http://book.51cto.com/art/201007/215560.htm

.1 Rational Rose 的四种视图模型

在 Rational Rose 建立的模型中包括四种视图， 分别是用例视图 (Use Case View) 、逻辑视图 (Logical View) 、构件视图 (Component View) 和部署视图 (Deployment View) 。创建一个 Rational Rose 工程的时候，会自动包含这四种视图，如图 5-1 所示。

图 5-1 Rose 模型中的四种视图

每一种视图针对不同的模型元素，具有不同的用途。在下面的几个小节中将分别对这四种视图进行说明。

.1.1 用例视图（ 2）

用例图 (Use Case Diagram) 。在用例视图中，用例图显示了各个参与者、用例以及它们之间的交互。在用例图下可以连接与用例图相关的文件和 URL地址。在浏览器中选择某个用例图，右键单击，可以看到在该用例图中允许创建的元素，如图 5-7 所示。

图 5-7 用例图可以关联文件和 URL

类图 (Class Diagram) 。在用例视图下，允许创建类图。类图提供了结构图类型的一个主要实例，并提供了一组记号元素的初始集，供所有其他结构图使用。在用例视图中，类图主要提供了各种参与者和用例中对象的细节信息。与在用例图下相同，在类图下可以创建连接类图的相关文件和 URL地址。在浏览器中选择

某个类图，右键单击，可以看到在该类图中允许创建的元素，如图 5-8 所示。

图 5-8 类图下可以关联文件和 URL

协作图 (Collaboration Diagram) 。在用例视图下，也允许创建协作图，来表达各种参与者和用例之间的交互协作关系。与在用例图下相同，在协作图下可以创建连接与协作图相关的文件和 URL地址。在浏览器中选择某个协作图，右键单击，可以看到在该协作图中允许创建的元素，如图 5-9 所示。

图 5-9 协作图下可以关联文件和 URL

序列图 (Sequence Diagram) 。在用例视图下，也允许创建序列图，和协作图一样表达各种参与者和用例之间的交互序列关系。与在用例图下相同，在序列图下也可以创建连接与序列图相关的文件和 URL地址。在浏览器中选择某个序列图，右键单击，可以看到在该序列图中允许创建的元素，如图 5-10 所示。

图 5-10　序列图下可以关联文件和　URL

状态图 (Statechart Diagram)　。在用例视图下，状态图主要用来表达各种参与者或类的状态之间的转换。在状态图下也可以创建各种元素，包括状态、开始状态和结束状态以及连接状态图的文件和　URL地址等。在浏览器中选择某个状态图，右键单击，可以看到在该状态图中允许创建的元素，如图　5-11 所示。

图 5-11　状态图下可以创建的内容

活动图 (Activity Diagram)　。在用例视图下，活动图主要用来表达参与者的各种活动之间的转换。同样，在活动图下也可以创建各种元素，包括状态　(State) 、活动 (Activity)　、开始状态 (Start State)　、结束状态(End State) 、泳道 (Swimlane) 和对象 (Object)　等，还有包括连接活动图的相关文件和　URL地址。在浏览器中选择某个活动图，右键单击，可以看到在该活动图中允许创建的元素，如图　5-12 所示。

图 5-12　活动图下可以创建的内容

文件 (File)　。文件是指能够连接到用例视图中的一些外部文件。　　它可以详细介绍用例视图的各种使用信息，甚至可以包括错误处理等信息。

URL地址 (URL)。URL地址是指能够连接到用例视图的一些外部　　URL地址。这些地址用于介绍用例视图的相关信息。

在项目开始的时候，项目开发小组可以选择用例视图来进行业务分析，确定业务功能模型，完成系统的用例模型。客户、系统分析人员和系统的管理人员根据系统的用例模型和相关文档确定系统的高层视图。一旦客户同意分析用例模型，就确定了系统的范围。然后就可以在逻辑视图　　(Logical View)　中继续开发，关注在用例中提取的功能的具体分析。

5.1.2　逻辑视图

逻辑视图关注系统是如何实现用例中所描述的功能的，主要是对系统功能性需求提供支持，即在为用户提供服务方面，系统所应该提供的功能。在逻辑视图中，用户将系统更加仔细地分解为一系列的关键抽象，将这些大多数来自于问题域的事物通过采用抽象、封装和继承的原理，使之表现为对象或对象类的形式，借助于类图和类模板等手段，提供系统的详细设计模型图。类图用来显示一个类的集合和它们的逻辑关系有关联、使用、组合、继承关系等。相似的类可以划分成为类集合。类模板关注于单个类，它们强调主要的类操作，　并且识别关键的对象特征。　　如果需要定义对象的内部行为，　　则使用状态转换图或状态图来完成。公共机制或服务可以在工具类　(Class　Utility)　中定义。对于数据驱动程度高的应用程序，可以使用其他形式的逻辑视图，例如　E-R 图，来代替面向对象的方法　(OO Approach) 。

在逻辑视图下的模型元素包括类、类工具、用例、接口、类图、用例图、协作图、顺序图、活动图和状态图等。其中有多个模型元素与用例视图中的模型元素是相同的，这些相同的模型元素的界面请参考用例视图中的相关图示，这里只给出不重复的图形示例。充分利用这些细节元素，系统建模人员可以构造出系统的详细设计内容。在　Rational Rose　的浏览器中的逻辑视图如图　　5-13 所示。

图 5-13　逻辑视图

在逻辑视图中，同样可以创建一些模型元素。在浏览器中选择 Logical View(逻辑视图) 选项，右键单击，可以看到在该视图中允许创建的模型元素，如图 5-14 所示。

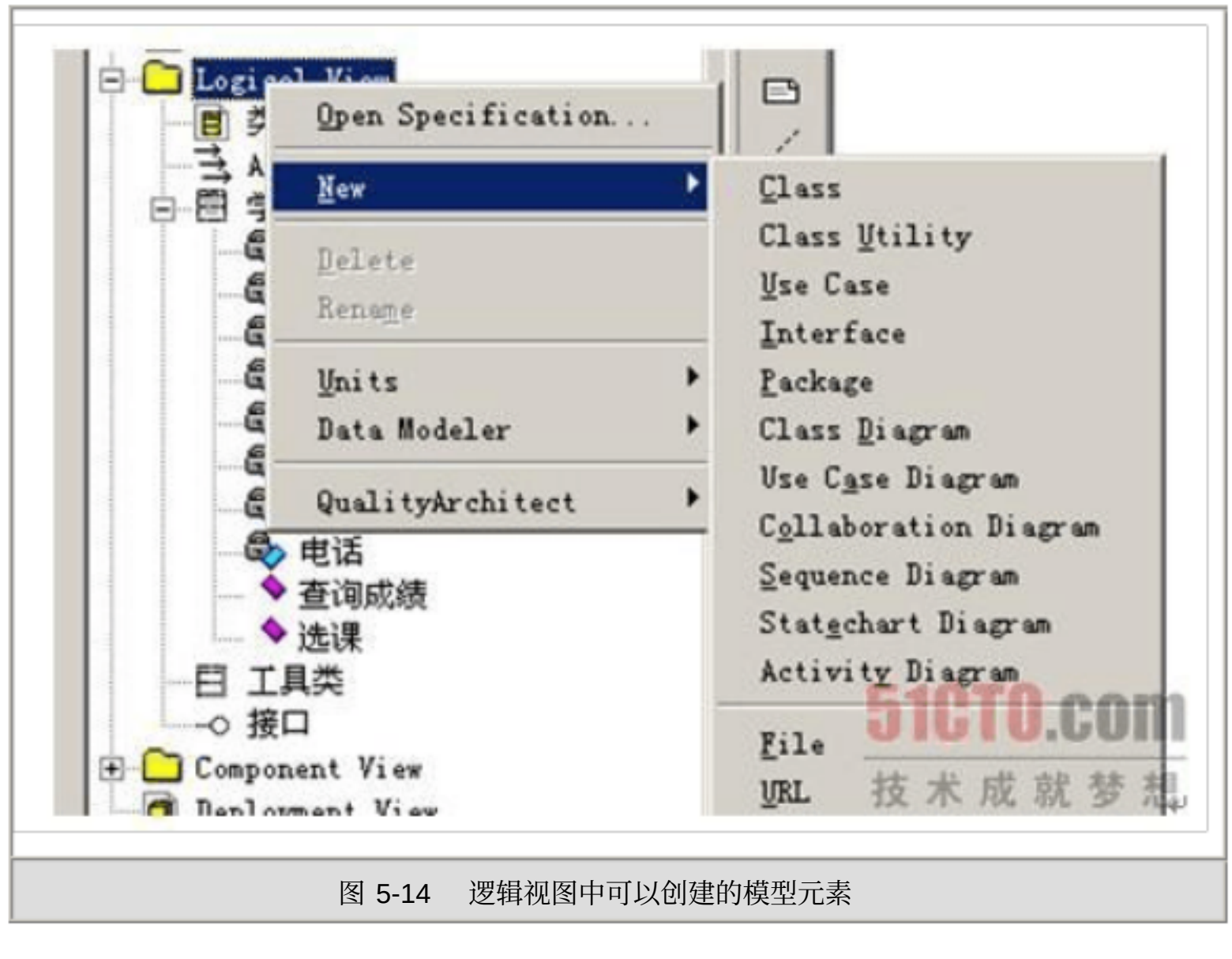

图 5-14　逻辑视图中可以创建的模型元素

类(Class) 。在逻辑视图中主要是对抽象出来的类进行详细的定义，包括确定类的名称、方法和属性等。系统的参与者在这个地方也可以作为一个类存在。在类下，也可以创建其他的模型元素，这些模型元素包括类的属性 (Attribute) 、类的操作 (Operation) 、嵌套类 (Nested Class) 、状态图 (Statechart Diagram) 和活动图 (Activity Diagram) 等，与前面在用例视图中创建的信息相同。

工具类 (Class Utility) 。工具类仍然是类的一种，是对公共机制或服务的定义，通常存放一些静态的全局变量，用来方便其他类对这些信息进行访问，如图 5-15 所示。

图 5-15　工具类下可以创建的模型元素

括协作图、序列图、类图、用例图、状态图和活动图等。在浏览器中选择某个用例，右键单击，可以看到在该用例中允许创建的元素。

接口 (Interface)　。接口和类不同，类可以有它的真实实例，然而接口必须至少有一个类来实现它。和类相同，在接口可以创建接口的属性　(Attribute)　、操作 (Operation)　、嵌套类 (Nested Class)　、状态图 (Statechart Diagram)　和活动图 (Activity Diagram)　等。在浏览器中选择某个接口，右键单击，可以看到在该接口中允许创建的元素，如图　5-16 所示。

图 5-16　接口下可以创建的元素

包(Package) 。使用包可以将逻辑视图中的各种　UML图或模型元素按照某种规则划分。在逻辑视图的包下，仍然可以创建所有能够在逻辑视图下创建的各种图和模型元素。

类图 (Class　Diagram) 。类图用于浏览系统中的各种类、类的属性和操作，以及类与类之间的关系。类图在建模过程中是一个非常重要的概念，必须了解类图的重要理由有两个：第一个是它能够显示系统分类器的静态结构，系统分类器是类、接口、数据类型和构件的通称；第二个理由是类图为　UML描述的其他结构图提供了基本标记功能。开发者可以认为类图是为他们特别建立的一张描绘系统各种静态结构代码的表，但是其他的团队成员将会发现它们也是有用的，比如业务分析师可以用类图为系统的业务远景建模等。其他

的图，包括活动图、序列图和状态图等，也可以参考类图中的类进行建模和文档化。与在用例视图下相同，在类图下也可以创建连接与类图相关的文件和 URL地址。在浏览器中选择某个类图，右键单击，可以看到在该类图中允许创建的元素。

用例图 (Use Case Diagram) 。在逻辑视图中也可以创建用例图，其功能和在逻辑视图中介绍的一样，只是放在不同的视图区域中罢了。与在用例视图下相同，在用例图下可以创建连接用例图的相关文件和 URL地址。在浏览器中选择某个用例图，右键单击，可以看到在该用例图中允许创建的元素。

协作图 (Collaboration Diagram) 。协作图主要用于按照各种类或对象交互发生的一系列协作关系，显示这些类或对象之间的交互。协作图中可以有对象和主角实例，以及描述它们之间关系和交互的连接和消息。通过说明对象间是如何通过互相发送消息来实现通信的，协作图描述了参与对象中发生的情况。你可以为用例事件流的每一个变化形式制作一个协作图。与在用例视图下相同，在协作图下也可以创建连接协作图的相关文件和 URL地址。 在浏览器中选择某个协作图， 右键单击， 可以看到在该协作图中允许创建的元素。

序列图 (Sequence Diagram) 。序列图主要用于按照各种类或对象交互发生的一系列顺序，显示各种类或对象之间的交互。它的重要性和类图很相似，开发者通常认为序列图对他们非常有意义，因为序列图显示了程序是如何在时间和空间中在各个对象的交互作用下一步一步地执行下去的。当然，对于组织的业务人员来讲，序列图显示了不同的业务对象如何交互，对于交流当前业务如何进行也是很有帮助的。这样，除了记录组织的当前事件外，一个业务级的序列图还能被当作一个需求文件使用，为实现一个未来系统传递需求。在项目的需求阶段，分析师能通过提供一个更加正式层次的表达，把用例带入下一层次。在那种情况下，用例常常被细化为一个或者更多的序列图，这样对于组织的技术人员来讲，序列图在记录一个未来系统的行为应该如何表现时非常有用。在设计阶段，架构师和开发者能使用序列图，挖掘出系统对象间的交互，如何充实整个系统设计，这就是序列图的主要用途之一，即把用例表达的需求转化为进一步、更加正式层次的精细表达。用例常常被细化为一个或者更多的序列图。序列图除了在设计新系统方面的用途外，还能用来记录一个存在系统 (称它为 " 遗产 ") 的对象现在是如何交互的。当把这个系统移交给另一个人或组织时， 这个文档很有用。 与在用例视图下相同， 在序列图下也可以创建连接序列图的相关文件和 URL地址。在浏览器中选择某个序列图，右键单击，可以看到在该序列图中允许创建的元素。

状态图 (Statechart Diagram) 。状态图主要用于描述各个对象自身所处状态的转换，用于对模型元素的动态行为进行建模，更具体地说，就是对系统行为中受事件驱动的方面进行建模。状态机专门用于定义依赖于状态的行为 (即根据模型元素所处的状态而有所变化的行为) 。其行为不会随着其元素状态发生变化的模型元素不需要用状态机来描述其行为 (这些元素通常是主要负责管理数据的被动类) 。状态机由状态组成，各状态由转移标记连接在一起。状态是对象执行某项活动或等待某个事件时的条件。转移是两个状态之间的关系，它由某个事件触发，然后执行特定的操作或评估并导致特定的结束状态。与在用例视图下相同，在状态图下也可以创建各种元素，包括状态、开始状态和结束状态以及连接状态图的相关文件和 URL地址等。在浏览器中选择某个状态图，右键单击，可以看到在该状态图中允许创建的元素。

活动图 (Activity Diagram) 。在一个活动图中可以包括以下元素。

活动状态，表示在工作流程中执行某个活动或步骤。

状态的转移，表示各种活动状态的先后顺序。这种转移可称为完成转移。它不同于一般的转移，因为它不需要明显的触发器事件，而是通过完成活动 (用活动状态表示) 来触发。

活动决策，为其定义了一组警戒条件。这些警戒条件决定在活动完成后将执行一组备选转移中的哪一个转移。我们也可以使用判定图标来表示线程重新合并的位置。决策和警戒条件使您能够显示业务用例的工作流程中的备选线程。

同步联接，用于联接平行分支流。 与在用例视图下相同， 在活动图下也可以创建各种元素， 包括状态 (State) 、活动 (Activity) 、开始状态 (Start State) 、结束状态 (End State) 、泳道 (Swimlane) 和对象 (Object) 等，还包括连接活动图的相关文件和 URL地址。在浏览器中选择某个活动图，右键单击，可以看到在该活动图中允许创建的元素。

文件 (File) 。文件是指能够连接到逻辑视图中的一些外部文件， 用来详细地介绍使用逻辑视图的各种信息。

URL地址 (URL)。URL地址是指能够连接到逻辑视图的一些外部 URL地址。这些地址用于介绍逻辑视图的相关信息。

在逻辑视图中关注的焦点是系统的逻辑结构。在逻辑视图中，不仅要认真抽象出各种类的信息和行为，还要描述类的组合关系等，尽量产生出能够重用的各种类和构件来，这样就可以在以后的项目中，方便地添加现有的类和构件，而不需要一切从头再开始一遍。一旦标识出各种类和对象并描绘出这些类和对象的各种动作和行为，就可以转入构件视图中，以构件为单位勾画出整个系统的物理结构。

5.1.3 构件视图

构件视图用来描述系统中各个实现模块以及它们之间的依赖关系。构件视图包含模型代码库、执行文件、运行库和其他构件的信息，但是按照内容来划分，构件视图主要由包、构件和构件图构成。包是与构件相关的组。构件是不同类型的代码模块，它是构造应用的软件单元，构件可以包括源代码构件、二进制代码构件以及可执行构件等。在构件视图中也可以添加构件的其他信息，例如资源分配情况以及其他管理信息等。构件图显示各构件及其之间的关系，构件视图主要由构件图构成。一个构件图可以表示一个系统全部或者部分的构件体系。 从组织内容上看， 构件图显示了软件构件的组织情况以及这些构件之间的依赖关系。

构件视图下的元素包括各种构件、构件图以及包等。在 Rational Rose的浏览器中的构件视图如图 5-17 所示。

在构件视图中，同样可以创建一些模型元素。在浏览器中选择 "Component View"(构件视图) 选项，右键单击，可以看到在该视图中允许创建的模型元素，如图 5-18 所示。

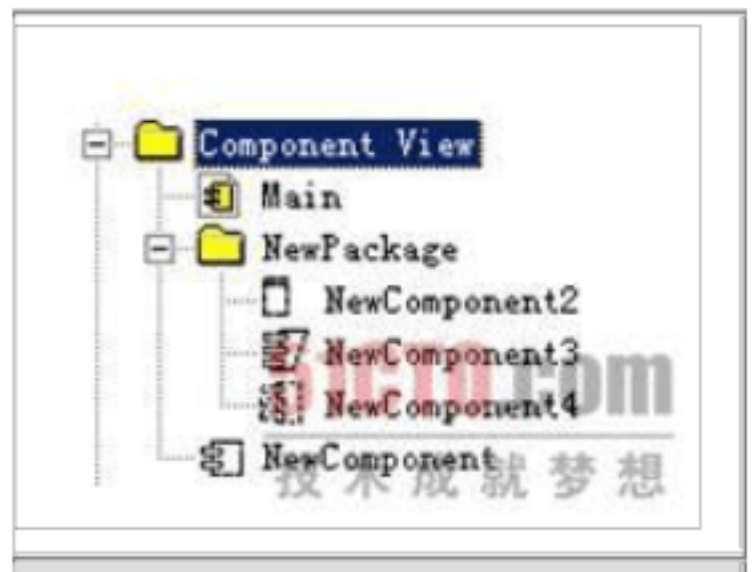

图 5-17 构件视图示例

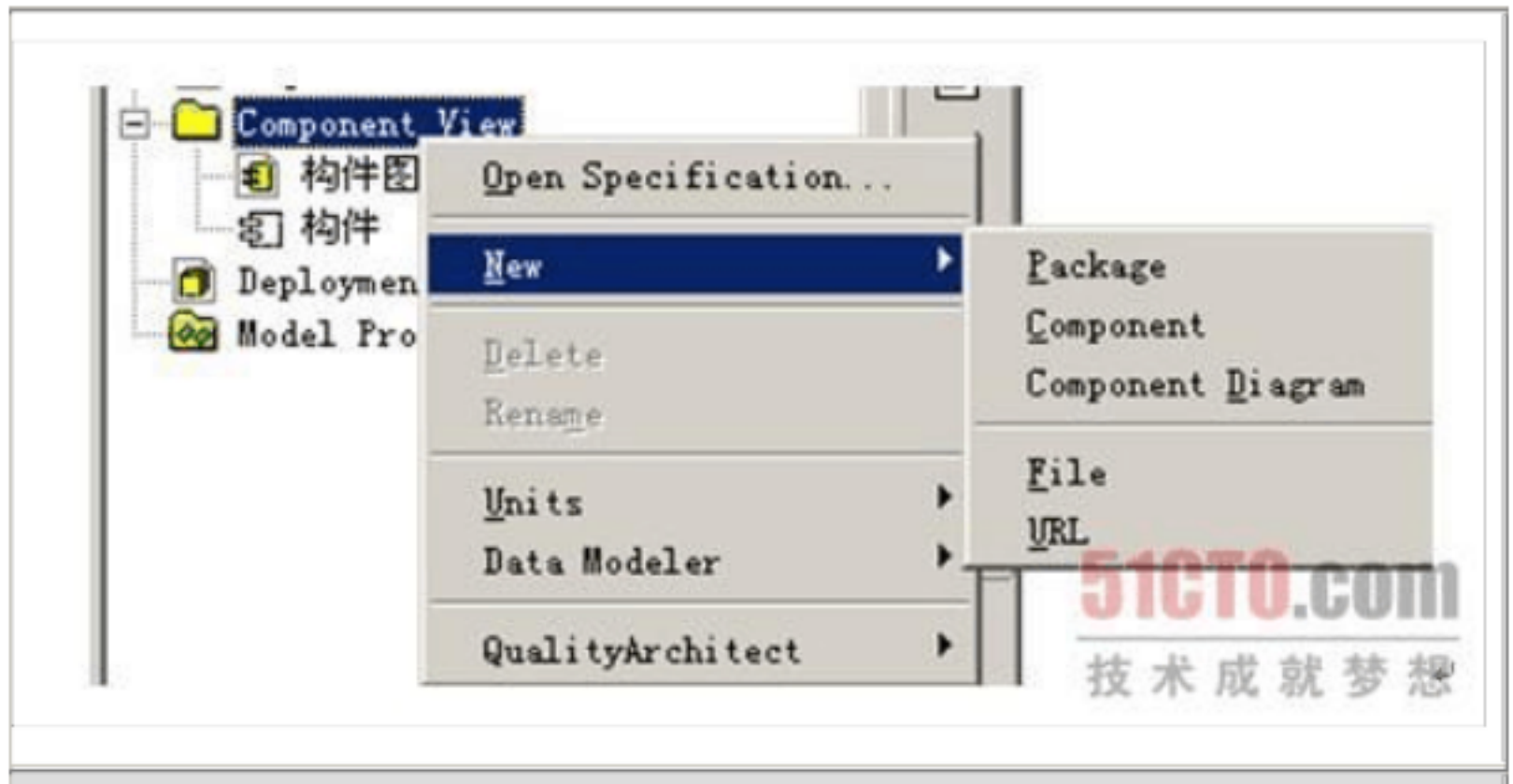

图 5-1 Rose 模型中的四种视图

包(Package) 。包在构件视图中仍然担当的是划分的功能。 使用包可以划分构件视图中的各种构件, 不同功能的构件可以放置在不同逻辑视图的包中。在将构件放置在某个包中的时候，需要认真考虑包与包之间的关系，这样才能达到在以后的开发程序中重用的目的。

构件 (Component) 。构件图中最重要的模型要素就是构件, 构件是系统中实际存在的可更换部分, 它实现特定的功能， 符合一套接口标准并实现一组接口。 构件代表系统中的一部分物理实施, 包括软件代码 (源代码、二进制代码或可执行代码) 或其等价物 (如脚本或命令文件)。在图中，构件使用一个带有标签的矩形来表示。在构件下可以创建连接构件的相关文件和 URL地址。在浏览器中选择某个构件，右键单击，可以看到在该构件中允许创建的元素，如图 5-19 所示。

构件图 (Component Diagram) 。构件图的主要目的是显示系统构件间的结构关系。在 UML1.1 中，一个构件表现了实施项目，如文件和可运行的程序。但是同时，构件这个术语通常和 COM构件这些更为普遍的指代相冲突。随着时间的推移及 UML连续版本的发布， UML构件已经失去了最初的绝大部分含义。在 UML 2中，构件正式改变了原本概念的一些本质意思，它被认为是在一个或多个系统或子系统中，能够独立地提供一个或多个接口的封装单位。虽然在 UML 2 中没有严格地规范它，但是一旦要呈现事物的更大设计单元的时候，这些事物一般是使用可更换的构件来实现的。现在，构件必须有严格的逻辑，设计时必须进行构造，其主要思想是能够很容易地在设计中被重用或被替换成一个不同的构件来实现，因为一个构件一旦封装了行为，实现了特定接口，那么这个构件就围绕实现这个接口的功能而存在，而功能的完善或改变意味着这个构件需要改变。在构件图下也可以创建连接构件的相关文件和 URL地址。在浏览器中选择某个构件图，右键单击，可以看到在该构件图中允许创建的元素，如图 5-20 所示。

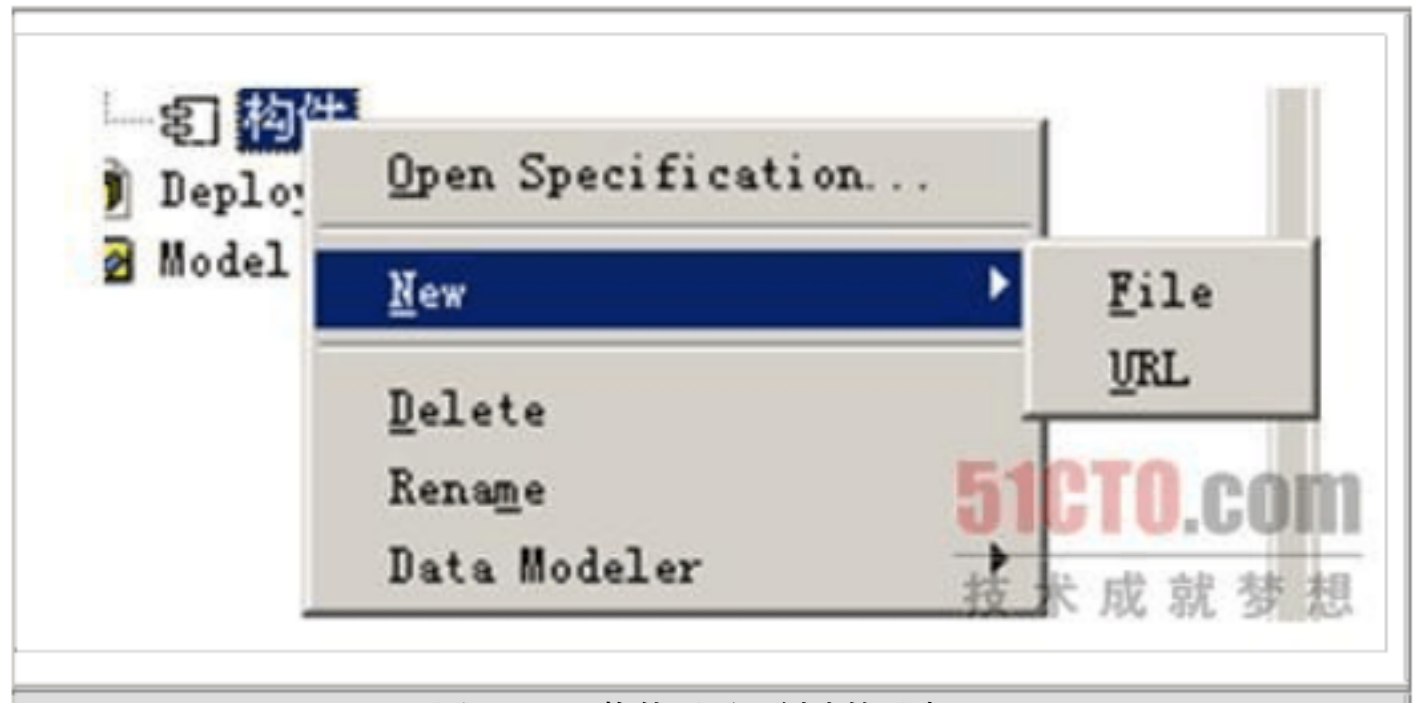

图 5-19　构件下可以创建的元素

图 5-20　构件图下可以创建的元素

文件 (File)　。文件是指能够连接到构件视图中的一些外部文件，　用来详细地介绍使用构件视图的各种信息。

URL地址 (URL)。URL地址是指能够连接到构件视图的一些外部　URL地址。这些地址用于介绍构件视图的相关信息。

在以构件为基础的开发　(CBD)中，构件视图为架构设计师提供了一个开始为解决方案建模的自然形式。　构件视图允许架构设计师验证系统的必需功能是否是由构件实现的，这样确保了最终系统将会被接受。除此之外，构件视图在不同小组的交流中还担当交流工具的作用。对于项目负责人来讲，当构件视图将系统的各种实现连接起来的时候，构件视图能够展示对将要被建立的整个系统的早期理解。对于开发者来讲，构件视图给他们提供了将要建立的系统的高层次的架构视图，这将帮助开发者开始建立实现的路标，并决定关于任务分配及　(或)增进需求技能。对于系统管理员来讲，他们可以获得将运行于他们系统上的逻辑软件构件的早期视图。虽然系统管理员将无法从图上确定物理设备或物理的可执行程序，但是，他们仍然能够通过构件视图较早地了解关于构件及其关系的信息，了解这些信息能够帮助他们这一些系统管理员轻松地计划后面的部署工作。如何进行部署那就需要部署视图来帮忙了。

5.1.4　部署视图

与前面的那些显示系统的逻辑结构不同，部署视图显示的是系统的实际部署情况，它是为了便于理解系统在一组处理节点上的物理分布。在系统中，只包含有一个部署视图，用来说明各种处理活动在系统各节点

的分布。但是，这个部署视图可以在每次迭代过程中都加以改进。部署视图中包括进程、处理器和设备。进程是在自己的内存空间执行的线程；处理器是任何有处理功能的机器，一个进程可以在一个或多个处理器上运行；设备是指没有任何处理功能的机器。如图 5-21 所示，显示了是一个部署视图结构。

在部署视图中，可以创建处理器和设备等模型元素。在浏览器中选择 "Deployment View"(部署视图) 选项，右键单击，可以看到在该视图中允许创建的模型元素，如图 5-22 所示。

图 5-21 部署视图示例

图 5-22 在部署视图中可以创建的模型元素

处理器 (Processor) 。处理器是指任何有处理功能的节点。节点是各种计算资源的通用名称，包括处理器和设备两种类型。在每一个处理器中允许部署一个或几个进程，并且在处理器中可以创建进程，它们是拥有自己内存空间的线程。线程是进程中的实体，一个进程可以拥有多个线程，一个线程必须有一个父进程。线程不拥有系统资源，只运行必须的一些数据结构；它与父进程的其他线程共享该进程所拥有的全部资源。可以创建和撤消线程，从而实现程序的并发执行。

设备 (Device) 。设备是指没有处理功能的任何节点，例如打印机。

文件 (File) 。文件是指能够连接到部署视图中的外部文件，用来详细地介绍使用部署视图的各种信息。

URL地址 (URL)。URL地址是指能够连接到部署视图的外部 URL地址，用于介绍部署视图的相关信息。

部署视图考虑的是整个解决方案的实际部署情况，所描述的是在当前系统结构中所存在的设备、执行环境和软件运行时的体系结构，它是对系统拓扑结构的最终物理描述。系统的拓扑结构描述了所有硬件单元，以及在每个硬件单元上执行的软件的结构。在这样的一种体系结构中，可以通过部署视图查看拓扑结构中任何一个特定的节点，了解正在该节点上组件的执行情况，以及该组件中包含了哪些逻辑元素（例如类、对象、协作等），并且最终能够从这些元素追溯到系统初始的需求分析阶段。