Journal of Advanced Research in Wireless, Mobile and Telecommunication
Volume 6, Print Issue 1 - 2023, Pg. No. 20-26
Peer Reviewed Journal

Research Article

Organizing Web Service Program Elements in Accordance with Their Requirements for Testing

Amit Kumar Singh[1], Agastya Singh[2]

[1]Assistant Professor CSE department.
[2]Student CSE department, Chandigarh University.

I N F O

Corresponding Author:
Amit Kumar Singh, Assistant Professor CSE department.
E-mail Id:
singhdramitkumar916@gmail.com
Orcid Id:
https://orcid.org/0009-0009-3827-1715
How to cite this article:
Singh AK, Singh A. Organizing Web Service Program Elements in Accordance with Their Requirements for Testing. *J Adv Res Wire Mob Telecom* 2023; 6(1): 20-26.

Date of Submission: 2023-03-12
Date of Acceptance: 2023-04-22

A B S T R A C T

Testing is crucial for identifying problems in software products using Webservices. However, it is not possible to guarantee a software is free of errors, as it is impossible to test every possible combination of inputs and outputs. Testing is an effective way to reduce system defects and increase users' trust in a built system. Residual errors in an object-oriented program can be scattered throughout the code. To organize the components of a program, time spent testing can be divided into components with the highest failure rate.

This approach is often used in coding, debugging, designing test cases, maintaining software. Web services are built on the interaction between a service provider and a service requestor. Discovery agencies are not required, but they can occur in some situations. The web services architecture portrays discovery agencies as a cloud, providing access to a vast universe of data and query services. Metadata is important in relation to online services, as it helps describe the interaction between a requestor and a provider.Keywords: Web Services, Testing, software testing, test case, prioritization, object-oriented programming are some of the keywords that are important here.

Keywords: Web Services, Testing, Software Testing, Test Case, Prioritization, Object-Oriented Programming Are Some Of The Keywords That Are Important Here

Introduction

The term "Web Services" may have a variety of meanings depending on whom you ask. In the end, there will be a set of standards that will let us carry out activities that we were unable to carry out in the past; nevertheless, in the meanwhile, many individuals and businesses will approach these standards from a variety of vantage points and with a variety of expectations. In the years 2001 and 2002, "Web Services" has also been a term that has been used often and is said to be one of the emerging technologies with the most potential. These are the recurring ideas.

A shift away from seeing the web as a place for storing information that is mostly static and toward viewing it as one in which interactions are the key paradigm.

The usage of hypertext transfer protocol (HTTP), extensible markup language (XML), other web architectural standards as the fundamental building elements

A customary concentration on enterprise-wide and inter-enterprise activities.

The phrase "Internet Services" would be a better fit for what are now being referred to as "Web Services," which has the word "Web" in its name. The usage of the HTTP and XML

is already in use as a well-understood and well-debugged set of protocols that enable the Web, so it makes sense to reuse them in delivering remote activities and those things that are associated with them. This is where the Web originates from. The third element is what differentiates the requirements for a web service so drastically from those of a local RPC system. The fact that data is traded for commercial reasons as well as between other social bodies indicates that accountability, in addition to dependable transmission, is essential.

The providers of software perceive web services as a method to repackage already existing capabilities in a manner that makes them compatible with systems developed by other companies.

The trust environments whether intranet, business-to-business, or business-to-consumer, etc. are the ones that determine the security needs for online services. For business-to-business transactions, reliability is not enough; responsibility is also required.

In Survey[1,2] a number of different strategies for prioritizing and choosing test cases and affecting nodes have been presented, these strategies have also been experimentally explored. the state of knowledge in these fields now. According to the data, the majority of the techniques that are now in use apply structural or functional coverage criteria to the source code that is executed during test cases. In this particular respect, our thoughts are distinct from those presented in the aforementioned body of work.

Even while testing software is a costly operation in and of itself, the cost of releasing software without testing it might be considerably greater. This is particularly true if the program impacts the safety of humans. Testing software encompasses any activity that examines the functionality of a program or system to determine whether or not it performs the tasks for which it was designed and whether or not it does so effectively. Software testing is still often seen as an art form, despite the fact that software principles are essential to the quality of software and are used frequently by testers as well as programmers. Testing software may be challenging due to the complexity of the program being tested. We won't be able to do a thorough examination of the software if it isn't too difficult. The process of testing includes a number of steps, one of which is called debugging. It is possible to test anything in order to determine whether or not it is dependable, whether or not the quality is satisfactory, or in order to verify and authenticate something. The term "testing" may also refer to a more comprehensive method of "measuring" something. Testing something to ensure that it is right and testing something to ensure that it is reliable are two of the most significant forms of testing. When it comes to software testing, you are going to have to make a choice between money, time, quality.

When errors are discovered and repaired early on in the software life cycle, a lower total cost is incurred as a result of these actions. Software applications are becoming more integrated into the fabric of our everyday life. Software companies are under a huge amount of pressure to provide solutions that are incredibly dependable and have very little room for mistakes. Testing is often performed on several levels for software products in order to uncover any and all bugs. the application running on the machine. It is difficult to guarantee that a software product is devoid of errors for the vast majority of real systems, even after the testing procedure has been completed successfully and passed with flying colors. This issue is brought about by the fact that the majority of software products have a relatively broad input data domain to work with.

In addition, any endeavor to produce a software product is subject to limitations placed on both its timeline and its financial resources. You cannot thoroughly test a piece of software by providing it with every conceivable value for the input data since it is not feasible. At the present, testing consumes an average of fifty percent of both the time and the money spent on development.[10]

Therefore, it is quite improbable that the total quantity of testing that is done will increase much more. In order to completely test each component of the software product, conventional testing approaches are used. This indicates that the software flaws are dispersed across the program in an equal manner. However, when bugs are present in some sections, they generate issues that are more severe and occur more often than they do in other sections. For instance, if one statement produces vital information that many other claims rely on, then an error in one statement might have repercussions for a great number of other assertions. As a result, one of our primary objectives is to determine which aspects of a program are the most crucial and should be subjected to more stringent testing. We argue that the level of importance and seriousness of an element is measured by how much of an effect it has. We devised a system that enables us to assess both the significance of a statement and the significance of a procedure. We are able to determine the significance of a class by using these two parameters. The process of developing, creating, testing, maintaining software may all benefit from coding that is characterized. In order to demonstrate how the code operates in the middle, we make advantage of the Extended System Dependent Graph. It would seem that there is no way to do the testing task in a more efficient manner. Given that testing is essentially a sample, it is critical to make decisions on what should be tested, what should not

be tested, the amount of work that should be done. The vast majority of systematic testing approaches, including white box testing and black box testing approaches like equivalence partitioning, boundary value analysis, cause-effect graphing, produce an excessively large number of test cases.[7]

Motivation for Our Work

Because computers and software are so regularly used in essential applications, even the smallest error might have very negative repercussions. It's possible that bugs may cause enormous losses. Errors in critical computer systems have resulted in the destruction of airplanes, the failure of the systems that operate the space shuttle, the suspension of stock market transactions. An insect is lethal. Bugs have been known to trigger catastrophes.

In a culture in which everything is computerized, the reliability and quality of software are of the utmost importance. This may be accomplished only via the administration of exhaustive tests. The current generation is one that lives on the internet and makes use of IoT devices. As a result, we make use of firmware, which is essential in fields ranging from medicine to agriculture. Everywhere we go, we use software for analyzing data and making decisions; hence, dependable software is required each and every hour.[9,11,12,13,14]

Objective of Our Work

The effect that different aspects of the programming have on the system's dependability as a whole might vary greatly from case to case. Therefore, the effect of the various components must be explained, the components with the highest degree of trustworthiness must be subjected to comprehensive testing. to locate previously unknown flaws by using the standards as a guide. Before distributing the product or making it available to the public, check to see that it is free of any problems. "Your Satisfaction Is 100% Guaranteed."

Our research is focused mostly on developing efficient algorithms with the intention of determining how a statement, a method, a class influence an object-oriented program. This is the fundamental objective of our study. Finding and fixing software bugs as early as possible in the process of developing software is one of the primaries focuses of our work. This is done to ensure that the product in question does not experience frequent or severe failures.

Our work is focused on locating and isolating software flaws at the earliest feasible stages of the software development cycle in order to reduce the frequency and severity of software failures. This is the overarching aim of our efforts. Our goal is to cut down on the number of times a system fails tests while maintaining the existing testing budget. There are two components of this that are included in the test plan.

1. The most important aspects of the program should be checked for functionality first.
2. Perform exhaustive testing on the portions of the code where the existence of even a single bug raises the probability of an unsuccessful outcome.

The first one may be figured out by having a look at the function's visibility, the use frequency, the possible failure costs. Regarding the second one, we have devised a method to locate the essential components of the source code [2], which uses an algorithm. The topic of how to organize test cases has been the subject of a significant amount of research.[4,5] However, there hasn't been much discussion regarding the work that was done to determine which aspects of the code are the most significant.

Prior to the creation of test cases, there haven't been a lot of studies done on how to make testing better. In the context of this topic, one of the areas of research that may be pursued is the creation of software that can be evaluated. In this work, an attempt is made to describe how to develop software that is not only simple to test but also, ideally, less expensive to test.

The preconditions, postconditions, assertions that apply to each module are decided upon at the design phase of the development life cycle for this particular piece of work. The second component of pre-testing is determining the order of testing priorities for the code. I proposed a method for determining priorities that would place the most significant aspects of the code that need testing at the front of the list and highlight them to draw attention to their significance. Increasing the amount of code that is covered by this approach would not take much time. Code coverage is a measure that indicates what percentage of an application's source code is executed when the application's unit tests are executed. In principle, the more efficiency with which the code operates, the greater the amount of ground it covers. However, having 100% code coverage does not guarantee that an application is free of bugs in actual use. The number of other objects in the given programming that utilize an item either directly or indirectly may be used as a proxy for determining how strong that object is. By the value that it sends back, a method on an object may sometimes instruct other objects in a program to carry out a certain action. Therefore, the power of an object is proportional to the number of other objects in the program that rely on it for either control or data, this dependence might be direct or indirect. To begin, we will build an intermediate representation using the source code. This representation is called a "control dependency graph." After that, we put

the data and the software through its paces by executing the program. We demonstrate our suggested method, which has the ability to determine the influence value of any item and get the dynamic slice of any object at any moment in the execution of the process. To ensure that the software testing process produces high-quality results while staying within its financial constraints, one of the things we do is implement prioritized testing. In this part, the primary emphasis is placed on research findings that were presented in the context of prioritizing approaches, during the process of test case selection in test suites, or prior to the construction of test cases.[2]

Control Flow

The control flow graph (CFG) is a programmable representation that may be used as a step in various different optimization code transformations, such as the elimination of similar subexpressions, the propagation of copies, the transfer of loop invariant code.

Through the use of the Program Control Reliance Graph, we are able to determine the dependence and which factors impact others. This is one of the strategies that we may utilize to identify the reliance that we have on one other and impact one another.

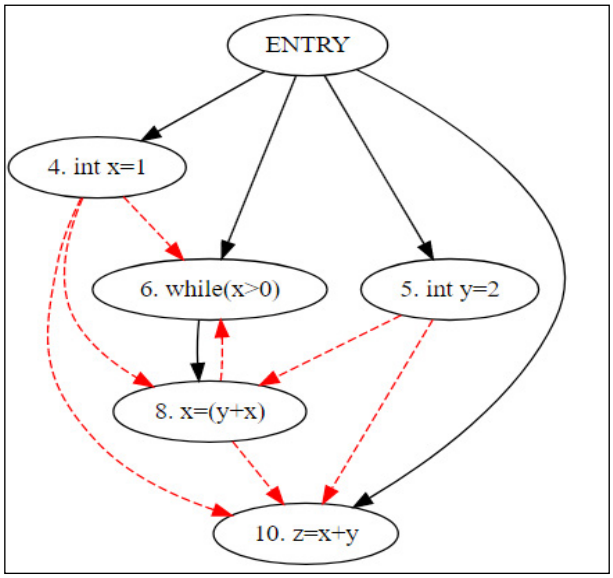

Figure 1.Program Control Dependence Graph

Profiling

When we profile a piece of software, we are able to determine where the software spent its time and which functions it called when those functions were operational. This data may help to identify the portions of the software that are operating more slowly than planned and may benefit from having their code rewritten in order to speed up performance. In addition, it might disclose whether functions are being utilized more often or less frequently than you had expected using them. You may be able to see bugs that you would have missed in any other circumstance. Earlier work could have required a significant amount of time (a month or a year), depending on the size of the test suite and the amount of time it takes to execute each test case individually.

However, in order to detect flaws more often, testers might rearrange the order of the test cases provided they make use of an efficient prioritizing approach. In the technique that was presented in this study, a prioritization algorithm was utilized to determine the order in which the test cases were executed. During the time we were confined, we were instructed to look for as many errors as we possibly could. In this part, we will discuss how we choose the order of programmed components depending on how thoroughly each of those elements should be evaluated. To begin, we will describe the process that we want to use to complete the tasks. After that, we will demonstrate how we compute the impact of the statement, the effect of the method, the effect of the class.

A Critical Analysis of Our Working Procedures

The classes of object-oriented software are constructed using code. Variables and methods are presumed to be present in each and every class. The influence wielded by a class is built up of the accumulated impacts of all of the class's individual aspects. As a consequence of this, we evaluate the consequences of every statement, if any of those statements call a method, we also evaluate the consequences of that method. Our strategy makes no use of variable values and instead relies on doing a static examination of the source code. As a direct consequence of this, it has difficulty processing recursive function calls and loops. The influence of class is equivalent to the total of the fluences of all relevant assertions and methods. A class's impact may be determined statically using this method. The effect of a comment is covered first, followed by the impact of a method, finally, the influence of a class is covered. The results of one statement in a program might be contingent on the results of another statement in the same program. When the impact is greater, the remark takes on a more significant significance. The statement's level of impact is determined by the number of other statements in the provided program that make direct or indirect use of the variable in question. In light of the fact that there is no call vertex, we provide a measure for determining influence. If a statement is labeled as a vertex, its influence will be computed separately using the method metric's influence. This influence will then be added to the desired statement's impact to establish the desired statement's total influence. The proportion of the statement's effect that may be attributed to each of the following elements is determined:

The total number of nodes that were affected.

_____ × 100

The total number of vertices and edges that make up the graph.

Algorithm

Input: The code for the programmed statement as well as the code.

The result or consequence of what has been said.

StmtInfluence(statement) 1. Create the ESDG for the application in a manner that is static.

2. The for statement should iterate over all of the edges that rely on it and mark them when they are completed.

3. Repeat step 2 for each of the nodes that were previously indicated until there are no longer any edges that rely on the nodes.

4. Determine the significance of a marked node by determining whether or not it is a call vertex, then utilize the MethodInfluence tool (call vertex

5. Determine the effect using an equation after counting the nodes that have been tagged (1).

6. Stop.

}

The Effects of a Procedure

When one method in a program achieves a result, that result has an impact on the other methods and statements in the program. It's possible for one method in the program to have an impact on another method or line in the code. If the approach is going to have a more significant impact, then you should prioritize it more. The "impact of a method" is the name of a measure that we have developed for object-oriented programming.

The significance of a method may be evaluated based on the number of other statements and methods inside a particular program that make use of the method's findings, either directly or indirectly.

If the method whose impact we are interested in is one that calls other methods, then the overall influence of the method will be equal to the sum of the effect of the method itself and the influence of any other methods that the method calls.

You may determine what proportion of an impact an approach has by:

The total number of nodes that were affected.

_____ × 100

The total number of vertices and edges that make up the graph.

Algorithm

Both the name of the technique used by a programmed and the name of the method used by the programmed are required as input.

What the technique really accomplished.

Method

The power of influence (call vertex)

1. The ESDG should be included in the program.

2. After you have traversed all of the edges, make a note of the ones you have visited before to serve as the beginning point for the procedure.

3. After going over all of the edges of each node you visit, mark the node that it belongs to as visited. If the node in question is not a call-vertex node, mark it as influenced if you haven't done so previously.

4. Determine whether each visited node also serves as a call vertex. In such case, continue down the call's edge and perform the actions listed below:

(a) If the node that follows is polymorphic, call the vertex, traverse over each polymorphic edge, add the matching node to a queue. Q

b) If it does not, include the node in the Q list.

5. Take out the nodes from the Q. Make a note of the damaged node, then carry call vertex–4 once again for that node.

6. You must continue to do Step 5 until there is nothing left in Q.

7. After each affected node has been marked, go to the next node in the chain and mark each of its edges as influenced.

if it hasn't been carried out as of yet.

8. Determine what the strategy will achieve by using the phrase to help you (2).

9. Stop.

}

The Power Wielded by a Social Group

The aggregate of the effects produced by all of the other components of a particular program that make use of the results produced by a class is what is meant to be understood as the influence of that class. We keep track of how many nodes are impacted by this. It is possible to determine the effect of nodes that include function calls by using the measure known as Method Influence(call vertex). To determine the effect that each and every other statement has, the StmtInfluence (statement) measure is used.

The following are examples of the impact that a class has:

The total number of nodes that were affected.

_____ × 100

The total number of vertices and edges that make up the graph.

Algorithm

A sample that has been coded and the name of the class are both inputs.

The results of the lesson and how things were altered.

Class Influence(class name)

{

1. Create the ESDG for the application in a manner that is static.

2. While moving through the class entry vertex, go to each individual member of the class, nd mark them as visited as you reach them.

3. For each node that has already been visited, it is necessary to traverse each of that node's edges and then mark the visited status of the associated node. Mark the node as influenced if it hasn't already been done so if the node in question is not a call-vertex.

4. Determine whether or not each previously visited node is a call vertex. If this is the case, you may use MethodInfluence to determine how significant this statement is (call vertex).

5. For each node that has been tagged as being impacted, it is now necessary to go through all of that node's edges and mark each one as being influenced if this step has not yet been completed.

6. Determine the impact of the specified group by using the phrase to do so.[3]

7. Stop.

}

We have chosen an example program that has 134 nodes for the purpose of doing experimental research.

Table 1:Experimental Studies using Web Services

The sum of all nodes that were impacted	The Program's Total Nodes	Relevance percentage
96	134	71.64
22	134	16.41
8	134	5.9
7	134	5.2
1	134	.7

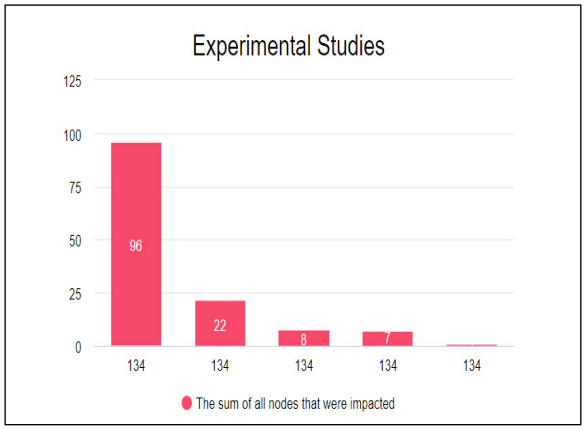

Figure 2.Experimental Studies using Web Services

We have identified the node with the greatest amount of influence, because this node has an effect on the highest node, we are required to test it first. In order to do this, we need to locate the statement, method, class that has the biggest impact and test appropriately. The following is a list of the several goals that may be helped by giving testing more priority:

- Errors with a high potential for damage being uncovered early on in the testing process
- To raise the probability that certain code alterations may result in errors at an earlier stage in the testing process
- In order to enhance the frequency with which code that is capable of being covered is covered
- To make a system more reliable by increasing its stability

Conclusion

We created a metric for the programme that assesses the importance of its many components. Which elements of the programme are impacted more than others can be determined depending on the effect. As a result, the components that have a higher impact are more important, the addition of these elements will raise the possibility of a programme malfunction. As a result, the intended metrics are very helpful in identifying which parts are the most important, they also warn us that we must proceed with utmost caution when building the parts that will have the biggest impact throughout the software development cycle. This indicates that compared to testing the components of greater significance, testing the less important components may be accomplished with a smaller number of test cases. This frees up time that could be used to test the more important components. It is reliant on a program's static analysis. This is crucial for creating and organising test scenarios. Knowing how each component of the programme affects the other is useful. As a result, we can conduct rigorous testing with more dependable components at our disposal.

References

1. Prioritization of Program Elements Based on Their Testing Requirements, Computer Science and Engineering, Kanhaiya Lal Kumawat, National Institute of Technology Rourkela (2009)
2. Reliability Improvement Based on Prioritization of Source Code, Mitrabinda Ray and Durga Prasad Mohapatra, Department of Computer Science and Engineering, National Institute of Technology Rourkela (2010)
3. Danjun Zhu, Gangtian Liu, "Deep Neural Network Model-Assisted Reconstruction and Optimization of Chinese Characters in Product Packaging Graphic Patterns and Visual Styling Design", Scientific Programming, vol. 2022, Article ID 1219802, 12 pages, 2022. https://doi.org/10.1155/2022/1219802
4. Guertin SM. Board Level Proton Testing Book of Knowledge for NASA Electronic Parts and Packaging Program. Accessed: Oct. 2018.
5. Guertin SM. Lessons and recommendations for board-level testing with proton in Process Small Satell Conf Logan UT USA 2018.
6. A. Coronetti et al., "Radiation Hardness Assurance Through System-Level Testing: Risk Acceptance, Facility Requirements, Test Methodology, Data Exploitation," in IEEE Transactions on Nuclear Science, vol. 68, no. 5, pp. 958-969, May 2021, doi: 10.1109/TNS.2021.3061197.
7. Jordi Roglans-Ribas, Kemal Pasamehmetoglu & Thomas J. O'Connor (2022): The Versatile Test Reactor Project: Mission, Requirements, Description, Nuclear Science and Engineering, DOI: 10.1080/00295639.2022.2035183
8. E. W. Dijkstra. 2022. On the Reliability of Programs. Edsger Wybe Dijkstra: His Life, Work, Legacy (1st ed.). Association for Computing Machinery, New York, NY, USA, 359–370. https://doi.org/10.1145/3544585.3544608
9. Srivastava, A., Kumar, A. (2022). A Review of Network Optimization on the Internet of Things. In: Saini, H.S., Sayal, R., Govardhan, A., Buyya, R. (eds) Innovations in Computer Science and Engineering. Lecture Notes in Networks and Systems, vol 385. Springer, Singapore. https://doi.org/10.1007/978-981-16-8987-1_6
10. N. Srivastava, U. Kumar and P. Singh (2021) Software and Performance Testing Tools. Journal of Informatics Electrical and Electronics Engineering, Vol. 02, Iss. 01, S. No. 001, pp. 1-12, 2021. https://doi.org/10.54060/JIEEE/002.01.001
11. Kumar, G., Singh, G., Bhatanagar, V., & Jyoti, K. (2019). SCARY DARK SIDE OF ARTIFICIAL INTELLIGENCE: A PERILOUS CONTRIVANCE TO MANKIND. Humanities & Social Sciences Reviews, 7(5), 1097-1103. https://doi.org/10.18510/hssr.2019.75146
12. Gupta, R., Bhatnagar, V., Kumar, G., & Singh, G. (2022). Selection of suitable IoT-based End-devices, tools, technologies for implementing Smart Farming: Issues and Challenges. International Journal of Students' Research in Technology & Management, 10(2), 28-35. https://doi.org/10.18510/ijsrtm.2022.1024
13. Singh, G. and Yogi, K.K. (2022a). Internet of Things-Based Devices/Robots in Agriculture 4.0. In: Karrupusamy P., Balas V.E., Shi Y. (eds) Sustainable Communication Networks andApplication. Lecture Notes on Data Engineering and Communications Technologies, vol 93. Springer, Singapore. https://doi.org/10.1007/978-981-16-6605-6_6
14. Singh, G. and Yogi, K.K. (2022b). Usage of Internet of Things Based Devices in Smart Agriculture for Monitoring the field and Pest Control. 2022 IEEE Delhi Section Conference (DELCON), pp.1-8. https://doi.org/10.1109/DELCON54057.2022.9753021