Estratégias de Camadas

Peter Eeles

Rational Software White Paper

TP 199, 08/01

Rational®
the software development company

Índice Analítico

Resumo

Existem várias técnicas para decompor sistemas de software. A divisão em camadas é uma exemplo e é descrita neste documento. Essas técnicas abordam duas preocupações principais: a maioria dos sistemas é complexa demais para compreensão em sua totalidade e as perspectivas diferentes de um sistema que são requeridas para públicos diferentes.

A divisão em camadas foi adotada em vários sistemas de software e está exposta em muitos textos e também no RUP (Rational Unified Process). No entanto, a divisão em camadas é freqüentemente mal interpretada e aplicada incorretamente. Este documento esclarece o que a divisão em camadas significa e discute o impacto da aplicação de diferentes estratégias de camadas.

O que é a "Divisão em Camadas?"?

Vamos começar definido o que significam as "camadas." O termo *camada* refere-se à aplicação de um padrão arquitetural geralmente conhecido como o padrão *"Camadas"*, que é descrito em vários textos ([Buschmann], [Herzum], [PloP2]), e também no RUP. Um *padrão* representa uma solução para um problema comum que existe em um contexto específico. Uma visão geral do padrão Camadas é fornecida na Tabela 1.

Tabela 1: Visão Geral do Padrão "Camadas"

	Padrão Camadas
Contexto	Um sistema que requer decomposição
Problema	Um sistema que é muito complexo para compreensão em sua totalidade Um sistema que é difícil de manter Um sistema cujos elementos não estão isolados Um sistema cujos elementos mais reutilizáveis são difíceis de identificar Um sistema que deve ser construído por equipes diferentes, possivelmente com habilidades diferentes
Solução	Estruturar o sistema em camadas

Um dos exemplos mais familiares da divisão em camadas é o modelo de camada OSI 7, definido pelo ISO (International Standardization Organization). Esse modelo, mostrado na Figura 1, define um conjunto de protocolos de rede — cada camada enfatiza um aspecto específico de comunicação e constrói os recursos da camada abaixo dela. O modelo de camada OSI 7 utiliza uma estratégia de camadas com base em responsabilidade: cada camada tem uma responsabilidade específica. Essa estratégia é descrita detalhadamente neste documento.

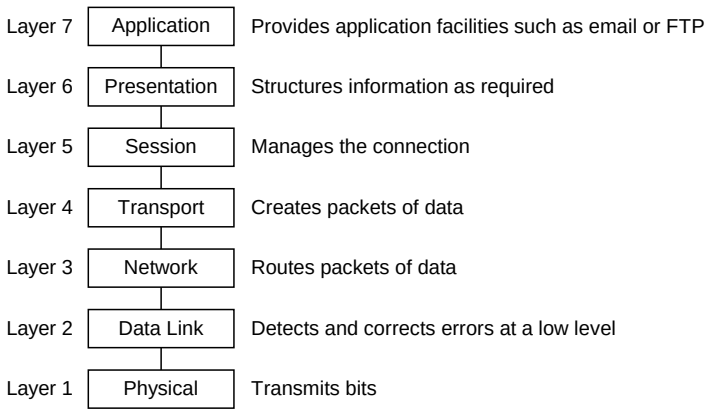

Figura 1: Modelo de Camada OSI 7 (Camada com Base em Responsabilidade)

Figura 2 mostra outro exemplo de camada com base em responsabilidade.

- A *camada Lógica de Apresentação* contém os elementos responsáveis pelo fornecimento de alguma forma de exibição para um ser humano, como um elemento na interface com o usuário.
- A *camada Lógica de Negócios* contém os elementos responsáveis pela execução de algum tipo de processamento de negócios e pela aplicação das regras de negócios.
- A *camada Lógica de Acesso aos Dados* contém os elementos responsáveis pelo fornecimento de acesso a uma origem de informações, como um banco de dados relacional.

Deveria ser observado que as camadas podem ser modeladas de várias maneiras, conforme descrito posteriormente neste documento. Agora, representaremos explicitamente uma camada utilizando um pacote UML com o estereótipo «camada».

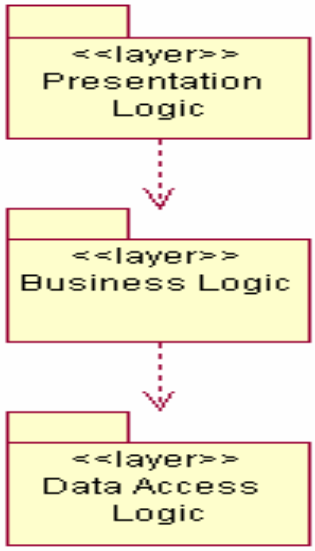

Figura 2: Camada com base em responsabilidade

As camadas mostradas neste exemplo específico de camada com base em responsabilidade são freqüentemente chamadas de "tiers" e são um conceito conhecido no desenvolvimento de sistemas distribuídos em que os sistemas *2 níveis*, *3 níveis*, e *n-tier* são encontrados.

Um aspecto importante da Figura 2 é a *direção das dependências* mostradas, já que indica uma determinada regra que é uma característica dos sistemas em camadas — um elemento em uma camada específica pode acessar somente elementos na mesma camada ou nas camadas abaixo dela.[1]. No exemplo fornecido aqui, os elementos na *camada Lógica de Negócios* não podem acessar os elementos na *camada Lógica de Apresentação*. Além disso, os elementos na *camada Lógica de Acesso aos Dados* não podem acessar elementos na *camada Lógica de Negócios*. Essa estrutura é freqüentemente utilizada como referência para um DAG (Direct Acyclic Graph). É *dirigida* de modo que as dependências sejam unidirecionais e *acíclicas* para que um caminho de dependências nunca seja circular.

[1] Embora uma notificasção de evento possa resultar em uma mensagem de um elemento em uma camada sendo enviada para um elemento em uma camada superior, não existe nenhuma dependência explícita nesta direção.

Em uma observação mais específica, é importante ser preciso em relação ao significado de cada camada ao definir uma estratégia de camada para que os elementos sejam colocados corretamente na camada apropriada. A falha para designar corretamente um elemento para a camada apropriada diminuirá o valor da aplicação da estratégia. Já que cada estratégia de camadas é discutida mais detalhadamente, algumas diretrizes gerais são dadas sobre o significado de cada uma das camadas.

Modelagem de Camadas

Enquanto investigamos diferentes estratégias de camadas, ficará claro que é apropriado comunicar cada estratégia utilizando *modelos* específicos (e, portanto, elementos UML específicos). Um *modelo* representa uma descrição completa de um sistema a partir de uma perspectiva específica. A Figura 3 mostra um exemplo de quatro modelos que representam diferentes perspectivas do sistema em consideração:

- Modelo de Caso de Uso: captura os requisitos do sistema
- Modelo de Análise: captura a análise de requisitos do sistema
- Modelo de Design: captura o design do sistema
- Modelo de Implementação: captura a implementação do sistema

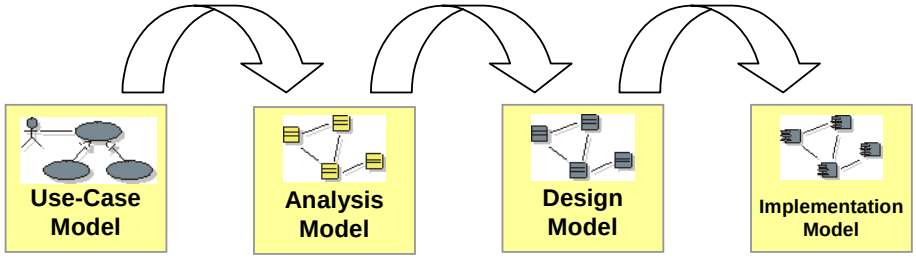

Figura 3: Quatro Modelos Representando Refinamento Gradual

Os modelos adicionais incluem:

- *Modelo de Implementação:* captura os aspectos de distribuição de um sistema
- *Modelo de Dados:* captura os aspectos persistentes de um sistema

Estratégias de Camadas

A divisão em camadas pode ter como base várias características. Essa seção discute a divisão em camadas com base nas seguintes características:

- responsabilidade
- reutilização

A representação de cada estratégia será considerada na medida em que cada estratégia for discutida em detalhes.

Camadas com base em Responsabilidade

Provavelmente a estratégia de camadas utilizadas mais freqüentemente é uma com base na responsabilidade. Essa estratégia específica pode aprimorar o desenvolvimento e a manutenção de um sistema em que várias responsabilidades do sistema estão isoladas umas das outras. Como exemplo (consulte a Figura 2), um sistema pode ser dividido em camadas com base nas seguintes responsabilidades:

- lógica de apresentação
- lógica de negócios
- lógica de acesso de dados

Cada uma dessas responsabilidades pode ser representada por uma camada, conforme mostrado na Figura 4, que exibe algum conteúdo de amostra para cada camada. Aqui, consideramos três conceitos em um sistema de processamento de pedidos—Cliente, Pedido e Produto. Como exemplo, o conceito *Cliente* abrange o seguinte:

- *Classe VisualizaçãoCliente:* responsável pela *lógica de apresentação* associada a um cliente, como a exibição de um cliente na interface com o usuário
- *Classe Cliente*: responsável pela *lógica de negócios* associada a um cliente, como a validação dos detalhes do cliente
- *Classe DadosCliente*: responsável pela *lógica de acesso aos dados* associada a um cliente, como tornar o estado de um cliente persistente

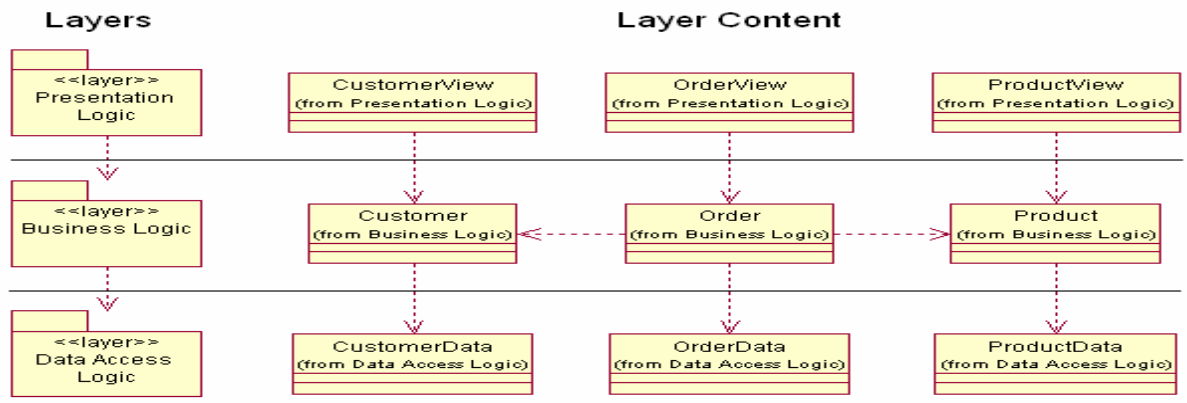

Figura 4: Camadas e Conteúdo para uma Camada com Base em Responsabilidade

Agora consideraremos alguns “mitos” dessa estratéfia de camada específica.

Mito 1: Camadas diferentes

Esse mito específico é normalmente a origem da confusão. O fato é que uma camada(tier) *é* uma camada, embora uma camada com base em uma estratégia específica—uma de responsabilidade. A confusão é composta pelo fato de que os conceitos de camadas podem ser aplicados de várias maneiras, conforme mostrado na Tabela 2.

Tabela 2: Definições de Camada

Aplicativo	Camadas (Tiers)
2 níveis	lógica de apresentação combinada e lógica de negócios lógica de acesso de dados
3 níveis	lógica de apresentação lógica de negócios lógica de acesso de dados

n-tier	lógica de apresentação lógica de negócios (*distribuída*) lógica de acesso de dados

Mito 2: A camada (tiers) implica em uma distribuição física

Outro equívoco comum é que a camada lógica implica em uma distribuição física. Considere uma camada de 3 níveis. Embora vários elementos residam em uma das camadas, cada camada pode ser aplicada de várias maneiras, conforme mostrado na Tabela 3, que utiliza nomes freqüentemente utilizados para caracterizar uma distribuição física específica (como "cliente thin").

Tabela 3: Aplicativo de Camada de 3 Níveis

Aplicativo	Camadas	
	Lado cliente	Lado servidor
Sistema único	lógica de apresentação lógica de negócios lógica de acesso de dados	
Cliente leve	lógica de apresentação	lógica de negócios lógica de acesso de dados
Cliente Fat	lógica de apresentação lógica de negócios	lógica de acesso de dados

Também é verdade dizer que um sistema único pode empregar mais de uma estratégia de distribuição física, em que determinados elementos seriam classificados como a redução de uma distribuição do "cliente thin" e outros como uma distribuição do "cliente fat". Normalmente, a escolha tem como base requisitos não funcionais como o desempenho.

Modelagem das camadas com base em responsabilidade

Como veremos, o aplicativo dessa estratégia pode influenciar o *modelo de design*e *o modelo de implementação* . O *modelo de design* é normalmente estruturado utilizando uma das duas abordagens.

A **primeira abordagem** mostra os elementos sendo "contidos" na camada. O resultado é exibido na Figura 5, uma captura de tela do navegador Rational Rose, que mostra:

- *classes de apresentação* (VisualizaçãoCliente, VisualizaçãoPedido e VisualizaçãoProduto) residindo em um pacote de Lógica de Apresentação
- *classes de lógica de negócios* (Cliente, Pedido e Produto) residindo em um pacote de Lógica de Negócios
- *classes de lógica de acesso aos dados* (DadosCliente, DadosPedido e DadosProduto) residindo em um pacote de Lógica de Acesso aos Dados

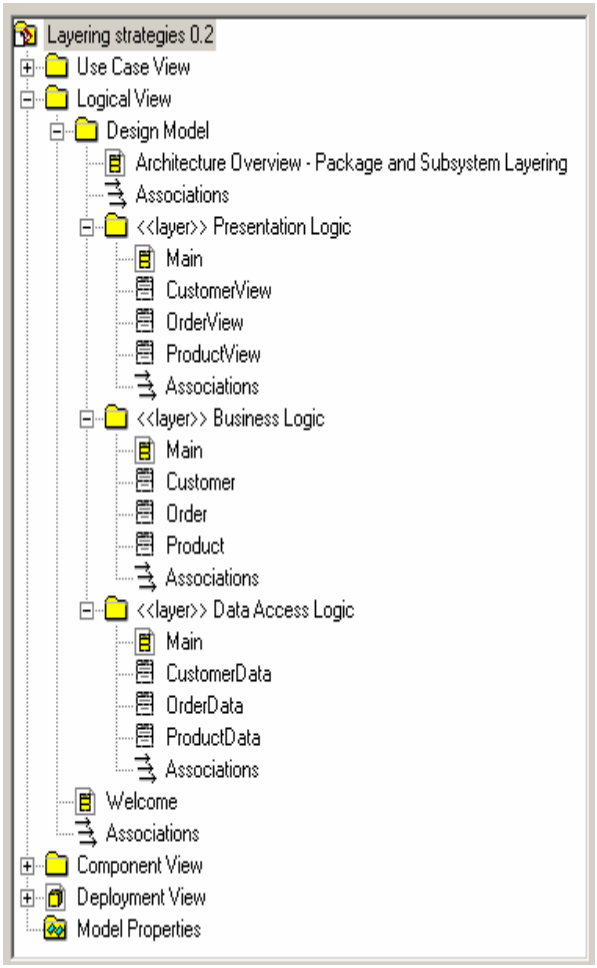

Figura 5: Elementos Contidos em Camadas

A segunda abordagem incorpora o conceito de um *componente de negócios* (neste caso, o Cliente, o Pedido e o Produto) como um cidadão de primeira classe, de acordo com o qual os lementos principais de preocupação são os conceitos relacionados ao domínio suportado pelo sistema. Por exemplo, o conceito *Cliente* pode ter elementos de lógica de apresentação, lógica de negócios e lógica de acesso aos dados associados. Esse conceito de um componente de negócios é discutido em [Eeles] e [Herzum]. Essa forma de pensar resulta na estrutura de modelo mostrada na Figura 6. Nesse exemplo, a camada é *indicada* pelos nomes do elemento. Por exemplo, todas as classes de *Visualização*(como a *VisualizaçãoCliente*) indicam uma camada lógica de apresentação e todas as classes de *Dados*(como a *DadosCliente*) indicam uma camada lógica de acesso aos dados. Nomes de classe não qualificados (como *Cliente*) indicam uma camada lógica de negócios.

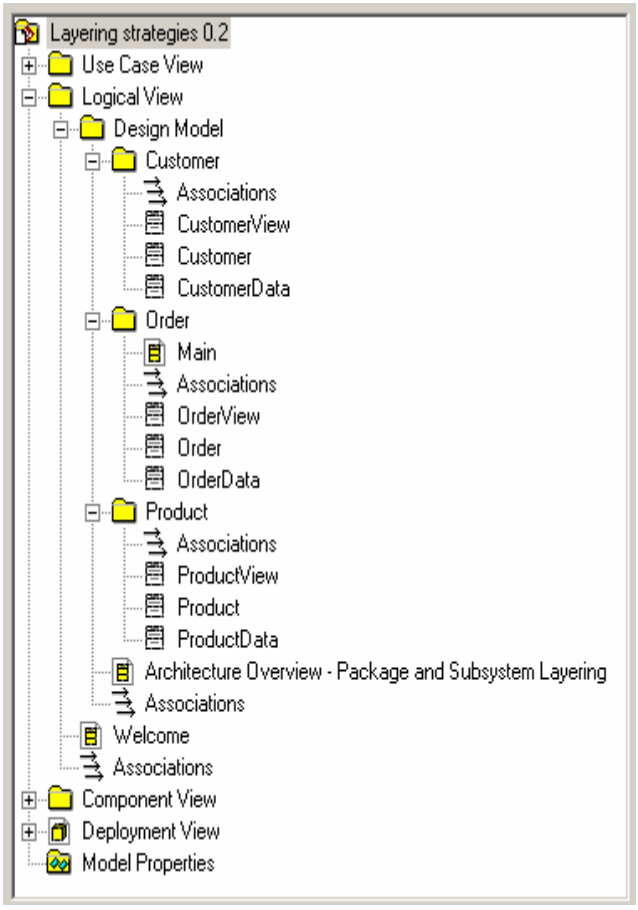

Figura 6: *Camada* Implícita em Cada Pacote de Componentes de Negócios

A camada também poderia ser representada explicitamente em cada pacote, representando um componente de negócios conforme mostrado na Figura 7. *Essa estruturação é preferível quando há um número de elementos envolvidos em cada camada de um determinado componente de negócios*. Embora somente o pacote de componente de negócios do Cliente tenha sido expandido neste exemplo, os pacotes de Pedido e de Produto teriam uma estrutura semelhante.

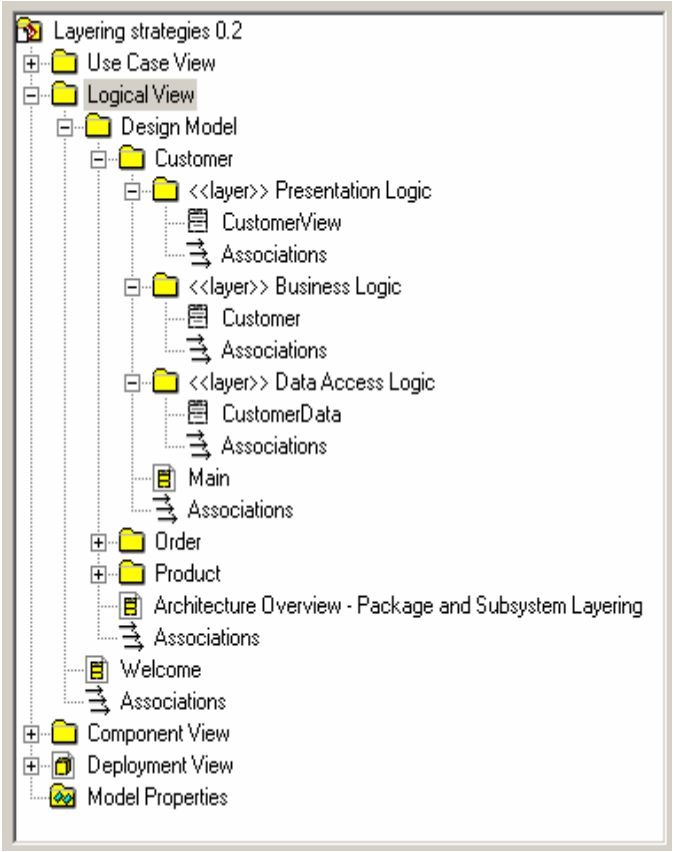

Figura 7: *Camada* Explícita em um Pacote de Componente de Negócios

Uma estratégia de camadas com base em responsabilidade normalmente influencia o *modelo de implementação* além do modelo de design quando há uma necessidade de particionar fisicamente os elementos que implementam cada responsabilidade. Por exemplo, considere um sistema que exibe uma distribuição física de um "cliente thin": é útil identificar as unidades de implementação requeridas para suportar a execução em um cliente e as requeridas para suportar a execução no servidor. Neste exemplo, os elementos na *camada lógica de apresentação* residem em um aplicativo que é implementado em um cliente e todos os elementos na *camada lógica de negócios* e *na camada lógica de dados* residem em outro aplicativo que é implementado em um servidor.

Esse cenário indica que um *modelo de implementação*, conforme mostrado na Figura 8, apresenta uma imagem do navegador Rational Rose e um diagrama de componente exibindo os elementos do aplicativo que é implementado no cliente. Neste exemplo, acontece uma mapeamento um a um entre uma classe no modelo de design e um componente UML no modelo de implementação. No entanto, observe que esse mapeamento normalmente depende da tecnologia de implementação utilizada.

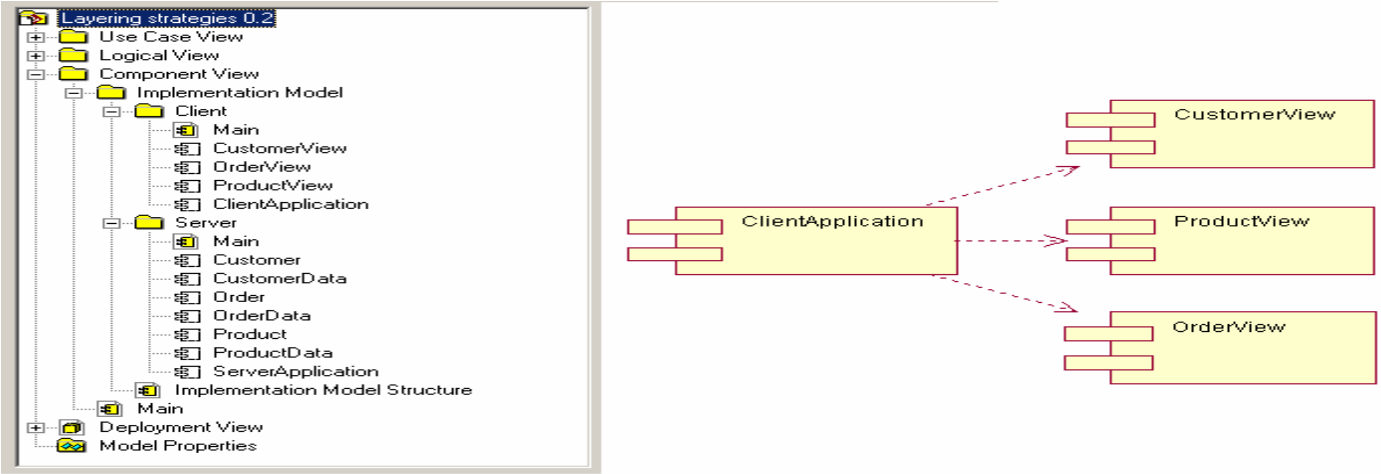

Figura 8: *Camada* Implícita no Modelo de Implementação

De maneira semelhante, uma estratégia de camada com base em responsabilidade também pode influenciar o *modelo de implementação* quando houver a necessidade de descrever a distribuição física das responsabilidades. Na Figura 9, e utilizando o exemplo acima, podemos ver que seis nós foram definidos. Cada um dos três *nós do Cliente* abriga um processo de AplicativoCliente. O nó do ServidordaExtremidadeFrontal abriga um processo BalanceadordeCarga que é responsável pela distribuição dos pedidos do cliente para um dos dois nós do Servidor. Cada *nó do Servidor* abriga um processo AplicativoServidor.

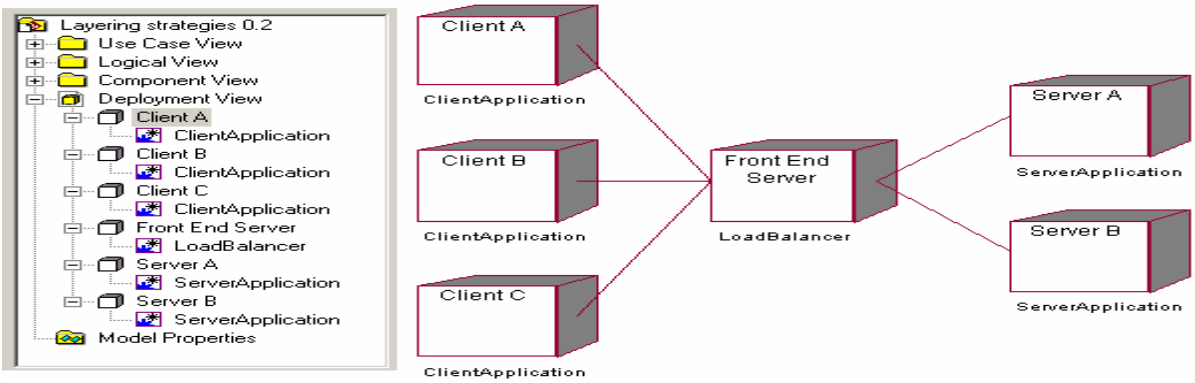

Figura 9: Modelo de Implementação Descrevendo a Distribuição Física das Responsabilidades

Modelagem com base na reutilização

Outra camada utilizada normalmente é uma com base na reutilização. Essa estratégia é particularmente relevante para as organizações que possuem uma meta identificável para reutilizar componentes em toda a organização. O impacto da utilização dessa estratégia de camadas é que a reutilidade de componentes é altamente visível, desde que os componentes estejam agrupados explicitamente de acordo com o seu nível de reutilização. Uma camada de exemplo, derivada de uma estratégia descrita em [Jacobson], é mostrada na Figura 10. Aqui nós vemos três camadas: Base, Específica para Negócios e Específica para Aplicativos.

- A *camada Base* contém elementos que pode ser aplicados em organizações (como Matemática). Esses elementos podem ser amplamente reutilizados.

- A *camada Específica para Negócios* contém esses elementos que se aplicam a uma organização específica, mas são independentes do aplicativo (como o Catálogo de Endereços). Esses elementos serão reutilizados em aplicativos da mesma organização.
- A *camada Específica para Aplicativos* contém elementos que se aplicam a um aplicativo ou projeto específico (como o Organizador Pessoal). Esses elementos são pelo menos reutilizáveis.

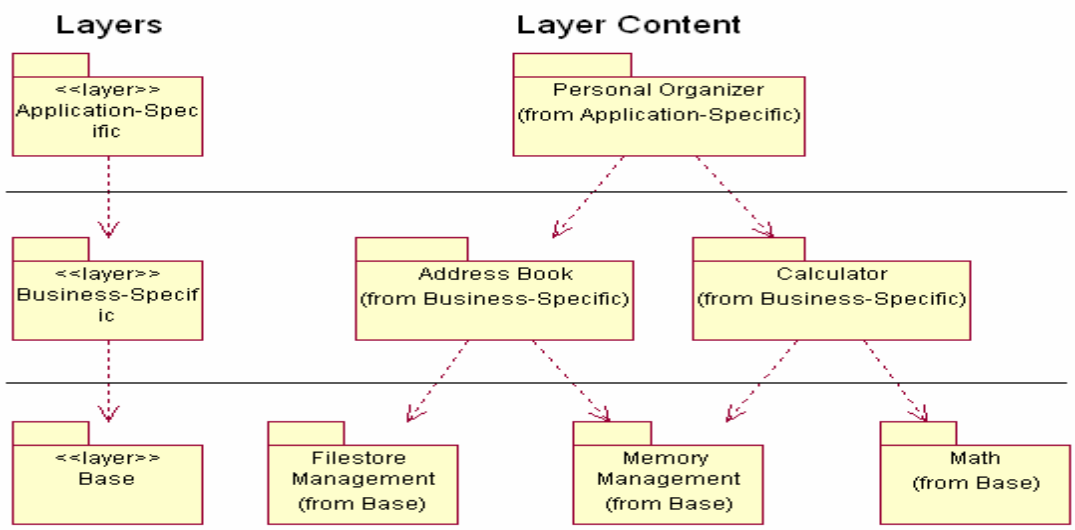

Figura 10: Exemplo da Camada com Base em Reutilização

Podemos dizer que os elementos na camada Base são os mais reutilizáveis, enquanto que os elementos da camada Específica para Aplicativos são mais específicos para o projeto e, portanto, menos reutilizáveis.

Modelagem das camadas com base em reutilização

O aplicativo de uma estratégia de reutilização influencia principalmente o *modelo de design*. A estrutura de um modelo de design que incorpora a camada com base em reutilização é de fácil visualização e é mostrada na Figura 11, que reflete o exemplo na Figura 10.

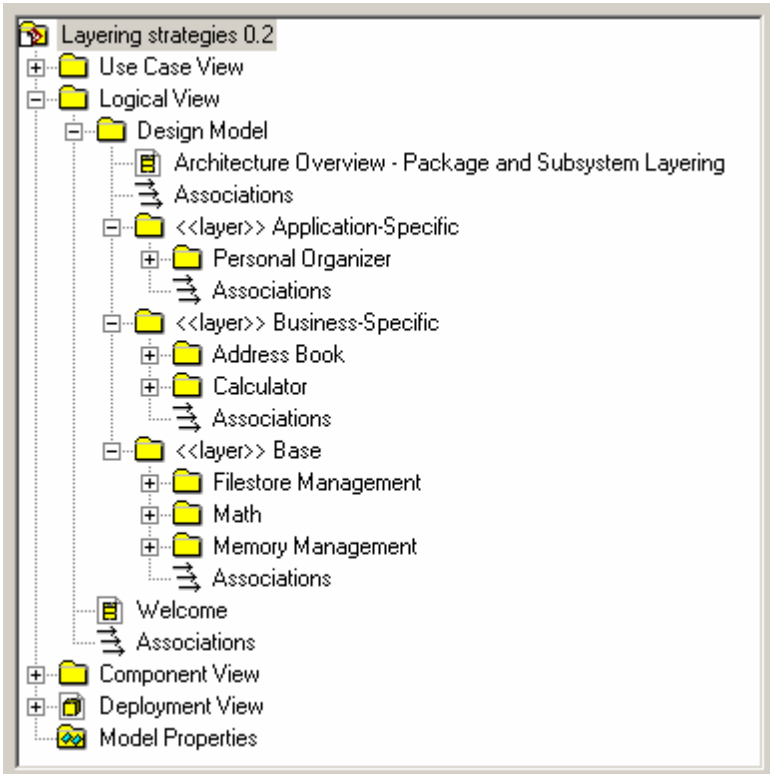

Figura 11: Modelo de Design Incorporando a Camada com Base em Reutilizaçãoi

Outras Estratégias de Camadas

Esse documento pretende simplesmente fornecer uma “versão” das diferentes estratégias de camadas que existem, utilizando as estratégias mais utilizadas como exemplos. No entanto, abordagens semelhantes poderia ser tomadas como estratégias que reconhecem características como segurança, propriedade e conjunto de habilidades.

Camada multidimensional

As estratégias descritas anteriormente também podem ser combinadas para criar novas estratégias de camadas. O exemplo na Figura 12 mostra:

- duas das camadas com base em reutilização do exemplo anterior
  - específica para aplicativos
  - específica para negócios
- três camadas com base em responsabilidade (tiers)
  - lógica de apresentação
  - lógica de negócios
  - lógica de acesso de dados

As dependências, presentes na estratégia de camadas com base em reutilização, normalmente resultam das dependências entre os elementos nas camadas de lógica de negócios, conforme descrito na Figura 12, em que vemos a dependência entre o OrganizadorPessoal e o CatálogodeEndereços.

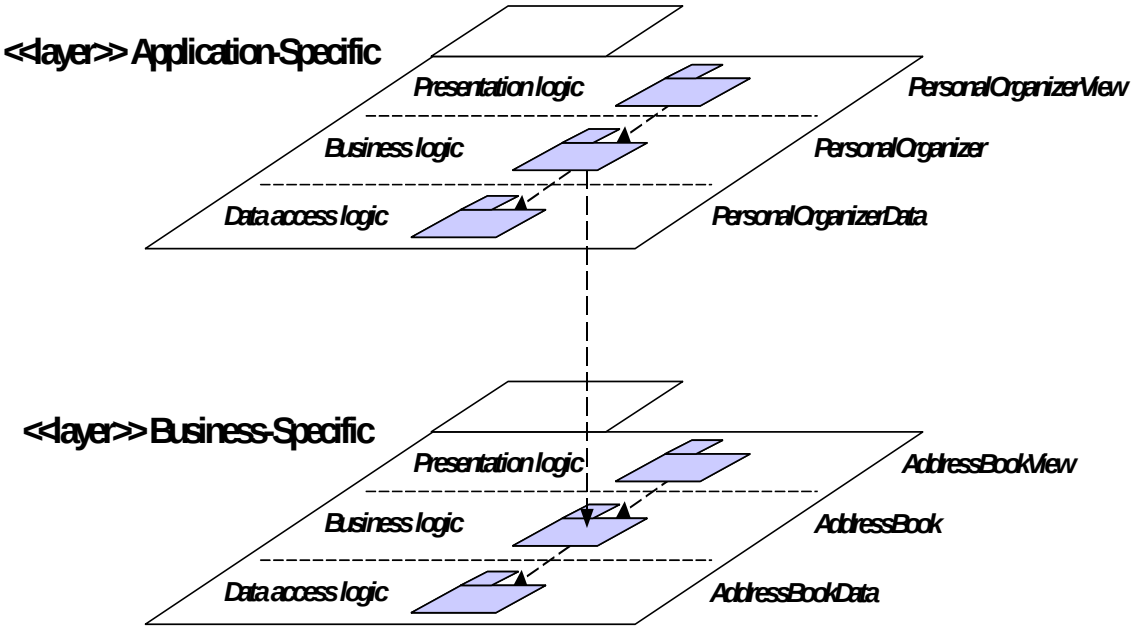

Figura 12: Camada Multidimensional

Modelagem de camadas multidimensionais

Aqui, consideramos a representação dos aspectos multidimensionais das camadas em um *modelo de design* bidimensional. Consideramos também a estrutura em que o conceito do *componente de negócios* é incorporado.

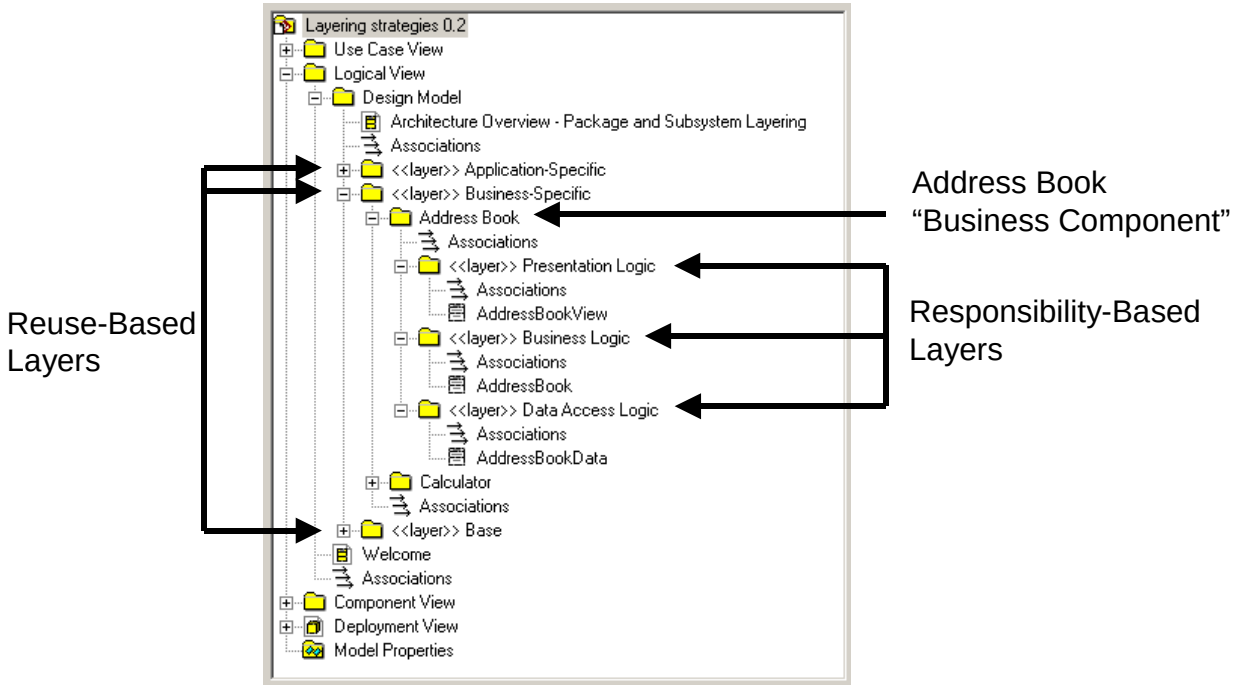

Figura 13: Modelo de Design Incorporando Camadas Multidimensionais

Adotar uma estratégia de camadas multidimensionais requer que uma estratégia principal seja identificada. Em nosso exemplo, a estratégia de camadas principal tem como base a reutilização. O modelo de design é organizado com base nessa estratégia, fornecendo as camadas *Específica para Aplicativos*, *Específica para Negócios*, e *Base*. Cada uma dessas camadas é organizada posteriormente pelos elementos que residem em cada uma dessas camadas, por exemplo, a Figura 13 mostra a camada Específica para Negócios contendo o *Catálogo de Endereços* e a *Calculadora*. Cada um desses elementos é organizado posteriormente com base em uma estratégia secundária: camada com base em responsabilidade. Por exemplo, o pacote do *Catálogo de Endereços* con´tem as três camadas *Lógica de Apresentação*, *Lógica de Negócios* e *Lógica de Acesso aos Dados*.

Cada uma dessas camadas contém elementos que residem nesta camada:

- a *camada da lógica de apresentação* contém a classe VisualizaçãoCatálogodeEndereços
- a *camada da lógica de negócios* contém a classe CatálogodeEndereços
- a *camada da lógica de acesso aos dados* contém a classe DadosdoCatálogodeEndereços

Conclusão

Uma das decisões mais importantes que um arquiteto deve tomar é escolher uma estratégia de camadas apropriada, já que ela terá uma influência maior na estrutura dos modelos produzidos. No entanto, de maior significância é o fato de que o negócios se beneficia, com a sustentabilidade e a reutilização, que podem ser suportadas diretamente pela estratégia de camadas escolhida. Por exemplo, mais sistemas de sustentabilidade provavelmente serão desenvolvidos, as diferentes responsabilidades do sistema devem ser isoladas umas das outras por meio da adoção de uma estratégia de camadas com base na responsabilidade. Além disso, os elementos do sistema reutilizável podem ser claramente identificados utilizando uma estratégia de camadas com base na reutilização.

Agradecimentos

O autor gostaria de agradecer as contribuições de Kelli Houston, Wojtek Kozaczynski, Philippe Kruchten, Bran Selic e Catherine Southwood (todos do Rational Software) por seus criteriosos comentários nos primeiros rascunhos desse documento.

Bibliografia

[Buschmann] Buschmann, Frank, et al. *A System of Patterns*. 1996. Nova Iorque: 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 e Oliver Sims. *Building Business Objects*. 1998. New York: John Wiley & Sons. ISBN 0-471-19176-0.

[Herzum] Herzum, Peter e 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 e Norman Kerth. *Pattern Languages of Program Design 2*. 1996. Reading, Massachusetts: Addison-Wesley. ISBN 0-201-89527-7.

Rational®
the software development company

Duas Sedes:

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

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

Sem custo: (800) 728-1212
E-mail: info@rational.com
Web: www.rational.com
Localização Internacional: www.rational.com/worldwide