AUTOMATIC TEST CASE GENERATION ON CLIENT SIDE IN SERVICE ORIENTED ARCHITECTURE

Dissertation submitted in fulfilment of the requirements for the Degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

HEMENDRA SAHU

11509550

Supervisor

Mr. MAKUL MAHAJAN

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Month April, Year 2017

ABSTRACT

As the enlargement of web services are continuously increasing day by day, the Service Oriented Architecture is widely used to overcome the communication problem, which arises between the systems having different configuration over the network. The testing of Service Oriented Architecture is an important challenge, which is done in service, interface and end to end levels. For testing any system, test cases are used but the large number of test cases increases the overheads. Therefore, there is a need to optimize the test cases. In this research, we generate the optimal number of test cases in the Service Oriented Architecture using client side testing with the help of genetic algorithm. Firstly, the web data and elements are extracted from web pages and with the help of these data digraph is generated. Using diagraph, the test cases are generated and the test cases are optimized using genetic algorithm. At last, we have compared the traditional approach with genetic one and it has been observe that with genetic approach lesser number of test cases are required as compare to the traditional one.

DECLARATION STATEMENT

I hereby declare that the research work reported in the dissertation entitled "AUTOMATIC TEST CASE GENERATION ON CLIENT SIDE IN SERVICE ORIENTED ARCHITECTURE" in partial fulfilment of the requirement for the award of Degree for Master of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab is an authentic work carried out under supervision of my research supervisor Mr. Makul Mahajan. I have not submitted this work elsewhere for any degree or diploma.

I understand that the work presented herewith is in direct compliance with Lovely Professional University's Policy on plagiarism, intellectual property rights, and highest standards of moral and ethical conduct. Therefore, to the best of my knowledge, the content of this dissertation represents authentic and honest research effort conducted, in its entirety, by me. I am fully responsible for the contents of my dissertation work.

Signature of Candidate

Hemendra Sahu

R.No 11509550

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the M.Tech Dissertation entitled "**AUTOMATIC TEST CASE GENERATION ON CLIENT SIDE IN SERVICE ORIENTED ARCHITECTURE"**, submitted by **Hemendra Sahu** at **Lovely Professional University, Phagwara, India** is a bonafide record of his / her original work carried out under my supervision. This work has not been submitted elsewhere for any other degree.

Signature of Supervisor

Mr. Makul Mahajan

Date:

Counter Signed by:

1) **Concerned HOD:**
   HoD's Signature: _____

   HoD Name: _____

   Date: _____

2) **Neutral Examiners:**

   External Examiner

   Signature: _____

   Name: _____

   Affiliation: _____

   Date: _____

   Internal Examiner

   Signature: _____

   Name: _____

   Date: _____

ACKNOWLEDGEMENT

Words are often too less to reveal ones deep regards. An understanding of the work like this is never the outcome of the efforts of single person.

I would like to express the deepest appreciation to my dissertation mentor **Mr. Makul Mahajan**, who has shown the attitude and the substance of a genius. He continually and persuasively conveyed a spirit of adventure in regard to research and an excitement in regard to teaching. Without his supervision and constant help this dissertation would not have been possible.

I am highly indebted **Mr. Dalwinder Singh** HOD (CSE) and **Dr. Rajeev Sobti** HOS (CSE) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my family for their kind co-operation and encouragement which help me in completion of this project.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Checklist for Dissertation-II Supervisor

Name: _____ UID: _____ Domain: _____
Registration No: _____Name of student:_____

Title of Dissertation:
_____

- ☐ Front pages are as per the format.
- ☐ Topic on the PAC form and title page are same.
- ☐ Front page numbers are in roman and for report, it is like 1, 2, 3…….
- ☐ TOC, List of Figures, etc. are matching with the actual page numbers in the report.
- ☐ Font, Font Size, Margins, line Spacing, Alignment, etc. are as per the guidelines.
- ☐ Color prints are used for images and implementation snapshots.
- ☐ Captions and citations are provided for all the figures, tables etc. and are numbered and center aligned.
- ☐ All the equations used in the report are numbered.
- ☐ Citations are provided for all the references.
- ☐ **Objectives are clearly defined.**
- ☐ Minimum total number of pages of report is 50.
- ☐ Minimum references in report are 30.

Here by, I declare that I had verified the above mentioned points in the final dissertation report.

Signature of Supervisor with UID

CHAPTER 1
INTRODUCTION

1.1 SERVICE ORIENTED ARCHITECTURE

Nowadays, due to the exponential growth of the web services and heterogeneity between systems, there is a need of building the high-quality systems for the services. The performance of these systems is highly dependent on the distant web services and the unpredictable nature of the Internet. The web service is the method of communication between the heterogeneous systems [20]. To overcome the problems, Service Oriented Architecture (SOA) is introduced. Most of the SOA based implementations are built with the web services. SOA is a style of architecture in which the components of application provide service to other applications by a communication protocol over the network. The SOA principles are independent of any technology, vendor or product. SOA is an Information technology along with model driven architecture and business process management which facilitates business globalization, virtualization, digitization and agility [15].

A service is an operation which may be invoke distinctly and is a complete set of functionality. In Web Service Description Language (WSDL), several distinct operations are listed by an interface called service. It is a component which is encapsulated with interface in the background. With the help of SOA, it is easy for software components to connect over the network to communicate. Each computer can run number of services and each service can exchange information over the network with the help of other services, without human interaction and without any changes in program.

In Figure 1.1, the SOA triangle shows that the SOA provides the interaction between the service providers and the clients. In SOA the provider publish the service description which is specified by WSDL, in the registry. The registry is mostly implemented by UDDI standard and it acts as a mediator between the clients and the service providers. After the publishing of service by provider, the client sends the find query for desired service. The registry processed the client query and returns the desired service according to the requirements. The client selects the service and binds it with providers that performs the service citation and receive the desired response.

As the client can decide which service is going to be implemented at the design time, the static binding is considered [13].

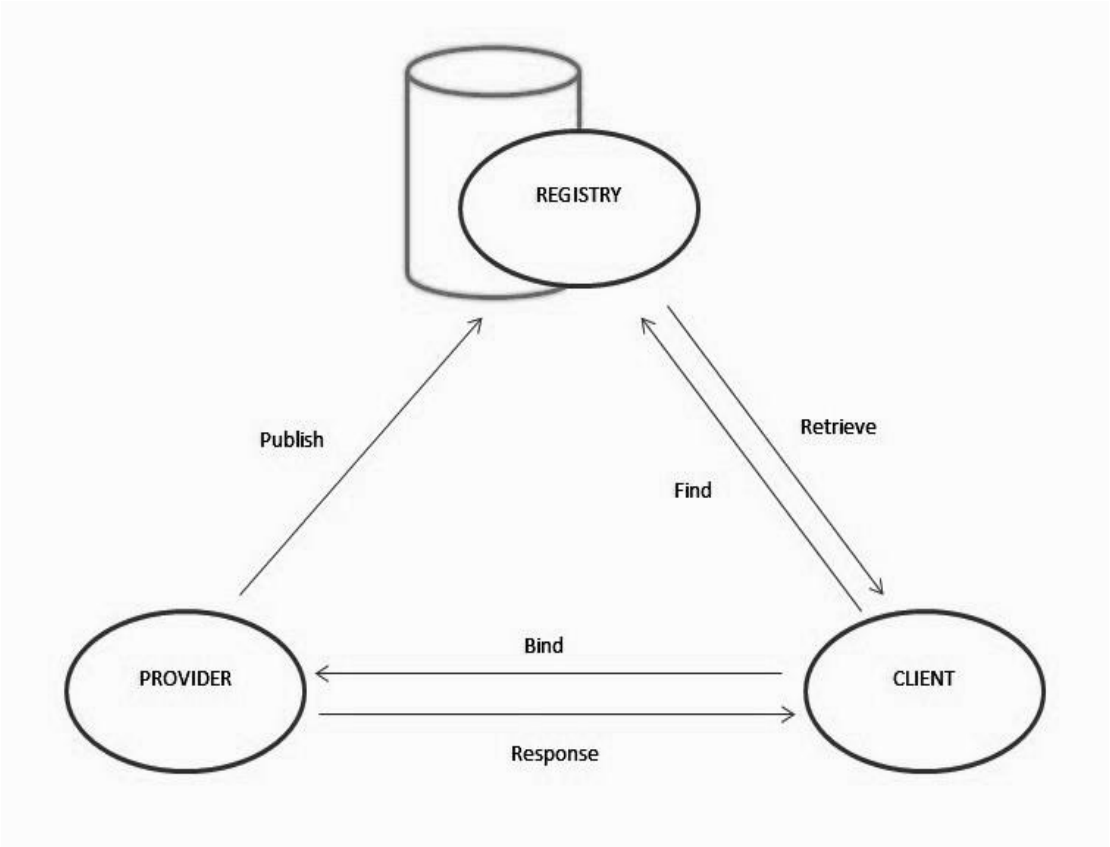


Figure 1.1: Operations and Roles of SOA

1.1.1 DESIGN CONCEPT

During the software development and its integration the SOA design principles are used. It provides a way for customers to be aware of service provided by the SOA, such as Web Based Applications. For example, in a company, different departments develop and deploy the SOA by implementing in different languages and with the help of well-defined interface their respective clients and users are benefited.

The integration of various Web Based Applications and the implementation using different platform are defined by SOA. Instead of defining the API, it defines the interface for the functionality and protocols. The entry point of such SOA implementation is an endpoint.

There is a loose coupling of services is required by SOA to interact with the operating systems and other technologies, which uses web applications. It separates the functions of different units or services, which are made by the developers to

access over the network by client. The communication between the services and the corresponding users is done by sharing data in the well-defined format.

1.2 TESTING IN SOA

Every software product is supposed to be going through a quality assurance cycle to make sure the product that is delivered lives up to its quality standards. Therefore, testing is a major effort required in any software development project.

Most people consider web service testing as SOA testing. However, as mentioned before, SOA is not only about the web services. It is about the overall architecture. In other words, SOA testing should not be restricted to just web services testing.

An SOA solution is generally an integrated set of products that can be a collection of legacy applications, third-party components, or custom developed components and among others. Testers are expected to test not only the individual products and its functionality, but also the overall integrated solution. Therefore, an SOA tester has to consider the bigger picture of SOA for testing, including service providers, service directories, service consumers, communication between SOA components, and authentication providers.

Nowadays, all the business depends on the IT systems for their everyday operations and the SOA acts as a backbone of these enterprise systems. If any single error occurs in the system, it may result for enterprise to lose lots of money and damaging their brand. Because of this reason the testing of SOA is very crucial.

Testing of SOA is emphasis on three system layers which are as follows:

- **Services layer: -** This layer consists of services, services exposed by the system derived from business functions.
- **Process layer: -** This layer includes the processes, collection of services which are the part of a single functionality. The processes might be a part of user interface (for example A search engine), a part of ETL (Extract, Transform & Load) tool (for getting data from the database). The main focus will be in user interfaces and process.
- **Consumer layer: -** This layer mainly includes the user interfaces.

Based on the layer, the testing of an SOA application is distributed into three levels.

- **Service level: -** It includes testing the component for functionality, performance, security and interoperability. Every Service needs to be first tested independently.
- **Interface level: -** It includes all possible business scenarios. The Non-Functional testing of the application should be done once again in this phase.
- **End to End level: -** It ensures that the application verifies the business requirements both functionally and non-functionally. The below items are ensured to be tested during end to end testing
    - All services working as presume after integration
    - Handling the exceptions
    - Application's User Interface
    - Proper data flow via all the components
    - Business process

Other types of SOA testing: -

- **Functional Testing: -**Functional Testing should be done on each and every service to:
    - Ensure that the service provides right response to each request.
    - Proper errors are received for requests with Invalid data, bad data, etc.
    - Check for each request and response for each and every operation.
    - When an error occurs at the server, client or network level, validate the fault messages.
    - Check the responses received are in the right format.
    - Check the data received on the response correspond to the requested data.

- **Security Testing: -** It is an important characteristic during service level testing of the SOA application; this confirms the safety of the application.
  During testing the following factors need to be covered:
    - Industry Standard defined by WS-Security testing should be observed by the Web Service.
    - Security measures should work perfectly.
    - Digital signatures on the documents and Encryption of data.
    - Authorization and Authentication.

- SQL Injection, Malware and other exposures are to be tested on the XML.
- Denial of Service attacks.

- **Performance Testing: -**It needs to be done since the services are reusable and multiple applications might be using the same service.

  The following factors are considered during testing:

  - Functionality and performance of the service need to be tested under heavy load.
  - The performance of the service needs to contrast while working individually and with the application, it is coupled with.
  - Load testing should be performed
    - To verify the response time.
    - To check for bottlenecks situation.
    - To verify CPU and memory utilization.
    - To predict the scalability.

The major issues that make the testing of SOA difficult are as follows [13]:-

1) Lack of source code observability.
2) Lack of control in services.
3) Testing cost when services are not deployed in the infrastructure of testers
4) Cost derived from the exhaustive test suite could not be manageable.
5) Time for testing is limited.
6) The solutions of SOA are heterogeneous and complex in nature.
7) Simulation of testing environment is difficult.
8) Reporting issues are difficult
9) It is more data driven than the traditional testing.

The difference between the traditional testing and the SOA testing approach is that, in traditional approach the tester publishes the message as request, through the proxy services and checks, if the required response and reply sequence is getting or not while in SOA testing approach the tester need to consider all the possible scenarios which are required for that system like –

1) Receive reply sequence is according to the respective response message or not.
2) Check how the system performs when the backend service is shut down.
3) Check the system performance during the slowdown of backend services.

4) Check for message duplication, loss and modification in the system.
5) System behavior when the message sent in different format.
6) Check individual message forwarding and storing features.
7) System behavior on high loads.

There are some essential qualities which are required in the software tester as well as in the SOA tester:

1) Fully attention on details.
2) Ability to quickly detect the problems.
3) Able to understand product and its integrated parts.
4) Willingness and passion in breaking the software systems.
5) Willingness to question and object in every aspects of the software.
6) Willingness to complete everything in well manner and in perfect way.
7) Having good knowledge in all aspects of IT basically database and administration.
8) Having good knowledge of software development.
9) Test the application as a mindset of developer, not as a normal user.

1.2.1 TEST CASE GENERATION

A test case is defined as "a documentation specific inputs, predict results, and a set of execution conditions for a test item". For generating the test cases, the test data can be generated by different sources like source code, system specifications, business scenarios, system requirements and use cases which describe the SUT (System/Service Under Test) [2].

There are three approaches for test data generation which are as follows:-

1) Based on specification.
2) Based on contract.
3) Based on partition.

Testing based on specification is the verification of the SUT on a reference document such as a description of user interface, a design specification, and a list of requirements or a user manual. Generally, in this testing, using the available system specifications, the test cases are generated.

In testing based on contract, the contracts define the preconditions for a component to be accessed and the postconditions that need to hold after the execution methods of that component with the specified preconditions. The contracts are used to

detect some unexpected behavior of the SUT, and contracts information can also be used to enhance the testing process itself.

Partition based testing is a technique to find subsets of the test cases that can fully test a system. The aim of this testing is to distribute the input domain of SUT into subdomains, so that generating or selecting the number of test cases from each subdomain will be enough for testing the whole domain.

CHAPTER 2

REVIEW OF LITERATURE

Wenbin Dai et al. [4] proposed a systematic model for the applications of SOA in the distributed architecture domain to achieve flexibility in automation systems. There was the mapping between the SOA and IEC 61499, in which some basic SOA principles like loose coupling and discover-ability were used. This standard defines the set of management commands to provide the skill of dynamic reconfiguration without influencing the normal operations. The Figure 2.1 shows the design pattern of IEC 61499 based on SOA. The characteristics of flexibility and re-configurability were also showed with the help of case study. Optimization of SOA was needed at runtime to reduce communication overhead and memory consumption.

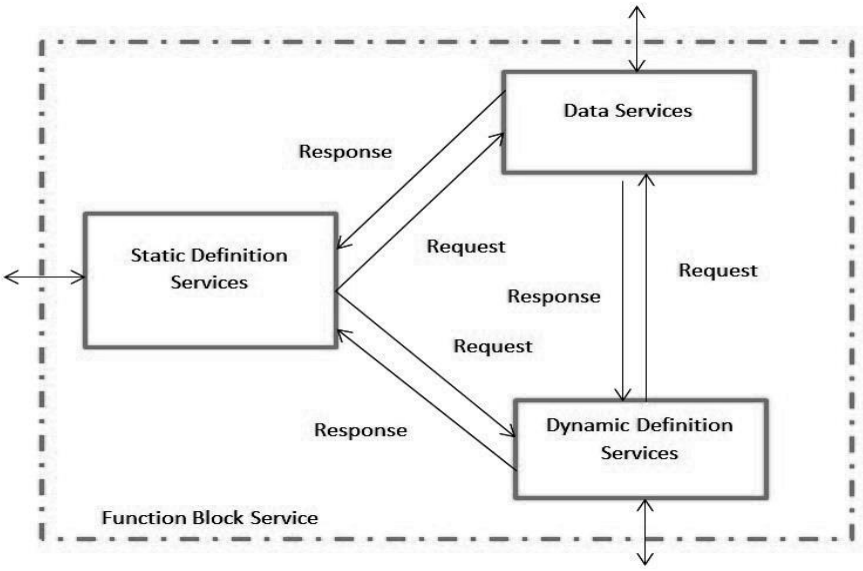


Figure 2.1: Design pattern for SOA-based IEC 61499

Zeng et al. [15] proposed a Model Driven Architecture (MDA) and Service Oriented Enterprise (SOE) architecture in which there were three stages for performance analysis based on simulation and optimization of method for business process. Firstly, analyze the overall performance based on Analytic Hierarchical Process (AHP) for service binding and matching. Secondly, evaluate the performance based on simulation for the business process composition. Lastly, optimizing the performance of Service oriented enterprise network based on business process

simulation. The predictability, flexibility, robustness, agility and measurability of SOE and SOEN (Service Oriented Enterprise Networks) were enhanced by the combination of MDA, SOA and PAO (Performance Analysis and Optimization).

Deivamani et al. [9] proposed the algorithm called a User Preference based Web Service Ranking algorithm (UPWSR algorithm) which provides rank to the user preference based web services and also the QoS (Quality of Service) aspect of web services. The proposed framework overcome the issues of we service composition. It provided the flexibility for satisfying multiple QoS requirements and also considers user preferences dynamically. It provides lower execution time and supports the user preferences. This framework was implemented for the travel application only.

Zhang et al. [20] proposed a framework called an Online performance prediction (OPred) framework which can efficiently predict the service oriented systems performance based on the past consumption experience from different users. It personalized the performance prediction at runtime. The Figure 2.2 shows the Online Performance Prediction procedures. The drawback of the proposed technique of performance prediction of service was risky to identifying the poor services and unable to maintain the performance timely which can be overcome by techniques like data smoothing, utilizing information based on content and so on.

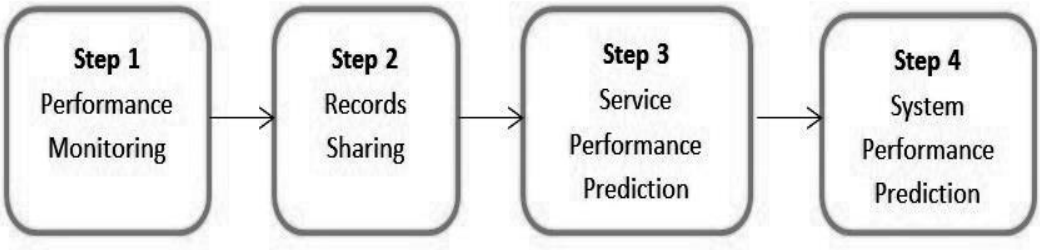


Figure 2.2: Online Performance Prediction procedures

For business process the SOA is getting strength as an arising distributed system architecture which can be observed in both academic and industry research. Bozkurt et al. [2] examined all the previous work related to testing and service oriented systems verification in which the pros and cons of current strategies along with the testing tools and future works issues were showed. The fundamental testing methodologies such as unit testing, regression testing, integration testing and interoperability testing and other testing methods such as model based testing ,fault

based testing, formal verification, partition testing, contract based testing and test case generation are surveyed. Due to increasing nature of web services the solutions provided by the survey were not effective.

Endo et al. [6] proposed a technique for testing web services based on event that enabled the systematic generation of test cases for structural testing with the aim of code coverage. That approach was the combination of view based on event and coverage which formed the combination of white box and black box testing i.e. the grey box testing. The whole strategy is illustrated with the help of an example of bank service. As for measuring the code coverage the source code is necessary, the applied approach was recommended for service developers. The drawback of proposed strategy was that it was limited to web services based on JAVA.

Palacios et al. [13] performed a systematic mapping study to evaluate the research in the scope of testing in service oriented architecture. The 33 studies were selected from the 392 studied for quality assessment and primary studies. Because of different factors, there were different limitations. Review protocol which includes the research questions had developed and guides the collection of search terms to identify the existing literature. There was a risk to select search term and keywords before starting the search. To overcome those risks during the review process, the protocol was refined and modified.

Wotawa and Leitner [18] proposed a complementary workflow for tackling the detecting and testing of SOA, discussed the test case generation of SOA in detailed and reported the preliminary researches on that. Due to limited observability, controllability and dynamic style of SOA, where actual implementation of services used at runtime is not known in advance, it is very hard to test the SOA. To overcome these entire problems there is a need to acquire the grey box testing technique. The author contributed in SOA by this paper in twofold:

i. Depends on more or less testing based on classical model.
ii. Test at runtime the dynamic improvement in the service implementation.

Jehan et al. [8] adopted a grey box testing approach based on model, which utilize the different description levels for individual services that required workability for successful development workflows. In this, the authors proposed the testing approach which consider SOA model and defines the constraints fulfillment problems for the steps of test case generation. The generation of test case approach was mainly

focus on testing functionality and combined constraints that were extracted from the Business Process Execution Language (BPEL) process activities. This approach was feasible for small BPEL processes but it was infeasible for the larger BPEL processes. Improvement required in generation of test cases.

Weir et al. [17] used the service oriented architecture which provides interoperable solutions to overcome the time consuming problems of the testing processes in NPI (New Product Introduction). The proposed structure was fully scalable and deployed in single site or global applications and installed in cloud or behind cooperate firewalls. Companies that had not well-defined data management capability were targeted the Agile Test Platform. It provided the end to end data management to the users and was not restricted to any particular installation. It was the powerful and simple tool that allowed companies to reduce the resources requirement of test data management.

Sneed et al. [16] proposed the new way of measuring the test coverage as the testers had not access of source code due to which they were not able to measure test coverage at the code level. The proposed approach satisfied the constraints of black box testing and focused on structure and content of service interface without regarding the code. The paper presented the different alternate methods for measuring the test coverage in software services without accessing the code. The limitation was no guarantee for the error free services and still there were the corners of the code left with unforeseen defects lurk.

Herbold et al. [7] proposed a Model and Interface Driven Automated testing of Service architectures (MIDAS) Testing as a Service for testing the SOAs in the cloud platform. The focus of this system was to cover the whole SOA system, complete task because of potential service interactions and each service complexity of the systems. Testing based on model approach was employed based on Unified Modeling Language (UML) and UML Testing Profile (UTP). MIDAS was skilled to generate and executed the composite test scenarios which would be useless to run locally. In Cloud the MIDAS provides testing in the form of service on the cloud to the clients or end users on rent in the basis of pay-per-use business policy.

Buchgeher et al. [3] proposed the architectural viewpoint based prioritization and selection of test cases approach that provides required architectural information to the software tester. The Architectural Based Testing (ABT) approach improved the

managing of test process for the enterprise SOA. This paper contributed a exclusive policy called Test Prioritization and Selection Explorer (TePSEx) that addressed the concern for software tester and the case study in which that point of view was applied in large-scale enterprise SOA testing process. The prioritization of test cases was supported by computing load factors for each usage called graph. The TePSEx was used by testers for defining and managing the applicable test case manually. Limitation of this approach was that it was based on static code analysis.

Liu et al. [11] optimized the web application test case and proposed a method called User Session Clustering based on Hierarchical Clustering (USCHC) algorithm for optimization of test case. The USCHC algorithm calculated the gap between the user sessions to employ the hierarchical clustering algorithm that makes clusters of test suites. It minimizes the number of test cases to improve the efficiency. In web application efficient testing along with optimization are needed for its reliability and quality. The USCHC method provided all this facility. USCHC was significantly the better than the other test case generation method for streamlining the initial test cases and had a good stability between the test time and its performance. As it was not easy to achieve the effectual practical tool, additional works were needed in this method.

User session based testing is an impressive way for web applications testing. Generation of test cases based on user session was not qualified in Black box testing approach. To overcome the problem, Peng [14] proposed the approach called User Session Data with Request Dependence Graph (US-RDG), which combined with Grey box testing method. In this method the test cases are automatically generated with the help of Genetic Algorithm. In the small test suite it covered the fault detection rate and the high path coverage. This paper used a concept called transition relation in the formed of "page-> request-> page" to charge the test case generation process. It showed the Empirical results for generation of test cases for web application but it won't meet the full coverage according to the structure point of view.

Dynamic contents of web application increase the new challenges in web application testing. For testing the dynamic content of web application Sinha and Arora [1] combined the approach of testing based on state and user session. They proposed the approach called user session based state testing that captured the web application dynamic contents. Finite State Machine was generated based on user

session to generate test cases for detecting faults. The limitation was that the generated test cases were not optimized.

In [19] Zhang and Qiu proposed an automatic test case generation by step-by-step fault injection approach for reliability assessment of web services. The proposed approach was capable of testing the web service hosts for its mischievous acts at the time of invocation. The faulty test cases were also used to test other web service attributes (eg. interoperability). This approach was appropriate for the limited exposed web services interfaces and the primarily focused was on reliability assessment only.

In [10] li et al. proposed an approach based on Grey box testing which involves data structures and algorithms and had access to the internal workings for designing test cases at user level as black box testing. The proposed approach was for business process and better SOA solutions whose main focus was in BPEL. The BPELTester tool was used to implement the approach with three major enabled – exploration of test cases, traces analysis and test selection. Improvement was needed in preciseness of test cases and level wise generation of test cases.

In [12] Noikajana and Suwannasart suggested a method to generate test cases for web service based on decision table. This method overcomes the generation of large number of test cases by black box testing. According to the web service requirements and description, the test cases were generated. The result of this approach showed that the generated test cases were decreased and still able to cover the expected outcomes of the web services with the help of TAD(Testing by Automatically generate Decision table).

In [5] deng et al. proposed Isomerous software based on SOA which covered the complete testing process and the conformity of different testing model was also implemented. It integrates the whole development process of software and effectively improves the effects of software automation testing. Further improvement was needed in modeling of automation testing process.

In [21] Wu and Lee proposed a framework, which efficiently generates the test cases based on SOA for the cloud services. It used source code comments for creating the WSDL-Ss (Web Service Semantics), which were further translated to ESGs (Event Sequence Graphs) by which test cases are generated and cloud services were verified. In this framework, the authors integrated the advantages of model based testing as well as the WSDL based testing. They develop a prototype system for this

framework and discussed all the strategies for improving the testing efficiency. In this approach, they integrate the coverage based, event based and the black box testing, but the error handling testing was not available.

In [22] Dancheng et al. proposed the data structure testing with predefined patterns, the basic algorithms used for test case generation and the deep study about the test data generation. As web services are more a standards rather than technology, this research was good for combining the development framework and the data which are tested. It reduces the development difficulty of testing the web services and increased the web test efficiency but it was limited to finish an algorithm independently of each type.

In [24] Liqiu and Yuhang implemented a method on testing framework which overcome the challenges arises due to the turning of web development into the agile development. It reduces the coupling between testing modules and ensure the quality of services as well as increase the coverage rate and test efficiency.

In [23] Dessislava et al. proposed an approach to automate the software tool which supports data driven test definition for generation of test cases using XML schema based testing for web services. It derives the XML instances from the schema of XML. This proposed approach was capable of deriving both the correct and incorrect instances of XML. The incorrect instances of XML data was applied for robustness testing. The tool was capable of doing the functional as well as robustness testing. This approach provides the limited set of test inputs only. The future work of this approach is to apply it in domains like population database, testing of web application and the generation of bench mark.

In [25] Tung et al. proposed a two phase approach for the automatic generation of test cases for web application by analyzing its structure. In this, the authors defined the relationship between the data and control dependency in the web application for reducing the number of test cases under the state explosion situations. It improves the way of test case generation by detect the relationship between the web application and its source code. Evaluating the defect reveal effectiveness and enhancing the model for web application along with to accommodate test in analysis and design phase of software requirement.

In [30] Wen et al. worked on increasing the adaptability of service oriented systems by overcoming the inefficiency of resources and handling runtime

exceptions. They established the software architecture, which worked according to the customer requirements and the service providers provide customized services, according to the requirements just in time effectively without any failures. They achieved the following progresses:

1. They investigate the adoptive SOA based on service requester along with run time exception handling.
2. Algorithm for adoptive SOA based on mathematics was built and verified.

In [26] Rastogi et al. worked on handling one or more faults in SOA based on QoS. They proposed a model for handling multiple faults by reconfiguring it in the faulty regions on the bases of different values of QoS, search algorithm computation and the optimal path from source to the destination. This model is depends on the concept of finding the path from source to destination, which is optimal and short. This model reduced the overhead time and complexity instead of regenerating the whole web service.

In [29] Kumar et al. proposed a testing framework which worked on reliability and fault free services based on SOA. The framework also focused on generating test report and explaining service provider role during testing. It provides solution for different aspects like test case selection and finalization, missed coverage items identification and web services versioning issues. Since for ensuring the robustness in web services the fault handling is needed, the working on fault handling part of this framework is their future work.

In [28] Jehan et al. proposed an algorithm for test case generation in SOA based on the selection of path randomly in a BPEL process. For this approach the test suite size was the most important because it was used for calculating the standard deviation. The future work of this algorithm is that it needs to consider the invalid paths percentages and combine the concepts of structural and random views for providing the attractive solutions.

In [27] Katsikogiannis et al. proposed a model which manages the multilevel identity integration formal policy based authorization and authentication modules. It simplified the management, improved the accuracy, and integrates the capability of automatic controls. In this model, they ensured and apply the suitable policy decisions to integrate the different components of access control.

The Table 2.1 shows all the applications used in the above paper along with tools or algorithms used in it.

Table 2.1 : Table for Application and its tools/algorithm used

Application	Tools/Algorithm used for Application
Model-driven and service oriented enterprise architecture	• MDA tool [15] • ETL tool [15] • Optimization tool [15]
Framework for travel application	• UPWSR algorithm [9] • QAWSC algorithm [9]
Online Performance Prediction Framework	• WSMonitor tool [20] • WSDL2Java tool [20] • Online prediction algorithm [20]
Banking Service Application	• JaBUTi tool [6]
Bank Loan Application	• BPEL test case generation algorithm [10] • BPELUnit tool [10]
Agile Applications	• Industry standard encryption algorithms [17]
Banking Application	• Research evaluation and validation tool [16] • TESTBED tool [16]
Cloud Application	• MBT tool [7]
Internet Banking Application	• TePSEx viewpoint [3]
Web Application	• USCHC algorithm [11] • GA algorithm [14] • Prototype tool [25]
Ajax Web Application	• Selenium tool [1] • GVEdit tool [1]

CHAPTER 3

PRESENT WORK

This section is divided into 3 parts i.e. problem formulation, objective of the study and research methodology.

3.1 PROBLEM FORMULATION

In the proposed approach we worked on generating the optimal number of test cases for the SOA based web application. In our approach the test cases are generated automatically for the testing of web application.

In our work we studied about the SOA and its processes, benefits, its working environment and the testing techniques possible in it. We analyze the different testing techniques along with algorithms and decided to precede work on client side testing. For that we apply a method in which, by using the graph, the test cases are generated for an individual path according to web page wise. As we all know, for testing any web application from client side generally two types of testing technique are possible i.e. black box testing and grey box testing. We are using both as per our requirements for generating test cases.

Firstly, we generate the test cases with the traditional approach then for optimizing it we apply the genetic algorithm in it. By this way, the test cases are generated with the proposed approach. At last we compare out results with the existing one.

3.2 OBJECTIVE OF THE STUDY

The objectives of our research are as follows: -

1) To propose an automatic test case generation approach based on genetic algorithm for SOA.
2) To apply the proposed approach and generate optimal test cases.
3) To compare the test cases generated using proposed method with already existing method.

3.3 RESEARCH METHODOLOGY

3.3.1 PROPOSED APPROACH

The proposed approach of our optimal test case generation is shown below in Figure 3.1. The process start from webpage of the website whose page content are taken as input by DOM Tree Inspector, which extracts all the elements that change state. All the elements along with the complete paths are collected using the selenium web driver. From the collected data a text file is generated from which the digraph is generated. By using this directed graph, for individual path test cases are generated. Then, these test cases are optimized by using the genetic algorithm.

3.3.2 STEPS FOR GENERATING TEST CASES

1. Open web page in web browser (Mozilla Firefox).
2. Extract elements by using DOM tree inspector.
3. Collect all the extracted elements and complete paths by using selenium webdriver.
4. By using the collected data, generate a text file for generating digraph by linking all the paths.
5. Taking text file as input, generate digraph using the Graphviz tool.
6. Using this digraph, generate the test cases for the individual paths by selenium webdriver.
7. These test cases are optimized by using genetic algorithm.

3.3.3 FLOW CHART OF TEST CASE GENERATION PROCESS

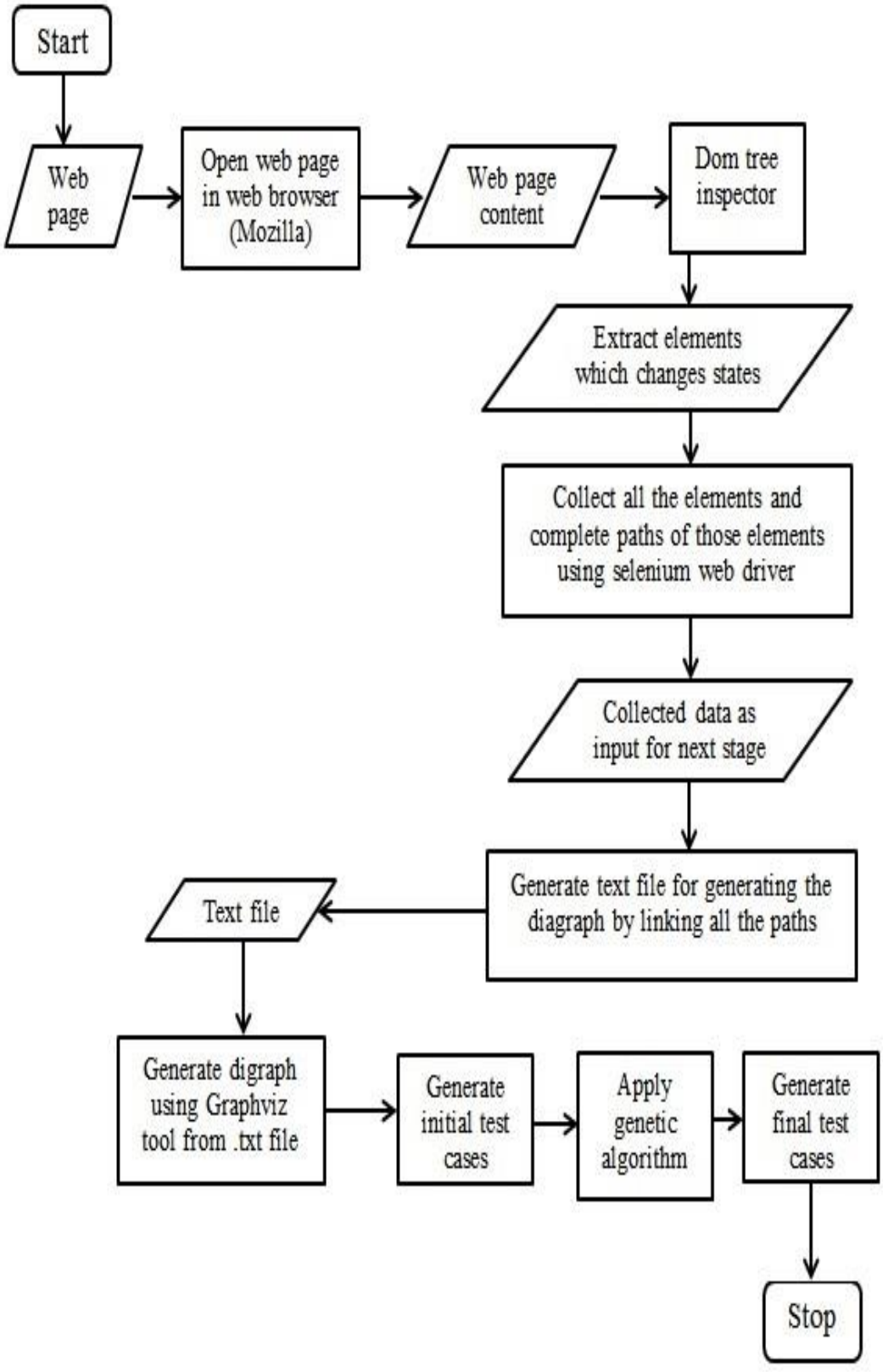


Figure 3.1: Flow chart shows the test cases generation process

In Figure 3.1, the flow chart shows the whole process of test case generation step by step starting from opening the web page in the browser and ends at the generation of final test cases from our approach. It also shows the tools and genetic algorithm which are used during process. These all are explained below.

3.3.4 GENETIC ALGORITHM

GA is an optimization technique which works on the principles of natural selection and genetics. It is a method of solving the constraints and unconstraint problem based on the process of natural selection. It continuously modifying a population until it gets the optimal output of an individual solution. Optimization refers to the searching of input values in such a way, so that we get the best possible output. We use initial test cases as input population for the GA and after optimizing it generates the final test cases as output. The Figure 3.2 shows the simple processes of GA.

Population is the solution subset of the current generation which defined the chromosomes set. For using GA we need to consider that the population size must be maintained and the size of population must not be too large as it may slow down the generation process. In GA the off springs are generated and replace the population for next iteration until the required output is not generated.

Selection is a process of selecting parents for creating the off springs of next generation. It is very crucial to select the good patents for better and fitter solution.

The crossover is an analogous biological crossover process in which more than one parent is selected and generates one or more than one off springs using the genetic properties of parents.

Mutation is a process of small random changes in the chromosomes to get the desired solution. Mutation is applied with low probability value in the GA because if the probability is high it reduces the random searches.

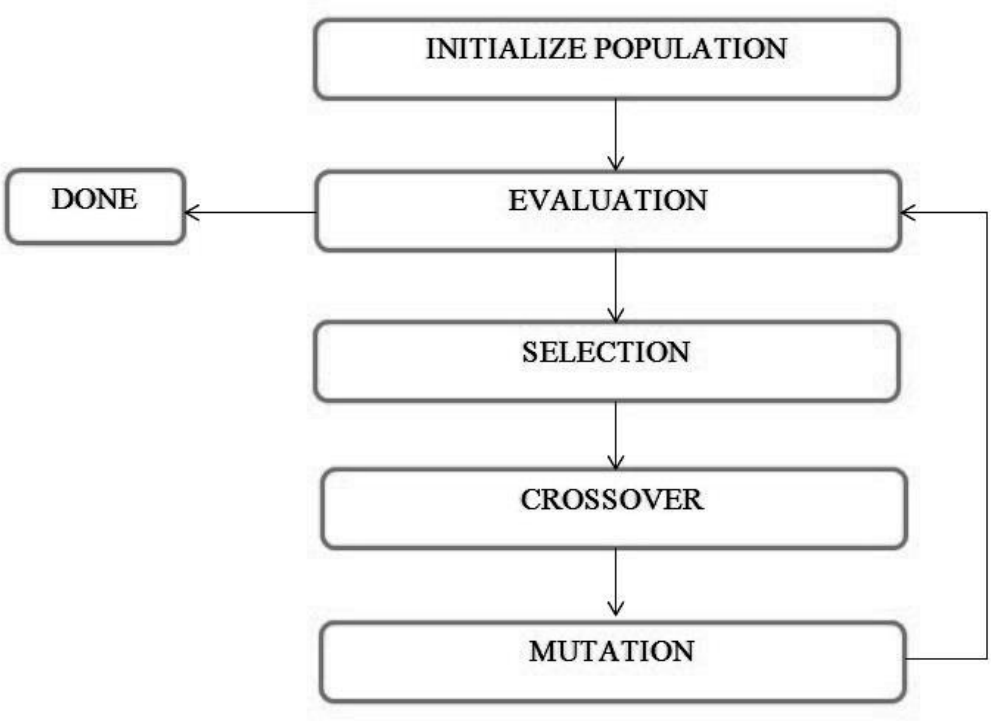


Figure 3.2: GA Algorithm

3.3.5 TOOLS

DOM Tree Inspector

The client side is generally consists of HTML, CSS, Java script and DOM. For displaying web content in a proper sequence, the HTML and CSS are used. Dynamic changes of documents are handle by Java script and DOM according to change in hierarchical structure of objects and to send and receive messages. For constructing the request which is sent to the server, the data located on an initial DOM object are used. The DOM inspector is available as extension in the Mozilla Firefox and it is easily accessible. All toolkit applications are supported by DOM inspector. It is used for verifying the DOM state by serve as a sanity checking and for manipulating the DOM structure. It visualizes the whole web page in the DOM tree form so that it is easy to inspect elements from tree.

The dynamic web application which leaves user interfaces active and responsive is dependent on asynchronous requests and combined with the probability that the pages dynamically updated through the DOM. When a response receives by

client, it updates the web document by adjusting the DOM objects. After further processes the document get back to the present page. From the web application file the dynamic objects are extracted to construct the DOM tree.

Eclipse

Eclipse is an Integrated Development Environment (IDE) which is mostly used for the java application development. It is used for other programming languages also. It supports different languages like – C, C++, Java, Python, Ruby, JavaScript, R, etc.

The Eclipse Software Development Toolkit (SDK) is basically used by java developers and it can extend by using different plugins available for the eclipse platform. These plugins are easily available in the Eclipse Marketplace. It also provides the feature that you can add plugins by setting path to the jar files. In our approach we use the Selenium Webdriver in eclipse, for that we add the Selenium package in the Eclipse.

Eclipse is the best IDE for java programming and other languages also because it provides the different types of features which are generally not known to us. There are different versions of Eclipse are available in the market like – Luna, Helios, Neon, Jupiter, etc. The best part of this very useful tool is that it is freely available or an open source tool. But it's a best practice to use the stable and updated version, so that, it provides the more new features.

Selenium tool

Selenium was developed by Jason Huggins as a library of JavaScript in 2004 to automate the testing routines. Initially it has functionality of Selenium IDE and the Selenium RC (Remote Server) but due to the browser security it was restricted to specific functionality only.

Later on a Google developer named Simon Stewart developed the Selenium webdriver which allows the browser to communicate with the operating systems using respective methods.

In 2008, the Selenium and webdriver are merged for providing the best possible framework of test automation and then it is called as Selenium suite or Selenium.

Selenium is a test automation tools suite for the web application. It consists of Selenium IDE, Selenium Webdriver, Selenium RC and Selenium Grid.

1) Selenium IDE is the Replay/Playback tool which is used by developers for writing simple tests quickly. It requires very little or no programming. It can export the test cases for Webdriver and RC. It doesn't provide iteration facility for the test scripts.
2) Selenium RC or the Selenium 1 uses the JavaScript functions when the browser is loaded and then uses the JavaScript for AUT (Application Under Test) in the browser. It supports mainly the maintenance mode and different languages like Java, Python, etc.
3) Selenium Webdriver or Selenium 2 provides the simple and concise programming interface which addresses the limitations of Selenium RC API. It makes directly calls to the browser and provides better flexibility and portability. It can be used for simple main method as well as for unit testing also.
4) Selenium Grid is used for parallel launching multiple browsers with their supporting operating systems using the scripts. It works in the heterogeneous environments for multiple machines. It reduces the time taken of completion for test suite to pass the test.

Graphviz tool

The Graphviz is an open source tool which is used for visualizing the graph. The visualization of graph is a way of showing the structural information in the form of abstract graphs and diagrams of networks. It takes the description of graph in simple text language and makes graphs in required format such as SVG (Scalable Vector Graphics) for web pages and images. It has different useful features for diagrams such as custom shapes, fonts, colors, layout, etc. Different roadmaps or plans for Graphviz are as follows:

1) **Dot -** It is a hierarchical drawing of the directed graph. It is the default tool if directed edges are used.

2) **Neato –** It is a spiral model layout. It is used for small graph (less than or equal to 100 nodes) and for minimizing the global energy function.
3) **Fdp –** It's also a spiral model like neato but instead of working with energy it reduces the forces.
4) **Sfdp –** It's a multi scale version of the fdp and is used for large graphs layout.
5) **Twopi –** It is a radial layout in which the nodes are arrange in the concentric circles equidistance from the root node.
6) **Circo –** It's a circular layout which is suitable for multiple cyclic structures like telecommunication networks.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 EXPERIMENTAL RESULTS

As per the literature review it is observed that in SOA the testing of web application from the client side is less and it requires more improvement. For web application testing from the client side only black box testing and grey box testing techniques are possible. In this research, we apply these testing techniques as per our requirements.

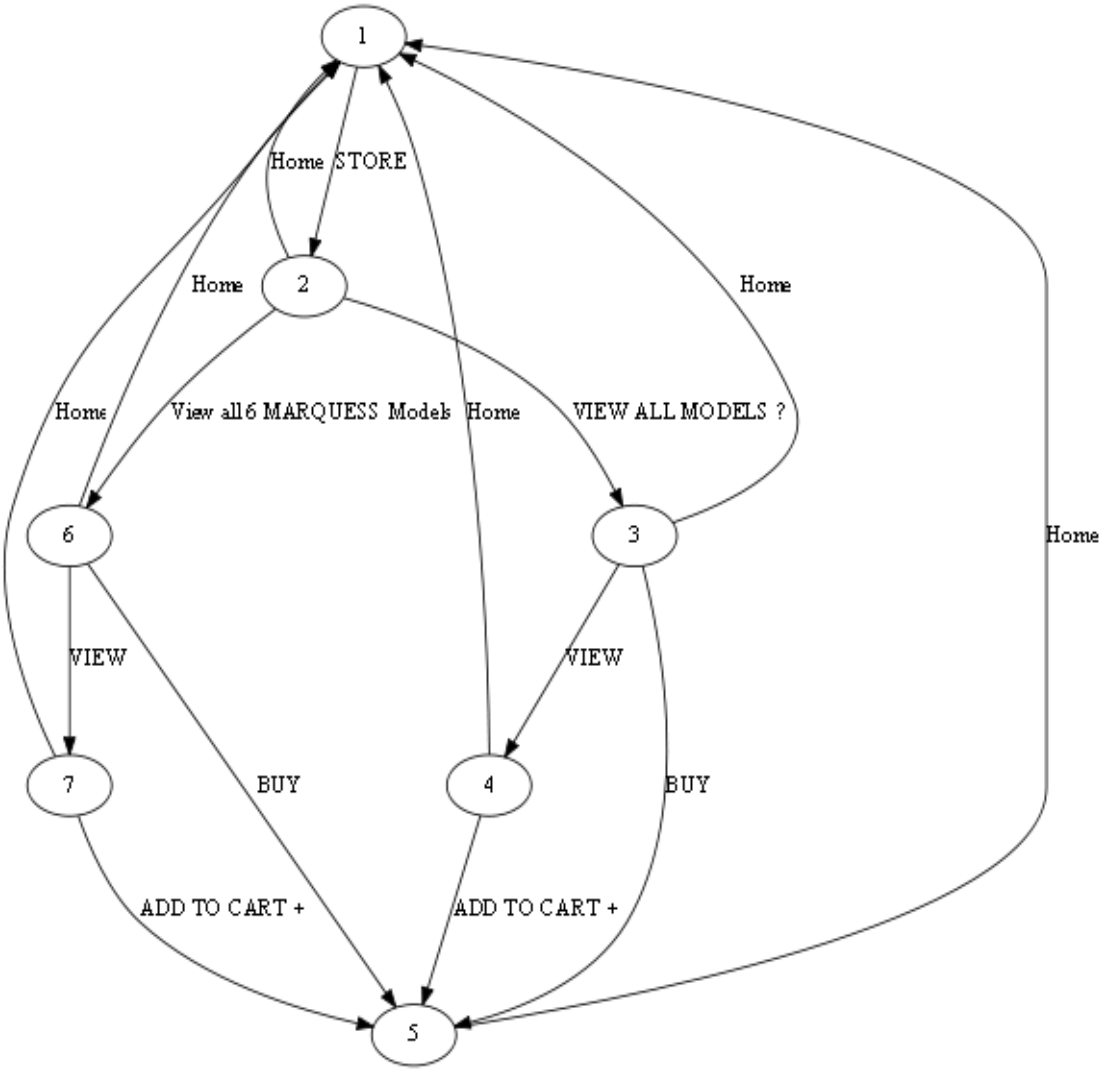


Figure 4.1: Digraph shows the whole path covered

First of all, we extract the elements and all related complete paths from a SOA based e-commerce website using Selenium Webdriver and eclipse. Then merge all the links and related information, by which we create a text file in the Graphviz format. Now by this file, a digraph is created which is shown in the Figure 4.1. From a web page of a website there are different paths are possible but we take some most common paths that the user can follow.

The Table 4.1 gives the details about the nodes used in the digraph in Figure 4.1. Each node represents a web page of a website which is used for creating digraph.

Table 4.1 : Diagraph nodes description

Nodes	Details
1	It represents the home page of an ecommerce website.
2	It represents the store page, in which products are arrange according to the category wise.
3	It represents the page which shows all the models of products.
4	It represents the page which gives the details of a specific product.
5	It represents the cart page in which the products are added for buying.
6	It represents the page which shows all the models of the specific category of product.
7	It represents the page which gives the details of the specific product of the specific category.

Using this digraph, the independent complete paths are taken for testing and test cases are generated for them. The Figure 4.2 shows all the independent complete paths taken from the digraph. In our approach testing of pages are done in path wise.

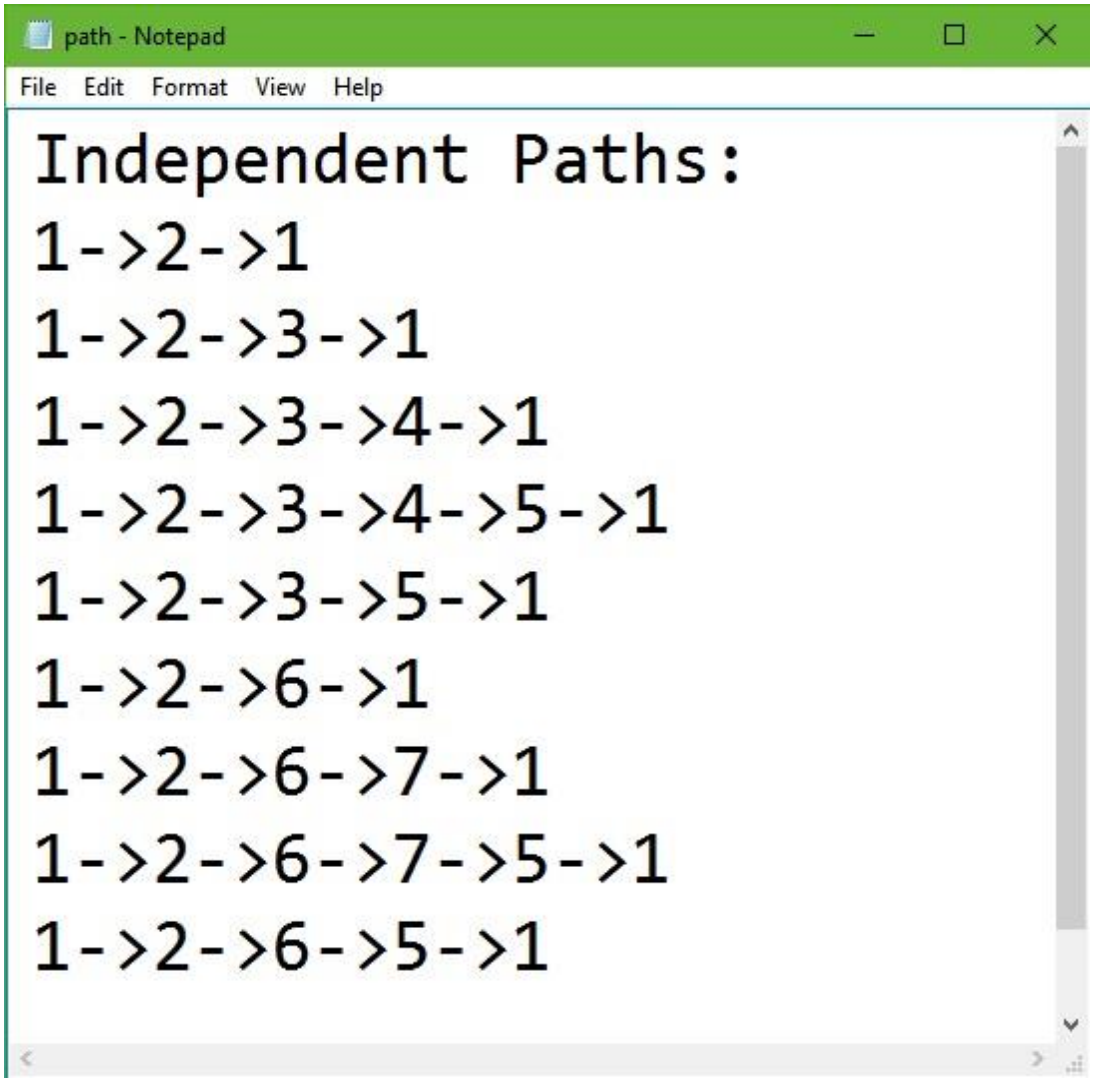


Figure 4.2: Independent Complete Paths

In the above Figure 4.2, each path like 1->2->3->1 show that the path is starts from the home page and also ends at the home page. These paths show the behavior of the user or the path generally followed by them. As the user can close the website when he/she on any page, but here we consider that the all path are ends with home page i.e. home page. Generally, user searches all types of products or he/she search the product in category wise if he/she knows some details about it.

There are number of test cases for testing are possible in the web pages of a website. But we take some of them which are shown in the Table 4.2.

Table 4.2 : Possible test cases in a web page

S. No.	Test Cases used in a web page
1.	Check the response time of a web page.
2.	Count the total number of links.
3.	Check number of invalid links.
4.	Count number of images in a web page.
5.	Check for invalid images in the web page.
6.	Count number of category of products.
7.	Verify and count the number of products.
8.	Check whether products are in sorted order or not.
9.	Check product images are clearly visible or not.
10.	Check product details are clearly visible or not.
11.	Check and verify the cart.

According to the type of web page the test cases are applied on it. In our approach we take these test cases only, but along with these test cases some other test cases are also possible. We take only these test cases because it takes too much time to execute like for example if we check for invalid links or the invalid images, it checks all the links or images for finding out the invalid link or images.

The first test case checks the response time of the page so that it opens in the acceptable time or not. It gives the timeout message if the server not response or the speed of net connection is very low. The net connection speed is very important during testing from client side because all the test cases are directly or indirectly depends on it. Likewise other test cases are for counting the total links and images available in the pages, checking for the broken or invalid links or images, checking the products details and its images are clearly visible or not, and we need to check the cart page also.

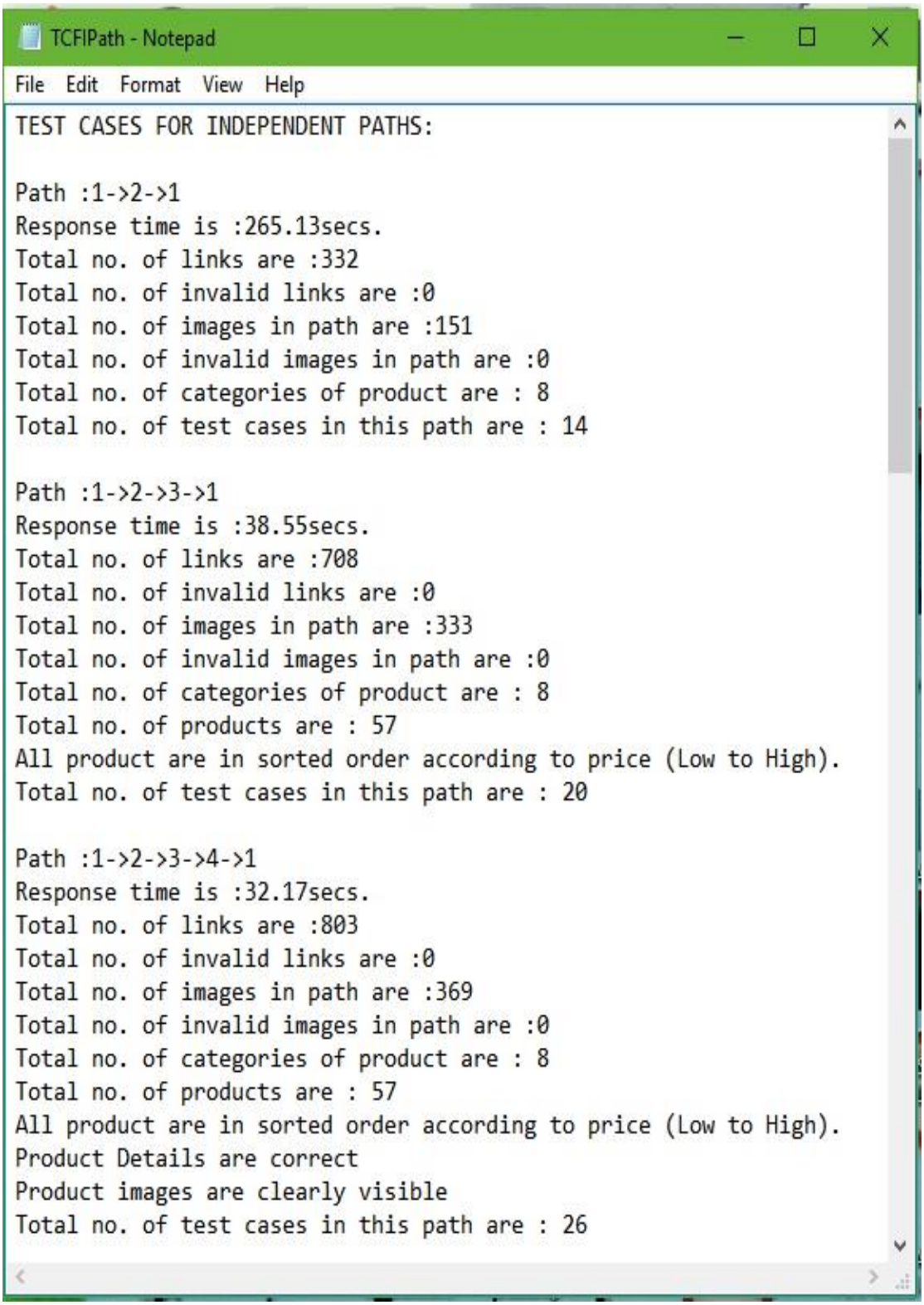


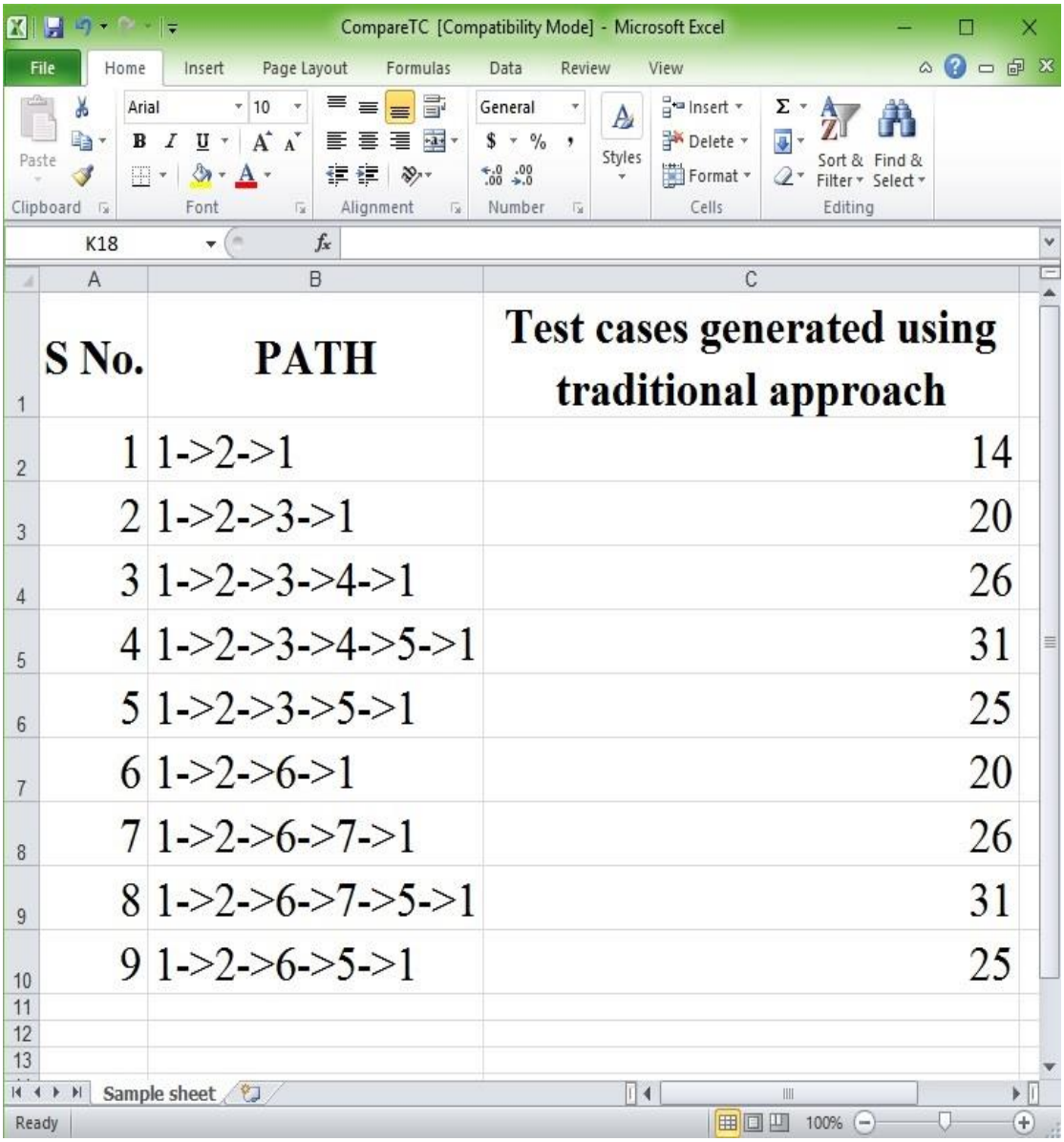
Figure 4.3: Test cases for Independent Paths

As in our approach we are generating test cases for the individual paths. All the test cases are executed and summarized path wise as shown in the Figure 4.3. It summation all the test cases results according to the type of test cases.

The above Figure 4.3 shows the path wise number of test cases for a specific task like- response time, total number of links, total number of invalid links, total number of images, total number of invalid image, etc.

4.2 COMPARISON WITH EXISTING TECHNIQUE

In this section we compare the result of our proposed approach with the traditional approach.

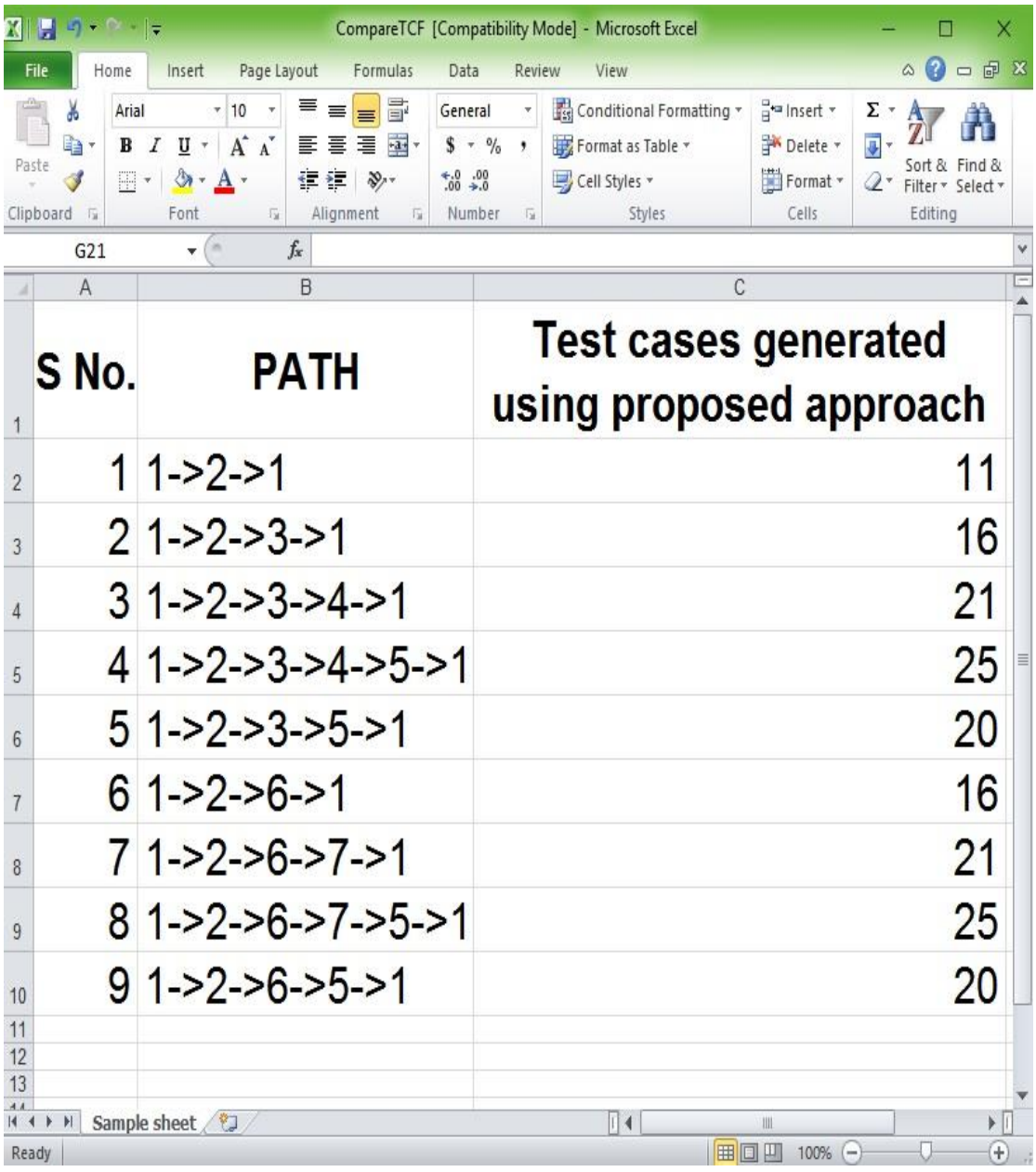


S No.	PATH	Test cases generated using traditional approach
1	1->2->1	14
2	1->2->3->1	20
3	1->2->3->4->1	26
4	1->2->3->4->5->1	31
5	1->2->3->5->1	25
6	1->2->6->1	20
7	1->2->6->7->1	26
8	1->2->6->7->5->1	31
9	1->2->6->5->1	25

Figure 4.4: Test cases generated by traditional approach

The above Figure 4.4 shows the test cases generated by the traditional approach. It shows the total number of test cases generated in a particular path.

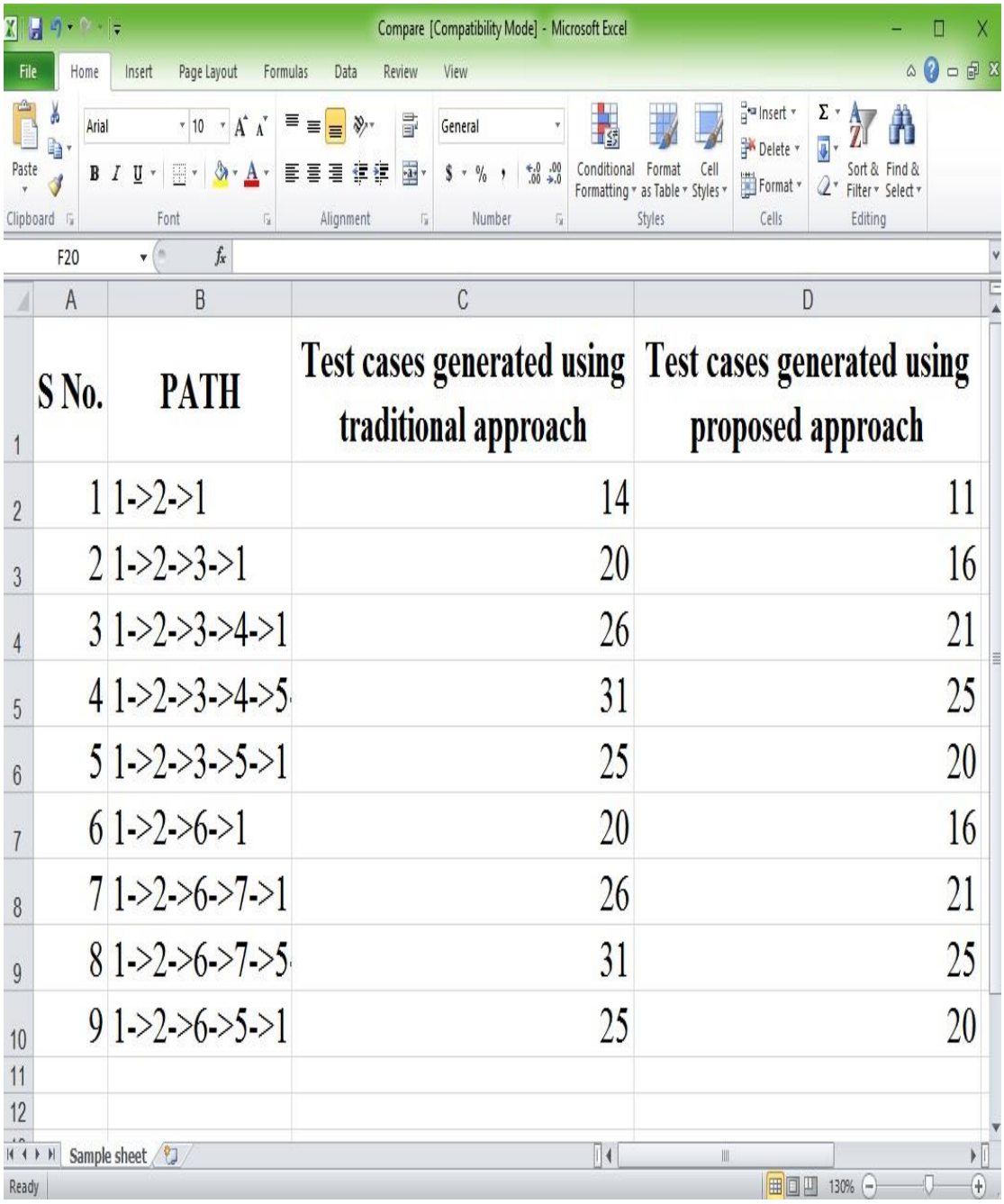
Using these test cases as initial test cases, in our approach we apply the GA for optimizing the number of these test cases. The optimal number of test cases is required for testing because if we have unwanted or useless test cases, it unnecessarily increases the execution time which is not acceptable. The optimal number of test cases is required for efficiently testing any type of application. It reduces the wastes of time, cost, resources as well as the number of execution.



S No.	PATH	Test cases generated using proposed approach
1	1->2->1	11
2	1->2->3->1	16
3	1->2->3->4->1	21
4	1->2->3->4->5->1	25
5	1->2->3->5->1	20
6	1->2->6->1	16
7	1->2->6->7->1	21
8	1->2->6->7->5->1	25
9	1->2->6->5->1	20

Figure 4.5: Test cases generated by proposed approach

The above Figure 4.5 shows the test cases generated by our proposed approach which are generated by applying the GA in the traditional one.



S No.	PATH	Test cases generated using traditional approach	Test cases generated using proposed approach
1	1->2->1	14	11
2	1->2->3->1	20	16
3	1->2->3->4->1	26	21
4	1->2->3->4->5	31	25
5	1->2->3->5->1	25	20
6	1->2->6->1	20	16
7	1->2->6->7->1	26	21
8	1->2->6->7->5	31	25
9	1->2->6->5->1	25	20

Figure 4.6: Comparison of test cases generated using traditional and proposed approach

Now we are comparing our result with the traditional one on the basis of number of test cases generated by both approaches. The Figure 4.6 shows the comparison of both approaches.

On the basis of above Figure 4.6, the comparison of test cases generated by both approach is graphically represents by various charts which are as follows.

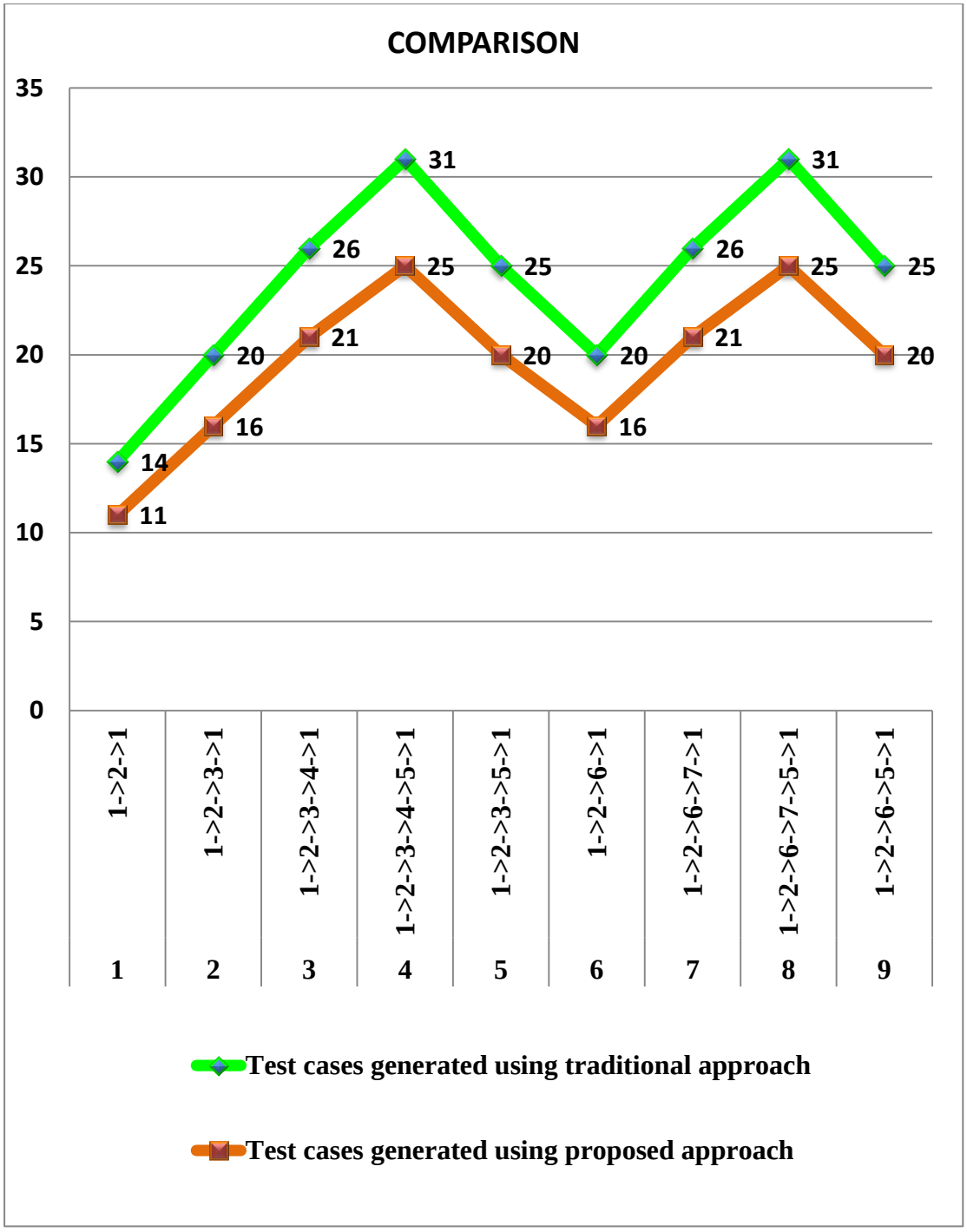


Figure 4.7: Line Chart shows the comparison path wise

In Figure 4.7, the line chat shows the comparison of both traditional and proposed approaches for each path.

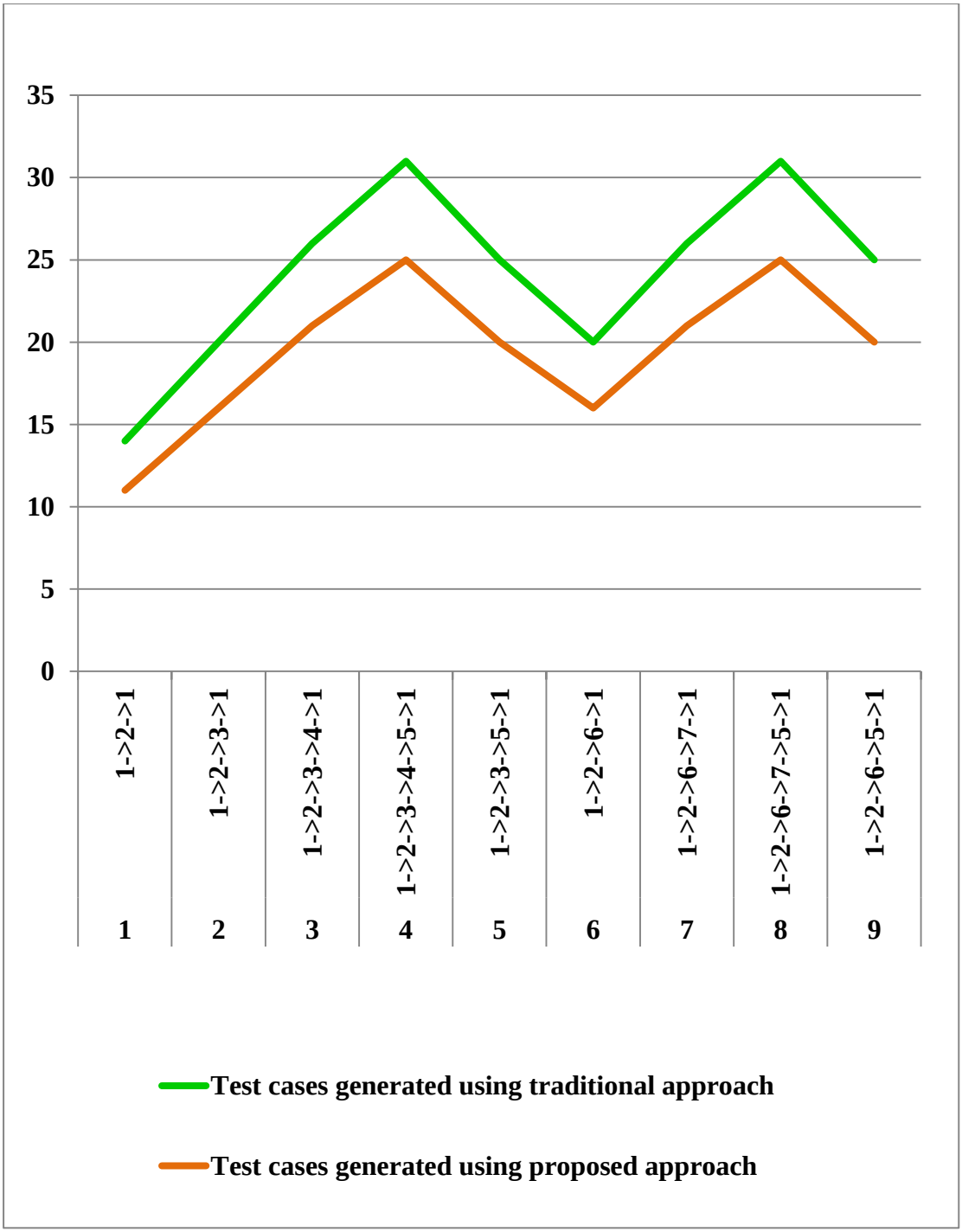


Figure 4.8: Line Chart shows overall comparison

In Figure 4.8, the line chart shows that the test cases generated by proposed approach are less as compared to the traditional one.

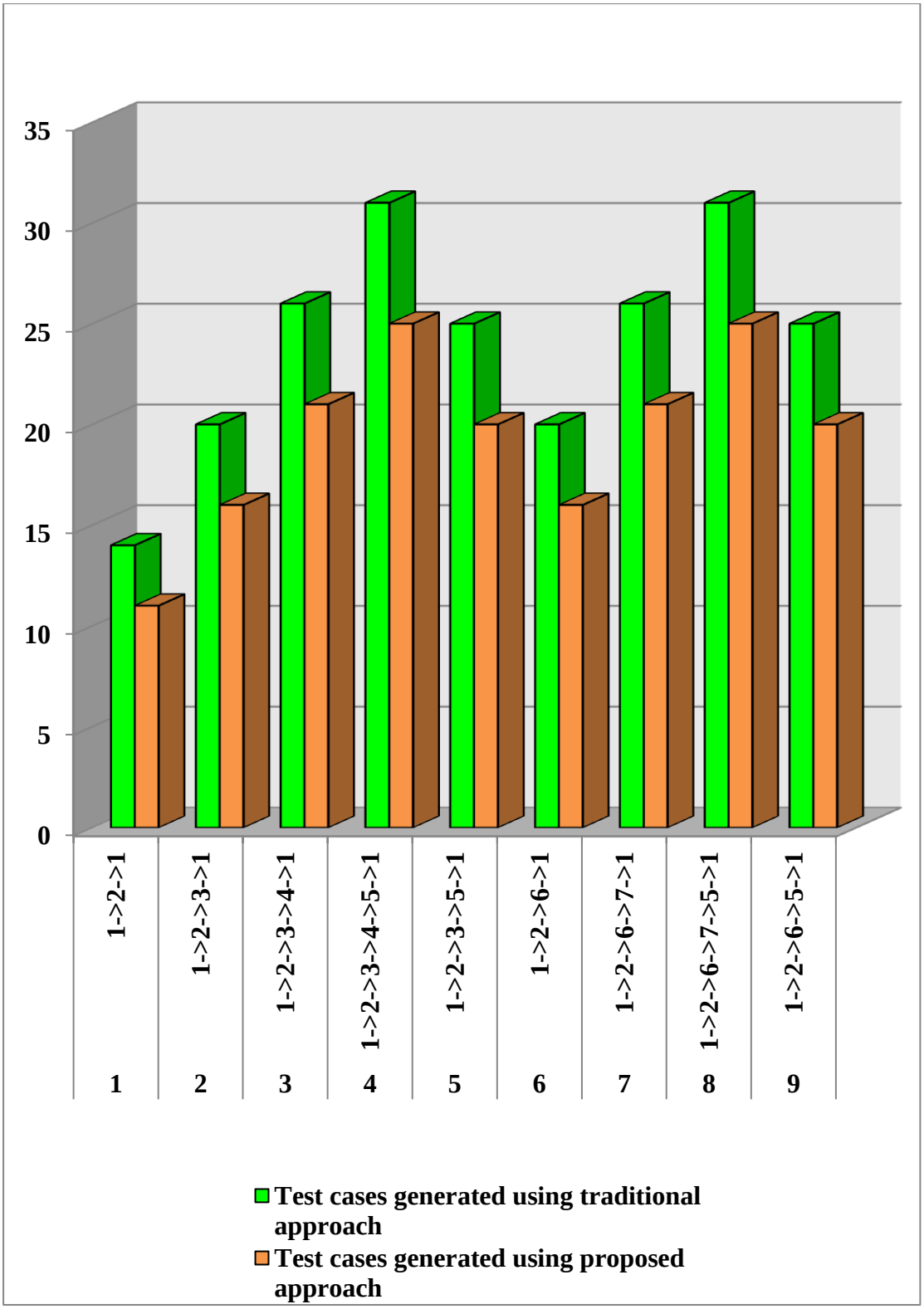


Figure 4.9: 3D Bar Chart

In Figure 4.9, the 3D bar chart shows that for the individual paths the generated test cases are less in proposed approach as compared to the traditional one.

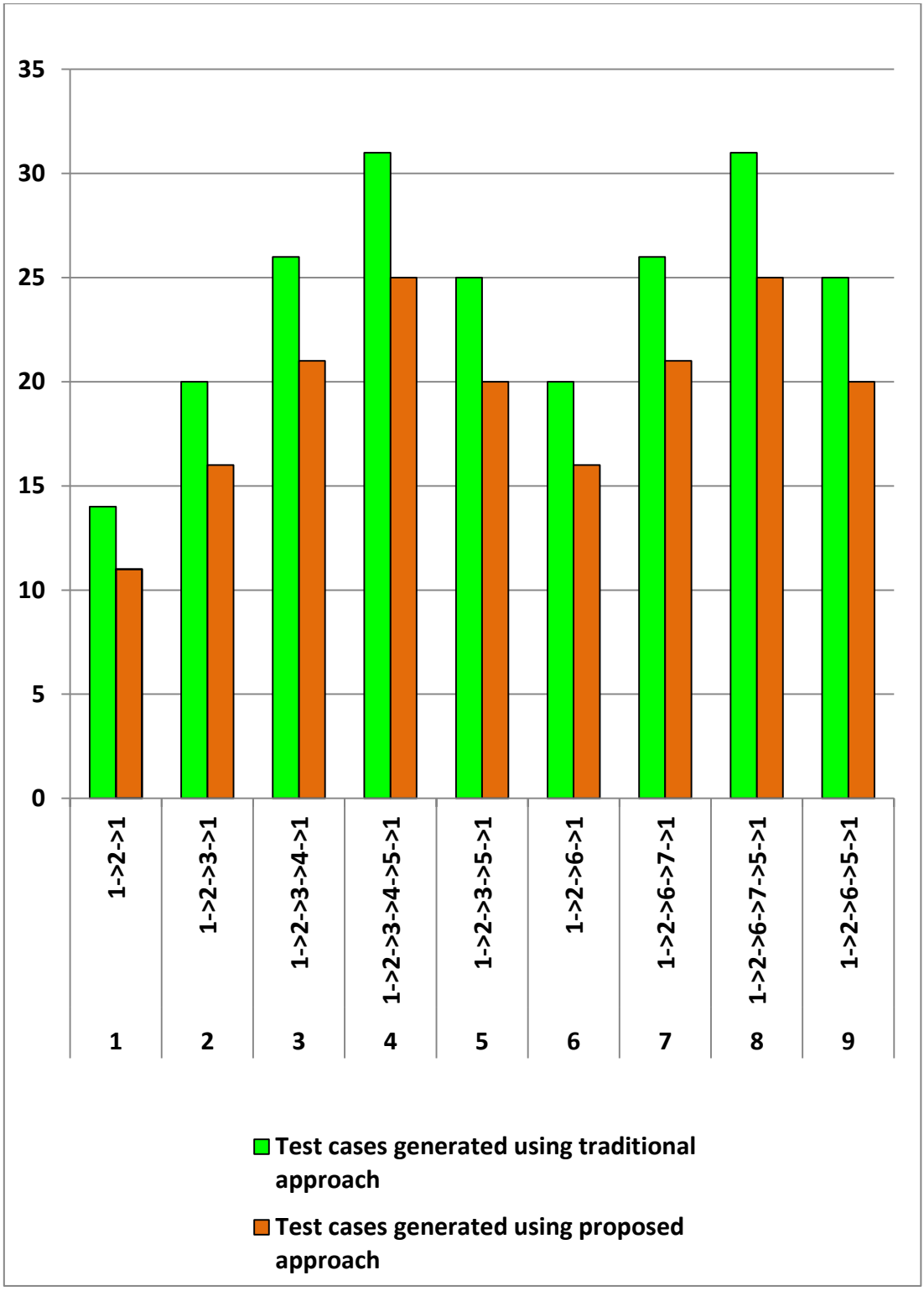


Figure 4.10: 2D Bar Chart

The above Figure 4.10, shows the 2D bar chart of test case generated for both approaches and it is clearly shows that the less number of test cases are generated by the proposed approach.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

In this report we study about SOA which overcome the problems of communication between systems over the network due to non-homogeneous nature of the systems. The main focus is on testing the SOA, in which particularly focus is on test cases generation. For testing any application, test cases are most important requirement, by which the testing is to be done. In our research, we take an e-commerce website for testing and on the basis of web pages we generate the test cases and applied on it. For optimizing the number of test cases, the GA is used. During this research, there are two main challenges we face. The first one is, to extract data and elements from webpages based on its DOM structure, which is the time taking process and the second one is, after the generation of test cases, the execution of some test cases take too much time. But the main focus is on generating and optimizing test cases which works fine.

5.2 FUTURE SCOPE

In our research, we are considering only the general test cases which are mostly apply in the web pages. But there are lots of test cases which can be possible in the web pages. For optimization, we are using only the GA. Like GA, other algorithms are also available for optimization. In future, we will try to consider and generate other test cases and along with that, we try to use the other optimization algorithms.

REFERENCES

[1] A. Arora and M. Sinha, “Dynamic Content Testing of Web Application Using User Session Based State Testing,” pp. 22–28.

[2] A. D. Brucker and J. Julliand, “Testing and verification in service-oriented architecture: a survey”, vol. 24, no. 8. pp. 591–592, 2014.

[3] G. Buchgeher, C. Klammer, W. Heider, M. Schuetz, and H. Huber, “Improving Testing in an Enterprise SOA with an Architecture-Based Approach,” 2016 13th Work. IEEE/IFIP Conf. Softw. Archit., pp. 231–240, 2016.

[4] W. Dai, V. Vyatkin, S. Member, J. H. Christensen, and V. N. Dubinin, “Bridging Service-Oriented Architecture and IEC 61499 for Flexibility and Interoperability,” vol. 11, no. 3, pp. 771–781, 2015.

[5] W. Deng, S. Liu, and J. Liu, “Automation Testing Process Modeling Method of SOA-based Isomerous Software,” pp. 129–132, 2009.

[6] A. T. Endo, M. Linschulte, A. Da Silva Simão, and S. Do Rocio Senger De Souza, “Event- and coverage-based testing of web services,” SSIRI-C 2010 - 4th IEEE Int. Conf. Secur. Softw. Integr. Reliab. Improv. Companion, pp. 62–69, 2010.

[7] S. Herbold et al., “The MIDAS Cloud Platform for Testing SOA Applications,” 2015 IEEE 8th Int. Conf. Softw. Testing, Verif. Valid., pp. 1–8, 2015.

[8] S. Jehan, I. Pill, and F. Wotawa, “SOA grey box testing - A constraint-based approach,” Proc. - IEEE 6th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2013, pp. 232–237, 2013.

[9] T. S. W. Journal, “Retracted: An Automatic Web Service Composition Framework Using QoS-Based Web Service Ranking Algorithm.” Scientific World Journal., vol. 2016, p. 6902846, 2016.

[10] Z. J. Li, H. F. Tan, H. H. Liu, J. Zhu, and N. M. Mitsumori, “Business-process-driven gray-box SOA testing,” vol. 47, no. 3, pp. 457–472, 2008.

[11] Y. Liu, K. Wang, W. Wei, B. Zhang, and H. Zhong, “User-session-based test cases optimization method based on agglutinate hierarchy clustering,” Proc. - 2011 IEEE Int. Conf. Internet Things Cyber, Phys. Soc. Comput. iThings/CPSCom 2011, pp. 413–418, 2011.

[12] S. Noikajana and T. Suwannasart, “Web Service Test Case Generation Based on Decision Table,” Eighth Int. Conf. Qual. Softw., pp. 321–326, 2008.

[13] M. Palacios, J. García-Fanjul, and J. Tuya, “Testing in Service Oriented Architectures with dynamic binding: A mapping study,” Inf. Softw. Technol., vol. 53, no. 3, pp. 171–189, 2011.

[14] X. Peng and L. Lu, “A New Approach for Session-based Test Case Generation by GA.pdf,” pp. 91–96, 2011.

[15] Z. Sen, H. Shuangxi, and F. A. N. Yushun, “Service-Oriented Enterprise Network Performance Analysis *,” vol. 14, no. 4, pp. 492–503, 2009.

[16] H. M. Sneed and C. Verhoef, “Measuring test coverage of SoA services,” 2015 IEEE 9th Int. Symp. Maint. Evol. Serv. Syst. Cloud-Based Environ. MESOCA 2015 - Proc., pp. 59–66, 2015.

[17] M. Weir and R. Kulak, “Testing Environment,” 2013.

[18] F. Wotawa et al., “Fifty shades of grey in SOA testing,” Proc. - IEEE 6th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2013, pp. 154–157, 2013.

[19] J. Zhang, R. G. Qiu, and S. Member, “Fault Injection-based Test Case Generation for SOA-oriented Software,” pp. 1070–1078.

[20] Y. Zhang, S. Member, Z. Zheng, and M. R. Lyu, “An Online Performance Prediction Framework for Service-Oriented Systems,” vol. 44, no. 9, pp. 1169–1181, 2014.

[21] C. S. Wu and Y. T. Lee, “Automatic SaaS test cases generation based on SOA in the cloud service,” CloudCom 2012 - Proc. 2012 4th IEEE Int. Conf. Cloud Comput. Technol. Sci., pp. 349–354, 2012.

[22] L. I. Dancheng, J. I. N. Weipeng, L. I. U. Guoqi, S. Xiang, Y. E. Tengqi, and Z. H. U. Zhiliang, “The Research on Automatic Generation of Testing Data for Web Service,” Structure, pp. 0–3, 2010.

[23] D. Petrova-Antonova, K. Kuncheva, and S. Ilieva, “Automatic generation of test data for xml schema-based testing of web services,” ICSOFT-EA 2015 - 10th Int. Conf. Softw. Eng. Appl. Proceedings; Part 10th Int. Jt. Conf. Softw. Technol. ICSOFT 2015, pp. 277–284, 2015.

[24] X. Dawei, J. Liqiu, X. Xinpeng, and W. Yuhang, “Web Application Automatic Testing Solution,” pp. 1183–1187, 2016.

[25] Y.-H. T. Y.-H. Tung, S.-S. T. S.-S. Tseng, T.-J. L. T.-J. Lee, and J.-F. W. J.-F. Weng, “A Novel Approach to Automatic Test Case Generation for Web Applications,” Qual. Softw. QSIC 2010 10th Int. Conf., pp. 399–404, 2010.

[26] S. Rastogi, “A QoS based methodology for multiple fault handling in SOA,” pp. 300–304, 2016.

[27] G. Katsikogiannis, S. Mitropoulos, and C. Douligeris, “An Identity and Access Management approach for SOA,” pp. 1–6, 2016.

[28] S. Jehan, I. Pill, and F. Wotawa, “SOA Testing Via Random Paths in BPEL Models,” pp. 260–263, 2014.

[29] C. Engineering, “A Novel Testing Framework for SOA Based Services,” pp. 1–4, 2014.

[30] B. Wen and Z. Luo, “Active Customization-oriented Adaptive SOA with Runtime Exception Handling,” 2016.

APPENDIX

SOA	- Service Oriented Architecture
WSDL	- Web Service Definition Language
UDDI	- Universal Description, Discovery and Integration
API	- Application Program Interface
ETL	- Extract, Transform and Load
WS	- Web Service
SUT	- System/Service Under Test
MDA	- Model Driven Architecture
AHP	- Analytic Hierarchical Process
SOE	- Service Oriented Enterprise
SOEN	- Service Oriented Enterprise Network
PAO	- Performance Analysis and Optimization
UPWSR	- User Preference based Web Service Ranking
QoS	- Quality of Services
OPred	- Online performance prediction framework
BPEL	- Business Process Execution Language
NPI	- New Product Introduction
MIDAS	- Model & Interface Driven Automated testing of Service architectures
UML	- Unified Modeling Language
ABT	- Architectural Based Testing
TePSEx	- Test Prioritization and Selection Explorer

USCHC	- User Session Clustering based on Hierarchical Clustering
US-RDG	- User Session Data with Request Dependence Graph
TAD	- Testing by Automatically generate Decision table
QAWSC	- QoS Aware automatic Web Service Composition
DOM	- Document Object Model
GA	- Genetic Algorithm
SVG	- Scalable Vector Graphics
IDE	- Integrated Development Environment
SDK	- Software Development Toolkit
Selenium RC	- Selenium Remote Control