I.J. Information Technology and Computer Science, **2013, 08, 109-122**
Published Online July 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2013.08.11

Improvement of Component Integration Testing Technique

Khulood Salem Albeladi, M. Rizwan Jameel Qureshi
Faculty of computing and Information Technology, King Abdulaziz University, Saudi Arabia
E-mail: khuloodsalem@yahoo.com, anriz@hotmail.com

***Abstract*—** Component-based technology can increase reuse and productivity, but high-quality component-based systems are often difficult to implement. Component developers do not know the systems where the components will be used, while software engineers must develop new systems with limited knowledge on available components. We propose a new testing technique that generates, at the time of component development, integration test cases from the specification of the behavior expected from other components of the system. The technique presented in this paper supports both the component developer, who can early test the integration of the components with the system, and the software engineers, who can test concrete components at deployment time, simply re-using existing test cases.

The technique presented in this paper supports both the component developer, who can early test the integration of the components with the system, and the software engineers, who can test components at deployment time. We used questionnaires to validate the proposed solution.

***Index Terms*—** Integration, CBSE, CBSD, Software Development, Test Cases Component-Based Application

I. Introduction

The idea with component integration is that separate components are combined into a working system. However, this process of assembling parts into bigger units, products and systems is not well performed in industry, especially not when a substantial part of the product functionality is implemented in software. Many faults that are introduced in early phases are found as late as in the product integration phase, or even worse, in the verification or validation of the final delivery, or after delivery of the product or system. This leads to high costs for error correction and additional efforts for re-testing [1].

Today's software larger in size, design complex and time consuming to implement them, for this we need a prominent solution to overcome these problems. Component-based software development (CBSD) has emerged as an Object Oriented (OO) Software Engineering approach that forced rapid software development. Using CBSE approach we can eliminate these problems largely. To build the application using CBSE approach we can develop the software with lowest price, reduced in size and we can reduce the time also. The component-based application may be implemented in house or by different vendors, integrate them in a different environment is still challenging. The nature of the component is Heterogeneous, so the integration is a bit complicated [3]. The rest of the paper is organized as follows. Section 2 reviews related literature. In Section 3, we define the problem statement. Section 4 presents the proposed solution. Section 5 describes the validation of proposed solution.

II. Related Work

Integration test cases can be automatically derived with the technique by Mariani, Pezz`e and Willmor [1]. They automatically infer a model of component interactions and they automatically generate the corresponding integration test cases. Test cases cover aspects related to both the protocol and data values used for interactions, but neglect coverage of the assumptions that single components perform on state evolution of the other components of the system.

An approach [2] was developed to integrate distributed components in different languages and on different platforms in the implementation phase with the known component interactions. Ports and links are use to specify the inter-component communication so that core component functions are separated from intercomponent communication .In component-based testing, Rosenblum [3] proposed a formal model for adequate testing of component-based software, in which a "C-adequate" criterion is defined to determine the adequacy of a test set for a given component-based software system as well as for a single component.

Gao et al. [4] have proposed a component test model to analyze API-based component validation and testing. The test model uses the concepts of the component function access graph to represent components access patterns. Further, a set of API-based test criteria is also proposed to evaluate the models. They have also proposed a component regression test approach [5] to

identify component changes and their impact on CBS. They have also developed a tool called COMPTest which supports automatic identification analysis of API-based component changes and black box test selection.

An approach [6] was developed test method that facilitates the design, execution, assessment, and report generation of test cases for automotive systems that show continuous behavior. At the same time, it provides a systematic approach for test case selection that helps to reveal redundancies and missing, but relevant aspects in test sets. In [7], a testing method which utilizes the Service-oriented architecture to support testing of complex and safety-critical systems is presented. However, this testing approach focuses on the distribution and performance of testing process, e.g., distributed testing among testing hosts, rather than how to model testing as a service. The Self-testing COTS components [8] strategy proposes to augment a component with functionality of analysis and testing tools thus enabling it to be capable of conducting some or all activities of the component user's testing processes. Reiko and Leonardo [9] discussed issues in testing distributed component-based systems and suggested an interface and exception coverage-based testing strategy. In [10], Stephen and Bing have presented an approach to object-oriented distributed component software development. Based on the distributed component architecture we have defined, we have developed the integration process of distributed software and the use of component adapter in connecting components' methods/events across different interfaces of the components.

III. Problem Definition

There are a lot of challenges that facing developer during using approach of component-based development, one of the biggest challenges is how to integrate various components in software systems without error. Although Component-Based System (CBS) increases the efficiency of development and reduces the need for maintenance, but even good quality components could fail to compose good product if the integration is not managed appropriately. In real world, such as industrial automation domain, this probability is unacceptable because additional measures, time, efforts, and costs are required to minimize its impacts.

IV. The Proposed Solution

Test cases are executed at an early stage to validate the integration of the component with the expected behavior of the system, and then are re-executed with concrete components at deployment time. The technique presented in this paper supports both the component developer, who can early test the integration of the components with the system, and the software engineers, who can test concrete components at deployment time. When executing a component, we submit an external input to the integrated system, observe the external output. At the same time, by observing the internal interfaces, we also obtain input/output sequences of every component in the system. After executing process, we check whether the earned result has been respected. If not, we identify the problematic component replace it and start all over again.

To reduce integration errors we need for iteratively trying to determine errors to reach for component integration with highest level of quality. The Proposed scenario to reduce integration problem as much as possible is:

1. Both of component developer and software engineer are responsible about testing process.
2. Software engineer has to create a package which is consist of two Tables and check list. Tables are Expected Result Table (see Table 2) and earned Result Table (see Table 3). While check list consists of number of criteria that is required to test integration of each component.

Example of check list:

Criteria

- All High prioritized bugs fixed and closed
- All Modules to be code completed and integrated successfully.
- Successful Testing of Integrated Application.

2.1 software engineers will describe the Expected result of integration of each two components in Expected result field.

2.2 component developers will describe the earned result of integration of each two components in earned result field.

3. Software engineer has to provide the package to component developer.
4. Component developer will integrate the components (according to Expected Result Table), which means that some of the outputs of one component will appear as inputs on the connected interface of another component.
5. Record the result of step 4 in 'Earned Result Table' and compare it with expected result that is stated in 'Expected Result Table'.

5.1 If the results are same indicating integration of those components is done without errors i.e. they don't need maintenance.

5.2 If the results are not same reflecting error at least in one of those components and there is maintenance required.

6. Component developer is responsible for error correction and to resend the package to software engineer.
7. Software engineer will check integration of each component using the check list.

By that we save cost and time because the testing process divided between component developer and software engineer. There is ability of determining exactly where the error which facilitates maintenance process and error correction is.

Table 1: Comparison of Related Work

Paper Title with reference number	Problem Found
Generation of integration tests for self-testing components [1]	Since all modules are tested at once, high risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.
An Approach to Distributed Component-based Real-time Application Software Development [2]	Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
Adequate Testing of Component-Based Software [3]	Since the integration testing can commence only after "all" the modules are designed, testing team will have less time for execution in the testing phase.
Merging components and testing tools: The Self-Testing COTS Components (STECC) Strategy [4]	Limited extension of object functionality objects can be extended only via inheritance and multiple inheritance cannot expose the same interface more than once, nor can it alone determine which interface should be exported to clients
A Systematic Regression Testing Method and Tool For Software Components [5]	No standard way to deploy object implementations in server processes.
Systematic Testing of the Continuous Behavior of Automotive Systems [6]	Limited standard support for common design patterns: Provides a rich set of features to implement servers
An Integrated Testing Technique for Component-Based Software [7]	Software is developed during the implementation phase, so no early prototypes of the software are produced. If any changes happen in midway, then the test documents along with requirement documents has to be updated.
Integration Testing of Components Guided by Incremental State Machine Learning [8]	Proposed test technique requires high dependency on modeling skills and inapplicable to cheaper projects as cost of modeling and automated code generation is very high.
Component Integration Testing by Graph Transformations [9]	Interlinking of Components and the container are very complicated and difficult to understood
A Component-Based Approach to Object-Oriented Distributed Application Software Development [10]	A middleware is needed in which the component is supposed to work. (Middleware: A communication layer which enables components to interact with higher level component in a network).

Table 2: Expected Result Table

Test case ID	Test Case Objective	Test Case Description	Expected Result
1	Check the interface link between the Login and Mailbox module	Enter login credentials and click on the Login button	To be directed to the Mail Box
2	Check the interface link between the Mailbox and Delete Mails Module	From Mail box select the an email and click delete button	Selected email should appear in the Deleted /Trash folder

Table 3: Earned Result Table

Test case ID	Test Case Objective	Test Case Description	Earned Result
1	Check the interface link between the Login and Mailbox module	Enter login credentials and click on the Login button	Move directly to the sent Box
2	Check the interface link between the Mailbox and Delete Mails Module	From Mail box select the an email and click delete button	Nothing

V. Validation of the Proposed Solution:

A questionnaire is consisting of 20 closed ended questions that are divided into 4 goals. Questions are arranged according to their relevancy to specific goals. It is statistically analyzed and interpreted to find support to Proposed Solution.

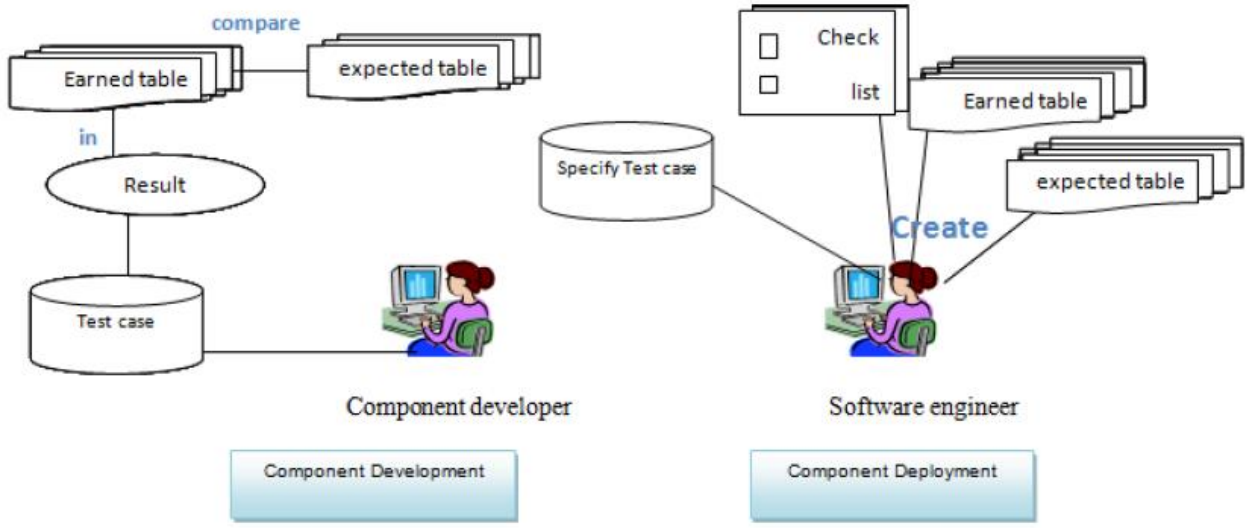

Fig. 1: Component developer and software engineer responsibilities

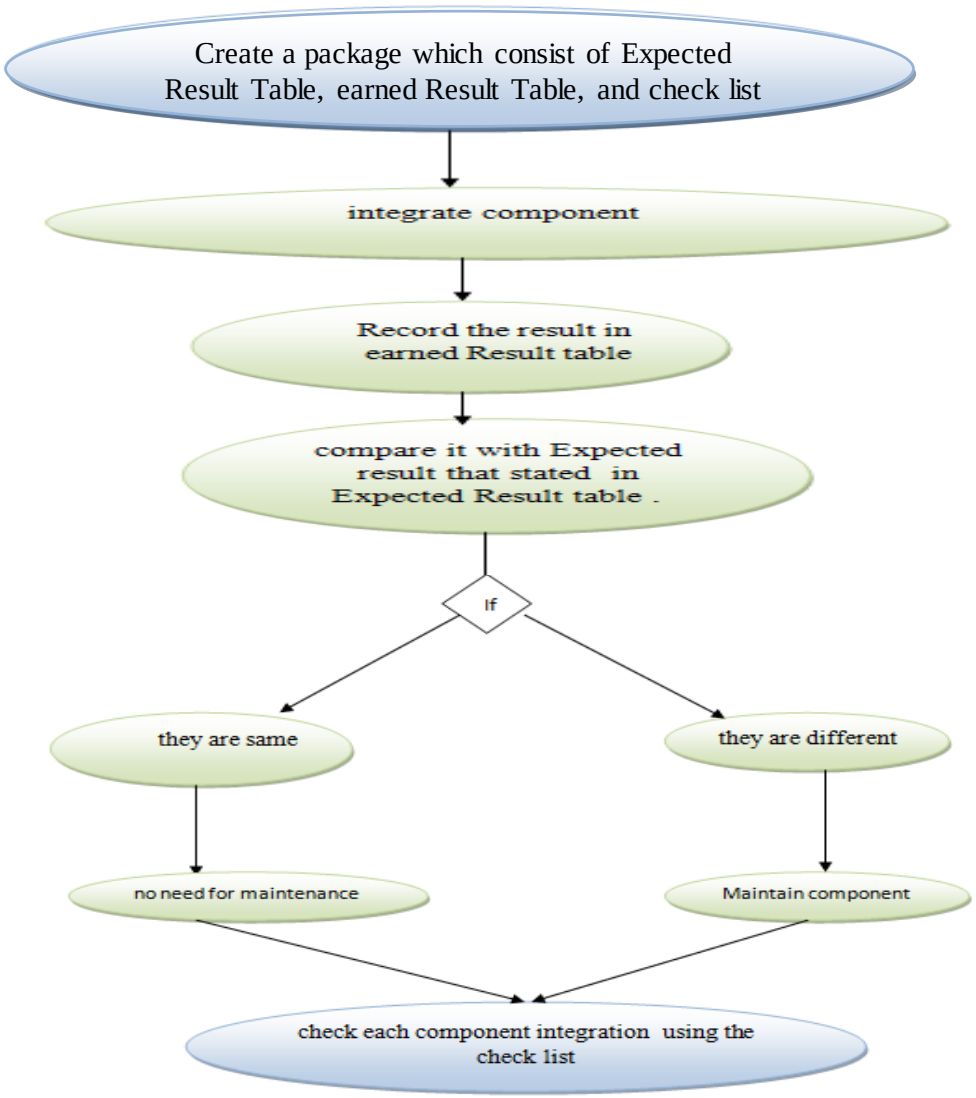

Fig. 2: Diagram of Proposed Solution (Blue: done by software engineer. Green: done by component developer)

Likert Scale

Table 4: Likert Scale

5	Strongly Agreed
4	Agreed
3	Neither Agreed Nor Disagree
2	Disagree
1	Strongly Disagree

5.1 Validation Using Survey

Statistical analysis is made on the basis of gathered data through the distribution of questionnaire. The analysis form is represented through frequency tables and bar charts showing the exact degree of analysis. We describe the validation results on the basis of our results below.

Goal: 1

Component-based development is more popular than traditional development.

Goal: 1

Question1 shows how much usefulness of component based development

Table 5: Frequency Table of Question1

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	25	83%	83%	83%
4	2	7%	7%	90%
3	3	10%	10%	100%
2	0	0%	0%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 83% strongly agreed, 7% agreed where as 10% neither agreed nor disagree from the proposed statement. And 0% is disagreeing and 0% is strongly agreed.

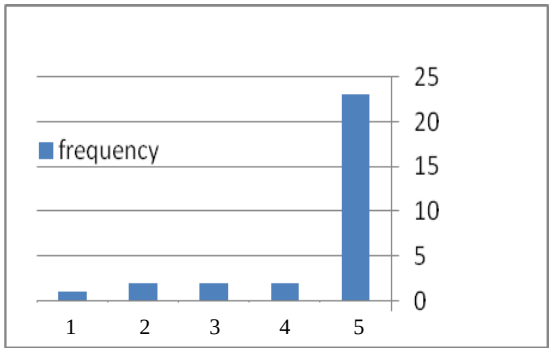

Fig. 2: Graphical representation of question 1

Goal: 1

Question-2 shows how much effectiveness of component based development

Table 6: Frequency Table of Question2

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	20	67%	67%	67%
4	5	17%	17%	84%
3	2	7%	7%	91%
2	2	7%	7%	98%
1	1	3%	3%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 67% strongly agreed, 17% agreed where as 7% neither agreed nor disagree from the proposed statement. And 7% are disagreeing and 3% are strongly agreed.

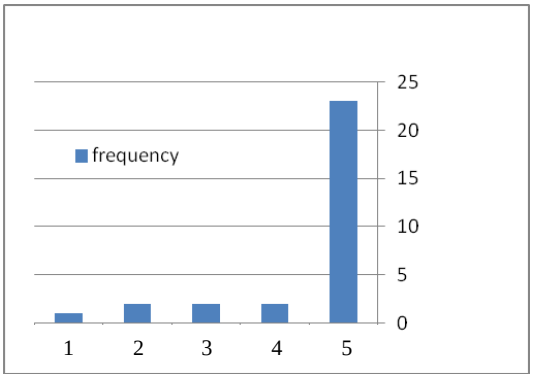

Fig. 3: Graphical representation of question 2

Goal: 1

Question-3 shows how much the users prefer component based development.

Table 7: Frequency Table of Question 3

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	15	50%	50%	50%
4	10	33%	33%	83%
3	5	17%	17%	100%
2	0	0%	0%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 50% strongly agreed, 33% agreed where as 17% neither agreed nor disagree from the proposed statement. And 0% is disagreeing and 0% is strongly agreed.

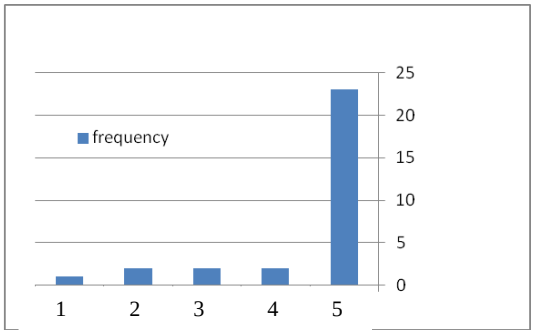

Fig. 4: Graphical representation of question 3

Goal: 1

Question-4 shows the component based development benefits in point of users view.

Table 8: Frequency Table of Question4

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	27	90%	90%	90%
4	2	7%	7%	97%
3	1	3%	3%	100%
2	0	0%	0%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 90% strongly agreed, 7% agreed where as 3% neither agreed nor disagree from the proposed statement. And 0% is disagreeing and 0% is strongly agreed.

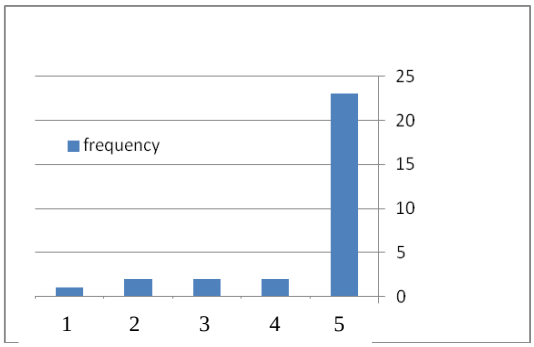

Fig. 5: Graphical representation of question 4

Cumulative Survey of Goal 1

In this phase we divide the tables in to issue wise the first issue covers that component based development most popular than traditional development. That's issue we resolved using survey through questioner. That's show 69.6% are strongly disagreed 15.2% are disagree 8.8% are neither agreed nor disagree 4% are agreed and 2.4 % are strongly disagreed.

Table 9: Cumulative analysis of Goal 1

Q.n	Strongly Agreed	Agree	Not Sure	Disagree	Strongly Disagree
1	25	2	3	0	0
2	20	5	2	2	1
3	15	10	5	3	2
4	27	2	1	0	0
Total	87	19	11	5	3
Avg.	69.6	15.2	8.8	4	2.4

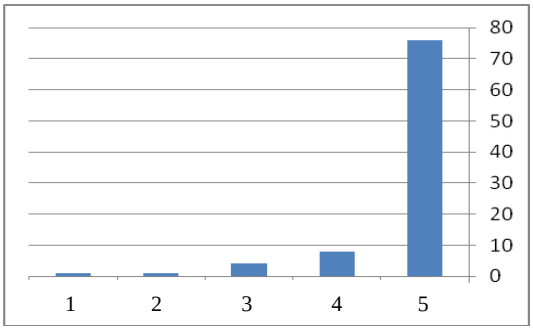

Fig. 6: Graphical representation of goal 1.

Goal: 2

The biggest challenge (that a developer faces in component-based development) is how to integrate various components in software systems without error.

Question-5 shows how much difficulty of component integration process.

Table 10: Frequency Table of Question 5

Liker	Frequency	Percent	Valid Percent	Cumulative Percent
5	11	37%	37%	37%
4	5	17%	17%	54%
3	10	33%	33%	87%
2	4	13%	13%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 37% strongly agreed, 17% agreed where as 33% neither agreed nor disagree from the proposed statement. And 13% are disagreeing and 0% is strongly agreed.

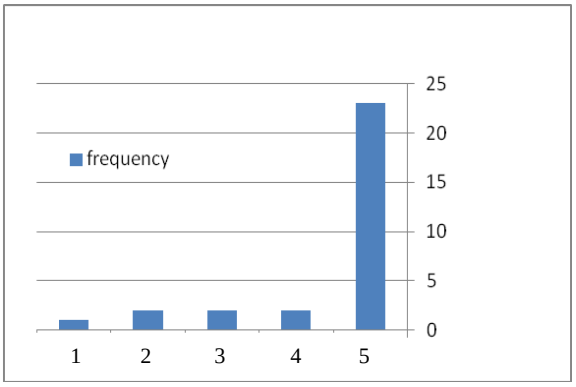

Fig. 7: Graphical representation of question 5

Goal: 2

Question-6 shows amount of reliability of component integration process.

Table 11: Frequency Table of Question 6

Liker	Frequency	Percent	Valid Percent	Cumulative Percent
5	18	60%	60%	60%
4	5	16%	16%	77%
3	2	7%	7%	84%
2	2	7%	7%	90%
1	2	10%	10%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 60% strongly agreed, 16% agreed where as 7% neither agreed nor disagree from the proposed statement. And 7% are disagreeing and 10% are strongly agreed.

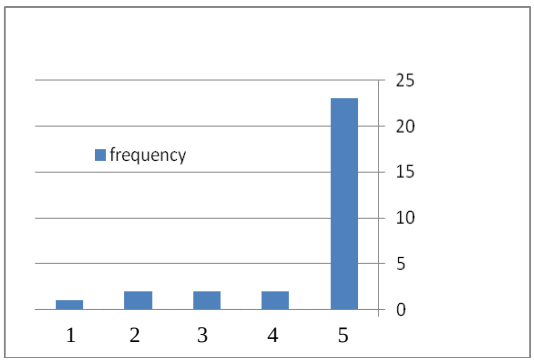

Fig. 8: Graphical representation of question 6

Goal: 2

Question-7 shows approximate amount of errors that appear during component integration process.

Table 12: Frequency Table of Question 7

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	13	43%	43%	43%
4	5	17%	17%	60%
3	5	17%	17%	77%
2	4	13%	13%	90%S
1	3	10%	10%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 43% strongly agreed, 17% agreed where as 17% neither agreed nor disagree from the proposed statement. And 13% are disagreeing and 10% are strongly agreed.

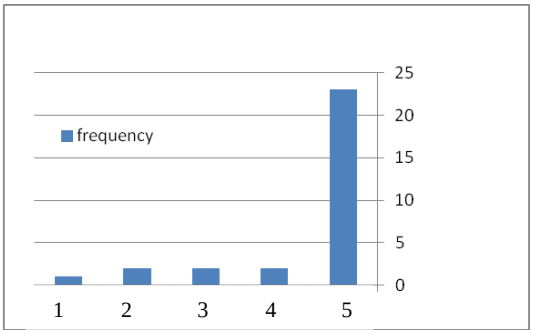

Fig. 9: Graphical representation of question 7

Goal: 2

Question-8 shows importance of testing phase of component integration process

Table 13: Frequency Table of Question 8

Liker	Frequency	Percent	Valid Percent	Cumulative Percent
5	25	83%	83%	83%
4	4	13%	13%	97%
3	1	3%	3%	100%
2	0	0%	0%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 83% strongly agreed, 13% agreed where as 3% neither agreed nor disagree from the proposed statement. And 0% is disagreeing and 0% is strongly agreed.

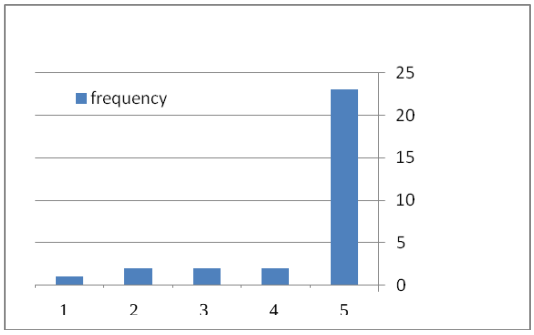

Fig. 10: Graphical representation of question 8

Cumulative Survey of Goal 2

In second issue we define that the biggest challenge that facing the developer of component-based development is how to integrate various components in software systems without error. That's show 56.77% are strongly disagreed 16.101% are disagree 15.2% are neither agreed nor disagree 8.47% are agreed and 3.38% are strongly disagreed.

Table 14: Cumulative analysis of Goal 2

Q.n	Strongly Agreed	Agree	Not Sure	Disagree	Strongly Disagree
5	11	5	10	4	0
6	18	5	2	2	1
7	13	5	5	4	3
8	25	4	1	0	0
Total	67	19	18	10	4
Avg.	56.77	16.101	15.2	8.47	3.38

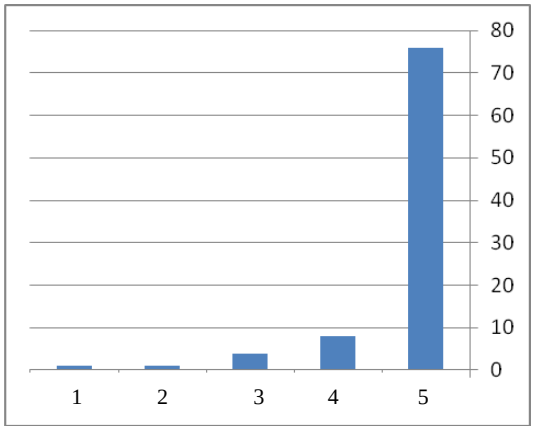

Fig. 11: Graphical representation of goal 2

Goal: 3 Measure the acceptance of proposed component integration testing methodology.

Goal: 3

Question-9 shows amount of difficulty that facing a component developer during component integration process.

Table 15: Frequency Table of Question 9

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	24	80%	80%	80%
4	4	13%	13%	93%
3	1	3%	3%	96%
2	1	4%	4%	100%
1	0	0%	%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 80% strongly agreed, 13% agreed where as 3% neither agreed nor disagree from the proposed statement. And 0% is disagreeing and 3% are strongly agreed.

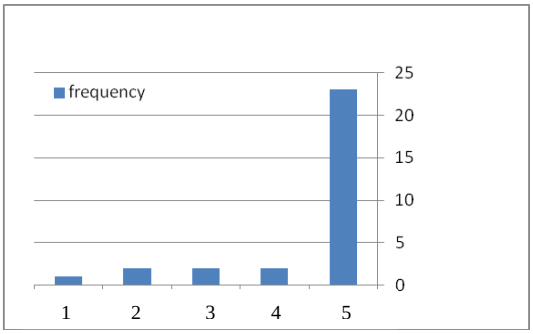

Fig. 12: Graphical representation of question 9

Goal: 3

Question-10 shows amount of effectiveness of collaboration between component developer and software engineer.

Table 16: Frequency Table of Question 10

Liker	Frequency	Percent	Valid Percent	Cumulative Percent
5	27	90%	90%	90%
4	2	7%	7%	97%
3	1	3%	3%	100%
2	0	0%	0%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 90% strongly agreed, 7% agreed where as 3% neither agreed nor disagree from the proposed statement. And 0% is disagreeing and 0% is strongly agreed.

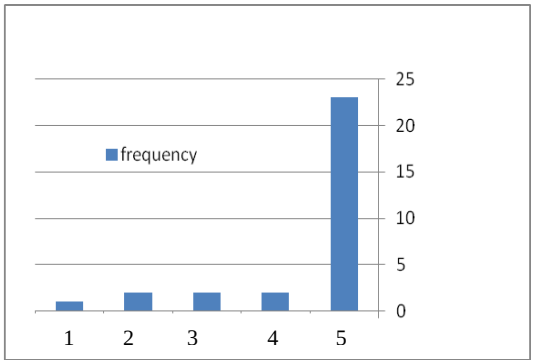

Fig. 13: Graphical representation of question 10

Goal: 3

Question-11 shows how much users accept of dividing testing process between component developer and software engineer.

Table 17: Frequency Table of Question 11

Liker	Frequency	Percent	Valid Percent	Cumulative Percent
5	25	83%	83%	
4	2	7%	7%	
3	2	7%	7%	
2	1	3%	3%	
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 83% strongly agreed, 7% agreed where as 7% neither agreed nor disagree from the proposed statement. And 3% are disagreeing and 0% is strongly agreed.

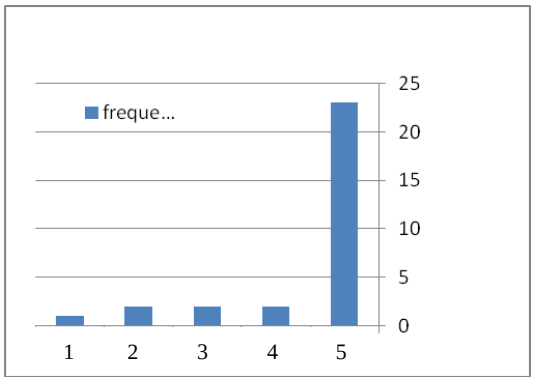

Fig: 14: Graphical representation of question 11

Cumulative Survey of Goal 3

In third issue that Measure the acceptance of proposed component integration testing methodology. That's show 84.44% are strongly agreed 8.88% are disagree 4.44% are neither agreed nor disagree 1.11% are agreed and 1.11% are strongly disagreed

Table 18: Cumulative analysis of Goal 3

Q.n	Strongly Agreed	Agree	Not Sure	Disagree	Strongly Disagree
9	24	4	1	0	1
10	27	2	1	0	0
11	25	2	2	1	0
Total	76	8	4	1	1
Avg.	84.44	8.88	4.44	1.11	1.11

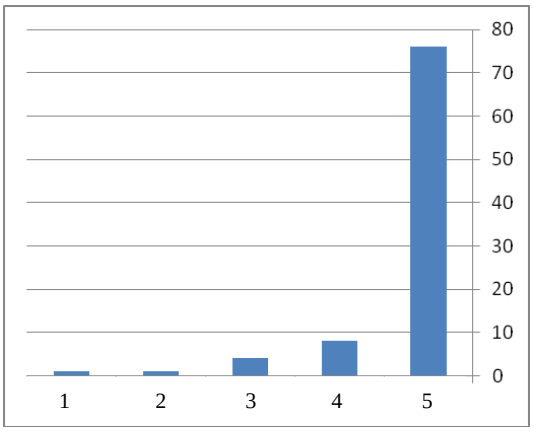

Fig. 15: Graphical representation of goal 3

Goal: 4

Measure the effectiveness of proposed component integration testing methodology.

Question-12 shows how much users accept the idea of Creating expected result Table to each test case by software engineer

Table 19: Frequency Table of Question 12

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	20	67%	67%	67%
4	4	13%	13%	80%
3	2	7%	7%	87%
2	4	13%	13%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 67% strongly agreed, 13% agreed where as 7% neither agreed nor disagree from the proposed statement. And 13% are disagreeing and 0% is strongly agreed.

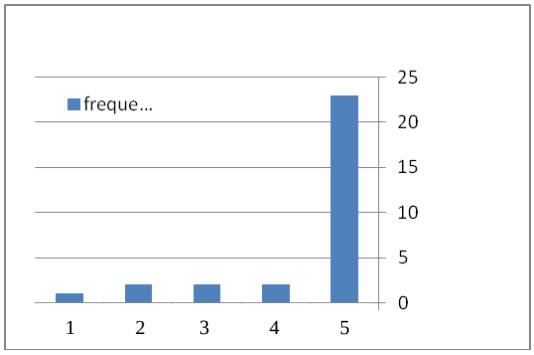

Fig. 16: Graphical representation of question 12

Goal: 4

Question-13 shows how much users accept the idea of earned result Table.

Table 20: Frequency Table of Question 13

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	26	87%	87%	87%
4	3	10%	10%	97%
3	1	3%	3%	100%
2	0	0%	0%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 87% strongly agreed, 10% agreed where as 3% neither agreed nor disagree from the proposed statement. And 0% is disagreeing and 0% is strongly agreed.

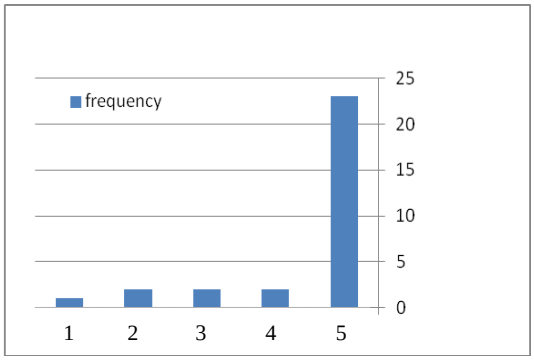

Fig. 17: Graphical representation of question 13

Goal: 4

Question 14 shows how much the proposed component integration testing methodology is suitable for the development of medium and large-scale software projects in point of users view.

Table 21: Frequency Table of Question 14

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	22	73%	73%	73%
4	4	13%	13%	86%
3	3	10%	10%	96%
2	1	4%	4%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 73% strongly agreed, 13% agreed where as 10% neither agreed nor disagree from the proposed statement. And 3% are disagreeing and 0% is strongly agreed.

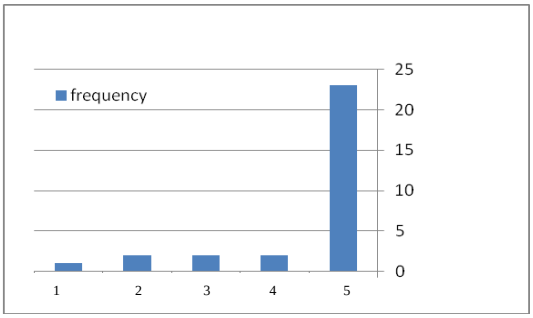

Fig. 18: Graphical representation of question 14

Goal: 4

Question-15 shows how much the proposed component integration testing methodology will save time in point of users view.

Table 22: Frequency Table of Question 15

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	21	70%	70%	70%
4	5	17%	17%	87%
3	2	7%	7%	94%
2	1	3%	3%	97%
1	1	3%	3%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 70% strongly agreed, 17% agreed where as 7% neither agreed nor disagree from the proposed statement. And 3% are disagreeing and 3% are strongly agreed.

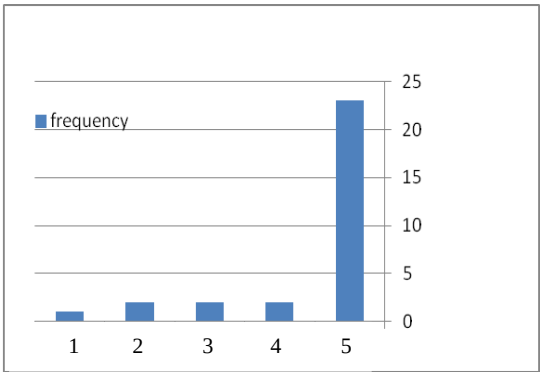

Fig. 19: Graphical representation of question 15

Goal: 4

Question- 16 shows how much the proposed component integration testing methodology will save cost in point of users view.

Table 22: Frequency Table of Question 16

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	24	80%	80%	80%
4	3	10%	10%	90%
3	1	3%	3%	93%
2	1	3%	3%	96%
1	1	3%	3%	100%
Total	30	100%	100%	

Considering the shape feature invariability of the reply through the frequency Table shows out of 30 questionnaires, 80% strongly agreed, 10% agreed where as 3% neither agreed nor disagree from the proposed statement. And 3% are disagreeing and 3% are strongly agreed.

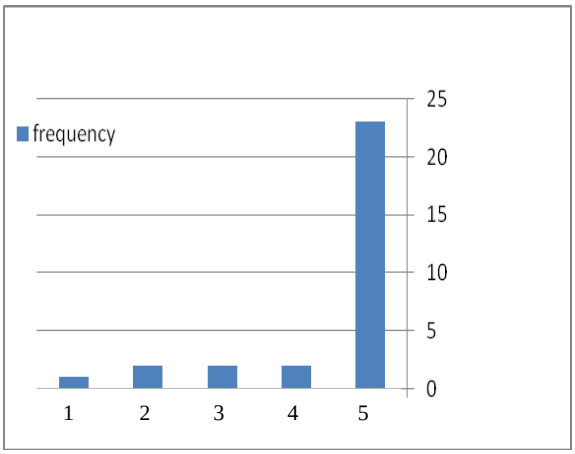

Fig. 20: Graphical representation of question 16

Goal: 4

Question 17 shows how much the proposed component integration testing methodology will reduce the complexity.

Table 23: Frequency Table of Question 17

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	20	67%	67%	67%
4	2	7%	7%	74%
3	4	13%	13%	87%
2	2	7%	7%	94%
1	2	7%	7%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 67% strongly agreed, 7% agreed where as 13% neither agreed nor disagree from the proposed statement. And 7% are disagreeing and 7% are strongly agreed.

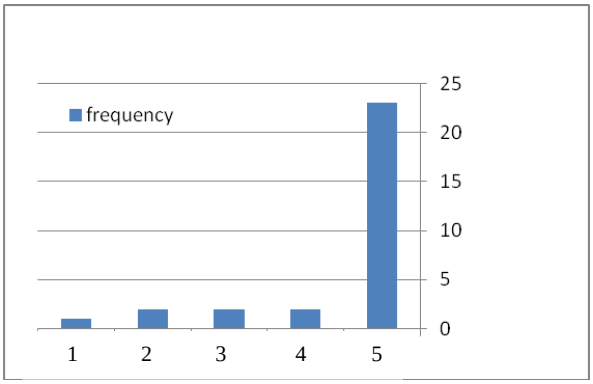

Fig. 21: Graphical representation of question 17

Goal: 4

Question 18 shows how much the proposed component integration testing methodology will increase the Reliability

Table 24: Frequency Table of Question 18

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	28	93%	93%	93%
4	1	3%	3%	96%
3	1	4%	4%	100%
2	0	0%	0%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 93% strongly agreed, 3% agreed where as 3% neither agreed nor disagree from the proposed statement. And 0% is disagreeing and 0% is strongly agreed.

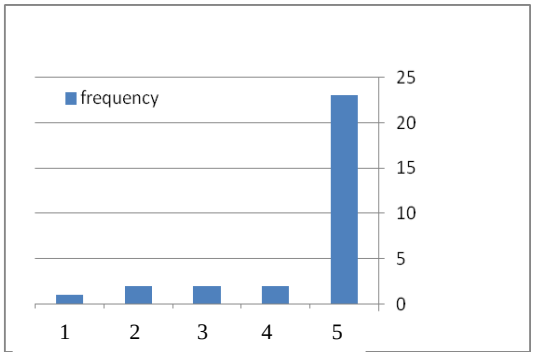

Fig. 22: Graphical representation of question 18

Question 19 shows how much the proposed component integration testing methodology will increase the quality

Table 25: Frequency Table of Question 19

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	26	87%	87%	87%
4	2	7%	7%	94%
3	1	3%	3%	97%
2	1	3%	3%	100%
1	0	0%	0%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 87% strongly agreed, 7% agreed where as 3% neither agreed nor disagree from the proposed statement. And 3% are disagreeing and 0% is strongly agreed.

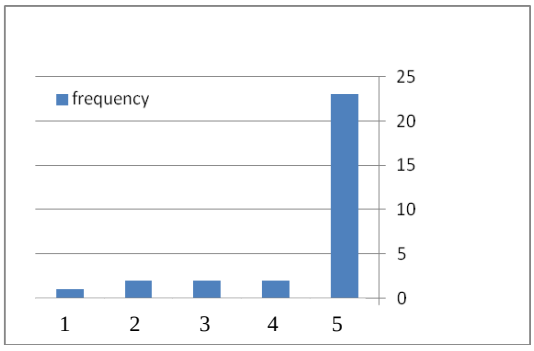

Fig. 23: Graphical representation of question 19

Question 20 shows how much the proposed component integration testing methodology will guide testing process towards the optimal solution.

Table 26: Frequency Table of Question 20

Liker Scale	Frequency	Percent	Valid Percent	Cumulative Percent
5	23	77%	77%	77%
4	2	7%	7%	84%
3	2	7%	7%	91%
2	2	7%	7%	97%
1	1	3%	3%	100%
Total	30	100%	100%	

The reply through the frequency table shows out of 30 questionnaires, 77% strongly agreed, 7% agreed where as 7% neither agreed nor disagree from the proposed statement. And 7% are disagreeing and 3% are strongly agreed.

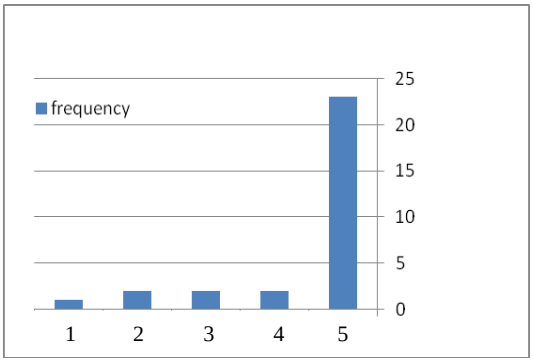

Fig. 24: Graphical representation of question 20

Cumulative Survey of Goal 4

In fourth issue that Measure the effectiveness of proposed component integration testing methodology. That's show 77.49% are strongly disagreed 9.59% are disagree 6.27% are neither agreed nor disagree 4.42% are agreed and 2.21% are strongly disagreed.

Table 27: Cumulative analysis of Goal 4

Q.n	Strongly Agreed	Agree	Not Sure	Disagree	Strongly Disagree
12	20	4	2	4	0
13	26	3	1	0	0
14	22	4	3	1	1
15	21	5	2	1	1
16	24	3	1	1	1
17	20	2	4	2	2
18	28	1	1	0	0
19	26	2	1	1	0
20	23	2	2	2	1
Total	210	26	17	12	6
Avg.	77.49	9.59	6.27	4.42	2.21

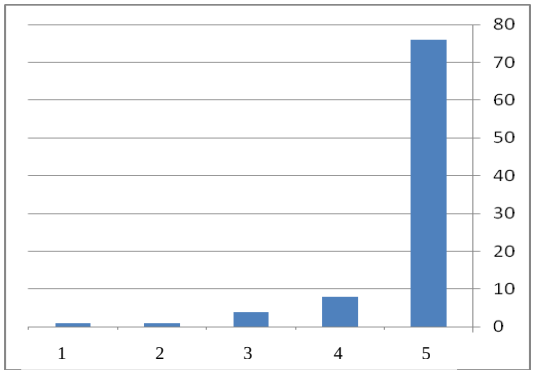

Fig. 25: Graphical representation of Goal 4

Cumulative Survey of Cumulative Survey of Goal 1, Goal 2, Goal 3 and Goal 4:

Table 28 shows that 72.14% are strongly disagreed 12.44% are disagree 8.65% are neither agreed nor disagree 4.47% are agreed and 2.26% are strongly disagreed.

Table 28: Cumulative analyses of all goals

Goal	Strongly Agreed	Agree	Not Sure	Disagree	Strongly Disagree
1	21.75	4.75	2.75	1.25	0.75
2	16.75	4.75	4.5	2.5	1
3	25.33	2.66	1.33	0.33	0.33
4	23.33	2.88	1.88	1.33	0.66
Total	87.16	15.04	10.46	5.41	2.74
Avg.	72.14	12.44	8.65	4.47	2.26

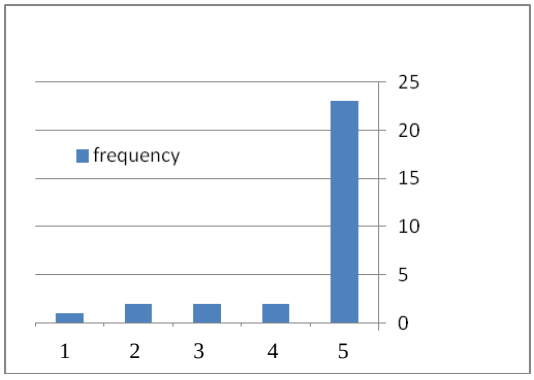

Fig. 26: Graphical representations of all goals

VI. Conclusion

Building a system from reused software components is the key idea introduced by the component-based software engineering (CBSE) approach. The systems developed from this approach are more flexible for facilitating maintenance, modifications and upgrades on their software components. Reuse of a poor quality software component, or not efficient use of a good quality software component, may lead to negative effects on the system users. Integration testing is important step of reliability and productivity assurance that focuses on the protection of integration errors and minimize the risk of the system as much as possible. In this paper, the reliability of some component testing techniques was evaluated to get a suitable integration testing technique for component-based software. Some of those techniques have their own drawbacks. Therefore, in this research, these testing techniques were extended to provide a more comprehensive testing technique that addresses these drawbacks. This work is supported with validation in which 84.58% people supported it while 6.73% disagreed to it as shown in "Fig. 27".

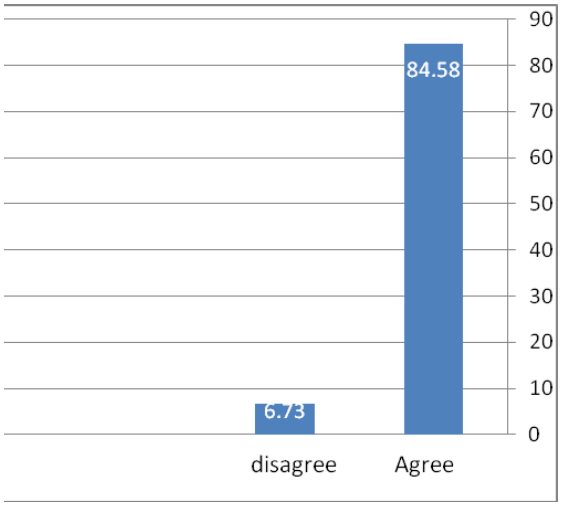

Fig. 27: Overall results for the validation of proposed solution

References

[1] L. Mariani, M. Pezzè, and D. Willmor, "Generation of Integration Tests for Self-Testing Components," [C]. in ITM '04: Proceedings of the 1st International Workshop on Integration of Testing Methodologies, 2008.

[2] S. S. Yau and B. Xia, "An Approach to Distributed Component-Based Real-time Application Software Development" [J]. IEEE. Object-Oriented Real-time Distributed Computing, 2009, 22(1):63-84.

[3] D.S. Rosenblum. "Adequate testing of component-based software."[C]. In: Proceedings of the Tenth European Symposium on Artificial Neural Networks (ESANN "2009), University of California at Irvine, 2009.

[4] Beydeda and V. Gruhn. Merging components and testing tools: the elf-testing cots components (stecc) strategy. [J]. IEEE. 12 (2), 2008.

[5] J. Gao, D Gopinathan, Quan Mai, and Jingsha He. “A systematic regression testing method and tool for software components"[C]. In: Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), 2006.

[6] Eckard Bringmann and Andreas Krämer. Systematic Testing of the Continuous Behavior of Automotive Systems. [J] IEEE. 12 (2), 2009.

[7] Sami Beydeda and Volker GruhnAn Integrated Testing Technique for Component-Based Software" [J]. IEEE. , 2008, 89(2):120-14.

[8] Keqin Li and Muzammil Shahbaz... Integration Testing of Components Guided by Incremental State Machine. [J].IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008, 22(1):63-84.

[9] Reiko Heckel and Leonardo Mariani. Component Integration Testing by Graph Transformations [C]. In: Proceedings of International Joint Conference in University of Paderborn, 2009, 2: 1449-1454.

[10] Stephen S. Yau and Bing Xia. Component- Based Approach to Object-Oriented. [J]. IEEE. , 2010, 89(2):120-14.

Authors’ Profiles

Dr. Rizwan Qureshi: Assistant Professor at Faculty of Computing & Information Technology, King Abdulaziz University, major in CBD and agile development.

Khulood Salem Albeladi: Master student in IT Department at King Abdulaziz University, interested in CBD and Technology Management.

How to cite this paper: Khulood Salem Albeladi, M. Rizwan Jameel Qureshi,"Improvement of Component Integration Testing Technique", International Journal of Information Technology and Computer Science(IJITCS), vol.5, no.8, pp.109-122, 2013. DOI: 10.5815/ijitcs.2013.08.11

