모델 구조 가이드라인
Rational Software Modeler 및 Rational Software Architect 용 (2004 릴리즈)

백서

Bill Smith, Model Driven Development, IBM Rational Software

V1.0
2004 년 9 월 8

목차

1. 소개

대상 독자

이 문서는 RUP(Rational Unified Process)에 있는 가이드를 RSA 사용에 적용하는 데 관심이 있는 RSA(Rational Software Architect) 제품 사용자를 지원하기 위해 설계되었습니다. RSM(Rational Software Modeler) 사용자인 경우에도 문서가 유용하기는 하지만 일부 섹션은 RSM 이 아닌 RSA 에만 유용한 기능을 반영한다는 점을 알아 두십시오. 이 문서는 기본적으로 RSA 에서 새로운 모델 세트를 작성 중인 경우에 추천됩니다. RSA 는 처음 사용하지만 이전에 Rational Rose 또는 Rational XDE 를 사용한 경험이 있으며 이러한 제품으로부터 모델을 가져온 경우, 사용자가 가져온 문서를 재구성하기 위한 안내서로도 이 책이 유용하다는 것을 알게 될 것입니다.

이 문서는 사용자가 UML 에 대해 광범위한 지식이 있을 뿐 아니라 RSA 조작의 기본 개념 및 이론(특히, 모델링, 변환 및 비주얼 코드 편집)에 대해서도 기본적인 지식이 있다고 간주합니다.

목적

RUP 는 시스템의 솔루션 및 문제점 도메인에 대해 제대로 정의된 Perspective 를 표시하는 모델 세트에 대해 설명합니다. 이 모델 구조 세트의 유틸리티는 여러 실제 프로젝트에서 입증되었으며 공식 프로세스(예: RUP)를 따르는지 여부에 상관없이 이러한 모델 구조는 고려할 가치가 있습니다. 이 문서에서는 RSA 를 사용하여 RUP 모델 결과물을 구현하는 방법에 대해 설명합니다.

제목이 암시하는 바와 같이, 이 문서에서 설명된 프로젝트 및 모델 구조는 명령이 아니라 가이드라인입니다. RSA 에서 특정 RUP 결과물을 모델화할지를 결정하는 것은 사용자 자신의 개발 프로세스에서 고려할 사항이며 종종 프로젝트에 특정한 결정 사항이기도 합니다. 한 가지 명심할 점은

RUP 가 엄격한 프로세스 규칙 세트가 아니라는 것입니다. RUP 는 공식적인 것에서부터 사소한 것까지 범위를 지정할 수 있는 프로세스 정의를 공식화하는 프로세스 *프레임워크*입니다.

UML 을 사용하는 방법 또한 매우 공식적에서 매우 비공식적일 수 있습니다. UML 모델을 건축 중에 엄격하게 따라야 하는 공식 건축 도면처럼 다루도록 선택하거나, 설계의 광범위한 개요를 제안하기는 하지만 프로젝트가 구현 단계로 이동하면 버려도 되는 것으로 간주되는 스케치처럼 다루도록 선택할 수도 있습니다. RSA 는 모델링 범위 및 프로세스의 양 끝에서 지원할 수 있습니다. 이러한 관점에서 이 문서에서 가이드라인을 제공하는 것은 사용자의 사고를 구속하기 위함이 아니라 RSA 기능을 사용하는 방법을 이해하도록 도와 사용자에게 가장 적합한 프로세스를 제공하기 위한 것입니다 .

RSA 는 모델을 단지 청사진뿐 아니라 구현을 자동으로 생성할 수 있는 솔루션의 스펙으로도 사용할 수 있음을 참고하십시오. 이는 RSA 모델 대 모델 및 모델 대 코드 변환으로 달성됩니다. MDD(Model-Driven Development)를 수행하기 위해 RSA 를 사용하면 모델 구조와 관련된 특수한 문제점이 발생합니다. RSA 모델과 변환을 사용하여 MDD(Model-Driven Development)를 수행할 경우 Developers Works 에서 사용 가능한 여러 RSA MDD 특정 자원도 참조해야 합니다.

범위

이 문서에서는 RSA 에서 RUP 모델 결과물을 표시하는 방법을 설명하고 이러한 결과물의 내부 조직 구조에 대한 가이드라인을 제공합니다. 다음은 시도하지 *않습니다.*

- RUP 모델 결과물의 개념적 기반을 다시 언급하거나 이에 대한 확장 설명을 제공합니다.
- 연관된 RUP 결과물의 의미론적 또는 도표화된 컨텐츠를 지정하는 프로세스 또는 기술에 대해 설명합니다.

RUP 결과물의 컨텐츠를 정의, 개발 및 모델화하는 방법에 대한 툴 중립적 정보는 RUP 를 참조하십시오.

RSA 모델의 컨텐츠 개발에 필요한 툴 특정 기술에 대한 정보는 다음을 참조하십시오.

- 제품 문서(자습서, 샘플, 온라인 도움말)
- 이 백서가 들어 있는 RSA 특정 RUP 구성의 툴 강좌
- Developer Works 의 RSA 관련 자원

인쇄상의 규정

IBM Rational Rose 또는 XDE 에서 이주 중인 RSA 사용자의 관심사항에 대한 토론은 음영이 있는

> *XDE/Rose*
> 이전 XDE 또는 Rose 사용자의 관심사항에 대한 토론

백그라운드의 테두리가 있는 텍스트 상자에 사이드바 방식으로 제공됩니다.

문서의 구성

다음에 오는 기본 개념 및 용어 섹션에서는 사용하는 어휘를 설정하고 RSA 제품에서 모델을 구현하는 방법에 대한 몇 가지 일반 정보를 제공합니다.

그 다음, RUP 모델을 RSA 모델에 맵핑 섹션에서는 RSA 가 RUP 를 사용하여 정의된 모델 유형을 지원하는 방법에 대해 설명합니다.

그 다음에는 RSA 에 있는 여러 유형의 모델을 구성하기 위한 가이드를 제공하는 여러 섹션이 있습니다. 이러한 섹션의 일부에서는 프로세스, 모델링 접근법 및 구조적 제어 관점에서 사용자가 선호하는 정도에 따라 모델 유형을 사용하는 여러 가지 방법에 대해 설명합니다.

마지막으로 모델을 여러 모델링 파일로 분해하는 것과 연관된 여러 가지 문제를 다루는 토론이 있습니다. 여기서는 파일 경합 및 충돌 병합을 최소화하기 위해 팀원 간에 모델을 공유할 수 있게 하고 규모를 관리하는 전략에 대해 다룹니다.

2. 기본 개념 및 용어

모델

RUP 에서 모델은 특정 Perspective 에서 솔루션 또는 문제점 도메인에 대한 완전한 스펙으로 정의됩니다. 문제점 도메인 또는 시스템은 도메인 또는 시스템의 여러 Perspective 를 표시하는 다수의 모델에 의해 지정될 수 있습니다. RUP 는 다음과 같은 특정 모델 세트를 제안합니다.

- 비즈니스 유스 케이스 모델
- 비즈니스 분석 모델
- 유스 케이스 모델
- 분석 모델(설계 모델에 포함될 수 있음)
- 설계 모델
- 구현 모델
- 전개 모델
- 데이터 모델

RUP 자체는 툴이 아닙니다. RUP 와 관련될 경우, 모델은 냅킨 또는 화이트보드에 그린 그림, 모델링 툴의 일부 또는 마음속의 이미지일 수 있습니다. 따라서 RUP Perspective 에서 모델은 논리적 개념입니다. RSA 관점에서 모델을 논리적 관점에서 설명할 수 있지만 실제적 관점에서도 설명할 수 있습니다.

다음과 같은 두 어플리케이션에 대해 작업하는 팀이 있다고 가정하십시오. Timesheet 관리 어플리케이션에 대해 작업하는 세 명의 분석가로 이루어진 한 팀과 Call Center 어플리케이션에 대해 작업하는 다섯 명의 분석가로 이루어진 두 번째 팀이 있습니다. 두 팀 모두 현재 요구사항을 캡처하기 위해 작업 중이며 유스 케이스 모델링에 RSA 를 사용합니다. RUP 관점에서 한 팀은 “timesheet 어플리케이션에 대한 유스 케이스 모델”을 빌드 중이고 다른 팀은 “call center 어플리케이션에 대한 유스 케이스 모델”을 빌드 중이라고 말할 수 있습니다. 그러나 팀이 RSA 를 사용 중인 경우 모델에 특정 실제 표현이 있음을 인식하는 것이 중요합니다. 이것이 다음 섹션의 주제입니다.

모델링 파일

RSA 에서 모델은 파일로서 존속됩니다. (Eclipse 용어에서 파일은 '자원'으로 간주됩니다. [1] 따라서 이 문서 또는 기타 소스에서 '모델링 자원' 이라는 용어가 나타나면 이는 '모델링 파일'과 동일한 의미를 갖습니다.) 가장 광범위한 의미에서 RSA 는 두 가지 유형의 모델링 파일을 지원합니다.

- "사전 구현" 모델링 파일. 이 파일에는 *직접적으로* 구현 결과물을 반영하지 않는 개념적 UML 컨텐츠가 들어 있습니다. 파일에는 모델의 UML 의미론과 의미론적 요소를 설명하는 UML 다이어그램 모두가 들어 있습니다.

- Java 소스 파일 또는 웹 페이지와 같은 보통 구현 결과물 및 Eclipse 프로젝트에 있는 구현 모델링 파일(Eclipse 자원). 이러한 경우에는 프로젝트가 구현 모델링 파일의 컨텐츠 범위를 표시한다고 생각할 수 있습니다. 모델 의미론은 구현 결과물 자체에 있습니다. 각 다이어그램은 프로젝트 내의 자체 실제 파일에 있습니다. 이러한 다이어그램은 UML 표기법을 사용할 수도 있으나 기타 표기법(예: 데이터 비주얼화에 사용되는 Information Engineering 또는 IDEF1X, 웹 티어 설계에 사용되는 Rational 자원 표기법)을 사용할 수도 있습니다. 3GL 코드 결과물의 설명을 지원하는 UML 다이어그램 유형에는 클래스 다이어그램과 순서 다이어그램(Java 전용)이 포함됩니다.

이 문서는 "사전 구현" 모델의 내부 구조를 조직하는 방법에 역점을 두며 **문서 내에서 모델링 파일이라는 용어는 "사전 구현" 모델 컨텐츠를 포함하는 파일에 사용하도록 정해져 있습니다**. 구현 프로젝트의 컨텐츠를 조직하는 데 필요한 가이드는 기타 소스(예: Rational Software Architect, Rational Application Developer 및 Rational Web Developer 에 대한 온라인 도움말)에서 찾을 수 있습니다.

"사전 구현" 모델링 파일이 한 모델에 대한 모든 정보를 포함하는 것은 아닙니다. 사실, 모델링 파일에 모델의 서브세트만이 포함되는 것은 흔한 경우입니다. 예를 들어, 위에 주어진 Timesheet 어플리케이션의 유스 케이스 모델에 대해 작업하는 세 명으로 구성된 팀의 예에서 팀은 유스 케이스 모델을 실제로 세 개의 모델링 파일로 파티션하도록 선택하여 각 팀원이 동일한 파일을 사용하기 위해 경쟁하지 않고 유스 케이스의 다른 서브세트에서 작업할 수 있게 합니다. 이 문서의 마지막 섹션에서는 모델 파티션 및 모델링 파일의 관리와 연관된 문제에 대해 설명합니다.

모델 유형

RUP 에서 모델에는 유스 케이스 모델, 분석 모델 또는 데이터 모델과 같은 특정 유형이 있습니다. RSA 에서는 모델링 파일이 유형(파일이 포함하는 모델(또는 모델 서브세트) 유형)을 갖고 있다고 생각할 수 있습니다. 모델링 파일의 유형은 다음 두 가지 방법 중 하나로 설정할 수 있습니다.

- "공백" 모델링 파일(아래 참조)에서 시작한 다음 단순히 이름 지정 방법 및 파일에 넣을 컨텐츠(적용한 UML 프로파일 포함) 유형만을 선택하여 유형을 설정하십시오.

- 특정 모델 유형을 표시하는 사전 정의된 "템플리트 모델"을 기반으로 작성하십시오. 모델링 파일의 "유형"은 파일의 컨텐츠에 관한 규정일 뿐임을 주의하십시오. 예를 들어, 툴은 유스 케이스 모델을 포함하는 파일에 유스 케이스를 구현하는 클래스가 포함되는 것을 막을 수는 없습니다. 그러나 이러한 가이드라인은 사용자가 모델 파일을 입력 중인 것으로 간주하도록 권장합니다.

작업공간, 프로젝트 및 프로젝트 유형

Eclipse, WebSphere Studio 제품 또는 Rational Application Developer 에 익숙한 사용자는 이미 파일이 프로젝트에 있으며, 해당 프로젝트는 여러 유형일 수 있고 (가상적으로) 그룹화되어 작업공간에서 관리될 수 있음을 알고 있습니다. RSA 모델링 파일은 기타 다른 파일과 마찬가지로 프로젝트에 있습니다.

[1] Eclipse 에서 자원은 파일이지만 Eclipse 환경 내의 추가 특성 및 동작도 있습니다. 여기에 설명된 모델링 파일은 Eclipse 에서 '자원'으로 다룹니다.

이 토론을 위해 Rational Software Architect, Rational Application Developer 및 Rational Software Modeler 에서 사용 가능한 모든 프로젝트 유형에 대해 자세히 설명할 필요는 없습니다. 광범위한 관점에서 프로젝트의 다음 두 카테고리에 관심이 있습니다.

- UML 프로젝트
- **구현 프로젝트**(엔터프라이즈 프로젝트, EJB 프로젝트, 웹 프로젝트 및 C++ 프로젝트와 같은 특수 유형 포함).

앞서 RSA 가 다음과 같은 두 종류의 모델링 파일을 지원한다고 설명했습니다.

- 구현(예: 요구사항, 분석 및 설계) 이상의 추상 레벨에 있는 모델링에 사용하는 UML 모델(의미론적 요소와 함께 해당 요소를 설명하는 다이어그램)을 포함하는 파일.
- Java 코드와 같은 구현 결과물 또는 웹 페이지와 함께 구현 결과물을 설명하는 다이어그램을 포함하는 다이어그램 파일(선택사항)이 들어 있는 프로젝트.

프로젝트에 모델을 할당하는 규칙은 다음과 같이 간단합니다. a) "사전 구현" 모델링 파일을 UML 프로젝트에 두십시오. b) 구현 모델은 실제로 다음과 같으므로 자연히 처리됩니다.

[구현 모델] = [구현 프로젝트]

이 규칙에는 몇 가지 예외사항이 있습니다. 다음 UML 모델링 파일이 언어 특정 구현 프로젝트에 배치될 후보입니다.

- 설계 '스케치 모델'(후반 섹션에서 설명)
- 프로젝트 내의 코드에 대해 실행될 테스트에 대해 기술하는 순서 다이어그램이 있는 모델

> *XDE/Rose*
> Rose 및 XDE 조작 이론은 사용자가 코드와 동등한 추상 레벨에 도달할 때까지 반복적으로 설계 모델을 정제한 다음 코드 모델 동기화 기술을 사용하여 코드 자체와 해당 모델의 의미론을 동일하게 유지하는 것을 포함합니다. 예를 들어, XDE 에서 구현 모델은 프로젝트의 코드 및 다이어그램으로도 존재하지만 구현 결과물에 독립적으로 존속되며 실제로 중복된 의미론 사본을 표시하는 '코드 모델' 파일로도 존재합니다.
>
> 조작의 RSA 이론은 코드보다 상위인 추상 레벨(엔터프라이즈 IT 설계 모델과 같은 설계 모델)에 있는 플랫폼 중립 모델과 해당 모델에서 코드를 생성할 때 변환을 사용하도록 권장합니다. 그런 다음, 추상 코드 레벨에서 RSA 를 사용하면 간단히 UML 다이어그램을 그릴 수 있으며 별도의 존속하는 의미론 모델을 사용하는 접근법이 필요하지 않습니다.
>
> RSA 로는 추상 코드 레벨에서 UML 모델을 정의하여 해당 모델에서 코드를 생성하는 것을 *막을 수 없음을* 참고하십시오. 사실, 이러한 유형의 사용법이 예상되기는 합니다. 그러나 RSA 는 이러한 모델이 코드와 동기화를 유지할 수 있는 기술을 제공하지 않습니다. 이 유형의 사용법은 일반적으로 코드가 생성되고 나면 설계 모델이 필요하지 않다고 생각되는 RUP 의 매우 '간단한' 변환에 해당합니다.

검토의 개념

다음 삽화에 앞에서 설명한 내용이 요약되어 있습니다. 삽화에 이전에 설명한 시나리오가 반영되어

있습니다. 한 팀은 Call Center 어플리케이션에 대해 작업하고 다른 팀은 Timesheet 관리 어플리케이션에 대해 작업하고 있습니다.

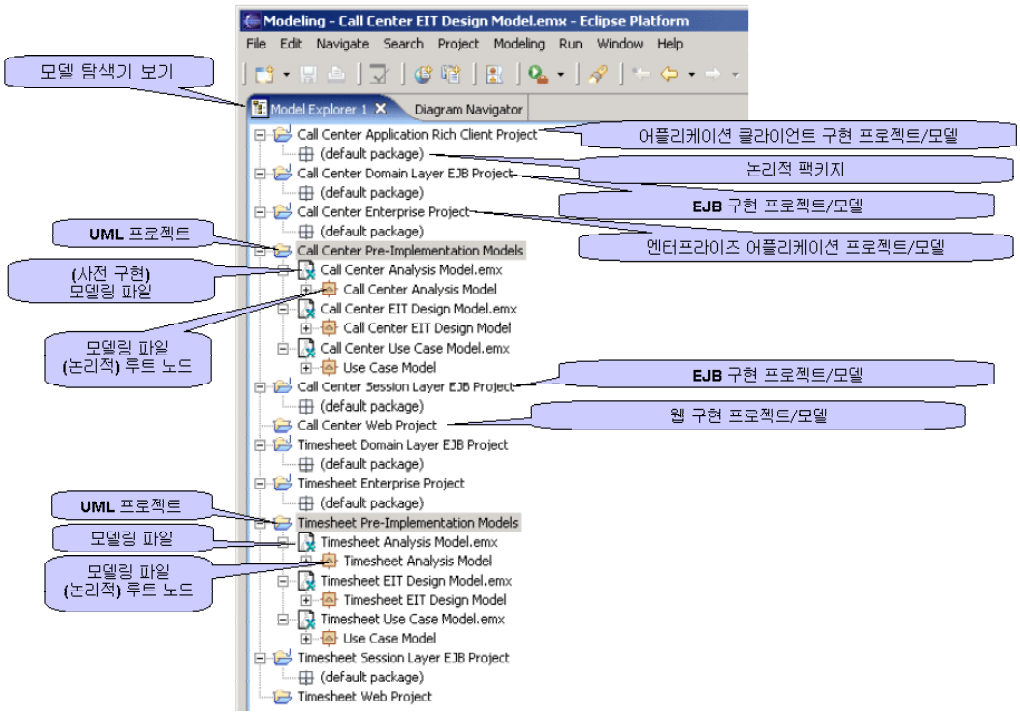


그림 2-1

RSA 는 모델의 결합된 실제 및 논리 보기를 제공하는 모델 탐색기 보기를 제공합니다. 모델 탐색기를 보면, 최상위 레벨 노드로 설명된 작업공간에 프로젝트가 표시되고 각 프로젝트 내에 해당 프로젝트에 속한 자원이 표시됩니다. **그림 2-1**의 모델 탐색기에 시나리오의 두 어플리케이션에 대응하는 프로젝트의 콜렉션이 나와 있습니다. UML 프로젝트가 사전 구현 모델에 사용되었습니다. 솔루션에 적합한 유형의 기타 프로젝트 콜렉션(예: 엔터프라이즈 어플리케이션 프로젝트, 웹 프로젝트 등)이 구현 모델에 사용되었습니다.

> *XDE/Rose*
> RSA 모델 탐색기와 대조적으로 Rose 및 XDE 의 모델 탐색기는 모델의 논리 보기만을 제공합니다. RSA 모델 탐색기에서 제공하는 자원의 보기는 Eclipse 탐색기 보기가 제공하는 '순수' 실제 보기가 아니라는 점을 참고하십시오. 일부 실제 자원은 모델 탐색기에 표시되나 이들은 대부분 자원의 *논리* 보기를 표시하는 아이콘으로 표시됩니다.

그림 2-2 에 내부적으로 Timesheet 유스 케이스 모델을 기능이 응집된 문제점 도메인의 서브세트를 표시하는 패키지로 조직할 수 있는 방법이 나와 있습니다.

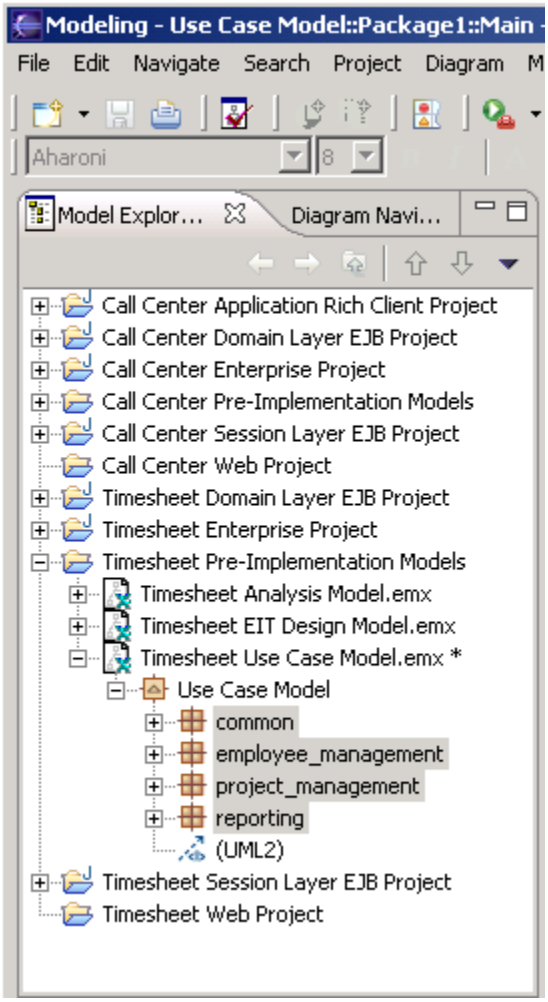


그림 2-2

그림 2-1 및 **2-2** 에서 각 사전 구현 모델은 단일 모델링 파일에 있습니다. 대안으로, 임의의 사전 구현 모델을 복수의 모델링 파일로 다시 세분할 수도 있습니다. 예를 들어, Timesheet 유스 케이스 모델을 각각 설명된 문제점 도메인의 서브세트 중 하나에 해당하는 네 개의 모델링 파일(“common”, “employee_management”, “project_management” 및 “reporting”)로 다시 세분할 수 있습니다. 이러한 경우, 각 모델링 파일의 루트 노드는 전체 유스 케이스 모델을 구성하는 모든 모델링 파일 전체에서 일치하는 네임스페이스를 유지하도록 이름이 지정됩니다. 예를 들어, 네 개의 모델링 파일의 루트 노드는 “timesheet.requirements.common”, “timesheet.requirements.employee_management”, “timesheet.requirements.project_management” 및 “timesheet.requirements.reporting”일 수 있습니다.

3. RUP 모델을 RSA 모델에 맵핑

다음 표에 가장 일반적으로 사용되는 RUP 모델을 RSA 모델 유형으로 맵핑하는 방법이 나와 있습니다. 일반적으로 맵핑은 간단하지만 RSA 와 함께 RUP 를 실행할 때 가이드로서 이 문서를 사용하는 것이 관건입니다. 표에서 언급된 RSA 모델 유형이 테이블 바로 다음에 설명됩니다. 여러 모델 유형의 내부 조직 및 해당 모델 유형을 보유할 프로젝트 종류에 대한 가이드라인은 나중 섹션에 제공됩니다. 이러한 나중 설명은 여기에 나열된 RSA 모델 유형과 관련하여 제공됩니다.

RUP 모델	RSA 모델 유형
유스 케이스 모델	유스 케이스 모델
분석 모델	분석 모델 (대안: 설계 모델의 «analysis» 패키지)
설계 모델	N 계층 비즈니스 어플리케이션의 경우: 엔터프라이즈 IT 설계 모델 기타 유형의 어플리케이션의 경우: 설계 모델로 사용하는 공백 모델 설계 '스케치'의 경우: 설계 '스케치' 모델에 사용하는 공백 모델 선택적 보충사항: 구현 개요 모델로 사용하는 공백 모델
구현 모델	구현 결과물 및 다이어그램 파일을 포함하는 구현 프로젝트
전개 모델	전개 모델에 사용하는 공백 모델

RSA 모델 유형

공백 모델

RSA 는 "공백 모델"을 작성하는 옵션을 제공합니다(파일→새로 작성→UML 모델→공백 모델). "공백 모델"은 모델 템플리트를 기반으로 하지 않는 모델링 파일입니다. 특수 프로파일이 적용되지 않았으며 단일 "기본"(자유 양식) 다이어그램을 제외한 기본 컨텐츠가 없습니다. **공백 모델링 파일을 임의의 모델 유형 에 대한 시작점으로 사용할 수 있습니다**. 이름 지정 방법, 정의할 컨텐츠 및 적용할 프로파일을 선택함으로써 유스 케이스 모델, 분석 모델, 설계 모델, 전개 모델 또는 기타 유형의 RUP 모델을 빌드할 때 공백 모델링 파일을 사용할 수도 있습니다.

유스 케이스 모델

RSA 는 모델 템플리트를 기반으로 "유스 케이스" 모델 파일을 작성하는 옵션을 제공합니다. 템플리트는 *그림 3-1*에 설명된 기본 컨텐츠를 제공합니다. ("빌딩 블록" 컨텐츠와 검색 문자열을 사용하는 방법에 대한 설명은 이 문서에서 다루는 분야가 아닙니다. 템플리트에는 주로 자체로도 설명이 가능한 지시사항이 포함됩니다.)

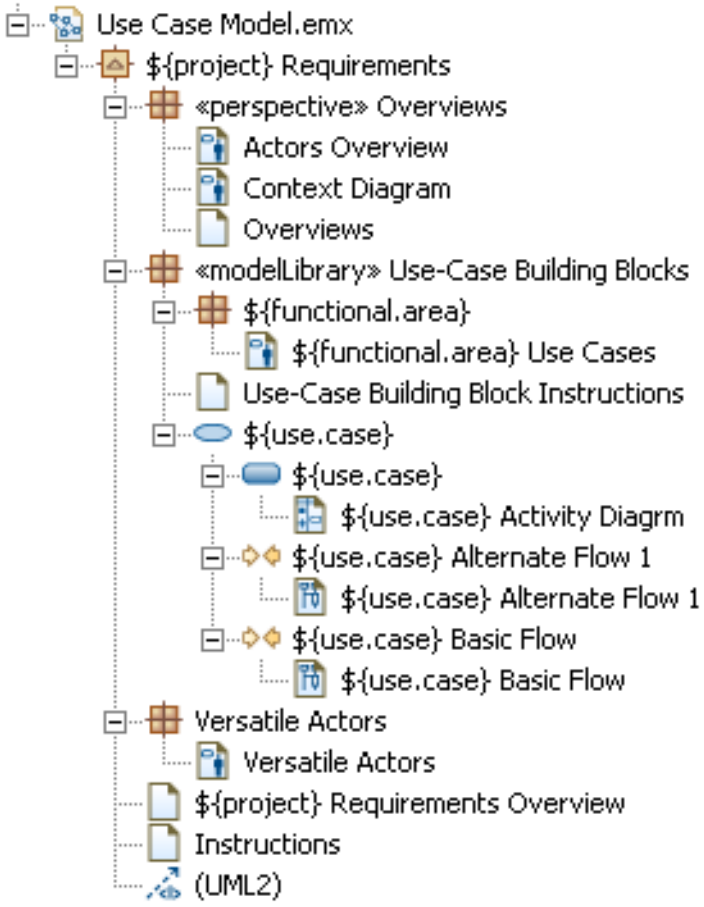


그림 3-1

분석 모델

RSA 는 모델 템플리트를 기반으로 "분석 모델" 파일을 작성하는 옵션을 제공합니다. 템플리트는 *그림 3-2*에 설명된 기본 컨텐츠를 제공합니다. 또한 "분석" 프로파일이 이 템플리트에서 작성된 모델 파일에 적용됩니다.

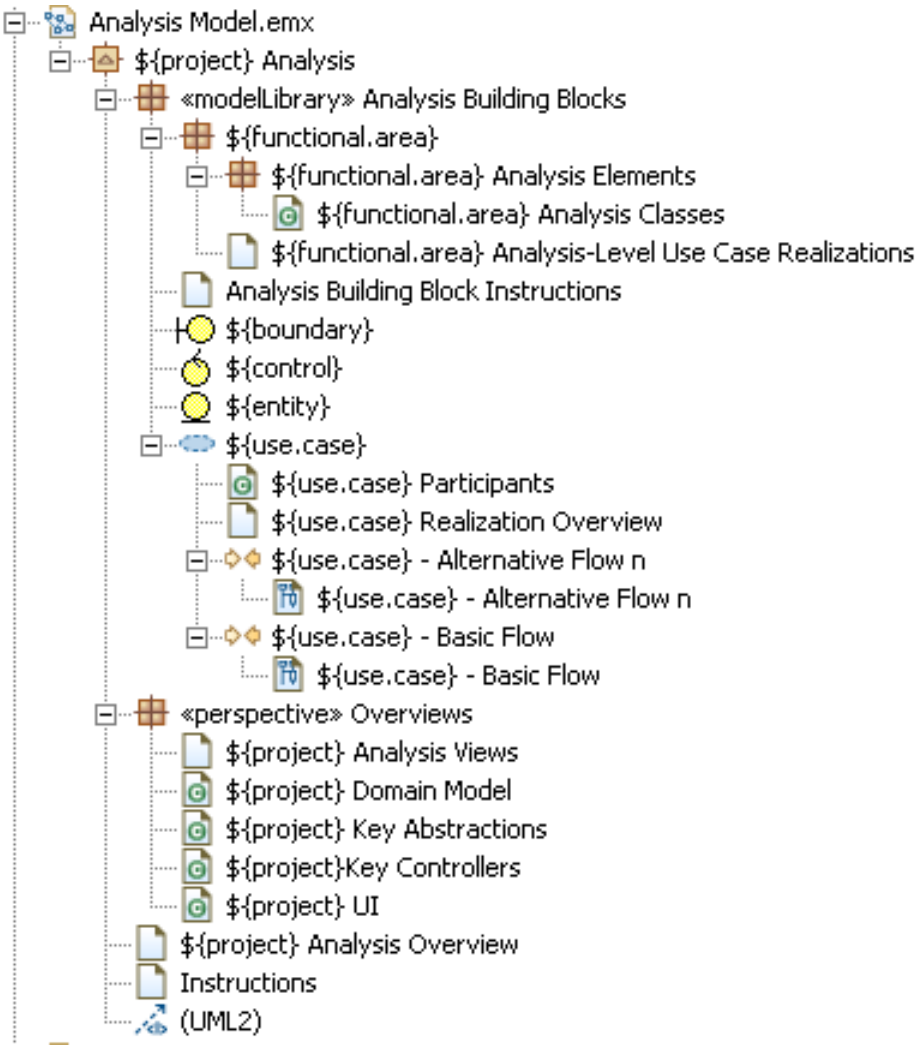


그림 3-2

엔터프라이즈 IT 설계 모델

RSA 는 모델 템플리트를 기반으로 EITDM(“Enterprise IT Design Model”)을 작성하는 옵션을 제공합니다. 템플리트는 **그림 3-3**에 설명된 기본 컨텐츠를 제공합니다. 또한 “EJB 변환” 프로파일[2] 이 이 템플리트에서 작성한 모델 파일에 적용됩니다. 이는 비즈니스 어플리케이션을 대상으로 하고 RSA 코드 생성 변형을 사용하여 해당 어플리케이션의 작성을 지원할 때 설계(선택적으로 분석)에 사용하기에 적합한 템플리트입니다.

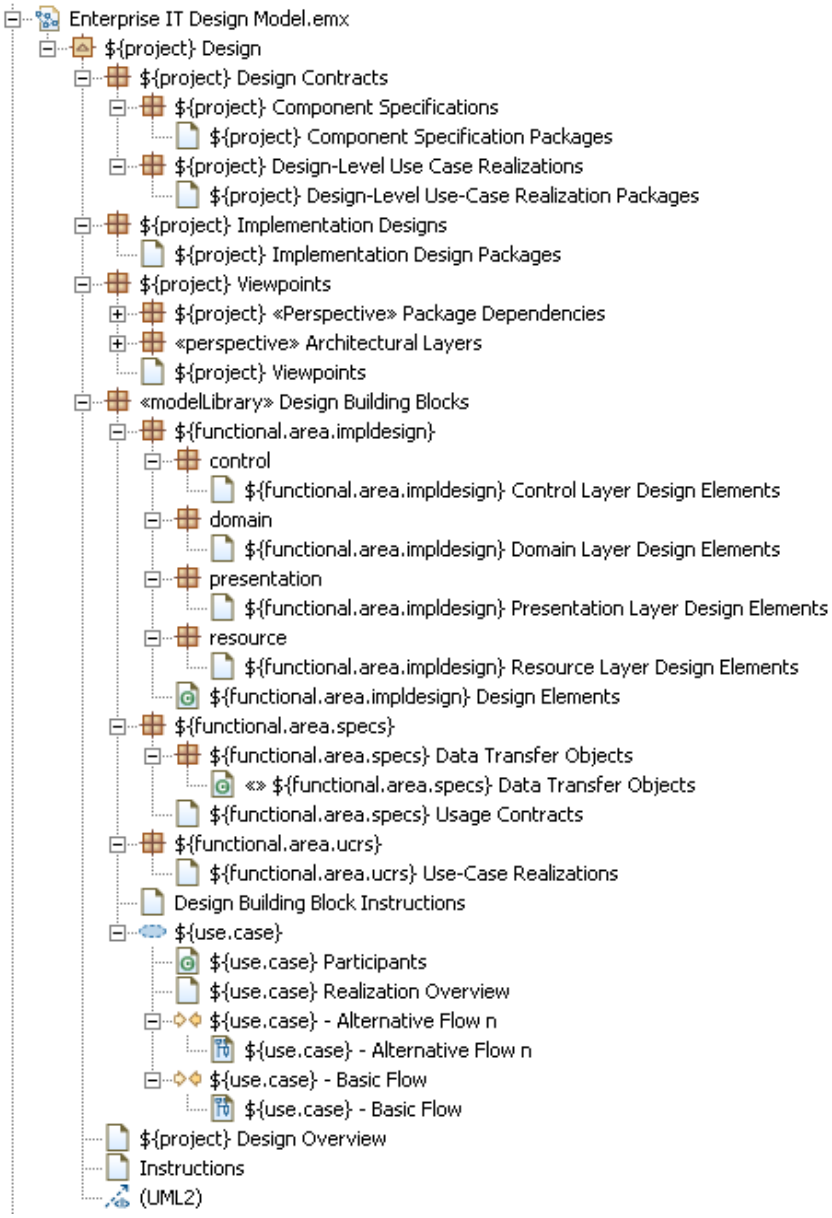


그림 3-3

[2] EIT 설계 모델 템플리트의 일부로 제공되는 변환 가능 프로파일 세트는 제품의 갱신사항이 릴리스됨에 따라 전개될 가능성이 있습니다.

구현 개요 모델

설계 모델의 일부로서, 구현을 조직할 방법에 대한 상위 레벨 보기를 캡처하는 "구현 개요 모델" 을 정의하는 것이 유용하다는 것도 알게 됩니다. 코드 및 관련 파일(메타데이터, 전개 설명자 등)이 있을 것으로 예상하는 실제 RSA 프로젝트 및 폴더/패키지를 표시할 수 있도록 구현 개요 모델은 설계 단계의 초기(코드를 생성 또는 작성하기 전)에 사용됩니다. 이 모델을 사용하여 이러한 프로젝트 및 패키지 간의 예상 종속성을 표시할 수도 있습니다. 종속성은 시스템 빌드 요구사항을 식별하는 데 유용하다는 것을 입증할 수 있습니다. 구현 개요 모델은 솔루션 구조의 비공식적 개념 다이어그램을 보존할 장소가 될 수도 있습니다.

구현 모델

앞서 언급한 바와 같이, RSA 에서 구현 모델은 구현 결과물과 해당 결과물을 설명하는 다이어그램(선택적)을 포함하는 프로젝트로 구성되어 있습니다.[3]

" 스케치" 모델

"기본 개념 및 용어" 섹션에서 언급한 바와 같이, 설계 모델을 시스템 수명 동안 유지보수하고 구조적 제어를 지원/시행하는 데 사용되는 공식적 설계 도면과 같이 처리하도록 선택할 수 있습니다. 아니면 설계를 제시하는 역할을 하고 이를 명백히 하며 통신하는 데 도움은 되지만 실제 구현이 원래 설계에서 분기하기 시작하면 버려도 되는 스케치와 같이 다룰 수 있습니다. RSA 는 두 접근법 모두를 지원합니다. 해당 기능은 일반적으로 한 접근법 또는 다른 접근법을 대상으로 하지는 않지만 사용자가 선택한 설계 모델 사용 방법은 사용할 RSA 기능 및 해당 기능의 사용 방법을 결정하는데 확실히 도움이 됩니다. 이 문서에 표시된 가이드라인의 컨텍스트에서 구분을 해야 한다면 "스케치 모델"이라는 용어는 모델을 '버리기 쉬운' 방식으로 사용한다는 것을 나타낼 때 사용됩니다.

---

[3] 이러한 다이어그램을 작성하려면 모델 작성 시 파일→새로 작성→UML 모델을 사용하지 말고 파일→새로 작성→클래스 다이어그램을 사용하여 UML(또는 기타) 표기법으로 코드의 '보기'를 구성할 수 있는 다이어그램을 작성하십시오. 각 개별 다이어그램은 확장자가 .dnx 인 별도의 파일로 존재하며 코드 파일과 동일한 정도로 제어되는 버전일 수 있습니다. 이러한 다이어그램에는 어떠한 의미론적 정보가 아닌 표기법만을 포함합니다. 모든 관련 의미론적 정보는 코드 자체에 있습니다. 이러한 다이어그램 중 하나에서 클래스 이름 또는 조작 서명과 같은 사항을 변경하는 경우, 실제로 기반 코드 자체를 변경합니다. 이처럼 텍스트 편집기를 사용하여 코드를 변경할 경우, 변경된 코드가 들어 있는 다이어그램은 자동으로 갱신됩니다.

4. 모델의 내부 구조를 조직하기 위한 기술 및 일반 가이드라인

UML 모델의 컨텐츠를 조직하는 기본 툴은 패키지입니다. UML 패키지는 두 가지 기본 용도로 사용됩니다.

- 모델 정보를 레이블링, 조직 및 파티션
  - 문제점 또는 솔루션 도메인의 특정 주제 문제에 대응하는 요소 그룹화
  - 모델 정보의 다른 유형(예: 인터페이스, 구현, 다이어그램 등)을 분리
  - 기타 요소에 대한 종속성을 정의하고 제어하기 위해 요소 그룹화
  - 동일한 모델의 대체 보기를 제공하는 다이어그램 그룹화

- 네임스페이스 설정
  - 모델 요소용
  - 모델 요소에서 생성된 구현 결과물용(모델 및 구현 언어 네임스페이스 간 맵핑을 포함할 수도 있음)
  - 재사용 단위용

일반적으로 RUP 는 여러 모델 유형에 특정 패키징 전략을 제안합니다. 이러한 전략은 이 문서의 모델 유형 특정 섹션에 반영되어 있습니다. RSA 는 여기에 설명된 일부 추가 조직 툴도 도입합니다.

«perspective» 패키지를 사용하여 관점 표시

두 가지 이상의 방법으로 조직된 요소를 표시해야 할 경우, 대체 조직 설계를 설명하는 다이어그램을 사용하여 추가 패키지를 작성할 수 있습니다. 이 동일한 기술은 모델의 패키징 설계를 초월하여 모델 컨텐츠의 특정 보기를 표시해야 하는 모든 경우에 유용할 수 있습니다. RSA 는 «perspective» 패키지 스테레오타입을 UML '기본 프로파일'의 일부로 제공하여 이 기술을 지원합니다. «perspective» 패키지가 일반적으로 시스템 엔지니어링 또는 IEEE 1417 "관점"의 RUP 에 상응한다고 생각할 수 있습니다.

«perspective» 패키지 내에 의미론적 요소(클래스, 패키지, 연관 등)를 배치하지 마십시오. 대체 조직 문제 또는 어플리케이션 관점을 기반으로 하는 보기를 설명하는 다이어그램만 배치하십시오. 패키지에 «perspective» 스테레오타입을 적용함으로써 여러 가지 작업을 수행할 수 있습니다. 특정 관점을 표시함으로써 해당 패키지를 가시적으로 식별합니다. 의미론적 요소가 «perspective» 패키지에 놓여 있으면 사용자에게 경고를 표시하는 모델 유효성 검증 규칙도 지원합니다. 이는 RSA 변환이 우회해야 하는 패키지의 지정자 역할도 합니다.

주제 다이어그램을 사용하여 특정 관련 사항에 대한 자체 갱신 표현 작성

설명할 요소를 수동으로 배치하는 '일반' 다이어그램과는 대조적으로 주제 다이어그램의 컨텐츠는 기존 모델 컨텐츠에 대해 실행되는 조회에 의해 판별됩니다. 주제 다이어그램을 작성하려면 '주제' 모델 요소를 선택한 다음 주제 요소에 대한 관계 유형을 기반으로 다이어그램에 표시할 기타 요소를 정의하십시오. 모델의 의미론적 컨텐츠가 변경되면 '주제' 다이어그램도 이에 맞게 조정됩니다.

찾아보기 다이어그램을 통해 모델 검사

찾아보기 다이어그램은 *명확히* 모델 조직용 툴이 아닙니다. 툴의 목적은 수동으로 다이어그램을 작성하지 않고도 모델 컨텐츠의 발견 및 이해를 용이하게 하는 것입니다. 모델 조직의 컨텍스트에서 찾아보기 다이어그램을 사용하면 존속되는 다이어그램을 작성해야 하는 필요성을 줄일 수 있으므로 이를 인지하는 것이 유용합니다. 이에 따라 모델의 크기 및 복잡도가 줄어들기 때문에 조직하기도 쉬워집니다.

찾아보기 다이어그램은 주제 다이어그램과 유사하지만 이는 지속적이지 않으며 언제나 작동 중에 생성된다는 차이점이 있습니다. 찾아보기 다이어그램을 작성하려면 모델 요소를 선택하고(다이어그램 또는 모델 탐색기에서) 컨텍스트 메뉴를 사용하여 "찾아보기 다이어그램을 탐색"하십시오. 이로써, 선택한 요소를 초점 주변에 방사형 레이아웃으로 표시되는 관련 요소와 함께 '초점'으로 간주되는 다이어그램을 작성합니다. 그런 다음, 해당 찾아보기 다이어그램에서 연관된 요소 중 하나를 선택하여 다른 찾아보기 다이어그램의 초점으로 만드십시오. 원하는 만큼 이러한 방식으로 계속할 수 있습니다.

다이어그램 내 탐색

RSA 에는 다이어그램 내 탐색을 위한 두 가지 메커니즘이 있습니다.

- 모델 탐색기에서 일부 기타 '호스트' 다이어그램으로 다이어그램 노드를 끌어 놓을 수 있습니다. 그런 다음, 호스트 다이어그램에 결과로 생긴 아이콘을 두 번 눌러 참조된 다이어그램을 열 수 있습니다.

- 모델에서 새 UML 패키지를 작성할 때마다 "기본" 다이어그램(자유 양식 다이어그램)이 자동으로 작성됩니다. 기본적으로, 이 "기본" 다이어그램이 패키지의 "기본" 다이어그램으로서 작성됩니다. 다이어그램을 "기본" 이 아닌 다른 이름으로 다시 이름을 지정할 수 있으며 해당 다이어그램은 여전히 "기본"으로 취급됩니다. 패키지에서 다른 다이어그램을 선택하여 해당 패키지의 "기본" 다이어그램으로 만들 수도 있습니다. "기본" 다이어그램의 용도는 다음과 같습니다. 패키지 자체를 일부 기타 '호스트' 다이어그램에 배치한 경우 패키지를 두 번 눌러 기본 다이어그램을 열 수 있습니다.

이러한 메커니즘은 임의의 유형의 모델에 적용될 수 있는 다음과 같은 조직적 **가이드라인**을 지원합니다.

1. 설명할 각 모델링 파일의 기본 다이어그램(또는 기타 기본 다이어그램)을 작성하십시오.
    a. 모델링 파일의 각 최상위 레벨 패키지
    b. 모델링 파일의 루트 패키지에 있는 모든 기타 다이어그램의 다이어그램 아이콘(기본 다이어그램 자체에 대한 아이콘을 설명하지 마십시오)

2. 설명할 각 최상위 레벨 패키지의 기본 다이어그램(또는 기타 기본 다이어그램)을 작성하십시오.
    a. 직접 포함하는 패키지
    b. 직접 포함하는 모든 기타 다이어그램의 다이어그램 아이콘

3. 하위 레벨의 각 후속 패키지에 이 패턴을 반복하십시오.

5. 유스 케이스 모델의 내부 조직용 가이드라인

주: 이 섹션과 그 다음의 기타 모델 유형 특정 섹션에서는 RSA 에 포함된 경매 전시 예제에서 채택한 예를 사용하여 가이드라인을 설명합니다. 추가 조직 세부사항 및 다이어그램의 컨텐츠를 보려면 예를 학습하십시오.

유스 케이스 모델 상위 레벨 조직

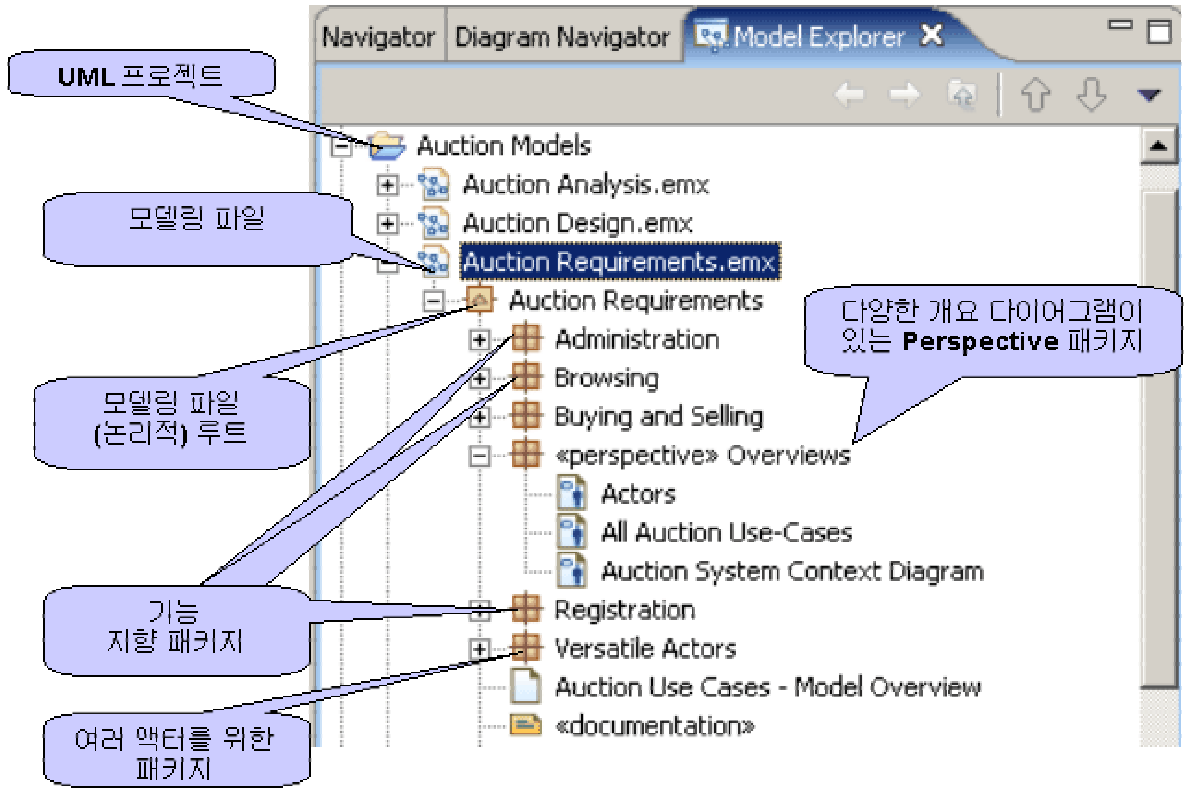


그림 5-1

그림 5-1 유스 케이스 모델을 구성할 수 있도록 다음 가이드라인을 설명합니다.

1. 최상위 레벨 패키지를 사용하여 기능 지향 그룹화를 설정하십시오. **근거**:

   - 팀원이 유스 케이스 모델에 대해 작업하는 경우, 이는 일반적으로 분업 관련사항에 잘 맵핑됩니다. 나중에 파일 경합이 문제가 되어 유스 케이스 모델을 여러 모델링 파일로 나누도록 결정한 경우에도 제대로 지원합니다(최상위 레벨 패키지마다 별도의 모델링 파일을 작성하면 됨).

   - 기타 접근법과 비교하여 일반적으로 최종 구현 조직에 보다 잘 맵핑됩니다. 이는 변환을 사용하여 연속된 각 하위 레벨의 추상을 시드(seed)할 경우에 중요합니다. 특히, 유스 케이스 모델을 기반으로 하는 분석 모델에 대한 시드(seed) 컨텐츠를 생성하려는 경우, 유스 케이스 모델의 패키징 구조가 대상 분석 모델의 원하는 패키징 구조에 제대로 맵핑되기를 원합니다.

그 다음에는 분석 모델의 패키징 구조가 설계 모델에 제대로 맵핑되고 설계 모델의 패키징 구조가 구현을 포함하는 프로젝트 세트에 보다 잘 맵핑되기를 원합니다. 맵핑이 간단할수록 한 추상 레벨에서 다음 레벨로의 변환을 구성하는 데 필요한 작업의 양이 줄어들게 됩니다.

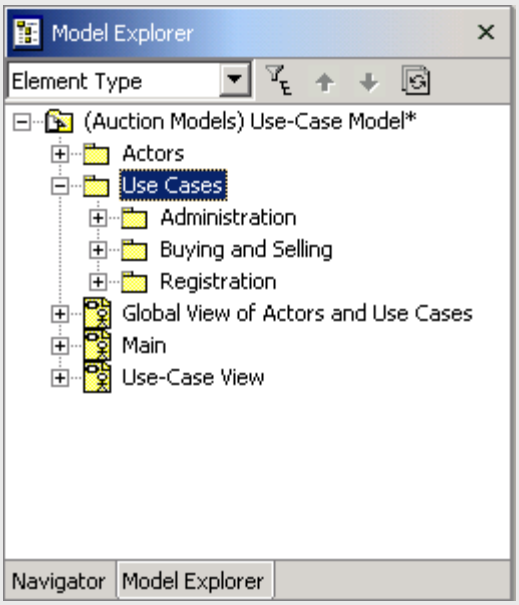
2. 다른 최상위 레벨 패키지를 사용하여 '광범위'하거나 '다양한' 권한을 가진 액터를 캡처하십시오.

3. «perspective» 패키지의 다이어그램을 사용하여 유스 케이스에 대한 상위 레벨의 교차 보기를 캡처하십시오. **근거**:

   - 기능 지향 그룹화로 조직된 모델의 의미론적 요소를 유지함과 동시에 교차 보기 및 '구조적으로 중요한' 유스 케이스에 대한 보기를 제공하십시오.

XDE/Rose

RSA 가이드는 유스 케이스의 액터 및 기타 요소에 대한 패키지를 작성하는 데 사용되었던 유스 케이스 모델의 상위 레벨 조직에 대한 기존의 가이드를 약간 개정합니다. 그런 다음, 모델이 요구하는 크기 및 복잡도에 따라 보다 낮은 레벨의 패키지를 사용하여 다음 XDE 기반 예제에 표시된 대로 기능 지향 그룹화를 설정합니다.



유스 케이스 모델 컨텐츠

이 문서는 좋은 유스 케이스를 작성하는 방법 또는 좋은 유스 케이스 모델링을 수행하고 있는지 여부에 대한 자세한 자습서의 역할은 하지 않습니다. 그러나 여기에 액터 및 유스 케이스와 함께 유스 케이스 모델에 포함될 수 있는 사항에 대한 간단한 설명이 나와 있습니다.

- **권장사항:** 모델 루트에 모델의 기타 패키지를 설명하고 이러한 패키지 및 각 '기본' 다이어그램에 대한 드릴 다운을 지원하는 '기본' 다이어그램을 작성하십시오.

- **권장사항:** 각 유스 케이스 패키지에 패키지의 유스 케이스, 해당 유스 케이스의 관계 및 여기에 참여하는 액터를 설명하는 다이어그램을 포함시키십시오. (경우의 수가 큰 경우 둘 이상의 다이어그램이 적합합니다.)

- **권장사항:** 문서 필드에 각 유스 케이스의 기본 및 대체 플로우를 설명하십시오.[4] (**그림 5-2** 참조)

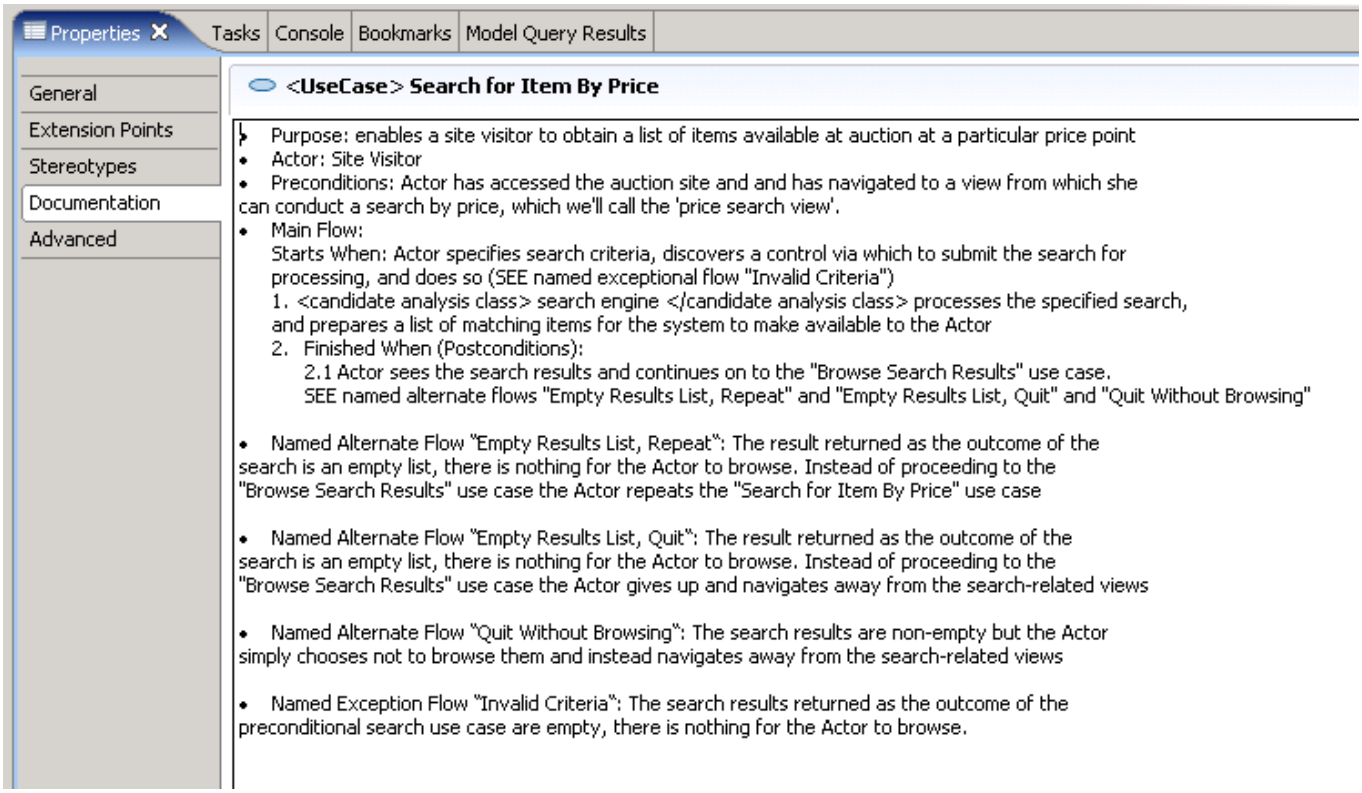


그림 5-2

- **선택사항:** 유스 케이스가 복잡하여 이를 허용할 경우, 활동 다이어그램을 추가하고 유스 케이스의 전체 활동 플로우를 반영하도록 작성하십시오(**그림 5-3** 참조). **근거:** 이는 각 (기본 및 대체/예외) 플로우에 대응하는 조건을 표시하는데 도움이 되며 모든 다양한 플로우가 궁극적으로 다시 합류하는지를 확인하는 데 도움이 됩니다. (RSM/RSA 에 활동 다이어그램을 추가하면 활동이 활동 아래 다이어그램과 함께 유스 케이스에 자동으로 추가됩니다)

- **선택사항:** 유스 케이스의 이름 지정된(기본, 대체 및 예외) 각 플로우의 '블랙 박스' 구현을 모델화하십시오. 유스 케이스에 협업 발생을 추가하십시오. 유스 케이스의 기본 플로우에 대응하는 상호작용 인스턴스와 함께 각 대체 및 예외라고 이름 지정된 플로우의 상호작용 인스턴스를 추가하십시오. 각 상호작용 인스턴스의 순서 다이어그램(또는 대체적으로 통신 다이어그램)을 작성하십시오. 이러한 유스 케이스 협업 인스턴스를 분석 레벨 유스 케이스 구현(분석 모델에서 설명됨) 또는 설계 레벨 유스 케이스 구현(설계 모델에서 설명됨)과 혼동하지 마십시오. 이는 유스 케이스의 "화이트 박스" 구현이며 솔루션의 내부 요소간 상호작용을 설명합니다. 여기서 유스 케이스 모델에 대해 제안된 협업 발생은 엄격히 액터와 시스템 간 "블랙 박스" 상호작용입니다. (**그림 5-3**

[4] 유스 케이스 설명 예제에 묘사된 형식은 RTF 가능 편집기를 사용하여 유스 케이스 설명에 대해 텍스트로 된 ' 템플리트'를 작성한 다음 해당 템플리트를 복사하여 유스 케이스 설명 필드에 붙여넣어 완성됩니다.

참조.) **근거:** 이는 기술적이지 않은 스테이크 홀더에게 시스템 사용자가 시스템과 상호작용하는 방법에 대한 상위 레벨 그림을 제공합니다. 이는 구현의 일부로 필요한 여러 보기(화면 또는 페이지)를 식별하는 데 도움이 될 수도 있습니다. 의미론적 모델 요소(협업 발생)에 이름을 지정하여 유스 케이스의 다양한 플로우(시나리오)에 이름을 공식적으로 설정합니다.

> *XDE/Rose*
> UML 1.x 에서는 이러한 목적으로 "협업 발생" 대신 "협업 인스턴스"를 사용했습니다.

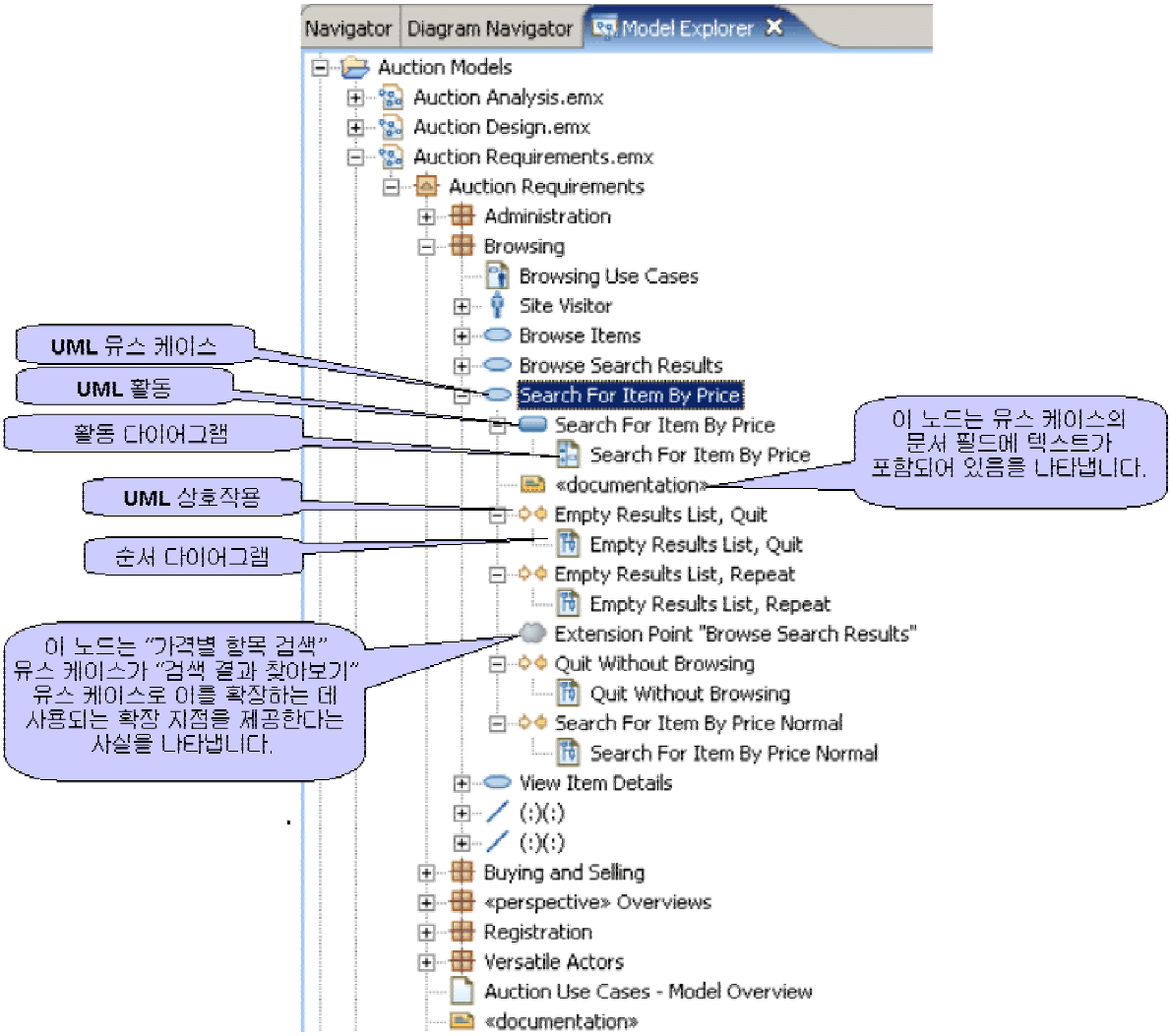


그림 5-3

- **선택사항:** RUP 가이드에 따라 사용자 구조의 '구조적으로 중요한 보기"를 식별하고 특히 소프트웨어 구조 문서를 유지보수하려는 경우, 최상위 레벨의 **«perspective»** 패키지를 추가하여 구조적으로 중요한 유스 케이스를 설명하는 유스 케이스 다이어그램을 포함시키십시오. "구조의 유스 케이스 보기" 패키지에 이름을 지정하고자 할 수도 있습니다.

6. 분석 모델의 내부 조직용 가이드라인

분석 모델은 솔루션의 첫 번째 판'을 나타냅니다. 이는 요구사항에서 최종 설계에 도달하기 위한 디딤돌로서 비즈니스 도메인에 대한 정보를 캡처하는 데 중점을 두고 비즈니스에 근접한 상위 추상 레벨의 후보 솔루션 요소를 표시합니다. 이는 분석 클래스 및 분석 레벨 유스 케이스 구현이 있는 위치입니다. 이는 솔루션에 필요한 클래스(특히 순서 다이어그램에서 필요하다고 느낀 라이프라인에 대응하는 클래스)를 *찾기* 시작하는 모델링 유스 케이스 구현(기본적으로 순서 다이어그램 사용) 프로세스를 통해 이루어집니다. 또한 유스 케이스 모델의 컨텐츠를 기반으로 하는 분석 모델 컨텐츠를 제안하는 데 적용할 수 있는 몇 가지 개략적인 규칙이 있습니다. 이는 이 섹션의 후반에서 다룹니다.

RUP 에서 설계 모델에서 분석 모델을 개별적으로 유지보수해야 하는지 여부는 프로젝트에 고유한 결정사항이며 별도의 분석 모델을 유지보수해야 할 가치가 있는지 여부를 기반으로 내려진 결정은 투자할 시간을 보증합니다. 별도의 분석 모델을 작성했지만 유지보수하지 않으면 분석 클래스는 설계 모델로 이동하여 정제됩니다. 또는 분석 모델이 점차 전개되어 설계 모델이 될 수도 있습니다.[5] 제품 특정 관점에서 고려할 수 있는 몇 가지 옵션이 있습니다.

1. 분석 모델 템플리트를 기반으로 모델링 파일(또는 파일 세트)에 있는 분석 모델을 작성하십시오. 그런 다음, 수동 프로세스 또는 자동 변환을 사용하여 정제된 버전의 분석 요소를 엔터프라이즈 IT 설계 모델 템플리트를 기반으로 두 번째 모델 파일에 작성한 다음, 분석 모델 파일을 제거하십시오. 별도의 분석 모델을 유지보수할지 또는 제거할지를 선택할 수 있는 옵션이 제공됩니다.
2. 분석 프로파일을 적용하는 엔터프라이즈 IT 설계 모델 템플리트를 기반으로 모델링 파일(또는 파일 세트)에서 분석 레벨 모델링을 수행하십시오. 이로써, 분석 클래스를 사용하여 유스 케이스 구현 모델링을 시작한 다음 설계 인터페이스가 행위에서 역할을 수행하도록 반복하여 해당 구현을 정제할 수 있습니다.
3. 두 번째 및 세 번째 옵션을 혼합한 것은 설계 모델과 동일한 모델링 파일에 분석 모델을 유지보수하기 위해서입니다. 이를 수행하기 위해 분석 컨텐츠를 «analysis» 키워드를 적용할 패키지로 분리합니다. 이는 분석 레벨 결과물을 보다 정제된 설계 레벨 사본(counterpart)과 동일한 모델링 파일 내에 보유할 수 있는 기회를 제공합니다.

RSA 변환을 사용하여 구현을 생성할 때 알아야 할 고려사항은 이러한 변환이 여러 경우에 분석 레벨 요소를 입력으로 받아들이므로 수동으로 해당 요소를 설계 요소로 정제하는 일부 단계를 생략할 수 있다는 것입니다. 이러한 식으로 RSA 를 사용할 경우, 위의 옵션 2 또는 3 이 선호됩니다. RSA 의 일부로 패키지된 표준 코드 생성 변환에서는 «analysis» 키워드가 있는 모델 패키지는 무시합니다.

[5] 실제로 RUP 가 분석 클래스 및 분석 레벨 유스 케이스 구현을 설계 모델에 작성하는 옵션을 소집한 다음 설계 양식으로 직접 전개합니다. 해당 접근법 하에서 설계 모델을 "발견"함에 따라 일부 "순수 분석" Perspective 를 보존하는 방식으로 패키지를 작성할 수 있습니다.

분석 모델 상위 레벨 조직

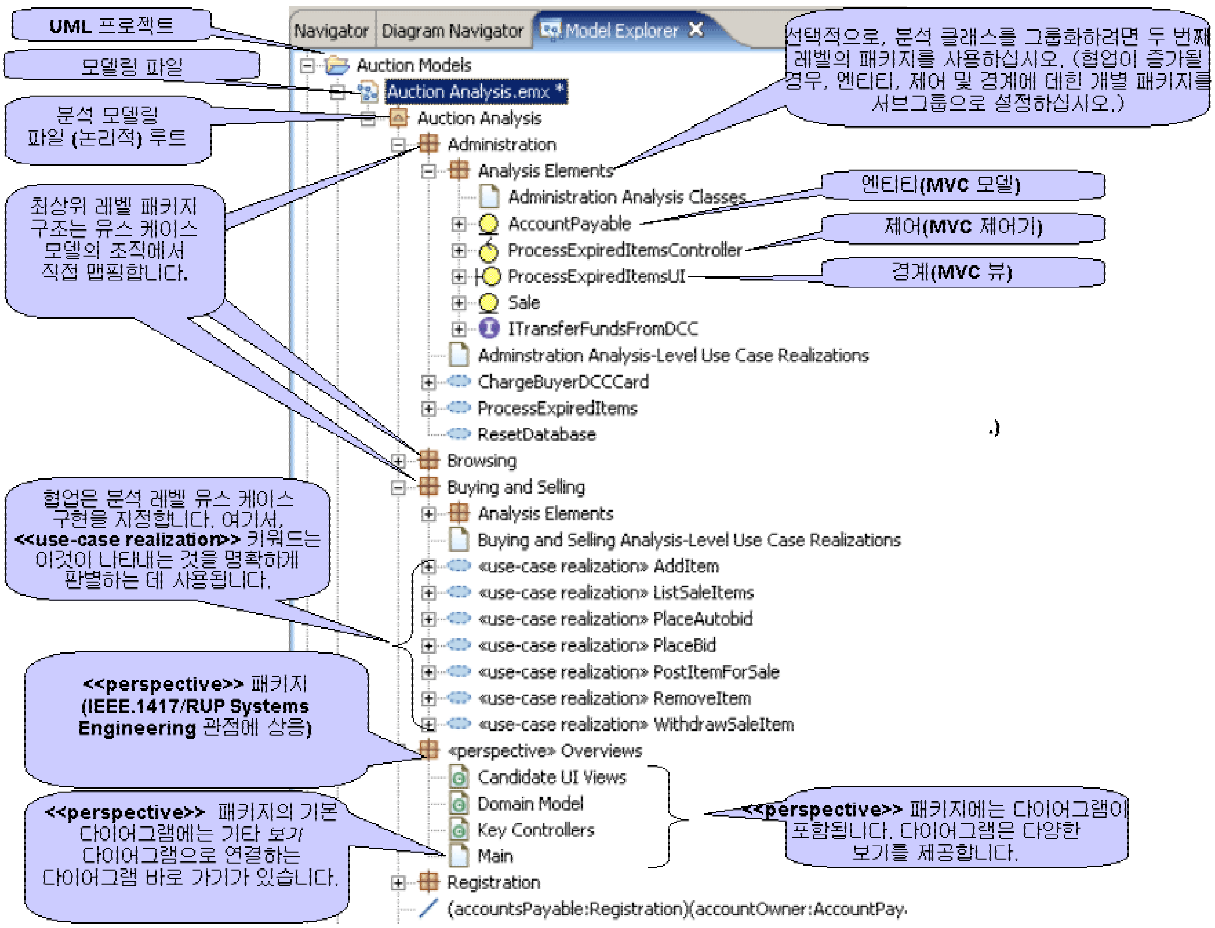


그림 6-1

그림 6-1 분석 모델을 구성하도록 다음 가이드라인을 설명합니다.

1. 최상위 레벨 패키지를 사용하여 분석 클래스에 대한 기능 지향 그룹화를 설정하십시오. **근거**: 유스 케이스 모델과 동일한 근거를 가집니다.

2. 선택적으로 최상위 레벨 패키지에서 서브 패키지를 사용하여 분석 클래스를 수집하고 조직하십시오.

3. «perspective» 패키지의 다이어그램을 사용하여 분석 요소의 대체, 상위 레벨 또는 교차 보기를 캡처하십시오. **근거**: 기능 지향 그룹화로 조직화된 모델의 의미론적 요소를 유지함과 동시에 다른 스테이크 홀더에게 다른 Perspective 를 제공하십시오.

최상위 레벨 패키지를 사용하여 분석 클래스로부터 유스 케이스 구현을 분리하는 것을 표시하는 약간 변환된 접근법이 **그림 6-2**에 설명되어 있습니다. 해당 최상위 레벨 패키지 내에 최상위 레벨 패키지와 일치하는 기능 지향 서브 패키지 세트가 있습니다. 이런 식으로 유스 케이스 구현을 분리함으로써 유스 케이스 구현의 조직에 영향을 미치지 *않고도* 분석 클래스를 포함하는 패키지 구조를 다시 세분할 수 있습니다. (특히, 분석 모델을 *원래 위치에서* 설계 모델로 전개할 경우, 클래스의 패키지 조직이 전개되어 유스 케이스에 원래 사용된 모델과 일치하지 않을 가능성이 있습니다.)

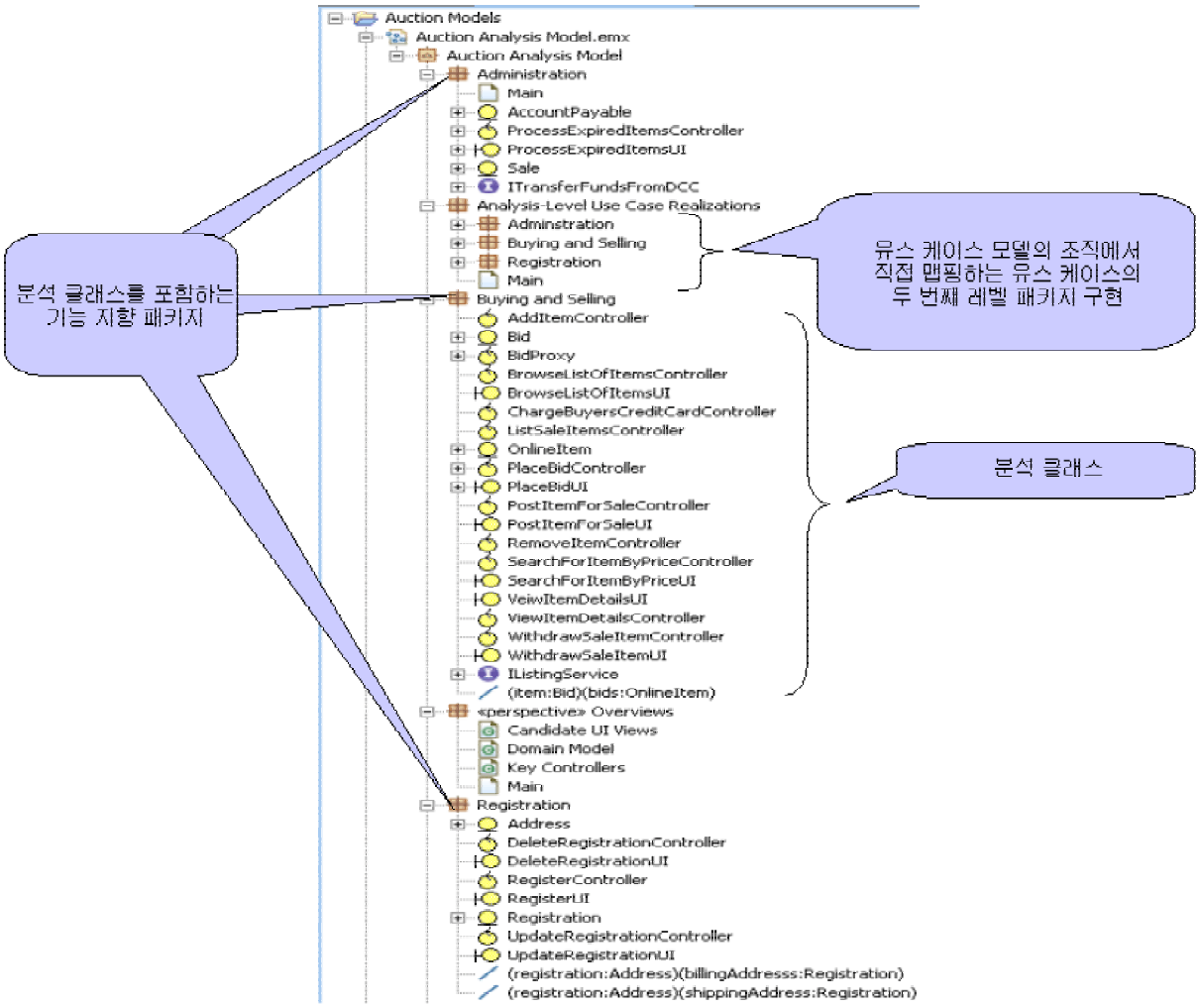


그림 6-2

사용자의 상황에 따라, 다른 (파트너) 비즈니스 내의 그룹을 비롯하여 여러 개별 그룹이 작성한 모델 컨텐츠를 병합하고 재사용하는 이름 지정 규칙을 도입해야 할 필요가 있을 수 있습니다. 이런 문제가 있는 경우, **그림 6-3**에 설명된 대로 역변환된 인터넷 도메인 네임스페이스 규정을 사용하는 것을 고려하십시오. 이는 *본질적으로* 분석 모델에 대해서는 주요 관심사가 아니지만 분석 모델을 *원래 위치*에서 설계 모델로 전개하는 접근법을 채택하여 설계 레벨에서 재사용 또는 비즈니스 통합이 예상될 경우에는 미리 계획을 세워야 함을 참고하십시오. 이 접근법을 채택하는 다른 잠재적 장점은 분석/설계에서 생성된 코드 조직에 제대로 맵핑되기 때문에 코드 생성 변환의 후속 형상을 단순화할 수 있다는 것입니다.

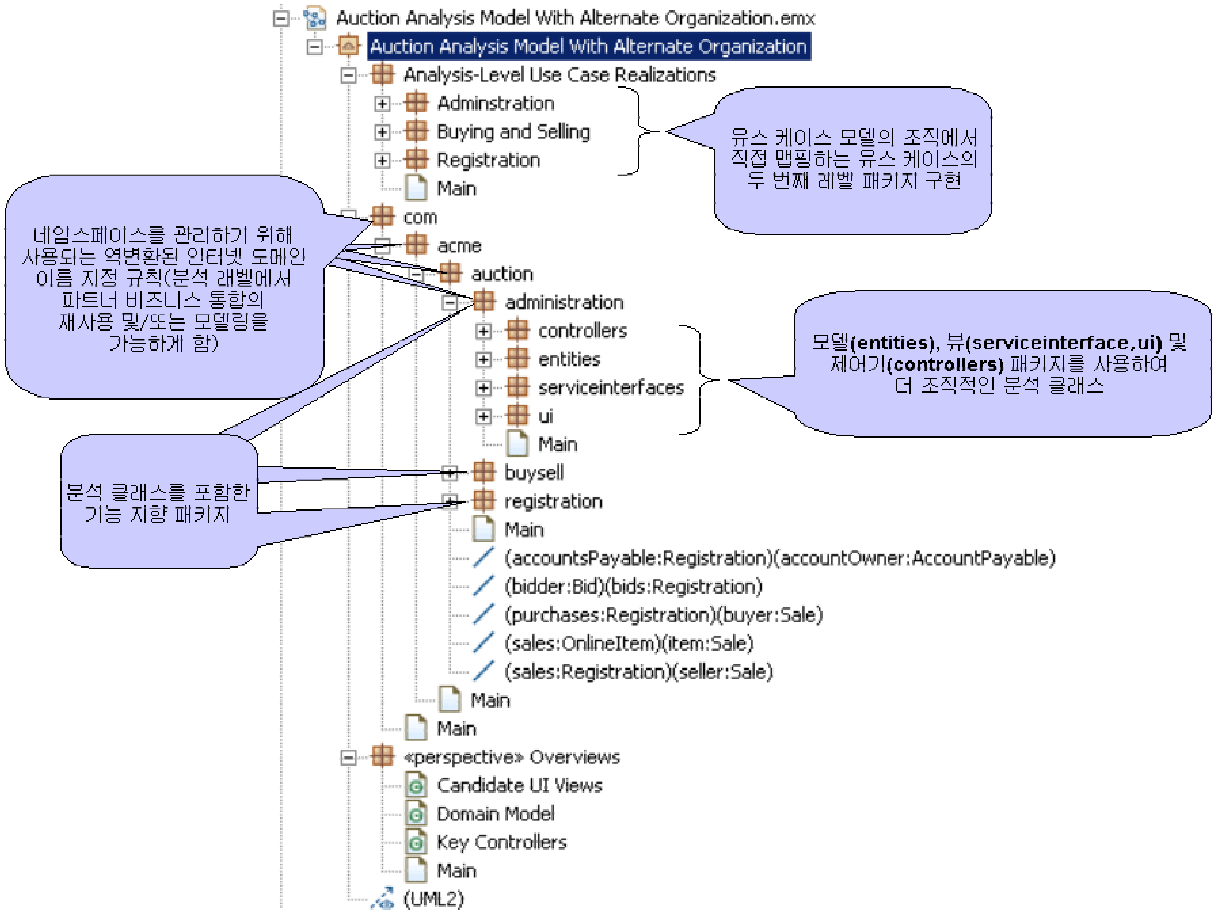


그림 6-3

분석 모델 컨텐츠

어떤 분석 클래스가 있는지를 찾을 수 있는 여러 가지 방법이 있습니다. 한 가지 방법은 유스 케이스 구현을 제안하는 순서 다이어그램을 그리기 시작하는 것입니다. 이를 수행함에 따라 어떠한 라이프라인이 필요한지를 발견하게 되며, 일반적으로 각 라이프라인은 분석 클래스에 해당합니다. 이런 방식으로 클래스를 발견하는 경우, 해당 클래스를 분석 모델의 유스 케이스 구현 패키지에 작성할 수는 있지만 해당 패키지에 그대로 두어서는 안됩니다. 앞서 분석 모델의 상위 레벨 조직용 가이드라인에서 설명한 바와 같이 분석 클래스를 기능 지향 패키지로 이동하려면 모델을 '다시 세분'해야 합니다(*그림 6-1* 참조).

분석 클래스를 발견하는 다른 유용한 접근법: 이러한 개략적인 규칙에 기초한 클래스로 분석 모델을 '시드(seed)'하십시오.

- 각 유스 케이스의 경우(유스 케이스 모델 내), «control» 클래스를 분석 모델에 추가하십시오. «control» 클래스는 유스 케이스와 연관된 비즈니스 논리를 표시합니다. (나중에 설계 시 세션 관리와 같은 고려사항에도 맵핑됨).

- 각 액터/유스 케이스 관계의 경우(유스 케이스 모델 내) «boundary» 클래스를 분석 모델에 추가하십시오. «boundary» 클래스는 솔루션 및 휴먼 액터 간 또는 솔루션 및 기타 외부 시스템 간 인터페이스를 표시합니다. 휴먼 액터에 대응하는 «boundary» 클래스는 결국에는 설계 및 구현의 하나 이상의 사용자 인터페이스 결과물에 맵핑될 가능성이 있습니다. 외부 시스템에 대응하는 «boundary» 클래스는 결국 설계 및 구현의 특정 유형의 어댑터 계층에 맵핑될 수도 있습니다.

- CRC 카드 분석 또는 유스 케이스 설명의 단어 분석과 같은 프로세스를 통해 추가 «control» 클래스(동사)와 «entity» 클래스(명사)를 식별하십시오.

이 시딩(seeding) 접근법을 사용하여 분석 클래스를 식별할 경우, 이전에 분석 모델의 상위 레벨 조직용 가이드라인에서 설명한 바와 같이 클래스를 기능 지향 패키지에 직접 배치할 수 있습니다(*그림 6-1* 참조).

그러나 사용자는 분석 클래스를 찾으려고 하며 원래 작동 패키지 조직에 변경이 필요하다는 사실을 인식하고 있습니다.

선택사항: 이러한 패키지의 컨텐츠를 더욱 체계화하려면 분석 클래스 패키지의 두 번째 레벨 패키지를 사용하십시오(**그림 6-4** 참조).

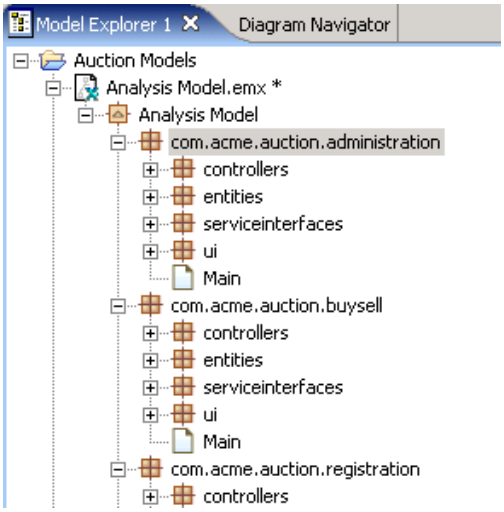


그림 6-4

권장사항: 분석 모델은 분석 클래스 관점에서 유스 케이스를 수행하는 방법에 대해 설명하는 분석 레벨 유스 케이스 구현을 **포함해야 합니다**. 각 분석 유스 케이스 구현(UML 협업에 의해 표시됨)은 유스 케이스 모델에서 유스 케이스를 구현하며 해당 유스 케이스와 동일한 이름을 갖습니다. **그림 6-5**를 참조하십시오. 분석 레벨 구현으로 모델화되어야 한다고 생각되는 각 이름 지정된 유스 케이스 플로우[6]의 경우, 순서 다이어그램(고유 상호작용을 자동으로 추가함)을 추가하십시오. **그림 6-6**에 순서 다이어그램을 작성함에 따라 모델에 추가되는 의미론적 컨텐츠의 유형이 표시됩니다. (모델 탐색기 보기에서 임의의 UML 요소 유형을 필터하여 **그림 6-6**에 설명된 'clutter'의 대부분을 숨길 수 있음을 참고하십시오.)

[6] 유스 케이스 모델에서 이미 설정된 대로

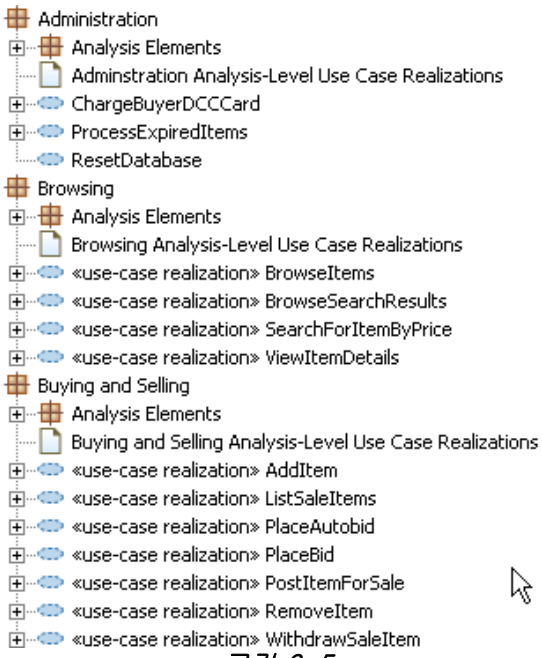


그림 6-5

선택사항: 유스 케이스 플로우에 대한 순서 다이어그램을 작성한 경우, 모델 탐색기에서 소유하는 UML 상호작용을 선택하고 의사소통 다이어그램을 추가할 수 있습니다. 새 의사소통 다이어그램은 순서 다이어그램에 참여한 분석 클래스 인스턴스로 자동으로 채워집니다.

권장사항: 각 유스 케이스 구현(UML 공동 작업)에서 구현 종속성 관계를 작성하고 유스 케이스 모델에서 대응하는 유스 케이스를 작성하십시오(**그림 6-6** 참조). 주제 다이어그램 및 추적성 분석과 같은 기능을 사용하여 모델의 추적성 관계를 이해할 수 있으므로 추적성 관계를 설명하는 영구적인 다이어그램을 반드시 보유할 필요는 없습니다. 예를 들어 다음과 같은 특정 유형의 '버리기(throw-away)' 다이어그램을 사용하여 관계를 작성하도록 권장합니다.

- 공동 작업에 자유 양식 다이어그램을 추가하십시오.
- 공동 작업을 여기에 끌어 놓으십시오.
- 유스 케이스를 여기에 끌어 놓으십시오.
- 종속성 관계를 그리십시오.
- 마지막으로, (모델 탐색기에서) 공동 작업에서 다이어그램을 삭제하십시오.

권장사항: 구현에 참가하는 분석 클래스(인스턴스가 유스 케이스의 구현에 대해 설명하는 상호작용 다이어그램에 표시되는 분석 클래스) 및 상호작용 다이어그램에 설명된 공동 작업을 지원하는 클래스 간 관계를 표시하려면 각 유스 케이스 구현에 대한 "참가자" 다이어그램을 포함시키십시오. **그림 6-6**을 참조하십시오.

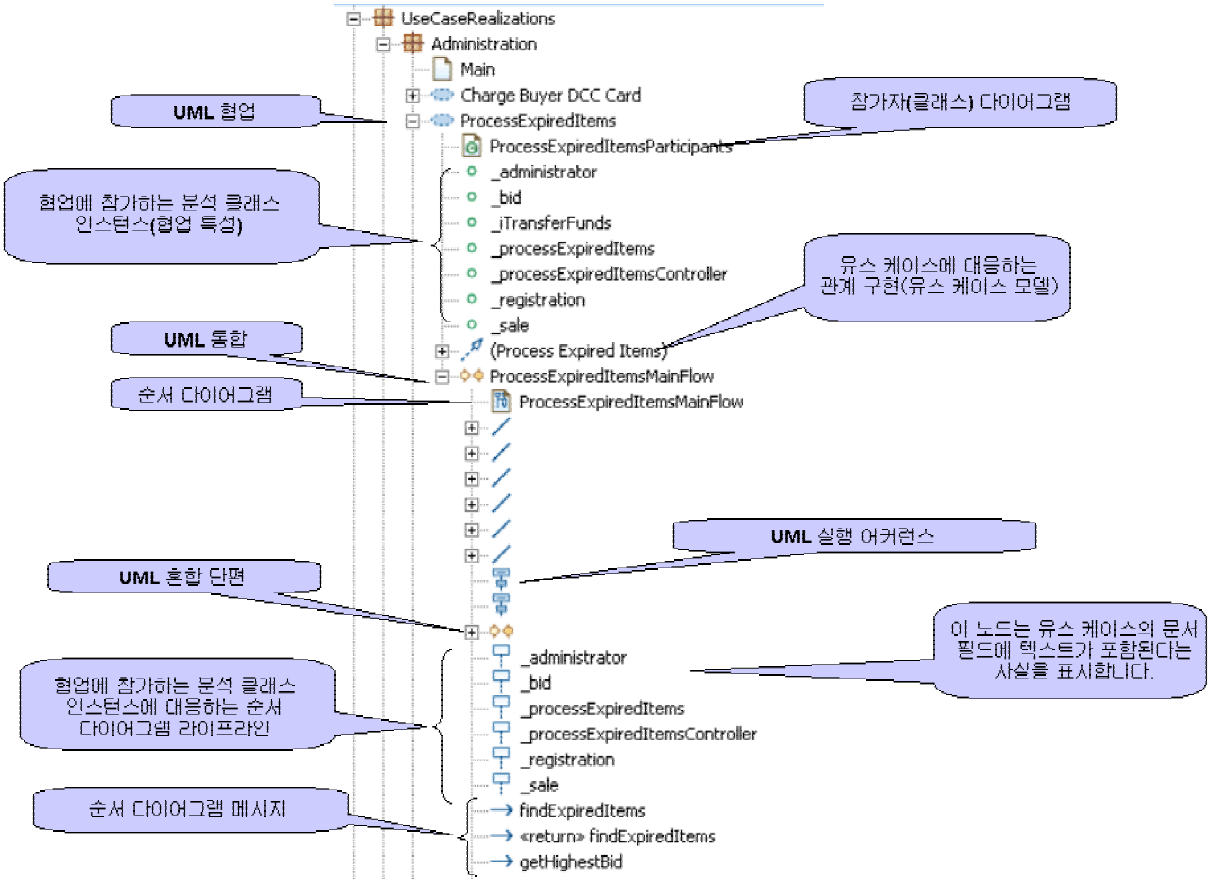
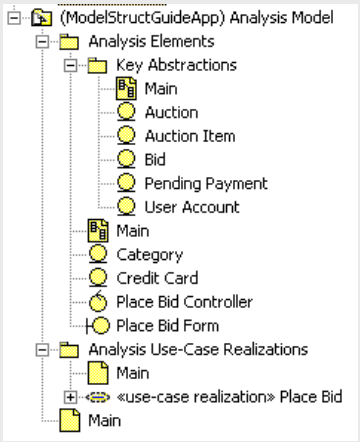


그림 6-6

XDE/Rose

아래에 표시된 대로 **분석 모델**에 대해 전통적으로 권장되는 구조가 분석 클래스의 기능 지향 패키지 조직을 강조하는 RSA 에 맞게 수정되었습니다. 핵심 추상 패키지를 사용하던 것이(다른 기능 지향 패키징 접근법과 절충)이 «perspective» 패키지의 핵심 추상 다이어그램을 사용하는 것으로 대체되었다는 점도 참고하십시오.



7. 설계 모델의 내부 구조용 가이드라인

설계 모델 상위 레벨 조직

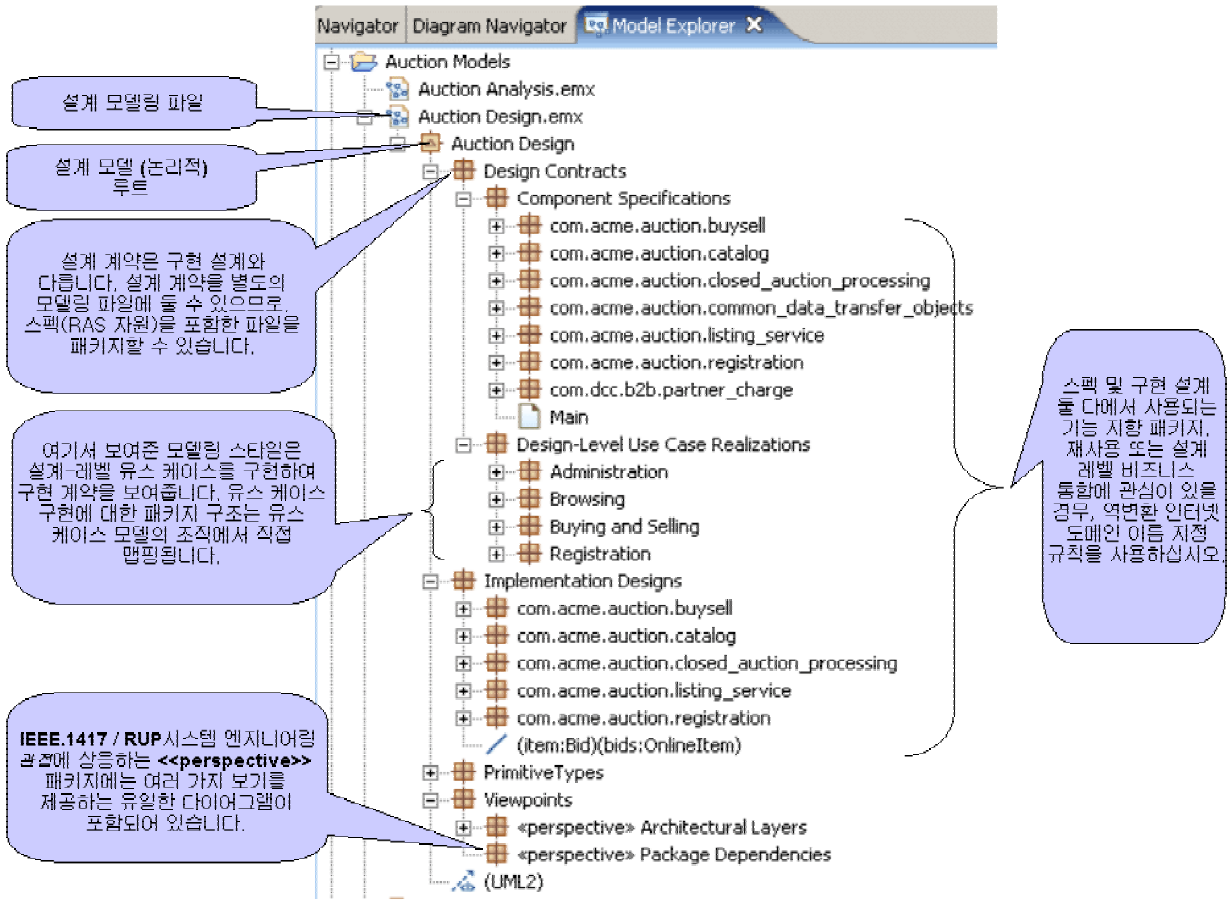


그림 7-1

그림 7-1 설계 모델을 구성하기 위한 다음 가이드라인을 설명합니다.

1. 구현 설계에서 스펙을 분리하십시오. 그림에 최상위 레벨의 “설계 계약 ” 및 “구현 설계” 패키지를 사용하여 이를 수행하는 방법이 표시되어 있습니다.

2. 보다 낮은 레벨 패키지를 사용하여 기능 지향 그룹화를 설정하십시오. 예를 들어, 분석 중에 사용한 조직에서 시작하여 분석 클래스를 실제 설계 클래스, 컴포넌트 및 서비스에 맵핑하는 방법에 따라 해당 조직이 전개되게 할 수 있습니다. (임의의 초기 조직 설계가 설계 중에 전개될 가능성이 있습니다. 자세한 내용은 아래의 설명을 참조하십시오.)

   여기서 서브시스템이라는 용어에 대해 설명을 하도록 하겠습니다. 버전 2 이전의 UML 에서 서브시스템은 특수한 유형의 패키지입니다. UML 2 에서 서브시스템은 특수한 유형의 컴포넌트이며 컴포넌트는 패키지를 포함할 수 있습니다. UML 2 에서 «subsystem» 컴포넌트는 패키지에 대한 실행 가능한 조직적/네임스페이스 대안이지만 서브시스템이 패키지에 해당한다고는 아직 확언할 수

없습니다. 제안사항: 특정 어플리케이션의 설계 서브시스템과 같은 세분 레벨에서는 패키지를 사용하고 엔터프라이즈 전반의 구조 보기에 전체 어플리케이션(예: CRM 또는 SCM)을 표시할 때에는 서브시스템을 사용하도록 정하십시오.

> *XDE/Rose*
> 이 문서를 작성할 때에는 Rose 및 XDE 모델 가져오기 툴이 UML 1.x 서브시스템을 UML2 서브시스템 또는 «subsystem» 키워드가 적용된 패키지에 맵핑하는 옵션을 제공할 것이라고 예상했습니다.

3. 설계 요소의 조직은 시스템의 유스 케이스를 구성하는 방법으로부터 전개될 가능성이 큽니다(별도의 분석 모델이 유지보수되는 경우, 유스 케이스 모델에서 또는 분석 모델에서도 가능). 패키지를 사용하여 설계 계약을 설계 요소 스펙(사용법 계약) 및 설계 레벨 유스 케이스 구현(구현 계약)으로 세분하고 유스 케이스 자체의 조직을 계속 반영하는 유스 케이스 구현의 패키지 서브 구조를 유지보수하십시오.

4. 구조 계층을 기능 영역의 구현 설계 및 스펙을 구성하는 요소의 두 번째 레벨 조직 설계의 기본으로서 사용하는 것을 고려하십시오(자세한 설명은 아래 참조).

5. 의미론적 모델 요소를 그룹화하는 UML 컴포넌트 및 패키지 내에 해당 그룹에 특정한 보기를 제공하는 다이어그램을 배치하십시오. 이 가이드라인에는 해당 그룹화가 비즈니스 도메인의 기능 지향 서브세트를 기반으로 하는지, 구조 계층을 기반으로 하는지 또는 사용자가 수행한 작업을 기반으로 하는지가 포함됩니다. '기본' 다이어그램이 패키지 또는 컴포넌트 자체와 동일한 이름을 갖도록 하고 패키지의 컨텐츠 개요를 표시하도록 작성하십시오. 이렇게 하면, 일부 다이어그램이 설명하는 내용과 근접하게 되어 모델을 탐색 및 이해하는데 용이하게 됩니다.

6. 설계 모델에 역변환된 인터넷 도메인 네임스페이스의 사용을 도입하고자 할 수도 있습니다. **근거**:
   - 기본적으로 이를 수행하는 동일한 이유가 다음과 같은 언어 특정 구현에 대해서도 중요합니다.
     a. 관련된 여러 모델 위주 어플리케이션(특히 파트너 회사와)이 있는 통합 작업에 관련된 시나리오
     b. 재사용 시나리오
   - 이는 변환의 후속 구성을 단순화할 수 있습니다(소스 대 대상 위치 및 이름 맵핑).

7. *네임스페이스 맵핑의 부담 및 잠재적 혼동을 없애기 위해* 대상 구현 플랫폼에서 유효한 패키지 이름을 사용하는 것을 고려하십시오. (대부분의 경우, 이는 단순히 이름에 밑줄 외에 공백이나 구두점을 사용하지 않는 것을 의미함).

8. 패키지 이름에 소문자를 사용하여 패키지의 클래스 이름과 쉽게 구분할 수 있게 하십시오.

9. 인터페이스 및 이를 구현하는 클래스 또는 컴포넌트에 대해 다른 이름을 사용하는 것을 고려하십시오. 인터페이스 및 구현 이름에 ILoan 과 Loan 또는 Loan 과 LoanImpl 을 사용하십시오. 사실상 이는 모델에서 필요하지 않지만 종종 생성된 코드에서는 유용한 아이디어입니다. 따라서 이는 일부 후속 변환 형상 작업을 수행하는 수고를 덜 수 있는 또 다른 영역입니다.

10. 다음 시나리오에서 코드를 생성하지 않게 되어 있는 모든 분석 레벨 컨텐츠는 «analysis»로 스테레오타입화된 패키지에 분리되어 있어야 합니다.[7].

[7] 이러한 패키지는 변환 시 무시됩니다.

- A) 별도의 분석 모델 사용을 생략하도록 선택하고 설계 모델을 분석 레벨 컨텐츠로 채우며 동일한 모델에 설계 레벨 컨텐츠를 작성함과 동시에 추상 분석 레벨에서 컨텐츠를 유지보수해야 합니다.

- B) EIT 설계 모델에서 모델에서 코드로의 변환을 수행합니다.

11. «perspective» 패키지의 다이어그램을 사용하여 설계 요소에 대한 상위 레벨의 교차 보기를 캡처하십시오. **근거**: 기능 지향 그룹으로 조직된 모델의 의미론적 요소를 유지하는 동시에 스테이크홀더의 다른 유형에 적합한 보기, '구조적으로 중요한' 컨텐츠 및 교차 보기를 제공하십시오.

설계 모델의 패키징 구조는 계속해서 전개된다는 것을 인식하는 것이 중요합니다. 궁극적으로, 조직이란 사용자 구조를 컴포넌트 및 서비스로 구성하는 방법에 해당합니다. 설계의 *후반* 조직에 대한 이 접근법은 일반적으로 재사용 가능 자원과 설계에서 프로젝트 및 폴더 세트(설계에서 생성된 구현 결과물(코드, 메타데이터, 문서)을 보유함)로의 가장 간단한 맵핑을 패키징하기 위한 최적의 잠재성을 제공합니다.

그러나 *초기* 조직은 유스 케이스 모델에 사용한 후 분석 중에 개정된 조직 접근법에 다소 직접적으로 해당됩니다.[8] 사실(이전 섹션인 "분석 모델의 내부 조직용 가이드라인" 에 설명된 대로), 분석 모델을 원래 위치에서 설계로 전개하도록 선택할 수 있습니다. 즉, 설계의 초기 조직은 기능이 응집되었으면서도 느슨하게 결합된 비즈니스 고려사항 세트로 그룹화한 다음 이를 교차 또는 재사용 가능한 요소로 분리하려는 경향이 있습니다. 초기 조직에 대한 이 접근법은 다음과 같은 이유로 효율적임이 입증되었습니다.

- 분석 또는 유스 케이스 모델 컨텐츠에서 설계 모델 컨텐츠를 생성하는 변환을 사용하려는 경우 소스 패키지에서 대상 패키지로의 맵핑이 단순하고 간단해집니다.

- 기능이 응집되고 느슨하게 결합된 패키지를 기반으로 하는 초기 조직 접근법은 최종 컴포넌트 지향 조직에 맵핑될 가능성이 있으며, 이는 설계 프로세스의 일부로서 수행해야 하는 세분화의 양을 줄일 수 있음을 의미합니다.

- 느슨한 패키지 결합은 팀 워크플로우를 개선하고 설계가 여러 모델링 파일로 분해되는 경우 재사용을 용이해집니다.

대체 접근법도 사용 가능하며 다음과 같은 일부 경우에는 *후반* 조직으로서 권장됩니다.

- EJB 를 포함한 J2EE 기반 웹 어플리케이션을 대상으로 하는 경우, 설계 조직은 RSA 및 J2EE 프로젝트 관련 Rational Application Developer 의 규정을 기대할 수도 있습니다.[9] 특히, 구조 계층에 해당하는 최상위 레벨 설계 패키지를 정의하도록 선택할 수도 있습니다(비즈니스 및 프리젠테이션은 세션 및 도메인으로 계층화됨). 이는 확실히 플랫폼 중립적 접근법이 아니므로 설계 중인 솔루션이 J2EE 외의 플랫폼에서는 구현되지 않는다는 것을 아는 경우에만 권장됩니다.

---

[8] 분석 클래스의 패키징은 재사용 및 예상치 않은 작동 요구사항을 보다 잘 지원할 수 있도록 종종 개발과 동시에 다시 세분됩니다.

[9] 간략하게 말하면, 시스템 또는 어플리케이션 또는 대형 서브시스템당 엔터프라이즈 프로젝트 및 각 엔터프라이즈 프로젝트용, 프리젠테이션 티어용 웹 프로젝트 및 EJB 프로젝트가 일반적으로 컴포넌트 또는 소수의 서브시스템에 대응하며 일반적으로 분리된 EJB 프로젝트가 컴포넌트 또는 서브시스템당 세션(세션 EJB) 및 도메인(엔티티 EJB) 계층에 사용되는 여러 EJB 프로젝트 . 자세한 정보는 이 문서의 섹션 9 를 참조하십시오.

- 보다 일반적으로, 개발자 전문가 및 분업이 프리젠테이션 및 비즈니스 계층에 대응하는 n 티어 어플리케이션을 빌드하는 경우가 빈번하므로 다시 이러한 구조 계층에 해당하는 최상위 레벨 패키지를 사용하도록 선택할 수도 있습니다. 특정 *구조* 를 지원하기 위해 특정 *비즈니스 기능*을 지원하는 클래스를 조직할 때에는 주의를 기울이십시오. 이런 경우 변경하기가 훨씬 힘들어 집니다.

- 비컴포넌트/서비스/서브시스템 지향 조직 접근법을 사용해야 할 이유가 있는 경우, 코드 생성 변환을 구성하는 데 좀더 노력을 기울여 설계 조직을 대상 프로젝트 및 폴더 세트에 계속해서 맵핑할 수 있어야 합니다. '맵핑 모델'이라고 하는 특수 유형의 컴패니언(COMPANION) 모델은 특히 복잡한 맵핑을 정의하는 데 사용할 수 있습니다.

설계 모델 컨텐츠

설계 모델에 있어야 하는 사항에 대한 엄격한 규칙은 없지만 다음 제안이 유용합니다.

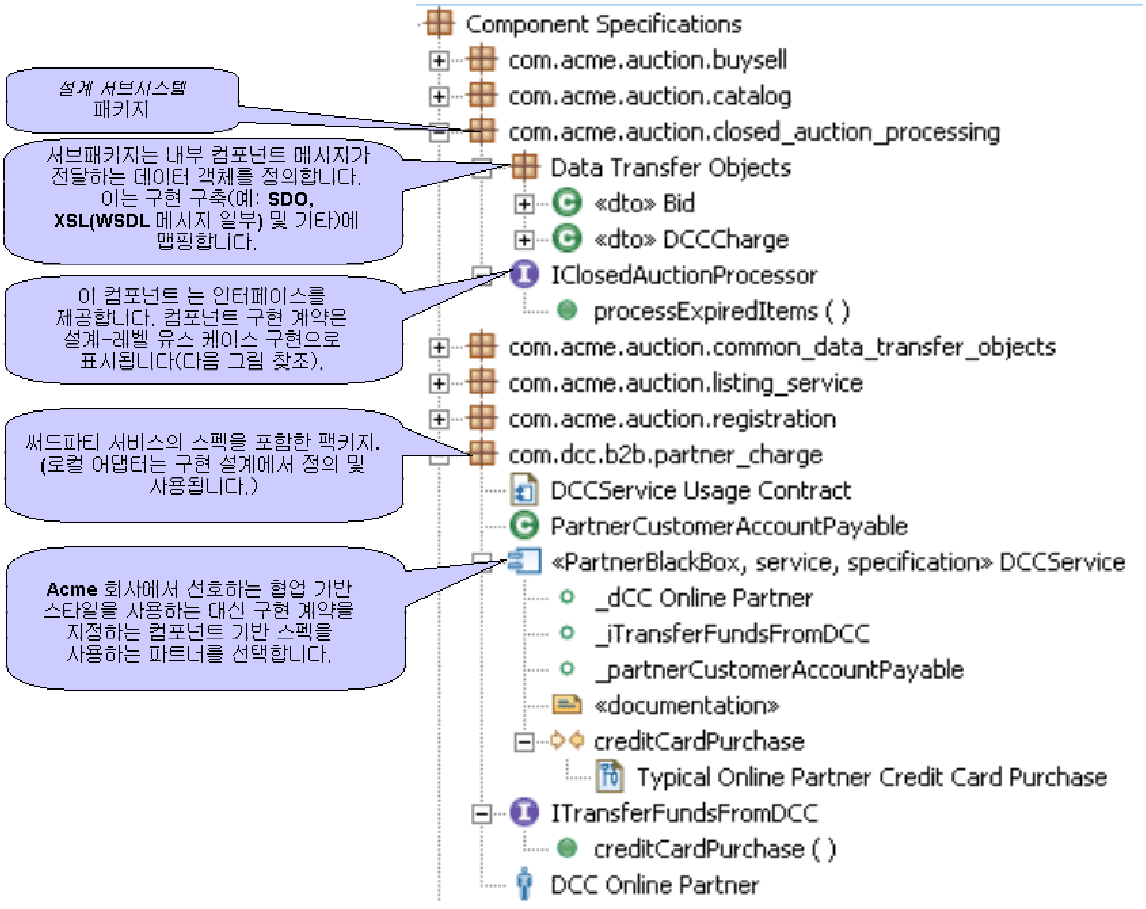


그림 7-2

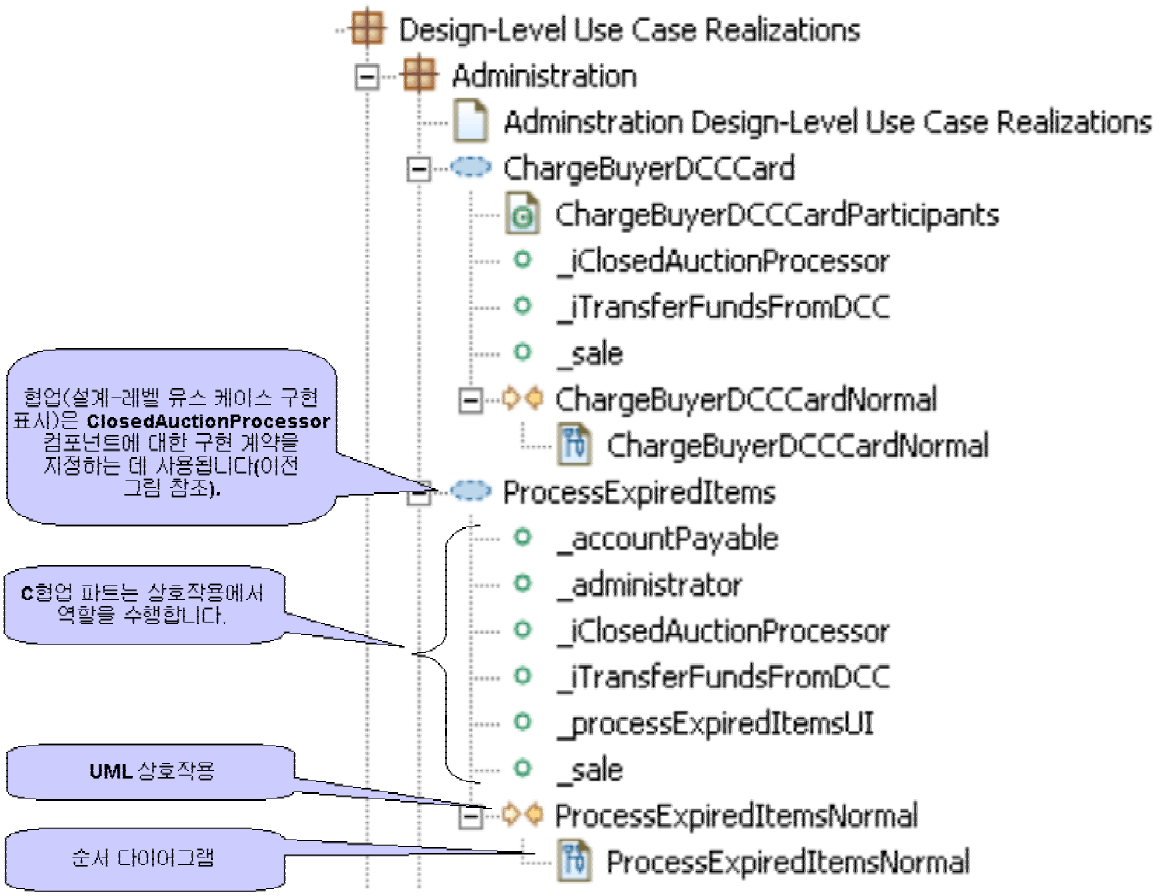


그림 7-3

그림 7-2 및 **그림 7-3** **그림 7-1**에 설명된 조직 구조를 준수하고 설계 계약을 지정할 수 있는 방법을 설명합니다.

- "ClosedAuctionProcessor" 컴포넌트의 사용법 계약은 단일 인터페이스로 표현됩니다.[10] (**그림 7-2**). 대응하는 구현 계약은 협업으로 표현되는 단일 설계 레벨 유스 케이스 구현에 의해 지정됩니다. [11] (**그림 7-3**) 분석 레벨 유스 케이스 구현은 분석 클래스 사이의 협업을 표시하는 반면 설계 레벨 구현은 덜 추상적인 설계 요소 사이의 협업을 표시함을 참고하십시오.[12] 설계 모델의 스펙 서브세트를 구현 설계 서브세트와 개별적으로 패키지해야 하는 경우, 설계 레벨 유스 케이스 구현은 해당 역할에서 분석 또는 설계 스펙 요소만 사용하고 구현 설계 요소는 사용하지 않아야 합니다.

[10] 물론 컴포넌트에는 여러 제공된 인터페이스가 있을 수 있으나 이 예에서는 오직 한 개만 있는 것으로 간주합니다.

[11] 기타 컴포넌트가 여러 시스템 유스 케이스에 참여할 수도 있으므로 해당 구현 계약이 여러 유스 케이스 구현에 있을 수 있습니다. 이러한 경우, 이러한 유스 케이스의 유스 케이스 구현을 구성하는 여러 다이어그램에 대한 링크를 배치하는 "{component name} 사용 위치"라는 다이어그램을 컴포넌트의 인터페이스와 동일한 패키지에 포함시킬 수 있습니다.

[12] 기타 가능한 차이점: 설계 레벨 구현의 일부 "참가자" 다이어그램은 분석 레벨 유스 케이스 구현에 대해 제안된 관련 클래스 다이어그램 대신(또는 이에 추가하여) 컴포넌트 배치를 묘사하는 컴포넌트 다이어그램일 수도 있습니다.

- 써드파티 “DCCService”의 사용법 및 구현 계약은 모두 한 패키지에 있습니다[13]. 사용법 계약이 단일 인터페이스로 구성되었음을 다시 한 번 확인하지만, 이런 경우 구현 계약은 «specification» 컴포넌트를 사용하여 표현됩니다(**그림 7-2**). (그렇지 않으면, 구현 계약의 스펙은 거의 비슷하며 행위(이 경우, “creditCardPurchase”라는 상호작용)를 사용하여 표현됩니다. 협업 대신 컴포넌트를 사용하는 기타 예제가 **그림 7-4**에 나와 있습니다.

- 조작은 «specification» 컴포넌트(사용하는 경우) 또는 인터페이스를 구현하는 구현 설계의 분류자에 의해 구현될 수 있는 인터페이스에 정의됩니다.

- 데이터 전송 오브젝트의 스펙(제공된 조작의 매개변수 유형 역할을 하며 XML 스키마 또는 SDO와 같은 구현 구조에 맵핑될 수 있음)은 사용법 계약의 일부로서 포함될 수 있습니다. 분배할 수 있게 설계되지 않은 컴포넌트의 경우, 데이터 전송 오브젝트를 조작 매개변수로 사용하는 유형의 스펙으로 지정하도록 선택할 수도 있고 선택하지 않을 수도 있습니다. 분배 가능한 서비스의 경우(예: 웹 서비스), 서비스 조작은 로컬 주소 공간에 있는 오브젝트를 참조하지 않는 것이 기본이므로 DTO를 사용해야 합니다.

> *XDE/Rose*
> 이전 버전의 UML 에서 유스 케이스 구현 가이드는 각 유스 케이스에 대한 협업 인스턴스와 중요한 각 구현 플로우에 대한 상호작용 및 순서 다이어그램을 사용하기 위한 것이었습니다.
>
> 이제 UML2 순서 다이어그램은 대체 실행 경로에 대한 표기법을 지원하므로 Rational Software Architect 에서는 하나의 상호작용 및 다이어그램을 사용할 수 있어야 합니다.
>
> 또한 UML 2 에는 더 이상 ‘협업 인스턴스’가 없습니다. 대신, ‘협업 사용’이 있으며 이는 해당 유형이 협업이어야 합니다. 그러므로 Rational Software Architect 에서 유스 케이스 구현을 표시하려면 협업을 사용하십시오.

- [13] 여기서 세운 가설은 DCC사가 Acme사에 UML 스펙을 제공하면 Acme사가 설계 모델에 통합된다는 것임을 참고하십시오. 역변환된 인터넷 도메인 공백을 사용할 수 있는 시나리오 유형입니다.
-

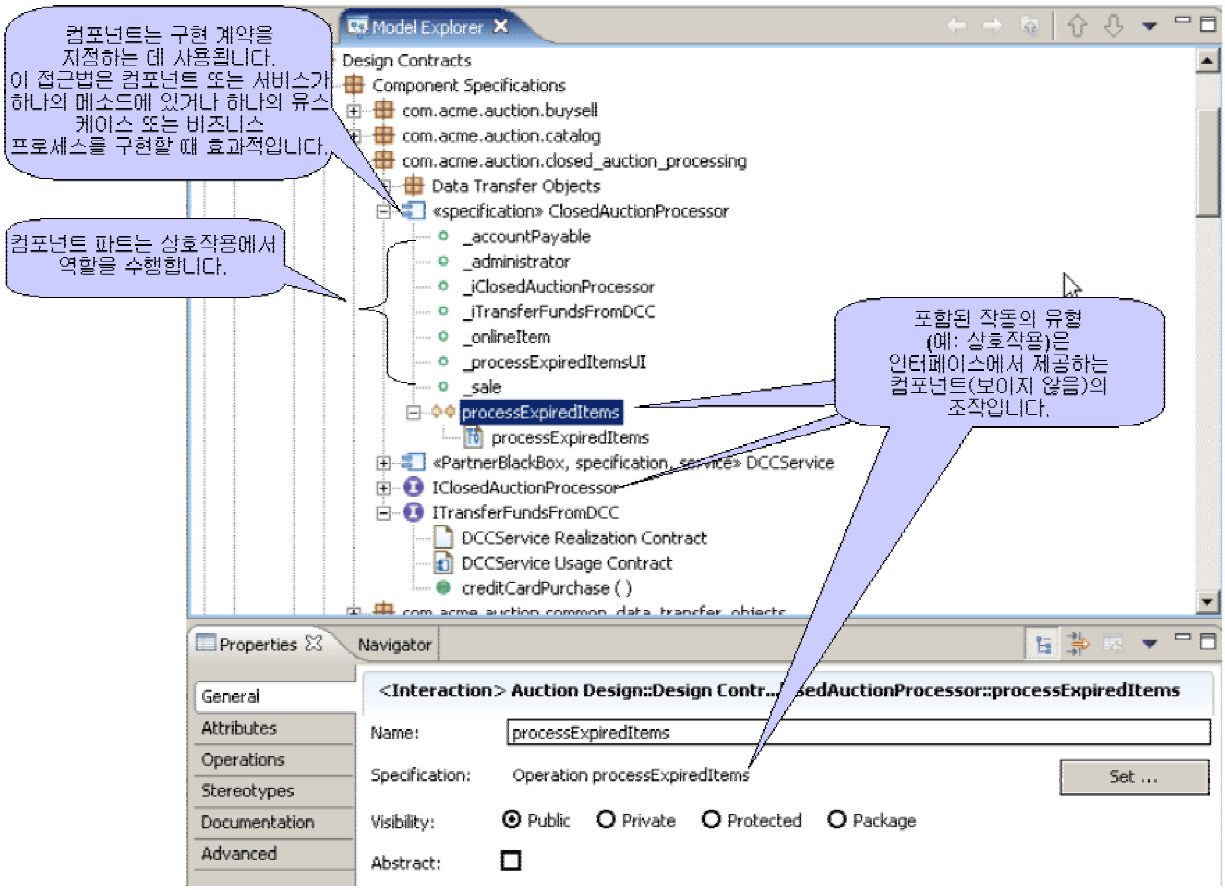


그림 7-4

- 구현 설계 지정에 가능한 접근법이 **그림 7-5**에 표시되어 있습니다. 구현 구조는 조작을 포함하는 단순한 클래스를 사용하여 정의됩니다. 이 접근법은 UML 1.x를 사용하여 작성한 매우 전형적인 설계 모델입니다. UML 2의 목표에 더 잘 부합하는 두 번째 가능한 접근법이 **그림 7-6**에 표시되어 있습니다. 여기서는 클래스 대신 컴포넌트가 사용되며 컴포넌트가 조작을 소유하지 않는 대신 행위(이 경우에는 상호작용)를 소유함을 알 수 있습니다.

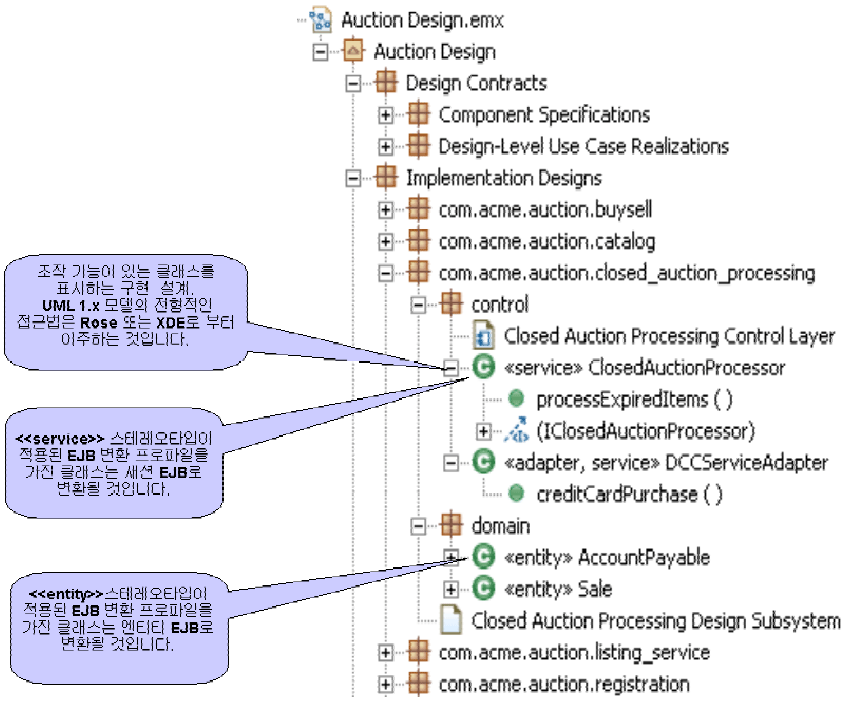

Auction Design.emx
Auction Design
Design Contracts
Component Specifications
Design-Level Use Case Realizations
Implementation Designs
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
control
Closed Auction Processing Control Layer
«service» ClosedAuctionProcessor
processExpiredItems ( )
(IClosedAuctionProcessor)
«adapter, service» DCCServiceAdapter
creditCardPurchase ( )
domain
«entity» AccountPayable
«entity» Sale
Closed Auction Processing Design Subsystem
com.acme.auction.listing_service
com.acme.auction.registration
조작 기능이 있는 클래스를 표시하는 구현 설계. UML 1.x 모델의 전형적인 접근법은 Rose 또는 XDE로 부터 이주하는 것입니다.
<<service>> 스테레오타입이 적용된 EJB 변환 프로파일을 가진 클래스는 세션 EJB로 변환될 것입니다.
<<entity>>스테레오타입이 적용된 EJB 변환 프로파일을 가진 클래스는 엔티티 EJB로 변환될 것입니다.

그림 7-5

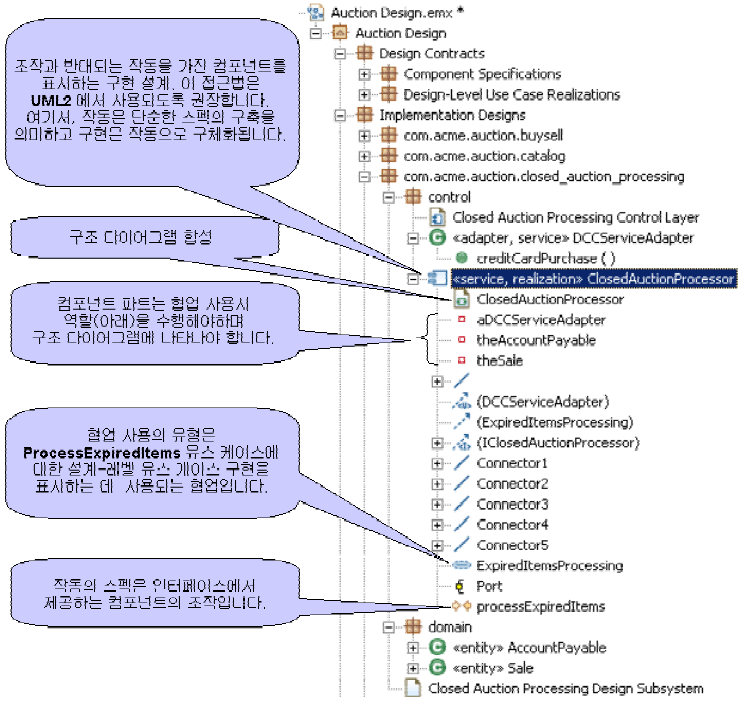

Auction Design.emx *
Auction Design
Design Contracts
Component Specifications
Design-Level Use Case Realizations
Implementation Designs
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
control
Closed Auction Processing Control Layer
«adapter, service» DCCServiceAdapter
creditCardPurchase ( )
«service, realization» ClosedAuctionProcessor
ClosedAuctionProcessor
aDCCServiceAdapter
theAccountPayable
theSale
(DCCServiceAdapter)
(ExpiredItemsProcessing)
(IClosedAuctionProcessor)
Connector1
Connector2
Connector3
Connector4
Connector5
ExpiredItemsProcessing
Port
processExpiredItems
domain
«entity» AccountPayable
«entity» Sale
Closed Auction Processing Design Subsystem
조작과 반대되는 작동을 가진 컴포넌트를 표시하는 구현 설계. 이 접근법은 UML2 에서 사용되도록 권장합니다. 여기서, 작동은 단순한 스펙의 구축을 의미하고 구현은 작동으로 구체화됩니다.
구조 다이어그램 합성
컴포넌트 파트는 협업 사용시 역할(아래)을 수행해야하며 구조 다이어그램에 나타나야 합니다.
협업 사용의 유형은 ProcessExpiredItems 유스 케이스에 대한 설계-레벨 유스 케이스 구현을 표시하는 데 사용되는 협업입니다.
작동의 스펙은 인터페이스에서 제공하는 컴포넌트의 조작입니다.

그림 7-6

8. 구현 개요 모델의 내부 조직용 가이드라인

> *XDE/Rose*
> XDE 모델 구조 가이드라인에서 구현의 서브시스템 레벨 개요를 제공하는 장치로서 구현 개요 모델이 권장됩니다. 그런 다음 각 서브시스템의 세부사항이 서브시스템을 구현하는 프로젝트의 코드 모델에 지정됩니다.

엄격히 말해서, RSA에서는 구현 개요 모델을 사용할 필요가 없어야 합니다. 설계 모델 조직 가이드라인을 따르는 경우 설계 모델의 (후반) 조직은 자연적으로 컴포넌트(더 중요한 «subsystem» 및 더 분배하기 쉬운 «service» 유형 포함) 주변에 구체화되어야 합니다. 그런 다음, 변환을 통해 설계 패키지를 프로젝트에 맵핑할 수 있습니다. 예를 들어, J2EE 구현의 경우, 패키지는 여러 Java, EJB, 웹, J2EE 어플리케이션 및 구현이 개발되는 기타 프로젝트에 맵핑됩니다. (사실상 이러한 프로젝트는 이 문서의 기본 개념 및 용어 섹션에서 설명한 바와 같이 솔루션의 구현 모델을 표시합니다.)

그러나 사용자가 여전히 초기 단계에 프로젝트 구조를 스케치하는 것을 선호하거나 단순히 시각적으로 보다 명백한 프로젝트 구조를 설명하는 것을 선호할 수 있습니다(예: 프로젝트 및 폴더를 표현하는 결과물이 «project» 및 «folder» 또는 «EJB Project» 및 «Web Project»로서 명시적으로 키워드화된 경우). 또 다른 고려사항은 정밀한 구현 결과물(예: JAR)을 설명하는 것이 설계 모델(조작의 Rational Software Architect 이론이 플랫폼 중립적이어야 하는 경우)에는 적합하지 않다는 것입니다. 그러나 이러한 결과물은 구현 개요 모델에 완전히 포함시킬 수 있습니다. 따라서 구현 개요 모델을 사용하려는 몇 가지 이유가 있습니다. 그림 8-1에 샘플 구현 개요 모델이 설명되어 있습니다.

마지막으로 구현 개요 모델은 솔루션의 다양한 측면의 비공식적 다이어그램을 캡처할 수 있는 좋은 위치가 될 수도 있습니다. 그림 8-2에 이 문서에 있는 대부분의 샘플이 기반으로 하는 경매 시스템의 비공식적 상위 개념 다이어그램이 표시되어 있습니다.

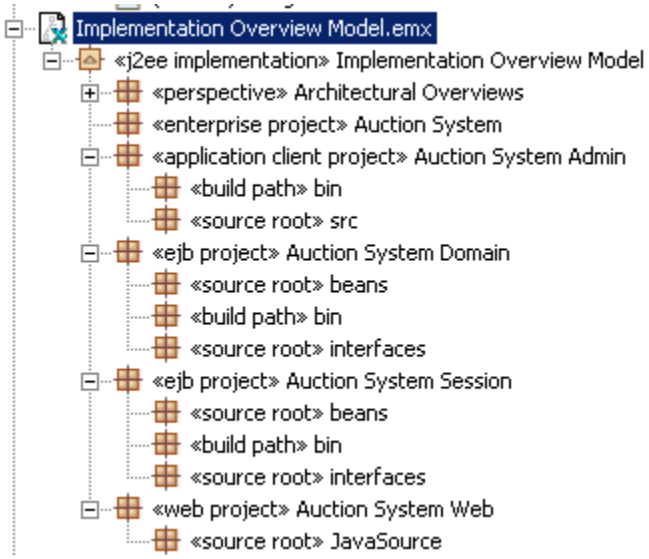

Implementation Overview Model.emx
«j2ee implementation» Implementation Overview Model
«perspective» Architectural Overviews
«enterprise project» Auction System
«application client project» Auction System Admin
«build path» bin
«source root» src
«ejb project» Auction System Domain
«source root» beans
«build path» bin
«source root» interfaces
«ejb project» Auction System Session
«source root» beans
«build path» bin
«source root» interfaces
«web project» Auction System Web
«source root» JavaSource

그림 8-1

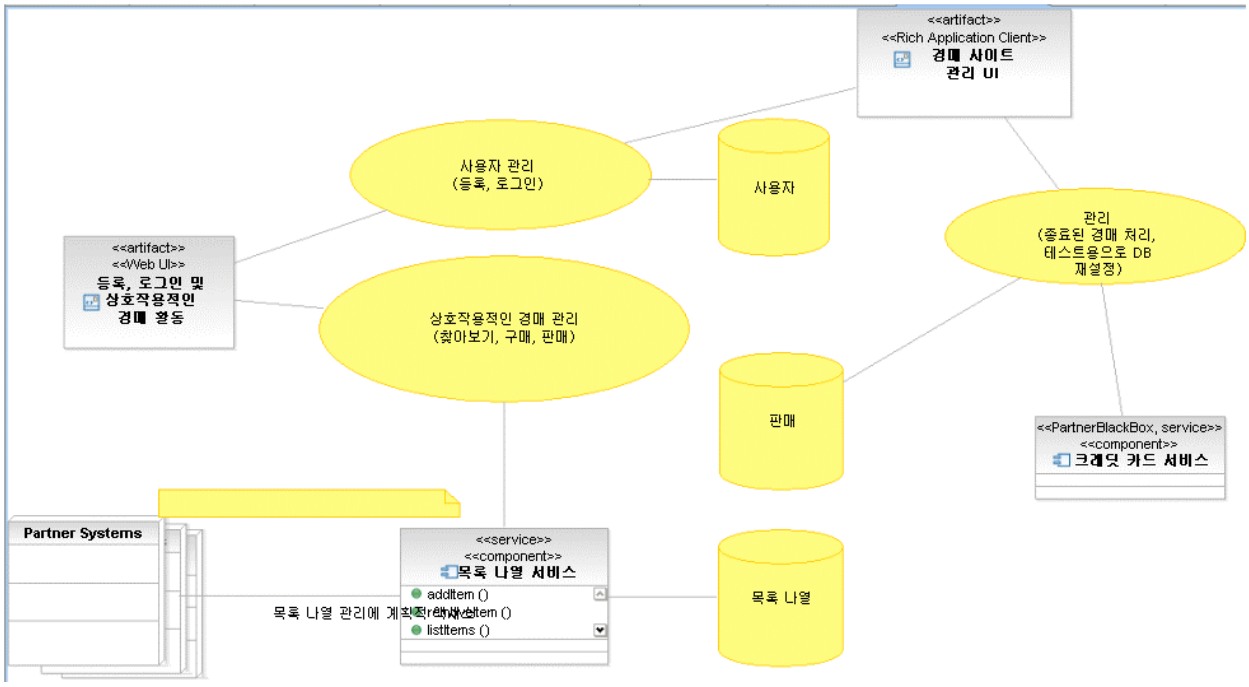

<<artifact>>
<<Rich Application Client>>
경매 사이트
관리 UI
사용자 관리
(등록, 로그인)
사용자
<<artifact>>
<<Web UI>>
등록, 로그인 및
상호작용적인
경매 활동
관리
(종료된 경매 처리,
테스트용으로 DB
재설정)
상호작용적인 경매 관리
(찾아보기, 구매, 판매)
판매
<<PartnerBlackBox, service>>
<<component>>
크레딧 카드 서비스
Partner Systems
<<service>>
<<component>>
목록 나열 서비스
addItem ()
listItems ()
목록 나열

그림 8-2

9. 전개 모델의 내부 조직용 가이드라인

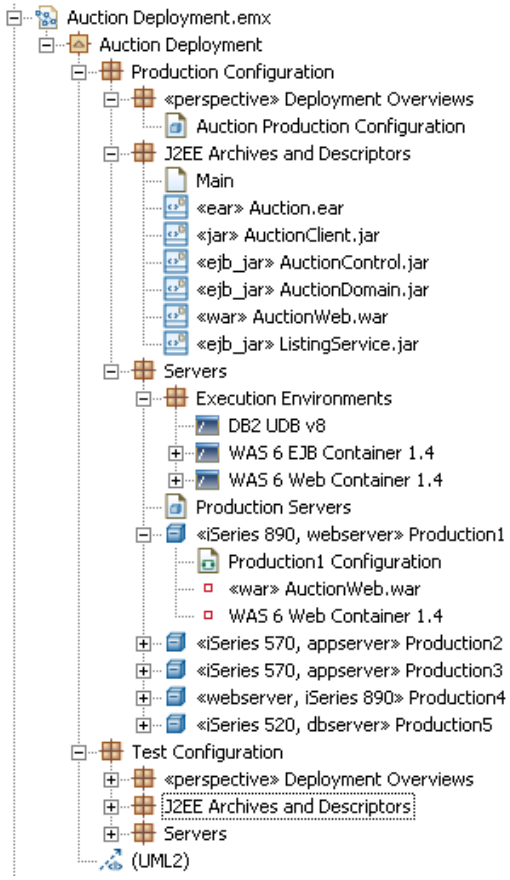


그림 9-1

이 문서에서 언급된 임의의 기타 모델에 대해서 보다 전개 모델에 대해 언급해야 할 필요가 오히려 적습니다. 일반적으로 전개 모델링 조직 및 컨텐츠 선택사항에 대한 다운스트림 관계가 거의 없으므로 합당한 조치를 수행하십시오. 사용자가 계속해서 생각을 떠올릴 수 있도록 가능한 전략 및 일부 대표적인 컨텐츠가 **그림 9-1**에 설명되어 있습니다. 이 예에서 다음을 참고하십시오.

1. 제품 형상의 스펙이 테스트 형상 스펙에서 분리되었습니다.

2. 개요(예: 클러스터, 데이터 센터 또는 엔터프라이즈)가 **«perspective»** 패키지에 유지보수됩니다.

3. 노드 및 결과물의 특수화 및 분류(패키징 조합 및 키워드 사용)와 관련하여 다음과 같이 간단한 접근법이 채택됩니다. 보다 복잡한 접근법은 사용자 고유 환경에 사용된 자원의 유형을 설명하고 문서화하는 데 적합한 특성과 특수 스테레오타입을 정의하는 특수 UML 프로파일을 개발하는 것입니다.

10. 모델링 파일을 사용하여 소프트웨어 구조 문서 표시

모델을 조직하는 데 필요한 툴이 제공된다면(예: 다이어그램 링크와 교차 모델 참조를 포함하는 여러 모델 파일 지원), RUP 소프트웨어 구조 문서와 "4+1 구조 보기"를 표시하는 모델을 작성하는 것은 사실상 너무도 간단한 작업이 됩니다.

가장 단순한 양식에서 **그림 10-1**의 라인을 따라 일부 작업을 수행할 수도 있습니다. 모델링 파일을 작성한 후 이를 4+1 보기에 대응하는 단순 패키지 세트로 채우십시오. (이 예의 시스템은 동시에 많은 내용을 표시하지 않으므로 프로세스 보기용 패키지 없이 예가 표시됩니다.)

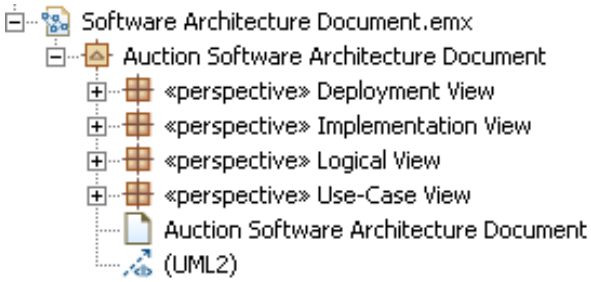


그림 10-1

그런 다음, **그림 10-2**에 제안된 라인을 따라 기본 다이어그램을 작성하십시오. 이 다이어그램에 추가 참고사항 또는 텍스트를 추가할 수도 있습니다.

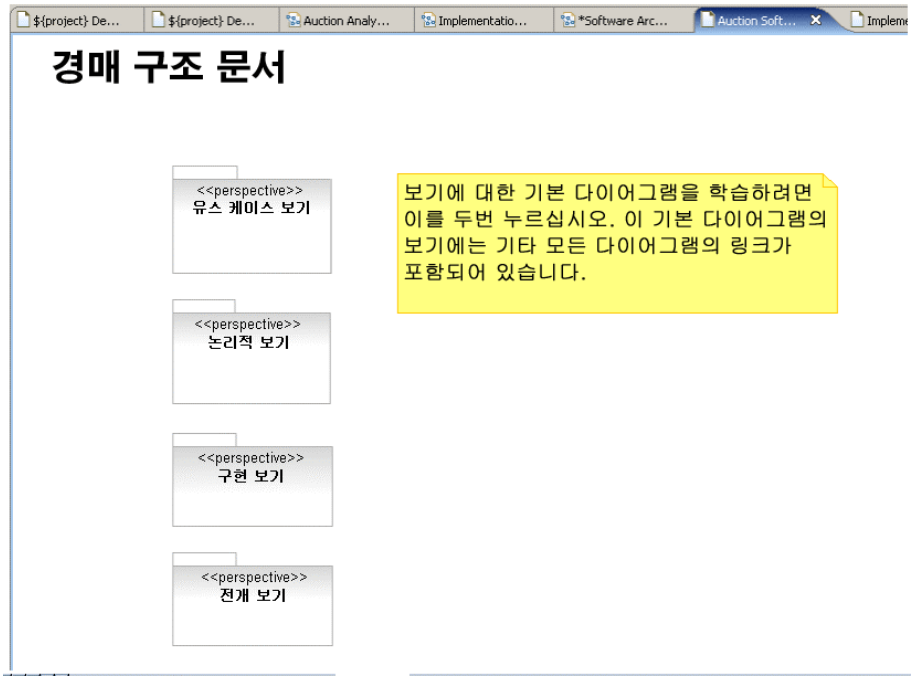


그림 10-2

그런 다음, 다음 접근법을 사용하여 소프트웨어 구조 문서 모델링 파일에 다이어그램을 작성하십시오.

- 기타 모델링 파일의 UML 의미론적 요소를 사용하여 작성되고 해당 기타 모델링 파일에는 없지만 구조 문서의 일부로서 필요한 새 보기를 설명하는 다이어그램을 작성하십시오.

- 소프트웨어 구조 문서 모델링 파일에 있는 기하학적 모양 및/또는 "임시" UML 요소로 구성된 다이어그램을 작성하십시오. (이러한 UML 요소는 문서화 또는 설명 용도로만 사용되어야 하며 설명되는 솔루션의 실제 구현에 대해 의미론적으로 중요하지 않아야 합니다.)

- 단순히 기타 모델링 파일의 기존 다이어그램에 대한 링크를 포함하는 다이어그램을 작성하십시오. (이 기술은 구조 문서 모델링 파일이 독자가 사용하는 기타 모델링 파일과 함께 분배되는 경우에는 제대로 작동합니다. 그렇지 않고 구조 문서를 웹으로 출판할 예정이면 기타 접근법 중 하나를 따르십시오.)

11. 팀 개발 고려사항

"기본 개념 및 용어" 섹션에서 RUP 가 인식하는 여러 모델(예: 유스 케이스, 분석 및 설계)에 대해 설명했습니다. RSM 또는 RSA 의 "사전 구현" 모델이 하나 이상의 모델링 파일로 존속할 수 있으며 여러 유스 케이스 모델, 여러 분석 모델 등을 유지보수할 수 있는 지점을 설명하는 예제도 제공되었습니다. 이 때 각 해당 모델은 단일 모델링 파일 또는 여러 모델링 파일로 존속합니다. 이 섹션에서는 모델을 여러 모델링 파일로 존속시키도록 선택해야 하는 경우와 이유에 대한 몇 가지 고려사항을 소개합니다. 이러한 문제에 대해서는 RSA 온라인 도움말에서 좀더 포괄적으로 다루어집니다.

팀의 모델링

여러 사용자가 한 모델에서 병렬로 작업해야 하는 경우, 모델을 여러 모델링 파일(.emx 파일)로 나누는 것이 보다 효율적이라는 것을 알 수 있습니다. 이렇게 하면 배타적 액세스를 위해 모델링 파일을 체크아웃하여 팀이 해당 작업을 수행할 수 있습니다. 그러면 둘 이상의 사용자가 동일한 모델링 파일을 수정함으로써 병합이 발생할 가능성도 줄어들게 됩니다.

한편, 트레이드오프가 있습니다. 여러 모델링 파일을 사용할 경우에는 병합을 자주 수행할 필요가 없으나, 그래도 병합이 발생하면 여전히 이를 수행해야 할 가능성이 있습니다. 병합은 모델링 *파일을* 그룹이 아닌 개별적으로 처리합니다. 즉, 모델을 여러 모델링 파일로 저장하면 개별 파일이 전체(논리) 모델의 "컨텍스트 외부 "에서 처리됩니다. 병합 세션이 전체 모델의 정보 컨텐츠에 더 많이 더 액세스할 경우(즉 모델이 *보다 적은 수의* 모델링 파일에 나뉘어져 있는 경우) 병합이 더욱 잘 이루어집니다.

궁극적으로, 모델을 여러 파일로 파티션하는 것은 병합 중에 해결해야 하는 충돌 변경사항을 초래하지 않고 여러 팀원이 모델에 대해 작업할 수 있게 한다는 면에서 모델 구조화만큼 중요하지는 않습니다. 병합을 현저히 줄일 수 있는 경우에만 모델(*해당 논리 인스턴스*)을 여러 실제 파일로 파티션하도록 권장합니다. 일반적으로, 팀 구성원이 *독점적*(오직 팀의 한 구성원만이 한 시점에 파일을 체크아웃함)이고 *개별적*(각 팀 구성원이 모델의 완전한 논리 컨텍스트를 구성하는 기타 파일에 액세스하지 않고도 개별 파일을 변경할 수 있음)으로 개별 파일에 대해 작업할 수 있는 경우에만 병합을 줄일 수 있습니다.

모델을 파티션하기 위한 두 가지 접근법

RSM 및 RSA 에서 모델을 세분하는 툴 기능을 사용하여 임시로 논리 모델 인스턴스를 여러 실제 파일로 나눌지 여부를 결정할 수 있습니다. 그러나 *미리 계획*을 세우도록 권장합니다. 다음 단락에 두 접근법이

비교 대조되어 있습니다.

계획된 접근법: 맨 처음에 모델 분해

팀의 공동 요구사항을 예측하는데 최선을 다하고 이에 따라 모델을 논리 서브세트로 세분하십시오. 예를 들면, 다음과 같습니다.

- 팀의 분석가 사이에 기능 전문가가 분포되어 있는 상태를 조사하여 각 유스 케이스 모델을 별개의 모델링 파일로 세분하고 이에 따라 모든 사항을 분류하십시오(본질적으로 모델 내 실제 유스 케이스를 조직하기 위해 패키지를 사용할 때마다 별도의 파일을 작성함). 예를 들어, 의료용품 제공업체 어플리케이션에는 환자 등록 유스 케이스를 포함하는 모델링 파일과 주문 처리 유스 케이스용 모델링 파일이 있을 수 있습니다. 또는, 주문 처리 케이스를 연구소 주문, 방사선과 주문, 약국 주문 등의 별도 모델링 파일로 세분할 수도 있습니다.

- 각 분석 모델을 여러 파일로 세분하십시오. 여기서 다시 권장되는 접근법은 팀 구성원 사이에 할당된 전문가 또는 문제점 도메인에서 자연스럽게 찾은 논리 그룹화에 따라 분석 모델을 나누는 것입니다 (가장 자주 조정해야 하는 두 개의 고려사항).

- 각 설계 모델을 사용자 구조의 주요 서브시스템, 서비스 또는 구성요소를 기초로 여러 파일로 세분하십시오(구현 작업을 팀 또는 개인에 할당하는 방법에 따라).

이러한 각각의 경우에, 각 모델에 대해 파티션(서브시스템/패키지/서비스) 경계를 교차하는 개요를 제공하는 다이어그램 및 공유/공통 요소를 포함하는 추가 모델링 파일 및 다이어그램을 유지보수할 수도 있습니다.

모델 파티션에 대한 추가 가이드는 RSM/RSA 온라인 도움말에서 찾을 수 있습니다.

임시 접근법: 모델 세분화

RSM 및 RSA 에서 하나의 파일로 이루어진 전체 모델에서 시작한 다음 나중에 모델의 분기를 '분리'하여 추가 파일을 작성함으로써 모델에 대한 여러 파일을 작성할 수 있습니다. 파티션 전략을 미리 계획했다 하더라도 팀 워크플로우를 향상시킬 수 있는 새로운 기회를 인식한 경우, 이 접근법이 유용할 수 있습니다.

이 토론을 위해, 모델의 논리 컨텐츠에 관심을 갖고 있으므로 분리된 모델의 분기를 "서브 모델"이라고 합니다. 서브 모델이 이를 포함하는 모델에서 분리되면 원래 모델이 서브 모델을 가리키는 '바로가기'를 유지보수합니다. 서브 모델은 원래 모델에 대한 역 포인터를 유지보수하지 않습니다. 기본값으로 새 서브 모델의 요소는 원래 상위 모델의 네임스페이스에 더 이상 존재하지 않습니다. 그러나 원래 모델의 네임스페이스를 새 서브 모델로 전달하는 분리 프로시저 중에 옵션이 제공됩니다. 원래 모델에서 서브 모델을 분리하여 작성하는 단계에 대해서는 RSM/RSA 온라인 도움말을 참조하십시오.

교차 파일 참조

임의의 두 RSA 모델링 파일이 다른 모델의 요소를 참조할 수 있습니다. 프로젝트에 걸친 참조를 포함할 수 있습니다. 이러한 참조 중 일부는 명시적으로 작성한 관계(예: 유스 케이스 간의 종속성 또는 클래스 간의 연관)을 나타냅니다. 나머지는 RSA 가 작성하며 일부 경우에는 숨겨져 있습니다. 예를 들어, 변환 어플리케이션을 통해 추적성 링크를 생성할 수 있으며, 이러한 링크는 일반적인 가시적 모델 요소로 작성됩니다. 또 다른 예제로서, RSA 다이어그램에는 해당 다이어그램에 설명된 의미론적 요소를 가리키는 참조가 포함됩니다(한 의미론적 요소가 여러 다이어그램에 표시될 수 있음). 표시 요소 참조는

“숨겨진” 참조입니다(모델 탐색기에 표시되지 않음).

각 교차 파일 참조는 다음과 같은 경우에 잠재적 중단점을 표시합니다.

- 하나 이상의 교차 참조된 파일을 제대로 저장할 수 없습니다(예: 시스템 충돌로 인해).
- 파일이 RSM/RSA 모델 서버가 모르는 상태에서 파일 시스템에서 이동합니다.

따라서 모델을 여러 파일로 나눌 경우 한 결과로 인해 교차 파일 참조가 확산되는 것은 아닌지를 고려하는 것이 중요합니다. 또한 선호하는 기능이 응집되고 느슨하게 결합된 조직 단위로 모델을 조직하면 팀의 개발 경험을 향상시키는 데 도움이 될 수 있습니다.