계층화 전략

Peter Eeles

Rational Software 백서

TP 199, 08/01

Rational®
the software development company

목차

요약

소프트웨어 시스템을 분해하는 많은 설명이 있습니다. 계층화가 한 가지 예이고 이 문서에서 설명됩니다. 이런 설명은 두 가지 주요 관심사를 제시합니다. 대부분의 시스템이 전체적으로 이해하기 너무 어렵고 여러 사용자에 맞는 여러 시스템 관점이 필요합니다.

계층화는 다수의 소프트웨어 시스템에서 채택되었고 많은 텍스트 및 RUP 에서 지지되고 있습니다. 그러나 종종 계층화가 잘못 이해되거나 올바르지 않게 적용됩니다. 이 문서는 계층화의 의미를 명확히 설명하고 여러 계층화 전략 적용의 영향에 대해 논의합니다.

계층화의 의미

계층화의 의미를 정의해 봅시다. *계층*이라는 용어는 일반적으로 *계층화 패턴*으로 알려진 구조적 패턴의 어플리케이션을 가리킵니다. 이것은 많은 텍스트([Buschmann], [Herzum], [PloP2])로 설명되고 RUP 에서도 설명됩니다. *패턴*은 특정 컨텍스트에 존재하는 일반적인 문제점에 대한 솔루션을 나타냅니다. 계층 패턴의 개요는 표 1 에서 제공됩니다.

표 1: 계층화 패턴의 개요

	계층 패턴
컨텍스트	분해가 필요한 시스템
문제점	완전히 이해하기에 복잡한 시스템 유지보수하기에 어려운 시스템 가장 덜 안정된 요소가 고립되지 않은 시스템 최대로 재사용 가능한 요소를 식별하는 데 어려운 시스템 될 수 있는 한 여러 기술을 가진 여러 팀에서 빌드되어야 하는 시스템
솔루션	시스템을 계층으로 구축

가장 익숙한 계층화의 예 중 하나는 ISO(International Standardization Organization)에서 정의된 OSI 7-계층 모델입니다. 이 모델은 그림 1 에 표시된 대로 네트워킹 프로토콜 세트를 정의합니다. 각 계층은 특정 통신 측면에 중점을 두고 계층 아래에 있는 계층의 기능을 토대로 합니다. OSI 7-계층 모델은 책임 기반 계층 전략을 사용합니다. 각 계층에는 특정 책임이 있습니다. 이 전략은 이 문서의 후반부에서 자세히 설명됩니다.

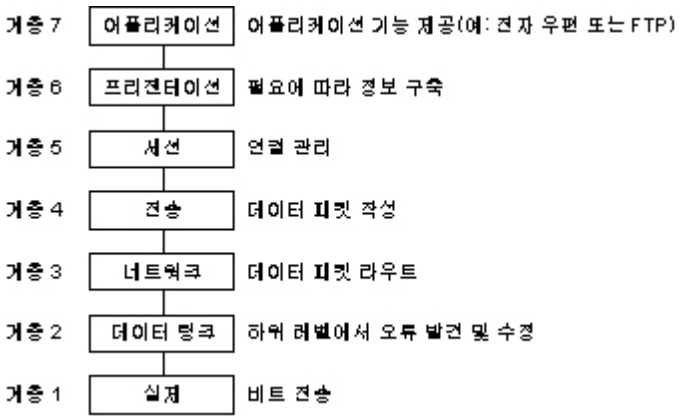


그림 1: OSI 7-계층 모델(책임 기반 계층)

그림 2 는 책임 기반 계층의 또 다른 예를 표시합니다.

- *프리젠테이션 논리 계층*은 사용자 인터페이스 요소와 같이 인간에 대한 몇 가지 표현 양식을 제공하는 책임을 맡는 요소를 포함합니다.
- *비즈니스 논리 계층*은 일부 유형의 비즈니스 처리 수행 및 비즈니스 규칙 적용을 담당하는 요소를 포함합니다.
- *데이터 액세스 논리 계층*은 관계형 데이터베이스와 같은 정보 소스에 대한 액세스 제공을 담당하는 요소를 포함합니다.

계층이 이 문서 후반에 설명된 대로 여러 가지 방법으로 모델화될 수 있음에 주의해야 합니다. 이제 스테레오타입 계층을 포함한 UML 패키지를 사용하여 계층을 명시적으로 표시합니다.

그림 2: 책임 기반 계층화

책임 기반 계층의 특별한 예를 표시하는 계층을 종종 층이라고 하며 *2 층*, *3 층* 및 *n 층* 시스템이 발견되는 분산 시스템 개발에 익숙한 개념입니다.

그림 2 에서 중요한 부분은 표시되는 *종속성의 방향*입니다. 왜냐하면 이것이 계층화된 시스템의 특성인 일정한 규칙을 암시하기 때문입니다. 특정 계층의 원소는 동일한 계층의 요소나 아래 계층의 요소만 액세스할 수

있습니다. [1]여기에 제공된 예에서 *비즈니스 논리 계층*의 요소는 *프리젠테이션 논리 계층*의 요소에 액세스할 수 없습니다. 또한 *데이터 액세스 논리 계층*의 요소는 *비즈니스 논리 계층*의 요소에 액세스할 수 없습니다. 이 구조를 종종 DAG(Directed Acyclic Graph)라고 합니다. 이것은 종속성이 단방향이라는 점에서 *지시적*이고 종속성의 경로가 절대 순환되지 않는다는 점에서 *비순환*입니다.

특정 참고로, 계층와 전략을 정의할 때 요소가 적절한 계층에 올바르게 놓일 수 있도록 각 계층의 의미에 대해 명확히 하는 것이 중요합니다. 요소를 적절한 계층에 올바르게 지정하는 데 실패하면 첫째로 전략을 적용하는 가치가 줄어듭니다. 각 계층화 전략이 세부적으로 논의되며 일부 각 계층의 의미에 대한 일반 가이드가 제공됩니다.

계층 모델링

여러 계층화 전략을 조사한 결과, 특정 *모델*(및 그 결과 특정 UML 요소)을 사용하여 각 전략을 전달하는 데는 계층 모델링이 적합하다는 것이 명확해졌습니다. *모델*은 특정 관점에서의 완전한 시스템 설명을 나타냅니다. 그림 3 은 고려 중인 시스템의 여러 관점을 나타내는 4 가지 모델의 예를 표시합니다.

- 유스 케이스 모델: 시스템 요구사항 캡처
- 분석 모델: 시스템 요구사항 분석 캡처
- 설계 모델: 시스템 설계 캡처
- 구현 모델: 시스템 구현 캡처

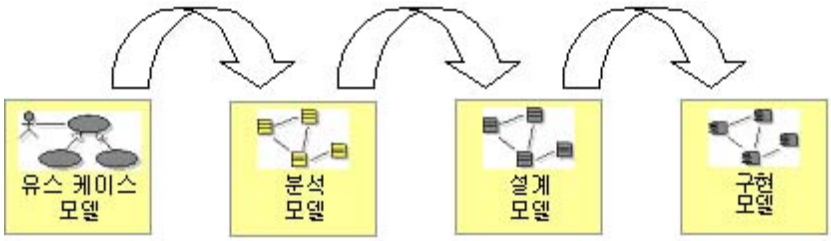


그림 3: 점진적인 정제를 표시하는 4 가지 모델

추가 모델에는 다음이 포함됩니다.

- *전개 모델:* 시스템의 분배 측면 캡처
- *데이터 모델:* 시스템의 지속적 측면 캡처

계층화 전략

계층화는 다양한 특성을 기반으로 할 수 있습니다. 이 섹션에서는 다음 특성을 기반으로 하는 계층화에 대해 논의합니다.

- 책임
- 재사용

[1] 이벤트 공고는 한 계층에 있는 요소로부터의 메시지가 상위 계층에 있는 요소로 전송되도록 할 수 있지만, 이 방향에서 명시적인 종속성은 없습니다.

각 전략이 세부적으로 논의될 때 각 전략의 표시가 고려됩니다.

책임 기반 계층

아마도 가장 일반적으로 사용되는 계층화 전략은 책임에 기반한 계층화 전략일 것입니다. 다양한 시스템 책임이 서로로부터 분리되므로 이 특별한 전략이 시스템의 개발 및 유지보수를 개선할 수 있습니다. 예를 들어(그림 2 참조), 시스템은 다음 책임을 기반으로 계층화될 수 있습니다.

- 프리젠테이션 논리
- 비즈니스 논리
- 데이터 액세스 논리

이런 각각의 책임은 그림 4 에 표시된 대로 각 계층에 대한 일부 샘플 컨텐츠를 표시하는 계층으로 나타낼 수 있습니다. 여기에서는 주문 처리 시스템에서 고객, 주문 및 제품의 세 가지 개념을 고려합니다. 예를 들어, *고객* 개념은 다음으로 이루어집니다.

- *CustomerView 클래스:* 사용자 인터페이스에서의 고객 렌더링과 같이 고객과 연관된 *프리젠테이션 논리* 담당.
- *Customer 클래스*: 고객 세부사항의 검증과 같이 고객과 연관된 *비즈니스 논리* 담당.
- *CustomerData 클래스*: 고객 상태를 지속적이게 하는 것과 같이 고객과 연관된 *데이터 액세스 논리* 담당.

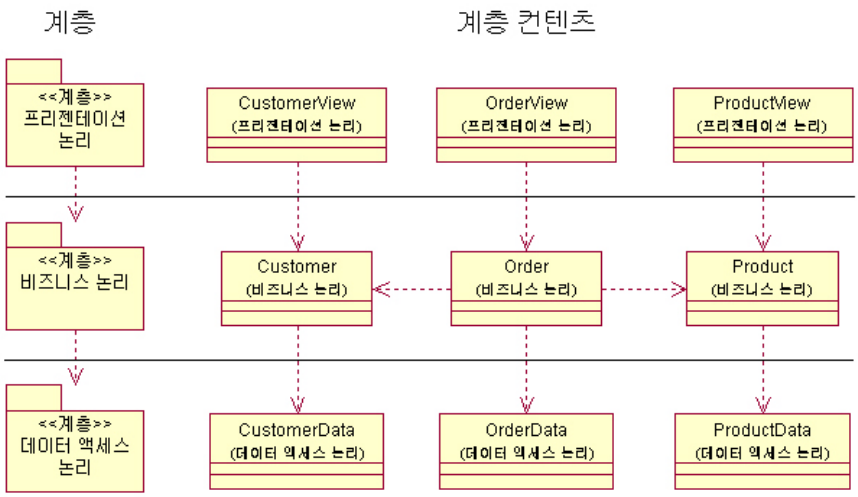


그림 4: 책임 기반 계층화에 해당하는 계층 및 컨텐츠

이제 이 특별한 계층화 전략에 관해 몇 가지 가설을 고려해 봅니다.

가설 1: 계층 및 층이 다름

이 특별한 가설은 일반적으로 마주치게 되는 혼란의 원인입니다. 특정 전략(책임 중 하나)에 기반한 계층이지만 사실상 *계층*입니다. 수많은 방법으로 층의 개념이 적용될 수 있기 때문에 혼란이 가중됩니다(표 2 참조).

표 2: 층 정의

어플리케이션	계층(층)
2 층	결합된 프리젠테이션 논리 및 비즈니스 논리 데이터 액세스 논리
3 층	프리젠테이션 논리 비즈니스 논리 데이터 액세스 논리
N 층	프리젠테이션 논리 비즈니스 논리(*분배됨*) 데이터 액세스 논리

가설 2: 계층(층)이 실제 분배를 포함함

또 다른 일반적인 오해는 논리 계층화가 실제 분배를 포함한다는 것입니다. 3 층 계층화를 생각해 보십시오. 다양한 요소가 계층 중 하나에 있더라도 각 계층 자체는 표 3 에 표시된 많은 방법(씬 클라이언트)으로 적용될 수 있습니다. 이것은 종종 특정 물리적 분산을 특징화하는 데 사용되는 이름을 사용합니다.

표 3: 어플리케이션 3 층 계층화

어플리케이션	계층	
	클라이언트측	서버측
단일 시스템	프리젠테이션 논리 비즈니스 논리 데이터 액세스 논리	
씬 클라이언트	프리젠테이션 논리	비즈니스 논리 데이터 액세스 논리
팻 클라이언트	프리젠테이션 논리 비즈니스 논리	데이터 액세스 논리

또한 단일 시스템이 여러 개의 물리적 분산 전략을 사용할 수 있으며, 여기에서 특정 요소가 씬 클라이언트 분산 및 다른 팻 클라이언트 분산을 요약하는 것으로 분류될 것입니다. 일반적으로, 선택은 성능과 같은 비기능적 요구사항에 따라 이루어집니다.

책임 기반 계층 모델링

뒤에 설명된 대로, 이 전략의 어플리케이션은 *설계 모델*, *구현 모델* 및 *전개 모델*에 영향을 줄 수 있습니다. *설계 모델*은 일반적으로 두 가지 방법 중 하나를 사용하여 구축됩니다.

첫 번째 방법은 계층 내에 포함되는 요소를 표시합니다. 그 결과는 다음을 표시하는 그림 5, Rational Rose 브라우저 스크린샷에 나타나 있습니다.

- *프리젠테이션 논리 패키지 내부에 있는 프리젠테이션 클래스*(CustomerView, OrderView 및 ProductView)
- *비즈니스 논리 패키지 내부에 있는 비즈니스 논리 클래스*(Customer, Order 및 Product)
- *데이터 액세스 논리 패키지 내에 있는 데이터 액세스 논리 클래스*(CustomerData, orderData 및 ProductData)

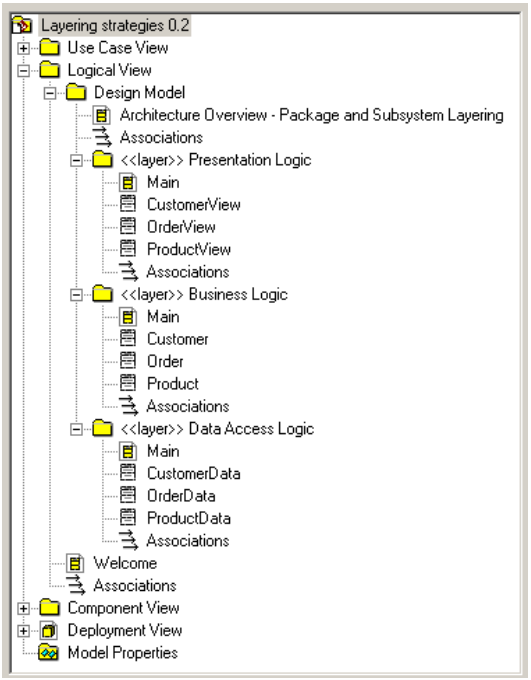


그림 5: 계층 내에 포함된 요소

두 번째 방법은 *비즈니스 컴포넌트*(이 경우, Customer, Order 및 Product)의 개념을 일급 시민으로 통합합니다. 따라서 중요한 기본 요소는 시스템에서 지원되는 도메인 관련 개념입니다. 예를 들어, *Customer* 개념에는 프리젠테이션 논리, 비즈니스 논리 및 데이터 액세스 논리의 관련 요소가 포함될 수 있습니다. 이 비즈니스 컴포넌트 개념은 [Eeles] 및 [Herzum]에서 자세히 논의됩니다. 모델 구조의 결과를 생각하는 방법은 그림 6 에 표시됩니다. 이 예에서는 계층화를 요소 이름으로 *나타냅니다*. 예를 들어, 모든 *View* 클래스(예: *CustomerView*)는 프리젠테이션 논리 계층을 나타내고, 모든 *Data* 클래스(예: *CustomerData*)는 데이터 액세스 논리 계층을 나타냅니다. 부적당한 클래스 이름(예: *Customer*)은 비즈니스 논리 계층을 나타냅니다.

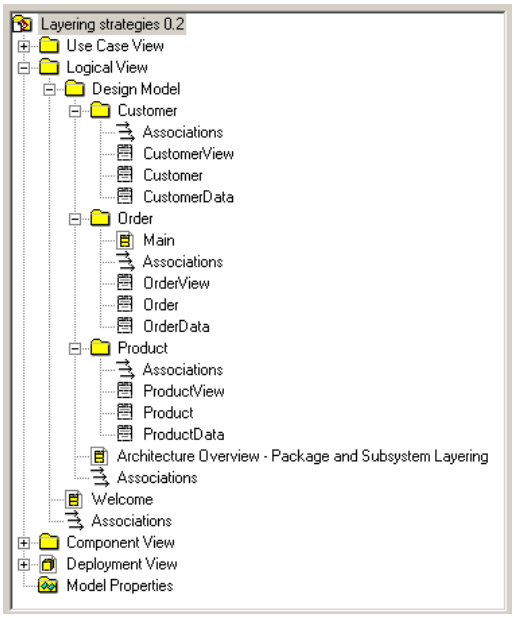


그림 6: 각 비즈니스 컴포넌트 패키지 내의 *명시적* 계층화

또한 계층화는 그림 7 에 표시된 대로 비즈니스 컴포넌트를 나타내는 각 패키지 내에 명시적으로 표시될 수 있습니다. *이 구조는 해당 비즈니스 컴포넌트의 각 계층에 다수의 요소가 포함되어 있을 때 보다 바람직합니다.* 이 예에서는 고객 비즈니스 컴포넌트 패키지만 확장되었지만 주문 및 제품 패키지도 유사한 구조를 가질 수 있습니다.

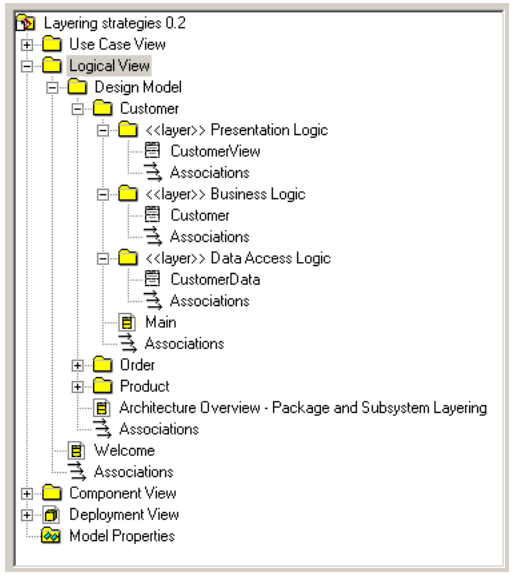
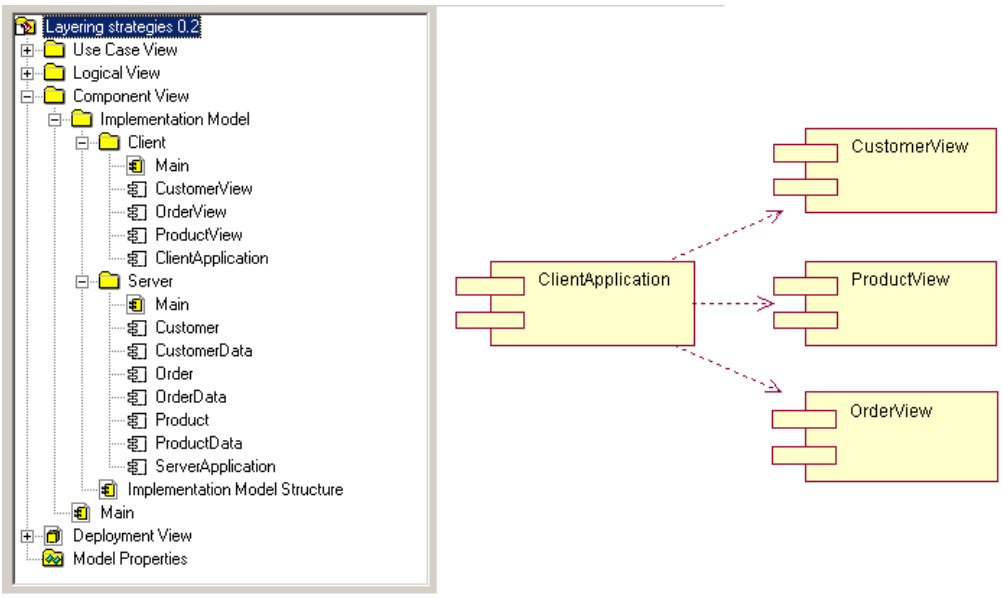


그림 7: 비즈니스 컴포넌트 패키지 내의 *명시적* 계층화

일반적으로 책임 기반 계층화 전략은 각 책임을 구현하는 요소를 물리적으로 분리해야 하는 경우 설계 모델과 더불어 *구현 모델*에 영향을 미칩니다. 예를 들어, 씬 클라이언트의 실제 분산을 나타내는 시스템을 고려하십시오. 이것은 클라이언트에서의 실행을 지원하는 데 필요한 구현 단위와 서버에서의 실행을 지원하는 데 필요한 구현 단위를 식별하기에 유용합니다. 이 예에서 *프리젠테이션 논리 계층*의 요소는 클라이언트에 전개되는

어플리케이션에 있고 *비즈니스 논리 계층* 및 *데이터 논리 계층*의 모든 요소는 서버에 전개되는 또 다른 어플리케이션에 있습니다.

이 시나리오는 그림 8 에 표시된 대로 *구현 모델*을 의미하며 Rational Rose 브라우저 이미지 및 클라이언트에 전개되는 어플리케이션의 요소를 표시하는 클라이언트 다이어그램을 나타냅니다. 이 예에서는, 설계 모델의 클래스와 구현 모델의 UML 컴포넌트 간에 1 대 1 맵핑이 발생합니다. 그러나 일반적으로 이 맵핑은 사용된 구현 설명에 따라 다름에 주의하십시오.



구현 8: 구현 모델 내의 *암시적* 계층화

유사하게, 실제 책임 분배를 설명할 필요가 있을 때 책임 기반 계층화 전략도 *전개 모델*에 영향을 미칩니다. 그림 9 및 위의 예 사용에서 6 개 노드가 정의된 것을 볼 수 있습니다. 세 가지 *클라이언트 노드* 각각은 ClientApplication 프로세스를 포함합니다. FrontEndServer 노드는 두 개의 서버 노드 중 하나에 클라이언트 요청을 분배하는 책임을 맡은 LoadBalancer 프로세스를 포함합니다. 각 *서버 노드*는 ServerApplication 프로세스를 포함합니다.

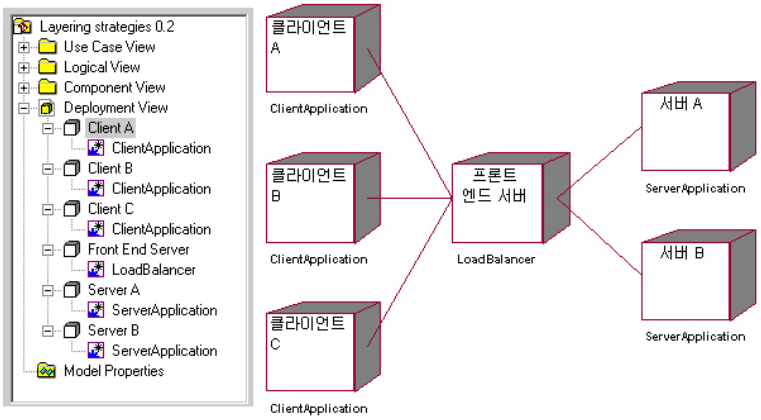


그림 9: 실제 책임 분배를 설명하는 전개 모델

재사용 기반 모델링

일반적으로 사용되는 또 다른 계층은 재사용에 기반한 것입니다. 이 전략은 특히 조직 전체에서 컴포넌트를 재사용하기 위한 식별 가능한 목표를 가진 조직과 관련되어 있습니다. 이 계층화 전략을 사용할 경우, 컴포넌트가 재사용 레벨에 따라 명시적으로 그룹화되므로 컴포넌트의 재사용 가능성이 가시화됩니다. [Jacobson]에 설명된 전략에서 발췌한 계층화 예제가 그림 10 에 표시되어 있습니다. 여기에서는 기본, 비즈니스 특정 및 어플리케이션 특정 등 세 가지 계층이 설명되어 있습니다.

- *기본 계층*은 여러 조직에 걸쳐 적용될 수 있는 요소를 포함합니다(예: 수식). 이와 같은 요소는 광범위하게 재사용됩니다.
- *비즈니스 특정 계층*은 특정 조직에 적용되지만 어플리케이션에 독립적인 요소를 포함합니다(예: 주소록). 이와 같은 요소는 동일한 조직의 어플리케이션 내에서 재사용됩니다.
- *어플리케이션 특정 계층*은 특정 어플리케이션 또는 프로젝트에 적용되는 요소를 포함합니다(예: 전자 수첩). 이런 요소는 최소한으로 재사용 가능합니다.

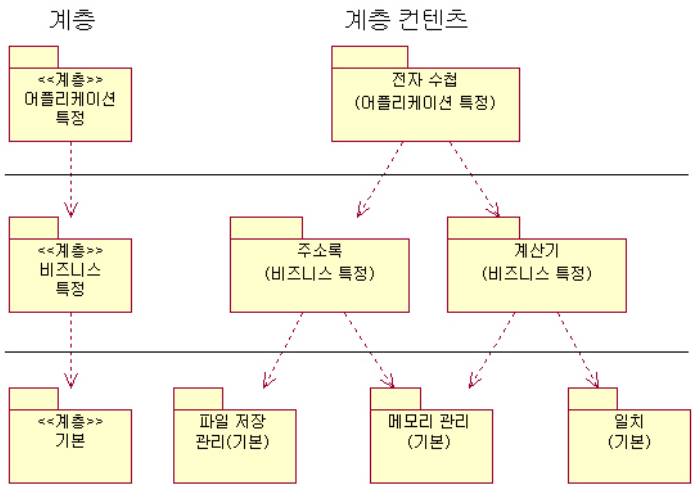


그림 10: 재사용 기반 계층화의 예

기본 계층의 요소가 가장 많이 재사용할 수 있습니다. 어플리케이션 특정 계층의 요소가 프로젝트 특정 계층의 요소보다 많으므로 적게 재사용할 것이기 때문입니다.

재사용 기반 계층 모델링

재사용 전략의 어플리케이션은 주로 *설계 모델*에 영향을 미칩니다. 재사용 기반 계층화를 통합하는 설계 모델의 구조는 파악하기 수월하며 그림 10 의 예를 반영하는 그림 11 에 표시됩니다.

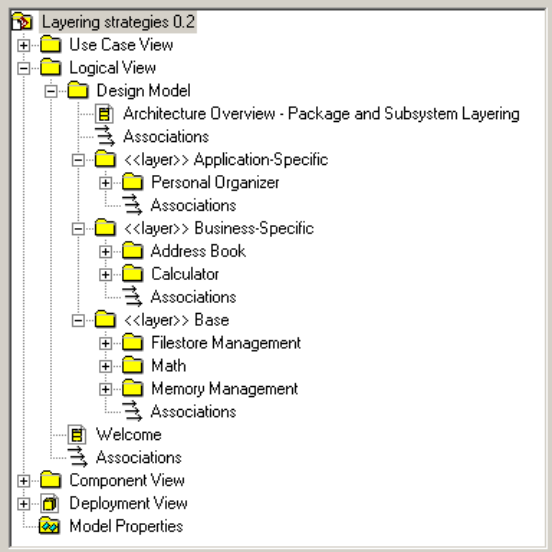


그림 11: 재사용 기반 계층을 통합하는 모델 설계

기타 계층화 전략

이 문서는 예를 들어, 광범위하게 사용되는 전략을 사용하여 존재하는 여러 계층화 전략의 " 특징" 을 제공할 것입니다. 그러나 보안, 소유권 및 설명 세트와 같은 특성을 승인하는 전략에 대해 유사한 방법을 취할 수 있었습니다.

다차원 계층화

이전에 설명된 전략은 새로운 계층화 전략을 작성하기 위해 결합되기도 합니다. 그림 12 의 예는 다음을 표시합니다.

- 이전 예의 두 가지 재사용 기반 계층
  - 어플리케이션 특정
  - 비즈니스 특정
- 세 가지 책임 기반 계층(층)
  - 프리젠테이션 논리
  - 비즈니스 논리
  - 데이터 액세스 논리

재사용 기반 계층화 전략에 있는 종속성은 일반적으로 PersonalOrganizer 및 AddressBook 간의 종속성을 볼 수 있는 그림 12 에 설명된 대로 비즈니스 논리 계층에 있는 요소 간의 종속성에서 기인합니다.

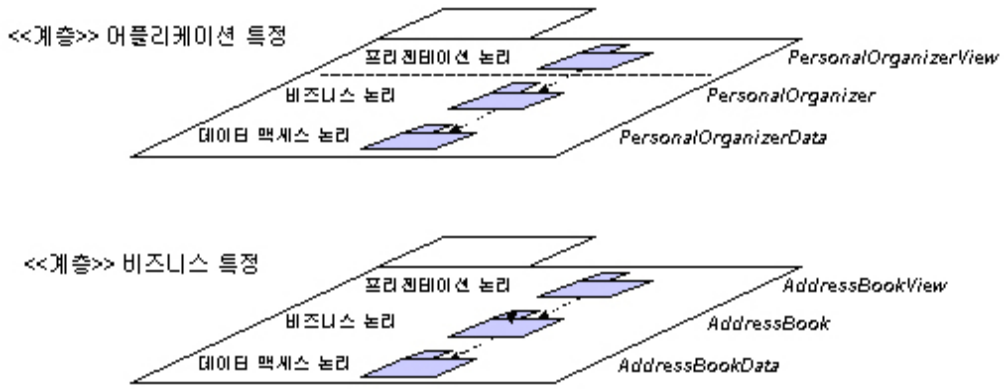


그림 12: 다차원 계층화

다차원 계층 모델링

여기서는 2 차원 *설계 모델* 내의 계층화를 다차원 측면으로 표시하는 것을 고려합니다. 또한 이를 통해 *비즈니스 컴포넌트* 개념이 통합되는 구조를 고려합니다.

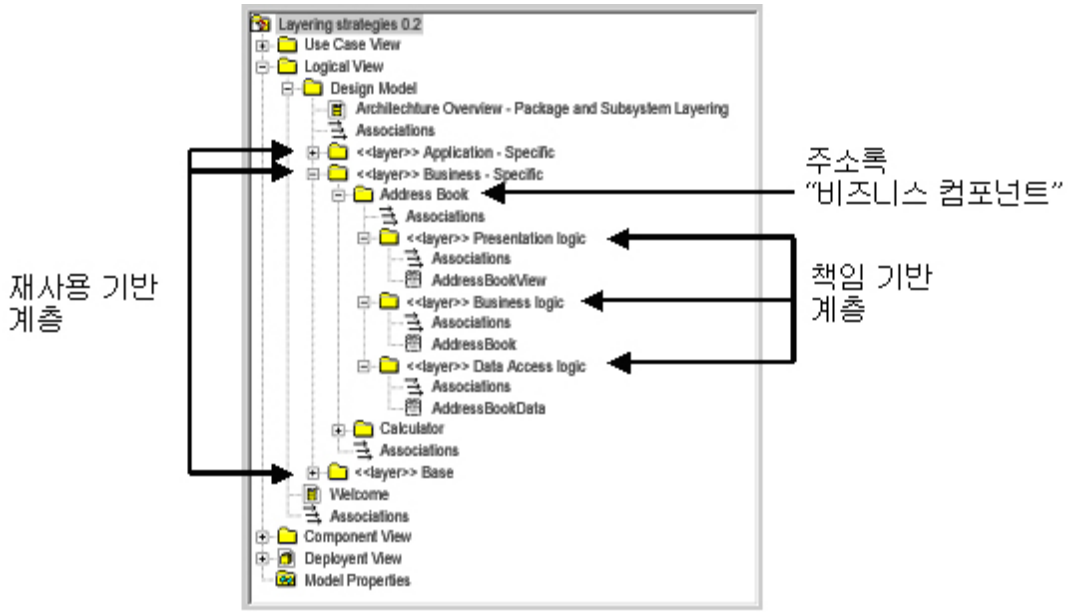


그림 13: 다차원 계층을 통합하는 모델 설계

다차원 계층화 전략을 채택하려면 기본 전략이 식별되어야 합니다. 이 예에서 기본 계층화 전략은 재사용에 기반한 것입니다. 설계 모델은 우선 이 전략을 기반으로 *어플리케이션 특정*, *비즈니스 특정* 및 *기본* 계층을 제공하여 구성됩니다. 그런 다음, 각 계층에 있는 요소를 통해 각 계층이 더 자세히 구성됩니다. 예를 들어, 그림 13 은 *주소록* 및 *계산기*를 포함하는 비즈니스 특정 계층을 표시합니다. 그런 다음, 2 차 전략을 기반으로 각 요소가 더 자세히 각 자세히 구성됩니다(책임 기반 계층화). 예를 들어, *주소록* 패키지는 *프리젠테이션 논리*, *비즈니스 논리* 및 *데이터 액세스 논리* 등 세 가지 계층을 포함합니다.

그런 다음, 이런 각 계층이 이 계층에 있는 요소를 포함합니다.

- *프리젠테이션 논리 계층*은 주소록 보기 클래스를 포함합니다.

- *비즈니스 논리 계층*은 주소록 클래스를 포함합니다.
- *데이터 액세스 논리*는 주소록 데이터 클래스를 포함합니다.

결론

아키텍트의 가장 중요한 결정 중 하나는 적절한 계층화 전략을 선택하는 것입니다. 이것은 계층화 전략이 생성되는 모델의 구조에 주요한 영향을 미치기 때문입니다. 그러나 가장 중요한 것은 유지보수성 및 재사용과 같은 비즈니스 장점이 선택된 계층화 전략에 의해 직접 지원될 수 있다는 사실입니다. 예를 들어, 시스템의 여러 책임이 책임 기반 계층화 전략의 채택을 통해 서로에게서 분리되면 유지보수성이 더욱 높은 시스템이 개발될 것입니다. 또한 재사용 시스템 요소는 재사용 기반 계층화 전략을 사용하여 명확하게 식별될 수 있습니다.

감사

저자는 이 문서의 초안에 대해 통찰력 있는 비평을 해준 Kelli Houston, Wojtek Kozaczynski, Philippe Kruchten, Bran Selic 및 Catherine Southwood(Rational Software 의 모두)의 호의에 감사드립니다.

참조서

[Buschmann] Buschmann, Frank, et al. *A System of Patterns*. 1996. New York: John Wiley & Sons. ISBN 0-471-95869-7.

[Edwards] Edwards, Jeri. *3-Tier Client/Server at Work*. 1999. New York: John Wiley & Sons. ISBN 0-471-31502-8.

[Eeles] Eeles, Peter, and Oliver Sims. *Building Business Objects*. 1998. New York: John Wiley & Sons. ISBN 0-471-19176-0.

[Herzum] Herzum, Peter, and Oliver Sims. *The Business Component Factory*. 2000. New York: John Wiley & Sons.

[Jacobson] Jacobson, Ivar, et al. *Software Reuse*. 1997. Reading, Massachusetts: Addison-Wesley. ISBN 0-201-92476-5.

[PLoP2] Vlissides, John, James Coplien, and Norman Kerth. *Pattern Languages of Program Design 2*. 1996. Reading, Massachusetts: Addison-Wesley. ISBN 0-201-89527-7.

Rational®
the software development company

본사:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
전화번호: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
전화번호: (781) 676-2400

무료 전화번호: (800) 728-1212
전자 우편: info@rational.com
웹: www.rational.com
월드와이드: www.rational.com/worldwide