공학박사 학위논문

스마트홈 컨텍스트에서 에니어그램을 이용한 서비스지향 아키텍처의 설계 및 평가

Design and Evaluation of Service Oriented Architecture using Enneagram in Smart Home Context

지도교수 이 상 배

2007년 2월

한국해양대학교 대학원

전자통신공학과 최성욱

本 論文을 崔盛旭의 工學博士 學位論文으로
認准함.

위원장 : 임 재 홍 (인)

위 원 : 심 준 환 (인)

위 원 : 백 승 면 (인)

위 원 : 손 홍 근 (인)

위 원 : 이 상 배 (인)

2006년 12월 19일

한국해양대학교 대학원

목 차

키 워 드

SOA, Enneagram, Smart Home, OSGi, ATAM, 품질속성, Fuzzy

표 목 차

그림 목차

Abbreviations

AA : Application Architecture
ABAS : Attribute-Based Architectural Styles
API : Application Programming Interface
ATAM : Architecture Tradeoff Analysis Method
A/V : Audio/Video
BAM : Business Activity Monitoring
BCE : Boundary-Control-Entity
BPA : Basic Probability Assignment
BPM : Business Process Management
CBAM : Cost Benefit Analysis Method
COTS : Commercial Off-The-Shelf
CSF : Critical Success Factor
DA : Data Architecture
DMS : Data Mediator Service
DLNA : Digital Living Network Alliance
EA : Enterprise Architecture
EAP : Enterprise Architecture Planning
EIS : Enterprise Information Systems
EMF : Eclipse Modeling Framework
ESB : Enterprise Service Bus
eRCP : embedded Rich Client Platform
eSOA : embedded Service Oriented Architecture
EDA : Event Driven Architecture
EMR : Electric Heath Record
EVMS : Earned Value Management System

GENA : General Event Notification Architecture
HANA : High Definition Audio Video Network Alliance
HAVi : Home Audio Video interoperability
HL7 : Health Level 7
IHE : Integating the Health Enterprise
IP-VPN : Internet Protocol - Virtual Private Network
iQA : included Quality Attribute
KPI : Key Performance Index
LAAAM : Lightweight Architecture Alternative Assessment Method
LAMP : Linux, Apache, My-SQL, Php
LnCP : Living network Control Protocol
MECE : Mutually Exclusive, Collectivery Exhaustive
MOT : Moment Of Truth
MVC : Model-View-Controller
MVP : Model-View-Presenter
NAT : Network Address Translation
NHII : National Health Information Infrastructure
OHF : Open Helathcare Framework
OEOR : One Event One Response
ORM : Object-Relational Mapping
OSMU : One Source Multi Use
OSGi : Open Service Gateway intiative
QA : Quality Attribute
QAR : Quality Attribute Refinement
PAC : Presentation-Abstraction-Control
PLC : Power Line Communication
PNA : Phoneline Networking Alliance
POC : Proof Of Concept
RFM : Recency, Frequency, Monetory
RCP : Rich Client Platform
RNS : Role-Norm-Sanction
RRC : Resource-Role-Context
RTE : Real Time Enterprise

SaaS : Software as a Service
SAF : SPAMMED Architecture Framework
SBC : Server Based Computing
SCA : Service Component Architecture
SDO : Service Data Object
SLC : Service Life Cycle
SOA : Service Oriented Architecture
SOAP : Simple Object Access Protocol
SOP : Service Oriented Programming
SPAMMED : Stakeholders, Principles, Attributes, Model, Map, Evaluate, Deploy
SPC : Service Profit Chain
SSDP : Simple Service Discovery Protocol
STD : State Transition Diagram
STP : Service oriented architecture Tool Platform
TA : Technical Architecture
TDD : Test-Driven Development
UDDI : Universal Description Discovery and Integration
UML : Unified Modeling Language
UI : User Interface
UPnP : Universal Plug and Play
URL : Uniform Resource Locator
VM : Virtual Machine
WOPA : Write Once Publish Anywhere
WORA : Write Once Run Anywhere
WSDL : Web Service Description Language
XML : eXtensible Markup Language

Abstract

In the smart home environment designed for many users, not only a single context, but also multi-context simultaneously exists, which continues to generate and perish with the passage of time. Therefore, this paper has designed a Service Oriented Architecture based MVC and composed a smart home scenario on the basis of the Enneagram principle, which can aware of conflict resolution between the RRC(Resource, Role, Context) and also which is able to respond effectively according to the scope of domain.

The Enneagram, which provides a theoretical idea for designing the Service Oriented Architecture demanded under a smart home context, is able to provide an excellent prototype that well describes the relationship between the entirety and rotational process of an event. It is described in the Monad(0), Triad(9-3-6), and Heptad(7-1-4-2-8-5-7). These make contribution to the whole event, and Enneagram organizes these principles into a diagram. The Service Oriented Architecture suggested in this paper has been designed by taking into consideration the UPnP, OSGi, and SOA Tool Platform for eclipse platform. A service provider manages related kinds of services on an integrative basis from various sensors, puts each service in a WSDL/SOAP message, and registers them to the UDDI server of service mediator. Under the environment of an OSGi service platform, various context-aware services are dynamically being mapped from various sensors; new services are being offered for the asking of users; existing services are changing; and bundled services are available.

By applying the modified ATAM, which is the service evaluation methodology for scenario-based architecture, this paper has made a utility tree from a business objective, designed the scenario for system quality, and thus providing a systematic approach to quality element. Therefore, if transaction throughput is 546 TPS and system availability is 98% and resuability COTS is 35%, then service requester satisfaction is 85.1% in maximum value set, so and if data latency is 2000 ms and system reliability is 1 Min and modify COTS is 1 Man/Day , then service requester satisfaction is 70.1% in minimun value set from ATAM while remaining stable other condition. Also this paper provides a useful idea for plug-in architecture modeling and for an eclipse-based OHP, which has adopted the HL7 and IHE standard by domain size or by stakeholder in the U-Healthcare field.

제 1 장 서 론

1.1 연구배경

유비쿼터스 컴퓨팅의 발달은 홈네트워크의 발전과 밀접한 영향을 가지고 있으며 스마트홈 환경은 기존의 수동적 공간에서 능동적 공간으로 변화됨에 따라 환경이 서비스 대상이 아닌 서비스 제공자 또는 서비스 중개자가 될 것이고 여기에 필요한 응용 및 서비스는 컴퓨팅 및 커뮤니케이션 능력을 가진 스마트 객체들이 동적인 컨텍스트 변화를 인식하고 가상화하여 이에 적응할 수 있는 특성, 즉 상황인식 특성을 갖게 될 것이다[2,11,17].

스마트홈은 “생활 환경의 지능화, 환경 친화적 주거생활” 거주공간으로 정의되며 다양한 센서와 네트워크 기술을 바탕으로 거주자 및 주거환경에 대한 컨텍스트 정보를 수집하여 서비스 사용자 에게 차별화된 개인화 서비스를 제공한다. 홈네트워크 환경에서는 홈오토메이션 관점에서 장치간 상호작용에 맞추어져 있지만 스마트홈 환경에서는 서비스 사용자의 의도에 맞는 상황인식이 중요해지기 때문이다. 특히 다수의 서비스 사용자가 생활하는 스마트홈 환경에서는 컨텍스트가 하나만 존재하는 것이 아니라 시간의 흐름에 따라 생성되고 소멸되는 다중 컨텍스트가 동시에 존재함에 따라 컨텍스트, 자원, 역할의 충돌을 감지하여 도메인의 범위에 따라 이를 효율적으로 가상화하여 관리해줄 수 있는 서비스지향 아키텍쳐가 요구된다[2,3,17].

스마트홈 컨텍스트에서 요구하는 서비스지향 아키텍쳐를 설계하기 위하여 사상적인 배경이 되는 에니어그램은 어떤 전체사건의 조직을 그것의 본질적인 원리인 전체성과 순환적 과정과 연관된 훌륭한 원형을 제시해주며 그것은 수 1, 3, 7로 표현되는 모나드, 트리아드, 헵타드로서 이들은 각각 어떤 전체적인 사건에 기여하며 에니어그램은 이들의 원리를 조직하여 하나의 다이아그램으로 만든다[8,12,18,30].

스마트홈 컨텍스트에서 요구하는 MVC(Model-View-Controller) 아키텍쳐에 충실한 서비스지향 아키텍쳐의 서비스 발견 및 전달, 서비스 조립, 서비스 가상화를 우선적으로 고려하기 위하여 홈네트워크 서비스 미들웨어와 개방형 서비스 게이트웨이, 통합형 서비스 플러그인, 웹서비스의 표준화가 요구되는데, 외부에서 서비스를 제공하는 사람의 입장에서는 각 가정에 있는 홈네트워크가 달

라짐에 따라 다른 형태의 서비스를 제공해 주어야 하며 외부 네트워크가 n개의 표준이 있고 홈네트워크가 m개의 표준이 존재하면 서비스를 제공해주는 방법은 $n \times m$ 개가 존재하게 된다. 하지만 서비스 미들웨어, 서비스 게이트웨이, 서비스 플러그인, 웹서비스를 표준화시킨다면 그 복잡도는 $n+m$으로 떨어지게 되며 서비스발견을 위해 SSDP(Simple Service Discovery Protocol), SOAP(Simple Object Access Protocol) 같은 인터넷기반 통신프로토콜을 정의한 미들웨어인 UPnP(Universal Plug and Play) 서비스전달을 위한 실행환경을 정의한 OSGi(Open Service Gateway intiative), 서비스 플러그인과 툴-통합 플랫폼인 이클립스 플랫폼, 서비스 가상화를 위한 ESB(Enterprise Service Bus)를 이용한 웹서비스가 사실상 표준이 되고 있다[1,7,10,11,39].

소프트웨어 아키텍쳐의 라이프사이클에서 후반부에 발생하는 요구사항의 오류를 수정하는 비용은 개발 초기단계인 요구공학 단계에서 오류를 검출하고 수정하는 비용의 200배가 소요되며 인식된 소프트웨어 문제점 중 약 50%이하가 요구사항 명세에 있었고, 약 50%정도가 요구사항관리에 있었다. 따라서 본 논문에서는 서비스 사용자의 요구사항을 분석하고 명세화하는 시나리오기반의 아키텍쳐 분석 요구공학 평가방법론의 하나인 ATAM(Architecture Tradeoff Analysis Method)을 이용하였다. ATAM의 스텝 7에서 해당 문헌조사와 스테이크홀더간의 충분한 협의를 하였고 CBAM(Cost Benefit Analysis Method)의 품질속성의 반응값을 선택한 후 이의 최대가치의 반응값 집합과 최소가치의 반응값 집합을 분리하여 MATLAB 퍼지시뮬레이션을 적용하였다. 이러한 품질속성간의 조합으로 비즈니스 목표로 부터 유틸리티 트리를 만들고 시스템 품질을 위한 시나리오를 설계함으로써 중요한 품질속성을 찾는 체계적인 접근법으로 제안된 서비스지향 아키텍쳐에 대한 가상의 서비스 사용자의 만족도를 조사하여 평가하였다. 이는 서비스 사용자가 MOT(Moment Of Truth)의 서비스 접점 순간에는 가치지향적이므로 서비스에 대해 지불된 가격과 그 서비스를 얻기 위한 획득비용보다 더 큰 서비스 결과에 대한 가치와 서비스 전달과정에서의 품질을 원하고 있으며 SPC(Service Profit Chain)의 이익증가는 서비스 사용자의 충성도로부터 파생되고 서비스 사용자의 충성도는 서비스 가치에 대한 서비스 사용자의 만족도로 평가되기 때문이다[9,14,19.20,35].

1.2 연구 목적 및 내용

본 연구의 목적은 스마트홈 다중 컨텍스트에서 컨텍스트, 자원, 역할의 충돌을 감지하여 도메인의 범위에 따라 번들화된 서비스를 가상화하여 효율적으로 관리하기 위해 에니어그램 사상을 이용한 서비스지향 아키텍쳐를 설계하는데 있다. 따라서 이의 설계 및 검증하기 위하여 다음과 같이 서비스 사용자에 대해 제안된 6가지 품질속성별 최적화된 조합이 필요하다.

연구문제 1. 에니어그램을 이용한 서비스지향 아키텍쳐는 도메인의 범위에 따라 시스템, 컴포넌트 레벨에서 다양한 성능(Performance) 또는 관리효율성(Managability) 즉, 시스템레벨에서 다수의 동시 서비스 사용자와 자동화 및 실시간처리 등에 대한 **신속하고 관리효율적인** 품질속성의 정책관리가 서비스 중개자 관점에서 서비스 사용자와 서비스 제공자의 요구에 따라 가능한가?

연구문제 2. 에니어그램을 이용한 서비스지향 아키텍쳐는 도메인의 범위에 따라 시스템레벨에서 가용성(Availability), 보안성(Security) 등 **정확하고 안전한** 품질속성의 정책관리가 서비스 제공자 관점에서 서비스 사용자의 요구에 따라 가능한가?

연구문제 3. 에니어그램을 이용한 서비스지향 아키텍쳐는 도메인의 범위에 따라 시스템레벨에서 유연성(Agility), 사용성(Usability) 등 **유연하고 편리하게 통합되는** 품질속성의 정책관리가 서비스 사용자관점에서 서비스 제공자 및 외부의 변화요구에 따라 민첩한 대응이 가능한가?

본 연구를 위한 가설의 구체적인 내용은 다음과 같다.

가설. 에니어그램을 이용한 서비스지향 아키텍쳐는 1,000세대 이하 소규모 스마트홈(서비스 사용자, 아파트관리사무소 등) 컨텍스트에서 3가지 이상 품질속성의 상충요소를 분리하고 적절하게 조합할 수 있으면 서비스 사용자는 만족할 것이다.

본 논문에서는 상기 가설을 검증하기 위하여 품질속성중에서 성능, 가용성, 유연성을 살펴보겠다. 여기서 유연성측면에서 서비스 사용자는 상황변화에 따른 민첩한 대응성을 높이는 가운데 충분히 선택할 수 있는 자유를 서비스 받기 원하기 때문에 가용성측면에서 서비스 공급자는 보증할 수 있는 약속된 서비스

를 제공하고자 불확실성을 극복하면서 최대한 필요한 것을 백업, 보안, 준비하며 스마트홈 컨텍스의 객체간의 충돌할 수 있는 의도를 사전에 감지하여 파악함으로써 전체 최적화를 향상시켜 복잡도와 성능을 개선하고자 한다.

본 연구의 범위는 다음과 같다.

첫째, 본 연구는 에니어그램을 이용한 서비스지향 아키텍쳐의 설계가 목적이므로 도메인범위별 실제 구현에 따라 영향을 미칠 수 있는 다른 내적 요인과 환경적인 요인들은 적절히 고려하지 못하였으며 해당 서비스지향 아키텍쳐 설계에 대한 시스템레벨의 실증적인 검증은 소규모 스마트홈에서 z256 PLC(Power Line Communication) 프로토콜을 이용한 외출모드에서 수정된 ATAM으로 검증하였다. 부록의 데모에서 홈서버는 고정 IP(Internet Protocol)이며, 백업, 인증 등 보안품질속성을 고려하지 않았으며 서비스 사용자의 유연성 품질속성을 중심으로 웹서비스 프로토타입 데모를 실시하였다.

둘째, 본 연구의 에니어그램을 이용한 서비스지향 아키텍쳐 설계 및 수정된 ATAM 검증단계에서 유틸리티 트리 전문가 참여규모 2 ~3명과 시나리오 브레인스토밍 전문가 참여규모를 5 ~10명으로 추천하고 있으나 4개월간 9회에 걸쳐 모두 5명으로 선정함에 따라 스테이크홀더 선정인원의 제한점이 있었다.

셋째. 본 연구는 한정된 스테이크홀더 수로 인하여 에니어그램을 이용한 서비스지향 아키텍쳐 설계를 도메인 범위별 다중 컨텍스트에서 일반화하는데 한계성을 가지고 있었다.

본 연구의 기대효과는 다음과 같다.

상기의 제약조건에도 불구하고 제안된 에니어그램을 이용한 서비스지향 아키텍쳐는 도메인범위별 서비스 사용자에 대해 스마트홈을 통한 웹서비스와 U헬스분야에서 아키텍쳐모델링을 통한 비즈니스모델링을 하는데 흥미있는 사상을 제공하여 준다.

먼저 웹서비스를 통한 서비스 가상화를 통하여 관점에 따라 서비스지향 원칙인 “관심의 분리”가 가능하며, 웹서비스 URL(Uniform Resource Locator) 객체에 대한 서비스 생명주기에 따라 서비스 제어권을 관리할 수 있었으며, 가상의 서비스 사용자의 만족도를 도출하는 과정에서 포함품질속성을 추출하여 스마트홈에서 요구하는 표준서비스 시나리오로부터 제어가전을 위한 조명관리, 가전제어, 가스밸브관리의 프로토타입 데모를 제시하였으며, 이를 근거로 설계된 서

비스지향 아키텍쳐는 정보가전, A/V(Audio/Video)가전의 디지털 컨버전스에서 컨텍스트 충돌이 발생하였을 때에도 효과적으로 대응할 수 있다.

현재 IT 인프라 및 유비쿼터스 관련 기술의 빠른 보급 환경, 아파트 중심의 거주 환경, 고령화 시대를 맞는 우리나라에서는 U헬스를 통한 서비스 사용자 삶의 질 향상 욕구와 서비스 제공자의 새로운 사업의 이해관계가 있다. 실제 서비스 사용자의 요구에 부응하는 U헬스를 위하여 면밀한 서비스 사용자 요구 분석과 의료 서비스 구성원의 이해 관계 매듭을 풀지 않고서는 성공할 수 없는데 U헬스를 통한 서비스 사용자(환자)의 이익 대비 진료 서비스 공급자인 의료인의 이익이 선결되어야 하며, 의료인도 U헬스의 기대 효과를 인지하고 스스로 준비해야 한다. 따라서 스마트홈 다중 컨텍스트관리를 통하여 U헬스분야에서 HL7(Health Level 7)과 IHE(Integating the Health Enterprise) 분석을 한 후 이클립스기반의 OHF(Open Helathcare Framework)를 제안된 에니어그램을 이용한 서비스지향 아키텍쳐에 플러그인함으로써 새로운 사업 모형 및 진료 모형을 제시할 수 있다. [1,27,42,43]

본 논문의 구성과 전개순서는 <표 1-1>과 같이 비고의 주요 그림을 중심으로 9단계로 분류하였으며 1장의 서론에 이어, 2장에서는 관련연구로서 ① 비트루비우스 삼각형으로부터 에니어그램과 퍼지이론을 먼저 살펴보고 전사적 구조와 ② MVC를 찾아서 스마트홈의 특성으로부터 ③ RRC(Resource-Role-Context)의 컨텍스트, 자원, 역할을 파악하여 해당 ④ SLC(Service Life Cycle)에 따라 서비스 발견 및 전달, 서비스 전달, 서비스 가상화를 정의하며 ⑤ SOA(Service Oriented Architecture)를 정리하였다. 이를 근거로 3장에서는 ⑥ 에니어그램을 이용한 서비스지향 아키텍쳐와 컨텍스트를 분석하여 ⑦ 5W1H(Who, Why, What, When, Where, How)에 따라 임베디드와 엔터프라이즈환경에서 범위별 서비스지향 아키텍쳐를 설계한 후 ⑧ 주방모드와 외출모드의 시나리오를 각각 설계하였다. 이를 4장에서는 ATAM으로 ⑨ 품질속성에 따라 MATLAB 퍼지시뮬레이션으로 가상의 서비스 사용자의 만족도를 평가하였다. 본 논문은 시스템 개발논문이 아니라 아키텍쳐 설계 논문인 관계로 설계된 아키텍쳐의 프로토타입 데모와 상세한 스마트홈의 개요 및 ATAM 평가방법론의 개요는 부록에서 다루었다.

<표 1-1> 주요 속성별 에니어그램 모형에 따른 논문전개순서

<Table 1-1> Paper Process of Enneagram model by attribute

Triad / Process	9	3	6	비고
①에니어그램, 비트루비우스 삼각형	조정자, 설계자 (아름다움)	성취자, 작업자 (짜임새)	충성가, 사용자 (쓸모)	<그림 2-1>, <그림 2-2>
② *MVC*	***Controller***	***Model***	***View***	**<그림 *2-8*>**
③ RRC	컨텍스트 (시간, 장소)	자원 (돈)	역할 (사람)	<그림 2-9>
④ SLC	서비스 발견 및 전달	서비스 조립	서비스 가상화	<그림 2-10>
⑤ *SOA*	***Service Registry***	***Service***	***Application Front-end***	**<그림 *2-19*>** **<그림 *2-20*>**
⑥ SOA Context	Context Conflict Manager	Context Interpreter	Service Manager	<그림 3-1>
⑦ *5W1H*	***When,Where, How***	***What***	***Who, Why***	**<그림 *3-7*>** **<그림 *3-9*>** **<그림 *3-11*>**
⑧-1.주방모드 시나리오	주방	음식재료	먹는사람	<그림 3-16>
⑧-2.외출모드 시나리오	**집**	**제어가전** 정보가전,*A/V*가전	**집주인**	**<그림 *3-17*>**
⑨ 품질속성	**성능 (복잡성)**	**가용성 (불확실성)**	**유연성 (변화대응성)**	**<그림 *4-4*>** **<그림 *4-7*>** **<그림 *4-10*>**

* 굵은 표시는 핵심전개단계임.

제 2 장 관련연구

2.1 에니어그램과 퍼지정보이론

2.1.1 개요

에니어그램(Enneagram)이란 희랍의 '에니어(ennear, 9, 아홉)'라는 단어와 '그라모스(grammos, 도형·선·점)'라는 단어의 합성어로써 인간의 기본적인 9가지 성격, 유형, 성향에 대한 이론이다. 즉 에니어그램은 그리스어로 '아홉 개의 점이 있는 그림'이라는 뜻이다. 원과 아홉 개의 점, 그리고 그 점들을 잇는 선으로만 구성된 단순한 도형이지만 숨어 있는 내부의 연결을 통해 전체 사건들을 외부의 순서인 수 1, 3, 7로 표현되는 모나드, 트리아드, 헵타드로 바라본다. 트리아드는 전체적인 구조와 균형을 제공하지만, 과정의 세부적인 것은 제공하지 않는다. 헵타드는 과정을 안내하지만, 균현잡힌 양극성의 원리는 결여되어 있다. 에니어그램은 이 둘을 합쳐 아홉개의 수 모두를 통합하여 모나드가 되며 에니어그램의 원자체는 열번째수인 0을 나타낸다[8,12].

역사상 수학은 많은 애로 사항들을 해결해주는 주요 연장이었다. 그 중에 애매성이나 불확실성을 나타내는 문제는 과학화하기가 힘든 것들이지만 이런 불확실성의 문제를 과학화하여 그 실현 가능성에 명쾌한 답을 준 것이 확률이었다. 더불어 불확실성의 문제 혹은 과학에서 도저히 다룰 수 없는 문제들을 다룬 것이 퍼지 이론이다. 확률은 시간과 공간에 지배를 당하는 불확실성을 가지지만 퍼지는 시간과 공간에 상관없이 불확실성을 영원히 가지는 속성이 있다. 이러한 퍼지이론 중에서 퍼지측도는 집합의 경계가 분명한 확실한 원소들이 있을 때 어떤 하나의 원소가 어떤 집합에 속하는 지 애매한 상황으로서 경계가 희미한 상황을 다루는 퍼지집합과는 구분된다. 여기서 가법성(additive)은 확률의 본질적인 성질이지만 이것은 르베그(Lebesgue) 측도의 성질이며 측도라는 것은 면적이나 길이를 측정하는 척도를 말하고 판단의 확실성을 나타내는 주관확률도 물론 덧셈성을 충족시켜야 한다[30].

따라서 일반집합 X 중의 경계가 확실한 영역 B인 에니어그램의 모나드에서 잘 알지 못하는 요소 x_j인 헵타드가 모나드에 포함되는 확실성의 정도를 생각하면 x_j가 B에 포함되는『확실성』을 $g_{x_j}(B)=x_j\in B$의 정도처럼 표시해서 퍼지측도로서 관리할 수 있다. 또한 퍼지 집합은 에니어그램의 품질요소인 트리아

드의 y가 모나드 B에 대한 함수로서 관리할 수 있다.

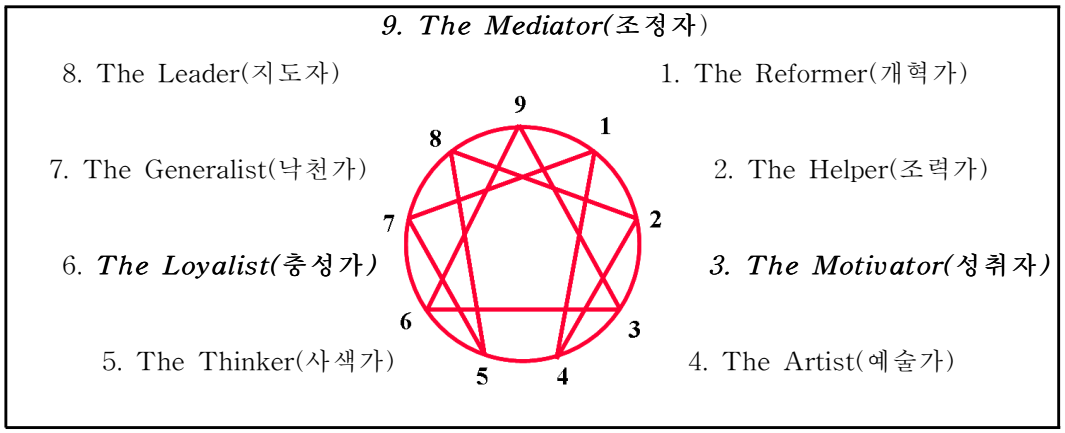
2.1.2 에니어그램

에니어그램은 에니어의 원리로 아홉개의 점을 원 주위에 배열하는 것이다. 에니어그램은 20세기에 구르지예프(Georges I. Gurdjieff)가 서양에 소개한 놀라운 다이아그램인데, 그것은 숨어 있는 내부의 연결을 통해 전체 사건들을 외부의 순서로 바라보는 간단한 방법이라는 점에서 아주 놀랍다.

본 논문에서는 에니어그램을 실용적인 예인 스마트홈 컨텍스트에서 주방모드과 외출모드에서 에니어그램을 적용해보기로 한다. 이것은 구르지예프의 제자인 베넷(J. G. Bennett)이 만든 모형에 기초한 것이다.

구르지예프는 “우리는 왜 여기에 있으며, 우리는 무엇을 원하고, 우리를 지배하는 법칙은 무엇인지를 알아야 한다” 라고 말한 것은 자신의 내면에서 요구하는 것이 무엇이며, 내가 존재하는 이유가 무엇이고, 나를 통제하고 지배하는 법칙이 무엇인지를 아는 것에 삶의 목적을 두어야 한다는 뜻이다. 따라서 모나드의 원리에 따라 자신의 삶의 목적을 알고, 헵타드의 원리에 따라 그 목적을 달성하기 위해 무엇을 해야 하는지, 트리아드의 원리에 따라 자신이 원하는 욕구가 무엇인지를 알아야 한다는 뜻이다[8,12].

에니어그램의 역사는 추정된 것에 의해서만 보면 기원전 2500년 전에 중동

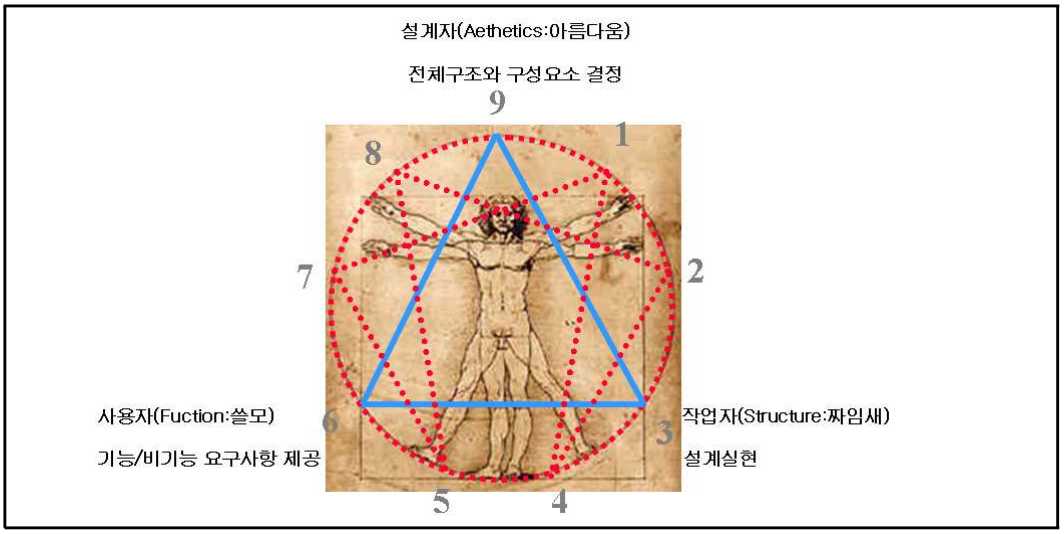


<그림 2-1> 에니어그램의 원형

<Fig. 2-1> Prototype of Enneagram

지방에서 발생되었으며 현재의 에니어그램 이론은 어떤 한 가지 근원에서 나오는 것은 아니지만 1920년대에 최초로 유럽사회에 소개한 이는 러시아의 구르지에프로서 1960년대에는 미국으로까지 퍼지게 되었다. 현재 에니어그램이 가장 많이 보급된 나라는 미국으로 현대사회에서의 심리학을 중심으로 다양한 산업분야에서 응용에 관한 연구가 활발히 전개되고 있으며 심리학의 에니어그램은 <그림 2-1>과 같이 인간의 유형이 아홉 가지라고 주장한다[8].

구체적으로 살펴보면 외면적인 사건들은 원의 다른 부분, 곧 과정중의 다른 단계들과 분명하지 않은 내면적 관계를 지니고 있다. 헵타드의 원리, 곧 음악의 옥타브 음계와 같이 전체 순환의 내면에 존재하는 일곱겹의 조화를 적용함으로써 그것을 발견한다. 7이 단일성으로 돌아가고자 하는 방식은 7을 1로 나누어 주기만 하면 발견할 수 있다. 그 결과는 0.1428571428571…로, 1-4-2-8-5-7의 여섯자리를 순환고리로 하는 순환소수이다. 이 순환고리는 트리아드의 점을 이루는 3의 배수를 제외한 1에서 9사이의 모든 수를 포함하고 있다. 트리아드와 헵타드는 다른 쪽이 결여하고 있는 과정을 보충하여 준다. 각각의 점으로 표기된 영역 또는 단계에서는 다른 단계를 미리 보거나 뒤돌아보게 안내함으로써 과정을 생각할 수 있도록 도와주는 안내자의 역할을 한다. 만약 그 내면의 트리아드와 그 과정을 하나의 순환으로 만드는 중간단계를 알 수 있다면, 에니어



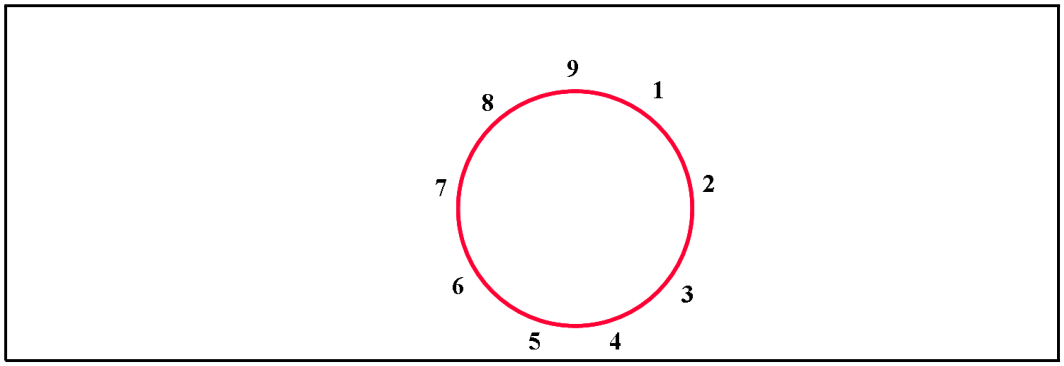
<그림 2-2> 비트루비우스 삼각형

<Fig. 2-2> Triangle of Vitruvius

그램의 구조를 자기인생의 어떤 순환적 사건에도 적용할 수 있다. 트리아드와 헵타드는 각자 상대방이 갖고 있지 못한 것을 갖고 있는 보완물이며, 애니어드의 아홉 단계에서 이해된다. <그림 2-2>는 비트루비우스(Vitruvius) 삼각형의 아키텍쳐 사상과 에니어그램의 트리아드 사상을 비교한 모형이다. 여기서 설계자, 작업자, 사용자간의 역할분담을 알 수 있다[8,12,18,30].

(1) 모나드

<그림 2-3>에서와 같이 원의 의미에 대한 해석으로, 원은 통합과 전체, 단일성을 나타내며 과정의 연속을 나타낸다. 성경의 요한복음에서도 “그들이 모두 하나가 되게 해 달라” 고 간구할 때 하나라는 표현을 쓴다. 이것은 모든 것은 하나에서 비롯됨을 나타내는 것으로 공간, 시간, 형태에서 범위를 한정짓는 것을 나타낸다고 볼 수 있다. 원은 닫혀 있음으로써, 확실한 공간을 차지하고 울타리를 친 것 같기 때문이다. 원전체는 항상 ‘지금’을 나타내는 반면, 원둘레는 순간의 연속인 ‘시간의 흐름’을 나타내고, 독립적인 전체로서 그 자체의 형태 또는 특징이 있다. 본 논문에서는 서비스 아키텍쳐의 사상을 정의할 때 5W1H의 Why/Who, MVC의 View를 의미할 때 쓰이며 시스템 다이나믹스에서는 사건 영역으로서 문제정의 및 브레인스토밍에 해당한다[8,12,18,30].



<그림 2-3> 에니어그램의 모나드

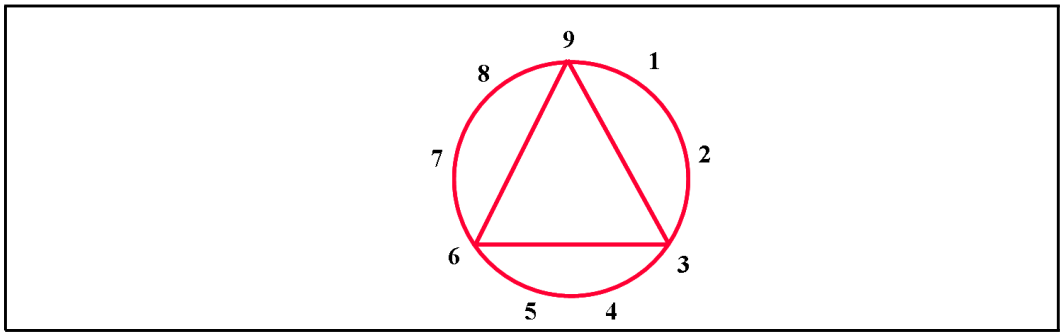
<Fig. 2-3> Monad of Enneagram

(2) 트리아드

삼각형의 의미에 대한 해석으로 구르지예프의 해석을 보면, 삼각형은 세 힘의 상호작용의 결과로 인해 존재함을 상징하며 결합을 의미하는 데 성경의 전도서에서도 “삼겹줄은 쉽게 끊어지지 않는다” 고 하여 이를 3의 법칙이라 하는데, 세 힘에는 능동적인 힘, 수동적인 힘, 조화로운 중립의 힘이 있다. 물리학에서 원자는 양성자, 전자, 중성자로 구성되어 있으며, 현대물리학에서는 강한 힘, 약한 힘, 전자기 힘 등 세 힘이 존재한다.

<그림 2-4>에서 원에 삼각형이 결합되는 것과 같이 한 체계에 두 개의 그림이 결합되면 주요 과정이 된다. 여기서 점 9, 3, 6에서 주요과정이 변화를 위해 개방적으로 일어나고, 점 9는 원의 시작과 끝이며 삼각형의 정점을 이루는 완전한 형태로서 근원의 목적이 실현되는 자리이다. 즉, 점 9에서 제1과정이 시작되고, 점 3에서 제2과정이, 점 6에서 제3과정이 시작되므로 제1과정이 완성되어야만 제2과정이 진행되고, 제2과정이 완성되면 제3과정이 진행되어 이 세 과정이 모두 이루어지면 에니어그램이 완성된다.

3의 법칙에 대한 계산 원리는 십진법과 관련이 있는데, 전체가 세 힘으로 분리되는 것을 계산하려면, 1을 3으로 나누면 되는데 이때, 새로운 순환소수가 다음 페이지와 같이 나타난다. 이것은 완전한 합일과 통일을 의미하는 절대를 나타내는 숫자 1을 향해 소수 9가 끝없이 순환하는 체제임을 알 수 있다.



<그림 2-4> 에니어그램의 트리아드

<Fig. 2-4> Triad of Enneagram

1:3＝1÷3＝0＋1/3＝.333…또는 .$\dot{3}$
여기에다 다른 1/3 을 더하면 아래와 같이 6이 끝없이 계속이어지고,

2:3＝2÷3＝1/3＋1/3＝.666…또는 .$\dot{6}$
똑같은 방법으로 마지막 1/3을 더하면 9가 계속 이어진다.

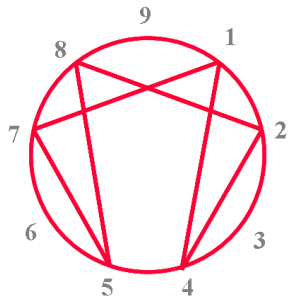
3:3＝3÷3＝2/3＋1/3＝.999…또는 .$\dot{9}$

본 논문에서는 아키텍쳐의 사상을 정의할 때 5W1H의 What, MVC의 Model을 의미할 때 쓰인다. 시스템 다이나믹스에서는 구조 영역으로서 인과모형과 시뮬레이션 설계 및 구현에 해당한다[8,12,18,30].

(3) 헵타드

<그림 2-5>와 같이 칠각형의 의미에 대한 해석으로 주기적인 숫자 7-1-4-2-8-5-7의 순환으로 아주 다양한 단계로 연결되어 있음을 의미하고, 이 다양한 단계들이 어떻게 서로 연결되어 있는지를 나타내는데 이를 '7의 법칙'이라고 한다. 성경의 잠언에서도 "참지혜가 그 집을 짓고 일곱 기둥을 깍아 만들었다"고 한다. 이것은 절대를 상징하는 하나의 원이 어떻게 나뉘어지고 다시 통합되는 지를 나타내며 존재하는 모든 것은 정지되어 있지 않고 움직이며 뭔가 다른 것으로 변형되는 것을 상징한다. 이는 변화에 작용하는 힘과 그 자체의 성질에 따라서 변화되는 방식은 다르지만 화학에서의 원소 주기율표, 음악에서의 옥타브 등으로 모든 것은 변화하여 재생되고 발전해 나감을 의미한다.

7의 법칙에 대한 계산 원리는 십진법과 관련이 있는데, 전체가 7수준의 세계



<그림 2-5> 에니어그램의 헵타드

<Fig. 2-5> Heptad of Enneagram

로 나뉘어지는 것을 계산하려면, 1을 7로 나누면 되는데 이때, 새로운 순환소수가 아래와 같이 나타난다.

1:7 = 1÷7 = 0 + 1/7 = .142857
여기에다 다른 1/7 을 더하면 아래와 같이 숫자의 형식이 끝없이 반복된다.

2:7 = 2÷7 = 1/7 + 1/7 = .285714

3:7 = 3÷7 = 2/7 + 1/7 = .428571

4:7 = 4÷7 = 3/7 + 1/7 = .571428

~~5:7 = 5÷7 = 4/7 + 1/7 = .714285~~ (본 논문에서 주요 알고리즘의 사상이 됨)

6:7 = 6÷7 = 5/7 + 1/7 = .857142

7:7 = 7÷7 = 6/7 + 1/7 = .857142 + .142857 = .99999 = $.\dot{9}$

따라서, 이것도 완전한 합일과 통일을 의미하는 절대를 나타내는 숫자 1을 향해 끝없이 순환하는 체제임을 알 수 있다.

본 논문에서는 아키텍쳐의 사상을 정의할 때 5W1H의 Where, When/How, MVC의 Controller를 의미할 때 쓰이며 밑줄 그은 *"7-1-4-2-8-5-7"*의 헵타드는 주요 알고리즘의 사상을 제공한다. 시스템다이나믹스에서는 행태 영역으로서 행태의 역사적 결과와 패턴분석에 해당한다[8,12,18,30].

2.1.3 퍼지집합과 퍼지측도

수학적으로는 두 개의 기본적인 Fuzziness의 개념이 있다고 한다. 하나는 퍼지 집합이고 또 하나는 퍼지 측도이다. 두 개의 개념을 추상적으로 표시하면 다음의 <그림 2-6>의 (a), (b)처럼 된다[48].

(a)는 일반집합 X 중의 경계가 확실치 않는 영역 A에, 잘 알고 있는 요소 x가 포함되는 『확실성』의 정도를 생각하는 것이다. (b) 는 일반집합 X 중의 경계가 확실한 영역 B 에, 잘 알지 못하는 요소 y가 B 에 포함되는 확실성의 정도를 생각하는 것이다. 여기서 잘 알지 못하는 요소라는 의미는 y가 구체적으로 어느 요소인가 하는 것이 특정되지 않았다는 의미이다.

y가 B에 포함되는『확실성』을 $g_y(B) = y \in B$의 정도처럼 표시해서 퍼지 측도라고 한다.

즉 퍼지 측도는 요소 y가 용기 B 에 대한 함수이다. 한편, 퍼지 집합의 멤

버십 함수는 요소 x 에 대한 함수이다. 퍼지 집합과 퍼지 측도는 두 가지의 접점을 가진다. 먼저, 만약 용기도 알맹이도 모두 『애매성』이 없으면 양자는 명확히 일치한다. 실제, 크리스프 집합 (보통의 집합) B 의 정의함수를

$$x_B(X)=\begin{cases}1 & x\in B\text{일때}\\ 0 & x\notin B\text{일때}\end{cases}$$ 라고 하면 "$x\in B$" 의 정도 (2.1)

는 퍼지 집합으로서의 정도인 동시에 퍼지 측도로서의 정도다.

그리고 $g_x(B)=x_b(x)$ 가 된다.

집합의 정의 함수와 동일한 $g_x(B)$는 디렉의 측도라는 것이다.

한편, 용기도 알맹이도 모두 『애매성』이 있으면 양자는 일치하지만 "$y\in A$" 의 정도를 정의하는 데에는 퍼지 적분의 개념이 필요하다.

따라서 퍼지 정보이론은 수학적으로 퍼지 측도를 바탕으로 전개되는데 이를 정의하면, 전체집합을 X , 공집합을 ∅ 라고 정의하면, X의 원소로 구성될 수 있는 가능한 모든 조합의 집합을 멱집합 P(X) 로 정의할 수 있다.

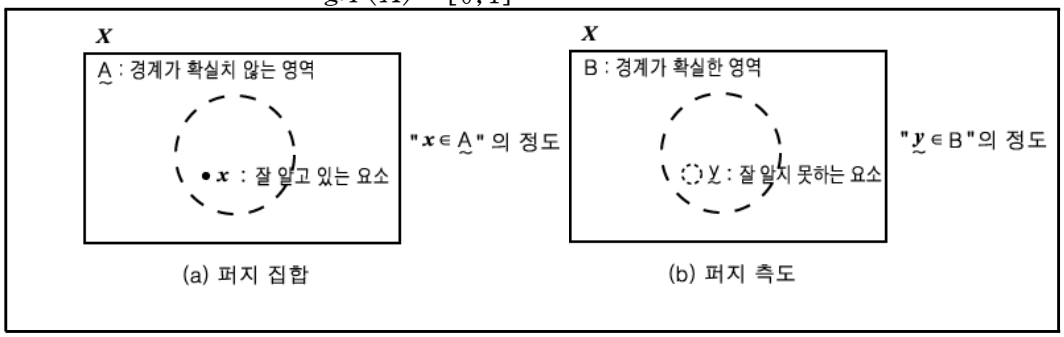
퍼지 측도는 다음과 같이 (i) 경계조건, (ii) 단조성의 Sugeno 측도와 (iii) 연속성을 기본 공리로 한다.

퍼지 측도: (i) 경계조건: $g(\varnothing)=0, g(X)=1$ (2.2)

(ii) 단조성: $\forall A, B\in P(X),\ A\subseteq B\Rightarrow g(A)\le g(B)$

(iii) 연속성: if $A_1\subseteq A_2\subseteq\ldots$ or $A_1\supseteq A_2\ldots\supseteq\ldots$ then $\lim_{i\to\infty} g(A_i)=g(\lim_{i\to\infty} A_i)$

위의 퍼지 측도 g 는 $g: P(X)\to[0,1]$의 함수이고, 이 중에서 하한 확률이라고



<그림 2-6> 퍼지집합과 퍼지측도의 비교

<Fig. 2-6> Comparison of fuzzy set and fuzzy measure

할 수 있는 Belief 측도, 상한 확률이라고 할 수 있는 Plausibility 측도, 그리고 확률 측도는 각각 다음의 세부 공리(iv)를 만족한다.

Belief 측도: (iv-1) $g \leftarrow Bel, Bel(A \cup B) \geq Bel(A) + Bel(B) - Bel(A \cap B)$ (2.3)

Plausibility 측도: (iv-2) $g \leftarrow Pl, Pl(A \cap B) \leq Pl(A) + Pl(B) - Pl(A \cup B)$

확률 측도: (iv-3) $g \leftarrow \Pr, \Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$

[정의] 기본 확률 부과치 (BPA): $m: P(X) \rightarrow [0,1]$ 인 함수이며,

① $m(\emptyset) = 0$ ② $\sum_{A \in P(X)} m(A) = 1$ 의 성질을 만족한다. (2.4)

[정리 1] $Bel(A) = \sum_{B \subseteq A} m(B) \ \forall A, B \in P(X)$

[정리 2] $Pl(A) = \sum_{A \cap B \neq \emptyset} m(B) \ \forall A, B \in P(X)$

증거에 대한 정도를 수치로 나타낸 것을 위의 정의와 같이 "기본 확률 부과치"라고 하고 약자로 BPA(Basic Probability Assignment)로 정의한다.

정리 1에서 Belief 측도는 원소 집합 B 가 집합 A 와 그의 특별한 부집합들에 속하는 전체 믿음 또는 증거를 나타내며, 정리 2에서 Pausibility 측도는 원소 집합 B 가 집합 A 와 그의 특별한 부집합에 속하는 믿음 또는 증거 뿐만 아니라 집합 A 와 중첩된 다른 원소 집합들의 믿음이나 증거를 모두 합한 수치를 나타낸다. 여기서, 임의의 원소 집합 $A \in P(X)$ 를 선택했을 때, m(A) > 0 이면 증거를 포함한 것으로 이러한 원소 집합을 "초점 원소(Focal Element)"라고 하며, 초점 원소의 집합 F 와 해당 기본 확률 부과치인 BPA m 의 쌍 {F,m} 을 "증거체(Body of Evidence)"라고 한다.

역으로 Belief 측도에 의해 BPA m 을 구할 수 있으며, 다음 식이 성립한다

$$m(A) = \sum_{B \subseteq A} (-1)^{|A-B|} Bel(B)$$ 여기서, |A-B|는 집합A-B의 개수를 나타낸다. (2.5)

Belief 측도와 Plausibility 측도는 앞에서 언급한 바와 같이 확률의 상한값과 하한값을 결정짓는 수치이며 증거체의 기본 확률 부과치 m 을 적절히 사용하여 확률 분포 함수의 범위를 예측하는 데 중요한 자료가 된다.

2.2 전사적 구조

2.2.1 개요

다수의 서비스 사용자가 생활하는 스마트홈 환경에서는 컨텍스트가 하나만 존재하는 것이 아니라 시간의 흐름에 따라 생성, 소멸되는 다중 컨텍스트가 동시에 존재함에 따라 컨텍스트, 자원, 역할의 충돌을 감지하며 도메인의 범위에 따라 이를 효율적으로 관리해줄 수 있는 전사적 아키텍쳐(Enterprise Architecture)가 요구된다.

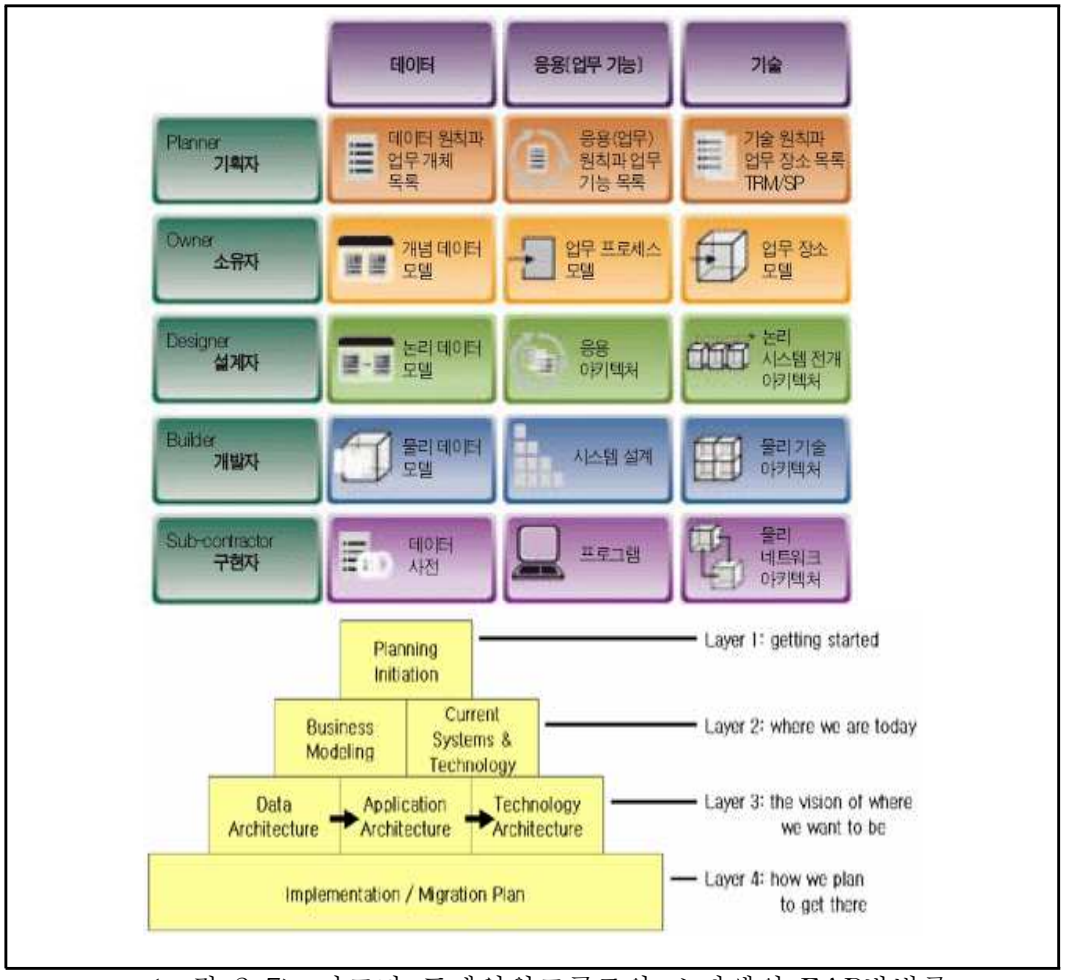
자크만이 1987년 처음 프레임워크 개념을 발표한 이후 EA 분야는 스티븐 스피왁 박사(Steven H. Spewak Ph.D)의 EAP(Enterprise Architecture Planning) 방법론으로 자크만이 제시한 아키텍처 개념과 프레임워크를 어떻게 구현할 것인가에 대해서 "방법론은 일관되고 조리 있고 반복적인 방법으로 활동을 수행할 수 있는 문서화된 접근"이라고 정의하며, "일상적으로는 프로세스와 산출물을 정의한 것"이라고 설명한다[9,19].

이러한 EA는 '지역 최적화'가 아니라 '전역 최적화'를 추구하므로 EAP 수행과정에서도 '효율'보다는 '효과'를 중시해야 한다. 따라서 본 절에서는 자크만 프레임워크에서 EA 구조에서 다양한 스테이크홀더의 관점(Perspective)의 차이를 살펴보고, 스티븐 스피왁의 EAP 방법론에서 절차를 검토하여 이를 MVC기반의 에니어그램의 트리아드와 헵타드의 원리와 접목시켜 본다. 본 논문에서 아키텍쳐는 기술분야마다 적용되는 지식의 요체로서 어떤 목적을 달성하기 위해서 상호유기적인 요소들이 MECE(Mutually Exclusive, Collectivery Exhaustive)하게, 즉 중복없이 누락없이 분리가능하게, 집합되어 있는 구조를 말한다. 검토된 아키텍쳐에서 이벤트관리의 EDA(Event Driven Architecture) 및 OEOR(One Event One Response)와 컨텐츠관리의 OSMU(One Source Multi Use)와 컴포넌트 코드관리의 WORA(Write Once Run Anywhere)의 자바사상 및 WOPA(Write Once Publish Anywhere)의 플래시사상을 반영하여 유연성, 가용성의 품질속성을 중심으로 다중 컨텍스트에서 컨텍스트, 자원, 역할의 스마트홈 컨텍스의 객체간의 충돌할 수 있는 의도를 사전에 감지하여 파악함으로써 전체 최적화를 향상시켜 복잡도와 성능을 개선하는 MVC기반의 전사적아키텍쳐를 살펴본다.

2.2.2 엔터프라이즈 아키텍쳐

(1) 자크만 프레임워크 구조와 스피왁 EAP 방법론의 관계

자크만이 1987년 처음 프레임워크 개념을 발표한 이후 스피왁 박사의 EAP 방법론은 <그림 2-7>과 같이 자크만의 프레임워크에서 상위 2개의 뷰(행, 열)인 볼파크 뷰(Ballpark View)와 오너 뷰(Owner's View)를 대상으로 한다. 3번째 뷰 이하는 시스템 설계 단계로 간주하며 EAP 완료 이후에 진행할 업무로 본다. EAP 방법론은 계획의 시작 단계, 현황 파악, 목표 내용, 목표에 도달하는 방법의 순서로 위에서 아래로 정의되어 있다[9,19,20].



<그림 2-7> 자크만 프레임워크구조와 스피왁의 EAP방법론

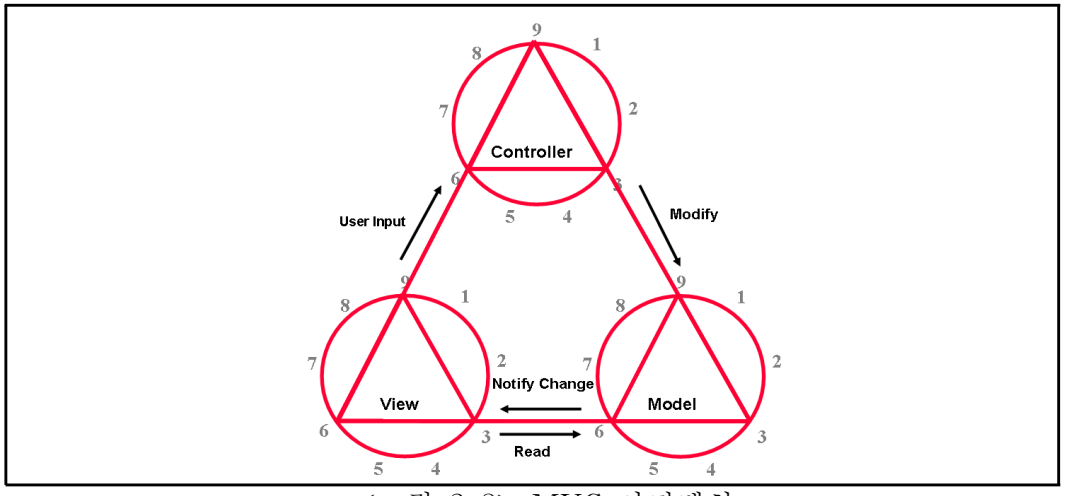
<Fig 2-7> Zachman framework and EAP of Spewak

여기서 3번째 계층의 DA → AA → TA가 화살표로 연결되어 있는데, 프로젝트의 효율을 중요시한다면 DA(Data Architecture), AA(Application Architecture, TA(Technical Architecture)를 3개의 팀이 동시에 작성해 프로젝트 기간을 최소화해야 할 것이다.

그러나 동시 작업은 위에서 언급한 EAP의 장점을 심각하게 훼손한다. 즉, 팀들 간의 용어 정의, 이해 수준, 목표 시스템에 대한 시각 차이가 존재하기 때문에 EAP의 일치성, 통합성을 훼손할 수 있다. 또한 AA와 TA는 데이터에 대한 의존도가 있기 때문에 DA가 완성되기 전에는 완전한 AA와 TA를 작성할 수 없다. 따라서 3가지 아키텍처는 차례대로 작성돼야만 하기에 화살표가 프로세스를 강조하는 것이다. 따라서, 전통적인 EA를 통하여 구조에서 방법론을 분리하여 제시함으로써 완전하게 해석이 가능함을 보여주고 있다[19,20].

(2) MVC 아키텍처

<그림 2-8>과 같이 MVC(Model-View-Controller)란 사용자 인터페이스를 성공적이며 효과적으로 데이터 모형에 관련시키기 위한 방법론 또는 설계 방식 중 하나이다. MVC 방식은 역할기반 모델링의 RNS(Role-Norm-Saction)와 에이전트 인터페이스 모델링의 PAC (Presentation-Abstraction-Control)와



<그림 2-8> MVC 아키텍처

<Fig 2-8> MVC Architecture

UML(Unified Modeling Language) 모델링의 BCE(Boundary-Control-Entity), 유저인터페이스의 MVP(Model-View-Presenter) 등에서 널리 사용된다. MVC 형식은 목적 코드의 재사용에 유용한 것은 물론, UI(User Interface)와 응용프로그램 개발에 소요되는 시간을 현저하게 줄여주는 형식이라고 많은 개발자와 서비스 제공자들이 평가하고 있다[25,44,45,47].

MVC 형식은 소프트웨어 개발에 사용될 세 가지 구성요소를 제안한다.

① Model(모형) : 소프트웨어 응용과 그와 관련된 고급 클래스 내의 논리적 데이터 기반 구조를 표현하며 이 목적 모형은 UI에 관한 어떠한 정보도 가지고 있지 않음.

② View(뷰) : UI 내의 구성요소들을 표현하는 클래스들의 집합

③ Controller(제어기) : 모형과 뷰를 연결하고 있는 클래스들을 대표하며, 모형과 뷰 내의 클래스들 간에 통신하는데 사용됨

2.2.3 스마트홈

스마트홈은 “생활 환경의 지능화, 환경 친화적 주거생활” 거주공간으로 정의되며 다양한 센서와 네트워크 기술을 바탕으로 거주자 및 주거환경에 대한 컨텍스트 정보를 수집하여 서비스 사용자에게 차별화된 개인화 서비스를 제공한다. <표 2-1>과 같이 홈네트워크 환경에서는 홈오토메이션 관점에 장치간의 상호작용에 맞추어져 있지만 스마트홈 환경에서는 서비스 사용자의 의도에 맞는 상황인식이 중요해지기 때문이다.

<표 2-1> 스마트홈과 홈오토메이션의 비교

<Table 2-1> Comparison of Smat Home and Home Automation

	스마트홈	홈 오토메이션
서비스 유형	통합된 정보	단일 정보
취득 정보량	센서 > 사용자입력	사용자입력>가전장치
사용자 ID	○	×
컨텍스트	○	×

홈오토메이션은 미리 입력해놓은 명령을 수행하는 가전제품을 기반으로 서비스 사용자의 움직임이나 주위환경의 변화에 따라 자동으로 동작한다. 하지만 모든 서비스 사용자에게 동일한 서비스를 제공하여 다양한 요구를 모두 만족시키지 못하고 서비스 사용자의 단순한 동작에 반응하여 적절하지 못한 서비스를 제공하는 경우도 있다. 스마트홈 또한 서비스 사용자의 쾌적한 생활을 위해 서비스를 제공한다는 목적은 동일하다. 하지만 그 대상을 구분하여 개개인에게 맞춤형 서비스를 제공하고 컨텍스트를 이용하여 서비스 사용자의 요구에 부응하는 서비스를 제공한다. 스마트홈에서 사용하는 컨텍스트에는 위치, 신원, 시간, 동작 등이 있다. 스마트홈에 관련된 연구의 공통된 부분은 거주자와 거주환경정보를 수집, 인식, 행동하는 시스템을 개발하는 것이다. 이러한 시스템은 다양한 센서를 이용하여 정보를 수집하고, 수집된 정보를 이용하여 적절한 서비스를 제공하기 위해 퍼지 또는 신경망 등의 인공지능기술들이 사용될 수 있다. 그러나 정확하고 효율적인 시스템을 구축하기 위해서는 거주자와 거주환경정보에 대한 최소정보단위가 무엇이며, 이를 어떠한 센서를 사용하여 수집, 인식을 해야 하는 것에 대한 연구가 절실히 필요하다[1,2,4,17].

집안의 가전제품을 모두 연결해 한꺼번에 제어하는 스마트홈 서비스가 보안, 엔터테인먼트를 넘어 헬스케어, 실버케어로까지 확장되고 있다. 시장현황을 보면 2007년도 스마트홈 세계시장은 1,027억 달러를 예상됨에 따라 스마트홈은 이미 특정 국가의 특정 기업이 독식하기엔 너무 방대한 시장이므로 세계적인 전자 업체들이 주요 표준 컨소시엄에서 영향력을 확대하기 위해 노력하고 있다[1,42].

LG전자와 대우일렉은 국내 업체들 간의 협력을 통해 시장을 키워 세계가 한국을 벤치마킹하게 하자는 전략을 추진 중이다. 2005년 10월 구성돼 이미 회원사가 44개로 불어난 LnCP(Living network Control Protocol) 컨소시엄이 대표적이며 LG전자가 개발한 LnCP 솔루션을 44개 회원사들과 공유하는 컨소시엄이다. 또한 LG전자는 DLNA(Digital Living Network Alliance), UPnP 등 글로벌 컨소시엄에 주도적으로 참여하고 있으며 대우일렉도 다양한 전자, 통신 기기 간의 서비스 호환이 가능한 OSGi UFK 운영위원으로 활동하고 있다.

삼성전자는 2001년 홈비타라는 이름으로 스마트홈 사업을 시작해 2006년 2만 가구에 서비스를 공급하고 있으며 HANA(High Definition Audio Video

Network Alliance) 컨소시엄, DLNA 등 글로벌 표준 컨소시엄에서 이사회 멤버로 활동하고 있다.

정보통신정책연구원(KISDI)이 최근 2006년에 홈네트워킹 전문가를 대상으로 설문조사한 결과에 따르면 국내 홈네트워킹에 가장 적합한 기술로 무선랜이 선택됐다. 다음으로 이더넷과 전력선이 적합한 것으로 평가됐다. 이는 전력선의 경우 다양한 기술간 융합이 문제가 되며 기타 유선솔루션은 배선문제로, 무선 중 블루투스는 현재 홈 환경에 적합하지 않기 때문이다. 그러나 특정 기술이 독자적으로 발전하기보다는 여러 기술이 주택특성, 사업자의 의지와 맞물려 혼재된 형태로 발전할 것이다.

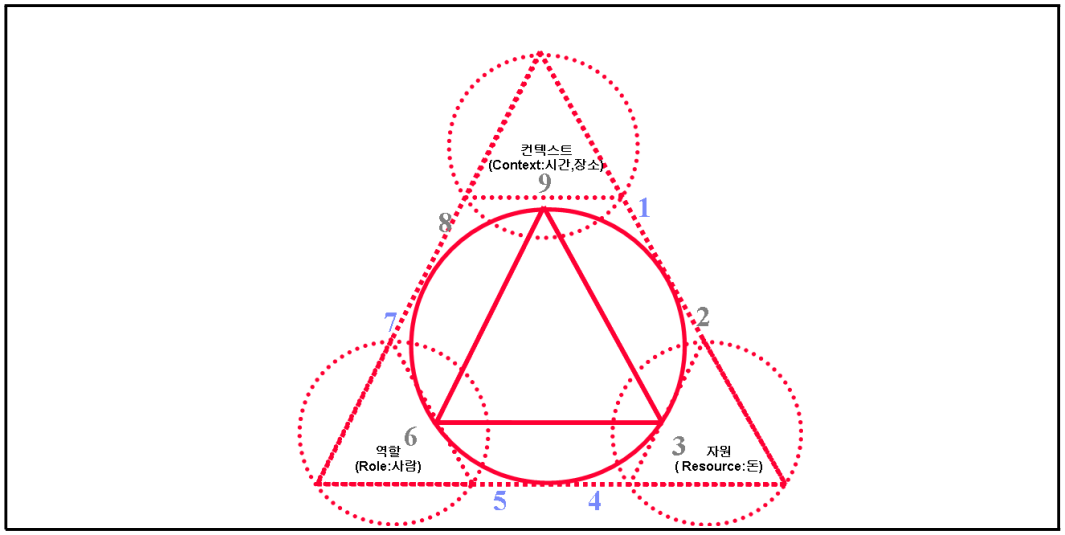
개별기기간 네트워킹이나 컨트롤 네트워킹의 경우 기술적 특성으로 인해 블루투스나 무선랜과 같은 무선계 기술이 주도할 것이다. 또한 신규 주택은 유선과 무선 등 다양한 댁내 기술을 선택할 수 있지만 기존 노후화된 주택에는 무선기술이 더욱 선호될 것으로 예측된다. 유선 전송 기술 분야에서는 A/V기기, PC간 연결은 IEEE1394 우세하며 전체 네트워크 부문에서는 홈PNA(Home Phoneline Networking Alliance)와 PLC가 경합할 것이다.

가정내 서비스 다양화에 대응하기 위해 서버기능이나 축적기능까지 포함되면서 정보가전 시스템의 중심으로서 홈게이트웨이가 가정내 접속 단말의 연결이나 프로토콜의 다양성을 해결해야 하고 항상 인터넷에 접속해 있어야 하기 때문이다. 국내에서 개발중인 홈게이트웨이 관련 기기들은 다수 업체가 개발하고 있으나 이러한 기기들의 상호운용성에 대한 통합표준은 부재한 상황이다. 일례로 국내 PLC 방식은 업체마다 다른 표준을 사용해 홈게이트웨이의 표준화된 모델 개발이 곤란하고 일반인의 가전기기 선택의 폭을 제한한다. 삼성의 경우 Lonworks PLC 표준을 채택하고 있으며 LG전자는 LnCP 표준을 사용하고 있다. 이 때문에 홈게이트웨이에 탑재되는 미들웨어간 상호운용성을 확보하기 위한 통합 미들웨어 개발이 절실하다. 이와 관련 한국정보통신기술협회(TTA)는 PLC포럼과 연계해 PLC모뎀 등에 대한 표준화를 2006년도 상반기 마무리한다는 방침이며 2006년도 하반기에 홈네트워크 기기인증제도가 홈네트워크산업협회(www.hna.or.kr) 산하 'U홈건설협의회' 소속에서 진행한다[42].

2.2.4 컨텍스트, 자원, 역할

다수의 서비스 사용자가 생활하는 스마트홈 환경에서는 다중 컨텍스트가 동시에 존재함에 따라 <그림 2-9>와 같이 컨텍스트, 자원, 역할의 충돌을 사전에 감지하여 도메인의 범위에 따라 이를 효율적으로 관리해야 한다[2,4,17].

상황인식 기술을 효과적으로 사용하려면, 본질적으로 컨텍스트가 무엇이고, 이를 어떻게 사용할 수 있는지 그리고 이를 사용하기 위한 기술 구조에 대한 이해가 필요하다. 컨텍스트에 대한 이해는 응용 개발자가 응용 서비스에 어떤 컨텍스트를 활용할지를 선택할 수 있도록 할 것이며, 컨텍스트를 활용하는 방법을 이해하면 개발자가 응용 서비스에서 지원할 상황인식 행위가 무엇이 될지를 결정할 수 있을 것이다. 또한 컨텍스트와 관련된 기술 구조에 대한 이해는 개발자가 응용 서비스를 용이하게 구축하는 것을 지원할 것이다. 이러한 기술 구조의 이해를 위하여 컨텍스트를 기반으로 하는 '서비스'와 '추상화'의 이해가 우선 요구된다. 이러한 분야를 포괄적으로 포함한 상황인식 컴퓨팅의 기술적 배경을 이해하기 위하여 본 항에서는 컨텍스트의 개념에 대하여 논한다[28,32].



<그림 2-9> 컨텍스트, 자원, 역할과 에니어그램과의 관계

<Fig. 2-9> Relation of Context, Resource, Role and Enneagram

컨텍스트에 대한 정의에 대한 연구 중 최초로 상황인식에 용어와 정의를 성공적으로 소개한 사례인 Schilit와 Theimer의 경우, 컨텍스트는 '위치'를 의미하는 것으로 근접한 사람과 사물의 확인 및 이러한 실체에 대한 변화를 의미하였다. 이러한 예를 이용하여 컨텍스트를 정의하는 방식은 다른 응용에 적용하기에는 어려움이 있다. 한편 컨텍스트의 동의어를 단순히 활용한 경우, 즉, 컨텍스트를 주위 환경 또는 처해 있는 상황으로 언급하여 정의한 경우가 있다. 여기에서 표현한 정의와 함께 컨텍스트를 위한 동의어를 단순하게 사용한 정의들은 실제 적용에 많은 어려움이 있다. 이런 의미에서 Schilit와 Pascoe의 정의는 실제 적용적 측면에서 가장 근접한 정의로 보인다. Schilit는 컨텍스트의 중요한 측면으로 '어디에 존재하고 누구와 함께 있으며, 주변에 무슨 자원이 있는지'를 컨텍스트로 정의하였다. Pascoe는 컨텍스트를 '특정 관심이 가는 실체의 물리적 개념적 상태의 부분 집합'으로 정의하였다. 이러한 정의는 한정적으로 활용가능하다. 컨텍스트는 응용 및 서비스 사용자 집합과 관련 있는 모든 주변 상황에 대하여 관여하게 된다. 이러한 컨텍스트는 주변 상황에 따라 수시로 변하기 때문에 모든 주변 상황과 어느 측면이 중요한지를 열거할 수 없게 된다.

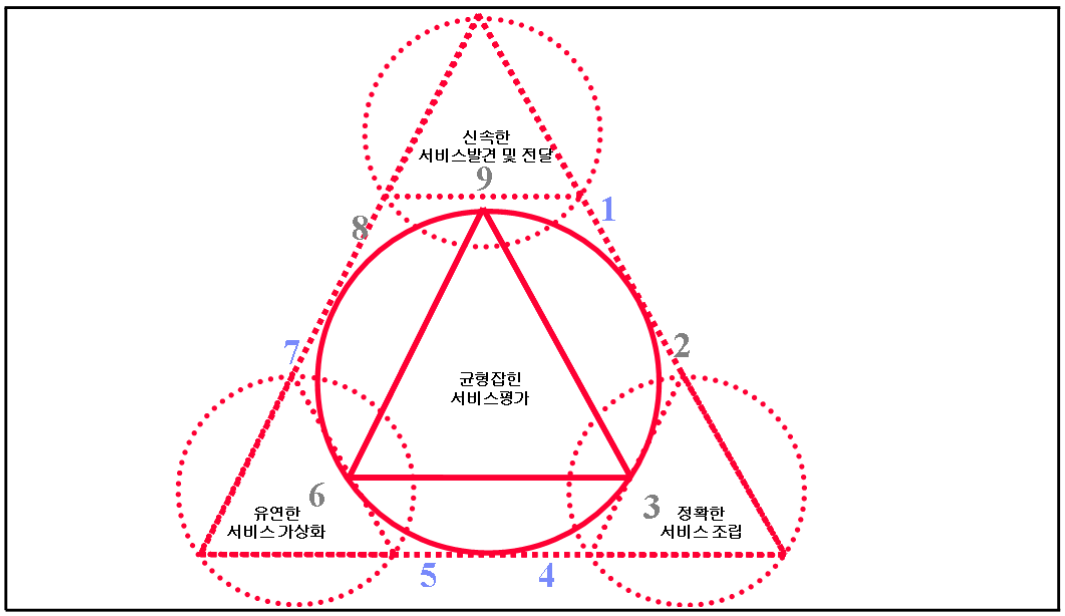
컨텍스트의 본질적인 정의는 "실세계에 존재하는 실체의 상태를 특징화 하여 정의한 정보"라고 정의할 수 있으며, 여기서 실체란 인간, 장소 또는 사람과 서비스간의 상호 작용을 의미한다고 할 수 있다. 이러한 정의는 개발 시 주어진 응용 서비스 시나리오를 위한 컨텍스트 전개 작업을 용이하게 할 수 있다. 만약 이러한 정보가 상호 작용하여 참여자의 컨텍스트를 특성화 할 수 있으면, 그 정보가 컨텍스트가 된다고 볼 수 있다[26,36,37].

따라서 다중 컨텍스트가 동시에 존재함에 따라 컨텍스트, 자원, 역할의 충돌을 사전에 감지하기 위하여 광주과학기술원에서 제안한 컨텍스트를 5W1H형태로 정형화하여 응용서비스에 따라 사용가능하게 한 개념과 본 논문에서 제안한 개념을 조합하여 제3장에서 자세하게 컨텍스트 및 역할 충돌을 해결하는 아키텍쳐를 설계하고 구성할 것이다[2,17].

2.3 서비스생명주기

2.3.1 개요

<그림 2-10>과 같이 스마트홈 컨텍스트에서 요구하는 MVC 아키텍쳐에 충실한 서비스지향 아키텍쳐의 신속한 서비스 발견 및 전달, 정확한 서비스 조립, 유연한 서비스 가상화를 우선적으로 고려하기 위하여 홈네트워크 서비스 미들웨어와 개방형 서비스 게이트웨이, 통합형 서비스 플러그인, 웹서비스의 표준화가 요구되는데, 외부에서 서비스를 제공하는 사람의 입장에서는 각 가정에 있는 홈네트워크가 달라짐에 따라 다른 형태의 서비스를 제공해 주어야 하며 외부 네트워크가 n개의 표준이 있고 홈네트워크가 m개의 표준이 존재하면 서비스를 제공해주는 방법은 $n \times m$ 개가 존재하게 된다. 하지만 서비스 미들웨어, 서비스 게이트웨이, 서비스 플러그인, 웹서비스를 표준화시킨다면 그 복잡도는 $n+m$으로 떨어지게 되며 서비스발견을 위해 SSDP, SOAP 같은 인터넷기반 통신프로토콜을 정의한 미들웨어인 UPnP 서비스전달을 위한 실행환경을 정의한 OSGi, 서비스 플러그인과 툴-통합 플랫폼인 이클립스 플랫폼, 서비스 가상화를 위한 ESB를 이용한 웹서비스가 사실상 표준이 되고 있다[1,7,10,11,39].



<그림 2-10> 서비스생명주기 개념도

<Fig. 2-10> Concept diagram of SLC

그리고, 일반적인 요구사항에 대한 정의를 살펴보면, Robertson은 요구사항을 '제품을 만들기 전에 발견해야 할 모든 것들', '제품이 해야 할 모든 것들과 가져야 할 품질'이라고 정의하였다. 소프트웨어 아키텍쳐의 라이프사이클에서는 후반부에 발생하는 요구사항의 오류를 수정하는 비용은 개발초기단계인 요구공학 단계에서 오류를 검출하고 수정하는 비용의 200배가 소요되며 인식된 소프트웨어 문제점 중 약 50%이하가 요구사항 명세에 있었고, 약 50%정도가 요구사항관리에 있었다. 또한 프로젝트성과측정방법론인 EVMS(Earned Value Management System)에서 공사계약이 15% 이상 진행된 상태에서 요구사항변경은 예산초과를 만회하기가 불가하다는 경험을 1977년 이후에 미국 국방부에 의해서 수행된 700건 이상의 주요 계약을 분석한 결과에서 말하고 있다.

이처럼 서비스 사용자의 요구사항을 올바로 분석하고 명세화하는 시나리오기반의 아키텍쳐 분석 요구공학중에서 LAAAM(Lightweight Architecture Alternative Assessment Method)과 SAF(SPAMMED(Stakeholders, Principles, Attributes, Model, Map, Evaluate, Deploy) Architecture Framework)의 특성과 CBAM의 품질속성의 반응값을 수정한 ATAM을 적용하여 비즈니스 목표로 부터 유틸리티 트리를 만들고 시스템 품질을 위한 시나리오를 설계함으로써 최대가치와 최소가치로부터 균형잡힌 최적가치의 품질속성을 찾는 체계적인 접근법으로 서비스생명주기에 따른 제안된 아키텍쳐를 평가한다[9,14,19,20,22,25].

2.3.2 서비스 발견 및 전달

(1) UPnP

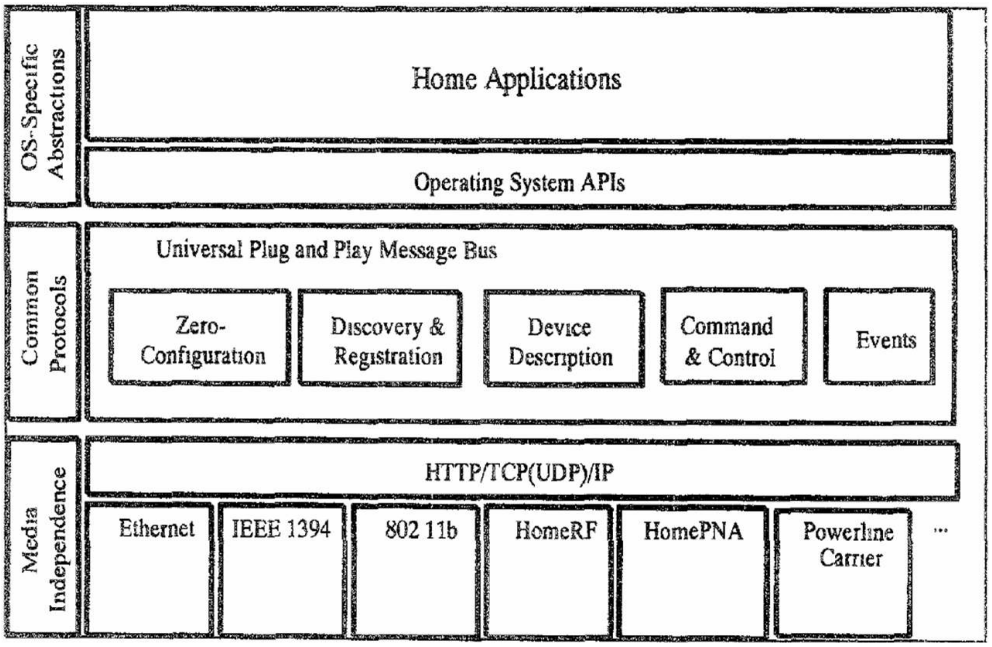
<그림 2-11>과 같이 서비스발견을 위해 인터넷기반 통신프로토콜을 정의한 미들웨어인 UPnP는 홈네트워크에서 지능형 어플라이언스들, Wireless 장치들, PC와 같은 다양한 스마트홈 장치들 사이의 Inter Networking을 보다 쉽게 지원하기 위하여 고안된 것이다. 1995년 Microsoft사에서 발표한 Plug and Play의 확장된 형태로서 진일보하여 UPnP는 다양한 장치의 Networking 대하여 Zero-configuration을 지원한다. 즉, 장치를 발견하고 장치를 통하여 사용 가능한 서비스를 탐색한 후 Universal Control Point를 사용하여 각종 스마트홈 어플라이언스들을 제어하는 목적으로 고안된 것이다[5,11,46].

UPnP의 장점은 다음과 같다.

1) 사용의 편리성 : 다양한 주변장치에 대하여 일련의 셋업과정 없이 사용할 수 있었던 Plug and Play의 진화된 형태로서 다양한 Networking device들 사이의 Zero-configuration을 제공함으로서 수많은 벤더로부터 제조된 수많은 장치들에 대하여 configuration 작업 없이 바로 동작 가능하게 하는 강력한 장점을 지닌다.

2) 표준기반 : UPnP는 TCP/IP, DHCP, NAT(Network Address Translation), Web, SSDP, GENA(General Event Notification Architecture), SOAP, HTTP와 같은 기존에 개발된 기술과 프로토콜을 그대로 또는 일부 수정하여 사용함으로써 여러 표준을 토대로 개발되어 개발자에게 친숙함을 제공하고 표준으로의 가능성을 한 단계 높이고 있다.

3) 융통성 : PnP가 PC에 대하여 주변장치에 대한 연결성를 제공한 반면 UPnP는 다양한 종류의 스마트홈 장치뿐만 아니라 레거시 장치에까지 그 영역을 확대함으로서 모든 종류의 장치에까지 확장할 수 있다[5,11,46].

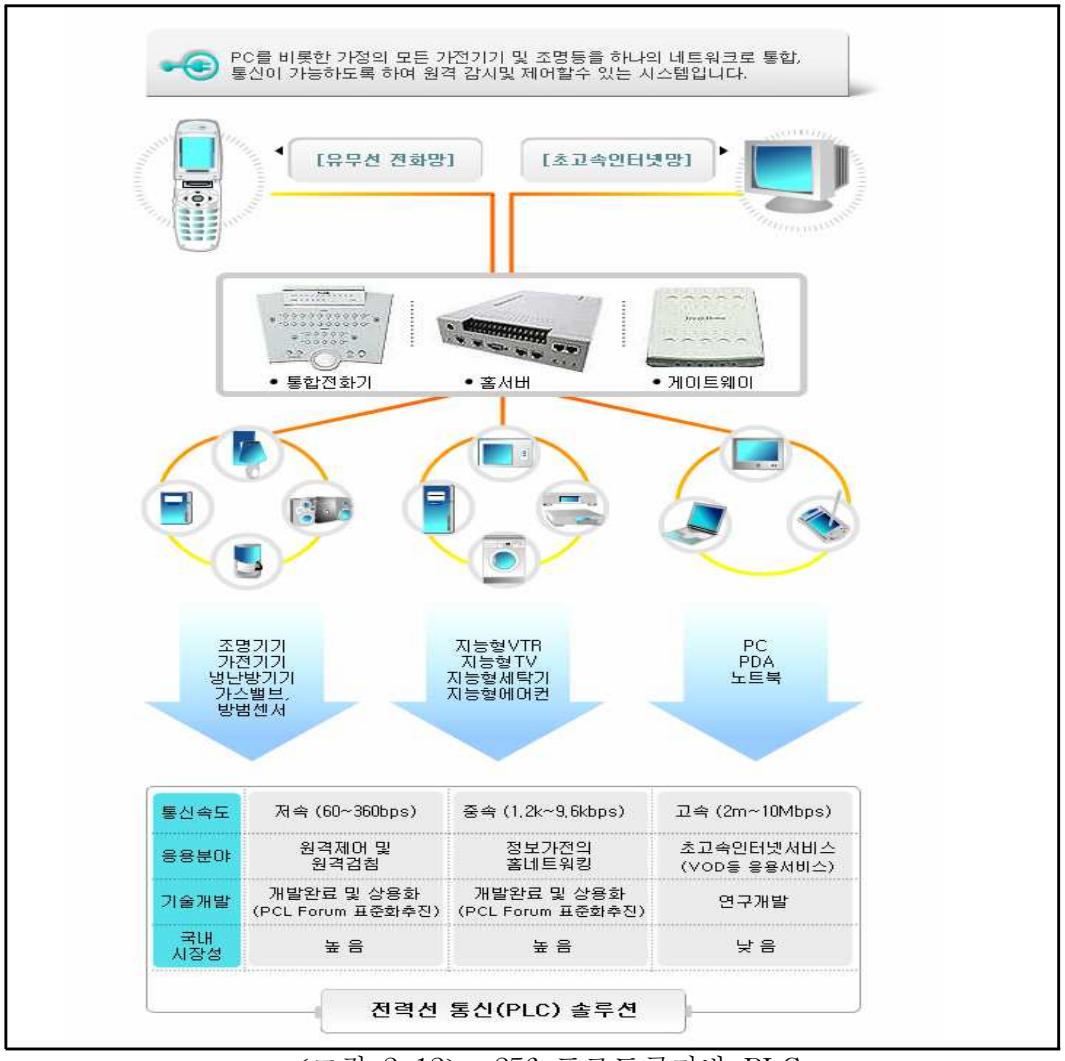


<그림 2-11> UPnP 계층

<Fig 2-11> UPnP Stack

(2) z256

2006년 현재 LG전자의 LnCP, 삼성전자의 HANA, 홈네트워크포럼의 RS485 직렬통신 등 홈네트워크 표준화 전쟁중이다. 이 가운데 <그림 2-12>와 같이 플레넷(www.planet-int.com)이 개발한 360bps급 z256 protocol은 2000년에 특허 등록 되어 홈오토메이션을 위한 제품들(전력선 스위치, 콘센트, IR Generator, 전화기 모듈 등)에 내장되어 시장에 공급되고 있으며, 데이터 통신을 위한 9.6Kbps급의 z256Protocol 개발 및 ASIC을 개발 완료하여 사업화를



<그림 2-12> z256 프로토콜기반 PLC

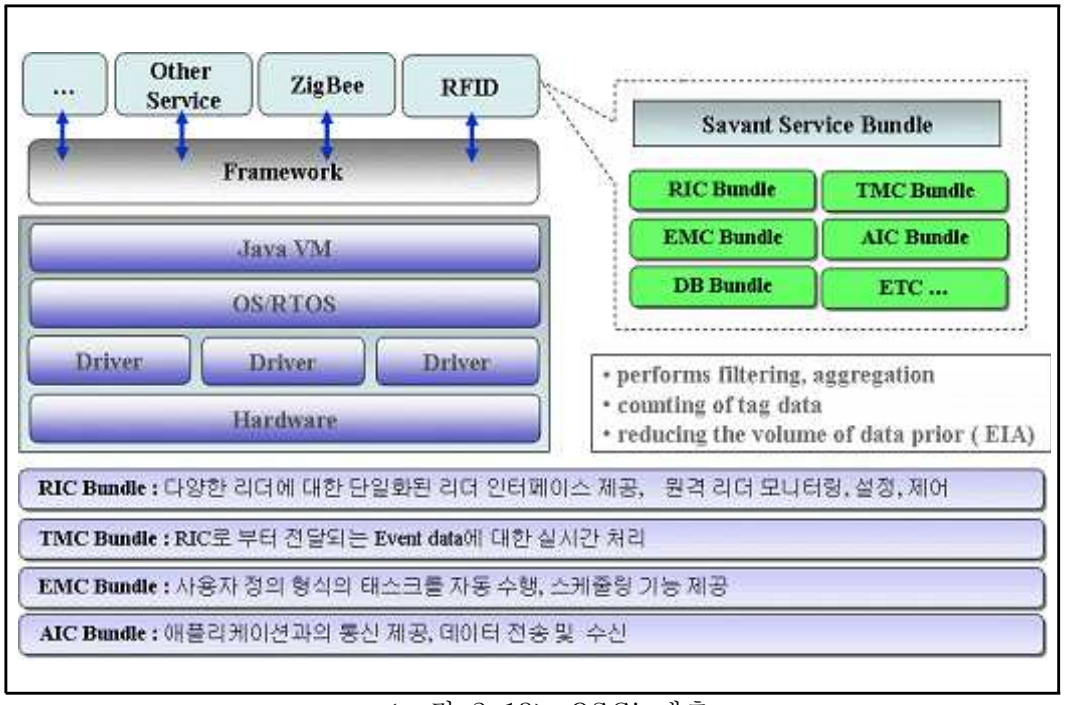
<Fig 2-12> PLC based z256 protocol

추진 중에 있다. 국내의 플레넷을 포함해 해외의 Echellon, Intellon, Domosys 등 9.6Kbps급 통신 기술을 보유하고 있는 업체이고, 고속 통신은 관련법의 규제로 인해 상용화에 상당한 시일이 소요 될 것으로 예상되고 있다. 전력선 통신이란 가정이나 사무실에 포설되어 있는 전력선을 통하여 통신신호를 100KHz ~ 30MHz의 고주파 신호로 바꿔 실어 보내고 이를 고주파 필터를 이용, 따로 분리해 신호를 수신하는 방식을 말한다. 국내에서 사용되는 전력은 60Hz의 교류신호로서 가전제품은 이를 전력변환기를 통해 직류로 바꿔 사용하며, 전력선 통신에서의 고주파 신호는 저출력의 신호이므로 일반 가전기기 작동에는 어떠한 영향을 미치지 않는다[11].

(3) OSGi

1) OSGi의 배경과 개요

<그림 2-13>과 같이 서비스전달을 위한 실행환경을 정의한 OSGi는 개방



<그림 2-13> OSGi 계층

<Fig. 2-13> OSGi Stack

형 서비스 게이트웨이의 표준을 지향하는 업계들이 모여 1999년에 만든 사실상의 표준화 단체이다. 개방형 서비스 게이트웨이란 전 세계적으로 퍼져있는 컴퓨터위주의 인터넷을 가전제품, 조명기기, 계량기 등 집에서 사용하는 모든 가전제품과 설비에 이르기까지 연결시켜 일반 가정을 인터넷의 한 부분으로 편입시켜주는 일종의 관문 역할을 해주는 기기이다. 뿐만 아니라 Bluetooth, HomePNA, HomeRF, UPnP, Jini, HAVi(Home Audio Video interoperability), z256 등 가정내의 홈네트워크의 표준도 다양하다[40].

2000년 5월에 OSGi의 첫규격인 OSGi Spec 1.0을 발표하였고 2001년 10월에 OSGi Spec 2.0을 발표하였다. 2003년 5월에는 OSGi Spec 3.0을 발표하였는데 BMW, AMI-C 등 텔레메틱스 위주로 많이 확장되었으며, 2005년 10월에는 OSGi Spec 4.0을 발표하면서 모토롤라, 노키아 등 모바일기기에 초점을 맞추었다. 그리고, 복수개의 표준을 채택한 어느 가정에도 OSGi 는 지원을 할 수 있기 때문에 홈네트워크의 단일 표준화에 너무 많은 노력과 집착을 할 필요는 없다. 실제로 오디오/비디오 기기에는 HAVi, PC와 같은 정보기기에는 HomePNA나 Ethernet 라인을 통하여 정보를 주고 받을 수 있게하는 UPnP, 냉장고나 마이크로웨이브 오븐 같은 간단한 콘트롤만 요하는 가전제품에는 에셜론의 론워크, 플레넷의 z256과 같은 PLC 표준으로 여러개 구성할 수 있다[11].

2) OSGi의 정의

다음은 OSGi에서 정의한 기본 용어들을 살펴보기로 한다.

① 서비스 프레임워크 : 서비스가 돌아가기 위하여 소프트웨어들이 돌아가는 틀이다. 소프트웨어의 묶음을 번들이라고 칭한다. 번들들은 언제든지 다른 곳에서 서비스를 받아 시작하고 끝나고 제거될 수 있다.

② 번들 : 서비스 소프트웨어의 묶음이다. 프레임워크 안에서 동작한다.

③ 서비스 : 서비스 소프트웨어의 최소단위이며 일반적으로 번들의 형태로 구동된다.

3) OSGi의 주요특징

① OSGi 서비스 플랫폼은 네트워크 서비스를 위한 표준화되고, 서비스 중심의 구성모델과 컴퓨팅 실행환경을 정의하며, 이를 이용하면 좀 더

쉽고 편리하게, 안정적이고, 보안성이 있으며, 확장성 있는, 유연하고, 잘 설계된 관리 가능한 어플리케이션/서비스를 비용과 시간을 절약해서 개발하고 운영할 수 있다.

② 네트워크 기기에 OSGi 서비스 플랫폼을 사용하면 네트워크를 통한 소프트웨어 서비스의 기간을 조절하는 기능이 부가된다.

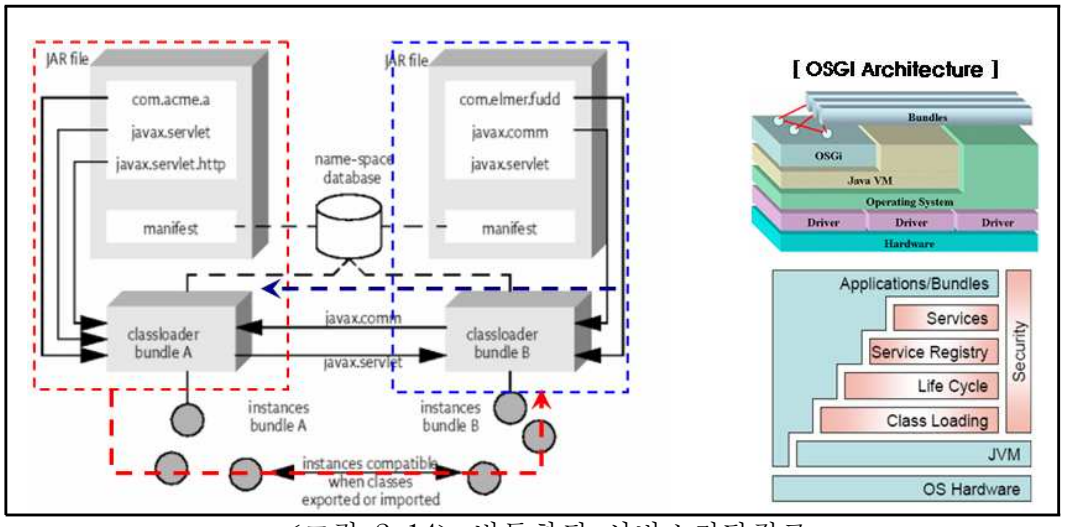
③ 소프트웨어 서비스는 특별한 기기 작동 없이 설치와 업데이트, 삭제가 가능하다.

④ 서비스지향의 구성모델인 OSGi 서비스 플랫폼의 특징은 네트워크상의 서비스가 다른 서비스와 함께 원하는 기능을 수행하는 것을 가능하게 한다.

4) OSGi 프레임워크 기술

① 프레임워크

OSGi에서 프레임워크는 번들들이 프로세스처럼 잘 돌아갈 수 있도록 해준다. 번들을 시작하고 종료하는 일, 업데이트 하는 일, 더 이상 필요 없는 번들을 제거하는 일 등, 번들을 관리하는 일을 프레임워크에서 한다. 또한 한가지 번들에서 제공하는 서비스를 필요로 하는 다른 번들이



<그림 2-14> 번들화된 서비스전달경로

<Fig. 2-14> Service routing path on bundled

그 서비스를 공유할 수 있도록 하는 일도 한다.

② 번들

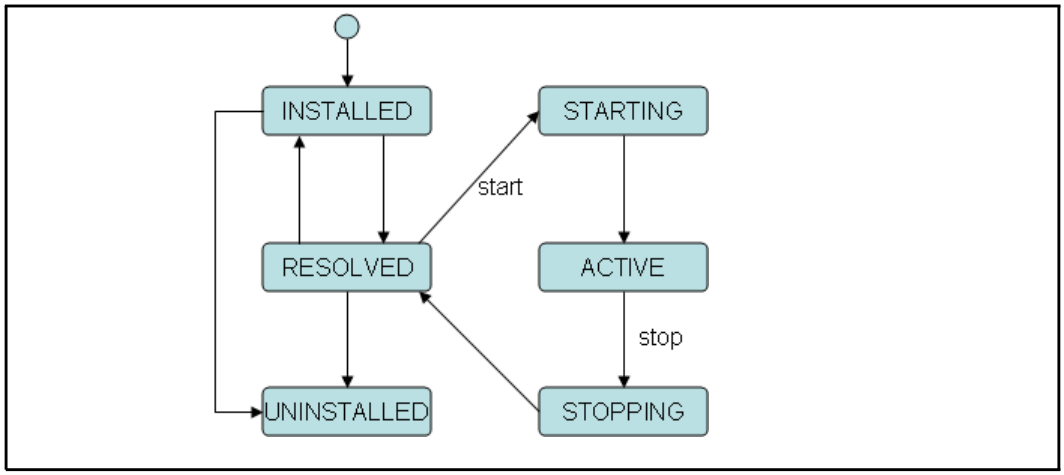
번들은 프레임워크에서 마치 프로세스처럼 돌아가는 서비스의 묶음이다. JAVA Archived File인 JAR(zip) 파일로 되어 있으며 <그림 2-14>와 같이 번들 안에 들어 있는 파일들의 정보를 갖고 있는 목록, 자바클래스, 리소스 등이 번들 안에 들어 있다. 공유 라이브러리의 DLL처럼 작동하며 프레임워크 위에서 <그림 2-15>와 같이 STD(State Transition Diagram)에서 상태를 바꾸어 가며 프로세스처럼 동작한다. 번들이 종료되면 서비스 레지스터리에 등록되어 있던 서비스는 사라지게 되며 그 서비스에 의존하던 다른 번들들에게 그 사실이 알려지게 된다.

번들이 시작되기 위해서는 'Resolved' 상태에 있어야 한다. 'Resolved' 상태로 되기 위해선 다음과 같은 세가지 의존관계가 문제없이 해결되어야 한다.

②-1. 클래스 경로 의존 : JAR 포맷으로 된 라이브러리로 가는 경로

②-2. 네이티브코드 의존 : 네이티브코드와 환경에 관한 의존

②-3. 패키지 의존 : expert(번들이 가능한 서비스를 공유가능하도록 내놓는 것)와 import(번들이 공유하고자 하는 필요로 하는 서비스를 찾는 것) 관계가 잘 맞아 떨어져야 함.



<그림 2-15> OSGi 번들 라이프사이클 상태전이도

<Fig 2-15> OSGi Bundle Lifecycle STD

③ 서비스

서비스는 번들이 다른 번들이 쓸 수 있도록 공유시켜 등록시킨 객체이다. 서비스 규격은 자바 인터페이스이다. OSGi 는 서비스의 표준세트를 정의한다.

④ 이벤트

이벤트에는 다음과 같은 세종류가 있다.

④-1. ServiceEvent

서비스의 등록, 제거, 속성 변화등의 일이 발생할 때 마다 나타나는 이벤트. ServiceListener에 의해 감지된다.

④-2. BundleEvent

번들의 상태 변화가 일어날 때마다 나타나는 이벤트.

BundleListener에 의해 감지된다.

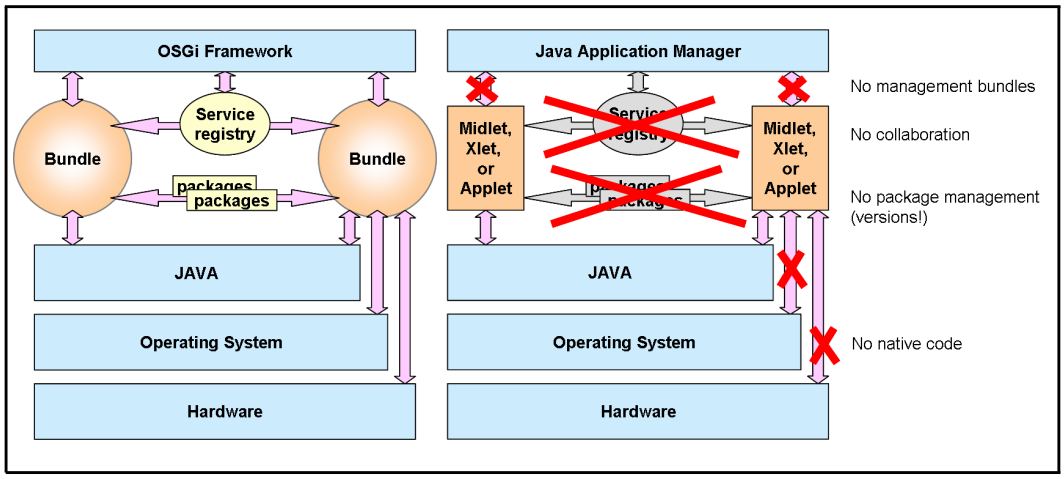
④-3. FrameworkEvent

프레임워크 자체가 시작되거나 에러가 발생할 때 일어나는 이벤트.

FrameworkListener에 의해 감지된다.

⑤ OSGi 프레임워크과 기본 자바 어프리케이션의 프레임워크의 비교

<그림 2-16>에서와 같이 OSGi와 기존 자바 프레임워크의 Applet, MIDlet, Xlet runner에 비하여 협업환경에서 관리가 용이하다[11,40,43].



<그림 2-16> OSGi 프레임워크과 자바 어프리케이션의 비교

<Fig 2-16> Comparison of OSGi framework and Java application

2.3.3 서비스 조립

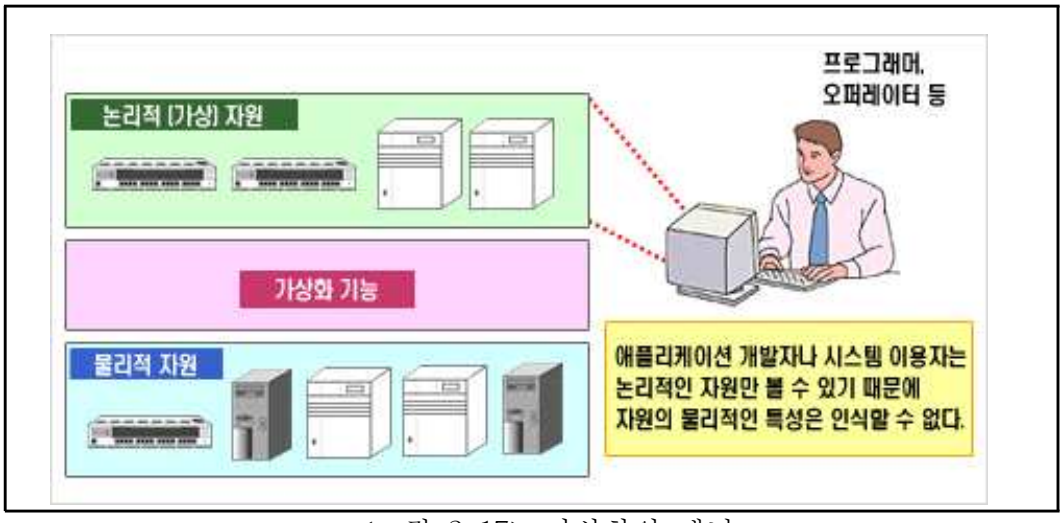
현재 대표적인 J2EE 오픈소스 개발 환경으로 Sun에서 오픈한 넷빈즈(NetBeans)와 IBM에서 오픈한 이클립스가 있으며, 본 논문에서는 이클립스를 선택하였는데, 이클립스의 비전은 '툴 중심'이 아닌 '플랫폼 중심'으로 사고하는 것이기 때문이다. 이클립스가 태양을 가려 어둡게 만든다는 의미를 가지고 있지만 본 논문에서는 생산성을 극대화하고 개인적인 스타일에 맞추어 변경하기 위해 유연성과 확장성, 이종의 소스로부터 구조화된 컴포넌트 통합성을 지원하는 것을 고려하여 설계된 이클립스 플랫폼을 제품 관리보다는 기술 혁신에 초점을 맞추었다. 즉, 이클립스에 새로운 툴을 만들 때, 자신이 그 '툴'을 어떻게 이클립스에 끼워 넣을 것인가가 아니라 이클립스 플랫폼에게 자신의 문제가 무엇인지를 어떻게 알려줄 것인가를 먼저 고민하고 이클립스에게 자신의 문제를 알려 줄 수 있는 방법은 잘 정의된 플러그인 포인트에 잘 걸릴 수 있는 툴 플러그인을 만드는 것이다. 결국 서비스 사용자는 이클립스에 추가된 새로운 툴을 보는 것이 아니라 매끄럽게 통합되어 이클립스 플랫폼에서 수행할 수 있게 된 새로운 기능을 보게 된다. 모든 이클립스의 기능은 플러그인에 의해 제공된다. 플러그인을 관리하는 이클립스 플랫폼 런타임은 이클립스의 시동을 다루고, 설치된 로컬 플러그인을 찾아서 각 설정 파일을 읽어 확장과 확장점을 일치시킨다. 메모리의 레지스터리는 모든 플러그인에서 활용 가능하다[13].

현재 수많은 조직들의 어플리케이션의 다수가 이클립스 RCP(Rich Client Platform)에 기반하고 있다. "리치클라이언트"라는 용어는 첫째로는 어플리케이션이 서비스 사용자에게 풍부한 경험을 제공하고, 둘째로는 어플리케이션이 몇몇 서버의 클라이언트임을 의미한다. 리치 클라이언트는 팻 클라이언트에 비해 많은 점에서 차이가 있는데, 팻 클라이언트가 배치 및 업데이트가 힘든 커다란 한 덩어리의 어플리케이션이 되려는 경향이 있는 반면, 리치 클라이언트는 좀 더 경량이며 그들 자신을 상대적으로 배치 및 업데이트에 용이한 컴포넌트 모델에 기반한다. 예전부터 팻 클라이언트는 특정한 플랫폼으로서 구축되어 왔는데, 오늘날의 리치 클라이언트 기술들은 아랫단에 위치한 플랫폼의 능력을 외부에 노출시키는 반면 플랫폼의 세부사항은 숨기는데, 이는 개발자로 하여금 어떤 특정한 플랫폼에 국한된 특수한 세부사항 보다는 주어진 과업에 집중할

수 있게끔 해준다. 이클립스 3.2 이상 RCP의 플랫폼런타임에는 강력함과 유연한 OSGi 규격을 따르는 컴포넌트 모델이 있다. 컴포넌트는 필요할 때 동적으로 발견되어 불러들여지며 그것들은 업데이트 및 확장 가능하다. 어플리케이션 코드가 여러 플러그인으로 분할되어 있다면 RCP 어플리케이션을 작성하는 것은 그러한 플러그인들을 연결된 하나의 완전한 어플리케이션으로 조립하는 과정이다. 따라서 이클립스 TDD(Test-Driven Development)와 이클립스 RCP에 따라 Flex로 리치클라이언트를 조립하며 자세한 내용은 2장 4절의 서비스지향 아키텍쳐에서 자세하게 설명하겠다[13,16].

2.3.4 서비스 가상화

<그림 2-17>과 같이 가상화는「통합화」와 함께 최근 정보 인프라에 관한 2대 테마가 되어 자원의 효율적인 활용, 전체의 최적화, 비용의 삭감 등의 이점이 있다. 그러나 통합화는 복수의 x86 서버를 한 대의 브레이드 서버에 집약하는 등, 물리적인 어프로치인데 비해 가상화는 물리적으로 다수 있는 것을 논리적으로는 하나로 보거나 물리적으로 하나인 것을 논리적으로는 복수로 보는 접근 방식이다. 따라서 가상화는 직접적으로 보이지 않는 기능에 의해 서비스 사용자가 물리적인 자원을 논리적인 자원으로 변환해 물리적인 제약으로부터 피할 수 있게 하고 보다 유연하게 IT자원을 이용할 수 있게 해주는 기술의 총칭



<그림 2-17> 가상화의 개념

<Fig. 2-17> Concept of Virtualization

이다.

이것으로 어플리케이션 개발자나 시스템 이용자에게서는 논리적인 자원만 보여 자원의 물리적인 특성은 보이지 않게 된다. IT세계에서 가상화는 일반적인 개념이지만 원래 IT의 역사는 가상화의 역사라고 해도 과언이 아니다. 현재 사람들은 프로그램을 작성하는데 물리적 메모리의 용량을 신경쓰지 않는다. 이것은 가상 메모리 기술에 의해 물리적 메모리의 제한에서 해방되었기 때문이며 네트워크에서도 TCP/IP등의 통신 프로토콜이 물리적 케이블이라는 물리적인 통신 수단을 숨겨 주기 때문이다[42].

가상화는 <표 2-2>와 같이 IT스택의 각 계층에 존재하며, 미들웨어도 물리적인 자원은 아니지만 어플리케이션과 데이터를 분할하고 어플리케이션으로부터 데이터로의 유연한 접속성을 제공한다는 의미에서 가상화적 메커니즘으로 생각할 수 있다. 미들웨어로의 가상화는 서비스 지향 아키텍처를 기초로 웹 서비스에 의해 SOAP나 XML(eXtensible Markup Language)을 이용하는 것으로써 어플리케이션으로부터의 네트워크나 데이터에 대한 가상적인 액세스를 제공한다. 또한 웹 어플리케이션의 경우도 웹 브라우저가 있으면, 클라이언트가 윈도우든 리눅스든 휴대 단말기든 웹 어플리케이션이 작동한다. 클라이언트의 물리 특성을 인식시키지 않는다는 점에서 어플리케이션의 가상화라 할 수 있는 것이다. 이러한 소프트웨어 가상화는 운영체제, 데스크톱 어플리케이션 등의 소프트웨어를 하드웨어로부터 분리하는 것으로 더욱 간편한 배치와 관리가 편리하다는 장점을 갖고 있다.

<표 2-2> 가상화의 계층

<Table 2-2> Stack of Virtualization

서비스	SaaS
데스크톱	SBC(Server Based Computing)
웹어플리케이션	웹브라우저
서비스 미들웨어	SOA
DBMS	클러스터, GRID
OS	VM(Virtual Machine), 하이퍼바이저
서버	블레이드 서버, 파티셔닝
스토리지	SAN
네트워크	IP-VPN(Internet Protocol - Virtual Private Network), TCP/IP

가상화는 어플리케이션의 요구에 대해 다양한 목적을 가진다.

① 성능: 자원들을 하나의 풀로 만들어 최대한 컴퓨팅 파워를 사용하고자 성능을 향상시킴

② 확장성: 서비스 요구량 및 트래픽의 폭발적인 증가에도 유연하게 대처할 수 있는 서비스의 확장성

③ 가용성, 신뢰성: 하루 24시간 1년 365일 동안 중단 없이 서비스를 제공할 수 있는 데이터 및 서비스의 가용성

④ 탄력성, 민첩성: 긴급한 문제 발생 시에도 빠르게 대처하여 실시간으로 장애 복구가 가능한 서비스 인프라의 탄력성과 어떠한 상황 변화에도 빠르게 대처할 수 있는 서비스 인프라의 민첩성

⑤ 자원 최적화: 물리적인 시스템의 재구성 없이도 다양한 비즈니스 요구에 맞는 자원의 활용을 최적화하게 구성

서비스의 가상화는 서비스 사용자가 SaaS(Software as a Service)에 의해 많은 물리 자원을 사외 서비스의 제공자로부터 얻는 방식이다. 서비스 사용자는 서버나 스토리지, 데이터기반이나 어플리케이션을 보유하고 있지 않아도 네트워크에 접속할 수 있는 클라이언트만으로 어플리케이션을 이용할 수 있다.

컴퓨팅 환경이 서비스 중심으로 발전하면서 서비스 사용자들은 표면에 들어난 가상화된 서비스만을 호출하여 서비스를 받고자하고 있다. 이를 가능하게 하기 위해서 서비스 가상화는 스토리지, 서버, 네트워크 가상화 기술과 복합적인 모델을 지원하여 서비스 중심 컴퓨팅 동작이 가능하게 구현되어야 한다. 이를 지원하기 위한 환경으로 적합하게 적용되는 기술이 웹 서비스이며 이질적인 시스템들간에 표준화란 연결고리를 사용하여 흩어진 자원들을 사용할 수 있도록 한다. 기존의 웹 접속이 사람에 의해 접속되어 사용된 것이라면 웹 서비스는 가상화된 서비스가 웹에 접속하여 정보를 요청하고 요청된 서비스는 관련되는 서비스들에 의해 요청된 정보의 응답을 받게 된다. 즉, 웹 서비스는 인터넷상에서 서로 통신할 수 있는 서비스를 지칭한다. 웹 서비스의 기술은 서비스 지향 구조를 기반으로 서비스 단위의 통합을 이루어 유연하고, 능동적인 웹 어플리케이션 수행 환경을 제공하는 기술이다. 웹 서비스는 가상화된 서비스를 전달하는 주요 요소라고 할 수 있으며 가상화된 서비스는 서로 이질적인 시스템 자원들을 투명성 있게 사용할 수 있도록 한다. 서비스 가상화를 이루기 위

해서는 웹 서비스를 활용하여 흩어진 서비스들을 구성하고 활용할 수 있도록 해야 할 뿐만 아니라 서비스들을 배치하고 관리하는 작업들이 이루어져야 한다. 따라서 서비스 가상화를 위해 서비스 지향 구조는 임의의 서비스 사용자가 임의의 플랫폼을 기반으로 하는 서비스 제공자에게 접근할 수 있는 기본적 아키텍처 패러다임을 전제하고 있다. 이는 서비스 사용자와 서비스 제공자를 절연할 수 있는 프로토콜과 중재 방안이 존재함을 의미한다.

서비스지향구조의 "고려사항의 분리" 원칙에 따라 시스템 레벨의 "Syntactic" 변환은 ESB에서 수행하고, 어플리케이션 레벨의 "Semantic" 변환은 BPM(Business Process Management)에서 수행한다. ESB를 SOA 환경에 도입하려 했던 애초의 목적은 바로 IT 인프라스트럭처를 가상화하고 비즈니스 고려사항을 IT 고려사항으로부터 분리하는 것이므로 ESB에 불필요한 로직을 구현하지 말고 프로세스 통합의 역할은 BPM 툴이 전담한다.

ESB를 이용하여 IP 주소, 포트, 기타 로우 레벨 머신 정보들을 비즈니스 로직으로부터 분리하고, 단순하고 안정적인 인터페이스를 제공할 수 있다. 그 결과로 비즈니스 고려 사항과 IT 고려 사항을 완벽하게 분리하고, IT/비즈니스 자산을 각각 독립적으로 관리할 수 있다.

또한 ESB를 이용하여 단일 API를 통해서만 접근 가능한 서비스에 새로운 채널을 추가하고자 하는 경우, 간단한 설정 작업만을 거치면 되고 동일한 아키텍처를 이용하여 멀티-채널 환경을 지원하기 때문에, 어떤 채널을 사용하든 관계없이 모든 액세스에 대해 동일한 글로벌 정책을 적용할 수 있다.

이러한 서비스 가상화는 로컬 투명성의 측면에서도 이점이 있다. 프록시를 공개함으로써, 서비스 사용자에게 영향을 주지 않은 상태에서 하부 인프라스트럭처를 변경하는 것이 가능하기 때문이다. 중간에 위치한 프록시 서버는 가상 서비스 엔드포인트 매핑을 직접 관리하거나, UDDI(Universal Description Discovery and Integration) 레지스터리를 통해 서비스 엔드포인트를 조회하게 된다[42].

2.3.5 서비스 평가

부록의 “ATAM 평가방법론의 개요” 참조.

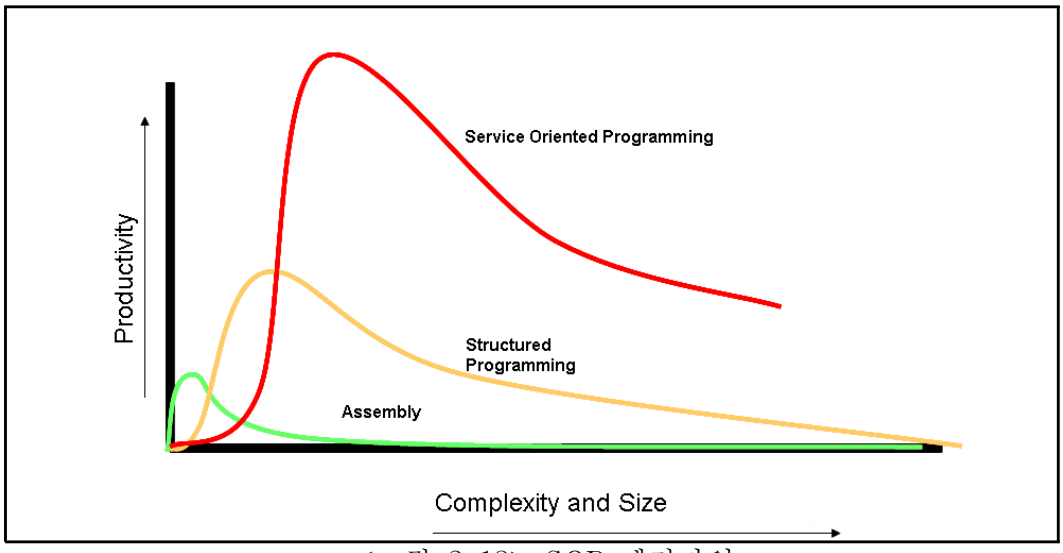
2.4 서비스지향 아키텍쳐

2.4.1 엔터프라이즈 서비스지향 아키텍쳐

(1) 서비스지향 아키텍쳐 개요

현재 <그림 2-18>과 같이 Assembly, C와 같은 절차지향형 프로그래밍에서, 구조적프로그래밍, 객체지향프로그래밍을 넘어 SOP(Service Oriented Programming)과 같은 패러다임을 겪고 있다. 이에 따라 서로 연동되기 힘든 다양한 플랫폼과 기술을 기반으로 구축되어 있으므로 서비스지향 아키텍쳐는 이렇게 다양한 컴퓨팅시스템 위에 구현된 어플리케이션과 같은 리소스계층의 여러 자원들을 효율적으로 통합시켜, 기능의 재활용이나, 배치의 유연성을 극대화하는 것을 목적으로 한다. 따라서, 객체지향 기술이 어플리케이션 내에서 기능을 효과적으로 모듈화하고, 재활용하는 것에 초점이 맞춰져 있는 반면, 서비스지향은 각 어플리케이션의 구현이 아니라, 이미 구현된 어플리케이션 간의 상호연동과 재활용에 초점을 맞춘다[29,41].

"서비스"란, "관심의 분리"라는 서비스지향원칙에 따라 어플리케이션의 기능 구현이 어떤 네트워크, 하드웨어, OS, 미들웨어, 프로그래밍 언어, 객체 모델에 기반하고 있던 관계없이 일관된 방법으로 종단간의 그 기능을 사용할 수 있도

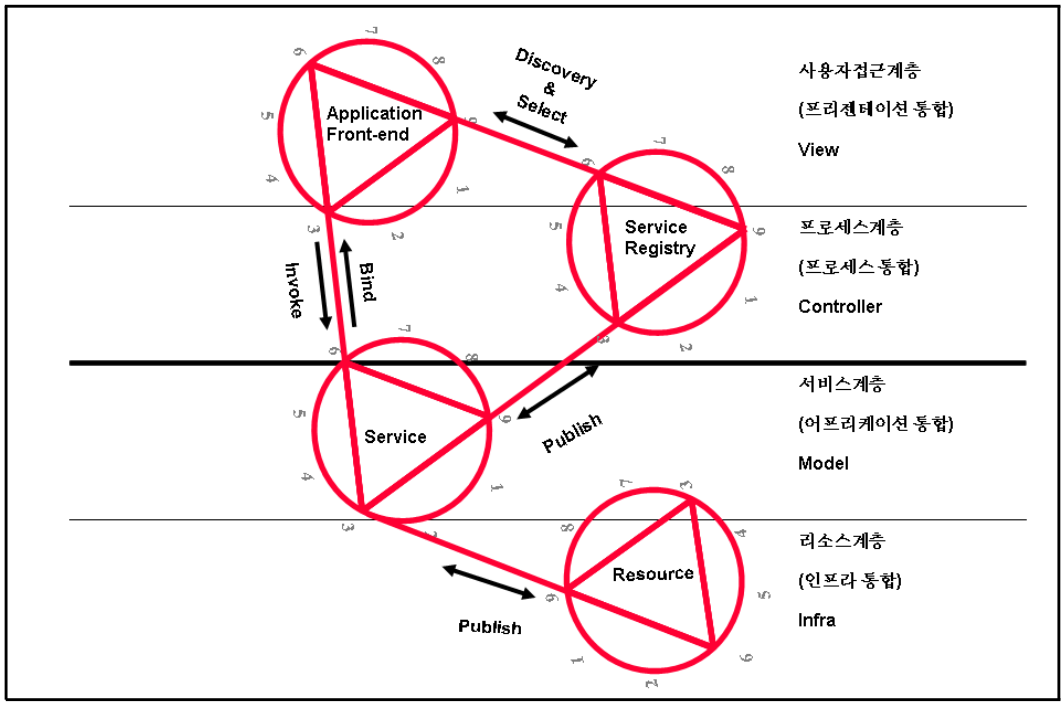


<그림 2-18> SOP 패러다임

<Fig. 2-18> SOP paradigm

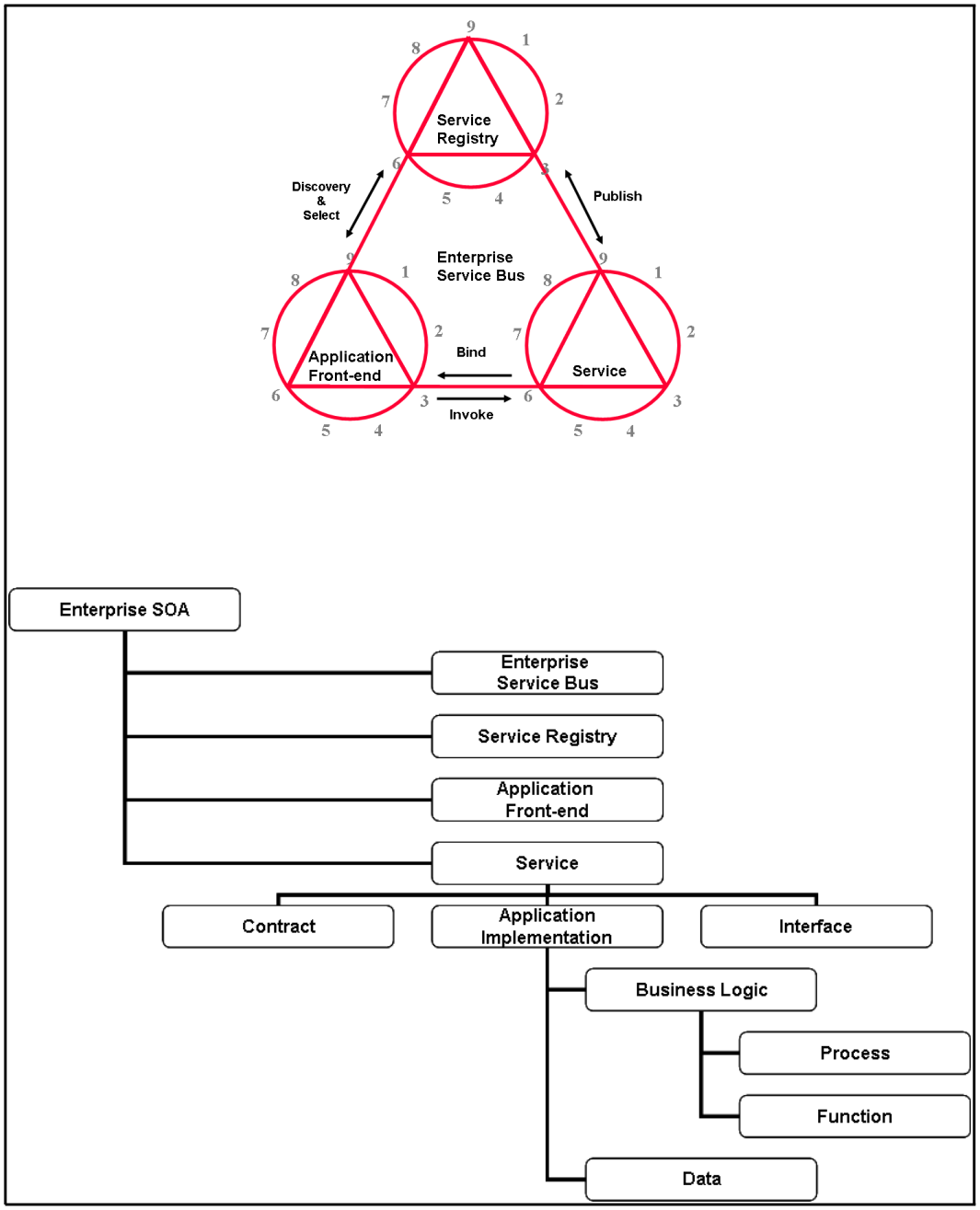
록 포장한 것이다. 또한 서비스의 활용을 중개할 수 있는 인프라스트럭쳐로서 서비스 레지스터리와 ESB를 도입하여, 프로세스 요건을 기능으로부터 분리하며 서비스의 조합으로 관리하여, 프로세스 요건 변화에 신속하게 대처할 뿐 아니라, 조직 간 프로세스가 융통성있게 상호운영하는 프로세스 수준의 통합이 가능하다. 서비스화된 어플리케이션 기능은 서비스 사용자와 인터페이스하는 여러 채널에 걸쳐, 채널의 종류에 관계없이 일관성있게 활용될 수 있어, 서비스 사용자 수준의 통합을 지원할 수 있다[14,15,41,21].

서비스지향 아키텍쳐의 개념 자체가 여러 계층을 포함하고 있어, 정의하기 힘들지만, 굳이 한마디로 정의하자면, "다양한 어플리케이션 기능을 균질한 서비스로 포장하여, 기능의 내부 구현에 영향받지 않고 활용할 수 있게 하고, 어플리케이션, 프로세스 그리고 서비스 사용자 수준의 통합을 가능하게 하는 인프라스트럭쳐를 구축하는 것"이라고 할 수 있다[14,15,29,41].



<그림 2-19> 엔터프라이즈 서비스지향 아키텍쳐의 수직적 구조

<Fig. 2-19> Vertical structure of enterprise SOA



<그림 2-20> 엔터프라이즈 서비스지향 아키텍처의 수평적 구조

<Fig. 2-20> Horizontal structure of Enterprise SOA

(2) 서비스 계층과 리소스계층

서비스지향 아키텍쳐에서 서비스의 의미는 전통적인 서비스지향 아키텍쳐구현과 새롭게 등장하고 있는 최신 서비스지향 아키텍쳐구현 사이에 약간 차이가 있다. 기존의 서비스가 기능의 내부 구현에 독립적이지만, 여전히 플랫폼에는 종속적이었다면, 웹 서비스라는 기술이 등장한 이후에 부각된 새로운 서비스 개념은 플랫폼에 조차 독립적일 것을 요구받고 있다.

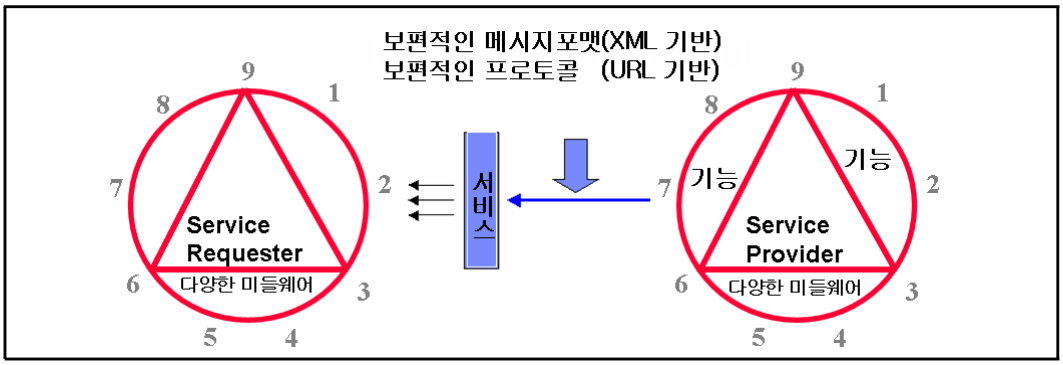
서비스지향 아키텍쳐의 바탕을 이루는 기본 사상는 "다양한 시스템에 분산되어 있는 어플리케이션끼리 직접 호출하는 대신, 서비스(인터페이스)를 통해 간접적으로 호출하게 하여, 호출되는 기능의 실제 위치와 내부 구현에 영향을 받지 않도록 하는 것" 이다.

서비스지향 아키텍쳐에서 "서비스"란 어떤 기능을

① 사용할 수 있도록 하는 수단인 동시에, 그 기능의 내부 구현이 그 기능을 사용하는데,

② 영향을 미치지 않도록 하는 보호막의 역할을 한다. 즉 필요한 기능이 동일한 시스템에 있든지 원격지의 다른 시스템에 있든지 상관없이(위치 투명성), 그리고 그 내부 구현이 어떻게 바뀌든지 상관없이, 사전에 약속된 인터페이스가 변경되지 않는 한, 수정할 필요가 없도록 한다.

이렇게 <그림 2-21>과 같이 서비스를 통해 기능을 활용하는 어플리케이션이나 시스템은 서비스 사용자(Service Requester)라 하고, 서비스를 통해 기능을 제공하는 어플리케이션이나 시스템은 서비스 제공자(Service Provider)라 부른



<그림 2-21> 서비스지향 아키텍쳐에서 서비스전달

<Fig. 2-21> Service delivery in SOA

다. 이러한 개념을 구현하기 위해서는, 서비스 사용자를 대신하여, 서비스 제공자의 위치를 찾고 기능 호출을 대행해주는 별도의 미들웨어가 필요하다.

여기에서 서비스의 구현이 가지는 두 가지 특징을 알 수 있다.

① 서비스는 명확하고 내부로직과 무관한 인터페이스를 사용하여 정의된다. 서비스를 이용하는 쪽은 서비스의 내부는 관심 없지만, 어떻게 해야 원하는 기능을 수행시키고 원하는 결과를 얻을 수 있는지 알 필요가 있다. 따라서, 서비스의 사용방법을 명확하게 기술하여, 서비스를 제공하는 쪽과 이용하는 쪽 모두에서 이를 공유한다.

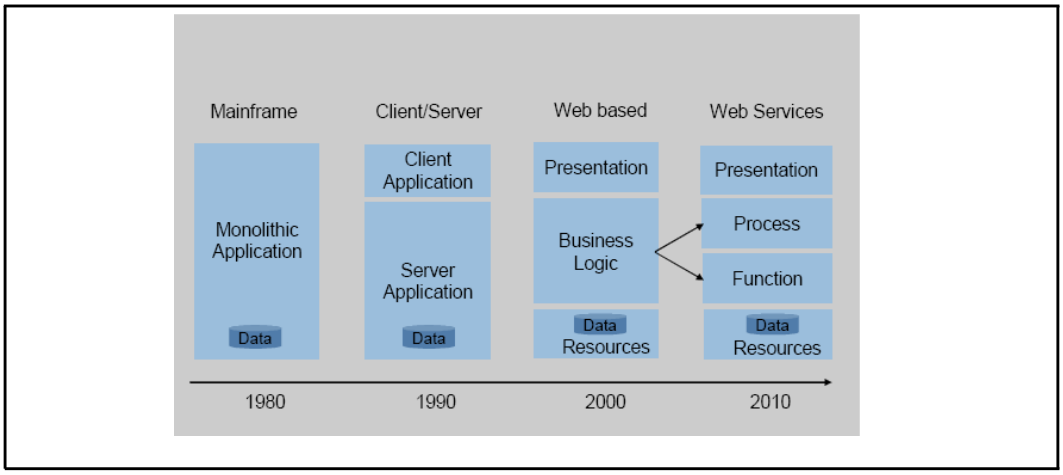
② 서비스는 위치투명성과 상호연동을 보장하는 통신 프로토콜을 통해 느슨하게 결합한다.

주고 받는 메시지의 포맷과 프로토콜을 통일하여 플랫폼으로부터 독립한 포맷과 프로토콜은 다양한 언어와 API(Application Programming Interface)에서 지원되는 형태여야 하고, 객체모델과는 무관하여야 한다. 무엇보다도, 서비스지향 아키텍쳐와 연관된 모든 이해당사자가 지원을 동의해야 하는데 "웹 서비스"가 그 해결책으로 이야기되고 있는 최신 서비스지향 아키텍쳐의 하나이다.

웹 서비스의 등장과 함께, 서비스지향 아키텍쳐에서 서비스의 의미는 플랫폼 독립성까지 포함하여 다음과 같이 확장할 수 있다. "기능의 위치, 내부 구현 뿐 아니라 사용하는 플랫폼에 영향을 받지 않고 일관성 있는 방법으로 기능을 사용할 수 있도록 제공되는 수단"

(3) 프로세스계층

<그림 2-22>와 같이 전형적인 비즈니스 프로세스는 어플리케이션의 코드 속에 숨겨지게 되어 비즈니스 요건의 변화 나아가 시장 환경의 변화에 엔터프라이즈 환경 컴퓨팅시스템이 따라가지 못하는 문제가 생긴다. 이에 대한 해결책은 프로세스를 기능으로부터 분리하여, 구현하는 것이다. 변화가 잦은 프로세스와 한 번 작성하면 꾸준히 사용할 수 있는 기능을 분리하여, 기능의 수정없이 프로세스만을 변경할 수 있도록 하는 것이다. 즉 모든 기능을 서비스화하고, 이 서비스를 조합하는 방법으로 비즈니스 프로세스 요건을 구현하는 것이다. 이것은 엔터프라이즈 환경 내 혹은 엔터프라이즈 환경 간의 비즈니스 프로세스를 유연하게 관리하고 통합하는 것을 가능하게 한다.



<그림 2-22> 프로세스와 기능의 분화과정

<Fig 2-22> Seperation of process and function

변화가 잦은 비즈니스 프로세스와 일단 만들어지면 상대적으로 오래 사용할 수 있는 기능을 분리하여, 비즈니스 요건이 바뀔 때 마다 비즈니스 프로세스만을 변경하는 것이다. 데이터(M)-표현(V)-비즈니스 로직(C)의 3-tier로 나누어 서로 별도로 변경될 수 있도록 한 것에서 더 나아서 비즈니스 로직을 프로세스와 기능을 분화시킨 것이다.

이렇게 프로세스와 기능을 분화시키기 위해서는 비즈니스 로직을 구성하는 기능들을 균질하고 독립적인 단위로 만들 필요가 있다. 그 내부 기능에 관계없이 일관되고 표준화된 방법으로 사용할 수 있어야 하고, 그 각각은 다른 기능과 어떻게 조합되든 관계없이 사용할 수 있어야 하므로 임의의 순서와 규칙으로 기능들을 쉽게 조합할 수 있기 때문이다.

프로세스와 기능을 분화시키면 다음과 같은 잇점이 있다.

① 기능의 변화없이 프로세스 규칙의 변경만으로 쉽게 비즈니스 프로세스를 변경할 수 있다.

② 프로세스가 코드에 숨어있을 필요가 없이, 정형화된 방식으로 표현하고, 다이어그램으로 시각화할 수 있다. 즉 현재의 프로세스를 현업 서비스 사용자도 쉽게 파악하고 그 개선에 참여할 수 있다.

③ 엔터프라이즈 환경 내 그리고 엔터프라이즈 환경 간 시스템 사이의 비즈니스 프로세스를 필요에 따라 쉽게 연결하고 관리할 수 있다.

1) 서비스 레지스터리

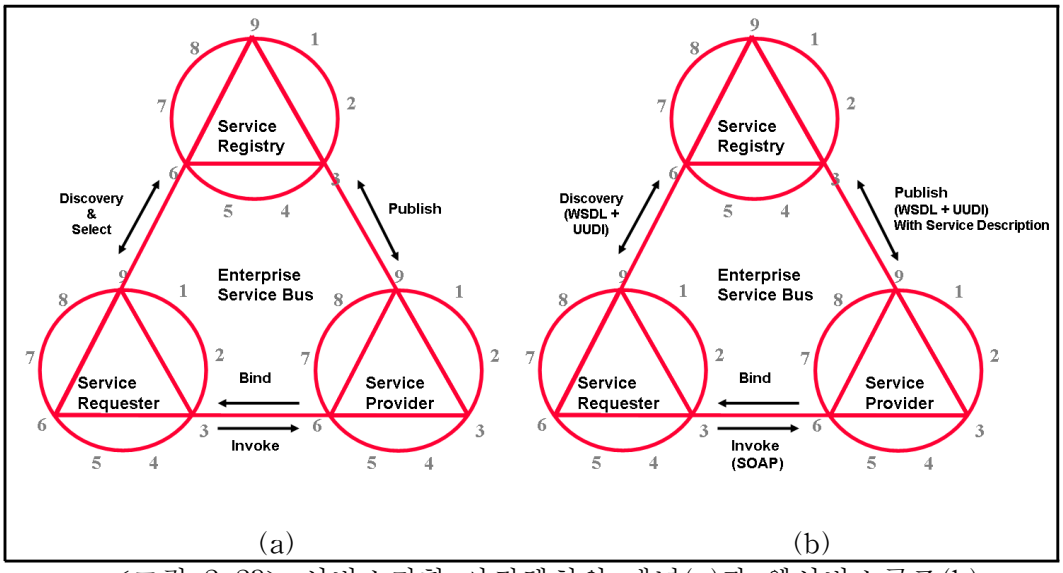
<그림 2-23>과 같이 서비스 제공자와 제공되는 서비스가 여럿일 경우에, 서비스들에 대한 정보를 한 곳에 모아 서비스 사용자를 구현할 때 참조할 수 있도록 할 필요가 있을 때 서비스 정보를 취합하여 관리하는 것이 서비스 레지스터리(Service Registry)이다. 현실적으로 서비스의 종류와 그 제공자의 수가 많고 점점 늘어나므로, 서비스 레지스터리가 필요하다.

서비스지향 아키텍쳐에서 서비스 레지스터리는 단순히 서비스 자체에 대한 정보, 서비스의 품질이나, 제공 정책 등 서비스 사용자가 어떤 서비스를 사용할 것인지 결정하기 위해 필요한 모든 정보를 담고 있어야 하며 서비스 제공자와 서비스 사용자 사이의 협상을 위한 표준을 제공할 필요가 있다.

그리고 서비스 제공자와 서비스 사용자는 서비스 레지스터리를 거쳐 다음과 같은 절차를 거쳐 서로를 발견하고 서로 연동한다.

서비스 레지스터리에는 서비스를 설명하는 서비스 설명서가 저장되며, 서비스를 이용하는 절차는 공개-검색-연결의 순서를 따른다.

① 서비스 제공자가 자신이 제공하는 서비스를 설명하는 서비스 설명서를 서비스 레지스터리로 공개한다.



<그림 2-23> 서비스지향 아키텍쳐의 개념(a)과 웹서비스구조(b)

<Fig. 2-23> Concept of SOA(a) and webservice Structure(b)

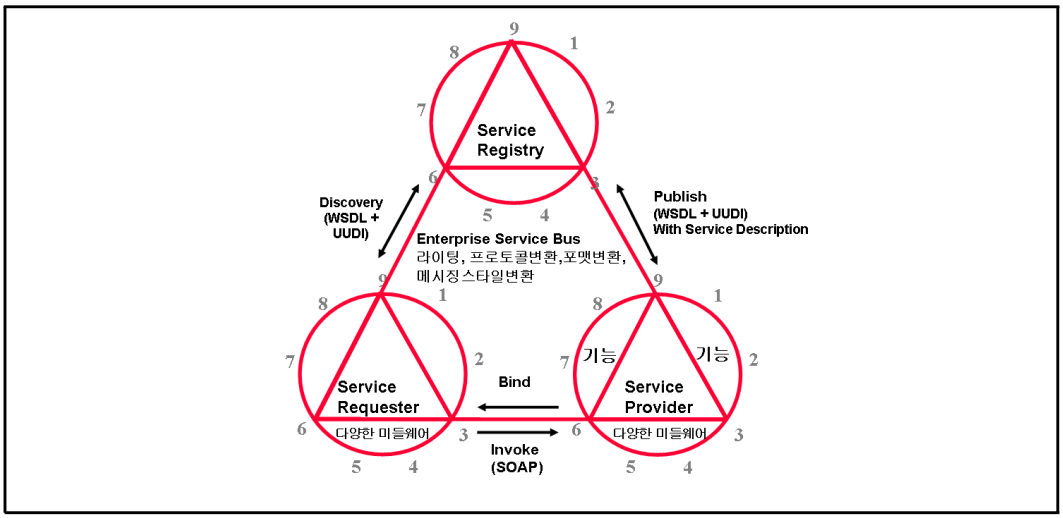
② 서비스 사용자는 원하는 서비스를 검색한다.

③ 검색된 서비스 설명서을 사용하여, 원하는 서비스 제공자에 연결하여 서로의 정책을 협상한 후, 서비스를 요청한다.

2) ESB

<그림 2-24>와 같이 서비스지향 아키텍쳐의 핵심 요소로서 ESB가 거론되고 있다. 서비스 사용자, 서비스 제공자와 서비스의 수가 점점 늘어나면, 그 연결의 수는 기하급수적으로 늘어나게 된다. 이것은 한 부서나 한 엔터프라이즈 환경 내에서 운영하는 시스템에서는 관리상의 문제가 될 수 있다. 이 문제를 해결하기 위해서는, 서비스의 사용을 중개하는 ESB를 두고, 모든 서비스 사용을 이를 통해서 이루지게 할 필요가 있다. 이렇게 하여, 서비스 연결의 수를 획기적으로 줄이고, 서비스 사용을 한 곳에 관리할 수 있다.

또한 RTE(Real Time Enterprise)환경에서 ESB를 사용하여 서비스 사용을 한 곳에서 관리하면 하나의 엔터프라이즈 환경의 프로세스를 기능과 분리하여 BPM으로 관리하거나, 엔터프라이즈 환경 프로세스의 수행 현황을 BAM (Business Activity Monitoring)으로 모니터링하는 시스템을 보다 용이하게 구축할 수 있다. ESB는 서비스 연결을 단순화할 뿐 아니라, 추가적인 유연성을



<그림 2-24> 서비스지향 아키텍쳐에서 ESB

<Fig. 2-24> ESB in SOA

제공해 줄 수있다. 서비스 제공자나 사용자가 모두 별도의 조직에 의해 관리되는 자율적인 당사자인 환경을 제외하고는, ESB의 개념은 관리적인 측면이나 그 유연성 면에서 매우 유용하다.

(4) 사용자 접근계층

컴퓨팅시스템에는 어플리케이션 간의 상호연동도 존재하지만, 서비스 사용자를 위한 인터페이스 역시 멀티채널로 제공한다. 서비스 사용자에게 인터페이스를 제공하는 시스템이 직접 비즈니스 로직을 가지는 대신, 서비스 사용자로서 비즈니스 프로세스에 참여하고, 프리젠테이션 로직만을 가짐으로써 보다 유연성있게 구현할 수 있다.

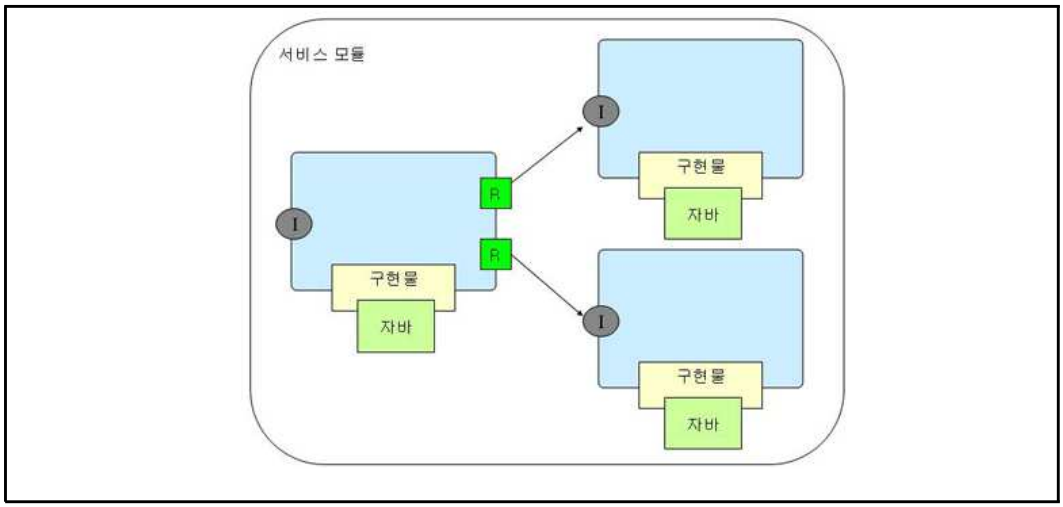
즉, Flex와 같은 X-interent 기술을 이용하여 웹, 리치 클라이언트나 모바일과 같은 서로 다른 사용자 인터페이스를 제공하는 시스템이 별도의 비즈니스 로직을 구현하는 대신, 서비스 형태로 비즈니스 로직을 활용할 수 있어, 재활용성을 높일 수 있다. 또한 비즈니스 프로세스의 변경에 대해 보다 영향을 적게 받아, 변화에 빨리 대응할 수 있다. 이것은 다양한 종류의 서비스 사용자들이 비즈니스 프로세스에 효과적으로 참여할 수 있는 기반이 된다.

2.4.2 이클립스 STP

(1) SCA

이클립스 STP(Service oriented architecture Tool Platform)에서 공식적으로 이 SCA(Service Component Architecture)가 서비스지향 아키텍쳐 기반의 개발을 위한 기본 플랫폼으로 채택되었으며 이러한 SCA는 서비스지향 아키텍쳐 환경에서 자신이 원하는 비즈니스 서비스를 쉽게 개발하기 위한 새로운 차원의 프로그래밍 모델이다. 이 표준은 2005년 11월 30일에 IBM과 BEA 그리고 오라클 등의 수많은 소프트웨어 벤더들의 주도로 발표되었다[25]. SCA의 주된 용도는 서비스지향 아키텍쳐 세계에서 어플리케이션과 복합 어플리케이션 개발에 따르는 어려움을 해결하기 위해 특정 프로그래밍 언어나 미들웨어의 API에 연연하지 않고 그러한 어플리케이션들을 묶어서 서비스지향 아키텍쳐 환경에서 실행하고 관리할 수 있는 환경을 제공하고자 한다[42,43].

서비스지향 아키텍쳐 환경에서 엔터프라이즈 환경 내의 이기종 서비스들을 연계하여 유연한 서비스 컴포넌트를 개발하는 과정은 다음과 같다. 엔터프라이즈 환경 내에는 다양한 컴포넌트와 프로토콜 그리고 프로세스들이 존재하며 이러한 여러 서비스들을 탑다운 방식으로 조직화하는 일련의 과정이 표준화되어 있지 않다면 유연한 엔터프라이즈 환경 환경을 기대할 수 없을 것이다. 즉, 엔

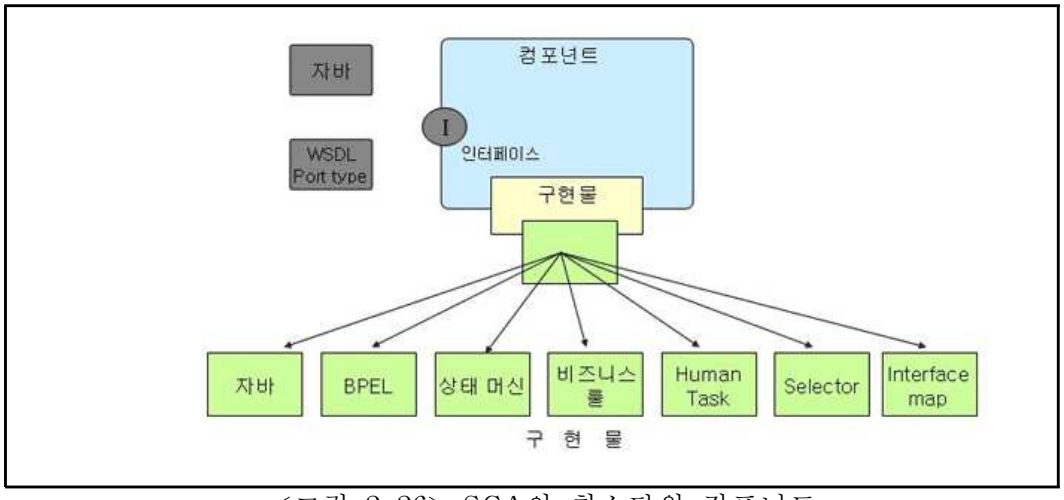


<그림 2-25> SCA의 서비스 모듈

<Fig. 2-25> Service module of SCA

터프라이즈 환경 내의 모든 IT 리소스들을 그 구현 형식과 내용에 관계없이 통일된 방식으로 표현하여 그들 간의 어플리케이션 조립을 위한 런타임 인프라스트럭처를 제공하는 것이 SCA의 목적이다. 따라서 SCA에서 가장 최소한의 단위는 서비스 컴포넌트로서. 이는 비즈니스 프로세스나 비즈니스 룰, 자바 구현물이나 그외 다양한 구현물에 대한 추상화된 컴포넌트라고 할 수 있으며, <그림 2-25>과 같이 이러한 서비스 컴포넌트가 묶여서 서비스 모듈이라는 단위가 완성된 모습이다.

WSDL(Web Service Description Language)이 어플리케이션들 간의 상호운영성을 높여주는 기술이나 WSDL은 오직 단일 서비스의 인터페이스에만 초점을 맞추고 있어서 이 서비스가 다른 서비스들 간에 어떠한 정책과 의존성으로 관계를 맺고 있는지에 대한 것을 보여주는 데에는 한계가 있다. SCA가 이러한 WSDL의 한계를 넘어 여러 서비스들을 엮어서 비즈니스 컴포넌트로 정의할 수 있는 모델을 만들고자 하는 데에 있다. 이들 또한 서비스 컴포넌트라 부르며 서비스 개발자는 SCA의 서비스 컴포넌트를 통해 서비스에 대한 인터페이스 설계뿐만 아니라 서로 다른 서비스들 간의 관계와 의존성 등에 대한 설정을 쉽게 해낼 수 있게 된다. 이는 개발자의 서비스지향 아키텍쳐 환경에 편리함을 주는 것으로서 특정 벤더의 독점적인 방식으로 각각의 서비스들에 대한 관계를 기술하거나 그러한 내용을 서비스 구현 자체에 포함시키지 않아도 표준적인 방



<그림 2-26> SCA의 최소단위 컴포넌트

<Fig 2-26> Unit Component of SCA

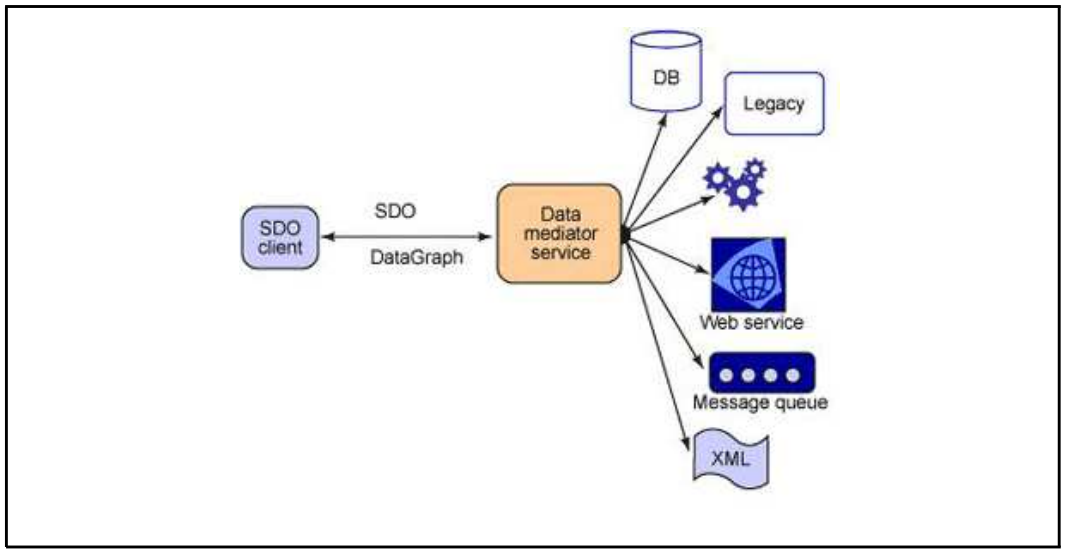
식으로 관리하는 것이 가능해졌기 때문이다.

SCA의 기본 구조는 <그림 2-26>와 같다. SCA의 최소 단위는 서비스 컴포넌트이다. 이 서비스 컴포넌트는 자바뿐만 아니라 다양한 구현물을 추상화하여 모두 동일한 서비스의 형태로 다룰 수 있게 한다. 따라서 자신이 할 일은 서비스로 만들 어떠한 구현체에 대하여 인터페이스를 정의하는 것뿐이다.

SCA가 제공하는 또 다른 기능은 이렇게 만들어진 서비스 컴포넌트들 간의 관계를 설정하는 것이 가능하다는 것이다. 이것은 WSDL이 제공할 수 없었던 서비스들 간의 정책과 관계를 설정함에 있어 복잡한 WS-*(Web Service-*) 표준을 알지 못한다 할지라도 매우 쉽게 가능하게 한다는 점에서 개발의 편리성을 증대시켜주는 결과를 가져온다.

(2) SDO

SCA를 통해서 가시화된 컴포넌트들 간의 데이터 교환은 또 다른 추상화된 데이터 계층을 필요로 하다. 즉, 각 서비스 컴포넌트들은 상대 컴포넌트가 내부적으로 어떠한 형태의 데이터 포맷을 필요로 하는지를 상관하지 않아야 하며 오로지 순수하게 모든 데이터를 데이터 자체로만 인식할 수 있는 표준이 바로



<그림 2-27> SDO의 구조

<Fig 2-27> Structure of SDO

SDO(Service Data Object)이다. SDO는 IBM과 BEA가 공동으로 참여한 JSR 235 표준으로서 그 뿌리는 CommonJ라는 구현물에 있다. SDO는 IT 업계에서 점차 그 중요함이 절실해지고 있는 서로 다른 데이터 소스에 대한 통합되고 단순화된 제어를 목적으로 만들어진 API이다. 데이터 소스가 DB이건 웹 서비스이건 LDAP이건 상관없이 단일화된 인터페이스를 통해서 개발 생산성을 높이자는 데 그 목적이 있다. SDO는 모든 데이터에 대하여 표준화된 구조와 객체 형태를 제공한다. 더욱이 수많은 데이터 소스와 서비스로 부터 표준화된 접근 방식을 제공한다. 이는 정보의 소스로부터 비즈니스 프로세스를 분리하여 데이터 중심적인 제어를 가능하게 한다.

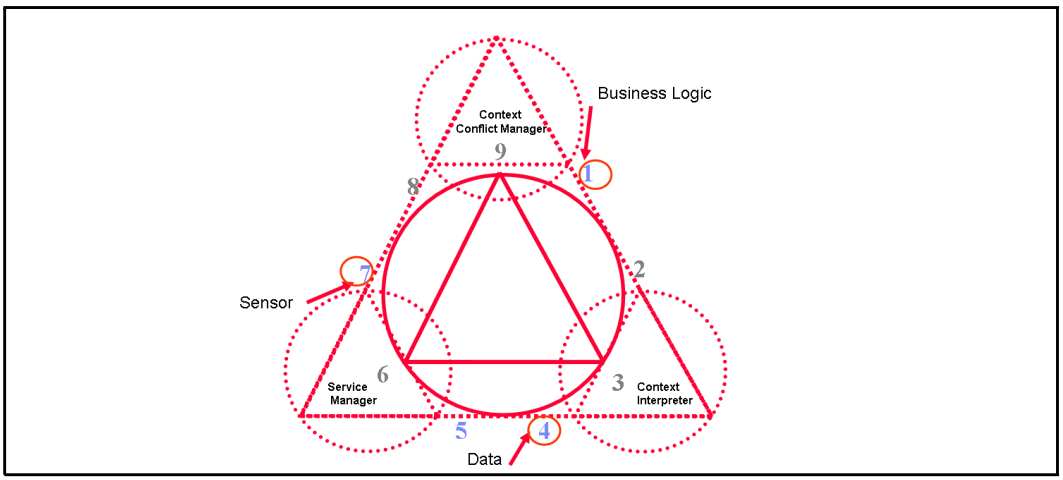
SDO가 만들어지게 된 가장 주된 목적은 단순한 데이터 제어에 있다. EIS(Enterprise Information Systems)의 다양한 데이터 제어 기술을 통합하자는 데 그 첫 번째 목적이 있는 것이다. 따라서 SDO를 사용하게 되면 데이터 소스로부터 독립적인 데이터 표현 인터페이스를 구축할 수 있게 된다. 이러한 SDO의 설계 이면에는 Domain Store라 불리는 자바 EE 패턴이 자리 잡고 있다. 이 패턴을 통해 SDO는 데이터 제어를 쉽게 하고 서로 다른 계층 간에 느슨한 연결을 보장하는 등의 여러 이점을 제공한다. 특히 느슨한 연결이라는 특징은 SDO의 가장 장점 중의 하나이다.

SDO의 구조는 <그림 2-27>과 같이 크게 보면 데이터 객체, 데이터 그래프 그리고 DMS(Data Mediator Service)로 이루어져 있다. 현재 이 SDO는 BEA와 IBM 등의 대형 벤더의 ESB 제품에서 핵심적으로 사용되고 있는 기본 데이터 모델이다. IBM의 경우 서비스지향 아키텍쳐 핵심으로서 이 SDO가 거의 모든 컴포넌트들 간의 정보 교환에 사용되고 있으며, BEA의 경우도 아쿠아로직 제품 중 데이터 서비스 제품에서 이 SDO가 핵심 데이터 처리 컴포넌트로 사용되고 있다. 또한 이클립스 EMF(Eclipse Modeling Framework) 프로젝트에서는 이미 오래 전부터 SDO의 구현물을 사용하여 이클립스 기반 프레임워크의 중요한 데이터 처리 컴포넌트로 사용 중에 있다. 따라서 아직 이 SDO가 산업 전반에 널리 사용되고 있는 것은 아니지만 서비스지향 아키텍쳐 환경이 점차 일반화되어 감에 따라 자바 기반의 미들웨어에서 제공하는 서비스 컴포넌트들 간에 메시지의 자유로운 교환과 변환에 가장 널리 사용될 기술이 될 것으로 보인다[42,43].

제 3 장 서비스지향 아키텍쳐의 설계

3.1 서비스지향 아키텍쳐의 분석

집이란 가족이라는 구성원이 함께 생활하는 공간으로 언제나 다중 컨텍스트들이 생성되고 사라지는 공간으로 컨텍스트간의 충돌이 발생하는 공간이다. 다중 컨택스트 관리란 스마트홈이 사용자에게 적절한 서비스를 제공할 수 있도록 컨텍스트간의 충돌을 관리하는 것이다. 센서의 정보를 활용해 컨텍스트간의 충돌을 사전에 감지하고 어플리케이션이 사용자에게 적절한 서비스를 제공하도록 관리하는 것이다. 본 논문에서 제안하는 에니어그램을 이용한 서비스지향 아키텍쳐 컨텍스트 관리정책은 2장 2.4.1절의 <그림 2-23>을 고려하여 분석하였다. 에니어그램을 이용한 서비스지향 아키텍쳐 컨텍스트 관리자는 트리아드의 경로인 9영역의 컨텍스트 충돌 관리자, 3영역의 컨텍스트 해석기, 6영역의 서비스 관리자로 구성되어 있다. 센서로 수집된 사용자와 주변환경의 정보를 헵타드경로인 7단계로 입력받아 1단계에서 컨텍스트를 감지하여 어프리케이션을 선택하고 4단계-2단계-8단계로 사용자에게 필요한 데이터를 분석하여 생성한 후 5단계-7단계로 서비스를 제공해준다. 단, 본 논문에서는 편의상 트리아드 경로는 '영역', 헵타드경로는 '단계' 로 경로구분을 하기로 한다[3,6,17].



<그림 3-1> 서비스지향 아키텍쳐의 컨텍스트 관리원형

<Fig. 3-1> Context management prototype of SOA

3.1.1 컨텍스트 해석기

컨텍스트 해석기는 2장 2.2.4절에서 논한 5W1H형태의 컨텍스트를 얻기 위해 센서를 통하여 수집한 스마트객체의 Who, What, Where, When의 정보를 이용하여 스마트객체의 How와 Why를 얻어낸다.

컨텍스트 해석기는 스마트객체의 관리주체인 사용자의 몸짓, 의도, 감성정보에 관한 라이브러리와 별도의 지능적인 기법을 사용하여 구할 수 있다.

<그림 3-2>와 <표 3-1>에서 컨텍스트 해석기가 사용자의 생활습관 등의 배경지식을 학습하고 추론한다.

아래는 이를 서비스지향 아키텍처의 컨텍스트 처리를 위한 모나드 시나리오로 단순화하여 원형을 제시하였다.

1단계 : 시스템으로 컨텍스트를 감지하여 모델변경을 준비한다.

2단계 : 시스템에서 컨텍스트의 모델 변경과 백업데이타 생성한다.

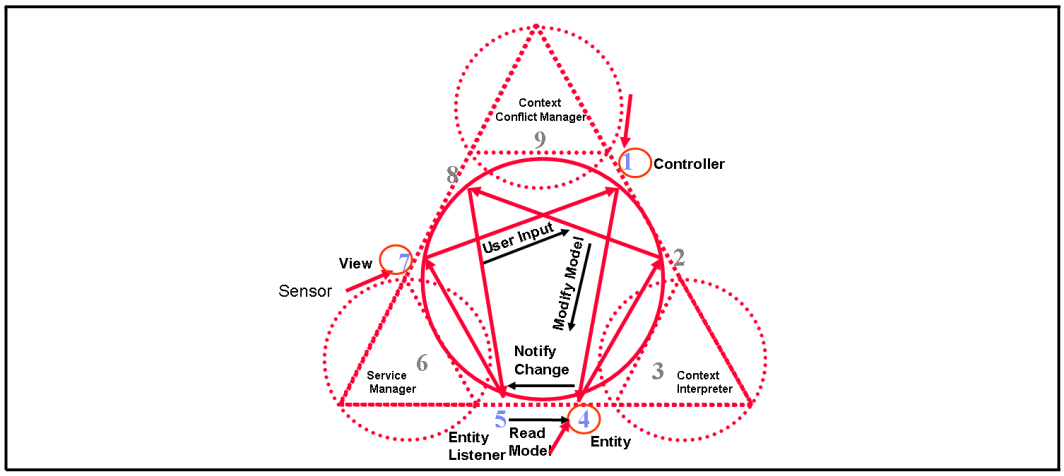
3영역 : 컨텍스트 해석기

4단계 : Who, What, Where, When의 컨텍스트를 준비한다.

5단계 : Who, What, Where, When의 컨텍스트를 처리한다.

6영역 : 서비스 관리자

7단계 : 사용자의 의도인 Why를 파악하고 검증한다.



<그림 3-2> 서비스지향 아키텍처의 컨텍스트 처리원형

<Fig. 3-2> Context processing prototype of SOA

8단계 : 사용자는 필요한 서비스를 조회하고 승인한다.

9영역 : 컨텍스트 충돌관리자

<표 3-1> 서비스지향 아키텍쳐의 컨텍스트 처리 헵타드 시나리오

<Table 3-1> Heptad Scenario of SOA Context Processing

Ennead	서비스 모드	내용
6		서비스 관리자 : 사용자의 컨텍스트 변화에 대하여 센싱대기
7→1	User Input	사용자는 시스템으로 컨텍스트 자동입력
9		컨텍스트 관리자 : 사용자 인증과 컨텍스트 충돌을 사전감지하여 모델변경준비
1→4	Modify Model	시스템은 컨텍스트 감지하여 모델변경
3		컨텍스트 해석기 : Who, What, Where, When의 컨텍스트 해석
4→2	Notify Changes	컨텍스트 모델 해석
2→8	Notify Changes	켄텍스트 모델 변경하여 백업데이타 생성
9		컨텍스트 관리자 : 변경된 컨텍스트에 대한 사용자 인증후 통보
8→5	Notify Changes	변경된 컨텍스트 사용자 리스너에게 통보
6		서비스 관리자 : 사용자의 의도인 Why를 파악하고 검증한다.
5→7	Notify Changes	사용자는 의도된 서비스인지 검증한다.
5→8	Read Model	사용자 인증을 받는다.
8→2	Read Model	사용자는 요청한 서비스를 조회한다.
2→4	Read Model	사용자는 요청한 서비스를 승인한다.

3.1.2 컨텍스트 충돌관리자

컨텍스트 충돌 관리자는 서비스의 충돌을 점검하고 서비스관리자와 협의하여 컨텍스트 충돌을 조정한다. 스마트홈 공간의 특성을 이용하여 컨텍스트간의 우선순위를 정하거나 조명 점등과 조명 소등의 중간치인 보조 조명 점등을 선택하는 것처럼 충돌이 일어난 서비스들의 중간치를 취하는 형태로 충돌을 해결한다. 공간의 특성은 미리 정의된 라이브러리 형태로 주어져야 한다.

(1) 컨텍스트에 따른 서비스의 모순을 스마트서비스 관리자에게 문의한다.

(2) 컨텍스트의 충돌이 발생하면 사용자가 위치한 공간의 특성에 따라 충돌 컨텍스트간의 우선순위를 결정하거나 모순이 일어난 서비스를 적절하게 조정하는데 다음의 2가지 경우가 있다.

1) 다수의 컨텍스트가 하나의 가전기기에게 그 가전기기가 동시에 제공할 수 없는 두가지 이상의 서비스를 요구할 때이며,

2) 서로 다른 컨텍스트가 같은 종류의 서비스를 제공하는 둘 이상의 가전기기에게 상반된 서비스를 요구하여 그 둘이상의 상반된 서비스가 일정공간에서 제공될 때, 즉, 한 컨텍스트가 가전기기에 요구하는 서비스가 다른 컨텍스트에게 부정적인 영향을 미치는 경우이다.

3.1.3 서비스관리자

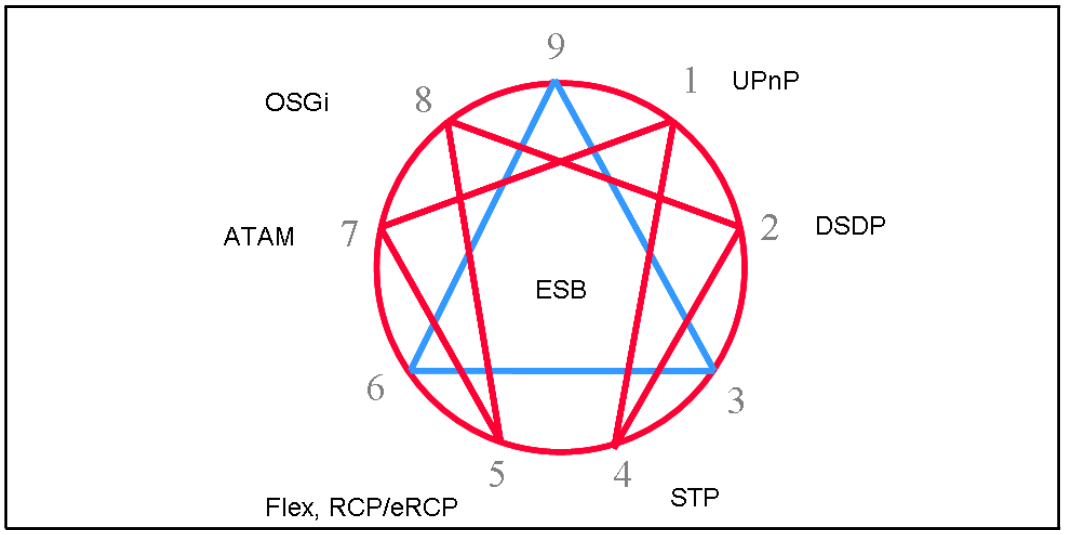
컨텍스트가 생성되면 스마트 서비스관리자는 각 컨텍스트와 관련된 서비스의 종류를 결정하고 서비스를 제공할 어프리케이션을 결정한다. 조명은 시각관련 서비스, 오디오는 청각관련 서비스, TV는 시각과 청각 서비스 등 각 어프리케이션은 제공할 수 있는 서비스의 종류가 정해져 있다. 서비스관리자는 가정에서 사용가능한 어프리케이션의 정보를 미리 가지고 있으며 컨텍스트에 따라 어프리케이션에 제어명령을 내려 서비스를 제공한다. 컨텍스트에 따른 서비스는 미리 정의되어 라이브러리 형태로 정의 될 수 있다.

3.2 서비스지향 아키텍쳐 품질속성

분석된 에니어그램을 이용하여 서비스지향 아키텍쳐 요소에 매핑하여 설계하면 <그림 3-3>과 같이 표현할 수 있으며 본 논문에서 다루고자 하는 품질속성을 <그림 3-4>와 같은 품질속성 다이아그램으로 가정하겠다. 여기서 본 논문의 서론에서 제기되었던 제안된 서비스지향 아키텍쳐의 설계 및 검증하기 위하여 다음과 같이 제안된 6가지 품질속성별 스테이크홀더(Stakeholder)별 정책관리의 조합문제를 생각해 본다.

연구문제 1. 에니어그램을 이용한 서비스지향 아키텍쳐는 도메인의 범위에 따라 시스템, 컴포넌트 레벨에서 다양한 성능(Performance) 또는 관리효율성(Managability) 즉, 시스템레벨에서 다수의 동시 서비스 사용자와 자동화 및 실시간처리 등에 대한 **신속하고 관리효율적인** 품질속성의 정책관리가 서비스 중개자 관점에서 서비스 사용자와 서비스 제공자의 요구에 따라 가능한가?

연구문제 2. 에니어그램을 이용한 서비스지향 아키텍쳐는 도메인의 범위에 따라 시스템레벨에서 가용성(Availability), 보안성(Security) 등 **정확하고 안전한** 품질속성의 정책관리가 서비스 제공자 관점에서 서비스 사용자의 요구에 따라 가능한가?



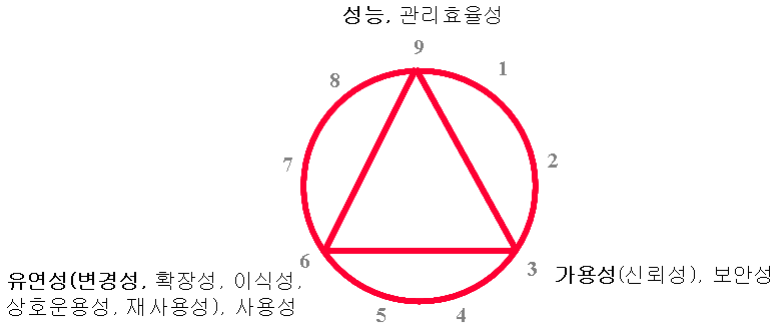
<그림 3-3> 서비스지향 아키텍쳐 Toolkit.

<Fig. 3-3> SOA Toolkit

연구문제 3. 에니어그램을 이용한 서비스지향 아키텍쳐는 도메인의 범위에 따라 시스템레벨에서 유연성(Agility), 사용성(Usability) 등 **유연하고 편리하게 통합되는** 품질속성의 정책관리가 서비스 사용자관점에서 서비스 제공자 및 외부의 변화요구에 따라 민첩한 대응이 가능한가?

먼저 현재 국제표준(ISO/IEC 9126-1)에서는 요구하는 소프트웨어 품질요소를 살펴보고 이것이 상기의 품질속성과 관련된 문제의 영역에서 관련성을 규명한다. ISO/IEC 9126-1에서 요구하는 소프트웨어 품질은 여섯 가지 특성(효율성, 유지보수성, 신뢰성, 기능성, 이식성, 사용성)으로 규정하고 있으며 이러한 특성들은 다시 세부적인 부특성들로 구성된다. 부특성은 소프트웨어 생명주기에 따라 개발단계에서의 품질관련은 9126-3의 내부 메트릭(개발단계의 측정)에서 규정하고 제품 개발완료 후 실행코드 상태에서의 품질관련은 9126-2의 외부메트릭(완성단계의 측정)에서 규정하고 있다[42].

ISO/IEC 9126-1에서는 정의된 품질특성은 펌웨어(firmware)에 수록된 프로그램과 데이터를 포함하여 모든 종류의 소프트웨어에 적용할 수 있다. 품질특



Ennead	Proposed	ISO/IEC 9126-1
9	성능	효율성
	관리효율성	유지보수성
3	가용성	신뢰성
	보안성	기능성
6	유연성	이식성
	사용성	사용성

<그림 3-4> 에니어그램을 이용한 품질속성 다이아그램

<Fig. 3-4> Quality Attribute Diagram using Enneagram

성과 부특성은 소프트웨어 품질에 대한 일관성 있는 체계를 제공하며, 소프트웨어에 대한 품질 요구사항을 상세화하거나 또는 기능성, 신뢰성, 사용성, 효율성, 유지보수성 및 이식성과 같은 소프트웨어 품질특성간의 균형을 맞추기 위한 기본적인 체계를 제공하고 있다. 프로그래밍 환경의 재사용성과 같이 오직 개발자만이 관심을 갖는 소프트웨어 속성은 ISO/IEC 9126의 범위를 벗어난다.

ISO/IEC 9126-1은 구매, 요구명세서, 개발, 사용, 평가, 지원, 유지보수, 품질보증 및 소프트웨어 감사 등과 관련된 사람들이 서로 다른 관점에서 소프트웨어 제품 품질을 정의하고 평가할 수 있도록 한다.

소프트웨어 품질은 정의된 품질 모델을 사용하여 평가해야 하며, 소프트웨어 제품이나 중간제품의 품질 목표를 설정할 경우에 사용된다. <표3-2>와 같이 ISO/IEC 9126-1은 품질과 관련된 문제점에 대한 점검표로서 사용 가능한 품질 모델을 제공한다. 소프트웨어 품질 부특성은 내부 메트릭이나 외부 메트릭에 의해 측정 가능하다. 그러나 규모가 큰 소프트웨어 제품의 모든 메트릭에 대하여 내부 및 외부적으로 모든 부특성을 측정한다는 것은 실제적으로 불가능하다. 마찬가지로 사용자 작업의 모든 가능한 시나리오에 대한 사용중 품질을 측정하는 것도 통상적으로 현실적이지 않다. 평가를 위하여 서로 다른 측정 유형간에 할당되는 자원도 사용목적과 제품이나 설계 프로세스의 속성에 따라 다르게 구성될 수 있다.

<표 3-2> ISO/IEC 9126-1에서 규정된 품질속성표

<Table 3-2> Provided QA table from ISO/IEC 9126-1

Ennead	ISO/IEC 9126-1	포함품질속성
9	효율성	시간반응성, 자원효율성, 준수성
	유지보수성	해석성, 변경용이성, 안정성, 시험성, 준수성
3	신뢰성	성숙성, 오류허용성, 복구성, 준수성
	기능성	적합성, 정확성, 보안성, 준수성
6	이식성	적응성, 설치성, 공존성, 대체성, 준수성
	사용성	이해성, 습득성, 운용성, 친밀성, 준수성

ISO9126-1에서 규정하는 6개 품질특성은 내부 혹은 외부 메트릭을 통하여 정량적인 값으로 측정되며 소프트코드 관점에서 측정 가능한 내부속성에 의한 내부 메트릭은 ISO/IEC 9126-3에 규정되어 있고, 실행소프트웨어 관점에서 시스템이 제공하는 능력을 측정하기 위한 외부 메트릭은 ISO/IEC 9126-2에 규정되어 있다. 따라서 소프트웨어 내, 외부 품질특성을 다음과 같이 분류하여 <그림 3-5>와 연관성을 갖도록 한다.

이를 ATAM평가에 필요한 <표 3-5>의 품질속성(QA:Quality Attribute)표와의 관련성을 찾기 위하여 <표 3-3>의 포함품질속성(iQA:included Quality Attribute)표로부터 유사한 포함품질속성을 <표 3-4>와 같이 추출한다.

<표 3-3> ISO/IEC 9126-1에서 추출된 포함품질속성표

<Table 3-3> Extrated iQA table from ISO/IEC 9126-1

Ennead	ISO/IEC 9126-1	추출된 포함품질속성	품질속성 정의
9	효율성 (Efficiency)	시간반응성 (Time Behaviour)	규정된 조건에서 그 기능을 수행할 때 적절한 반응 및 처리시간과 처리율을 제공하는 소프트웨어 제품의 능력
	유지보수성 (Maintainability)	시험성 (Testability)	변경된 소프트웨어를 확인할 수 있게 하는 소프트웨어 제품의 능력
3	신뢰성 (Reliability)	복구성 (Recoverability)	고장 발생시 규정된 성능수준을 재유지하고 직접적으로 영향받은 데이터를 복구하는 소프트웨어 제품의 능력
	기능성 (Functionality)	보안성 (Secuity)	사용자의 권한에 따라 시스템 또는 시스템 내 정보에 대한 접근제어를 통하여 대상정보를 보호하는 소프트웨어 제품의 능력
6	이식성 (Portability)	적응성 (Adaptability)	소프트웨어 제품이 기본적으로 제공하는 방법만으로 다른 환경으로 변경할 수 있는 소프트웨어 제품의 능력
	사용성 (Usability)	사용성 (Usability)	소프트웨어가 규정된 조건에서 사용될 때, 사용자에 의해 이해되고, 학습되며 선호될 수 있게하는 소프트웨어 제품의 능력

<표 3-4> 제안된 서비스지향 아키텍쳐의 품질속성표

<Table 3-4> Proposed QA Table of SOA

Ennead (Triad)	Proposed	포함품질속성	품질속성 정의
9	성능 (Performance)	**처리율 *(Throughput)*** **응답속도 *(Latency)***	시간에 관한 것으로 이벤트(메시지, 사용자 요청, 시간흐름)발생에 따른 응답의 수행에 있어 이벤트 발생시의 시스템의 응답 속도로 나타난다.
	관리효율성 (Managability)	유지보수성 (Maintainability) 테스트성 (Testability)	시스템의 운영 및 관리에 있어 필요한 역량으로 관리 효율성이 높으면 시스템의 관리 및 유지보수, 운영 검증을 위한 비용이 절감된다.
3	**가용성 *(Availability)***	**신뢰성 *(Reliability)*** 오류허용성 (Fault tolerance)	시스템의 실패 및 오류허용성에 연관된 품질로써 시스템 실패 및 오류의 발생시에 파급되는 효과로 도출된다.
	보안성 (Security)		합법적인 사용자 및 접근 시스템에게 서비스를 제공하면서 인가되지 않은 사용을 막는 역량으로 부인방지, 기밀성, 무결성, 보증, 감사 등의 요소가 있음.
6	유연성 (Agility)	**변경용이성 *(Modifiability)*** **재사용성 *(Reusuability)*** 이식성 (Portability) 상호운영성 (Interoperability)	비즈니스 및 업무의 변화에 따라 시스템이 수행하는 기능과 운영되는 환경, 프로토콜, 품질의 변경이나 변화에 유연하게 대응하는 역량으로 변경 및 변경에 관련된 금전적, 시간적, 기회적 비용으로 도출된다.
	사용성 (Usability)		원하는 작업을 수행하려는 사용자가 느끼는 편의성으로, 시스템의 효율적인 사용과 시스템의 사용법을 쉽게 배우고, 에러의 따른 영향을 최소화하고, 시스템에 적응하여 편안함을 느끼게 만드는 역량으로 늦게 발견될수록 교정에 많은 비용이 들어감

* 굵은표시 품질속성은 평가에서 사용되는 품질속성임.

<표 3-5> ISO/IEC 9126의 추출품질속성과 제안된 품질속성의 매핑

<Table 3-5> Mapping of Extrated iQA of ISO/IEC 9126 and proposed QA

Ennead (Triad)	ISO/IEC 9126-1	추출된포함품질속성	제안된 품질속성
9	효율성	시간반응성 (Time Behaviour)	성능 (Performance)
	유지보수성	시험성 (Testability)	관리효율성 (Managability)
3	신뢰성	복구성 (Recoverability)	가용성 (Availability)
	기능성	보안성 (Secuity)	보안성 (Security)
6	이식성	적응성 (Adaptability)	유연성 (Agility)
	사용성	사용성 (Usability)	사용성 (Usability)

<표 3-3>의 ISO/IEC 9126-1에서 추출된 포함품질속성표와 <표 3-4>에서 제안된 서비스지향 아키텍쳐 품질속성표로부터 품질속성의 유사성을 품질속성 정의를 통하여 확인하고 <표 3-5>와 같이 매핑시킨다. 따라서 <그림 3-4>의 에니어그램을 이용한 품질속성 다이아그램의 가정이 옳다는 것을 검증하였다.

3.3 에니어그램을 이용한 서비스지향 아키텍쳐의 설계

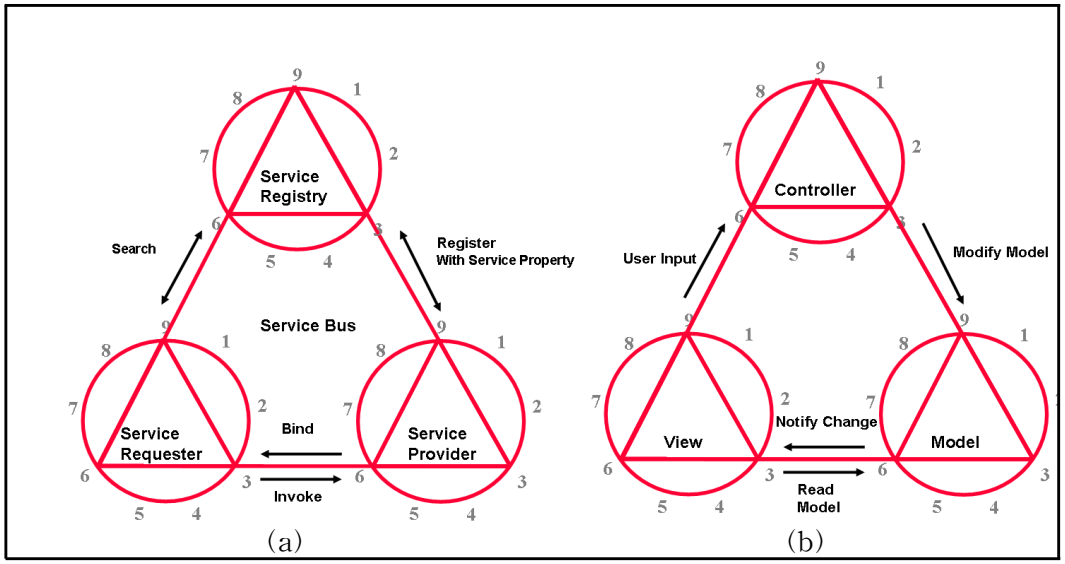
2장 4절의 이론적인 모형을 근거로 다음 절과 같이 임베디드 환경과 엔터프라이즈환경으로 구분하여 범위별 서비스지향 아키텍쳐를 설계하였다. 설계된 서비스지향 아키텍쳐가 서론에서 제기되었던 다음의 가설을 만족하는지는 4장의 서비스지향 아키텍쳐의 평가를 통해서 살펴보겠다.

> 가설. 에니어그램을 이용한 서비스지향 아키텍쳐는 1,000세대 이하 소규모 스마트홈(Stakeholder:서비스 사용자, 아파트관리사무소 등) 컨텍스트에서 3가지 이상 품질속성의 상충요소를 분리하고 적절하게 조합할 수 있으면 서비스 사용자는 만족할 것이다.

본 논문에서는 상기 가설을 검증하기 위하여 <그림 3-4>, <표 3-5>의 품질속성중에서 성능, 가용성, 유연성을 살펴보겠다. 여기서 유연성측면에서 서비스 사용자는 상황변화에 따른 민첩한 대응성을 높이는 가운데 충분히 선택할 수 있는 자유를 서비스 받기 원하기 때문에 가용성측면에서 서비스 공급자는 보증할 수 있는 약속된 서비스를 제공하고자 불확실성을 극복하면서 최대한 필요한 것을 백업, 보안, 점검, 준비하며, 스마트홈 컨텍스의 객체간의 충돌할 수 있는 의도를 사전에 감지하여 파악함으로써 전체 최적화를 향상시켜 복잡도와 성능을 개선하고자 한다.

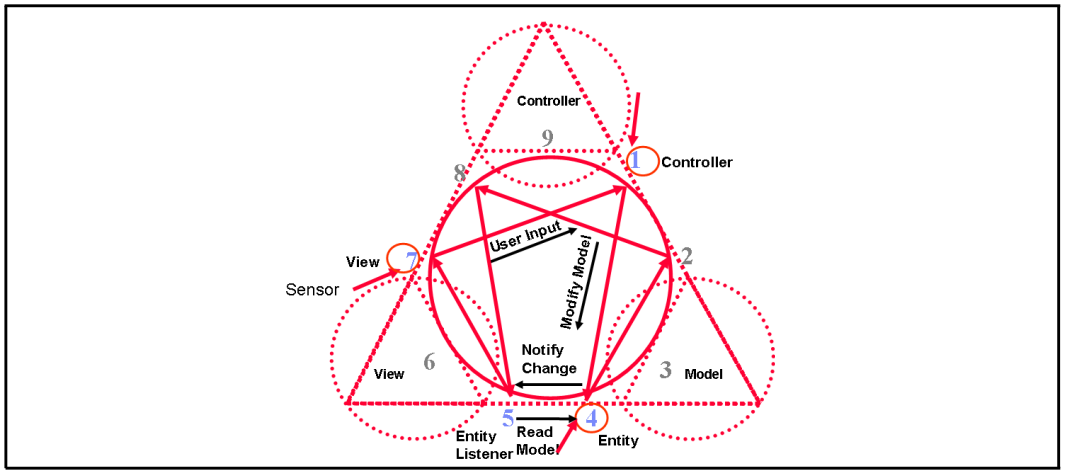
이는 서비스 사용자가 MOT의 서비스 접점순간에는 가치지향적이므로 서비스에 대해 지불된 가격과 그 서비스를 얻기 위한 획득비용보다 더 큰 서비스 결과에 대한 가치와 서비스 전달과정에서의 품질을 원하고 있으며 SPC의 이익증가는 서비스 사용자 충성도로부터 파생되고 서비스 사용자 충성도는 서비스 가치에 대한 서비스 사용자의 만족도로 평가되기 때문이다[23,34,39,43].

3.3.1 임베디드 서비스지향 아키텍쳐 설계



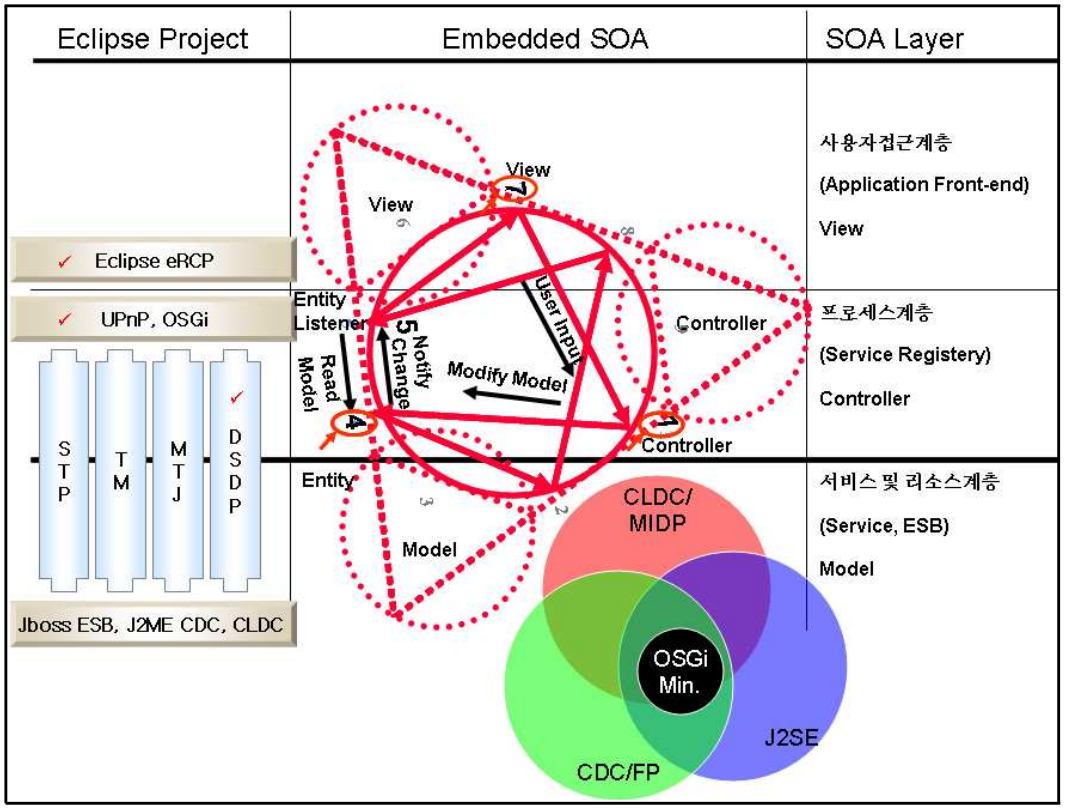
<그림 3-5> 임베디드 서비스지향 아키텍쳐(a)와 MVC 아키텍쳐(b)

<Fig. 3-5> eSOA(a) and MVC architecture(b)



<그림 3-6> 임베디드 서비스지향 아키텍쳐의 내부순회설계

<Fig. 3-6> Internal traveling design of eSOA



<그림 3-7> 임베디드 서비스지향 아키텍처 이클립스 프로젝트

<Fig 3-7> Eclipse project of eSOA

<그림 3-5>에서 OSGi 기반의 임베디드 서비스지향 아키텍처(a)와 MVC아키텍처(b)를 합성하여 <그림 3-6>과 같이 MVC 적용 임베디드 서비스지향 아키텍처(eSOA:embedded Service Oriented Architecture)의 내부순회 헵타드경로(7-1-4-2-8-5-7)를 설계하였다. 설계된 경로에 따라 서비스모드를 설정하고 <그림 3-3>의 서비스지향 아키텍처 툴킷 기본 배치도와 <그림 3-4>의 품질속성을 확인하여 <표 3-6>과 같이 임베디드 서비스지향 아키텍처의 서비스모드, 3가지의 품질속성 및 4종류의 이클립스 플러그인 프로젝트와 <표 3-7>과 같이 eRCP(embedded Rich Client Platform)를 포함한 상세한 계층별 이클립스 프로젝트 구성표를 정리하였으며 <그림 3-7>과 같이 임베디드 이클립스 프로젝트 스크린샷으로 해당 이클립스 사이트에서 확인하였다.

여기서 MVC 적용 임베디드 서비스지향 아키텍처의 내부순회 헵타드경로

<표 3-6> 임베디드 서비스지향 아키텍처의 서비스모드, 품질속성

<Table 3-6> Service Mode and QA of eSOA

Ennead (Heptad)	MVC Mode	Quality Attribute
	Service Mode	Eclipse plug-in project
7→1	View→Controller	성능(서비스 발견)
	User Input	프로시스트 UPnP
1→4	Controller→Model	성능(서비스 조립)
	Modify Model	이클립스 DSDP
4→5	Model→View	가용성(서비스 전달)
	Notify Changes	프로시스트 OSGi
5→4	View→Model	유연성(서비스 가상화)
	Read Model	이클립스 eRCP

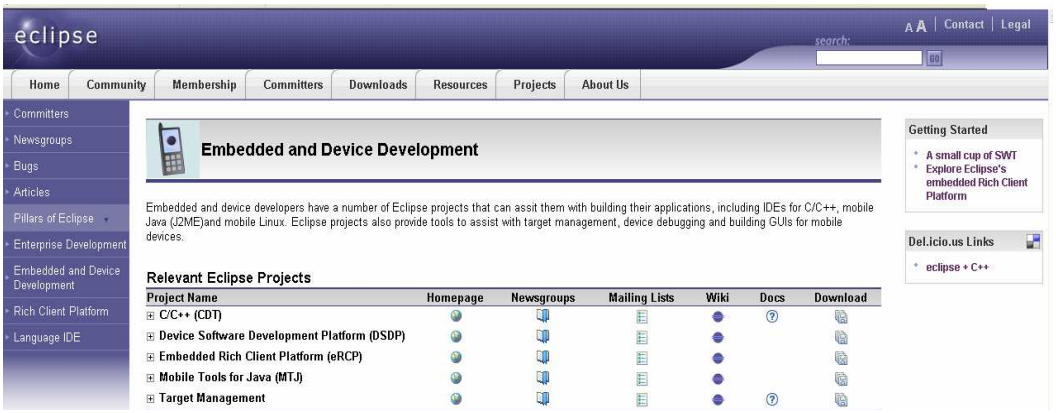
(7-1-4-2-8-5-7)를 살펴보면 “7-1-4”의 경로는 사용자의 요청이 View에서 입력되어 Controller에게 전달되어 해당 로직이 발생하며 Model의 수정에 이르는 명시적인 헵타드 경로이다. 정상적으로 문제가 처리되어 Model의 Entity에서 쿼리결과값을 “4-2-8-5”의 암묵적인 헵타드 경로를 통하여 사용자의 요청에 대한 변화를 사용자의 Entity Listener에게 공지하면, 감지된 사용자는 역헵타드 경로를 통하여 “5-8-2-4”의 암묵적인 헵타드 경로를 통하여 해당 값을 읽을 수 있게 된다. 이러한 이중적인 서비스관리정책이 필요한 이유는 서비스의 요청은 명시적으로 받아들이지만, 요청 결과값에 대한 처리는 백업·복구와 보안상의 필요에 따라 가용성의 품질속성이 요구되며, 유지보수성의 필요에 따라 성능의

<표 3-7> 임베디드 서비스지향 아키텍처의 이클립스 프로젝트

<Table 3-7> Eclipse Project of eSOA

Layer	Eclipse Project	Homepage URL
View	eclipse eRCP	eclipse.org/ercp
Controller	UPnP	upnp.org, prosyst.com
	OSGi	osgi.org, prosyst.com, eclipse.org/equinox
Model	DSDP	eclispe.org/dsdp
	MTJ	eclispe.org/dsdp/mtj
	TM	eclispe.org/dsdp/tm
	J2ME CDC,CLDC	java.sun.com/products/sjwtoolkit/

품질속성이 요구되고, 요청결과값의 조회에 대해서는 변경용이성의 필요에 따라 유연성의 품질속성이 요구되므로 암묵적인 헵타드 경로를 통하여 처리되는 메카니즘을 정하였다. 또한 <표 3-8>과 같이 해당서비스의 생산적인 서비스 조립을 위하여 이클립스 기반에서 필요한 임베디드 프로젝트를 제안된 임베디드 서비스지향 아키텍처와 프로시스트 번들을 관련함으로써 가용성과 유연성을 한층 더 확보하여 서비스 사용자와 서비스 제공자의 선택의 폭을 넓혔다.



<그림 3-8> 임베디드 이클립스 프로젝트의 스크린샷

<Fig. 3-8> Screenshot of embedded eclipse project

<표 3-8> z256 프로토콜기반 서비스지향 아키텍처의 프로시스트 jar 번들

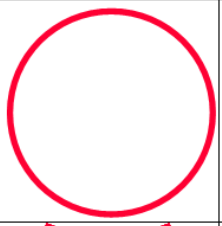
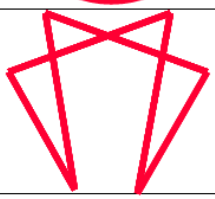
<Table 3-8> Prosyst jar bundle of SOA based z256 protocol

Layer	Prosyst jar Bundle	Servelet service URL
View	z256http.jar	*.class
	PDASocketHander.jar	*.class
Controller	commw32.jar	-
	connector.jar	-
Model	DLz256.jar	-
	z256.jar	-

3.3.2 엔터프라이즈 서비스지향 아키텍쳐 설계

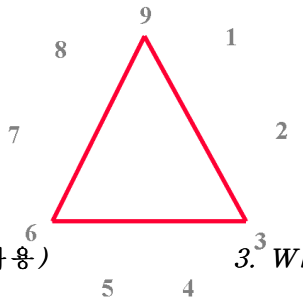
(1) 5W1H에 따른 RNS-PAC-MVC 아키텍쳐

6하원칙과 RFM(Recency, Frequency, Monetory)에 따라 분류된 역할기반 RNS와 PAC 에이전트 아키텍쳐는 엔터프라이즈 아키텍쳐의 구체적인 사상과 절차를 제공한다(단, 여기서 Role*은 자원관리의 개념에 한하여 What으로 매핑함).

Enneargram	5W1H	(6하원칙, RFM)	RNS-PAC-MVC
	Why Who	(의도) , Frequency	Sanction (Presentation-View)
	How Where, When	(방법) , Recency	Norm (Control-Controller)
	What	(구조) , Monetory	Role* (Abstraction-Model)

단, 트리아드는 다음과 같이 재구성할 수 있다.

9. *Where,When/How(필요충분조건,최적 · 관리)(매개변수)*



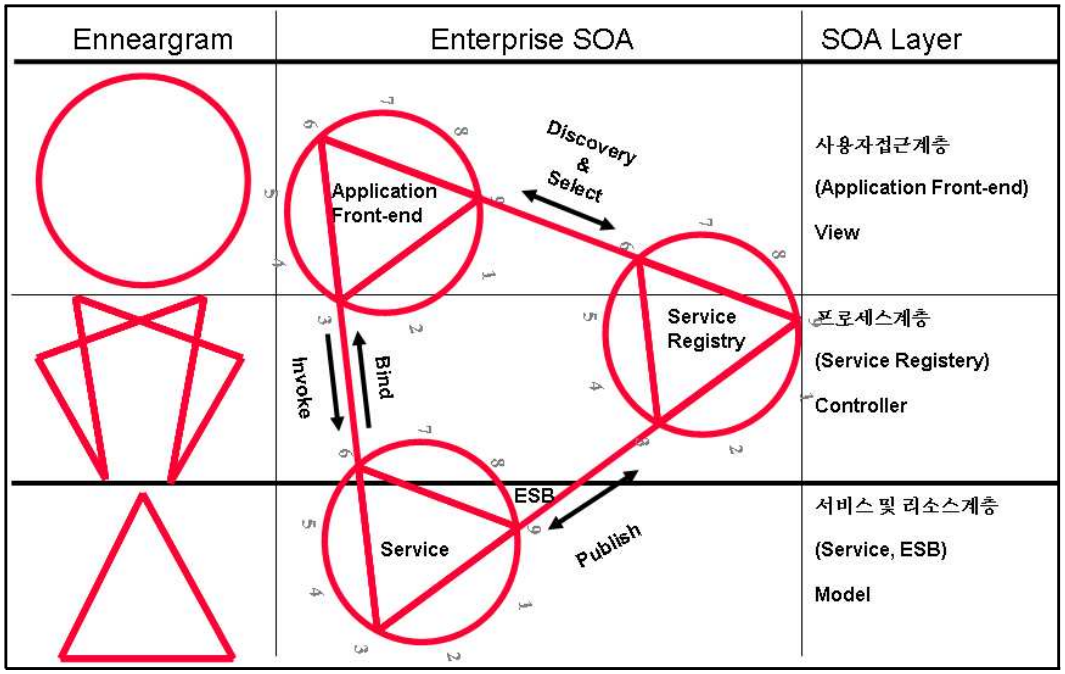
6. *Who/Why(충분조건,최소 · 사용)*
(독립변수)

3. What(필요조건,최대 · 소유)
(종속변수)

<그림 3-9> 5W1H을 이용한 RNS-PAC-MVC 아키텍쳐

<Fig. 3-9> RNS-PAC-MVC Architecture using 5W1H

(2) 수직구조 엔터프라이즈 서비스지향 아키텍처



<그림 3-10> 제안된 수직구조 엔터프라이즈 서비스지향 아키텍처

<Fig. 3-10> Proposed Vertical enterprise SOA

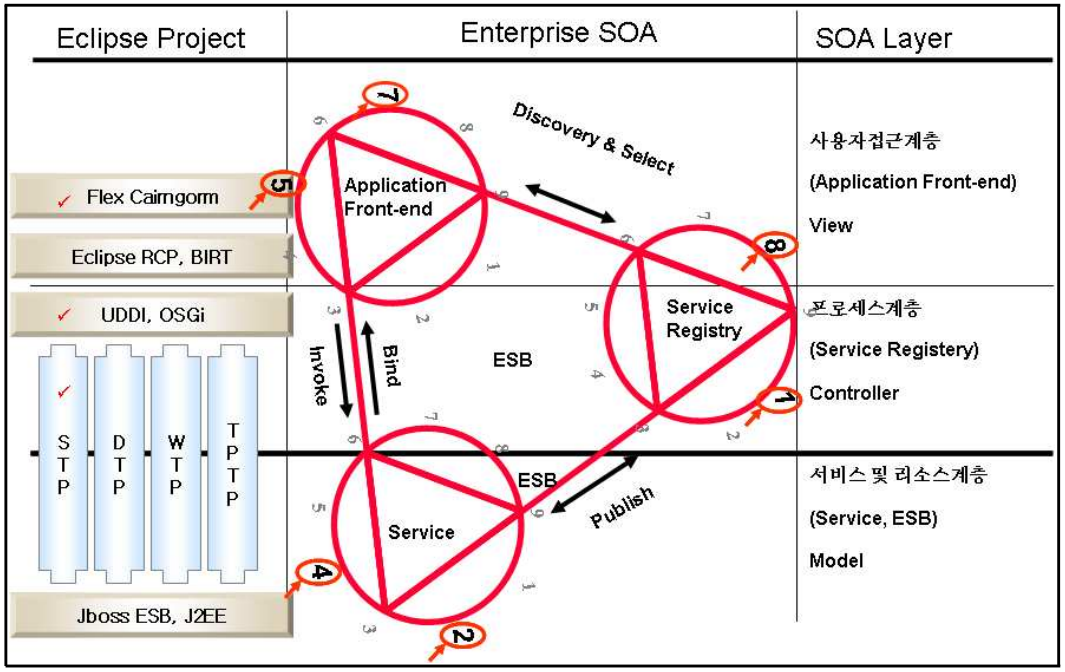
본 논문 2장 2.4.1절의 <그림 2-19>,<그림 2-20>에서 제시된 엔터프라이즈 서비스지향 아키텍처의 이론적인 모형을 근거로 리소스계층을 서비스계층과 통합하여 <그림 3-10>과 같이 수직구조의 엔터프라이즈 서비스지향 아키텍처를 제안한다. 이는 기존 서비스지향 아키텍처의 설계사상에서 MVC아키텍처의 개념을 합성함으로써 수평적, 수직적인 통찰력을 제공해준다[24,27,38,48,49].

(3) 엔터프라이즈 서비스지향 아키텍처 이클립스 프로젝트

<그림 3-11>과 같이 MVC 기반의 엔터프라이즈 서비스지향 아키텍처의 이클립스 프로젝트를 설계하였다. 설계된 경로에 따라 서비스모드를 설정하고 <그림 3-3>의 서비스지향 아키텍처 Toolkit과 <그림 3-4>의 품질속성을 확인하여 <표 3-9>과 같이 엔터프라이즈 서비스지향 아키텍처의 서비스모드, 5가지의 품질속성 및 4종류의 이클립스 플러그인 프로젝트와 <표 3-10>과 같이 ORM(Object-Relational Mapping) 이슈를 극복하기 위해 상세한 계층별 이클

립스 프로젝트 구성표를 정리하였으며 <그림 3-12>와 같이 엔터프라이즈 이클립스 프로젝트 스크린샷으로 해당사이트에서 확인하였다

여기서 MVC 적용 엔터프라이즈 서비스지향 아키텍처의 내부순회 트리아드 경로(6-9-3)의 주위를 맵도는 헵타드 경로(7-1-4-2-8-5-7)를 살펴보면 “7-1-4”의 경로는 사용자의 요청이 View에서 입력되어 Controller에게 전달되며 해당 로직이 발생하여 Model의 수정에 이르는 명시적인 헵타드 경로이다. 정상적으로 문제가 처리되어 Model의 Entity에서 쿼리결과값을 “4-2-8-5”의 암묵적인 헵타드 경로를 통하여 사용자의 요청에 대한 변화를 사용자의 Entity Listener에게 공지하면, 감지된 사용자는 역헵타드 경로를 통하여 “5-8-2-4”의 암묵적인 경로를 통하여 해당값을 읽을 수 있게 된다. 이러한 이중적인 서비스 관리정책이 필요한 이유는 서비스의 요청은 명시적으로 받아들이지만, 요청결과값에 대한 처리는 백업·복구와 보안상의 필요에 따라 가용성의 품질속성이 요구되며, 유지보수성의 필요에 따라 성능의 품질속성이 요구되고, 요청결과값의 조회에 대해서는 변경용이성의 필요에 따라 유연성의 품질속성이 요구되므



<그림 3-11> 엔터프라이즈 서비스지향 아키텍처의 이클립스 프로젝트

<Fig. 3-11> Eclipse project of enterprise SOA

<표 3-9> 엔터프라이즈 서비스지향 아키텍쳐의 서비스모드, 품질속성

<Table 3-9> Service Mode and QA of enterprise SOA

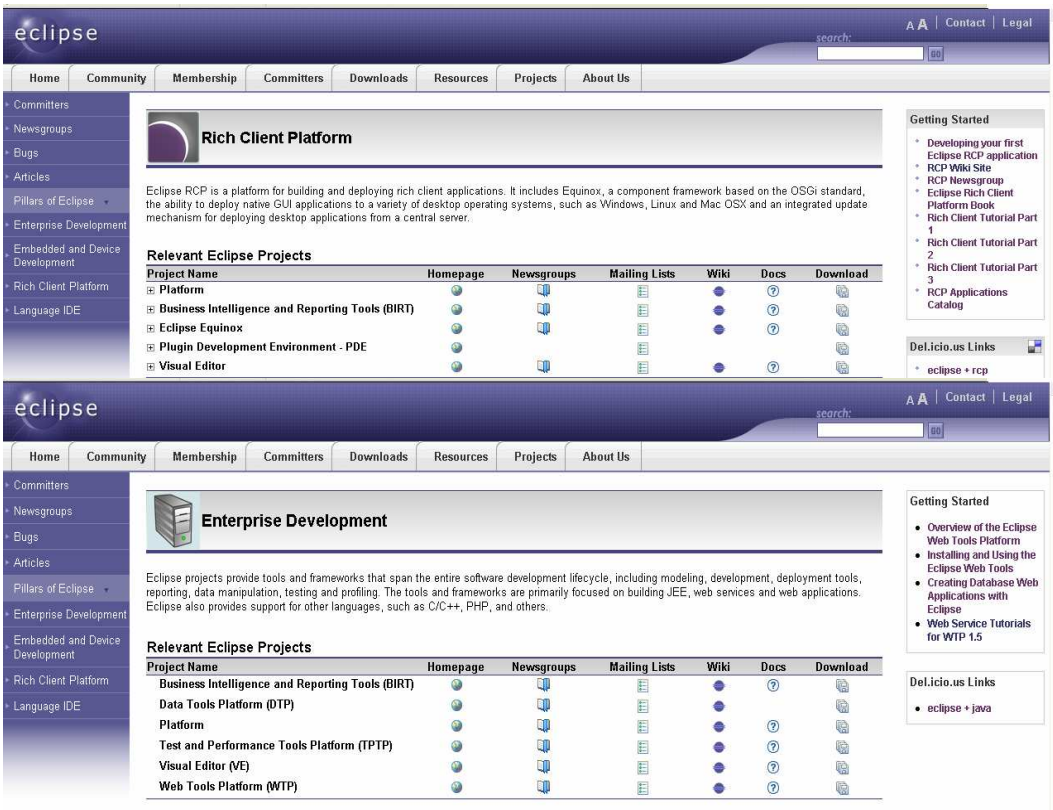
Ennead (Heptad)	MVC Mode	Quality Attribute
	Service Mode	Eclipse plug-in project
7 →1	View →Controller	성능(서비스 발견)
	Discovery & Select:: User Input	UDDI
1 →4	Controller →Model	성능(서비스 조립)
	Publish::Modify Model	이클립스 STP
4 →5	Model →View	가용성(서비스 전달)
	Bind::Notify Changes	프로시스트 OSGi
5 →4	View →Model	유연성(서비스 가상화 : 변경용이성, 상호운영성, 재사용성)
	Invoke::Read Model	Flex Cairngorm

로 암묵적인 헵타드 경로를 통하여 처리되는 메카니즘을 정하였다. 해당서비스의 생산적인 서비스 조립을 위하여 <표 3-10>과 같이 엔터프라이즈 이클립스 프로젝트에서 제안된 서비스지향 아키텍쳐와 관련함으로써 가용성과 유연성을 한층 더 확보하여 서비스 사용자와 서비스 제공자의 선택의 폭을 넓혔다.

<표 3-10> 엔터프라이즈 서비스지향 아키텍쳐의 이클립스 프로젝트

<Table 3-10> Eclipse project of enterprise SOA

Layer	Eclipse Project	Homepage URL
View	Flex Cairngorm	labs.adobe.com/wiki/index.php/cairngorm
	eclipse RCP,BIRT	eclipse.org/rcp, eclipse.org/birt
Controller	UDDI	uddi.org
	OSGi	osgi.org, prosyst.com, eclipse.org/equinox
Model	TPTP	eclispe.org/tptp
	WTP	eclispe.org/webtools, springframework.org
	DTP	eclispe.org/datatools, hibernate.org
	STP	eclispe.org/stp, polaris.ing.unimo.it/MOON/BRAIN
	JBoss ESB	labs.jboss.com/portal/jbossesb
	J2EE	java.sun.com/javaee



<그림 3-12> 엔터프라이즈 이클립스 프로젝트의 스크린샷

<Fig 3-12> Screenshot of enterprise eclipse project

(4) 이클립스 OHF 설계

유비쿼터스 컴퓨팅 기술의 주요 적용 도메인으로 U헬스는 매력적인 신사업 기회로 인식되고 있으며 IT 인프라 및 유비쿼터스 관련 기술의 빠른 보급 환경, 아파트 중심의 거주 환경, 고령화 시대를 맞는 우리나라에서는 U헬스를 통한 서비스 사용자 삶의 질 향상 욕구와 서비스 제공자의 새로운 사업의 이해관계가 있다. 실제 서비스 사용자의 요구에 부응하는 U헬스를 위하여 면밀한 서비스 사용자 요구 분석과 의료 서비스 구성원의 이해관계 매듭을 풀지 않고서는 성공할 수 없는데 U헬스를 통한 서비스 사용자(환자)의 이익 대비 진료 서비스 공급자인 의료인의 이익이 선결되어야 하며, 의료인도 U헬스의 기대 효과를 인지하고 스스로 준비해야 한다. 따라서 스마트홈 다중 컨텍스트관리를 통하여 U헬스분야에서 HL7과 IHE 분석을 한 후 이클립스기반의 OHF를 제안된 에니어그램을 이용한 서비스지향 아키텍쳐에 플러그인함으로써 새로운 사업 모

형 및 진료 모형을 제시할 수 있다.

(1) 이용자 측 : 다양한 의료 정보 접근 용이, 웰빙 욕구 충족

(2) 제약 및 제조사 측 : 업무 프로세스 단축, 거래 투명성 제고

(3) 의료기관 측 : 신속한 진료 서비스 체계 구축, 진료 서비스 향상, 진료 수입 확대

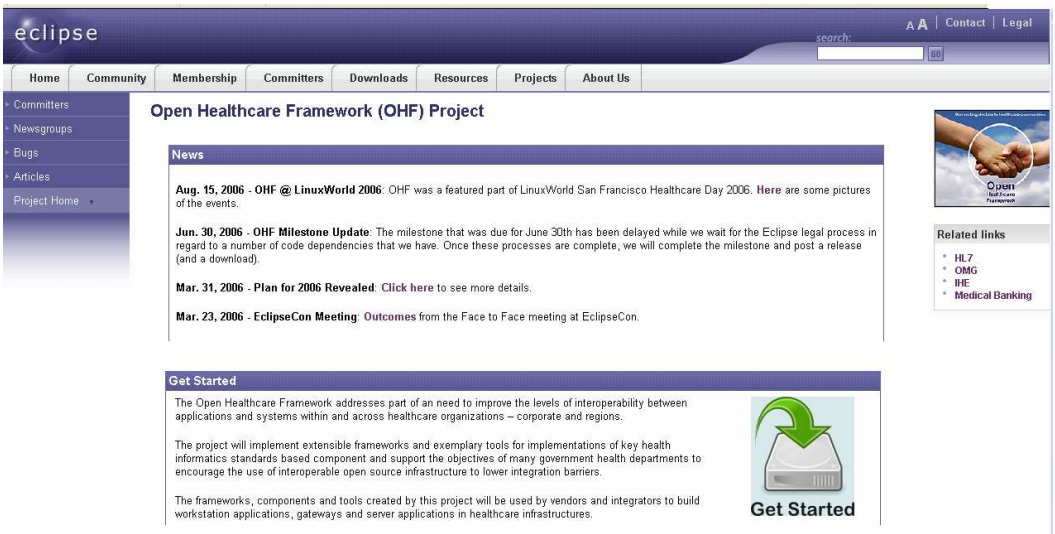
(4) 통신사업자 측 : 신규 사업 캐시카우 발굴, 의료 시장 진출 초석 마련

(5) 정부 측 : 의료비 지출 절감, 의료 서비스 국제 경쟁력 강화

<표 3-11> 이클립스 OHF 프로젝트의 MVC

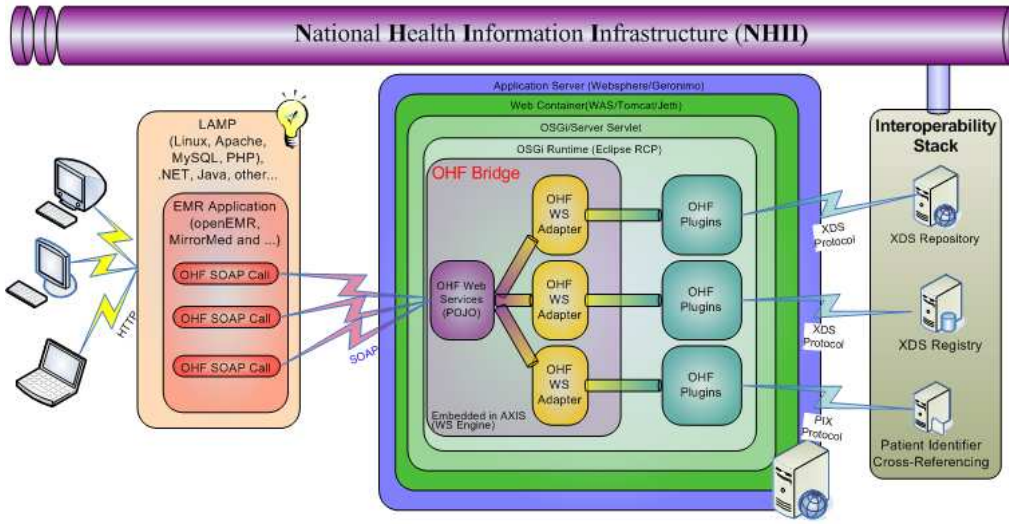
<Table 3-11> MVC of eclipse OHF project

Layer	Name	Contents
View	LAMP	Linux, Apache, MySQL, PHP
	EMR Application	Open EMR(Electric Heath Record)
Controller	Application Server	Webshpere
	Web Container	Tomcat
	OSGi Server	Servlet
	OSGi Runtime	Eclipse RCP
Model	Interoperability Stack	XDS Repository, XDS Registry, Patient Identifier Cross-Referencing



<그림 3-13> 이클립스 OHF 프로젝트의 스크린샷

<Fig. 3-13> Screenshot of eclipse OHF project



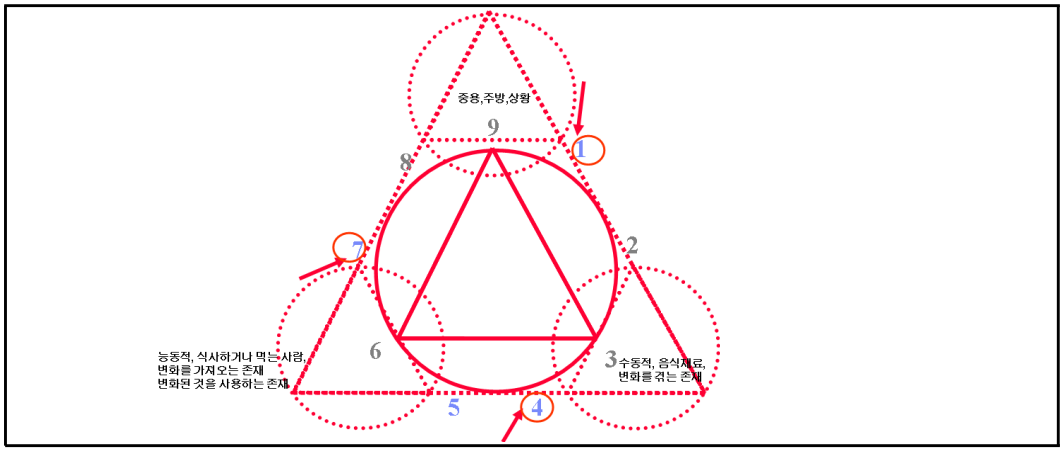
<그림 3-14> 이클립스 OHF 아키텍처

<Fig. 3-14> eclipse OHF architecture

3.4 스마트홈 시나리오 설계

3.4.1 개요

스마트홈에서 요구하는 표준서비스 시나리오를 추출하기 위하여 지능형 스마트홈의 사업관점에서 서비스와 기술에 대한 이해를 하고 이를 면밀하게 분류할 필요가 있다. 본 논문에서는 경상남도 지역산업진흥사업의 지능형홈 중점육성 대상서비스의 분류기준으로 하였으며 자세한 구조와 분류는 부록에 별도로 정리하였다. 앞장에서 설계된 범위별 서비스지향 아키텍쳐의 설계를 ATAM 평가방법론으로 검증하기에 앞서서 먼저 시나리오를 구체화하는 단계에서 부록에 있는 바와 같이 스마트홈의 5대 서비스에서 편리한 생활을 위한 원격제어와 안전한 생활을 위한 방범방재에 적용함으로써 건강한 생활을 위한 원격 홈헬스케어로 확장할 수 있는 가능성을 확인하고자 한다. 이를 위하여 스마트홈 주택유형중에서 사이버아파트를 선택하여 댁내외 구역별 서비스중에서 다중컨텍스트가 발생할 수 있는 개연성이 있는 공동공간인 거실이 유리하나 현실적인 실험환경 여건상 <부록그림 4>의 구역별 대상서비스에서 정리하였듯이 현관 및 통로, 거실, 주방, 침실, 파우더 룸, 외출에서 공통적으로 가장 많이 존재하는 조명관리 단위서비스와 거실, 주방, 외출에서 공통적으로 존재하는 가전제어 단위서비스, 주방과 외출모드에 공통적으로 존재하는 방재관리 단위서비스를 포함하여 총 3가지의 단위서비스를 선택하여 이를 주방모드와 외출모드의 표준시나



<그림 3-15> 에니어그램의 시나리오 개념도

<Fig. 3-15> Scenario concept of Enneagram

리오에서 에니어그램을 이용하여 재설계하겠다.

2장 2.2.1절에서 언급한 콘텐츠 측면에서 OSMU, 컴포넌트 측면에서 WORA, WOPA의 사상에 따라 에니어그램을 다양하게 응용해 보았다. 이제 <그림 3-15>의 에니어그램의 시나리오 개념도를 살펴보면서 부록의 주방모드와 외출모드를 재설계하는데 필요한 사상을 준비한다.

주방모드를 추상화해서 똑 같은 간격의 점 아홉개로 이루어진 원을 적용하는 것에서 시작하자. 이것은 모나드, 즉 전체과정의 순환을 나타내는 살이 아홉개인 바퀴이다. 이 예에서 주방은 음식을 준비하고 제공한 다음, 모든 것을 깨끗이 청소하고 이 과정을 다시 반복할 준비를 갖춘다[1,12].

숫자 9를 맨위에다 쓰고, 시계방향으로 나머지 점들에 순서대로 번호를 붙인다. 9, 3, 6 번의 점을 직선으로 연결하여 트리아드의 원리, 곧 모든 사건에 내재하는 세부분이 구조(수동적인 측면, 능동적인 측면, 중용적인 측면)를 나타내는 정삼각형을 그린다.

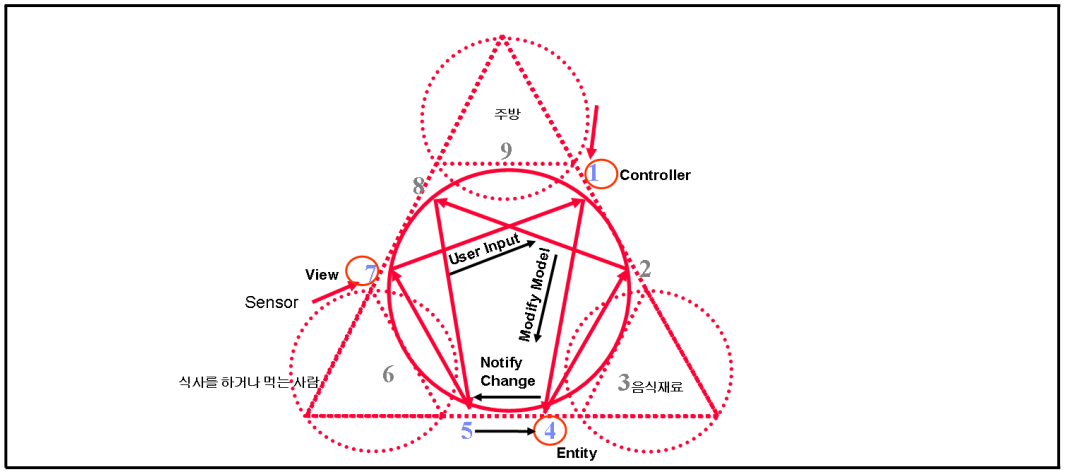
① 수동적인 요소는 변화를 겪는다. 그것을 준비하고, 요리하고, 차려놓아야 하는 음식재료이다.

② 이 경우에 능동적인 요소는 식사하는 사람들, 곧 준비된 음식을 먹는 사람들로 구성된다.

③ 중용적인 요소는 주방 그자체로, 양극의 반대되는 것들을 서로 건설적으로 반응하도록 해주고, 그것들을 묶어 새로운 차원에서 완전한 전체로 결합시킨다. 이 세가지 측면은 삼각형의 세 꼭지점을 이룬다.

에니어그램의 전체 모나드경로에서 트리아드 경로에 따라 음식재료는 3영역에서 들어오고, 식사하는 사람들은 6영역에서 들어온다. 주방은 9영역의 정점에 위치함에 따라 이 세 요소는 모든 주방의 구조를 이루고 있다. 헵타드경로에 속하는 연속되는 각각의 두 단계는 삼각형에서 이전의 꼭지점을 연장시킨다. 1단계와 2단계는 주방을 포함하고, 4단계와 5단계는 음식재료를 포함한다. 그리고, 7단계와 8단계는 식사하는 사람을 포함한다. 주방에서의 경험은 사건들의 외면적인 순서를 구성하는 단계를 채우게 도와준다.

3.4.2 주방모드



<그림 3-16> 주방모드 에니어그램

<Fig. 3-16> Kichen mode Enneagram

아래는 <표 3-12>의 서비스지향 아키텍쳐 컨텍스트 처리 모나드 시나리오로 단순화하였다.

1단계 : 주방에서 일을 할 준비를 한다.

2단계 : 주방에서 일을 한다.

3영역 : 음식재료, 조명, 주방가전, 가스밸브

4단계 : 음식을 준비한다.

상황발생시 조명관리, 주방가전제어, 방재관리를 한다.

5단계 : 음식을 요리한다.

6영역 : 식사하는 사람들, 식사를 먹는 사람들

7단계 : 식사하는 사람들, 식사를 먹는 사람들에게 음식을 차려 제공한다.

8단계 : 식사하는 사람들 또는 식사를 먹는 사람들은 음식을 먹은 후 모든 것을 깨끗이 청소한다.

9영역 : 주방

남아있는 점을 트리아드의 경로에 따라 "7-1-4-2-8-5-7"대로 직선으로 연결한 다음, 순환을 처음부터 다시 시작하게 된다. 이 여섯 내부의 선들은 과정중의 어떤 단계에서도 '앞을 생각할' 수 있게 도와준다.

<표 3-12> 주방모드를 위한 헵타드 시나리오

<Table 3-12> Heptad scenario for kichen mode

Ennead	서비스 모드	내용
6		서비스 관리자 : 사용자의 컨텍스트 변화에 대하여 센싱대기 (조명관리, 주방가전제어, 방재관리)
7→1	User Input	사용자는 시스템으로 컨텍스트 자동입력
9		컨텍스트 관리자 : 사용자 인증과 컨텍스트 충돌을 사전감지하여 모델변경준비 (조명관리, 주방가전제어, 방재관리)
1→4	Modify Model	시스템은 컨텍스트 감지하여 모델변경
3		컨텍스트 해석기 : Who, What, Where, When의 컨텍스트 해석 (조명관리, 주방가전제어, 방재관리)
4→2	Notify Changes	컨텍스트 모델 해석
2→8	Notify Changes	켄텍스트 모델 변경하여 백업데이타 생성
9		컨텍스트 관리자 : 변경된 컨텍스트에 대한 사용자 인증후 통보
8→5	Notify Changes	변경된 컨텍스트 사용자 리스너에게 통보
6		서비스 관리자 : 사용자의 의도인 Why를 파악하고 검증한다.
5→7	Notify Changes	사용자는 의도된 서비스인지 검증한다.
5→8	Read Model	사용자 인증을 받는다.
8→2	Read Model	사용자는 요청한 서비스를 조회한다.
2→4	Read Model	사용자는 요청한 서비스를 승인한다.

예를 들면, 자신이 주방에서 일을 준비하는 1단계에 있다고 가정한다.

그곳에서 실제로 어떤 일을 해야 하는지 결정하려면, 거기서 4단계로 뻗어가는 선을 향해 바라보고, 음식을 준비하기 위해 무엇이 필요한지 생각하게 해준다. 따라서, 자르고 써는 도구와 그 밖의 조리도구, 그릇, 양념, 그리고 음식재료의 준비에 필요한 것들을 준비한다.

또한 7단계로 뻗어가는 선을 향해 바라보면 음식을 내놓기 위해 무엇이 필요한지 생각하며 쟁반, 수저, 냅킨, 유리컵 등을 준비한다.

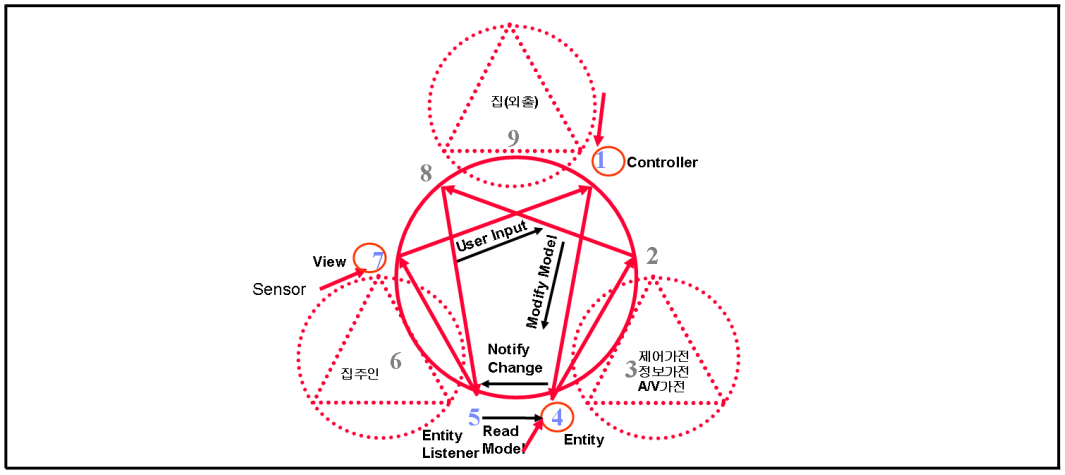
이렇게 주방을 준비한 다음에는 외면적인 순서에 따라 2단계로 나아가 일을 한다. 2단계에서는 선들이 4단계와 8단계로 나아간다.

8단계는 음식을 먹는 것과 특정 종류의 음식, 손님들의 음식기호에 대해 생각한다. 그것은 자신이 어떤 음식을 제공할 것인지. 그리고 그것을 어떻게 제공할 것인지 살펴본다.

방문자가 방문시에 적절한 음식의 리스트를 요청하고, 3장 3.1.1절에 있는 <그림 3-2>의 컨텍스트 해석기에게 보고하고, 컨텍스트 해석기는 이를 서비스 매니저에게 보고한다. 4단계는 보고받은 서비스매니저는 음식을 준비하기 위해 실제로 어떻게 해야 할지, 해당 어프리케이션 리스트를 호출하여 즉 음식재료를 구울지, 튀길지, 찔지 들을 생각한다. 재료의 재고를 확인하기 위해 냉장고에 저장된 재료의 종류를 다시 요청하고 부족하면 컨텍스트 충돌관리자에게 해당 자원에 대한 주문예약을 실시한다.

3영역에서 음식재료가 등장하며, 이제 서비스 제공자는 음식을 준비하고 요리하는 4단계와 5단계로 나아갈 수 있다. 그다음에 6영역에서는 식사를 할 사람들이 등장하고, 7단계, 8단계에서 음식을 차려놓고 먹게 된다. 그런 다음 서비스 제공자는 주방을 청소하고 9영역에 이르러, 다음에 그것을 반복할 수 있도록 식당을 깨끗하게 준비된 상태에서 순환을 완성하는 목표를 달성한다.

3.4.3 외출모드



<그림 3-17> 외출모드 에니어그램

<Fig. 3-17> Outdoor mode Enneagram

아래는 <표 3-13>의 서비스지향 아키텍쳐의 모나드 시나리오로 단순화하였다.

1단계 : 댁내를 모니터 할 준비를 한다.

2단계 : 댁내의 카메라 및 센서가 모니터 한다.

3영역 : 조명기기, 제어가전(선풍기), 가스밸브, 정보가전(PC), A/V가전(TV)

4단계 : 조명기기, 제어가전(선풍기), 가스밸브, 정보가전(PC), A/V가전(TV)를 감시하고 상황충돌을 인식한다.

5단계 : 조명기기, 제어가전(선풍기), 가스밸브,정보가전(PC), A/V가전(TV)를 조건에 따라 처리한다.

6영역 : 집주인

7단계 : 집주인은 조명기기, 제어가전(선풍기), 가스밸브, 정보가전(PC), A/V가전(TV)의 오동작 및 상황충돌을 처리하고 외출을 계속한다.

8단계 : 집주인은 외출(퇴근)후 귀가한다.

9영역 : 집

본 시나리오는 <그림 3-18>과 같이 인터넷을 사용하는 10대의 두 자녀를 둔 40대 맞벌이 부부가 전날 회사 업무차 마감시간을 맞추느라 무리하게 밤샘을 하였고 아침에 늦잠을 자서 일어나보니 비가 내려서 날씨가 어둡고 좋지 않는 날이라고 가정하겠다. 급히 아침식사를 하고 자녀들은 학교에 등교시키고 부부는 출근하였는데 급히 나오다 보니 댁내의 가스밸브의 개폐상태와 큰방의 제어가전(선풍기)의 상태를 알 수 없는 상황이다. 그리고, 자녀들의 게임 등 교육목적상 정보가전(PC)를 거실에 두면 좋다는 권고에 따라 A/V가전(TV)과 같이 거실에 두었다고 가정한다. 오후에 두 자녀가 학교에서 돌아와서 거실에 있는 정보가전(PC)와 A/V가전(TV)에 대하여 한 자녀가 독점하고자 하거나, 한 자녀는 본인이 원하는 프로를 A/V가전(TV)를 보고 있고 다른 자녀는 상당히 조용하게 정보가전(PC)로 인터넷 어학강의를 듣고자 하는 상황충돌이 발생한다.

스마트홈에서는 해당 장치 기본정보를 바탕으로 컨텍스트 관리 서비스를 실행하며 아침에는 원격에서 PDA나 웹으로 네트워크 카메라를 통하여 부엌의 조명기기를 켜서 가스밸브를 확인하고 켜져 있으면 스위칭제어를 하며 거실, 안방, 작은방의 제어가전(선풍기)도 켜져 있으면 스위칭제어를 실시한다. 이때, 센서는 조명기기, 제어가전(선풍기), 가스밸브 등 제어가전(선풍기)의 동작을 감

<표 3-13> 외출모드를 위한 헵타드 시나리오

<Table 3-13> Heptad scenario for outdoor mode

Ennead	서비스 모드	내용
6		서비스 관리자 : 사용자의 컨텍스트 변화에 대하여 센싱대기
7→1	User Input	사용자는 시스템으로 컨텍스트 자동입력 (원격조명관리, 원격가전제어, 원격방재관리)
9		컨텍스트 관리자 : 사용자 인증과 컨텍스트 충돌을 사전감지하여 모델변경준비 (원격조명관리, 원격가전제어, 원격방재관리)
1→4	Modify Model	시스템은 컨텍스트 감지하여 모델변경
3		컨텍스트 해석기 : Who, What, Where, When의 컨텍스트 해석 (조명관리, 제어,정보,A/V가전제어, 방재관리)
4→2	Notify Changes	컨텍스트 모델 해석
2→8	Notify Changes	켄텍스트 모델 변경하여 백업데이타 생성
9		컨텍스트 관리자 : 변경된 컨텍스트에 대한 사용자 인증후 통보
8→5	Notify Changes	변경된 컨텍스트 사용자 리스너에게 통보
6		서비스 관리자 : 사용자의 의도인 Why를 파악하고 검증한다.
5→7	Notify Changes	사용자는 의도된 서비스인지 검증한다.
5→8	Read Model	사용자 인증을 받는다.
8→2	Read Model	사용자는 요청한 서비스를 조회한다.
2→4	Read Model	사용자는 요청한 서비스를 승인한다.

지하고 3장 3.1.1절에 있는 <그림 3-2>의 컨텍스트 해석기에게 보고하고, 컨텍스트 해석기는 이를 서비스매니저에게 보고하며 간단하게 아침에는 스위칭제어를 실시한다. 만약 <그림 3-18>과 같이 오후에 스위칭제어로 간단하게 수행할 수 없는 경우, 서비스매니저는 집주인에게 조명기기, A/V가전(TV), 정보가전(PC)의 인증을 요청하고, 네트워크 카메라를 통하여 두자녀의 역할과 의도를 분석하여 OSGi 기술규격 문서에서 외부에서 내부로 가는 5가지 통신기법에 따라 상황충돌을 처리한다.

① 제어 : “전원을 켜라” 같은 운영명령어

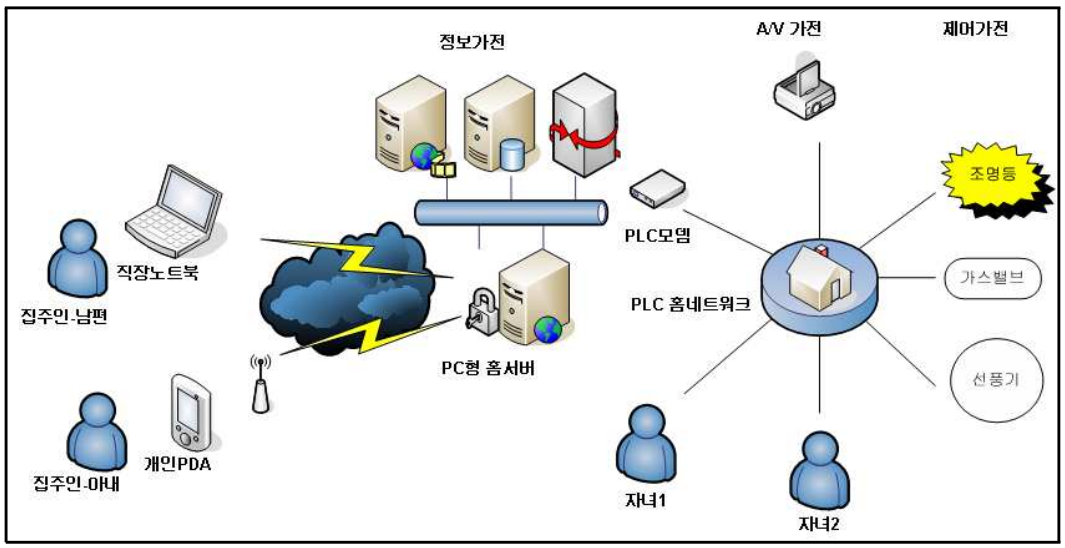
② 질의 : 특정 질의에 응답을 요구하는 통신

③ 비동기적 : 이벤트 통지 제공

④ 발견 : 기기들이 기기나 서비스가 요구한 프로파일에 맞는지 인지하도록 허용

⑤ 미디어스트리밍 : 세션기반의 통신

정상적으로 처리되었다는 것을 서비스 매니저는 집주인에게 알리고 집주인이 상황에 따라 거실의 A/V가전(TV), 정보가전(PC)의 운영을 위해 컨텍스트 충돌관리자에게 해당 자원에 대한 예약을 요청한 후 해당 응용프로그램을 호출하여 이러한 이벤트를 작동시킨다.



<그림 3-18> 외출모드 시스템 개념도

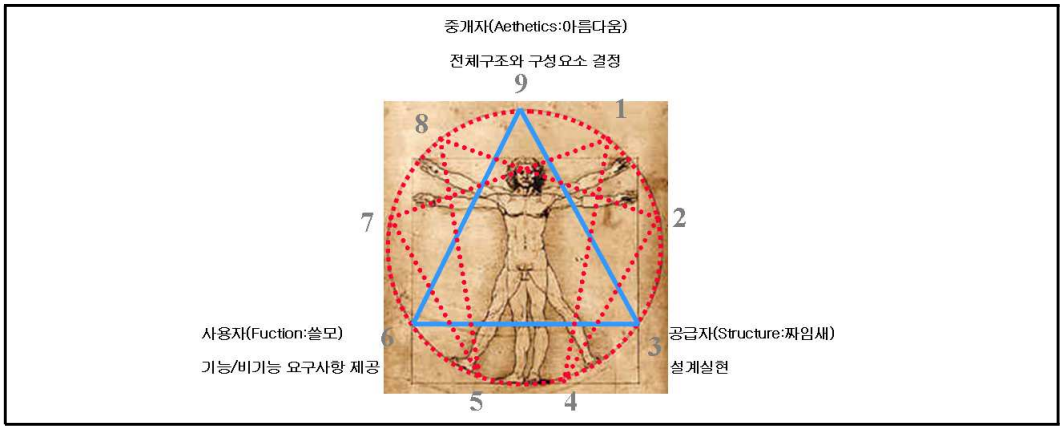
<Fig. 3-18> Outdoor mode system concept diagram

제 4 장 서비스지향 아키텍쳐의 평가

4.1 평가 방법론

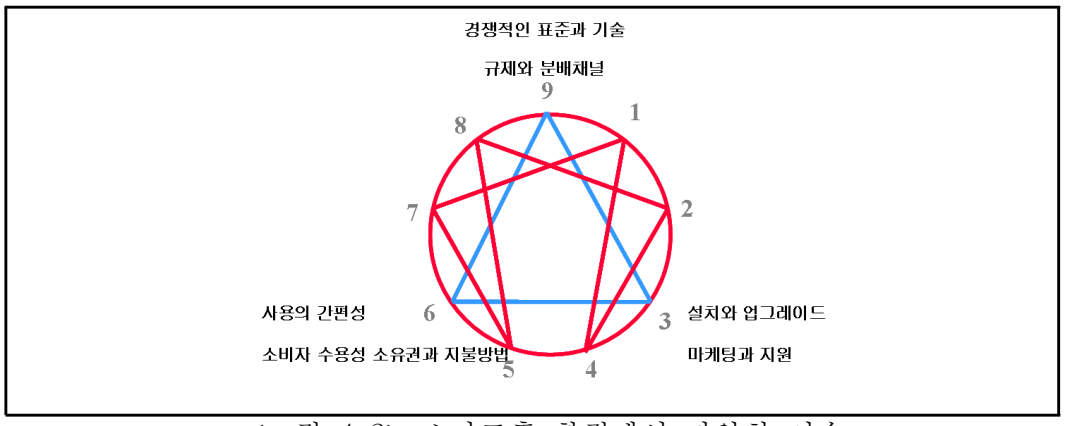
스마트홈에서 요구하는 표준서비스 시나리오로부터 조명관리 단위서비스와 가전제어 단위서비스, 가스밸브 방재관리 단위서비스 등 총 3가지의 단위서비스를 선택하여 이를 외출모드의 표준시나리오로부터 에니어그램을 이용하여 재설계하였다. 앞장에서 설계된 엔터프라이즈 서비스지향 아키텍쳐의 설계를 ATAM평가방법론으로 검증하기 위해 실제 프로젝트 보다는 POC(Proof Of Concept)형태의 프로젝트로서 단계 0에서 시작한다. ATAM을 이용하여 본 논문에서 제안한 가설에 대한 평가를 위해 스마트홈 관련 전문가 3명 및 소프트웨어 공학 관련 전문가 2명의 도움으로 문헌검토를 시작으로 2006년 8월 14일부터 2006년 12월 9일까지 9회에 걸쳐 브레인스토밍을 수행하면서 이들을 각각 서비스 중개자, 서비스 사용자, 그리고 서비스 제공자(논문저자)의 스테이크홀더로 선임하고 가상회의를 실시하였다.

이 회의에서 서비스 제공자는 부록의 ATAM 평가방법론에 따라 절차와 아키텍처를 소개하였고, 서비스 사용자와 함께 초기 품질속성 요구사항과 시나리오를 설명하였다[19,31,33,35].



<그림 4-1> ATAM 스테이크홀더의 역할

<Fig. 4-1> Role of ATAM stakeholder



<그림 4-2> 스마트홈 환경에서 다양한 이슈

<Fig. 4-2> Issue of smart home environment

4.1.1 스텝 1: 수정된 ATAM 소개

서비스 제공자는 품질속성간의 상충성을 분석하기 위하여 해당 문헌조사와 스테이크홀더간의 충분한 협의를 통하여 CBAM에서 이용하는 품질속성의 반응값을 수립하여 수정된 ATAM을 설명해주었고 스테이크홀더들과 방법론, 평가 산출물, 평가 목표에 대한 질의응답시간을 가졌다.

4.1.2 스텝 2: 비즈니스 드라이버 소개

서비스 사용자는 비즈니스 드라이버(Business driver)를 소개하였다. 여기서는 시스템이 지원해야 하는 단일세대 또는 다세대의 스마트홈 상황에 대한 정보와 상황 특유의 요구사항이 비지니스 드라이버를 결정짓는 요인이 되었다. 스마트홈에서 요구하는 표준서비스 시나리오로부터 조명관리 단위서비스와 가전제어 단위서비스, 방재관리 단위서비스 등 총 3가지의 단위서비스를 선택하였다. 비지니스 드라이버를 소개하는 자리에서는 <그림 4-2>와 같은 스마트홈 환경에서의 다양한 이슈들을 성능 목표, 가용성, 유연성 등 앞으로 따라야 될 하드웨어와 소프트웨어 기준에 대한 요구사항(사실상 제약조건)과 연관하여 개략적인 설명이 있었다.

4.1.3 스텝 3: 아키텍처 소개

다음으로는 서비스 제공자가 아키텍처를 소개하였고, 서비스 사용자와 함께

프로젝트 초기에 도출한 품질속성 요구사항과 그것들의 초기 시나리오에 대해서 토론하면서 본 프로젝트의 CSF(Critical Success Factor)를 고민하였다. 즉, OSGi의 요구사항 분석과 시나리오와 서비스도출을 위한 아키텍처를 판단하기 위하여 OSGi 시장 요구사항 분석과 스마트홈 가치사슬로 사업자를 각각 서비스 제공자, OSGi 운영자, 서비스 애그리게이터, 게이트웨이 판매자, 로컬 네트워크 제공자 및 운영자, 엑세스 네트워크 제공자, 인터넷 서비스 제공자, 지불 제공자 등 다양한 스테이크홀더를 고려하여 개방형 게이트웨이의 가치를 판단하였다. 또한 OSGi의 기능적 요구사항을 프레임워크 기능요구사항, 운영관리기능요구사항, 클라이언트 엑세스 기능 요구사항, 데이터 관리 기능요구사항, 자원관리기능요구사항, 기기관리기능 요구사항 등을 고려하여 설명하였다. 전형적인 MVC 아키텍처 구조에서 얼마나 충실하게 확장가능한 형태로 구성하였는지도 토론하였다.

4.2 조사 및 분석

4.2.1 스텝 4: 아키텍처 접근법 식별

본 스텝에서 서비스 제공자는 본 프로젝트의 CSF를 시스템의 유연성(Agility), 성능(Performance), 가용성(Availability)의 목표로 제안하고 설명하였으며 시스템은 앞장에서 설계한 것을 근거로 부록에서 보는 바와 같이 PLC기반에서 웹서비스로 구성되어 있다. 여기서 하드웨어 아키텍처와 프로세스 아키텍처상에서 시스템의 성능 속성에 영향을 끼친다고 가정하겠다. 이를 근거로 도출된 아키텍처 접근법으로는 다음과 같다.

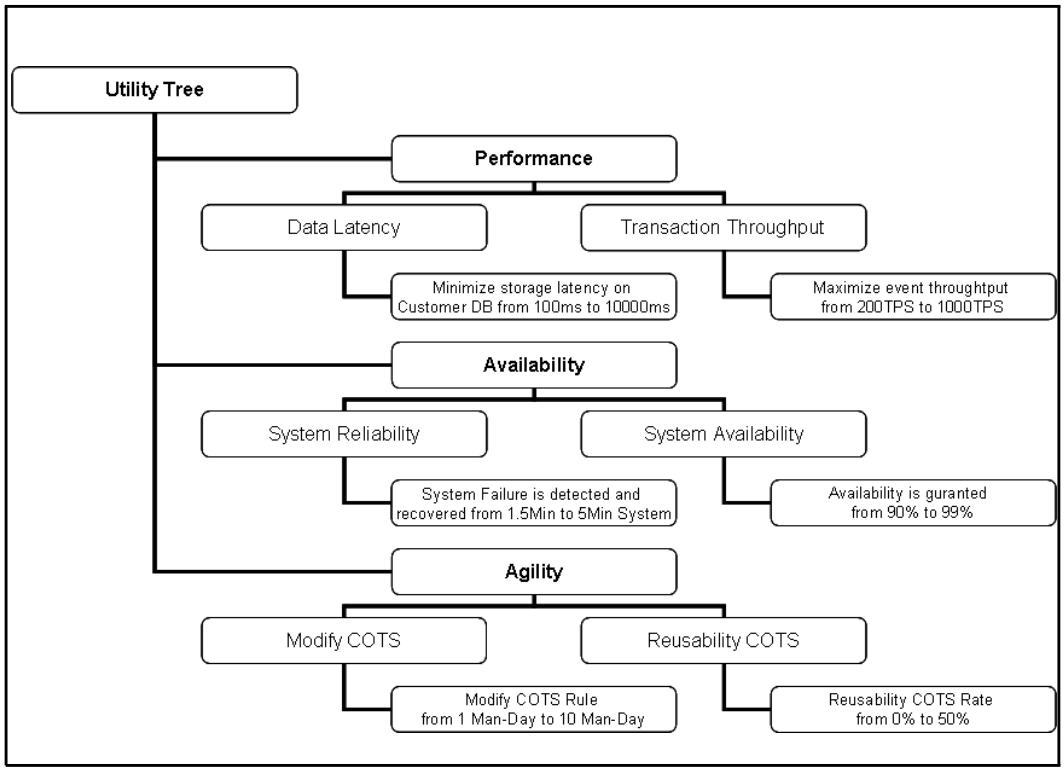
(1) (시스템의 성능 목표를 달성하기 위한) OSGi 기술표준규격에 충실하게 서로 독립적으로 통신하는 z256 PLC 번들에서 신속한 서비스 발견을 위해 최대평균 처리율과 데이터 응답속도의 범위설정 접근.

(2) (시스템의 가용성 목표를 달성하기 위한) 정확한 서비스전달을 위해 시스템 실패시의 복구시간과 보장된 총 가용시간 범위설정 접근.

(3) (시스템의 유연성 목표를 달성하기 위한) MVC 디자인패턴 기반에서 이클립스 및 프로시스트의 COTS(Commercial Off-The-Shelf) 번들의 유연한 서비스조립을 위해 변경용이성과 재사용성의 범위설정 접근.

4.2.2 스텝 5: 품질속성 유틸리티 트리 만들기

평가에 참여한 스테이크홀더들은 유연성과 성능을 가장 중요하게 생각한다고 말했다. 비지니스 드라이버에 정의되었던 것처럼 평가 초기에는 이러한 속성들이 가장 큰 관심사항이었으나 면밀한 조사를 통해서 가용성 또한 매우 중요하다는 것을 알게 되었다. 초기에 스테이크홀더들이 진술한 내용과 브레인스토밍 활동을 기반으로 <그림 4-3>과 같이 본 논문의 유틸리티 트리를 KPI(Key Performance Index) 시나리오로 만들었다. 유틸리티 트리 도출 프로세스의 일환으로 유틸리티 트리에 있는 각각의 시나리오가 특정한 자극(Stimulus)과 그것과 관련한 반응(Response)을 가지고 있음을 확인하였다.



<그림 4-3> 도출된 품질속성 유틸리티 트리

<Fig. 4-3> Etracted QA Utility Tree

시나리오번호	품질속성	포함품질속성
#9	성능	
#1	(복잡성)	처리율
#8		응답속도
#3	가용성	가용성
#4	(불확실성)	
#2		신뢰성
#6	유연성	
#7	(변화대응성)	재사용성
#5		변경용이성

<그림 4-4> 품질속성 에니어그램

<Fig. 4-4> QA Enneargram

<표 4-1> 유틸리티 트리의 KPI 시나리오

<Table 4-1> KPI Scenario of Utility Tree

품질속성	포함품질속성	KPI 시나리오
성능	처리율	#1 이벤트 처리율을 200TPS에서 1000TPS까지 최대화한다.
성능	응답속도	#8 고객DB 응답속도는 100ms에서 10,000ms 까지 최소화한다.
가용성	가용성	#3 시스템 가용성은 90% 에서 99%이상까지 보장되어야 한다.
가용성	신뢰성	#2 시스템 실패시 감지하고 복구하는 시간은 1.5분 에서 5분 이내로 보장되어야 한다.
유연성	재사용성	#7 COTS 번들의 재사용율은 0% 에서 50%이상은 되면 만족한다.
유연성	변경용이성	#5 COTS 번들의 변경시 1 Man-Day 에서 10 Man-Day이내가 되면 만족한다.

<표 4-2> 유틸리티 트리의 KPI 시나리오의 분석결과

<Table 4-2> Analysis result of KPI scenario of Utility Tree

Analysis Output	포함품질속성	KPI 시나리오 설명
위험성	가용성	#3 분산객체 환경에서 시스템 가용성 문제 발생 가능
민감성	응답속도	#8 개별적인 트랜잭션에 대한 실행성능 문제 발생가능
상충성	변경용이성	#5 프로세스나 룰의 변경이 많을 경우 코드 수정과 배포에 따른 장애 발생 가능성
	재사용성	#7 인터페이스 대상 장비가 많을 경우 여러 시스템간의 호환성과 재사용성 문제발생가능성

4.2.3 스텝 6: 아키텍처 접근법 분석

이 단계에서 앞서 식별한 몇가지 아키텍처 접근법들을 포함한 아키텍처 참고 문헌은 가지고 있었지만, 스마트홈의 컨텍스트 충돌을 관리하기 위하여 아키텍처 접근법에서 검증하고 있는 구체적인 방법에 대해서는 스테이크홀더 상호간에 제대로 인지하지 못했다. 그래서 서비스 제공자는 접근법에 대한 이해를 돕고 접근법들이 가지고 있는 위험성, 민감성, 상충성에 대한 조사를 하기 위하여 품질 속성 특성으로부터 <표 4-2>와 같이 분석하였다.

4.3 시험 및 보고

4.3.1 스텝 7: 퍼지를 이용한 시나리오 분석

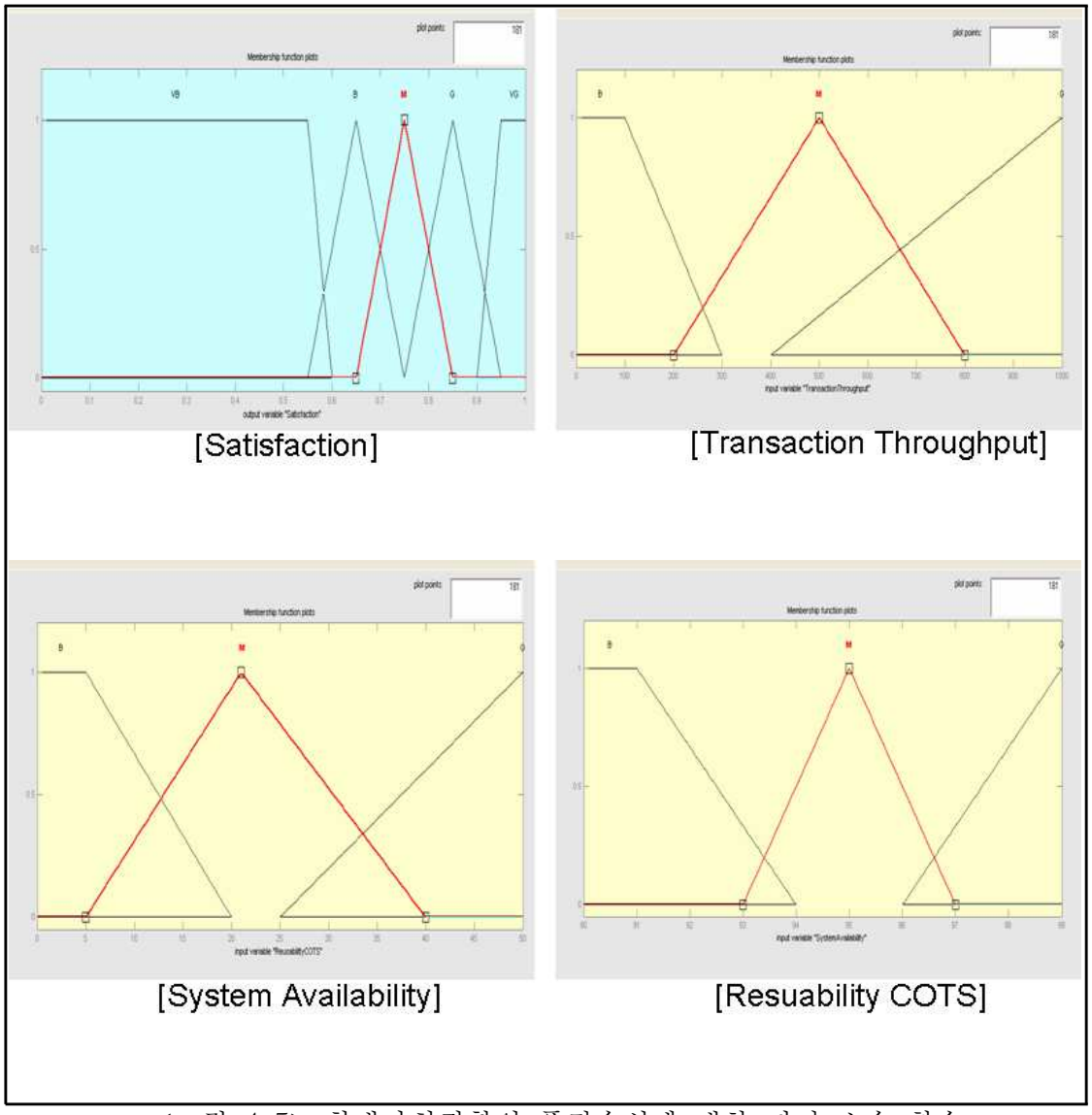
품질속성 유틸리티 트리로부터 도출된 KPI 시나리오는 아키텍처가 기능적인 요구사항을 충족시키는지 확인하는데 쓰일 뿐만이 아니라 시스템의 아키텍처 접근법을 보다 면밀히 이해하고 이들 접근법이 성능, 가용성, 유연성 등의 품질 속성 요구사항을 충족시키는지를 알아보는데 쓰인다. 본 논문에서는 차별화된 품질속성간의 상충성을 분석하기 위하여 해당 문헌조사와 스테이크홀더간의 충분한 협의를 통하여 CBAM에서 이용하는 <표 4-3>의 품질속성의 반응값을 수립하였다. 이를 최대가치의 반응값 집합과 최소가치의 반응값 잡합을 분리하

여 퍼지규칙에 적용하였으며 이러한 품질속성간의 조합으로 <그림 4-5>에서 <그림 4-10>까지 서비스 사용자의 만족도를 MATLAB을 이용하여 퍼지시뮬레이션하였다. 스마트홈 시스템의 ATAM 평가를 위한 시나리오는 순환 브레인스토밍 활동을 통해서 수집하였다.

<표 4-3> 유틸리티 트리의 KPI 시나리오의 반응값

<Table 4-3> Reaction of KPI scenario of Utility Tree

가치집합	시나리오#	포함품질속성	Bad	Medium	Good
최대가치집합	#1	처리율	200TPS	400TPS	1,000TPS
			10	70	100
	#3	가용성	90%	95%	99%
			10	80	100
	#7	재사용성	0%	20%	50%
			0	70	100
최소가치집합	#8	응답속도	10,000ms	250ms	100ms
			10	70	100
	#2	신뢰성	5Min	3Min	1.5Min
			10	80	100
	#5	변경용이성	10 M/D	2.5 M/D	1 M/D
			10	70	100



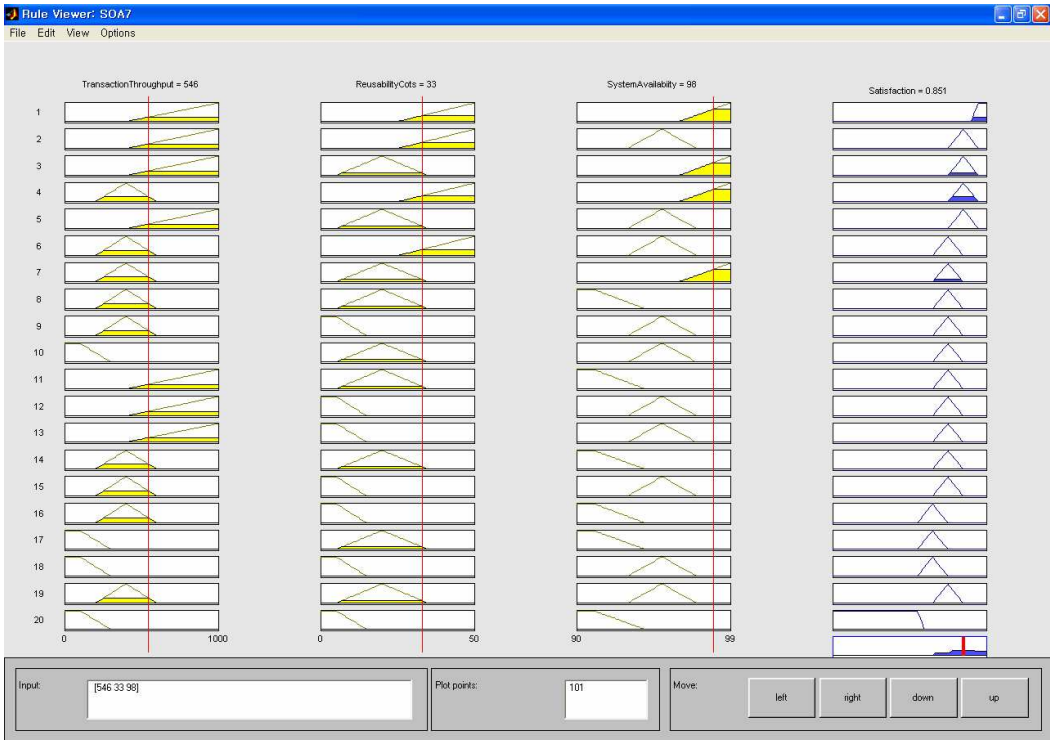
<그림 4-5> 최대가치집합의 품질속성에 대한 퍼지 소속 함수

<Fig. 4-5> Fuzzy membership function for QA of maximum value set

1. If (TransactionThroughput is G) and (ReusabilityCOTS is G) and (SystemAvailability is G) then (Satisfaction is VG) (1)
2. If (TransactionThroughput is G) and (ReusabilityCOTS is G) and (SystemAvailability is M) then (Satisfaction is G) (1)
3. If (TransactionThroughput is G) and (ReusabilityCOTS is M) and (SystemAvailability is G) then (Satisfaction is G) (1)
4. If (TransactionThroughput is M) and (ReusabilityCOTS is G) and (SystemAvailability is G) then (Satisfaction is G) (1)
5. If (TransactionThroughput is G) and (ReusabilityCOTS is M) and (SystemAvailability is M) then (Satisfaction is G) (1)
6. If (TransactionThroughput is M) and (ReusabilityCOTS is G) and (SystemAvailability is M) then (Satisfaction is M) (1)
7. If (TransactionThroughput is M) and (ReusabilityCOTS is M) and (SystemAvailability is G) then (Satisfaction is M) (1)
8. If (TransactionThroughput is M) and (ReusabilityCOTS is M) and (SystemAvailability is B) then (Satisfaction is M) (1)
9. If (TransactionThroughput is M) and (ReusabilityCOTS is B) and (SystemAvailability is M) then (Satisfaction is M) (1)
10. If (TransactionThroughput is B) and (ReusabilityCOTS is M) and (SystemAvailability is M) then (Satisfaction is M) (1)
11. If (TransactionThroughput is G) and (ReusabilityCOTS is M) and (SystemAvailability is B) then (Satisfaction is M) (1)
12. If (TransactionThroughput is G) and (ReusabilityCOTS is B) and (SystemAvailability is M) then (Satisfaction is M) (1)
13. If (TransactionThroughput is G) and (ReusabilityCOTS is B) and (SystemAvailability is M) then (Satisfaction is M) (1)
14. If (TransactionThroughput is M) and (ReusabilityCOTS is M) and (SystemAvailability is B) then (Satisfaction is M) (1)
15. If (TransactionThroughput is M) and (ReusabilityCOTS is B) and (SystemAvailability is M) then (Satisfaction is M) (1)
16. If (TransactionThroughput is M) and (ReusabilityCOTS is B) and (SystemAvailability is B) then (Satisfaction is B) (1)
17. If (TransactionThroughput is B) and (ReusabilityCOTS is M) and (SystemAvailability is B) then (Satisfaction is B) (1)
18. If (TransactionThroughput is B) and (ReusabilityCOTS is B) and (SystemAvailability is M) then (Satisfaction is B) (1)
19. If (TransactionThroughput is M) and (ReusabilityCOTS is M) and (SystemAvailability is M) then (Satisfaction is M) (1)
20. If (TransactionThroughput is B) and (ReusabilityCOTS is B) and (SystemAvailability is B) then (Satisfaction is VB) (1)

<그림 4-6> 최대가치집합의 품질속성에 대한 퍼지규칙

<Fig. 4-6> Fuzzy rule for QA of maximum value set



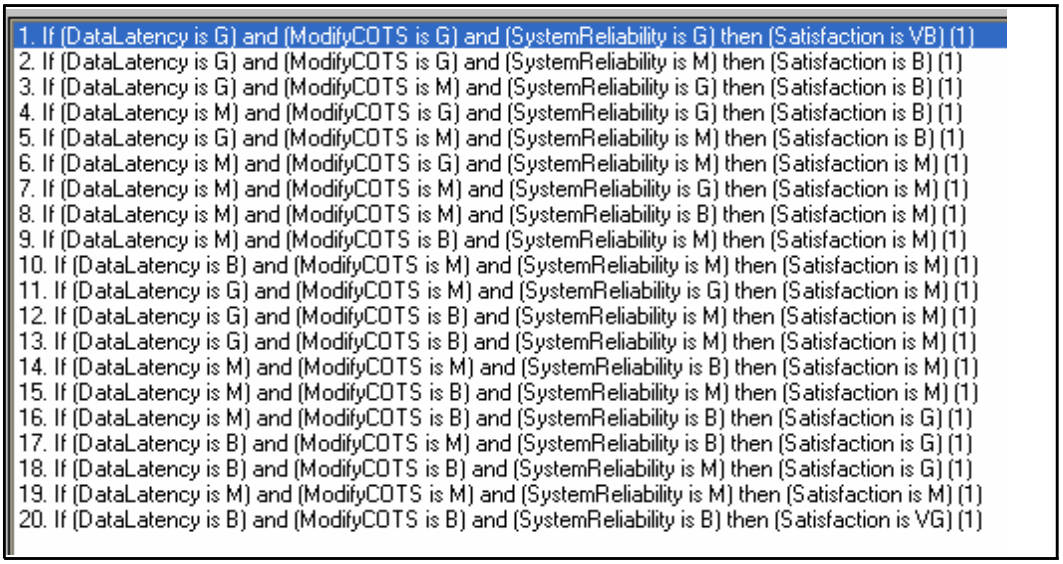
<그림 4-7> 최대가치집합의 품질속성에 대한 퍼지 추론과정

<Fig. 4-7> Fuzzy rule viewer for QA of maximum value set

<표 4-4> 최대가치집합의 퍼지규칙 만족도

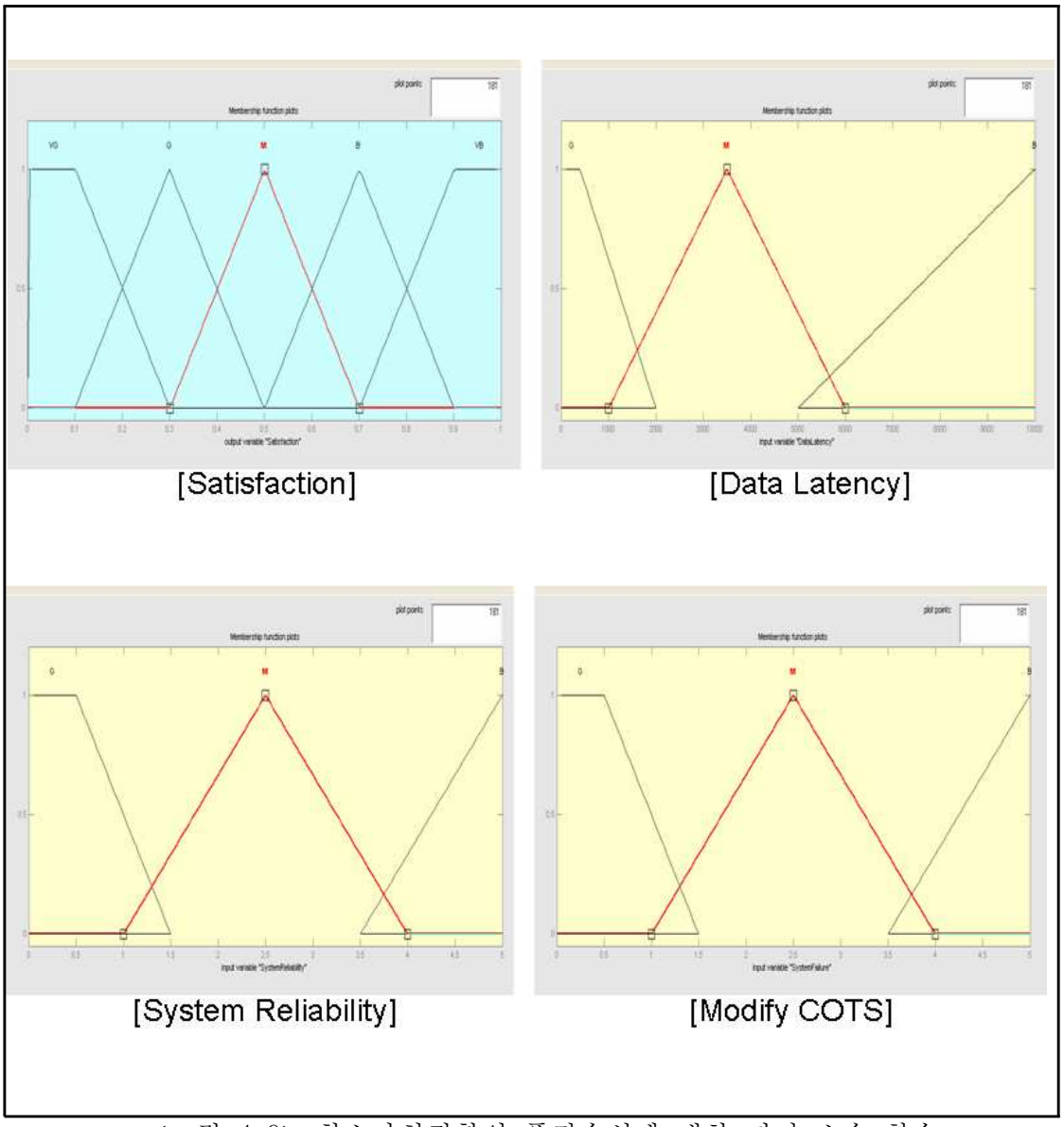
<Table 4-4> Fuzzy rule satisfaction for maximum value set

가치집합	시나리오#	포함품질속성	시뮬레이션
최대가치집합	#1	처리율	546TPS
	#3	가용성	98%
	#7	재사용성	33%
결론	-	만족도	85.1%



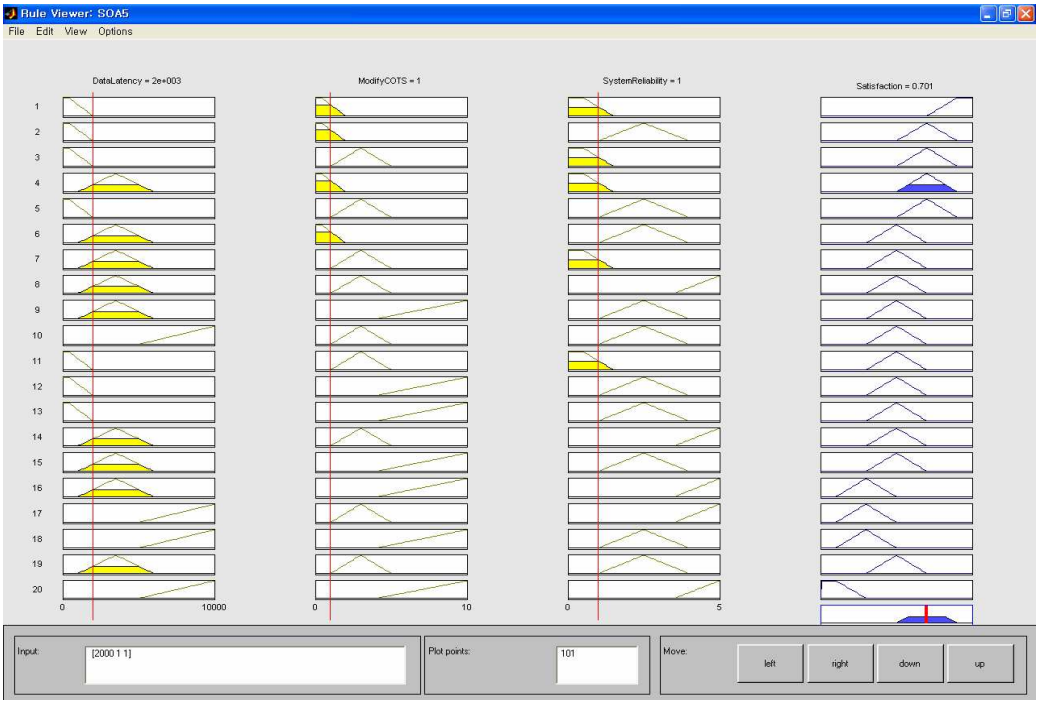
<그림 4-8> 최소가치집합의 품질속성에 대한 퍼지규칙

<Fig. 4-8> Fuzzy rule for QA of minimum value set



<그림 4-9> 최소가치집합의 품질속성에 대한 퍼지 소속 함수

<Fig. 4-9> Fuzzy membership function for QA of minimum value set



<그림 4-10> 최소가치집합의 퍼지 추론 과정

<Fig. 4-10> Fuzzy rule viewer for QA of minimum value set

<표 4-5> 최소가치집합의 퍼지규칙 만족도

<Table 4-5> Fuzzy rule satisfaction for minimum value set

가치집합	시나리오#	포함품질속성	시뮬레이션
최소가치집합	#8	응답속도	2000ms
	#2	신뢰성	1Min
	#5	변경용이성	1 M/D
결론	-	만족도	70.1%

단, 퍼지만족도는 70.1%는 임의의 퍼지평균만족도인 50.1%를 상회하였슴.

4.3.2 스텝 8: 아키텍처 접근법 분석

본 단계에서 부록의 시험 단계를 거쳐 서비스 제공자는 제안된 에니어그램을 이용한 MVC 아키텍처 접근법과 성능, 가용성, 유연성 간에 관련성을 보다 깊게 조사하고 정제할 수 있었다.

4.3.3 스텝 9: 결과 보고

서비스 제공자는 아키텍처 결정 사항의 내포하고 있었던 성능이나 가용성 측면을 고려하고 있지 않았으며, 성능과 가용성은 스테이크홀더의 관심사항이 아니었기 때문에 비지니스 드라이버에 대해서 토의하는 동안에 대부분의 스테이크홀더들은 시스템의 유연성에 대해서만 고려하고 있었지만, 메시지 형태가 수시로 변경되더라도 스마트홈 컨텍스트 관리시스템이 서비스조립과 전달 등을 제대로 변화에 대응 해주기를 기대하고 있었다. 따라서 상충요소를 고려한 품질속성에 대한 시뮬레이션 결과 최대가치집합에서 85.1%, 최소가치집합에서 70.1%의 서비스 사용자의 만족도를 도출하는 포함품질속성을 추출하여 스마트홈에서 요구하는 표준서비스 시나리오로부터 제어가전을 위한 조명관리, 가전제어, 가스밸브 방재관리의 프로토타입 데모를 <부록그림 7>~<부록그림 14>에서 제시하였듯이 아래 <표 4-6>의 URL 웹서비스가 정상적으로 처리됨을 확인할 수 있다. 설계된 서비스지향 아키텍쳐는 정보가전, A/V가전의 디지털컨버전스에서 상황충돌이 발생하였을 때에도 효과적으로 대응할 수 있을 것이다.

<표 4-6> 프로시스트 z256http.jar 번들의 URL 서버렛 서비스

<Table 4-6> URL servelet service of prosyst z256http.jar

Layer	Prosyst jar Bundle	URL servelet service
View	z256http.jar	Ha_lamp_on.class, Ha_lamp_only.class, Ha_valbe_on.class,Ha_consent_on.class,
	PDASocketHander.jar	
Controller	commw32.jar	-
	connector.jar	-
Model	DLz256.jar	-
	z256.jar	-

제 5 장 결 론

본 논문에서는 OSGi 플랫폼기반에서 다양한 센서들로부터 수많은 상황인식 서비스가 동적으로 움직이게 되고 사용자 요구 및 필요성에 따라 신규서비스의 제공 및 기존 서비스의 변경과 제공된 서비스간에 데이터 공유, 서비스 라이프 사이클, 서비스분배의 효과적인 관리를 하고자 한다. 제안된 에니어그램을 이용한 서비스지향 아키텍쳐에서 서비스공급자는 다양한 센서들로부터 상대적인 서비스들을 통합하여 각각 서비스를 SOAP 메시지로 묶어서 웹서비스를 서비스 중개자의 UDDI서버에 등록하면, 서비스요청자는 UDDI서버에서 특정한 서비스를 검색하고, 서비스공급자에게 해당 SOAP메시지를 호출한다.

이를 위해 제안된 서비스지향 아키텍쳐는 1,000세대 이하 소규모 스마트홈 컨텍스트에서 3가지 이상 품질속성의 상충요소를 최대가치집합과 최소가치집합으로 분리하여 유틸리티 트리의 반응값에 대하여 MATLAB을 이용한 퍼지시뮬레이션 결과 최대가치집합에서 85.1%, 최소가치집합에서 70.1%의 서비스 사용자의 만족도에 해당하는 포함품질속성을 추출하였으며 스마트홈에서 요구하는 표준서비스 시나리오로부터 조명관리, 가전제어, 가스밸브 방재관리 방재관리의 프로토타입 데모를 제시하였고, 설계된 서비스지향 아키텍쳐는 정보가전, A/V가전의 디지털컨버전스에서 상황충돌이 발생하였을 때에도 효과적으로 대응할 수 있을 것이다.

많은 제약사항에도 불구하고 제안된 에니어그램을 이용한 서비스지향 아키텍쳐는 도메인범위별, 스테이크홀더별로 스마트홈에서 웹서비스를 통한 서비스 가상화를 통하여 관점에 따라 서비스지향 원칙인 관심의 분리가 가능하며, URL 웹서비스 객체에 대한 서비스 생명주기에 따라 서비스 제어권을 관리할 수 있었다. 또한 스마트 U헬스분야에서 아키텍쳐모델링을 통한 비즈니스모델링을 하는데 흥미있는 사상을 제공하여 주었으며, 스마트홈 다중 컨텍스트관리를 통한 U헬스는 HL7과 IHE분석을 통한 이클립스기반의 OHF를 제안된 에니어그램을 이용한 서비스지향 아키텍쳐에 플러그인함으로써 새로운 사업 모형 및 진료 모형을 제시할 수 있을 것이다.

참고문헌

[1] 김동철, “시나리오 중심의 Ubiquitous Home Network 구축방안,” 고려대학교 공학대학원 전자컴퓨터공학전공 석사학위논문 (2005)

[2] 김용, “유비쿼터스 홈네트워크 환경에서의 상황인식서비스 미들웨어 개발 및 연구,” 중앙대학교 대학원 전자전기공학과 석사학위논문 (2005)

[3] 남승좌, “유비쿼터스 컴퓨팅 환경의 역할 기반 접근제어에서 발생하는 컨텍스트 충돌,” 서강대학교 대학원 석사학위논문 (2003)

[4] 남창우, “지능형 홈네트워크환경에서의 상황인지기반 서비스 관리엔진 개발,” 중앙대학교 대학원 전자전기공학과 석사학위논문 (2005)

[5] 양수경, “홈네트워크상에서 IEEE1394 정보가전제어를 위한 UPnP 브릿지 구현,” 조선대학교 대학원 전자공학과 박사학위논문 (2002)

[6] 엄윤식, “상황인지기반 지능형 유비쿼터스 홈네트워크 서비스와 보안구현을 위한 추론엔진 개발,” 중앙대학교 대학원 전자전기공학과 석사학위논문 (2005)

[7] 이경모, “유비쿼터스 컴퓨팅 환경을 위한 OSGi 상의 서비스 이동 관리 시스템,” 인하대학교 대학원 컴퓨터 .정보공학과 석사학위논문 (2006)

[8] 이순자, “구르지예프, 베어 및 리소의 에니어그램 비교,” 창원대학교 대학원 교육학과 박사학위논문 (2003)

[9] 이재호, “유스케이스 모델링을 위한 시나리오 근간의 목표지향 분석 방안,” 서강대학교 대학원 컴퓨터학과 석사학위논문 (2000)

[10] 이지훈, “OSGi 서비스 플랫폼 기반의 컨텍스트 서비스 구조설계,” 한양대학교 대학원 석사학위논문 (2005)

[11] 이해준, 서대영, “OSGi Programming,” (주)프로시스트테크놀로지코리아, 도서출판 토비스 (2006)

[12] 이충호번역, 마이클슈나이더원저, “자연, 예술, 과학의 수학적원형,” 경문사, p.316-322 (2004)

[13] 안강범외 4인 번역, 짐 단주외 4인 원저, “자바 개발자를 위한 이클립스 바이블,” 피어슨 에듀케이션 코리아 (2005)

[14] 임철홍, 홍도석, 최정준, "ESB기반 SOA Application에 대한 S/W Architecture 관점의 평가와 개선방안에 대한 연구," 한국IT서비스학회, Volume 5, Issue 2 (2006)

[15] 장세영외 3인 번역, 토마스 얼 원저, “SOA 개념에서 설계, 구현까지,” 에이콘 (2006)

[16] 옥상훈, “예제로 배우는 Adobe 플렉스 2,” 에이콘 (2006)

[17] 전진형, “스마트홈 환경에서의 다중 컨텍스트관리와 서비스제어,” 중앙대학교 대학원 제어계측학과 석사학위논문 (2004)

[18] 정병국, “에니어그램 교육 프로그램이 자아존중감과 학업성취목표에 미치는 영향 -사상체질과 비교연구-,” 창원대학교 대학원 대체의학과 석사학위논문 (2005)

[19] 홍태기, “아키텍쳐 트레이드오프 분석 방법을 지원하는 웹기반의 시나리오 관리도구의 구현,” 서강대학교 대학원 컴퓨터학과 석사학위논문 (2000)

[20] 최봉균, “엔터프라이즈 환경아키텍쳐를 지원하는 요구사항 분석 프로세스에 관한 연구,”연세대학교 대학원 컴퓨터과학 .산업시스템공학과 석사학위논문 (2002)

[21] Sang Chul *A*hn, "A SOA-based Context Managent Middleware Architecture," The Graduate School Yonsei University Depart. of Computer Science Master of Science (2005).

[22] *A*rnon Rotem-Gal-Oz, "SPAMMED Architecture Framework (SAF)," http://www.rgoarchitects.com/saf/ (2006).

[23] de *A*lmeida D.R., de Souza Baptista, C., da Silva, E.R., Campelo, C.E.C., de Figueiredo, H.F., Lacerda, Y.A. "A context-aware system based on service-oriented architecture," Advanced Information Networking and Applications (2006).

[24] Giacomo *C*abri, Luca Ferrari, Letizia Leonardi, "Rethinking Agent Role : Extending the Role Definition in the BRAIN Framework," IEEE

International Conference on SMC '04 (2004).

[25] Jeromy *C*arriere, "Lightweight Architecture Alternative Assessment Method(LAAAM)", http://blogs.msdn.com/jeromyc/archive/2005/08/27/457081.aspx (2006).

[26] Guanling *C*hen and David Kotz. "A survey of context-aware mobile computing research," Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College (2000).

[27] S.W. *C*hoi, A.S. Oh, O.H. Kwon, S.H. Kang, S.G. Hong, H.R. Choi, " Study on Context-Aware SOA based Open Service Gateway initiative platform," Journal of the Korean Institute of Maritime Information & Communication Sciences, Vol. 10, N0. 11., (2006).

[28] A.K. *D*ey, D. Salber, M. Futakawa and G. Abowd, "An architecture to support context-aware applications," UIST '99 (1999).

[29] *D*irk Krafzig, Kal Banke, Dirk Slama, "Enterprise SOA," Pretice Hall PTR (2004).

[30] Walter J. *G*eldart, M. Eng., M. Div.,"Fuzzy Information, Man, And The Enneagram," IEEE Systems, Man, and Cybernetics, 'Computational Cybernetics and Simulation'. 1997 IEEE International Conference on Volume 1, 12-15 , p.657 - 662 (1997).

[31] Nam-Ki *K*im, Kyung-Chul Chae, "One the Discrete-Time Version of the Distributional Little's Law," Journal of the Korean Institute of Industrial Engineers. Vol.27, No.4., (2001).

[32] *K*orkea-aho, M., "Context-Aware Applications Survey," Department of Computer Science, Helsinki University of Technology (2000).

[33] Won Young *L*ee, "Mathematical Approach to Performance Analysis for Web-based Enterprise System," JavaService Consulting, http://www.javaservice.net (2005).

[34] Gu, T., *P*ung, H.K., Zhang, D.Q., "Toward an OSGi-based infrastructure for context-aware applications," Pervasive Computing, IEEE, Volume 3, Issue 4 (2004).

[35] *R*ick Kazman, Mark Klein, Paul Clements, "ATAM^{SM} Method for Architecture Evaluation," Carnegie Mellon Software Engineering Instituete (2000).

[36] Bill *S*chilit, Norman Adams and Roy Want, "Context-aware computing applications," In proceedings of IEEE Workshop on Mobile Computing Systems and Applications (1994).

[37] A. *S*hehzad, H. Q. Ngo, K. A.Pham, S.Y. Lee, "Formal Modeling in Context-Aware Systems," KI-04 Workshop (2004).

[38] Gerhard *W*eiβ, Michael Rovatsos, Matthias Nickles, "Capturing Agent Autonomy in Roles and XML," ACM International Conference on AAMAS'03 (2003).

[39] *Y*ang, S.J.H. Lan, B.C.W. Chung, J.-Y., "A new approach for context aware SOA," e-Technology, e-Commerce and e-Service, IEEE '05. Proceedings (2005).

[40] *O*SGi, .http://OSGi.org, http://www.prosyst.co.kr/

[41] *S*OA 개요, http://www.jeongsik.name

[42] 기사검색어 : *S*OA, 가상화현상 등, http://www.zdnet.co.kr

[43] eclipse *S*TP, http://www.eclipse.org/STP, http://www.eclipse.org/equinox

[44] 용어검색어 : MVC 등, http://www.*t*erms.co.kr

[45] MVC, http://www.dossier-andreas.net/software_architecture/mvc.html

[46] *U*PnP, http://www.upnp.org

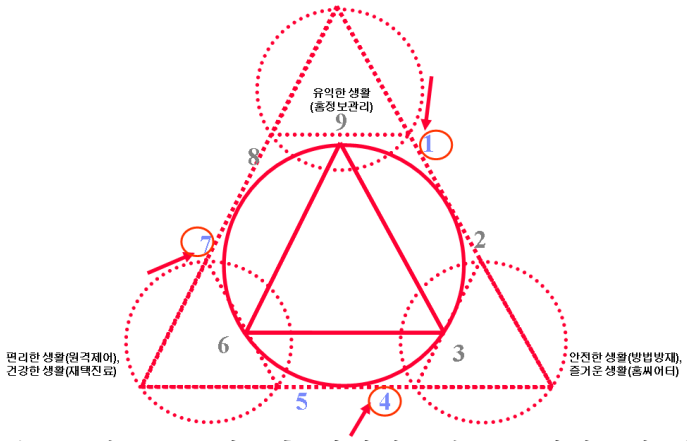
[47] Software Architecture, http://www.cs.cmu.edu/afs/cs/project/able/ftp/intro_softarch/intro_softarch.pdf

[48] RNS.8core Framework, http://8core.com

[49] PAC.xidsys Framework, http://xidsys.com

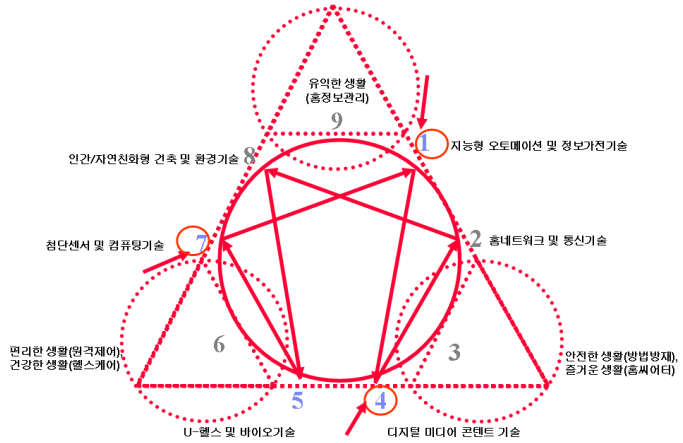
부 록

1. 스마트홈 개요



<부록그림 1> 스마트홈 산업의 5대 주요서비스의 분류

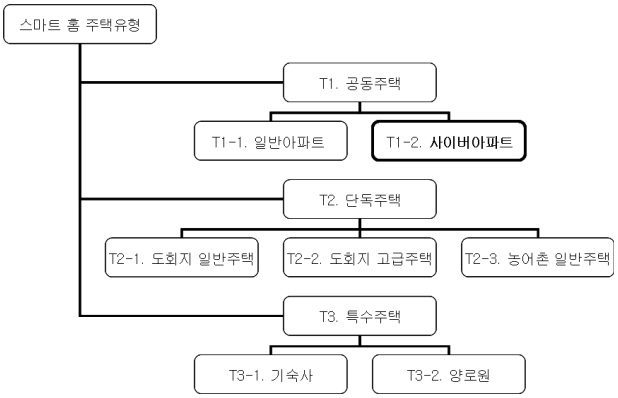
<Fig. 1> Classification of the five main service of smart home industry



<부록그림 2> 스마트홈 산업의 6대 핵심기술의 분류

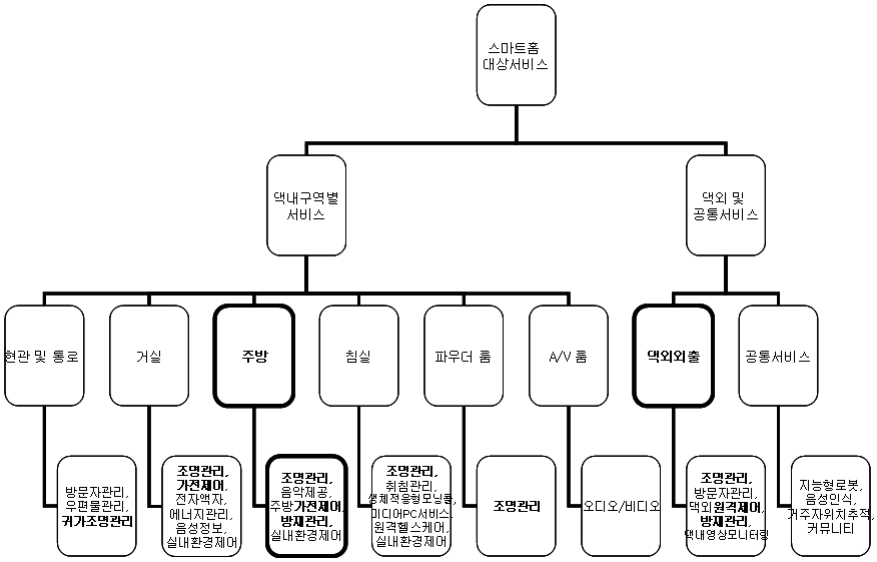
<Fig. 2> Classification of the six core service of smart home industry

지능형 스마트홈 산업을 주요서비스와 핵심기술로 분류하면 <부록그림 1, 2>와 같다.



<부록그림 3> 스마트홈의 주택유형

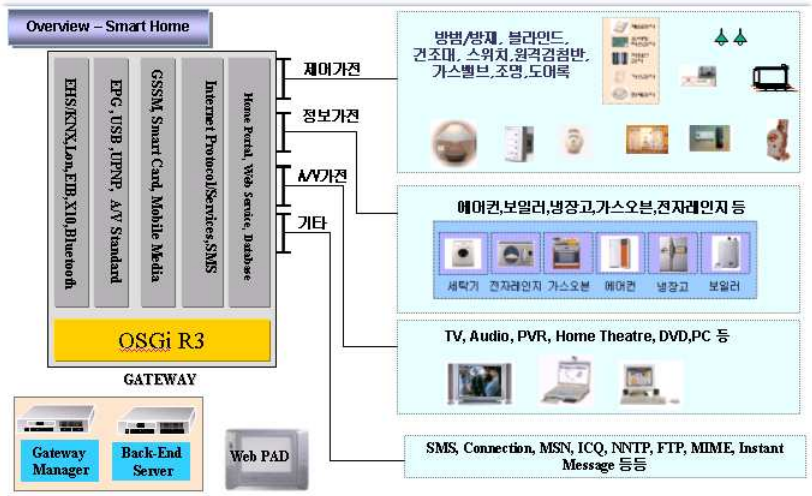
<Fig. 3> House type of smart home



<부록그림 4> 스마트홈의 구역별 대상서비스 분류

<Fig. 4> Classification of Object service of smart home by zone

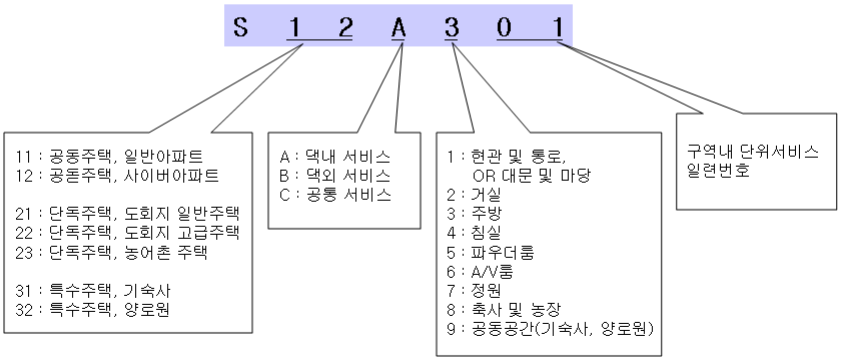
스마크홈 대상 주택유형과 대상서비스를 분류하면 <부록그림 3, 4>와 같다.



<부록그림 5> 스마트홈의 가전기기별 대상 서비스별 분류

<Fig. 5> Classification of Object service of smart home by home appliance

구분	단위 서비스	단위 서비스 정의 및 설명	사이버용 서비스코드
주방 (A3)	1. 조명관리 - 조도 조절 & 켈빈 제어	1. 조명관리 거주자가 식단(메뉴)에 따라 조명 기구의 조도와 켈빈을 조절하여 무드를 설정하는 서비스	S12A301 : 조명관리(주방) - 리모콘, PDA, 터치스크린, 매뉴얼로 조작 - 조도 및 켈빈 제어



• 본 자료는 경상남도 지역산업진흥사업의 지능형 홈 중점육성 대상서비스의 분류코드에서 일부인용하였음.

<부록그림 6> 스마트홈 서비스 코드의 샘플

<Fig. 6> Sample of smart home service code

스마트홈의 가전기기 제어서비스별 분류와 분류코드는 <부록그림 5, 6>과 같으며, 주방모드와 외출모드의 표준시나리오는 <부록표 1, 2>와 같다.

<부록표 1> 주방모드 표준 단위서비스 시나리오

<Table 1> Kichen mode standard scenario

구분	단위 서비스	단위서비스 정의 및 설명	사이버용 서비스 코드
주방 (A3)	**1. 조명관리** **- 조도 조절 &** **캘빈 제어**	**1. 조명관리** **거주자가 식단(메뉴)에 따라 조명 기구의** **조도와 캘빈을 조절하여 무드를 설정하는 서비스**	**S12A301 : 조명관리(주방)** **- 리모콘, PDA, 터치스크린, 매뉴얼로 조작** **- 조도 및 캘빈 제어**
	2. 음악제공	2. 음악제공 식사 중 음악을 듣고 싶을 때 음성으로 음악을 요청하면 음성인식 시스템과 연계된 오디오시스템이 음악을 제공하는 서비스	S12A302 : 음악제공(주방) - 네트워크 오디오시스템+음성 - 음악 제공
	3. 가전기기 제어 및 요리정보 - 생활 가전 모니터링 및 제어	3. 가전기기 제어 및 요리정보 음성, PDA, 디지털 TV 및 터치스크린 등을 통해서 전자레인지, 가스오븐레인지, 세탁기, 에어컨 등 생활 가전을 제어하거나, 상태를 모니터링 또는 요리정보를 제공하는 서비스	S12A303 : 가전기기 제어 및 요리정보(주방) - 음성, PDA, 디지털 TV 및 터치스크린 - 생활 가전 제어 및 상태 모니터링
	4. 방재관리 **- 가스누출, 화재 감지 및 진화**	**4. 방재관리** **주방 내에 LPG 혹은 LNG가 누설되는 것이 감지되면, 가스밸브를 자동으로 차단하고, 열감지 센서가 화재를 감지하면 자동식 소화기가 작동하여 화재를 진화하는 서비스**	**S12A304 : 방재관리(주방)** **- 자동식 소화기로 감지 및 대처** **- 가스누출, 화재감지 및 진화**
	5. 실내온도 및 환경관리 - 실내 온도 및 공기오염 관리	5. 실내온도 및 환경관리 벤틸레이션 시스템, 냉난방 시스템, 전동 커튼 및 전동 창호가 연계되어 실내 온도와 공기를 최적의 상태로 자동 조절할 수 있는 서비스	S12A305 : 실내온도 및 환경관리(주방) -벤틸레이션, 냉난방 시스템, 전동 커튼, 전동 창호 - 실내온도 및 환경관리

• 본 자료는 경상남도 지역산업진흥사업의 지능형 홈 중점육성 대상서비스에서 일부인용하였음.
• 본 논문에서는 조명관리, 방재관리를 일부 구현하였음.

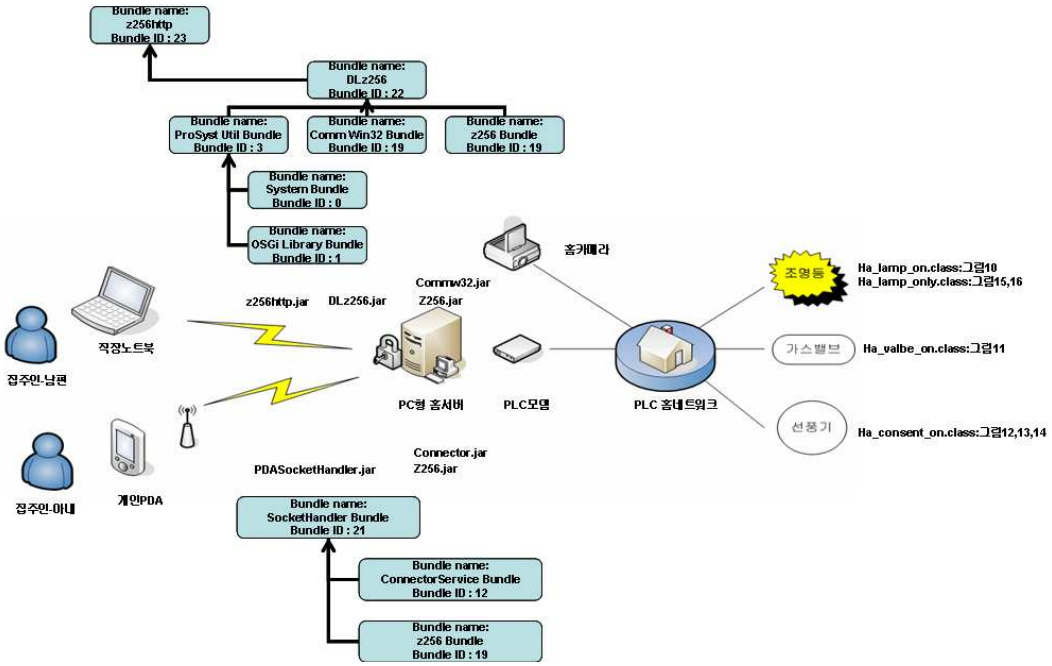
<부록표 2> 외출모드 표준 단위서비스 시나리오

<Table 2> Outdoor mode standard scenario

구분	단위 서비스	단위서비스 정의 및 설명	사이버용 서비스 코드
댁외 외출 (B7)	1. 방문자관리 - 방문자 영상 저장 및 원격 통화	1. 방문자관리 방문자가 현관폰을 누르면 댁내에 있는 비디오폰과 연계된 홈 서버가 자동으로 방문자의 동영상을 저장하고 비디오폰에 내장 되어 있는 전화 시스템은 거주자에게 전화를 걸어 방문자와 통화를 가능하게 해주는 서비스	S12B701 : 방문자관리(댁외) - 현관폰+비디오폰+홈서버 - 방문자 영상 저장 및 원격 통화
	2. 조명관리 **- 원격 조명 제어**	**2. 조명관리** **원격으로 댁내 전체조명을 On/Off 할 수 있는 서비스**	**S12B702 : 조명관리(집안전체:댁외)** **- 인터넷, 인터넷폰** **- 집안전체 조명 원격 On/Off**
	3. 네트워크 가전기기 댁외 원격제어 - 네트워크 가전기기 원격제어	3. 네트워크 가전기기 댁외 원격제어 냉난방기를 포함 한 생활가전기기를 원격에서 On/Off 제어 뿐만 아니라 모든 기능을 제어 할 수 있는 서비스	S12B703 : 네트워크 가전기기 댁외 원격제어(집안전체:댁외) -인터넷, 인터넷폰 -네트워크 생활가전 전기능 원격제어 및 모니터링
	4. 방재관리 **- 원격 방재 관리 및 이벤트 통보**	**4. 방재관리** **방재 관리 필요성을 인식했을 경우 원격에서 방재관련 기기를 제어 하며, 이벤트 발생시 관리실 및 원격지에 있는 거주자에게 이벤트를 통보하는 서비스**	**S12B704 : 방재관리(집안전체:댁외)** **-인터넷, 인터넷 폰** **- 원격 방재 관리 및 이벤트 통보**
	5. 댁내영상 모니터링 - 웹 카메라를 통한 집안을 영상 모니터링	5. 댁내영상 모니터링 웹 카메라를 통해 자신이 보고자 하는 구역을 모니터링 할 뿐만 아니라 제어하고자 하는 기기를 영상 모니터링 하며 제어 할 수 있는 서비스	S12B705 : 댁내영상 모니터링 (집안전체:댁외) -인터넷, 인터넷 폰 -집안 상태 및 가전기기 영상 모니터링 제어

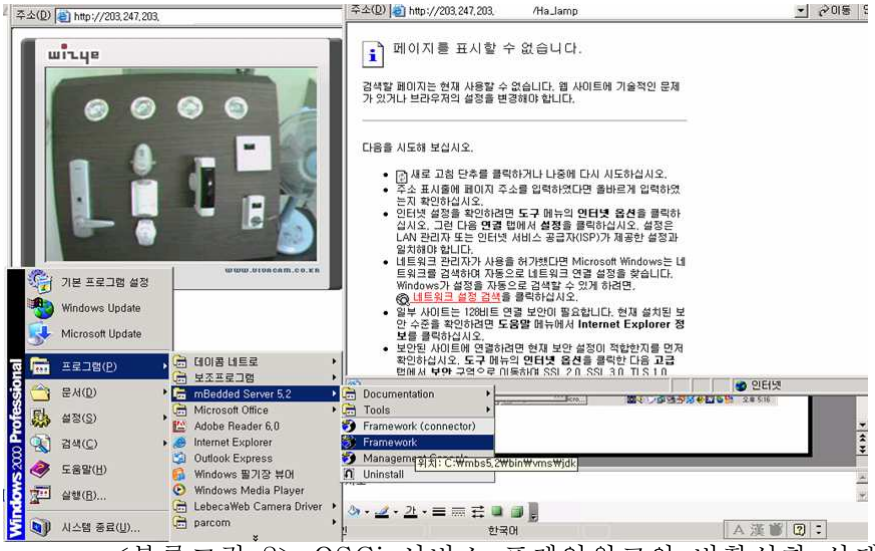
• 본 자료는 경상남도 지역산업진흥사업의 지능형 홈 중점육성 대상서비스에서 일부인용하였음.
• 본 논문에서는 조명관리, 방재관리를 일부 구현하였음.

2. 스마트홈 컨텍스트의 외출모드 데모



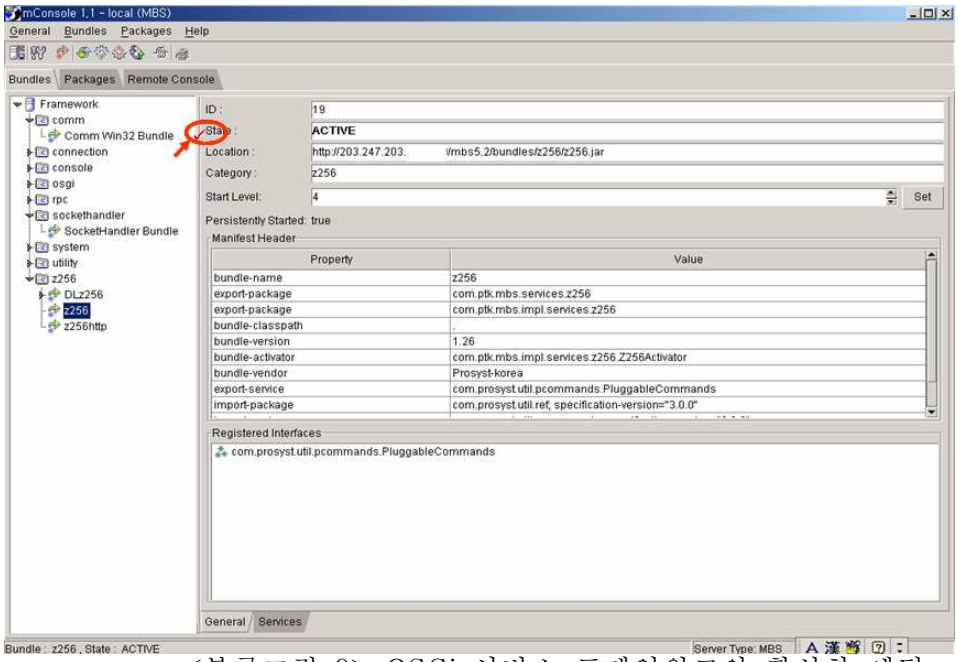
<부록그림 7> z256기반 OSGi 서비스의 외출모드 개념도

<Fig. 7> Outdoor mode concept diagram of OSGi service based z256



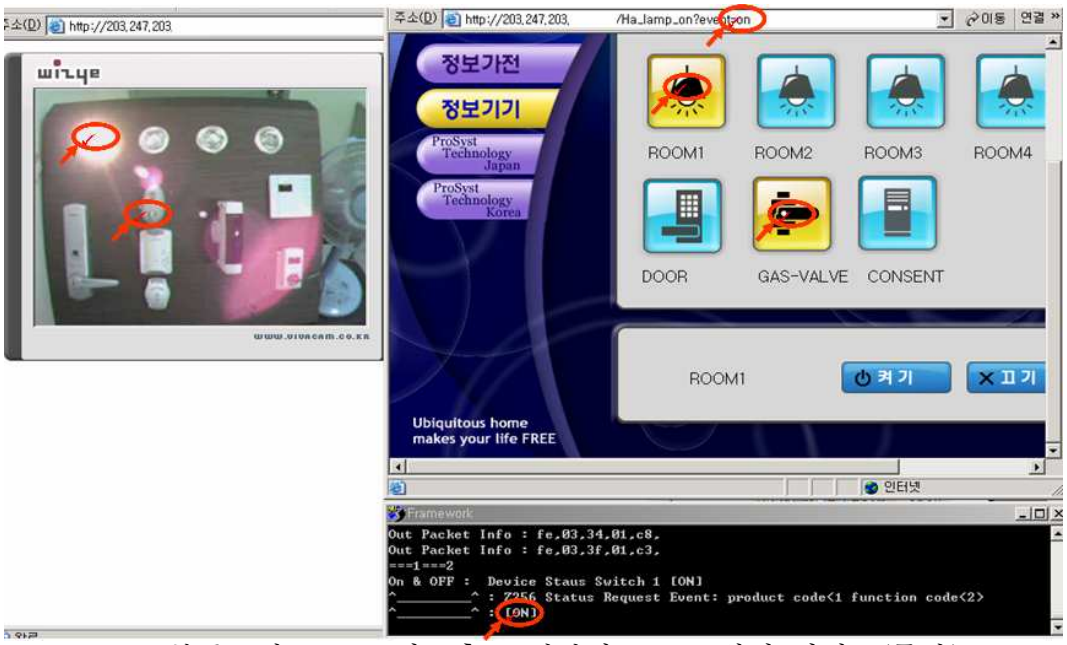
<부록그림 8> OSGi 서비스 프레임워크의 비활성화 상태

<Fig. 8> Inactive state of OSGi service framework



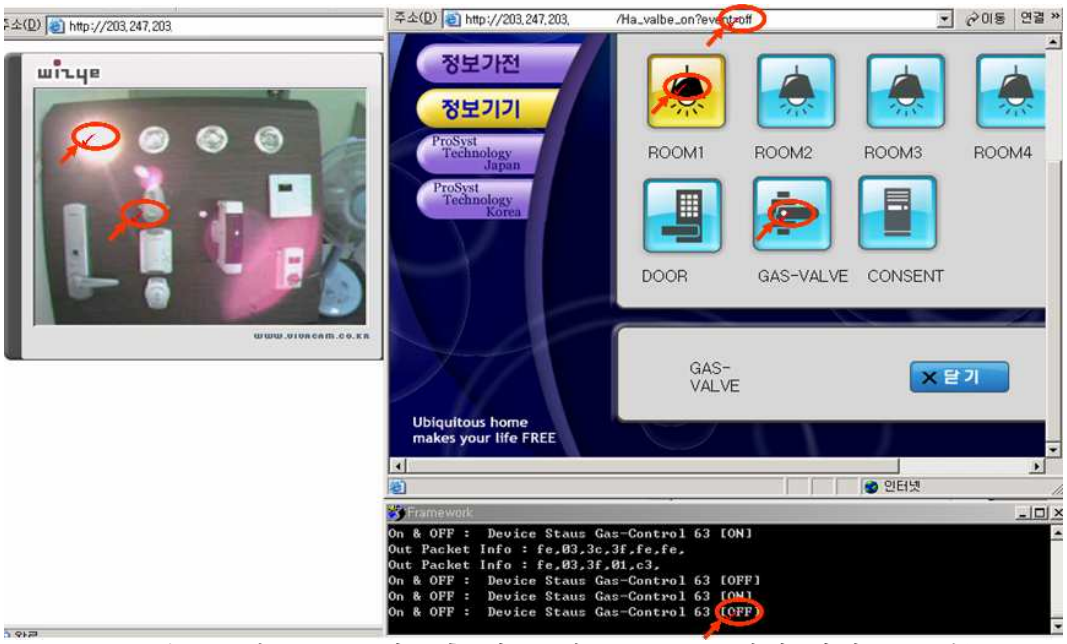
<부록그림 9> OSGi 서비스 프레임워크의 활성화 세팅

<Fig. 9> Active setting of OSGi service framework



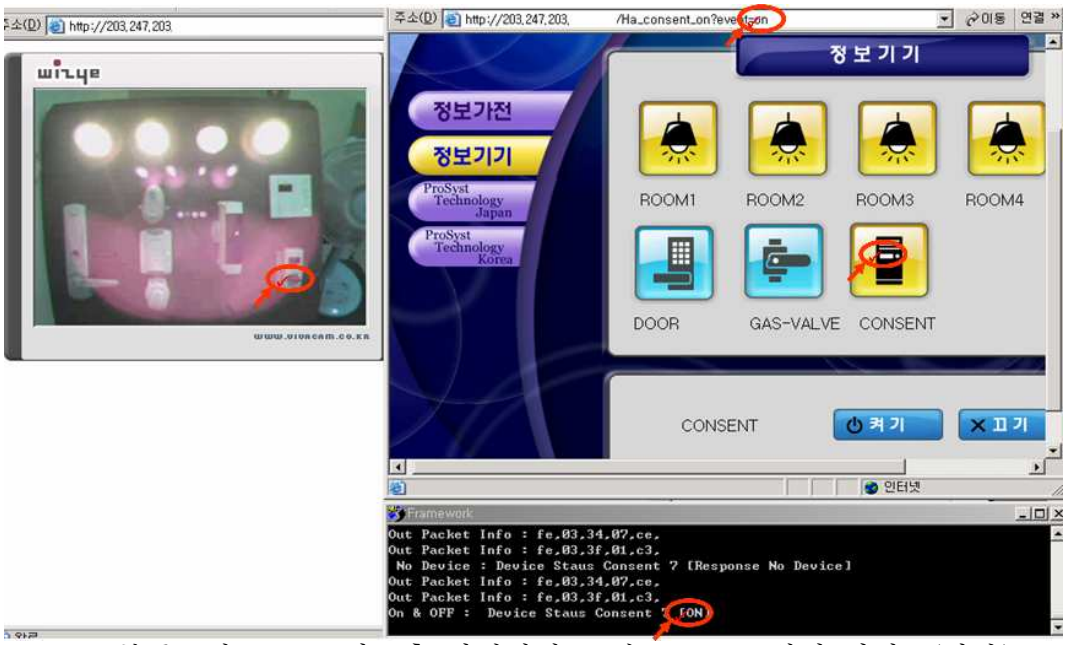
<부록그림 10> 스마트홈 조명관리 ON 스위치 서비스(주방)

<Fig. 10> Smart home lighting management ON switch service(Kichen)



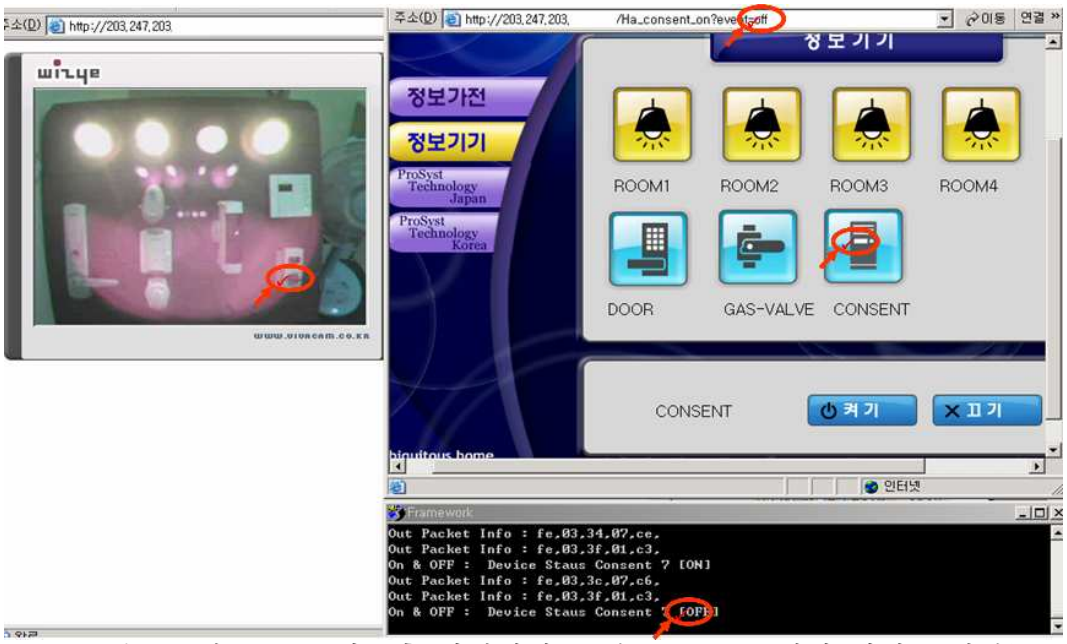
<부록그림 11> 스마트홈 가스 밸브 OFF 스위치 서비스(주방)

<Fig. 11> Smart home gas value OFF switch service(Kichen)



<부록그림 12> 스마트홈 가전기기 콘센트 ON 스위치 서비스(안방)

<Fig. 12> Smart home appliance consent ON switch service(Room)



<부록그림 13> 스마트홈 가전기기 콘센트 OFF 스위치 서비스(안방)

<Fig. 13> Smart home appliance consent OFF switch service(Room)

<부록그림 14> 스마트홈 조명관리 일괄소등 서비스

<Fig. 14> Smart home lighting management all OFF switch service

3. ATAM 평가방법론의 개요

아키텍처가 특정 품질 목표를 만족하는지에 대해 분석할 수 있을 뿐만 아니라, 품질 목표들 간에 발생하는 충돌에 대해서도 분석할 수 있다. 품질 목표들 간에 충돌을 프로젝트 초기(요구사항 정의 또는 설계 단계)에 알아냄으로써, 비교적 경제적인 방법으로 문제를 해결할 수 있다. 변경, 다른 시스템과의 통합, 업그레이드 등이 필요한 레거시 시스템을 분석하는데도 쓰일 수 있다.

가용한 인원과 아키텍처 정보에 따라서 다양한 일정 조절이 가능하며, ATAM 절차는 엄격한 폭포수형 프로세스는 아니므로 필요에 따라서 전단계로 회귀, 다음단계로 건너뛰기, 단계 간에 반복 등이 가능하고. 절차의 순서가 중요한 것이 아니라, 수행해야 하는 활동이 무엇이고 어떤 결과를 얻게 되는 지가 중요하다[19,35].

스텝 1. ATAM 소개

(1) 평가선임자가 스테이크홀더에게 ATAM을 소개한다.

(2) 특히 선임자는 다음의 내용을 설명해야 한다:

1) ATAM 스탭 개요

2) 평가 및 분석 기법

유틸리티 트리 생성 기법, 아키텍처 기반의 분석 기법, 시나리오 브레인스토밍 기법, 시나리오 우선순위 매기기 기법

스텝 2. 비지니스 드라이버 소개

(1) 프로젝트 의사결정권자가 비지니스적인 관점에서 시스템 개요를 설명한다.

1) 비지니스 목표와 가장 중요한 아키텍처적인 요구사항

(2) 다음의 내용을 설명해야 한다:

1) 가장 핵심적인 시스템의 기능

2) 관련된 기술적, 관리적, 경제적, 정치적 제약사항

3) 프로젝트의 비지니스 목표와 컨텍스트

<부록표 3> ATAM 절차

<Table 3> ATAM procedure

단계 *1.* 소개	
스텝 1. ATAM 소개:	평가 선임자가 참석자들에게 평가 방법에 대한 설명을 한다. 기대치를 정하고, 질의응답 시간을 갖는다.
스텝 2. 비지니스 드라이버 소개:	프로젝트 대표자(프로젝트 관리자 혹은 고객)가 비지니스 목표와 가장 중요한 아키텍처적인 요구사항이 무엇인지 설명한다.
스텝 3. 아키텍처 소개:	아키텍트가 비지니스 목표를 달성하기 위한 아키텍처에 대한 설명을 한다.
단계 *2.* 조사와 분석	
스텝 4. 아키텍처 접근법 식별:	아키텍트가 아키텍처적인 접근법들을 찾는다. 이것에 대한 분석은 아직 하지 않는다.
스텝 5. 품질속성 유틸리티 트리 만들기:	시스템이 획득해야 하는 품질속성을 도출, 정의한다.
스텝 6. 아키텍처 접근법 분석:	스탭 5에서 식별한 높은 우선순위의 시나리오를 기반으로, 이것에 알맞는 아키텍처적 접근법을 도출해내고 분석한다. 이 단계에서는 아키텍처적인 Risk, Nonrisk, Sensitivity Point, 그리고 Tradeoff Point도 함께 식별한다.
단계 *3.* 시험	
스텝 7. 브레인스토밍과 시나리오 우선순위 매기기:	전체 스테이크홀더 그룹으로부터 시나리오를 도출하고, 투표에 의해서 이것의 우선순위를 매긴다.
스텝 8. 아키텍처 접근법 분석:	스탭 6의 되풀이. 그러나, 스탭 7에서 도출한 높은 우선순위의 시나리오를 중심으로 수행한다.
단계 *4.* 보고	
스텝 9. 결과 보고:	평가팀은 ATAM 평가를 통해 수집한 정보를 관련 스테이크홀더에게 보고한다.

4) 주요 스테이크홀더

5) 아키텍처 드라이버 - 주요 품질속성 목표

스텝 3. 아키텍처 소개

(1) 선임 아키텍트(혹은 아키텍처팀)가 비지니스 목표에 맞는 아키텍처를 적절한 수준에서 설명한다.

1) 적절한 수준(Appropriate level)이란,

① 설계 및 문서화된 아키텍처의 양

② 가용한 시간

③ 기능적 특성과 품질 요구사항에 따라서 달라진다.

(2) 아키텍트는 다음의 내용을 설명해야 한다:

1) (사용하기로 한 운영체제, 하드웨어, 또는 미들웨어와 같은) 기술적인 제약사항

2) 시스템과 상호작용하는 다른 시스템

3) 품질속성 요구사항을 만족시키기 위해 사용되어진 아키텍처 접근법

스텝 4. 아키텍처 접근법 식별

(1) 평가팀이 아키텍처 접근법을 "식별"한다. 아직 "분석"은 하지 않는다.

1) 아키텍트에게 물어보거나, "스탭 3: 아키텍처 소개" 단계에서 언급되었던 접근법.

(2) 최우선순위의 품질속성을 달성하기 위한 아키텍처 접근법과 아키텍처 스타일을 제시해 본다.

1) 아키텍처 접근법은 중요한 시스템 구조와 시스템의 성장, 변경, 공격에 대한 저항, 통합 방법 등을 설명한다.

2) 아키텍처 스타일은 컴포넌트 타입, 이들의 위상, 테이타 패턴, 컴포넌트간에 상호작용 제어, 해당 스타일을 적용하였을 때 장/단점을 설명한다.

스텝 5. 품질속성 유틸리티 트리 만들기

(1) 스탭 5에서 하는 일?

1) 시스템이 획득해야 하는 품질속성을 도출, 정의하기 위해서 평가팀은 프로젝트 의사결정권자(아키텍처팀, 매니저, 고객)와 함께 작업을 수행한다.

(2) 스탭 5에서 만들어지는 것?

1) 이 단계의 산출물은 품질속성 요구사항에 대한 우선순위와 시나리오가 만들어진다.

(3) 왜 이런 일을 해야 하는가? 어떻게 하는가?

1) 평가팀은 아키텍처를 평가하기 전에, 이러한 시스템 목표를 반드시 보다 정확하고 구체적으로 정의할 필요가 있다.
2) 더욱이 아키텍처를 평가하는 동안에 평가팀이 더욱 주의를 기울여야 하는 부분을 알아내기 위해서, 다른 목표와 비교했을 때에 상대적인 중요도를 알아야 한다.
3) 유틸리티 트리는 품질목표를 구체적으로 만들어주고, 우선순위를 결정하는데 도움을 준다.

(4) 유틸리티 트리

1) 비지니스 드라이버를 구체적인 품질속성 시나리오로 변경하기 위한 매커니즘을 제공.
2) 트리 구성 : Utility(Root), Quality Attribute Refinement(QAR), Scenario(Leaf)
3) 시나리오에 우선순위를 매기는 방법 (퍼지를 이용한 품질측정이용)
4) 우선순위는 2차원으로 구성
   성공적인 시스템을 위해 해당 시나리오의 중요도(Importance)
   해당 시나리오를 달성하는데 대한 난이도(Difficulty)

(5) 스탭 5의 결과물은 자신에게 어떤 도움을 주나?

1) ATAM의 나머지 부분에 대한 계획을 세우는데 도움을 줌.
2) 어느 부분에 시간을 집중 투자해야 하는지 결정하는데 도움을 줌.
3) 높은 우선순위의 시나리오를 만족시키기 위한 아키텍처 접근법이 무엇인지 찾을 수 있도록 도와준다.
4) 시나리오를 통해 품질 요구사항을 보다 구체적이게 정의할 수 있도록 도와준다.

(6) 시나리오

1) 품질 목표를 도출해내기 위한 매커니즘.

2) 스테이크홀더와 시스템간에 상호작용을 묘사하고 있는 짧은 문장.

3) 스테이크홀더의 관점에서 본, 시스템이 상호작용하는 줄거리.

(7) 적용분야 : 아키텍처 평가, 시스템 분석, UI 엔지니어링, 요구사항 도출, 퍼포먼스 모델링, 안정성 검사 등 다양한 분야.

(8) 장점 : 작성 및 이해하기 쉬우며, 경제적이고 효율적이다.

(9) 시나리오는 무엇을 묘사하는가?

시나리오는 시스템을 개발하는 과정과 실행되는 동안의 품질을 예측함.

(10) 시나리오의 형식은 다음의 3개로 이루어진다.(ABAS:Attribute-Based Architectural Styles)

1) 자극(Stimulus)

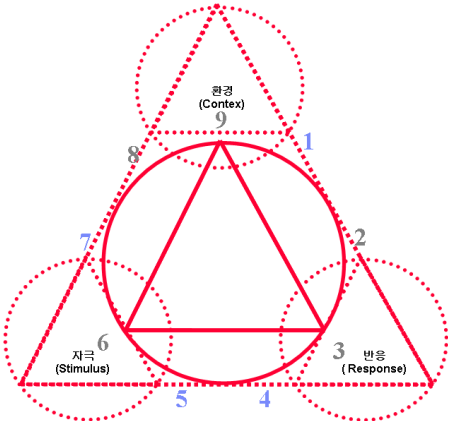
누가(스테이크홀더) 시스템과의 상호작용을 일으키는가?

2) 환경(Context)

자극을 받은 시점에서 이에 대한 반응에 영향을 끼치게 되는 시스템의 상태,조건,환경

3) 반응(Response)

스테이크홀더 자극에 대해서 시스템은 어떻게 반응하는가?



<부록그림 15> 시나리오 형식과 에니어그램의 관계

<Fig. 15> Relationship of Scenario style and Enneagram

(11) ATAM에서는 3가지 서로 다른 각도에서 시스템을 분석하기 위해서 3가지 유형의 시나리오 타입을 사용한다:

1) 유스케이스(Use case) 시나리오 : 일반적인 시스템의 사용 사례
2) 성장(Growth) 시나리오 : 예상되는 시스템의 변경 사례
3) 사전탐사(Exploratory) 시나리오 : 생각하기도 싫은 시스템의 극단적인 변경 사례

(12) 왜 3개의 시나리오 타입을 사용하는가?

1) 아키텍처의 서로 다른 측면을 조망하는데 도움을 줌.
2) 시스템을 진화시키기 위한 전략을 제공해줌.
3) 스테이크홀더간에 의사소통을 원활하게 해주고, 각자의 의견을 체계화하고 이해하기 쉽게 해줌.

스텝 6. 아키텍처 접근법 분석

(1) 전단계에서 수행한 일들:

1) 우선순위가 결정된 품질 요구사항들 (스탭 5)
2) 아키텍처에서 이용된 접근법들 (스탭 4)

(2) 여기서 수행해야 하는 일:

1) 아키텍처 접근법이 중요한 품질속성(높은 우선순위의 시나리오)을 달성할 수 있는지 엄밀하게 분석한다. 즉, 아키텍처가 요구사항에 적절한지 분석.
2) Risk, Nonrisk, Sensitivity Point, Tradeoff Point도 함께 식별한다.

스텝 7. 브레인스토밍과 시나리오 우선순위 매기기

(1) ATAM의 시험 단계(스탭 7~ 스텝 8)는 시나리오에 의해서 주도된다.

1) 유틸리티 트리를 만드는 목적이 주로 품질속성을 다루기 위한 것인데 반해서, 시나리오 브레인스토밍은 대규모 스테이크홀더 집단의 의중을 파악하기 위한 기법이다.
2) 스탭 5에서의 시나리오 우선순위와 스탭 7의 브레인스토밍을 통한 우선순위 비교했을 때

(2) 이 단계에서 평가팀은 3가지 종류의 시나리오를 브레인스토밍한다.

1) 유스케이스(Use case) 시나리오

2) 성장(Growth) 시나리오

3) 사전탐사(Exploratory) 시나리오

(3) 브레인스토밍을 하기 위한 시나리오는 어떻게 수집하는가?

1) 유틸리티 트리를 만들면서 기술하였던 시나리오 중에서 아직 충분한 분석이 이루어지지 않은 시나리오.

(4) 본 논문에서는 이 스텝에서 해당 문헌조사와 스테이크홀더간의 충분한 협의를 통하여 CBAM(Cost Benefit Analysis Method)의 품질속성의 반응값을 선택한 후 이의 최대가치의 반응값 집합과 최소가치의 반응값 집합을 분리하여 MATLAB 퍼지시뮬레이션을 적용하였다.

스텝 8. 아키텍처 접근법 분석

(1) 스탭 7에서 새롭게 찾은 시나리오를 기반으로, 이것에 알맞은 아키텍처적인 접근법을 도출해내고 분석.

스텝 9. 결과 보고

(1) ATAM을 통해 수집한 정보를 정리하고, 스테이크홀더에게 보고한다.

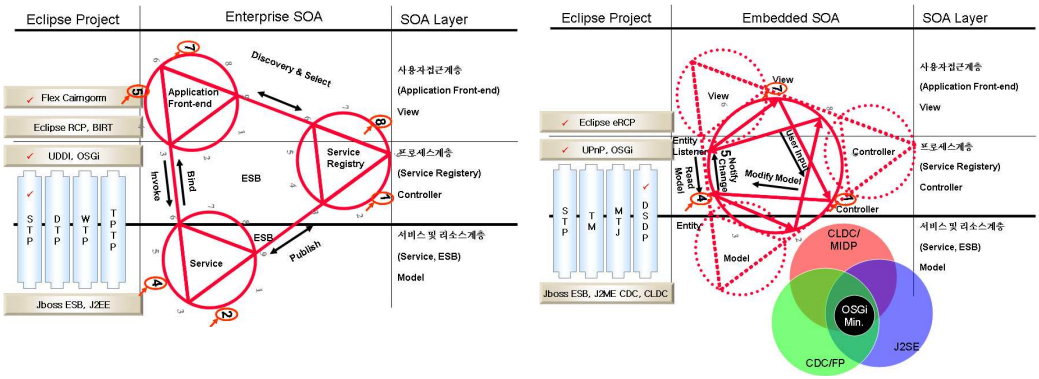
제안된 SOA 요약문

제출자	소 속	전자통신공학과	학번 : 19951410	성 명	최 성 욱
주 제	스마트홈 컨텍스트에서 에니어그램을 이용한 서비스지향 아키텍처의 설계 및 평가				

목 적 :

이클립스 프로젝트 기반에서 범위별 서비스지향 아키텍처를 아래와 같이 설계하고 수정된 ATAM을 이용하여 utility tree를 만들어 최대 및 최소가치의 품질속성을 파악한 후 요구되는 스마트홈 환경에 적합한 KPI를 이용하여 MATLAB 퍼지시뮬레이션을 했다. 설계된 SOA는 Triad 경로 "9-3-6" 및 Heptad 경로 "7-1-4-2-8-5-7"의 에니어그램 순회를 통하여 최적의 유연성, 가용성, 성능을 만족하는 서비스 발견 및 전달, 서비스 조립, 서비스 가상화의 논리적인 뷰를 제공해 준다.

범위별 SOA 개 요



MVC Layer	Enterprise SOA	Homepage URL
View (Presentation)	Flex Cairngorm	labs.adobe.com/wiki/index.php/cairngorm
	eclipse RCP, BIRT	eclipse.org/rcp, eclipse.org/birt
Controller (Logic)	UDDI	uddi.org
	OSGi	osgi.org, eclipse.org/equinox
Model (Data)	TPTP	eclispe.org/tptp
	WTP	eclispe.org/webtools, springframework.org
	DTP	eclispe.org/datatools, hibernate.org
	STP(SCA,SDO)	eclispe.org/stp
	JBoss ESB	labs.jboss.com/portal/jbossesb
	J2EE	java.sun.com/javaee

MVC Layer	Embedded SOA	Homepage URL
View	eclipse eRCP	eclipse.org/ercp
Controller	UPnP	upnp.org, prosyst.com
	OSGi	osgi.org, prosyst.com
Model	DSDP	eclispe.org/dsdp
	MTJ	eclispe.org/dsdp/mtj
	TM	eclispe.org/dsdp/tm
	J2ME CDC,CLDC	java.sun.com/products/sjwtoolkit/

감사의 말씀

먼저 여호와 하나님께 감사드리며 제가 대학원에 들어와서 졸업하기 까지 많은 가르침을 주시고 이끌어 주신 이상배 교수님께 진심으로 깊이 감사드립니다. 바쁘신 가운데에도 많은 격려와 조언으로 본 논문을 심사하여 주신 임재홍 교수님, 심준환 교수님, (주)엘지전자의 백승면 CE님, (주)풍산마이크로텍의 손홍근 사장님께 머리 숙여 감사드리며, 대학원 전공과정에서 많은 가르침으로 저를 지도하여 주신 모든 교수님께도 감사드립니다. 아울러 본 논문을 준비하는 가운데 서비스지향 아키텍쳐의 사상에 관하여 허심탄회한 토론 및 공동연구를 허락하신 동아대학교의 최형림 교수님, 김현수 교수님, 홍순구 교수님과 스마트홈 관련 인프라스트럭쳐 및 공동연구를 허락하신 동명대학교의 권오현 교수님, 오암석 교수님과 OSGi와 UPnP 관련 유용한 자료를 제공하여 주신 (주)프로시스트테크놀로지코리아의 이해준사장님, (주)씨드시스템의 손석길 사장님, 안병선 이사님께 깊이 감사드립니다. 퍼지뉴로제어연구실(FNCL)의 김일 선배님, 탁한호 선배님과 연구실생활을 함께했던 김관형, 이재현, 강성인, 김영탁 후배박사와 김성주, 강동우, 채명기, 문주영, 김태영, 이주원, 방은오, 박승현, 조병일, 이용수, 손창우, 공석민, 조동민, 이주상 이하 모든 후배들께 감사드립니다.

본 논문이 나올 때까지 아내 현지연의 헌신적인 노력이 없었다면 지금의 제 모습이 아니었으며, 더불어 사랑스러운 아내에게 약속한 초심을 언제까지나 기억하며 변함없이 사랑한다는 말을 전합니다. 또한 불효자식 때문에 늘 세빈이, 세은이를 지금까지 대신 보살펴 주시며 사랑으로 건강하게 키워 주신 어머님께 감사드리며, 건강하지 못하신 몸이지만 살아계신 것만으로도 저에게는 힘이 되어 주시는 아버님에게도 감사드린다는 글을 전하고 싶습니다. 나에게 아버지라는 이름으로 존재의 이유를 상기시켜주며 큰 힘이 되어준 아들 세빈이, 딸 세은이 에게도 고맙다는 말을 하고 싶습니다.

마지막으로 저를 사랑해 주시고 아껴주신 장인어르신과 장모님, 남동생 성렬이와 큰 제수씨, 여동생 소영이와 류지매제, 성길이와 작은 제수씨, 현동수처남과 현화영처제, 또한 조카들인 혜원이, 아야까, 유리까, 수빈이, 타이요오, 원빈이, 혜빈이, 그리고 엄궁회중 성원들에게 완성의 기쁨을 함께 하고자 합니다.