DOI: 10.3745/KIPSTD.2008.15-D.3.361

적응형 소프트웨어를 위한 프로덕트 라인 아키텍처 개발

예 은 숙[†] · 염 근 혁[††] · 문 미 경[†††]

요 약

최근의 유비쿼터스 컴퓨팅 환경에서 요구되는 새로운 패러다임의 소프트웨어인 적응형 소프트웨어는 외부 환경 즉 문맥의 변화를 인식하고 자신의 운영 상태를 평가하여 스스로 동작행위를 바꾸는 능력을 가진 소프트웨어이다. 적응형 소프트웨어를 개발하기 위해서는 문맥에 대한 분석 및 설계와 더불어 동적인 문맥에 따라 적절히 대응할 수 있는 소프트웨어의 구조 및 행위를 설계하는 과정이 필요하다. 이것은 일반적인 소프트웨어보다 소프트웨어가 적응해야 하는 상황을 모델링 하는 기술이 필요하기 때문에 더 많은 노력이 필요하다.

본 논문에서는 재사용을 통해 개발의 효율성을 높이고자 적응형 소프트웨어를 위한 프로덕트 라인 아키텍처와 아키텍처 산출물의 템플릿을 제안한다. 아키텍처 산출물들은 프로덕트 라인의 공통성과 가변성을 명시적으로 표현하여 체계적인 핵심자산의 재사용 활동을 지원한다.

키워드 : 소프트웨어 프로덕트 라인, 적응형 소프트웨어, 프로덕트 라인 아키텍처, 핵심자산, 공통성과 가변성

Product-Line Architecture Development for Self-Adaptive Software

Eunsuk Ye[†] · Keunhyuk Yeom[††] · Mikyeong Moon[†††]

ABSTRACT

In the latest intelligent computing environments, the self-adaptive software, as new software paradigm, which modifies its own behavior in response to changes in its operating environment is needed. To develop the self-adaptive software, it is necessary to analyze and design the context of software as well as the structure and the behavior of software. We need more efforts for self-adaptive software development than for traditional software development because we need more activities and technologies like context modeling and adaptation to develop the self-adaptive software.

In this paper, we present the product line architecture for self-adaptive software and templates of artifacts to improve the efficiency of development through a reuse methodology. The artifacts of the architecture support the systematic reuse activities of core assets by expressing the commonality and variability of product line.

Key Words : Software Product Line, Self-Adaptive Software, Product Line Architecture, Core Asset, Commonality And Variability

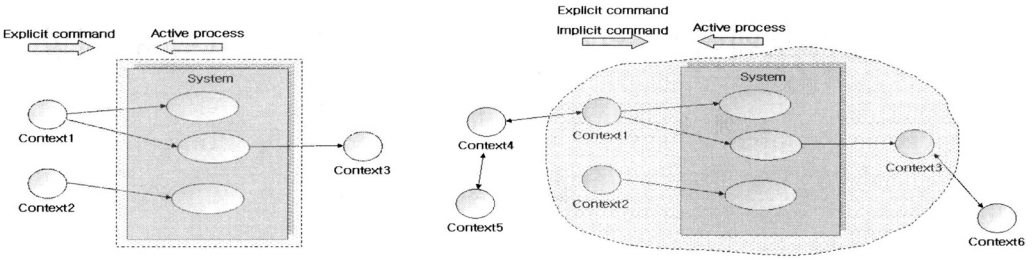
1. 서 론

소프트웨어 프로덕트 라인은 일련의 관련된 소프트웨어들이 공유할 수 있는 핵심자산을 이용하여 소프트웨어를 개발하는 재사용 방법의 하나이다[1]. 도메인 내에서 재사용될 가능성이 높은 공통 부분을 식별하고 시스템마다 상이하게 나타나는 가변되는 부분을 분석하여 모든 자산에 이들을 명시적으로 나타내어 핵심자산으로 관리한다. 프로덕트 라인 아키텍처는 소프트웨어 프로덕트 라인의 주요한 핵심자산 중 하나이며 여러 소프트웨어의 전체적인 구조를 나타낸다.

※ 이 논문은 2006년도 정부(교육과학기술부)의 재원으로 한국과학재단 특정기초연구사업의 지원을 받아 수행된 연구임(No. R01-2006-000-10536-0)
† 정 회 원 : 부산대학교 컴퓨터공학과(석사)
†† 정 회 원 : 부산대학교 정보컴퓨터공학부 교수
††† 정 회 원 : 동서대학교 컴퓨터정보공학부 전임강사(교신저자)
논문접수 : 2008년 1월 16일
수 정 일 : 1차 2008년 4월 1일, 2차 2008년 4월 24일
심사완료 : 2008년 4월 24일

유비쿼터스 컴퓨팅 환경에서는 사용자와 상호 작용하며 환경의 변화에 지능적으로 동작하는 컴퓨팅 기술이 요구되고 있다. 이러한 요구를 만족하는 특성을 가진 것이 적응형 소프트웨어이다. 적응형 소프트웨어란 소프트웨어가 환경의 변화를 인식하고 자신의 운영 상태를 평가하여 스스로 동작행위를 환경에 맞추어 바꾸는 능력을 가진 소프트웨어이다[2]. 이러한 소프트웨어를 개발하기 위해서는 그 설계의 산출물부터 일반적 소프트웨어와는 다르게 구성된다. (그림 1-(a))처럼 기존의 소프트웨어는 시스템의 범위(회색 점선 상자) 내부만을 설계하고 개발하였지만, 적응형 소프트웨어는 (그림 1-(b))처럼 시스템의 외부 환경 즉 문맥(context)도 개발범위(회색 점선 타원)에 포함된다.

적응형 소프트웨어가 되기 위해 문맥 변화에 대한 정보를 수집하고 분석하는 능력과 자신의 행위를 적절하게 변경할 수 있는 구조를 갖춰야 한다. 시스템에 관련된 여러 이해관계자(stakeholder)의 요구사항을 반영하는 것뿐만 아니라 이



(a) 일반적 소프트웨어 (b) 적응형 소프트웨어

(그림 1) 일반 소프트웨어와 적응형 소프트웨어의 개발 범위

러한 적응성을 위한 능력과 구조를 갖춘 소프트웨어는 기술적으로나 관리적으로나 그 복잡성이 커진다. 소프트웨어 아키텍처는 복잡한 소프트웨어의 전체적인 모습을 보여주고 시스템 수준의 중요한 속성과 제약사항을 담고 있다[3]. 따라서 아키텍처는 적응형 소프트웨어의 내부 구조를 설계하고 이해하며 제어할 수 있는 효과적인 개발 자산이 된다. 또한 프로덕트 라인의 아키텍처는 그것을 재사용함으로써 유사한 소프트웨어의 개발 시에 드는 시간과 비용을 절감할 수 있게 한다.

본 논문에서는 적응형 소프트웨어를 재사용을 통해 효과적으로 개발하기 위한 프로덕트 라인 아키텍처를 제시한다. 자산화된 적응형 소프트웨어 프로덕트 라인 아키텍처는 공통적으로 이용될 수 있는 분석된 문맥 정보를 재사용할 수 있게 하고 이와 관련된 컴포넌트의 중복적인 개발을 줄일 수 있다.

본 논문의 구성은 2장에서 적응형 소프트웨어를 위한 프로덕트 라인 아키텍처 뷰의 전체적 구성을 제시하고, 3장에서 6장까지는 아키텍처를 구성하는 각 뷰와 명세에 대해서 특정 도메인에 적용한 예제(테마파크 PDA 시스템 도메인)를 제시하여 설명한다. 7장에서는 본 연구와 관련된 기존 연구를 소개하고 비교 평가한다. 마지막으로 8장에서는 결론 및 향후 연구 과제를 제시한다.

2. 적응형 소프트웨어를 위한 프로덕트 라인 아키텍처

본 장에서는 적응형 소프트웨어가 갖추어야 하는 요소인 문맥 인식을 위한 구성요소, 소프트웨어 기능 수행을 위한 구성요소, 소프트웨어의 적응을 담당하는 구성요소를 가진 아키텍처 구조에 대해 설명한다. 그리고 각 아키텍처 구성요소마다 명시적으로 표현되는 공통성과 가변성에 대하여 설명한다. 본 논문에서는 테마파크 PDA 시스템 도메인에 적용한 아키텍처의 구성을 사례로 보인다. 테마파크 PDA 시스템 도메인은 사용자가 PDA를 가지고 테마파크에서 운영되는 모든 놀이기구 및 관람시설의 기다리는 시간을 절약하도록 대기 예약할 수 있으며, 대기 예약된 시설물에 대해 자동 스케줄링 하여 시간을 관리하도록 도와준다. 그리고 사용자의 정보, 날씨 등의 변화에 반응하여 그 구조와 동작을 변형시킨다.

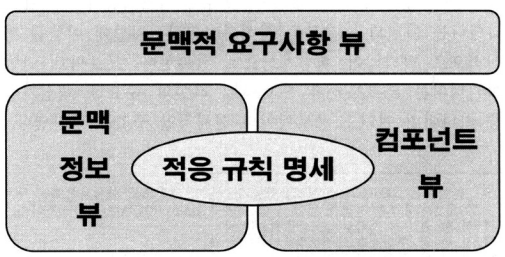
2.1 아키텍처 뷰의 구성

본 논문에서 제시하는 적응형 소프트웨어를 위한 프로덕트 라인 아키텍처는 (그림 2)와 같이 세 개의 뷰와 적응 규칙 명세로 구성된다. 각 뷰에서는 그 뷰의 관점을 나타내는 모델을 통해 소프트웨어의 구조를 볼 수 있다[4].

문맥적 요구사항 뷰(contextual requirements view)에서는 적응형 소프트웨어에 대한 요구사항을 정의한다. 이는 적응형 소프트웨어의 기능 및 외부 문맥 구성 정보, 컴포넌트 구성 정보, 소프트웨어의 적응 행동에 대한 근거를 제공한다. 문맥적 요구사항 뷰는 문맥적 요구사항 다이어그램과 문맥적 요구사항 명세서로 이루어진다.

문맥 정보 뷰(context information view)에서는 적응형 소프트웨어가 관심을 가지는 문맥을 분석하고 설계한다. 관심을 가지는 문맥이란 동작 중인 소프트웨어와 상호작용하면서 동작에 영향을 주고받는 환경요소를 의미한다. 소프트웨어가 문맥과 상호작용하기 위해서는 필요한 문맥이 무엇인지 알아야 하고, 소프트웨어가 인식할 수 있는 문맥 정보의 형태가 정의되어야 한다. 이러한 지식들을 문맥 정보 뷰에서 문맥 객체 모델과 문맥 정보 명세를 통해 나타낸다. 문맥 객체 모델은 적응형 소프트웨어와 상호작용을 하는 문맥 객체들의 구성을 나타내는 모델이며, 분석된 문맥 정보에 대한 상세 내용을 문맥 정보 명세서로 작성한다.

컴포넌트 뷰(component view)에서는 소프트웨어의 기능



(그림 2) 적응형 소프트웨어를 위한 프로덕트 라인 아키텍처 뷰의 구성

을 제공하는 컴포넌트의 논리적 구성 정보를 보인다. 소프트웨어는 구성하고 있는 컴포넌트들이 상호작용함으로써 사용자에게 필요한 기능을 제공한다. 컴포넌트 뷰는 구성하고 있는 컴포넌트의 구조적인 정보와 그들의 상호작용하는 행위 정보를 나타내는 모델로 구성되며, 각 컴포넌트에 대한 자세한 정보를 제공하는 명세가 포함된다.

적응 규칙 명세(adaptation rule specification)에서는 소프트웨어의 적응성을 나타낸다. 관심 있는 문맥의 변화를 인식하여 소프트웨어의 구조나 행위가 어떻게 적응해야 하는가를 정의한다.

2.2 가변성의 특징

핵심자산의 공통성과 가변성을 잘 정의하는 것은 프로덕트 라인 개발에서 핵심적 활동이다. 가변성은 소프트웨어 재사용을 위해 개발되는 자산들 중에서 프로덕트 라인 멤버들 간의 달라짐을 나타내는 성질이다. 본 논문에서는 다음의 세가지 특징으로 가변성을 정의한다.

- **가변성 유형 정의** - 핵심 자산들은 소프트웨어 개발의 각기 다른 단계에서 생산되며 이를 표현하는 구성요소가 다르기 때문에 가변성이 각기 다른 형태로 나타나게 된다[5]. 그러므로 본 논문에서도 아키텍처 뷰에 따라 기술되는 아키텍처 구성요소가 달라지기 때문에 각 뷰마다 식별될 수 있는 가변성 유형을 분석하여 이를 가변성 유형으로 정의한다.
- **가변성 수준** - 가변성의 상세화 정도를 두 단계 수준으로 나누어 나타낸다. 상위 수준에서는 기능의 선택성을 나타내는 속성으로 가변성을 표현한다. 선택성은 애플리케이션 산출물에 존재할 것인지 여부를 나타낸다. 상세 수준에서는 하나의 기능 내부에서 가변될 수 있는 요소를 가변점으로 나타난다. 가변점은 가변되는 지점을 표시하며, 가변점에 바인딩 될 수 있는 값을 가변치(variant)로 정의한다.
- **가변성 바인딩 타임** - 바인딩 타임이란 핵심자산으로부터 목표 애플리케이션을 개발(Instantiation)할 때 가변점에 대해 가변치를 결정하는 시기를 말한다[6]. 이는 설계 시점(design time), 컴파일 시점(compile time), 실행파일의 링킹 시점(linking time), 런타임 시점(runtime) 등에서 이루어질 수 있다. 본 논문은 프로덕트 라인 아키텍처의 논리적 수준의 뷰를 제시하고 있으므로 가변성의 바인딩 타임을 설계 시점과 런타임 시점으로 구분한다. 설계 시점에서 가변성의 바인딩이 이루어진다는 것은 특정 목적의 애플리케이션을 개발할 때 핵심자산의 가변성에 대한 결정이 설계 단계에서 모두 완료되어야 함을 의미한다. 런타임 시점의 바인딩은 가변성에 대한 결정이 애플리케이션 동작 중에 행해짐을 의미한다. 따라서 런타임 시점 바인딩 가변성은 대응될 수 있는 모든 가변치들을 애플리케이션의 실행 중에 포함하고 있어야 한다. 적응형 소프트웨어는 동작하는 중에 관심 문맥의 변화로 인해 자신의 상태가 목적을 이루지 못하게 되면 런타임 가변성에 포함되어 있는 가변치들 중 변화된 문맥에 적절한 것으로 바인딩되어 적응성을 이룰 수 있다.

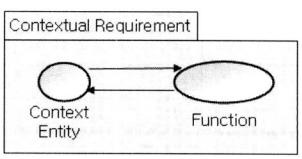
3. 문맥적 요구사항 뷰

일반적인 소프트웨어의 요구사항은 상호작용하는 수행 환경의 정적인 상태만을 반영하여 정의된다. 적응형 소프트웨어는 정적인 것뿐만 아니라 변화하는 환경에 적응할 수 있어야 하기 때문에 동적인 환경요소와 소프트웨어 간의 상호작용에 대한 요구사항을 명확하게 파악해야 한다. 그러므로 적응형 소프트웨어의 요구사항 정의에는 개발할 소프트웨어의 기능적 요구사항과 더불어 문맥의 변화에 대해 소프트웨어의 적응 행위의 요구사항을 기술한다. 소프트웨어의 적응 행위는 사용자에게 보여지는 관점에서 기술된다. 본 논문에서는 적응형 소프트웨어의 요구사항을 기술하기 위해 문맥적 요구사항을 정의한다. 문맥적 요구사항은 (그림 3)과 같이 소프트웨어의 단위기능과 기능 수행에 영향을 주는 문맥 객체, 그리고 이들 간의 상호작용을 포함하는 개념이다. 문맥적 요구사항 정의 활동의 산출물은 문맥적 요구사항 다이어그램과 문맥적 요구사항 명세서로 구성된다.

3.1 문맥적 요구사항의 가변성 유형

공통성과 선택성을 표현하는 상위 수준의 가변성은 문맥적 요구사항 다이어그램에서 표현된다. 문맥적 요구사항 다이어그램을 구성하는 각 패키지마다 문맥적 요구사항 명세서가 기술되는데, 여기에서 상세 수준의 가변성이 표현된다. 문맥적 요구사항 뷰에서 나타나는 가변성 유형은 다음과 같다.

- **문맥적 요구사항 속성 가변성(상위 수준)** - 적응형 소프트웨어의 요구사항 단위 기능인 문맥적 요구사항에 대한 선택성을 나타낸다.
- **문맥 객체 가변성(상위 수준)** - 소프트웨어와 상호작용을 하는 문맥 객체가 선택적으로 존재할 수 있는 경우를 의미한다.
- **문맥 변화 가변성(상세 수준)** - 문맥 객체의 상태가 변하게 될 때 그 변화량이나 변화상태 등의 임계범위가 가변적으로 결정이 될 수 있을 경우를 의미한다. 임계범위를 초과한 문맥 변화는 소프트웨어에 영향을 미치게 되고 소프트웨어는 이에 대한 적응 행동을 하게 된다.
- **문맥 측정 가변성(상세 수준)** - 문맥 객체의 상태를 소프트웨어가 인식할 수 있도록 센서 등의 문맥 측정 매체나 측정방법이 가변적일 수 있다. 즉, 해당 문맥



(그림 3) 문맥적 요구사항의 개념

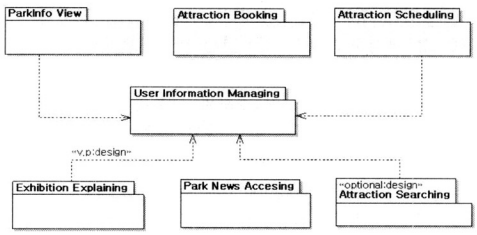
객체에 대해 모든 소프트웨어가 관심을 가지고 있지만, 감지하기 위한 방법에서 가변이 되는 경우를 의미한다. 이 가변 유형은 센서 등의 물리적인 장치 문맥 객체에 대한 가변으로 주로 나타나게 된다.

- **흐름 가변성(상세 수준)** - 처리 흐름에서 특정 흐름이 가변적일 경우에 해당된다.
- **데이터 가변성(상세 수준)** - 시스템의 데이터의 구조에서 나타나는 가변성을 의미한다.
- **제어 가변성(상세 수준)** - 프로세스의 제어흐름을 결정하는 기준규칙 또는 판단조건이 가변되는 경우이다.
- **품질 요구사항 가변성(상세 수준)** - 시스템의 품질 요구사항에 대한 가변성을 의미한다.

3.2 문맥적 요구사항 다이어그램

문맥적 요구사항 다이어그램(contextual requirement diagram)은 파악된 문맥적 요구사항을 표현한 모델이다. 적응형 소프트웨어가 제공하는 기능에 대한 범위를 명확하게 구분하기 위한 목적을 가진다. 즉, 개발할 소프트웨어의 경계를 결정하기 위한 요구사항 정의 활동의 산출물이다. 문맥적 요구사항 다이어그램의 작성은 그림 3과 같은 문맥적 요구사항을 기본 요소로 하며 패키지 형태로 표현한다. 패키지의 표현은 소프트웨어의 기능적 요구사항과 함께 문맥 요소가 포함되어 묶여 있음을 나타낸다. 문맥적 요구사항 사이의 관계는 의존 관계(depend on)를 나타낸다. 그림 4는 테마파크 PDA 시스템 도메인에서 추출된 문맥적 요구사항을 나타낸 다이어그램으로, 테마파크 정보 보기(*ParkInfo View*), 시설물 예약하기(*Attraction Booking*), 예약된 시설물의 스케줄링(*Attraction Scheduling*), 사용자 정보 관리(*User Information Managing*) 및 전시물 안내(*Exhibition Explaining*) 등의 기능을 가지고 있음을 알 수 있다.

모든 문맥적 요구사항들은 공통성 또는 선택성을 가진다.



(그림 4) 테마파크 PDA 시스템 도메인의 문맥적 요구사항 다이어그램

선택성을 가진 문맥적 요구사항은 그 기능이 프로덕트 라인에 포함된 소프트웨어 모두가 가지는 필수적인 것이 아님을 의미한다.

선택성은 스테레오타입<<optional>>을 이용하여 표현한다. (그림 4)에서 시설물 검색 기능(*Attraction Searching*)은 선택적으로 제공되는 기능임을 알 수 있다. 이 가변성은 시스템을 설계할 때 바인딩 되어야 함을 *design* 기호를 덧붙여 나타내고 있다.

3.3 문맥적 요구사항 명세

문맥적 요구사항 다이어그램에 정의된 각 요구사항들의 이름만으로는 요구사항을 명확하게 정의할 수 없다. 문맥적 요구사항에 대한 자세한 정보, 즉 기능의 세부 흐름, 관련된 문맥 객체, 상호작용 흐름 등의 상세한 요구내용을 <표 1>과 같은 형식으로 기술한다.

<표 2>는 테마파크 PDA 시스템 도메인의 *ParkInfo View* 문맥적 요구사항에 대해 작성된 명세서를 보이고 있다. 이 명세서에서 가변성은 각 문맥 시나리오와 소프트웨어 시나리오의 가변치(*variants*)항목에 기록되어 있다. 문맥 시나리오의 A항목(사용자의 위치 정보 파악)에서 문맥 측정 가변

〈표 1〉 문맥적 요구사항 명세서

CR ID	<CR의 식별번호>	**CV_property**	<CR의 가변속성>
CR 개요	<CR에 대한 설명>		
관련 문맥 객체	<CR의 수행에 영향을 주고 받는 문맥 객체들>		
CR 시나리오	**문맥 시나리오**		**소프트웨어 시나리오**
	A.<관련된 문맥의 변화흐름1> B.<관련된 문맥의 변화흐름2> C.<관련된 문맥의 변화흐름3>		1.<소프트웨어 수행 흐름1> 2.<소프트웨어 수행 흐름2> 3.<소프트웨어 수행 흐름3>
variants	[문맥 시나리오no][#][variability type]<관련 시나리오 지점에서 나타날 수 있는 가변값을 기술>		[SW시나리오no][#][variability type]<관련 시나리오 지점에서 나타날 수 있는 가변값을 기술>
선행조건	<CR가 수행되기 전에 만족해야 하는 조건>		
후행조건	<CR의 수행 완료될 때 만족해야 하는 조건>		
제약사항	<CR의 수행에 관련된 문맥 요소나 소프트웨어의 제약사항 및 비기능적 요구사항>		

[#]: cardinality
[variability type]: 요구사항의 가변성 유형

〈표 2〉 ParkInfo View 문맥적 요구사항 명세서

CR ID	ParkInfo View	CV_property	common
CR 개요	PDA기기를 통해 공원 지도를 열람하고 사용자의 위치와 시설물의 정보를 볼 수 있다.		
관련 문맥 객체	사용자의 위치, 날씨 정보, 시설물 정보 DB, 사용자 정보, GPS모듈, WL-LA Toolkit		
CR 시나리오	**문맥 시나리오**	**소프트웨어 시나리오**	
	A.[mm]사용자의 위치를 파악한다(공원내 위치) B.현재 날씨 상태를 인식한다 C.이용마감시간이 임박한 시설물을 안내해 준다 D.사용자의 키 등의 제약사항 등에 의해 제한받을 수 있는 시설물을 안내해 준다 E.빨리 이용 가능한 시설물을 안내해 준다	사용자는 프로그램의 공원지도보기 메뉴를 선택한다. 1.사용자의 위치를 지도 화면의 중심으로 나타낸다. [control]PDA화면크기에 따라 적절한 해상도의 맵 이미지를 요청한다.[quality]PDA는 사용자가 메뉴선택 후 3초이내 데이터를 보여야 한다. 2.전체보기,자세히보기는 다른 해상도를 지원하여 공원 지도를 안내한다 3.[control]지도의 각 시설물에 대해 사용자의 상황에 적합한 안내가 표시돼야 한다 4.[control]공원지도에 나타난 시설물 아이콘을 클릭하면 시설물의 상세정보를 볼 수 있다.	
variants	A.[1..n][mm]사용자위치정보:GPS모듈/WL-LA Toolkit(GPS이용중 시계가 좋지 않을 때에는 WL-LA로 변경 운용되어야 한다)	1.[1][control]맵 이미지 해상도선택-PDA화면 크기에 따라 선택됨 (160x160/240x320/640x480) 1.[0..1][quality]3초이내 응답 3.[1..n][control]사용자에게 적합한 시설물 안내:지도의 시설물 지점에 표시/시설물 상세정보에 표시 4.[1][control]시설물상세정보보기흐름:일반정보만 나타낸다/일반정보와 실시간정보를 함께 나타낸다/일반정보를 먼저 나타내고 실시간정보는 선택적으로 나타낸다	
선행조건	사용자의 정보가 등록되어 있어야 한다.		
후행조건	지도에 나타난 시설물을 선택하여 시설물 상세 정보를 열람할 수 있다.		
제약사항	PDA의 화면화질은 최소3만 화소 이상 되어야 한다.		

<요구사항 가변성 유형>
[element] 문맥 객체 가변성
[change] :문맥 변화 가변성
[mm]: 문맥 측정 가변성
[computation]: 시나리오의 흐름 가변성
[data]: 시나리오의 데이터 가변성
[control]: 시나리오의 제어 가변성
[quality]: 품질요구사항 가변성

성이 나타나고 가변치는 2가지(*GPS모듈, WL-LA Toolkit*)를 가지고 있다. 가변치의 선택은 한가지 이상이 가능하므로 *GPS모듈*이나 *WL-LA Toolkit* 중 한가지만 선택하거나 두 가지 모두 채택할 때는 대체 운용하여 사용할 수 있음을 알 수 있다.

4. 문맥 정보 뷰

문맥 정보 뷰는 문맥 정보의 표현과 그 구조를 명시적으로 나타내며 문맥 객체 모델과 문맥 정보 명세의 산출물을 가진다.

4.1 문맥 정보 가변성

공통성과 선택성을 표현하는 상위 수준의 가변성은 문맥 객체 모델에서 표현된다. 상세 수준의 가변성은 문맥 정보 명세에서 표현된다. 문맥 정보 뷰에서 나타나는 가변성 유형은 다음과 같다.

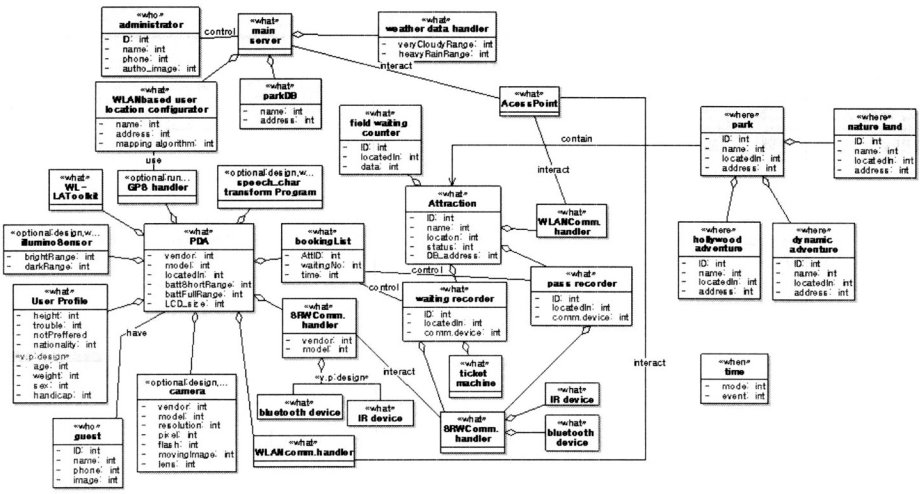
- **문맥 속성 가변성(상위 수준)** - 문맥 객체 또는 문맥 정보가 선택적으로 나타나는 경우에 해당한다.
- **문맥 계산(computation) 가변성(상세 수준)** - 문맥 정보를 얻기 위해 여러 문맥 정보를 통합·추론하는 규칙에서 가변이 발생하는 경우이다.
- **문맥 데이터 가변성(상세 수준)** - 문맥 정보의 데이터 구조가 가변되는 경우이다.

4.2 문맥 객체 모델

문맥 객체 모델(context entity model)은 소프트웨어 외부에서 소프트웨어와 상호작용하며 영향을 주는 문맥 객체를 식별함으로써 관심 문맥의 범위를 나타낸다. 문맥 객체 모델에는 문맥 객체와 객체의 범주 및 속성, 그들 간의 관계가 표현된다.

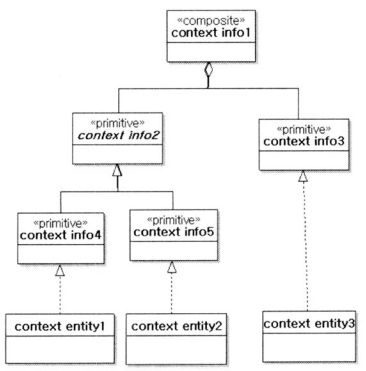
문맥 객체란 적응형 소프트웨어와 상호작용하는 실 세계의 실체를 말하며 물리적·논리적인 공간이나 사람, 물리적 장치, 외부 시스템 및 환경적 요소 등이 될 수 있다. 문맥 객체는 문맥 정의의 범주에 따라 분류될 수 있다. 문맥의 범주화는 소프트웨어 설계자로 하여금 그들의 애플리케이션이 관심을 가질 문맥의 많은 부분을 발견할 수 있게 한다[7]. 문맥은 그 정의에 따라서 Who, What, Where, When으로 분류할 수 있다. Who는 사용자의 신원, What는 외부의 물리적 객체, Where는 사용자나 객체의 위치, When은 시간에 대한 문맥으로 정의한다.

문맥 객체들은 설치 및 구동에 필요한 세부속성이 식별되어 모델에 표현된다. What 카테고리에 속하는 물리적 장치라면 그 속성은 벤더, 모델명, 위치, 데이터 타입 등이 될 수



(그림 5) 테마파크 PDA 시스템 도메인의 문맥 객체 모델

〈표 3〉 문맥 정보 명세서

Spec No	<명세서의 구분번호>		
Spec Name	<명세서 이름>	Related CR	<관련된 문맥적 요구사항 문서 번호>
Context information diagram		Context information specification	
		Context info ID	<문맥 정보의 ID>
		Context info name	<문맥 정보의 이름>
		Description	<문맥 정보에 대한 설명>
		CV_property	<문맥 정보의 가변속성>
		Req_s#	<문맥 정보의 출처>
		Value type	<문맥 정보의 데이터 형>
		Value Driven Rule	
		<문맥 정보의 구성 규칙>	

(그림 6) 문맥 정보 다이어그램

있다. 필요한 문맥 객체를 찾고 각각의 속성을 분석한 후에는 문맥 객체들 간의 관계를 분석한다. 문맥 객체들 사이의 관계는 일반화 관계나 집합관계 그리고 연관관계가 있다.

문맥 객체의 가변성은 요구사항의 문맥 시나리오에서 나타난 가변성에 의존되어 결정된다. 스테레오 타입을 사용하여 문맥 객체의 범주와 가변성 속성을 나타낸다.

(그림 5)은 테마파크 PDA 시스템 도메인의 문맥 객체 모델을 UML 클래스 다이어그램으로 작성한 일부분을 나타내고 있다. 이 시스템은 *GPS handler*처럼 PDA 장치 내부에 장착되어 독립적인 기능을 제공하는 센서 및 프로그램뿐 아니라 외부 장치들인 *main server*와 *DB*, *테마파크 attractions* 및 통신 장치도 시스템에 영향을 주는 what 객체로 식별되었으며 이들 사이의 관계가 모델에 표현되었다. 또한 PDA를 사용하여 서비스를 제공받는 사용자가 who 객체로, 사용자가 위치하는 테마파크의 공간은 where 객체로 식별되었다.

4.3 문맥 정보 명세

문맥 정보(context information)는 문맥 객체로부터 획득한 데이터들을 이용하여 의미 있는 문맥을 인식할 수 있게 하며 명세서에서 문맥 정보의 의미나 그 구성을 상세히 나타낸다. 문맥 정보 명세서의 형식은 <표 3>에서 보이고 있다.

문맥 정보는 primitive 문맥 정보와 composite 문맥 정보로 분류한다. primitive 문맥 정보는 단일의 문맥 객체에서 1차적으로 획득되는 정보이고, composite 문맥 정보는 1가지 이상의 문맥 정보를 통합·추론하여 얻을 수 있는 고수준의 논리적인 정보를 의미한다. composite 문맥 정보는 primitive 문맥 정보뿐 아니라 composite 문맥 정보의 조합으로 표현할 수 있으므로 요구사항 수준의 추상적 의미의 정보도 인식할 수 있다. 이러한 문맥 정보의 통합 구성정보와 데이터의 형식은 *Value Driven Rule*에서 정의된다.

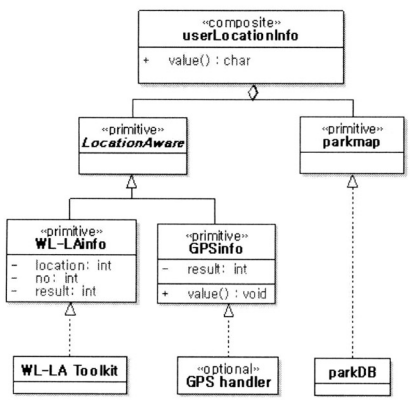
Value Driven Rule이란 두 가지 이상의 문맥 정보들이 어떤 규칙에 의해 통합되어 새로운 문맥 정보의 값을 추론할 때 그 구성 규칙을 의미한다. 이것은 문맥들을 object로 표현하여 OCL(Object Constraint Language)을 사용해서 작성할 수 있다. 문맥 정보 다이어그램(*Context information diagram*)은 두 가지 이상의 문맥 정보로 조합된 고차원 수준의 문맥 정보의 구성을 도식화한 것이다. 문맥적 요구사항(CR) 문서에서 도출된 문맥 정보의 대부분은 시스템이 인식하기에 불분명한 추상적인 용어이거나 고차원 수준의 추론이 필요한 정보이다. 따라서 시스템이 추상적이거나 고차원 수준의 문맥 정보를 저차원 수준의 물리적 문맥 정보들을 토대로 추론할 수 있도록 분석되어야 한다. (그림 6)에서 보는 것과 같이 문맥 정보 다이어그램은 UML의 클래스 다이어그램의 구성요소를 간소화하였다. 다이어그램의 기본 단위는 클래스이고 클래스는 문맥 정보나 문맥 객체를 나타낸다. 문맥 정보나 문맥 객체들 사이에서 나타나는 관계는 composition, realization, generalization이다. composition관계는 상위 수준의 정보를 구성하는 하위 수준 정보의 관계를 나타내고, realization관계는 가장 하위 수준으로만 나타나는 문맥 객체와 이로부터 획득되는 primitive 문맥 정보의 관계를 나타낸다. generalization관계는 특정 문맥 정보의 획득을 위한 여러 가지 대체될 수 있는 문맥 객체가 존재하고 있어서, 상황에 적응하거나 혹은 프로덕트 라인 핵심자산의 가변점으로써 추상화된 문맥 정보에 대해 구체화될 수 있는 정보와의 관계를 나타낸다. <표 4>는 ParkInfoView요구사항에 대해 문맥 정보 명세서를 작성한 예이다.

5. 컴포넌트 뷰

컴포넌트 뷰는 논리적 측면에서 단위 기능을 제공하는 컴포넌트로 구성되며 컴포넌트 구조(structure) 모델과 컴포넌트 행위(behavior) 모델로 소프트웨어를 나타낸다. 소프트웨어의 적응은 소프트웨어를 구성하는 컴포넌트들의 상호작용을 재구성하거나 컴포넌트 내부 요소의 변경으로 컴포넌트

〈표 4〉 ParkInfoView 요구사항의 문맥 정보 명세서

Spec No	CIS_parkInfoView		
Spec Name	parkInfoView	Related CR	parkInfoView
Context information diagram		Context information specification	



(그림 7) 사용자 위치정보의 구성을 보이는 문맥 정보 다이어그램

Context info ID	CI_01
Context info name	UserLocationInfo
Description	사용자의 공원 내 위치
CV_property	Common
Req_s#	ParkInfoView.A
Value type	(x:int,y:int)
Value Driven Rule	

Context UserLocationInfo::value() : (x:int,y:int)
post:
result=mapping(LocationAware.value(), parkmap.value())
Context LocationAware::value() : (x:int,y:int)
post:
[v1]result=transform(WL-LAinfo.value(), geoCoordinate)
[v2]result=GPSinfo.value()

상태를 변경시킴으로 수행된다. 소프트웨어는 관심 있는 문맥의 변화에 대해 적절한 적응을 하기 위해 필요한 모든 후보 컴포넌트를 포함하고 있어야 하며 컴포넌트 자체도 적응에 필요한 모든 후보 요소들을 미리 갖추고 있어야 한다. 소프트웨어의 적응 행위는 적응 규칙(Adaptation Rule)에 의해 가동되고 이행된다. 컴포넌트 뷰는 컴포넌트 구조 모델 및 행위 모델과 함께 컴포넌트에 대한 명세서를 포함한다.

5.1 컴포넌트 가변성

공통성과 선택성을 표현하는 상위 수준의 가변성은 컴포넌트 구조모델에서 표현된다. 상세 수준의 가변성은 컴포넌트 행위모델과 컴포넌트 명세에서 표현된다. 컴포넌트 뷰에서 나타나는 가변성 유형은 다음과 같다.

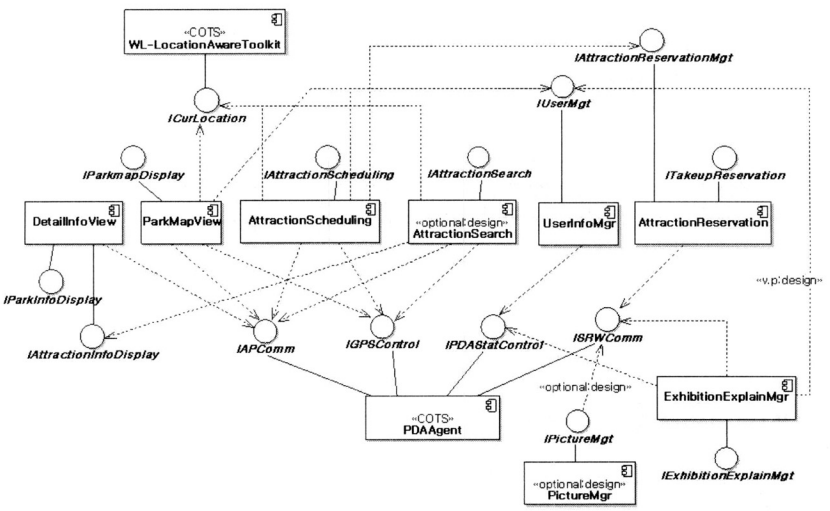
- **컴포넌트 속성 가변성(상위 수준)** - 컴포넌트가 선택적으로 나타날 수 있는 경우 첫 번째 가변성 유형이 적용된다.
- **컴포넌트 관계 가변성(상위 수준)** - 컴포넌트 사이에 존재하는 관계가 가변적으로 나타나는 경우이다. 이는 제공하는 기능의 일부의 차이나 컴포넌트의 선택성으로 인해 파급되어 생성될 수 있는 가변성 유형이다.
- **인터페이스 속성 가변성(상위 수준)** - 컴포넌트가 제공하는 기능이 가변되면 그 기능을 위해 제공되는 인터페이스가 가변성을 가지게 된다.
- **컴포넌트 상호작용 가변성(상세 수준)** - 각 컴포넌트들은 그들이 서로 상호작용하여 사용자에게 제공하는 기능을 실현하는데, 컴포넌트들 사이의 흐름의 패턴에 가변이 나타날 수 있다.
- **오퍼레이션 속성 가변성(상세 수준)** - 컴포넌트의 세부적인 기능이나 액션이 가변적이고 이를 하나의 오퍼레이션으로 구현하고 있을 경우 그 오퍼레이션은 가변성을 가지게 된다.

5.2 컴포넌트 구조 모델

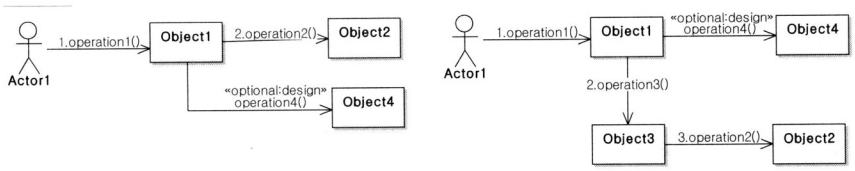
컴포넌트 구조 모델은 소프트웨어를 구성하는 컴포넌트의 전체적 구성을 나타낸다. 구조 모델을 구성하는 기본 단위는 컴포넌트와 컴포넌트가 가지고 있는 인터페이스이다. 인터페이스는 컴포넌트의 기능을 사용할 수 있도록 제공하는 오퍼레이션의 집합이다. 구조 모델에서 나타나는 관계는 컴포넌트와 인터페이스 사이의 realization관계와 컴포넌트와 컴포넌트 사이의 dependency 관계가 있다. 컴포넌트는 인터페이스를 구현하고 있는 실체화(realize) 관계를 가지며 다른 컴포넌트에 의존(depend on)하고 있는 컴포넌트는 다른 컴포넌트가 가지고 있는 인터페이스와 dependency 관계가 있다. 구조 모델은 UML의 컴포넌트 다이어그램을 사용하여 표현한다.

컴포넌트 뷰에서 나타나는 가변성은 바인딩 타임에 따라 설계 시점 가변성과 런타임 시점 가변성으로 분류된다. 바인딩 타임은 가변성을 나타내는 스테레오타입 뒤에 명시한다. 컴포넌트 뷰에서 나타나는 모든 가변성은 바인딩 타임을 함께 표기하여 개별 소프트웨어의 개발(instantiation)시 혼동을 하지 않도록 해야 한다. 프로덕트 라인 아키텍처를 재사용하기 위해 가변성의 바인딩 결정을 할 때 런타임 시점 가변성에 대해서는 엔지니어가 결정을 하지 않는다. 런타임 시점 가변성은 문맥의 변화로 인해 소프트웨어가 적응을 해야 할 필요가 있을 때 문맥에 적합한 가변치로 소프트웨어 스스로 바인딩 결정을 한다.

(그림 8)은 테마파크 PDA 시스템 도메인의 컴포넌트 구조 모델을 보인다. 시설물 검색 기능을 제공하는 *AttractionSearch* 컴포넌트와 사용자의 사진 관리 기능을 제공하는 *PictureMgr*



(그림 8) 테마파크 PDA 시스템 도메인의 컴포넌트 구조 모델



(그림 9) 컴포넌트 행위 모델

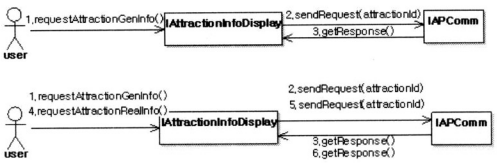
컴포넌트가 선택성을 가진 컴포넌트이며 설계 바인딩 타임을 가짐을 알 수 있다. *ExhibitionExplainMgr* 컴포넌트의 *UserInfoMgr* 컴포넌트에 대한 의존관계가 가변적임을 통해 컴포넌트 구조 모델에서는 구체적으로 볼 순 없지만 *Exhibition ExplainMgr* 컴포넌트가 제공하는 어떤 오퍼레이션이 가변성을 가지고 있음을 알 수 있다.

5.3 컴포넌트 행위 모델

적응형 소프트웨어 아키텍처의 행위 모델은 구성하는 컴포넌트들 간의 상호작용을 나타낸다. 컴포넌트들 간의 상호작용은 사용자를 위한 기능을 실행하기 위한 컴포넌트의 동작 과정 및 오퍼레이션의 수행 순서를 의미한다. 행위 모델을 구성하는 기본 단위는 컴포넌트 인스턴스와 컴포넌트 인스턴스들 사이에서 주고받는 메시지가 된다. 컴포넌트 인스턴스는 생성된 소프트웨어가 운영되는 중에 만들어지는 실체화된 컴포넌트의 개체이다. 행위 모델에서 나타나는 관계는 컴포넌트 인스턴스 사이에서 메시지들이 오고 감을 나타내는 링크이다. 링크된 컴포넌트 인스턴스 사이에는 순서와 방향이 함께 나타난 메시지가 표시된다. (그림 9)처럼 UML 인터랙션 다이어그램을 사용하여 컴포넌트의 상호작용을 모델링할 수 있다. 행위 모델에 나타나는 가변성은 컴포넌트 상호작용 가변성이 있다.

컴포넌트 상호작용 가변성은 컴포넌트 사이의 메시지 흐름 패턴이 가변적으로 결정될 수 있는 경우이다. 메시지 흐름의 패턴은 여러 상황에서 가변이 될 수 있다. 메시지가 선택적으로 나타나는 컴포넌트 인스턴스로 가는 흐름이 존재하면 그 흐름도 가변적으로 나타나게 된다. 인터페이스에 포함된 선택적 속성을 가진 오퍼레이션으로부터 다른 컴포넌트 인스턴스로 메시지가 전송되는 경우에도 흐름은 가변적으로 발생할 수 있다. 또한 메시지를 주고받는 양쪽 컴포넌트는 공통적이지만 메시지의 흐름 순서 자체가 가변이 될 수 있다. 이러한 모든 상황에 대해 가변적인 메시지 흐름 패턴이 발생하면 모델에 표현되어야 한다. 메시지 흐름의 패턴 가변성도 메시지에 스테레오타입 <<optional>> 또는 <<v.p>>를 표기하여 나타낸다. (그림 9)의 왼쪽 모델은 Object1에서 메시지가 Object2로 직접 가는 패턴을 보이고 있고 오른쪽 모델은 Object1의 메시지가 Object3로 전달되어 Object2로 가는 다른 패턴을 보이고 있다. 이때 Object4로의 메시지 흐름은 선택적으로 반영될 수 있음을 <<optional>>을 표기하여 표현하고 있다.

(그림 10)는 *ParkInfoView* 요구사항의 시설물 상세정보 안내 기능을 위한 컴포넌트 상호작용의 2가지 패턴을 보여주는 컴포넌트 행위 모델이다. 시설물 상세정보 보기는 수신감도가 좋지 않은 상태에서는 기본적인 패턴인 시설물의 일반정보 보기만을 지원한다(그림 10의 위). 수신감도가 양호할 경우는 빠른 응답이 가능하기 때문에 시설물의 기본적 정보 보기와 더불어 실시간 정보 보기를 제공할 수 있다(그림 10의 아래).



(그림 10) 시설물 상세정보 보기를 위한 컴포넌트 행위 모델 패턴

5.4 컴포넌트 명세

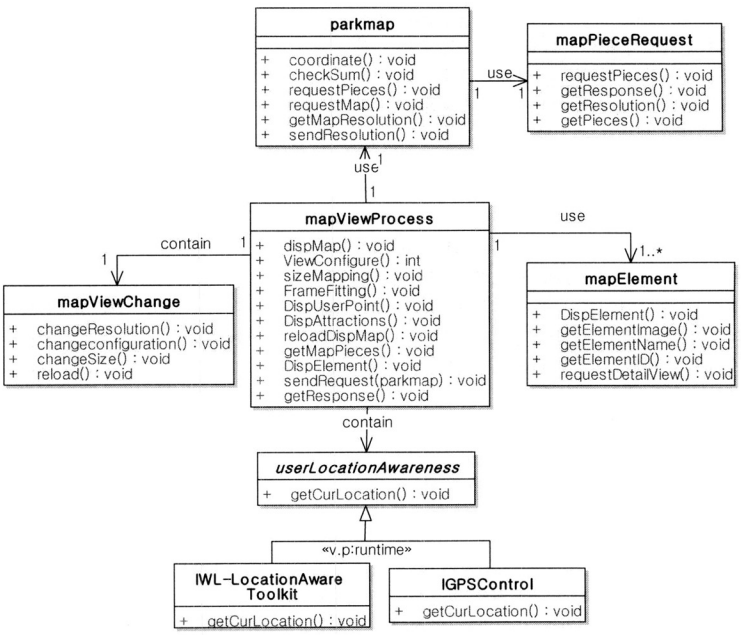
컴포넌트들은 그것이 제공하는 기능, 인터페이스, 오퍼레이션의 기술과 함께 컴포넌트 내부 구현 시의 존재하는 가변성에 대해 자세히 기술되어야 한다. 테마파크 PDA 시스템 도메인에서 추출된 *ParkmapView* 컴포넌트의 명세서의 일부를 <표 5>에서 보이고 있다. 이 컴포넌트는 공통성을 가지며 *ParkInfoView* 요구사항의 일부 기능을 구현한다. 컴포넌트의 내부 구현할 때 가변되는 부분이 있는데, 요구사항 시나리오 A항목에 기술된 '*사용자 위치정보*'를 얻기 위해 필요한 컴포넌트가 문맥의 변화에 의해 달라짐을 Variation Point 항목에서 기술하고 있다. 이 컴포넌트를 구현하는데 포함되는 클래스의 구성을 (그림 11)에서 클래스 다이어그램으로 작성하고 기능을 수행할 때 클래스의 상호작용 행위를 (그림 12)에서 상호작용 다이어그램으로 작성하였다.

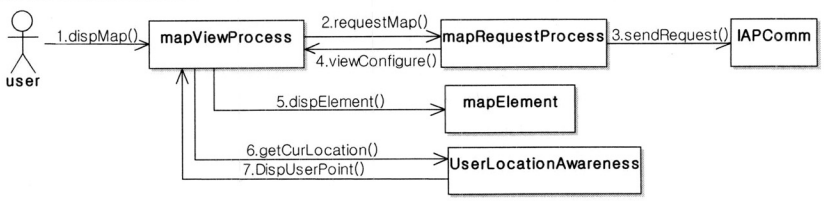
6. 적응 규칙 명세

적응 규칙은 문맥 변화가 발생하였을 때 소프트웨어의 동작이나 상태를 재구성하는 행위를 정의한다. 문맥 정보 뷰에 명시된 문맥 정보의 발생 이벤트로 인해 관련 규칙의 적용이 판단되면 소프트웨어의 동적 재구성을 가동하고 제어한다. 적응 규칙은 문맥 정보와 컴포넌트와의 관계를 정의하여 적응형 소프트웨어로서 문맥에 반응하여 재구성하는 기능을 갖추도록 하는 전략이다.

적응 규칙 명세도 프로덕트 라인 아키텍처의 부분으로서

〈표 5〉 ParkmapView 컴포넌트 명세서

<table>
<tr><td>Name</td><td colspan="2">ParkmapView</td><td>CV_property</td><td>Common</td></tr>
<tr><td>Description</td><td colspan="4">사용자는 PDA기기를 사용하여 테마파크의 지도를 볼 수 있다. 지도에는 사용자의 현재 위치와 각종 시설물의 위치를 볼 수 있다.</td></tr>
<tr><td>Related CR</td><td colspan="4">ParkInfoView</td></tr>
<tr><td rowspan="2">Interface description</td><td>Provided interface</td><td colspan="3">IParkmapDisplay</td></tr>
<tr><td>Required interface</td><td colspan="3">IAPComm: 무선랜통신을 위한 AP와의 통신 인터페이스
IGPSControl: GPS정보를 받기 위한 인터페이스
IUserMgt: 사용자의 정보를 관리하는 인터페이스
ICurLocation: 무선랜 통해 사용자 위치정보를 인식하는 인터페이스</td></tr>
<tr><td rowspan="4">Variation Point</td><td>ID</td><td colspan="3">[runtime]CurrentLocationAwareness</td></tr>
<tr><td>Description</td><td colspan="3">사용자 위치정보를 얻기 위한 타 시스템 사용 가변성-날씨상태 등에 따라 런타임시 바인딩됨</td></tr>
<tr><td>Related scenario no.</td><td colspan="3">ParkInfoView.A.사용자 위치정보</td></tr>
<tr><td>constraints</td><td colspan="3">Multiplicity:1 / Relation: [v1] exclusiveOR [v2]
{[v1]IGPSControl.getCurLocation(),
[v2]IWS-LocationAwareToolkit.gotCutLocation()}</td></tr>
<tr><td>Component Specification Model</td><td colspan="4">Structure model
 

(그림 11) ParkmapView컴포넌트의 클래스 구조 모델
Behavior model
 

(그림 12) ParkmapView컴포넌트의 클래스 상호작용 모델</td></tr>
</table>

〈표 6〉 적응 규칙 형식

```
1_level_varExpr  rule    ruleName {
2_level_varExpr  when    variantsID   contextInformationExpr
2_level_varExpr  if      variantsID   currentStateExpr
2_level_varExpr  then    variantsID   reconfigurationActionExpr
}
```

〈표 7〉 테마파크 PDA 시스템 도메인의 적응 규칙 명세

```
rule  Location_r1{
when (weatherStatus.value() = 'veryCloudy'
or    weatherStatus.value() = 'heavyRainy')
and   IGPSControl.getCurLocation() is null
then CurrentLocationAwareness->ICurLocation.getCurLocation()
}
[optional]rule Handicap_r1{
when userProfile.handicap = 'hearing_difficulty'
then  IPDAStatControl.setSound(off)
and   ISpeech-charTrans.initial()
}
rule  Reception_r1{
when receiveSensitivityStatus.value() = 'poor'
[v.p] then
 [v1]RequestGenInfo/RealInfoPattern->RequestAttractionGenInfoPattern
 [v2]IPDAStatControl.NotifyImage("reception is poor")
}
```

프로덕트 멤버들 간의 공통으로 적용되는 부분과 가변되는 부분을 명시적으로 나타내어야 한다. 적응 규칙의 가변성은 두 가지 상세화 수준의 가변성 유형을 가진다. 첫 번째는 적응 규칙 자체에 대한 공통성 혹은 선택성의 속성을 결정하는 것이고, 두 번째는 조건문의 연산에서 가변성이 나타나게 된다.

적응 규칙에서는 문맥 정보 뷰와 컴포넌트 뷰에서 나타난 모델의 요소를 사용하여 기술하며 이들 요소들은 이미 공통성 및 가변성이 결정되어 있다.

따라서 선택적인 속성을 가지거나 가변점을 포함하는 모델 요소가 적응 규칙에 기술되었다면 그들의 가변성에 대한 결정에 따라 그 적응 규칙에도 변형이 발생한다. 예를 들어 선택성을 가진 요소가 제품 개발 단계에서 선택되지 않음으로 결정되면 그 요소를 이용한 조건이나 행위를 기술한 적응 규칙에서도 제거된다. 가변성을 포함하는 적응 규칙의 표현 형식은 아래의 <표 6>과 같다.

rule절에서는 정의하는 적응규칙의 이름을 작성하게 되며 rule 식별자 앞에 공통성 혹은 선택성의 속성을 명시해준다. 이후 when, if, then의 항목에서 가변점을 포함하고 있을 경우에 각 식별자 앞에 가변성 표현을 명시할 수 있다.

When절에서는 해당 rule을 가동시키는 조건을 기술한다. 이 조건은 획득한 문맥 정보가 임계범위의 초과 여부를 결정한다. 결정의 결과에 의해 문맥 변화 발생을 인식한다.

if 절에는 현재 상태가 소프트웨어의 재구성이 필요한 상태인지 아닌지를 판단하는 조건문을 기술한다.

then 절은 규칙이 가동되었을 때 수행될 재구성 행위들을 기술한다. 1_level_varExpr은 적응 규칙의 공통성 혹은 선택성의 속성에 따라 [common]또는 [optional]로 대체될 수 있다. 2_level_varExpr은 규칙의 연산에 가변성 포함여부에 따라 [v.p]로 대체되어 가변점임을 표현할 수 있으며 가변점에 대응될 수 있는 가변치들을 나타내기 위해 [v1], [v2],...,[vn] 식으로 variantsID의 위치에 표기한다.

<표 7>는 테마파크 PDA 시스템 도메인의 적응 규칙 명세의 일부분이다. 첫 번째 적응 규칙 Location_r1은 날씨 상태에 따라 시계가 좋지 않을 때에는 GPS 정보가 정확하지 못하기 때문에 무선랜 위치 인식 장치로 변경하여 서비스를 제공하게 한다. 두 번째로 정의된 Handicap_r1 적응 규칙은 청각 장애자로 정보가 입력될 시에 배터리 소모를 줄이기 위해 PDA사운드를 자동적으로 사용안함 상태로 바꾸게 한다. 이 규칙은 시스템이 장애자의 지원이 가변적이라는 속성 때문에 규칙 자체도 가변 속성을 가지게 된다. 따라서 장애자를 지원하지 않는 시스템이면 이 규칙은 그 시스템의 적응 규칙에 포함되지 않게 된다. 마지막에 나타난 적응 규칙 Reception_r1은 수신전파의 세기가 약한 위치에서는 PDA의 정보 요청에 대해 느린 응답속도가 나타날 것이기 때문에 DB 접속에 많은 시간이 소요되는 실시간 정보의 요청이 제외된 프로세스로 바꾸도록 한다.

〈표 8〉 관련연구와의 비교평가

관련 분류	비교항목	DDA[10]	MADAM[12]	FORM[8]	KobrA[1]	본 연구
적응형 소프트웨어 특징	정형화된 문맥 정보 및 표현 구조 제시	X	X	X	X	○
	고수준 문맥 정보 표현 지원	X	X	X	X	○
	적응 단위의 상세 수준	중간	낮음	지원안함	지원안함	중간
	적응 전략 제시	○	○	X	X	○
	적응 대안 구조의 명시적 표현	X	△	△	X	○
아키텍처 특징	관점 분리의 아키텍처 뷰 제시	X	X	○	○	○
	아키텍처 모델의 템플릿 제시	X	X	○	○	○
프로덕트라인 특징	가변점의 상세정보 명시	△	△	X	○	○
	모델과 명세에서 가변성의 명확한 표현 제시	X	△	△	○	○
	자산별 가변성 유형 분류	X	X	X	X	○

○ 명확하게 지원함
△ 제시하지만 미흡함
X 존재하지 않음

7. 관련 연구

7.1 소프트웨어 프로덕트 라인 공학

KobrA(Komponentenbasierte Anwendungsentwicklung)[1]는 컴포넌트를 식별하고 실체화하기까지의 개발과정에서 산출되는 모델 및 명세에서 공통성과 가변성을 식별하여 표현하고 있으나, 요구사항을 도출하고 정의하는 단계가 명확히 제시되지 않아서 요구사항 산출물에서 나타나는 가변성에 대해서도 다루지 못하고 있다. FORM(Feature-Oriented Reuse Method)[8]은 도메인 자산에서의 가변성을 명시적으로 표현하지 못하며 피처 그래프와의 매핑을 통해 암시적으로 가변성을 다루고 있다[9].

이들은 모두 일반적인 소프트웨어에 대한 프로덕트 라인 아키텍처를 제시하고 있으므로 적응형 소프트웨어의 개발에 적용하기에는 한계가 있다. 따라서 적응형 소프트웨어의 중요한 특징인 문맥을 분석하는 활동과 소프트웨어의 재구성을 위한 구조 및 재구성 행위에 대한 표현을 지원하는 프로덕트 라인 아키텍처가 제시되어야 할 필요가 있다.

7.2 적응형 소프트웨어

기존의 적응형 소프트웨어에 대한 연구들은 프로그래밍 언어적 기술이나 알고리즘에 기반한 소프트웨어의 적응을 제시하고 있다. 가변적인 구현 부분이 외부 조건에 따라 선택되고 대체되는 DDA(Dynamic Domain Architecture)[10]가 대표적이다. 아키텍처 기반으로 소프트웨어의 적응을 제어하는 Rainbow Project[11]는 범용적 미들웨어로서 아키텍처 모델을 제시하지만 추상적이며 이를 구현하기 위한 구체적인 구성요소와 구조를 제시하지 않는다. MADAM(Mobility and adaptation-enabling middleware) Project[12]도 외부적 적응 기법(external adaptation approach)을 통해 미들웨어 기반의 적응 방식을 지원하는 아키텍처를 제시한다. 그러나 Rainbow Project와 마찬가지로 개발 시 적용하기에는 추상적이다.

7.3 비교 평가

<표 8>은 본 연구와 관련연구의 비교 결과이다. 각 비교 항목은 적응형 소프트웨어에서 필요한 특징, 프로덕트 라인 아키텍처로서 갖춰야 할 특징과 아키텍처로서 필요한 특징을 추출한 것이다.

본 연구는 관점의 분리를 지향한 아키텍처 뷰와 구체적인 모델의 템플릿을 제시하여 개발에 적용 가능한 수준으로 나타내었다. 즉, 고수준 문맥을 포함하는 문맥 정보의 정형화된 표현 구조를 제시하였으며, 아키텍처 뷰 모델의 산출물 템플릿을 제공한다. DDA와 MADAM은 이러한 문맥 정보의 표현을 자체적으로 지원하지 않고 있다. 본 연구의 적응 단위 수준은 컴포넌트 및 인터페이스와 오퍼레이션 단위로 제어할 수 있지만 코드 수준까지 지원하지 않으므로 '중간'으로 평가되었다. DDA는 코드 혹은 알고리즘 대체를 통한 접근방식이므로 아키텍처 수준에서 적응력을 제공하지 못하므로 '중간'으로 평가되었다. MADAM project는 컴포넌트 타입 단위에서 소프트웨어의 적응을 지원하며 대체가능 컴포넌트 타입을 명시적으로 나타내고 있다. 그러나 컴포넌트 단위의 적응 구조는 세부적인 변화를 지원하지 못할 수 있으며 컴포넌트 내부 요소가 중복될 수 있으므로 적응 단위 수준이 '낮음'으로 평가되었다.

본 연구에서는 프로덕트 라인의 재사용성을 위해 아키텍처 요소의 가변성에 대한 명확한 표현 및 상세 정보를 나타내고 있다. 가변성의 설계는 프로덕트 라인의 핵심 부분이며 다양한 가변 요소와 정확한 정보를 제공할수록 프로덕트

라인의 효율성이 증가하기 때문에 가변성 정보의 표현은 본 연구의 중요한 특징이다. 또한 개발 자산의 수준에 따라 다르게 나타나는 가변성의 유형을 분류하여 가변성의 식별을 편리하게 한다는 것도 본 연구의 장점이라 할 수 있다.

8. 결론 및 향후 연구

본 논문에서는 적응형 소프트웨어의 아키텍처를 구성하는 산출물을 정의하고 프로덕트 라인 아키텍처로서 재사용을 지원하기 위해 아키텍처 요소의 공통성과 가변성을 명시적으로 표현하였다. 적응형 소프트웨어의 아키텍처 뷰는 문맥적 요구사항 뷰, 문맥 정보 뷰, 컴포넌트 뷰, 적응 규칙 명세로 구성된다. 문맥적 요구사항 뷰는 소프트웨어의 기능적 요구사항과 더불어 기능 수행시 상호작용 하는 문맥에 대한 반응을 정의한다. 문맥 정보 뷰에서는 관심있는 문맥 정보를 분석하고 그것을 얻기 위해 필요한 문맥 객체의 모델을 정의한다. 소프트웨어 기능을 수행하고 문맥의 변화에 반응하게 되는 컴포넌트의 구조와 행위를 표현하는 컴포넌트 뷰와 문맥의 변화에 적응하는 행위를 정의한 규칙 명세로 적응형 소프트웨어의 전반적 구조를 나타낼 수 있다. 또한 각 뷰에서 제시되는 산출물들은 프로덕트 라인의 핵심자산으로 공용될 수 있도록 자산 수준별, 상세화 수준별로 식별된 가변성이 명확히 반영됨을 보였다. 향후 연구 과제로는 제시된 적응형 소프트웨어의 프로덕트 라인 아키텍처의 개발을 지원하는 도구를 개발하여 각 모델 간의 가변성 연관이나 자산 관리 및 재사용을 자동화하는 것이 요구된다.

참 고 문 헌

[1] Atkinson, C. et al., *Component-based product line engineering with UML*. Addison-Wesley, London, New York, 2002.

[2] Self Adaptive Software, DARPA, BAA 98-12, Proposer Information Pamphlet, 1997.

[3] Clements, P., Garlan, D., Little, R., Nord, R., and Stafford, J., *Documenting software architectures: views and beyond*, Addison-Wesley, September 2002

[4] IEEE, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. Institute of Electrical and Electronics Engineers, Sept. 2000. IEEE Std 1471-2000.

[5] 문미경, "소프트웨어 프로덕트 라인에서 가변성 분석을 통한 요구사항 및 아키텍처 개발," 부산대학교 컴퓨터공학과 박사 학위 논문, 2005.

[6] Pohl, K., Bockle, G., and van der Linden, F., *Software Product Line Engineering: Foundations, Principles, and Techniques*, Springer, 2005.

[7] Dey, A. K. and Abowd, G. D., "Towards a Better Understanding of Context and Context-Awareness," Technical Report, GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999.

[8] Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., and Huh, M., "FORM: A Feature-Oriented Reuse Method with Domain Specific Reference Architectures," *Annals of Software Engineering*, Vol.5, No.1, 1998, pp.143-168(26).

[9] 문미경, 염근혁, "소프트웨어 프로덕트 라인에서 가변성 분석을 통한 도메인 아키텍처 개발 방법," *정보과학회논문지: 소프트웨어 및 응용* 제34권 제4호, 2007, pp.328-341.

[10] Laddaga, R., Robertson, P., and Shrobe, H. E., "Probabilistic dispatch, dynamic domain architecture, and self-adaptive software," *Self-Adaptive Software*, pp.227-237. Springer-Verlag, 2003.

[11] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., and Steenkiste, P., "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," *IEEE Computer* Vol.37 No. 10, october 2004, pp.46-54.

[12] Hallsteinsen, F. J., Stav, S., Eliassen, E., Lund, F., Gjorven, K. E., "Using architecture models for runtime adaptability," *IEEE Software*, March-April 2006 Vol.23, pp.62-70.

예 은 숙

e-mail : yes27@pusan.ac.kr
2003년 부경대학교 전자계산학과(학사)
2008년 부산대학교 컴퓨터공학과(석사)
관심분야 : 소프트웨어 프로덕트 라인 공학, 적응형 소프트웨어 개발, 비즈니스 프로세스 모델링, SOA 등

염 근 혁

e-mail : yeom@pusan.ac.kr
1985년 2월 서울대학교 계산통계학과 (학사)
1992년 8월 Univ. of Florida 컴퓨터공학과 (석사)
1995년 8월 Univ. of Florida 컴퓨터공학과 (박사)
1985년 1월~1988년 2월 금성반도체 컴퓨터연구실 연구원
1988년 3월~1990년 6월 금성사 정보기기연구소 주임연구원
1995년 9월~1996년 8월 삼성SDS 정보기술연구소 책임연구원
1996년 8월~현 재 부산대학교 정보컴퓨터공학부 교수
관심분야 : 소프트웨어 재사용, 프로덕트 라인 공학, 소프트웨어 아키텍쳐, 컴포넌트 기반 소프트웨어 개발, 적응형 소프트웨어 개발, RFID기반 미들웨어 등

문 미 경

e-mail : mkmoon@dongseo.ac.kr

1990년 이화여자대학교 전자계산학과 (학사)

1992년 이화여자대학교 전자계산학과 (석사)

2005년 부산대학교 컴퓨터공학과 (박사)

2005년 3월~2005년 8월 부산대학교 차세대물류IT기술사업단 박사후 연구원

2005년 9월~2006년 8월 부산대학교 컴퓨터 및 정보통신 연구소 기금교수

2006년 9월~2008년 2월 부산대학교 정보컴퓨터공학부 연구교수

2008년 3월~현 재 동서대학교 컴퓨터정보공학부 전임강사

관심분야 : 소프트웨어 프로덕트 라인 공학, 적응형 소프트웨어 개발, RFID 미들웨어 개발 및 RFID 기반 애플리케이션 개발 등