Chapter 8-2

集成测试与系统测试

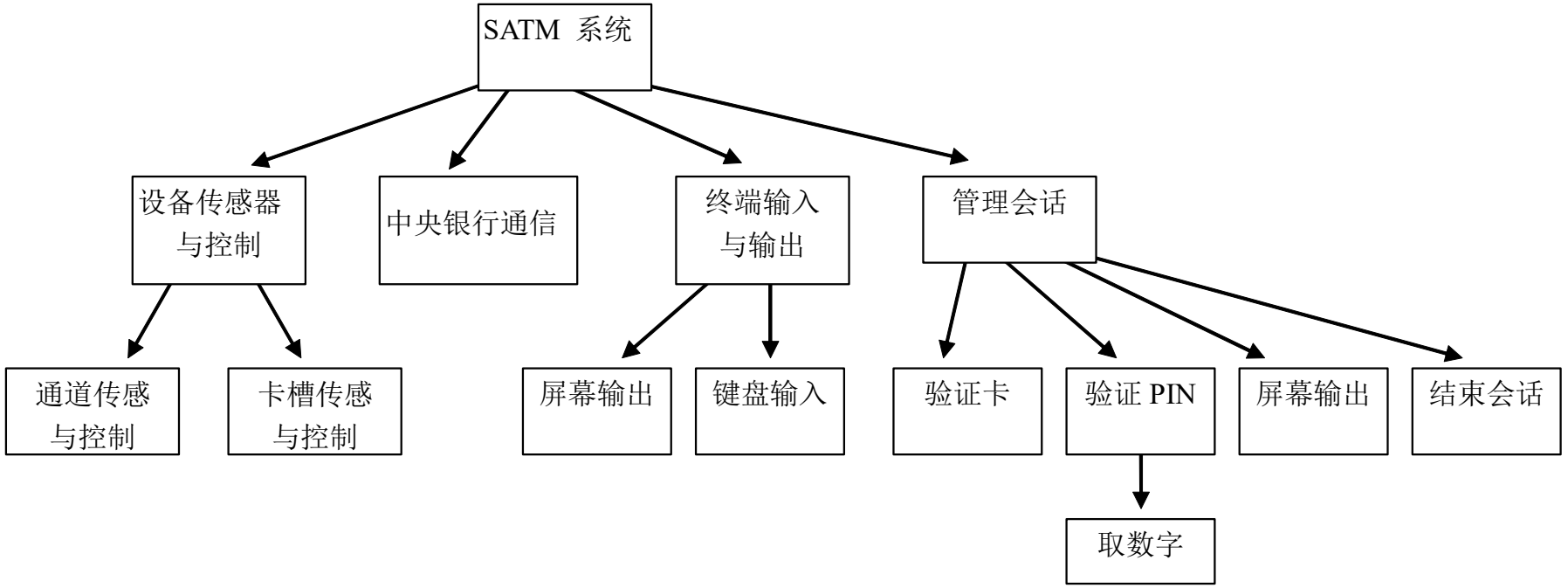
8.2 集成测试

8.3 系统测试

将集成测试与系统测试分开

- 从前面的介绍,我们知道集成测试针对的是模块之间的关系;而系统测试针对的是整个系统的功能。
- 集成测试需要了解程序的结构，是一种结构化的测试方法，有路径覆盖的含义。
- 系统测试不需要了解程序的结构，是一种黑盒的测试方法，是功能覆盖的意义。
- 集成测试是由软件开发人员完成的；而系统测试往往是需要用户的参与的。

事例 SATM


SATM 系统
设备传感器与控制
中央银行通信
终端输入与输出
管理会话
通道传感与控制
卡槽传感与控制
屏幕输出
键盘输入
验证卡
验证 PIN
屏幕输出
结束会话
取数字

8.2 集成测试

8.2.1 集成测试的方法

1）自顶向下集成 从主程序(顶层)开始,所有下层程序都以“桩程序”出现。完成顶层测试后，以真实程序代替“桩程序”，向下进行下一层测试。

“桩程序”(stub)：？

模拟被调用程序的代码。一般以表格形式存在。

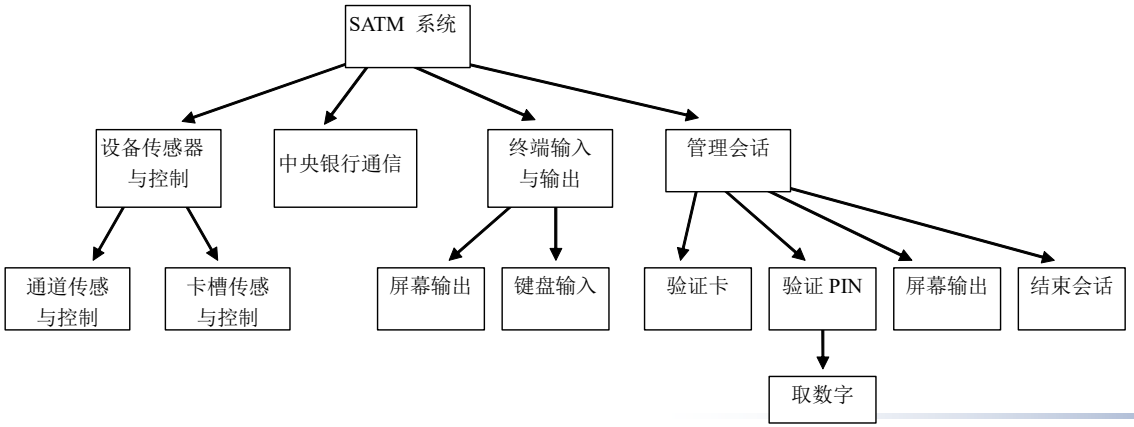
2）自底向上集成 从程序的最下层节点（叶子）开始，通过编写“驱动器”完成测试，然后以真实程序代替“驱动器”，向上进行上一层测试。

“驱动器”：模拟对测试节点的调用驱动。

3）三明治集成 是自顶向下和自底向上测试的组合，即可以同时从顶和底向中间层集成，可以减少桩程序和驱动的数量。

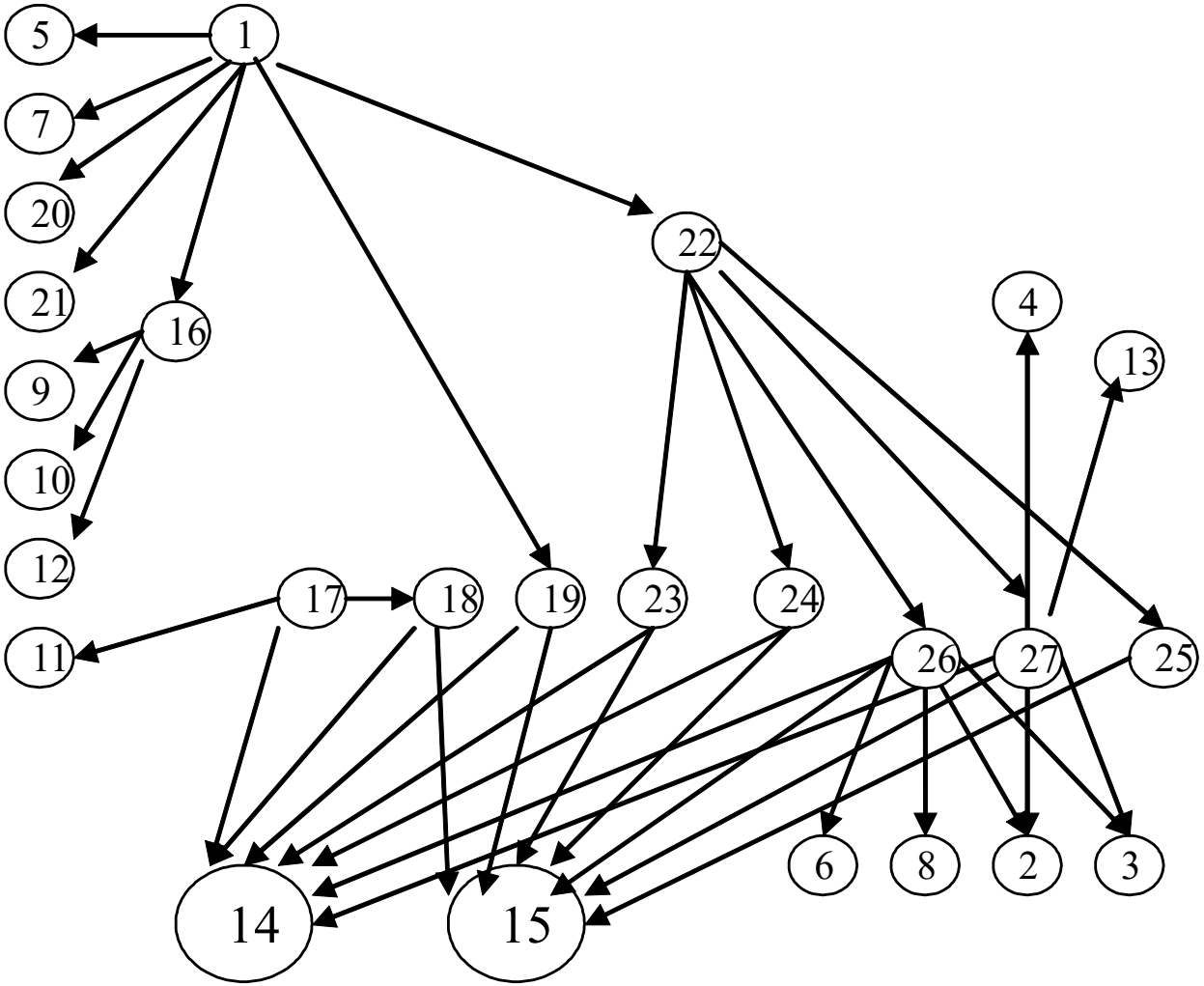
4）大爆炸测试 不分层次，将所有单元放在一起编译，并进行一次性测试。

对于SATM系统，我们知道需要开发（节点-1）个桩程序：？个；需要开发（非叶子节点）个驱动器：？个。



8.2.2 基于调用图的集成

换一个角度，从模块之间调用关系的角度，我们可以得到SATM的调用图。



1) 成对集成

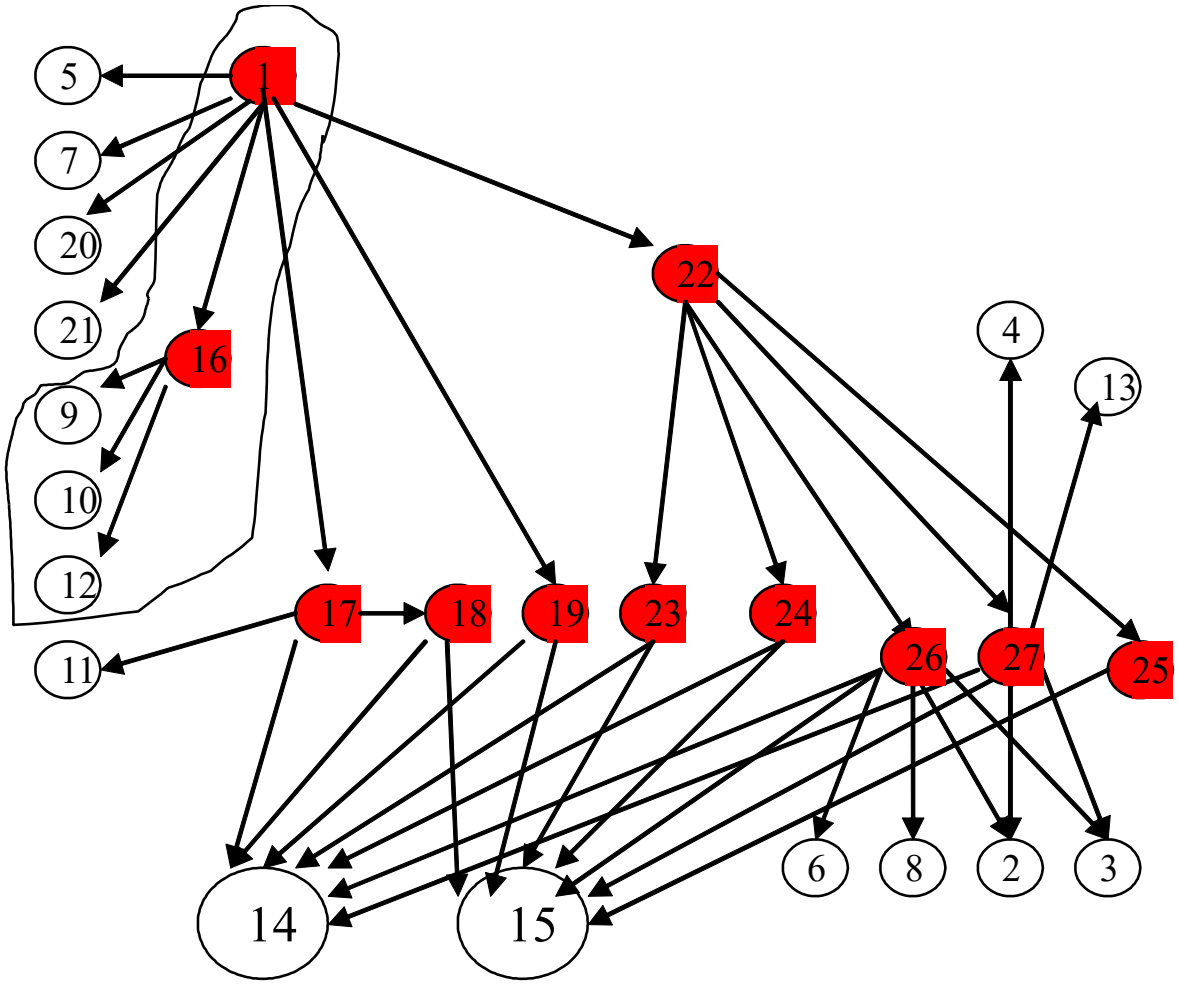
为免除桩程序和驱动器的开发,可采用调用对的测试方法。SATM的成对测试集，就是调用图中边的数量，共40个。

2) 相邻集成

为减少测试的数量，以相邻节点为集合，进行测试。

相邻节点：包括所有直接前驱和所有直接后继节点。

对于SATM，显然就是除去叶子节点的所有节点的相邻集合，共？个。


5
1
7
20
21
16
9
10
12
11
22
4
13
17
18
19
23
24
26
27
25
14
15
6
8
2
3

8.2.3 基于路径的集成

在面向结构的测试中,我们经常采用路径覆盖的方法,在集成测试中,我们也有类似的概念。

◆ 概念

定义：程序中的源节点是程序开始或重新开始处的语句片段。

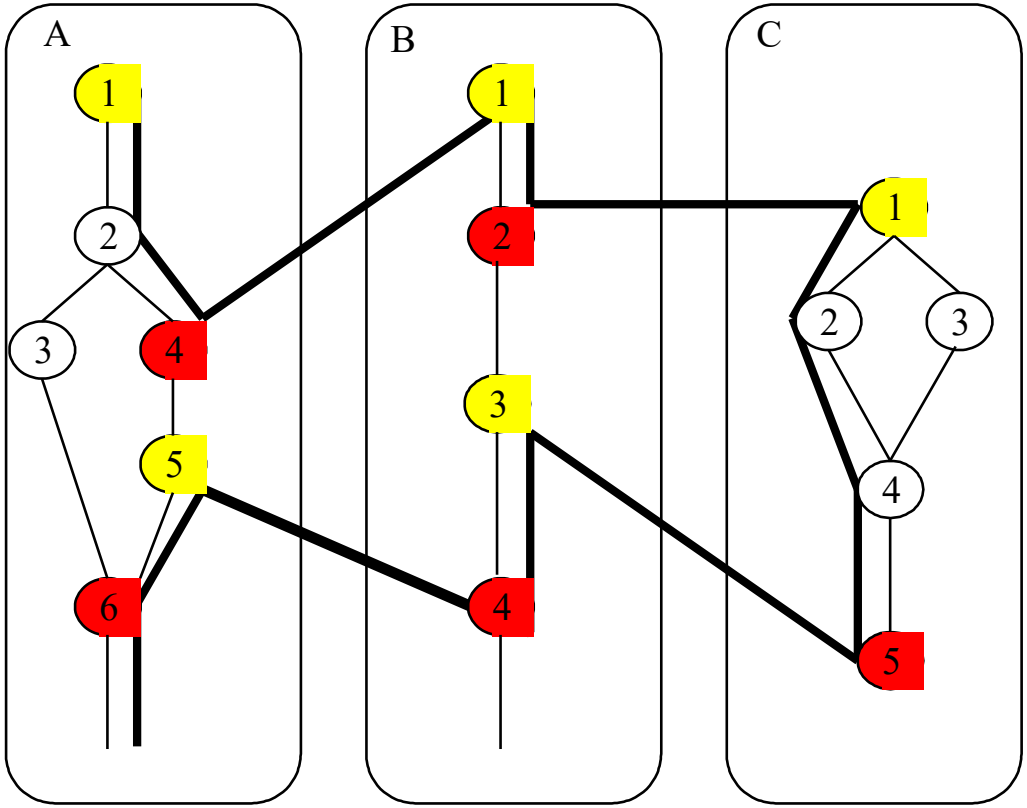
汇节点是程序执行结束处的语句片断。

模块执行路径是以源节点开始，以汇节点结束的一系列语句，中间没有插入汇节点。

消息是一种程序设计语言机制，通过它，一个单元将控制转移给另一个单元。

MM-路径是穿插出现模块执行路径和消息的序列。

下图就是在模块A、B、C之间控制转移的MM-路径



源节点：　　汇节点：

上图共有7条模块执行路径：

MEP（A，1）=<1,2,3,6>

MEP（A，2）=<1,2,4>

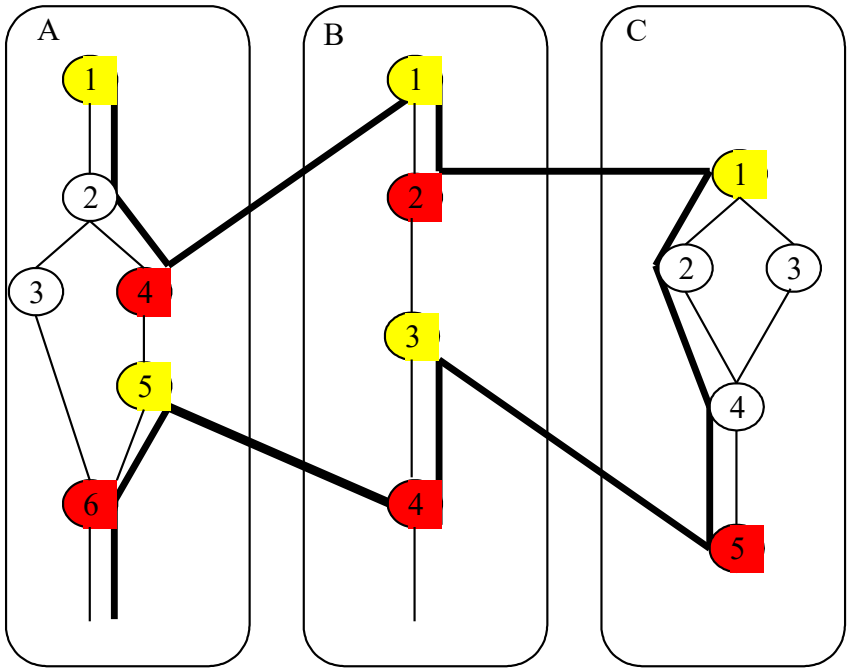
MEP（A，3）=<5,6>

MEP（B，1）=<1,2>

MEP（B，2）=<3,4>

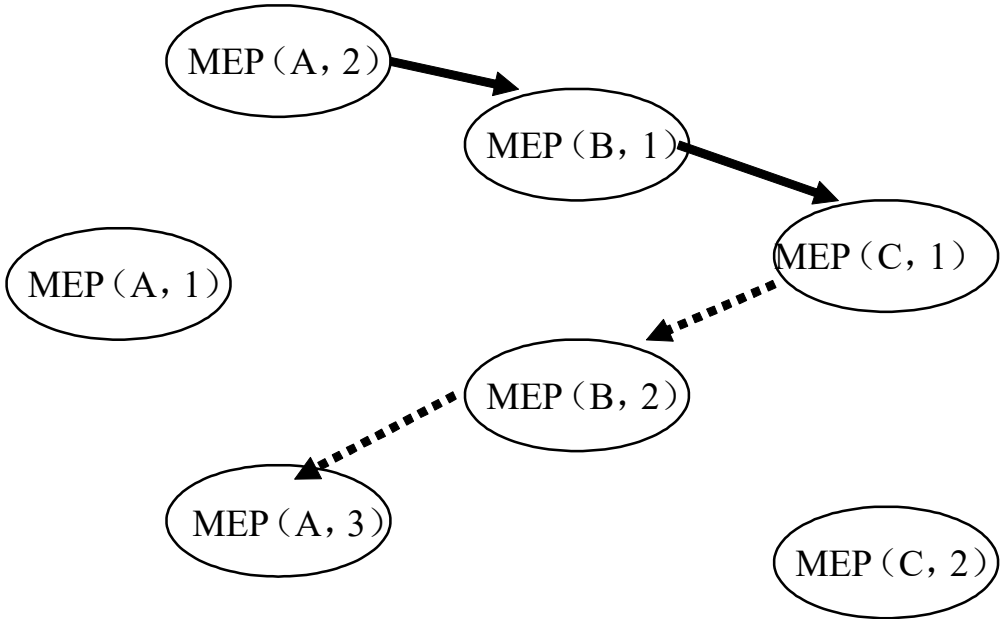
MEP（C，1）=<1,2,4,5>

MEP（C，2）=<1,3,4,5>



定义:给定一组单元,其MM-路径是一种有向图,其中的节点表示模块执行路径,边表示消息和单元之间的返回。

那么，上面的MM-路径就表示为：



实线箭头表示消息，返回由虚线箭头表示。

- DD-路径：模块内的程序执行路径；
- MM-路径：模块间的模块执行路径序列。

请同学自己阅读NextDate的例子（P169~P174 ）

其中，UML序列图是MM-路径图的表示方式，因此，UML序列图可以作为集成测试的路径覆盖标准。

集成测试策略比较

策略	对接口的测试能力	对交互功能的测试能力	故障分离辨别力
功能分解	满意但有可能不可靠	有限单元对	好，尤其是有故障单元
调用图	满意	有限单元对	好，尤其是有故障单元
MM路径	优秀	全部单元	优秀，尤其对有故障单元执行路径

- MM-路径复杂度

  圈复杂度的计算：V（G）=e-n+2p

  其中，e为边数，n为节点数。

  大家很容易得到P168页对图13-13的计算结果。

  双向箭头表示2条边。

8.3 系统测试

从前面的介绍我们知道有：

- 单元测试：白盒(侧重)、详细设计文档；
- 集成测试：白盒、概要设计文档；
- 系统测试：从外部特性对软件进行测试，功能测试。

标准：需求规格说明！

测试方法？

- 如何描述系统的功能，其实是通过系统的输入输出来分析。
- 换言之，我们可以通过系统从输入到输出的行为线索来描述系统。

8.3.1 线索(thread)

线索有不同的层次,单元级线索被理解为指令执行路径,或DD-路径;集成测试线索是MM-路径,即模块执行和消息交替序列。那么系统级线索，就是原子系统功能序列。

这样的定义并不令人满意，换个角度，单元测试的线索是模块内的路径执行序列（有意义的最小单元）；集成测试的线索是模块间的路径执行序列（是小于系统级的意义单元）；系统级的线索，是系统输入到输出的路径（是功能的最小单元）。

本章，我们讨论系统级线索。

SATM中的可能线索：？

- 数字输入-屏幕输出
- 密码（PIN）验证
- 单一事务：存款、取款、查询余额
- 多个事务：包括两个或多个业务活动

- 当然，仅对于密码验证而言，还可以细化为插卡和密码输入2个线索。
- 而密码可以有三次尝试。

定义：原子系统功能（ASF）：是一种在系统层可以观察得到的端口输入和输出事件的行动。

- ASF具有事件静止特性，ASF开始于一个端口输入事件，遍历一个或多个MM路径，以一个端口输出事件结束。
- ASF具有事件序列原子化（不愿再细分）特性。

- 卡输入
- PIN输入？是ASF吗？

我们总是希望有图形化的方法，来描述系统需求。因此，我们就需要有能够描述系统功能的图。

定义：给定通过原子系统功能描述的系统，系统的ASF图是一种有向图，其中的节点表示ASF，边表示串行流。

定义：源ASF是一种原子系统功能，在系统ASF图中作为源节点出现。汇ASF也是一种原子系统功能，在系统ASF图中作为汇节点出现。

定义：系统线索在系统的ASF图中，是从源ASF到汇ASF的路径。

8.3.2 需求规格说明的基本概念

数据：整数、浮点数、字符串、数组、结构体等；

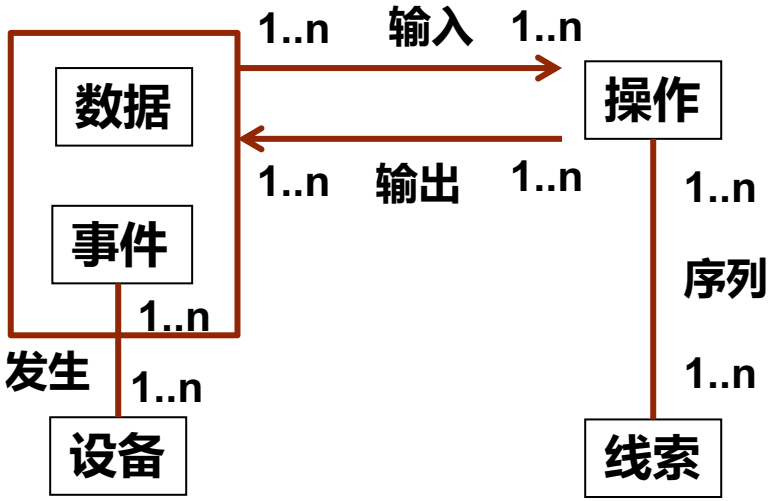
操作：输入、输出、转换、处理、活动、任务、方法、服务；

设备：端口设备、系统I/O接口；

事件：发生在端口设备上的系统级输入（或输出），是操作+数据。

线索：最难把握的概念，根据定义，是从源ASF到汇ASF的路径。因此，要测试线索，就必须画出系统ASF图。

基本概念之间的E/R图（图14-3）



基于模型的线索

- 建立系统的ASF图，是软件建模的方式之一。换言之，是建立需求规格的形式化方法之一。

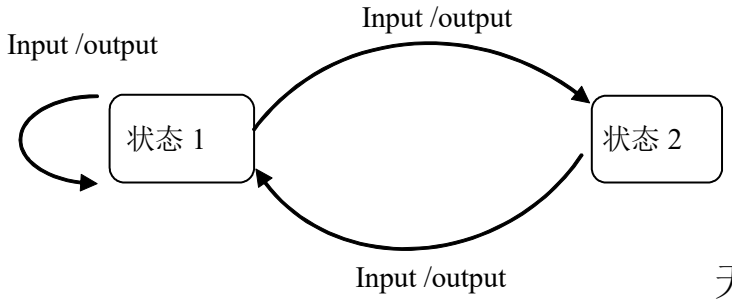
- 有限状态机（FSM）是研究系统功能级线索的有效手段。换言之，有限状态机恰是我们前面定义的ASF图。

  FSM：节点=ASF

  边=事件和行动

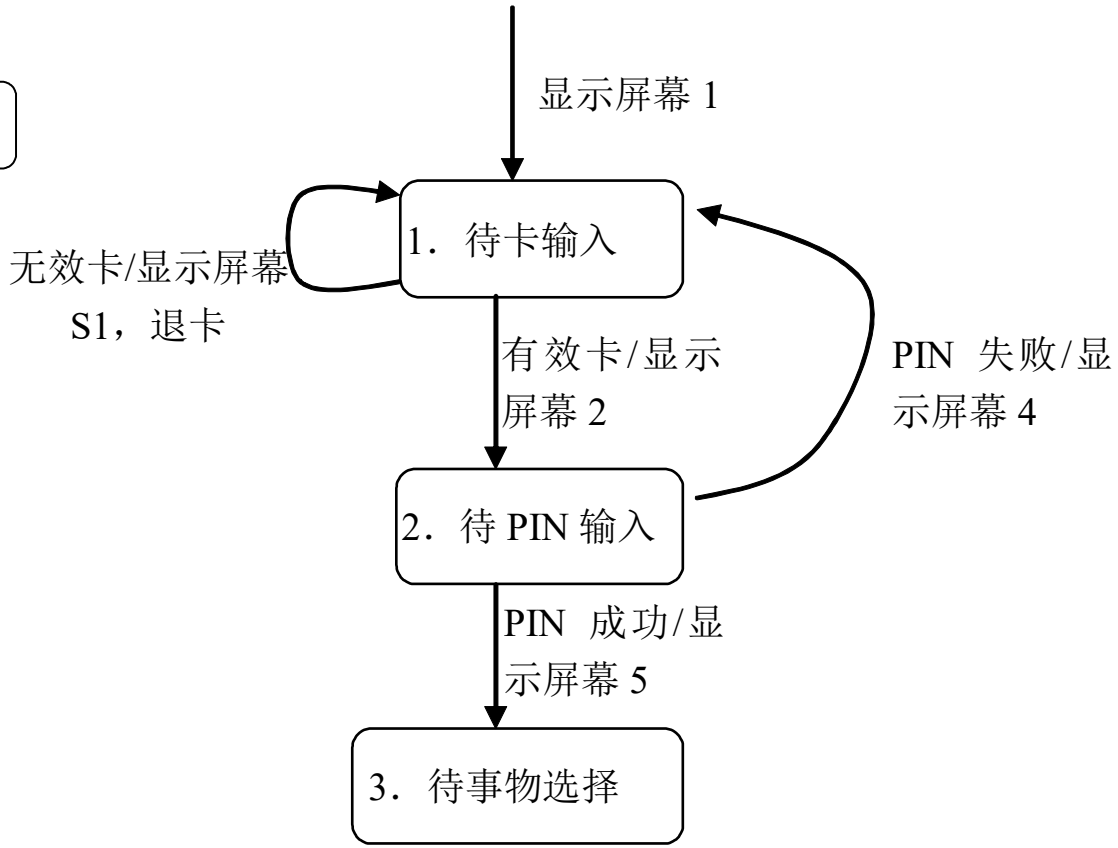
8.3.3 建立系统ASF图

FSM的一般形式

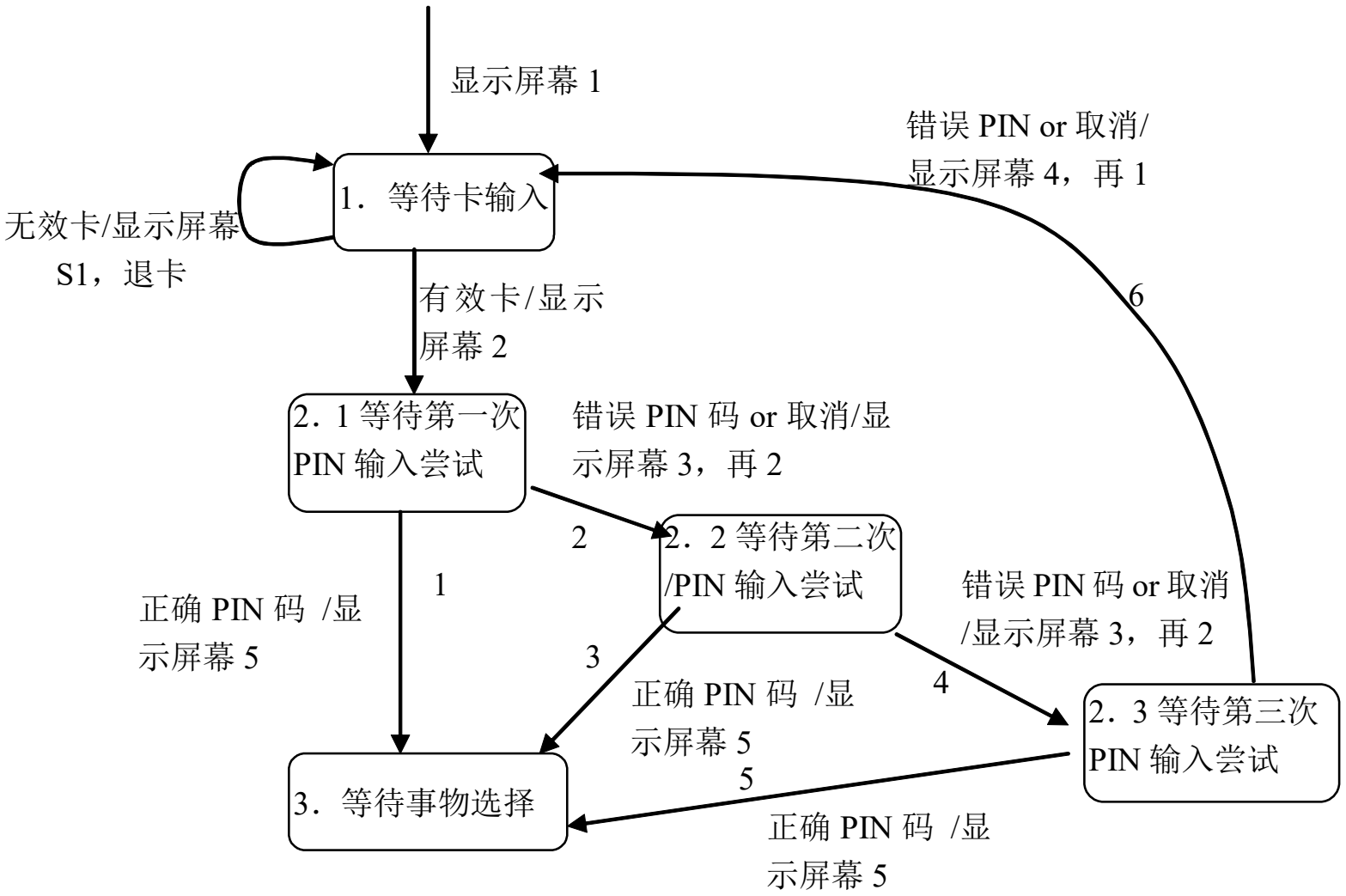


注：Input，output可以是多值组合

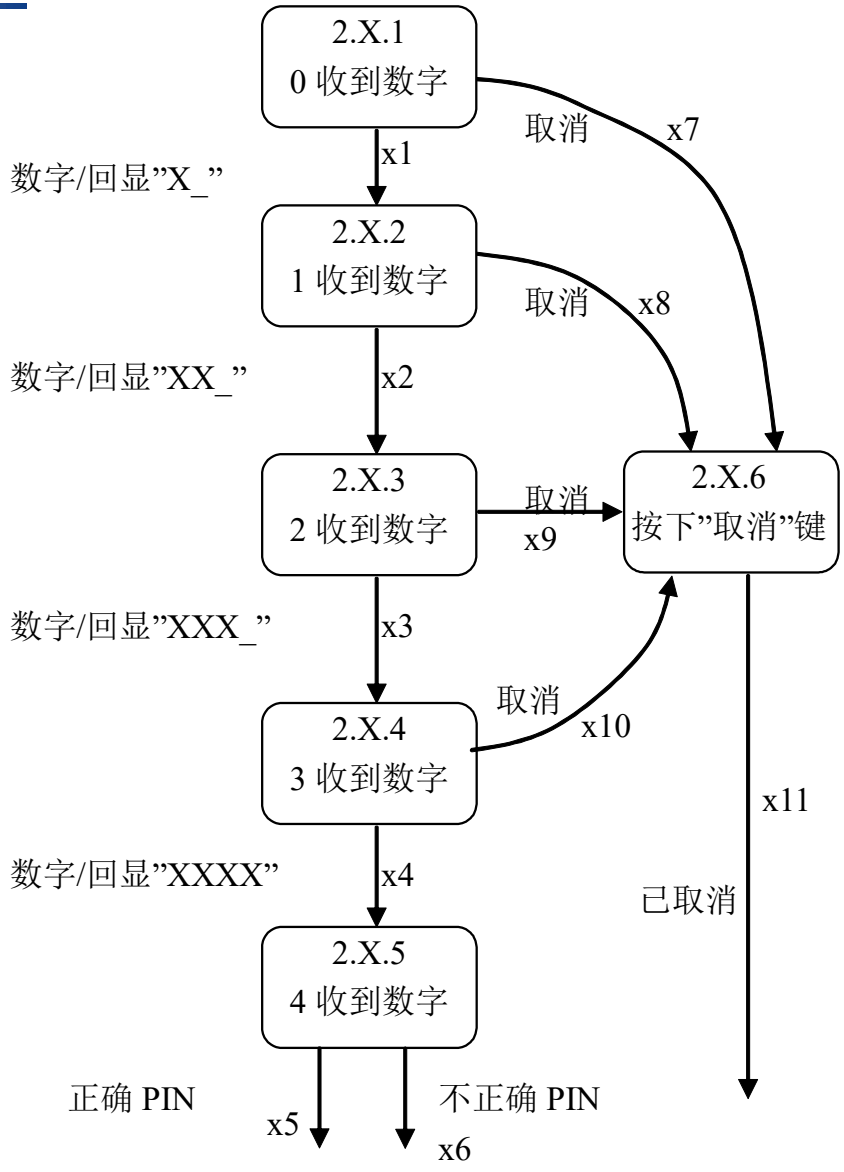
顶层的SATM的FSM图：



对PIN输入的细化状态机



- 进一步的细化,我们知道每次PIN需要有4位数字输入,而且在<4个数字输入后,都可以按“取消”键返回。则PIN的过程可以细化为右图。
- 为了使测试用例明确,我们不妨设定PIN密码为:1234



我们来算算有多少不同的线索（从源到汇的路径）

仅计算一下正确PIN的路径：

第1次PIN正确：1条；

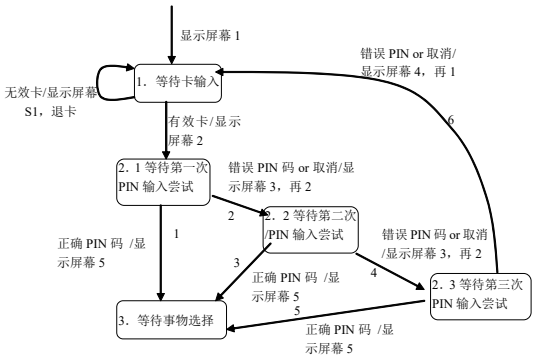
第2次PIN正确：5条；

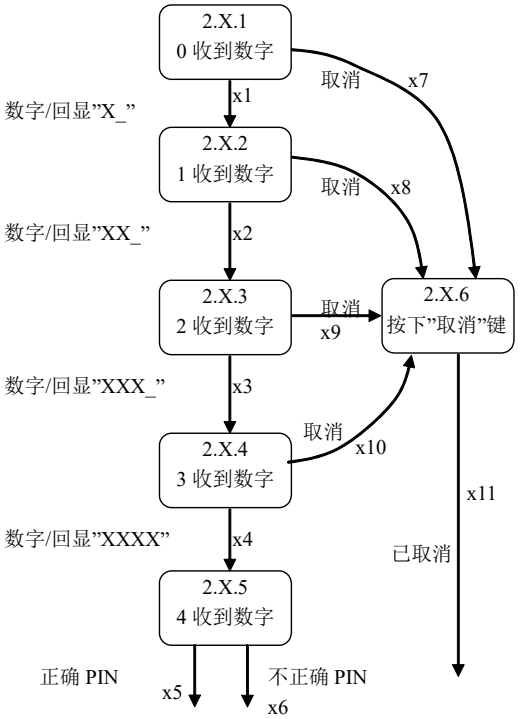
第3次PIN正确：25条；

共31条；

不正确的路径：125条，为什么？

第1次PIN正确的事件序列：P183 表14-1





8.3.4 线索测试的结构策略

在获得系统状态图之后,如何确定测试用例呢?

既然已经有系统的有限状态图,当然测试用例的选择就是考虑对路径的覆盖了。

对于PIN的细化状态图，共有6条路径。

输入事件序列	路 径
1234	x1、x2、x3、x4、x5
1235	x1、x2、x3、x4、x6
C	x7、x11
1C	x1、x8、x11
12C	x1、x2、x8、x11
123C	x1、x2、x3、x10、x11

基于用例的线索

- 用例的层次
  - 高级用例（非常类似于一个敏捷开发故事）
  - 基本用例
  - 基本扩展用例

- 问题：
  - 如何利用需求分析中的用例进行系统测试？
  - 用例=用例图+用例描述（？）

8.3.5 基于规格说明系统测试的覆盖率

基于状态图的测试方法很有效，但并不是所有的软件系统都能够方便的得到状态图。而系统功能确是各个系统都十分明确的线索，下面我们就从功能定义的三个基本构件：事件、端口和数据，讨论测试的线索。

1) 基于事件的线索测试

从端口的输入事件考虑，有5个覆盖标准：

- PI1：每个端口输入事件发生。
- PI2：端口输入事件的常见序列发生。
- PI3：每个端口输入事件在所有“相关”数据语境中发生。
- PI4：对于给定语境，所有“不合适”的输入事件发生。
- PI5：对于给定语境，所有可能的输入事件发生。

- PI1是易于达到的，PI2是基本可行的。而PI3就有可能形成测试爆炸，而PI4和PI5往往是供参考的选项。

从端口的输出事件考虑，可以有两种覆盖指标：

- PO1：每个端口输出事件发生。
- PO2：每个端口输出事件在每种情况下发生。

- PO1是基本的要求，PO2不仅要求所有的输出事件，而且要考虑导致这种输出的所有可能原因，这个要求往往也是难以完全满足的。

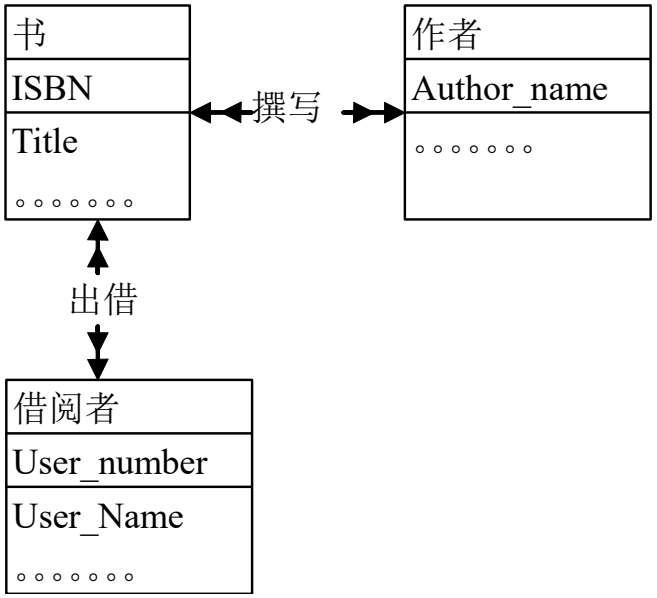
2) 基于端口的线索测试

- 基于端口的线索测试是基于事件的测试的有用补充。
- 基于事件的测试以事件为中心，考虑的是事件到端口的一对多测试；
- 基于端口的测试以端口设备为中心，考虑的是端口到事件的一对多测试。

3) 基于数据的线索测试

- 基于端口和事件的测试适合主要以事件驱动的系统，但并不是所有的系统都是事件驱动系统，例如以数据库为基础的系统，主要是数据的操作，此时就可以采用基于数据的测试。

  以E-R模型作为测试的线索。



覆盖指标：

- DM1：检查每个关系的基数。
- DM2：检查每个关系的参与。
- DM3：检查关系之间的函数依赖关系。

  基数指：关系间的一对一、一对多、多对一、多对多关系。

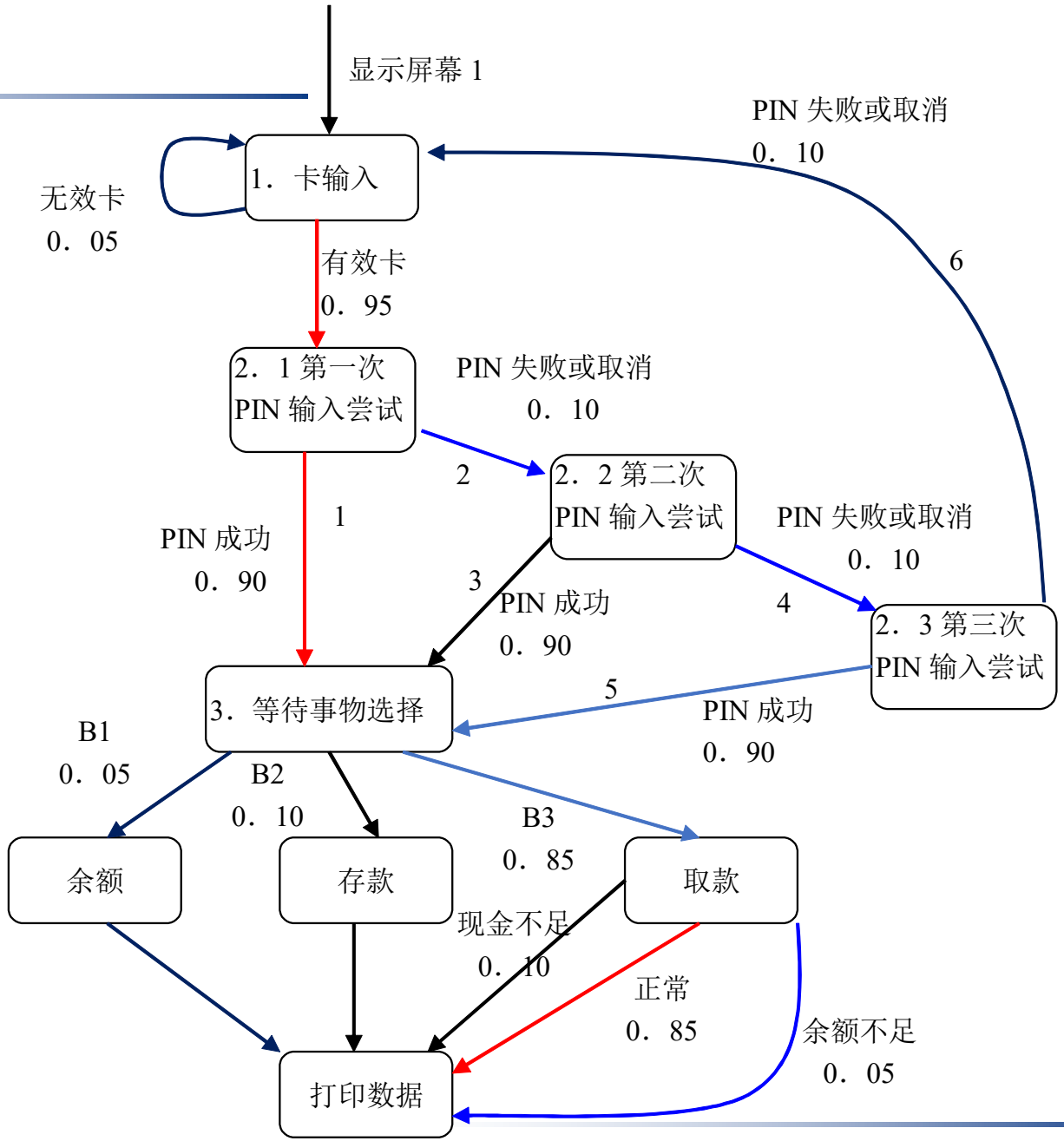
SATM 实例（由学生阅读）

8.3.6 系统测试指导方针

- 伪结构系统测试
  - 伪结构是系统的行为模型，是系统实际情况的近似模型。
  - 决策表、状态图、Petri网是系统功能性测试的常见选择。
- 运行剖面
  - 齐夫定理（Zipf's Law）在大多数情况下都成立。即80%的活动发生在20%的空间里（二八定理）。
  - 因此，确定各种线索的执行频率，对事件发生频率高的线索执行测试，可以大大提高测试的效率。

常见线索：

罕见线索：



概率树计算

基于风险的测试

- 运行剖面提供了系统的使用情况，对优化系统测试有意义。
- 更进一步的，对客户使用情况的了解，是改进系统设计的重要依据。对于发生概率极小甚至为零的线索，也许恰恰占用了系统的极大资源！

风险=代价*发生概率

- 失效代价通过四种风险类别给定：如1代表失效代价低、3为中间值、10为代价高
- 表14-14给出了SATM系统的风险评估结果

Q&A