계층화 전략

Peter Eeles

Rational Software 백서

TP 199, 08/01

Rational®
the software development company

목차

요약

소프트웨어 시스템을 분해하는 많은 기법이 있습니다. 계층화가 한 가지 예이고 이 문서에서 설명됩니다. 이 기법은 대부분의 시스템이 너무 복잡하여 완전하게 이해할 수 없으며 사용자에 따라 다른 시스템 관점이 필요하다는 두 가지 관심사항을 처리합니다.

계층화는 다수의 소프트웨어 시스템에서 채택되었고 많은 텍스트 및 Rational Unified Process 에서 지지되고 있습니다. 그러나 종종 계층화가 잘못 이해되거나 올바르지 않게 적용됩니다. 이 문서는 계층화의 의미를 규정하며 다양한 계층 전략 적용에 따른 영향에 대해 설명합니다.

"계층화"의 개념?

먼저 "계층화"의 정의에 대해 설명합니다. *계층*은 일반적으로 *"계층" 패턴*이라고 하는 아키텍처 패턴의 응용프로그램을 의미합니다. 계층 패턴에 대해서는 여러 텍스트([Buschmann], [Herzum], [PloP2])와 RUP 에서도 설명합니다. *패턴*은 특정 의미의 일반적인 문제점에 대한 솔루션을 나타냅니다. 표 1 은 계층 패턴에 대한 개요 정보를 제공합니다.

표 1: "계층" 패턴 개요

	계층 패턴
컨텍스트	분해가 필요한 시스템
문제점	너무 복잡하여 완전히 이해할 수 없는 시스템 유지보수가 어려운 시스템 가장 불안정한 요소가 분리되지 않은 시스템 가장 많이 재사용할 수 있는 요소를 식별하기 어려운 시스템 여러 팀에서 일반적으로 다양한 스킬로 빌드할 시스템
솔루션	시스템을 계층으로 구조화

가장 널리 알려진 계층화 예 중 하나는 ISO(International Standardization Organization)에서 정의한 OSI 7-계층 모델입니다. 이 모델(그림 1 참조)은 한 세트의 네트워크 프로토콜을 정의합니다. 각 계층은 통신의 특정 측면에 초점을 맞추며 하위 계층의 기능에 따라 빌드됩니다. OSI 7-계층 모델은 책임 기반 계층화 전략을 사용합니다. 즉, 각 계층마다 특정 책임을 갖습니다. 이 전략은 이 문서 후반부에서 자세히 설명합니다.

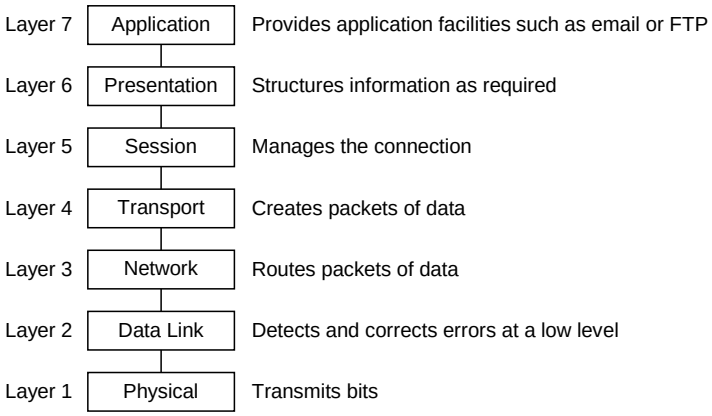

그림 1: OSI 7-계층 모델(책임 기반 계층화)

그림 2 는 책임 기반 계층화의 또 다른 예제를 보여줍니다.

- *프리젠테이션 로직 계층*에는 사용자에 대한 특정 양식의 렌더링을 제공해야 하는 요소(예: 사용자 인터페이스의 요소)가 포함됩니다.
- *비즈니스 로직 계층*에는 특정 유형의 비즈니스 처리를 수행하고 비즈니스 규칙을 적용해야 하는 요소가 포함됩니다.
- *데이터 액세스 로직 계층*에는 정보 소스(예: 관계형 데이터베이스)에 대한 액세스를 제공해야 하는 요소가 포함됩니다.

계층은 여러 가지 방법으로 모델링할 수 있으며 해당 내용은 이 문서 후반부에서 설명합니다. 여기서는 스테레오타입 "계층"의 UML 패키지를 사용하여 계층을 명시적으로 나타냅니다.

그림 2: 책임 기반 계층화

이 특정 책임 기반 계층화 예제에 표시되는 계층은 일반적으로 "층"이라고 하며 *2 층*, *3 층* 및 *n 층* 시스템을 사용하는 분산 시스템 개발에서 일반적으로 사용되는 개념입니다.

그림 2 에서 중요한 점은 표시된 *종속성의 방향*입니다. 이 방향은 계층화 시스템의 특성인 특정 규칙(특정 계층의 요소가 동일한 계층 또는 하위 계층의 요소에만 액세스할 수 있음)을 나타냅니다.[1]. 즉, 여기 제시된 예제의 경우 *비즈니스 로직 계층*의 요소는 *프리젠테이션 로직 계층의* 요소에 액세스할 수 없습니다. 또한 *데이터 액세스 로직 계층*의 요소는 *비즈니스 로직 계층*의 요소에 액세스할 수 없습니다. 이러한 구조를 일반적으로 방향 그래프(DAG)라고 합니다. 이 구조에서는 종속성의 *방향*이 일방향이며 종속성의 경로가 순환 경로가 아닌*비순환* 경로입니다.

즉, 계층화 전략을 정의할 때 각 계층의 의미를 정확하게 정의해야 요소가 해당 계층에 올바르게 배치될 수 있습니다. 요소를 해당 계층에 올바르게 지정하지 않으면 이 전략을 우선적으로 적용하는 데 따른 가치가 줄어듭니다. 각 계층화 전략을 자세히 설명함에 따라 각 계층의 의미에 대한 일반 안내가 제공됩니다.

계층 모델링

다양한 계층화 전략을 조사하면서 특정 *모델*(및 그에 따른 특정 UML 요소)을 사용하여 각 전략을 설명하는 것이 바람직함을 알 수 있습니다. *모델*은 시스템에 대한 완벽한 설명을 특정 관점에서 나타냅니다. 그림 3 은 고려 대상 시스템의 다른 관점을 나타내는 네 가지 모델 예제를 보여줍니다.

- 유스 케이스 모델: 시스템 요구사항을 캡처합니다.
- 분석 모델: 시스템 요구사항 분석을 캡처합니다.
- 디자인 모델: 시스템 디자인을 캡처합니다.
- 구현 모델: 시스템 구현을 캡처합니다.

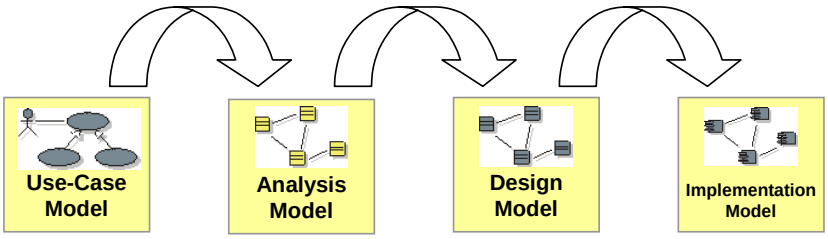

그림 3: 점진적인 정제를 나타내는 네 가지 모델

추가 모델은 다음과 같습니다.

- *배치 모델:* 시스템의 분포 측면을 캡처합니다.
- *데이터 모델:* 시스템의 지속적 측면을 캡처합니다.

계층화 전략

계층화는 여러 가지 특성을 기반으로 할 수 있습니다. 이 섹션에서는 다음 특성을 기반으로 한 계층화에 대해 설명합니다.

---

[1] 이벤트 알림으로 특정 계층의 요소에서 상위 계층의 요소로 메시지가 전달될 수 있지만 이 방향의 명시적인 종속성은 존재하지 않습니다.

- 책임
- 재사용

각 전략에 대한 자세한 논의를 통해 전략에 대해 설명합니다.

책임 기반 계층화

가장 일반적으로 사용되는 계층화 전략은 책임을 기반으로 하는 계층화 전략입니다. 이 특정 전략의 경우 다양한 시스템 책임이 서로 분리되므로 시스템의 개발 및 유지보수 기능을 개선할 수 있습니다. 예를 들어, 다음 책임을 기반으로 시스템 계층화를 수행할 수 있습니다(그림 2 참조).

- 프리젠테이션 로직
- 비즈니스 로직
- 데이터 액세스 로직

이러한 책임은 그림 4 와 같이 각각 하나의 계층으로 나타낼 수 있습니다. 이 그림에서는 각 계층에 대한 일부 샘플 컨텐츠가 표시됩니다. 여기서는 주문 처리 시스템의 세 가지 개념(고객, 주문 및 제품)을 고려합니다. 예를 들어, *고객* 개념은 다음과 같이 구성됩니다.

- *CustomerView 클래스:* 고객과 연관된 *프리젠테이션 로직*(예: 사용자 인터페이스의 고객 렌더링)을 담당합니다.
- *Customer 클래스*: 고객과 연관된 *비즈니스 로직*(예: 고객 세부사항의 유효성 검증)을 담당합니다.
- *CustomerData 클래스*: 고객과 연관된 *데이터 액세스 로직*(예: 고객 상태를 지속적 상태로 작성)을 담당합니다.

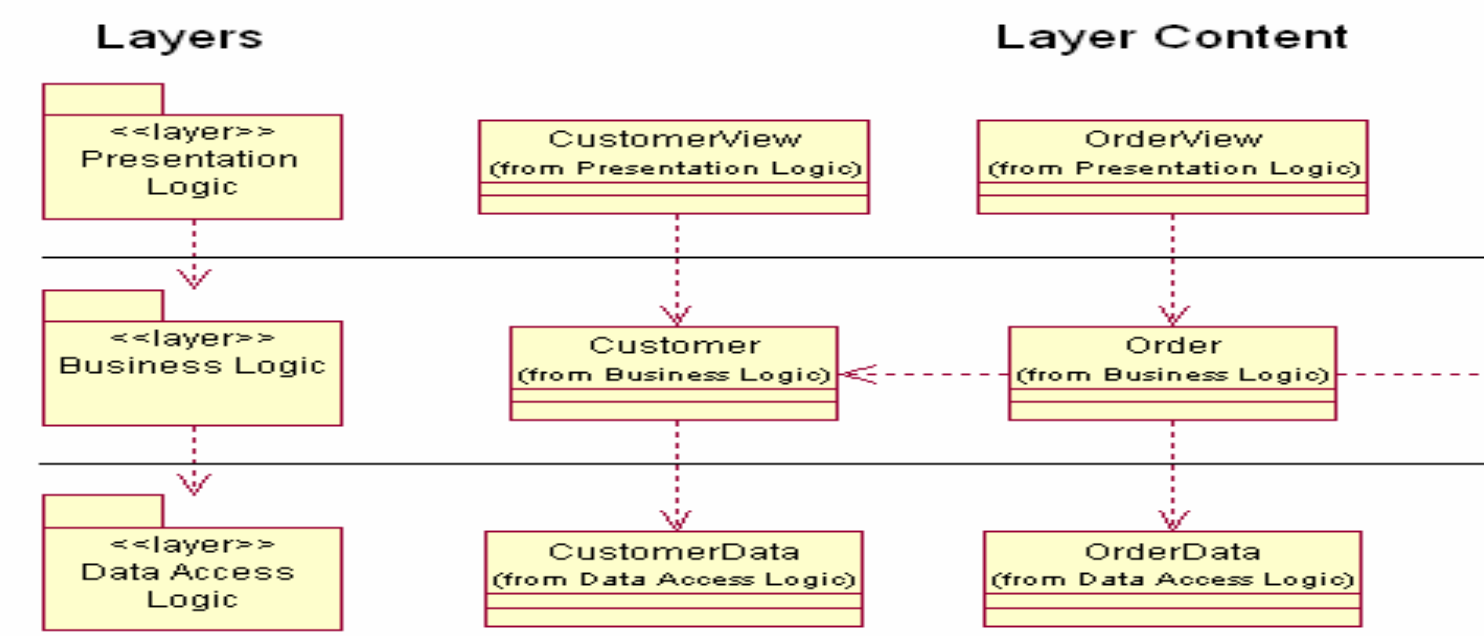

그림 4: 책임 기반 계층화의 컨텐츠 및 계층

다음은 이 특정 계층화 전략에 대한 몇 가지 "오해"에 대한 설명입니다.

오해 1: 계층(layer)과 층(tier)은 다르다.

이 오해는 가장 많은 혼란을 일으키는 원인으로서 실제로는 층이 *곧* 계층입니다. 그러나 계층은 특정 책임 전략을 기반으로 합니다. 이러한 혼란은 층의 개념을 표 2 와 같이 여러 가지 방식으로 적용할 수 있다는 점에서 비롯됩니다.

표 2: 층 정의

응용프로그램	계층(층)
2 층	결합된 프리젠테이션 로직 및 비즈니스 로직 데이터 액세스 로직
3 층	프리젠테이션 로직 비즈니스 로직 데이터 액세스 로직
n 층	프리젠테이션 로직 비즈니스 로직(*분포*) 데이터 액세스 로직

오해 2: 계층(층)은 실제 분포를 나타낸다.

일반적인 또 하나의 오해는 논리 계층화가 실제 분포를 나타낸다는 것입니다. 예를 들어, 3 층 계층화를 고려할 수 있습니다. 하나의 계층에 다양한 요소가 상주하지만 각 계층 자체는 표 3 과 같이 여러 가지 방법으로 적용할 수 있습니다. 이 표에서는 특정 실제 분포의 특성을 나타내는 데 자주 사용되는 이름을 사용합니다(예: "Thin 클라이언트").

표 3: 3 층 계층화의 응용프로그램

응용프로그램	계층	
	클라이언트측	서버측
단일 시스템	프리젠테이션 로직 비즈니스 로직 데이터 액세스 로직	
Thin 클라이언트	프리젠테이션 로직	비즈니스 로직 데이터 액세스 로직
Fat 클라이언트	프리젠테이션 로직 비즈니스 로직	데이터 액세스 로직

또한 단일 시스템이 여러 가지 실제 분포 전략을 채택할 수 있습니다. 이러한 경우 특정 요소가 "Thin 클라이언트" 분포 요약으로 분류되고 다른 요소는 "Fat 클라이언트" 분포로 분류됩니다. 일반적으로는 성능과 같은 비기능적 요구사항을 기반으로 선택합니다.

책임 기반 계층 모델링

실제로, 이 전략을 적용하면 *디자인 모델*, *구현 모델 및 배치 모델*에 영향을 줍니다. *디자인 모델*은 일반적으로 두 가지 접근 방식 중 하나를 사용하여 구조화됩니다.

첫 번째 접근 방식에서는 요소가 계층에 "포함"됩니다. 해당 결과는 그림 5(Rational Rose 브라우저 스크린샷)에 다음과 같이 표시됩니다.

- *프리젠테이션 로직 패키지에 상주하는 프리젠테이션 클래스*(CustomerView, OrderView 및 ProductView)
- *비즈니스 로직 패키지에 상주하는 비즈니스 로직 클래스*(Customer, Order 및 Product)
- *데이터 액세스 로직 패키지에 상주하는 데이터 액세스 로직 클래스*((CustomerData, OrderData 및 ProductData)

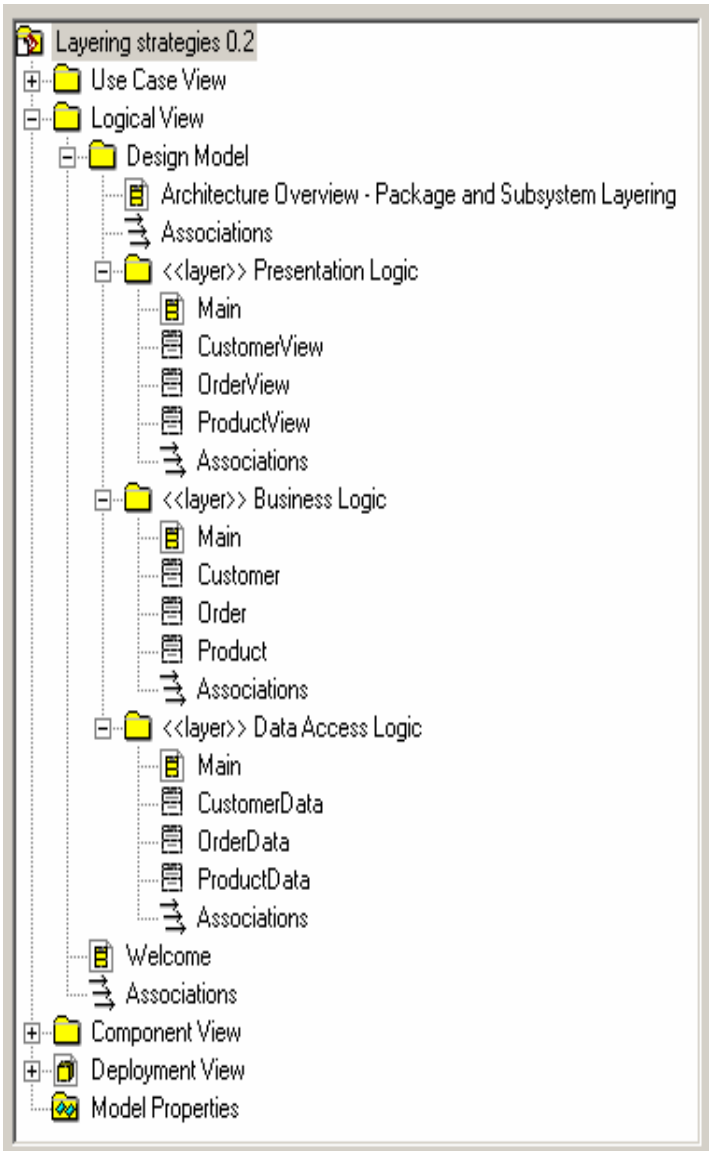

그림 5: 계층에 포함된 요소

두 번째 접근 방식은 *비즈니스 컴포넌트*(이 경우 고객, 주문 및 제품)의 개념을 일등석 고객으로서 통합합니다. 여기서, 기본 관심사항 요소는 시스템에서 지원하는 도메인 관련 개념입니다. 예를 들어, *고객* 개념은 프리젠테이션 로직, 비즈니스 로직 및 데이터 액세스 로직의 요소와 연관됩니다. 비즈니스 컴포넌트의 이 개념은 [Eeles] 및 [Herzum]에서 자세히 설명합니다. 이러한 접근 방식을 적용하면 그림 6 과 같은 모델 구조가 생성됩니다. 이 예제에서 계층화는 요소 이름으로 *나타냅니다*. 예를 들어, 모든 *View* 클래스(예: *CustomerView*)는 프리젠테이션 로직 계층을 나타내고 모든 *Data* 클래스(예: *CustomerData*)는 데이터 액세스 로직 계층을 나타냅니다. 규정되지 않은 클래스 이름(예: *Customer*)은 비즈니스 로직 계층을 나타냅니다.

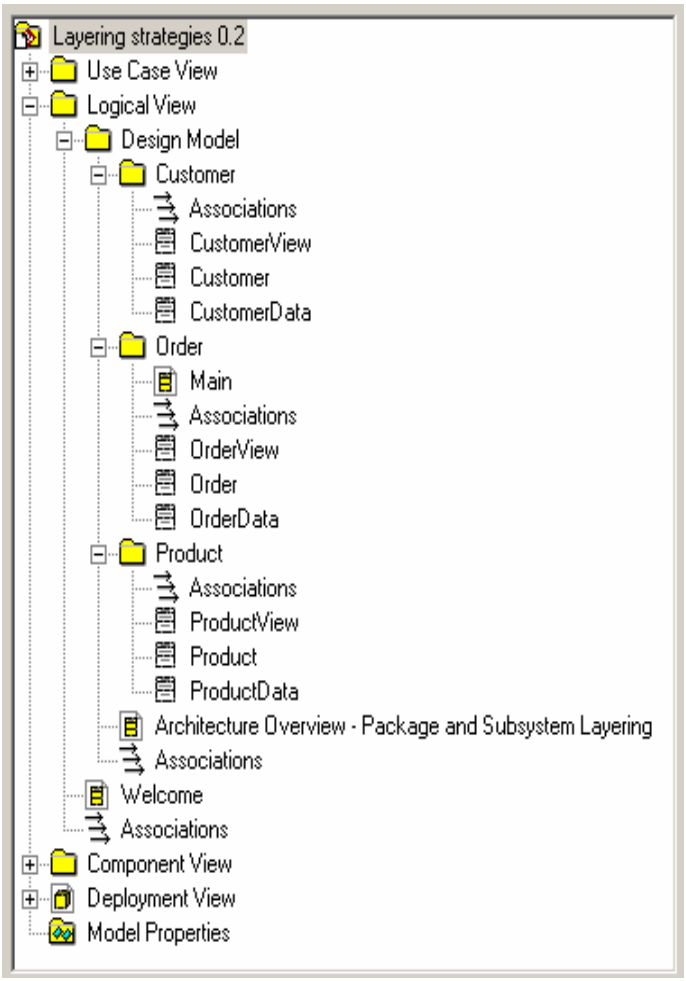

그림 6: *내재적*인 각 비즈니스 컴포넌트 패키지 내 계층화

계층화는 그림 7 과 같이 비즈니스 컴포넌트를 나타내는 각 패키지 내에 명시적으로 나타낼 수도 있습니다. *이 구조화는 특히 해당 비즈니스 컴포넌트의 각 계층과 여러 요소가 관련된 경우에 바람직한 방법입니다*. 이 예제에서는 고객 비즈니스 컴포넌트 패키지만 펼쳐져 있지만 주문 및 제품 패키지도 유사한 구조를 갖습니다.

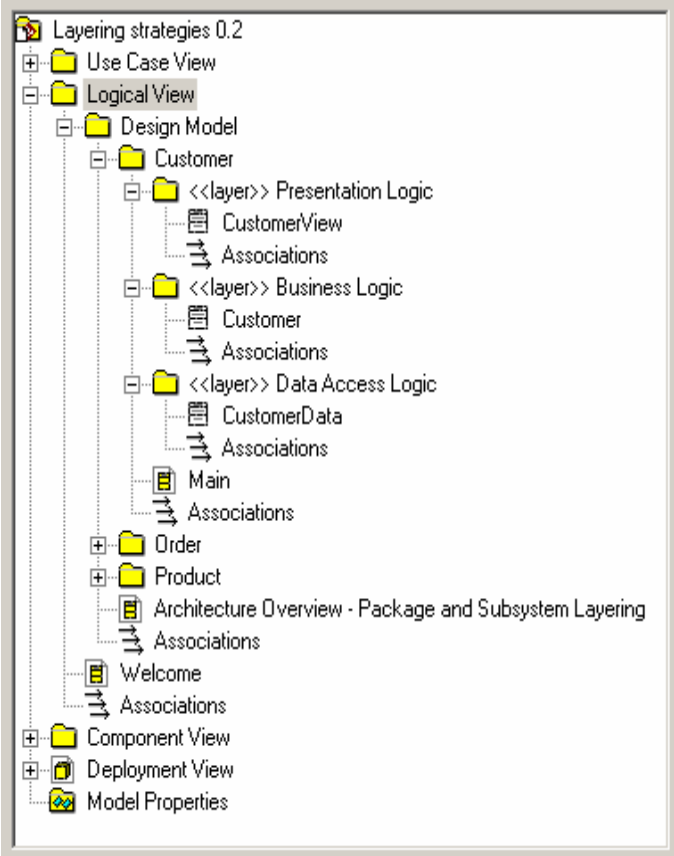

그림 7: *명시적*인 비즈니스 컴포넌트 패키지 내 계층화

책임 기반 계층화 전략은 각 책임을 구현하는 요소를 실제로 분할해야 하는 경우 일반적으로 *구현 모델*과 디자인 모델에 영향을 줍니다. 예를 들어, "Thin 클라이언트" 실제 분포를 나타내는 시스템을 고려할 수 있습니다. 이러한 경우 클라이언트에서의 실행을 지원하는 데 필요한 구현 단위와 서버에서의 실행을 지원하는 데 필요한 구현 단위를 식별하는 것이 좋습니다. 이 예제에서 *프리젠테이션 로직 계층*의 요소는 클라이언트에 배치되는 응용프로그램에 상주하므로 *비즈니스 로직 계층* 및 *데이터 로직 계층*의 모든 요소는 서버에 배치되는 다른 응용프로그램에 상주합니다.

이 시나리오는 *구현 모델*(그림 8 참조)을 나타냅니다. 이 그림에는 클라이언트에 배치되는 응용프로그램 요소를 표시하는 컴포넌트 다이어그램과 Rational Rose 브라우저 이미지가 표시됩니다. 이 예제에서는 디자인 모델의 클래스와 구현 모델의 UML 컴포넌트 간에 일 대 일 맵핑이 존재할 수 있습니다. 그러나 이 맵핑은 일반적으로 사용된 구현 기술에 따라 결정됩니다.

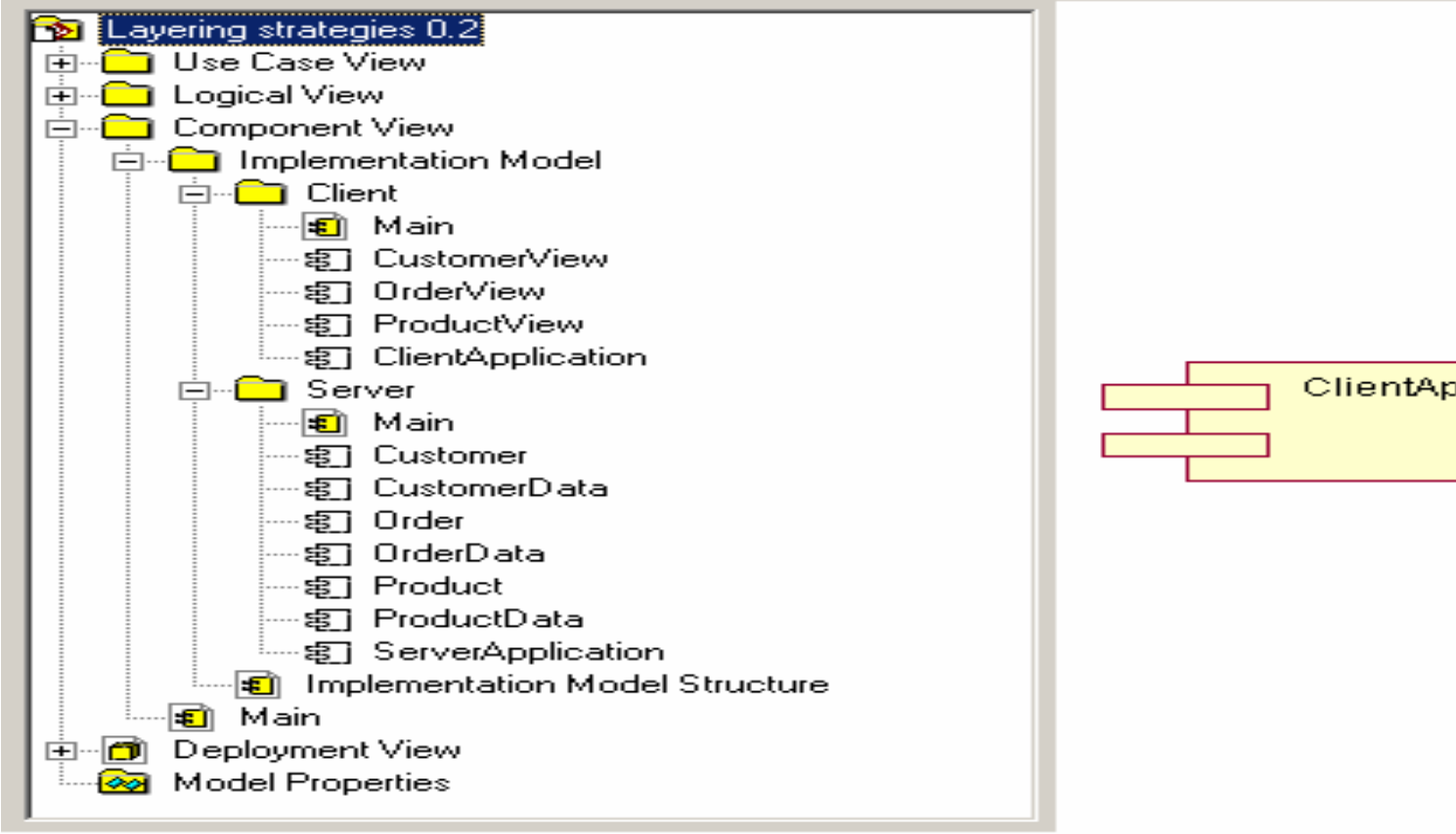

그림 8: *내재적인* **구현 모델 내 계층화**

마찬가지로, 실제 책임 분포를 설명해야 하는 경우 책임 기반 계층화 전략 또한 *배치 모델*에 영향을 줄 수 있습니다. 그림 9 와 위의 예제를 참조하면 여섯 개의 노드가 정의된 것을 알 수 있습니다. 세 개의 *클라이언트 노드* 각각에는 ClientApplication 프로세스가 포함됩니다. FrontEndServer 노드에는 두 개의 서버 노드 중 하나에 클라이언트 요청을 분배하는 LoadBalancer 프로세스가 포함됩니다. 각 *서버 노드*에는 ServerApplication 프로세스가 포함됩니다.

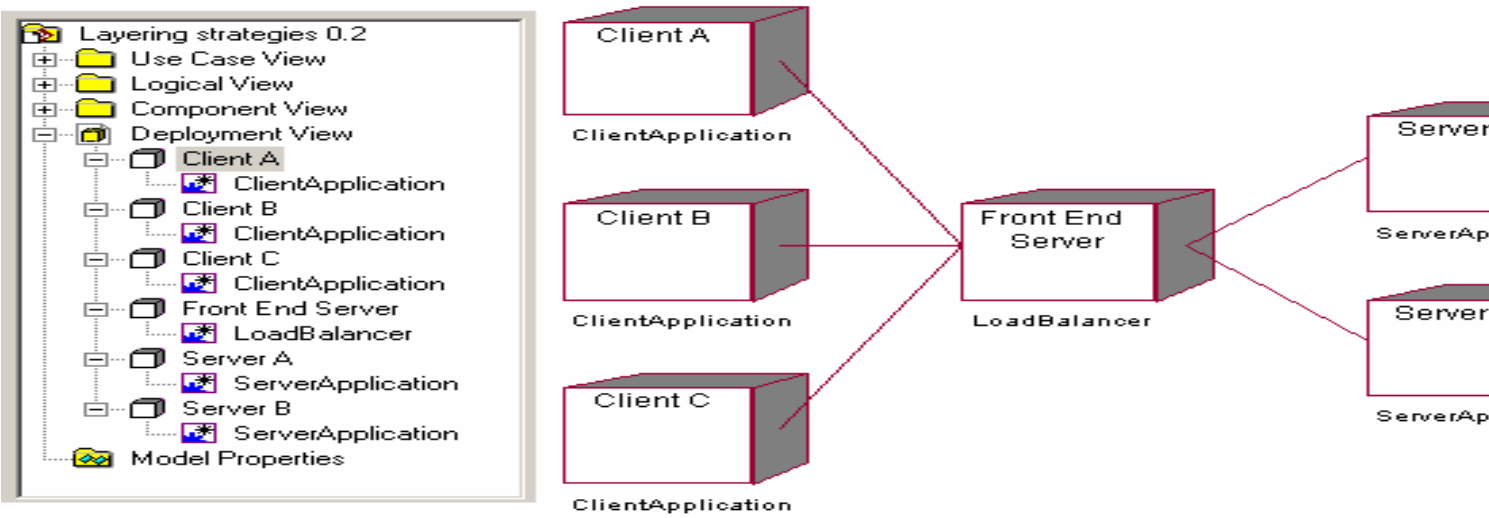

그림 9: 실제 책임 분포를 설명하는 배치 모델

재사용 기반 모델링

일반적으로 사용되는 또 다른 계층화 전략은 재사용을 기반으로 하는 계층화 전략입니다. 이 전략은 특히 조직 전체에서 컴포넌트를 재사용하려는 뚜렷한 목적을 갖고 있는 조직과 관련이 있습니다. 이러한 계층화 전략을 사용하는 경우 컴포넌트가 해당 재사용 레벨에 따라 명시적으로 구분되므로 컴포넌트의 재사용가능 상태를 시각적으로 확인할 수 있습니다. 그림 10 은 [Jacobson]에서 설명한 전략에서 파생된 계층화 예제를 보여줍니다. 이 그림에는 세 가지 계층(기본, 비즈니스 특정 및 응용프로그램 특정)이 표시됩니다.

- *기본 계층*에는 여러 조직에 적용할 수 있는 요소(예: 수학)가 포함되며 이러한 요소는 다방면으로 재사용됩니다.
- *비즈니스 특정 계층*에는 특정 조직에 적용되지만 응용프로그램과 무관한 요소(예: 주소록)가 포함됩니다. 이러한 요소는 동일한 조직의 여러 응용프로그램에서 재사용됩니다.
- *응용프로그램 특정 계층*에는 특정 응용프로그램 또는 프로젝트에 적용되는 요소(예: 개인 전자수첩)가 포함됩니다. 이러한 요소는 재사용율이 가장 낮습니다.

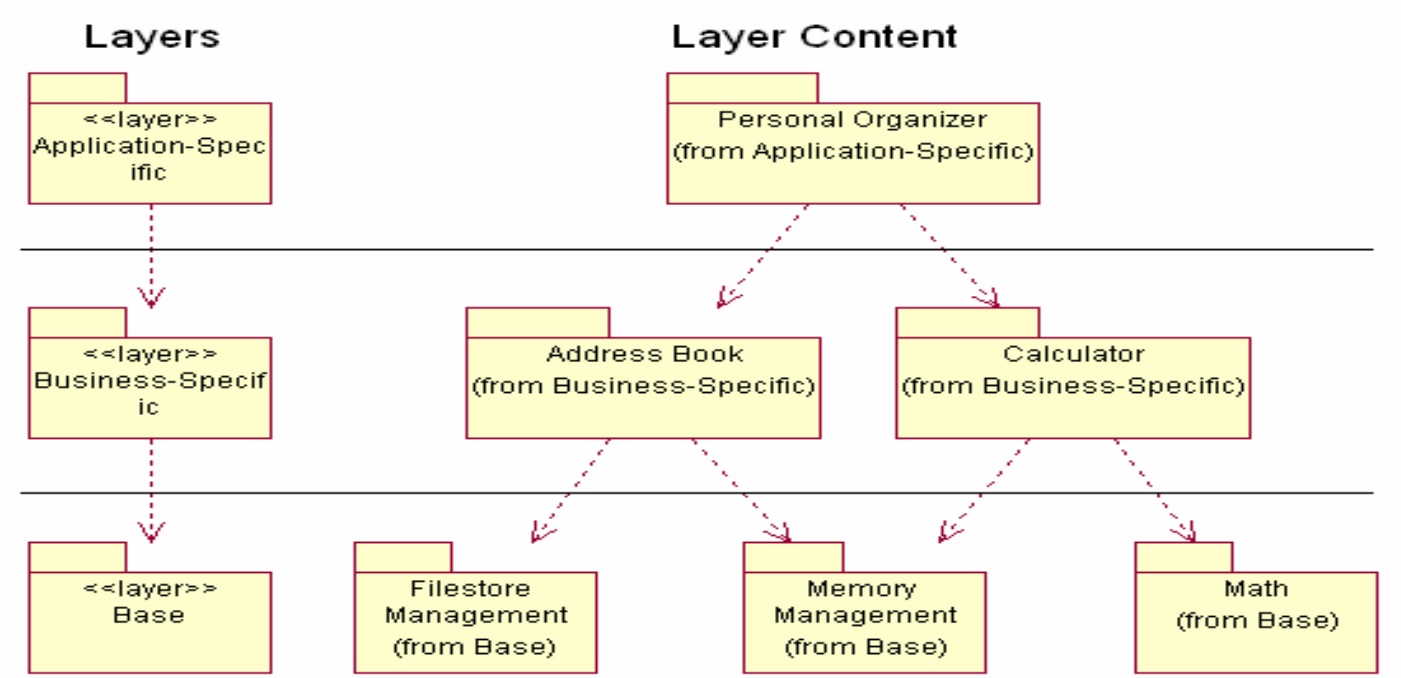

그림 10: 재사용 기반 계층화 예제

위의 그림을 통해 기본 계층 요소의 재사용율이 가장 높은 반면 응용프로그램 특정 계층 요소의 경우 프로젝트에 대한 의존도가 높아 재사용율이 낮음을 알 수 있습니다.

재사용 기반 계층 모델링

재사용 전략 응용프로그램은 일반적으로 *디자인 모델*에 영향을 줍니다. 재사용 기반 계층화를 통합하는 디자인 모델의 구조는 그림 11 과 같이 쉽게 파악할 수 있습니다. 이 그림에는 그림 10 의 예제가 적용됩니다.

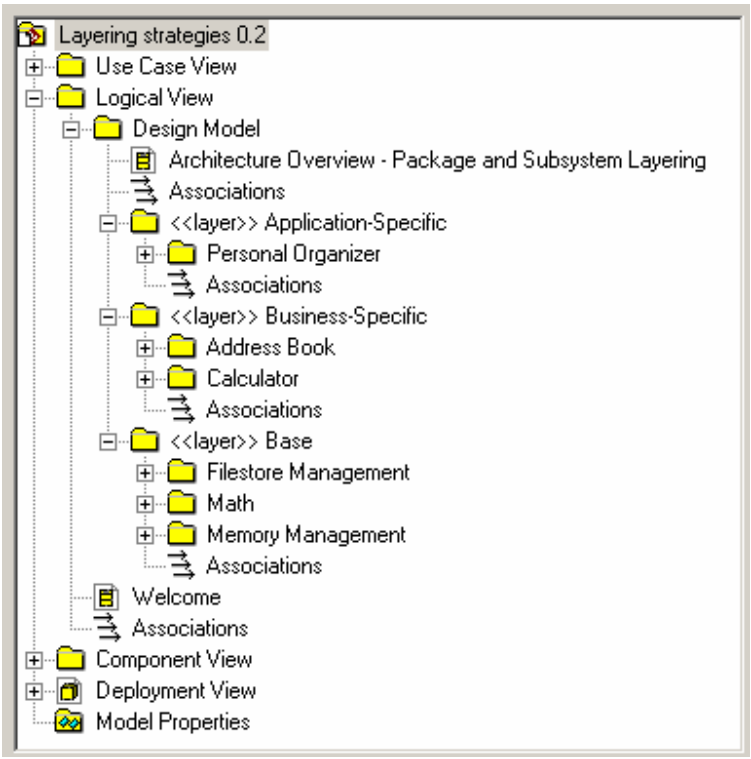

그림 11: 재사용 기반 계층화를 통합하는 디자인 모델

기타 계층화 전략

이 문서는 가장 널리 사용하는 전략을 예제로 사용하여 현재 존재하는 다양한 계층화 전략의 "개념" 정도만 설명하기 위해 작성되었습니다. 그러나 보안, 소유권 및 스킬 세트와 같은 특성을 인정하는 전략에도 유사한 접근 방식을 적용할 수 있습니다.

다차원 계층화

앞에서 설명한 전략을 새 계층화 전략에 결합할 수도 있습니다. 그림 12 의 예제에는 다음과 같은 계층이 표시됩니다.

- 이전 예제의 재사용 기반 계층 두 개
  - 응용프로그램 특정
  - 비즈니스 특정
- 세 개의 책임 기반 계층(층)
  - 프리젠테이션 로직
  - 비즈니스 로직
  - 데이터 액세스 로직

재사용 기반 계층화 전략에 나타나는 종속성은 일반적으로 그림 12 와 같이 비즈니스 로직 계층의 요소 간의 종속성에서 비롯된 것입니다. 이 그림에서는 개인 전자수첩과 주소록 간에 종속성이 존재합니다.

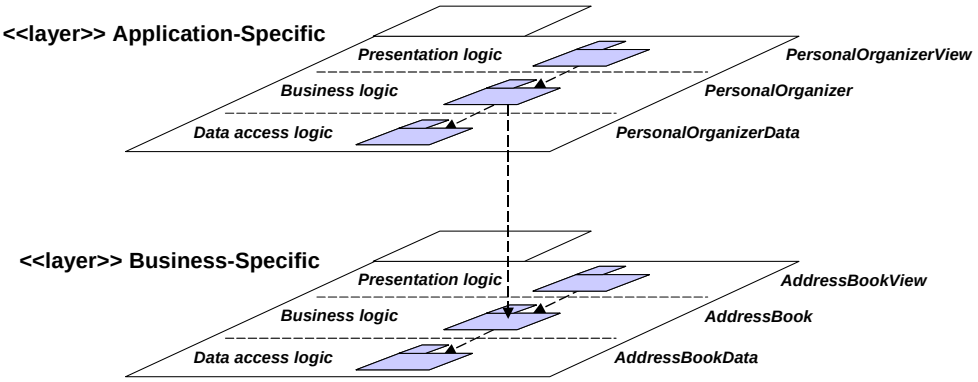

그림 12: 다차원 계층화

다차원 계층 모델링

다음은 2 차원 *디자인 모델*에서 계층화의 다차원적 측면의 표현과, *비즈니스 컴포넌트* 개념이 통합되는 구조에 대한 설명입니다.

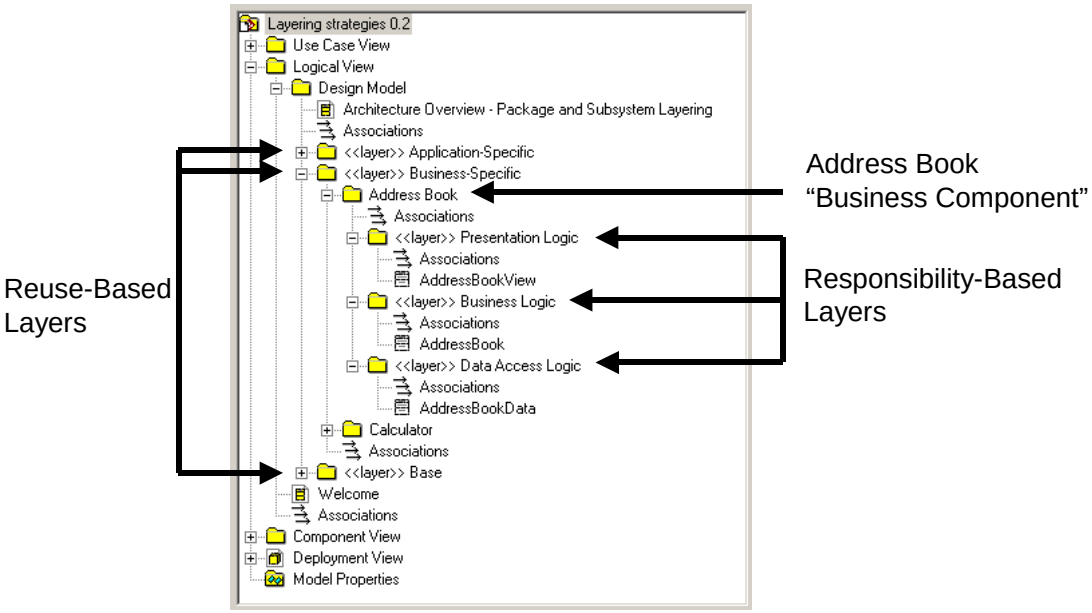

그림 13: 다차원 계층화를 통합하는 디자인 모델

다차원 계층화 전략을 채택하려면 기본 전략을 식별해야 합니다. 이 예제의 기본 계층화 전략이 재사용 기반의 계층화 전략입니다. 디자인 모델은 먼저 이 전략을 기반으로 구성되며 *응용프로그램 특정*, *비즈니스 특정* 및 *기본* 계층이 지정됩니다. 이러한 계층은 각각 해당 계층에 상주하는 요소로 구성됩니다. 예를 들어, 그림 13 에는 *주소록* 및 *계산기*를 포함하는 비즈니스 특정 계층이 표시됩니다. 이 요소는 각각 보조 전략인 책임 기반 계층화에 따라 구성됩니다. 예를 들어, *주소록* 패키지에는 세 가지 계층인 *프리젠테이션 로직*, *비즈니스 로직* 및 *데이터 액세스 로직* 계층이 포함됩니다.

이 계층은 각각 다음 계층에 상주하는 요소를 포함하게 됩니다.

- *프리젠테이션 로직 계층에는* AddressBookView 클래스가 포함됩니다.

- *비즈니스 로직 계층에는* AddressBook 클래스가 포함됩니다.
- *데이터 액세스 로직 계층에는* AddressBookData 클래스가 포함됩니다.

결론

설계자가 수행해야 하는 가장 중요한 결정은 올바른 계층화 전략을 선택하는 것입니다. 이는 전략에 따라 생성된 모델의 구조에 큰 영향을 주기 때문입니다. 그러나 무엇보다 선택한 계층화 전략에서 유지보수성 및 재사용과 같은 비즈니스 이점을 직접 지원할 수 있어야 합니다. 예를 들어, 책임 기반 계층화 전략을 채택함으로써 시스템 책임을 서로 분리할 수 있는 경우 유지보수가 용이한 시스템을 개발할 가능성이 더 높습니다. 또한 재사용 기반 계층화 전략을 사용하여 재사용가능한 시스템 요소를 명확히 식별할 수 있습니다.

감사의 글

이 문서 초본을 감수해주신 Rational Software 의 Kelli Houston, Wojtek Kozaczynski, Philippe Kruchten, Bran Selic 및 Catherine Southwood 의 노고에 감사드립니다.

참고 문헌

[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 안내:

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

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

수신자 부담 전화번호: (800) 728-1212
전자 우편: info@rational.com
웹: www.rational.com
전세계 지사 안내: www.rational.com/worldwide