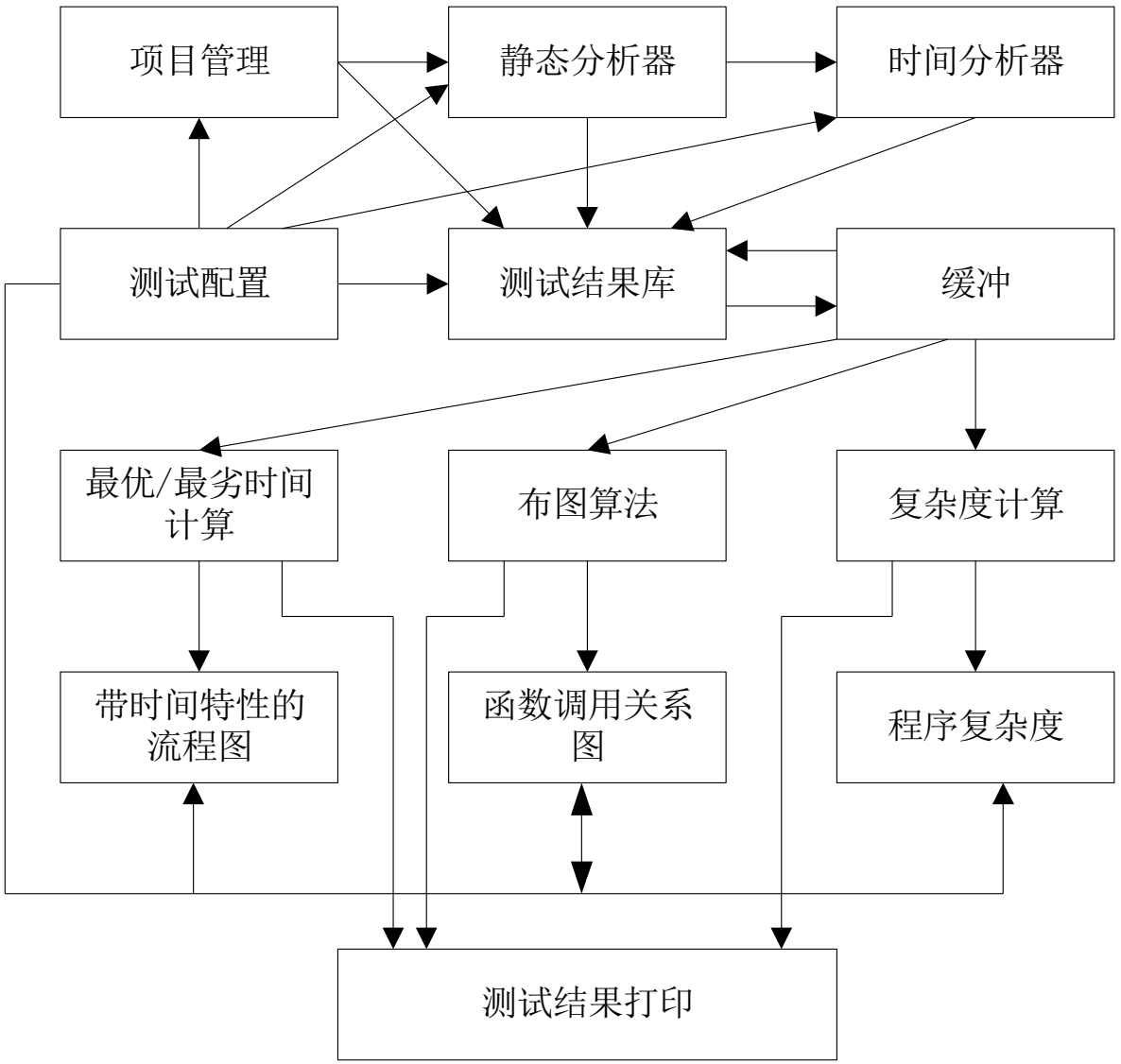
第4章 软件体系结构描述

描述方法的分类

- 图形表达工具
- 模块内连接语言
- 基于软构件的系统描述语言
- 软件体系结构描述语言

图形表达工具



某软件辅助理解和测试工具部分体系结构描述

模块内连接语言

- 采用将一种或几种传统程序设计语言的模块连接起来的模块内连接语言。由于程序设计语言和模块内连接语言具有严格的语义基础，因此它们能支持对较大的软件单元进行描述，诸如定义/使用和扇入/扇出等操作。例如，Ada语言采用use实现包的重用，Pascal语言采用过程（函数）模块的交互等。
- MIL方式对模块化的程序设计和分段编译等程序设计与开发技术确实发挥了很大的作用。但是由于这些语言处理和描述的软件设计开发层次过于依赖程序设计语言，因此限制了它们处理和描述比程序设计语言元素更为抽象的高层次软件体系结构元素的能力。

基于软构件的系统描述语言

- 基于软构件的系统描述语言将软件系统描述成一种是**由许多以特定形式相互作用的特殊软件实体构造组成的组织或系统。**
- 例如，一种多变配置语言就可以用来在一个较高的抽象层次上对系统的体系结构建模，Darwin最初用作**设计和构造复杂分布式系统的配置说明语言**，因具有动态特性，也可用来描述动态体系结构。
- 这种表达和描述方式虽然也是较好的一种以构件为单位的软件系统描述方法，但是他们所面向和针对的系统元素仍然是一些**层次较低的以程序设计为基础的通信协作软件实体单元**，而且这些语言所描述和表达的系统一般而言都是面向特定应用的特殊系统，这些特性使得基于软构件的系统描述仍然**不是十分适合软件体系结构的描述和表达。**

软件体系结构描述语言

- 软件体系结构的第四种描述和表达方法是参照传统程序设计语言的设计和开发经验，**重新设计、开发和使用**针对软件体系结构特点的**专门的软件体系结构描述语言。**
- 由于ADL是在吸收了传统程序设计中的**语义严格精确**的特点基础上，针对软件体系结构的**整体性和抽象性**特点，定义和确定适合于软件体系结构表达与描述的有关抽象元素，因此，ADL是当前软件开发和设计方法学中一种发展很快的软件体系结构描述方法，目前，已经有几十种常见的ADL。

软件体系结构描述框架标准 – IEEE P1471

- IEEE P1471于2000年9月21日通过IEEE-SA标准委员会评审。
- IEEE P1471适用于软件密集的系统，其目标在于：便于体系结构的表达与交流，并通过体系结构要素及其实践标准化，奠定质量与成本的基础。
- IEEE P1471详细介绍了一套体系结构描述的概念框架，并给出建立框架的思路。但如何描述以及具体的描述技术等方面缺乏更进一步的指导。

软件体系结构描述框架标准 – Rational

- Rational起草了可重用的软件资产规格说明，专门讨论了体系结构描述的规格说明，提出了一套易于重用的体系结构描述规范。
- 基于RUP（Rational United Process）、采用UML模型描述软件的体系结构，认为体系结构描述的关键是定义视点、视图以及建模元素之间的映射关系。
- 与IEEE P1471相比，该建议标准的体系结构描述方案涉及面比较窄，所注重的层次比较低，因而更具体。由于将体系结构的描述限于UML和RUP，具有一定的局限性，但该建议标准结合了业界已经广泛采用的建模语言和开发过程，因而易于推广，可以有效实现在跨组织之间重用体系结构描述结果。

ADL与其他语言的比较

- **构造能力**：ADL能够使用较小的独立体系结构元素来建造大型软件系统；
- **抽象能力**：ADL使得软件体系结构中的构件和连接件描述可以只关注它们的抽象特性，而不管其具体的实现细节；
- **重用能力**：ADL使得组成软件系统的构件、连接件甚至是软件体系结构都成为软件系统开发和设计的可重用部件；
- **组合能力**：ADL使得其描述的每一系统元素都有其自己的局部结构，这种描述局部结构的特点使得ADL支持软件系统的动态变化组合；
- **异构能力**：ADL允许多个不同的体系结构描述关联存在；
- **分析和推理能力**：ADL允许对其描述的体系结构进行多种不同的性能和功能上的多种推理分析。

ADL与其他语言的比较

典型元素含义比较

程序设计语言		软件体系结构	
程序构件	组成程序的基本元素及其取值或值域范围	系统构件	模块化级别的系统组成成分实体，这些实体可以被施以抽象的特性化处理，并以多种方式得到使用
操作符	连接构件的各种功能符号	连接件	对组成系统的有关抽象实体进行各种连接的连接机制
抽象规则	有关构件和操作符的命名表达规则	组合模式	系统中的构件和连接件进行连接组合的特殊方式，也就是软件体系结构的风格
限制规则	一组选择并决定具体使用何种抽象规则来作用于有关的基本构件及其操作符的规则和原理	限制规则	决定有关模式能够作为子系统进行大型软件系统构造和开发的合法子系统的有关条件
规范说明	有关句法的语义关联说明	规范说明	有关系统组织结构方面的语义关联说明

ADL与其他语言的比较

常见的软件体系结构元素

系统构件元素		连接件元素	
纯计算单元	这类构件只有简单的输入/输出处理关联，对它们的处理一般也不保留处理状态，如数学函数、过滤器和转换器等	过程调用	在构件实体之间实现单线程控制的连接机制，如普通过程调用和远程过程调用等
数据存储单元	具有永久存储特性的结构化数据，如数据库、文件系统、符号表和超文本等	数据流	系统中通过数据流进行交互的独立处理流程连接机制，其最显著的特点是根据得到的数据来进行构件实体的交互控制，如 Unix 操作系统中的管道机制等
管理器	对系统中的有关状态和紧密相关操作进行规定与限制的实体，如抽象数据类型和系统服务器等	隐含触发器	由并发出现的事件来实现构件实体之间交互的连接机制，在这种连接机制中，构件实体之间不存在明显确定的交互规定，如时间调度协议和自动垃圾回收处理等
控制器	控制和管理系统中有关事件发生的时间序列，如调度程序和同步处理协调程序等	消息传递	独立构件实体之间通过离散和非在线的数据（可以是同步或非同步的）进行交互的连接机制，如 TCP/IP 等
连接器	充当有关实体间信息转换角色的实体，如通讯连接器和用户界面等	数据共享协议	构件之间通过相同的数据空间进行并发协调操作的机制，如黑板系统中的黑板和多用户数据库系统中的共享数据区等

经典软件体系结构描述语言

- UniCon
- Wright
- C2
- Rapide
- SADL
- Aesop
- ACME

C2 – 概述

- C2和其提供的设计环境（Argo）支持采用基于时间的风格来描述用户界面系统，并支持使用可替换、可重用的构件开发GUI的体系结构。
- 在C2中，连接件负责构件之间消息的传递，而构件维持状态、执行操作并通过两个名字分别为“*top*”和“*bottom*”的端口和其它的构件交换信息。
- 每个接口包含一种可发送的消息和一组可接收的消息。构件之间的消息要么是请求其它构件执行某个操作的请求消息，要么是通知其他构件自身执行了某个操作或状态发生改变的通知消息。
- 构件之间的消息交换不能直接进行，而只能通过连接件来完成。每个构件接口最多只能和一个连接件相连，而连接件可以和任意数目的构件或连接件相连。
- 请求消息只能向上层传送而通知消息只能向下层传送。
- 通知消息的传递只对应于构件内部的操作，而和接收消息的构件的需求无关。
- C2对构件和连接件的实现语言、实现构件的线程控制、构件的部署以及连接件使用的通讯协议等都不加限制。

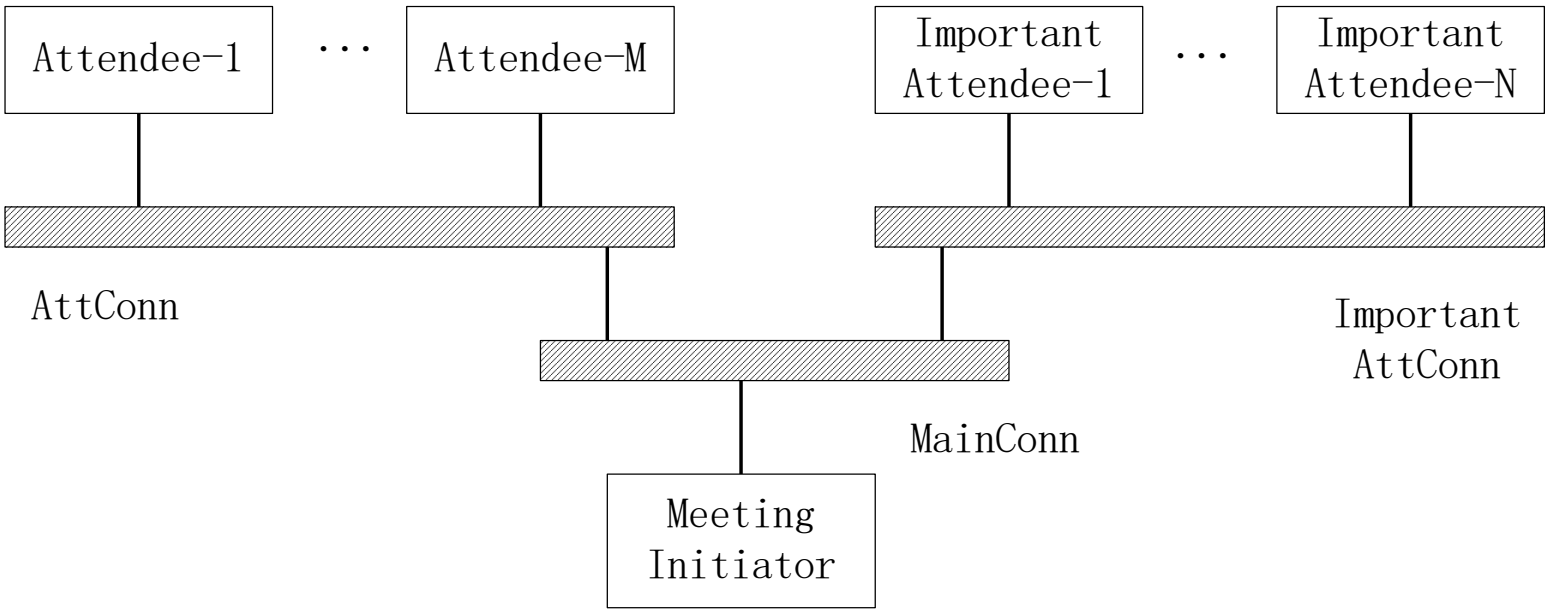
C2 – 对构件的描述

```
Component ::=
    component component_name is
        interface component_message_interface
        parameters component_parameters
        methods component_methods
        [behavior component_behavior]
        [context component_context]
     end component_name;
```

C2 – 对构件接口的描述

```
component_message_interface ::=
    top_domain_interface
    bottom_domain_interface

top_domain_interface ::=
    top_domain is
        out interface_requests
        in interface_notifications

bottom_domain_interface ::=
    bottom_domain is
        out interface_notifications
        in interface_requests
interface_requests ::=
    {request;} | null;

interface_notifications ::=
    {notification;} | null;

request ::=
    message_name(request_parameters)

request_parameters ::=
    [to component_name][parameter_list]

notification ::=
    message_name[parameter_list]
```

C2 – 会议安排系统


Attendee-1
...
Attendee-M
Important Attendee-1
...
Important Attendee-N
AttConn
Important AttConn
MainConn
Meeting Initiator

C2 – 对MeetgingInitiator构件的描述

```
component MeetingInitiator is
   interface
      top_domain is
        out
            GetPrefSet();
            GetExclSet();
            GetEquipReqts();
            GetLocPrefs();
            RemoveExclSet();
            RequestWithdrawal(to Attendee);
            RequestWithdrawal(to ImportantAttendee);
            AddPrefDates();
            MarkMtg(d:date; l:lov_type);
        in
            PrefSet(p:date_mg);
            ExclSet(e:data_mg);
            EquipReqts(eq:equip_type);
            LocPref(l:loc_type);
     behavior
        startup always_generate GetPrefSet, GetExclSet, GetEquipReqts, GetLocPrefs;
        received_messages PrefSet may_generate RemoveExclSet xor RequestWithdrawal xor MarkMtg;
        received_messages ExclSet may_generate AddPrefDates xor RemoveExclSet xor RequestWithdrawal xor MarkMtg;
        received_messages EquipReqts may_generate AddPrefDates xor RemoveExclSet xor RequestWithdrawal xor MarkMtg;
        received_messages LocPref always_generate null;
end MeetingInitiator;
```

C2 – 对Attendee构件的描述

```
component Attendee is
   interface
      bottom_domain is
        out
            PrefSet(p:date_mg);
            ExclSet(e:date_mg);
            EquipReqts(eq:equip_type);
        in
            GetPrefSet();
            GetExclSet();
            GetEquipReqts();
            RemoveExclSet();
            RequestWithdrawal();
            AddPrefDates();
            MarkMtg(d:date;  l:loc_type);
    behavior
          received_messages GetPrefSet always_generate PrefSet;
          received_messages AddPrefDates always_generate PrefSet;
          received_messages GetExclSet always_generate ExclSet;
          received_messages GetEqipReqts always_generate EqipReqts;
          received_messages RemoveExclSet always_generate ExclSet;
          received_messages ReuestWithdrawal always_generate null;
          received_messages MarkMtg always_generate null;
end Attendee;
```

C2 – 对ImportantAttendee构件的描述

```
component ImportantAttendee is subtype Attendee (in and beh)
    interface
        bottom_domain is
          out
              LocPrefs(l:loc_type);
              ExclSet(e:date_mg);
              EquipReqts(eq:equip_type);
          in
              GetLocPrefs();
     behavior
           received_messages GetLocPrefs always_generate LocPrefs;
 end ImportantAttendee;
```

C2 – 对体系结构的描述

```
architecture MeetingScheduler is
    conceptual_components
        Attendee；ImportantAttendee；MeetingInitiator；
    connectors
        connector MainConn is message_filter no_filtering；
        connector AttConn is message_filter no_filtering；
        connector ImportantAttConn is message_filter no_filtering；
    architectural_topology
        connector AttConn connections
            top_ports Attendee；
            bottom_ports MainConn；
        connector ImportantAttConn connections
            top_ports ImportantAttendee；
            bottom_ports MainConn；
        connector MainConn connections
            top_ports AttConn；ImportantAttConn；
            bottom_ports MeetingInitiator；
end MeetingScheduler；
```

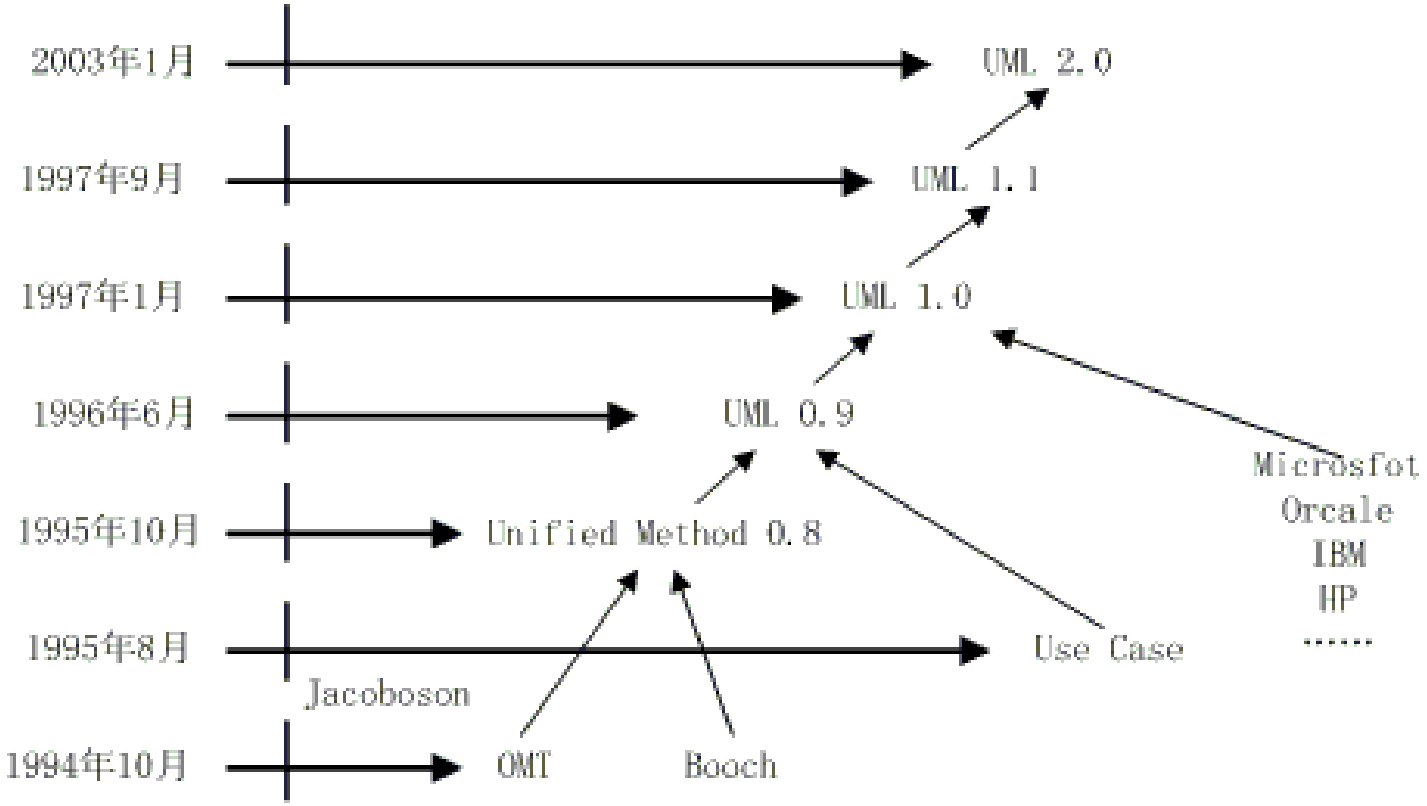
C2 – 对会议安排系统的描述

```
system MeetingScheduler_1 is
  architecture MeetingScheduler with
     Attendee instance Att_1，Att_2，Att_3；
     ImportantAttendee instance ImpAtt_1，ImpAtt_2；
     MeetingInitiator instance MtgInit_1；
end MeetingScheduler_1；
```

UML概述

- UML是一种语言
- UML是一种可视化语言
- UML是一种可用于详细描述的语言
- UML是一种构造语言
- UML是一种文档化语言

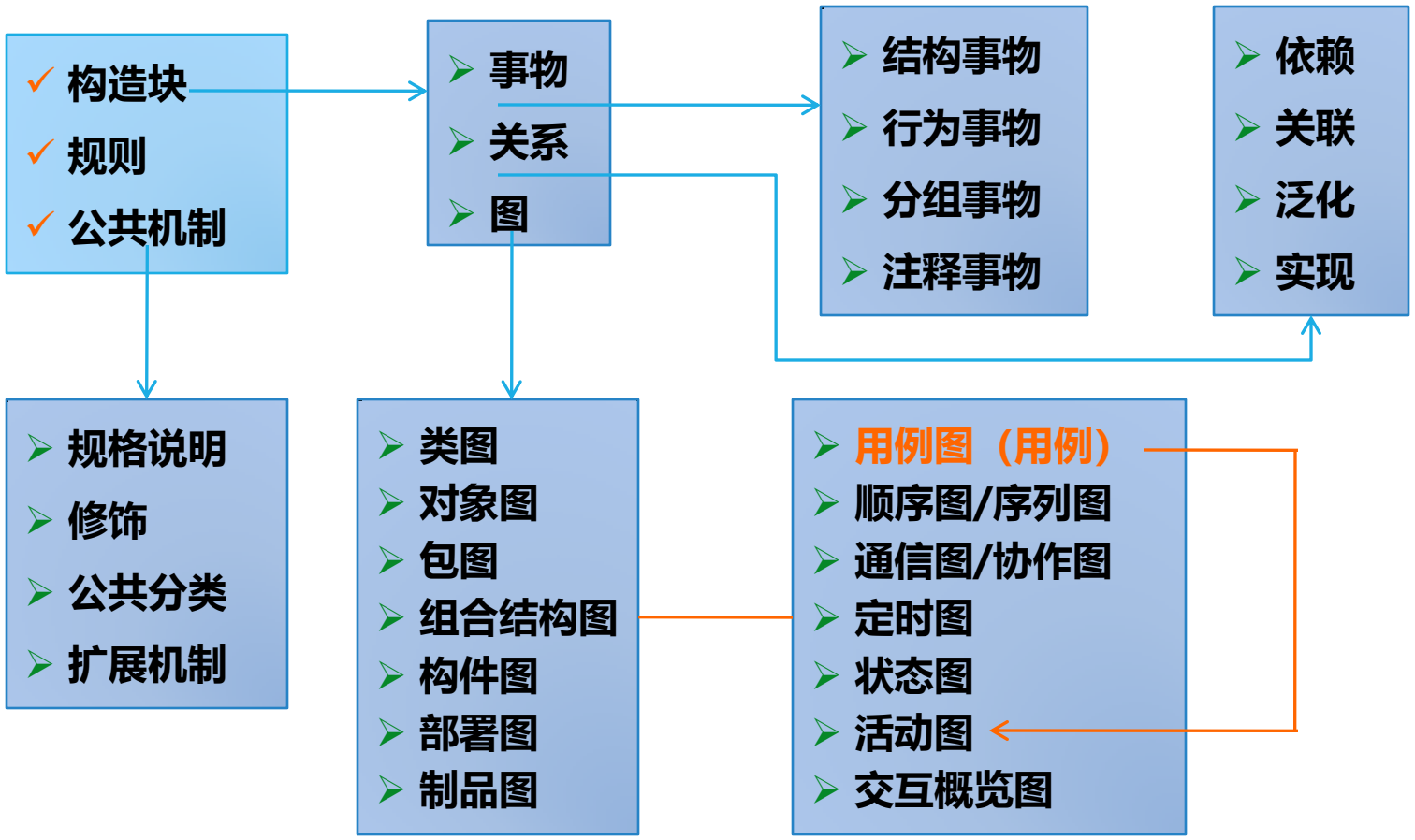
UML的发展历史


2003年1月
UML 2.0
1997年9月
UML 1.1
1997年1月
UML 1.0
1996年6月
UML 0.9
1995年10月
Unified Method 0.8
1995年8月
Use Case
Jacoboson
1994年10月
OMT
Booch
Microsfot
Orcale
IBM
HP
……

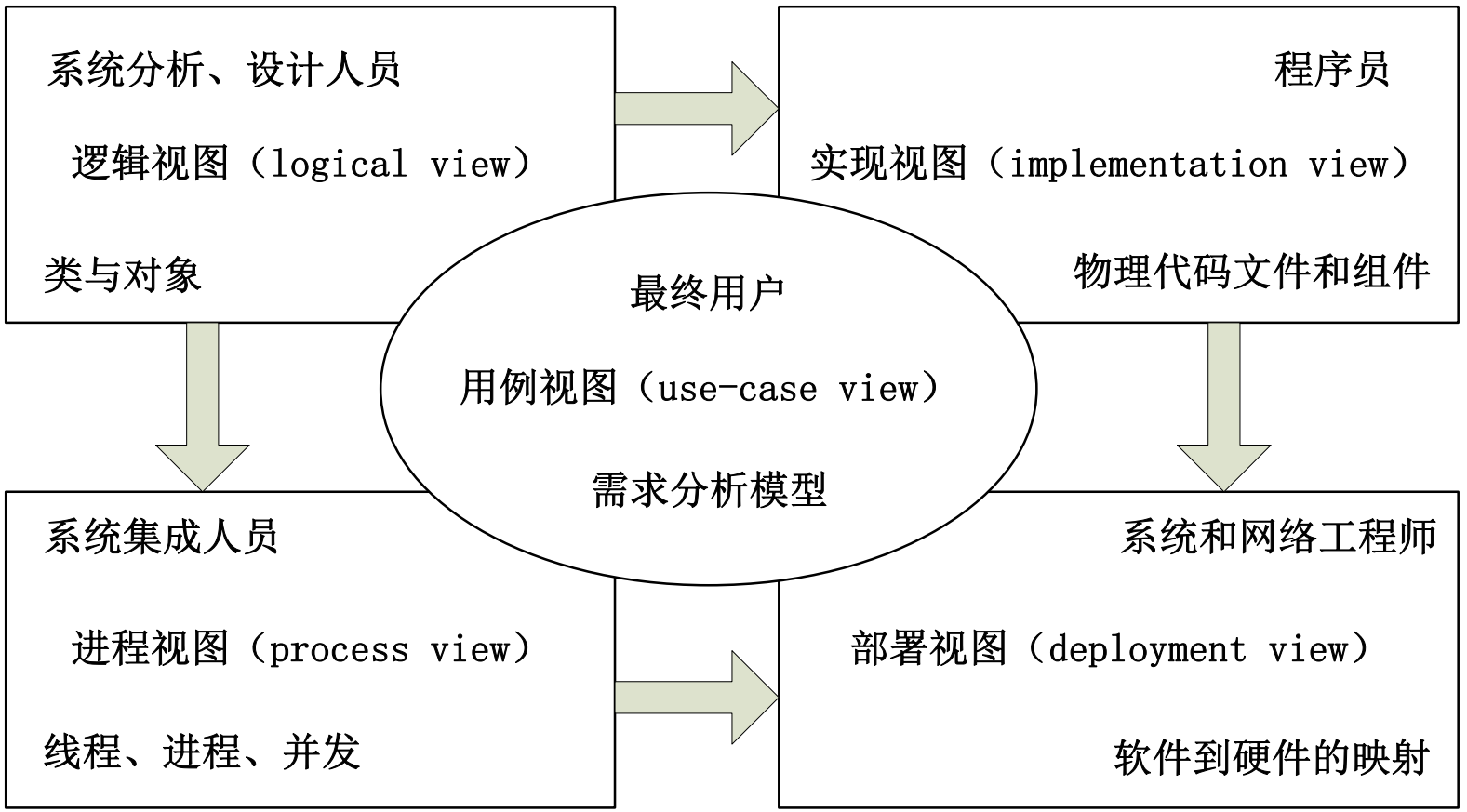
UML的应用领域

- UML是一种**建模语言**而不是一种方法，其中并不包括过程的概念，其本身是独立于过程的，可以在任何过程中使用它。
- UML能够用**面向对象的方法**描述任何类型的系统，并对系统开发从需求调研到测试和维护的各个阶段进行有效的支持。

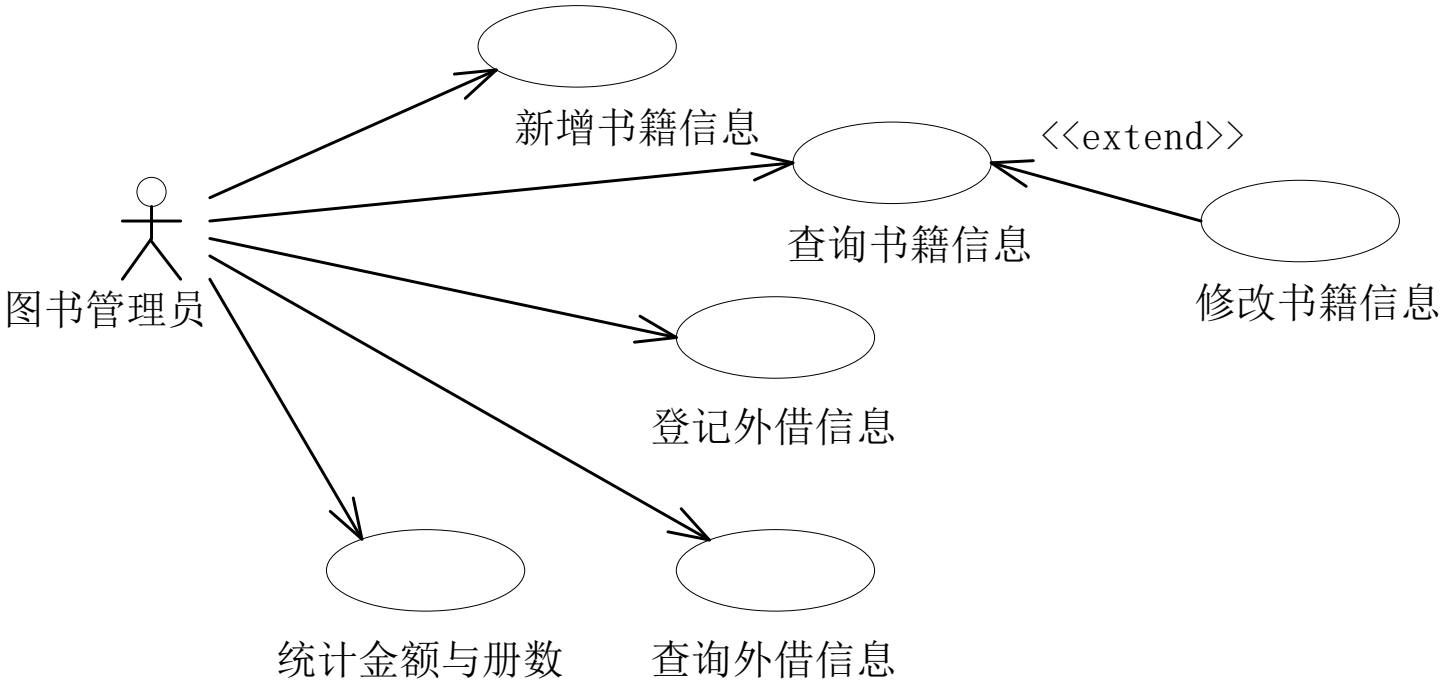
UML的结构


构造块
规则
公共机制
事物
关系
图
结构事物
行为事物
分组事物
注释事物
依赖
关联
泛化
实现
规格说明
修饰
公共分类
扩展机制
类图
对象图
包图
组合结构图
构件图
部署图
制品图
用例图（用例）
顺序图/序列图
通信图/协作图
定时图
状态图
活动图
交互概览图

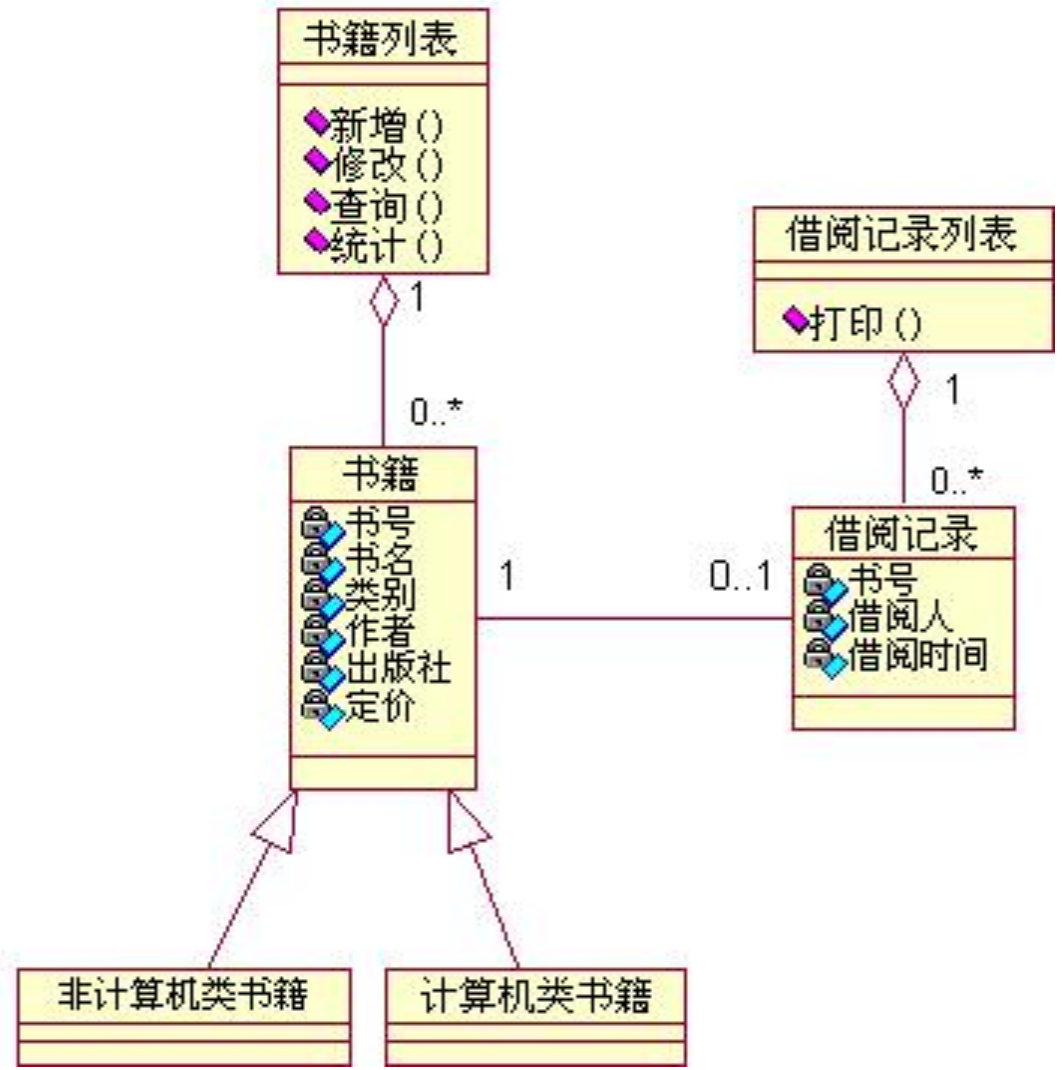
UML的结构


系统分析、设计人员
逻辑视图（logical view）
类与对象
程序员
实现视图（implementation view）
物理代码文件和组件
最终用户
用例视图（use-case view）
需求分析模型
系统集成人员
进程视图（process view）
线程、进程、并发
系统和网络工程师
部署视图（deployment view）
软件到硬件的映射

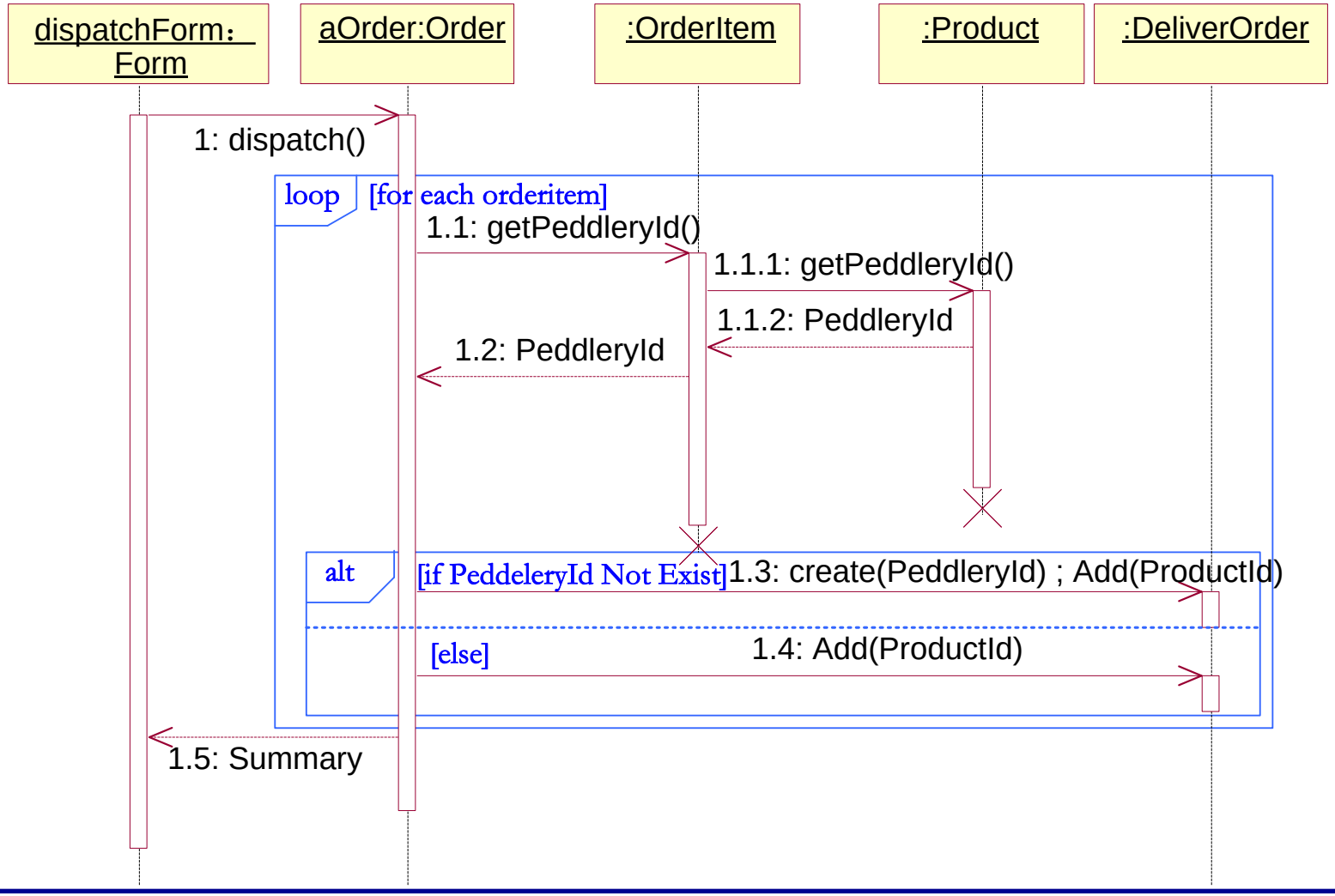
用例图



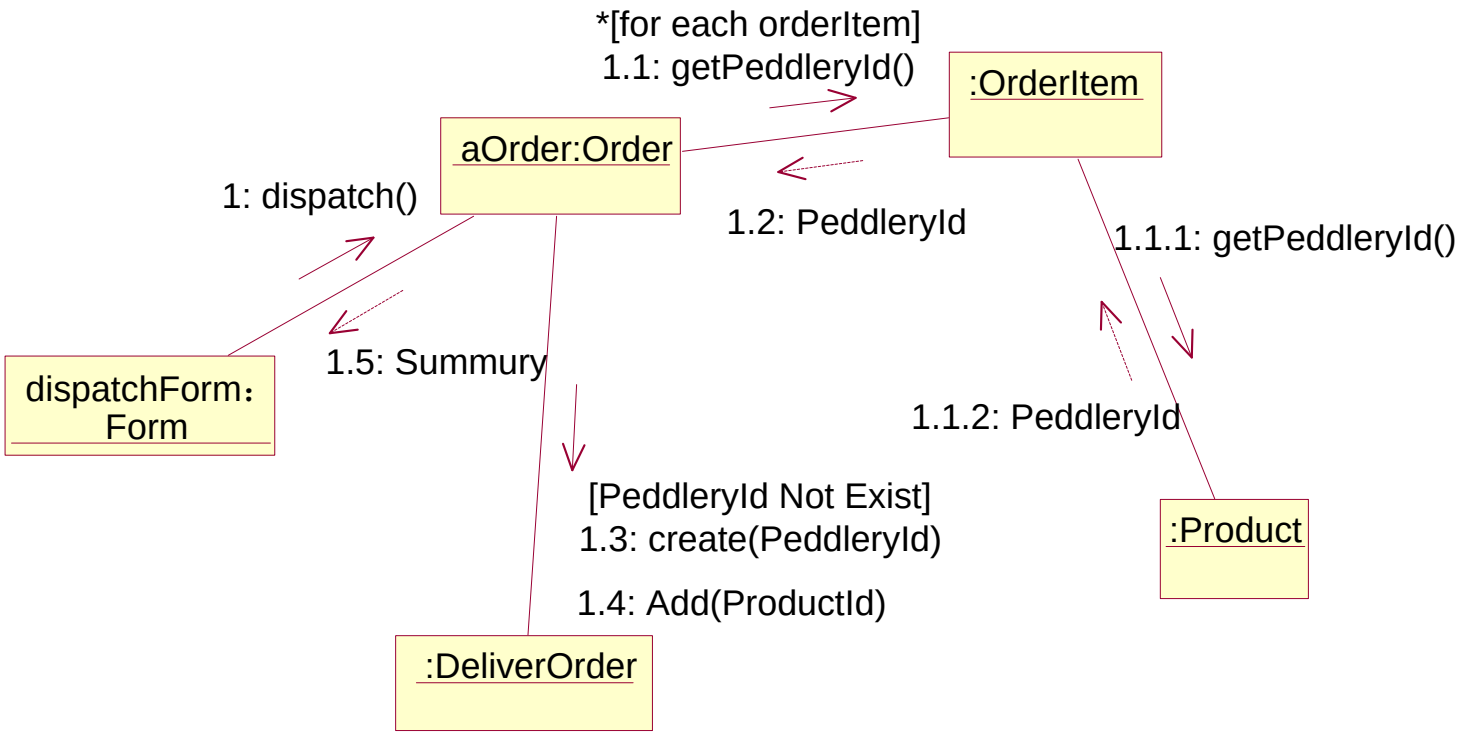
类图


书籍列表
新增()
修改()
查询()
统计()
1
0..*
借阅记录列表
打印()
1
0..*
书籍
书号
书名
类别
作者
出版社
定价
1
0..1
借阅记录
书号
借阅人
借阅时间
非计算机类书籍
计算机类书籍

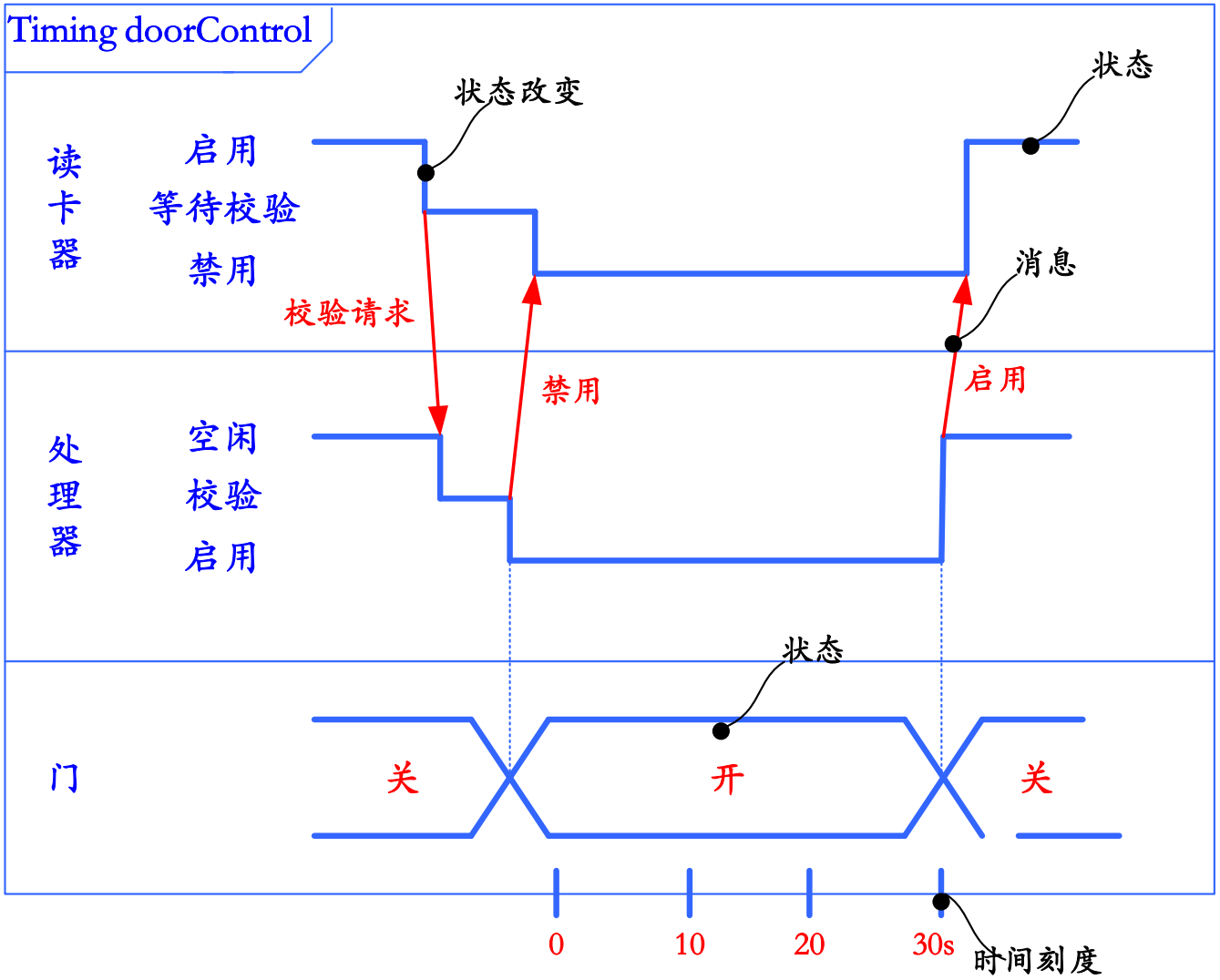
顺序图


dispatchForm：Form
aOrder:Order
:OrderItem
:Product
:DeliverOrder
1: dispatch()
loop
[for each orderitem]
1.1: getPeddleryId()
1.1.1: getPeddleryId()
1.1.2: PeddleryId
1.2: PeddleryId
alt
[if PeddeleryId Not Exist]
1.3: create(PeddleryId) ; Add(ProductId)
[else]
1.4: Add(ProductId)
1.5: Summary

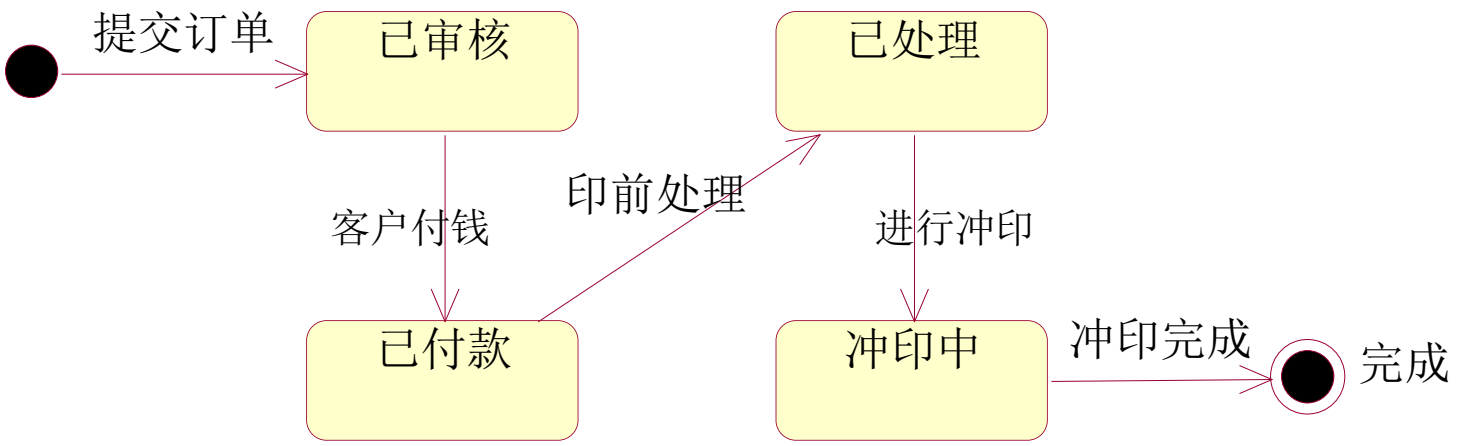
通信图


*[for each orderItem]
1.1: getPeddleryId()
:OrderItem
aOrder:Order
1: dispatch()
1.2: PeddleryId
1.1.1: getPeddleryId()
dispatchForm：Form
1.5: Summury
1.1.2: PeddleryId
[PeddleryId Not Exist]
1.3: create(PeddleryId)
1.4: Add(ProductId)
:Product
:DeliverOrder

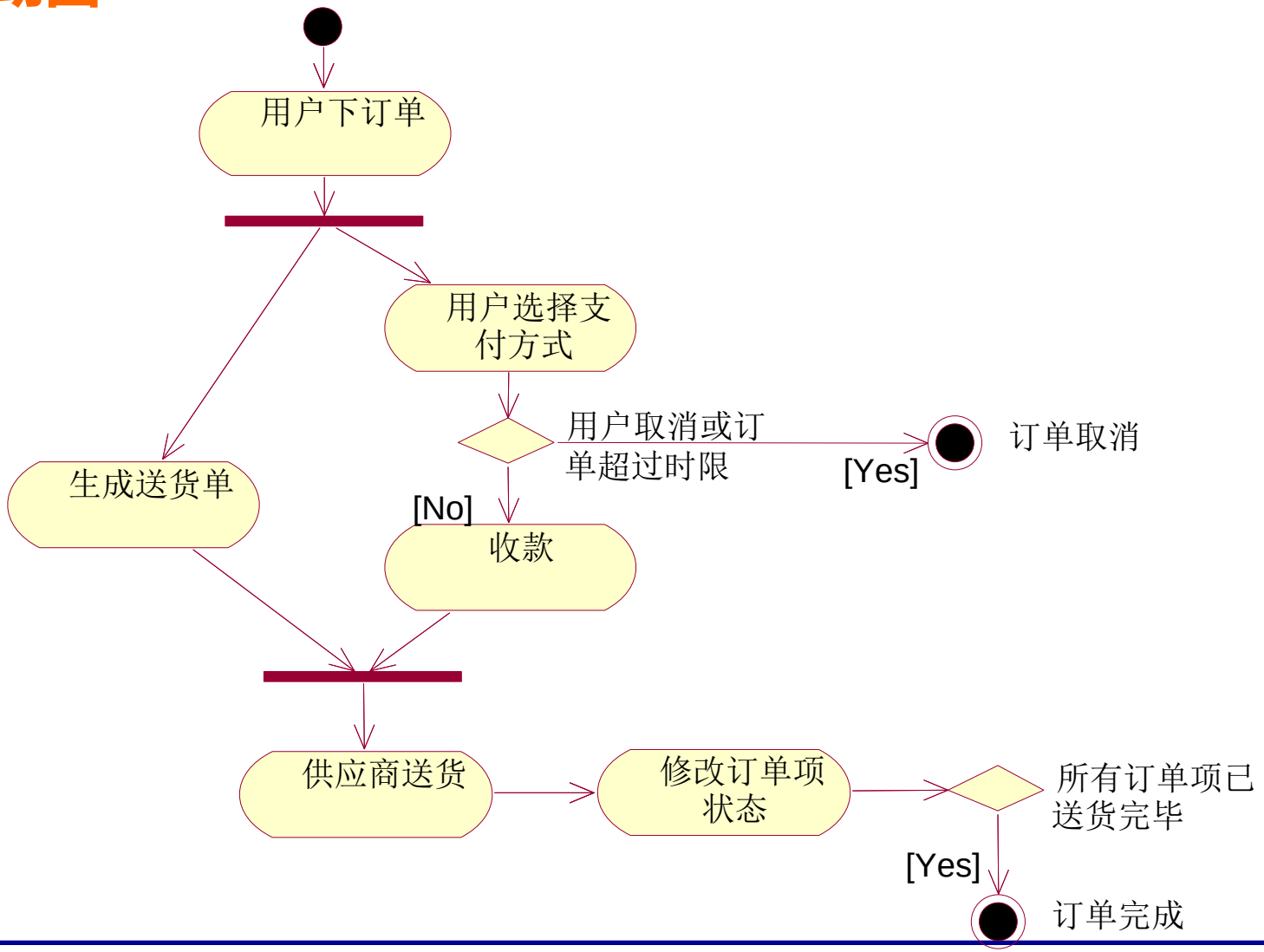
定时图


Timing doorControl
读卡器
启用
等待校验
禁用
状态改变
状态
消息
校验请求
禁用
启用
处理器
空闲
校验
启用
门
状态
关
开
关
0
10
20
30s
时间刻度

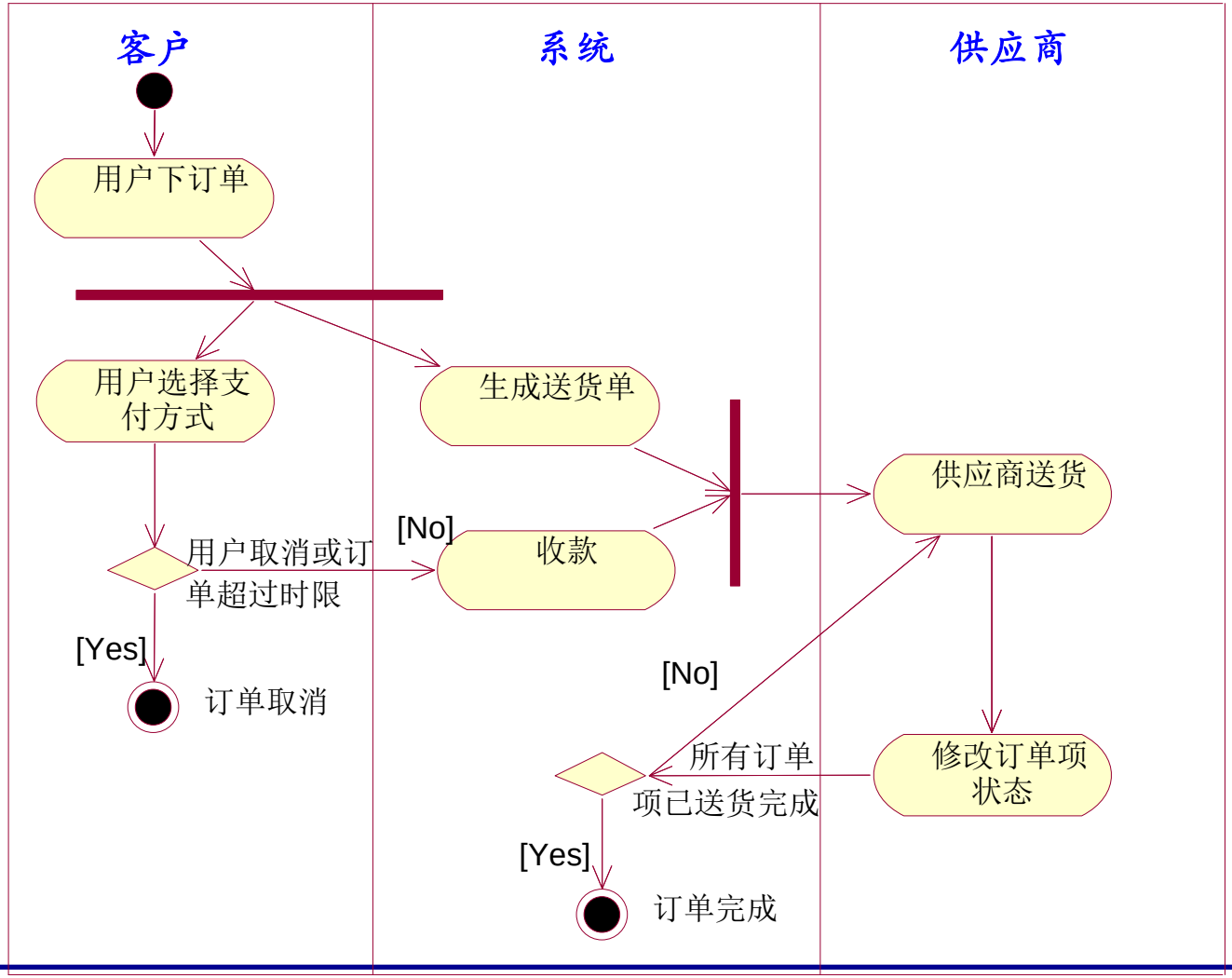
状态图



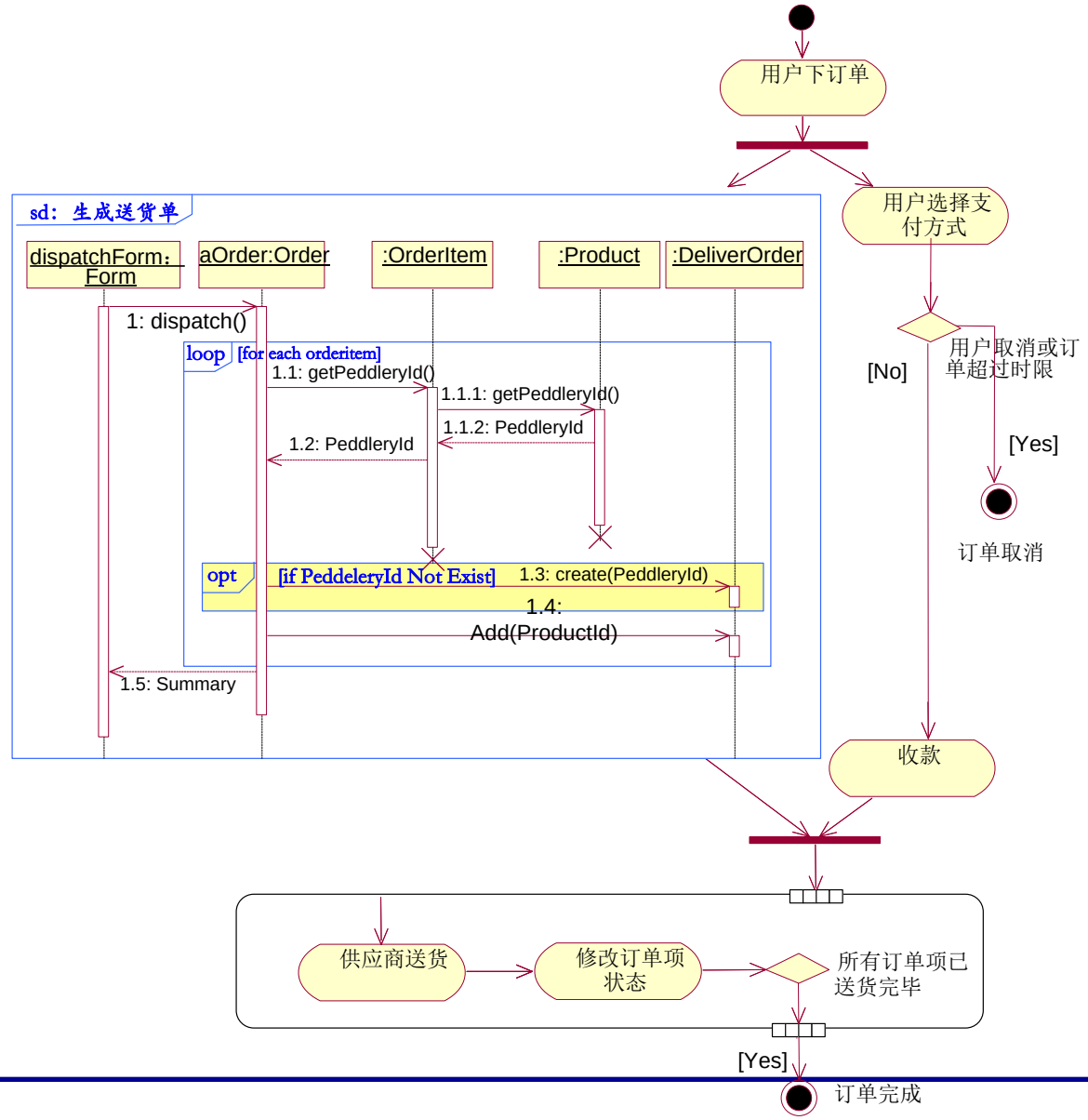
基本活动图


用户下订单
用户选择支付方式
用户取消或订单超过时限
[Yes]
订单取消
[No]
生成送货单
收款
供应商送货
修改订单项状态
所有订单项已送货完毕
[Yes]
订单完成

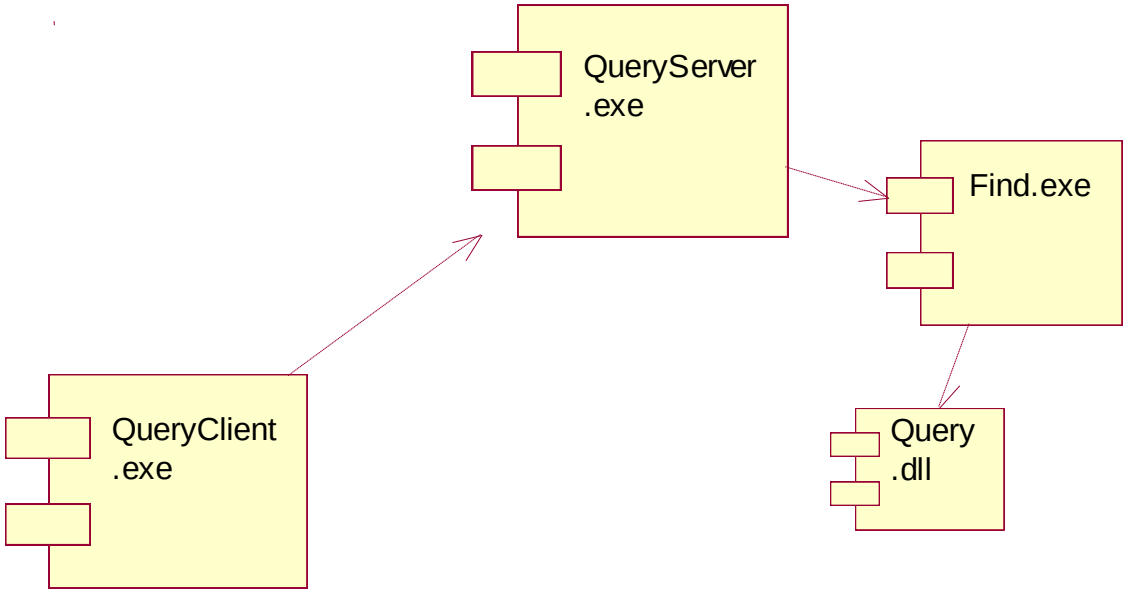
带泳道的活动图


客户
系统
供应商
用户下订单
用户选择支付方式
生成送货单
供应商送货
[No]
收款
用户取消或订单超过时限
[Yes]
订单取消
[No]
所有订单项已送货完成
修改订单项状态
[Yes]
订单完成

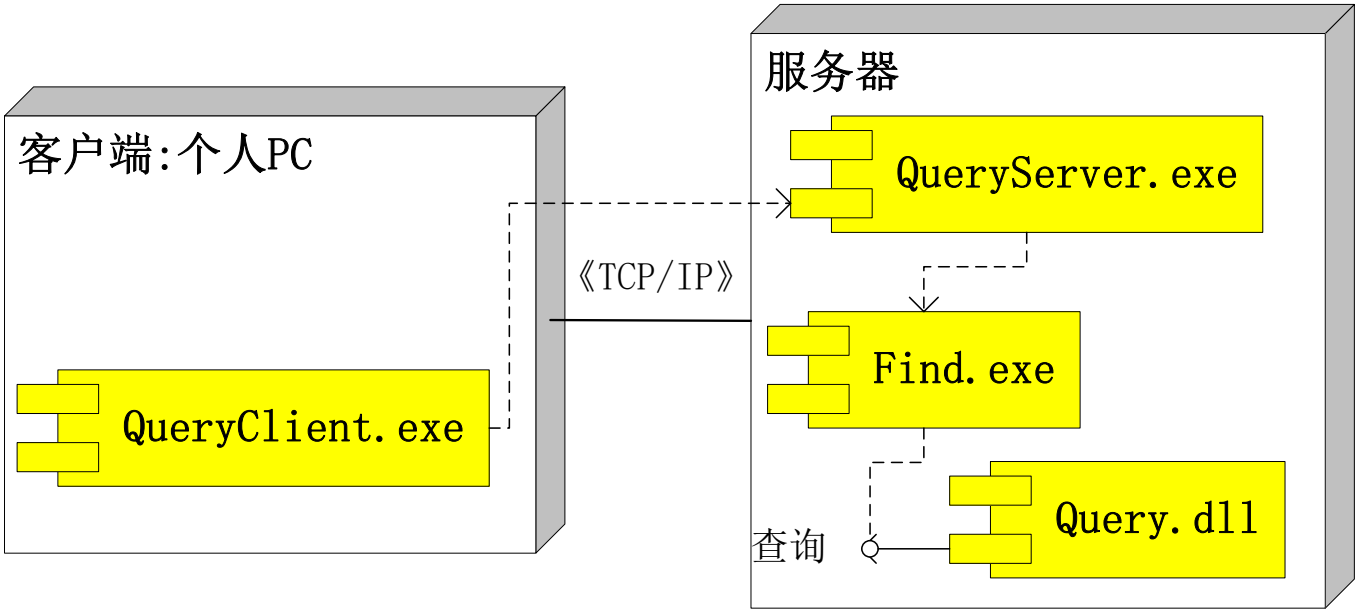
交互概览图


用户下订单
用户选择支付方式
sd: 生成送货单
dispatchForm: Form
aOrder:Order
:OrderItem
:Product
:DeliverOrder
1: dispatch()
loop [for each orderitem]
1.1: getPeddleryId()
1.1.1: getPeddleryId()
1.1.2: PeddleryId
1.2: PeddleryId
opt [if PeddeleryId Not Exist]
1.3: create(PeddleryId)
1.4: Add(ProductId)
1.5: Summary
[No]
用户取消或订单超过时限
[Yes]
订单取消
收款
供应商送货
修改订单项状态
所有订单项已送货完毕
[Yes]
订单完成

构件图



部署图


客户端:个人PC
QueryClient.exe
《TCP/IP》
服务器
QueryServer.exe
Find.exe
Query.dll
查询

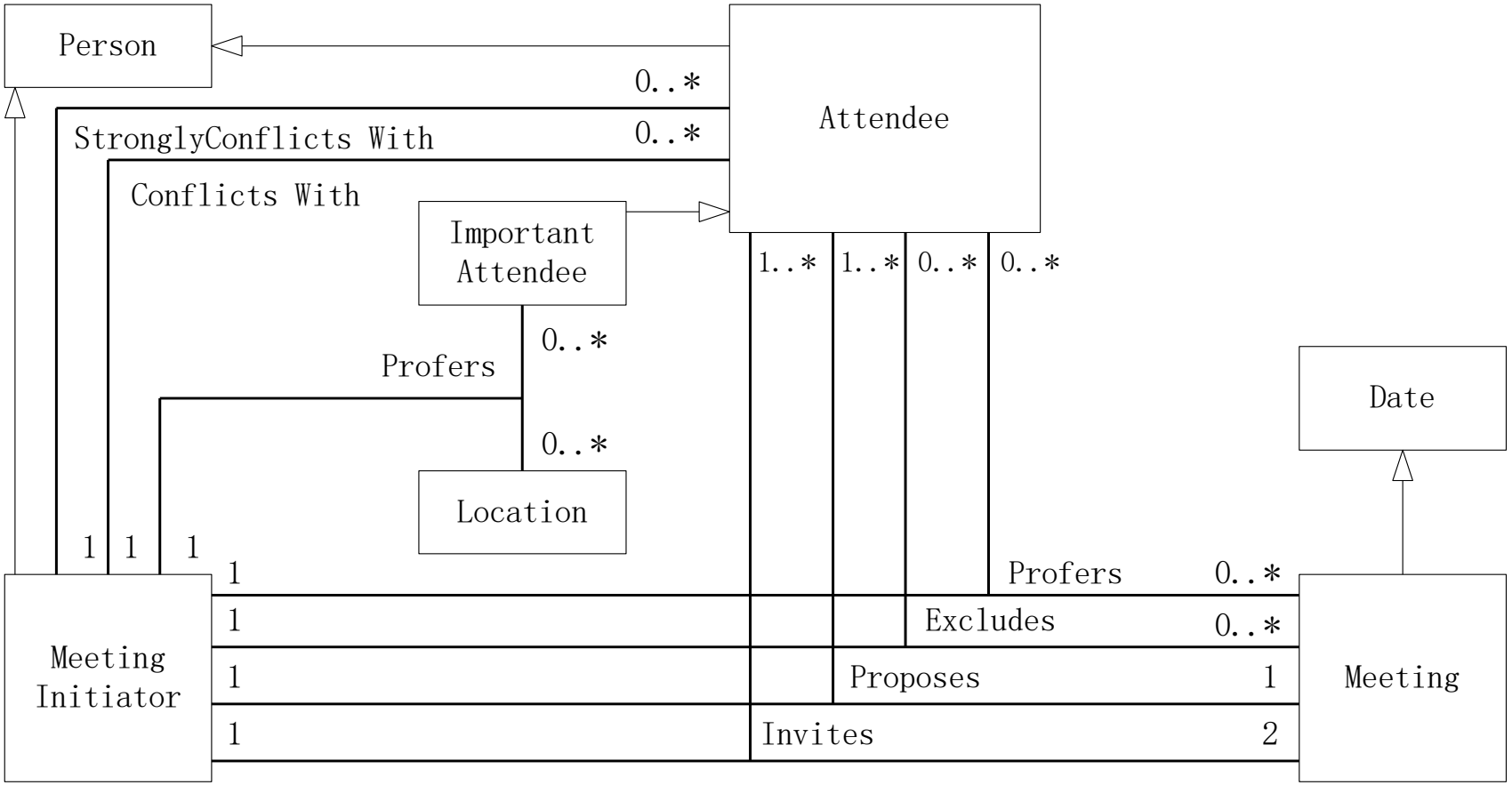
直接使用UML建模 – UML中的通用表示

- 字符串：表示有关模型的信息；
- 名字：表示模型元素；
- 标号：不同于编程语言中的标号，是用于表示或说明图形符号的字符串；
- 特殊字符串：表示某一模型元素的特性；
- 类型表达式：声明属性、变量及参数，含义同编程语言中的类型表达式；
- 实体类型：它是UML的扩充机制，运用实体类型可定义新类型的模型元素。

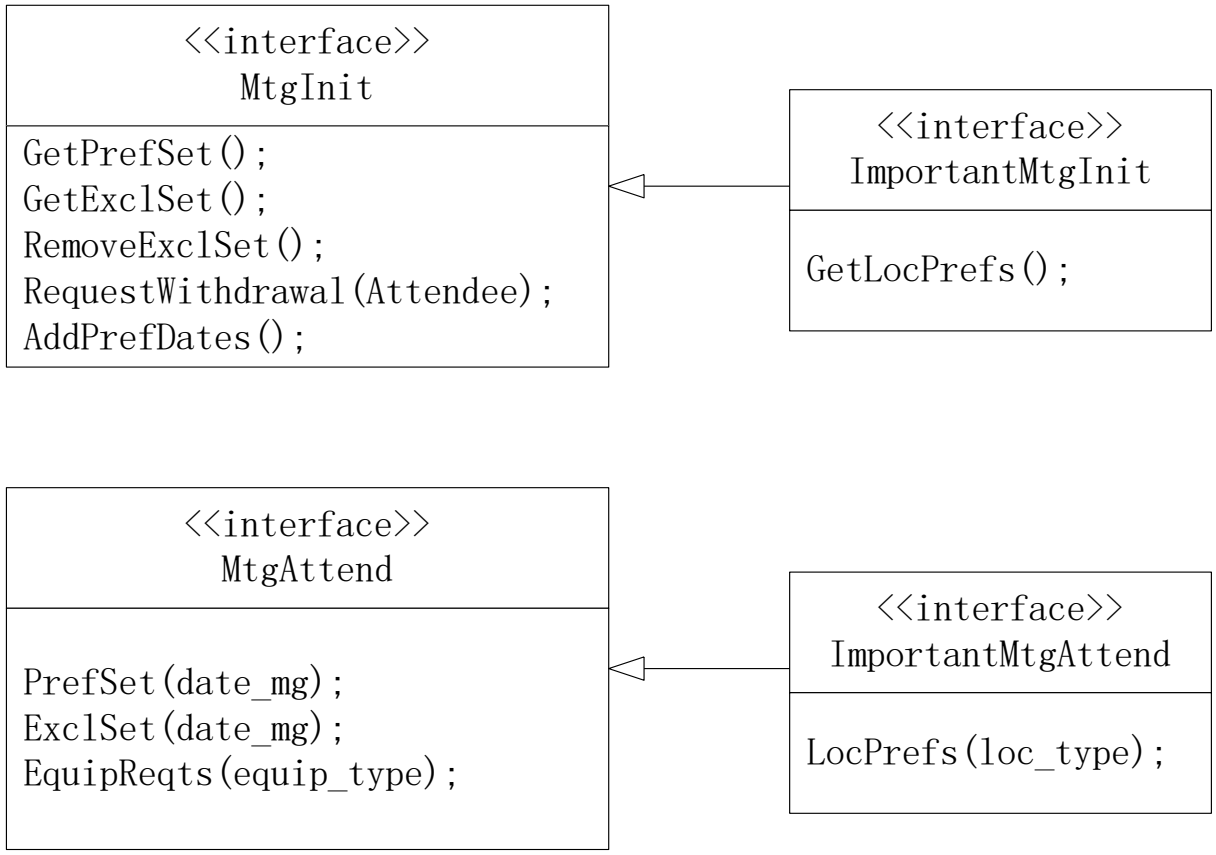
直接使用UML建模 – UML语义部分

- 通用元素：主要描述UML中各元素的语义。通用元素是UML中的基本构造单位，包括模型元素和视图元素，模型元素用来构造系统，视图元素用来构成系统的表示成分；
- 通用机制：主要描述使UML保持简单和概念上一致的机制的语义。包括定制、标记值、注记、约束、依赖关系、类型-实例、类型-类的对应关系等机制；
- 通用类型：主要描述UML中各种类型的语义。这些类型包括布尔类型、表达式类型、列表类型、多重性类型、名字类型、坐标类型、字符串类型、时间类型、用户自定义类型等。

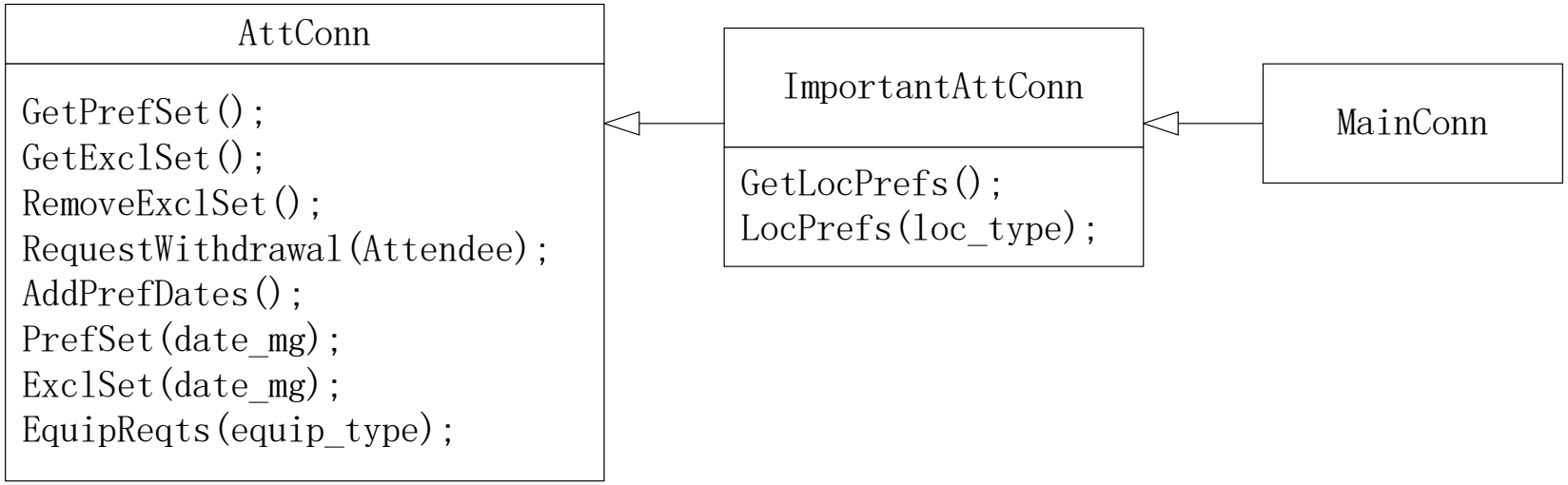
直接使用UML建模 – 会议安排系统的类图



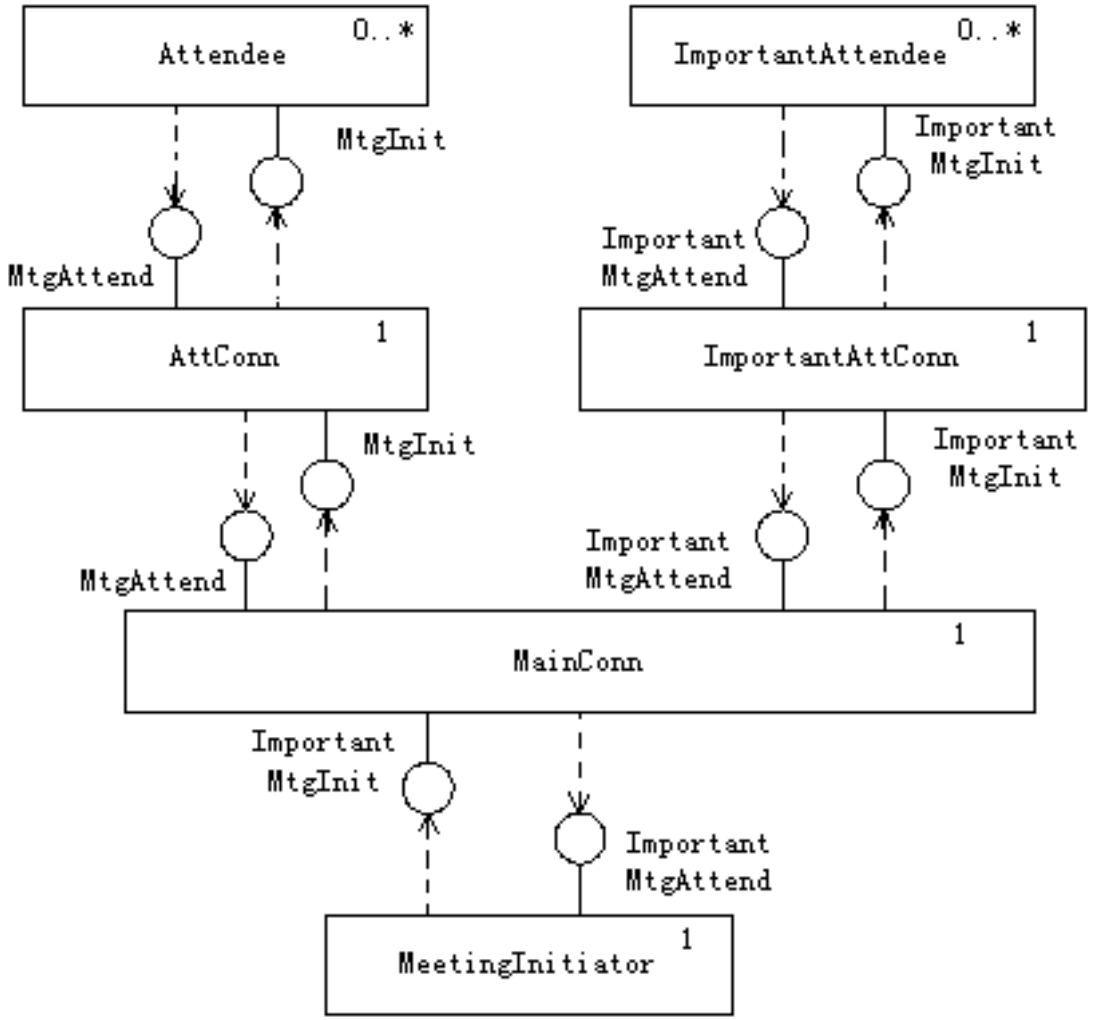
直接使用UML建模 – 会议安排系统的类接口



直接使用UML建模 – C2连接件模型



直接使用UML建模 – 细化的类图



直接使用UML建模 – 会议安排系统的通信图/协作图