Rational Software Modeler 和 Rational Software Architect 的模型结构指南（2004 发行版）

白皮书

Bill Smith，Model Driven Development，IBM Rational Software

V1.0
2004 年 9 月 8 日

目录

1. 简介

目标读者

本白皮书用来支持有兴趣将 Rational Unified Process（RUP）中的指南应用到他们对 Rational Software Architect（RSA）的使用中的 RSA 的用户。如果您是 Rational Software Modeler（RSM）的用户，您也会发现本白皮书很有用，但是要注意，一些章节反映的功能只在 RSA 中可用，而不能在 RSM 中使用。本白皮书主要用于您在 RSA 中创建一组新模型的情况。如果您是第一次使用 RSA，但以前使用过 Rational Rose 或 Rational XDE，并且从这些产品中导入了模型，您可能会发现本白皮书中的可以用作重构您导入的模型的指南的价值。

本白皮书假设您已经对 UML 有了广泛的了解，并且基本熟悉 RSA 操作的基本概念和理论，特别是建模、转换和可视代码编辑。

目的

RUP 描述了表示对系统的问题和解决方案领域的定义明确的透视图的一组模型。本组模型结构的效用已经在许多真实的项目中得到了验证，而且不管您是否遵照正式的流程（如 RUP），这些模型结构都值得考虑。本文档讨论如何用 RSA 来实现 RUP 模型工件。

如标题所示，本白皮书中描述的项目和模型结构是指南，而不是必须遵守的规则。您是否决定在 RSA 中建模一个特殊的 RUP 工件是在您自己的开发过程要考虑的事项，而且往往是一个因项目而异的决策。应该牢记的是 RUP 不是一套死板的流程规则。它是一个流程*框架*，在其中形成的流程定义可以是非常正式的，也可以是非常不正式的。

用户使用 UML 的方式可以非常正式，也可以非常不正式。您可以像对待需要在构造过程中严格遵守的正式的体系结构图一样来对待 UML 模型；或者您也可以像对待标识设计的大体框架的草图（一旦项目进入实施

阶段，就可以废弃了）一样来对待您的模型。RSA 既可以在流程结束时支持您，也可以在建模时支持您。从这个出发点来看，本白皮书中的指南并非要束缚您的思维，而是要帮助您了解如何使用 RSA 的功能来帮助实现看上去最适合您的流程。

请注意，RSA 使您不但能将模型用作蓝图，还能用作从中可以自动生成实施的解决方案规范，。这是通过使用 RSA 的“模型到模型”和“模型到代码”转换实现的。使用 RSA 转换来实行“模型驱动开发”（MDD）提出了关于模型结构的要特别关注的问题。如果您要用 RSA 模型和转换来实行“模型驱动开发”，您还应该查询可以在 Developer Works 上获得的各种特定于 RSA MDD 的资源。

范围

本白皮书描述了如何在 RSA 中表示 RUP 工件，并提供了这些工件的内部组织结构的指南。其目的*不*是为了：

- 重申 RUP 模型工件的基本概念，也不是为了提供这些工件的各种描述
- 描述指定相关 RUP 工件的详细的语义或图形内容的流程或技巧

有关如何定义、开发并建模 RUP 工件的不特定于工具的信息，请参阅 RUP。

有关开发 RSA 模型内容的特定于工具的信息，请参阅：

- 产品文档（教程、样本、联机帮助）
- 特定于 RSA 的 RUP 配置中的工具向导（本白皮书即属于其中一部分）
- Developer Works 上与 RSA 相关的资源

印刷约定

对于从 IBM Rational Rose 或 XDE 迁移过来的 RSA 用户有用的讨论表示为工具条的形式，文字用边框框起来，并且背景有阴影：

> ***XDE／Rose***
> 对以前的 XDE 或 Rose 用户有用的讨论。

白皮书结构

接下来的“基本概念和术语”部分确定了使用的词汇表，并提供了关于模型如何在 RSA 产品中实施的一些一般信息。

接下来，“RUP 模型到 RSA 模型的映射”部分讨论 RSA 如何支持 RUP 定义的模型类型。

再接下来的几个部分提供了在 RSA 中构建各种类型的模型的指南。这些部分中的一些讨论了根据您希望流程、建模方法和体系结构控件方面如何精确来使用不同模型类型的方法。

最后，还有一部分讨论与将模型进行分解为多个建模文件相关的各种问题。这初步涉及了管理规模以及在团队成员间共享模型以便将文件争用和冲突合并降到最低的策略。

2. 基本概念和术语

模型

在 RUP 中，模型的定义是从某个角度看到的问题或解决方案领域的完整的规范。问题领域或系统可以由在领域或系统上代表不同角度的一些模型来指定。RUP 提出了特定的一组模型：

- 业务用例模型
- 业务分析模型
- 用例模型
- 分析模型（可以包含在设计模型中）
- 设计模型
- 实施模型
- 部署模型
- 数据模型

请注意，RUP 本身是不特定于工具的。在涉及 RUP 的地方，模型可以是餐巾纸上或白板上的一幅画，可以是建模工具中的什么东西，甚至可以是想象中的图像。所以，从 RUP 的角度看，模型是个逻辑概念。在 RSA 的环境中，我们可以从逻辑的角度讨论模型，但也可以从物理的角度进行讨论。

假设您有几个团队在处理两个应用程序的工作：一个团队由三个分析人员组成，他们处理时间表管理应用程序，另一个团队由五个分析人员组成，他们处理呼叫中心应用程序。两个团队现在的工作都是捕获需求，并正在使用 RSA 进行用例建模。用 RUP 术语，您会说，一个团队正在构建“时间表应用程序的用例模型”，另一个正在构建“呼叫中心应用程序的用例模型”。但是，如果团队使用的是 RSA，就要认识到他们的模型有特殊的物理表现。这是下一部分的主题。

建模文件

在 RSA 中，模型保留为文件。（用 Eclipse 的术语来讲，文件被认为是“资源”[1]，所以如果您在本白皮书或其它地方遇到了“建模资源”这个术语，它的意思就与“建模文件”相同）。从广义来讲，RSA 支持两种建模文件：

- “实施前”建模文件。这些文件包含不*直接*反映实施工件的概念 UML 内容。这些文件既包含模型的 UML 语义，也包含描述语义元素的 UML 图。

- 实施建模文件（Eclipse 资源），它们是正常的实施工件（如 Java 源文件或 Web 页面），位于 Eclipse 项目中。在这种情况下，您可以认为项目代表实施建模文件的内容的范围。模型语义位于实施工件本身中。每张图都位于项目内其自身的物理文件中。这些图可能使用 UML 符号表达式，但它们也可能使用其它符号表达式（例如，用于数据可视化的 IDEF1X 或“信息工程”符号表达式，或者用于设计 Web 层的 Rational 专用符号表达式）。受支持用于描述 3GL 代码工件的 UML 图类型包括类图以及（仅用于 Java）序列图。

本白皮书的重点是如何组织“实施前”模型的内部结构，**在本白皮书中，“建模文件”这个术语继续用于包含“实施前”模型内容的文件**。组织实施项目的内容的指导信息可以在其它地方找到，如 Rational Software Architect、Rational Application Developer 和 Rational Web Developer 的联机帮助。

“实施前”建模文件不一定包含一个模型的所有信息。实际上，经常出现的情况是建模文件仅包含模型的一部分。例如，在上面的例子中，我们的三个人的团队处理时间表应用程序的用例模型，团队可以选择将

[1] 在 Eclipse 中，资源是文件，但在 Eclipse 环境中还有另外的属性和行为。此处描述的建模文件被 Elipse 视为“资源”。

其用例模型分成三个建模文件，这样团队的每个成员可以负责用例的不同部分，而不会争用同一个文件。本白皮书的最后部分讨论与将模型分区和管理建模文件相关的问题。

模型类型

在 RUP 中，模型是有具体的类型的，如用例模型、分析模型或数据模型。在 RSA 中，您可以认为建模文件是有类型的：文件包含的模型（或部分模型）的类型。建模文件的类型可以用两种方法之一确定：

- 从“空白”建模文件（见下）开始，然后只要用命名方式以及您放入其中的内容（包括您对其应用的 UML 概要文件）就可以确定其类型了。
- 根据代表某种模型类型的预定义的“模板模型”来创建它。请注意，建模文件的“类型”实际上只是关于文件的内容的约定。例如，工具不会阻止包含用例模型的文件同时包含实现用例的类。但是，这些指南建议您将模型文件作为有类型的文件进行处理。

工作区、项目和项目类型

熟悉 Eclipse、WebSphere Studio 产品或 Rational Application Developer 的读者已经知道文件位于项目内，项目可以有各种类型，并且项目在工作区内进行（虚拟的）分组和管理。RSA 建模文件和其它文件一样位于项目内。

出于本次讨论的目的，将不需要详细说明可以在 Rational Software Architect、Rational Application Developer 和 Rational Software Modeler 使用的所有项目类型。从最广义的角度讲，我们对两类项目感兴趣：

- **UML 项目**
- **实施项目**，这样的项目包含特殊类型，如 Enterprise 项目、EJB 项目、Web 项目和 C++ 项目

我们在前面说了 RSA 支持两种建模文件：

- 包含 UML 模型的文件（语义实施加上描述它们的图），用于在实施上的抽象级别进行建模（如需求、分析和设计）
- 包含像 Java 代码或 Web 页面这样的实施工件的项目加上（可选）包含描述实施工件的图的图文件。

将模型分配给项目的规则很简单：**a)** 将“实施前”建模文件放到 UML 项目，**b)** 实施模型自我管理，因为从本质上来讲：

[实施模型] = [实施项目]

这个规则有一些例外：以下 UML 建模文件是可以放置在语言特定的实施项目内的候选文件：

- 设计“草图模型”（将在后面的部分讨论）

- 带有序列图的模型，其中序列图描述将针对项目中的代码执行的测试

> ***XDE／Rose***
> 操作的 Rose 和 XDE 理论包括迭代优化设计模型直到您达到与代码相当的抽象级别的做法，然后使用代码模型同步技术以使该模型的语义与代码本身保持一致。例如，在 XDE 中，实施模型不仅作为代码和图存在于项目中，而且作为“代码模型”文件存在，它们独立于实施工件继续存在，并且在本质上代表它们的语义的冗余副本。
>
> 操作的 RSA 理论鼓励在高于代码的抽象级别使用不特定于平台的模型（即设计模型，如“企业 IT 设计模型”）并使用转换从那些模型中生成代码。然后，在代码抽象级别，RSA 只让您画 UML 代码图，并且省去了使用独立存在的语义模型的方法。
>
> 请注意，RSA 不*阻止*您在抽象的代码级别定义 UML 模型并从中生成代码；实际上，RSA 希望您这样使用。但是 RSA 不提供将这样的模型与代码保持同步的技术。这种类型的用法一般对应于非常“轻量级”的 RUP 变量，而一旦生成了代码，设计模型就被认为是无足轻重的了。

回顾概念

下面的图总结了前面的讨论内容。图反映了较早时描述的场景，即我们有两个团队，一个处理呼叫中心应用程序，另一个处理时间表管理应用程序。

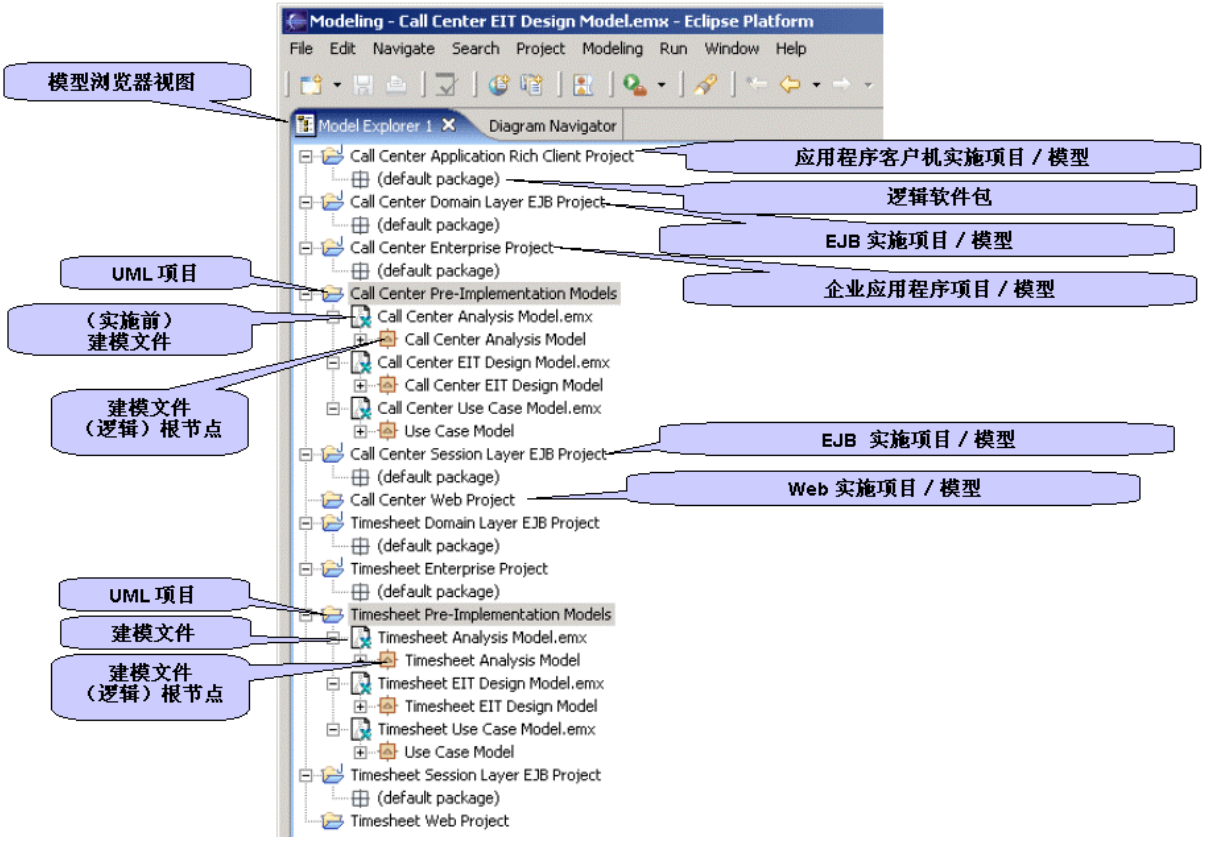


图 2-1

RSA 提供“模型浏览器”视图，它提供了模型的物理和逻辑组合视图。在“模型浏览器”中，您看到在顶级节点描述的工作区中的项目，在每个项目中您看到属于该项目的资源。**图 2-1** 在“模型浏览器”中描述了于我们的场景中的两个应用程序相对应的项目的集合。UML 项目已经用于实施前模型。与解决方案相应的类型的其它项目（如企业应用程序项目、Web 项目等等）的集合已经用于实施模型。

> ***XDE／Rose***
> 与 RSA 模型浏览器不同，Rose 和 XDE 中的模型浏览器仅提供模型的逻辑视图。请注意，RSA 模型浏览器提供的资源的视图不是 Eclipse 导航器视图提供的“纯”物理视图。虽然一些物理资源在模型浏览器中可视，但它们主要用表示资源的*逻辑*视图的图标进行表示。

图 2-2 显示了时间表用例模型可以如何内部组织为代表问题领域一些功能上连贯的部分的软件包。

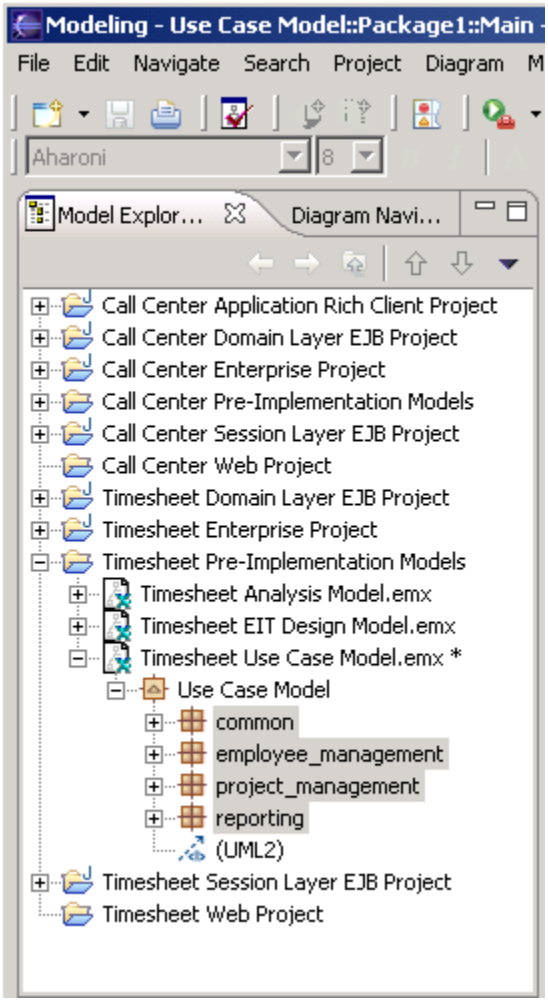


图 2-2

在**图 2-1** 和 **2-2** 中，每个实施前模型都位于单个建模文件中。或者，它们中的任何一个都可以重构成多个建模文件。例如，时间表用例模型可以重构成四个建模文件，其中每个都对应于描述的问题领域的一部分（“common”、“employee_management”、“project_management”和“reporting”）。在这种情况下，每个建模文件的根节点将被命名为在组成完整的用例模型的所有建模文件中保持一致的名称空间。例如，四个建模文件的根节点可以为“timesheet.requirements.common”、“timesheet.requirements.employee_management”、“timesheet.requirements.project_management”和“timesheet.requirements.reporting”。

3. RUP 模型到 RSA 模型的映射

下表显示最常用的 RUP 模型如何映射到 RSA 模型类型。映射一般是直接的，但是关键是要将本白皮书用作用 RSA 来练习 RUP 的指南。表中提到的 RSA 模型类型在紧接着表后进行讨论。各种模型类型的内部组织以及什么种类的项目使它们在里面的指南在后面的部分提供。那些后来讨论的内容在这里从 RSA 模型类型的角度列出来。

RUP 模型	RSA 模型类型
用例模型	用例模型
分析模型	分析模型 （或者：设计模型中的《analysis》软件包）
设计模型	对于 n 层业务应用程序：企业 IT 设计模型 对于其它类型的应用程序：用作设计模型的空白模型 对于设计“草图”：用作设计“草图”模型的空白模型 可选的补充内容：用作实施概览模型的空白模型
实施模型	包含实施工件和图文件的实施项目
部署模型	用作部署模型的空白模型

RSA 模型类型

空白模型

RSA 提供创建“空白模型”的选项（文件→新建→UML 模型→空白模型）。“空白模型”是不基于模型模板的建模文件。它不应用特殊的概要文件，而且除了单个“主”（自由格式）图没有缺省内容。**您可以将空白建模文件用作任何类型的模型的起点**。通过选择您对它的命名方式、您在其中定义的内容以及您对其应用的概要文件，就可以用空白建模文件来构建用例模型、分析模型、设计模型、部署模型或任何其它类型的 RUP 模型。

用例模型

RSA 提供基于模型模板创建“用例模型”文件的选项。模板提供的缺省内容如图 ***3-1*** 所示。（说明如何使用“构建块”内容和搜索字符串不在本文档的范围之内。模板包含指示信息，您会发现它们大多数简洁明了。）

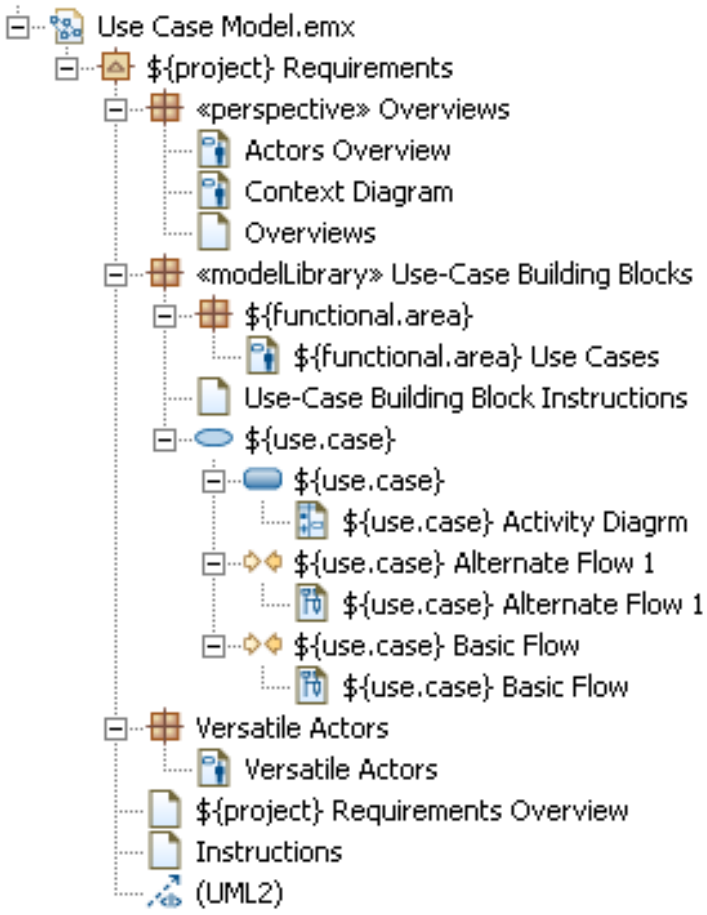


图 3-1

分析模型

RSA 提供基于模型模板创建“分析模型”文件的选项。模板提供的缺省内容如图 ***3-2*** 所示。此外，“分析”概要文件适用于从该模板创建的模型文件：

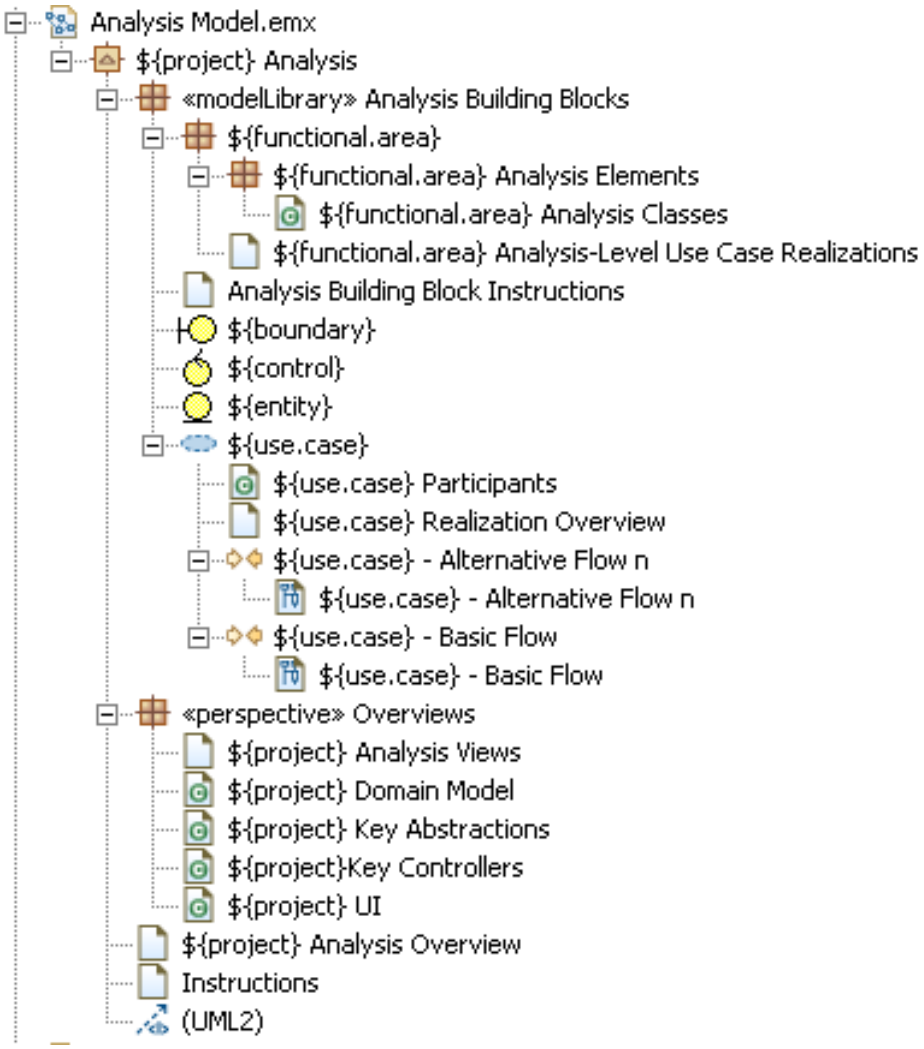


图 3-2

企业 IT 设计模型

RSA 提供基于模型模板创建“企业 IT 设计模型”（EITDM）文件的选项。模板提供的缺省内容如**图 3-3** 所示。此外，“EJB 转换”概要文件[2] 将适用于从该模板创建的模型文件。在以业务应用程序为目标时以及用 RSA 代码生成转换类支持这样的应用程序的创建时，这是用于设计（或者用于分析）的恰当模板。

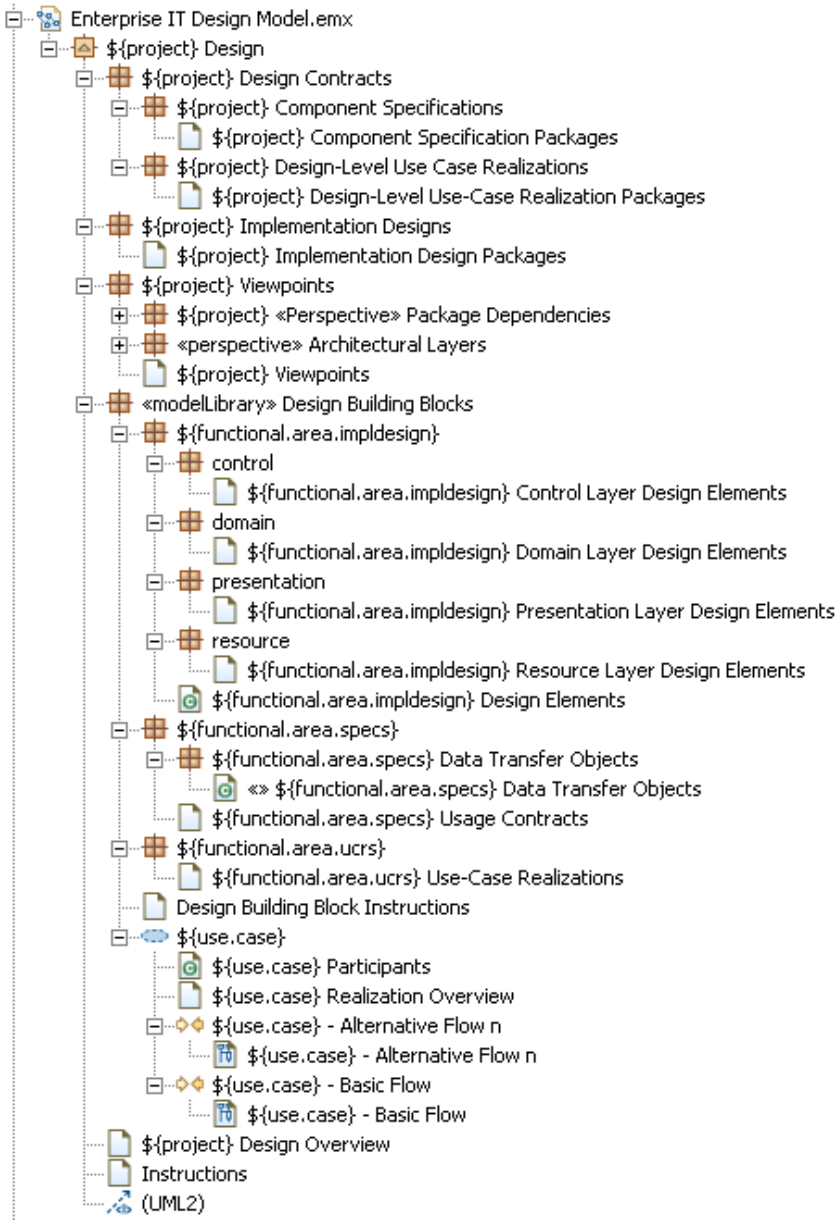


图 3-3

[2] 作为 EIT 设计模型模板一部分提供的实现转换的概要文件集合可能会在发行产品的升级时发生演变。

实施概览模型

作为设计模型的一部分，您还会发现定义“实施概览模型”来捕获实施将如何组织的高级视图很有用。“实施概览模型”将在设计阶段早期使用——在生成或撰写代码前——来代表您期望代码和相关文件（元数据、部署描述符等）所在的实际的 RSA 项目和文件夹 / 软件包。您也可以用本模型显示那些项目和软件包之间预计的依赖关系，这在确定系统构建需求时会很有用。“实施概览模型”也可以是保存解决方案体系结构的非正式的概念图的地方。

实施模型

如前所述，在 RSA 中，实施模型由包含实施工件和（可选）描述这些工件的图的项目组成[3]。

“草图”模型

如“基本概念和术语”部分的注释所述，您可以为将设计模型当作在系统生存期进行维护并用来支持 / 加强体系结构控制的正式体系结构图。或者，您可以将它们当作用来建议设计并帮助澄清并交流设计，但被认为在实际的实施开始从原始设计分离时就可以被废弃的草图。RSA 同时支持两种方法。其功能一般不特别针对两种方法中的任何一种，但是您关于如何使用设计模型的选择当然会有助于确定您将使用什么 RSA 功能部件以及将如何使用。区别在本白皮书中提出的指南环境中变得很重要的地方，将使用“草图模型”这个术语来表示模型正在用更“可废弃”的方式进行使用。

[3] 要创建这些图，而不使用文件→新建→UML 模型来创建模型，您使用文件→新建→类图来创建图，在其中您可以用 UML（或其它）符号表示法来构建代码“视图”。 每个单个的图都保留为独立的文件，扩展名为 .dnx，并且可以与代码文件一样进行版本控制。 这些图不包含任何语义信息，而只有符号表示法。所有相关的语义信息都位于代码本身之中。当您更改这些图中的一个图的类名或操作签名这样的内容时，您实际上是在更改底层的代码本身。当您在代码中作出这样的更改时（用文本编辑器），更改后的代码出现的图会自动更新。

4. 组织模型内部结构的一般指南和技巧

组织 UML 模型的内容的基本工具是软件包。UML 软件包主要用于两个目的：

- 将模型信息分区、组织并标注
    - 将对应于问题或解决方案领域的某个主题的元素分组
    - 将不同类型的模型信息（如接口、实施、图等等）分隔开
    - 将元素分组来定义并控制它们对其它元素的依赖关系
    - 将提供同一个模型的多个视图的图分组

- 确定名称空间
    - 用于模型元素
    - 用于从模型元素生成的实施工件（这可能涉及模型和实施语言名称空间之间的映射）
    - 用于一组复用

一般情况下，RUP 已经为各种模型类型提议了具体的封装策略。这些策略反映在本白皮书的特定于模型类型的部分。RSA 还引进了一些其它组织工具，描述如下：

用《透视图》软件包表示观点

如果期望看到用多种方法组织的元素，就可以用描述其它组织模式的图来创建其它软件包。这一相同的技巧可以在需要表示超越模型的封装模式的模型内容的某个视图的地方都可以使用。RSA 通过提供《透视图》软件包构造型作为其 UML“基本概要文件”的一部分来支持这种技巧。您可以将《透视图》软件包作为用于系统工程或 IEEE 1417“观点”的 RUP 的等同物。

不要将语义元素（类、软件包、关联等等）放到《透视图》软件包中。只要将根据组织问题或应用程序观点描述视图的图放置其中就可以了。将《透视图》构造型应用到软件包是做了几件事情。它可视地将该程序包识别为表示某个观点。它还支持将语义元素放到《透视图》软件包时向您发出警告的模型验证规则。它还用作软件包的标志符，这样的软件包应该被 RSA 转换绕开

用主题图创建特殊关注问题的自我更新的描述

在“正常”的图中，您可以手动放置要描述的元素。与其相反，主题图的内容是由针对现有模型内容运行的查询确定的。要创建主题图，您选择“主题”模型元素，然后根据元素与主题元素的关系的类型定义您希望出现在图中的其它元素。随着模型的语义内容的更改，主题图也相应地调整。

通过浏览图检验模型

浏览图不是专门用于模型组织的工具。其目的是使模型内容的发现和了解变得更容易，而不必手动构建图。但是在模型组织的环境中，知道它们是有好处的，因为它们可以减少您构建持久图的需要。这反过来可以减少模型的大小、降低模型的复杂性，使它们更易于组织。

浏览图优点像主题图，但有一个主要区别，就是浏览图永远都不是持久的，它们总是即时生成。要生成浏览图，您选择模型元素（从图中或模型浏览器中），并使用上下文菜单来“探索浏览图”。这样生成的图将选中的元素描述为“中心点”，其中相关元素围绕中心点呈辐射状布局。当然，然后您就可以选择该浏览图中的一个相关元素，使其成为另一个浏览图中的中心点，而且如果您愿意的话可以继续这么做。

图间浏览

在 RSA 中，有两种机制用于图间浏览：

- 可以将图节点从模型浏览器拖动到另一个“主机”图中。然后您可以双击主机图上产生的图标来打开引用的图。

- 不管您何时在模型中创建新的 UML 软件包，总会自动创建“主”图（自由格式图）。缺省的情况下，该“主”图创建为软件包的“缺省”图。您可以将图命名为“主”之外的名称，它还是会被当作“缺省”图。您也可以选择软件包中的另一个图，并使它成为该软件包的“缺省”图。“缺省”图的目的是：如果您将软件包本身放入某个其它“主机”图中，然后您就可以双击软件包，这将打开其缺省图。

这些机制支持以下组织**指南**，这些指南可以应用于任何类型的模型：

1. 构建每个建模文件的“主”图（或其它缺省图）来描述：
    a. 建模文件中的每个顶级软件包
    b. 位于建模文件的根软件包中的任何其它图的图图标（换句话说，不要描述缺省图本身的图标）

2. 构建每个顶级软件包的“主”图（或其它缺省图）来描述：
    a. 它直接包含的软件包
    b. 它直接包含的任何其它图的图图标

3. 对软件包的每个后续的低级别重复该模式。

5. 用例模型的内部组织指南

注：在本部分以及后面的其它特定于模型类型的部分中，指南将使用从随 RSA 一起提供的“拍卖”展示场景示例中采用的示例进行说明。研究示例以查看其它组织详细信息以及图的内容。

用例模型高级组织

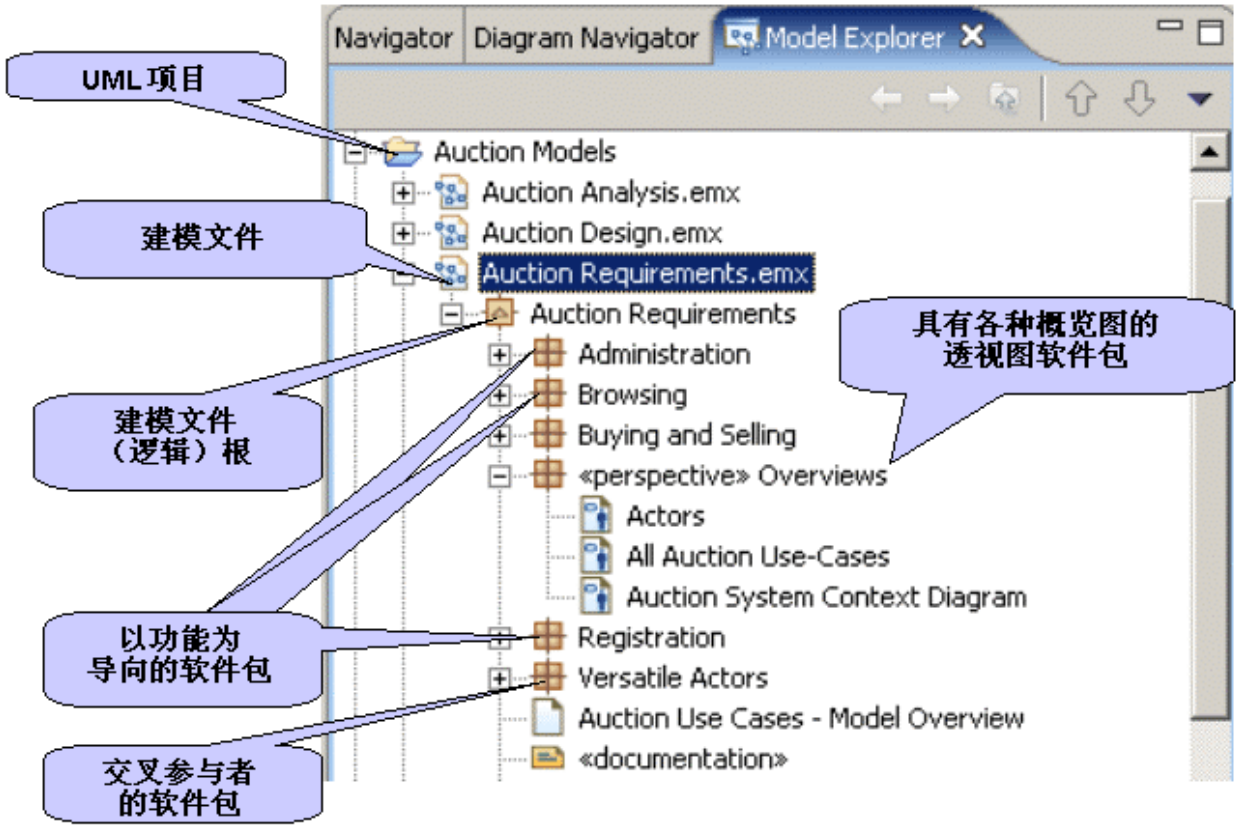


图 5-1

图 5-1 说明用于构建用例模型的以下指南：

1. 使用顶级软件包来确立以功能为导向的分组。**理由：**

   - 当一个团队的人员将处理用例模型的工作时，这一般很好地映射到关心的分工的问题上。而且，如果后来由于文件争用成为了问题，您决定将用例模型分成多个建模文件，它可以帮助您很好地做到这一点（您只要为每个顶级软件包创建独立的建模文件就可以了）。

   - 于其它组织方法相比，这一般会更好地映射到最终实施的组织。如果您将使用转换来种植抽象的每个后续的低层，这就很重要。特别是，如果您要根据用例模型在分析模型中生成种子值内容，您就会希望用例模型的封装结构能很好地映射到目标分析模型的期望的封装结构上。反过来，您希望分析模型的封装结构很好地映射到设计模型，设计模型的封装结构很好地映射到将组成实施的那组项目上。这些映射越简单，从一个抽象级别到下一个级别配置转换所要求的工作就越少。

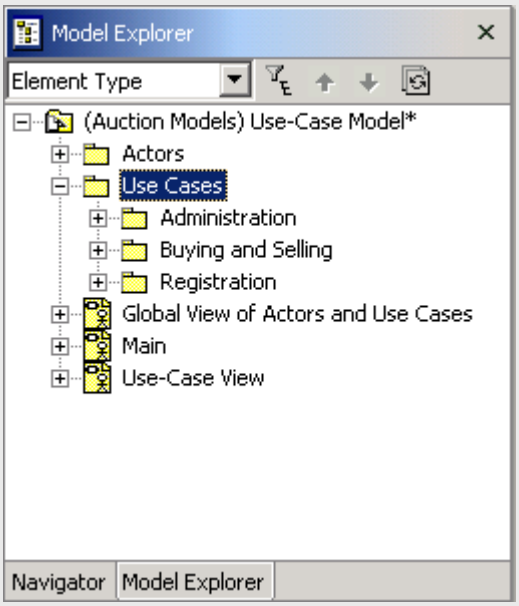
2. 使用另一个顶级软件包来捕获“充分授权”或“万能”的参与者。

3. 使用《透视图》软件包中的图来捕获用例的高级、交叉的视图。**理由：**

   - 提供交叉视图和“体系结构上有重要意义”的用例的视图，同时保持模型的语义元素组织为以功能为导向的分组。

XDE／Rose

RSA 指导信息在某种程度上改进了用例模型的高级组织的传统指导信息，即为参与者创建软件包并为用例创建另一个软件包。然后，按照模型的大小和复杂性的要求，您将使用低级软件包来确立以功能为导向的分组，如这一基于 XDE 的示例所示：



用例模型内容

本文档并非用作如何撰写好的用例或好的用例建模应该做和不该做的事情的教程。但是，我们在下面提供了用例模型除了包含参与者和用例外还可以包含的内容的简单讨论。

- **建议：**在模型根位置创建“主”图，描述模型的其它软件包并支持到那些软件包以及它们各自的“主”图的下寻。

- **建议：**在每个用例软件包中，包含描述软件包的用例、它们之间的任何关系以及参与到它们之中的参与者的图。（如果用例的数量很大，使用多个图可能比较合适。）

- **建议：**在每个用例的“文档”字段描述其主流程和备用流程[4]。（请参阅**图 5-2**。）

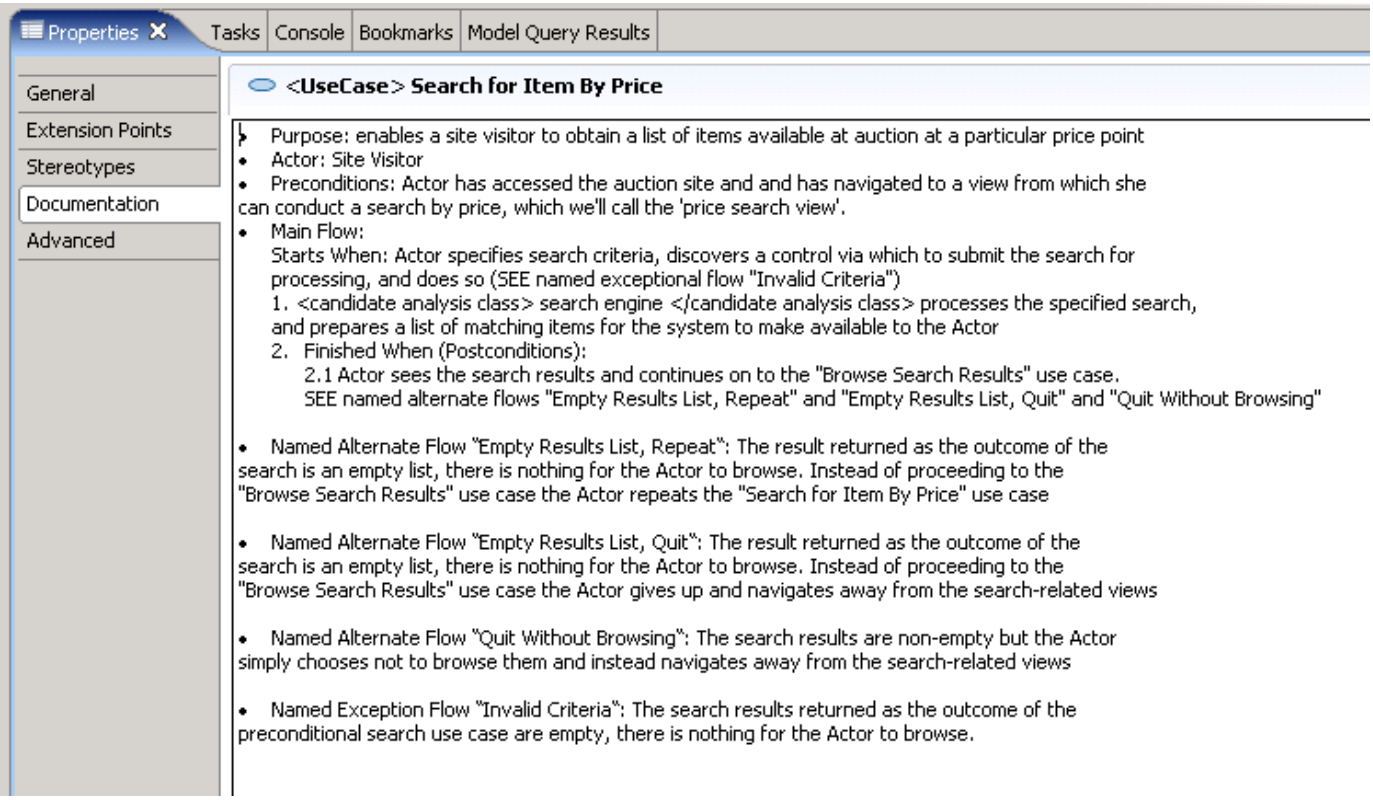


图 5-2

- **可选：**当用例足够复杂时，请添加“活动”图并使其反映用例的整体活动流程。（请参阅**图 5-3**。）
  理由：这有助于显示与每个（主流程和备用 / 异常）流程相对应的条件，并有助于确保各种流程最终会合并到一起。（在 RSM / RSA 中添加活动图将使活动自动添加到用例，其中图在活动下。）

- **可选：**为用例的每个命名的（主流程、备用和异常）流程建模“黑匣”实现；将协作发生添加到用例；将与用例的主流程相对于的交互实例加上每个命名的备用和异常流程的交互实例添加其中；为每个交互实例构建序列图（或者通信图）。这些用例协作实例不应该与分析级别用例实现相混淆（如分析模型中所述），也不应该与设计界别的用例实现相混淆（如设计模型中所述）。那些是用例的“白匣”实现，它们描述解决方案的内部元素之间的交互。这里为用例模型提议的协作发生是严格意义上的参与者和系统之间的“黑匣”交互。（请参阅图 **5-3**。） **理由：**这向非技术涉众提供了用户系统将如何用系统交互的高级图片。它还可以帮助您识别作为实施的一部分而要求的各种视图（屏幕或页面）。它还通过将名称分配给语义模型元素（即，分配给协作发生），正式确定用例的各种流程（场景）的命名。

> ***XDE / Rose***
> 在 UML 1.x 中，您应该已经将“协作实例”（而非“协作发生”）用于本目的。

[4] 用例说明示例中描述的格式编排通过以下方式实现：用支持 RTF 的编辑器创建用例说明的文本“模板”，然后将模板复制并粘贴到用例说明字段。

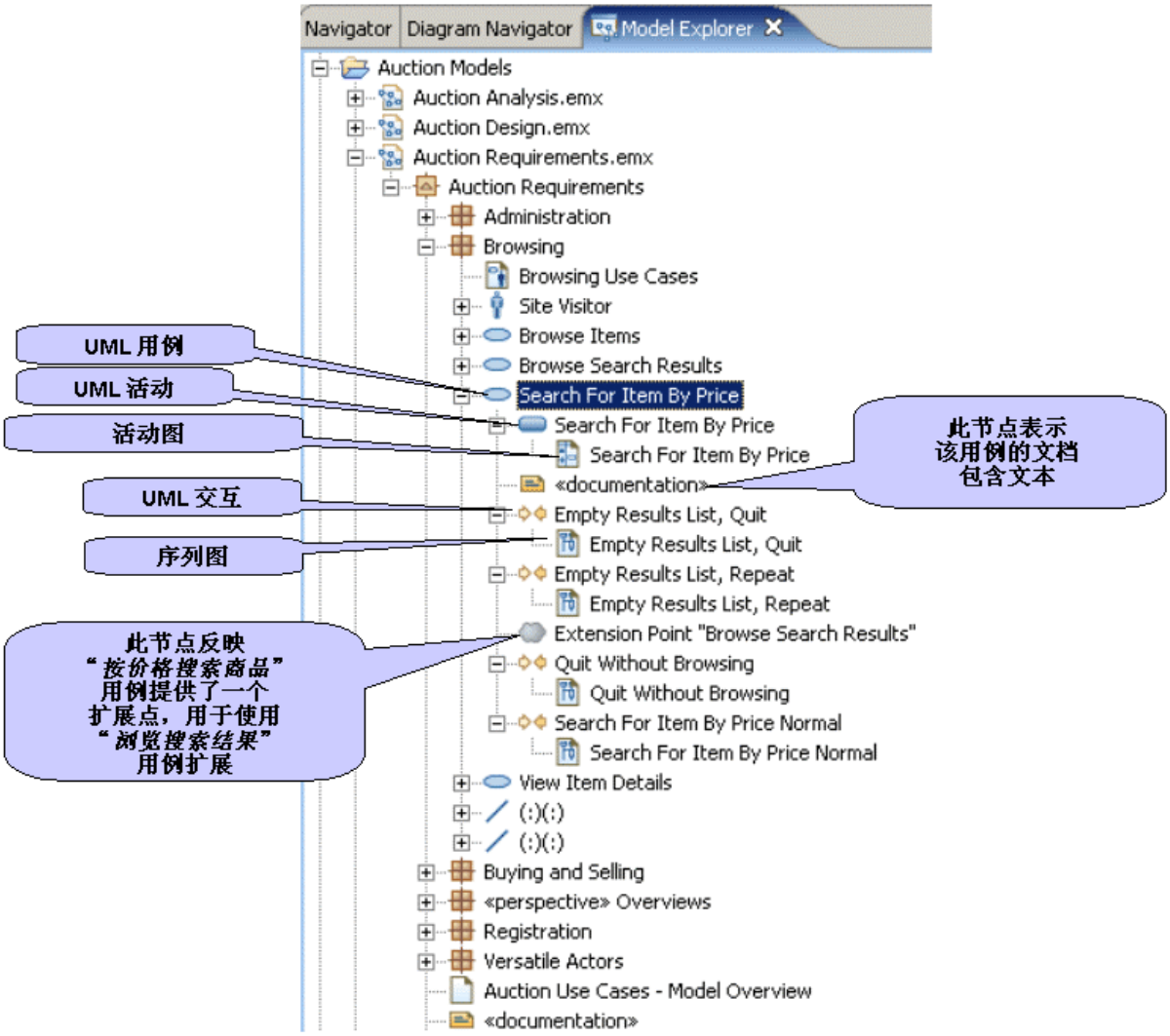


图 5-3

- **可选：**如果您在遵照 RUP 指导信息来识别您的体系结构的“体系结构上有重要意义的”视图，特别是如果您将维护“软件体系结构文档”，请添加顶级《透视图》软件包以包含描述体系结构上有重要意义的用例。您可能希望将软件包命名为“体系结构的用例视图”。

6. 分析模型的内部组织指南

分析模型代表解决方案的“第一步”。它是从需求到最终设计的第一步，重要侧重捕获关于业务领域的信息，并显示接近业务的高级抽象的候选解决方案元素。它是分析类和分析级别用例实现所在的地方。您是通过建模用例实现的过程（主要使用序列图）开始*发现*解决方案需要什么类的 —特别是，这些将是与您发现在序列图中需要的生命线相对应的类。也可以应用一些凭经验得出的方法来根据用例模型的内容建议分析模型。这些将在本部分的后面谈到。

在 RUP 中，分析模型是否应该独立于设计模型进行维护是一个特定于项目的决策，是您根据您是否相信维护独立分析模型的价值是否值得投入的时间而作出的决策。如果创建独立的分析模型，但没有得到维护，那么分析类 就将移动到设计模型 并进行优化。或者，分析模型可能会慢慢演变成为设计模型[5]。从特定于产品的角度讲，以下是一些您可能要考虑的选项:

1. 根据分析模型模板创建位于建模文件（或一组文件）内部的分析模型。然后根据企业 IT 设计模型模板用手动过程或自动化转换来创建分析元素的优化版本，然后除去分析建模文件。这使您可以选择保留进行中的独立的分析模型，也可以将它抛弃。
2. 根据您对其应用了分析概要文件的企业 IT 设计模型模板在建模文件（或一组文件）中进行分析级别的建模。这样您就可以用分析类开始建模用例实现，然后随着时间的发展优化它们，这样设计接口就可以采用行为中的角色。
3. 第二和第三个选项加起来是将各种分析模型作为设计模型保留在相同的建模文件中。要做到这一点，您将把分析内容隔离到您应用了关键字《分析》的软件包中。这提供了将分析级别的工件在相同的建模文件中保留为它们更优化的设计级别的对等物的机会。

在使用 RSA 转换生成实施时要知道的一个注意事项是：那些转换在许多情况下可以将分析级别的元素接受为它们的输入，从而为您省去了手动将那些元素优化到设计元素的一些步骤。用这种方法使用 RSA 时，最好选用上述的选项 2 或 3。作为 RSA 一部分的标准代码生成转换将绕过有《分析》关键字的模型软件包。

[5] RUP 实际上提出了在设计模型中创建分析类 和分析级别用例实现 然后直接将它们从那里演变为它们的设计格式的选择。。用这种方法，当设计模型“被发现”时，您可以用您保留一些“纯分析”透视图的方法来创建软件包。

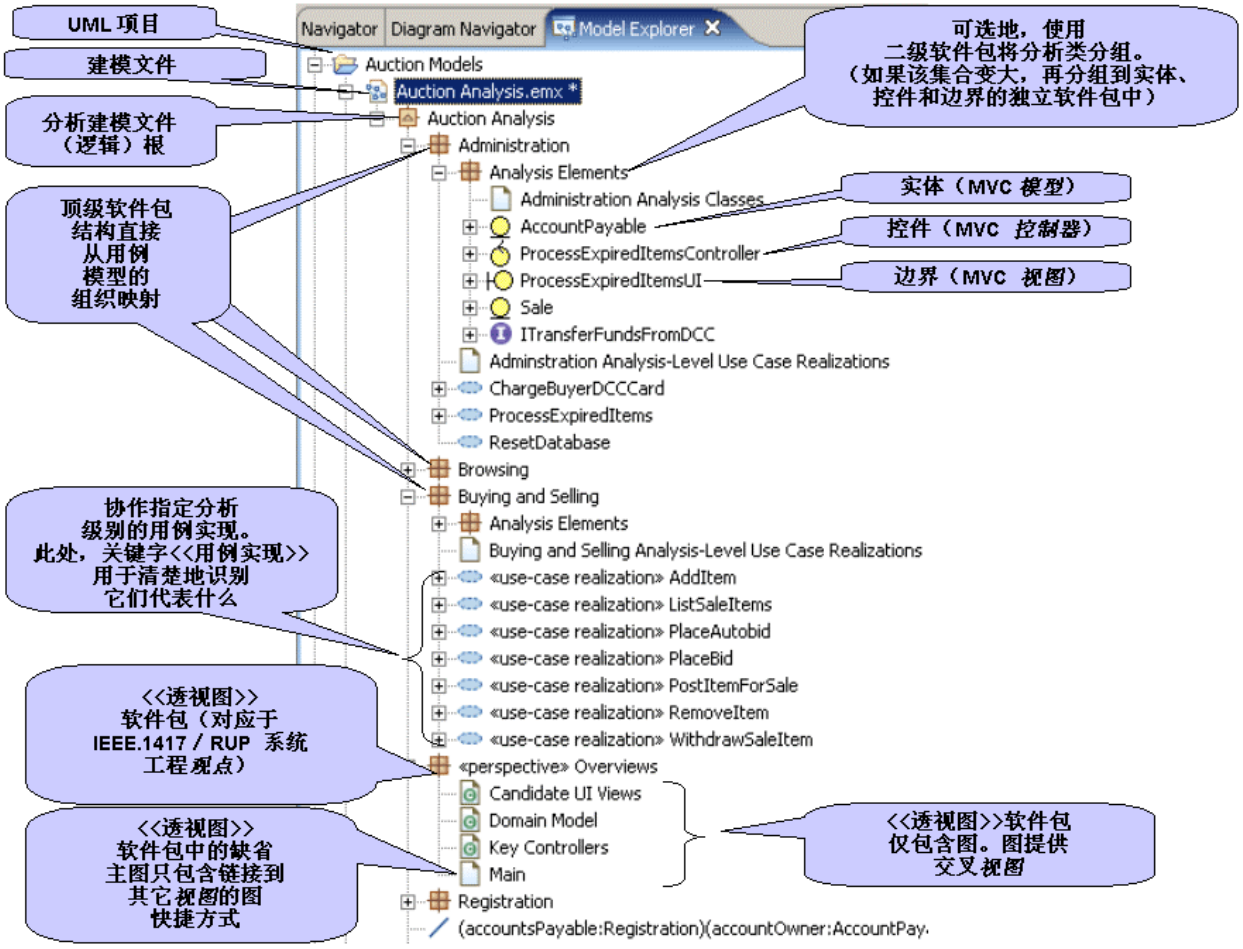


图 6-1

图 6-1 说明用于构建分析模型的以下指南：

1. 使用顶级软件包来为分析类确定以功能为导向的分组。**理由**：与用例模型的理由相同。

2. 可选地，在顶级软件包内使用子软件包来收集并组织分析类。

3. 使用《透视图》软件包中的图来捕获分析元素的备用、高级或交叉视图。**理由**：为不同的涉众提供不同的透视图，同时保持模型的语义元素组织成以功能为导向的分组。

这一方法的稍有不同的变体在**图 6-2** 中描述，它显示了使用顶级软件包将用例实现从分析类隔离。在该顶级软件包中是一组以功能为导向的子软件包，它们与那组顶级软件包相匹配。用这种方法分离用例实现使您能够重构包含分析类的软件包结构，而不*一定*影响用例实现的组织。（特别是如果分析模型将*在原处*演变为设计，那么类的软件包组织就可能会演变，这样它就不与原来用于用例的软件包相匹配。）

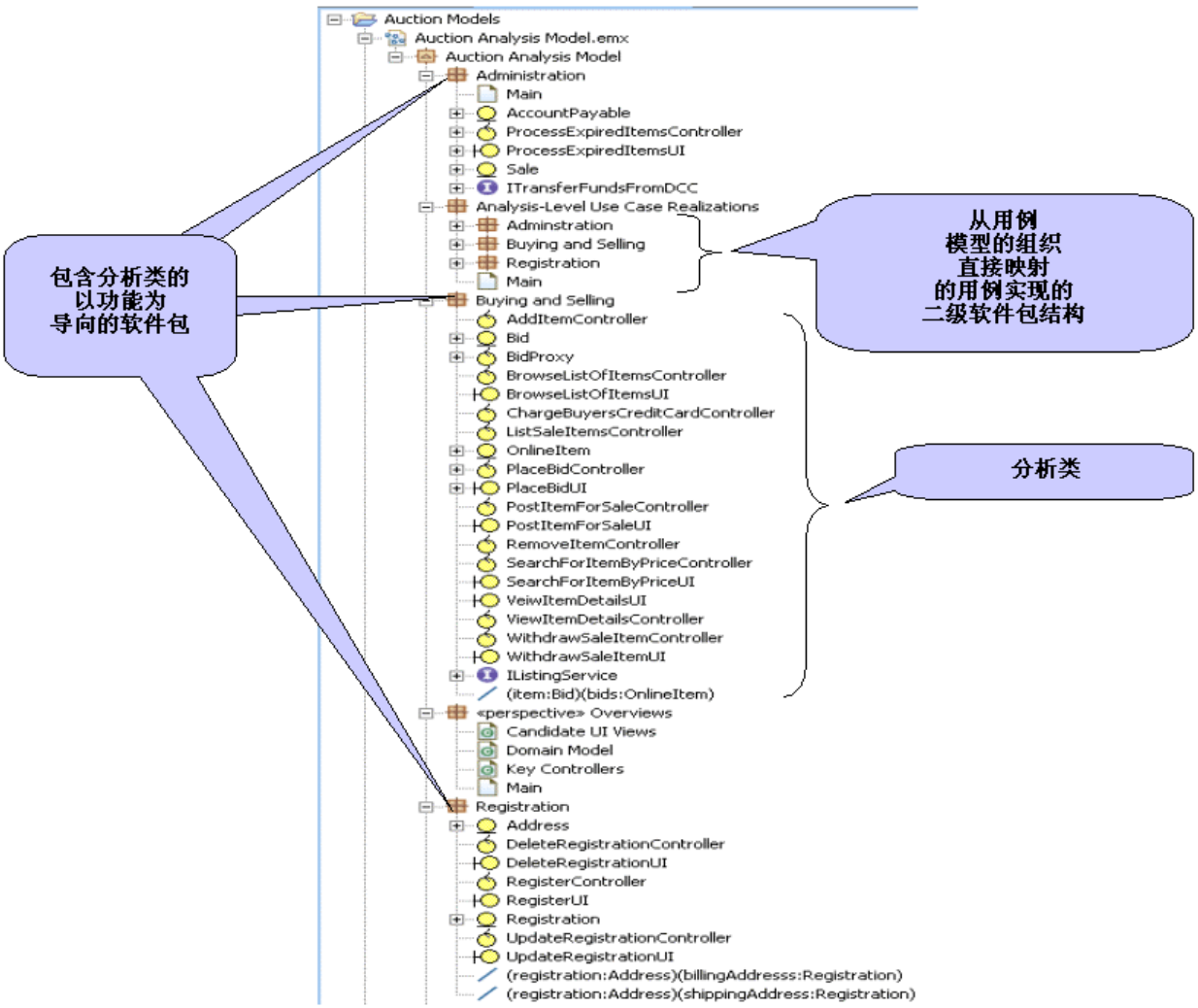


图 6-2

根据您的情形，可能有理由使用一个命名约定，该约定预计多个独立组创建的模型内容的合并和复用，甚至包括不同（合作伙伴）业务中的组。如果这是一个问题，请考虑使用如**图 6-3** 中所述的倒转的因特网域名称空间约定。请注意，这对分析建模*本身*可能不是一个大问题，但是如果您采用的是让分析模型*在原处*演变为设计模型的方法，并且预计设计级别的复用或业务集成，您就可能要提前计划了。采用这种方法的另一个潜在的优势是：因为它可能很好地映射到从分析 / 设计生成的代码的组织，它就可以简化生成代码的转换的后续配置。

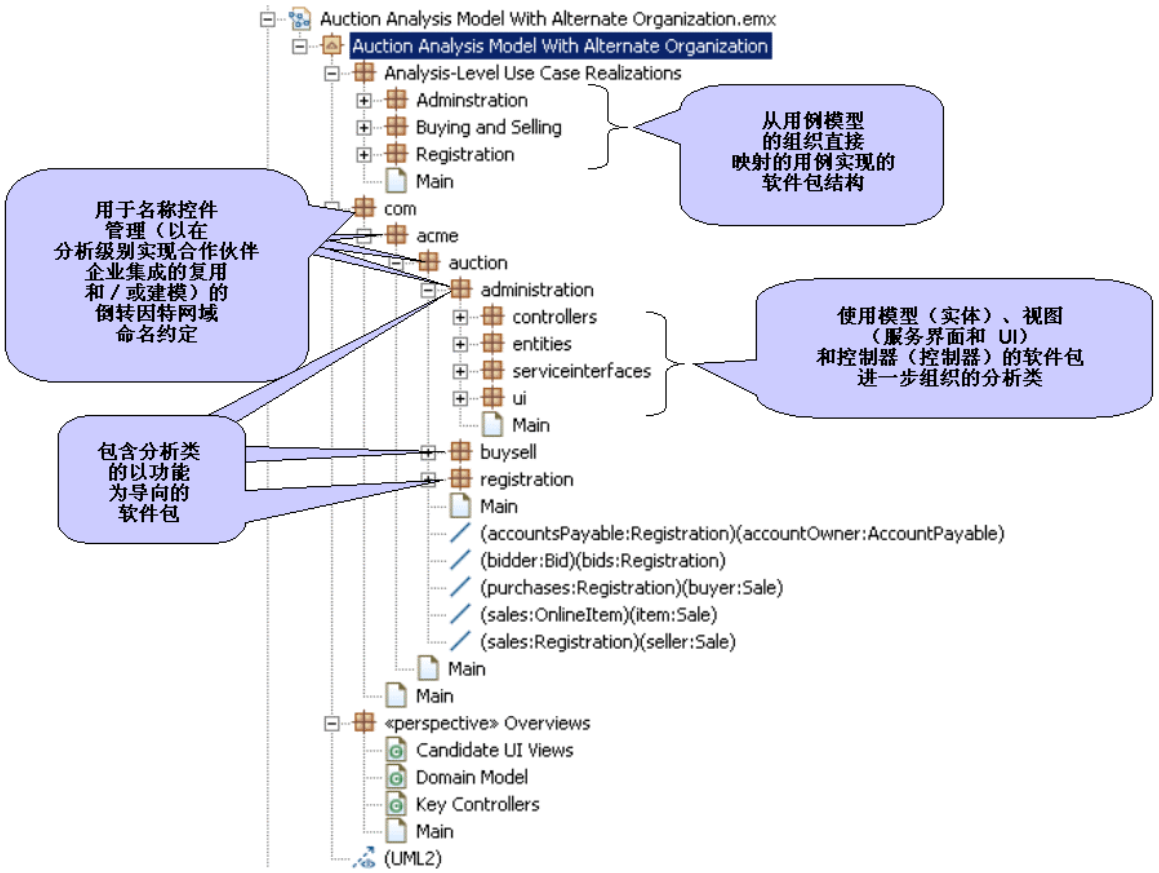


图 6-3

分析模型内容

发现分析类是什么有多种方法。一种方法是开始画建议用例实现的序列图。在这么做的时候，您会发现需要什么生命线，而且一般每条生命线都与一个可能的分析类相对应。当您用这种方法发现类时，您可能会在分析类的用例实现软件包种创建它们，但是您不应该把它们留在那里。您应该“重构”模型，将分析类移动到以功能为导向的软件包，如较早时在分析模型的高级组织指南中所描述的那样（请参阅 ***图 6-1*** ）。

发现分析类的另一个有用的方法：根据以下凭经验得出的规则用类来“种植”分析模型：

- 对于每个用例（在用例模型中）将《控制》类添加到分析模型。《控制》类代表与用例相关的业务逻辑。（稍后，在设计中，它们还将映射到会话管理这样的事务中。）

- 对于每个参与者 / 用于关系（在用例模型中）将《边界》类添加到分析模型。《边界》类代表解决方案和人类参与者之间或解决方案和某个外部系统之间的接口。对应于人类参与者的《边界》类可能最终会映射到设计和实施中的一个或多个用户界面工件。对应于外部系统的《边界》类可能最终会映射到设计和实施中的某种适配器层。

- 通过 CRC 卡分析或用例说明的词分析这样的过程来识别其它《控制》类（动词）和《实体》类（名词）。

当您使用这种种植方法来识别分析类时，您可以将类直接放置到以功能为导向的软件包，如较早时在分析模型的高级组织指南中所述（请参阅***图 6-1***）。

不管您如何来发现分析类，您几乎可以确定地认识到需要对您的原始功能软件包作出更改。

可选：使用分析类软件包中的二级软件包进一步组织那些软件包的内容（请参阅**图 6-4**）。

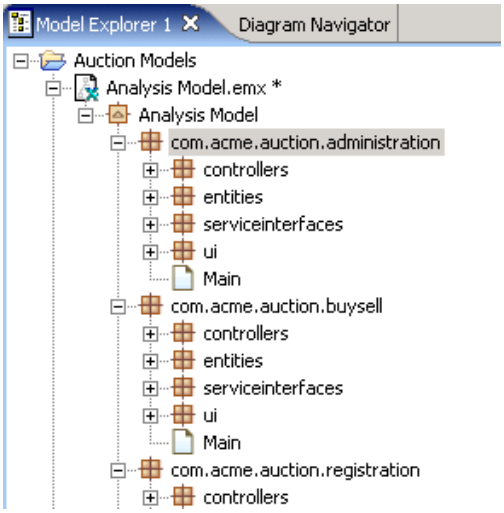


图 6-4

建议：分析模型应该包含分析级别的用例实现，这样的实现从分析类的角度说明用例如何执行。每个分析用例实现（用 UML 协作表示）在用例模型中实现用例并且拥有与该用例相同的名称。请参阅 **图 6-5**。有关每个命名的用例流程[6] 您感觉到的应该建模为分析级别的实现，要添加序列图（该序列图将自动添加拥有的交互）。**图 6-6** 显示在您创建序列图时将添加到模型中的语义内容的类型。（请注意，您可以从模型浏览器视图中过滤任何 UML 元素类型，因此隐藏了很多“杂波”，如 **图 6-6** 中所述。）

[6] 如前面在用例模型中所确立的，

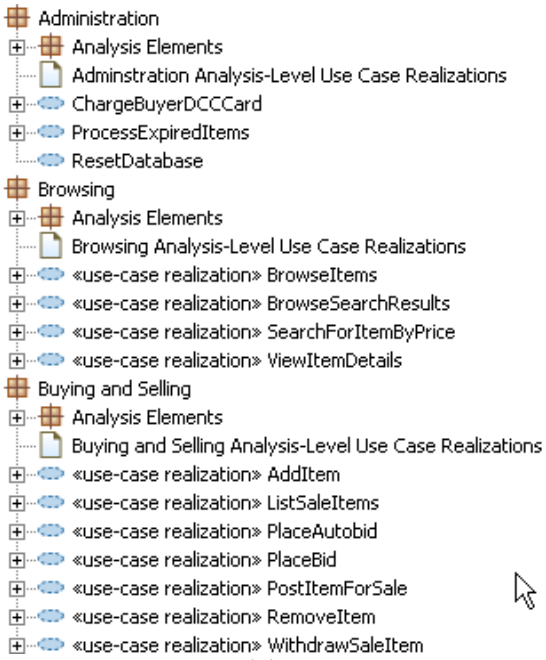


图 6-5

可选：一旦为用例流程创建了序列图，您就可以在模型浏览器中选择其拥有 UML 交互，并将通信图添加其中。新的通信图将自动用参与序列图的分析类实例进行填充。

建议：从每个用例实现（UML 协作）创建实现依赖关系，并从用例模型创建相应的用例（请参阅 **图 6-6** ）。因为您可以使用“主题图”和“可跟踪性分析”这样的功能部件来了解模型中的可跟踪性关系，您就不必保留永久图来描述可跟踪性关系，因此我们建议您用某种“废弃”图来创建关系，例如：

- 将自由格式图添加到协作。
- 将协作拖动到它上面。
- 将用例拖到上面。
- 画依赖关系。
- 最后，（在模型浏览器中）将图从协作中删除。

建议：包含每个用例实现的“参与者”图，来显示参与到实现中的分析类 （即，其实例出现在交互图上、描述用例实现的分析类）以及支持交互图中描述的协作的类之间的关系。请参阅 **图 6-6**。

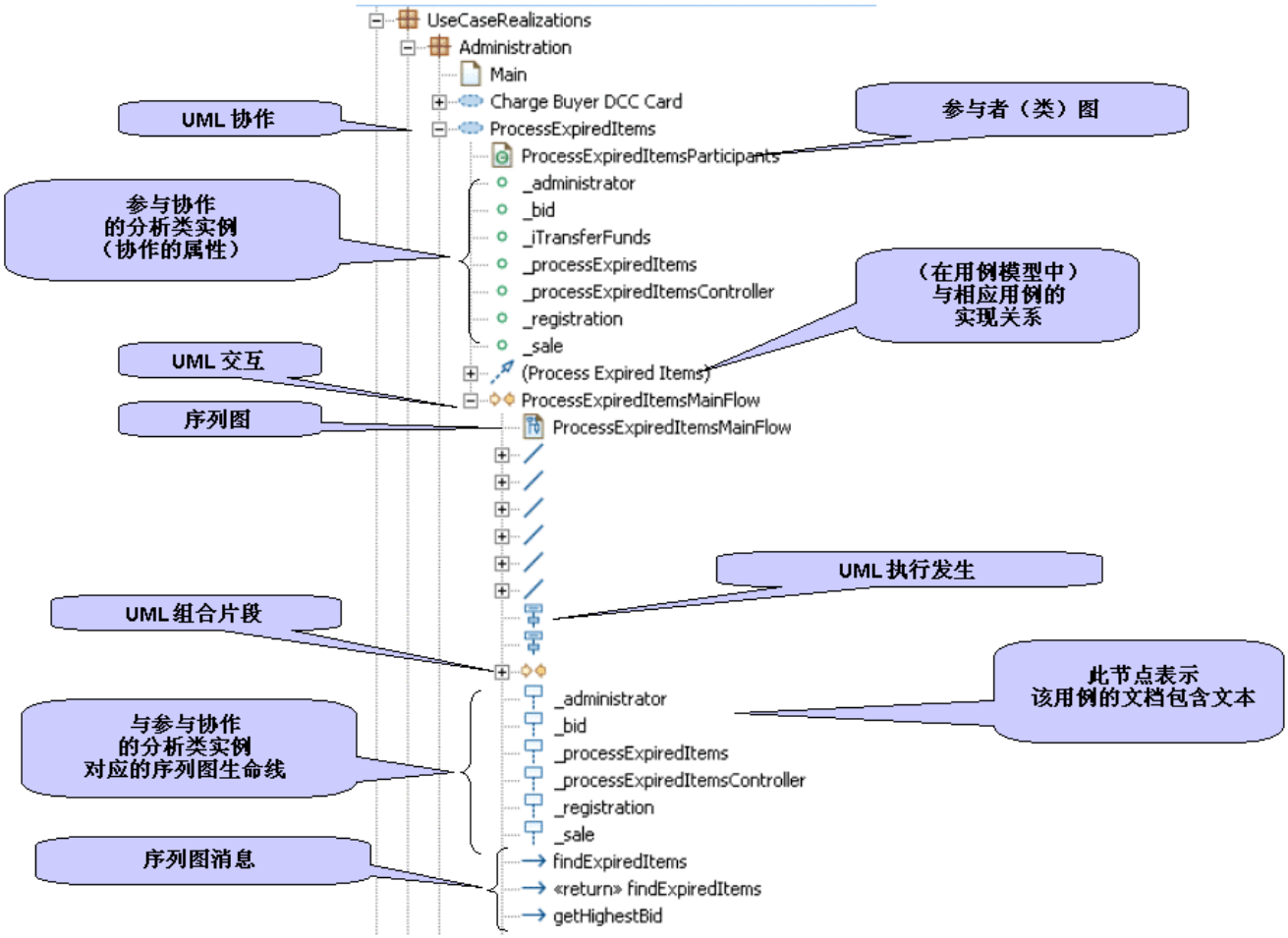
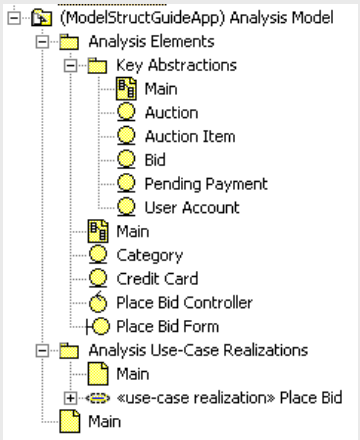


图 6-6

XDE／Rose

如下所示以前建议在**分析模型**中使用的结构为 RSA 进行了修改，将重点放在了分析类的以功能为导向的软件包组织上。同时请注意，关键抽象软件包（将牺牲一个以功能为导向的封装方法）的使用在《透视图》软件包中替换为关键抽象图的使用。



7. 设计模型的内部组织指南

设计 模型高级组织

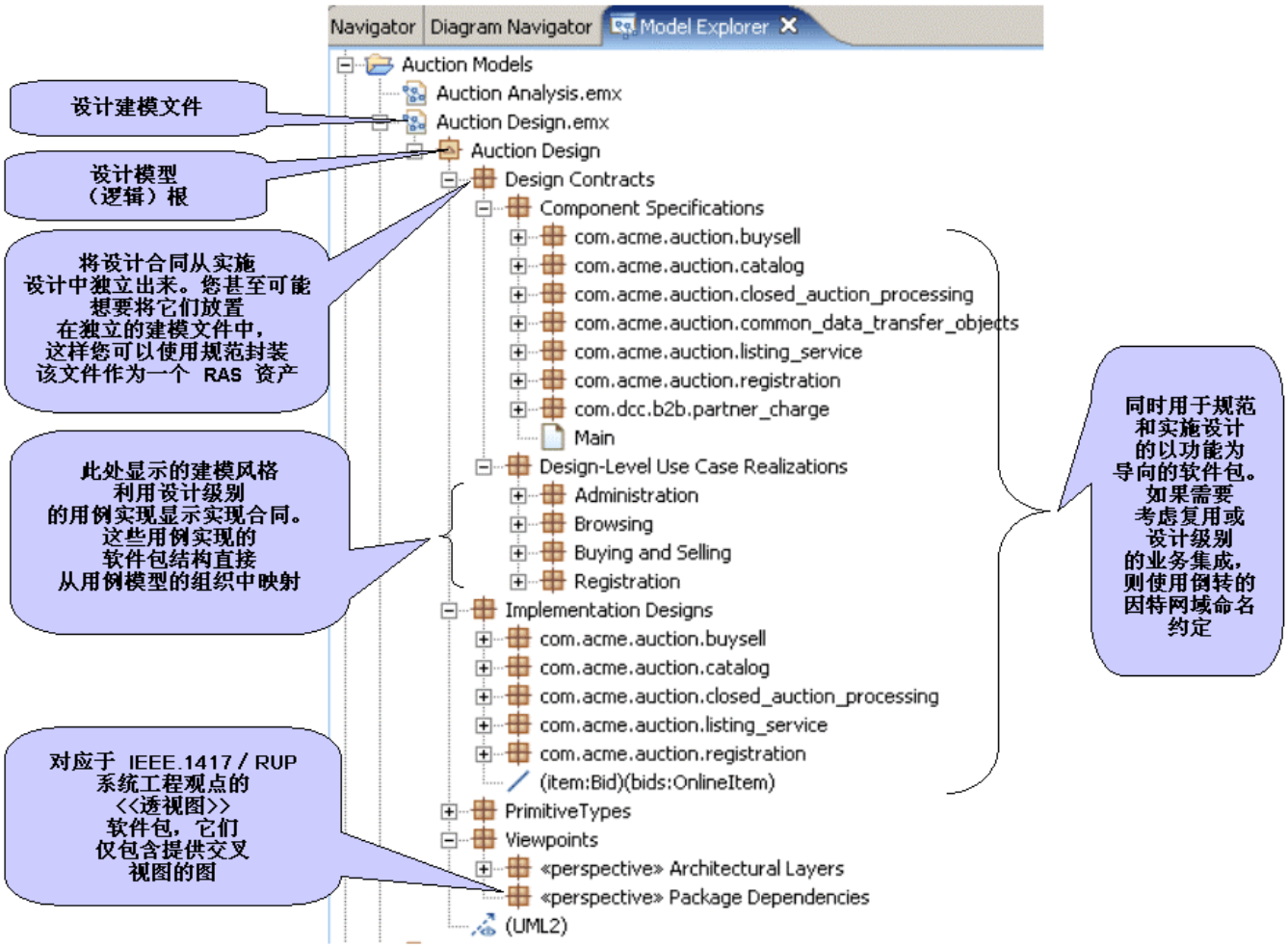


图 7-1

图 7-1 说明用于构建设计模型的以下指南：

1. 将规范从实施设计中独立出来。图显示了使用顶级“设计合同”和“实施设计”包来做到这一点。

2. 使用低级软件包来确立以功能为导向的分组。例如，您可以从在分析期间使用的组织开始，并让它随着您作出关于分析类如何映射到使用的设计类、组件和服务的决策进行演变。（任何初始的组织模式都可能在设计期间演变 —请参阅下面的深入讨论）。

   关于子系统的讨论在此时可能比较适宜。在 UML 2 以前的版本中，子系统是软件包的特殊类型。在 UML 2 中，子系统是组件的特殊类型，组件可以包含软件包。在 UML 2 中，《子系统》组件是软件包的可行的组织 / 名称空间替代品，但 UML2 关于子系统对软件包的恰当使用的内容很模糊。建议：使

用详细程度级别的软件包（如某个应用程序的设计子系统）并保留子系统来代表体系结构的企业范围的视图中的所有应用程序（如 CRM 或 SCM）。

> ***XDE／Rose***
> 在撰写本书时，我们期望 Rose 和 XDE 模型导入工具会提供将 UML 1.x 子系统映射到 UML2 子系统或应用了《子系统¬ 关键字的软件包的选项。

3. 设计元素的组织可能演变为系统的用例如何组织之外的内容（在用例模型以及可能在分析模型中，如果要保留独立的分析模型）。使用软件包将设计合同进一步分成设计元素规范（使用合同），和设计级别的用例实现（实现合同）并为继续反映用例本身的组织的用例实现保留软件包子结构。

4. *考虑*将体系结构层用作组成功能领域的规范和实施设计的元素的二级组织模式的基础（并请参阅下面的深入讨论）。

5. 在将语义模型元素分组的 UML 组件和软件包中，将提供特定于该分组的视图的图放置其中。本指南关于该分组是基于以功能为导向的业务领域部分、基于体系结构层还是基于您所拥有的其它内容。使“缺省”图拥有与软件包或组件相同的名称，并使它显示软件包的内容的概览。这使一些图接近它们描述的内容，使浏览和了解模型变得更加容易。

6. 您可能希望在设计模型中使用倒转的因特网域名称空间。**理由：**
   - 基本上，原因与这样做从特定于语言的实施的角度看很重要相同：
     a. 涉及集成工作的场景，其中涉及了多个模型驱动的应用程序（特别对于合作伙伴公司）
     b. 复用场景
   - 这可能会简化转换到实施的后续配置（源到目标的位置和名称映射）。

7. *考虑*使用将在目标实施平台中有效的软件包名称，来避免名称空间映射的负担和潜在的混乱。（从很大程度上来讲，这只意味着“名称中不要使用空格或除了下划线以外的标点”。）

8. 软件包名称请用小写，使它们更容易从软件包中的类名区别开来。

9. *考虑*为接口和实现它们的组件或类使用不同的名称。将 ILoan 和 Loan 或者 Loan 和 LoanImpl 用作接口和实施的名称。这在模型中实际上是没有必要的，但在生成的代码中是个不错的注意，所以这是您可以让自己省去一些后续转换配置工作的另一个地方。

10. 在以下场景中，任何分析级别的内容（不会从中生成代码）应该隔离在构造型为《分析》的软件包中[7]。

    - A) 您必须选择绕过独立分析模型的使用，并且用分析级别的内容填充设计模型，并将该内容保留在抽象的分析级别，同时在同一模型中创建设计级别的内容，而且

    - B) 您将从 EIT 设计模型驱动模型到代码的转换。

11. 使用《透视图》软件包中的图来捕获设计元素的高级、交叉的视图。**理由：**提供交叉视图、“体系结构上有重要意义的”内容的视图、吸引不同涉众的视图，同时保持模型的语义元素组织成以功能为导向

[7] 这样的软件包将被转换绕过.

的分组。

认识到设计模型的封装结构将随时间演变，这一点很重要。最终，组织应该对应于您将体系结构构建为组件和服务的方式。然后，设计的这种*游戏结束*组织的方法一般会提供封装可复用资产的最佳可能方法，并提供从设计到项目和文件夹的集合，这些项目和文件夹将持有从设计生成的实施工件（代码、元数据、文档）。

但是，*初始*组织应该多多少少直接对应于您用于用例模型、然后在分析期间改进的组织方法[8]。实际上（如前一部分“分析模型的内部组织指南”所述），您可以选择让您的分析模型在原处演变为设计。换句话说，设计的初始组织往往会将紧密相关的和不紧密耦合的业务问题集合组成一组，并将交叉或可复用元素隔离开来。这种用于初始组织的方法证明很有效，因为：

- 如果您希望使用从分析或用例模型内容生成设计模型内容的转换，则源软件包到目标软件包的映射将简单而直接。

- 基于功能关系和软件包的不紧密耦合的初始组织方法明显是映射到最终面向组件的组织的最佳选择，意味着它应该减少您作为设计过程的一部分必须执行的重构量。

- 程序包的不紧密耦合可能会改进团队工作流，并且可能在设计重构成多个建模文件的情况下使复用变得更加简单。

当然可以使用其它方法，而且在一些情况下作为*游戏结束*组织还是建议使用的：

- 如果您以基于 J2EE 的 Web 应用程序（包括 EJB）为目标，设计的组织就可以预计 RSA 和 Rational Application Developer 关于 J2EE 项目的约定。[9] 特别是，您可能会选择定义与体系结构层相对应的顶级设计包（展示和业务，其中业务进一步分层为会话和域）。这显然不是个不特定于平台的方法，因此建议只有在您知道您在设计的解决方案将不会在 J2EE 之外的平台上实施时才使用这种方法。

- 更一般地讲，在构建 n 层应用程序的地方经常出现的情况是开发人员专长和工作分区对应于展示和业务层，因此您可能还是要选择使用与那些体系结构层相对应的顶级软件包。但是要小心组织用来支持特殊*业务功能*来支持特殊*体系结构*的类。它会使两者中的任何一个都会变得更难更改。

- 如果您找到了使用以非组件 / 服务 / 子系统为导向的原因，您应该还是能够通过在配置代码生成转换中多做一些工作讲设计的组织映射到一组目标项目和文件夹。一种特殊类型的称为“映射模型”的同类模型可以用来定义特别复杂的映射。

设计模型内容

对于设计模型中应该有什么没有严格的规则，但是以下建议可能会有用。

[8] 分析类的封装在被发现时往往会有重大的重构，以更好地支持复用和未预料的功能要求。

[9] 不严格地讲：每个系统或应用程序或大的子系统一个企业项目，对于每个企业项目，展示层一个 Web 项目，以及每个组件或子系统多个 EJB 项目，其中 EJB 项目一般与组件或小的子系统相对应，而且一般独立的 EJB 项目用于会话（会话 EJB）和域（实体 EJB）。请参阅本白皮书的第 9 部分获得更多信息。

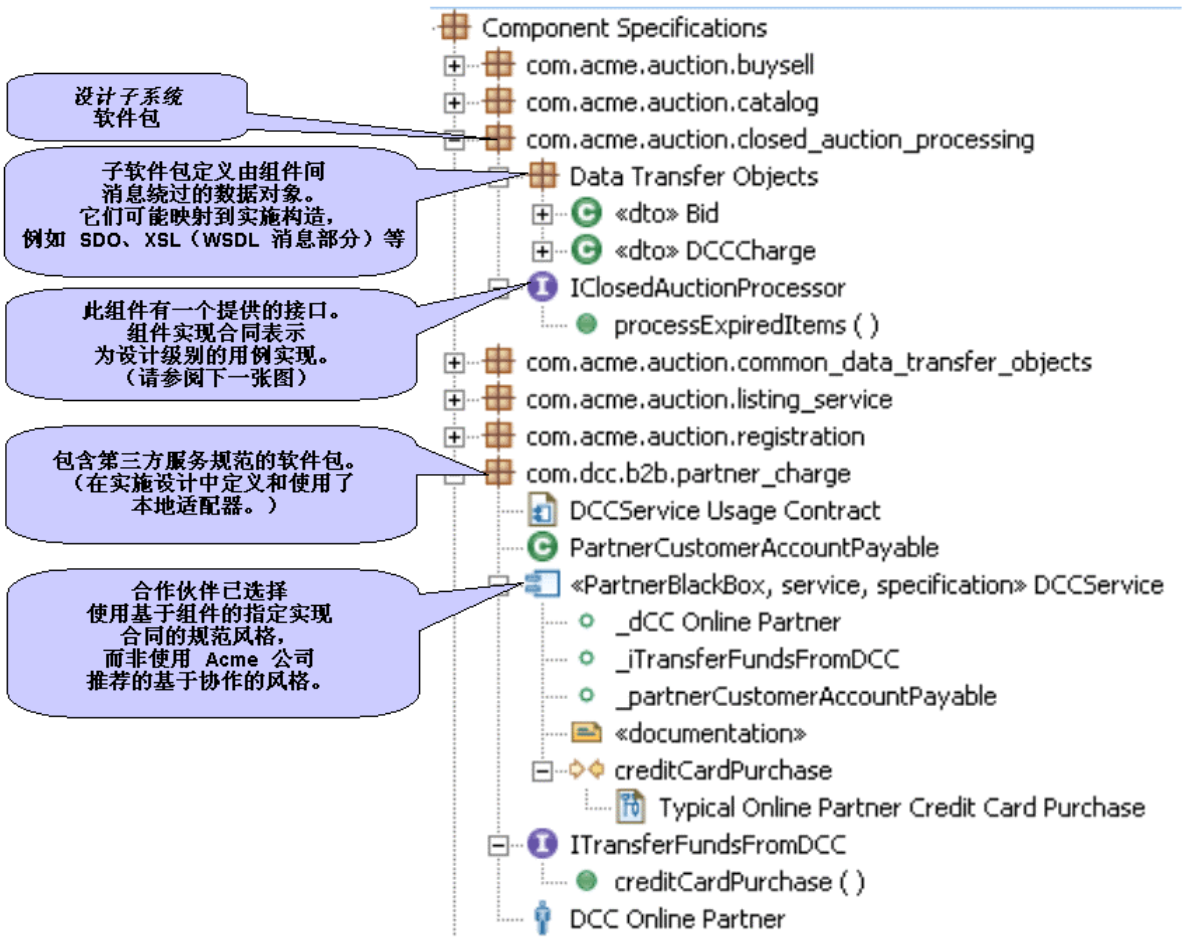

设计子系统
软件包
子软件包定义由组件间
消息绕过的数据对象。
它们可能映射到实施构造，
例如 SDO、XSL（WSDL 消息部分）等
此组件有一个提供的接口。
组件实现合同表示
为设计级别的用例实现。
（请参阅下一张图）
包含第三方服务规范的软件包。
（在实施设计中定义和使用了
本地适配器。）
合作伙伴已选择
使用基于组件的指定实现
合同的规范风格，
而非使用 Acme 公司
推荐的基于协作的风格。
Component Specifications
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
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

图 7-2

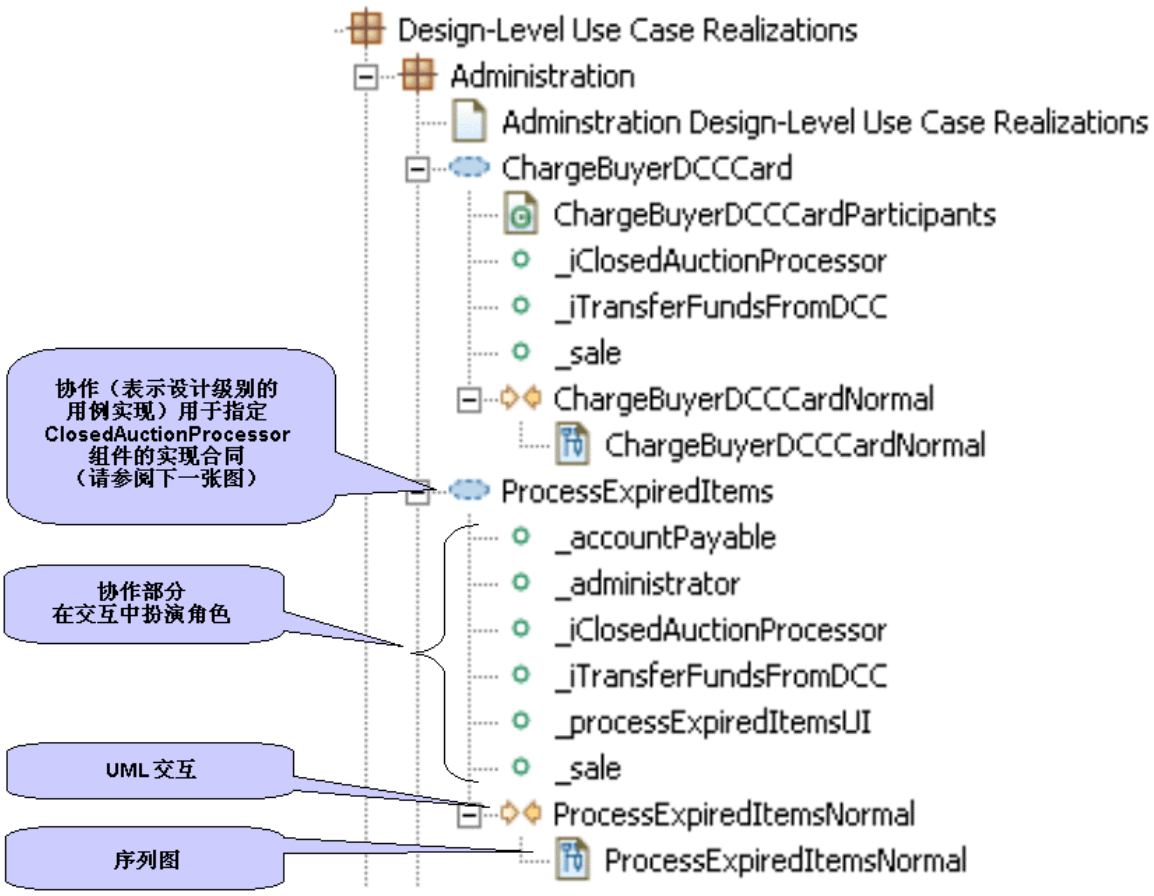


图 7-3

图 7-2 and **图 7-3** 遵守**图 7-11** 中描述的组织结构，并描述可以如何指定设计合同。

- “ClosedAuctionProcessor”组件的使用合同表示为单个接口[10] (**图 7-2**）。相应的实现合同由单个设计级别的、称为“协作”的用例实现指定[11] (**图 7-3**）请注意，分析级别的用例实现显示分析类之间的协作，而设计级别的实现显示不太抽象的设计元素之间的协作[12]。如果期望设计模型的规范子集独立于实施设计子集封装，那么设计级别的用例实现最好只在角色中使用分析或设计规范元素—而不是实施设计元素。
- 第三方“DCCService”的使用和实现合同一起装在一个软件包中。[13]。我们再次看到使用合同由单个接口组成，但是在这种情况下，实现合同用《规范》组件表示（**图 7-2**）。（否则，实现合同的规范就是相同的，都用行为来表示 —在这种情况下，交互被称为“creditCardPurchase”）。另一个使用组件而不使用协作的示例显示在**图 7-4** 中。

[10] 组件当然可以有多个提供的接口，只是碰巧本示例只有一个接口。

[11] 其它组件可能会参与多个系统用例，所以它们的实现合同可能位于多个用例实现中。在这样的情况下，您也可以在与组件的接口相同的软件包中包含称为“{组件名称}使用的地方”的图，在图上您可以放置到组成那些用例的用例实现的各种图的链接。

[12] 另一个可能的区别：设计级别实现中的一些“参与者”图可能是描述组件连线的组件图，而不是（或者除了）为分析级别的用例实现建议的参与者类图。

- [13] 请注意，这里假设的情形是 DCC 公司向 Acme 公司提供 UML 规范，然后 Acme 将该规范合并到其设计模型中。那是倒转因特网域空间可以派上用场的场景类型。

- 操作在接口中定义，接口可以由《规范 》组件（如果使用）或者由实施设计中实施接口的分类器实现。

- 数据传输对象（将用作提供的操作的参数类型，并且可以映射到 XML 模式或 SDO 这样的实施构造）的规范也可以包括为使用合同的一部分。对于设计为不可分发的组件，您可以选择将数据传输对象指定为或者不指定为用作操作参数的类型的规范。对于可分发的服务（例如 Web Service），强制要求服务的操作不能是本地地址空间的引用对象，因此必须使用 DTO。

XDE／Rose

在先前版本的 UML 中，用例实现的指导信息是每个用例一个协作实例，而且每个重要的实现流程一个交互和序列图。

在 Rational Software Architect 中，您应该经常能只使用一个交互和图，因为 UML2 序列图现在支持备用执行路径的符号表示法。

同时，在 UML 2 中已经没有“协作实例”了。取而代之的是“协作使用”，它要求将“协作”作为其类型。因此在 Rational Software Architect 中，请使用“协作”来代表用例实现。

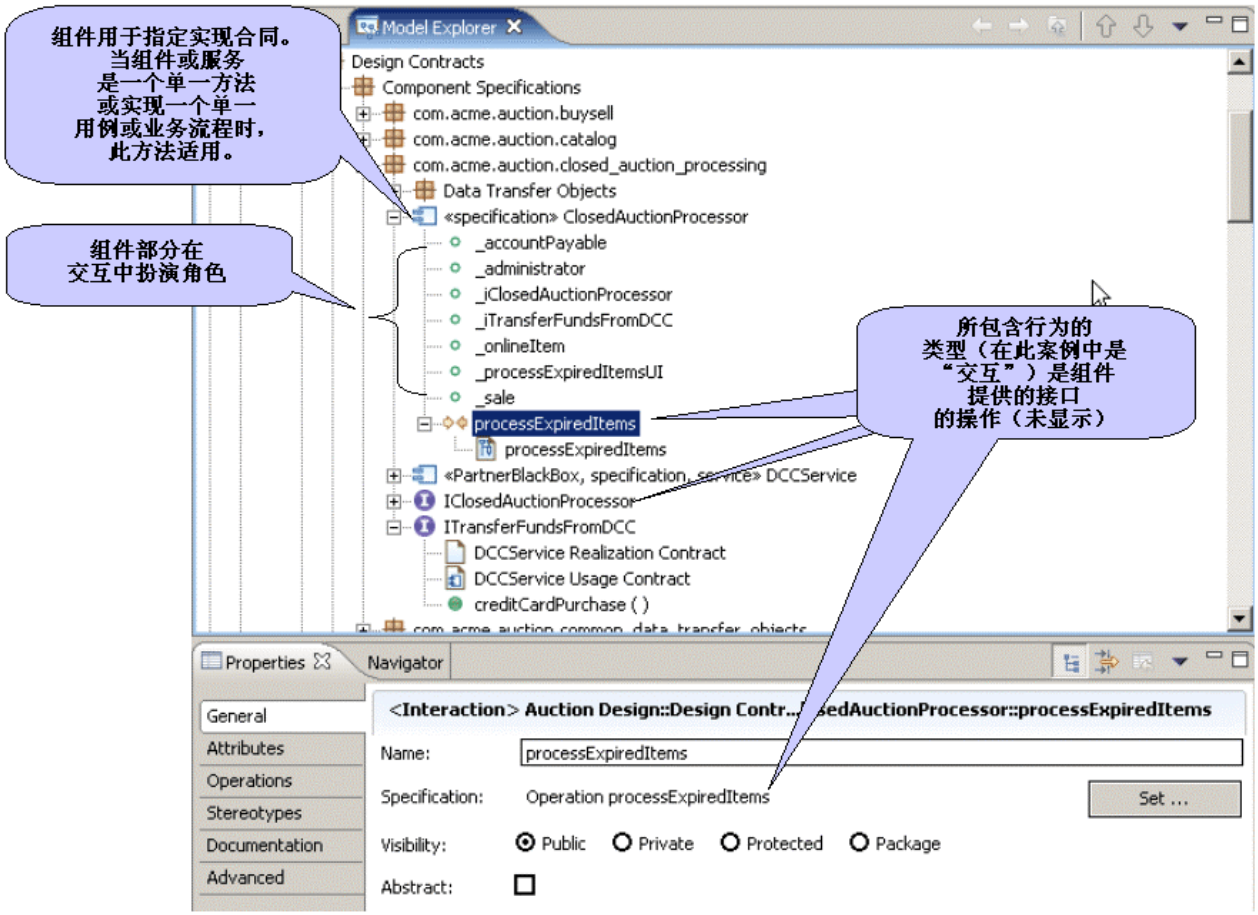


图 7-4

- 指定实施设计的可能的方法显示在**图 7-5** 中。实施结构是使用包含操作的简单类定义的。这种方法对用 UML 1.x 创建的设计模型非常典型。第二种可能的方法可能更注意保持 UML2 的目标，这种方法显示在**图 7-6** 中。在这里，我们看到的是“组件”而不是“类”，看到“组件”不拥有“操作”否，而是拥有行为（在这种情况下是“交互”）。

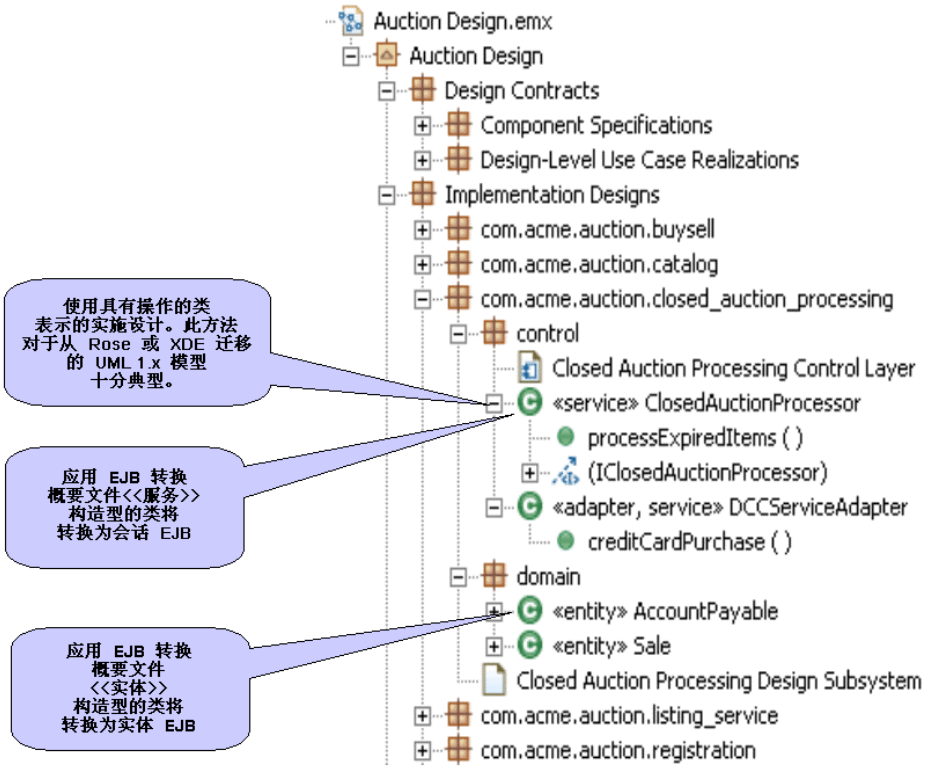

Auction Design.emx
Auction Design
Design Contracts
Component Specifications
Design-Level Use Case Realizations
Implementation Designs
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
control
Closed Auction Processing Control Layer
«service» ClosedAuctionProcessor
processExpiredItems ( )
(IClosedAuctionProcessor)
«adapter, service» DCCServiceAdapter
creditCardPurchase ( )
domain
«entity» AccountPayable
«entity» Sale
Closed Auction Processing Design Subsystem
com.acme.auction.listing_service
com.acme.auction.registration
使用具有操作的类表示的实施设计。此方法对于从 Rose 或 XDE 迁移的 UML 1.x 模型十分典型。
应用 EJB 转换概要文件<<服务>>构造型的类将转换为会话 EJB
应用 EJB 转换概要文件<<实体>>构造型的类将转换为实体 EJB

图 7-5

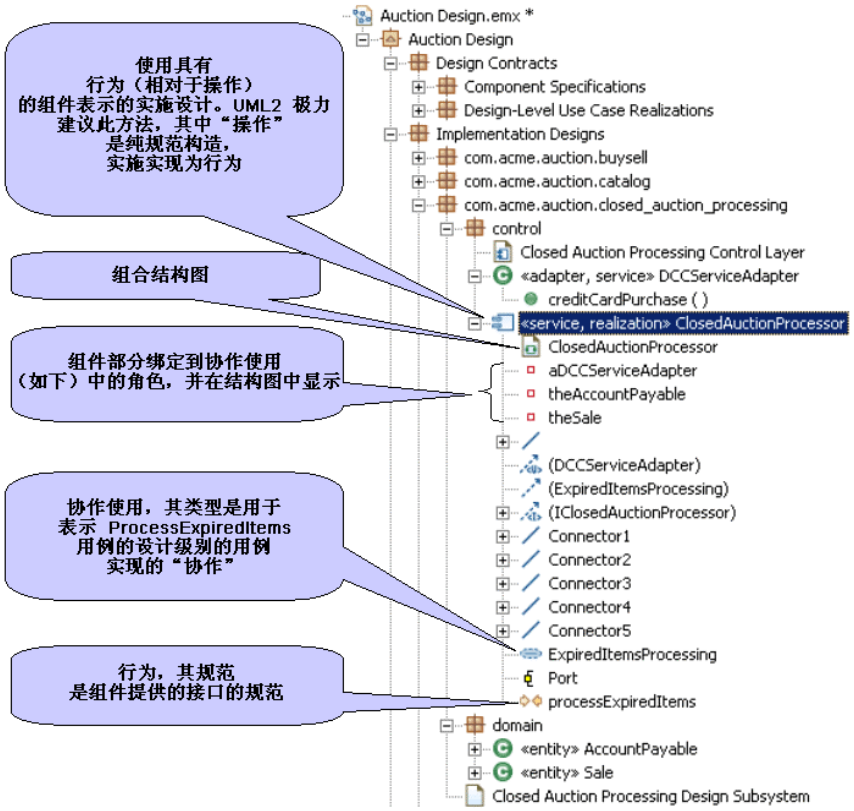

Auction Design.emx *
Auction Design
Design Contracts
Component Specifications
Design-Level Use Case Realizations
Implementation Designs
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
control
Closed Auction Processing Control Layer
«adapter, service» DCCServiceAdapter
creditCardPurchase ( )
«service, realization» ClosedAuctionProcessor
ClosedAuctionProcessor
aDCCServiceAdapter
theAccountPayable
theSale
(DCCServiceAdapter)
(ExpiredItemsProcessing)
(IClosedAuctionProcessor)
Connector1
Connector2
Connector3
Connector4
Connector5
ExpiredItemsProcessing
Port
processExpiredItems
domain
«entity» AccountPayable
«entity» Sale
Closed Auction Processing Design Subsystem
使用具有行为（相对于操作）的组件表示的实施设计。UML2 极力建议此方法，其中“操作”是纯规范构造，实施实现为行为
组合结构图
组件部分绑定到协作使用（如下）中的角色，并在结构图中显示
协作使用，其类型是用于表示 ProcessExpiredItems 用例的设计级别的用例实现的“协作”
行为，其规范是组件提供的接口的规范

图 7-6

8. 实施概览模型的内部组织指南

> ***XDE／Rose***
> 在 XDE 模型结构指南中，建议将实施概览模型用作提供实施的子系统级别的概览的方法。每个子系统的详细信息在实施子系统的项目的代码模型中指定。

严格地讲，应该没必要在 RSA 中使用实施概览模型。如果遵照设计模型组织指南，那么设计模型的（游戏结束）组织应该自然拥有围绕组件的形状（包括较强大的《子系统》和更可分发的《服务》变体）。然后，通过转换，设计的软件包可以映射到项目。例如，如果是 J2EE 实施，它们将映射到各种 Java、EJB、Web、J2EE 应用程序以及其它开发实施的项目中。（那些项目实际上代表解决方案的实施模型，如本白皮书的“基本概念和术语”部分所述。）

但是，您可能还是喜欢在较早时画出项目结构的草图，或者您可能只喜欢看到更直观的项目结构的描述 — 例如，代表项目和文件夹的工件用关键字直观地表示为《项目》和《文件夹》，或者甚至是《EJB 项目》和《Web 项目》的情况。另一个考虑事项是描述细颗粒实施工件（例如 JAR）对设计模型（在 Rational Software Architect 的操作理论中是独立于平台的）不适合。但这样的工件完全适合包含在实施概览模型中。这样，您想要使用实施概览模型就有一些原因。**图 8-1** 描述实施概览模型样本

最后的一个想法是，实施概览模型可能是捕获解决方案的各个方面的非正式图的好地方。**图 8-2** 显示拍卖系统的非正式高概念图，本白皮书的大多数样本是根据该图得出来的。

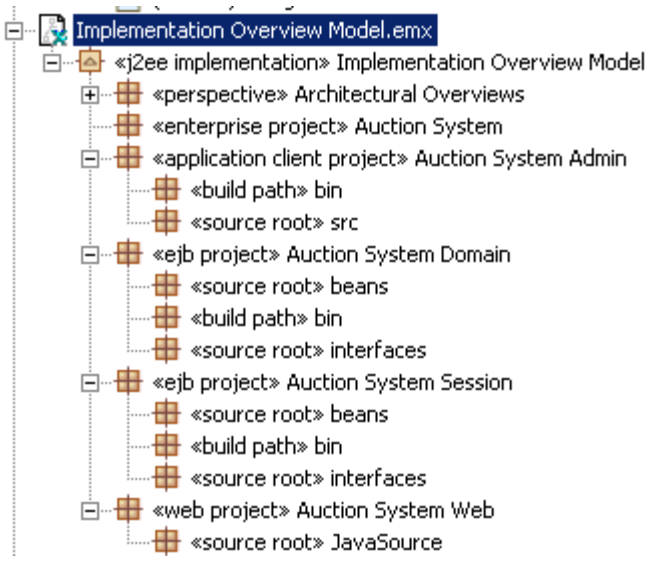


图 8-1

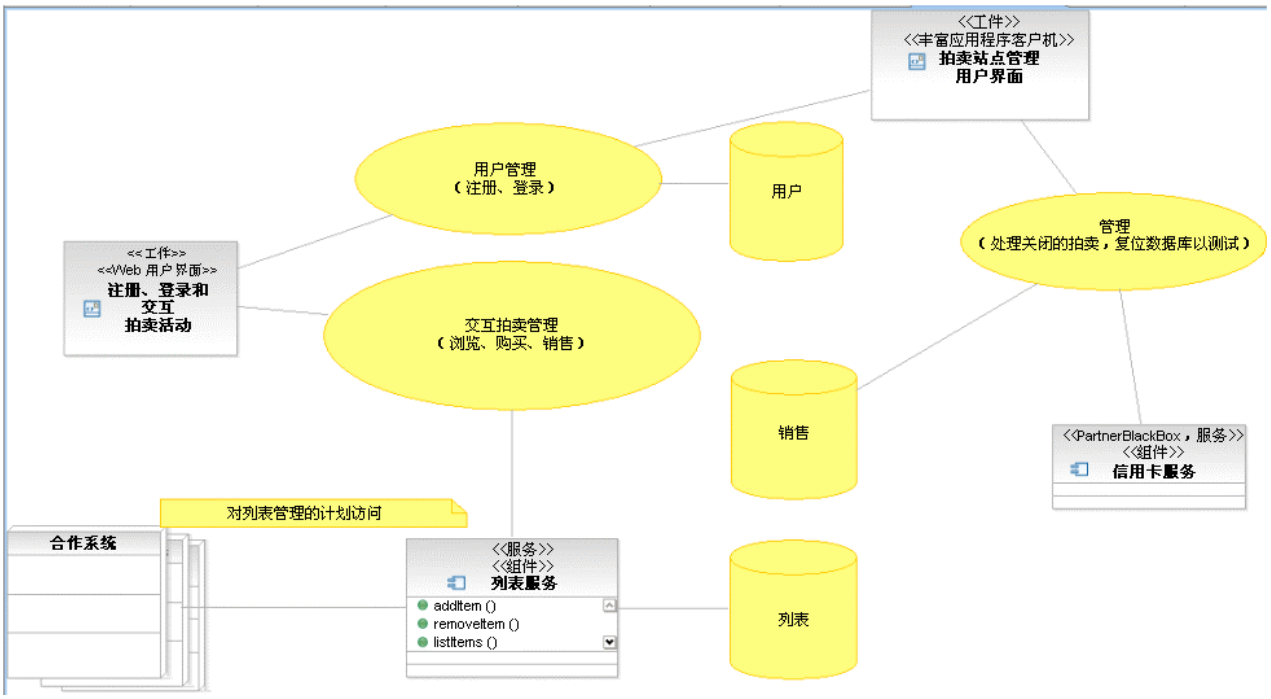

<<工件>>
<<丰富应用程序客户机>>
拍卖站点管理
用户界面
用户管理
（注册、登录）
用户
管理
（处理关闭的拍卖，复位数据库以测试）
<<工件>>
<<Web 用户界面>>
注册、登录和
交互
拍卖活动
交互拍卖管理
（浏览、购买、销售）
销售
<<PartnerBlackBox，服务>>
<<组件>>
信用卡服务
对列表管理的计划访问
合作系统
<<服务>>
<<组件>>
列表服务
addItem ()
removeItem ()
listItems ()
列表

图 8-2

9. 部署模型的内部组织指南

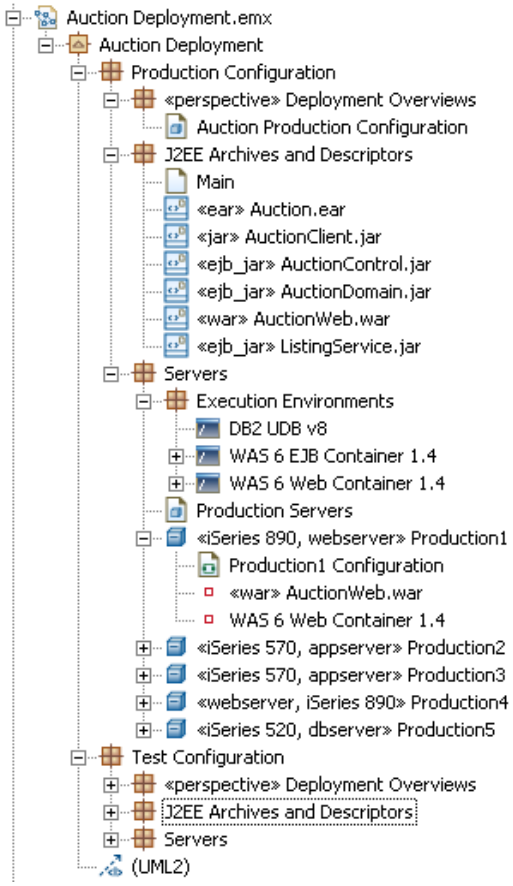


图 9-1

关于部署模型要说的内容可能比本白皮书中提到的任何其它模型都要少。您的部署建模组织和内容选项一般应该只有很少的下游含义，因此只要做有意义的就可以了。但 – 只是为了让您思考 – 在上述**图 9-1** 中还是描述了可能的策略和一些代表内容。在本示例中，请注意：

1. 生产配置的规范已经独立于配置测试的规范。

2. 概览（例如集群、数据中心或企业的概览）保留在《 透视图》软件包中。

3. 在将节点和工件中已经采用了轻量级的方法：封装和使用关键字的组合。更成熟的方法是开发特殊的 UML 概要文件定义用来描述和记录用在您自己的环境中的资源类型的特殊化的构造型和属性。

10. 使用建模文件来表示软件体系结构文档

考虑到它用于组织模型的工具（如图链接以及用跨模型引用对多个模型文件的支持），创建实际上代表 RUP 软件体系结构文档和“4+1 体系结构视图”的模型几乎是个小问题了。

用最简单的格式，您可以按照**图 10-1** 的内容执行一些操作。创建建模文件并用与 4+1 视图相对应的一组简单的软件包来填充它。（显示的示例没有用于进程视图的软件包，因为本示例中的系统在并行方面展示的信息不多。）

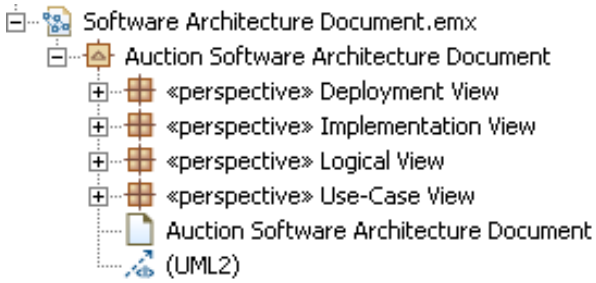


图 10-1

然后可能按照**图 10-22** 中的内容构建缺省图。您也可用将其它注释或文本添加到该图。

图 10-2

然后只要用以下方法在软件体系结构文档建模文件中创建图就可以了：

- 创建用来自其它建模文件的 UML 语义元素构建的、描述在那些其它建模文件中找不到的但作为体系结构文档的一部分所需的图。

- 创建由几何形状和 / 或位于软件体系结构文档建模文件的“即兴”UML 元素组成的图。（这样的 UML 元素应该仅用于记录或澄清的目的，应该对正在描述的解决方案的实际实施没有语义意义。）

- 创建仅包含到其它建模文件中现有图的链接的图。（该技巧在体系结构文档建模文件随其它建模文件一起分发供读者使用时能起很好的作用。如果体系结构文档将在 Web 上发布，请使用其它方法。）

11. 团队开发考虑事项

“基本概念和术语”部分讨论了各种模型，如 RUP 识别的“用例”、“分析”和“设计”。还给出了一个示例，说明 RSM 或 RSA 中的“实施前”模型可以保留为一个或多个建模文件，而且您可以保留多个用例模型、多个分析模型等，其中每个模型保留为单个建模文件或多个建模文件。本部分介绍您何时为为什么会选择将模型保留为多个建模文件的一些考虑事项。这些问题的更综合的处理方法可以在 RSA 联机帮助上找到。

团队建模

当多个个人必须同时处理一个模型的工作时，您可能会发现将模型分成多个建模文件（.emx 文件）会更有效。这样做使团队更可能在工作时检查建模文件的互斥存取。这反过来降低了由于两个或多个个人修改同一个建模文件而造成的必须执行合并的机会。

另一方面，还有一个折衷方法。虽然在使用多个建模文件时较少要求合并，但它们可能偶尔还是必须进行。合并对建模*文件*进行单个处理，而不是成批处理。换句话说，当模型存储为多个建模文件时，单个文件“独立于”整个（逻辑）模型的环境进行处理。当合并会话能更多地访问完整模型的信息内容时，合并能更好地起作用 —换句话说，当模型分隔在*较少的*建模文件中时。

最后，将模型分成多个文件不像将模型用以下方式构建模型那么重要：这样的方式使多个团队成员能处理这些模型的工作，而不会带来必须在合并期间解决的冲突更改。我们建议您仅在能大大降低执行合并的需要时将模型的（*逻辑实例*）分成多个物理文件。一般情况下，您仅在团队成员可以成功地*独占式*（即，不管何时只有一个团队成员检出文件）或*隔离式*（即，每个团队成员都可以对单个文件作出更改，而不要求访问组成模型的完整逻辑环境的其它文件）用单个文件工作可以降低执行合并的需要。

模型分区的两个方法

在 RSM 和 RSA 中，您可以使用工具的重构模型的功能，作出即兴将逻辑模型示例分成多个物理文件的决定。但是，我们的建议是*提前计划*。以下段落将两种方法进行了比较和对比。

计划方法：一开始分解模型

尽力预计团队的共同需要，并相应地将您的模型构建成逻辑子集。例如：

- 通过查看功能专长如何在团队分析人员之间分发，将每个用例模型分解成独立的建模文件，并相应地将内容分解（在实质上在您本来可以使用软件包来组织模型中的实际用例的地方创建独立的文件）。例如，在卫生保健产品供应商应用程序中，您可能拥有一个包含患者登记用例的建模文件以及另一个用于订单处理用例的建模文件。或者您可以进一步将用于订单处理的建模文件分解成独立的建模文件，用于实验室订单、放射科订单、药房订单等等。

- 将每个分析模型分解成多个文件，这里建议的方法还是根据专长在团队成员间或者在文件领域自然发现的逻辑分组（两个经常一起出现的考虑事项）中的分配将内容分解。

- 根据您的体系结构的主要子系统、服务或组件将每个设计模型分解成多个文件（换句话说，根据您预计实施工作将如何分配给团队或个人）。

在每种情况下，对于每个模型，您也可以保留另一个建模文件，包含提供跨越分区（子系统 / 软件包 / 服务）界线的概览的共享 / 常用元素和图。

关于模型分区的其它指导信息可以在 RSM / RSA 联机帮助中找到

即兴方法：模型重构

在 RSM 和 RSA 中，您可以从一个文件中的整个模型开始为模型创建多个文件，然后“切断”模型分支来创建其它文件。即使您提前计划分区策略，该方法在您识别提供改进团队工作流的新机会时可能很有用。

出于本次讨论的目的，我们关心模型的逻辑内容，所以模型的切断的分支称为“子模型”。当子模型从包含它的模型被切断时，原始模型仍然保留为指向子模型的“快捷方式”。子模型不保留反指向原始模型的指示。缺省的情况下，新的子模型中的元素将不再位于原始父模型的名称空间。但是，在切断过程中还是提供将原始模型的名称空间传递给新的子模型的选项。有关将子模型从原始模型切断来创建子模型的步骤，请参阅 RSM / RSA 联机帮助。

跨文件引用

任何两个 RSA 建模文件都可以引用彼此的元素。这可以包含跨项目的引用。这些引用中的一部分将代表您显式创建的关系，如用例之间的依赖关系或类之间的关联。其它引用由 RSA 创建，并且在某些情况下是隐藏的。例如，可跟踪性链接可以通过应用转换来生成，那些链接创建为正式、可视的模型元素。再例如，RSA 图包含指向该图中描述的语义元素的引用（任何语义元素都可以出现再多个图中）。显示元素引用是“隐藏的”引用（即，您不会在“模型浏览器”中看到它们）。

在以下情况中，每个跨文件引用代表可以分解的点：

- 一个或多个交叉引用文件不能正常保存（例如，由于系统崩溃）。

- 文件在文件系统中移动，而 RSM / RSA 模型服务器并不知情。

因此，在计划将模型分解成多个文件时要记住，是否可能造成大量的跨文件引用的结果是值得考虑的。再次重申，将模型组织为支持组织单位（软件包或建模文件）的功能连贯和不紧密耦合对于团队开发经验会有所共享。