비 기능적 특성 강화를 위한 새로운 엔터프라이즈 아키텍처

매니쉬*, 최승준*, 이봉재*, 박종서*

요약

엔터프라이즈 아키텍처의 주목표는 기술의 발전과 비즈니스 프로세스의 변화에 따르는 아키텍처 구성요소들을 조정하고 연결시켜 간극을 좁히는데 있다. 하지만 이러한 지속적인 연구, 대중화에도 불구하고 기존의 엔터프라이즈 아키텍처에는 신뢰성, 경제성, 유지보수, 상호운영성에 관한 단점들을 여전히 가지고 있다. 비즈니스와 기술의 변화에 따라 대처하는 것은 매우 중요하다. 본 논문에서는 서비스 지향 아키텍처 (SOA), 제품계열 아키텍처 (PLA), 전문가 시스템 (ES), 클라우드 컴퓨팅의 사용에 있어 주요한 문제를 논하고자 한다. 본 논문에서 제안하는 아키텍처는 적응성, 재사용성, 유지 보수성 및 확장성과 같은 특징들을 제공한다.

A Noble Enterprise Architecture to enhance the Non-Functional Properties

Manish Pokharel*, Seung-Jun Choi*, Bong-Jae Lee*, and Jong-Sou Park*

ABSTRACT

The main objective of Enterprise Architecture (EA) is to coordinate among the components of the architecture and to bridge the gap between the changes in business process and advancement of technology. Despite its use, popularity and continuing researches, there are still shortcomings in exiting EA in the areas such as addressing legacy system, reliability, affordability, maintainability, and interoperability. Adaptation in line with changes in business process and technology is still considered a critical challenge. In this paper, we address such critical challenges with use of Service Oriented Architecture (SOA), Product Line Architecture (PLA), Expert System (ES) and also Cloud Computing. The proposed architecture provides the features such as adaptability, reusability, maintainability and scalability.

Key Words : Enterprise Architecture, Service Oriented Architecture, Cloud Computing, Product Line Architecture, e-government

* 한국항공대학교 컴퓨터공학과 (✉manishpokharel@gmail.com)
· 제1저자(First Author) : 매니쉬(Manish Pokharel) · 교신저자(Correspondent Author) : 매니쉬(Manish Pokharel)
· 접수일(2010년 11월 19일), 수정일(1차 : 2010년 12월 31일), 게재확정일(2011년 1월 4일)

Ⅰ. Introduction

One of the major problems in using ICT in majority of the sectors including government offices is concerned with the issue of the degree of adaptability of ICT tools, with their systems and applications. The innovation in technologies and changing demand of users poses additional challenges in managing the technology and the system and this has been a serious concern for the experts. E-government, which is by its very nature a multi-stakeholder system requires a serious thought on adaptability of technologies with the system. The importance of the requirement of adaptability has drawn significant attention because the concept of e-government is heading towards the philosophy of connected government and is expected to provide seamless services. The existing Enterprise Architecture (EA) provides the government to be connected and alleviates some of these challenges of adaptability to some extents but not in its entirety.

The main goal of EA is to achieve the organization's mission through the optimal performance of its business process with optimal use of ICT tools and technologies.

In this paper, we introduce Enterprise Architecture and cloud computing in the beginning of this paper. We put the current work done in this area in Section 2 as a literature review. The existing problems of Enterprise Architecture are given in Section 3 and propose the solution as a new architecture in Section 4. We use Markov model to analyze one part of architecture and use SHARPE [1] as a modeling tool to verify our obtained data in Section 4. The features of proposed architecture are given in Section 5 We use cloud computing in a part of our proposed new architecture.

Ⅱ. Literature Review

Enterprise Architecture has become an established discipline both in industry and academia [2]. It is the architecture of both business process and Information Communication Technology (ICT) services delivery platform. During 1987, an IBM researcher J.A. Zachman initiated the concept of Enterprise Architecture. He mentioned the need of architecture to bind the different components in order to meet the changing business environment in his framework [3]. He proposed a two-dimensional scheme in his framework, the columns for abstractions and the rows for perspectives. Many organizations later started adopting the same framework with little or no modifications. Adhering to the basic concept of Zachman's framework, the US developed the Federal Enterprise Architecture Framework (FEAF) for their Enterprise Architecture in 1999. The main objective of this framework was to provide the guidelines to the Federal agencies in initiating, using, developing, and maintaining the EA [4]. The EA tries its best to meet the goal of business process with available technologies. The technology like Service Oriented Architecture (SOA) is one of the key technologies in EA [5].

As the popularity of EA increases, more challenges have emerged regarding its interoperability in semantics and syntax, reliability, security, disaster recovery, etc. The existing EA claims to have addressed the issues of reliability, security, disaster recovery and adaptability but the level of such features are not convincing. Reliability is achieved by incorporating redundancy in the system. Security is achieved to some level through the traditional security approaches. In order to obtain such non-functional properties in more satisfying level, a significant work has been done with virtualization technology in other areas of system development. None of the existing EAs is based upon virtualization. Virtualization makes the system more portable, durable, reliable and available [6]. These properties are essential for any EA. Stephen C. Gay has put the possibilities of using virtualization technology in managing disaster recovery [7].

III. Problems in EA

We have analyzed the present available EAs and identified the problems currently existing. There are a number of problems such as legacy to traditional system, syntactic and semantic interoperability, horizontal and vertical interoperability, security, reliability and many more. Some of them are critical and some of them are not. In spite of maximum use and high level of maturity, there are still many problems other than these in existing EA but our focus is on addressing the issue of adaptability. It covers all most all problems in EA. Ensuring high adaptability can also address the critical problems like modifiability, mobility, integrity and scalability.

3.1 Adaptability

Adaptability is the property of EA that can change itself according to the change in technology and change in business processes. Existing EA can predict the future technology and future business process but a small change requires significant change in the existing technical infrastructure, such as hardware, software, telecommunication, policy, business process etc. The impact of small change is tremendous. It induces the stress and risks in the architecture itself. The present EA cannot handle the change nor can it be updated by itself. It has very low adaptability. So, whenever there is a change in business process, it necessitates corresponding changes in technical, software, data architectures. To achieve maximum adaptability, we propose new EA.

IV. Architecture

In order to get maximum adaptability in the architecture we need to ensure features like high availability, high survivability and maximum reuse. We have incorporated such features in our proposed new architecture.

The features like availability, survivability, reusability and minimum downtime collectively makes a system adaptable. Absence of any one of those features does not ensure high adaptability in architecture. We propose the following architecture in Section 4.1 to address the above mentioned features.

4.1 The Proposed Architecture

This architecture is designed to address the existing problems and incorporate the required features of adaptability in EA.

The architecture depicted in Figure 1 is divided into three tiers. Each tier has been assigned significant role. Here, we consider electronic government as a case study. There are two main units on it. One is service provider and another is service user. We have developed this architecture with the consideration of service provider and service user as two main actors.

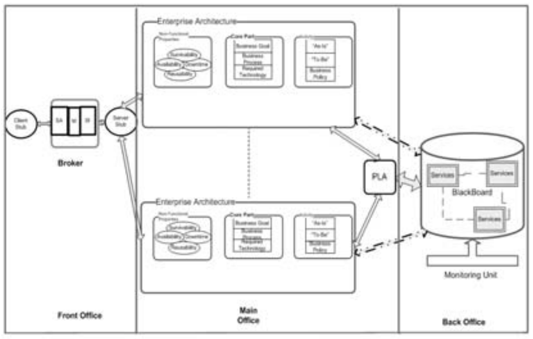

그림 1. 메인 아키텍처
Fig. 1. The Main Architecture

We see the first tier is "Front Office" for user section, second tier is "Main office" for Enterprise Architecture with Product Line Architecture (PLA) application, and third tier is "Back office" for common repository for enterprises. Each tier has been assigned significant role but back office is more important as compared to others in this architecture because it consists of sensitive government data, information, and services. We give special attention to it.

4.2 The Front Office

This tier belongs to the users of the system. In our system, citizen is a user of the system. Citizen sends request for service at the beginning of this tier. The request goes to the client side stub. It converts the citizen's request into system

understandable format i.e. marshal the request and convert it into standard format and send it to the broker. The request is accepted by the service assembly (SA) at the broker. The broker consists of the database of the government offices along with the list of services provided by them. The middleware in the broker checks citizen's request and verify whether the requested service is available in the service provider domain or not. If it finds the service then the service interface (SI) checks the interface and forwards such request to the service provider after de marshalling it with the help of server side stub. Here, broker needs some time to conduct Search. Once the search is complete then it forwards to the server side stub and finally to the enterprise.

The enterprise (government office) in the main office needs to be updated about its services to the broker regularly. It is not shown in the Figure 1 but it is assumed that this is done periodically. Here in this tier, there are three main components i.e. client side stub, broker and server side stub. These components interact with each other with Simple Object Access Protocol (SOAP) protocol and interfaces are designed in Web Services Description Language (WSDL).

4.3 Service Oriented Architecture in EA

Since the proposed architecture is based upon the SOA, each component plays the role in service receiving and service providing as needed. A component may be a service requester and at the same time it may be a service provider as well. It is possible in SOA through the concept of Meyer's Contract Model. In our architecture a broker is a service provider for client and at the same time it is a service requester for the server stub and the same applies for enterprise and blackboard as well.

One component can identify another component with its standards interface and communicates through a standard protocol like SOAP. The interfaces as well as SOAP are written in XML code. Interface is written in WSDL which is simply XML code.

4.4 Main Office

This tier consists of two units, one is the set of enterprises, i.e., Enterprises and second one is application of PLA. We discuss each unit separately.

4.4.1 Enterprises

In this unit, there are more than one government offices. Each office maintains its own enter-prise architecture. It has three sections. The middle section is the core of the architecture and other left and right sections provide support for the smooth functioning of the architecture. We explain each section in details.

- Core Part This is the main part of the enterprise architecture. It has three sub components. The first component is business goal. It defines the target or mission of the enterprise.
- If the mission in business goal is changed or up-dated then the business process also needs to be changed. The third component is for the technology specially ICT tools. It is more on selection of proper ICT tools which comprise proper data structure, application software or architecture, hardware infrastructure, network topology etc.
- Activity: It has three main components. All these components work along with business goal, business process and ICT tools. The first compo-nent is "As-Is". The main task of this component is to analyze the present status of the enterprise in terms of business goal, business process and technology. It gives the idea of present scenario of the enterprise like what goal that enterprise is trying to achieve now, what are the business processes that are being followed, and what is the status of ICT tools and how are they being used in current business process. The second component is "To-be". It has the clear picture for tomorrow like what exactly the enterprise would like to be in certain period of time, what the mission of the enterprise is and how do we make or select the business process and what are the technologies that we require for this. This component gives us the guidelines to make a plan. These first two components in this section show us the clear gap between the existing systems and would be system. The third component contains the rules and regulation of the enterprise.
- Non-Functional Properties: This section is for the quality of services which is provided by this architecture and also quality of entire architecture. There are four non-functional properties: availability, survivability, reuse, downtime in this section.
-

4.4.2 Product Line Architecture (PLA)

It is the architecture of product line and product family. A product line is a set of related systems that together addresses a market segment and a product family, is the set of related systems that are built from a common set of core assets [8]. In our case, product line is a set of enterprises whereas a product family is a blackboard repository.

This unit provides the major contribution in this part of the architecture. It takes the functional properties or main services from each of the enterprise then it finds out the commonality and differences in functional properties or services among the enterprises [9]. Many organizations in the government share the same data, same information and provide almost same type of services to the users and also have almost the same objectives but they do it differently. The product line architecture considers each one of them and tries to find out the commonality and variants among the organizations. It exploits the commonalities and manages the differences [10]. The Figure 2 shows the working mechanism of PLA.

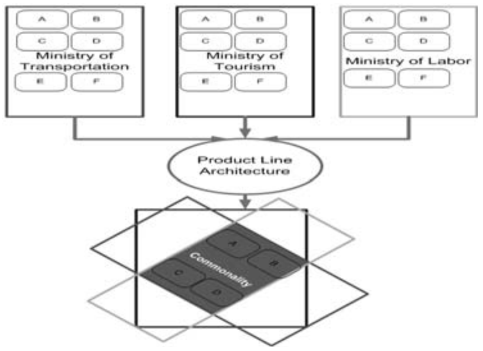

그림 2. 메인아키텍처의 PLA구성
Fig. 2. Equivalent PLA of Main Architecture

The dark part on the bottom of Figure 2 is commonalities and remaining hollow part is variants. In Figure 2, we see each ministry has some set of services i.e. A, B, C, D, E and F. We relate these services to the components of business process in Figure 3. “A” corresponds to first component, “B” corresponds to second component, and “C” corresponds to third component and so on in Figure 3. The information on the types of service is obtained by the knowledge source. In Figure 2, each service is analyzed by product line architecture along with the information obtained through knowledge source and creates the list of commonalities and variants. We find components “A”, “B”, “C” and “D” are common sharable components among the three ministries. If we focus only on the commonality parts, there will be no need to develop these components from scratch. we can achieve good percentage of reuse. These commonalities are stored in Blackboard of back office for the objective of proper reuse of these components. The blackboard stores these components based upon the features. Feature Oriented Product Line Engineering manages the commonalities and differences based upon the features or services [10].

4.4.2.1 Case Study

As we have said that we consider e-government as a case study, we take an example of it. Let us take an example of a service as a case that gives the legal authorization to the citizen. We call such service as a License Service. It provides rights to practice certain discipline. Citizen makes a request to the government for permission of doing some events. We have listed three examples below related to the legal authorization.

- Citizen is in need of permission to drive.
- Citizen is in need of permission to be a tourist guide.
- Citizen is in need of permission to run a factory.

These permissions are provided by the government to its citizen. A single government department normally does not provide all the three permissions. There are three different departments for these three different permissions. In general, Ministry of Transport provides permission to drive, Ministry of Tourism provides the permission to be a tourist guide and Ministry of Industries and Commerce provides permission to run a factory.

We found that each ministry has different criteria for giving permission but the mechanism is almost same and there are many similar steps in the procedure they follow.

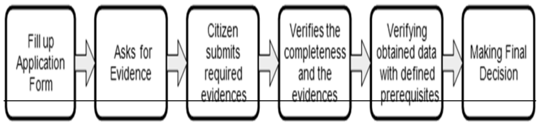
Fill up Application Form
Asks for Evidence
Citizen submits required evidences
Verifies the completeness and the evidences
Verifying obtained data with defined prerequisites
Making Final Decision

그림 3. 업무 흐름
Fig. 3. Business Process

There are six business steps in business process, given in Figure 3. The process flows from left to the right. Like, a citizen has to fill up the application form for all three above mentioned permissions. They are required to provide evidences of their personal information and they submit to their respective ministries or departments within the ministry. Ministry verifies the completeness of application form and other supporting evidences. Each ministry has different criteria for analyzing the obtained data from citizen like; Ministry of Transport may need the health information of the applicant as one of the prerequisites to give driving license whereas Minis-try of Industries and Commerce may not require it to give factory license. Along with these, each ministry has different policy to make the final decision.

In spite of differences among ministries in verifying obtained data and making final decision, but they are common in some aspects like, in filling up application form, asking and collecting evidences, and verifying collected evidences.

4.5 Back Office

It has two main units i.e. blackboard and monitoring unit. Blackboard is based upon the blackboard architecture. It is one of the popular architectures in the area of software architecture. There are two main parts in this architecture: one is repository and another is knowledge source (KS). It is also known as knowledge based agent. It has a knowledge base that contains the sentences which are used to verify the types of services. The sentences are represented with first order logic. The process of inference is used to make a decision on the services. The knowledge source interacts with the repository directly and it may interact with each other through repository. Knowledge source puts its features or services into the repository and shares the features provided by other knowledge sources as per the need. Here, the government organizations of "Main Office" are knowledge source and they put the services to it through product line architecture. As mentioned in Section 4.4.2, the commonalities among the different organizations are kept in the repository. The dotted line between organizations and black board shows the relation between them.

The blackboard takes some time to search service from its repository, to verify the authentication and authorization from the enterprise. Once the service is found and authentication and authorization from enterprise are obtained then the service will be given back to the enterprise. We use some set of primitives to interact between them. Monitoring unit monitors the performance of blackboard and its details are given in the next section.

4.5.1 Blackboard

Blackboard as a whole contains two main parts: one is blackboard itself and another is KS. KS is the government enterprise but its main work is carried out in collaboration with blackboard. We define some set of primitives with signature for the proper interaction between enterprise and blackboard. The primitives like: Discover, Inform, Initiates and Transfer are used for every transaction between enterprise and blackboard.

The blackboard architecture does not only contain the services but also has security mechanism to protect the data or services from unauthorized access. One organization may want to share some set of services to only particular organization and not to every other organization. In this case the blackboard architecture takes care of this with identity based access control mechanism [11]. Access to the service and data is only granted to the requested organization if such an association exits in the blackboard.

4.5.2 Knowledge Source

It has been mentioned in the previous Section that the main work of knowledge source is carried out in collaboration with blackboard. Knowledge source considers each service from the organization and decides on the types of services. Once the types of services are identified then PLA takes the role to construct the feature matrix. Here, we use the concept of expert system in which the decision is made based upon the inference of the sentences in knowledge base. Knowledge base is the collection of sentences that are used to make a decision of the type of service in our architecture. We use domain experts in finalizing the sentences.

4.5.3 Monitoring Unit

We see that the monitoring unit is very close to blackboard in Figure 1. The monitoring unit monitors the blackboards and provides fault tolerance capability to the blackboard. As per the nature of performance and architecture the blackboard suffers from outages due to fragmentation and accumulation of errors which is nothing but software ageing. The software ageing brings high costs, in system with high availability requirement. In order to counteract we use manager in our system to take care of this phenomenon [12]. We use cloud computing here as a pilot concept. The blackboard gives the signal in every 20 seconds to monitoring unit. It means it is alive. If monitoring unit does not get the signal in 20 seconds then it switches to another backup system. We can use the following approach for this as shown in Figure 4.

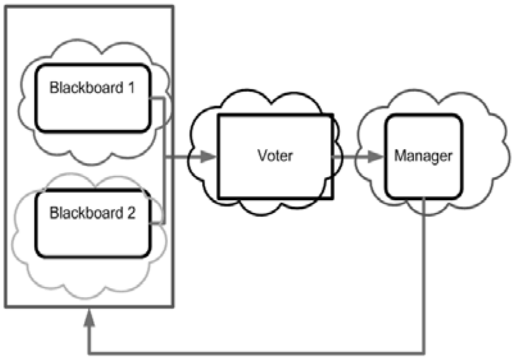

그림 4. 제안된 감시유닛
Fig. 4. Proposed Monitoring Unit

As shown in the Figure 4, the proposed monitoring system has three main entities, i.e. Blackboard, Voter, and Manager. Each entity is in the cloud. The interaction among the entities means interaction among the clouds. In total there are four clouds in Figure 4. We define set of standards API in order to communicate among the components in monitoring system.

4.5.3.1 Analysis

Manager has the vital role in monitoring unit that is why; we carry out the analysis based upon the presence and absence of manager. We assume manager as a reliability component in monitoring system.

- With Manager:

Blackboard consists of stack of services in its repository. It keeps the services from enterprise, provides the services back to enterprise, protects the services from unauthorized entities, and updates the services periodically. Most of the core activities of EA are done by blackboard in collaboration with KS. Performance of EA therefore depends upon the proper functioning of blackboard and necessitates mandatory monitoring of it to ensure smooth functioning of the blackboard. In order to achieve it we propose monitoring unit as a master unit to observe and monitor the blackboard. Here, we analyze blackboard with monitoring unit using Constant Time Markov Chain (CTMC) model depicted by the state diagram shown in Figure 5. We perform the analysis based upon the CTMC. We develop the corresponding state diagram. We use Markov model for the purpose of analysis.

The Figure 5 is the equivalent state diagram of Figure 4. It has twelve states. The transition from one state to another takes place with a probabilistic mean rate.

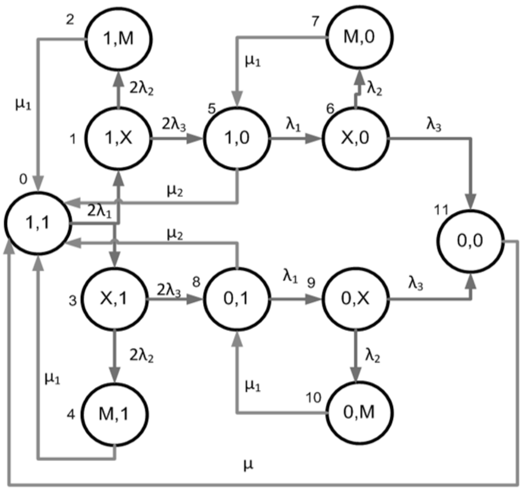

그림 5. 상태천이도 (관리자가 있는)
Fig. 5. State Diagram (with Manager)

1: **Blackboard is operating**

X: **Blackboard is in abnormal condition but working**

M: **Blackboard is getting repaired but not working**

0: **Blackboard is down i.e. not working**

Good State: 1, 1

Defective State: 1,X; X, 1; X,0; 0,X

Partial Failure State: 1,0; 0,1

Maintenance State: 1, M; M, 1; M, 0; 0, M

Failure State: 0, 0

Each state has a certain probability to be in its own state. They are represented with π0, π1, π2, π3, π4, π5, π6, π7, π8, π9, π10, and π11 for respective states. Each state is marked with two numbers (i.e. the status of two blackboards) and the combination of these two numbers indicates whether the system is in running state or not. In Figure 4, there are three main units i.e. black-board, voters and manager. Blackboard contains the stack of sharable services and it provides these services to the enterprise. Voters monitor the blackboard whereas manager monitors both voters and blackboard. It is assumed that voter monitors the blackboard regularly as per predefined duration and manager monitors blackboard in every 10 minutes. Manager is considered to be extremely reliable.

Blackboard is a kind of software component which is designed using a specific process. We consider process aging in this case but a catastrophic failure is ruled out. For this reason base longevity interval for the blackboard is considered 6 months. It means the process in blackboard application can survive up to 6 months. We expect both blackboards run throughout the life time of the system. During the life time of the system if one fails another takes care because of hot standby fault tolerance approach. In Figure 5, there are two possibilities for the transition. Transition is based upon the behavior of the blackboard. If first blackboard behaves abnormal then it follows the states in the right side of the state diagram and if second fails it follows the left. We assume the both blackboard does not fail in the initial phase of the system life time. The transition from a good state to a defective state takes place with a probabilistic mean rate of 2λ1. Once it is in the defective state voter identifies it and try to inform manager with probabilistic mean rate of 2λ2. This occurs in every 30 minutes. Voter can notice the defect but cannot repair it. Unfortunately, if it cannot switch to maintenance state then it goes to a partial failure state with probabilistic mean rate of $2\lambda_3$. Here, one of the blackboards is down but the system is operating because another blackboard is still in good condition. We assume this happens in a week. The first blackboard gets repaired with repair rate of μ_2. At the same time, the second blackboard may also get defect with a rate of λ_1. As earlier, the voter notices this defect also and moves to the maintenance state for getting maintenance from manager with probabilistic mean rate of λ_2. Here also manager does maintain or repair the abnormal blackboard. If the second blackboard does not get repaired by manager on time then it moves to the total failure state with probabilistic mean rate of λ_3. This is the meantime between two main failures (MTBF) in our monitoring system. Finally, if both of the blackboards go down then it is repaired with the rate of μ. The system does not provide the services if it is in state 7, 10 and 11 otherwise the services are available in other remaining states.

표 1. 운영 파라미터
Table 1. Operating Parameters

Parameters	Values
λ_1	6 Months
μ	1 time in a day
μ_2	2 times in a day
λ_2	1 time in 30 minutes
λ_3	1 time in a week
Time	1 year i.e. 8640 hours
μ_1	1 time in 30 mn

Calculations:

Let us calculate the probability of each state. The steady state balanced equations of state diagram are given below.

Good State:

$$\pi_0 = \frac{\pi_2\mu_1 + \pi_5\mu_2 + \pi_{11}\mu + \pi_8\mu_2 + \pi_4\mu_1}{(2\lambda_1 + 2\lambda_1)} \quad \text{(A)}$$

State 1:

$$\pi_1 = \frac{\lambda_1}{\lambda_2 + \lambda_3}\pi_0 \quad (1)$$

State 2:

$$\pi_2 = \frac{2\lambda_1\lambda_2}{\mu_1(\lambda_2+\lambda_3)}\pi_0 \tag{2}$$

State 3:

$$\pi_3 = \frac{\lambda_1}{\lambda_2+\lambda_3}\pi_0 \tag{3}$$

State 4:

$$\pi_4 = \frac{2\lambda_1\lambda_2}{\mu_1(\lambda_2+\lambda_3)}\pi_0 \tag{4}$$

State 5:

$$\pi_5 = \frac{\lambda_1\lambda_3}{(\lambda_2+\lambda_3)(\lambda_1+\mu_2)-\lambda_1\lambda_2}\pi_0 \tag{5}$$

State 6:

$$\pi_6 = \frac{\lambda_1{}^2\lambda_3}{(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)}\pi_0 \tag{6}$$

State 7:

$$\pi_7 = \left[\frac{\lambda_2\lambda_1^2\lambda_3}{\mu_1(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)\mu_1}\right]\pi_0 \tag{7}$$

State 8:

$$\pi_8 = \frac{\lambda_1\lambda_3}{(\lambda_2+\lambda_3)(\lambda_1+\mu_2)-\lambda_1\lambda_2}\pi_0 \tag{8}$$

State 9:

$$\pi_9 = \frac{\lambda_1{}^2\lambda_3}{(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)}\pi_0 \tag{9}$$

State 10:

$$\pi_{10} = \left[\frac{\lambda_2\lambda_1^2\lambda_3}{\mu_1(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)\mu_1}\right]\pi_0 \tag{10}$$

State 11:

$$\pi_{11} = \left[\frac{2\lambda_1^2\lambda_3{}^2}{\mu(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)\mu}\right]\pi_0 \tag{11}$$

Now, with following balanced equation, the steady state probability of vector π of the Markov Chain is given by:

$$\textstyle\sum_{i=1}^{P} \pi_i = 1$$

Where,

P is the total number of states of a system.

The equation for our state diagram can be obtained by summing up the probabilities of all five states and the sum is equal to 1.

$$\sum_{i=1}^{n}\pi_0+\sum_{i=1}^{n}\pi_1+\sum_{i=1}^{n}\pi_2+\sum_{i=1}^{n}\pi_3+\sum_{i=1}^{n}\pi_4+\sum_{i=1}^{n}\pi_5+\sum_{i=1}^{n}\pi_6+\sum_{i=1}^{n}\pi_7+\sum_{i=1}^{n}\pi_8+\sum_{i=1}^{n}\pi_9+\sum_{i=1}^{n}\pi_{10}+\sum_{i=1}^{n}\pi_{11}=1 \tag{12}$$

$$\pi_0+\pi_1+\pi_2+\pi_3+\pi_4+\pi_5+\pi_6+\pi_7+\pi_8+\pi_9+\pi_{10}+\pi_{11}=1 \tag{13}$$

Equating equation (1-11) into equation 12, we get,

$$\pi_0 = \left[\begin{array}{l} 1+\frac{\lambda_1}{\lambda_2+\lambda_3}+\frac{2\lambda_1\lambda_2}{\mu_1(\lambda_2+\lambda_3)}+\frac{\lambda_1}{\lambda_2+\lambda_3}+\frac{2\lambda_1\lambda_2}{\mu_1(\lambda_2+\lambda_3)}+\frac{\lambda_1\lambda_3}{(\lambda_2+\lambda_3)(\lambda_1+\mu_2)-\lambda_1\lambda_2}+\frac{\lambda_1^2\lambda_3}{(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)}+ \\ \frac{\lambda_1^2\lambda_3\lambda_2}{(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)\mu_1-\lambda_1\lambda_2(\lambda_2+\lambda_3)\mu_1}+\frac{\lambda_1\lambda_3}{(\lambda_2+\lambda_3)(\lambda_1+\mu_2)-\lambda_1\lambda_2}+\frac{\lambda_1^2\lambda_3}{(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)}+ \\ \frac{\lambda_1^2\lambda_3\lambda_2}{\mu_1(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)\mu_1}+\frac{2\lambda_1^2\lambda_3^2}{\mu(\lambda_2+\lambda_3)^2(\lambda_1+\mu_2)-\lambda_1\lambda_2(\lambda_2+\lambda_3)\mu} \end{array}\right]^{-1} \tag{14}$$

This is the equation of probability of our proposed monitoring system to be in "Good State" After keeping the parameters from Table 1, we get following features of monitoring system.

Availability: It is the likelihood that the system is delivering adequate service to its users. We consider EA and try to calculate the likelihood of service delivery. Since the blackboard is the core of our architecture and monitoring unit is keep on monitoring and supporting the blackboard in case of any type of failures, so non-functional properties of monitoring unit are very important. We calculate the availability, survivability and down time of monitoring unit with an hour repair rate.

If we divide the state space into the set of UP states and set of Down states, then the basic availability can be obtained with following equation,

$$A = \sum_{i \in UP} \pi_i \qquad (15)$$

Hence,

Availability = 1- unavailability

System does not provide service when it is in defect state, voter state and failure state. Only it provides when it is in manager state. So, the following expression calculates the availability of this system

$$\text{Availability} = 1 - (\pi_7 + \pi_{10} + \pi_{11}) \qquad (16)$$

Survivability: It is the likelihood that the system is in operating state in spite of any type of threat, attack or disaster. In our architecture, we consider the capability of monitoring unit to survive, which is given by:

Survivability = Availability-Probability of system to be in failure state.

Survivability = Availability- (Service Unavailable) (17)

Downtime: It is the period in which the system stops providing the service. We calculate the downtime with following equation:

$$\text{Downtime} = (\pi_7 + \pi_{10} + \pi_{11}) * L \qquad (18)$$

Where,

L = Period of 1 year i.e. (12*30*24) = 8640 hours

We calculate the availability, survivability and downtime as the function of repair rate i.e. rate of transition from voter to manager. Table 2 shows the results of them. As it is said that manager is most reliable and can repair the defects, we consider this repair rate as a parameter for achieving the availability. Hence, we carry out the repair rate from 1 time in a day till 24 times in a day i.e. Repair rate in every hour.

표 2. 얻어진 값
Table 2. Obtained Values

S.N.	Manager Rate ($\lambda_{2\text{ and }}\lambda_5$)	Availability	Downtime	Survivability
1	1 time in a day	0.999312853	0.01063	0.99851
2	2 times in a day	0.999631829	0.00372	0.99877
3	4 times in a day	0.999809006	0.00136	0.99891
4	12 times in a day	0.999934745	0.00033	0.99902
5	24 times in a day	0.999967164	0.00014	0.99904
6	48 times in a day	0.999993529	0.00007	0.99907

The above Figure 6 shows the availability verses the times of maintenance in a day. If we perform maintenance in a day then availability would be three 9′s. Once the rate of maintenance is increases the availability also gets increased. We can achieve up to five 9′s.

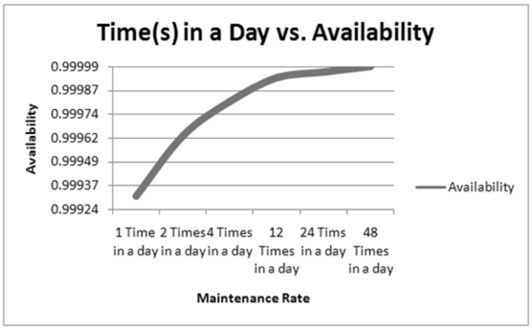

그림 6. 관리비율 대 가용성
Fig. 6. Maintenance Rate v.s. Availability

The Figure 7 shows the survivability verses times in a day. Survivability is increasing as we increase the rate of maintenance from in every 24 hours to every 30 minutes. It can be achieved

up to the three 9′s. It also shows that survivability can be achieved further more if we increase the rate of maintenance.

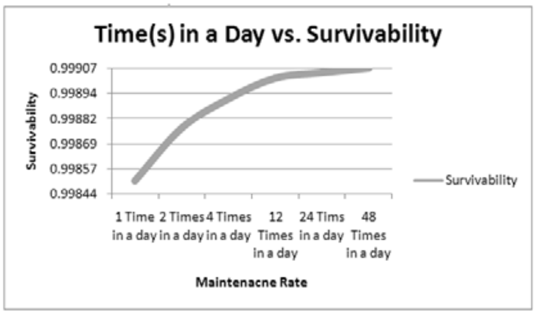

그림 7. 관리비율 대 생존성
Fig. 7. Maintenance Rate v.s. Survivability

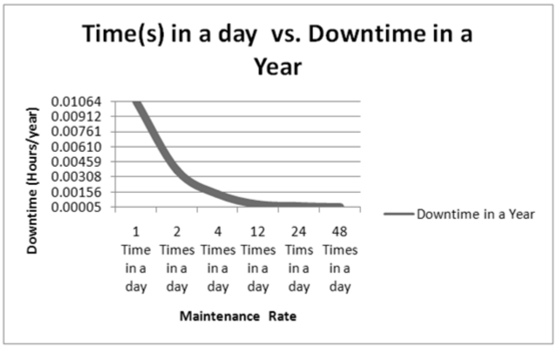

그림 8. 관리비율 대 고장시간
Fig. 8. Maintenance Rate v.s. Downtime

The Figure 8 shows the decreasing trend of downtime of a system. Here, we also compute the downtime against the maintenance rate. It clearly shows the system does not go down for a long time.

- Without Manager:

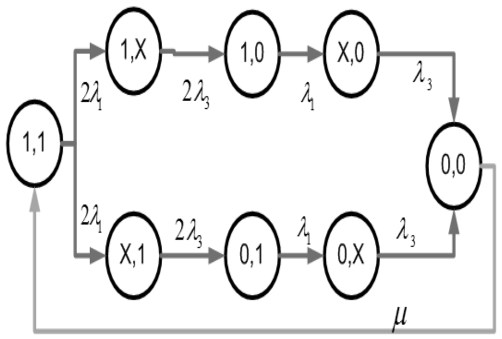

그림 9. 상태천이도 (관리자 없는)
Fig. 9. State Diagram (without Manager)

The Figure 9 shows the state diagram without manager. Here also, we use Markov Model and identify the state probability of each state and the probability of the system to be in good state is given below:

$$\pi_0 = \left[1 + \frac{2\lambda_1}{\lambda_3} + \frac{4\lambda_3}{\lambda_3} + \frac{4\lambda_1\lambda_3}{\lambda_3^2} + \frac{4\lambda_1\lambda_3^2}{\mu.\lambda_3^2} \right]^{-1} \tag{19}$$

We calculate the availability, survivability, and downtime of Figure 9 and compare it with Figure 5 with the help of following graph.

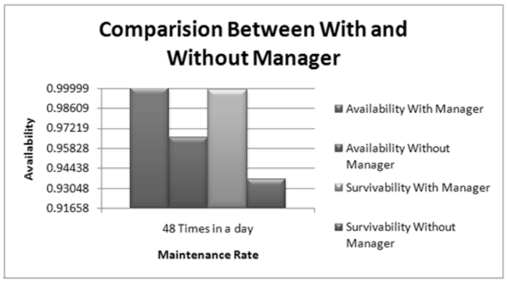

그림 10. 관리자 유무에 따른 가용성, 생존성 비교
Fig. 10. Comparison Availability, Survivability between with and without manager

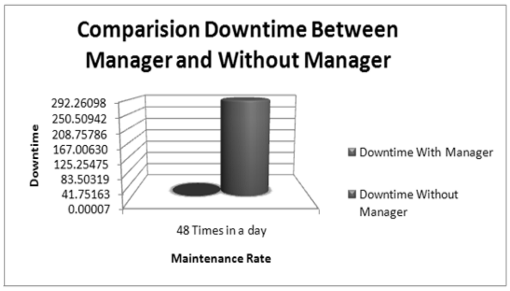

그림 11. 관리자 유무에 따른 고장시간 비교
Fig. 11. Comparison Downtime between with and without manager

The Figure 10 and Figure 11 are the graphs of comparison between with and without manager. It shows clearly the high availability, survivability and less downtime with manager as compare to without manager.

V . Features of Proposed Architecture

There are many features in the proposed architecture but we have listed only few of them:

- High Adaptability: The main challenge of EA is to enhance the degree of adaptability. We have calculated non-functional properties in our architecture and found very satisfying results. As per the analysis, this architecture can achieve availability of 99.957%, survivability of 99.910% and it goes down for four hours in a year which is the significant time for maintenance and repairing of the system.

- High Reusability: It is possible to get maximum reusability in this architecture because each service is analyzed by blackboard, knowledge source with the approach of expert system and product line architecture and tried to get maximum reuse.

- High Maintainability: SOA is used in component interaction. Each component interacts with other component with SOAP and WSDL thereby creating a loosely coupled architecture. A component can be updated, or replaced without disturbing other components or architecture. This feature ensures high maintainability.

- Scalability and Resource Management: It is the capability of architecture to expand its resources as per the need and reduce its when there is no need. We have obtained the feature of scalability in monitoring unit of architecture. That has been possible with cloud computing.

Ⅵ. Conclusion

The proposed architecture gives a new dimension to the existing enterprise architecture and is built upon the basic philosophical premise of EA consisting of components such as SOA, PLA, along with cloud computing. Blackboard architecture is proposed for managing sharable services. There are few components with some specific objectives and they can interact with each other through SOAP. The features like separation of concern, reusability, interoperability, scalability and cost effectiveness can be achieved in this architecture. Maximum reusability is achieved through PLA. We identified poor adaptability as the major challenge in existing EA and a solution to address this challenge has been proposed that makes use of available technologies. Now, there is no need to re-structure entire architecture with small change in one component. The proposed architecture also addresses the critical challenges of EA and enhances the non-functional properties.

References

[1] K. Trivedi, "SHARPE 2002: Symbolic Hie-rarchical Automated Reliability and Performance Evaluator", *2002*

[2] P. Narman, M. Schonherr, P. Johnson, M. Ekstedt, and M. Chenine, "Using Enterprise Architecture Models for System Quality Attributes",*Enterprise Distributed Object Computing Conference, IEEE 2008.*

[3] J. Zachman, "A Framework for Information System Architecture", *IBM System Journals 1987.*

[4] A Practical Guide to Federal Enterprise Architecture, *Version 1, Feb 2001*

[5] G. Engles, and M. Assmann, "Service Oriented Enterprise Architecture: Evolution of Concepts and Methods",*Department of Computer Science, University of Paderborn.*

[6] T. Thandar, and J. Park, "Availability Analysis of Application Servers Using Software Rejuvenation and Virtualization", *Journal of Computer Science and Technology March 2009, PP 339-346.*

[7] S. Gay, "An Examination of Virtualization's Role in Contingency Planning", *Information Security, Curriculum Development conference 07', ACM September 28-29, 2007*

[8] P. Clements, "Report of the Reuse and Product Lines" *Working Group of WISR8, CMU/SEI-97-SR-010 August 1997*

[9] K. Wang, G. Cai, Z. Li, and L. Zhou, "A Disaster Recovery System Model in an E-government System", *International Conference on Parallel and Distributed Computing, Application and Technology, [PDCAT 05] IEEE 2005*

[10] K. Kang, J. Lee, and P. Donohoe, "Feature Oriented Product Line Engineering", *IEEE Software, July/August 2002*

[11] Yuan, and J, Tong, "Attributed Based Access Control for Web Services", *IEEE, Conference on Web Services(ICWS' 05)*

[12] K. Trivedi, K. Vaidhyanathan, and K. Goseva-Popstojanova. "Modeling and Analysis of Software Aging and Rejuvenation", *33rd Annual Simulation Symposium, April 2000, PP. 270-279*

[13] J. Bosch, *Design & Use of Software Architecture*, Addison Wesley, 2000

[14] http://www.businessofgovernment.org/pdf/WyldC oudReportGov.pdf

[15] J. Laprie, "Dependable Computing and Fault Tolerance: Concepts and Terminology" *The Fifteenth International Symposium on Fault Tolerance Computing, 1985 PP 1-11.*

[16] S. Kaisler, F. Armour, and M. Valivullah, "Enterprise Architecture: Critical Problems", *The 38th Hawaii International Conference on System Sciences-2005*

저자소개

Manish Pokharel

In 1995, Bachelor of Engineering in Computer Science, Karnataka University
In 2000, Master of Engineering in Software System, Birla Institue of Technology and Science
2007 ~ present Ph.D. student in Korea Aerospace University
※ Interest : E-government, Enterprise Architecture, Fault Tolerance and Cloud Computing

최승준(Seung Jun Choi)

Bachelor of Electronic Engineering, Korean Air Force Academy
Master of Electronic Engineering, Yonsei University
2010 ~ present Brigadier General of the Republic of Korea Air Force
※ Interest : Ubiquitous Sensor Network(MAC Protocol), C4I(Network and Application) System and Analysis and IEEE802.16e(Wibro, WiMAX)

이봉재(Bong Jae Lee)

In 1990, Bachelor of Communication Engineering, Korea Aerospace University
In 1993, Master of Information and Communication Engineering, Korea Aerospace University
2003 ~ present Ph.D. student in Korea Aerospace University
※ Interest : u-City, E-government, Information security.

박종서(Jong Sou Park)

In 1986, Master of Electrical and Computer Engineering, North Carolina State University
In 1994, Ph.D of Computer Engineering, Pennsylvania State University
1996 ~ present Professor in Korea Aerospace University
※ Interset : information security, embedded system and hardware design.