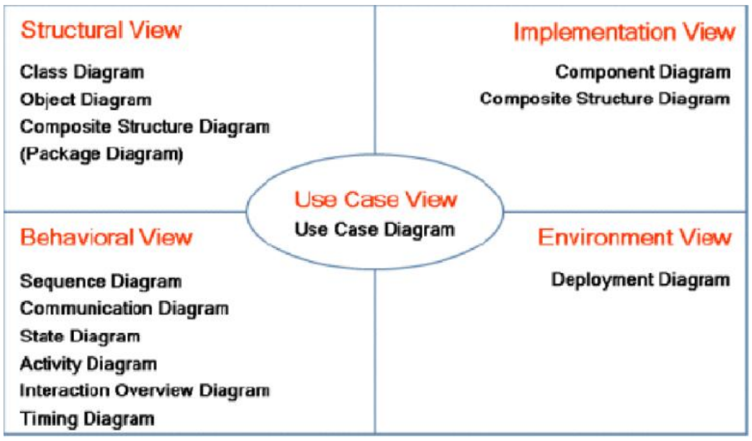
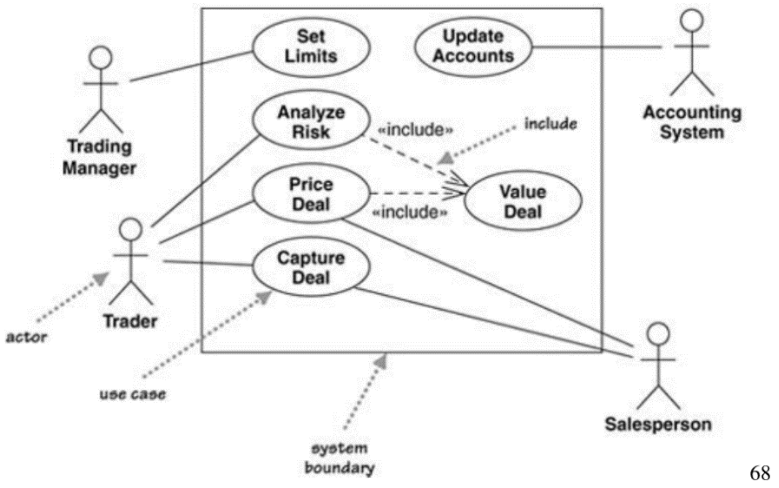
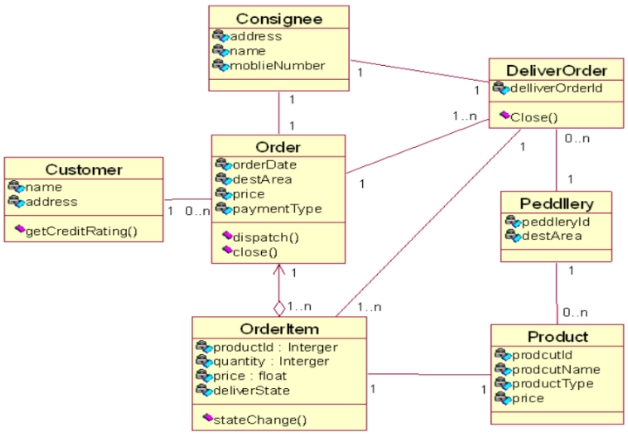


用例图 use case：用于显示若干角色以及这些角色与系统提供的用例之间的连接关系。用例是系统提供的功能的描述。Use cases are a valuable tool to help **understand the functional requirements of a system**. It is important to remember that use cases represent an external view of the system.

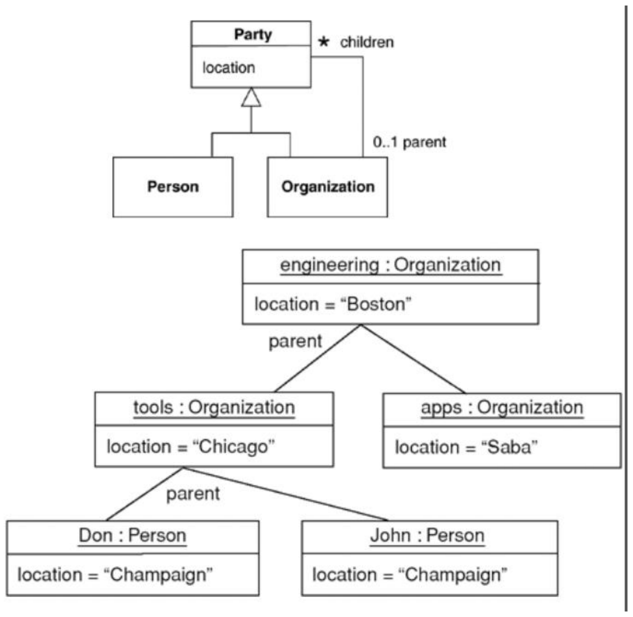


类图 class diagram：表示系统中的类和类与类之间的关系，它是对系统**静态结构**的描述；Class diagrams are the **backbone** of the UML

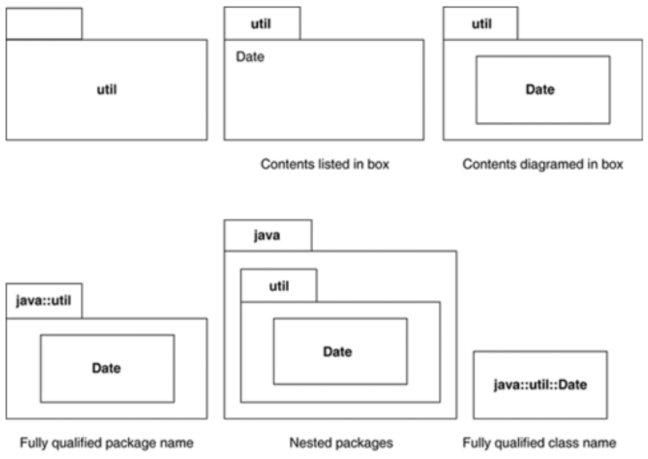


:

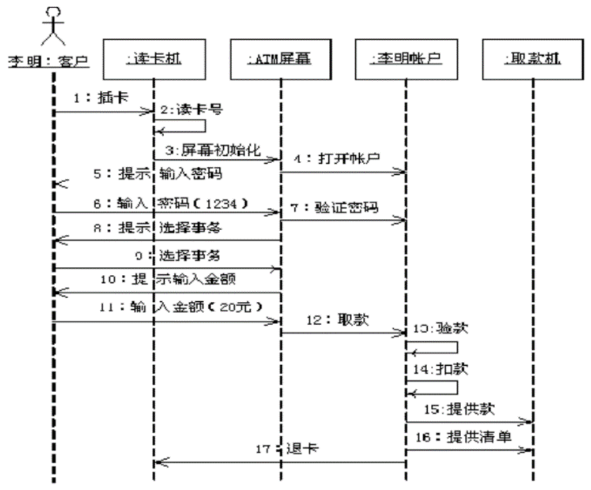
对象图 Object Diagram： object diagram is a snapshot of the objects in a system at a point in time. Because it shows instances rather than classes, an object diagram is often called an instance diagram.显示某个时间点的快照，不是类，是类的实例。Object diagrams are useful for showing examples of objects connected together 将对象连接起来



Package Diagram 包图：package is a **grouping construct** that allows you to take any construct in the UML and group its elements together into **higher-level units**. you can use packages for every other bit of the UML.

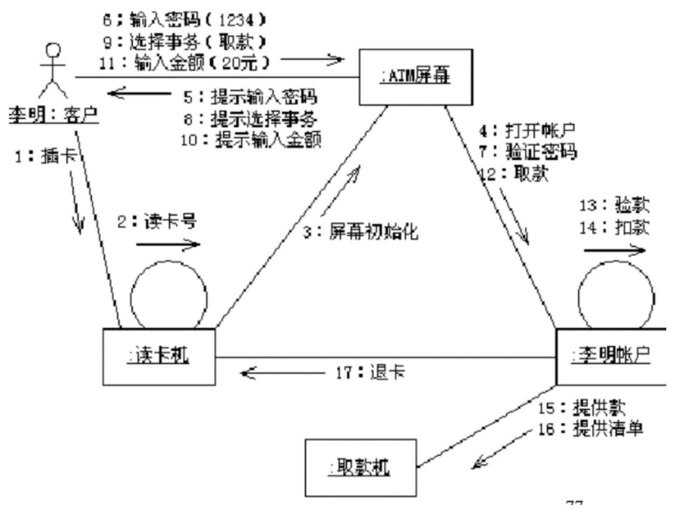


Sequence Diagram 序列图：用来反映若干个对象之间的动态协作关系，也就是随着时间的推移，对象之间是如何交互的

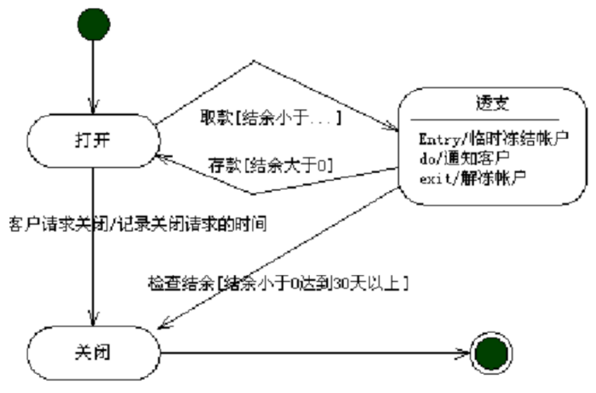


You should use sequence diagrams when you want to look at the **behavior of several objects within a single use case**. Sequence diagrams are good at showing collaborations among the objects; they are **not so good at precise definition of the behavior**.【序列图：一个用例中一系列对象的动作行为】
If you want to look at the **behavior of a single object across many use cases**, use a state diagram .【状态图：一个对象横跨多个用例的行为】
If you want to look at **behavior across many use cases or many threads,** consider an activity diagram【活动图：多个用例或线程之间的行为】

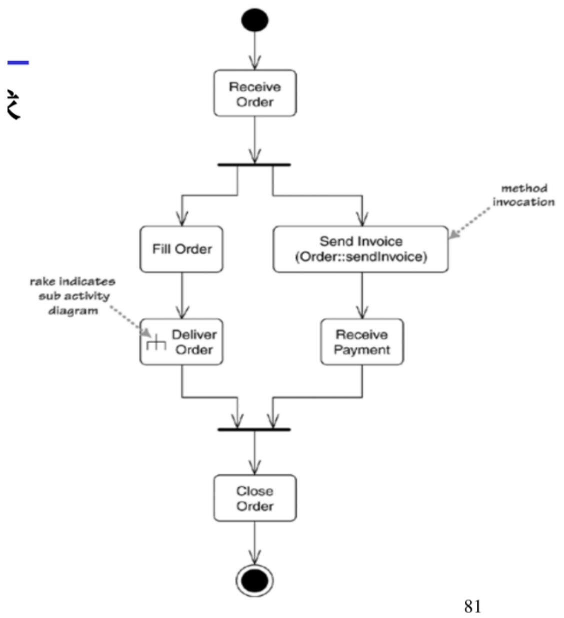
Communication Diagram：描述对象间的协作关系，协作图（通信图）跟序列图相似，显示对象间的动态合作关系。**如果强调时间和顺序，则使用序列图**；如果**强调上下级关系，则选择协作图**。这两种图合称为交互图；**sequence diagrams** are better when you want to **emphasize the sequence of calls** and that **communication diagrams** are better when you want to **emphasize the links**. Many people find that **communication diagrams** are **easier to alter** on a whiteboard, so they are a good approach for exploring alternatives.



State Diagram 状态图：描述类的对象所有可能的状态以及事件发生时状态的转移条件。通常，状态图是对类图的补充 State diagrams are good at describing the **behavior of an object across several use cases**. State diagrams are **not very good at** describing behavior that involves **a number of objects collaborating；don't try to draw them for every class in the system**

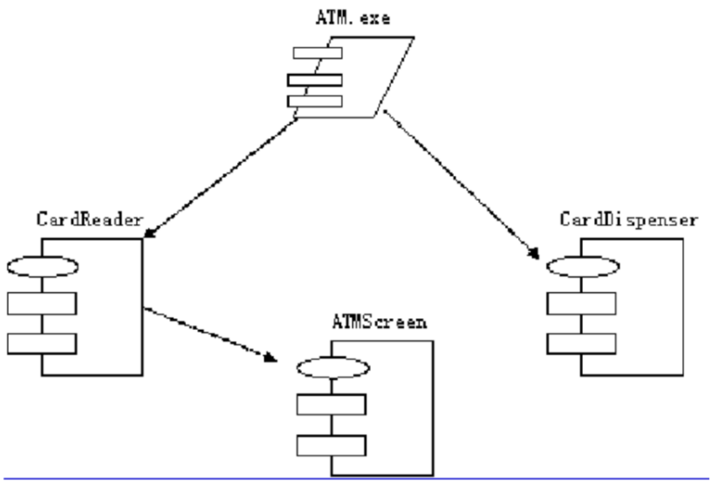


Activity Diagram 活动图：描述满足用例要求所要进行的活动以及活动间的约束关系，有利于识别并行活动；The great strength of activity diagrams lies in the fact that they **support and encourage parallel behavior**

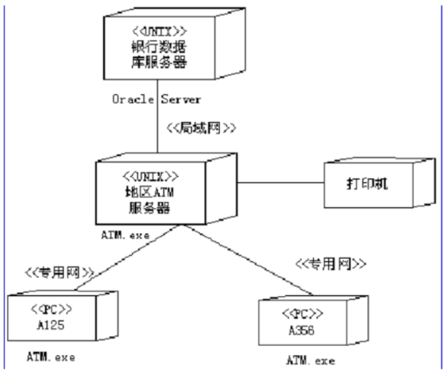


81

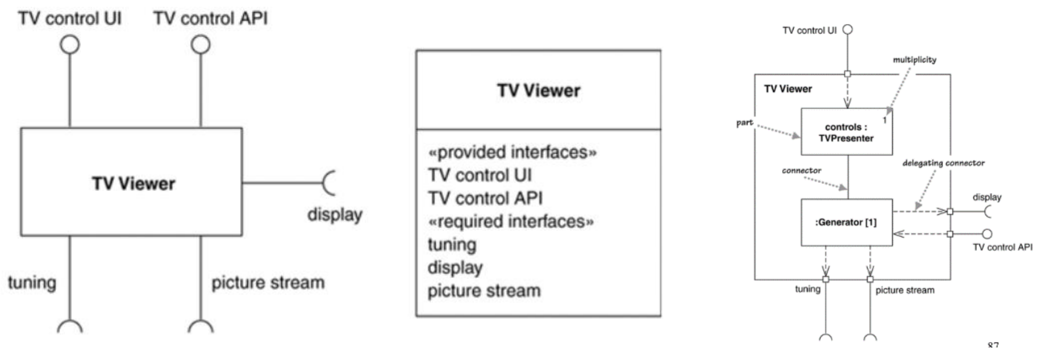
Component Diagram 组件图：描述代码构件的物理结构及各构件之间的依赖关系；Use component diagrams when you are **dividing your system into components** and want to **show their interrelationships through interface**s or the **breakdown of components into a lower-level structure**



Deployment Diagram 部署图：部署图定义系统中软硬件的物理体系结构

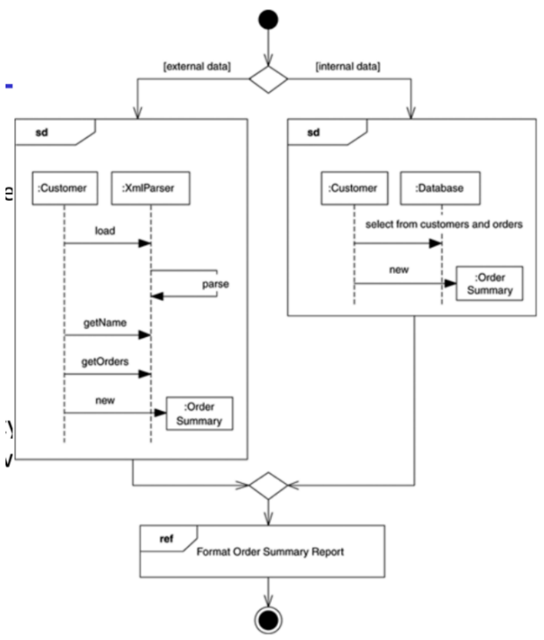


Composite Structures：复合结构：hierarchically decompose a class into an internal structure. This allows you to take a complex object and break it down into parts.

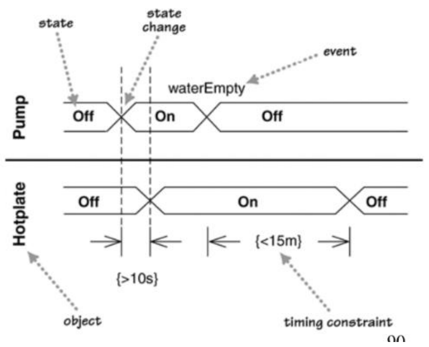


difference between packages and composite structures is that **packages are a compile-time grouping**, while **composite structures show runtime groupings**

Interaction Overview Diagrams 交互概览图：grafting together of activity diagrams and sequence diagrams

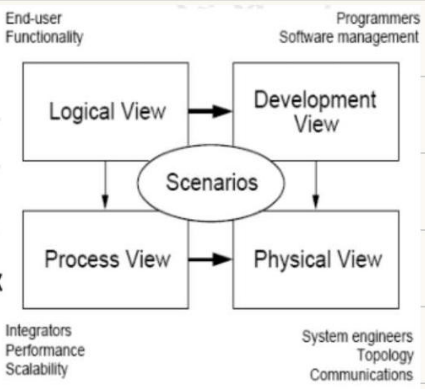


Timing Diagrams 时序图：Timing diagrams are another form of interaction diagram, where the focus is on timing constraints: either for a single object or, more usefully, for a bunch of objects；Timing diagrams are useful for showing **timing constraints between state changes on different objects**. The diagrams are particularly familiar to hardware engineers.



⑤ 4+1 模型

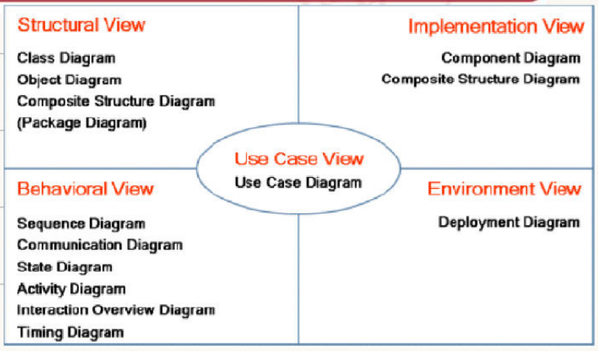
- 逻辑视图：**支持行为要求**。 关键抽象，是对象或对象类。
- 过程视图：**解决并发和分发**。 将线程映射到对象。
- 开发视图：**组织**软件模块，库，子系统，开发单元。
- 物理视图：将其他元素映射到**处理**和**通信节点**。
- 用例视图：将其他视图映射到重要的**用例**（这些用例被称作场景）上对体系结构加以说明，它们构成了第5个视图。



“4+1”视图模型从5个不同的视角包括逻辑视图、进程视图、物理视图、开发视图和场景视图来描述软件体系结构。每一个视图只关心系统的一个侧面，只有5个视图结合在一起才能反映系统的软件体系结构的全部内容。

2. UML

① UML 与 视图 对应.



- 用例视图 (场景)
  - 从外部世界的角度描述正在建模的系统的功能。
  - 需要使用此视图来描述系统应该执行的操作。 所有其他视图都依靠用例视图（场景）来指导它们，这就是将模型称为4 + 1的原因。
  - 该视图通常包含用例图，描述和概述图。

- 逻辑视图
  - 描述系统各部分的抽象描述。用于建模系统的组成部分以及各组成部分之间的交互方式。
  - 通常包括类图，对象图，状态图和协作图。

- 过程视图
  - 描述系统中的进程。 当 可视化系统中一定会发生的事情时，此视图特别有用。
  - 该视图通常包含活动图。

- 开发视图
  - 描述系统的各部分如何被组织为模块和组件。管理系统体系结构中的层非常有用。
  - 该视图通常包含包图和组件图。

- 物理视图
  - 描述如何将前三个视图中所述的系统设计实现为一组现实世界的实体。该视图中的图表展示了抽象部分如何映射到最终部署的系统中。
  - 该视图通常包含部署图。