University of Ibadan
Journal of Science and Logics in ICT Research

Qualitative Comparative Analysis of Software Integration Testing Techniques

Akinsola, J. E. T.[1], Adeagbo, M. A.[1], Abdul-Yakeen S. O. [1], Onipede, F. O.[1], Yusuf A. A.[2]
[1]*Department of Computer Sciences, First Technical University, KM 11, Ibadan – Lagos Expressway,* Ibadan, Nigeria
[2]*Department of Information and Communication Technology, Federal University of Petroleum Resources, Effurun, Nigeria*

*Corresponding Author's Email: akinsolajet@gmail.com

Abstract

Software testing is one of the core processes in software engineering. There are different types of testing which are unit testing, integration testing, system testing and acceptance testing. This study focusses on integration testing, its advantages, disadvantages, areas of application, guidelines, tools and approaches to attain a quality software. This study addressed the issue of determining which approach to use for a particular software project. Explicit characteristics of each approach were elucidated. Comparative analysis was carried out to determine the best and suitable approach of integration testing regarding a software project using twenty-six classification characteristics. TESSY, FitNesse, Rational Integration Tester are examples of automated tools for software integration testing discussed in this study. Unambiguous areas of application of protractor, rational integration tester and TESSY were discussed as well. The study therefore, recommends the use of machine learning paradigms for quantitative computation of the best software integration testing for further studies.

***Keywords*:** Acceptance testing, Integration testing, Rational integration tester, Software testing, System testing, Unit testing

1. INTRODUCTION

Development of a quality software is accomplished through a well-articulated software development life cycle model [1]. In order to achieve this goal, there is a need to follow an organized process. This process is known as software development life cycle, abbreviated as SDLC. The SDLC model includes some phases such as planning, analysis, requirement gathering, design, implementation, testing, among others [2]. Software performance evaluation and removal of ambiguity can be effectively done using formal methods [3]. Software testing is one of the phases of software development life cycle that comes after the implementation phase (coding).

Testing is done on a software product to assure its conformity with the user requirements. In the process of testing, software tester places the product under test by comparing its functionality with user functional requirements in the software requirements specification (SRS) document. It is an important aspect of software engineering that is used to ensure confidence over the product.

Software testing is also done to check for errors or defects in the software product that might occur during the implementation phase and also to proffer solution to the error. On time software testing helps to assure delivery of a good software product to the customer [4]. Test cases are required when checking the fitness of a product for use. These test cases contain the input, process and output. The system or software takes in data and processes it. The output of this is noted to know if it performs what it was developed for. There are different types of testing, which are unit testing, integration testing, system testing and acceptance testing. Three of these testing types are done by the development

Akinsola, J. E. T., Adeagbo, M. A., Abdul-Yakeen S. O., Onipede, F. O. and Yusuf A. A. (2022). Qualitative Comparative Analysis of Software Integration Testing Techniques, *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR),* Vol. 7 No. 2, pp. 67 – 82

team while the acceptance testing is done by the end user of the software.

Unit testing implies testing the modules independently, integration testing is to check the interface and interaction of the independently tested modules. System testing is for checking the combination of all the modules which forms a single system. Acceptance testing is done by the customer or by the end user of the software to confirm if truly the system meets their demand.

A software tester needs to be very careful while choosing any approach for integration testing. Making decision on which methodology to use to carry out integration testing on a software product needs to be carefully stated. Therefore, software engineering also provides some computer aided software engineering (CASE) tools to make the processes of software development easy and efficient. Software products always involve more than one module that interconnects together. Integration testing checks for the interaction of these modules and their interfaces. Among the tools for integration testing are Tessy, JMeter, Badboy, Protractor, Selenium software testing. These tools make integration testing to be done in a very much easy and efficient way.

Determination of appropriate integration testing techniques for software testing has been a herculean task due to the heterogenous nature of software codes and applications. Thus, proper characterization of the features in determining the suitable integration testing technique is desirable. The study characterized the four integration testing methodologies using qualitative comparison based on twenty-six features identified as shown in Table 2. Qualitative comparative analysis was conducted to categorize the important features for determining the right software integration testing approach. The characterized features will help software testers in choosing the right software testing approach and making effective decision for assuring quality of software products.

2. LITERATURE REVIEW

Software testing is one of the phases of software development that should not be underestimated. It contributes largely to the development of a quality software product. Testing a software product makes the developer to be confident that the right product has been built. One of the factors that is used in measuring the quality of a software is the usability. A quality software must be user friendly, simple and easy to use. Integration testing helps to confirm if the product is user friendly by testing the interfaces of different modules in the software.

In the research conducted by Sawant *et al.* [5], detailed description of software testing, the goals as well as principles of software testing in addition to software testing needs were considered. The best methodologies, practices, principles and standards essential for optimal software testing have been discovered. This makes the problem of software testing to be less rigorous with appropriate implementation of formal methods as exemplified by Akinsola *et al.* [3]. Anyone involves in testing should get familiarized with basic goals, limitations, principles as well as concepts of software testing in order to carry out effective and efficient software testing.

Tahvili [6] provides some methods for carrying out optimized test which are selection of test, prioritization of test as well as scheduling test execution. In the research, provision of a more effective and efficient way of carrying out integration testing process manually was outlined. To achieve these three phases, there are essential processes which must be provided; which are test cases properties for automatic measurement, test cases prioritization and scheduling for automatic execution in terms of a support system decided and lastly evaluation of the efficient approach proposed empirically.

Improved techniques such as automated testing and various metrics for software testing were discussed in the research carried out by Arumugam [7] for ensuring better quality assurance. The automated testing carried out are Test Driven Development (TDD) while the metrics are Prioritization Metrics as well as Process Quality Metrics. According to the study on integration testing in a software product line engineering by [8], automated integration testing method that can be used in engineering domain is called software product line platform. This method is proposed due to its ability of test cases and scenarios generation for integration testing.

Rastogi [9] discusses several most common software testing approaches with a brief summary of the essential requirements for software testing. Furthermore, comparative analysis of the software testing types was done in order to know the best one and how to find a particular error based on the systematic investigation called comparative study on testing techniques of software. It was revealed from the study that agile testing is the most efficient and effective testing technique among all the software testing techniques implemented. This is largely due to its adaptability and predictability characteristics.

2.1 *Types of Integration Testing*

Integration testing is the process of verifying combination of individual software modules. Integration testing is a phase in software testing that checks for the agreement of software modules in accordance with the requirements in the SRS document. Unit testing must have been done before integration testing. If an error had been detected during the unit testing phase, integration testing cannot be done until the error is fixed [10]. Integration testing also aims at checking the interaction between modules of a software product. For example, checking the connection of a login page in web application to the dashboard. The major focus or objective of the integration testing phase is to track the error in the interface of software. The interface is what makes interaction between software modules possible [11], and this is essential in software testing. Once a system passes the integration testing, it convinces the tester that, there is a good communication among the modules and that guarantees good functionality of the system [12]. Figure 1 shows pictorial representation of software testing types.

The most efficient and reliable way to carry out integration testing in a large system is to test modules in pairs. Testing the interfaces of all modules of a large system at a time may be too complex and accurate result may not be obtained. After the first pair has been tested, this forms a partially integrated system. Other modules can then be included in an incremental version to achieve a reliable test [13]. Integration testing is done based on test plan that has been prepared and documented during the design phase in software development life cycle (SDLC). Integration testing is done to make sure that the system fulfils the functional requirement (that is, the main purpose of creating the system), performance requirement (how well do the sections interrelate) and reliability (customers' satisfaction) [14]. During the design phase there must be proper identification of the right SDLC model. Figure 2 shows diagrammatic representation of software integration testing in relation to modules testing.

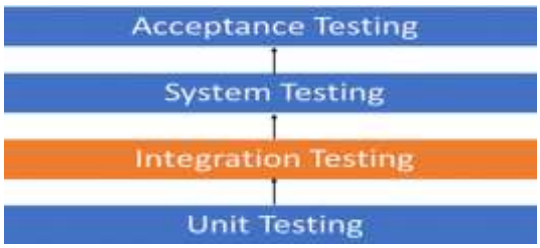

Figure 1: Types of software Testing

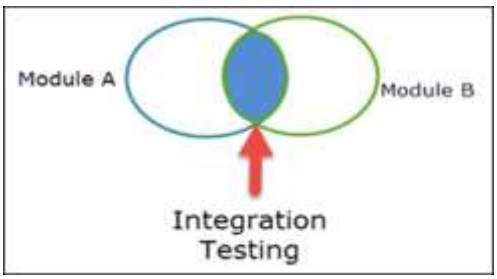

Figure 2: Software Integration Testing [15]

2.1.1 Advantages of Integration Testing

The following are the benefits of integration testing [16]:

i. **Production of a reliable system:** integration testing helps to mitigate the rate of getting failure, thereby ensuring the development of a reliable system.
ii. **Easy location of bugs:** integration testing makes it easier to find error as well as fixing it.
iii. **Reduction of defects:** defects can be easily reduced by correcting them after the collation of modules.
iv. **Early correction of defects:** integration testing gives the advantage of getting error on time and fixing them.
v. **Ensures basic testing:** integration testing helps to examine the operational features of the system.
vi. **Quick testing:** integration testing is faster than any other testing.
vii. **Modules interactions:** Integration testing assures proper interaction between modules.
viii. **Compatibility testing:** integration testing examines the suitability of hardware and software of a system.
ix. **Testing during development phase:** integration testing allows tester to begin testing the developed modules before the completion of the development phase.
x. **Proper integration functionality:** integration testing assures proper functioning of all combined components.

2.1.2 Disadvantages of integration testing

Given are the demerits of integration testing [17]**:**

i. **Difficulty of operation:** operating or performing integration testing requires expertise compared to system testing.
ii. **High resource utilization:** it involves an intensive use of resources while testing all the interfaces between linked modules.
iii. **Requires Studs and Drivers:** it needs studs as well as drivers' development, which if created wrongly can lead to inadequate testing.

2.2 Test Cases of Integration Testing

Login page, inbox as well as delete mails are the three assumption modules in an application. Individual modules' functionality is not the first thing to focus on while writing test cases of integration testing because unit testing would have tested the modules individually. The test cases must be properly determined for efficient integration testing. Wrong test case can lead to non-functional software product with the guise that the software is well integrated. Communication between modules is the major focus of integration testing. Due to the assumption above, the focus is how login page is connected to the inbox page as well as how inbox page is connected to the delete mails module [18]. Figure 3 shows how integration testing works. Integration test cases sample is shown in Table 1.

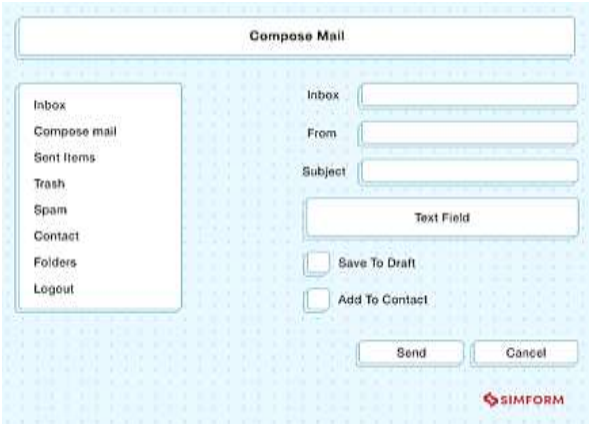

Figure 3: How Integration Testing Works [18]

Table 1: Test Cases Sample of Integration Testing [18]

ID of Test Case	Objective of Test Case	Description of Test Case	Result Expected
1.	Verification of integration testing between login as well as module of inbox.	Entering of credentials of login and then perform login.	Displaying of inbox.
2.	Integration between inbox as well as mails deleted for verification.	Selection of emails as well as clicking on delete button.	Disappearance of emails that has been deleted form inbox as well as appearance in trash box.

2.3 Integration Testing Entries

i. Modules that have been tested.
ii. All bugs to be corrected and ensure closure of those that are highly prioritized.
iii. All modules to be completely coded as well as successfully integrated.
iv. Documentation as well as sign off of integration scenarios, test cases and test plans.
v. Test environment that is required should be set up for integration testing.

2.4 Integration Testing Exit

i. Integration application that has been tested successfully.
ii. Documentation of test cases that has been executed.
iii. Corrected and closed highly prioritized errors.
iv. Documentations that are technical should be submitted and followed by notes released.

3. METHODOLOGY

Combination of mapping review and scoping review was used for typology of literature review methodology to ensure analytic frameworks construction in relation to integration testing techniques. Mapping review was done in relation to qualitative comparison with content analysis methods while scoping review focuses on the use of explicit inclusion and exclusion criteria. This is to ensure that there is no need to examine the quality or risk of bias of the primary studies that have been included. The goal of scoping review approach is to give a rough estimate of the size and extent of the existing research literature. It may be used to establish the quality and extent of research evidence, including ongoing research, in order to determine whether or not a comprehensive systematic review is warranted.

3.1 Integration Testing Methodologies

Integration testing has four different approaches or methodologies to enable an effective interaction and evaluation in a software project. To carry out integration testing, one of these approaches may be employed or combination of two or more techniques [19]. Integration testing techniques are done by combining several functional units and testing them for results investigation. Integration testing is divided into two classes as shown in Figure 4 which are incremental testing and big bang testing [16].

3.1.1 Big-Bang Methodology

Big-Bang type of integration testing is a straightforward method in software integration testing where all the modules are collated together and subjected to test. Big-bang checks for the correctness of the interface of all modules of the system at a time [19]. This approach can only be beneficial in case of a small project with less complexity and small number of modules. This is because error can easily be traced to a module compared to project of high complexity and larger number of modules. Error will be very much difficult to trace to any model in the situation of project with large number of modules [11].

Big-bang method requires that all the modules to be completed before the testing phase [20]. The diagrammatic representation of big bang is shown in Figure 5.

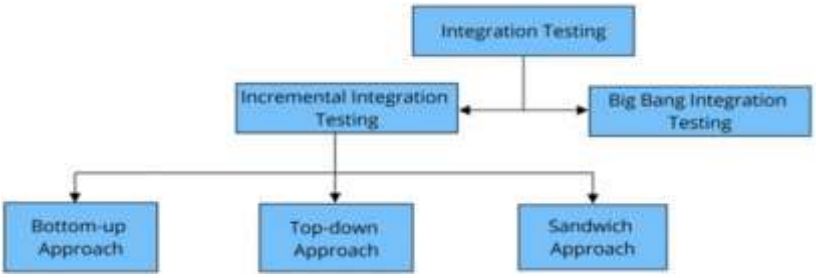

Figure 4: Types of Integration Testing Techniques

Figure 5: Big-Bang Integration Methodology [21]

a) Merits of Big-Bang Integration Testing Methodology

Big- Bang methodology has the following advantages:

i. **Small-sized project:** big-bang approach is appropriate for projects with small number of modules.
ii. **Saves time:** big-bang reduces time of testing each module incrementally by subjecting all modules to test at a stretch.
iii. **Minor planning:** very little testing plan is needed in this approach.
iv. **Component testing:** it assures complete module testing

b) Demerits of Big-Bang Integration Testing Methodology

The following are the shortcomings of big-bang integration testing methodology [22]:

i. **Error tracking**: one of the shortcomings of big-bang approach is that errors detected from the system cannot be easily traced to the module.
ii. **Project Size**: big-bang is not the best testing approach for a system with large number of modules.
iii. **Test Re-working:** once an error is detected, it requires splitting all the modules to find out the main source of the error
iv. **Risk:** it involves high rate of risk.
v. **Reliability:** this testing technique is not that dependable due to the level of risk involved.

c) When to Use Big-Bang Integration Testing Methodology

Big-bang methodology can be used under the following conditions:

i. When there is short time constraint.
ii. When the project is of small size.
iii. It can be used when there is little or less testing plan.
iv. When the project is of small risk.
v. When there is little cost of planning for the project.

3.1.2 Incremental Testing

This type of integration testing technique is done by integration of two or more modules that are logically related to each other as well as testing the application to ensure proper functioning. Then, the other modules are incrementally integrated and the process continues until all the modules that are logically related are tested as

well as successfully integrated [23]. In this type of integration testing, the relationship among modules or components that are dependent must be strong. For example, if two or more modules are chosen to validate the flow of data among them is working perfectly then, more modules or components are added as well as undergo testing again [24]. There are three approaches to incremental testing techniques, which are Bottom-up, Top-down and Sandwich or Mixed approach [23].

3.1.2.1. Bottom-Up Methodology

Bottom-up testing is one of the integration testing methodologies that follows an iterative process to check interaction of different components. This approach can also be classified as incremental technique. In this methodology, smaller modules are combined first and later on, more components are added to form a higher one. This approach make it easier and faster for the tester to carry out the testing within a short time range [25]. Modules are combined together in bottom-up approach. This combination is divided into two levels, the higher and lower levels. The interfaces of the modules in the lower level are first tested, then the modules at the higher level are also put to test [26]. In bottom-up approach, initial modules are tested together to form a module which will be tested against another integrated modules. All modules are finally integrated together to achieve the whole system. Figure 6 shows the diagrammatic representation of bottom-up approach.

a) When to Use Bottom-Up Methodology

In bottom-up methodology, unit testing of modules is carried out first. These modules are combined together to realize sub-systems. The subsystems' interface can then be tested. The following are the conditions that can warrant the usage of bottom-up integration methodology:

i. When the project involves many modules
ii. When the system can be split into sub-systems.
iii. When the project has medium to high risk.
iv. When reliability is of high demand.

b) Steps to Follow When using Bottom-Up Methodology

i. Combining or merging of low-level elements is known as clusters. They are known as builds that are responsible for certain subsidiary or secondary function performance of a software.
ii. Writing a control program for testing is essential.
iii. Testing is done on clusters that contains modules that are low-level or on entire clusters.
iv. Lastly, drivers are eliminated and clusters are integrated by upward moving from bottom to top in program structure with control flow help [27].

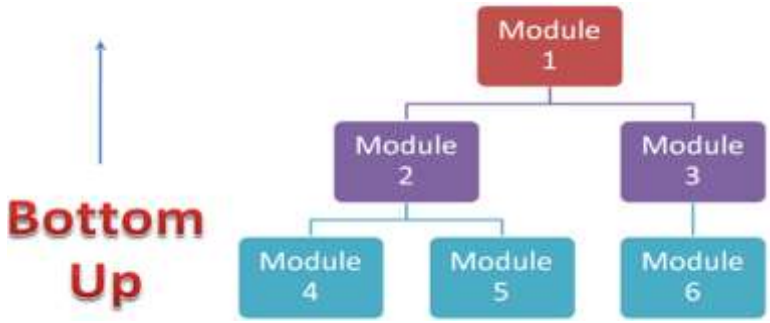

Figure 6: Pictorial Representation of Bottom-Up Approach [15]

c) *Merits of Bottom-Up Methodology*

The benefits of bottom-up approach are as follows [15], [25] :

i. **Easy tracking of errors**: errors can easily be noticed using bottom-up method
ii. **Size of project**: this methodology is very useful in large project size
iii. **Clarity of system's scope:** the entirety of the tested system is known to the testing team, which make it more convenient to test
iv. **Simplicity:** this approach makes the creation of test situation easier.
v. **Simultaneous testing:** This approach makes it possible for testing to be done simultaneously with the development.
vi. **Time utilization:** this approach ensure efficient management of time as testing is done on available unit unlike big-bang where it is compulsory that all modules are completed.
vii. **Reliability**: bottom-up technique assures high reliability because the test begins from the initial modules. All hidden errors can be tracked with this approach

d) *Demerits of Bottom-Up Integration Methodology*

The disadvantages of bottom-up integration approach are listed below [25], [28]:

i. **Complexity:** there is an increase in the complexity of the system as it comprises of too many components.
ii. **Rate of test case provision:** provision of test cases for this technique requires high fee.
iii. **Complete module testing:** Testing can only be satisfied when all modules are correctly tested.
iv. **Late generation of sample:** sample of the system is gotten late as this approach begin testing from individual component before getting to high level.
v. **Late data flow testing:** The flow of data is examined very late

3.1.2.2 Top-Down Methodology

Top-down technique is an approach that involves splitting the whole system into high-level and low-level. Unlike the bottom-up approach, where testing of modules in the lower level is done and gradually move to the high level which contain the main functionality of the system, Top-down approach begin testing from the higher level and combination of the lower-level modules are done repeatedly till the whole modules are combined. Top-down begins the combination of modules in the higher level division and later move to the lower level division to check the interaction between the modules [15].

While the higher-level testing is still in process, this technique makes use of artificial modules to work in place of the lower-level modules that are yet to be added. The artificial module is known as Stub. The introduced stubs are replaced with the main component when the process get to the lower-level phase. Perfect purpose of system is guaranteed with the finishing integration [29]. The higher-level division include the modules with the main functionality of the system. This is gotten by the prioritization of the system functionality. Figure 7 shows the diagrammatic representation of top-down approach.

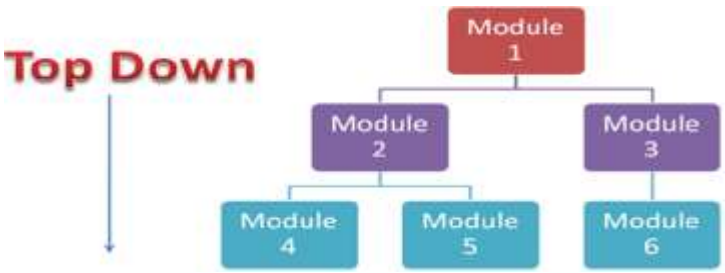

Figure 7: Top Down Approach to Software Integration Testing [15]

a) When to Use Top-down Integration Testing Methodology

The following are conditions that calls for the use of top-down:

i. When there is a need to get the sample of the system on time.
ii. When there is a need to test the system before the total completion of the modules.

b) *Top Down Integration Testing Methodology Strategies*

i. **Ensure that top-down testing approach match your bill:** other testing approaches must be kept in mind in order to set expectations that are realistic as well as time delivery achievement [30].
ii. **Creating and sticking to test plan:** global and detailed integration must be provided and the team must adhere to it for confusion avoidance as well as ensuring that everybody is not on different pages with the strategy of testing. Planned documents will serve as a documentation for focus areas, timelines as well as approaches to be used for the project. The team will navigate through structures of a system that is complicated with the use of detailed test plan in order to save effort and costs.
iii. **Left shifting and early testing:** launching of test activities in system development delivery cycle (SDDC) as early as possible will enhance reduction in cost associated with correcting and identifying errors.
iv. **Keeping stride with the constant delivery automation:** automation go along well with integration testing.
v. **Unit testing should not be mixed with integration testing:** environmental changes that affect the integration between modules are caused by failure in integration testing while unit testing exposes the errors in coding.
vi. **Various integration testing should be combined**: engagement model, project size as well as methodology can evolve and change and for successful continuation, the team must adapt to practices that guarantee reliable top - down software integration solutions. So, combinations of different testing strategies are required to arrive at the desirable outcome as well to a picture that is realistic.
vii. **Management of expectations:** expectations of accuracy should be well managed.

c) *Merits of Top-Down Methodology*

The following are the benefits of top-down integration approach [29]:

i. **Error detection:** errors in the system due to the design are easily discovered.
ii. **On time model of the system:** using top-down technique, it is convenient to acquire the model promptly.
iii. **Prioritization**: the main modules of the system are given higher priority and tested first.
iv. **Simple:** top-down integration testing approach is simple to implement.
v. **Correction of flaws:** correction can be made to the system very early since the main system is tested first.

d) *Demerits of Top-Down Integration Methodology*

The following are the disadvantages of the approach [29]:

i. **Too many stubs are required:** this approach uses many artificial modules at the lower level.
ii. **Complexity:** high number of stubs required makes this approach complex to implement.
iii. **Ineffectiveness of lower-level module:** there is high rate of the lower-level modules not being tested commendably.
iv. **Error tracking:** some errors that occur in the lower-level modules might not be traced.
v. **Restricted understandability:** the testing can only be done by the development team alone as the functionality might not be clear to another tester.
vi. **Testing restriction:** testing is restricted to only the module with the main functionality of the system.

3.1.2.3 *Sandwich, Hybrid or Mixed Methodology*

As the name implies, mixed methodology uses both the characteristics of top-down methodology and bottom-up methodology. This approach is used to address the drawback of the two approaches; where in top-down approach, the higher-level modules are tested first and in bottom-up approach, the lower-level modules are tested first. Mixed methodology involves testing both the higher level modules and the lower level modules simultaneously [12]. This approach is also referred to as the hybrid approach. This approach makes it possible to produce a good working version of the system in a short time. Unlike bottom-up and top-down approaches, where the system is shared into two levels, which are higher and lower levels. Hybrid approach divides the system into three levels based on the features of the system. The levels are high level, low level and the main level. The testing in this approach is focused on the middle level which is also the main level.

This approach is very useful in testing a software project with high complexity. It is very reliable when it is used in a large project due to the integration of two techniques.

Mixed approach does not only make use of the testing principles of the incremental approaches (that is, top-down and bottom-up); it also makes use of the non-incremental approach (big-bang approach). Big-bang approach is used at the middle division. Top-down approach is used to test the upper division of the software project down to the middle; while bottom-up is applied to test from the lower division up to the middle division where big-bang approach is then applied at the middle division to round up the integration testing process [31]. Pictorial representation of sandwich approach is shown in Figure 8.

a) When to Use Mixed Approach to Integration Testing

As the name implies, it is the combination of two integration testing approaches, top-down and bottom-up approach. It can be used in the following conditions:

i. When the size of the project is very large
ii. When the time for the delivery of the project is very short.
iii. When the estimated cost of the project is very high.
iv. When there is great demand for a quality software product.

b) Merits of Mixed Methodology

The following are the advantages of mixed methodology [32], [33]:

i. **Size of project:** mixed approach is very efficient for a project that is very large in size.
ii. **Parallelism:** this approach makes it possible to test using both top-down and bottom-up approaches simultaneously.
iii. **Saves time:** this approach uses less time when performing testing, because it uses the principles of two approaches at a time.
iv. **Test coverage:** there is high probability of covering all modules when using mixed approach for testing large software. This is because bottom-up tests the lower level while top-down approach tests from the top.

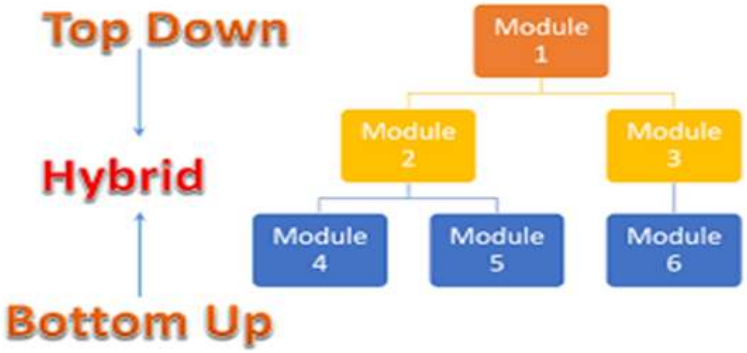

Figure 8: Mixed Approach to Integration Testing [15]

c) Demerits of Mixed Approach to Integration Testing

The following are the drawbacks of mixed approach to integration testing [33]:

i. **Cost:** mixed approach uses top-down and bottom-up approach which increases the cost associated with the software project.
ii. **Scope of the approach:** mixed approach cannot be used for software project with small size.
iii. **Complexity:** this approach is too complex due to the fact that the condition of both bottom-up and top-down approach must be fulfilled.
iv. **Late application:** this approach can only be used at the end of development phase, since all modules need to be completely developed before testing.

3.2 Comparative Analysis of Integration Testing Approaches

Software testing is one of the most essential phases in software development life cycle. Integration testing is one of the most important testing types that need to be carried out on a system to assure development of a quality software product. Integration testing is majorly aimed at tracking errors on interaction of different modules that may occur during the course of implementation. There are basically four approaches to integration testing which are big- bang approach, top-down approach, bottom-up approach and hybrid or mixed approach. Comparative analysis of these approaches is given in Table 2.

Table 2: Comparative Analysis of Big-Bang, Top-Down, Bottom-Up and Hybrid or Mixed Approach

S/N	Features	Software Integration Testing Approaches			
		Big-bang	**Top-down**	**Bottom-up**	**Hybrid**
1.	Project size	small	Large	Large	Large
2.	Project complexity	Low capability	Average capability	Average capability	High capability
3.	Project with risk	Not Suitable	Suitable	Suitable	Greatly suitable
4.	Project period	Short time	Average time	Average time	Short time
5.	Cost	Not expensive	Expensive	Expensive	More Expensive
6.	Project with low risk	Suitable	Less suitable	Less suitable	Not suitable
7.	Project with many sub-projects	Worst	Good	Good	Excellent
8.	Error discovery	Fast to track	Fast to track	A little bit late	Very fast
9.	Reliability	Not reliable	Low to high reliability	Less reliable	High reliability
10.	Test coverage	All modules are tested	Some modules might be missing	Some modules might be missing	All modules are covered
11.	Effectiveness	Less effective	Less efficient	Efficient	Highly efficient
12.	Testing while developing	Not allowed	Not allowed	Allowed	Not allowed
13.	Testing period	Late	Quite late	Early stage	At the end of development phase

S/N	Features	Software Integration Testing Approaches			
		Big-bang	**Top-down**	**Bottom-up**	**Hybrid**
14.	Prioritization of modules	No prioritization	Works based on prioritization	No prioritization	Little prioritization
15.	Testing basic functionality	Early	Early	Late	Quite early
16.	Module testing	Quite difficult	difficult	Easy	Easy
17.	Planning approach	Simple	Difficult	Simple	Difficult
18.	Incorporation of modules	Final level	At initial level	At initial level	At initial level
19.	Parallel testing	Not allowed	Allowed	Allowed	Allowed
20.	Component driver needed	Yes	No	Yes	Yes
21.	Work parallelism at beginning	High	Low	Medium	Medium
22.	Integration stage	Late	Early	Early	Early
23.	Testing a specific path capability	Easy	Easy	Hard	Medium
24.	Require stubs	Yes	Yes	No	Yes
25.	Planning and controlling of sequence capability	Easy	Hard	Easy	Hard
26.	Basic working program time	Late	Early	Late	Early

3.3 Areas of Application Integration Testing

Various areas where each integration testing approach can be applied in order to know where each approach is most suitable for execution of software project are discussed thus. The major application areas are web application and database application. Web application is for testing the output of merging client- and server-side code and accessing it through a web browser to determine its functionality within the context of web applications while database application is concerned with specifying test for queries results and the test to validate the database state after several update operations.

3.3.1 Areas of Application of Big-Bang Approach

It is applicable in cases where the project is not complex and need to be delivered in a short period of time. The application areas are given thus:

i. Database development
ii. Mobile app development
iii. Web applications development
iv. Networking
v. Server monitoring
vi. System analysis
vii. Reporting
viii. Code optimization
ix. Generation of test data
x. Database profiling

3.3.2 Areas of application of Top-down Approach

The top-down integration begins from the root node of the program module that is called by the main program known as stub. The following are the areas of application of top gown approach:

i. Database development
ii. Mobile app development

iii. Web applications development
iv. Networking
v. Server monitoring
vi. System analysis
vii. Reporting
viii. Code optimization
ix. Generation of test data
x. Database profiling

3.3.3 Areas of Application of Bottom-up Approach

Bottom-up can be applied in cases where an existing component is required and needs to be integrated into the product [34]. The application areas of bottom-up approach are highlighted thus [35]:

i. Database development
ii. Mobile app development
iii. Web applications development
iv. Networking
v. Server monitoring
vi. System analysis
vii. Reporting
viii. Code optimization
ix. Generation of Test data
x. Database profiling

3.3.4 Areas of Application of Mixed Approach

This approach can be termed as a smaller version of the big-bang approach since it is combining two approaches. Though, problem isolation is tedious as the total testing session varies with the total number of sub-trees [36]. The application areas are given below:

i. Database development
ii. Mobile app development
iii. Web applications development
iv. Networking
v. Server monitoring
vi. System analysis
vii. Reporting
viii. Code optimization
ix. Generation of Test data
x. Database profiling

4. AUTOMATION TOOLS FOR INTEGRATION TESTING

The interaction between components and their interfaces can be tested both manually and with the use of testing tools. Automation of integration testing makes it more convenient and faster to achieve the testing phase [37]. Effective conduct of integration testing sometimes is not easy as there is high probability of missing some parts due to oversight. It is not always easy to carry out integration testing on a large system with many sub-systems manually. The use of automation tools needs to be adopted in order to get a quality software product that meets user requirements. Some of integration testing tools are TESSY, FitNesse, rational integration tester, protractor and CITRUS.

4.1 TESSY

Tessy is an essential tool that is used to check the components' interfaces of a system under development. This tool takes in two inputs; the source code of the system and the requirement document that had been prepared before the implementation phase based on user requirements. This software checks the conformity of the built system with respect to the requirement document. This tool tests the system with the use of requirement document in order to ensure the system works as the user wants. The tool requires the tester to highlight all components' interfaces. The interfaces are then checked using a scenario to confirm the interaction between them [38].

4.1.1 Features of TESSY

The following are the characteristics of TESSY integration testing tool [38]:

i. **Need for requirement document:** TESSY tools requires the introduction of the requirement document alongside with the source code of the system to assure users' satisfaction.
ii. **Scenario formulation:** this is done when the main functionalities of each model and their interactions have been known. The tester creates a situation that can represent the main function of the system. The right and wrong inputs are subjected to the system to check it reliability.
iii. **Creation of interfaces:** the interface of all components is listed in the software in order to achieve the aim of integration testing which is to check for well-functioning of the system.

iv. **Generation of outcome report:** this testing tool is capable of producing the results of the testing after the completion.

4.2 Rational Integration Tester

This is a software used to automate integration testing. This testing tool helps to test a system in a very short time interval with a very moderate cost. This tool tests a system with incomplete modules. The missing modules in the system can be replicated using an artificial module known as stubs. This tool is best used in a top-down approach where there is a need for sub-modules to be replaced by stubs. This tool best suits the iterative approaches for integration testing, that is, top-down and bottom-up integration approaches.

4.2.1 Features of Rational Integration Tester

This testing tool is an automated version of the iterative approach to integration testing. The following are the distinguishing characteristics of this tool.

i. **Use of stubs:** rational integration tester allows the use of stubs to replicate the missing components in a system while undergoing integration testing.
ii. **Test-case design:** test case to use for the integration testing can be from the requirement document of the system or external test case.
iii. **Functional testing**: this tool can be used to check the functionality of the system. This is one of the main features of top-down approach.

4.3 Protractor

Protractor is an integration testing tool that is basically designed to test web application. The web-pages in a web application are put to test to check their interfaces and their inter-connectivity. For example, once a login page has been filed correctly, on clicking the sign-in button, it should move to the homepage of the web application. This tool is used to test the communication between web pages. It is mostly used with Angular JS application.

4.3.1 Features of Protractor

The following are the characteristics of Protractor as an integration testing tool:

a. **Angular JS based:** it is mainly used to test application that is Angular JS based.

b. **Multi-task:** protractor can be used on more than one browser simultaneously with the use of selenium grid.
c. **Simplicity:** this tool uses a simple syntax to automate the integration process.

4.4 Badboy Integration Testing Tool

This is used in CPU process measuring as well as consumption of memory, processes number, e.t.c.

4.5 JMeter Integration Testing Tool

This is useful in integration testing of different servers for example web servers, application servers as well as distributed and stand-alone databases.

4.6 Selenium Integration Testing Tool

This is used in program correction as well as quality of output tracking.

4.7 Worksoft Integration Testing Tool

This is used for testing of a whole program using several inputs, individual functions exercises as well as methods of object [39] , [40]**.**

4.8 Other Integration Testing Tools

There are other integration testing tools such as Citrus, Jasmine, FitNesse, VectorCAST / Ada, Validate MSG, LDRA, Smart Integration Test Accelerator (SITA), Cucumber, Steam, eZscript, Pioneerjs, VectorCAST/C++, TESSY and Spock for JAVA.

5. CONCLUSION

Software testing is generally important while working on a software project. Checking for the correctness is very essential to know how well the components in the software product interacts. Four approaches can be used to achieve software testing goals. Big-bang approach, top-down approach, bottom-up approach and mixed approach also known as hybrid approach have been carefully considered in this study. Some of the key factors to consider when choosing the software testing approach to work with are project size, complexity and time constraint;

reliability of the approach for the intended software project and the cost of carrying out integration testing. Big-bang approach is suitable for a small software project that has short life span. Long term project with many sub - projects and high complexity is peculiar to mixed approach. Bottom-up approach best fits a software project that requires simultaneous development and testing. Project risk also need to be considered while testing software. Mixed approach can be applicable for a high-risk software project with a very high-cost estimation. The detailed comparative analysis was given in Table 2 using twenty-six characteristics. It is therefore, recommended that the use of machine learning paradigms for the qualitative computation of the best software integration testing can be employed for future studies in relation to any given software project.

References

[1] Akinsola J. E. T., Ogunbanwo A. S., Okesola O. J., Odun-Ayo I. J., Ayegbusi F. D. and Adebiyi A. A. (2020). "Comparative Analysis of Software Development Life Cycle Models (SDLC)," *Springer*, vol. 1224 AISC, pp. 310–322, doi: 10.1007/978-3-030-51965-0_27.

[2] Adeagbo M. A., Akinsola J. E. T., Awoseyi A. A. and Kasali F. (2021). "Project Implementation Decision Using Software Development Life Cycle Models: A Comparative Approach," *J. Comput. Sci. Its Appl.*, vol. 28, no. 1, doi: 10.4314/jcsia.v28i1.10.

[3] Akinsola J. E. T., Kuyoro A. O., Adeagbo M. A. and Awoseyi A. A. (2020). "Performance Evaluation of Software using Formal Methods Global Journal of Computer Science and Technology," *Glob. J.*, vol. 20, no. 1.

[4] IBM (2022). "What is Software Testing and How Does it Work," *IBM*.

[5] Sawant A. A., Bari P. H. and Chawan P. M. (2012). "Software Testing Techniques and Strategies," *J. Eng. Res. Appl.*, vol. 2, no. 3, pp. 980–986.

[6] Tahvili S. (2018). *Multi-Criteria Optimization of System Integration Testing*..

[7] Arumugam A. K. (2019). "Software Testing Techniques & New Trends," *Int. J. Eng. Res. Technol.*, vol. 8, no. 12, pp. 708–713.

[8] Reis S., Metzger A. and Pohl K. (2007). "Integration Testing in Software Product Line Engineering : A Model-Based Technique" pp. 321–335.

[9] Rastogi S. (2021). "A comparative study of software testing techniques," *Int. J. Innov. Sci. Res. Technol.*, vol. 6, no. 7, pp. 114–119, doi: 10.1007/978-3-319-59647-1_27.

[10] Wikipedia (2021). "Integration Testing," *Wikipedia*,. .

[11] Geeks for Geeks (2020). "Software Engineering | Integration Testing," *Geeks for Geeks*.

[12] Pedamkar P. (2020). "Integration Testing _ Types & Approach with Advantages & Disadvantages," *EDUCBA*.

[13] Parmar K. (2014). "Integration Testing Techniques," no. SAP AG, pp. 1–15.

[14] Dhanalakshmi K. (2019). "Integration testing," Coimbatore.

[15] Hamilton T. (2021). "What is System Integration Testing (SIT) with Example".

[16] Selvan A. (2020). "Integration Testing Types & Approaches," *TechAffinity*.

[17] Rana K. (2020). "Integration Testing Types, Challenges, Benefits and Tools," *ArtOfTesting*.

[18] Rajkumar (2020). "Integration Testing - Big Bang, Top Down, Bottom Up & Hybrid Integration - Software Testing Material," *Software Testing Material*.

[19] Behera H. S., Sahu K. K. and Bhattacharjee G., "Lecture Notes On Course Code : BCS-306."

[20] QATestLab (2018). "Big-Bang Testing Specifics – QATestLab," *QATestLab*.

[21] Poonam (2019). "Big Bang Approach-The Way to Test Modules as Whole - TestOrigen," *TestOrigen*.

[22] Gupta Y. (2022). "ISTQB Foundation Level Exam Crash Course Part-3 - Software Testing Genius," *Software Testing Genius*.

[23] Hamilton T. (2022). "Integration Testing_ What is, Types, Top Down & Bottom Up Example," *Guru99*.

[24] JavaTpoint (2021). "Integration Testing - javatpoint," *JavaTpoint*.

[25] Geekforgeek (2013). "Steps in Bottom Up Integration Testing - GeeksforGeeks."

[26] Tutorialspoint (2021). "Bottom Up Testing - Tutorialspoint."

[27] GeeksforGeeks (2020). "Steps in Bottom Up Integration Testing," *GeeksforGeeks*.

[28] Singh R. and Khan I. A. (2012). "AN APPROACH F OR I NTEGRATION T ESTING I N," *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 3, pp. 141–158.
[29] Professionalqa.com (2022). "Top Down Integration Testing."
[30] Mikhalchuk O. (2020). "7 essential steps to top-down integration testing strategy _ Forte Group," *Forte Group*.
[31] Brown H. (2021). "What is Integration Testing: Approaches and Challenges Explained! Cyclr," *Cyclr*.
[32] Choudary A. (2020). "What is Integration Testing? | How to perform integration testing?," *Edureka*.
[33] MKS075 (2022). "Sandwich Testing _ Software Testing - GeeksforGeeks," *GeeksforGeeks*.
[34] ProfessionalQA.com (2022). "What is Bottom Up Approach in Software Testing_ _Professionalqa."
[35] Abdul M., Maniyar R., Hakeem M. A., Khalid M. and Zafar M. (2018). "Bottom-up Approach for Performance Testing of Software Applications or Products," vol. 6, no. 9, pp. 7812–7817, 2018, doi: 10.15680/IJIRCCE.
[36] Jorgensen P. C. (2014). "Integration Testing," *Softw. Test.*, pp. 225–250, doi: 10.1201/b16592-22.
[37] Javapoint (2021). "Integration Testing Tools - javatpoint."
[38] Embitel (2021). "How Tessy Tool Automate the Integration Testing of Automotive Software".
[39] Reddy M. R., Yalla P. and Chandra J. V., "Design and Implementation of Integrated Testing Tool," *Researchgate*, no. May, pp. 10464–10472, 2014.
[40] W3Softech (2019). "W3Softech," *W3Softech*.