

# A MODEL FOR RUN-TIME SOFTWARE ARCHITECTURE ADAPTATION

Fatemeh Khorasani<sup>1</sup>, Afshin Salajegheh, Ph.D<sup>2</sup>, Ali Moeini, Ph.D<sup>3</sup>,

1. M.Sc Student of Islamic Azad University South Tehran Branch, Tehran, Iran

2. Assistant Professor of Software engineering and Computer Science Islamic Azad University Tehran South Branch, Tehran, Iran

3. Associate Professor of Computer Science Tehran University, Tehran, Iran

## **Abstract:**

*Since the global demand for software systems and constantly changing environments and systems is increasing, the adaptability of software systems is of significant importance. Due to the architecture of software system is a high-level view of the system and makes the modifiability possible at an overall level, the adaptability of the software can be considered an effective approach to adapt software systems by changing architecture configuration. In this study, the architecture configuration is modified through xADL language which is a software architecture description language with a high flexibility. Software architecture reconfiguration is done based on existing rules of rule-based system, which are written with respect to three strategies of load balancing, fixed bandwidth and fixed latency. The proposed model of the study is simulated based on samples of client-server system, video conferencing system and students' grading system. The proposed model can be used in all types of architecture, include Client Server Architecture, Service Oriented Architecture and etc.*

## **Keywords:**

*Software architecture, Software adaptation, Rule-Based system, ADL, xADL*

## **1-INTRODUCTION**

Environmental changes in the world today have made the adaptability of software systems an integral principle of the software. Responding to these changes at every level of the software is possible, but the software architecture level can have a significant role in the adaptability of the system because it is the highest level of software and have an overview of the software at a high level [3]. The adaptability of the software architecture means architectural configuration change so that it makes the system resistant to changes through new configuration.

IBM has provided a general framework for adaptability that can be seen in Figure 1. The framework consists the following four phases: Monitor, Analysis, Plan, and Execute (MAPE). Thus, the system is first monitored the status of the running system is reviewed. When a quality constraint is violated, the system enters the analysis phase to evaluate the reason for this violation,

and then in the next phase, the adaptation operation appropriate to the violated limitation is selected and goes to Execute phase. The four phases make the system resistant to change [2].

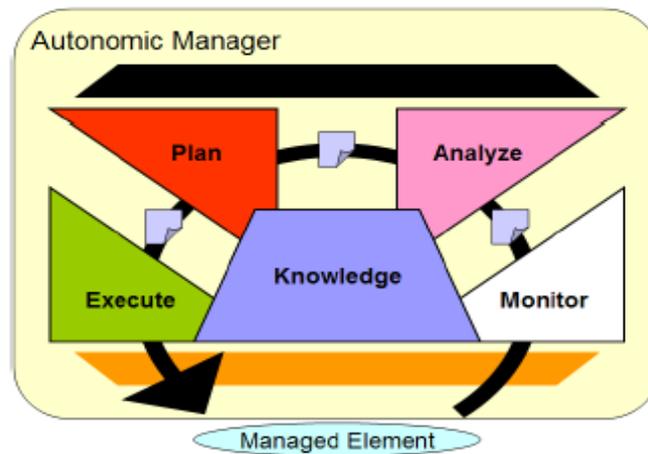


Figure 1: IBM Framework for adaptation

The systems can be adapted in each layer of the software where the software architecture is the highest level and can be considered as a software level to adapt the system. Software architecture can be known as a set of software components, subsystems, communications, interactions, characteristics of each component and a set of guiding principles that both sets form a series of basic features for a software system.

The architecture of a software system includes components, connectors, interfaces, links, and configuration [8]. To describe the architecture software and change the configuration, an architecture description language is required. An architecture description language (ADL) is a language (graphics or text, or both) to describe software systems with architectural elements and the relationships between them [8]. The architecture description language includes AADL, ACME, Rapide, Wright, xADL and etc. Due to the fact that changes in the architectural configuration of running software is desired, a language should be used by which the software architecture can be changed at the time of running.

To decide upon the current situation of the system how to respond to it, the rule-based systems are used. Rule-based systems are the systems in which the inference engine decides what happens in the system in the current situation using the facts contained in the database (data obtained from the environment and the running system) and rules in the knowledge base (written as conditions and quality constraints of software system). The rules are written in these systems as if <antecedent> then <consequent>. The condition of the rule is at the antecedent part, that if the condition is true, the consequent rule runs.

Garlan et al. [2] have proposed a comprehensive model called Rainbow to adapt architecture-based software systems that runs the adaptation process automatically.

The study aims to adapt software systems by an architecture configuration change. For architecture description of the software and changes in the configuration the xADL architectural description language has been used and conditions necessary to avoid reducing the efficiency of the system are considered in the form of rules, in the knowledge base of the rule-based system. The study has provided recommendations to optimize matching strategies to deal with changes in strategies by adopting a wider variety of tactics. Finally, to simulate the proposed model the case samples used for the Rainbow model has been used [3].

## 2-PROPOSED MODEL

The purpose of the model is the adaptability of the software architecture. Rule-based system for decision-making [6] and xADL language to describe the architecture [1] is used. The model is presented in framework proposed by IBM i.e. MAPE.

Performance of the proposed model in each phase is:

**Monitoring:** In this phase, the model checks the environment via the probes and collects information and gives them to the gauges, then the gauges measure the desired parameters and sends them to a rule-based system where they are placed in the database of the rule-based system.

**Analysis:** The rule-based system analyzes which limitations and conditions have been violated from normal state of the system which has removed the system from its natural state based on the predefined rules that exist in the knowledge base and the facts obtained by measurements. Note that if the system is working without any violated limitation the model then goes on to the monitoring of the system. However, if the violation occurred the model enters the next phase of planning.

**Planning:** In this phase, model searches for the desired adaptation strategy and thus the desired tactic and also its examination in xADL language based on the output of the rule-based system and send it to the execute phase.

**Execute:** In this phase, the model using effectors, changes the architecture configuration, depending on the desired tactics.

This model can be seen in Figure 2.

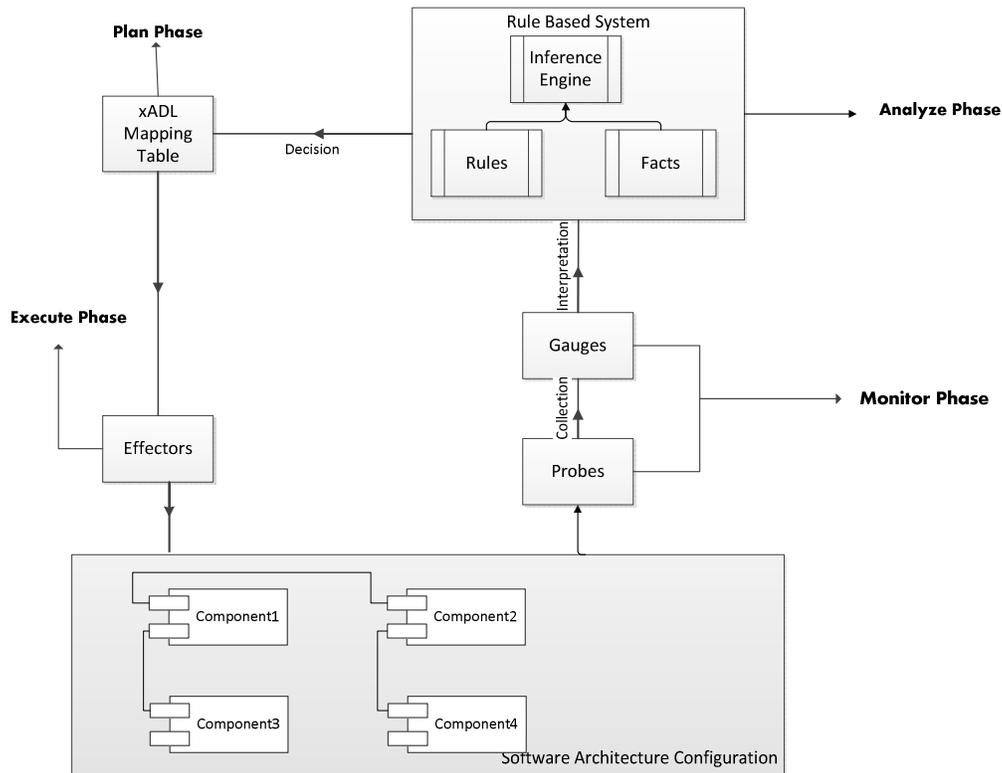


Figure 2: Proposed architectural model for software architecture adaptation

During the trend the four phases of system can be resistant to change and respond to changes in run mode.

However it should be noted that the rules of the rule-based system as mentioned in [4], must be written appropriate to adaption strategies. Desired strategies used in this study include three strategies of load balancing, fixed bandwidth, and fixed latency.

### 1-3-Load Balancing Strategy

The load balancing strategies flow is shown in Figure 3.

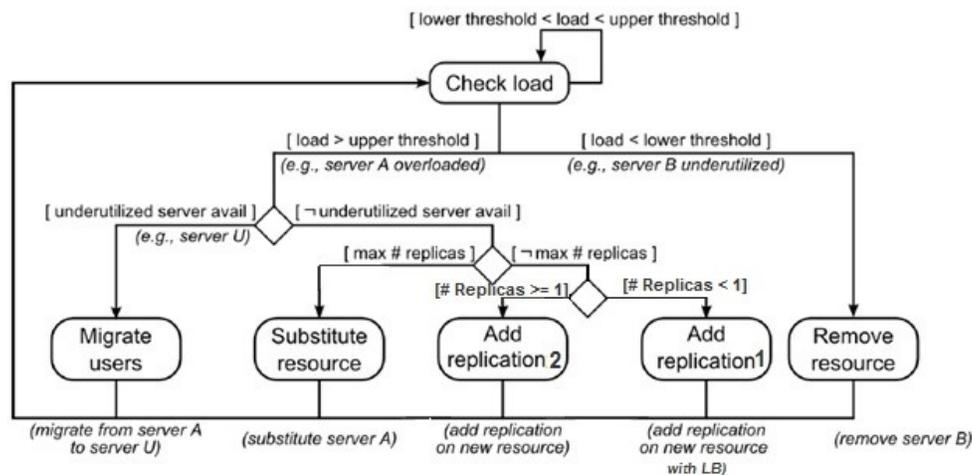


Figure 3: Load balancing strategy

In this strategy, at the first stage a list of all service providers is available. At each time slice when the system is monitored at each time slice, each service provider's load is measured. Model considers a threshold for the number of replications of each service provider and does not allow creating more than a certain number of replications. Because according to architectural environments, creating replications more than a certain number reduces efficiency and wastes space. When load is reduced and becomes lower the threshold, the service provider can be called underutilized and in this case, its mark is removed from the list of service providers and it is known as underutilized. It should be noted that if a service provider announced as underutilized, this means that the service provider replied to all requests for service. This means that if the service provider has a request to answer or bring it to a service provider that is working or low-load threshold is considered that there is no request for a response. In these two cases a service provider can be seen as underutilized. This tactic is known as the Remove Resource which is used to eliminate unnecessary service providers. If the load of a service provider exceeds a certain threshold search begins in the list of the providers and if there was an underutilized service provider and was able to respond to the current consumer, the consumer is migrated to it [5]. This tactic is known as the User Migration in which the consumer connections are replaced seamlessly between two service providers in the same area. If the model reached the unmarked service provider in the previous stage it meant that the search was aborted, the number of replications is checked. Then it checks whether the previous replication has existed for resource, or no replications has been created yet. If no replication has been created the model would create a new replication and add a load balancing component to it to divide the load between the available resources. This tactic can be seen in the diagram with the name of the Add Replication 1. But if a replication of the source has been already created and there was load balancing component and the maximum number of replications is not reached yet, new replication of the resource can be made connected to an existing load balancer. This tactic is called the Add Replication 2. At the last step if it was not allowed to create a replication of a resource the fifth tactic, i.e., Resource Substitution occurs that means the model replaces overloaded service provider with stronger service provider.

User migration and resource substitution tactics are the same in terms of code but are different in the way they are used that user migration takes place only when a component exists in the list of underutilized components that is able to respond to requests but the model to use resource substitution tactic seeks for stronger components without a component being underutilized and will replace it with the current component, this means that the replacement components may also be responding to requests.

## 2-2-Fixed Bandwidth Strategy

Bandwidth strategy diagram is shown in Figure 4:

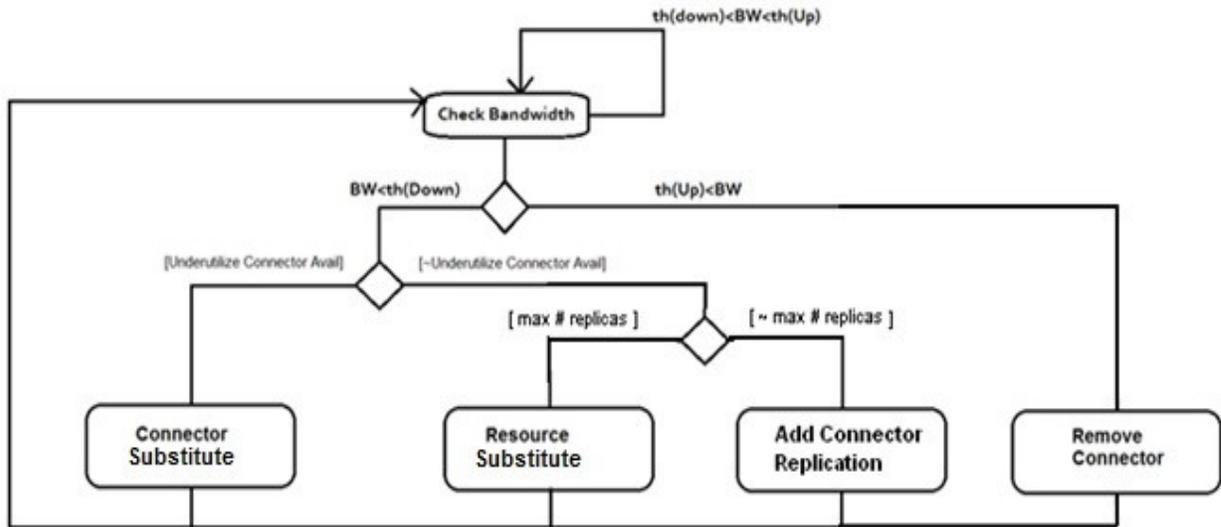


Figure 4: Fixed bandwidth strategy

In this strategy, the bandwidth of each working connection is investigated and compared with the threshold. If the bandwidth of a connection was higher than the threshold the connection is added to the list of high bandwidth connections to be used as necessary. When removing the connection it should be noted that there are no services transferring on the connection. This tactic is the first tactic called Remove Connector. In the evaluation of the bandwidth of each connection if the connection bandwidth is lower than the below threshold three tactics can be used. The first tactic is to search for high-bandwidth connections in the list and substitute low-bandwidth connections with high-bandwidth connections; this tactic is called Connector Substitute. The second tactic is to substitute the service provider component with a component that can respond with low bandwidth (Resource Substitute) [6], in the third tactic a replication is made of desired connection to create the same bandwidth and the original bandwidth can be doubled (Add Connector Replication).

## 2-3-Fixed Latency Strategy

Fixed latency strategy diagram [3] is Figure 5:

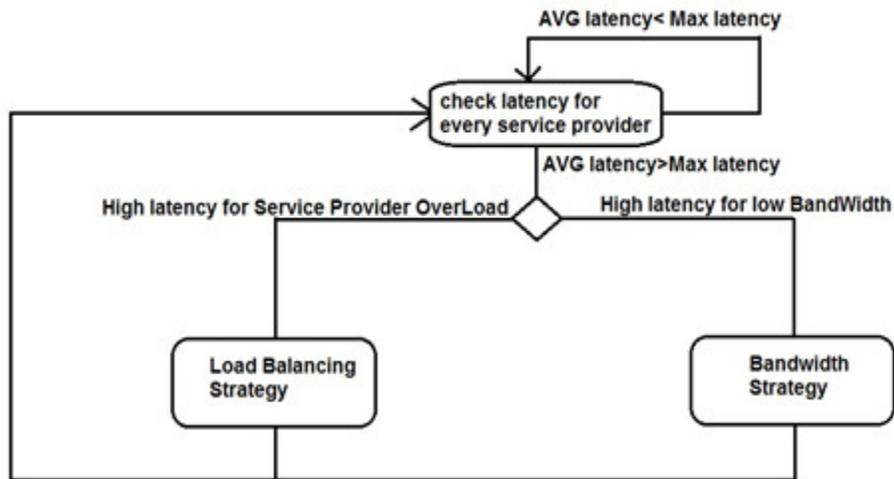


Figure 5: Fixed latency strategy

Latency to every service consumer by the service provider is measured; if the latency goes much higher than a given limit it means that the environment has undergone undesirable changes that must be responded. Two reasons may be accounted for; At first glance, increased latency may be due to overhead load of the service provider in which model goes on to load balancing strategy. On the other hand, increased latency may be due to communication delay (reduced bandwidth of the connection); in this case, the model seeks to apply the strategy of bandwidth. Case can be considered in which the increased latency is because of the problem in both load and bandwidth, thus if any of the problems are recognized it first goes on to the corresponding strategy and after executing the tactic and at the end of the first strategy, if the latency is not entered the desired range yet, model goes to another strategy and executes the relevant tactic.

According to the tactics mentioned, table of tactics can be formed as follows:

Table 1 Tactics in proposed model

<b>Tactic number</b>	<b>Tactic Name</b>	<b>Tactic operation</b>
Tactic 1	Remove Resource	Underutilized resources are removed and are added to Underutilized list.
Tactic 2	User Migration	Clients connections are switched seamlessly between two application servers replicating the same zone
Tactic 3	Replication Enactment1	New servers are added for more computing power in order to making replicates and a load balancer component.
Tactic 4	Replication Enactment2	New servers are added for more computing power in order to making replicates
Tactic 5	Resource Substitution	Running resources are substituted by more powerful resources
Tactic 6	Remove Connector	Underutilized connectors are removed and are added to Underutilized list
Tactic 7	Connector Substitution	Running connectors are substituted by more powerful connectors.
Tactic 8	Add Connector Replication	Connector is replicated for making high bandwidth.

Rules of the rule-based system can be written with respect to the strategies and tactics of interest as follows:

Table 2: Rules in Rule - Based System

Rules #	Rules
Rule 1	If $Load_{Down} \leq Load \leq Load_{Up}$ Then Back to probe;
Rule 2	If $Load < Load_{Down}$ Then Tactic 1;
Rule 3	If $Load > Load_{Up}$ And Underutilized Server List Is not Empty Then Tactic 2;
Rule 4	If $Load > Load_{Up}$ And Underutilized Server List Is Empty And Number of Replica Is Not MAX And Number of Replica $< 1$ Then Tactic 3 ;
Rule 5	If $Load > Load_{Up}$ And Underutilized Server List Is Empty And Number of Replica Is Not MAX And Number of Replica $\geq 1$ Then Tactic 4 ;
Rule 6	If $Load > Load_{Up}$ And Underutilized Server List Is Empty And Number of Replica Is MAX Then Tactic 5;
Rule 7	If $BW_{Down} \leq BW \leq BW_{up}$ Then Back to Probe;
Rule 8	If $BW > BW_{Up}$ Then Tactic 6;
Rule 9	If $BW < BW_{Down}$ And Underutilized Connector List IS Not Empty Then Tactic 7;
Rule 10	If $BW < BW_{Down}$ And Underutilized Connector List IS Empty And Number of Replicas Is Max Then Tactic 5;
Rule 11	If $BW < BW_{Down}$ And Underutilized Connector List IS Empty And Number of Replicas Is not Max Then Tactic 8;
Rule 12	If AVG Latency $<$ Max Latency Bach to Probe;
Rule 13	If AVG Latency $>$ Max Latency And High Latency is for low Bandwidth Then Rules 9-10 - 11 Can be used ;
Rule 14	If AVG Latency $>$ Max Latency And High Latency is for service Provider Overload Then Rules 3-4-5-6 Can be used ;

### 3-CASE STUDY

The proposed model is simulated based on samples of video conferencing system, University Grading system and Client-server system and its architecture describing codes are written in xADL language.

#### 3-1-Client-Server System

One case study on which the model process can be shown is the client-server model. This includes Clients who connect to a server by a connection and the server group is the head of several server components, for example, in the client-server system there are six clients and two server groups that the server group No.1 is attached to the three server components which receive

respond from the server if receiving a request and return it to the desired Client. Figure 6 shows a client-server system at the initial state.

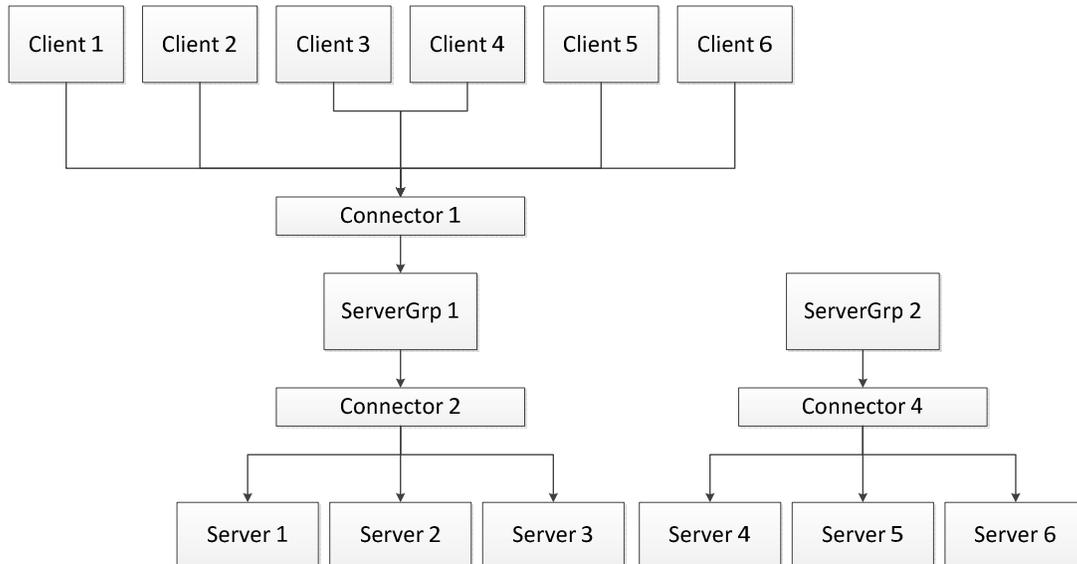


Figure 6: client - server system in initial state

**Scenario** Suppose that in the typical client-server system, by default, the request-response time of every Client is always measured at system run-time. When monitoring once model finds that the response time to request of Client 3 is higher than the threshold then model enters a new phase and track the limitation violation at the rules. According to rules 13 and 14 one shall follow the reason for the latency. Now suppose when monitoring system, the model finds that the latency is due to the heavy load of the responding server. As a result, the antecedent part of rule No. 14 has been created. According to the rules of the rule-based system, the consequent of the rule must be executed. The consequent of the rule goes to rules 3 to 6. Now model seeks to examine other characteristics. In the current system there are underutilized server groups that can respond to requests of Clients. The underutilized server group is group two. Server group two have also other server groups. As a result the tactic 2 i.e. user migration is executed (3) Therefore, Client No.3 is transferred to underutilized server group. The new architecture can be seen in Figure 7.

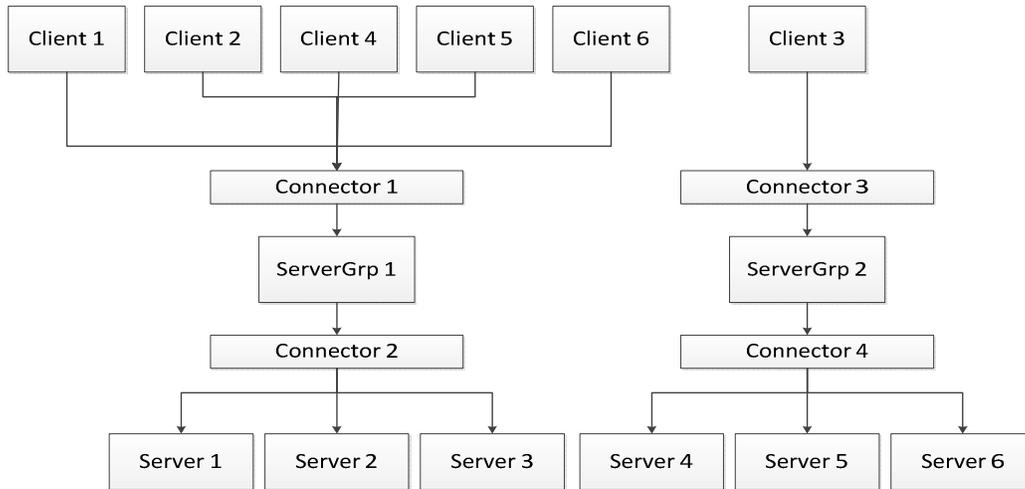


Figure 7: Client - server system with User Migration

In this scenario, assume that response-request time that the Client (1) experiences, is greater than the threshold and then system monitoring is characterized by model and the violation of the limitation is due to a communication delay. Thus, the model in the rule-based system executes condition 13 after examining the system. Model considers Bandwidth strategy in the condition 13. Then model checks the bandwidth conditions to decide which condition is realized in the system. Suppose the list of system underutilized connections is empty in this case, because there is no replication of the connection the model can create a replication of the connection with low bandwidth leave the Client to it, as a result tactic 8 is run. According to what was said client-server system is changed as Figure 8 and the latency is improved and the system problem is resolved.

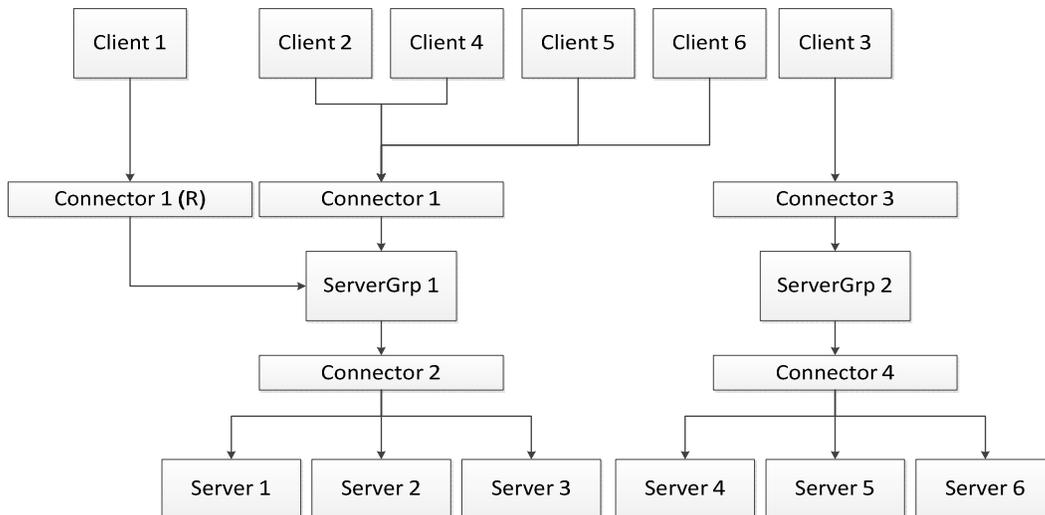


Figure 8: Client - server system with Connector Replication

All of the software architectures that is shown describes with xADL codes and design through Archstudio software.

#### 4-CONCLUSION AND FUTURE WORKS

This research tries to find a solution for software architecture adaptation against change. This was possible with proposed model to deal with three basic changes in software architecture. It has used flexible language xADL for describing software architecture and rule based system for decision making. Three strategies are intended that for each tactics are proposed. There are 8 tactics that are proposed in strategies and 14 rules that are written in rule based system. The proposed model is simulated through Archstudio software and xADL codes on three case studies for evaluation.

Finally, the proposed model can be compared in the following table with similar cases.

Table 3: compare proposed model with existing model

Parameters	Flexible ADL	Ability of development	Analyzable	Options in strategies	Easy to use	Generality	Automatic operation
Models							
Rainbow	×	✓	✓	×	✓	✓	✓
MADAM	×	✓	✓	×	×	✓	✓
Proposed Model	✓	✓	✓	✓	✓	✓	✓ *

\*The proposed model currently does not perform automatic adaptation operation but it would be possible in the future.

The proposed model can be automated by designing software for the above steps are performed automatically. On the other hand, this model can be expanded to other criteria such as security or cost, or other criteria based on their corresponding strategy.

#### REFERENCES

- [1] Eric M. Dashofy , January 2003 ,xADL 2.0 Distilled , Available from : <http://www.isr.uci.edu/>
- [2] Cheng , S-W. , 2008 ,Rainbow: Cost-Effective Software Architecture-Based Self-Adaptation , Ph.D. thesis , Carnegie Mellon University .
- [3] Garlan , D. , Schmerl , B. , Cheng , S-W. ,2008 , Chapter 1 Software Architecture-Based Self-Adaptation.
- [4] Sasikumar , M. , Ramani , S. , Muthu Raman , S. , KSR Anjaneyulu , Chandrasekar , R. ,2007 , A Practical Introduction to Rule Based Expert Systems , Narosa Publishing House, New Delhi .

- [5] D. Meiländer, A. Bucchiarone, C. Cappiello, E. Di Nitto and S. Gorlatch , 2011 , Using a Lifecycle Model for Developing and Executing Real-Time Online Applications on Clouds , University of Muenster, Germany Fondazione Bruno Kessler, Italy Politecnico di Milano, Italy .
- [6] Zheng , Y. , 2014 , ArchStudio , Software Methods and Tools , CS 490MT/5590MT .
- [7] Canal , C. , Murillo, J.M. , Poizat , P. , 2006 , Software Adaptation , University of Málaga, Department of Computer Science . RSTI - L'objet. WCAT'04, pages 9 to 31.
- [8] Bijlsma , A. , Heeren , B.J. , Roubtsova , E.E. , Stuurman , S. ,2011 , Software Architecture .