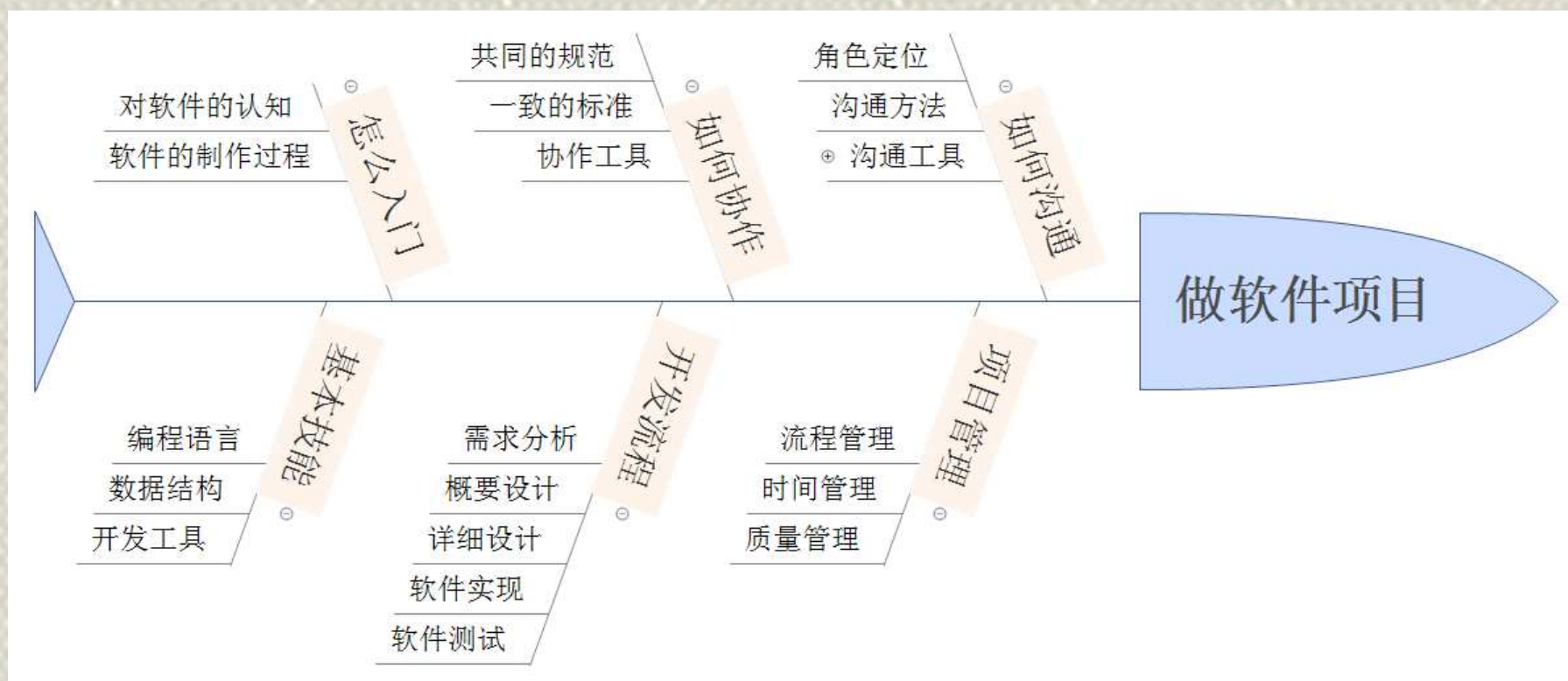


Ch. 5 项目与项目进程

2018 / 5

回顾



目录

- § 5.1 认识项目
- § 5.2 需求分析
- § 5.3 设计与模块化
- § 5.4 概要设计
- § 5.5 详细设计

认识项目

- 什么是项目
 - 项目 → **project** → 项目,工程; 计划,规划;

Project

From Wikipedia, the free encyclopedia

For the urban low-income housing buildings called projects, see [Public housing](#). For other uses, see [Wikipedia Projects](#). For [:Wikipedia:WikiProject](#), see [Project \(disambiguation\)](#).

In contemporary [business](#) and [science](#), a **project** is an individual or collaborative enterprise, possibly involving research or design, that is carefully [planned](#), usually by the project assigned team, to achieve a particular aim.^[1]

One can also define a project as a set of interrelated tasks to be executed over a fixed period and within certain cost and other limitations.^[2]

One can view projects as temporary (rather than permanent) [social systems](#) or as [work systems](#) that are constituted by [teams](#) within or across organizations to accomplish particular [tasks](#) under time constraints.^[3] An ongoing project is usually^[*quantify*] called (or evolves into) a [program](#).

项目

联合国工业发展组织《工业项目评估手册》对项目的定义是：“一个项目是对一项投资的一个提案，用来创建、扩建或发展某些工厂企业，以便在一定周期内增加货物的生产或社会的服务。

项目管理协会（Project Management Institute, PMI）认为：项目是为完成某一独特的产品或服务所做的一次性努力。

中国项目管理知识体系纲要（2002版）中对项目的定义为：项目是创造独特产品、服务或其他成果的一次性工作任务。

《项目管理质量指南(ISO10006)》定义项目为：“具有独特的过程，有开始和结束日期，由一系列相互协调和受控的活动组成。过程的实施是为了达到规定的目标，包括满足时间、费用和资源等约束条件”。

- 项目是在一定约束条件下（主要是限定资源、限定时间、限定质量），具有特定目标的一次性任务

项目的一些特点

- 一次性
- 有明确的目标
- 有限定条件
- 计划性

项目的目标

在限定的条件下完成一个产品：

一款商业保险

一款新的冰淇淋

一款软件

完成目标的途径

把大象关进冰箱里

- 整理出需求
- 找到解决方案
- 把方案完成

解决方案

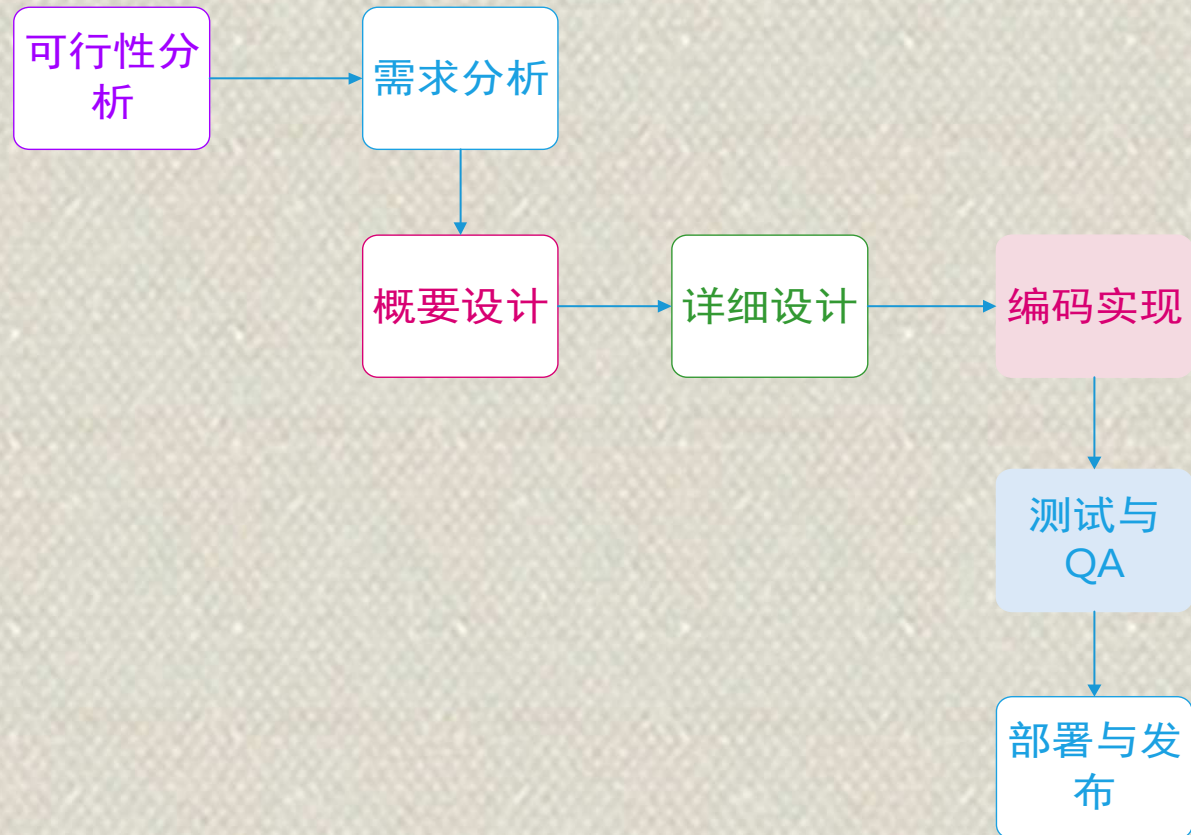
- 是不是软件项目一定要写代码呢？
- 用最低的成本解决问题吗？
- 有现成的“解决方案”，为什么还要做项目呢？

项目来源

- 客户需求
- 专业产品驱动
- 竞争产品刺激
- 作业和毕业设计

软件开发过程

- 需求分析
- 概要设计
- 详细设计
- 编码实现
- 测试/QA
- 部署
- 维护



目录

- § 5.1 认识项目
- § 5.2 需求分析
- § 5.3 设计与模块化
- § 5.4 概要设计
- § 5.5 详细设计

Let us go

- 自行设计一款计算器



- 进制转化
- 表达式求值
- 统计计算器
- 汇率换算计算器
- 税率（年终奖计算器）

一个段子

G20上午开会休息了，马老板对秘书说：中午帮我买肯德基，30分钟后，秘书回来说，买好了，一共4.6亿美元，咱是支付宝还是现金？马云眉头一皱，马上说：赶紧把小王追回来！我刚让他去买中南海了...告诉他：那是烟！是烟！！是烟！！！！



- 根据Standish Group对23000个项目进行的研究结果表明，28%的项目彻底失败，46%的项目超出经费预算或者超出工期，只有约26%的项目获得成功。
- 而在于这些高达74%的不成功项目中，有约60%的失败是源于需求问题。
- 也就是说，有近45%的项目最终因为需求的问题最终导致失败

可行性分析

- 更多市场层面和产品层面的分析
- 产品的定位（卖出去，还是实训项目？）
- 技术可行性与团队的技术实力考量
- 工期是否可控，投入是否可量化
- 竞争产品当前的状况（抄，还是改进？）
- 核心竞争力（技术上，还是产品设计上？）

怎样做需求分析

- 面对面访谈(face-to-face interviewing)
- 专题讨论会(workshop)
- 现场观察(observing on the scene)
- 头脑风暴(brainstorming)

这些和搞技术的有什么关系？

要做什么样的需求分析

- 明确：要做什么？过程中会不会更改？
- 无歧义：不同背景的人，对相同的词汇的理解可能是不同的（a/synchronization）
- 具体：越具体越好
- 合理转化：把“用户的需求”变成“产品需求”再转成“项目的需求”

需求分析的要点

- ✓ 功能性需求(Functional Requirements)
- ✓ 非功能性需求(Non-Functional Requirements)

功能性需求

- ❑ 完备性Completeness：软件能够支持用户所需求的全部功能的能力
- ❑ 正确性Correctness：软件按照需求正确执行任务的能力
- ❑ 健壮性Robustness：在异常情况下，软件能够正常运行的能力（容错能力/恢复能力）
- ❑ 可靠性Reliability：在一定的环境下，在给定的时间内，系统不发生故障的概率，或者是快速从错误状态恢复到正确状态的能力

非功能性需求

- ❑ 性能Performance Quality：时间/空间效率
- ❑ 易用性Usability：习惯性用户体验，特殊群体
- ❑ 清晰性Clarity：易读/易理解，可以提高团队开发效率，降低维护代价
- ❑ 安全性Safety：授权/密码；是否会被注入恶意代码？
- ❑ 可扩展性Expandability：添加一种新的运算是否方便？
- ❑ 兼容性Compatibility：你的计算器运行在什么环境下？你的计算器开了，是不是别的软件就不能用了？如果是页面版的，使用哪种浏览器打开
- ❑ 可移植性Portability：能方便地进行移植吗？网页版的可能更好一些

需求发生变化怎么办？

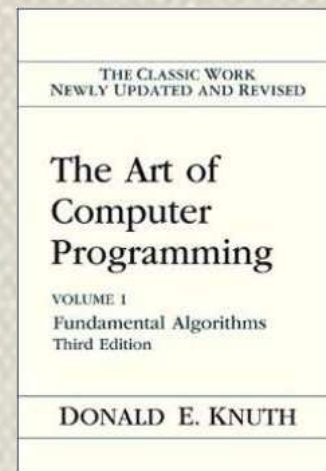
- Catastrophic ? New project : Update all;

目录

- § 5.1 认识项目
- § 5.2 需求分析
- § 5.3 设计与模块化
- § 5.4 概要设计
- § 5.5 详细设计

什么是设计

- 设计：指[设计师](#)有目标有计划的进行技术性的创作与创意活动。
- 设计的任务不只是为生活和商业服务，同时也伴有艺术性的创作。



为什么要做设计？

- 需求过程本身就是设计
- 需求要被整理
- 需求需要转化（什么是模态对话框？）
- 用一些固定的设计更清晰、高效
- 一般性问题和特殊性问题
- 项目总是有些特殊问题，特殊问题怎么解决？

都设计什么？

结构设计：定义软件系统的整体结构，

数据设计：数据结构、数据库、文件的定义。

过程设计：把结构成份（模块）转换成软件的过程性描述

UI/UE设计：是对系统边界的描述，是用户和系统进行交互的工具。

是不是一定要设计？

- 某些问题其实可以不必进行设计
- 根据经验动手先做，经验来源
- 不断优化
- 其实是有设计过程的

设计会不会发生变化？

- 确定后不再进行变化
- 一直在变化，不稳定的需求
- 需要本身要求具有较强的扩展性
- 潜在的变化

软件的设计思路

- 自顶向下与自底向上
- 模块化的设计思路

模块化思维

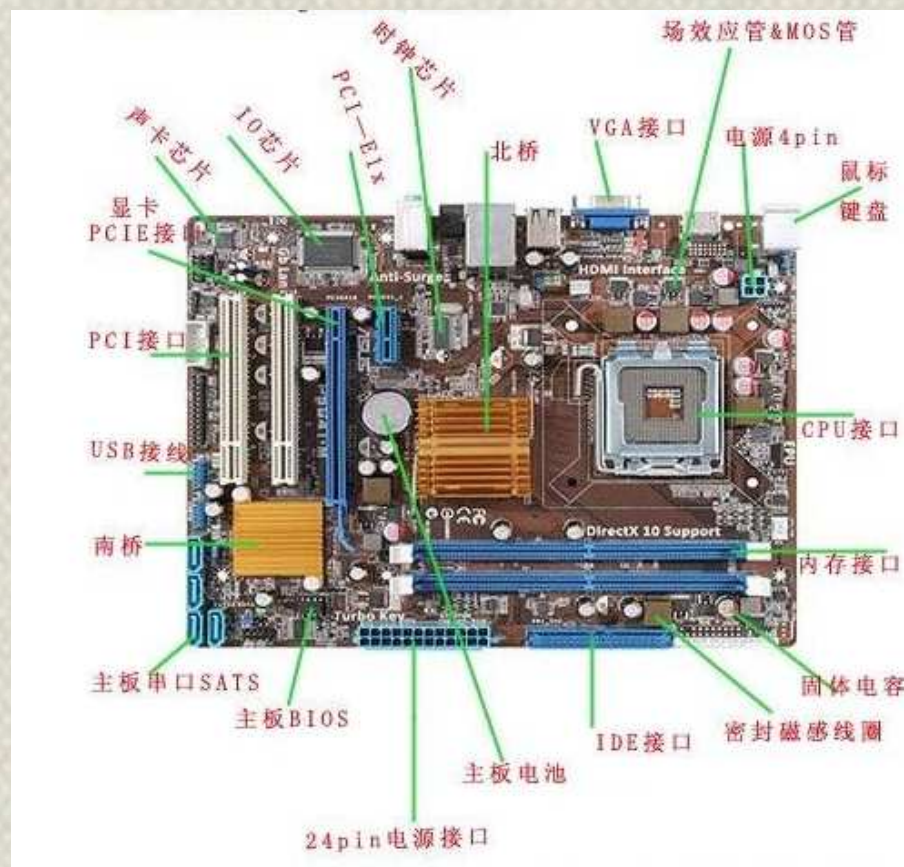
- 整体化思维与模块化思维



模块化



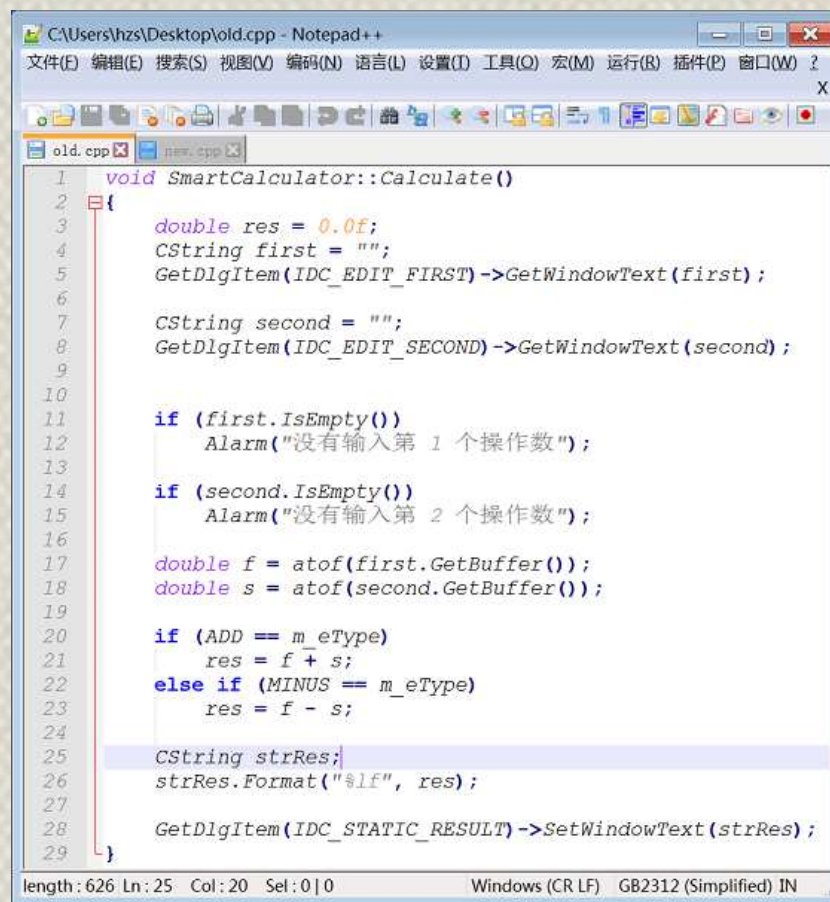
模块化



软件模块化方法

- 函数（过程）
- 面向对象的封装
- 源码包
- 静态库、动态库
- 组件

未经处理



```
1 void SmartCalculator::Calculate()
2 {
3     double res = 0.0f;
4     CString first = "";
5     GetDlgItem(IDC_EDIT_FIRST)->GetWindowText(first);
6
7     CString second = "";
8     GetDlgItem(IDC_EDIT_SECOND)->GetWindowText(second);
9
10
11     if (first.IsEmpty())
12         Alarm("没有输入第 1 个操作数");
13
14     if (second.IsEmpty())
15         Alarm("没有输入第 2 个操作数");
16
17     double f = atof(first.GetBuffer());
18     double s = atof(second.GetBuffer());
19
20     if (ADD == m_eType)
21         res = f + s;
22     else if (MINUS == m_eType)
23         res = f - s;
24
25     CString strRes;
26     strRes.Format("%lf", res);
27
28     GetDlgItem(IDC_STATIC_RESULT)->SetWindowText(strRes);
29 }
```

length: 626 Ln: 25 Col: 20 Sel: 0|0 Windows (CR LF) GB2312 (Simplified) IN

函数封装

```
1 void SmartCalculator::Calculate()  
2 {  
3     double res = 0.0f;  
4     double f = this->getFirst();  
5     double s = this->getSecond();  
6  
7     if (ADD == m_eType)  
8         res = f + s;  
9     else if (MINUS == m_eType)  
10        res = f - s;  
11  
12    out(res);  
13 }
```

```
15 double SmartCalculator::getFirst()  
16 {  
17     CString first = "";  
18     GetDlgItem(IDC_EDIT_FIRST)->GetWindowText(first);  
19     if (first.IsEmpty())  
20         Alarm("没有输入第 1 个操作数");  
21     return atof(first.GetBuffer());  
22 }
```

```
24 double SmartCalculator::getSecond()  
25 {  
26     CString second = "";  
27     GetDlgItem(IDC_EDIT_SECOND)->GetWindowText(second);  
28     if (second.IsEmpty())  
29         Alarm("没有输入第 2 个操作数");  
30     return atof(second.GetBuffer());  
31 }
```

```
32 void out(double res)  
33 {  
34     CString strRes;  
35     strRes.Format("%lf", res);  
36     GetDlgItem(IDC_STATIC_RESULT)->SetWindowText(strRes);  
37 }
```


多态设计

```
Calculator.java Operation.java Add.java Minus.java
1
2
3 public class Calculator
4 {
5     static Operation op = null;
6
7     public static void main(String[] args)
8     {
9         double f = 100.0f;
10        double s = 20.0f;
11
12        double value = op.calculate(f, s);
13
14        System.out.println(value);
15    }
16
17    static void ButtonDownAdd()
18    {
19        op = new Add();
20    }
21
22    static void ButtonDownMinus()
23    {
24        op = new Minus();
25    }
26 }
```

```
Calculator.java Operation.java Add.java Minus.java
1 public interface Operation
2 {
3     public double calculate(double f, double s);
4 }
```

```
Calculator.java Operation.java Add.java Minus.java
1 public class Add implements Operation
2 {
3     public Add(){}
4     public double calculate(double f, double s)
5     {
6         return f + s;
7     }
8 }
```

```
Calculator.java Operation.java Add.java Minus.java
1 public class Minus implements Operation
2 {
3     public Minus(){}
4     public double calculate(double f, double s)
5     {
6         return f - s;
7     }
8 }
```

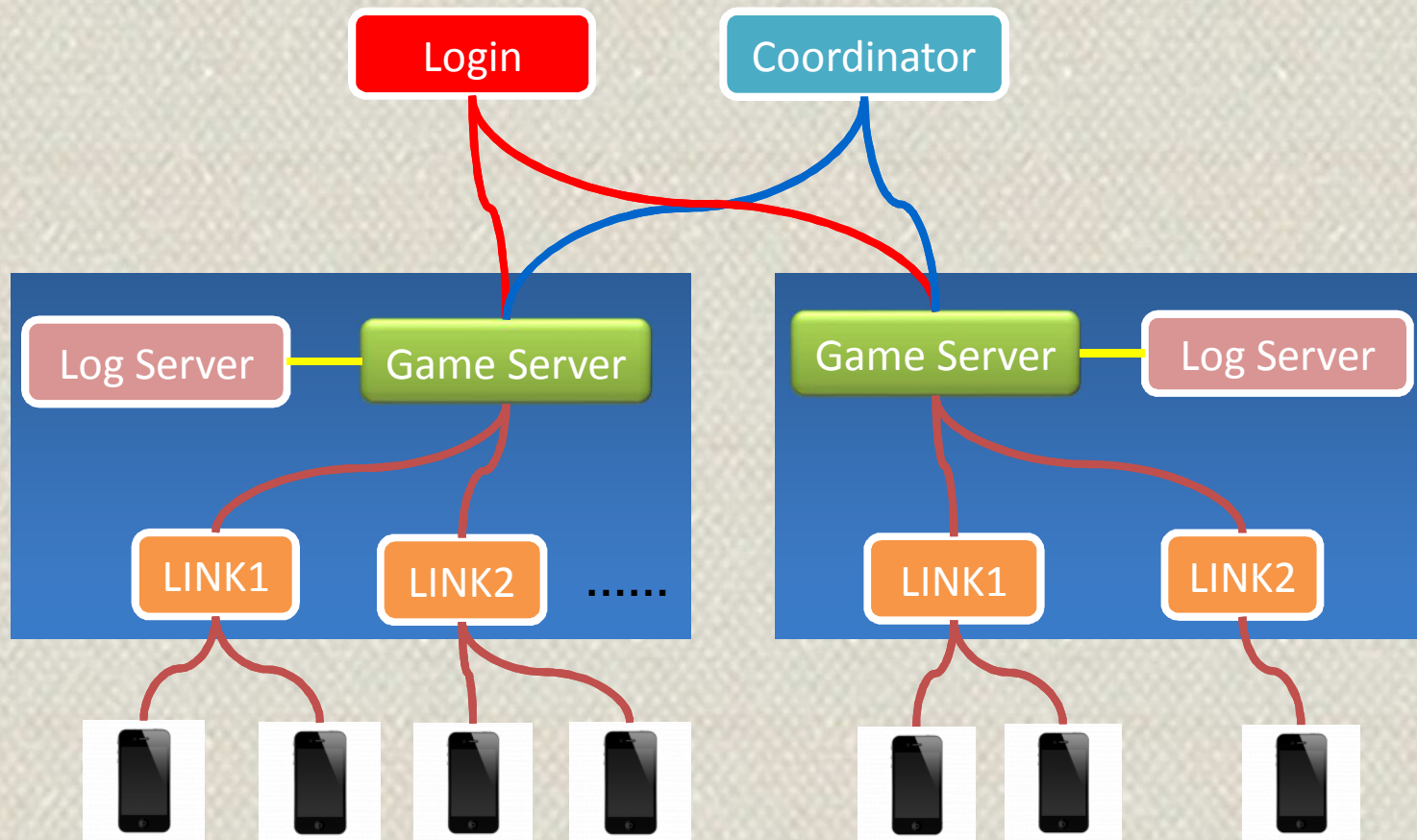
目录

- § 5.1 认识项目
- § 5.2 需求分析
- § 5.3 设计与模块化
- § 5.4 概要设计
- § 5.5 详细设计

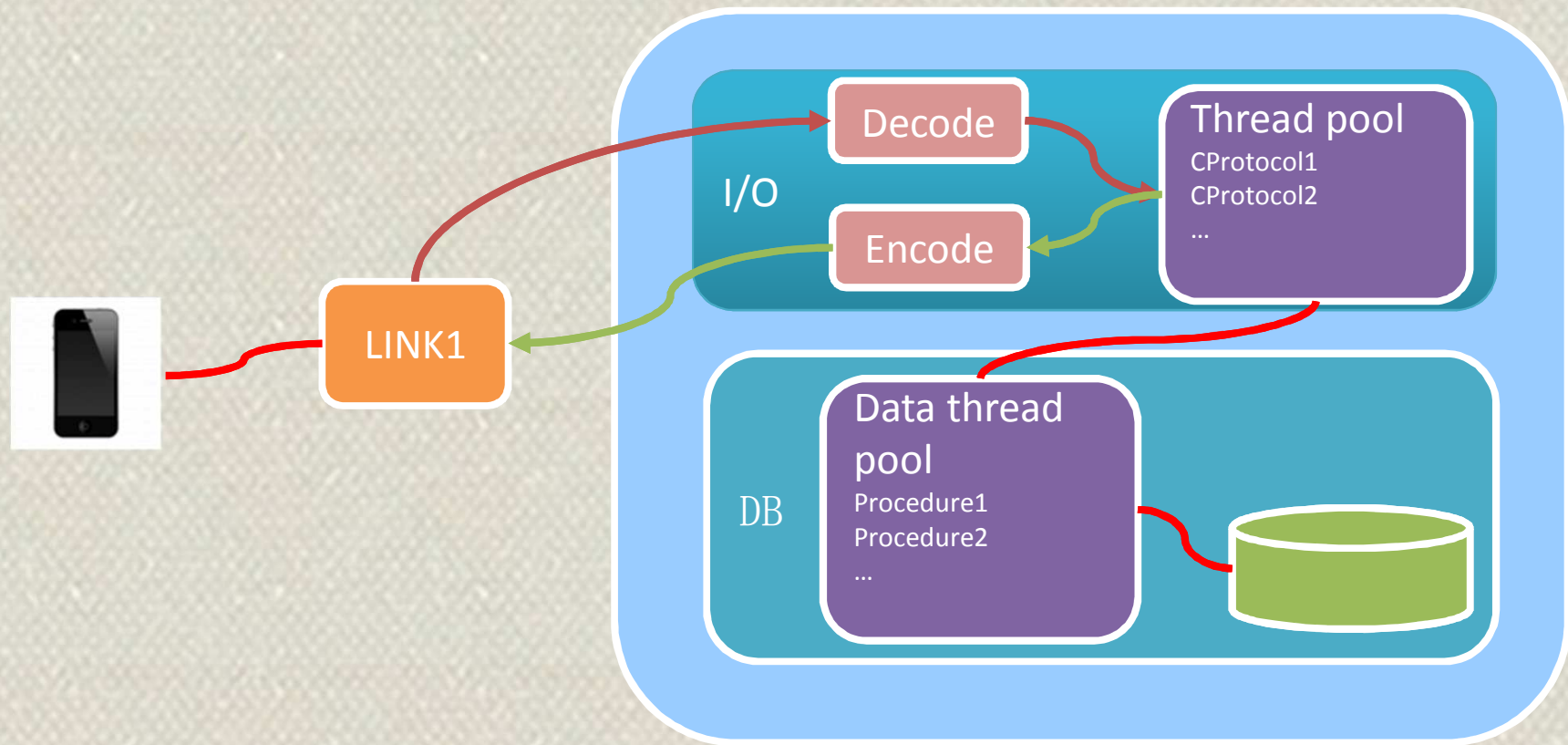
概要设计

- 系统结构
- 子系统
- 大模块
- 模块之间的关联

Architecture Overview



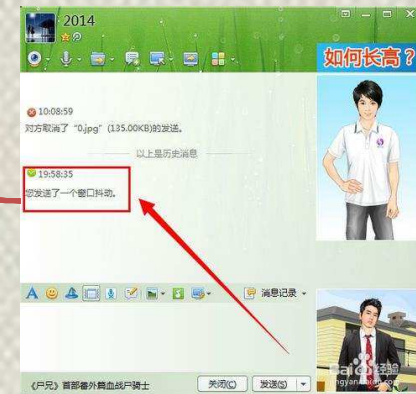
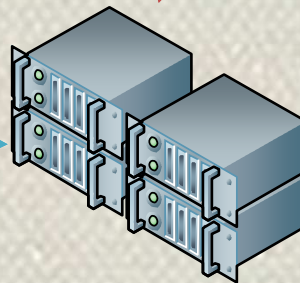
Game Server Architecture



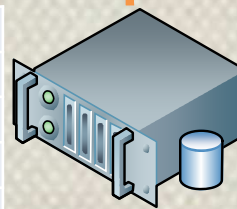
大型软件的架构

- 单机架构(Standalone)
- C/S架构(Client / Server)
- B/S架构(Browser/Server)
- Peer to Peer
- 分布式系统架构(Distributed System)
- 混合架构(Hybrid Architecture)

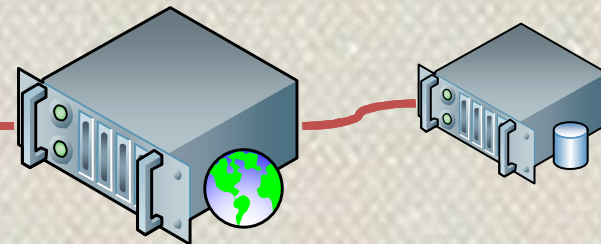
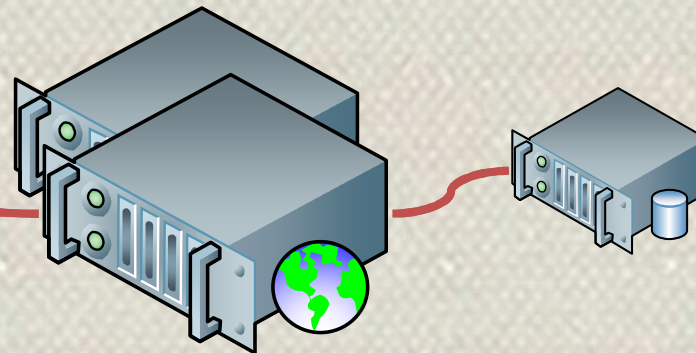
C/S架构



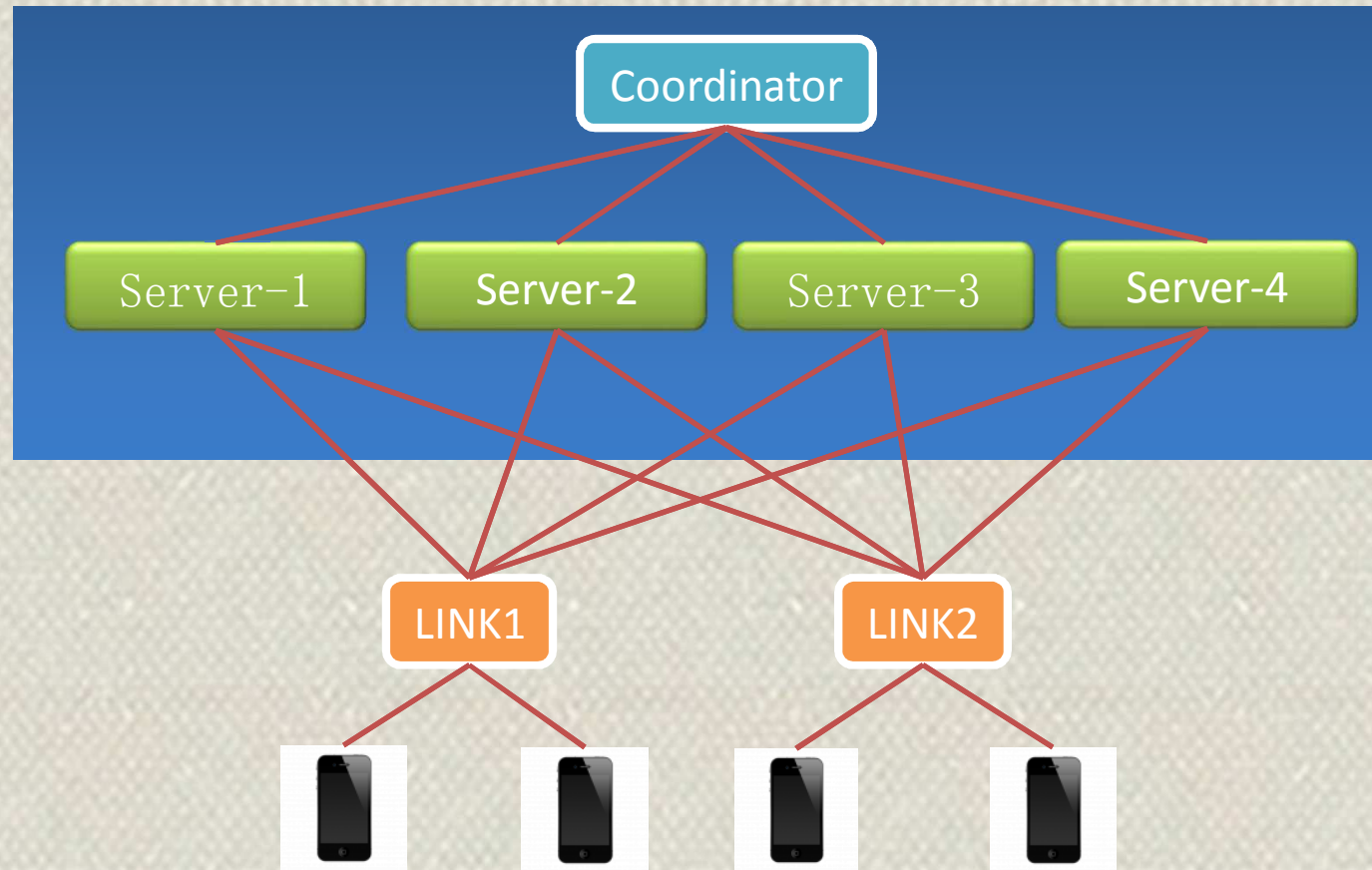
accid	password	name	signature	
1369234	#wulal896siw	快乐的程序员	之所以快乐，	
136954139	Tfit8^bfatIO	小鼠标	人生如戏，戏如人生	
13695414	5ajfit8bfrad	哈哈镜	照鬼不照人	
13695415	gwjit45eblgoa	蜜桃	你胖你先吃	
13695416	weafadbffwa	开心鬼 ^_^		
13695417	56esdt8OTe	秀个够够		



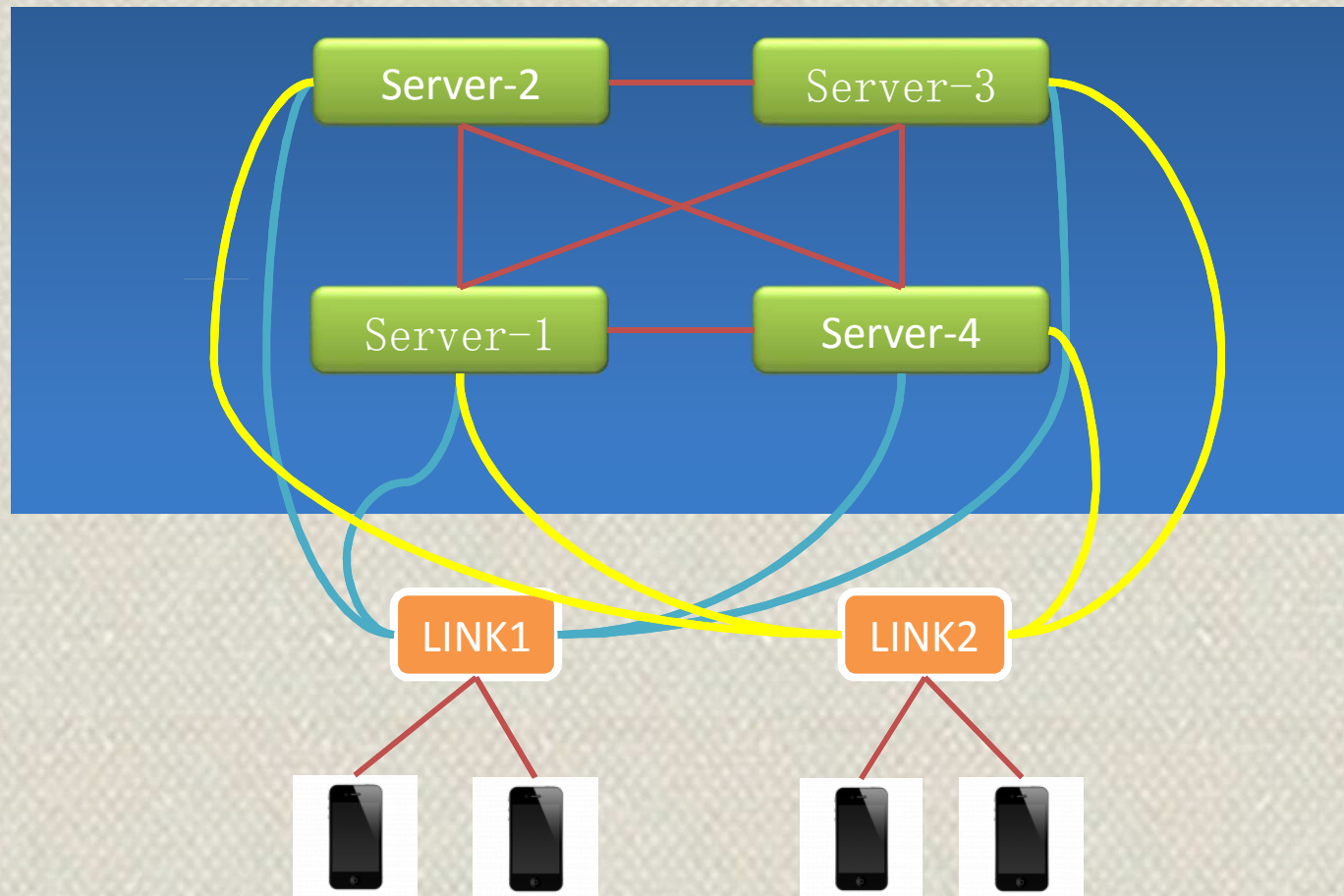
B/S架构



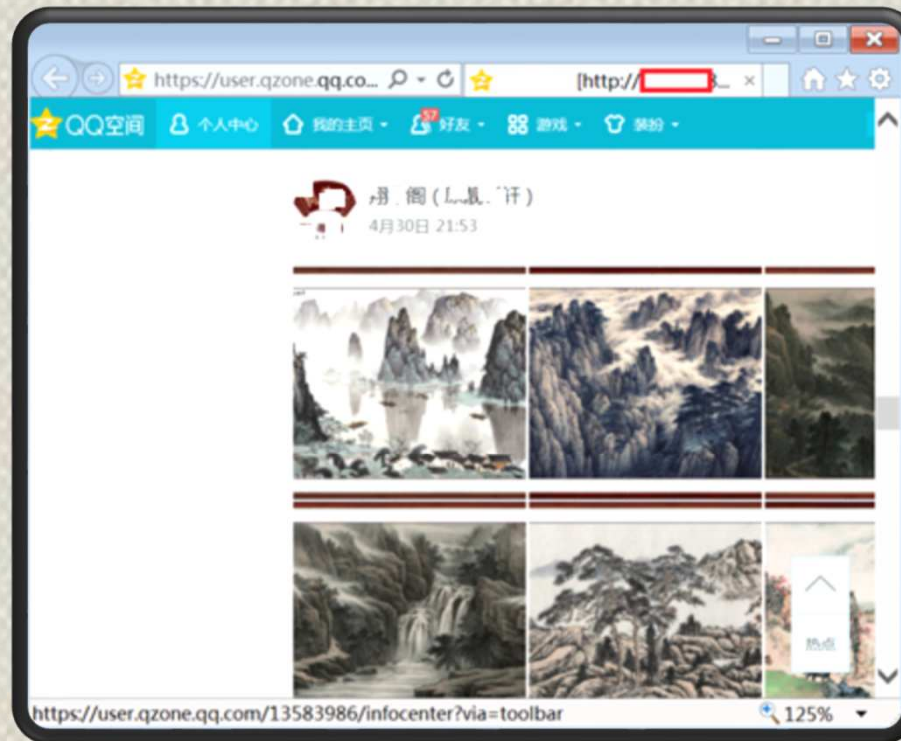
分布式系统



分布式系统



混合架构



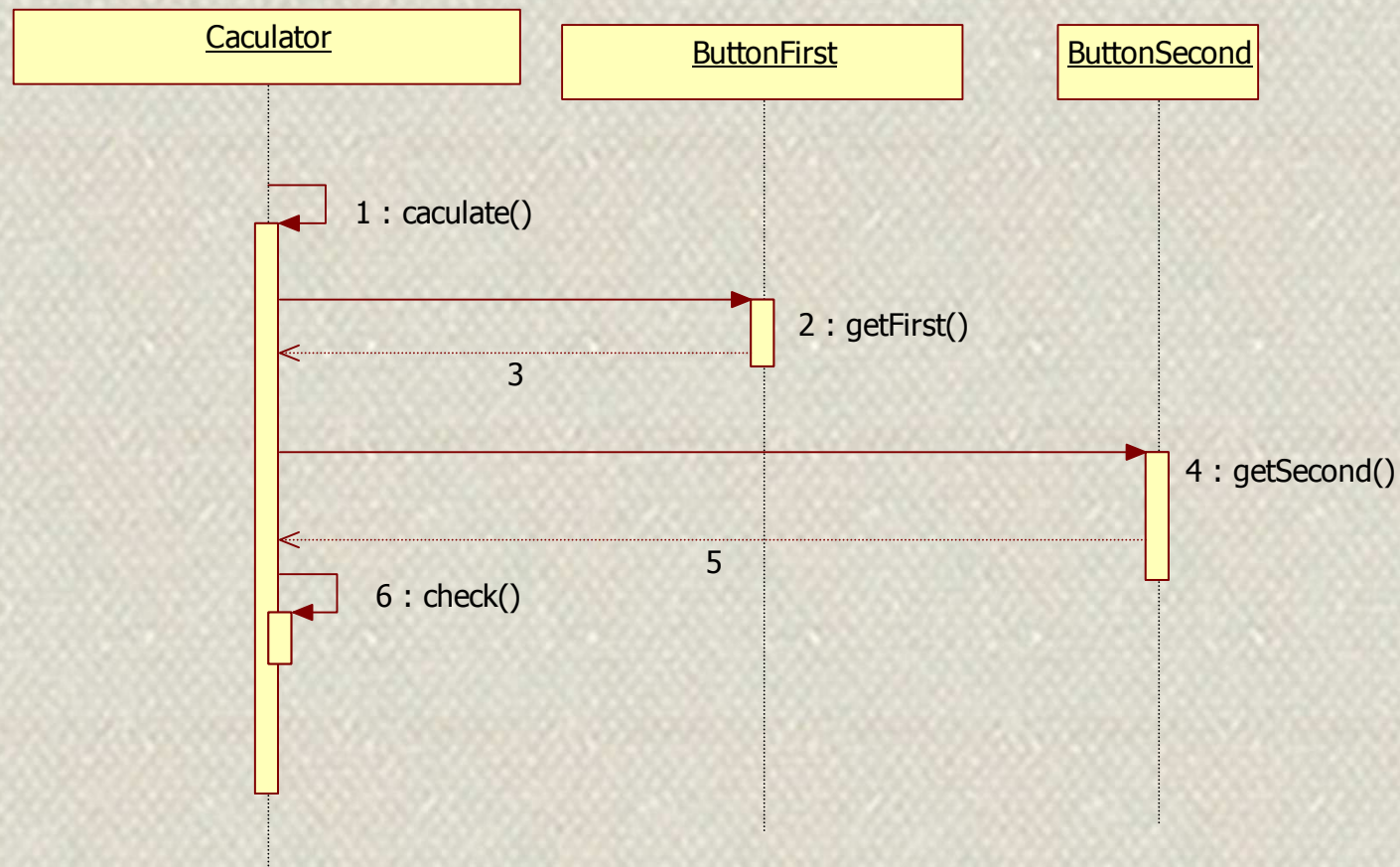
目录

- § 5.1 认识项目
- § 5.2 需求分析
- § 5.3 设计与模块化
- § 5.4 概要设计
- § 5.5 详细设计

详细设计

- 流程图
- 交互图（时序图、状态图、活动图.....）
- PAD（Problem Analysis Diagram）
- 伪代码
-

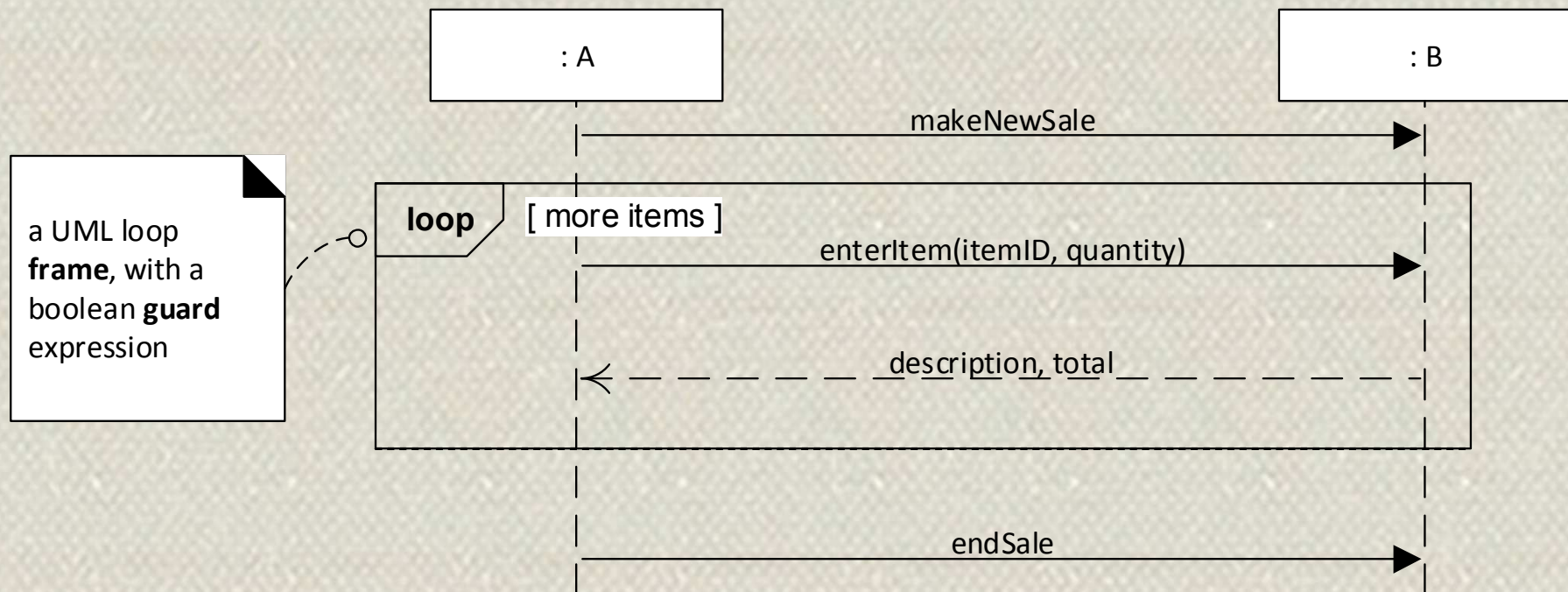
时序图



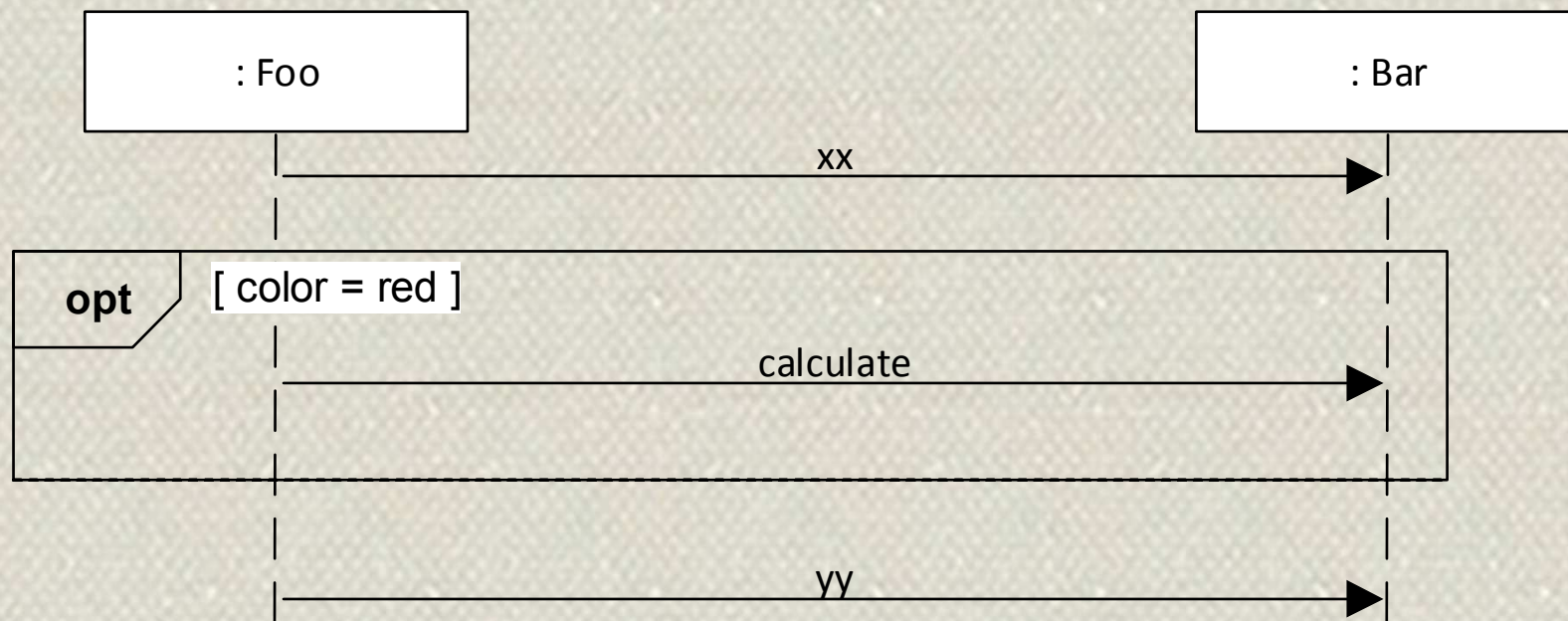
时序图

- 显示用例的行为顺序
- 显示了用例流程中不同对象之间的调用关系
- 对象、类和参与者都在时序图中进行描述

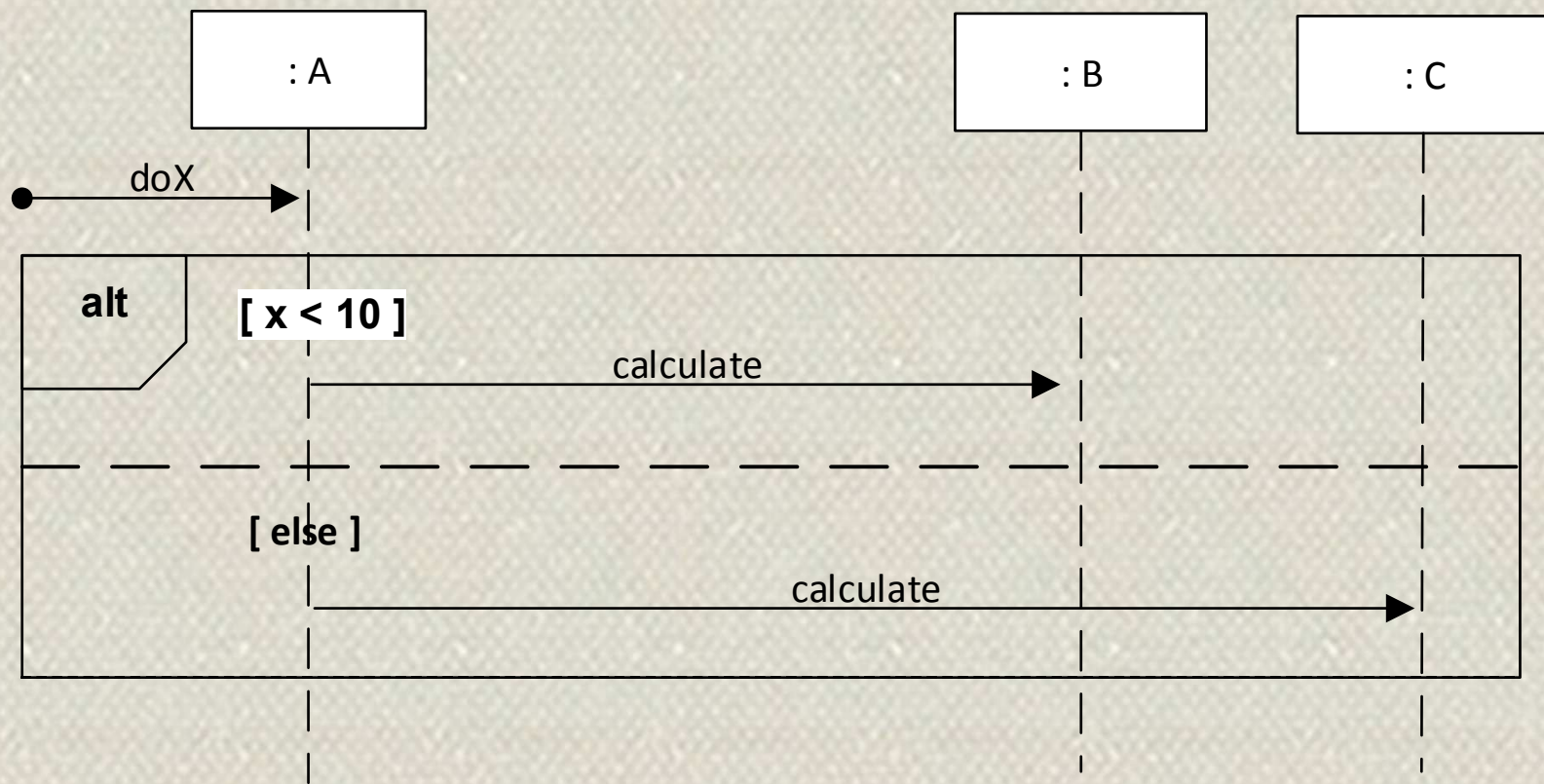
帧 (frame)



条件帧



互斥帧



伪代码Pseudocode

- $x \leftarrow y$
- $x \leftarrow 20 * (y + 1)$
- $x \leftarrow y \leftarrow 30$
- $x = y;$
- $x = 20 * (y + 1);$
- $x = y = 30;$

还记得敏捷吗？

- 有一种敏捷叫直接写代码！