第10章

系统顺序图

就理论而言，理论和实践并无差异。但真付诸实行，差异即开始显现。

——Jan L.A. van de Snepscheut

目标

- 确定系统事件。
- 为用例场景创建系统顺序图。

简介

系统顺序图（SSD）是为阐述与所讨论系统相关的输入和输出事件而快速、简单地创建的制品。它们是操作契约和（最重要的）对象设计的输入。

UML包含以顺序图为形式的表示法，用以阐述外部参与者到系统的事件。

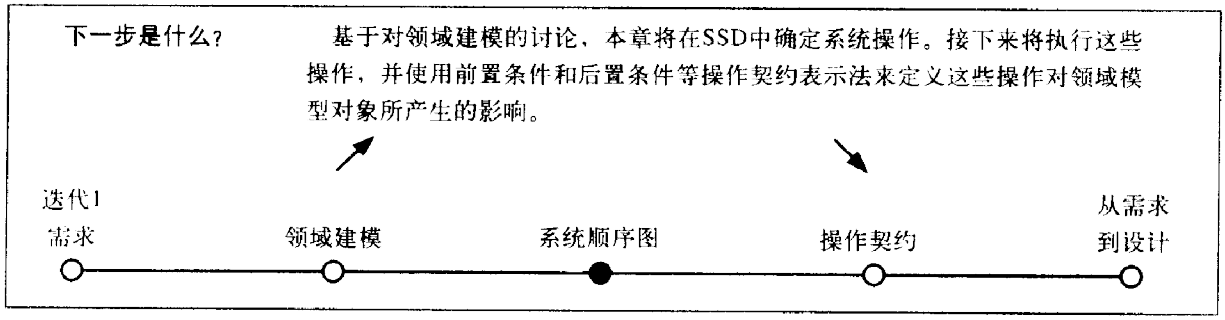

图10-1中所示的是强调系统顺序图的UP制品的相互影响。用例文本及其所示的系统事件是创建SSD的输入。SSD中的操作（例如enterItem）可以在操作契约中进行分析，在词汇表中被详细描述，并且（最重要的是）作为设计协作对象的起点。

10.1 示例：NextGen SSD

对于用例中一系列特定事件，SSD展示了直接与系统交互的外部参与者、系统（作为黑盒）以及由参与者发起的系统事件（如图10-2所示）。在图中，时间顺序是自上而下的，并且事件的顺序应该遵循其在场景中的顺序。

图10-2所示的示例是涉及现金支付的处理销售场景的主成功场景。其中给出了收银员发出的makeNewSale、enterItem、endSale和makePayment系统事件。这些事件是通过阅读用例文本而总结出来的。

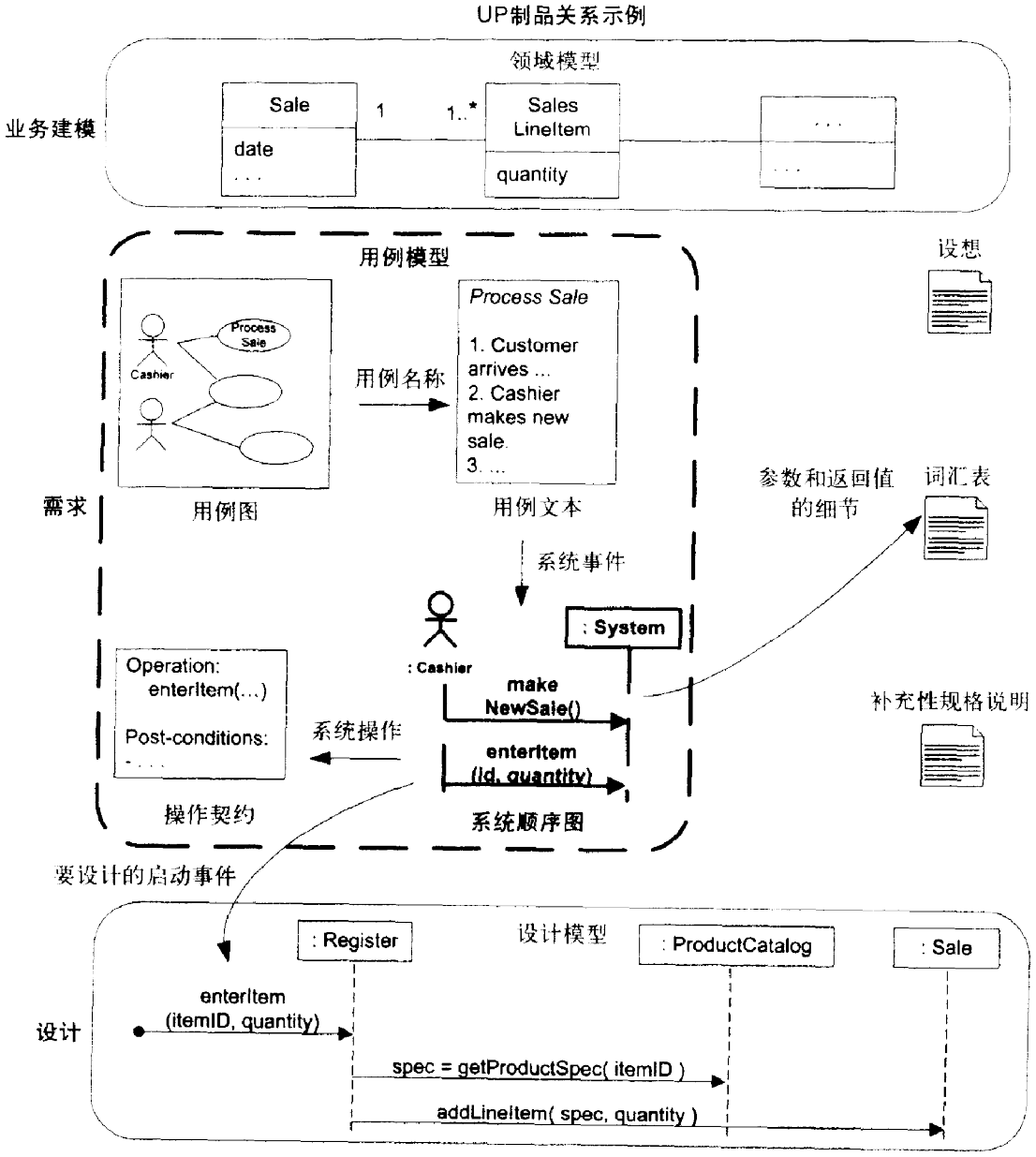

图10-1 UP制品示例的相互影响

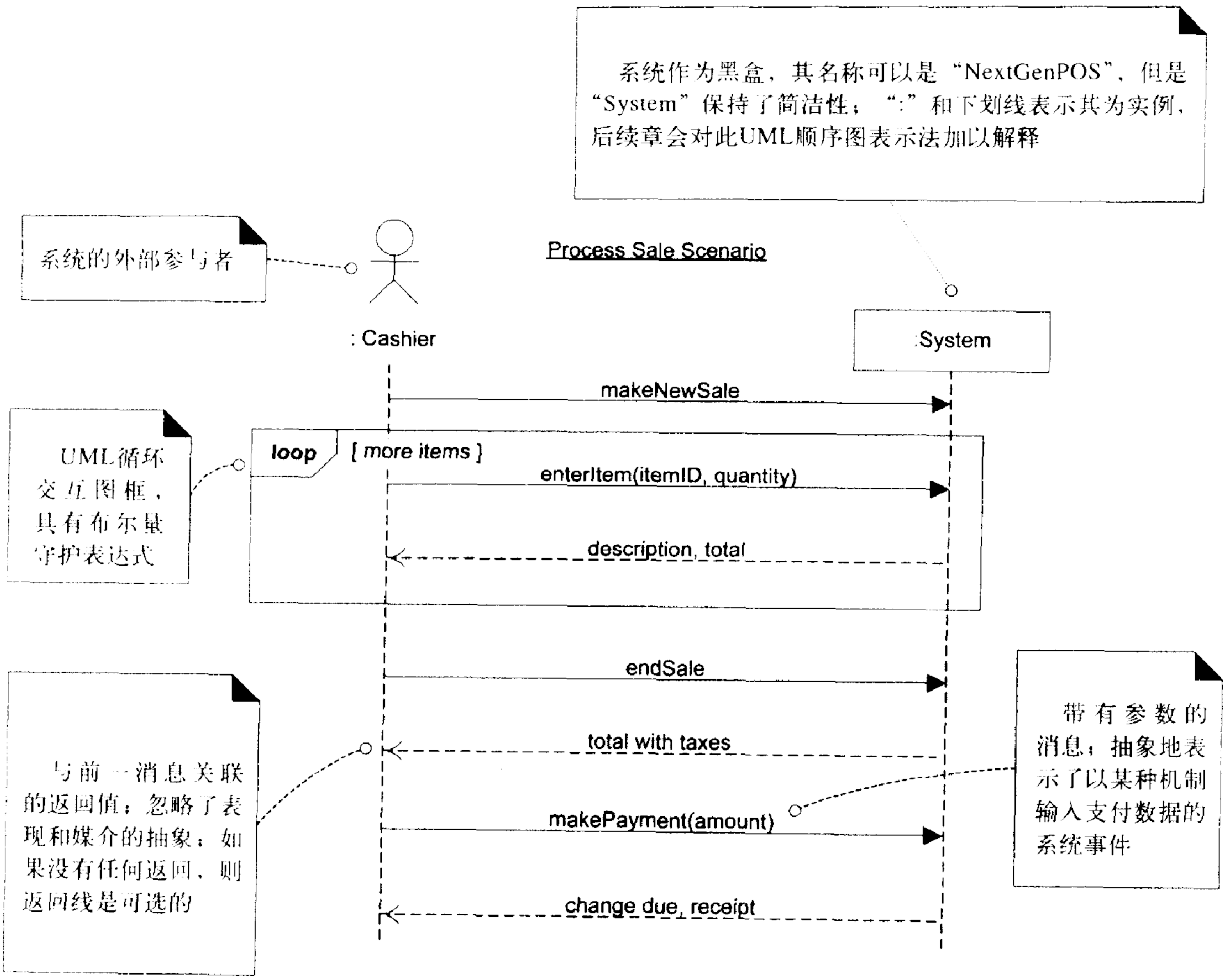

图10-2　处理销售场景的SSD

10.2　什么是系统顺序图

用例描述外部参与者是如何与我们所希望创建的系统进行交互的。在交互中，参与者对系统发起**系统事件**（system event），通常需要某些**系统操作**（system operation）对这些事件加以处理。例如，当收银员输入商品ID时，收银员要请求POS系统记录对该商品的销售（enterItem事件）。该事件引发了系统之上的操作。用例文本暗示了enterItem事件，而SSD将其变得具体和明确。

UML包含了**顺序图**作为表示法，以便能够阐述参与者的交互及参与者引发的操作。

系统顺序图表示的是，对于用例的一个特定场景，外部参与者产生的事件，其顺序和系统之内的事件。所有系统被视为黑盒，该图强调的是从参与者到系统的跨越系统边界的事件。

准　则
应为每个用例的主成功场景，以及频繁发生的或者复杂的替代场景绘制SSD。

10.3　动机：为什么绘制SSD

软件设计中一个有趣且有用的问题是：我们的系统中会发生什么事件？为什么？因为我们

必须为处理和响应这些事件（来自于鼠标、键盘、其他系统……）来设计软件。基本上，软件系统要对以下三种事件进行响应：1）来自于参与者（人或计算机）的外部事件，2）时间事件，3）错误或异常（通常源于外部）。

因此，需要准确地知道，什么是外部输入的事件，即**系统事件**。这些事件是系统行为分析的重要部分。

你可能对如何识别进入软件对象的消息非常熟悉。而这种概念同样适用于更高阶的构件，包括把整个系统（抽象地）视为一个事物或对象。

在对软件应用将如何工作进行详细设计之前，最好将其行为作为“黑盒”来调查和定义。**系统行为**（system behavior）描述的是系统做什么，而无需解释如何做。这种描述的一部分就是系统顺序图。

其他部分包括用例和系统操作契约（稍后进行讨论）。

10.4 应用UML：顺序图

UML没有定义所谓的“系统”顺序图，而只是定义了“顺序图”。这一限定强调将系统的应用视为黑盒。之后，我们会在另一个语境下使用顺序图，阐述完成工作的交互软件对象的设计。

顺序图中的循环

注意在图10-2中是如何使用**交互框**（interaction frame）来表示顺序图中的循环的。

10.5 SSD和用例之间的关系

SSD展示了用例中一个场景的系统事件，因此它是从对用例的考察中产生的（参见图10-3）。

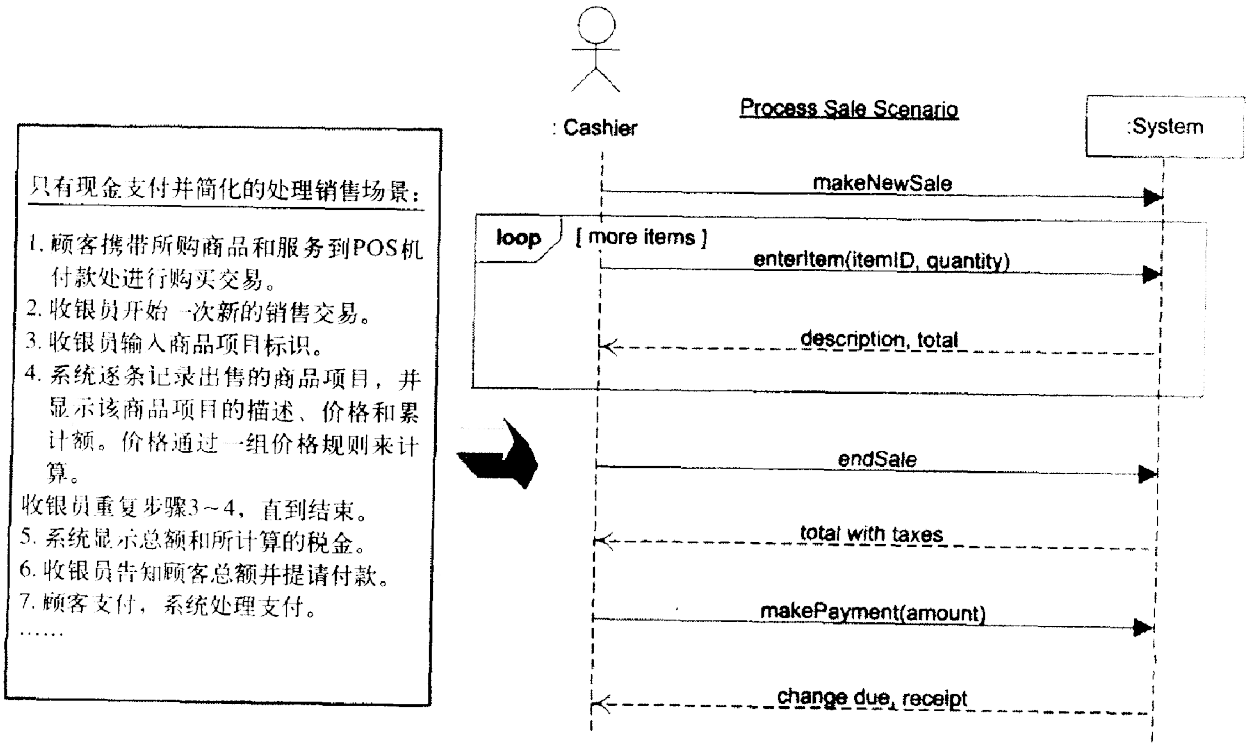

图10-3 SSD由用例导出，表示了一个场景

应用UML：是否应该在SSD中显示用例文本

通常不这么做。如果你为SSD适当地命名，可以指明对应的用例。例如，处理销售场景。

10.6　如何为系统事件和操作命名

scan（itemID）和enterItem（itemID）这两个名字，哪个更好？

系统事件应该在意图的抽象级别而非物理的输入设备级别来表达。

因此，“enterItem”要优于“scan”（也就是激光扫描），因为前者既捕获了操作的意图，又保留了抽象性，而不需要涉及使用什么样的接口来捕获系统事件。这一操作可以通过激光扫描器、键盘、声音输入设备或其他任何接口来完成。

如图10-4所示，系统事件的名称以动词开始（增加……，输入……，结束……，产生……），可以提高清晰程度，因为这样可以强调这些事件是命令或请求。

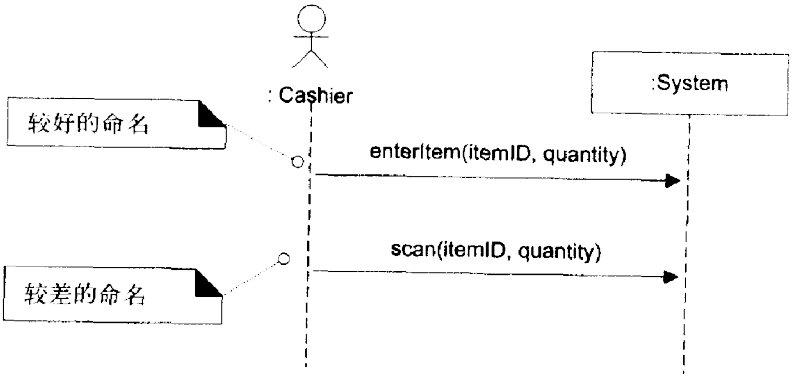

图10-4　在抽象级别上选择事件和操作的名称

10.7　如何为涉及其他外部系统的SSD建模

SSD也同样可以用来阐述系统之间的协作，例如NextGen POS和外部信用卡支付授权系统之间的协作。但这个问题我们会在案例研究的后继迭代中讨论，因为本次迭代不包括远程系统协作。

10.8　SSD的哪些信息要放入词汇表中

SSD中所示的元素（操作名称、参数、返回的数据）是简洁的。需要对这些元素加以适当的解释以便在设计时能够明确地知道输入了什么，输出了什么。词汇表是详细描述这些元素的最佳选择。

例如，在图10-2中，其中一条返回线含有描述：“change due, receipt”。其中关于票据（复杂报表）的描述是含糊的。所以，在UP词汇表中可以加入票据条目，显示票据样本（可以是数码图片）、详细内容和布局。

准　则
对大多数制品来说，一般在词汇表中描述其细节。

10.9　示例：Monopoly SSD

玩Monopoly游戏用例很简单，就是其主场景。观察者初始化游戏者的数量，然后请求模拟游戏，观察过程，直到产生赢家为止。参见图10-5。

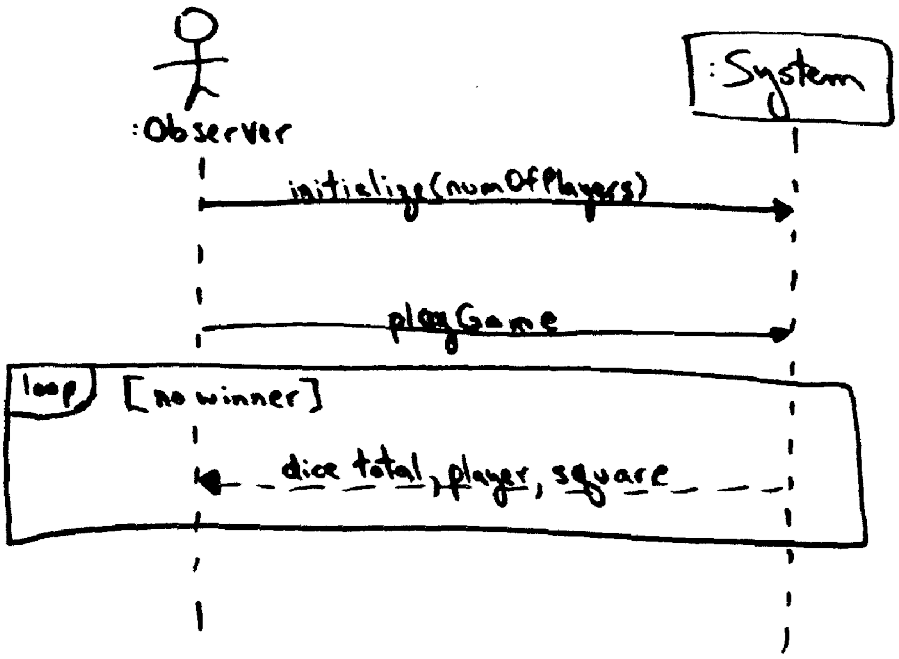

图10-5 玩Monopoly游戏场景的SSD

10.10 过程：迭代和进化式SSD

不用为所有场景创建SSD，除非你在使用需要识别所有系统操作的预算技术（例如功能点计数）。相反，只需为下次迭代所用的场景绘制SSD。同时，不应该花费太长时间来绘制SSD，用几分钟或半小时来绘制即可。

当需要了解现有系统的接口和协作时，或者将其架构记录在文档中时，SSD也是十分有效的。

UP中的SSD

SSD是用例模型的一部分，将用例场景隐含的交互可视化。尽管UP的创建者们意识到并理解这些图的用途，但最初的UP描述中并没有直接提到SSD。有大量可能有用和广泛使用的分析和设计制品或活动并没有在UP或RUP文档中提及，SSD就是其中之一。但是UP十分灵活，提倡引入任何能够增加价值的制品和实践。

UP阶段

初始——通常不会在该阶段引入SSD，除非你要对涉及的技术进行粗略的估算（不要指望初始阶段的估算是可靠的），这种估算的基础是对系统操作的识别，例如，**功能点**或COCOMO2（参见www.ifpug.org）。

细化——大部分SSD在细化阶段创建，这有利于识别系统事件的细节以便明确系统必须被设计和处理的操作，有利于编写系统操作契约，并且可能有利于对估算的支持（例如，通过未调整的功能点和COCOMO2进行宏观估算）。

10.11　历史和参考资料

确定软件系统公有操作的需求自来有之，所以数十年来，已经广泛使用了各种系统接口图，用以阐述将系统视为黑盒的系统I/O事件。例如在电信行业，这种图称为呼叫流程图。在OO方法中，首先普及使用这些图的是Fusion方法[⊖][Coleman+94]，该方法对SSD和系统操作与其他分析和设计制品之间的关系提供了大量示例。

⊖ UML前身之一。例如，UML的协作图是通过对Booch方法的对象图、Fusion方法的对象交互图以及其他一些方法中的相关图表改造而得到的。