I.J. Information Technology and Computer Science, **2013, 09, 108-117**
Published Online August 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2013.09.12

Validation of Novel Approach to Detect Type Mismatch Problem Using Component Based Development

M. Rizwan Jameel Qureshi, Ebtesam Ahmad Alomari

Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: anriz@hotmail.com, ebtesam07@yahoo.com

***Abstract*——** Adaptation software component is a crucial problem in component-based software engineering (CBSE). Components that assembled or reused sometimes cannot perfectly fit one another because of the incompatibility issues between them. The focus today is on finding adaptation technique, to solve the mismatch between component interfaces and to guarantee that the software components are able to interact in the right way. This paper will focus on detecting mismatch, which considers as an important step through adaptation process. We propose a solution to detect mismatch, by suggesting improvement in Symbolic Transition Systems that used in representing component interface, and synchronous vector algorithm to deal with parameters data type mismatch.

***Index Terms*—** Software; Component; Adaptation; Mismatch

I. Introduction

Component based software engineering (CBSE) indicates that assembling and reusing existing software components can develop the new systems. However, the assembling or reusing process may lead to interoperation among components that rise the needing to adaptation technique.

Mainly, the component adaptation process helps to guarantee that software components are able to interact with each other's successfully. Moreover, finding effective adaptation approach considers as a difficult problem today where CBSE indicates that components have to be reusable from its interface.

According to the description of component interface, there are several levels of mismatches [1]: technical level, signature level, behavioral level, semantic level and service level. The behavioral mismatch can be caused by not correspond message names, incompatible ordering of messages in two or more components, or by some messages in one component that have no match with several messages in another component.

This paper focuses on detecting mismatch appearing at the behavioral level by suggest improving Symbolic Transition Systems (STS) that used to represent component behavior interface and improving the synchronous vector algorithm to make it able to detect data type mismatch, in addition to, massage name and parameter mismatch.

The paper is organized as follows: section 2 provides the literature review and limitations, section 3 describes the problem and proposed solution, section 4 illustrates the validation of proposed solution and the last section gives the conclusion.

II. Related Work

Most components cannot be integrated directly into an application because they are incompatible. Several studies performed to propose a solution to this problem. One study proposed a model-based adaptation approach for software adaptation [1]. They proposed that behavioral interfaces are represented by means of Labeled Transition Systems (LTSs). The synchronous product of several component LTSs results in new LTS, which contains all of the possible interactions between the involved components. Moreover, they rely on synchronous vectors, which denote communication between several components. Their proposal supported by dedicated algorithms that automatically generates adaptor protocols. These algorithms have been implemented in a tool, called Adaptor. Their proposal is equipped with two algorithms, depending on whether reordering is necessary or not in the adaptation process. The first one is based on synchronous product computation and the second one on encodings into Petri.

The previous approach solved the problem of messages mismatch and messages reordering, but it cannot perfectly solve the mismatch problem when messages transmit with parameters. Other study proposed approach to solve this problem [2] by applying the composition operator to synchronous vector and make it include the entire mismatch relations of the transmitted messages with parameters, then generated automatically adaptor protocols to solve the

mismatch problems based on Petri net encoding and Tina tools.

To calculate the behavior protocol of adaptor, there is a need to calculate the synchronous product of the STS specification of components and the STS specification of adaptor. The synchronous vectors indicate communication between several components, where each event appearing in one vector is executed by one component and the overall result corresponds to synchronization between all of the involved components [2].

Other study set up Component Interaction Adaptation Model (ClAM) to remove behavioral mismatch [3]. They describe two phases to remove mismatches. The first one is detecting behavioral mismatch, by dividing different component groups, detect interaction behavior deadlock. The second phase is obtaining adapter, in this phase they define behavior rule, build adapter specification by behavior rule and get an adapter to solve deadlock.

Furthermore, adaptation focusing more on generating as automatically as possible adaptors to solve the mismatch between components interfaces. Prior studies [4] [5] proposed general and safe approach to solve the behavior mismatch and indicated that the behavior of adaptor can be calculated automatically from the adapted components and the adaptor specification. The adaptor can correct the component interaction with data exchange, and realize the reordering of message. Furthermore, researchers computed synchronous product of Symbolic Transition Systems (STS), which are the abstract specification of component behavior, to detect automatically deadlock mismatch [6].

The Symbolic Transition Systems (STS) is used as graph tool to specify the component behavior, in addition to solve the component behavior mismatch and analysis the exchanged data. As presented in [6], STS is a tuple (A, S, I, F, T):

a) A is an alphabet, each element in A is correspond to the name of event;

b) S is a finite set; each element in S is a state;

c) I is the only initial state;

d) F is a set of finite states;

e) T is a set of transition functions,

The alphabet of STS consists with the signature information of component. Each element in A can be an inner operation of component, named as tau, also can be a tuple (CI, M, D, PL) [6]:

1) CI is the identifier of component;

2) M is the identifier of message, namely the name of operation, the name of interface;

3) D is the behavior type (!/?) of operation. The symbol "!" means that the operation of component provide resource to the system, and the symbol "?" means that the operation of component receive resource from the system;

4) PL is the parameter list of message.

To more support the adaptation of components, researchers in [7] developed the technique depend on binary component adaptation techniques and adaptation components techniques. In addition to that, they developed a support tool to support an effective of the adaptation process.

Other researchers proposed a new approach to component adaptation by dealing with extra-functional mismatches [8]. Their approach was proposed analyzing functional mismatches and extra-functional mismatches appearing in the integration and assembly of software components. Then, generating adapter specifications, and producing the final adapter mediating the functional and extra-functional mismatches. So, this approach can successfully solve most of both functional and extra-functional mismatches.

Additional study performed for distributed applications when designers want to distribute the adaptation mechanisms themselves. Researchers propose a model for dynamic adaptation that clearly separates adaptation from business logic, and that can be customized by applications designers in order to satisfy adaptation needs [9].

For gray-box component model, although it has been successfully used in software engineering, its adaption in real-time embedded systems still raises serious challenges. A prior study presented a component-based framework that automated the integration of these components [10].

Table 1 illustrates the limitations founded in each paper presented in the literature review.

Table 1: Comparison of brief literature review

Paper Title	Limitations
Model-Based Adaptation of Behavioral Mismatching Components [1]	The study did not solve the mismatch about the messages with parameters. Also, it omitted the elements relative to data exchange in the signature interfaces.
Model-Based Adaptation of Component Behaviors [2]	Their approach was only for the message level to solve the parameter mismatch; it did not describe the one-to-one relation of the parameters.
A Behavior-Driven Model of Component Interaction Adaptation [3]	Researchers did not focus on description, detection and adaptation of the non-behavior properties.
Research on Safe Behavior Adaptation of Software Component [4]	Researchers need to realize an automatic general solution for component behavior mismatch.

Paper Title	Limitations
Safety Verification of Software Component Behavior Adaptation [5]	They need to provide a calculation for the synchronous relation among the component behaviors, and solution to remember the order of messages and make them reorder.
Research on Behavior Adaptation of Software Component [6]	They need to provide a calculation for the synchronous relation among the component behaviors, and solution to remember the order of messages and make them reorder.
Component Adaptation Mechanism [7]	Researchers need to improve the automation of the adaptation process, and prevent errors that may occur during the adaptation process.
A New Approach to Component Adaptation Dealing with Extra-Functional Mismatches [8]	Researchers in this paper need to explore general formal methods representing extra functional adaptation, which supports semi-automatic or automatic process of generation of extra-functional adapter.
A Distributed Dynamic Adaptation Model for Component-Based Applications [9]	Researchers can refinement their model by identifying other functionalities, and going deeper into the structure of their coordination tower.
Automatic Synthesis and Adaption of Gray-box Components for Embedded Systems — Reuse vs. Optimization [10]	This study requires implementing the interactions between components to ensure the respect of non-functional properties.

III. Problem and Proposed Solutions

As presented in section 2, there is several studies discussed components adaptation and provides a solution for behavioural mismatch by focusing on massages and parameters. However, they do not consider data type mismatch. So, this paper will focus on this problem and suggest a solution.

The proposed solution will focus into parameters data type. It divides into two parts, first part focuses on specify data type by suggesting improvement in Symbolic Transition Systems (STS) model proposed in [4], which used to represent component behavior interface. The second part extends the synchronous vector algorithm provided in [2], to detect data type mismatch.

3.1 Component Interface Representation

This proposed solution improves Symbolic Transition Systems (STS), which presented in section 2. The solution suggests adding parameter's data type in transition label. So, we suggest adding "TY" to element in tuple (CI, M, D, PL, TY).

For each component's operation, data type should specify in addition to the other three parts of STS transition label.

1-operation name

2-behavior type (!/?) where "!" mean send and "?" mean receive.

3- Parameters name

4- Parameter data type

For example: if component1 has operation with name "Login" which send two integer parameter username and password to other component, this operation describe as shown in Fig.1, which represent the STS of component1.

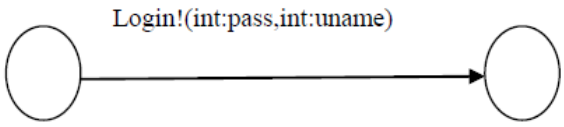

Fig. 1: STS of Component1

3.2 Detecting Mismatch

As presented in [2], behavior mismatch can be detected by synchronous vector that calculated after translating each component interface. This solution suggests extending synchronous vector algorithm to deal with data type by adding the data type name inside the parameter arc.

For each component i with STS Li,

"I" is the event, "P" is the parameter and "ty" is the data type

A) If l has the form a!

Then add an arc from the transition to place a (ty.P1, ...ty.Pn)

B) If l has the form a?

Then add an arc from the transition to place a (ty.P1, ...ty.Pn)

IV. Validation

Questioner shown in appendix A approves validation of proposed solution. It consists of 13 questions, which electronically distributed among different developers and designers through software engineering society.

Likert scale is ranging from 1 to 5 as the following.

- Strongly Disagree/ Very Low effect indicating 1
- Disagree/ Low effect indicating 2
- Neutral/ Normal effect indicating 3
- Agree/ High effect indicating 4
- Strongly agree/ Very High effect indicating 5

Statistical analysis performed after gathering data. Frequency tables and bar charts illustrate the analyses of the 30 questionnaires responses. The validation results describe below.

4.1 Cumulative Analysis of the First Goal

Goal 1: focusing on data type through adaptation components. Question 1: Is it important to consider parameter's data type through adaptation process?

Table 2: Frequency of question 1

	Frequency	Percent
1	1	3%
2	1	3%
3	5	17%
4	15	50%
5	8	27%

The frequency Table 2 shows out of 27% questionnaires strongly agreed, 50 % agreed where as 17 % neither agreed nor disagree. And 3% of people are disagreeing.

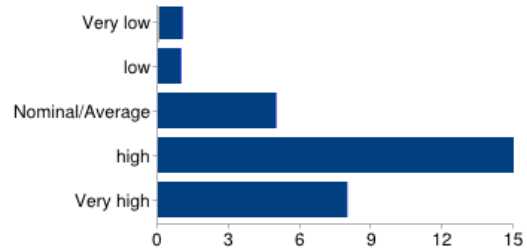

Fig. 2: Graphical Representation of question 1

Goal 1, Question 2: How much detecting data type mismatch improve component adaptation?

Table 3: Frequency of question 2

	Frequency	Percent
3	7	23%
4	14	47%
5	9	30%

The frequency Table 3 shows out of 47% questionnaires agreed, 30 % strongly agreed where as 23 % neither agreed nor disagree.

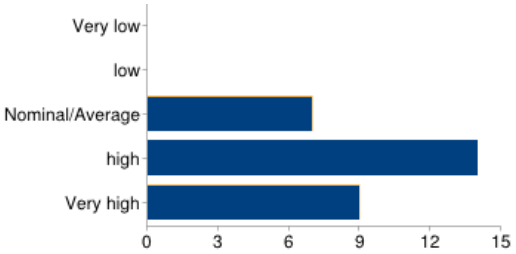

Fig. 3: Graphical Representation of question 2

Goal 1, Question 3: How much the current adaptation methods inefficient in detecting data type mismatch?

Table 4: Frequency of question 3

	Frequency	Percent
1	1	3%
2	2	7%
3	8	27%
4	9	30%
5	10	33%

As presented in frequency Table 4, 33% questionnaires strongly agreed, 30 % agreed where as 27 % neither agreed nor disagree. And 3% of people are disagreeing.

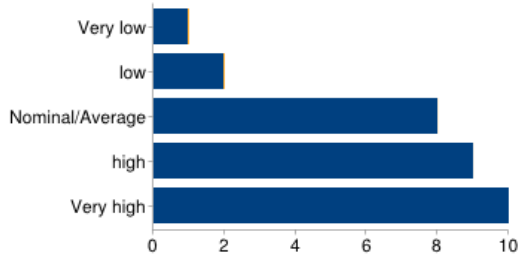

Fig. 4: Graphical Representation of question 3

Goal 1, Question 4: Do you think that the proposed solution will improve adaptation result?

Table 5: Frequency of question 4

	Frequency	Percent
3	8	27%
4	12	40%
5	10	33%

The frequency Table 5 shows out of 33% questionnaires strongly agreed, 40 % agreed where as 27 % neither agreed nor disagree.

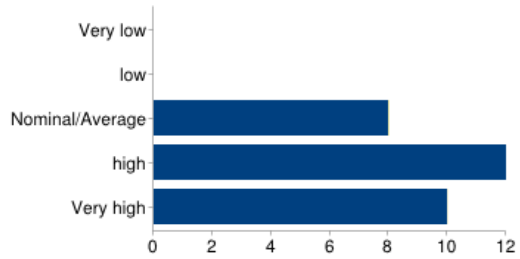

Fig. 5: Graphical Representation of question 3

Cumulative Survey of Goal 1

The result of the analysis of goal 1 is shown in Table 6.

Table 6: Cumulative Statistical Analysis of Goal 1

Q. number	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	1	1	5	15	8
2	0	0	7	14	9
3	1	2	8	9	10
4	0	0	8	12	10
Total	2	3	28	50	37
Average	**0.85%**	**0.85%**	**23.93%**	**42.74%**	**31.62%**

As shown in fig.6, it is clear from the cumulative descriptive analysis of goal 1 that 42.74% of the sample agreed that there is a need to focus on data type mismatch through adapt component behavior and 31.62% strongly agreed to that. In addition, 0.85 % disagreed and 0.85% strongly disagreed where 23.93% remained neutral.

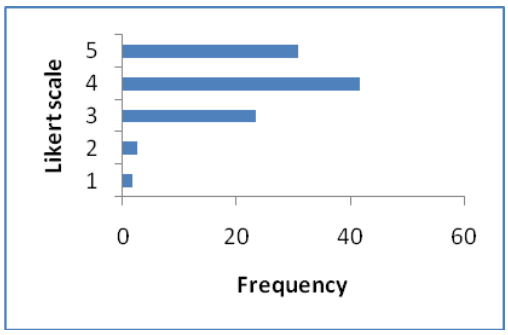

Fig. 6: Graph shown the Cumulative Results of Questionnaire for Goal 1

4.2 Cumulative Analysis of the Second Goal

Goal 2: considering data type through representing component interface. Question 5: Is specifying data type through describing component interface easy?

Table 7: Frequency of question 5

	Frequency	Percent
1	1	3%
2	4	13%
3	8	27%
4	10	33%
5	7	23%

The frequency Table 7 shows out of 23% people strongly agreed, 33 % agreed where as 27 % neither agreed nor disagree. In addition 13% of people are disagreeing, 3% strongly disagreed.

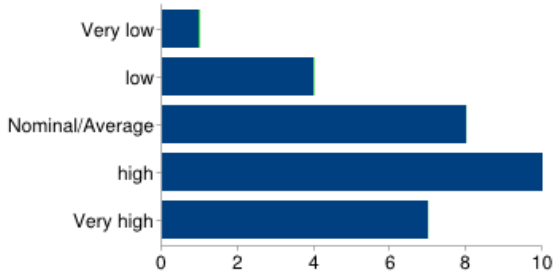

Fig. 7: Graphical Representation of question 5

Goal 2, Question 6: Is specifying data type through describing component interface dose not consuming time?

Table 8: Frequency of question 6

	Frequency	Percent
1	1	3%
3	9	30%
4	14	47%
5	6	20%

As shown in frequency Table 8, 20% of people are strongly agree, 47 % are agree while 30 % neither agreed nor disagree, and 3% are disagreeing.

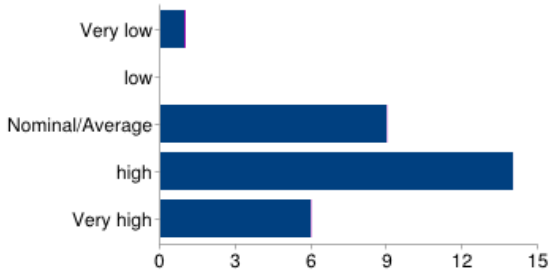

Fig. 8: Graphical Representation of question 6

Goal 2, Question 7: Is Symbolic Transition Systems (STS) efficient to represent component interface in adaptation process?

Table 9: Frequency of question 7

	Frequency	Percent
2	1	3%
3	8	27%
4	16	53%
5	5	17%

The Table 7 illustrates that 53% people agreed, 17 % strongly agreed where as 27 % neither agreed nor disagree. In addition 3% of people are disagreeing.

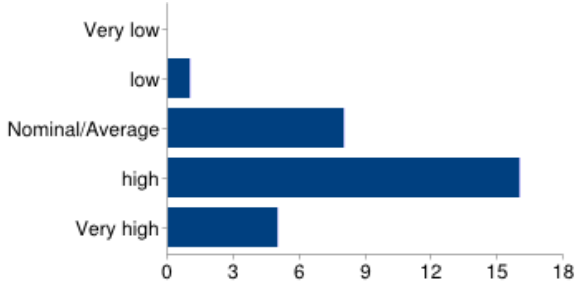

Fig. 9: Graphical Representation of question 7

Goal 2, Question 8: Is it efficient to use STS to specify data type?

Table 10: Frequency of question 8

	Frequency	Percent
3	9	30%
4	11	37%
5	10	33%

As presented in frequency Table 10, 33% questionnaires strongly agreed and 37 % agreed where as 30 % neither agreed nor disagree.

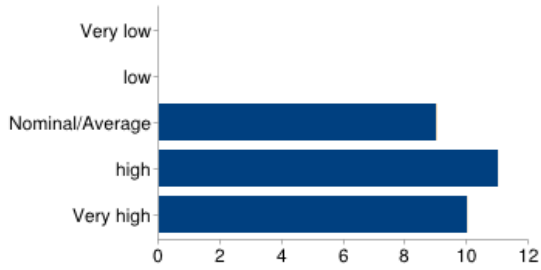

Fig. 10: Graphical Representation of question 8

Goal 2, Question 9: Do you agree that the time spend through considering data type in STS lead to save effort in the next step for detecting mismatch?

Table 11: Frequency of question 9

	Frequency	Percent
2	5	17%
3	10	33%
4	9	30%
5	5	17%

As presented in frequency Table 11, 17% of questionnaires are strongly agreed, 30 % agreed where as 33 % neither agreed nor disagree. And 17% are disagreeing.

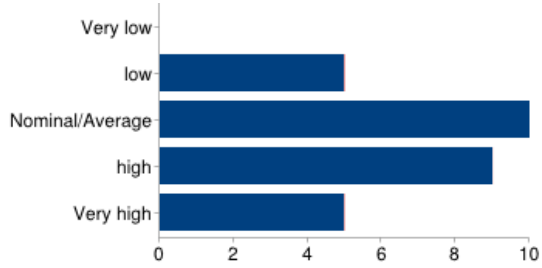

Fig. 11: Graphical Representation of question 9

Cumulative Survey of Goal 2

The result of survey of this goal is shown in Table 12.

Table 12: Cumulative Statistical Analysis of Goal 2

Q. number	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
5	1	4	8	10	7
6	1	0	9	14	6
7	0	1	8	16	5
8	0	0	9	11	10
9	0	5	10	9	5
Total	2	10	44	60	33
Average	**1.34%**	**6.71%**	**29.53%**	**40.27%**	**22.15%**

As shown in Fig. 12, 40.27% of people agreed that there is a need to specify data type by STS through representing component interface. While, 22.15 % strongly agreed, 6.71% disagreed and 1.34% strongly disagreed. The remained 29.53% are neutral.

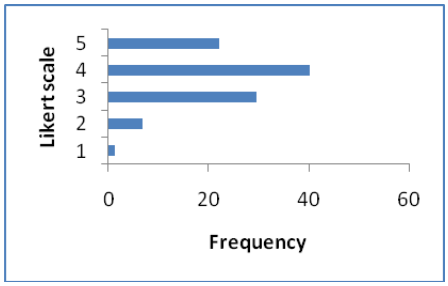

Fig. 12: Graph Shown the Cumulative Results of Questionnaire for Goal 2

4.3 Cumulative Analysis of the Third Goal

Goal 3: detecting data type mismatch.

Question 10: Is detecting data type mismatch difficult?

Table 13: Frequency of question 10

	Frequency	Percent
2	4	13%
3	11	37%
4	11	37%
5	3	10%

As shown in frequency Table 13, 37% of questionnaires are agreed and only 10 % strongly agreed where as 37 % neither agreed nor disagree. And 13% are disagreeing.

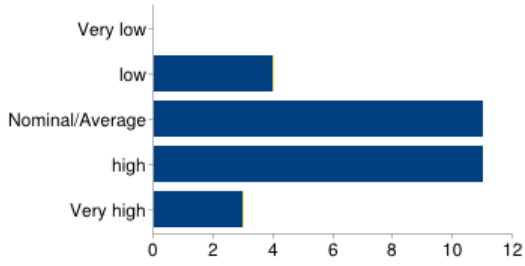

Fig. 13: Graphical Representation of question 10

Goal 3, Question 11: Is detecting data type mismatch consuming time?

Table 14: Frequency of question 11

	Frequency	Percent
3	11	37%
4	16	53%
5	3	10%

The frequency Table 14 shows that only 10% questionnaires strongly agreed while 53 % agreed, and 37 % neither agreed nor disagree.

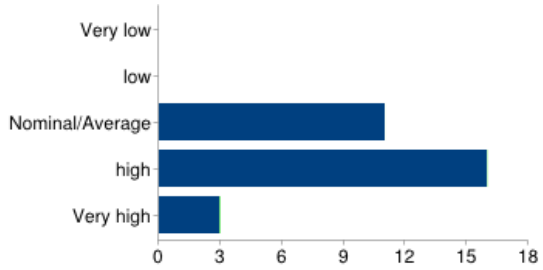

Fig. 14: Graphical Representation of question 11

Goal 3, Question 12: Is it efficient to consider data type through calculating synchronous vector to detect mismatch?

Table 15: Frequency of question 12

	Frequency	Percent
2	1	3%
3	11	37%
4	13	43%
5	5	17%

The frequency Table 15 shows out of 17% questionnaires strongly agreed, 43 % agreed where as 37 % neither agreed nor disagree. And 3% of people are disagreeing.

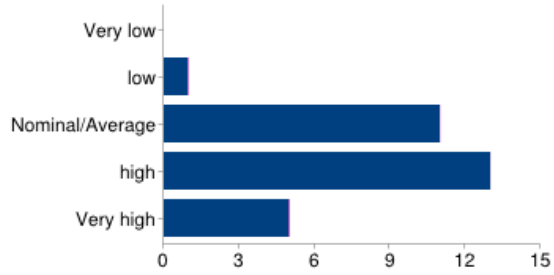

Fig. 15: Graphical Representation of question 12

Goal 3, Question 13: Do you agree that time spent to deal with data type in synchronous vector lead to provide better adaptation protocol?

Table 16: Frequency of question 13

	Frequency	Percent
2	3	10%
3	9	30%
4	12	40%
5	6	20%

The frequency Table 16 shows out of 20% questionnaires strongly agreed, 40 % agreed where as 30 % neither agreed nor disagree. And 10% of people are disagreeing.

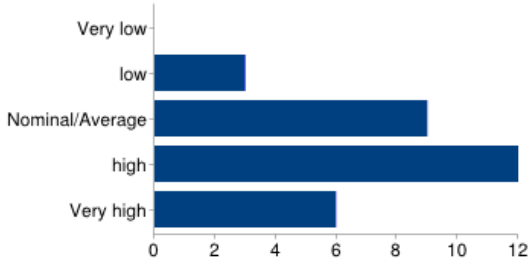

Fig. 16: Graphical Representation of question 13

Cumulative Survey of Goal 3

The result of survey of this goal is shown in Table 17.

Table 17: Cumulative Statistical Analysis of Goal 3

Q. number	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
10	0	4	11	11	3
11	0	0	11	16	3
12	0	1	11	13	5
13	0	3	9	12	6
Total	0	8	42	52	17
Average	0.00%	6.72%	35.29%	43.7%	14.29%

It is clear from the cumulative descriptive analysis of goal 3 that 43.7% of the sample agreed to detect data type mismatch and 14.29% strongly agreed while 35.29% remained neutral as shown in fig. 17.

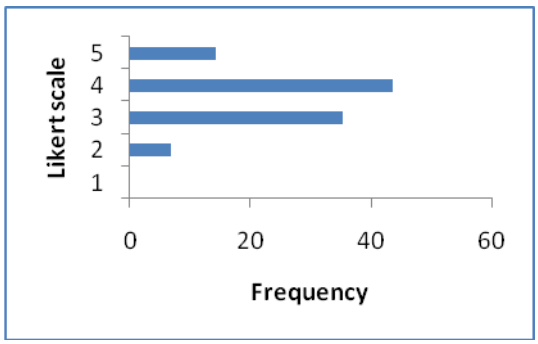

Fig. 17: Graph Shown the Cumulative Results of Questionnaire for Goal 3

4.4 Cumulative Analysis of the Three Goals

The result of survey of these three goals is shown in Table18. As shown in fig 18 that 41.75% of the sample agreed and 22.42% strongly agreed to that. In addition, 5.41% disagreed and 1.03% strongly disagreed where 29.38% remained neutral.

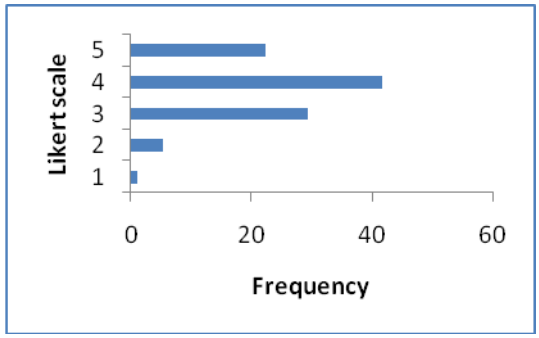

Fig. 18: Graph Shown the Cumulative Results of Questionnaire for the three goals

Table 18: Cumulative Statistical Analysis of the three goals

Q. number	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	1	1	5	15	8
2	0	0	7	14	9
3	1	2	8	9	10
4	0	0	8	12	10
5	1	4	8	10	7
6	1	0	9	14	6
7	0	1	8	16	5
8	0	0	9	11	10
9	0	5	10	9	5
10	0	4	11	11	3
11	0	0	11	16	3
12	0	1	11	13	5
13	0	3	9	12	6
Total	4	21	114	162	87
Average	1.03%	5.41%	29.38%	41.75%	22.42%

V. Conclusion

This paper focuses on detecting component behavior mismatch by proposing a solution and validates it by conduct a survey. The solution suggests focusing on data type through representing component interface by Symbolic Transition Systems (STS) and through calculating synchronous vector, where the behavior mismatch can be detected by synchronous vector that calculated after translating each component interface into STS.

As conclude from the cumulative analysis, data types have to consider through adaptation components. In addition, specifying data type through representation the component interface helps in the next step, for calculation the synchronous vector to detect data type mismatch, which provide better adaptor protocol.

In future, there is a need to consider data type through generating adaptor protocol and the implementing the adaptation algorithm in adaptor tool.

References

[1] Canal C, Poizat P, Salaun G. Model-Based Adaptation of Behavioral Mismatching Components, Software Engineering [J].IEEE Transactions on, 2008, 34(4):546-563.

[2] Zheng Jian, Jiang Jianhui. Model-Based Adaptation of Component Behaviors, Computational Intelligence and Software Engineering[C]. CiSE International, 2009, 1-4, 11-13.

[3] Qi Huacheng, Rong Mei, Zhang Guangquan. A behavior-driven model of component interaction adaptation[C]. Computer Science & Education. ICCSE '09. 4th International Conference on , 2009,871-875, 25-28

[4] Xiong Xie, Weishi Zhang, Xiuguo Zhang, Zhiying Cao, Jinyu Shi. Research on Safe Behavior Adaptation of Software Component[C]. Computational Intelligence and Software Engineering (CiSE), 2010,1-4, 10-12

[5] Xiong Xie, Weishi Zhang, Xiuguo Zhang, Zhiying Cao, Jinyu Shi. Safety Verification of Software Component Behavior Adaptation[C]. E-Product E-Service and E-Entertainment (ICEEE), 2010, 1-4, 7-9

[6] Xiong Xie, Weishi Zhang, Xiuguo Zhang, Zhiying Cao, Jinyu Shi. Research on Behavior Adaptation of Software Component, Parallel Architectures[C]. Algorithms and Programming (PAAP), Third International Symposium on , 2010,.412-416, 18-20

[7] Sae Hoon Kim, Jeong Ah Kim. Component Adaptation Mechanism[C]. Ubiquitous Computing and Multimedia Applications (UCMA), 2011, 90-95, 13-15

[8] Guorong Cao, Qingping Tan; Jingang Xie. A New Approach to Component Adaptation Dealing with Extra-Functional Mismatches[C]. Information Engineering and Computer Science, ICIECS, 2009, 1-4, 19-20

[9] Segarra M T, Andre F. A Distributed Dynamic Adaptation Model for Component-Based Applications[C]. Advanced Information Networking and Applications, AINA '09. International Conference on 2009, 525-529, 26-29

[10] Borde E, Carlson J. Automatic Synthesis and Adaption of Gray-Box Components for Embedded Systems - Reuse vs. Optimization[C]. Computer Software and Applications Conference Workshops (COMPSACW), IEEE 35th Annual , 2011, 224-229, 18-22

Appendix A

Questionnaire about Detecting Component Behavior Mismatch

Rizwan J. Qureshi, Ebtesam A. Alomari

Information Technology Department, King Abdul Aziz University

To adapt component, there is a need to detect mismatch.

Problem: The current adaptation model focuses on massages name mismatch. There is a need to improve this method to detect the data type mismatch.

Proposed Solution:

-The paper suggests focusing on specifying data type through representing component interface by Symbolic Transition Systems (STS)

-And consider data type through calculating synchronous vector, where the behavior mismatch can be detected by synchronous vector that calculated after translating each component interface into STS.

Note:

-STS is a graph tool to describe the component behavior.

-Synchronous vector denote communication between several components, where each event appearing in one vector is executed by one component.

To validate the proposed solution this questionnaire divided into 3 groups which are:

-Focusing on data type through adaptation components.

-Considering data type through representing component interface.

-Detecting data type mismatch.

I hope you help in answer the following questions.

Filling guideline

Likert scale is ranging from 1 to 5.

Very low effect indicating 1

Low effect indicating 2

Nominal/Average effect indicating 3

Very high effect indicating

*** Focusing on data type through adaptation components**	
1- Is it important to consider parameter's data type through adaptation process?	□Very low □ Low□ Nominal/Average □High □ Very high
2-How much detecting data type mismatch improve component adaptation?	□Very low □ Low□ Nominal/Average □High □ Very high
3-How much the current adaptation methods inefficient in detecting data type mismatch?	□Very low □ Low□ Nominal/Average □High □ Very high
4-Do you think that the proposed solution will improve adaptation result?	□Very low □ Low□ Nominal/Average □High □ Very high
*** Considering data type through representing component interface**	
5- Is specifying data type through describing component interface easy?	□Very low □ Low□ Nominal/Average □High □ Very high
6- Is specifying data type through describing component interface dose not consuming time?	□Very low □ Low□ Nominal/Average □High □ Very high
7-Is Symbolic Transition Systems (STS) efficient to represent component interface in adaptation process?	□Very low □ Low□ Nominal/Average □High □ Very high
8- Is it efficient to use STS to specify data type?	□Very low □ Low□ Nominal/Average □High □ Very high
9- Do you agree that the time spend through considering data type in STS lead to save effort in the next step for detecting mismatch?	□Very low □ Low□ Nominal/Average □High □ Very high
*** Detecting data type mismatch**	
10- Is detecting data type mismatch difficult?	□Very low □ Low□ Nominal/Average □High □ Very high
11- Is detecting data type mismatch consuming time?	□Very low □ Low□ Nominal/Average □High □ Very high
12- Is it efficient to consider data type through calculating synchronous vector to detect mismatch?	□Very low □ Low□ Nominal/Average □High □ Very high
13-Do you agree that time spent to deal with data type in synchronous vector lead to provide better adaptation protocol?	□Very low □ Low□ Nominal/Average □High □ Very high

Authors' Profiles

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

Ebtesam Ahmad Alomari: Master student in IT Department at King Abdulaziz University, interested in CBD and Technology Management.

How to cite this paper: M. Rizwan Jameel Qureshi, Ebtesam Ahmad Alomari,"Validation of Novel Approach to Detect Type Mismatch Problem Using Component Based Development", International Journal of Information Technology and Computer Science(IJITCS), vol.5, no.9, pp.108-117, 2013. DOI: 10.5815/ijitcs.2013.09.12