UML 2.0 기반의 가변요소 추출 및 표현 방법

최유희° 하수정 차정은 박창순

한국전자통신연구원 임베디드SW연구단

{yhchoi°, hsj, mary2743, cpark}@etri.re.kr

An approach to the identification and representation of variant elements based on UML 2.0

Youhee Choi° Sujung Ha, Jungeun Cha, Changsoon Park

Electronics and Telecommunications Research Institute Embedded S/W Division

요 약

제품계열 기반 개발 방법은 특정 제품 개발 시, 제품 계열 아키텍처를 요구사항에 따라 재정의하여 제품 아키텍처를 정의하고, 이에 따라 필요 기능의 컴포넌트를 조정, 조립 또는 신규 개발하여 제품의 빠른 생산을 가능하게 할 수 있는 방법으로 주목 받고 있는 기술이다. 제품계열 아키텍처로부터 제품 아키텍처를 쉽게 생성할 수 있도록 지원하기 위해서는 제품 계열 내의 여러 시스템에서 공통인 부분과 제품에 따라 다른 가변 부분의 표현이 명확해야 한다. 그러나 기존 연구들은 가변성의 이해 및 표현이 어렵거나 범용적이지 못하다는 등의 단점들이 존재한다. 따라서 본 연구에서는 UML 2.0을 이용하여 제품 계열 아키텍처의 가변 요소을 추출하고 표현하는 방법을 제안한다.

1. 서 론

컴포넌트 기반 개발(CBD)은 기능성의 단위 부품인 컴포넌트 개발에만 치중함으로써, 실제 S/W 생산을 위한 컴포넌트의 조립과 적용 시 많은 부가 비용을 요구한다. 이러한 이유로 다양한 컴포넌트를 조립하여 시스템을 개발하는데 중요한 설계 문제로 대두된 것이 소프트웨어 아키텍처이다. 이러한 추세에 따라 대표적인 S/W 아키텍처 기반 개발 방법론 중 하나인 제품계열 기반 개발 방법이 주목 받고 있다. 제품계열은 S/W 아키텍처 등의 핵심 자산을 구축하고 이를 재사용하여 S/W를 개발 하는 방식으로 특정 S/W 제품계열 내의 전체 제품들을 대표하는 S/W 아키텍처를 제품계열 아키텍처라고 한다. 특정 제품 개발 시, 제품계열 아키텍처를 요구사항에 따라 재정의 하여 제품 아키텍처를 정의하고, 이에 따라 필요 기능의 컴포넌트를 조정, 조립 또는 신규 개발하여 제품의 빠른 생산을 가능하게 할 수 있다. 즉, 제품계열 아키텍처는 하나의 제품 계열 내에 존재하는 다양한 제품들간의 공통성 활용의 장점을 극대화하면서 이들간의 차이점을 보장해주어야 한다. 따라서 제품 계열 아키텍처로부터 제품 아키텍처를 쉽게 만들 수 있도록 지원하기 위해서는 공통성과 가변성이 제품계열 아키텍처에 명시적으로 표현되어야 한다. 이를 위해 기존의 S/W 아키텍처 표현 방법에는 표준 모델링 언어인 UML과 ADL이 있고, 기존 여러 연구[1,2,3]에서 공통성과 가변성 표현에 대한 연구가 이뤄졌다. 그러나 기존 ADL에서는 공통성과 가변성 표현이 지원되지 않으며[1], UML을 이용한 공통성과 가변성 연구에서는[2,3], UML1.x을 기반으로 하여 소프트웨어 아키텍처 관점의 공통성과 가변성의 표현이 어렵고, 또한 공통성과 가변성 관리 및 추출 방법이 고려되지 않았다.

따라서 본 논문에서는 UML 2.0[4]을 기반으로 제품 계열 아키텍처를 모델링하기 위해서 가변요소를 추출하는 방법과 가변 요소들간의 의존관계를 형성하는 방법에 대해 제안하고, UML 2.0을 기반으로 제품 계열 아키텍처를 표현하는 방법에 대해 제안한다.

2. 관련 연구

2.1. UML 2.0을 이용한 아키텍처 모델링[5,6]

UML 2.0은 시스템의 아키텍처를 모델링할 수 있도록 지원하기 위해 컴포넌트, 커넥터, 포트와 같은 새로운 컨스트럭트를 정의하고 있으나, 커넥터의 경우, 컴포넌트간의 상호작용을 표현하기 어렵다. 이를 해결하기 위해 UML 2.0의 확장 메커니즘을 이용하여 커넥터를 표현하고자 하는 여러 연구들[5,6]이 있다. 기존 연구 [5]에서는 커넥터를 컴포넌트 메타클래스의 스테레오타입으로 정의하고 있으나 씨멘틱상의 불일치 문제가 있고, [6]에서는 커넥터를 collaboration의 스테레오타입으로 정의하고 있으며, 상호작용 패턴과 상호작용 구조를 표현할 수 있도록 지원한다. 따라서 본 연구에서도 컴포넌트간의 상호작용인 커넥터를 표현하기 위해 collaboration을 이용하고자 한다.

2.2. UML 확장 메커니즘을 이용한 가변성 표현[2]

Clauss는 UML 스테레오타입과 클래스 다이어그램을 기반으로 가변성을 모델링하는 연구를 제안하였다. 각 가변점은 <<variation point>>스테레오타입을 가진 클래스로 표현되며, 각 가변요소(variant)는 <<variant>>스테레오타입을 가진 클래스로 표현된다.

가변점과 가변요소사이의 관계는 상속 관계로 모델링되며 가변요소들간의 상호작용은 "mutex"와 "requires" 의존 관계를 이용하여 모델링된다. 또한 <<optional>>스테레오타입, tagged value와 조건이 선택적 요소를 표현하는데 사용된다. 그러나 이 연구는 클래스 추상 레벨의 가변성을 표현하고, 컴포넌트 추상 레벨의 가변성 표현을 지원하지 않는다.

2.3. 제품 계열에서의 가변성 분류[7]

이 연구에서는 제품 계열 레벨, 제품 레벨, 컴포넌트 레벨, 서브컴포넌트 레벨, 코드 레벨에서의 다양한 가변성 레벨을 분류하였고, 다양한 가변성 레벨을 다루는 방법을 제안하였으나, 다양한 가변성 레벨을 표현하는 방법을 제시하지 않았다.

3. UML 2.0 기반의 가변요소 추출 및 표현 방법

가변성은 크게 선택적 특성과 대안적 특성으로 나눌 수 있으므로, 본 연구에서는 아키텍처 주요 요소 단위별로 각 가변 특성을 나누어 추출하고 표현하는 방법을 제안한다.

3.1. 컴포넌트 분류

먼저 컴포넌트를 선택적 특성과 대안적 특성에 따라 분류한다. 이를 위해 사용자가 순차도 형태로 작성한 시나리오에 대해 각 단계별로 포함되는 컴포넌트 이름을 입력하여 컴포넌트 맵핑 테이블을 형성한다. 그림 1은 사용자로부터 입력 받는 입력물인 시나리오와 컴포넌트 맵핑 테이블과 결과물인 가변 특성별로 분류된 컴포넌트 모델을 나타낸 것이다.

컴포넌트와의 관계가 맵핑된 순차도에서 조건 선택 연산자인 opt 연산자가 사용된 경우 이를 선택적 기능으로 간주하고, alt 연산자가 사용된 경우 이를 대안 기능으로 간주한다.

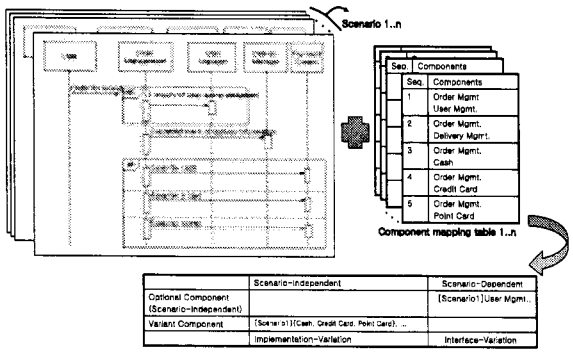

그림 1. 시나리오 분석 과정

이를 바탕으로 다음과 같은 기준으로 컴포넌트를 분류한다. opt 연산자가 사용된 단계에 맵핑된 컴포넌트가 다른 시나리오상에서 포함되어 있는지를 검사하여 다른 시나리오상에 포함되지 않는 컴포넌트인 경우에 독립 개별 기능 확장에 의한 선택 컴포넌트로 분류하고 포함되는 경우에는 시나리오상에서 선택적인 기능 수행을 위한 컴포넌트로 분류한다. alt 연산자가 사용된 단계에 맵핑된 컴포넌트들에 대해 아키텍처 모델에서의 인터페이스를 비교하여 인터페이스가 동일할 경우 인터페이스 변경 없이 구현만 변경되는 대안 컴포넌트로 분류하고, 동일하지 않을 경우 인터페이스가 변경되는 대안 컴포넌트로 분류한다.

3.2. 가변 요소들간의 관계 형성

다음으로 아키텍처 모델링 요소 중 컴포넌트는 선택적 가변 측면이나 대안적 가변 측면에 따라 다른 모델링 요소에 영향을 많이 주게 되므로 이러한 가변 요소들에 의해 영향을 받는 요소들의 관계를 관리하고 이를 표현 및 검증하는 것이 필요하다. 따라서 선택 컴포넌트와 대안 컴포넌트에 의해 영향을 받는 다른 요소들과의 의존 관계를 형성한다. 그림 2는 가변 요소 관계 형성 알고리즘을 나타낸 것이다.

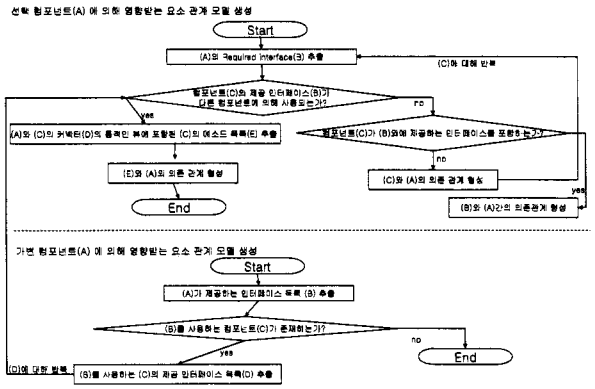

그림 2. 가변 요소 관계 형성 알고리즘

선택 컴포넌트에 의해 영향 받는 의존 관계를 형성하기 위해서 먼저 선택 컴포넌트(A)에서 사용하는 요구 인터페이스(Required Interface)를 추출한다. 그리고 해당 인터페이스(B)를 제공하는 컴포넌트(C) 측면에서 인터페이스(B)가 다른 컴포넌트에 의해 사용되는지를 확인한다.

다른 컴포넌트에 의해 사용되지 않는 경우에 해당되면 인터페이스(B)가 선택 컴포넌트(A)에 의해서만 필요한 경우에 해당된다. 이 경우 인터페이스(B)를 제공하는 컴포넌트(C) 단위에서의 의존 관계로 형성해야 되는지 확인하기 위해 컴포넌트(C)가 (B)외에 다른 인터페이스를 제공하는지 확인한다. 다른 인터페이스를 제공하지 않는 경우 컴포넌트(C) 단위에서의 의존관계를 형성하고 반대의 경우 인터페이스(B) 단위에서의 의존관계를 형성한다.

다른 컴포넌트에 의해 사용되는 경우에 해당되면

인터페이스(B)가 선택 컴포넌트(A)외에도 다른 컴포넌트에 의해서도 필요한 경우에 해당된다. 이 경우 인터페이스내의 메소드 단위의 의존 관계를 형성하기 위해 선택 컴포넌트(A)와 컴포넌트(C)와의 상호작용 부분인 커넥터(D)에서 컴포넌트(C)의 메소드 목록(E)을 추출하여 해당 메소드(E)에 대해 선택 컴포넌트와의 의존 관계를 형성한다.

다음으로 대안 컴포넌트에 의해 영향 받는 의존 관계를 형성하기 위해서 먼저 대안 컴포넌트(A)의 제공 인터페이스(Provided Interface)를 추출한다. 그리고 해당 인터페이스(B)를 사용하는 컴포넌트(C)가 존재하는지 확인하여 존재하지 않는다면 대안 컴포넌트에 의해 영향 받는 요소가 없다는 의미가 된다.

다른 컴포넌트(C)가 존재하는 경우라면 (B)를 요구 인터페이스로 사용하는 컴포넌트(C)가 존재한다는 의미가 되고 요구 인터페이스는 (C)의 제공 인터페이스(D)에 의해 필요한 것이므로 제공 인터페이스(D)를 추출한다. 컴포넌트 단위의 의존인지, 인터페이스 단위의 의존인지, 메소드 단위의 의존인지를 분류하기 위해 그림 2에서와 같이 앞 단계로 돌아가서 수행한다.

3.3. 가변 요소 표현

그림 3은 선택적 컴포넌트에 대해 아키텍처 모델의 정적인 뷰와 동적인 뷰상에서의 표현 형태를 나타낸 그림이다.

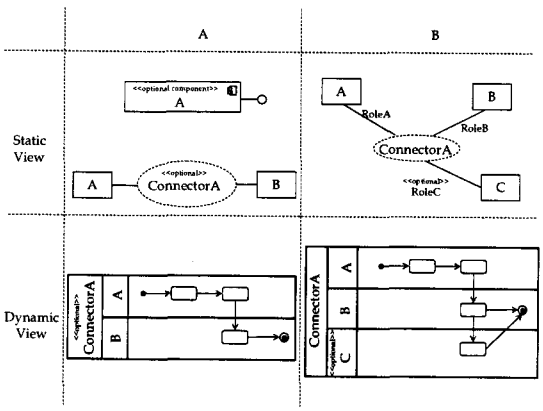

그림 3. 선택적 컴포넌트의 정적인 뷰와 동적인 뷰상에서의 표현 형태

선택적 컴포넌트인 경우는 다음과 같은 두 경우로 분류할 수 있다. 첫번째로는 해당 컴포넌트의 기능이 다른 기능들 및 다른 시나리오상에서 사용되지 않는 독립 개별 기능으로서 기능의 확장 등에 의한 경우가 있다. 두번째로는 기존의 기능에 의한 시나리오상에서 특정 컴포넌트가 수행하는 기능이 선택적인 경우이다.

첫 번째(A)의 경우, 다른 기능들과는 관계없는 독립적인 기능을 의미하는 것으로 해당 컴포넌트가 수행하는 기능이 필수 기능이 아닌 선택 기능일 경우로서 선택적인 하나의 시나리오에만 포함된 컴포넌트가 이에 해당될 수 있다. 이를 표현하기 위해 정적인 뷰에서는 해당 컴포넌트에 대해 UML의 확장 메커니즘인 스테레오타입 방식에 기반하여 <<optional component>> 스테레오타입을 정의하여 표현한다. 또한 해당 컴포넌트에 의해 시작되는 상호작용의 경우 선택적 컴포넌트의 포함 여부에 따라 영향을 받으므로 이를 표현해 줄 필요가 있다. 선택적 컴포넌트에 의해 시작되는 상호작용은 상호작용 자체가 선택적임을 의미하므로 커넥터에 선택적임을 표현하기 위해 <<optional>> 스테레오타입을 정의하여 표현한다. 동적인 뷰에서는 커넥터가 선택적이므로 해당 커넥터의 다이어그램 자체가 선택적임을 <<optional>> 스테레오타입을 지정하여 표현한다.

두번째(B)의 경우, 시나리오상에서 특정 컴포넌트가 선택적이라는 의미는 해당 컴포넌트가 시나리오상에서 하는 역할 자체가 선택적이라는 의미이므로 시나리오상에서의 역할을 표현할 때 선택적 역할(Optional Role)임을 표현함으로써 선택적 역할이 필요 없을 때 해당 컴포넌트는 시나리오상에서 제외될 수 있다는 의미로 표현될 수 있다. 즉 시나리오는 컴포넌트간의 상호 작용이므로 이는 커넥터로 표현되고, 커넥터를 표현하기 위해 사용하는 Collaboration에서 상호작용에 참여하는 역할을 나타내는 Role에 대해 <<optional>> 스테레오타입을 정의하여 나타냄으로써 선택적 역할임을 표현한다. 그리고 동적인 뷰에서도 해당 컴포넌트의 역할이 선택적임을 표현하기 위해 해당 컴포넌트의 역할 영역에 대해 <<optional>> 스테레오타입을 지정하여 표현한다.

그림 4는 대안 컴포넌트에 대해 아키텍처 모델의 정적인 뷰와 동적인 뷰상에서의 표현 형태를 나타낸 것이다.

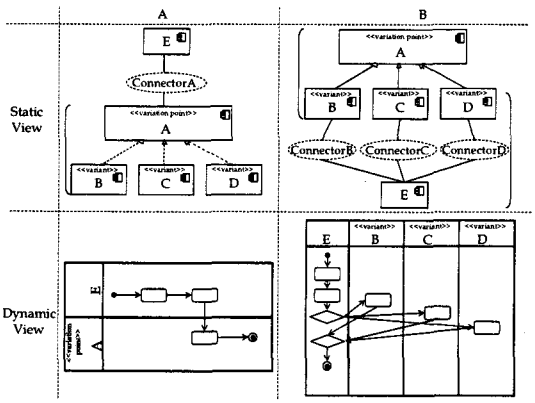

그림 4. 대안 컴포넌트의 정적인 뷰와 동적인 뷰상에서의 표현 형태

컴포넌트 단위에서의 대안인 경우는 다음과 같은 두 경우로 분류할 수 있다. 인터페이스의 변경 없이 여러 컴포넌트가 다양한 구현을 가지는 경우와 컴포넌트의 구현을 위해 요구되는 기능 및 제공하는 기능이 달라서 인터페이스가 변경되는 경우이다. 첫번째(A)의 경우, 공통의 인터페이스를 가지지만 인터페이스에 대한 구현이 다른 경우이므로 정적인 뷰에서는 대안 컴포넌트들의 그룹에 대한 하나의 추상 컴포넌트를 정의하고 이에 대해 <<variation point>> 스테레오타입을 정의하여 표현한다. 이에 대한 대안 컴포넌트들과의 관계는 Realization(실체화) 관계를 이용하여 표현한다. 이 경우 각 대안들과 다른 컴포넌트와의 상호작용은 동일한 인터페이스를 이용하므로 정적인 뷰에서 각 대안과 다른 컴포넌트와의 상호작용 대신 추상 컴포넌트와 다른 컴포넌트와의 상호작용에 대해 표현할 수 있다. 따라서 추상 컴포넌트와 다른 컴포넌트(E)와의 상호작용을 표현하는 대표 커넥터로 'ConnectorA'를 정의하여 표현한다. 또한 동적인 뷰에서는 대표 커넥터인 'ConnectorA'의 행위를 표현한다.

두번째(B)의 경우, 인터페이스가 변경되는 대안이므로 정적인 뷰에서는 대안 컴포넌트들의 그룹에 대한 하나의 추상 컴포넌트를 정의하는데 이 때, 모든 대안들이 공통으로 포함하는 인터페이스만을 가지고 있도록 정의한다. 추상 컴포넌트는 <<variation point>> 스테레오타입을 정의하여 표현한다. 이에 대한 대안 컴포넌트들과의 관계는 Specialization(상세화) 관계를 이용하여 표현하고 각 대안별로 추가적으로 필요한 인터페이스를 포함하도록 표현한다. 또한 첫번째 경우와 다르게 인터페이스가 변경될 수 있는 경우이므로 각 대안들마다 다른 컴포넌트와의 상호작용이 다를 수 있으므로 각각의 상호작용에 대해 각각의 대안에 대해 커넥터를 정의하여 표현한다. 동적인 뷰에서는 각각의 커넥터('ConnectorB', 'ConnectorC', 'ConnectorD')에 대해 하나의 독립된 다이어그램으로 나타낼 수 있고 또는 각 커넥터에서 일어나는 상호작용이 다른 상호작용의 일부분에 해당된다면 전체 다이어그램에서 대안 컴포넌트들의 상호 작용 부분은 각각의 대안에 대해 구획으로 나누어 표현할 수 있다.

그림 5는 선택적 커넥터에 대해 아키텍처 모델의 정적인 뷰와 동적인 뷰상에서의 표현 형태를 나타낸 것이다.

선택적 커넥터인 경우는 선택적 컴포넌트의 경우와 마찬가지로 두 경우로 분류할 수 있다. 다른 시나리오상에서 사용되지 않는 독립 개별 기능으로서 기능의 확장 등에 의한 선택적인 경우와 시나리오상에서 선택적인 일부 시나리오가 존재하는 경우이다.

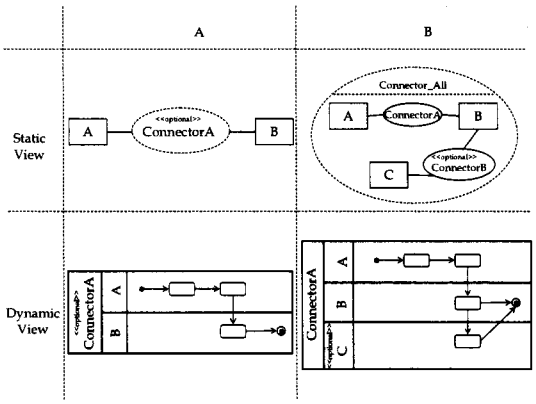

그림 5. 선택적 커넥터의 정적인 뷰와 동적인 뷰상에서의 표현 형태

첫번째(A)의 경우 정적인 뷰에서는 시나리오 자체가 선택적이므로 커넥터를 <<optional>> 스테레오타입을 정의하여 표현한다. 동적인 뷰에서는 시나리오를 표현하는 활동 다이어그램 전체에 대해 <<optional>> 스테레오타입으로 표현한다.

두번째(B)의 경우 일부 시나리오 즉 일부 커넥터가 선택적인 경우로서 정적인 뷰에서는 선택적인 일부 시나리오에 대해 전체 시나리오를 표현하는 Collaboration의 내부 구조에서 선택적인 시나리오에 해당되는 커넥터에 대해 <<optional>> 스테레오타입으로 표현한다. 동적인 뷰의 다이어그램에서는 전체 다이어그램의 선택적인 시나리오 부분에 대해 구획으로 나누어 표현하고 이를 <<optional>> 스테레오타입으로 표현한다.

그림 6은 대안 커넥터에 대해 아키텍처 모델의 정적인 뷰와 동적인 뷰상에서의 표현 형태를 나타낸 것이다.

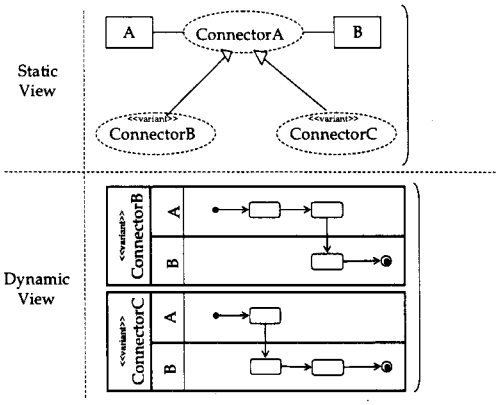

그림 6. 대안 커넥터의 정적인 뷰와 동적인 뷰상에서의 표현 형태

커넥터 단위에서의 대안인 경우는 상호작용 주체인 컴포넌트의 변경 없이 기능 수행 과정의 변경의 경우로 볼 수 있다. 정적인 뷰에서는 커넥터 단위의 대안임을 표현하기 위해 가장 많이 사용되는 시나리오에 해당되는 커넥터를 상위의 커넥터로 정의하고 이에 대해 대안커넥터를 표현한다. 동적인 뷰에서는 각 커넥터별로 표현한다.

4. 사례 적용

본 절에서는 본 방법을 의료 영역의 EMR 시스템의 아키텍처에 적용한 결과를 설명한다. EMR 시스템의 주요 기능으로는 진단 기록, 환자상태 기록, 간호기록 기능이 있다. 그 중에서 진단 기록 기능을 예제로 적용한다.

4.1. 컴포넌트 분류

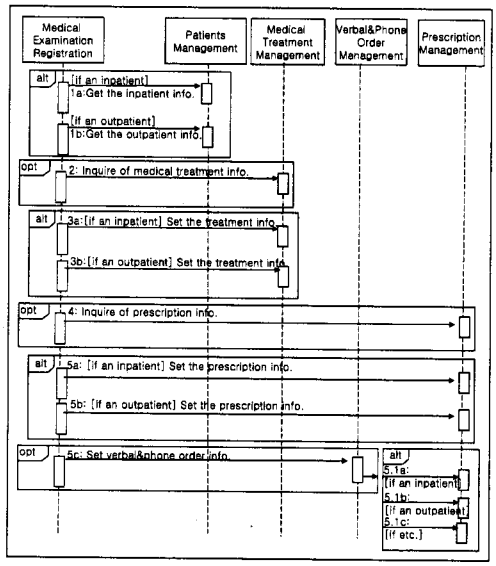

그림 7. 진단 기록 기능의 시나리오

그림 7은 사용자가 순차도 형태로 작성한 진단 기록 기능의 시나리오를 나타낸 것이다. 순차도에서 opt 연산자가 사용된 단계에 포함된 컴포넌트들을 분류하면 'Medical Treatment Management', 'Verbal&Phone Order Management'와 'Prescription Management' 컴포넌트가 해당된다. 'Medical Treatment Management'와 'Prescription Management' 컴포넌트는 진료 정보 조회 및 작성 기능과 처방 정보 조회 및 등록을 위한 컴포넌트들로서 진료 기록 조회 및 처방 정보 조회는 진단 기록 기능의 시나리오상에서만 선택적인 기능에 해당되므로 시나리오상에서 선택적인 기능 수행을 위한 컴포넌트로 분류한다. 그리고 'Verbal&Phone Order Management' 컴포넌트는 구두 처방기능만을 위한 컴포넌트로서 다른 시나리오에 포함되지 않는 독립적인 개별 기능에 해당된다. 따라서 독립 개별 기능 확장에 의한 선택 컴포넌트로 분류한다.

alt 연산자가 사용된 단계에 포함된 컴포넌트들을 분류하면 'Patients Management', 'Medical Treatment Management', 'Prescription Management' 컴포넌트가 해당된다. 이들 컴포넌트들이 다루는 정보들인 환자 정보, 진료 정보, 처방 정보 등은 각각의 대안에 대해 기본 정보는 동일하고 일부만 다른 정보들이므로 specialization 관계로 각 정보들을 구조화할 수 있다. 이러한 정보들을 다루는 컴포넌트는 각 특정 정보에 따라 인터페이스가 변경될 필요가 없으므로 각 대안 컴포넌트들이 인터페이스 변경 없이 구현만 변경되는 경우에 해당된다. 따라서 이 컴포넌트들을 인터페이스 변경 없이 구현만 변경되는 대안 컴포넌트로 분류한다.

4.2. 가변 요소 관계 형성

먼저 선택 컴포넌트에 의해 영향 받는 의존 관계를 형성하기 위해 선택 컴포넌트 중에서 다른 컴포넌트와 상호작용 관계를 가지는 컴포넌트를 분류한다. 본 예제에서 이에 해당되는 컴포넌트는 'Verbal&Phone Order Management' 컴포넌트다. 'Verbal&Phone Order Management'컴포넌트에 의해 사용되는 컴포넌트는 'Prescription Management' 컴포넌트로서 기존의 인터페이스인 외래환자 처방 인터페이스와 입원환자 처방 인터페이스를 사용할 수도 있고 기본 처방 정보와는 별개의 지시 정보 등이 구두처방에 포함될 수 있으므로 기타정보 처방을 위한 별개의 인터페이스가 'Prescription Management' 컴포넌트에 요구된다. 따라서 가변 요소 관계 형성 알고리즘을 적용하면 기타정보 처방을 위한 별개의 인터페이스와 'Verbal&Phone Order Management' 컴포넌트와의 의존관계를 형성한다.

다음으로 대안 컴포넌트에 의해 영향 받는 의존 관계를 형성하기 위해 대안 컴포넌트 중에서 다른 컴포넌트와 상호작용 관계를 가지는 컴포넌트를 분류한다. 본 예제에서 이에 해당되는 컴포넌트는 'Prescription Management' 컴포넌트다. 'Prescription Management' 컴포넌트의 인터페이스를 사용하는 컴포넌트는 'Verbal&Phone Order Management' 컴포넌트다. 이 컴포넌트의 제공 인터페이스는 'Prescription Management' 컴포넌트의 인터페이스를 요구하는 인터페이스므로 'Prescription Management' 컴포넌트와 의존 관계가 형성된다.

4.3. 가변 요소 표현

그림 8은 본 예제의 가변 요소가 표현된 아키텍처를 나타낸다. 그림 8에서 'Verbal&Phone Order Management' 컴포넌트의 경우 다른 시나리오에

포함되지 않는 독립 개별 기능에 의한 선택 컴포넌트로 그림 3(A)의 형태로 표현된다. 'Prescription Management'와 'Medical Treatment Management' 컴포넌트의 경우, 시나리오상에서 각각 'Get prescription info.'기능과 'Get medical treatment info.'기능이 선택적인 경우로 그림 3(B)의 형태로 표현된다. 'Prescription Management'의 'Set prescription info.', 'Medical Treatment Management'의 'Set medical treatment info.', 'Patients Management'의 'Get patients info.'의 경우, 인터페이스의 변경 없이 여러 컴포넌트가 다양한 구현을 가지는 경우에 해당되므로 그림 4(A)의 형태로 표현된다. 'The others' 컴포넌트의 경우 'Prescription Management'에서 다루는 기능 및 정보와 동일하지 않은 경우로 'Prescription Management'의 인터페이스와 다른 인터페이스를 통해 'Verbal&Phone Order Management' 컴포넌트와 상호작용하게 된다. 이러한 경우 그림 4(B)의 형태로 표현된다.

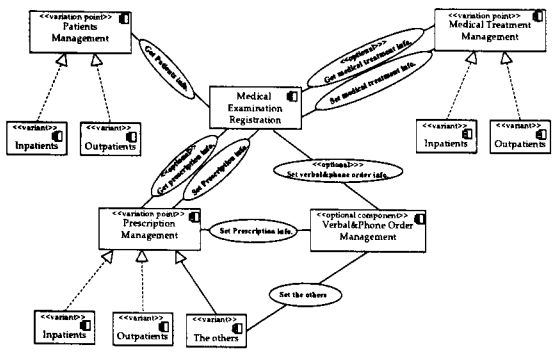

그림 8. 예제 아키텍처

5. Conclusions and Future Works

본 논문에서는 UML 2.0을 기반으로 가변요소를 추출하고 표현하는 방법을 제안하였다. 본 연구에서는 아키텍처 요소들간의 의존 관계를 고려하여 가변성을 다룰 수 있도록 하였고, 컴포넌트의 가변성뿐만 아니라 커넥터 측면의 가변성을 명시적으로 표현할 수 있도록 지원한다. 또한 아키텍처 요소의 특성에 따라 가변성을 분류하고 표현할 수 있도록 지원하여 제품 계열 아키텍처로부터 쉽게 제품 아키텍처를 만들 수 있도록 지원한다. 향후에는 더욱 실질적인 사례에 적용하여 본 연구를 평가하고 가변성 관리에 대한 연구의 보완이 필요하다.

References

[1] Bass, L., Clements, P., and Kazman, R., Software Architecture in Practice (Addison-Wesley, 1998).

[2] Clauss, M., Modeling variability with UML. GCSE 2001 - Young Researchers Workshop, September 2001.

[3] Gomma, H., Shin, M.E., Multiple-View Meta-Modeling of Software Product Lines. ICECCS 2002, pp. 238-246, 2002.

[4] OMG, UML 2.0 Superstructure. 3rd Revision, OMG document ad/03-04-01.

[5] Goulao, M., Abreu, F. B. e, Bridging the gap between Acme and UML 2.0 for CBD. Workshop at ESEC/FSE 2003, Sep. 2003.

[6] Roh, S, Kim, K., Jeon T., Architecture Modeling Language based on UML2.0. APSEC 2004, pp. Pages:663 - 669, Nov. 2004.

[7] Svahnberg, M. and Bosch, J., Issues Concerning variability in software product line, LNCS 1951, Jan 2000