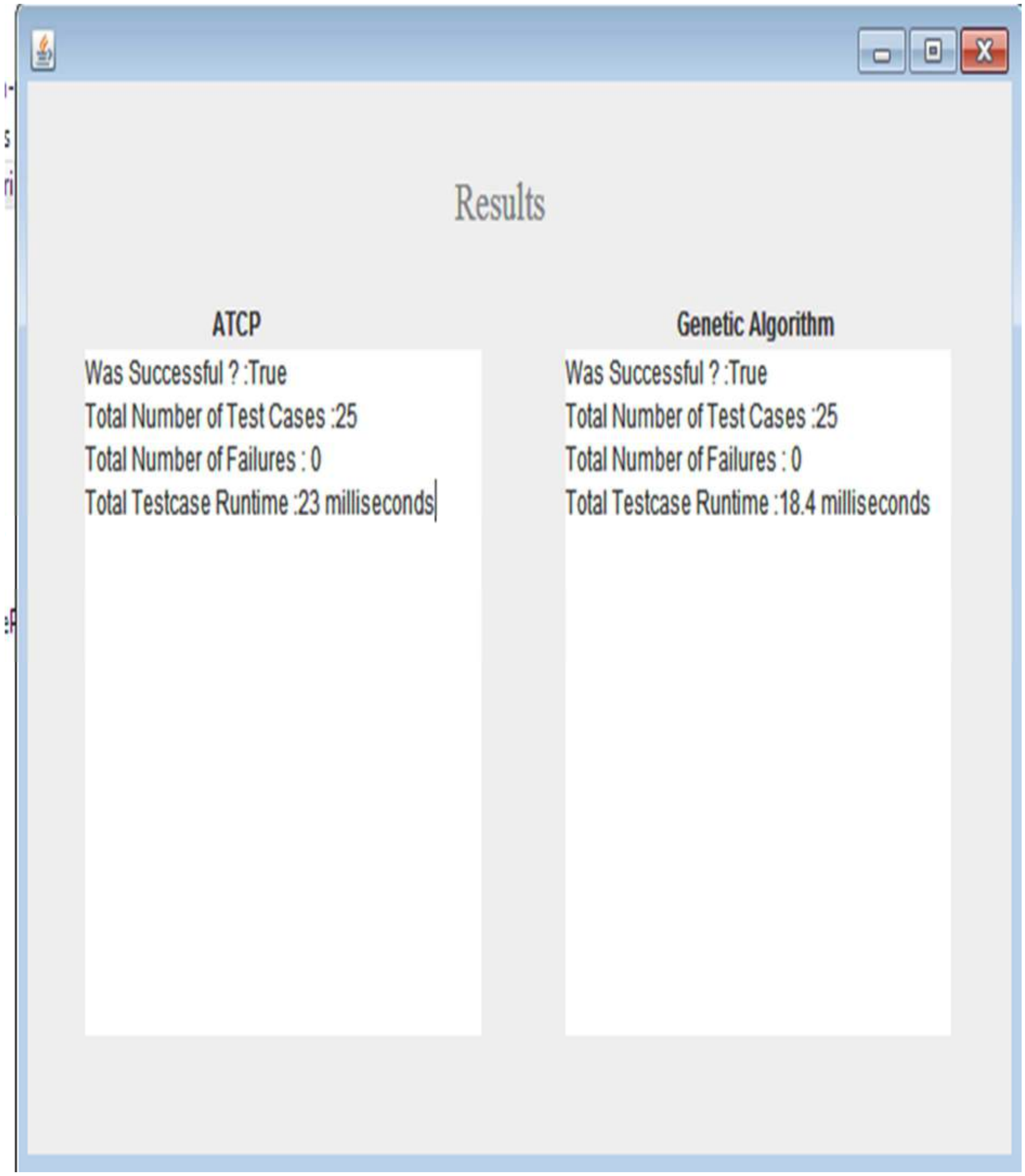
Fig 4.3 Apply Genetic Algorithm



```
Was Successful ? :true
Total Number of Test Cases :15
Total Number of Failures :0
Total Testcase Runtime :15 milliseconds
```

```
Was Successful ? :true
Total Number of Test Cases :15
Total Number of Failures :0
Total Testcase Runtime :12 milliseconds
```

Figure 4.4 Result Of Addtl_Tcp And Genetic Algorithm



Results

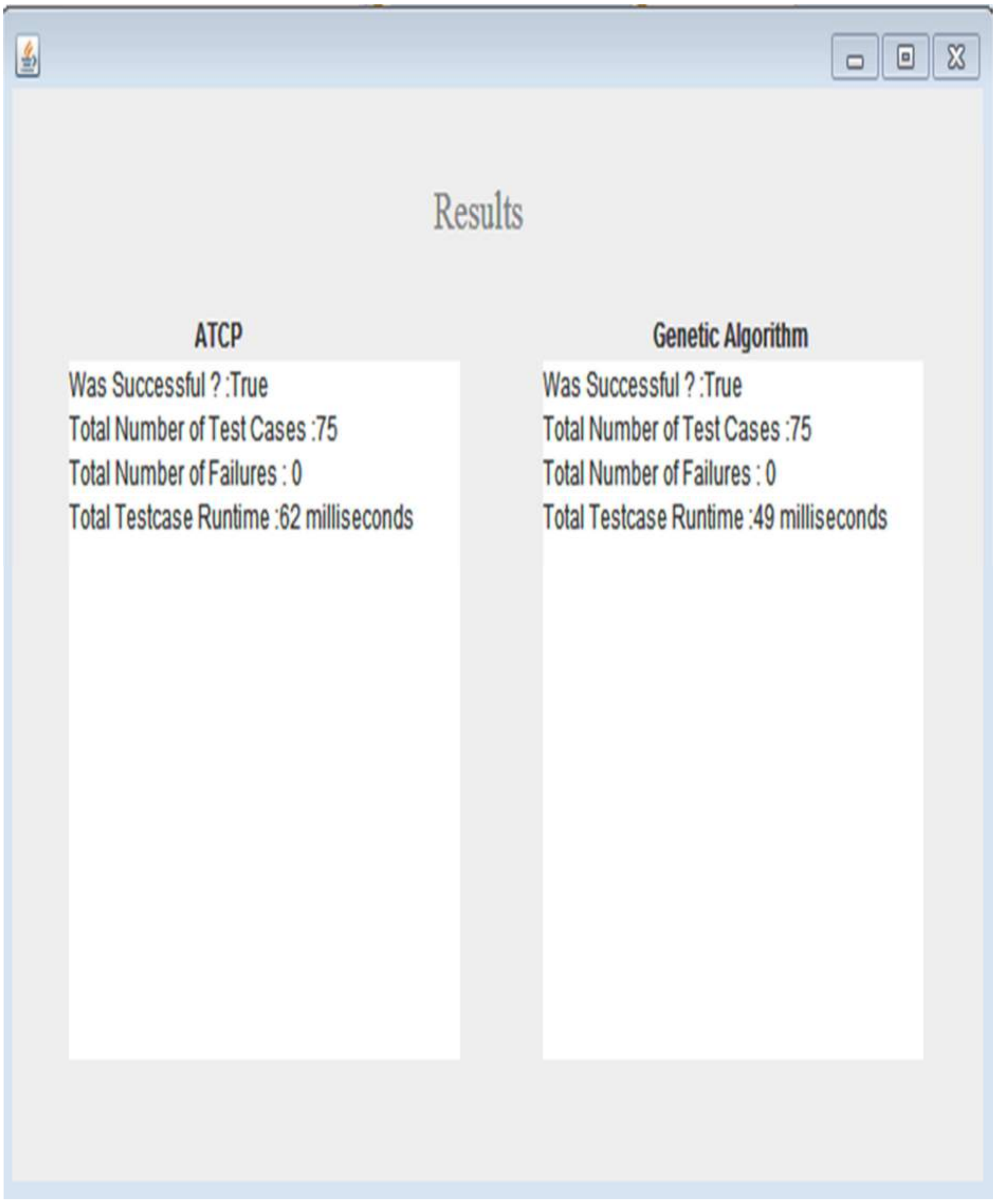
ATCP

```
Was Successful ? :True
Total Number of Test Cases :25
Total Number of Failures : 0
Total Testcase Runtime :23 milliseconds
```

Genetic Algorithm

```
Was Successful ? :True
Total Number of Test Cases :25
Total Number of Failures : 0
Total Testcase Runtime :18.4 milliseconds
```

Fig 4.5 Result Of Algorithms



Results

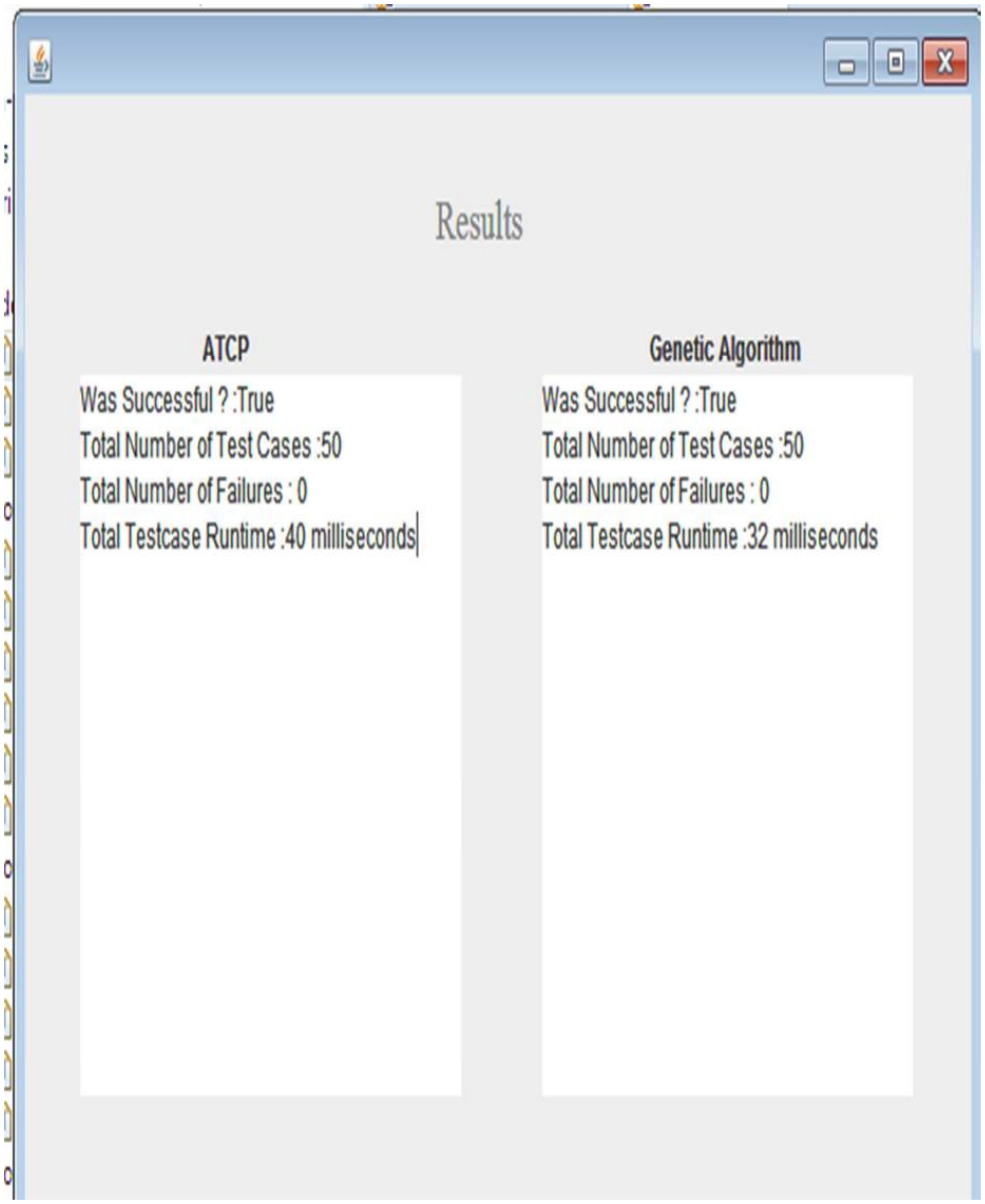
ATCP

```
Was Successful ? :True
Total Number of Test Cases :75
Total Number of Failures : 0
Total Testcase Runtime :62 milliseconds
```

Genetic Algorithm

```
Was Successful ? :True
Total Number of Test Cases :75
Total Number of Failures : 0
Total Testcase Runtime :49 milliseconds
```

FIG 4.6 ResultFor ADDTL_TCP And Genetic Algorithm

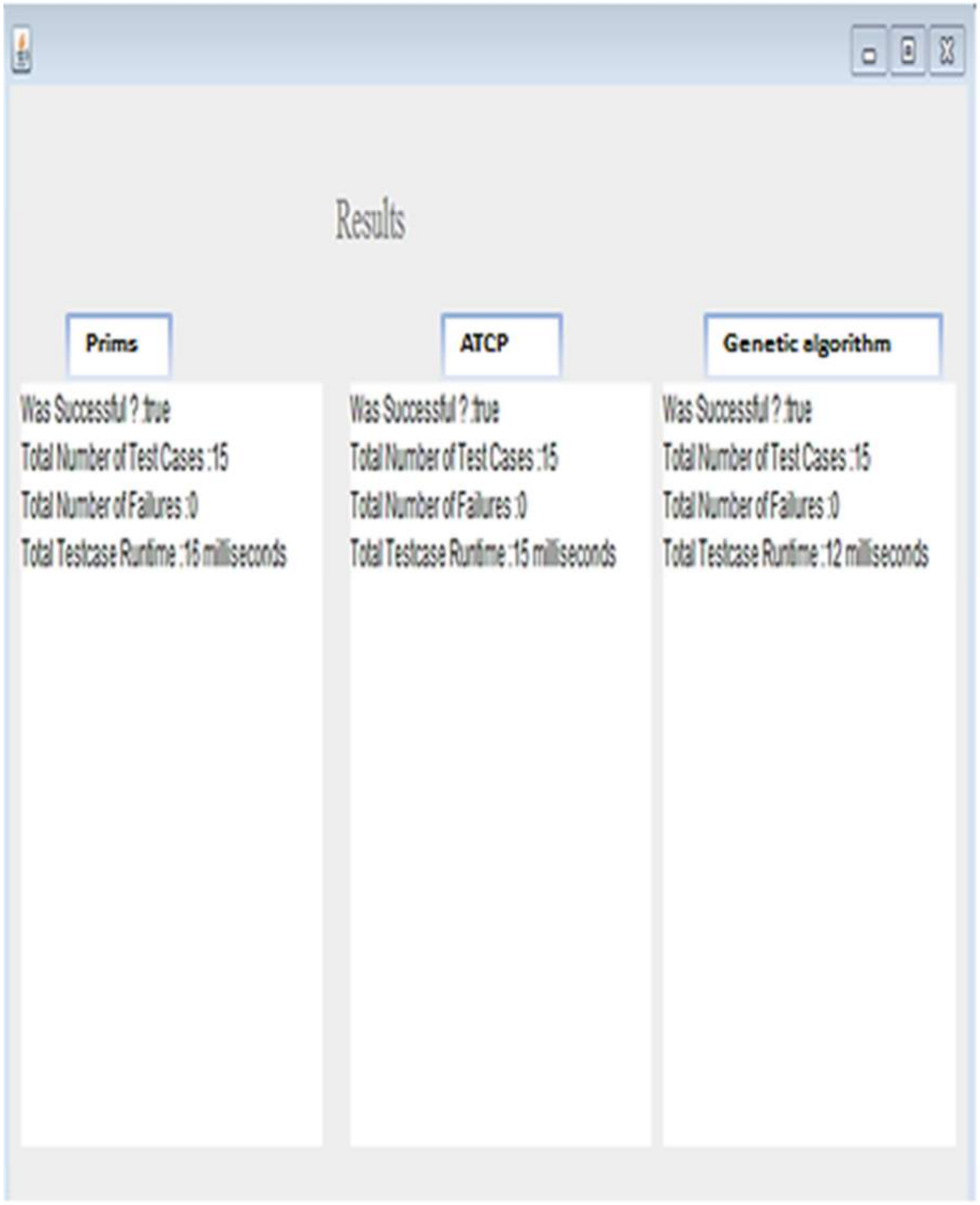


```
Was Successful ? :True
Total Number of Test Cases :50
Total Number of Failures : 0
Total Testcase Runtime :40 milliseconds
```

```
Was Successful ? :True
Total Number of Test Cases :50
Total Number of Failures : 0
Total Testcase Runtime :32 milliseconds
```

Fig 4.7Addtl_TcpAnd Genetic Algorithm Result

COMPARISON

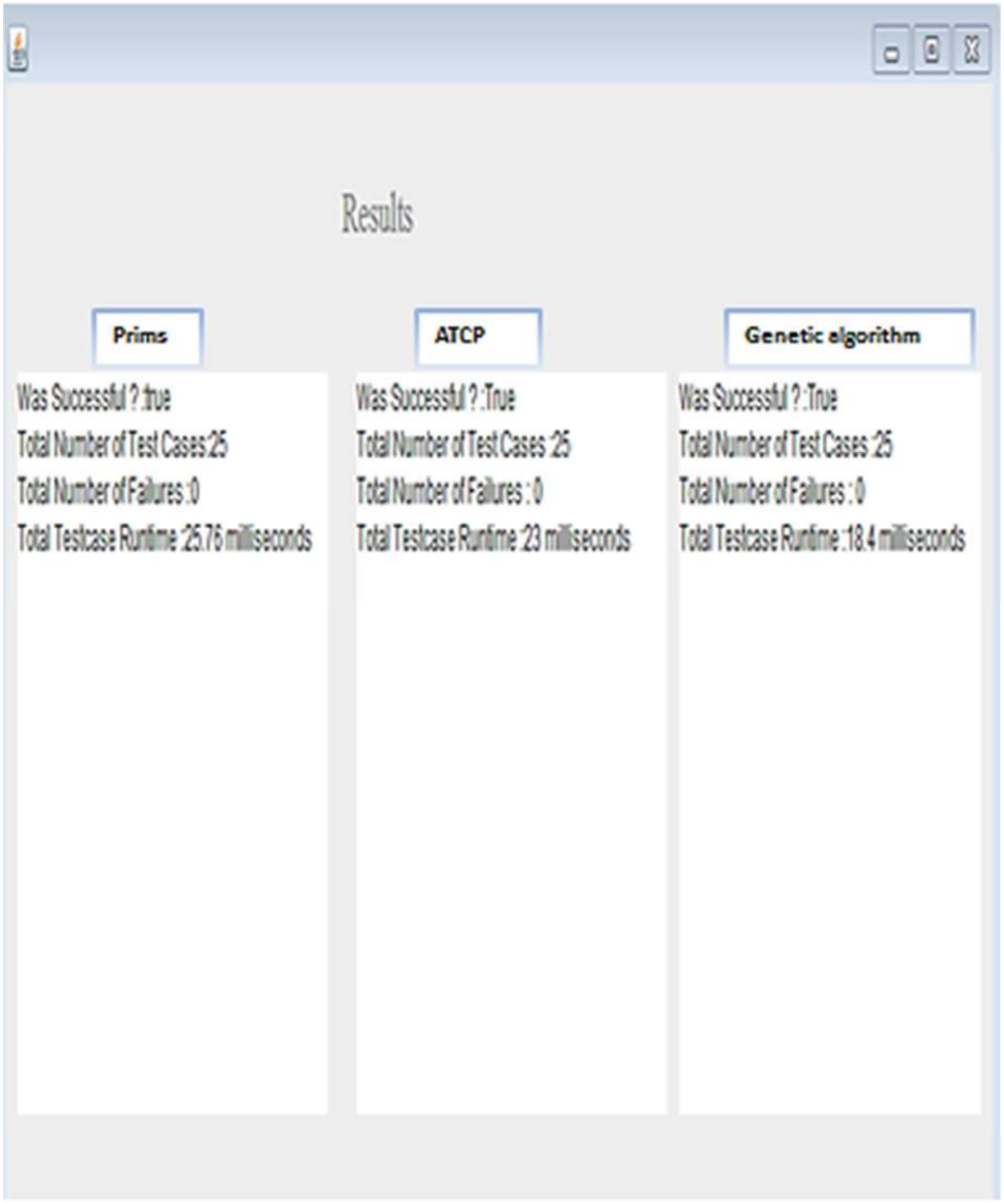


Was Successful ? :true
Total Number of Test Cases :15
Total Number of Failures :0
Total Testcase Runtime :16 milliseconds

Was Successful ? :true
Total Number of Test Cases :15
Total Number of Failures :0
Total Testcase Runtime :15 milliseconds

Was Successful ? :true
Total Number of Test Cases :15
Total Number of Failures :0
Total Testcase Runtime :12 milliseconds

Figure 4.8 Comparison With Previous Algorithm (Prims)



Prims

Was Successful ? :true
Total Number of Test Cases:25
Total Number of Failures :0
Total Testcase Runtime :25.76 milliseconds

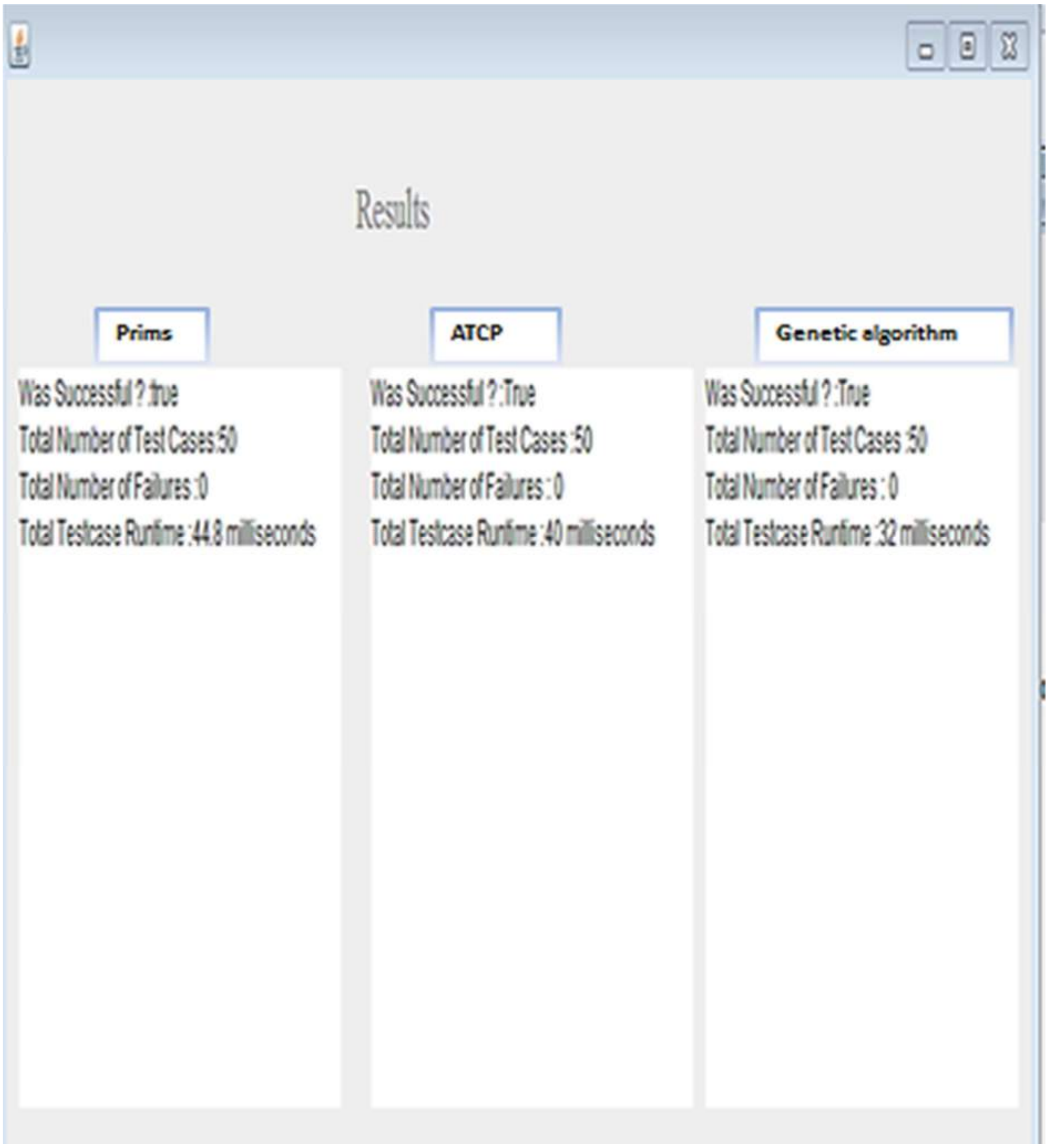
ATCP

Was Successful ? :True
Total Number of Test Cases :25
Total Number of Failures : 0
Total Testcase Runtime :23 milliseconds

Genetic algorithm

Was Successful ? :True
Total Number of Test Cases :25
Total Number of Failures : 0
Total Testcase Runtime :18.4 milliseconds

Figure 4.9ComparisionWith Prims Algorithm

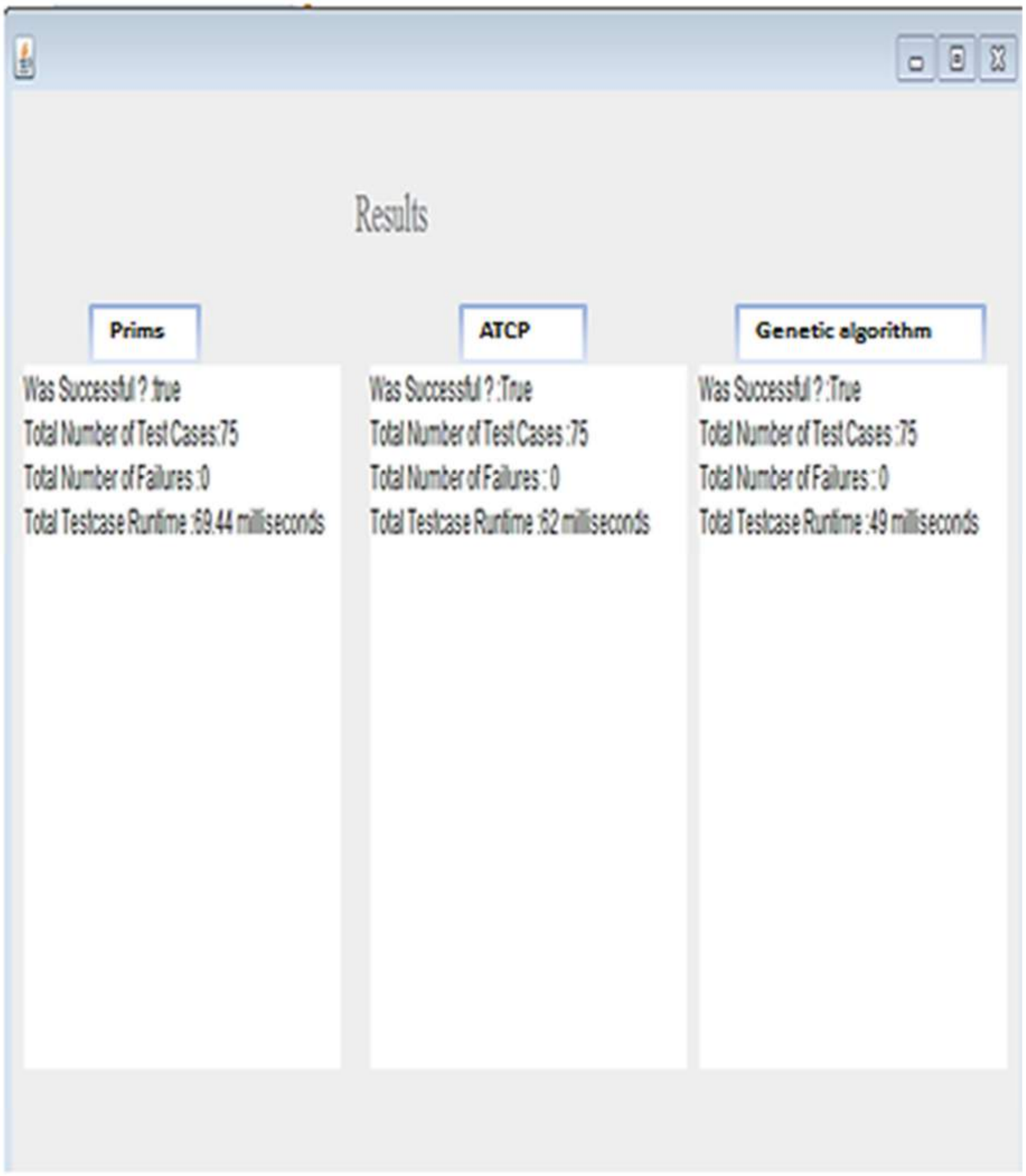


```
Was Successful ? :true
Total Number of Test Cases:50
Total Number of Failures :0
Total Testcase Runtime :44.8 milliseconds
```

```
Was Successful ? :True
Total Number of Test Cases :50
Total Number of Failures : 0
Total Testcase Runtime :40 milliseconds
```

```
Was Successful ? :True
Total Number of Test Cases :50
Total Number of Failures : 0
Total Testcase Runtime :32 milliseconds
```

Figure 4.10Comparison Of Algorithm Using Prims, Addtl_Tcp And Genetic


Results

Prims

```
Was Successful ? :true
Total Number of Test Cases:75
Total Number of Failures :0
Total Testcase Runtime :69.44 milliseconds
```

ATCP

```
Was Successful ? :True
Total Number of Test Cases :75
Total Number of Failures : 0
Total Testcase Runtime :62 milliseconds
```

Genetic algorithm

```
Was Successful ? :True
Total Number of Test Cases :75
Total Number of Failures : 0
Total Testcase Runtime :49 milliseconds
```

Fig4.11Comparison Between Algorithms

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

Software system has been used for various applications. This application has to perform various tasks on the basis of their computation. Software system has been designed by using various components. These components have been integrated for development of software system.

These components have to be tested by using various test cases. Test suit have been designed to perform testing. Test suit has to be executed by using different prioritization approach. In the purposed work web services has been developed to design software system. Test suit to test the developed system has been prioritization by using adaptive test cases prioritization approaches and Genetic algorithm. Genetic algorithm optimizes the test sequence that has to be executed o9n the basis of coverage area & belong to which components. The purposed work provides better results than that of previous approaches.

In the future reference this approach can be used in real world scenario. In the future various other artificial intelligence approaches can be used for optimize the test case prioritization sequence.

CHAPTER 6
REFERENCES

[1] Al-Masri, E, Mahmoud, Qusay H. "Discovering the best web service: A neural network-based solution" IEEE International Conference onSystems, Man and Cybernetics

[2] Benharref, A, Bouktif, S. Bentahar, J. "A New Approach for Quality Enforcement in Communities of Web Services" IEEE International Conference onServices Computing.

[3] Dan Hao, Xu Zhao, Lu Zhang "Adaptive Test-Case Prioritization Guided by Output Inspection" IEEE 37th Annual Conference on Computer Software and Applications, 2014.

[4] Shweta A. Josh "Effective Use of Prim's Algorithm for Model Based Test case Prioritization"international Journal of Computer Science and Information Technologies 2014.

[5] SajjadMahmood, Muhammad Ali Khan "A Degree Centrality-Based Approach to Prioritize Interactions of Component-Based Systems" IEEE International Conference on Computer & Information Science, 2009.

[6] Saddek Bensalem, Axel Legayy, AyoubNouri, DoronPeled "Synthesizing Distributed Scheduling Implementation for Probabilistic Component-based Systems" IEEE International Conference onFormal Methods and Models for Codesign.

[7] Shashank, S.P, Chakka, P. , Kumar, D.V. "A systematic literature survey of integration testing in component-based software engineering" IEEE International Conference onComputer and Communication Technology

[8] Singh, S. , Yu, C.L.K. "Efficient Component Based Software Engineering using the TCEM Methodology and the TCET Tool" 13th Asia PacificSoftware Engineering Conference

[9] Georgiana Macariu, Vladimir Cret¸u "Enabling Parallelism and Resource Sharing in Multi-core Component-based Systems" 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing ,2013.

[10] Guoyou Shi, Shuang Liu "Object-oriented design and implementation of communication software based on TCP/IP and multithread" 9th International Conference onFuzzy Systems and Knowledge Discovery (FSKD), 2012.

[11] Ning, J.Q. “Component-based software engineering (CBSE)” Fifth International Symposium on Assessment of Software Tools and Technologies.

[12] Pour, G. “Integrating component-based and reuse-driven software engineering business into software and information engineering curriculum” 30th Annual Frontiers in Education Conference.IEEE International Conference on E-Commerce

[13] Vu Tran, Dar-Biau Liu “A procurement-centric model for engineering component-based software systems” IEEE Proceedings Fifth International Symposium onAssessment of Software Tools and Technologies

[14] Yukyong Kim, Kyung-Goo Doh “Adaptable Web Services Modeling Using Variability Analysis” IEEE Third International Conference onConvergence and Hybrid Information Technology

[15] Zheng Yan, Christian Prehofer “Autonomic Trust Management for a Component-Based Software System” IEEE Transactions on Dependable and Secure Computing

Books

[16] Fundamental of software engineering by Rajibmall,first edition

[17] Fundamental of software engineering by rajibmall,third edition

[18] Oriented analysis and design by booch first edition

[19] software quality assurance by Daniel galin first eition

[20] Software testing Ronnpaton second edition

Sites

[21] https://en.wilkipedia.org/wiki /Software _ engineering

[22] https://www.google.co.in/

[23] https://www.google.co.in/?gfe_rd=cr&ei=IJd3vNvYBTv8geUroDwaq&gws_rd=ssl#qd=software+engineer

[24] https://www.weopedia.com/terms/s/software_engineering.html

[25] https://www.rspa.cpm/spi/

[26] https://en.wilkipedia.org/wiki/uml

[27] http://www.springer.com/content/?k=software+engineering .

[28] http://www.tutorialpoint.com/software_testing/testing_types.htm

[30] http://softwartestingfundamentals.com/software-quality-assurance

[31] http://ieeexplore.ieee.org/search/advsearch.jsp?reload=true

CHAPTER 7

APPENDIX

(CBSD)	component based software development
(SDLC)	Software development lifecycle
(CBSE)	Component-based enterprise software engineering
(VFT)	Visualization and formalizationtool
(CIG)	Component integration graph
(SBD)	Segment based programing
(CBS)	Componentbased system