顺序图与用例的关系 I

- 顺序图表达**单个情景**实例的行为。
- 每个用例对应一个顺序图。
- 顺序图表达对象间如何**协作**完成用例所描述的功能。
- 顺序图用于表示为完成用例而在系统边界输入输出的**数据**以及**消息**
- 顺序图也用于表示系统内部对象间的消息传递。

顺序图与用例对应

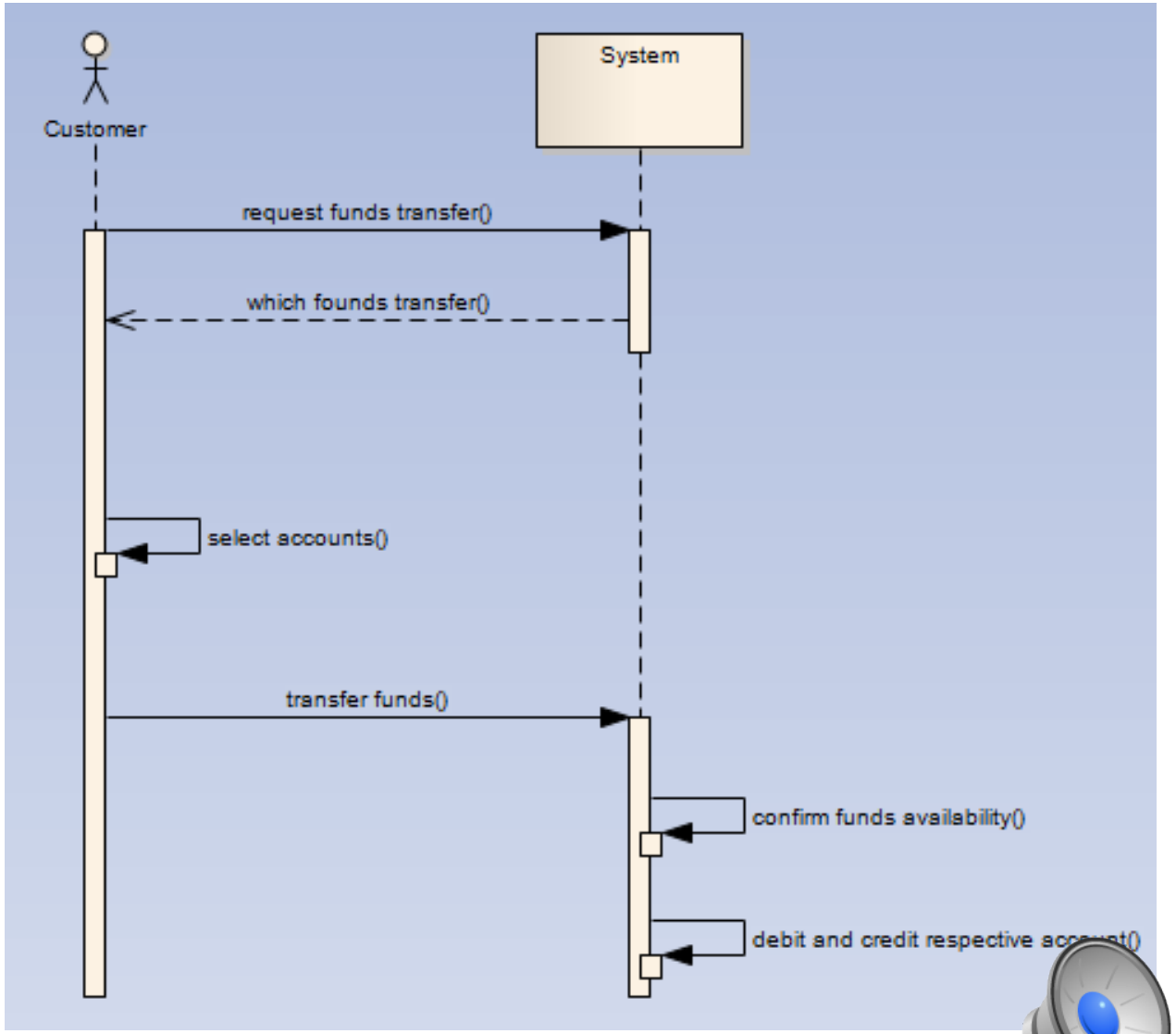
1. The customer requests a funds transfer.

2. The system asks the user to identify the accounts between which funds are to be transferred and the transfer amount.

3. Customer selects the account to transfer funds from, the account to transfer to, and then indicates the amount of funds to transfer.

4. The system checks the account from which funds are to be transferred and confirms that sufficient funds are available.

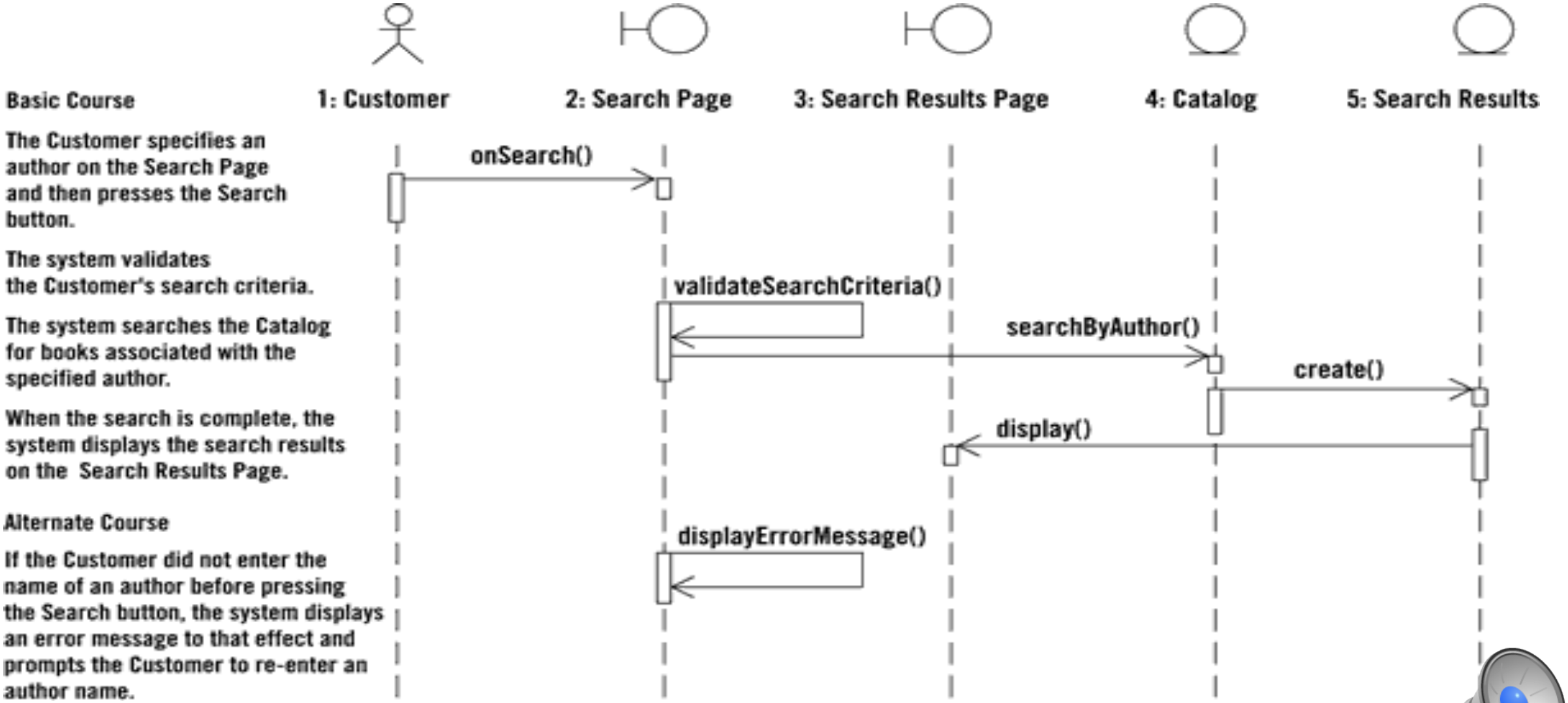
5. The amount is debited to the account from which funds are to be transferred and credited to the account previously selected by the customer by the system.



顺序图与用例的关系 II

- 顺序图可帮助分析人员对用例图进行扩展、细化和补遗
- 顺序图可用于开发周期的不同阶段，服务于不同目的，描述不同粒度的行为
- 分析阶段的顺序图**不要**
  - 包含设计对象
  - 关注消息参数

从用例中抽取顺序图


Basic Course
1: Customer
2: Search Page
3: Search Results Page
4: Catalog
5: Search Results
The Customer specifies an author on the Search Page and then presses the Search button.
onSearch()
The system validates the Customer's search criteria.
validateSearchCriteria()
The system searches the Catalog for books associated with the specified author.
searchByAuthor()
create()
When the search is complete, the system displays the search results on the Search Results Page.
display()
Alternate Course
If the Customer did not enter the name of an author before pressing the Search button, the system displays an error message to that effect and prompts the Customer to re-enter an author name.
displayErrorMessage()

顺序图建模意义

- 通过顺序图描述算法逻辑
- 高质量的顺序图是代码的抽象
- 顺序图是与语言无关的表示方式
- 可以绘制顺序图来描述业务逻辑
- 可以通过团队协作完成顺序图的绘制
- 可以在同一页浏览多个对象和类的行为

顺序图建模风格

- 建模风格1：把注意力集中于关键的交互。
  - 创建模型时要把注意力集中于系统的关键方面，而不要包括无关的细节。

    例如：

    如果顺序图是用于描述业务逻辑的，就没必要包括对象和数据库之间的详细交互。

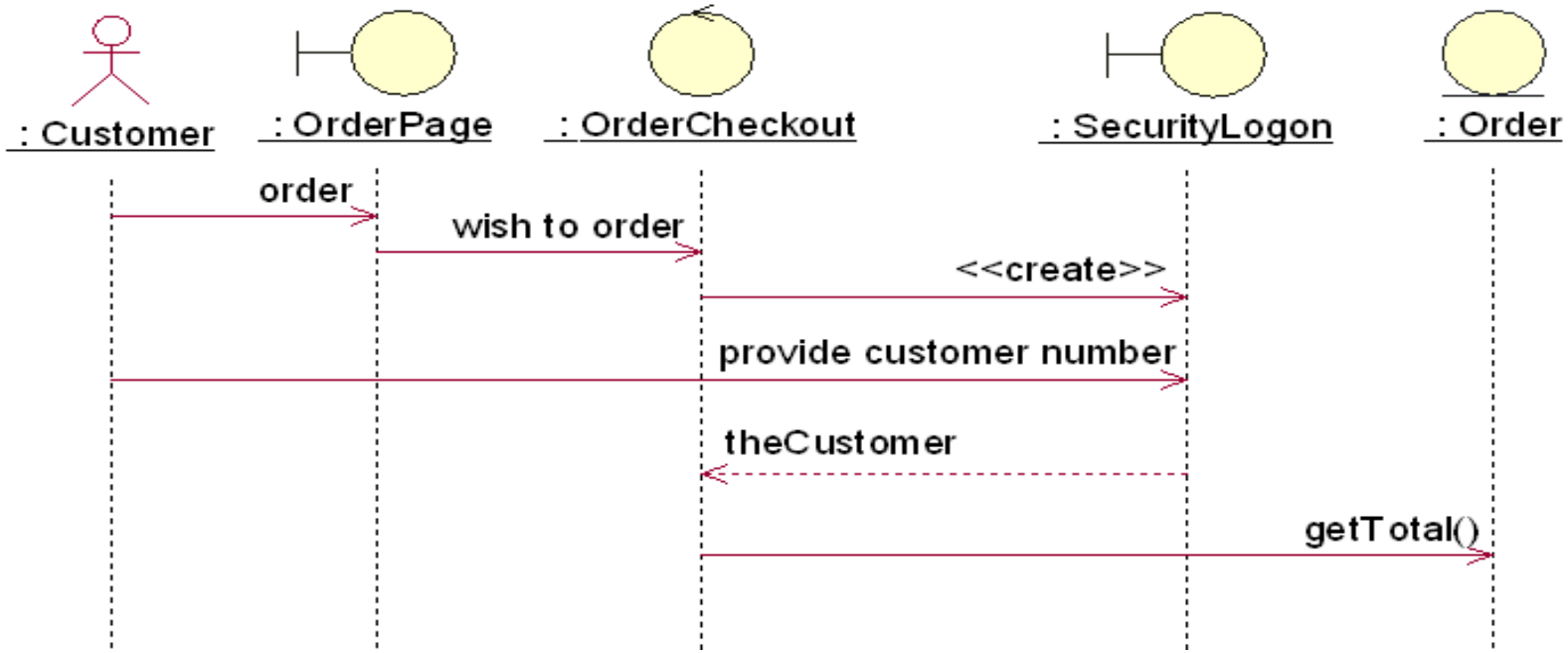
顺序图建模风格

- 建模风格2：对于参数，优先考虑使用参数名而不是参数类型。
  - 例如

    消息 addDeposit(amount, target)比 addDeposit(Currency, Account) 传递了更多的信息
  - 在消息中只使用类型信息不能传递足够的信息
  - 参数的类型信息用UML类图表示更好

顺序图建模风格

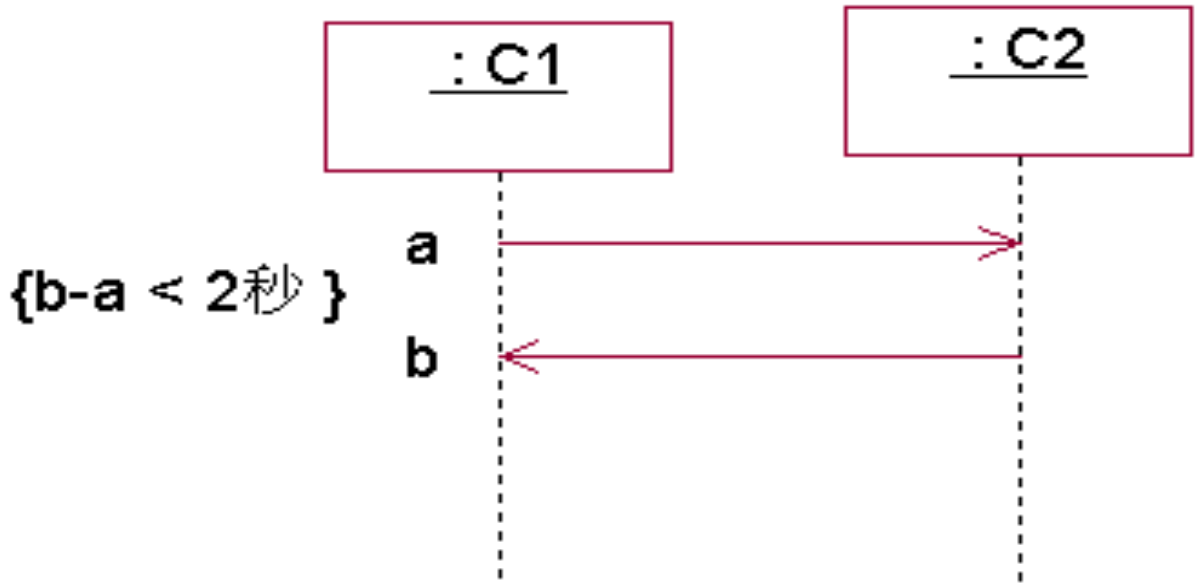
- 建模风格3：不要对明显的返回值建模。



- 建模风格4：可以把返回值建模为方法调用的一部分。

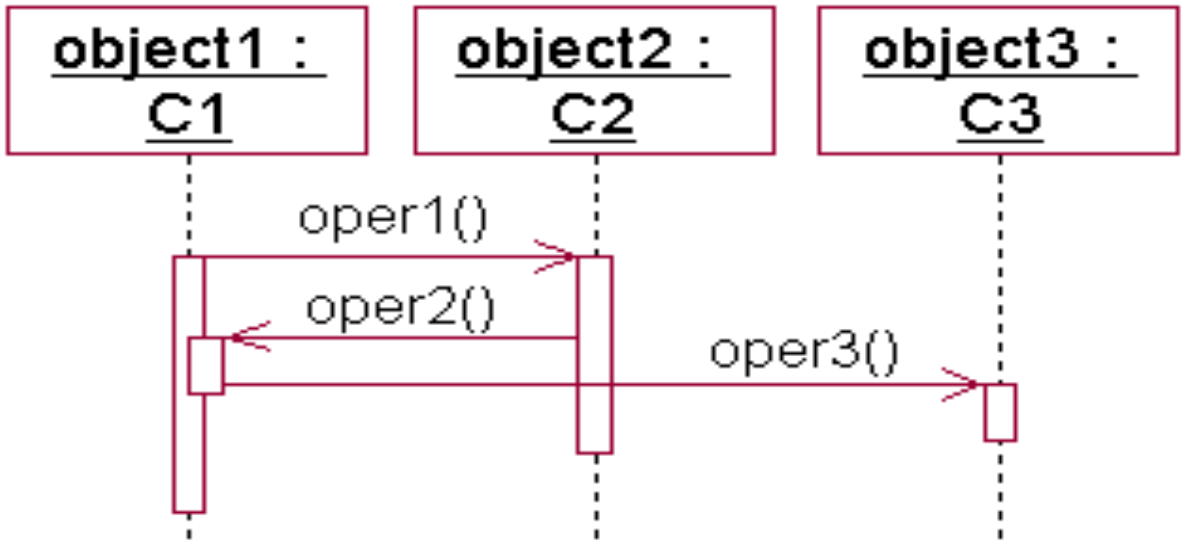
顺序图常见问题分析

- 顺序图中时间约束的表示
  - 用约束 (constraint)来表示。



控制焦点(focus of control)的嵌套

- 嵌套的FOC可以更精确地说明消息的开始和结束位置。
- 图例：



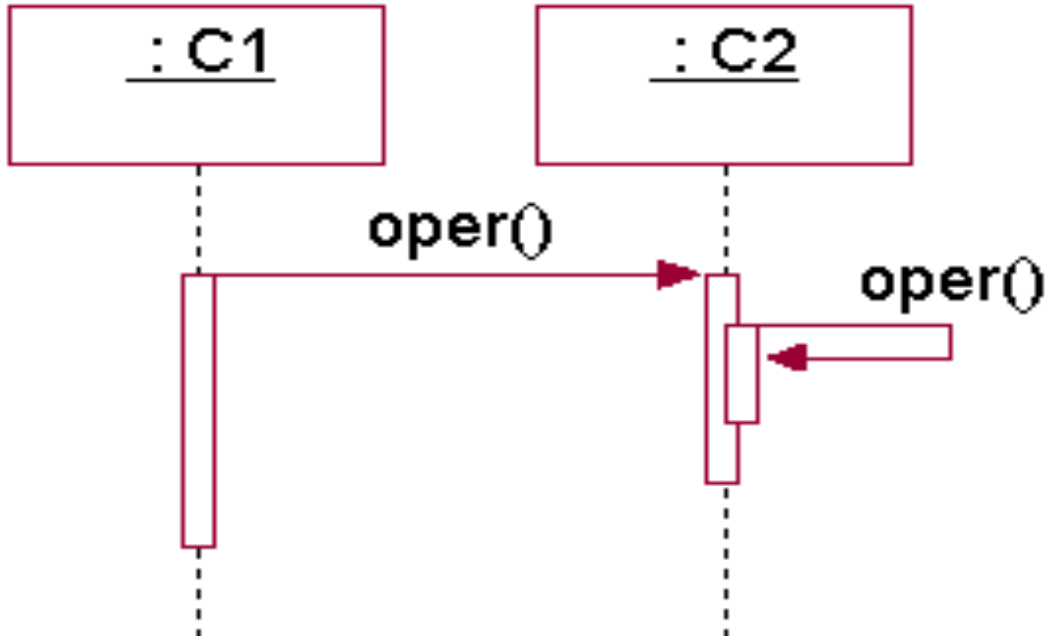
激活期(activation)：表示对象执行一个动作的期间（直接操作或者通过下级操作），也即对象激活的时间段。

控制焦点和激活期是同一个概念。

顺序图常见问题分析

- 顺序图中递归的表示
  - 利用嵌套的FOC表示

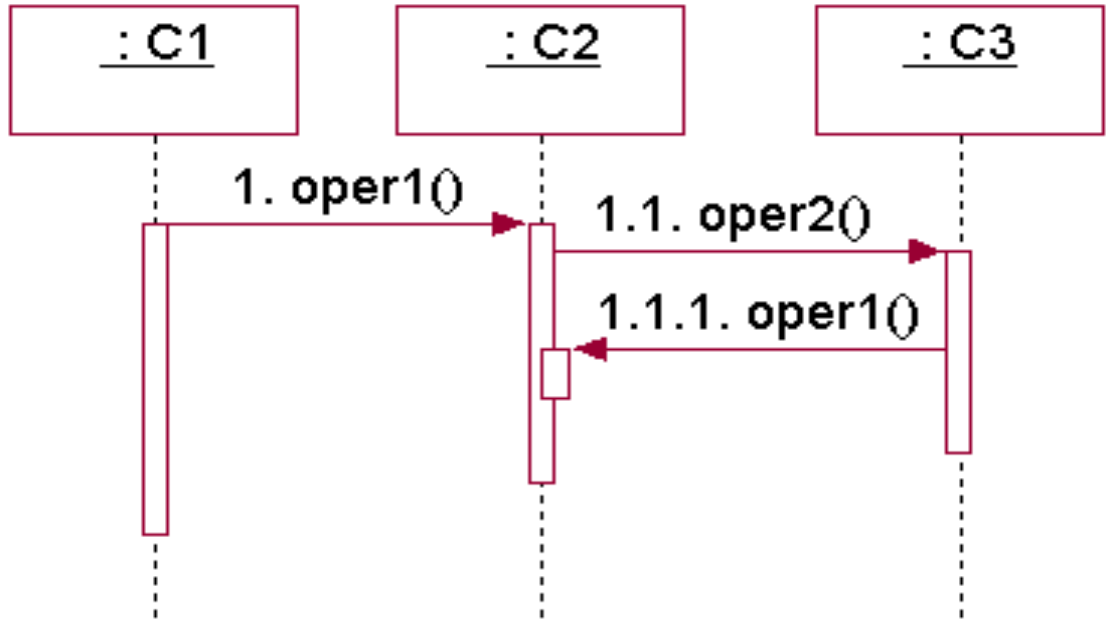
例1. 单个对象自身的递归。



顺序图常见问题分析

- 顺序图中递归的表示
  - 利用嵌套的FOC表示

例2. 多个对象间相互递归调用的表示。



顺序图的作用

- 帮助分析人员对照检查用例中描述需求，是否已经落实给具体对象去实现
- 提醒分析人员去补充遗漏的对象类或操作
- 帮助分析人员识别哪些对象是主动对象
- 通过对一个特定的对象群体的动态方面建模，深入地理解对象之间的交互