# Diretrizes da Estrutura do Modelo para o Rational Software Modeler e o Rational Software Architect (Release de 2004)

White Paper

Bill Smith, Model Driven Development, IBM Rational Software

V1.0
8 de setembro de 2004

# Índice Analítico

# 1. Introdução

### Público-alvo

O white paper foi projetado para suportar usuários do produto RSA (Rational Software Architect) cujo interesse é aplicar a orientação localizada no RUP® (Rational Unified Process) em seu uso do RSA. Se você for usuário do RSM (Rational Software Modeler), também achará o white paper útil, mas observe que algumas seções refletem os recursos disponíveis no RSA e não no RSM. Ele é dirigido principalmente para o caso em que se está criando um novo conjunto de modelos no RSA. Se você for iniciante no RSA, mas anteriormente tiver sido usuário do Rational Rose ou do Rational XDE e tiver importado modelos desses produtos, é possível que ache o white paper útil como origem de orientação para a reestruturação dos modelos importados.

O white paper assume que você tenha amplo conhecimento de UML, assim como uma familiaridade básica com os conceitos fundamentais e teoria de operações do RSA, especificamente, modelagem, transformações e edição de código visual.

### Finalidade

O RUP descreve um conjunto de modelos que representam perspectivas bem definidas sobre os domínios de problema e solução de sistemas. A utilidade desse conjunto de estruturas de modelos foi comprovada em vários projetos do mundo real e, independentemente de você seguir um processo formal (como o RUP), essas estruturas devem ser consideradas. Este documento discute como compreender os artefatos do modelo RUP utilizando o RSA.

Conforme sugere o título, as estruturas de projeto e de modelo descritas aqui são diretrizes, não ordens. A decisão de modelar um determinado artefato do RUP no RSA faz parte de seu próprio processo de desenvolvimento sendo, muitas vezes, uma decisão específica do projeto. É bom lembrar-se de que o RUP não é um conjunto rígido de regras de processos. Ele é uma *estrutura* do processo a qual permite formular definições de processos que podem variar de muito formais a muito reduzidos.

A maneira de utilização da UML também pode variar de muito formal a muito informal. É possível optar pelo tratamento de modelos UML como desenhos de arquitetura formais que devem ser cuidadosamente seguidos durante a construção ou tratá-los mais como esboços que sugerem os contornos amplos de um design, mas que são considerados descartáveis depois que o projeto vai para implementação. O RSA

oferece suporte em qualquer extremidade do espectro de processo e modelagem. A partir desse ponto de vista, as diretrizes apresentadas neste white paper não pretendem confundi-lo, mas sim ajudá-lo a entender como utilizar os recursos do RSA para facilitar o processo que parecer melhor a você.

Observe que o RSA possibilita a utilização de modelos não apenas como cópias de projetos, mas como a especificação de uma solução a partir da qual as implementações podem ser geradas automaticamente. Isso é realizado utilizando as transformações modelo-para-modelo e modelo-para-código do RSA. A utilização de transformações do RSA para praticar o MDD (Model-Driven Development) introduz questões especiais sobre estruturas de modelos. Se você for utilizar os modelos e as transformações do RSA para praticar o Model-Driven Development, consulte também os vários recursos específicos do MDD do RSA, disponíveis no Developer Works.

### Escopo

Este white paper descreve como representar os artefatos do modelo RUP no RSA e oferece diretrizes para as estruturas organizacionais internas desses artefatos. Ele *não* tenta:

- Especificar os suportes conceituais dos artefatos do modelo RUP ou fornecer descrições extensas sobre eles
- Descrever o processo ou as técnicas para especificação de conteúdo detalhado semântico ou diagramático dos artefatos associados do RUP

Para obter informações sobre técnicas específicas de ferramentas para desenvolver o conteúdo de modelos RSA, consulte:

- A documentação do produto (tutoriais, amostras, ajuda on-line)
- Os mentores de ferramentas na configuração do RUP específica do RSA, da qual este whitepaper faz parte
- Recursos relacionados ao RSA no Developer Works

### Convenções Tipográficas

As discussões que interessam aos usuários do RSA que estejam migrando do IBM Rational **Rose ou** do XDE são apresentadas na forma de barra lateral, dentro de uma caixa de texto com moldura e segundo

> ***XDE/Rose***
> Discussão de interesse de usuários anteriores do XDE ou do Rose.

### Organização do White Paper

A seção Conceitos Básicos e Terminologia a seguir estabelece um vocabulário de trabalho e fornece algumas informações gerais sobre a implementação de modelos no produto RSA.

A seguir, a seção Mapeamento do Modelo RUP para o Modelo RSA discute como o RSA suporta os tipos de modelos definidos pelo RUP.

As várias seções subseqüentes fornecem orientação para a estruturação de modelos de vários tipos no RSA. Algumas delas discutem as diferentes maneiras de utilização de um tipo de modelo, dependendo

do grau de severidade desejado com relação ao processo, abordagem de modelagem e controle de arquitetura.

Finalmente, há uma discussão que abrange várias questões associadas à formulação de modelos em vários arquivos de modelagem. Isso está relacionado a estratégias para o gerenciamento de escalas e ativação de compartilhamento de modelos entre os membros da equipe para minimizar a disputa de arquivos e a mesclagem de conflitos.

## 2. Conceitos Básicos e Terminologia

***Modelos***

No RUP, um modelo é definido com uma especificação completa de um domínio de problema ou solução a partir de uma determinada perspectiva. Um domínio de problema ou um sistema pode ser especificado por vários modelos que representam perspectivas diferentes no domínio ou no sistema. O RUP propõe um conjunto específico de modelos:

- Modelo de Caso de Uso de Negócios
- Modelo de Análise de Negócios
- Modelo de Caso de Uso
- Modelo de Análise (pode ser incluído no Modelo de Design)
- Modelo de Design
- Modelo de Implementação
- Modelo de Implantação
- Modelo de Dados

Observe que o próprio RUP é uma ferramenta agnóstica. No que diz respeito a ele, um modelo poderia ser um desenho em um guardanapo ou em um papel liso, algo em uma ferramenta de modelagem ou até mesmo uma imagem mental. Portanto, a partir da perspectiva do RUP, um modelo é um conceito lógico. No contexto do RSA, podemos discutir modelos em termos lógicos, mas também em termos físicos.

Suponha que você tenha equipes trabalhando em dois aplicativos: uma equipe de três analistas trabalhando em um aplicativo de gerenciamento de folha de horários e uma segunda equipe de cinco analistas trabalhando em um aplicativo de centro de chamada. As duas equipes estão trabalhando no momento para capturar requisitos e estão utilizando o RSA para modelagem de casos de uso. Com relação ao RUP, você diria que uma equipe está construindo “o modelo de caso de uso para o aplicativo de folha de horários” e a outra está construindo “o modelo de caso de uso para o aplicativo de centro de chamada”. Mas, se as equipes estiverem utilizando o RSA, é importante reconhecer que seus modelos possuem determinadas manifestações físicas. Este é o assunto da seção a seguir.

***Arquivos de Modelagem***

No RSA, os modelos permanecem como arquivos. (Na terminologia do Eclipse, um arquivo é considerado um ‘recurso’ [1], portanto, se você encontrar o termo ‘recurso de modelagem’ neste white paper ou em outras origens, significa o mesmo que ‘arquivo de modelagem’). No sentido mais amplo, o RSA suporta dois tipos de arquivos de modelagem:

- “Arquivos de modelagem de pré-Implementação”. Esses arquivos contêm conteúdo UML conceitual que não reflete *diretamente* os artefatos de implementação. Eles contêm a semântica de UML do

[1] No Eclipse, um recurso é um arquivo, mas também possui propriedades e comportamento adicionais dentro do ambiente Eclipse. Os Arquivos de Modelagem descritos aqui são tratados como ‘recursos’ pelo Eclipse.

modelo e os diagramas de UML que representam os elementos da semântica.

- Arquivos de modelagem de implementação (recursos do Eclipse), que são artefatos normais de implementação, como arquivos de origem Java ou páginas da Web residentes em um projeto Eclipse. Nesse caso, o projeto pode ser considerado uma representação do escopo do conteúdo do arquivo de modelagem de implementação. A semântica do modelo reside nos próprios artefatos de implementação. Cada diagrama reside em seu próprio arquivo físico no projeto. Esses diagramas podem utilizar notação UML, mas também podem utilizar outras notações (por ex., IDEF1X ou notações da Information Engineering para visualização de dados ou uma notação de propriedade exclusiva do Rational utilizada para projetar camadas da Web). Os tipos de diagramas de UML suportados para representar artefatos de código 3GL incluem os diagramas de Classe e os diagramas de Seqüência (apenas para Java).

O foco deste white paper é como organizar as estruturas internas de modelos de "pré-implementação" e, **dentro do white paper, o termo "arquivo de modelagem" é reservado para arquivos que contêm conteúdo de modelo de "pré-implementação".** É possível localizar orientação para organizar o conteúdo de projetos de implementação em outras origens, como a ajuda on-line do Rational Software Architect, do Rational Application Developer e do Rational Web Developer.

Um arquivo de modelagem de "pré-implementação" não contém necessariamente todas as informações para um modelo. Na verdade, muitas vezes, ele contém apenas um subconjunto de um modelo. Por exemplo, na apresentação anterior, onde temos uma equipe de três trabalhando em um modelo de caso de uso para um aplicativo de folha de horários, a equipe poderia escolher particionar fisicamente seu modelo de caso de uso em três arquivos de modelagem para que cada membro da equipe pudesse trabalhar em um subconjunto diferente dos casos de uso sem disputar o mesmo arquivo. A seção final do white paper discute as questões associadas ao particionamento de modelos e gerenciamento de arquivos de modelagem.

### *Tipos de Modelos*

No RUP, os modelos possuem tipos específicos, como modelo de caso de uso, modelo de análise ou modelo de dados. No RSA, o arquivo de modelagem pode ser considerado como tendo um tipo: o tipo do modelo (ou subconjunto do modelo) que o arquivo contém. O tipo de um arquivo de modelagem pode ser estabelecido em uma de duas maneiras:

- Iniciar com um arquivo de modelagem em "Branco" (consulte a seguir) e, em seguida, estabelecer seu tipo simplesmente pela forma que escolherá nomeá-lo e pelo tipo de conteúdo que colocará nele (incluindo os perfis de UML aplicados nele).
- Criá-lo com base em um "modelo de gabarito" predefinido que representa um determinado tipo de modelo. Observe que o "tipo" de arquivo de modelagem é realmente apenas uma convenção referente ao conteúdo do arquivo. Por exemplo, a ferramenta não evitará que um arquivo que contém um modelo de caso de uso também contenha as classes que realizam os casos de uso. Entretanto, essas diretrizes recomendam que você trate os arquivos de modelo como tendo um tipo.

### *Espaços de Trabalho, Projetos e Tipos de Projetos*

Os leitores familiarizados com o Eclipse, os produtos WebSphere Studio ou o Rational Application Developer já saberão que os arquivos residem em Projetos, que os Projetos podem ter vários tipos e que são agrupados e gerenciados (virtualmente) dentro de Espaços de Trabalho. Os arquivos de modelagem do RSA residem em projetos assim como outros arquivos.

Nesta discussão, nem todos os tipos de projetos disponíveis no Rational Software Architect, no Rational Application Developer e no Rational Software Modeler precisam ser explicados em detalhes. Em termos gerais, estamos interessados em duas categorias de projetos:

- **Projetos UML**

- **Projetos de implementação**, que incluem tipos especializados, como o Projeto Corporativo, o Projeto EJB, o Projeto da Web e o Projeto C++

Afirmamos anteriormente que o RSA suporta dois tipos de arquivos de modelagem:
- Arquivos que contêm modelos UML (elementos de semântica e diagramas que os representam) utilizados para modelagem em níveis de abstração acima da implementação (como requisitos, análise e design)

- Projetos que contêm artefatos de implementação, como código Java ou páginas da Web e (opcional) arquivos de diagrama que contêm diagramas representando os artefatos de implementação.

A regra de alocação de modelos para projetos é simples: **a)** coloque os arquivos de modelagem de "pré-implementação" em projetos UML e **b)** os modelos de implementação cuidam de si mesmos porque, essencialmente:

[implementation model] = [implementation project]

Há algumas exceções para essa regra. Os arquivos de modelagem UML a seguir são candidatos para posicionamento em um projeto de implementação específico do idioma:

- 'Modelos de esboço' de design (que serão discutidos em uma seção posterior)
- Modelos com diagramas de seqüência descrevendo testes que serão executados no código no projeto

> ***XDE/Rose***
> As teorias de operação do Rose e do XDE incluem a prática de aperfeiçoar iterativamente o Modelo de Design até você alcançar um nível de abstração equivalente ao código e, em seguida, utilizar a tecnologia de sincronização do modelo de código para manter a semântica desse modelo em acordo com o próprio código. Por exemplo, no XDE, os modelos de implementação existem não apenas como código e diagramas nos projetos, mas também como arquivos de 'modelo de código' que permanecem independentemente dos artefatos de implementação e representam, essencialmente, uma cópia redundante de sua semântica.
>
> A teoria de operações do RSA encoraja a utilização de modelos com plataforma neutra em níveis de abstração maiores do que o código (isto é, modelos de design, como um Modelo de Design de TI Corporativo) e a utilização de transformações para gerar código a partir desses modelos. Em seguida, no nível de abstração do código, o RSA simplesmente permite desenhar diagramas de UML de código e dispensar, com a abordagem de utilização de um modelo de semântica que permanece separadamente.
>
> Observe que o RSA não *impede* que você defina modelos UML em um nível de abstração do código e gere código a partir deles; na verdade, esse tipo de uso é esperado. Mas o RSA não fornece tecnologia para manter esses modelos sincronizados com código. Esse tipo de uso

### *Conceitos em Revisão*

As ilustrações a seguir resumem as discussões precedentes. Elas refletem o cenário descrito anteriormente, em que temos duas equipes, uma trabalhando em um aplicativo de centro de chamada e outra trabalhando em um aplicativo de gerenciamento de folha de horários.

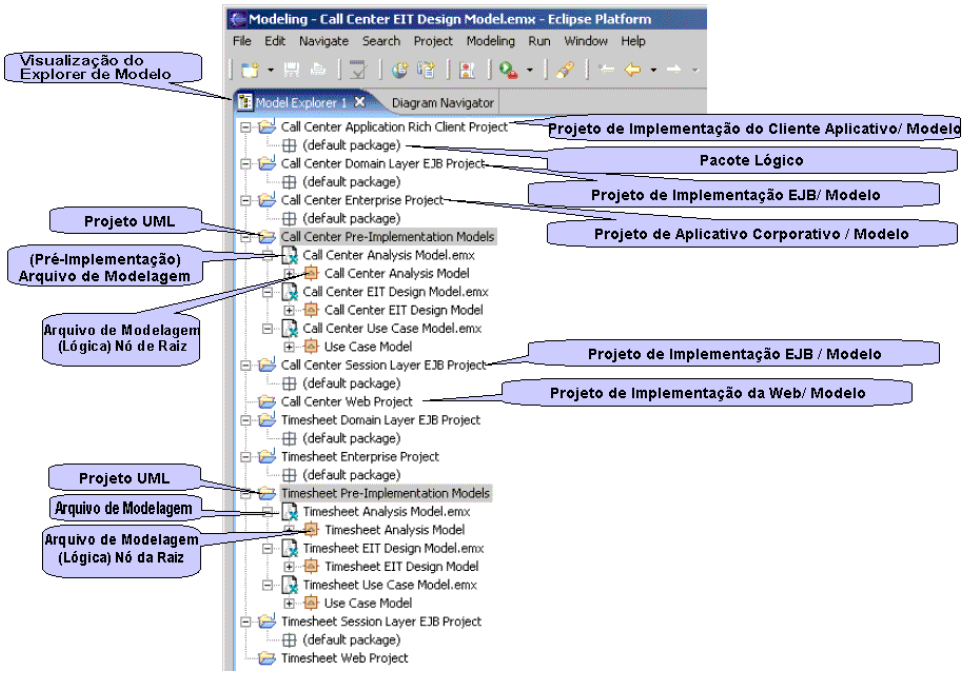



***Figura 2-1***

O RSA fornece uma visualização do Explorer de Modelo, que fornece uma visualização física e lógica combinada de modelos. No Explorer de Modelo, você vê os projetos no espaço de trabalho representados como nós de nível superior e, dentro de cada projeto, você vê os recursos pertencentes a esse projeto. **Figura 2-1** : representa no Explorer de Modelo uma coleta de projetos que correspondem aos dois aplicativos de nosso cenário. Os projetos UML foram utilizados para os modelos de pré-implementação. Uma coleta de outros projetos de tipos de solução apropriados (como projetos de Aplicativo Corporativo, projetos da Web, etc.) foram utilizados para os modelos de implementação.

> ***XDE/Rose***
> Ao contrário do explorer de modelo do RSA, os explorers de modelos no Rose e no XDE forneceram apenas uma visualização lógica dos modelos. Observe que a visualização de recursos fornecidos pelo Explorer de Modelo do RSA não é a visualização física 'pura' fornecida pela visualização Navigator do Eclipse. Enquanto alguns recursos físicos são visíveis no Explorer de Modelo, eles são, na maioria das vezes, representados por ícones que indicam visualizações *lógicas* dos recursos.

A Figura 2-2 mostra como o modelo de caso de uso de Folha de Horários poderia ser organizado internamente em pacotes que representam alguns conjuntos funcionalmente coesos do domínio de problema.

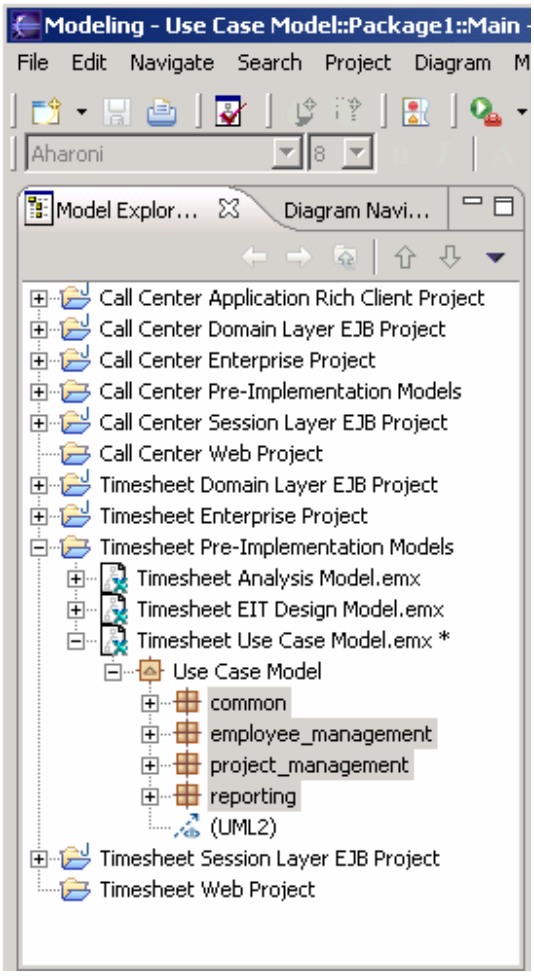



***Figura 2-2***

Na **Figura 2-1** e **2-2**, cada um dos modelos de pré-implementação reside em um único arquivo de modelagem. Alternativamente, qualquer um deles poderia ser reformulado em vários arquivos de modelagem. Por exemplo, o modelo de caso de uso de Folha de Horários poderia ser reformulado em quatro arquivos de modelagem, em que cada um correspondesse a um dos subconjuntos representados do domínio de problema (“common”, “employee_management”, “project_management” e “reporting”). Nesse caso, o nó de raiz de cada arquivo de modelagem seria nomeado para manter um espaço de nomes consistente em todos os arquivos de modelagem que compõe o modelo de caso de uso completo. Por exemplo, os nós de raízes dos quatro arquivos de modelagem poderiam ser “timesheet.requirements.common”, “timesheet.requirements.employee_management”, “timesheet.requirements.project_management” e “timesheet.requirements.reporting”.

## 3. Mapeamento do Modelo RUP para o Modelo RSA

A tabela a seguir mostra como os modelos RUP mais comumente utilizados são mapeados para os tipos de modelos RSA. O mapeamento é geralmente direto, mas é a chave para se utilizar este white paper como guia para praticar o RUP com o RSA. Os tipos de modelos RSA mencionados na tabela são discutidos imediatamente depois da tabela. As diretrizes para a organização interna dos vários tipos de modelos e os tipos de projetos nos quais mantê-los são fornecidos nas seções posteriores. Essas discussões são apresentadas com relação aos tipos de modelos do RSA listados aqui.

| Modelo RUP | Tipo de Modelo RSA |
|---|---|
| Modelo de Caso de Uso | Modelo de Caso de Uso |
| Modelo de Análise | Modelo de Análise<br><br>(alternativo: pacotes de «análise» no modelo de Design) |
| Modelo de Design | Para aplicativos de negócios com n camadas: Modelo de Design de TI Corporativo<br><br>Para outros tipos de aplicativos: Modelo em Branco utilizado como modelo de design<br><br>Para 'esboço' de design: Modelo em Branco utilizado como modelo de 'esboço' de design<br><br>Suplemento opcional: Modelo em Branco utilizado como Modelo de Visão Geral da Implementação |
| Modelo de Implementação | Projetos de implementação contendo artefatos de implementação e arquivos de diagrama |
| Modelo de Implantação | Modelo em branco utilizado como modelo de implantação |

### Tipos de Modelos RSA

#### *Modelo em Branco*

O RSA fornece a opção de criar um "Modelo em Branco" (Arquivo→Novo→Modelo UML→Modelo em Branco). Um "Modelo em Branco" é um arquivo de modelagem que não se baseia em um gabarito de modelo. Ele não possui perfis especiais aplicados e nenhum conteúdo padrão, exceto um único diagrama "Principal" (formato livre). **É possível utilizar um arquivo de modelagem em branco como ponto de partida para qualquer tipo de modelo**. Escolhendo o nome que dará a ele, o conteúdo que definirá dentro dele e os perfis que serão aplicados nele, você pode utilizar um arquivo de modelagem em branco para construir um modelo de caso de uso, um modelo de análise, um modelo de design, um modelo de implantação ou qualquer outro tipo de modelo RUP.

***Modelo de Caso de Uso***

O RSA fornece a opção de criar um arquivo de “Modelo de Caso de Uso” com base em um gabarito de modelo. O gabarito contribui com conteúdo padrão, conforme representado na ***Figura 3-1***. (Está fora do escopo deste documento explicar como o conteúdo do “bloco de construção” e as cadeias de procura são utilizadas. Os gabaritos contêm instruções auto-explicativas.)

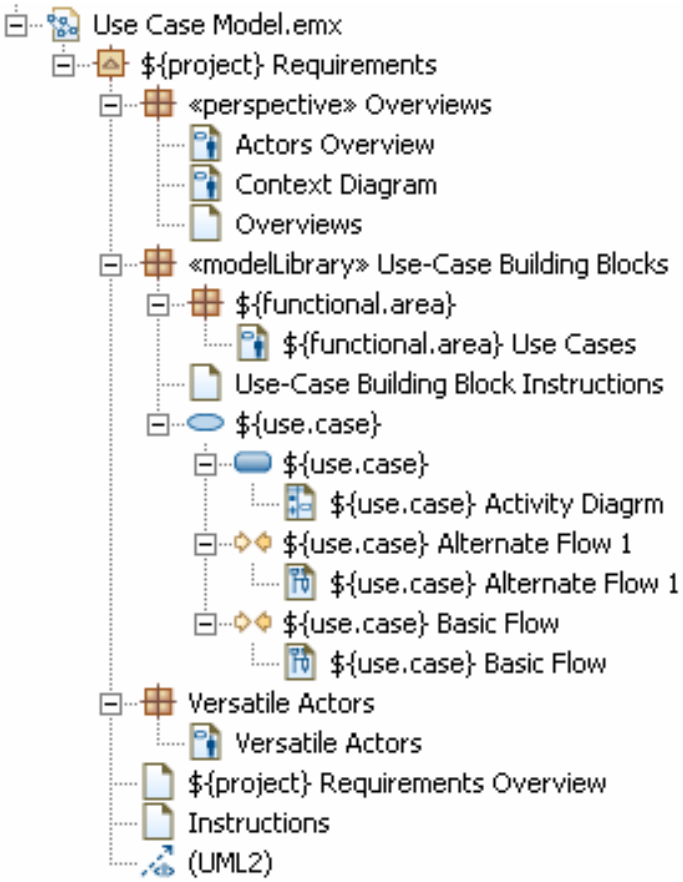



***Figura 3-1***

***Modelo de Análise***

O RSA fornece a opção de criar um arquivo de “Modelo de Análise” com base em um gabarito de modelo. O gabarito contribui com conteúdo padrão, conforme representado na ***Figura 3-2***. Além disso, um perfil de “Análise” é aplicado nos arquivos de modelo criados a partir deste gabarito:

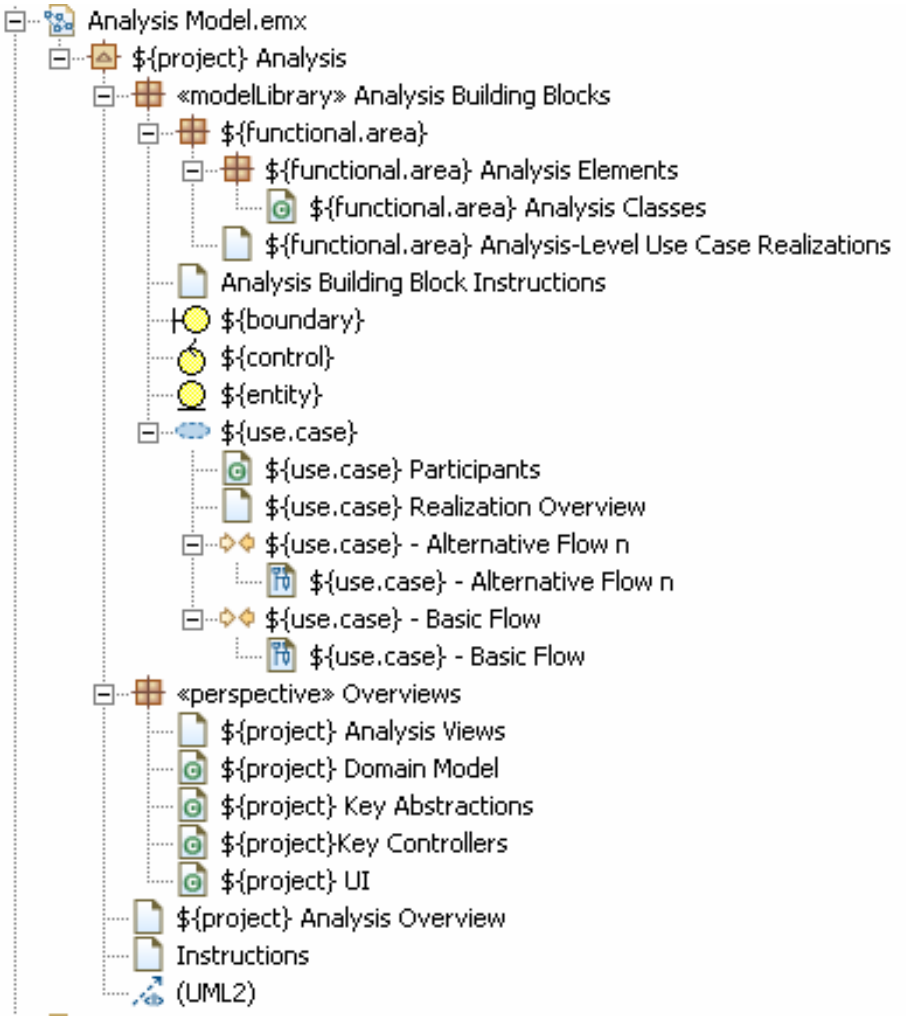



***Figura 3-2***

### *Modelo de Design de TI Corporativo*

O RSA fornece a opção de criar um arquivo EITDM (“Modelo de Design de TI Corporativo”) com base em um gabarito de modelo. O gabarito fornece conteúdo padrão, conforme representado na **Figura 3-3**. Além disso, um perfil[2] de “Transformação EJB” será aplicado nos arquivos de modelo criados a partir desse gabarito. Esse é o gabarito apropriado para utilizar para o design (e, opcionalmente, para a análise) ao direcionar aplicativos de negócios e utilizar transformações de geração de código do RSA para suportar a criação desses aplicativos.

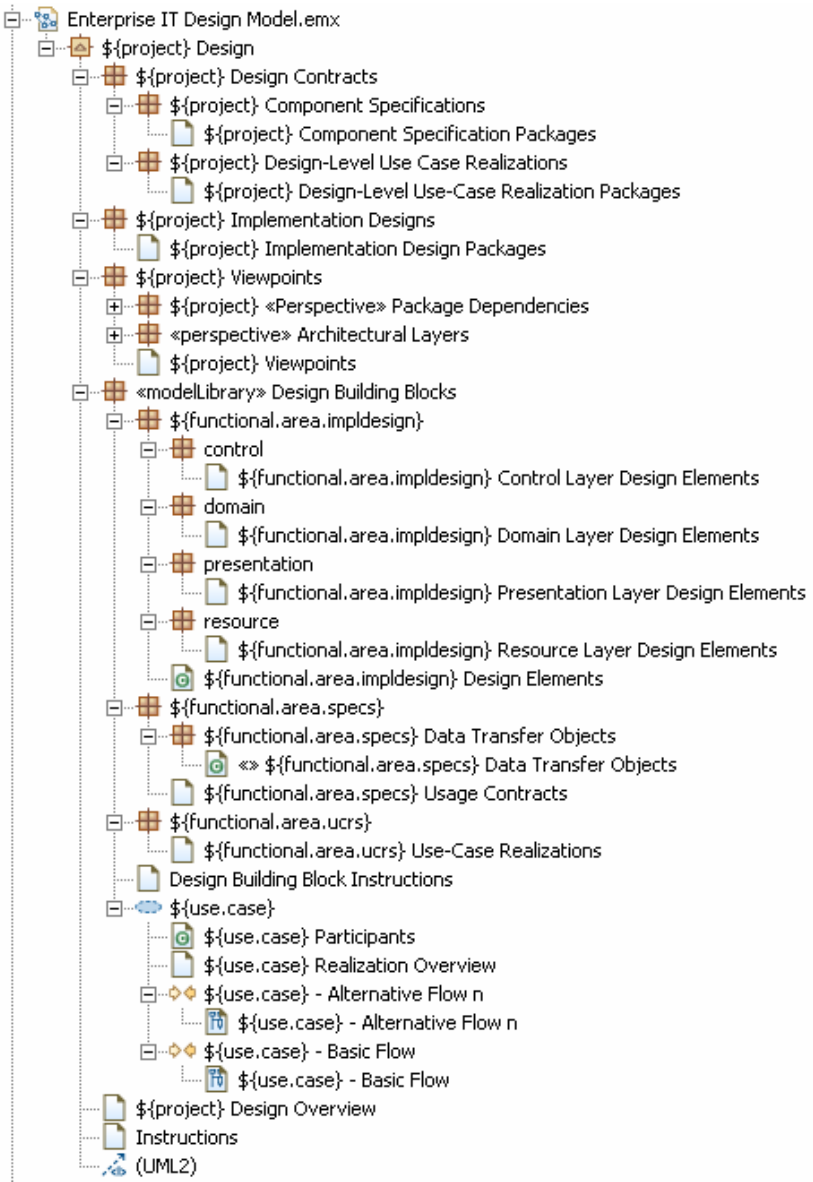



***Figura 3-3***

[2] O conjunto de perfis de ativação de transformação fornecido como parte do gabarito do Modelo de Design EIT provavelmente evoluirá com as liberações de atualizações para o produto.

### *Modelo de Visão Geral da Implementação*

Como parte de seu modelo de design, você ainda pode achar útil definir um "Modelo de Visão Geral da Implementação" para capturar uma visão geral de alto nível de como a implementação será organizada. O "Modelo de Visão Geral da Implementação" seria utilizado antecipadamente na fase de design—antes da geração ou gravação do código—para representar os projetos e pastas/pacotes reais do RSA nos quais você espera que o código e os arquivos relacionados (metadados, descritores de implementação, etc.) residam. Esse modelo também poderia ser utilizado para mostrar as dependências previstas entre esses projetos e pacotes, que podem ser úteis ao identificar requisitos de construção do sistema. O Modelo de Visão Geral da Implementação também pode ser um local para manter diagramas conceituais informais da arquitetura da solução.

### *Modelo de Implementação*

Conforme indicado anteriormente, no RSA um modelo de implementação consiste em um projeto que contém artefatos de implementação e (opcionalmente) diagramas que representam esses artefatos[3].

### *"Modelos de Esboço"*

Conforme observado na seção "Conceitos Básicos e Terminologia", você pode escolher tratar modelos de design como desenhos formais de arquitetura que são mantidos durante a existência do sistema e utilizados para suportar/forçar o controle da arquitetura. Ou poderia tratá-los mais como esboços que servem para sugerir um design e ajuda a esclarecer e a comunicá-lo, mas que são considerados descartáveis depois que a implementação real começa a divergir do design original. O RSA suporta as duas abordagens. Seus recursos geralmente não têm como destino específico uma ou outra abordagem, mas as opções feitas sobre como utilizar os modelos de design certamente ajudam a determinar quais recursos do RSA você utilizará e como os utilizará. Quando a distinção for importante no contexto das diretrizes aqui apresentadas, o termo "modelo de esboço" será utilizado para indicar que um modelo está sendo utilizado da maneira mais 'descartável'.

---

[3] Para criar esses diagramas, em vez de utilizar Arquivo→Novo→Modelo UML para criar um modelo, utilize Arquivo→Novo→Diagrama de Classe para criar um diagrama no qual você possa compor 'visualizações' de código em notação UML (ou outra). Cada diagrama individual permanece como um arquivo separado com uma extensão de .dnx e pode ser controlado por versão da mesma forma que um arquivo de código. Esses diagramas não contêm informações de semântica, apenas notação. Todas as informações relevantes de semântica residem no próprio código. Quando você altera algo como um nome de classe ou uma assinatura de operação em um desses diagramas, você está realmente alterando o próprio código de base. Ao fazer essas alterações no código (utilizando um editor de texto), os diagramas nos quais o código alterado aparece são atualizados automaticamente.

# 4. Diretrizes e Técnicas Gerais para Organização da Estrutura Interna de Modelos

A principal ferramenta para organizar o conteúdo de modelos UML é o pacote. Os pacotes UML servem a duas finalidades principais:

- particionar, organizar e identificar informações de modelos
    - agrupando elementos que correspondem a um assunto específico no domínio de problema ou solução
    - separando os diferentes tipos de informações de modelo, como interfaces, implementações, diagramas, etc
    - agrupando elementos para definir e controlar suas dependências de outros elementos
    - agrupando diagramas que fornecem visualizações alternativas no mesmo modelo

- estabelecer espaços de nomes
    - para elementos de modelo
    - para artefatos de implementação gerados a partir de elementos de modelo (isso poderia envolver mapeamentos entre os espaços de nomes de idioma do modelo e da implementação)
    - para uma unidade de reutilização

Tradicionalmente, o RUP propõe estratégias específicas de empacotamento para vários tipos de modelos. Essas estratégias são refletidas nas seções específicas do tipo de modelo deste white paper. O RSA também introduz algumas ferramentas organizacionais adicionais, que são descritas aqui:

**Representar Pontos de Vista Utilizando Pacotes de perspectiva**

Em casos em que se deseja ver os elementos organizados em mais de uma maneira, é possível criar pacotes adicionais com diagramas que representam os esquemas organizacionais alternativos. Essa mesma técnica pode servir em qualquer lugar onde haja a necessidade de representar uma determinada visualização no conteúdo do modelo que é recortado no esquema de empacotamento do modelo. O RSA suporta essa técnica fornecendo um estereótipo de pacote de «perspectiva» como parte de seu 'perfil base' de UML. Um pacote de «perspectiva» pode ser considerado geralmente o equivalente a um "Ponto de vista" da Engenharia de Sistemas RUP ou IEEE 1417.

Não coloque elementos de semântica (classes, pacotes, associações, etc.) dentro dos pacotes de «perspectiva». Coloque neles apenas diagramas que representem visualizações com base no ponto de vista da questão ou do aplicativo organizacional alternativo. A aplicação do estereótipo «perspectiva» em um pacote faz várias coisas. Identifica visualmente esse pacote como uma representação de um determinado ponto de vista. Também suporta uma regra de validação de modelo que alerta quando os elementos da semântica são colocados em um pacote de «perspectiva». Também serve como designador de pacotes que devem ser ignorados por transformações do RSA

**Criar Representações de Auto-atualização de Questões Específicas Utilizando Diagramas de Tópico**

Ao contrário de diagramas 'normais' nos quais você coloca manualmente os elementos que deseja representar, o conteúdo de um Diagrama de Tópico é determinado por uma consulta executada no conteúdo do modelo existente. Para criar um Diagrama de Tópico, selecione um elemento de modelo 'de tópico' e, em seguida, defina quais outros elementos deseja que apareçam no diagrama com base nos tipos de relacionamentos que eles têm com o elemento de tópico. Conforme ocorrem alterações no conteúdo semântico do modelo, os Diagramas de Tópico são ajustados adequadamente.

**Examinar Modelos via Diagramas de Navegação**

Os Diagramas de Navegação não são *especificamente* uma ferramenta para organização de modelos. Sua finalidade é facilitar a descoberta e o entendimento do conteúdo do modelo sem ter que compor manualmente os diagramas. Mas, no contexto da organização de modelos, é bom prestar atenção neles, uma vez que podem reduzir a necessidade de compor diagramas que permanecem. Isso, por sua vez, poderia reduzir o tamanho e a complexidade de seus modelos, deixando-os mais fáceis de organizar.

Os Diagramas de Navegação são um pouco como os diagramas de Tópico, mas com a principal diferença de que nunca permanecem, são sempre gerados automaticamente. Para produzir um Diagrama de Navegação, selecione um elemento de modelo (em um diagrama ou no Explorer de Modelo) e utilize o menu de contexto para "Explorar no Diagrama de Navegação". Isso produzirá um diagrama representando o elemento selecionado como o 'ponto focal' com elementos relacionados apresentados em um layout radial em torno do ponto focal. Naturalmente, você poderá, em seguida, selecionar um dos elementos relacionados nesse Diagrama de Navegação, torná-lo o ponto focal de outro Diagrama de Navegação e continuar dessa maneira o quanto quiser.

**Navegação entre os Diagramas**

No RSA, há dois mecanismos para navegação entre os diagramas:

- É possível arrastar um nó de diagrama do Explorer de Modelo para algum outro diagrama do 'host'. Em seguida, você pode dar um clique duplo no ícone resultante no diagrama do host para abrir o diagrama referenciado.

- Sempre que você criar um novo pacote UML em um modelo, um diagrama (de formato livre) "Principal" será criado automaticamente. Por padrão, esse diagrama "Principal" é criado como o diagrama 'padrão' do pacote. Você pode renomeá-lo para algo diferente de "Principal" e ele ainda será tratado como 'padrão'. Também é possível selecionar um diagrama diferente no pacote e torná-lo o diagrama 'padrão' para esse pacote. A finalidade do diagrama 'padrão' é esta: se você colocar o próprio pacote em algum outro diagrama do 'host', poderá então dar um clique duplo nele, o que abrirá seu diagrama padrão.

Esses mecanismos suportam as **diretrizes** organizacionais a seguir, que podem se aplicar a modelos de qualquer tipo:

1. Compor o diagrama Principal (ou outro diagrama padrão) de cada arquivo de modelagem a ser representado:
    a. cada pacote de nível superior no arquivo de modelagem
    b. os ícones do diagrama de quaisquer outros diagramas que residam no pacote raiz do arquivo de modelagem (ou seja, não representar o ícone do próprio diagrama padrão)

2. Compor o diagrama Principal (ou outro diagrama padrão) de cada pacote de nível superior a ser representado:
    a. os pacotes diretamente contidos
    b. os ícones do diagrama de quaisquer outros diagramas diretamente contidos

3. Repetir esse padrão para cada nível inferior de pacotes sucessivamente.

## 5. Diretrizes para Organização Interna do Modelo de Caso de Uso

**NOTA:** Nesta e nas outras seções específicas de tipo de modelo a seguir, as diretrizes serão ilustradas utilizando exemplos obtidos do exemplo de Leilão incluído com o RSA. Estude o exemplo para ver detalhes organizacionais adicionais e o conteúdo dos diagramas.

**Organização de Alto Nível do Modelo de Caso de Uso**

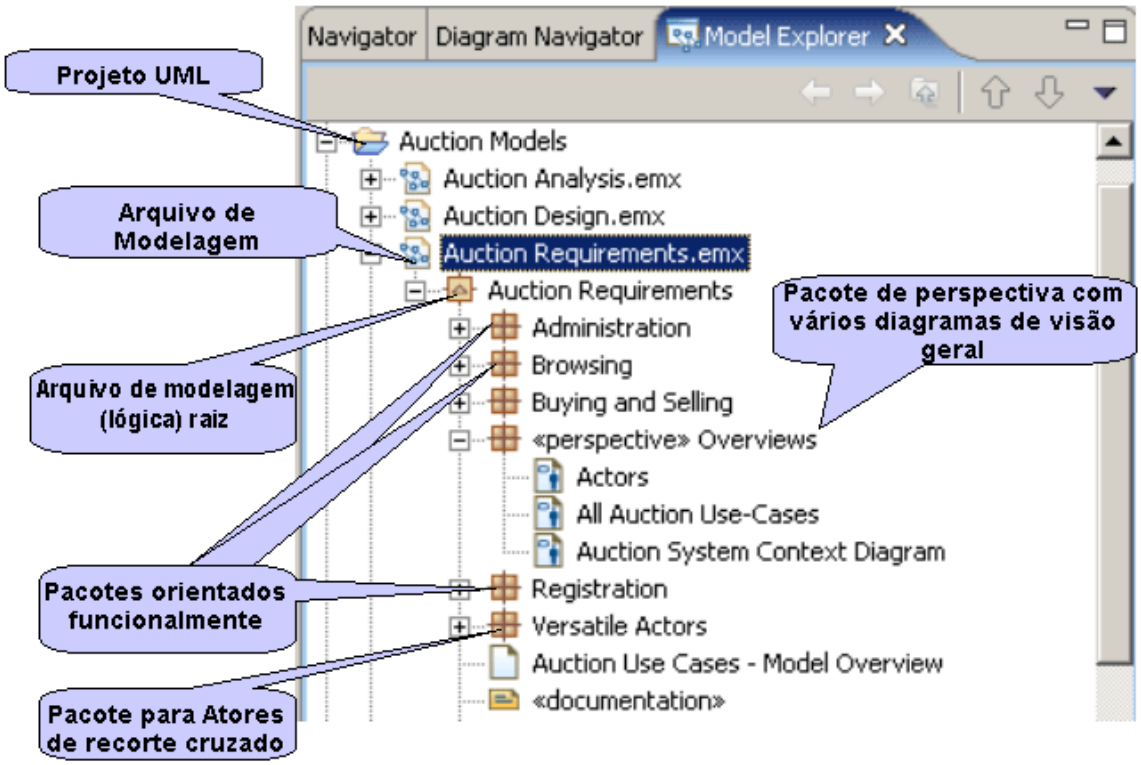



***Figura 5-1***

**Figura 5-1** : ilustra as diretrizes a seguir para estruturação de modelos de caso de uso:

1. Utilizar pacotes de nível superior para estabelecer agrupamentos orientados funcionalmente. **Análise Racional**:

   - Geralmente isso será bem mapeado para problemas de divisão de trabalho quando uma equipe de pessoas for trabalhar no modelo de caso de uso. E a configuração também será adequada no caso de você decidir mais tarde dividir o modelo de caso de uso em vários arquivos de modelagem em razão da disputa de arquivos ter se tornado um problema (é só criar um arquivo de modelagem separado por pacote de nível superior).

   - Comparada com outras abordagens organizacionais, geralmente o mapeamento desta será melhor para a organização da implementação eventual. Isso será importante se você for utilizar transformações para iniciar cada nível inferior sucessivo de abstração. Especificamente, se você for gerar conteúdo inicial em um modelo de análise com base no modelo de caso de

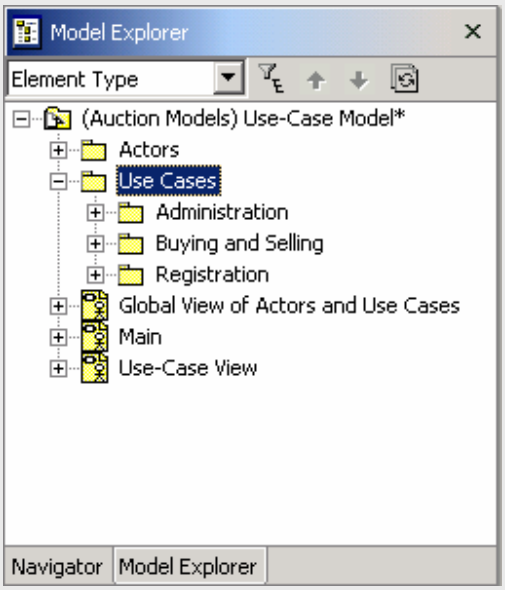
uso, desejará que a estrutura de empacotamento do modelo de caso de uso seja bem mapeada para a estrutura de empacotamento desejada do modelo de análise de destino. Por sua vez, você desejará que a estrutura de empacotamento do modelo de análise seja bem mapeada para o modelo de design e que a estrutura de empacotamento do modelo de design seja bem mapeada para o conjunto de projetos que conterá a implementação. Quanto mais simples esses mapeamentos, menos trabalho será necessário para configurar as transformações de um nível de abstração para o outro.

2. Utilizar outro pacote de nível superior para capturar Atores 'amplamente capacitados' ou 'versáteis'.

3. Utilizar diagramas em pacotes de «perspectiva» para capturar visualizações de recorte cruzado de alto nível dos casos de uso. **Análise Racional**:

   - Fornecer visualizações de recorte cruzado e visualizações de casos de uso 'significativos do ponto de vista da arquitetura' enquanto se mantém os elementos da semântica do modelo organizados em agrupamentos orientados funcionalmente.

***XDE/Rose***
A orientação do RSA revisa um pouco a orientação tradicional da organização de alto nível do modelo de caso de uso, que era criar um pacote para atores e outro para os casos de uso. Em seguida, conforme o tamanho e a complexidade do modelo pedia, seriam utilizados pacotes de nível inferior para estabelecer agrupamentos orientados funcionalmente, conforme mostrado neste exemplo do XDE:




**Conteúdo do Modelo de Caso de Uso**

Está fora do escopo deste documento servir como um tutorial detalhado sobre como gravar bons casos de uso ou os prós e contras da boa modelagem de casos de uso. Entretanto, a seguir está uma breve discussão sobre o que poderia ser incluído em um modelo de caso de uso além dos atores e dos casos de uso.

- **Recomendado:** Crie um diagrama 'principal' na raiz do modelo que represente os outros pacotes do modelo e suporte pesquisa detalhada nesses pacotes e em seus respectivos diagramas 'principais'.

- **Recomendado:** Em cada pacote de caso de uso, inclua um diagrama que represente os casos de uso do pacote, os relacionamentos entre eles e os atores participantes. (Talvez seja apropriado utilizar mais de um diagrama se o número de casos for grande.)

- **Recomendado:** Descreva os fluxos principal e alternativo de cada caso de uso em seu campo Documentação[4]. (Consulte a **Figura 5-2**.)

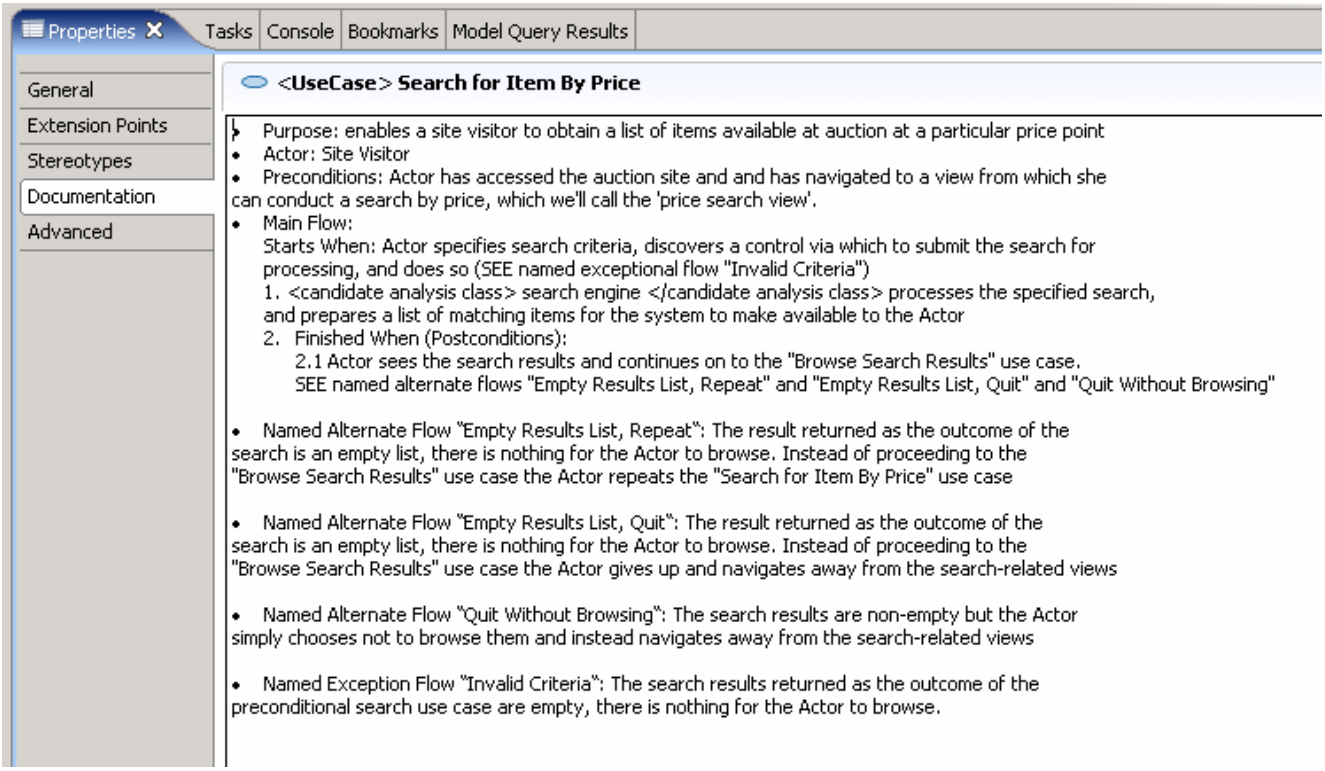



***Figura 5-2***

- **Opcional:** Quando a complexidade de um caso de uso permitir, inclua um diagrama de Atividade e componha-o para refletir os fluxos de atividade gerais do caso de uso. (Consulte a **Figura 5-3**.) **Análise Racional:** Isso ajuda a mostrar as condições correspondentes a cada um dos fluxos (principal e alternativo/excepcional), além de ajudar a assegurar que os vários fluxos sejam, enfim, reconvergidos. (A inclusão de um diagrama de Atividade no RSM/RSA resultará na inclusão automática de uma Atividade no caso de uso, com o diagrama sob a Atividade.)

- **Opcional:** Modele uma realização de 'caixa preta' de cada um dos fluxos nomeados (principal, alternativo e excepcional) do caso de uso; inclua uma ocorrência de colaboração no caso de uso; inclua nele uma instância de interação correspondente ao fluxo principal do caso de uso e uma instância de interação para cada um dos fluxos alternativo e excepcional nomeados; componha um

[4] A formatação representada no exemplo de descrição do caso de uso foi realizada pela criação do 'gabarito' textual para uma descrição de caso de uso, utilizando um editor para RTF e, em seguida, copiando e colando o gabarito no campo de descrição do caso de uso.

diagrama de seqüência (ou então, um diagrama de comunicação) para cada instância de interação. Essas instâncias de colaboração do caso de uso não devem ser confundidas com realizações de casos de uso em nível de análise (conforme descrito no modelo de análise) ou com realizações de casos de uso em nível de design (conforme descrito no modelo de design). Essas são realizações de "caixas brancas" do caso de uso e descrevem interações entre os elementos internos de uma solução. As ocorrências de colaboração propostas aqui para o modelo de caso de uso são estritamente interações de "caixa preta" entre os atores e o sistema. (Consulte a **Figura 5-3**.)
**Análise Racional:** Isso fornece aos investidores não-técnicos uma imagem de alto nível de como os usuários do sistema interagirão com o sistema. Também pode ser útil para identificar as várias visualizações (telas ou páginas) que serão necessárias como parte da implementação. Também estabelece formalmente a nomenclatura dos vários fluxos (cenários) do caso de uso pela designação desses nomes aos elementos do modelo de semântica (isto é, para as ocorrências de colaboração).

> ***XDE/Rose***
> Na UML 1.x, você teria utilizado "Instância de Colaboração", em vez de "Ocorrência de Colaboração" para essa finalidade.

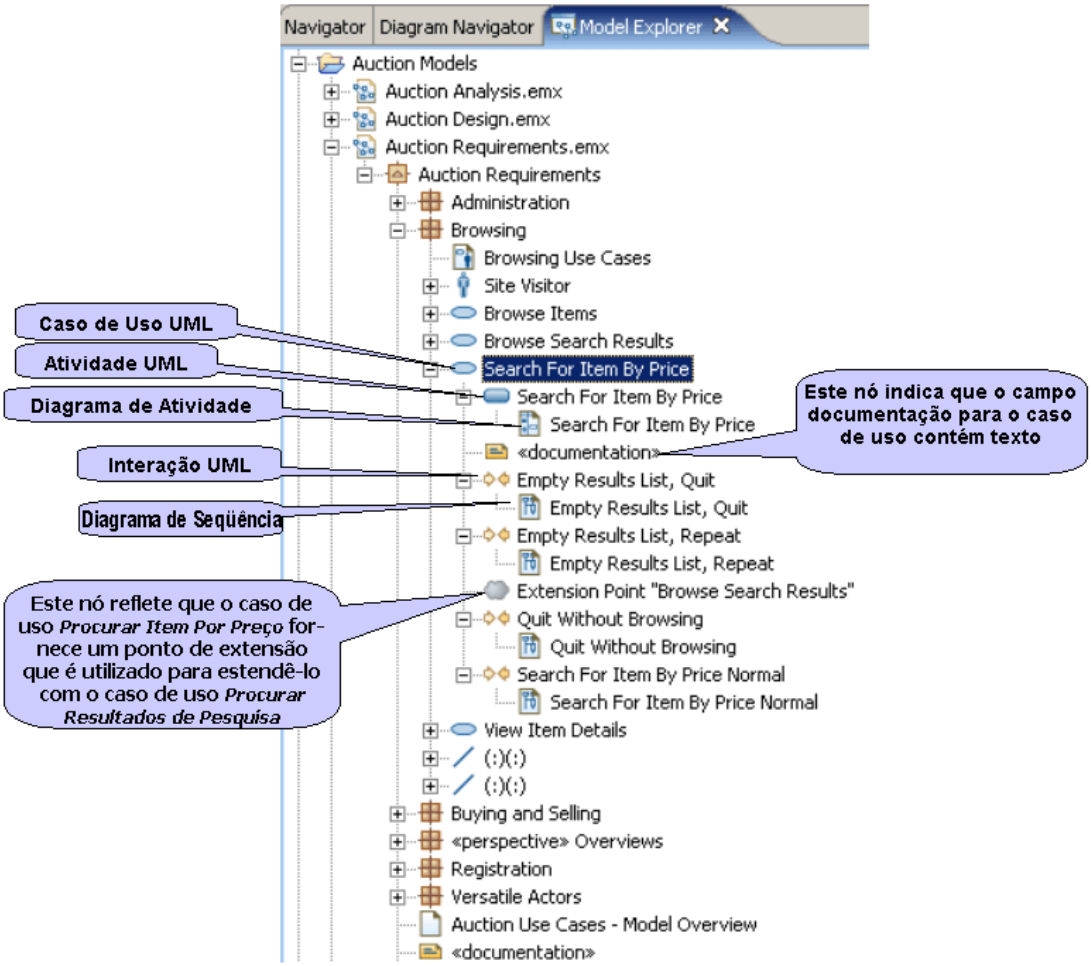



***Figura 5-3***

- **Opcional:** Se você estiver seguindo a orientação do RUP para identificar as visualizações 'significativas do ponto de vista da arquitetura" de sua arquitetura e, especificamente se você for manter um Documento de Arquitetura de Software, inclua um pacote de «perspectiva» de nível superior para conter diagramas de caso de uso que representem os casos de uso significativos do ponto de vista da arquitetura. Talvez você queira dar ao pacote o nome "Visualização do Caso de Uso da Arquitetura".

## 6. Diretrizes para Organização Interna do Modelo de Análise

O modelo de análise representa um 'primeiro recorte' em uma solução. É uma escalada ir dos requisitos até o design final, com foco na captura de informações sobre o domínio de negócios e na apresentação de sugestões de elementos de solução em um alto nível de abstração que esteja próximo dos negócios. É aí que as realizações de casos de uso de classes de análise e de nível de análise residem. É por meio do processo de modelagem de realizações de casos de uso (principalmente a utilização de diagramas de seqüência) que você começa a *descobrir* quais classes são necessárias para a solução— especificamente, essas serão classes correspondentes às linhas de vida que você descobre que precisa nos diagramas de seqüência. Há ainda algumas regras gerais que podem ser aplicadas para sugerir conteúdo do modelo de análise com base no conteúdo do modelo de caso de uso. Esse assunto será tratado posteriormente nesta seção.

No RUP, se um modelo de análise deve ou não ser mantido, independentemente do modelo de design ser uma decisão específica do projeto, um que você criar acreditando no valor de se manter o modelo de análise separado garantirá o tempo investido. Se um modelo de análise separado for criado, mas não mantido, as classes de análise serão movidas para o modelo de design e aperfeiçoadas. Ou talvez o modelo de análise evolua gradualmente para tornar-se o modelo de design[5]. Em termos específicos ao produto, seguem algumas opções que podem ser consideradas:

1. Crie um modelo de análise que resida em um arquivo (ou conjunto de arquivos) de modelagem com base no gabarito do Modelo de Análise. Em seguida, utilize um processo manual ou transformações automatizadas para criar versões aperfeiçoadas dos elementos de análise em um segundo arquivo (ou conjunto de arquivos) de modelo com base no gabarito do Modelo de Design de TI Corporativo e, em seguida, descarte os arquivos de modelagem de análise. Isso permite que você tenha a opção de manter o modelo de análise separado continuamente ou de descartá-lo.
2. Realize uma modelagem em nível de análise em um arquivo (ou conjunto de arquivos) de modelagem com base no gabarito do Modelo de Design de TI Corporativo, ao qual você aplicará o Perfil de Análise. Assim, você pode iniciar a modelagem das realizações de casos de uso utilizando as classes de análise e, então, periodicamente, aperfeiçoá-los para que as interfaces de design assumam as funções nos procedimentos.
3. Um misto da segunda e terceira opções é manter um modelo de análise de classificações no(s) mesmo(s) arquivo(s) de modelagem que o modelo de design. Para isso, seria necessário separar o conteúdo de análise em pacotes nos quais você aplicaria a palavra-chave «analysis». Isso permite a oportunidade de reter artefatos em nível de análise nos mesmos arquivos de modelagem que suas contrapartes mais aperfeiçoadas em nível de design.

[5] O RUP, na verdade, solicita a opção de criar classes de análise e realizações de casos de uso no nível de análise no modelo de design e, em seguida, desenvolve-os diretamente para seus formatos de design a partir daí . Sob essa abordagem, conforme o modelo de design for "descoberto", você poderá criar pacotes durante esse tempo, nos quais preservará algumas das perspectivas de "análise pura".

Um problema com o qual se preocupar ao utilizar transformações do RSA para gerar implementações é que elas podem, em vários casos, aceitar elementos em nível de análise como suas entradas, evitando algumas das etapas de aperfeiçoamento manual desses elementos para os elementos de design. Ao utilizar o RSA dessa maneira, prefira a opção 2 ou 3 anterior. As transformações de geração de código padrão empacotadas como parte do RSA ignorarão os pacotes de modelos que possuem a palavra-chave «analysis».

### Organização de Alto Nível do Modelo de Análise

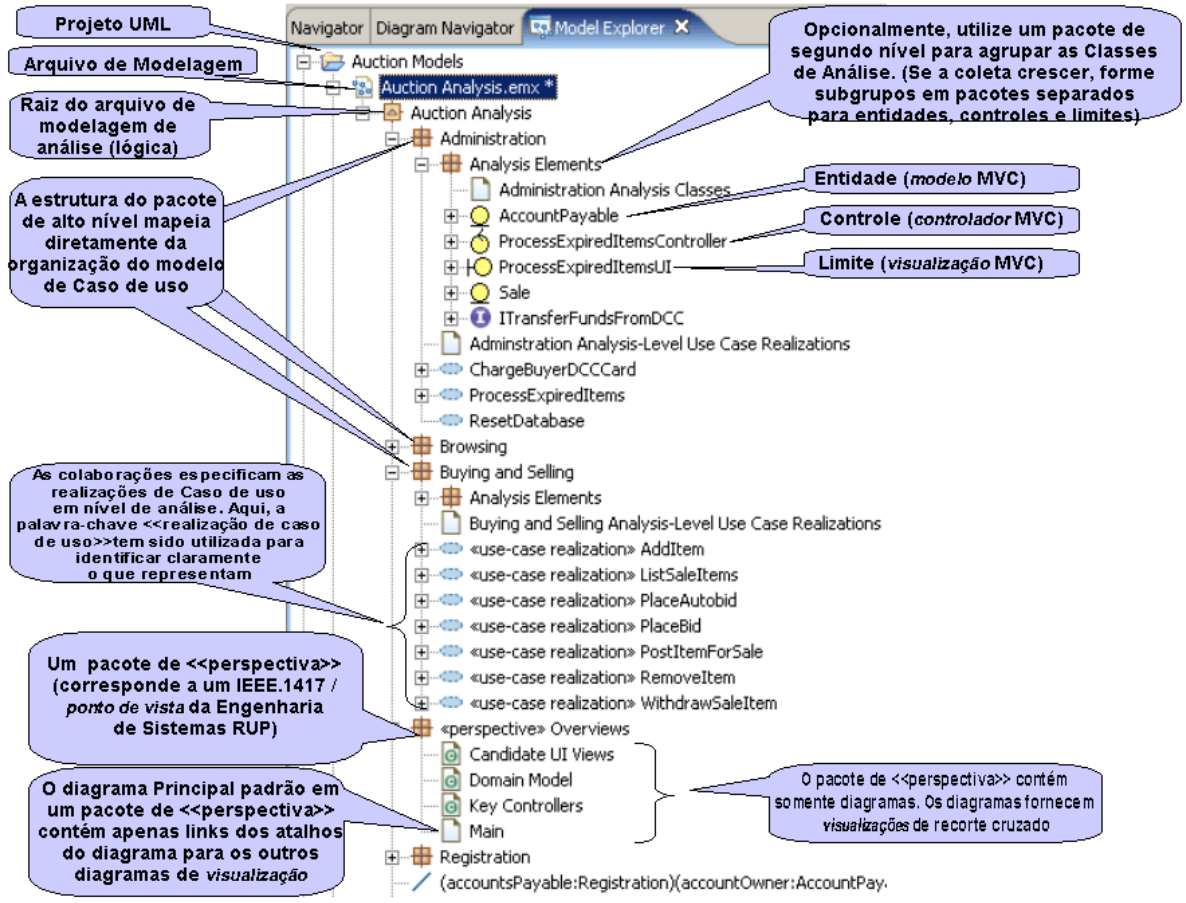



***Figura 6-1***

***Figura 6-1*** : ilustra as diretrizes a seguir para estruturação de modelos de análise:

1. Utilizar pacotes de nível superior para estabelecer os agrupamentos orientados funcionalmente para as classes de análise. **Análise Racional**: mesma análise racional do modelo de caso de uso.

2. Opcionalmente, nos pacotes de nível superior, utilizar subpacotes para coletar e organizar as classes de análise.

3. Utilizar diagramas em pacotes de «perspectiva» para capturar visualizações alternativas, de alto nível ou de recorte cruzado dos elementos de análise. **Análise Racional**: Forneça perspectivas diferentes para investidores diferentes enquanto mantém os elementos de semântica do modelo organizados em agrupamentos orientados funcionalmente.

Uma pequena variação dessa abordagem é representada na **Figura 6-2** que mostra a utilização de um pacote de nível superior para separar as realizações de casos de uso das classes de análise. Nesse pacote de nível superior há um conjunto de subpacotes orientados funcionalmente que correspondem ao conjunto de pacotes de nível superior. Isolar as realizações de casos de uso dessa maneira permite a reformulação da estrutura do pacote que contém as classes de análise, sem *necessariamente* afetar a organização das realizações de casos de uso. (Especificamente se o modelo de análise evoluir para design *em sua posição de origem*, é provável que a organização do pacote para as classes evolua de forma a não mais corresponder à utilizada originalmente para os casos de uso.)

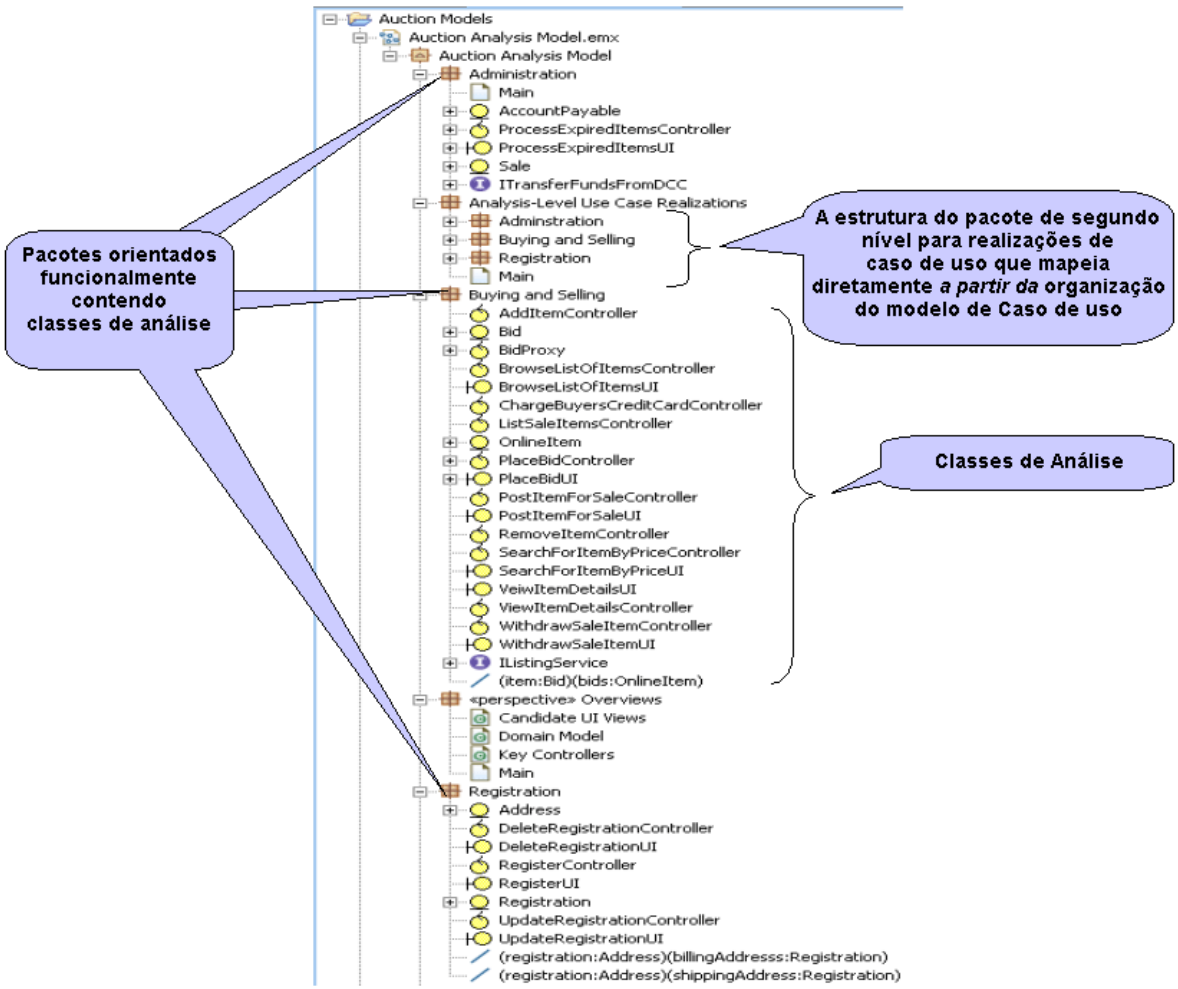



***Figura 6-2***

Dependendo de sua situação, pode haver motivos para introduzir a utilização de uma convenção de nomenclatura que preveja a mesclagem e reutilização de conteúdo de modelo criado por vários grupos independentes, mesmo incluindo grupos de empresas (parceiros) diferentes. Se isso for problema, pense em utilizar uma convenção de espaço de nomes de domínio de Internet invertida

**Figura 6-3**. Observe que, na verdade, isso não é um grande problema para a modelagem de análise *por si só*, mas se você estiver utilizando a abordagem para permitir que o modelo de análise evolua para seu modelo de design *em sua posição de origem* e prevê reutilização ou integração de negócios no nível de design, poderá avançar no planejamento. Outra provável vantagem para se adotar essa abordagem: como ela pode ser mapeada adequadamente para a organização de código gerado a partir da análise/design, ela pode simplificar a configuração subseqüente das transformações de geração de código.

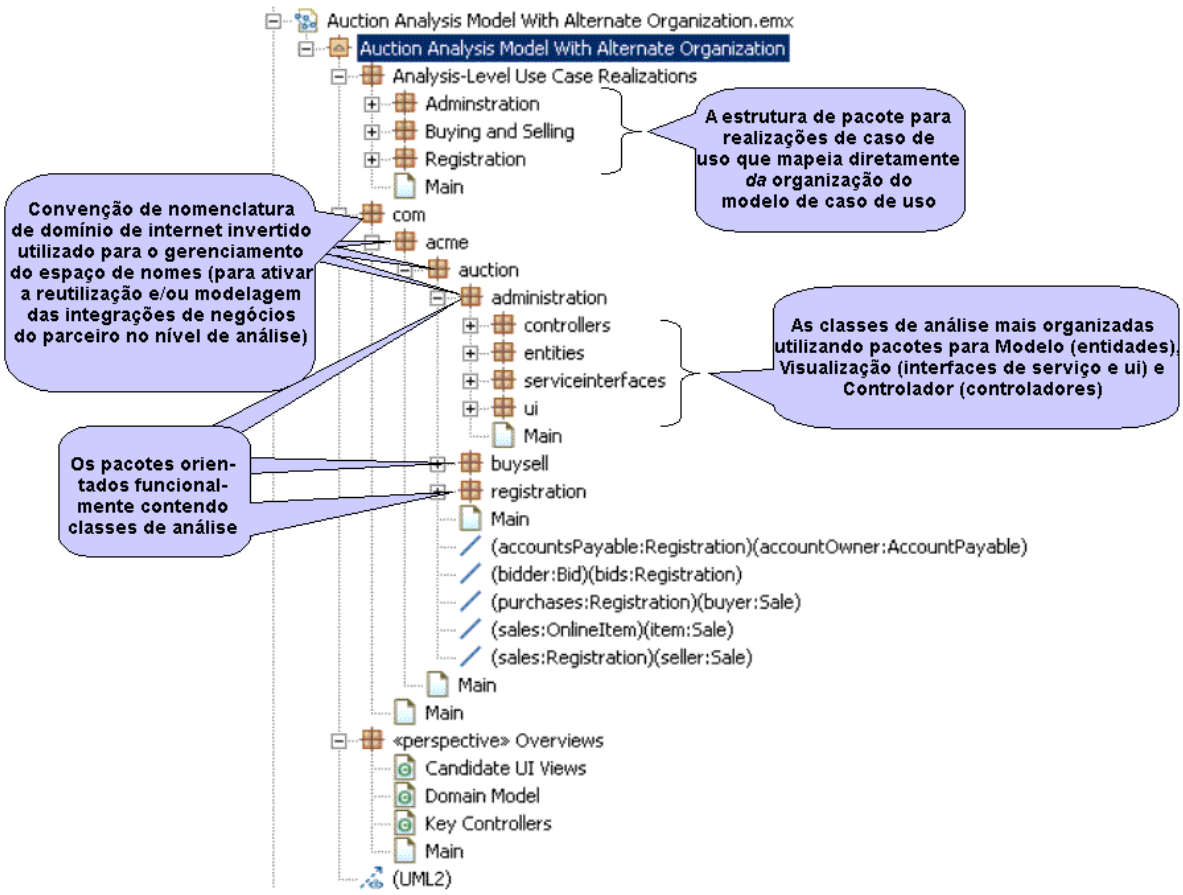



***Figura 6-3***

### Conteúdo do Modelo de Análise

Há várias maneiras de descobrir o que são as classes de análise. Uma delas é iniciar o desenho dos diagramas de seqüência que sugerem realizações de casos de uso. Durante o processo, você descobrirá quais linhas de vida precisa e, geralmente, cada uma delas corresponderá a uma provável classe de análise. Ao descobrir as classes assim, você poderá criá-las nos pacotes de realização de caso de uso do modelo de análise, mas não deverá deixá-las lá. Você deve 'reformular' o modelo para mover as classes de análise para os pacotes orientados funcionalmente, conforme descrito anteriormente nas diretrizes para organização de alto nível do modelo de análise (consulte a ***Figura 6-1***).

Outra abordagem útil para descobrir as classes de análise: 'iniciar' o modelo de análise com classes que se baseiam nestas regras gerais:

- Para cada caso de uso (no modelo de caso de uso), inclua uma classe de «controle» no modelo de análise. As classes de «controle» representam a lógica de negócios associada ao caso de uso. (Mais tarde, no design, elas também serão mapeadas para o gerenciamento de sessões.)

- Para cada relacionamento ator/caso de uso (no modelo de caso de uso), inclua uma classe «limite» no modelo de análise. As classes «limite» representam interfaces entre a solução e um ator humano ou entre a solução e algum sistema externo. É provável que as classes «limite» que correspondem a um ator humano sejam eventualmente mapeadas para um ou mais artefatos de interface com o usuário no design e na implementação. É possível que as classes «limite» que correspondem a um sistema externo sejam eventualmente mapeadas para algum tipo de camada de adaptador no design e na implementação.

- Por meio de um processo como a análise de placa CRC ou a análise de palavras das descrições de caso de uso, identifique classes (verbos) de «controle» e classes (nomes) de «entidade» adicionais.

Ao utilizar essa abordagem inicial para identificar classes de análise, você pode colocar as classes diretamente nos pacotes orientados funcionalmente, conforme descrito anteriormente nas diretrizes para organização de alto nível do modelo de análise (consulte a ***Figura 6-1***).

Por mais que você trabalhe na descoberta de classes de análise, é quase certo que reconheça a necessidade de alterações na organização de seu pacote funcional original.

**Opcional:** Utilize os pacotes de segundo nível dos pacotes da classe de análise para organizar ainda mais o conteúdo deles (consulte a **Figura 6-4**).

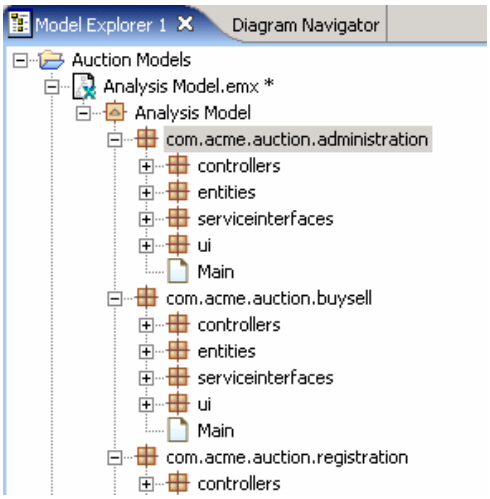



***Figura 6-4***

**Recomendado:** O modelo de análise deve conter realizações de casos de uso em nível de análise**,** que descrevem como os casos de uso são executados com relação às classes de análise. Cada uma das realizações de caso de uso de análise (representadas por uma Colaboração UML) realiza um caso de uso no modelo de caso de uso e tem o mesmo nome desse caso de uso. Consulte a **Figura 6-5**. Para

cada fluxo de caso de uso nomeado[6] que você achar que deveria ser modelado como uma realização em nível de análise, inclua um diagrama de seqüência (que incluirá automaticamente uma Interação própria). **Figura 6-6** : mostra os tipos de conteúdo de semântica que serão incluídos no modelo conforme você cria diagramas de seqüência. (Observe que você pode filtrar qualquer tipo de elemento UML a partir da visualização do Explorer de Modelo e, assim, ocultar boa parte da 'confusão' representada na **Figura 6-6**.)

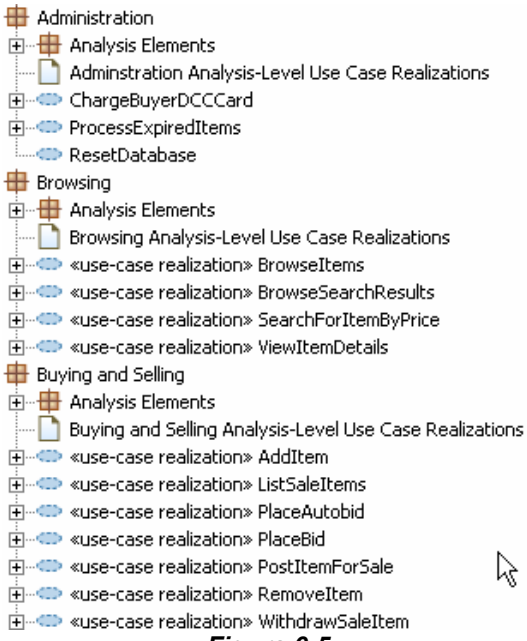



***Figura 6-5***

**Opcional:** Depois de ter criado o diagrama de seqüência para um fluxo de caso de uso, você poderá selecionar sua Interação UML própria no Explorer de Modelo e incluir um Diagrama de Comunicação nele. O novo Diagrama de Comunicação será preenchido automaticamente pelas instâncias de classe de análise que participaram do diagrama de seqüência.

**Recomendado:** Crie um relacionamento de dependência de Realização a partir de cada realização de caso de uso (Colaboração UML) e o caso de uso correspondente a partir do modelo de caso de uso (consulte a **Figura 6-6**). Como é possível utilizar recursos como Diagramas de Tópico e Análise de Rastreabilidade para entender os relacionamentos de rastreabilidade em seu modelo, você não precisa realmente reter diagramas permanentes para representar os relacionamentos de rastreabilidade, portanto, recomenda-se criá-los utilizando algum tipo de diagrama 'descartável', por exemplo:

- Inclua um diagrama de formato livre na Colaboração.
- Arraste a Colaboração até ele.
- Arraste o caso de uso até ele.
- Desenhe o relacionamento de dependência.
- Finalmente (no Explorer de Modelo), exclua o diagrama da Colaboração.

**Recomendado:** Inclua um diagrama de "Participantes" para cada realização de caso de uso para mostrar as classes de análise que participam da realização (isto é, as classes de análise cujas instâncias aparecem nos diagramas de interação que descrevem a realização do caso de uso) e os

[6] Conforme estabelecido anteriormente no Modelo de Caso de Uso

relacionamentos entre essas classes que suportam a colaboração descrita nos diagramas de interação. Consulte a **Figura 6-6**.

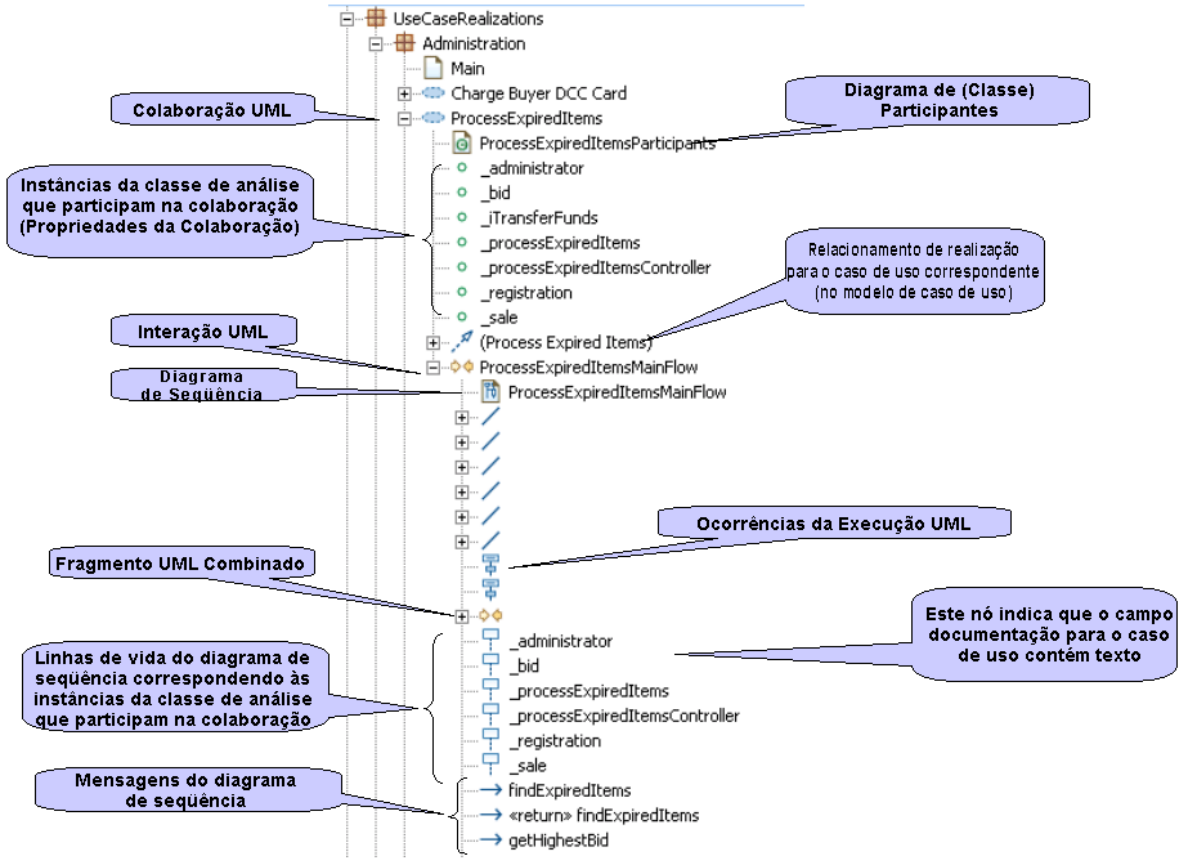
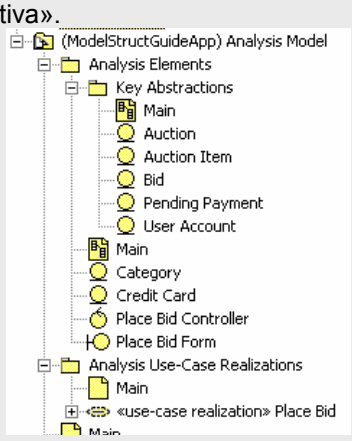



***Figura 6-6***

***XDE/Rose***

A estrutura tradicionalmente recomendada para o **Modelo de Análise** conforme mostrada a seguir, é modificada para que o RSA enfatize uma organização de pacotes orientada funcionalmente para as classes de análise. Observe também que a utilização de um pacote de Abstrações de Chaves (que poderia comprometer uma abordagem de empacotamento orientado funcionalmente de outra forma) é substituída pela utilização de um diagrama (ou diagramas) de Abstrações de Chaves em um pacote de «perspectiva».



# 7. Diretrizes para Organização Interna do Modelo de Design

### Organização de Alto Nível do Modelo de Design

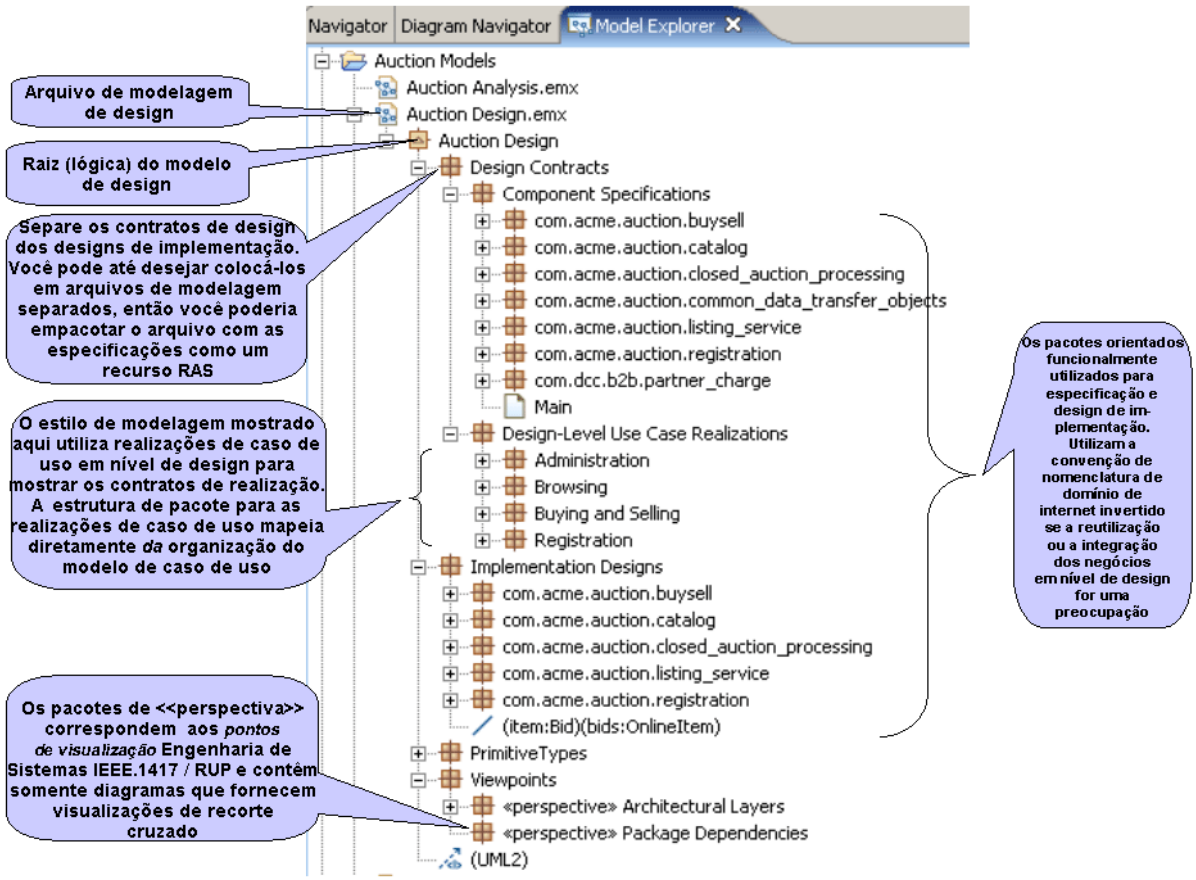



***Figura 7-1***

**Figura 7-1** : ilustra as diretrizes a seguir para estruturação de modelos de design:

1. Separar especificações a partir de designs de implementação. A ilustração mostra a utilização de pacotes de “Contratos de Design” e “Designs de Implementação” de nível superior para esta realização.

2. Utilizar pacotes de nível inferior para estabelecer agrupamentos orientados funcionalmente. Você pode, por exemplo, iniciar com a organização utilizada durante a análise e deixá-la evoluir enquanto toma decisões sobre como as classes de análise serão mapeadas para as classes de design, componentes e serviços reais. (É provável que algum esquema organizacional inicial evolua durante o design—consulte discussão adicional a seguir).

   Uma palavra sobre subsistemas é adequada neste ponto. Em versões de UML anteriores à 2, um subsistema é um tipo especializado de pacote. Na UML 2, um subsistema é um tipo especializado de componente e um Componente pode conter pacotes. Na UML 2, Componentes de «subsistema» são

alternativas organizacionais/de espaço de nomes viáveis para os pacotes; a UML2 ainda é indefinida sobre a utilização apropriada de subsistema versus pacote. Sugestão: Utilize Pacotes em níveis de granularidade, como os subsistemas de design de um determinado aplicativo e reserve os Subsistemas para representar aplicativos inteiros (por exemplo, CRM ou SCM) em visualizações de arquitetura em todo o corporativo.

> ***XDE/Rose***
> Na hora dessa escrita, esperava-se que as ferramentas de importação do modelo Rose e XDE oferecessem a opção de mapear subsistemas UML 1.x para Subsistemas UML2 ou para pacotes com a palavra-chave «subsystem» aplicada.

3. É provável que a organização de elementos de design evoluirá fora da forma de organização dos casos de uso do sistema (no modelo de caso de uso e, talvez, no modelo de análise, se um modelo de análise separado estiver sendo mantido). Utilize pacotes para subdividir ainda mais os contratos de design nas especificações de elementos de design (os contratos de uso) e as realizações de casos de uso em nível de design (os contratos de realização) e manter uma subestrutura de pacotes para as realizações de casos de uso que continuam a espelhar a organização dos próprios casos de uso.

4. *Considerar* a utilização de camadas de arquitetura como base para o esquema organizacional de segundo nível para os elementos que compõe as especificações e os designs de implementação das áreas funcionais (e consultar discussão adicional a seguir).

5. Nos componentes e pacotes de UML que agrupam elementos de modelo de semântica, coloque diagramas que forneçam visualizações específicas a esse agrupamento. Essa diretriz refere-se ao fato desse agrupamento basear-se nos subconjuntos de domínio de negócios orientados funcionalmente, em uma camada de arquitetura ou no que você tiver. Faça com que o diagrama 'padrão' tenha o mesmo nome do próprio pacote ou componente e componha-o para mostrar uma visão geral do conteúdo do pacote. Isso mantém alguns diagramas próximos do que representam, facilitando a navegação e entendimento do modelo.

6. Talvez você queira introduzir a utilização de um espaço de nomes de domínio de Internet invertido no modelo de design. **Análise Racional**:
   - Basicamente, os mesmos motivos de que fazer isso é importante com relação a implementações específicas do idioma:
     a. cenários envolvendo trabalho de integração onde haja vários aplicativos orientados ao modelo envolvidos (especialmente com empresas parceiras)
     b. reutilização de cenários
   - Isso provavelmente simplificará a configuração subseqüente de transformações em implementação (mapeamento de local e nome da origem para o destino).

7. *Considerar* a utilização de nomes de pacotes que serão válidos na(s) plataforma(s) de implementação de destino para evitar a carga e a provável confusão de mapeamento de espaço de nomes. (Na maioria das vezes, isso significa apenas "não utilizar espaços ou pontuação, exceto sublinhados, nos nomes".)

8. Utilizar minúsculas para os nomes de pacotes para facilitar distingui-los dos nomes de classes em um pacote.

9. *Considerar* a utilização de nomes diferentes para as Interfaces e para os Componentes ou Classes que os realizam. Utilize ILoan e Loan ou Loan e LoanImpl para os nomes de interfaces e

implementações. Isso realmente não é necessário no modelo, mas muitas vezes é interessante para o código gerado, portanto, esta é outra área na qual você pode poupar a si mesmo algum trabalho subseqüente de configuração de transformação.

10. No cenário a seguir, todo conteúdo em nível de análise a partir do qual não se pretenda gerar código deve ser separado nos pacotes estereotipados como de «análise»[7].

    - A) Você escolheu ignorar a utilização de um modelo de análise separado e preencher o modelo de design com conteúdo em nível de análise e manter esse conteúdo no nível de análise de abstração enquanto também cria conteúdo em nível de design no mesmo modelo e

    - B) Você estará tratando de transformações modelo para código a partir do Modelo de Design EIT.

11. Utilizar diagramas em pacotes de «perspectiva» para capturar visualizações de recorte cruzado de alto nível dos elementos de design. **Análise Racional**: Forneça visualizações de recorte cruzado, visualizações de conteúdo 'significativos do ponto de vista da arquitetura' e visualizações que atraiam tipos diferentes de investidores enquanto mantém os elementos de semântica do modelo organizados em agrupamentos orientados funcionalmente.

É importante reconhecer que as estruturas de empacotamento dos modelos de design evoluirão periodicamente. A organização deve corresponder, enfim, à forma como você estrutura sua arquitetura em componentes e serviços. Esta abordagem para a organização da *atividade final* do design geralmente propiciará o melhor potencial para empacotamento de recursos reutilizáveis e o mapeamento mais direto do design para o conjunto de projetos e pastas que conterão artefatos de implementação (código, metadados, documentação) gerados a partir do design.

Entretanto, a organização *inicial* deve corresponder mais ou menos diretamente à abordagem organizacional utilizada para o modelo de caso de uso e, em seguida, ser revisada durante a análise[8]. Na verdade (conforme descrito na seção anterior "Diretrizes para Organização Interna do Modelo de Análise"), você pode escolher deixar seu modelo de análise evoluir para o design na posição de origem. Ou seja, a organização inicial do design tenderá a agrupar conjuntos de questões de negócios coesos e desconectados e isolar os elementos de recorte cruzados ou reutilizáveis. Esta abordagem para a organização inicial prova ser eficaz porque:

- Se você espera utilizar transformações que gerem conteúdo de modelo de design a partir do conteúdo do modelo de análise ou de caso de uso, os mapeamentos do pacote de origem para o pacote de destino serão simples e diretos.

- Uma abordagem organizacional inicial com base na coesão funcional e na desconexão dos pacotes indica claramente a melhor oportunidade de mapeamento para a organização final orientada pelo componente, significando que deve reduzir a quantidade de reformulação que precisará ser feita como parte do processo de design.

- A desconexão de pacotes tem o potencial de aprimorar os workflows das equipes e facilitar a reutilização nos casos em que o design é formulado em vários arquivos de modelagem.

As abordagens alternativas são possíveis, naturalmente, e em alguns casos, aconselháveis como uma

[7] Esses pacotes serão ignorados pelas transformações.

[8] O empacotamento das classes de análise é, muitas vezes, reformulado significativamente assim que é descoberto, para suportar melhor a reutilização e os requisitos funcionais imprevistos.

organização de *atividade final*:

- Se você estiver direcionando aplicativos da Web com base no J2EE, incluindo os EJBs, a organização do design poderá prever as convenções do RSA e do Rational Application Developer em relação aos projetos J2EE.[9] Especificamente, você poderia escolher definir os pacotes de design de nível superior correspondentes às camadas de arquitetura (apresentação e negócios, com os negócios divididos em subcamadas na sessão e no domínio). Obviamente, esta não é uma abordagem com plataforma neutra e, assim, é aconselhável apenas se você souber que a solução que está projetando não será implementada em uma plataforma que não a J2EE.

- Em geral, muitas vezes é o caso quando aplicativos com n camadas estão sendo construídos que o conhecimento do desenvolvedor e a divisão do trabalho correspondem às camadas de apresentação e de negócios, portanto, novamente você pode escolher utilizar pacotes de nível superior correspondentes àquelas camadas de arquitetura. Mas, cuidado ao organizar classes destinadas a suportar determinadas *funções de negócios* para suportar uma determinada *arquitetura*. É ainda mais difícil alterar.

- Se você achar motivo para utilizar uma abordagem organizacional orientada a não-componentes/serviços/subsistema, ainda será possível mapear a organização do design para um conjunto de projetos e pastas de destino, investindo algum esforço adicional na configuração de transformações de geração de código. Um tipo especial de modelo parceiro referido como 'modelo de mapeamento' pode ser utilizado para definir mapeamentos particularmente complexos.

**Conteúdo do Modelo de Design**

Não há regras rigorosas para o que deve residir no modelo de design, mas as sugestões a seguir podem ser úteis.

[9] Em termos gerais: Um Projeto Corporativo por sistema ou aplicativo ou subsistema grande e para cada Projeto Corporativo, um projeto da Web para a camada de apresentação e vários projetos EJB nos quais os projetos EJB correspondem geralmente aos componentes ou subsistemas secundários e nos quais normalmente projetos EJB separados são utilizados para a camada de sessão (EJBs de sessão) e de domínio (EJBs de entidade) por componente ou subsistema. Consulte a seção 9 deste white paper para obter informações adicionais.

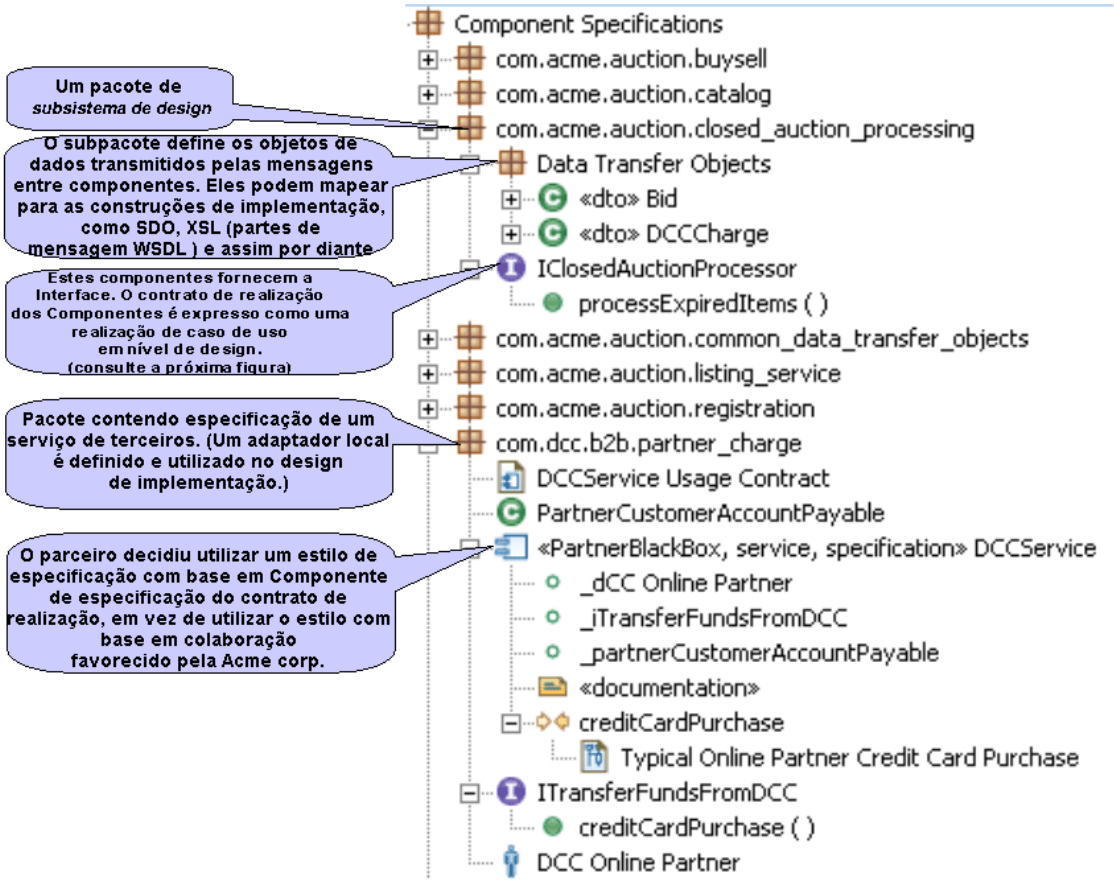

Um pacote de subsistema de design
O subpacote define os objetos de dados transmitidos pelas mensagens entre componentes. Eles podem mapear para as construções de implementação, como SDO, XSL (partes de mensagem WSDL ) e assim por diante
Estes componentes fornecem a Interface. O contrato de realização dos Componentes é expresso como uma realização de caso de uso em nível de design. (consulte a próxima figura)
Pacote contendo especificação de um serviço de terceiros. (Um adaptador local é definido e utilizado no design de implementação.)
O parceiro decidiu utilizar um estilo de especificação com base em Componente de especificação do contrato de realização, em vez de utilizar o estilo com base em colaboração favorecido pela Acme corp.
Component Specifications
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
Data Transfer Objects
«dto» Bid
«dto» DCCCharge
IClosedAuctionProcessor
processExpiredItems ( )
com.acme.auction.common_data_transfer_objects
com.acme.auction.listing_service
com.acme.auction.registration
com.dcc.b2b.partner_charge
DCCService Usage Contract
PartnerCustomerAccountPayable
«PartnerBlackBox, service, specification» DCCService
_dCC Online Partner
_iTransferFundsFromDCC
_partnerCustomerAccountPayable
«documentation»
creditCardPurchase
Typical Online Partner Credit Card Purchase
ITransferFundsFromDCC
creditCardPurchase ( )
DCC Online Partner

***Figura 7-2***

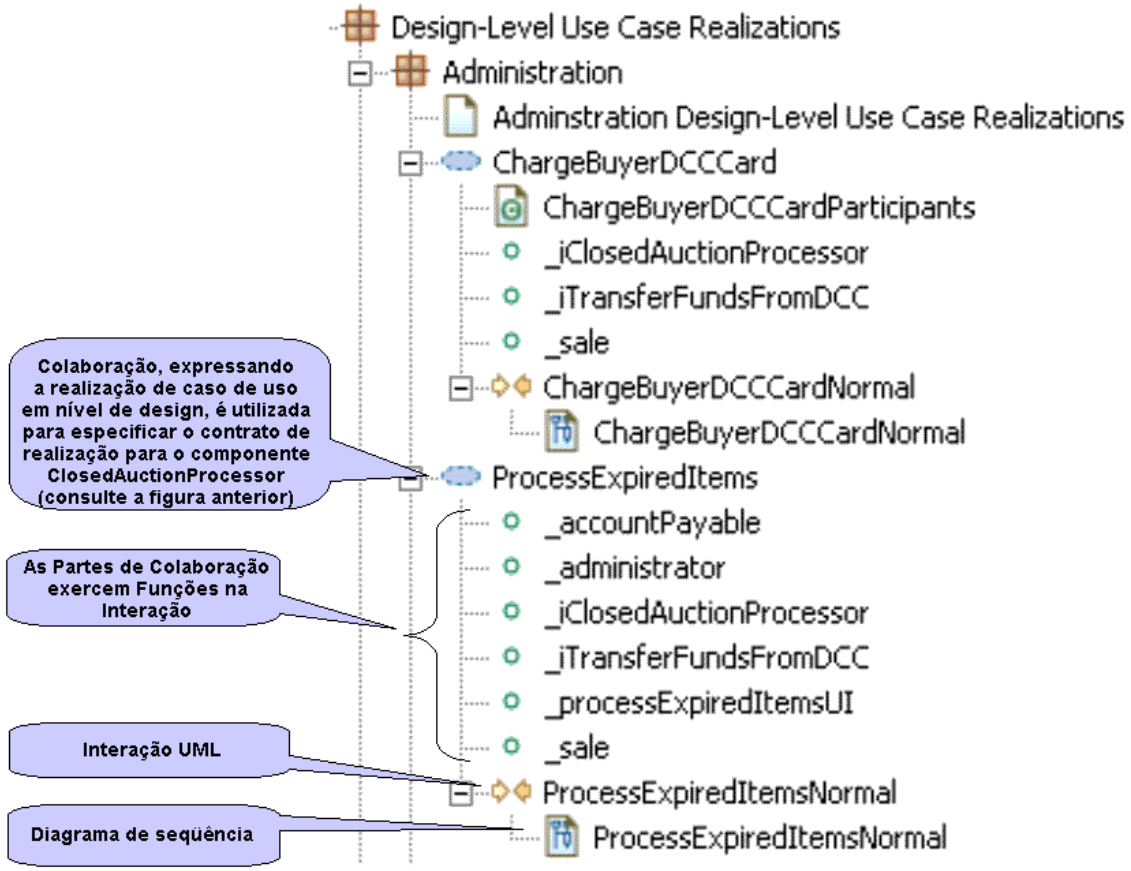



***Figura 7-3***

**Figura 7-2** e **Figura 7-3** : seguem a estrutura organizacional representada na **Figura 7-1** e descrevem como os contratos de design podem ser especificados.

- O contrato de uso para um componente “ClosedAuctionProcessor” é expresso como uma única Interface[10] (**Figura 7-2**). O contrato de realização correspondente é especificado por uma única realização de caso de uso em nível de design, expresso como Colaboração[11] (**Figura 7-3**) Observe que, enquanto as realizações de caso de uso em nível de análise mostram colaborações entre as classes de análise, as realizações em nível de design mostram colaborações entre os elementos de design menos abstratos[12]. Se for desejado que o subconjunto de especificação de um modelo de design seja empacotado independentemente do subconjunto de design de implementação, então é importante que as realizações de caso de uso em nível de design utilizem apenas elementos de especificação de análise ou de design—nunca

[10] Os componentes podem, naturalmente, ter várias interfaces fornecidas, mas este exemplo tem apenas uma.

[11] Outros componentes podem participar de vários casos de uso do sistema, portanto, seus contratos de realização podem residir em várias realizações de caso de uso. Nesses casos, você também poderia incluir no mesmo pacote com a Interface do componente um diagrama chamado “{nome do componente} Where-Used” no qual colocará links para os vários diagramas que compõe as realizações de caso de uso desses casos de uso.

[12] Outra provável diferença: Alguns dos diagramas “participantes” nas realizações em nível de design podem ser Diagramas de Componente que representam conexão física do componente, em vez de (ou além de) Diagramas de Classe participantes, conforme sugerido para as realizações de caso de uso em nível de análise.

elementos de design de implementação—em suas funções.

- Os contratos de uso e realização para o "DCCService" de terceiros estão juntos em um pacote[13]. Mais uma vez, vemos que o contrato de uso consiste em uma única Interface, mas, neste caso, o contrato de realização é expresso utilizando um Componente de «especificação» (**Figura 7-2**). (Caso contrário, a especificação do contrato de realização seria a mesma coisa, expressa com a utilização de procedimentos—neste caso, uma Interação chamada "creditCardPurchase"). Outro exemplo que utiliza Componentes, em vez de Colaborações, é mostrado na **Figura 7-4**.

- As operações são definidas nas interfaces, que podem ser realizadas por componentes de «especificação» (se utilizados) ou por classificadores no Design de Implementação que implementa as interfaces.

- A especificação de objetos de transferência de dados (que serviria como os tipos de parâmetros das operações fornecidas e poderiam ser mapeados para construções de implementação, como esquema XML ou SDOs) também pode ser incluída como parte do contrato de uso. Para componentes que não são projetados para serem distributivos, você poderia ou não escolher especificar objetos de transferência de dados como as especificações dos tipos utilizados como parâmetros da operação. Para serviços distributivos (por exemplo, Serviços da Web), é obrigatório que as operações do serviço não referenciem objetos em um espaço de endereçamento local, portanto, deve-se utilizar DTOs.

> ***XDE/Rose***
> Em versões anteriores da UML, a orientação para as realizações de caso de uso era utilizar uma Instância de Colaboração por caso de uso e uma interação e um diagrama de seqüência para cada um dos fluxos significativos da realização.
>
> No Rational Software Architect, você deve sempre conseguir utilizar apenas uma interação e um diagrama porque os diagramas de seqüência UML2 suportam agora notações para caminhos de execução alternativos.
>
> Além disso, na UML 2 não há mais uma 'Instância de Colaboração'. Em vez disso, há 'Utilização de Colaboração', que requer uma Colaboração como seu tipo. Portanto, no Rational Software Architect, utilize Colaborações para representar realizações de caso de uso.

- [13] Observe que a situação hipotética aqui é que a empresa DCC forneceu à empresa Acme a especificação UML, com a Acme então incorporada em seu modelo de design. Esse é o tipo de cenário no qual a utilização de espaços de domínio de Internet invertidos poderia ser conveniente.

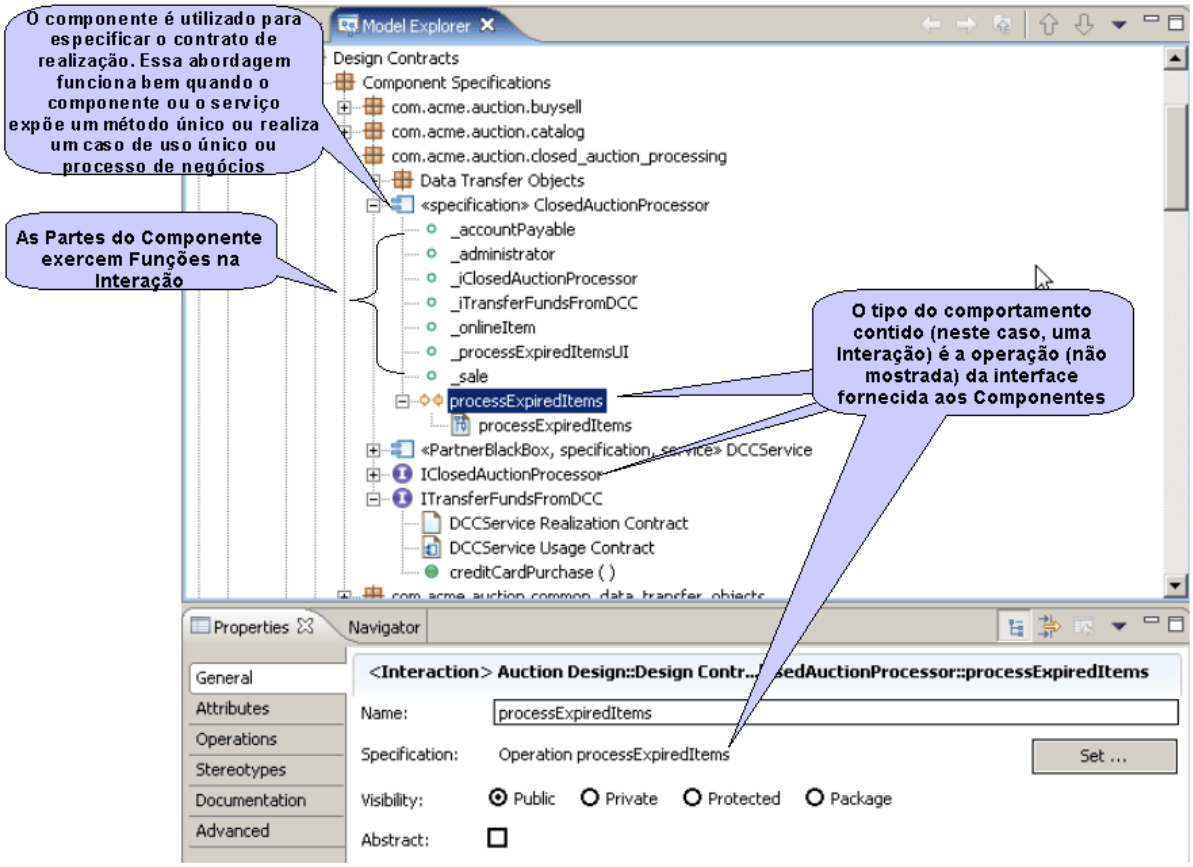



***Figura 7-4***

- Uma possível abordagem para especificar designs de implementação é mostrada na **Figura 7-5**. A estrutura da implementação é definida utilizando classes simples que contêm operações. Esta abordagem é bem típica de modelos de design criados com a UML 1.x. Uma segunda possível abordagem que poderia estar mais de acordo com os objetivos da UML2 é mostrada na **Figura 7-6**. Aqui, em vez de Classes, vemos que Componentes são utilizados e que os Componentes não possuem Operações e sim procedimentos próprios (neste caso, uma Interação).

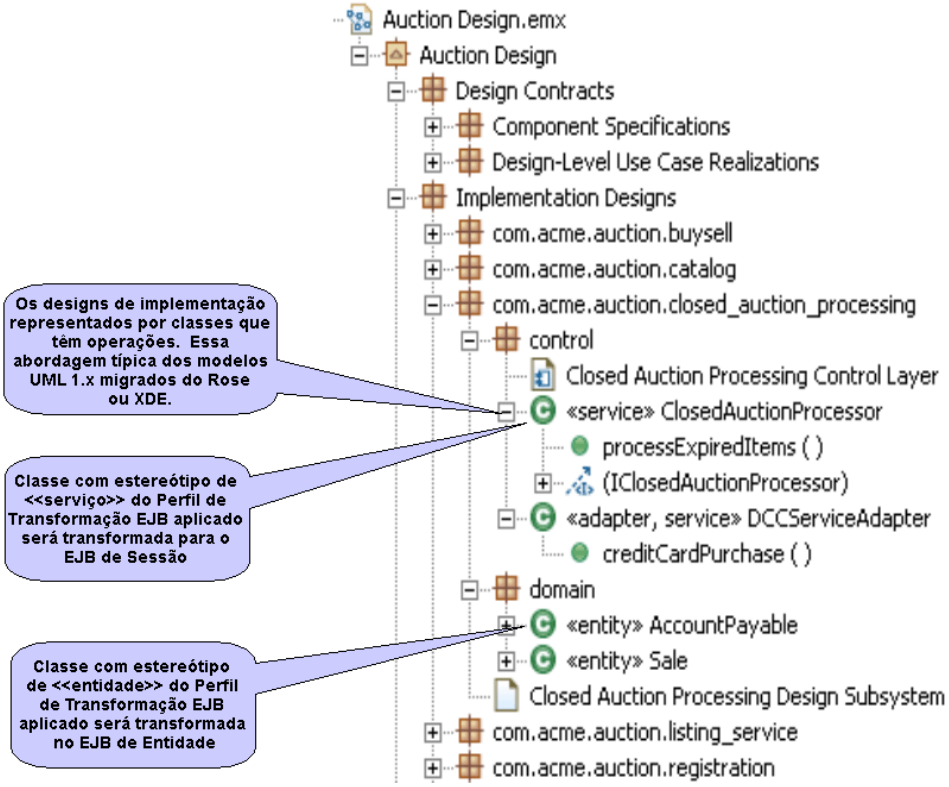



***Figura 7-5***

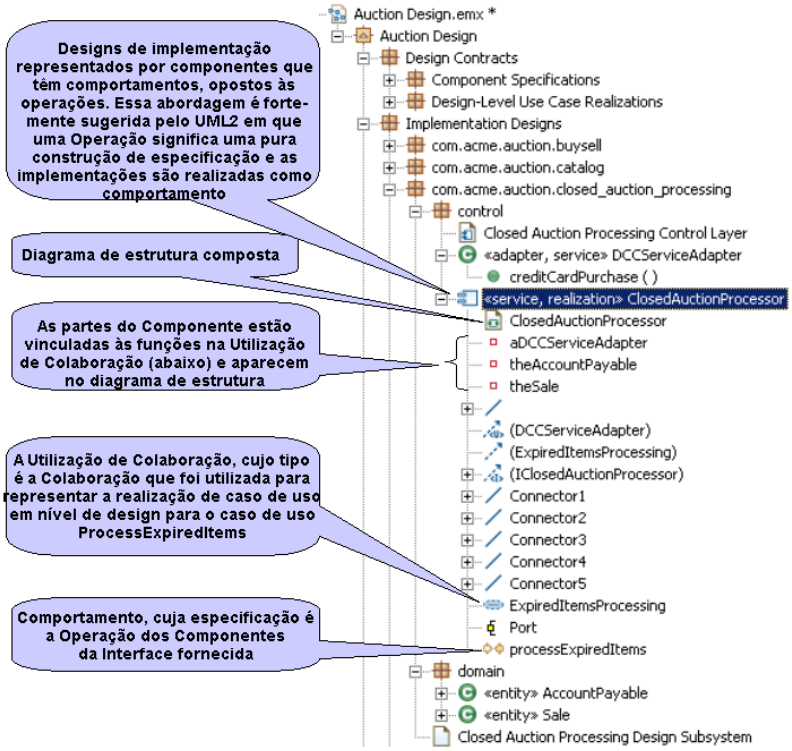



***Figura 7-6***

## 8. Diretrizes para Organização Interna do Modelo de Visão Geral da Implementação

> ***XDE/Rose***
> Nas Diretrizes da Estrutura do Modelo XDE, um modelo de Visão Geral da Implementação era recomendado como dispositivo para fornecer uma visão geral da implementação em nível de subsistema. Os detalhes de cada subsistema eram então especificados no modelo de código do projeto que implementou o subsistema.

Para ser mais exato, não deveria ser necessário utilizar um modelo de Visão Geral da Implementação no RSA. Se as diretrizes organizacionais do modelo de design forem seguidas, a organização (atividade final) do modelo de design deverá naturalmente tomar forma em torno dos componentes (incluindo o «subsistema» mais significativo e as variedades de «serviço» mais distributivas). Em seguida, por meio de transformações, os pacotes do design poderiam ser mapeados para os projetos. Por exemplo, no caso de uma implementação J2EE, eles serão mapeados para os vários projetos Java, EJB, Web, J2EE Application e outros, nos quais a implementação será desenvolvida. (E esses projetos representam realmente o modelo de implementação para a solução, conforme observado na seção Conceitos Básicos e Terminologia deste white paper.)

Entretanto, você ainda pode preferir esboçar a estrutura de seu projeto em um estágio antecipado ou pode simplesmente preferir ver uma representação de estruturas de projetos que seja mais explícita visualmente—por exemplo, uma na qual os artefatos que representam projetos e pastas tenham explicitamente a palavra chave «project» e «folder» ou até mesmo «EJB Project» e «Web Project». Outra consideração é que representar artefatos de implementação minuciosos (JARs, por exemplo) seria inapropriado para o modelo de design (que, na teoria de operação do Rational Software Architect, pretende ser neutro em relação à plataforma). Mas esses artefatos são perfeitamente aceitáveis para inclusão em um modelo de Visão Geral da Implementação. Sendo assim, há alguns motivos pelos quais você desejará utilizar o modelo de Visão Geral da Implementação. **Figura 8-1** : representa um modelo de Visão Geral da Implementação

Uma idéia final é que um modelo de Visão Geral da Implementação pode ser um local adequado para capturar diagramas informais de vários aspectos da solução. **Figura 8-2** : mostra um diagrama informal de alto conceito do sistema de leilão no qual a maior parte das amostras deste white paper se baseia.

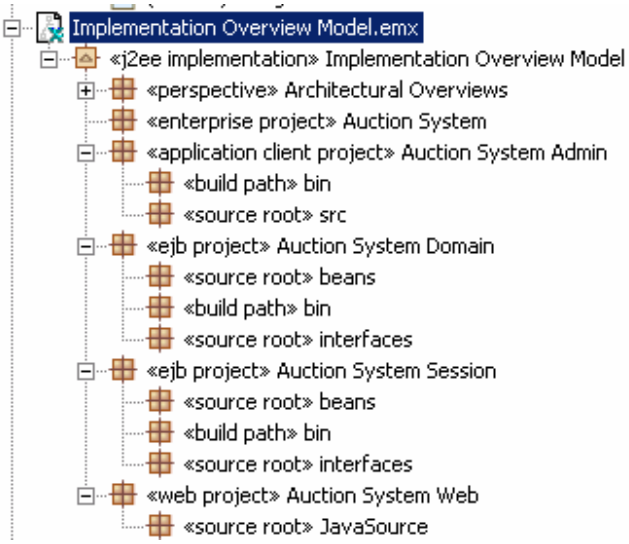



***Figura 8-1***

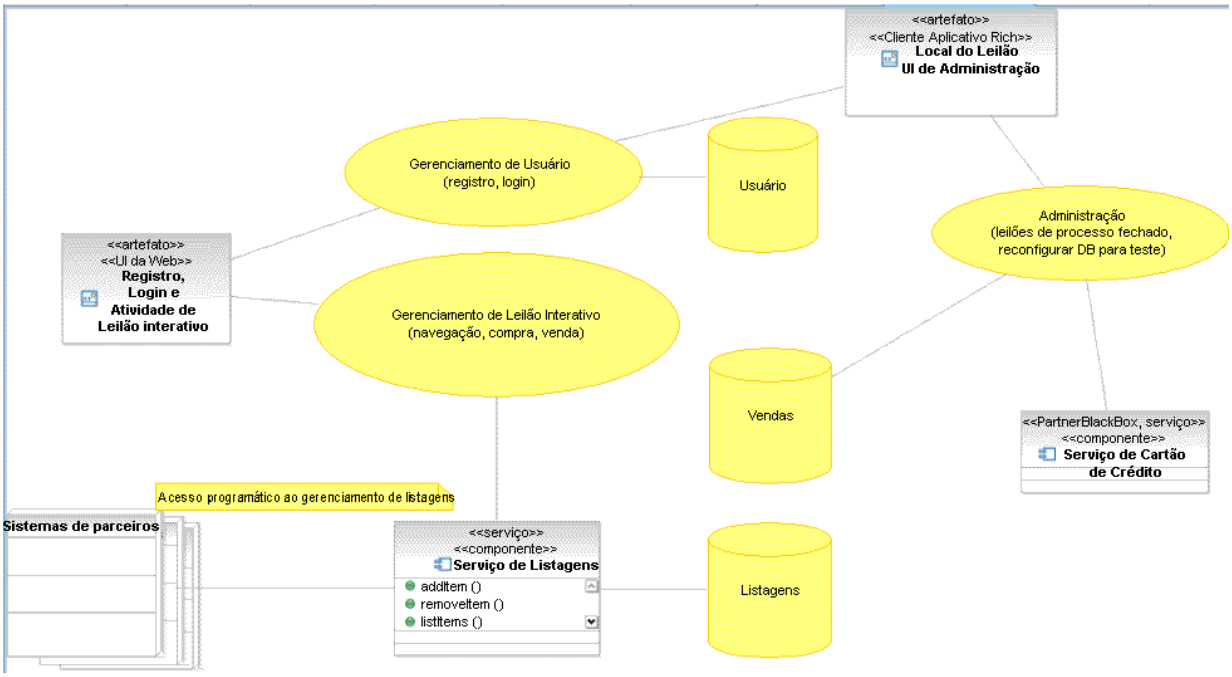



***Figura 8-2***

# 9. Diretrizes para Organização Interna do Modelo de Implantação

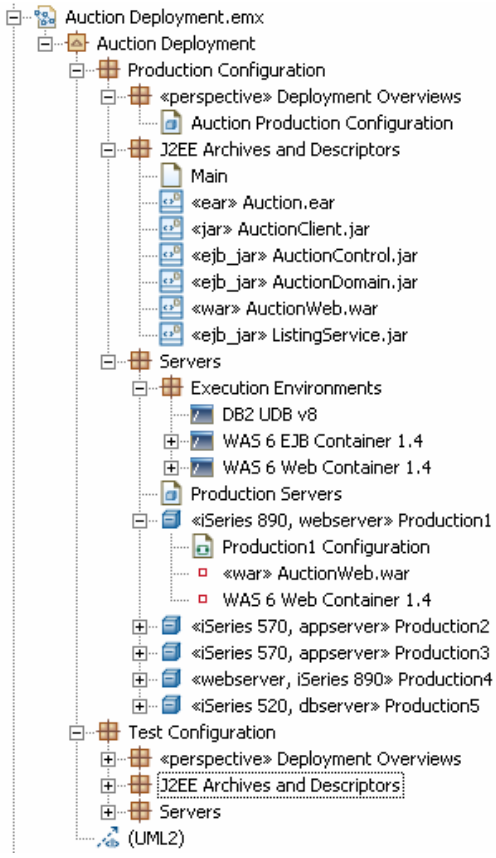



***Figura 9-1***

É provável que menos precise ser dito sobre o modelo de implantação do que sobre qualquer um dos outros modelos tratados aqui. Normalmente, deve haver muito poucas implicações no recebimento de dados de sua organização de modelagem de implantação e opções de conteúdo, portanto, faça apenas o que fizer sentido. Mesmo assim – só para mantê-lo pensando – uma possível estratégia e um pouco de conteúdo representativo são representados acima na **Figura 9-1**. Neste exemplo, observe:

1. As especificações de configurações de produção foram separadas daquelas de configurações de teste.

2. Visões gerais (por exemplo, de clusters, de centros de dados ou de corporativos) são mantidas em pacotes de «perspectiva».

3. Uma abordagem mais reduzida foi obtida com relação à especialização e classificações de nós e artefatos: uma combinação de empacotamento e a utilização de palavras-chave. Uma abordagem mais sofisticada seria desenvolver um perfil especializado de UML que definisse estereótipos especializados e propriedades apropriadas para descrever e documentar os tipos de recursos utilizados em seu próprio ambiente.

## 10. Utilizando um Arquivo de Modelagem para Representar o Documento de Arquitetura de Software

Dadas suas ferramentas para organizar modelos, como links de diagrama e suporte para vários arquivos de modelo com referências de modelo cruzado, torna-se um assunto quase trivial criar um modelo que, efetivamente, represente o Documento de Arquitetura de Software do RUP e as “4+1 Visualizações da Arquitetura”.

Na forma mais simples, seria possível fazer alguma coisa ao longo das linhas da **Figura 10-1**. Criar um arquivo de modelagem e preenchê-lo com um conjunto simples de pacotes correspondentes às 4+1 Visualizações. (O exemplo é mostrado sem um pacote para a Visualização de Processo, uma vez que o sistema nesse exemplo não exibe muito no caminho da coincidência.)

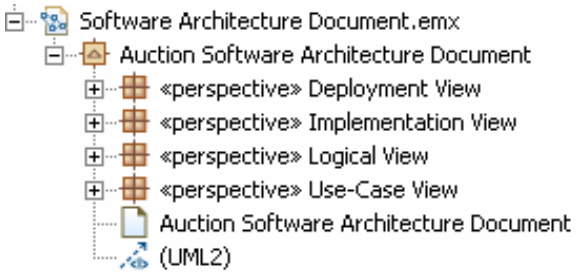



***Figura 10-1***

Depois, talvez, compor o diagrama padrão ao longo das linhas sugeridas na **Figura 10-2**. Também seria possível incluir notas adicionais ou texto neste diagrama.

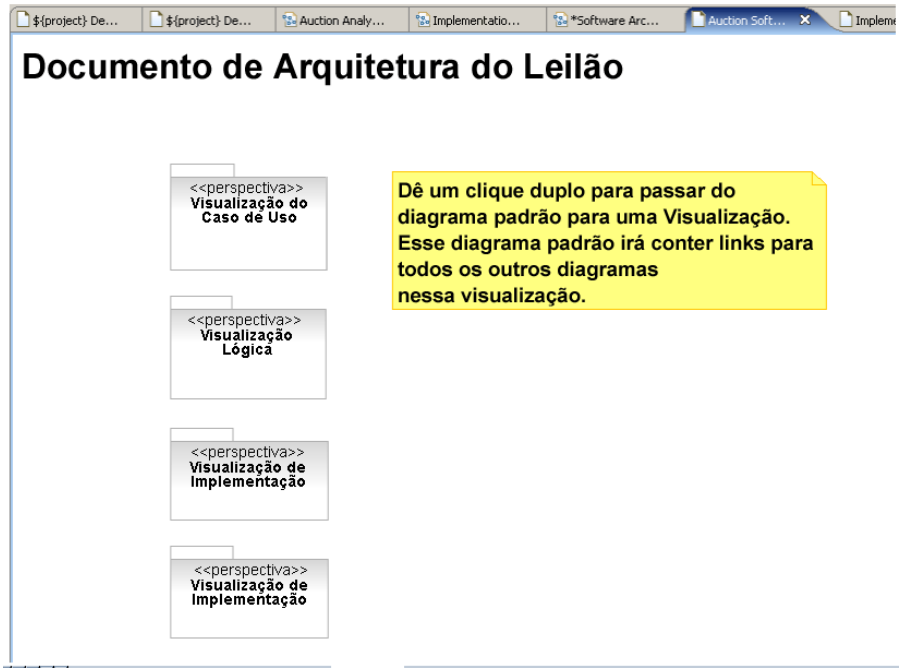



***Figura 10-2***

Em seguida, apenas criar diagramas no arquivo de modelagem do Documento de Arquitetura de Software, utilizando estas abordagens:

- Crie diagramas que sejam compostos pelos elementos de semântica de UML de outros arquivos de modelagem e que representem novas visualizações que não foram localizadas nesses outros arquivos de modelagem, mas que são necessários como parte do documento de arquitetura.

- Crie diagramas que sejam compostos por formas geométricas e/ou elementos de UML “ad-hoc” que residam no arquivo de modelagem do Documento de Arquitetura de Software. (Esses elementos UML devem servir apenas para finalidades de documentação ou de esclarecimento e não devem ter significado semântico para a implementação real da solução que está sendo descrita.)

- Crie diagramas que apenas contenham links para os diagramas existentes em outros arquivos de modelagem. (Essa técnica funcionará bem se o arquivo de modelagem do documento de arquitetura tiver que ser distribuído com outros arquivos de modelagem para consumo dos leitores. Se, em vez disso, o documento de arquitetura for ser publicado pela Web, siga uma das outras abordagens indicadas.)

## 11. Considerações sobre o Desenvolvimento da Equipe

A seção “Conceitos Básicos e Terminologia” discutiu os vários modelos, por exemplo, Caso de Uso, Análise e Design, como reconhecidos pelo RUP. Também foi fornecido um exemplo para ilustrar o ponto de que um modelo de “pré-implementação” no RSM ou RSA pode permanecer como um ou mais arquivos de modelagem e também que você poderia manter vários modelos de casos de uso, vários modelos de análise, etc., em que cada um desses modelos permaneceria como um único arquivo de modelagem ou como vários arquivos de modelagem. Esta seção introduz algumas das considerações sobre quando e por que você poderia escolher que um modelo permanecesse como vários arquivos de modelagem. Um tratamento mais abrangente sobre esses problemas deve ser localizado na ajuda on-line do RSA.

**Modelagem nas Equipes**

Quando várias pessoas tiverem que trabalhar em paralelo em um modelo, talvez você ache mais eficiente dividir o modelo em vários arquivos de modelagem (arquivos .emx). Com isso, é mais provável que a equipe possa fazer seu trabalho registrando a saída dos arquivos de modelagem para acesso exclusivo. Isso, por sua vez, diminui as chances de execução de mesclagens como resultado de duas ou mais pessoas que estejam modificando o mesmo arquivo de modelagem.

Por outro lado, há vantagens e desvantagens. As mesclagens, que serão requeridas com menos freqüência ao utilizar vários arquivos de modelagem, provavelmente ainda terão que ser feitas ocasionalmente. E as mesclagens processam os *arquivos* de modelagem individualmente, não em grupos. Ou seja, quando um modelo é armazenado como vários arquivos de modelagem, os arquivos individuais são processados “fora do contexto” do modelo completo (lógico). As mesclagens funcionam melhor quando a sessão de mesclagem tem acesso a mais do conteúdo informativo do modelo completo—ou seja, quando o modelo é dividido entre *menos* arquivos de modelagem.

Enfim, particionar modelos em vários arquivos não é tão importante quanto estruturar modelos de tal maneira a permitir que vários membros da equipe trabalhem neles sem introduzir alterações conflitantes que devam ser resolvidas durante as mesclagens. Recomendamos que você particione (*instâncias lógicas* de) modelos em vários arquivos físicos somente quando ele for reduzir significativamente a necessidade de executar mesclagens. Normalmente, e possível reduzir a necessidade de execução de

mesclagens somente quando os membros da equipe podem trabalhar com êxito com os arquivos individuais, *exclusivamente* (isto é, somente um membro da equipe tem um arquivo cujo registro de saída foi executado em algum momento exato) e *isoladamente* (isto é, cada membro da equipe pode fazer suas alterações nos arquivos individuais, sem também requerer acesso a outros arquivos que compõe o contexto lógico completo do modelo).

**Duas Abordagens para o Particionamento de Modelos**

No RSM e RSA, você pode decidir sobre a divisão de instâncias do modelo lógico em vários arquivos físicos em uma base ad-hoc, utilizando os recursos das ferramentas para reformulação de modelos. Entretanto, nossa recomendação é *planejar antecipadamente.* Os parágrafos a seguir comparam e contrastam as duas abordagens.

*Abordagem Planejada: Decompor Modelos no Início*

Faça o melhor possível para prever as necessidades de compartilhamento de sua equipe e formular adequadamente seus modelos em subconjuntos lógicos. Por exemplo:

- Formule cada modelo de caso de uso em arquivos de modelagem separados, observando como o conhecimento funcional é distribuído entre os analistas na equipe e divida as coisas adequadamente (essencialmente, criando um arquivo separado onde você possa ter, de outra forma, utilizado um pacote para organizar os casos de uso reais no modelo). Por exemplo, em um aplicativo provedor de saúde, você pode ter um arquivo de modelagem para conter os casos de uso de registro de pacientes e outro para casos de uso de processamento de pedidos. Ou pode decompor ainda mais os casos de processamento de pedidos em arquivos de modelagem separados para pedidos de laboratório, pedidos de radiologia, pedidos de farmácia, etc.

- Formule cada modelo de análise em vários arquivos e, novamente, a abordagem recomendada é dividir as coisas de acordo com a alocação de conhecimento entre os membros da equipe ou com os agrupamentos lógicos naturalmente localizados em seu domínio de problema (duas considerações que na maioria das vezes se alinham).

- Formule cada modelo de design em vários arquivos com base nos principais subsistemas, serviços ou componentes de sua arquitetura (ou seja, de acordo com a forma que espera que o trabalho de implementação seja designado às equipes ou pessoas).

Em cada um desses casos, para cada modelo você também poderia manter um arquivo de modelagem adicional que contivesse elementos e diagramas compartilhados/comuns que fornecesse visões gerais que fizesse partição (subsistema/pacote/serviço) cruzada dos limites.

Orientação adicional sobre particionamento de modelos pode ser localizada na Ajuda On-line do RSM/RSA

*Abordagem Ad-hoc: Reformulação de Modelos*

No RSM e RSA, você pode criar vários arquivos para um modelo, iniciando com o modelo inteiro em um arquivo e, em seguida, ‘separando’ ramificações do modelo para criar arquivos adicionais. Mesmo que você tenha planejado uma estratégia de partição antecipadamente, esta abordagem poderá ser útil quando você reconhecer uma nova oportunidade para aprimorar os workflows da equipe.

A finalidade desta discussão é o conteúdo lógico dos modelos, portanto, uma ramificação separada de um modelo é chamada de “submodelo”. Quando um submodelo é separado do modelo que o contém, o modelo original mantém um ‘atalho’ que aponta para o submodelo. O submodelo não mantém um ponteiro de volta para o modelo original. Por padrão, os elementos no novo submodelo não mais

residirão no espaço de nomes do modelo pai original. Entretanto, é fornecida a opção durante o procedimento de separação para propagar o espaço de nomes do modelo original para o novo submodelo. Para obter as etapas para criar um submodelo separando-o do modelo original, consulte a Ajuda On-line do RSM/RSA.

**Referências de Arquivos Cruzados**

Quaisquer dois arquivos de modelagem RSA podem referenciar seus elementos entre si. Isso pode incluir referências que estendem projetos. Algumas dessas referências representarão relacionamentos que você cria explicitamente, como dependências entre os casos de uso ou associações entre as classes. Outras são criadas pelo RSA e ficam ocultas em alguns casos. Por exemplo, os links de rastreabilidade podem ser gerados por meio da aplicação de transformações e esses links são criados como elementos de modelo visíveis normais. Como outro exemplo, um diagrama do RSA contém referências que apontam para os elementos de semântica representados nesse diagrama (qualquer elemento de semântica pode aparecer em vários diagramas). As referências do elemento de exibição são referências “ocultas” (isto é, você não as verá no Explorer de Modelo).

Cada referência de arquivo cruzado representa um provável ponto de interrupção em casos como:

- Um ou mais dos arquivos de referência cruzada não pode ser salvo adequadamente (por exemplo, em razão de travamento do sistema).

- Os arquivos são movidos no sistema de arquivo, sem que o servidor de modelos RSM/RSA saiba disso.

Portanto, ao planejar decompor um modelo em vários arquivos, lembre-se de que vale a pena considerar se um resultado pode ser uma proliferação de referências de arquivo cruzado. Novamente, organizar modelos para favorecer a coesão funcional e a desconexão de organizational units (pacotes ou arquivos de modelagem) pode contribuir com uma melhor experiência de desenvolvimento da equipe.