软件工程专业导论

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

第5讲 软件系统构造-对象-组件与软件框架

战德臣

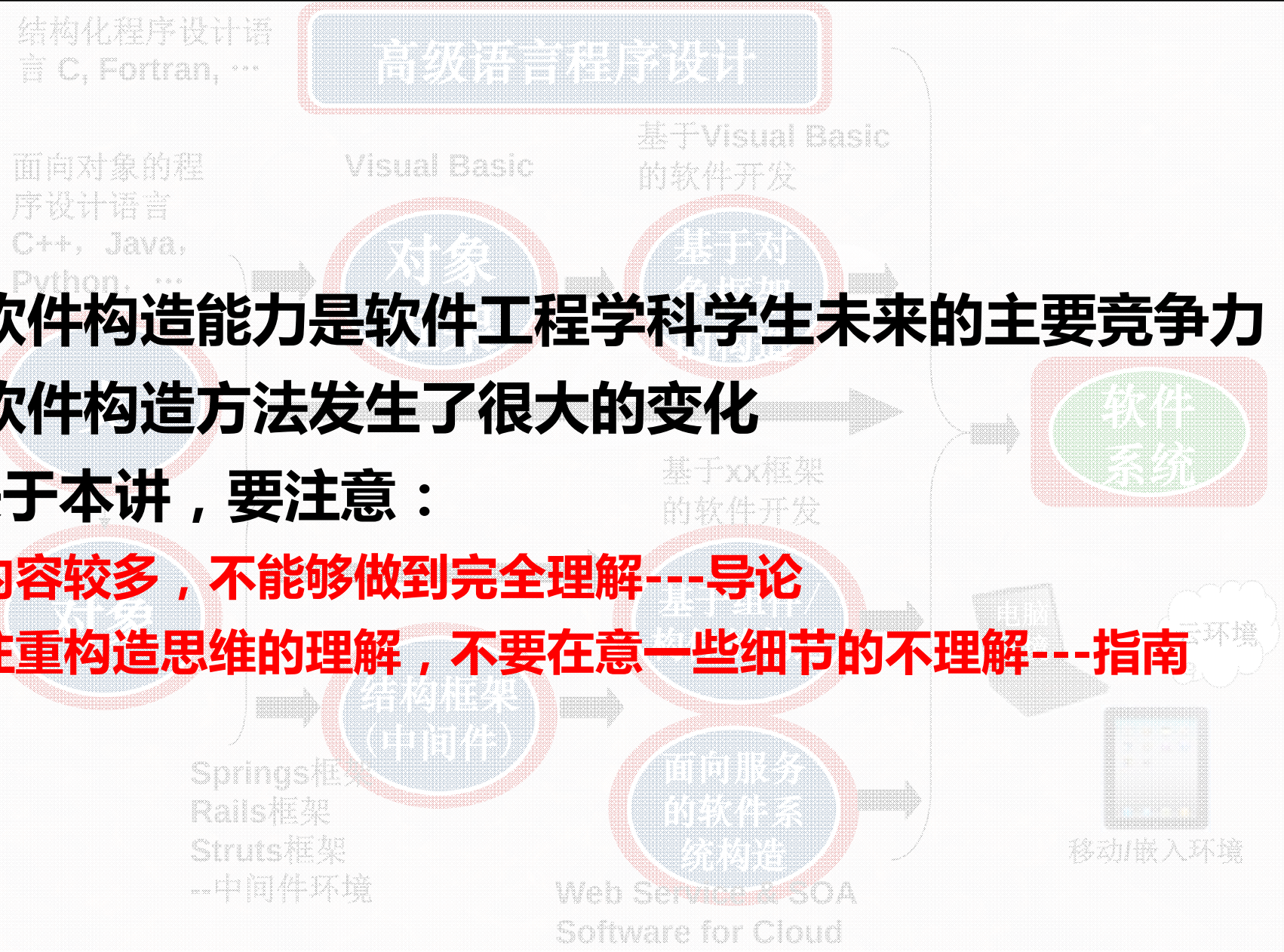
哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

本讲内容概览与学习方法

战德臣 教授



- **软件构造能力是软件工程学科学生未来的主要竞争力**
- **软件构造方法发生了很大的变化**

关于本讲，要注意：

- **内容较多，不能够做到完全理解---导论**
- **注重构造思维的理解，不要在意一些细节的不理解---指南**

面向对象的基本思想与概念

---类-对象-消息-事件-方法

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

面向对象的基本思想与概念--类-对象-消息-事件-方法
(1)为什么要理解面向对象的思维

战德臣 教授

- 面向对象的思维是构造复杂系统的基本思维模式
- 基于面向对象思维的程序设计语言，可以构造复杂及特色化的软件
- 面向对象的思维及程序设计语言是软件工程学科人才必须掌握的能力

(1)为什么要理解面向对象的思维

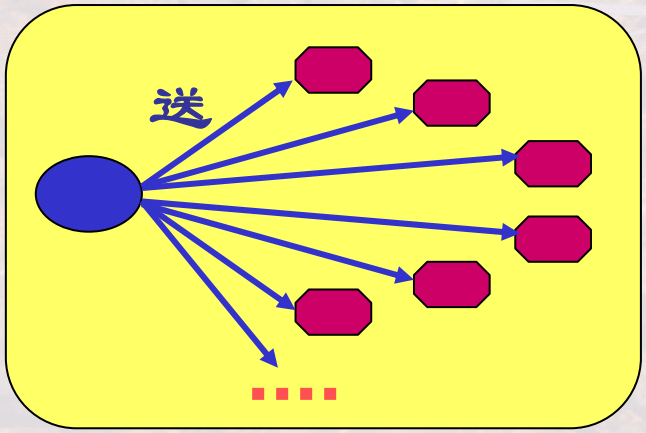
战德臣 教授

“取”与“送”的思维差别?

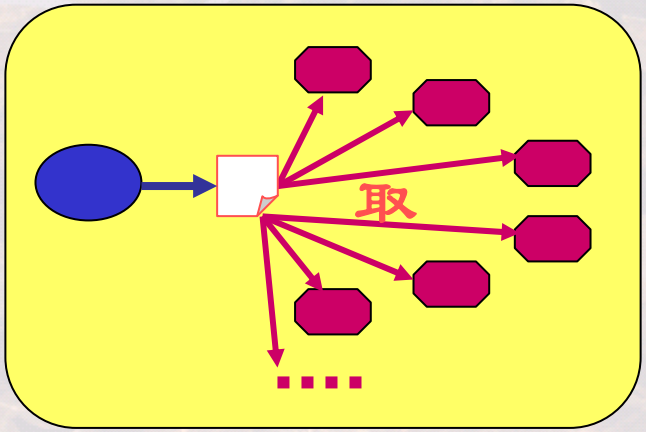
◆如：交通局制定了一个“某道路由双向改为单向”的通知，要让每一司机知道，它是采用送的策略还是采用取的策略呢？

◆“送”，即交通局要把通知送达每个司机的手中。显然，“送”是不可能完成的任务。

◆“取”，即交通局将该通知发布到一个地点，而由每一司机到此地点来获取通知。“取”则可达此目的。



以“过程”为中心



以“对象”为中心

面向对象的基本思想与概念--类-对象-消息-事件-方法
(1)为什么要理解面向对象的思维

战德臣 教授

现实世界(或者说系统)是由可区分的“对象”构成的…

- (从系统来看)对象可被看作是一个个具有某种功能/行为的个体
- **“(从软件系统来看)对象可被看作是一个个可执行某种程序的个体”**

--对象的“独立性/自主性”

- 对象自己执行自身的动作而无需其他对象干预…
- **“对象自己执行自身的程序而独立于其他对象…”**

--对象之间的“交互性”

面向对象的基本思想与概念--类-对象-消息-事件-方法
(2)什么是对象？

战德臣 教授

- “对象”是相互可区分的一个个“个体”
  - 每一对象有唯一的“**对象标识**”区分于其他对象
- “对象”是有状态属性的个体
  - 每一对象有一组状态属性，即相当于一组数据来刻画其自身的状态信息--**对象的属性**
- “对象”是有独立行为或功能的个体
  - 每一对象有自己的可以执行的程序--**对象的函数**(又称为**方法**)
- “对象”在接收到需服务的请求时，可为其他对象提供服务...
  - 每一对象有一组其可以处理的消息—**对象的消息**，而响应“消息”的则是对象的函数(或称方法)。

战德臣 教授

“类”与“对象”

- 有些对象具有相似的特性描述，组成一个“类”。
- 类是对象的抽象或称对象的“类型”，定义了同类型对象的框架(名字、属性和功能);
- 对象是类的实例，是实际运行的个体。

类

类名:	<>
(类的属性)	
性别:	<>
身高:	<>
体重:	<>
(类的功能)	
回答身高():	<>
回答体重():	<>
回答性别():	<>

同类对象的共性形式或者说对象的类型：类

各自独立运行的类的实例：对象

对象

对象名:	张三
(对象的属性)	
性别:	男
身高:	1.80m
体重:	70kg
(对象的功能)	
回答身高():	身高是1.80m
回答体重():	体重是70kg
回答性别():	性别是男

张三

.....

对象名:	王五
(对象的属性)	
性别:	男
身高:	1.64m
体重:	62kg
(对象的功能)	
回答身高():	身高是1.64
回答体重():	体重是62kg
回答性别():	性别是男

王五

战德臣 教授

■“对象”可隐藏其内部的各种细节

■对象可将其内部的属性和函数进行隐藏，即所谓的“封装”

■对象可以将内部复杂的数据结构和函数封装起来，对外只通过消息进行交互。---这非常重要

一车苹果。一箱苹果。一个苹果

对象B. 对象A. 常量
对象B. 对象A. 变量
对象B. 对象A. 函数();
对象B. 函数();

苹果
-- 数据：常量-变量

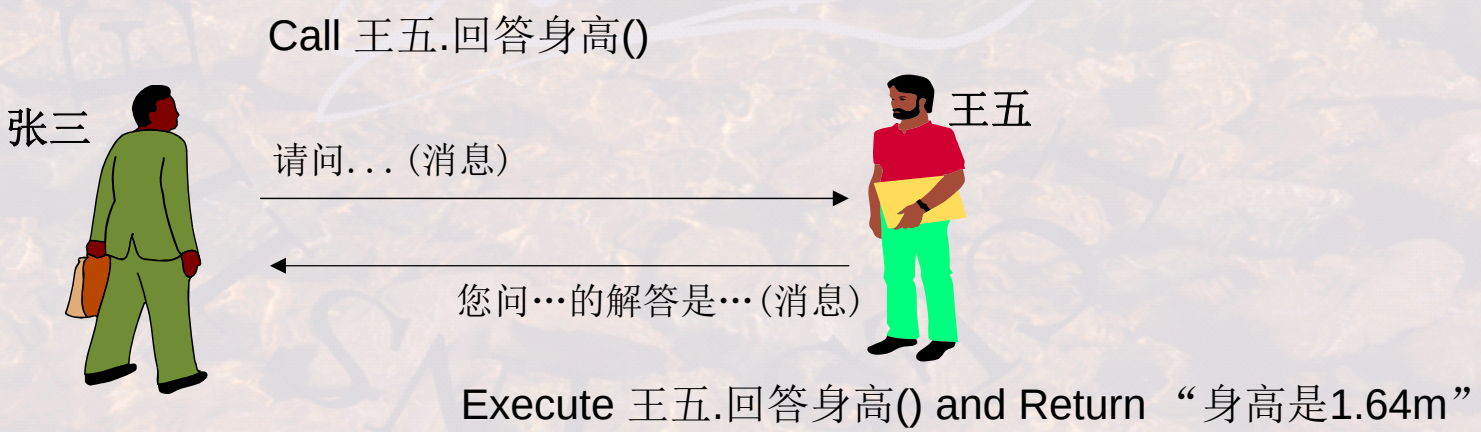
一箱苹果 ---- “对象A”
--对象 A的函数
•装入一个苹果，取出一个苹果
•移动一箱苹果

一车苹果---- “对象B”
--对象B的函数
•行使，暂停
•装载，卸载

面向对象的基本思想与概念--类-对象-消息-事件-方法
(5)对象之间的交互?

战德臣 教授

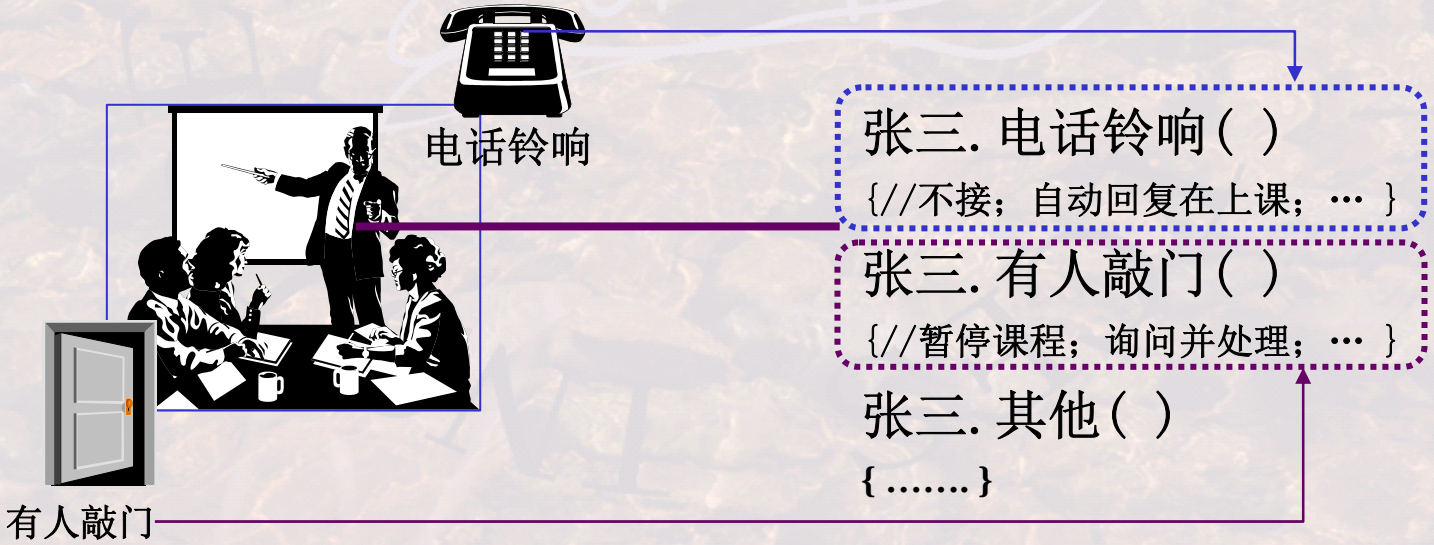
- **消息**是对象之间传递的内容，如指示、打听、请求 … …
- 消息是对象之间交互的唯一的内容
- 消息是 “对象.函数()”的调用和执行



(5)对象之间的交互?

ICES 战德臣 教授

●事件是外界产生的一种能够激活对象功能的特殊消息。当发生事件后将给所涉及对象发送一个消息，对象便可执行相应的功能---事件驱动；



面向对象的基本思想与概念--类-对象-消息-事件-方法
(6)对象的其他特性?

战德臣 教授

- 继承：一个对象(类)继承另一个对象(类)的属性与函数
- 多态：不同对象对同一消息的不同响应，即允许方法重名。有：虚函数型多态—子类可以重定义父类的函数；重载型多态--同一函数名的函数参数可以有不同的形式。
- … …
- 注意：这些特性不急于在本门课程中理解，本门课程要理解这种面向对象的思维…
- 学习面向对象的思维不仅仅是理解这些概念，更重要的是运用其进行构造…

计算机语言如何实现面向对象的思维呢?

面向对象构造的表达方法I：
面向对象程序设计语言

战德臣

哈尔滨工业大学　教授.博士生导师
教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

战德臣 教授

(传统的、面向过程的)计算机语言(程序)的基本构成要素

程序：变量与常量、表达式、语句与函数。

```
main()
{
  int D1,  D2;
  …
  f1();
  …
}

Int f1()  //独立的函数
{
  …  //f1()的程序语句集合
}
```

传统程序

主函数main()：主程序

函数调用：调用一个函数完成一个计算

函数定义：定义一个新函数，
即自己编写的程序段落

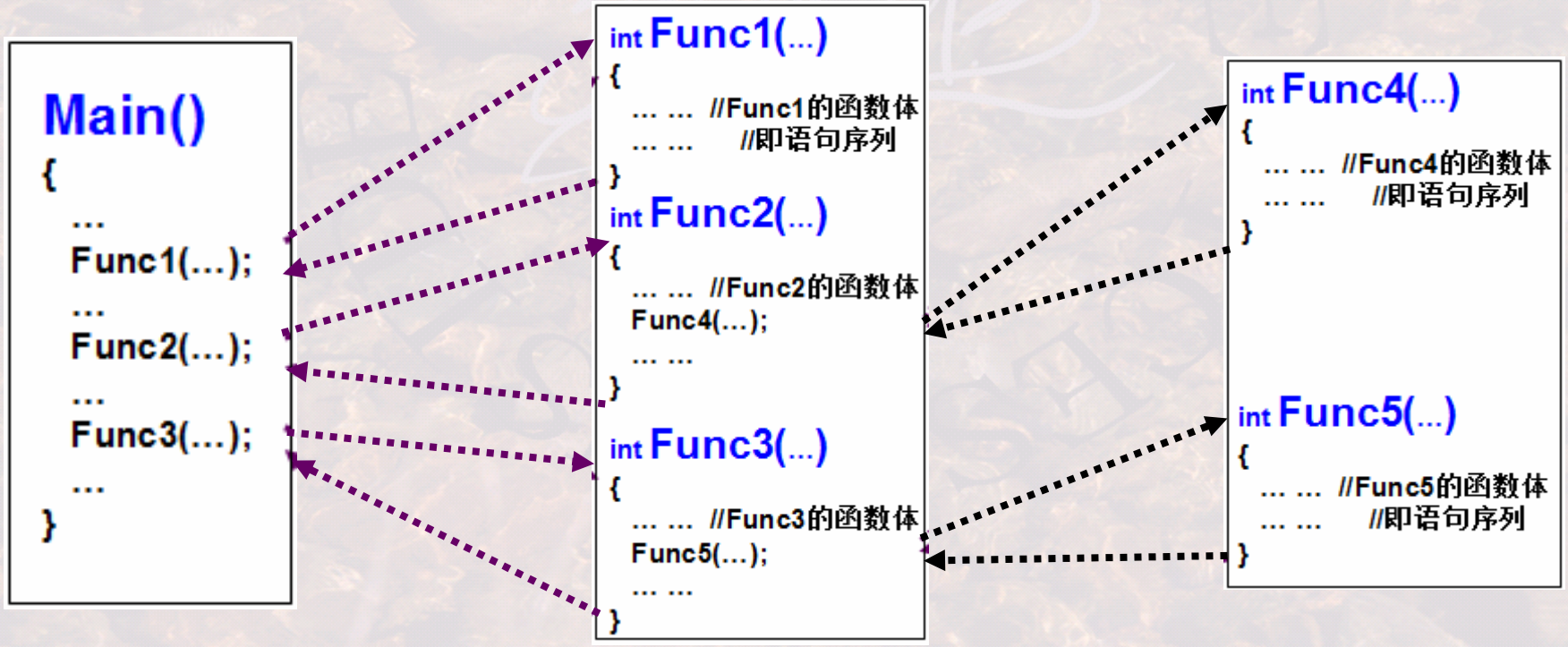
面向对象构造的表达方法I：面向对象程序设计语言
(1)面向过程的计算机语言 vs. 面向对象的计算机语言?

战德臣 教授

传统程序构造及其表达方法----由粗到细

为控制复杂性，先以函数来代替琐碎的细节，着重考虑函数之间的关系，以及如何解决问题

在前一阶段考虑清楚后或编制完成后，再编写其中的每一个函数。而函数的处理同样采取这种思路



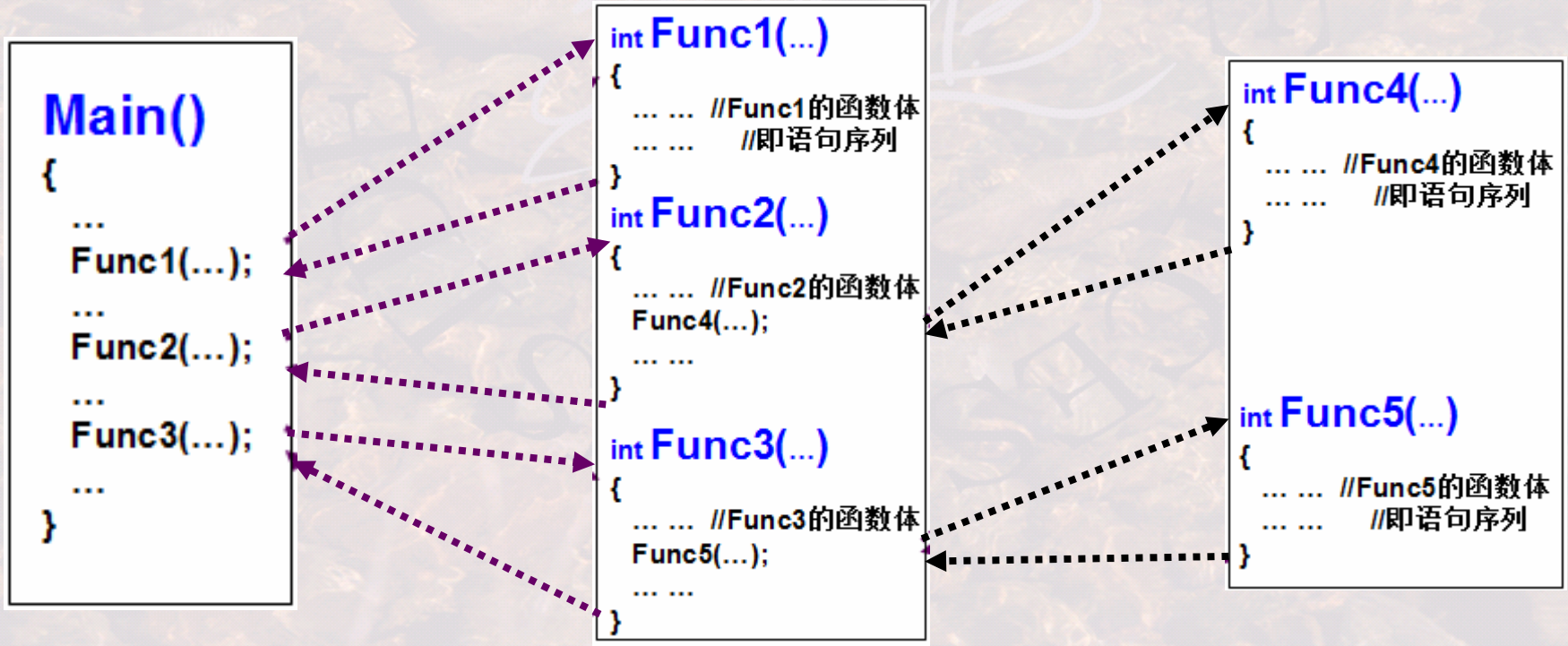
面向对象构造的表达方法I：面向对象程序设计语言
(1)面向过程的计算机语言 vs. 面向对象的计算机语言?

战德臣 教授

传统程序构造及其表达方法----也可以由细到粗

上一层次的函数依据下层函数来编写，确认正确后再转至更上层问题处理

首先编写一些基础性的函数，并确定其正确后，再处理上一层次的问题。



战德臣 教授

(传统的、面向过程的)计算机语言(程序)的基本构成要素

程序：变量与常量、表达式、语句与函数。

```
main()
{
  int D1,  D2;
  ...
  f1();
  ...
}

Int f1()  //独立的函数
{
  ...  //f1()的程序语句集合
}
```

传统程序

主函数main()：主程序

函数调用：调用一个函数完成一个计算

函数定义：定义一个新函数，
即自己编写的程序段落

程序 是 若干函数的集合；函数 是 若干语句的集合，是具有先后次序的若干语句的集合。

ICES 战德臣 教授

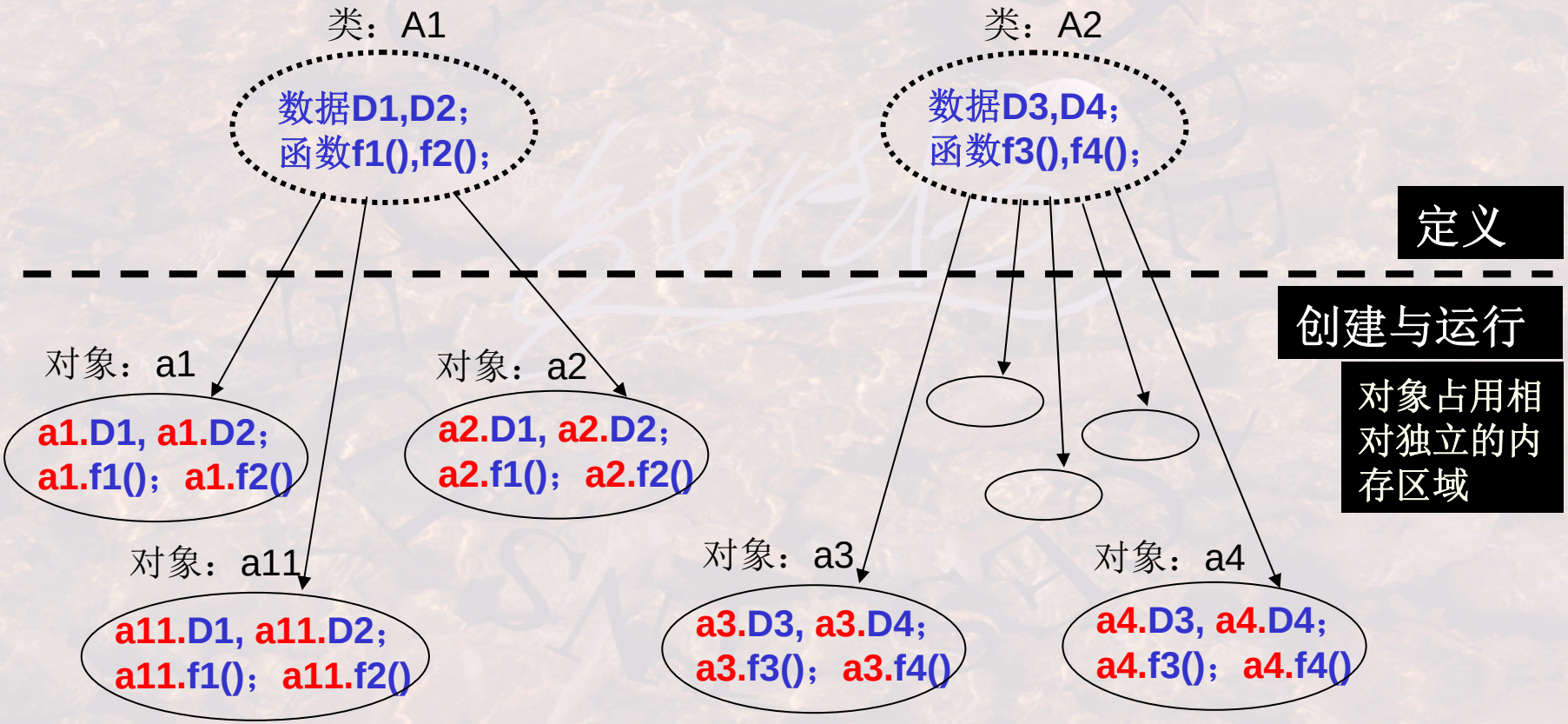
(现代的、面向对象的)计算机语言(程序)的基本构成要素

类与对象，消息与事件，函数或称方法

面向对象构造的表达方法I：面向对象程序设计语言
(2)对象和类要解决什么问题？

战德臣 教授

怎样实现“若干同类别对象，虽有相同的程序，但却处理不同的数据而产生不同的结果呢?”
两方面技术：一是抽象与封装，二是对象的自动产生与运行。

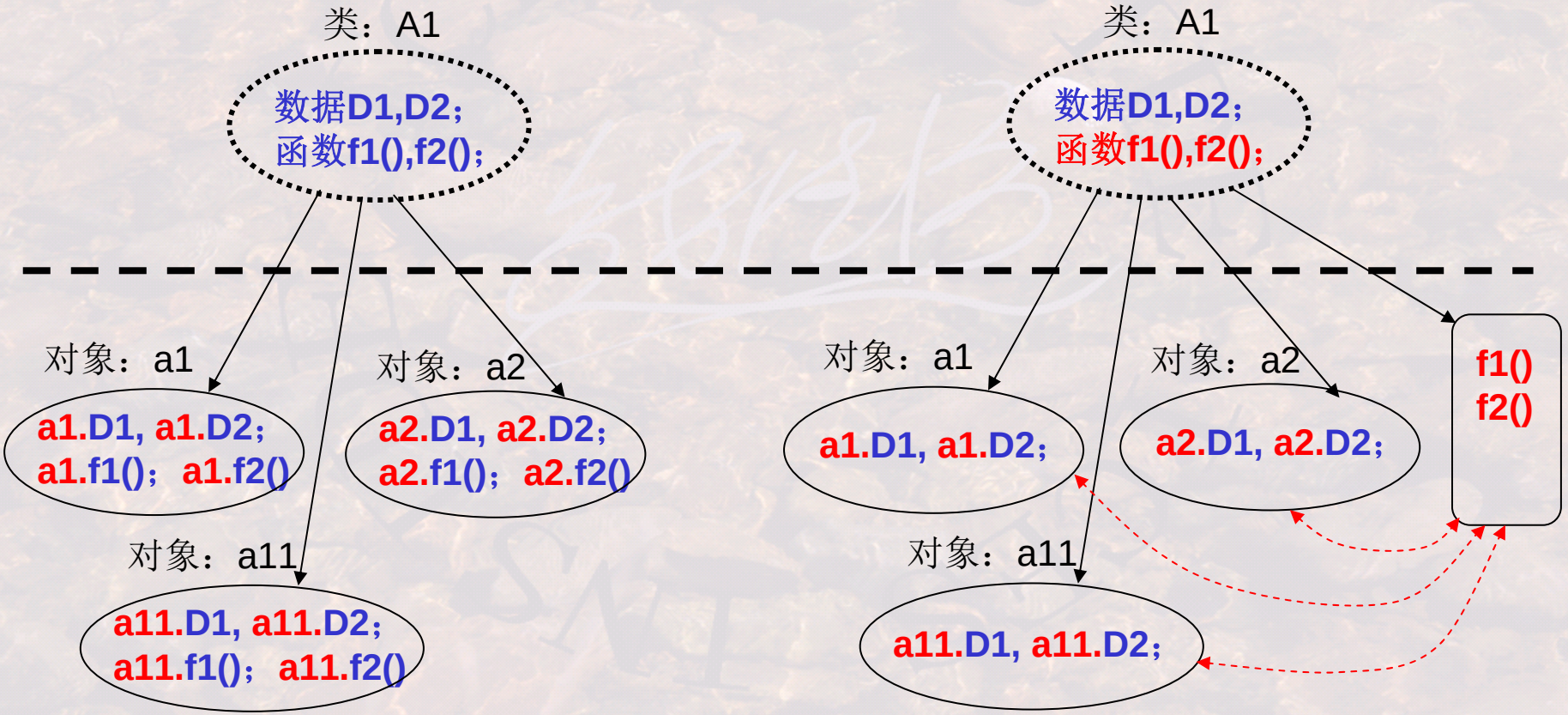


数据结构封装了若干个变量；函数封装了若干行的语句；
类/对象封装了若干数据结构和函数。

面向对象构造的表达方法I：面向对象程序设计语言
(2)对象和类要解决什么问题？

战德臣 教授

“若干同类别对象，虽有相同的程序，但却处理不同的数据而产生不同的结果呢？”
两方面技术：一是抽象与封装，二是对象的自动产生与运行。



(左侧图)—应该实现的，具体语言的物理实现方案之1；(右侧图)—具体语言的物理实现方案之2

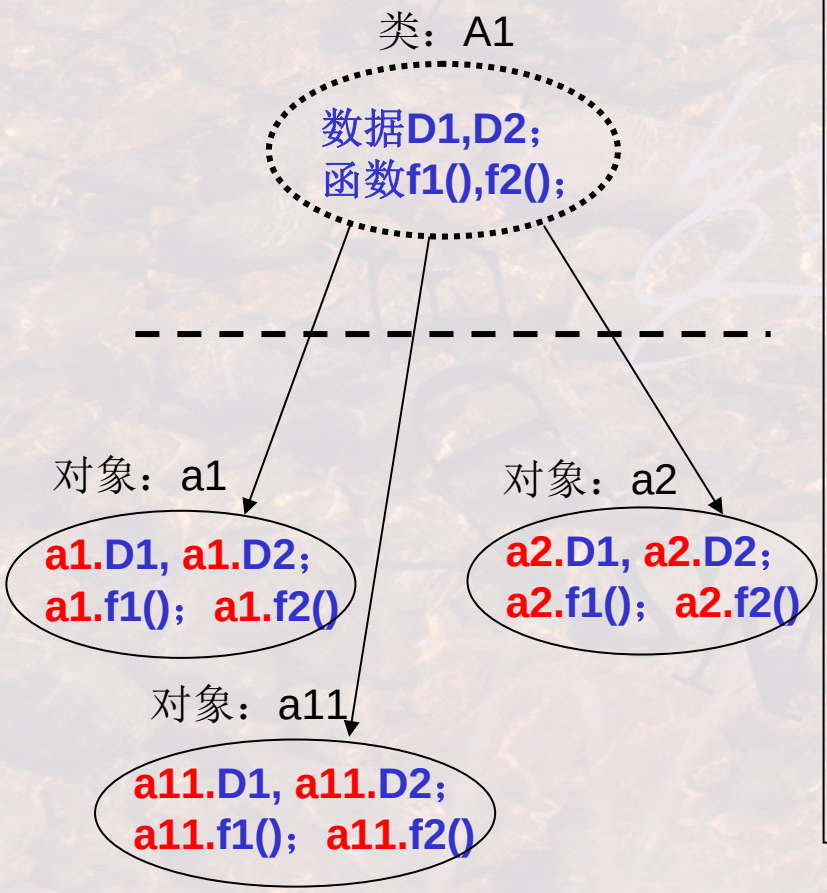
战德臣 教授

对象结构的定义---即类的定义


```
Class  A1                  //定义一个类
{
    int  D1, D2            //定义变量
    int  f1();             //定义类中的函数
    int  f2();             //定义类中的函数

    int  f1()              //类A1的函数f1的程序
    {  //f1的程序语句块  }

    int f2()               //类A1的函数f2的程序
    {   //f2的程序语句块1
        a4 = new  A2;     //创建类A2的对象a4
       //f2的程序语句块2
       D1 = call  a4.f3(); //调用另一对象的某一函数
       //f2的程序语句块3
    }
}
```

战德臣 教授

对象的创建与运行




```
Main()
{  int M1, M2;
   a1 = new A1;        //创建类A1的对象a1
   a2 = new  A1;       //创建类A1的对象a2
   a11 = new A1;       //创建类A1的对象a11

   M1 = call a1.f2();          //执行对象a1的f2的程序
   M2 = call a2.f1();          //执行对象a2的f1的程序
   a1.D1 = 15;         //对象a1的数据D1被赋值为15
   a2.D1 = 20;         //对象a2的数据D1被赋值为20
   … …
}
```

面向对象构造的表达方法I：面向对象程序设计语言

(5)消息与对象的交互?

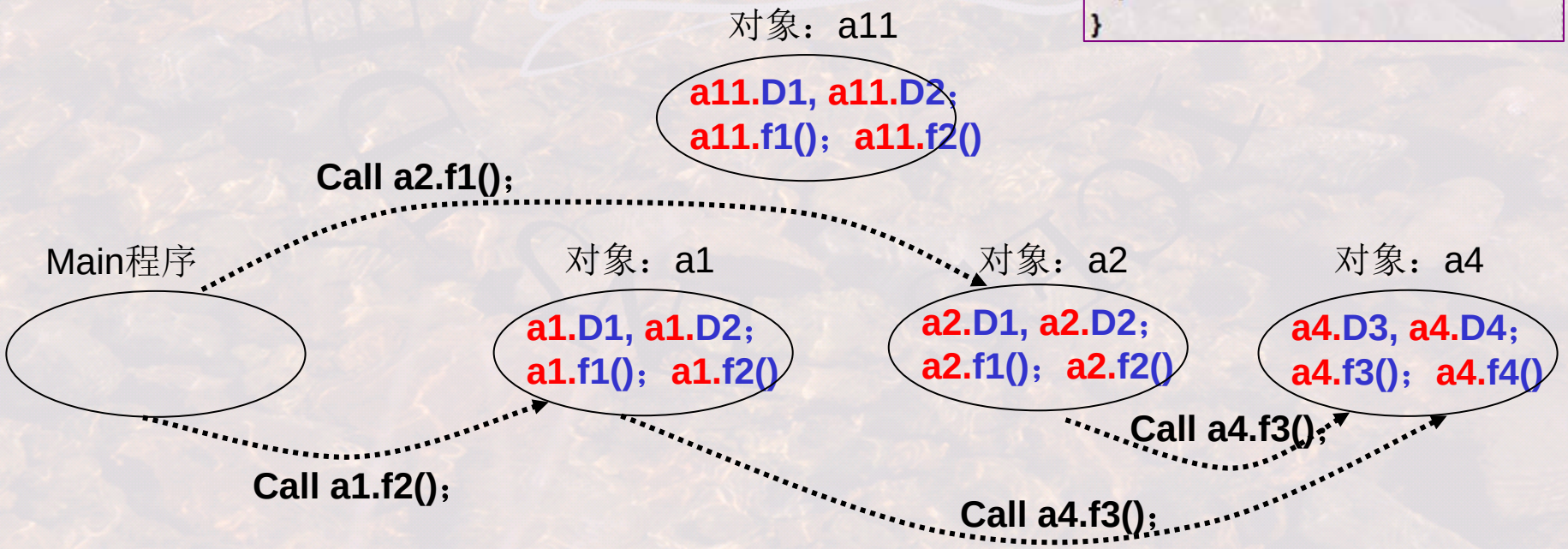
战德臣 教授

体验对象的产生

对象之间的消息交互

```
Main()
{ int M1, M2;
  a1 = new A1;      //创建类A1的对象a1
  a2 = new  A1;     //创建类A1的对象a2
  a11 = new A1;     //创建类A1的对象a11

  M1 = call a1.f2();          //执行对象a1的f2的程序
  M2 = call a2.f1();          //执行对象a2的f1的程序
  a1.D1 = 15;       //对象a1的数据D1被赋值为15
  a2.D1 = 20;       //对象a2的数据D1被赋值为20
  … …
}
```

```
Class A1          //定义一个类
{
   int  D1, D2    //定义变里
   int  f1();     //定义类中的函数
   int  f2();     //定义类中的函数

   int  f1()      //类A1的函数f1的程序
   {  //f1的程序语句块 }

   int f2()       //类A1的函数f2的程序
   {   //f2的程序语句块1
       a4 = new  A2;   //创建类A2的对象a4
      //f2的程序语句块2
      D1 = call  a4.f3(); //调用另一对象的某一函数
      //f2的程序语句块3
   }
}
```



面向对象构造的表达方法I：面向对象程序设计语言

(6)再看：面向过程的计算机语言 vs. 面向对象的计算机语言?

战德臣 教授

差别：面向过程的程序要素 vs. 面向对象的程序要素

```
Main()
{
     int  D1, D2
     int sum; sum=D1;
     call f1(sum);
     call f2();
}

int f1(int qty)
{  int temp;
   temp=qty*qty;
   return(qty);
    //f1的程序语句块 }

int f2()
{ //f2的程序语句块 }
```

全局变量

独立定义的函数

局部变量

```
Class  A1
{     int  D1, D2;
      int  f1(int qty)
      {   int temp; … …//f1的程序语句块  }
}
int A1:: f2()
{  //f2的程序语句块 }
int f2() { //f2的程序语句块 }
Main() {  …
    a1 = new A1;
    a2 = new A1;
    a1.D1 = M1;
    call f2();
    call  a1.f1(M1);
    call  a2.f1(M2);
}
```

定义“类”

类中定义的函数

类中定义的函数

独立定义的函数

用类产生对象

调用独立的函数

调用对象的函数

面向对象构造的表达方法II：
统一建模语言-UML

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

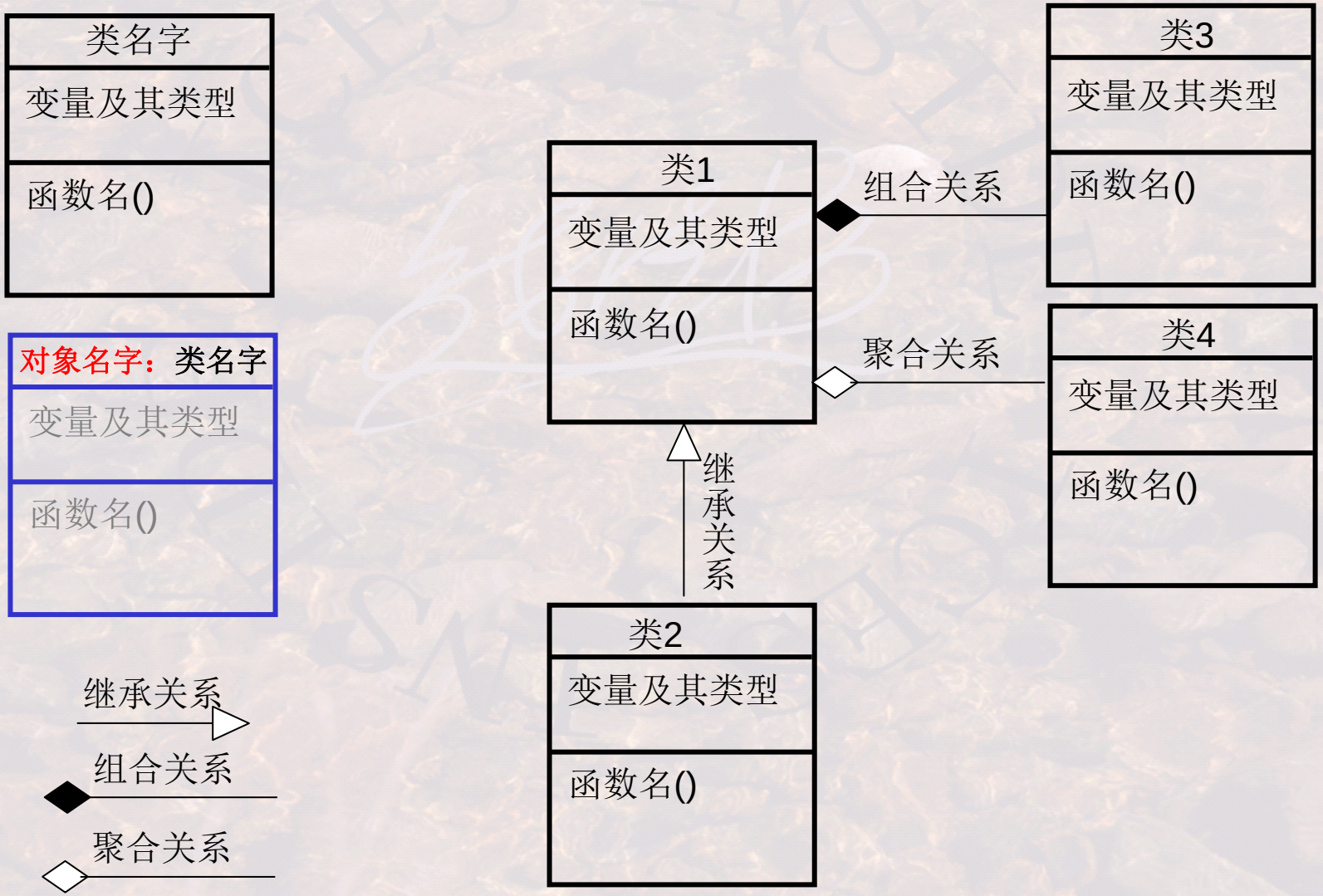
战德臣 教授

UML是什么?

- **Unified Modeling Language,**统一建模语言
- 面向对象程序分析、设计与构造的一种表达方法；
- 包含了类图/对象图、次序图、状态图、用例图等图形化的表达方法；
- 便于人们交流分析设计的成果；
- 软件工程领域的一种共用的表达方法；
- 软件工程专业学生必须要掌握的方法。

战德臣 教授

类图/对象图：描述类及其之间关系的一种图示化方法

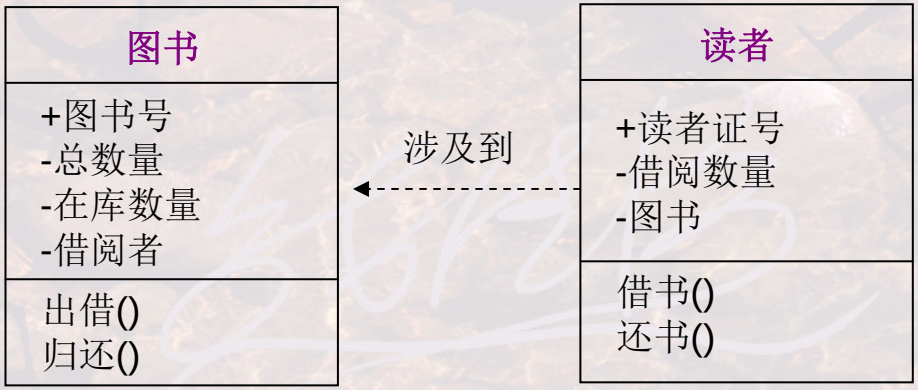


战德臣 教授

图书借阅管理(简化版)

一种图书被哪些读者借阅；一位读者借阅了哪些图书

可以这样构造




```
Class 图书 { ...
出借(读者证号)
{
    在库数量 = 在库数量 - 1;
    借阅者 = 借阅者+{读者证号}
}
归还(读者证号)
{
    在库数量 = 在库数量 + 1;
    借阅者 = 借阅者-{读者证号}
}           }
```

```
Class 读者 { ...
借书(图书号)
{   anObject = GetObject(图书号);
    call anObject.出借(读者证号);
    借阅数量 = 借阅数量 +1;
    图书 = 图书 + { 图书号 };
}
还书(图书号)
{   anObject = GetObject(图书号);
    call anObject.归还(读者证号);
    借阅数量 = 借阅数量 - 1;
    图书 = 图书 - { 图书号 };
}           }
```

面向对象程序构造的表达方法II：统一建模语言-UML

(3)用“类图”表达构造的一个例子

战德臣 教授

```
Class 图书 {
出借(读者证号)
{
   在库数量 = 在库数量 - 1;
   借阅者 = 借阅者+{读者证号}
}
归还(读者证号)
{
   在库数量 = 在库数量 + 1;
   借阅者 = 借阅者-{读者证号}
}
```

图书
+图书号 -总数量 -在库数量 -借阅者
出借() 归还()

涉及到

读者
+读者证号 -借阅数量 -图书
借书() 还书()

```
Class 读者 {
借书(图书号)
{  anObject = GetObject(图书号);
   call anObject.出借(读者证号);
   借阅数量 = 借阅数量 +1;
   图书 = 图书 +{图书号};
}
还书(图书号)
{  anObject = GetObject(图书号);
   call anObject.归还(读者证号);
   借阅数量 = 借阅数量 - 1;
   图书 = 图书 -{图书号};
}
```

表达的是这样的意思

张三：读者
+读者证号：R1 -借阅数量：3 -图书：{B1,B2,B3}
借书() 还书()

李斯：读者
+读者证号：R2 -借阅数量：2 -图书：{B2, B3}
借书() 还书()

王五：读者
+读者证号：R3 -借阅数量：1 -图书：{B2}
借书() 还书()

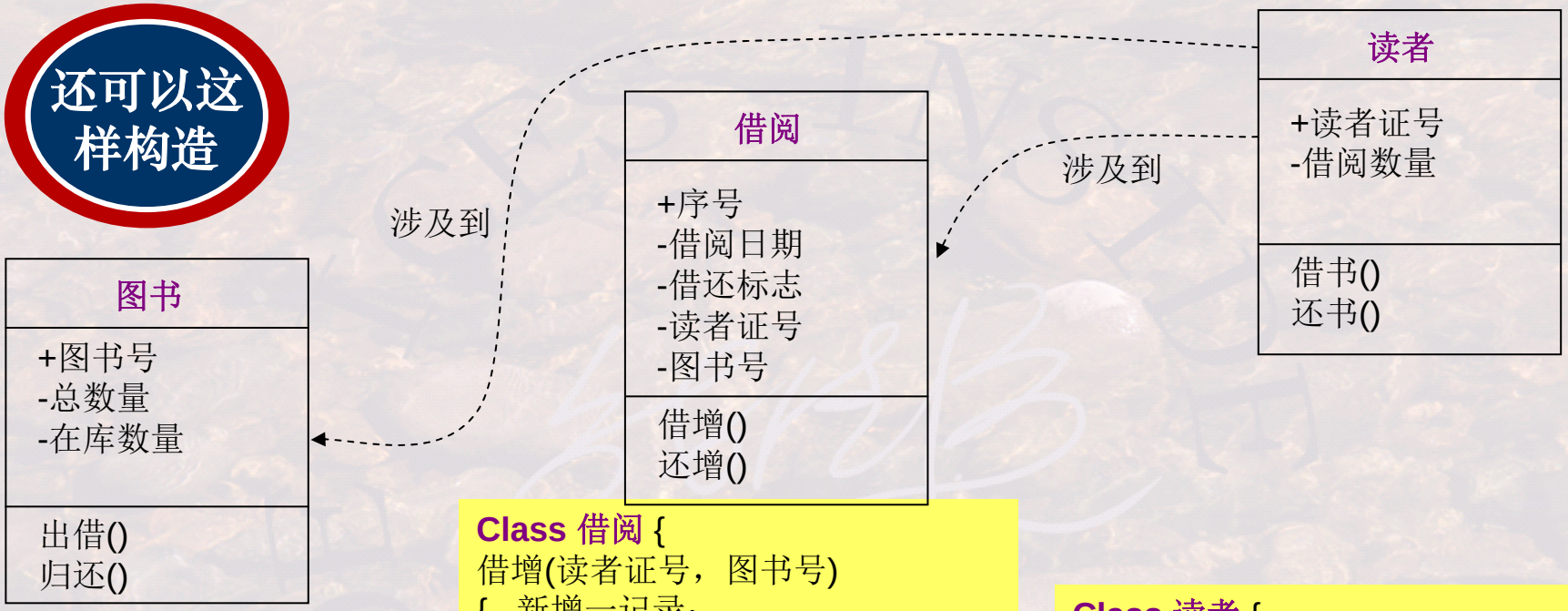
数据库：图书
+图书号：B1 -总数量：5 -在库数量：4 -借阅者：{R1}
出借() 归还()

操作系统：图书
+图书号：B2 -总数量：5 -在库数量：2 -借阅者：{R1,R2,R3}
出借() 归还()

离散数学：图书
+图书号：B3 -总数量：5 -在库数量：3 -借阅者：{R1,R2}
出借() 归还()

面向对象程序构造的表达方法II：统一建模语言-UML
(3)用“类图”表达构造的一个例子

ICES 战德臣 教授




```
Class 图书 {
出借()
{
   在库数量 = 在库数量 - 1;
}
归还()
{
   在库数量 = 在库数量 + 1;
}            }
```

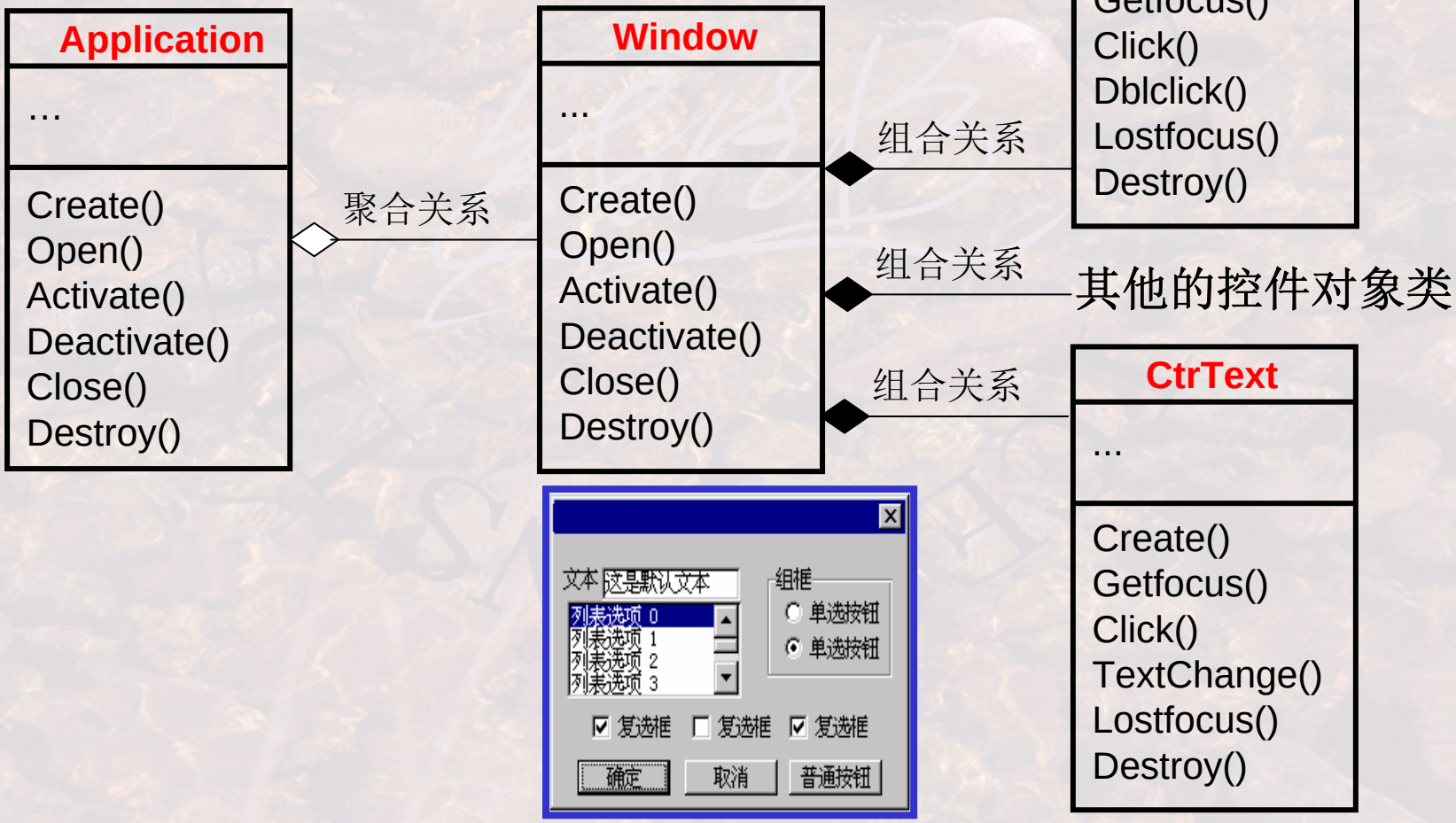
```
Class 借阅 {
借增(读者证号，图书号)
{  新增一记录;
   设置：“借还标志”=借出
         “读者证号” = 读者证号;
         “图书号” = 图书号;
         “借阅日期” = 当前日期;
}
还增(图书号，图书号)
{  新增一记录;
   设置：“借还标志”=归还;
         “读者证号” = 读者证号;
         “图书号” = 图书号;
         “借阅日期” = 当前日期;
}            }
```

```
Class 读者 {
借书(图书号)
{  anObject = GetObject(图书号);
    call anObject.出借();
   借阅数量 = 借阅数量 +1;
   call 借阅.借增(读者证号，图书号);
}
还书(图书号)
{  anObject = GetObject(图书号);
    call anObject.归还();
   借阅数量 = 借阅数量 - 1;
   call 借阅.还增(读者证号，图书号);
}            }
```

面向对象程序构造的表达方法II：统一建模语言-UML
(4)用“类图”表达构造的另一个例子

战德臣 教授

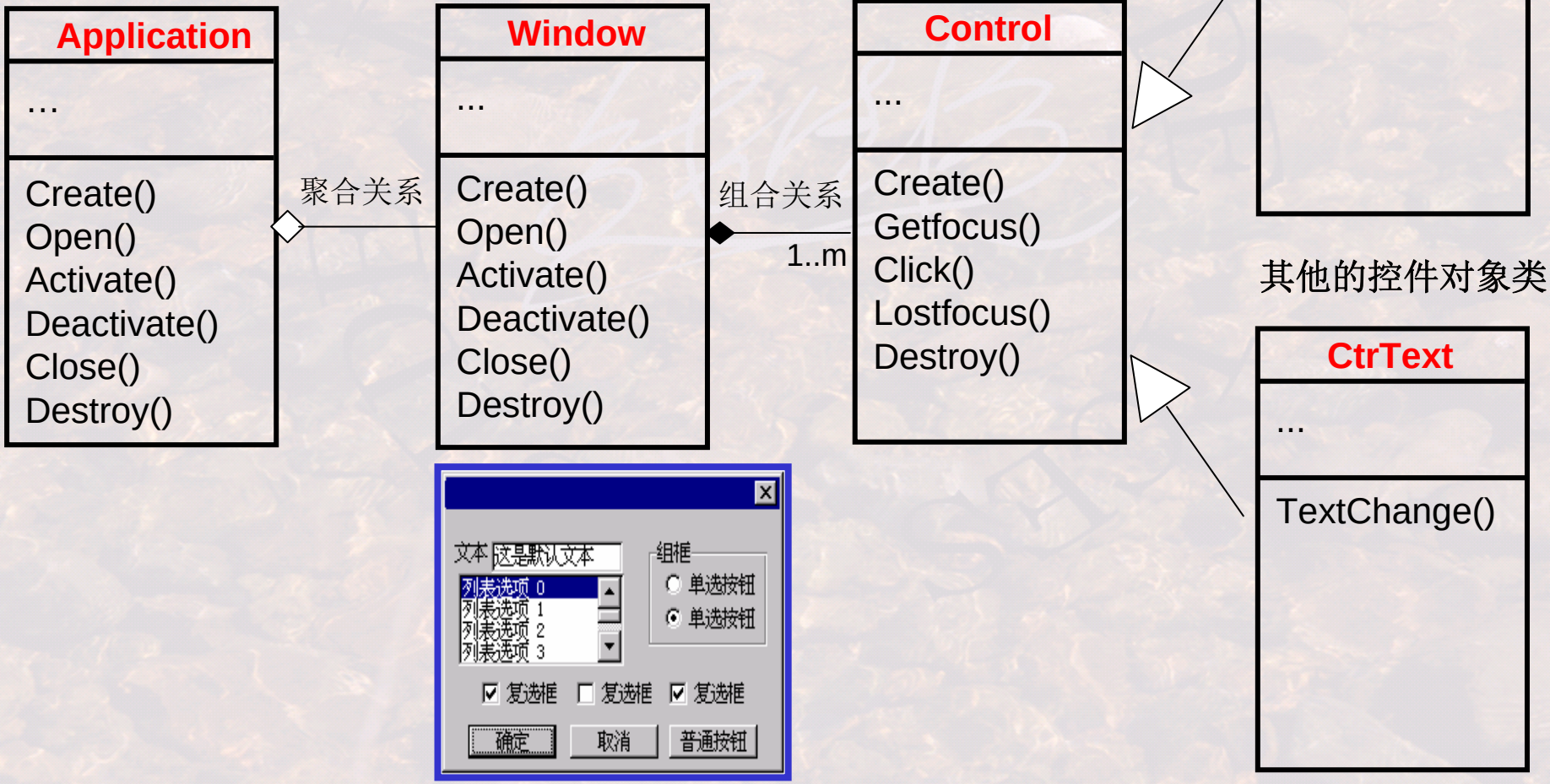
可以这样表达“窗口”



面向对象程序构造的表达方法II：统一建模语言-UML
(4)用“类图”表达构造的另一个例子

战德臣 教授

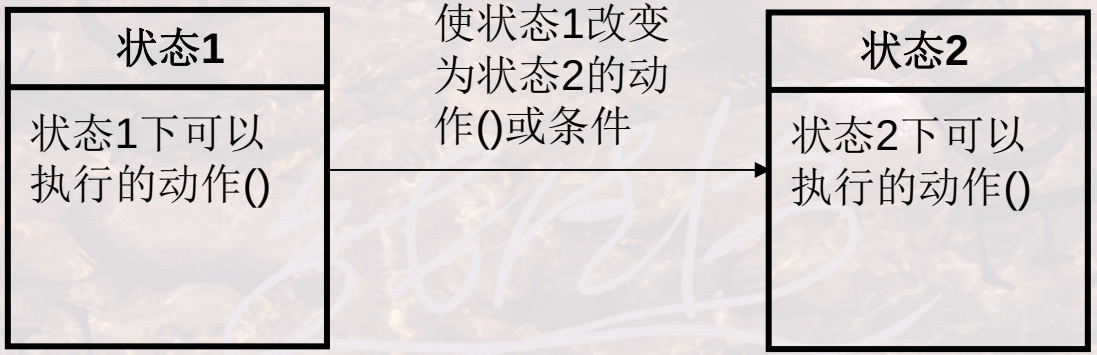
也可以这样表达“窗口”

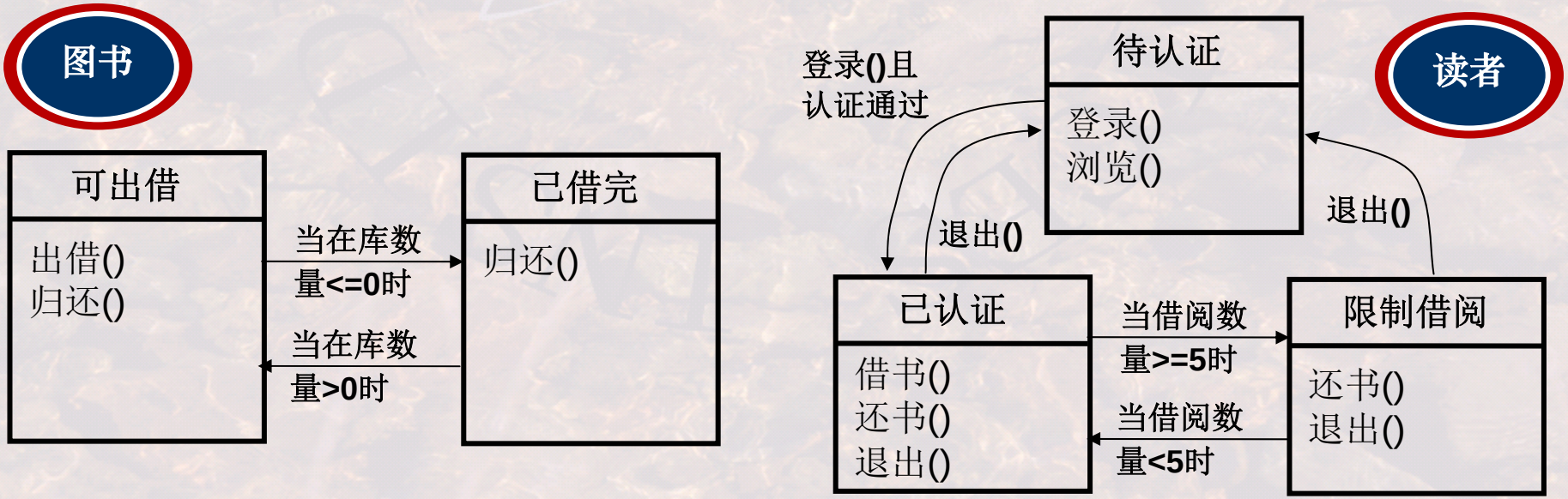


战德臣 教授

描述对象/类的状态变化关系

UML状态图：描述对象/类的状态变化关系的一种图示化方法





战德臣 教授

描述对象之间交互关系

序列图(又称次序图)：描述类的对象之间交互关系的一种图示化方法

类1的对象a发送一个消息给类1的对象b，即：对象a调用了对象b的一个函数()

类1的某一对象(注意类前面的冒号，表示该框中是一个对象)发送一个消息给类2的某一对象.即：类1的对象调用了类2某对象的一个函数()

战德臣 教授

图书借阅管理

某一“读者”对象与某一“图书”对象的交互过程

你是否还有疑问呢?

对象什么时候开始存在的?
对象什么时候消亡呢?
“图书”或“读者”对象消亡，则其保存的变量的信息是否也丢失了呢?

深入学习
(1)面向对象的程序设计语言
(2)统一建模语言
(3)软件设计模式
(4)软件体系结构
(5)软件工程

基于对象框架构造软件
--一种可视化编程示例
用Visual Basic开发一个应用程序

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

基于对象框架构造软件--一种可视化编程示例
(1)目标软件系统与其构造环境

战德臣 教授

软件系统的一种构造环境：Visual Basic

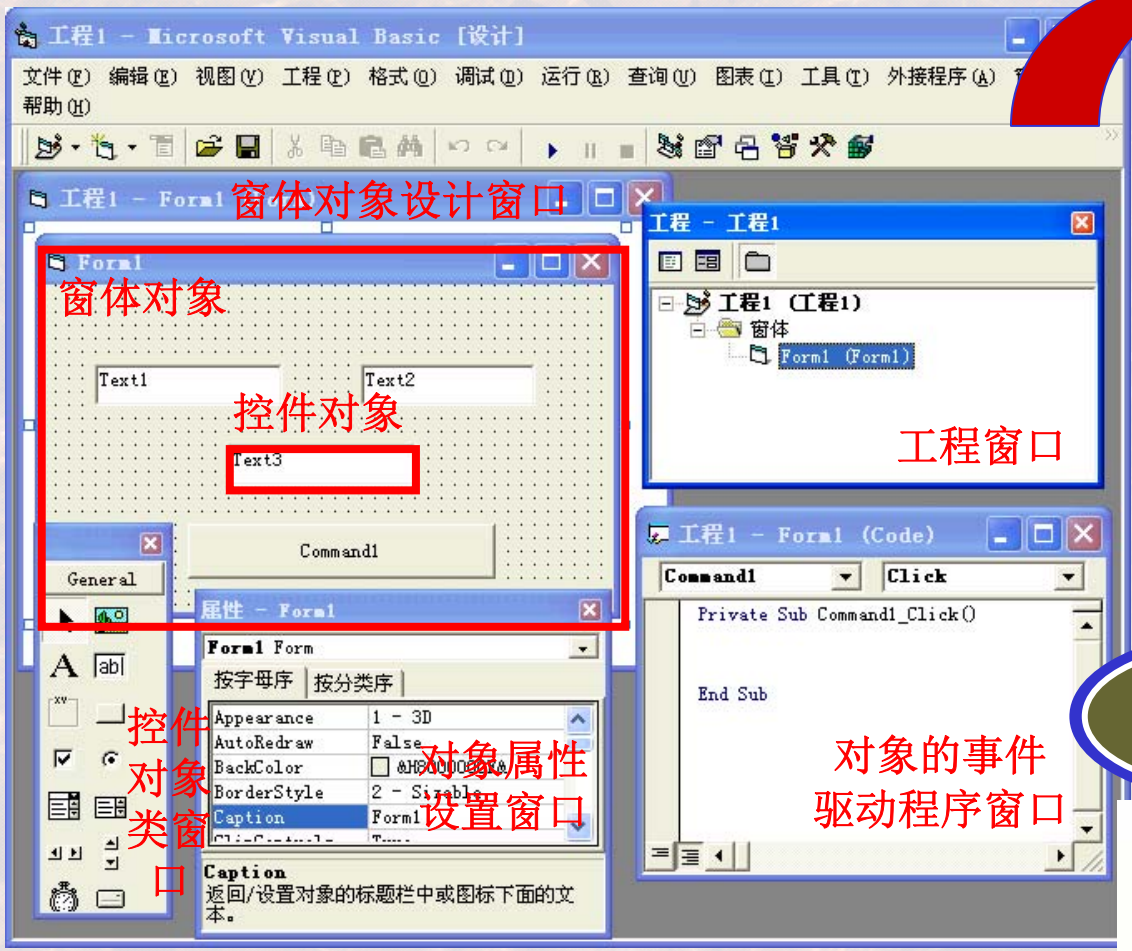
怎样构造呢?

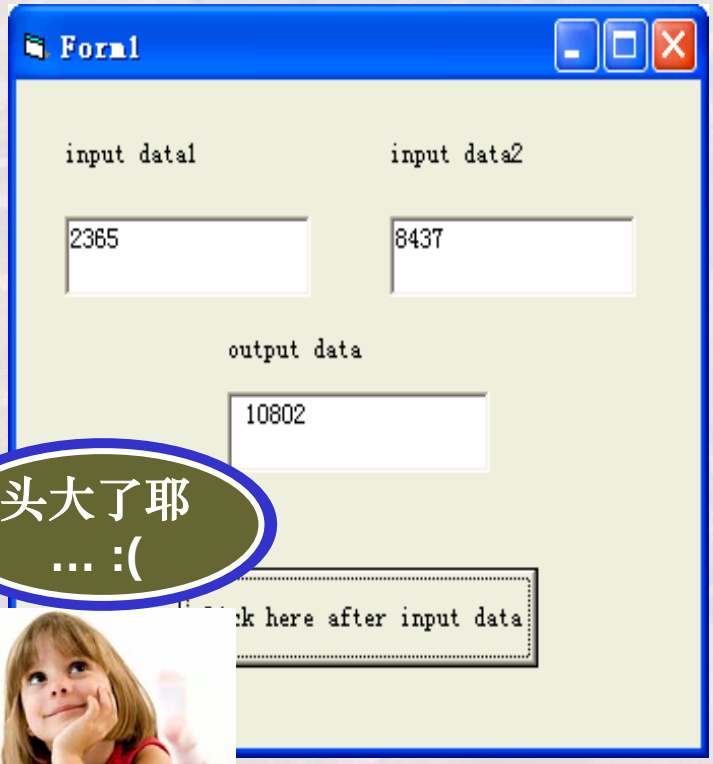
用户

使用

产生

(目标)软件系统



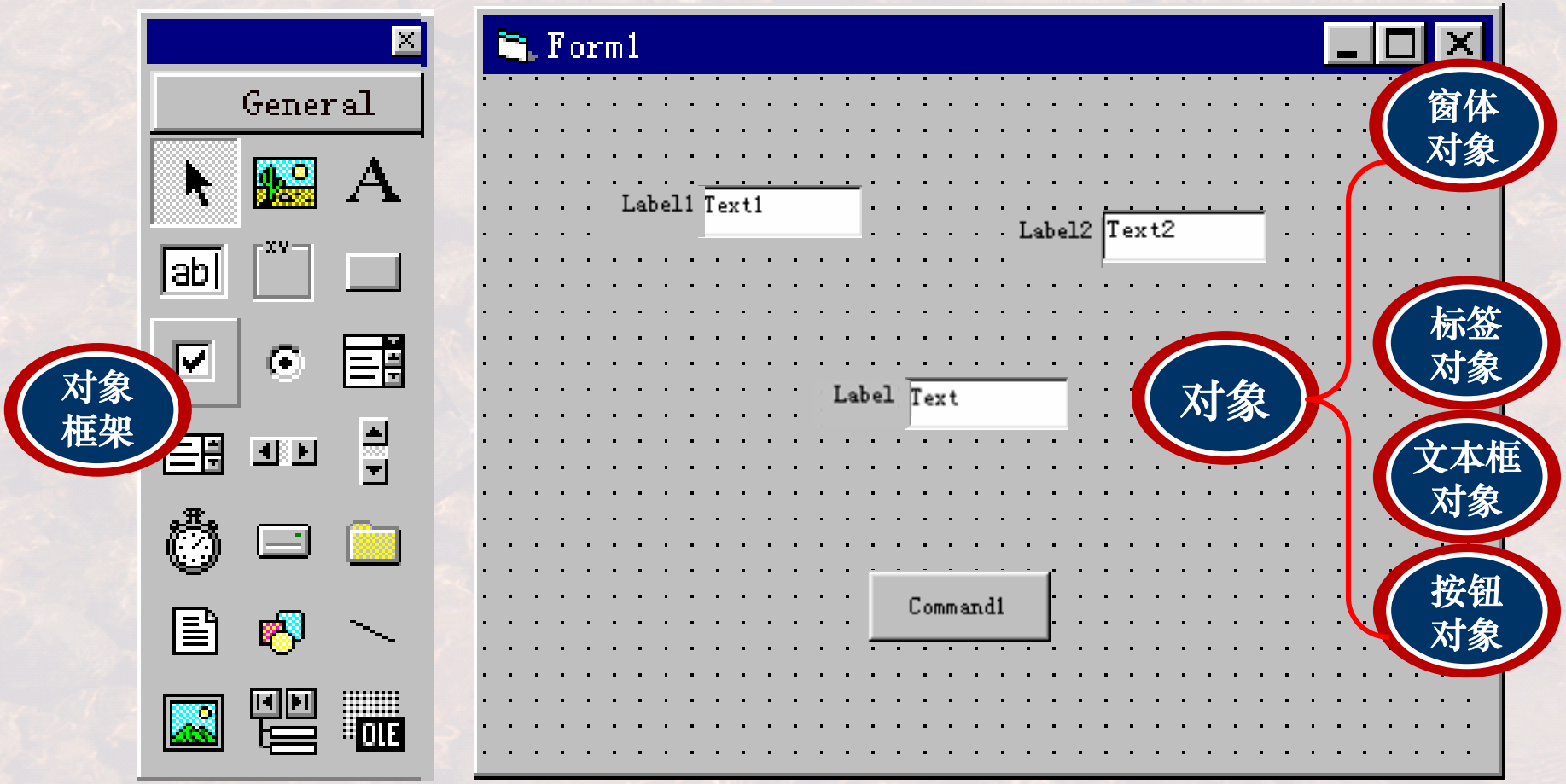


头大了耶 ... :(

基于对象框架构造软件--一种可视化编程示例
(2)如何利用环境提供的对象框架来构造程序

ICES 战德臣 教授

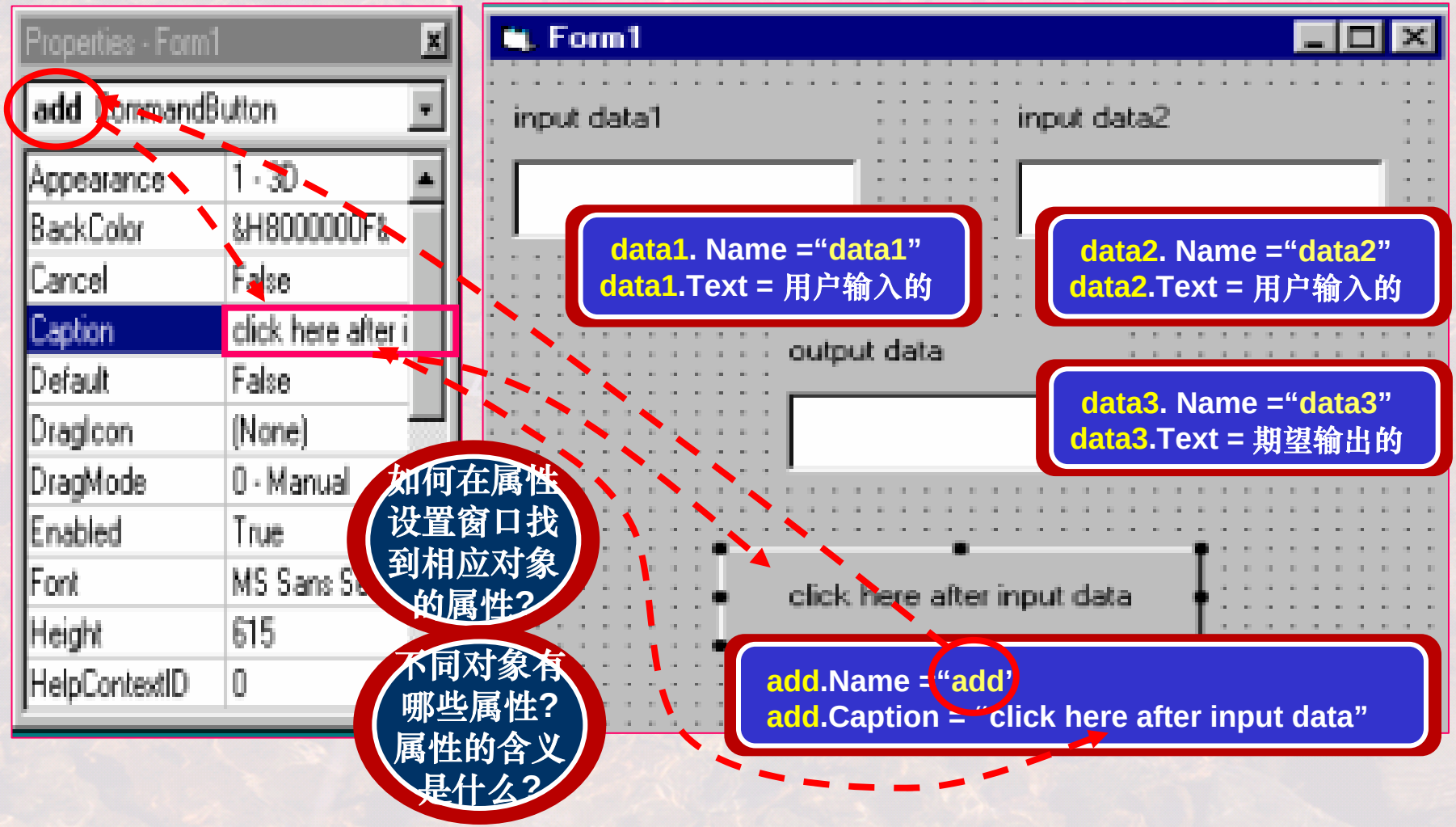
Visual Basic提供了一批部分功能已经实现的对象(对象框架)，从中选择合适的对象来构造自己的应用程序



基于对象框架构造软件--一种可视化编程示例
(2)如何利用环境提供的对象框架来构造程序

战德臣 教授

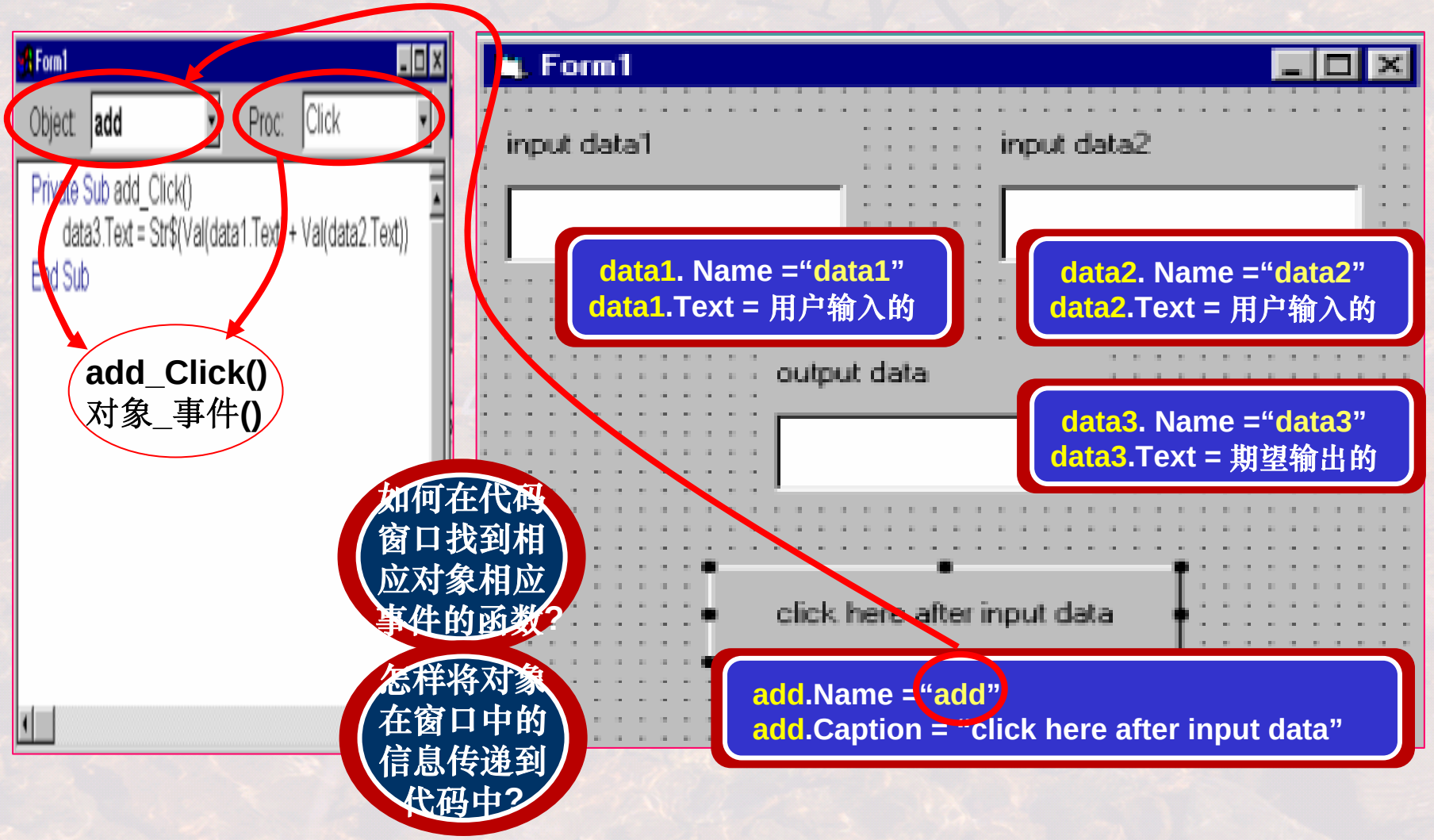
设置对象的属性，调整界面的布局，以反映应用程序的语义



基于对象框架构造软件--一种可视化编程示例
(2)如何利用环境提供的对象框架来构造程序

战德臣 教授

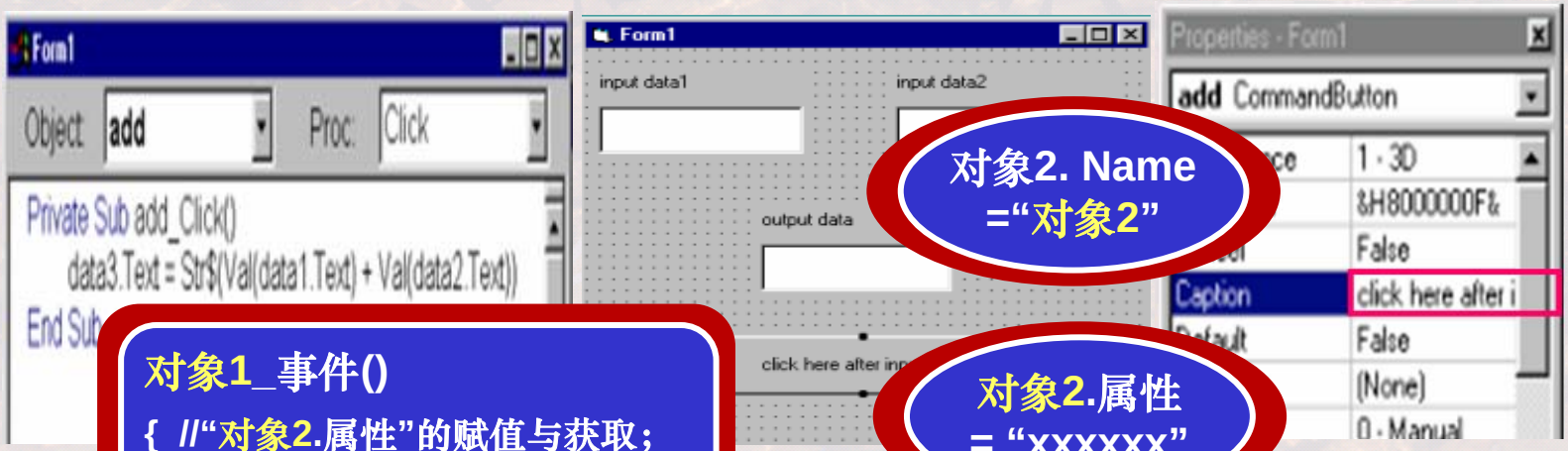
为不同对象的不同操作，编写不同的程序，以完成不同的功能



基于对象框架构造软件--一种可视化编程示例
(2)如何利用环境提供的对象框架来构造程序

战德臣 教授

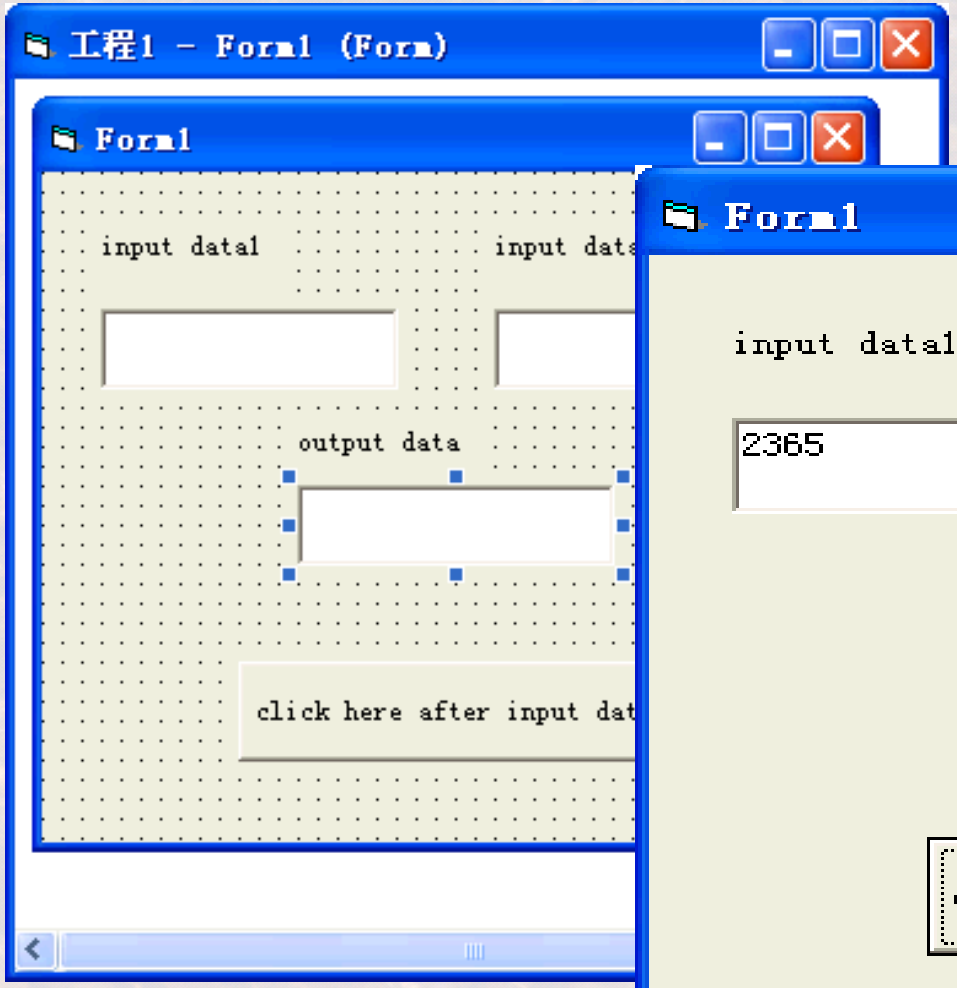
窗口-界面，与属性窗口-属性，与代码窗口-函数之间的关系



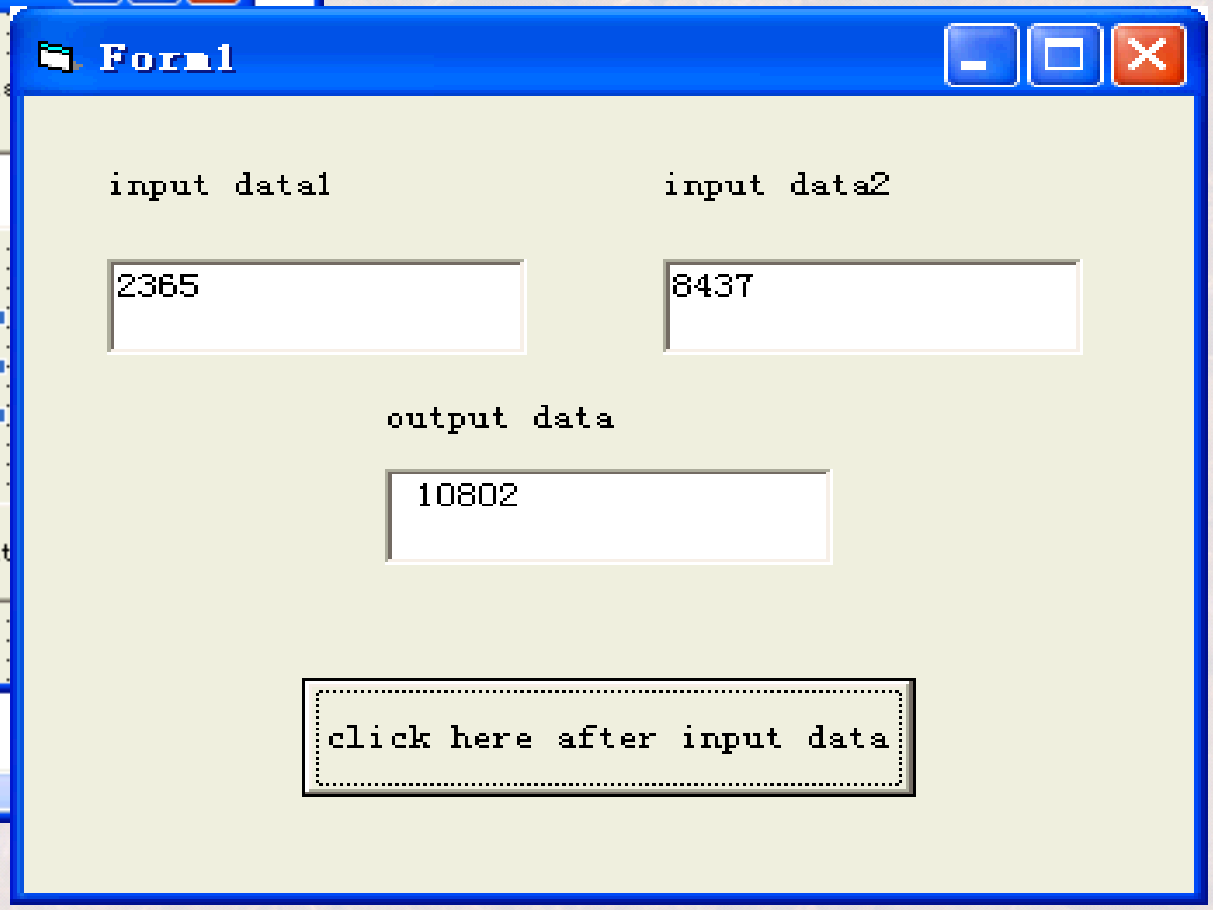
基于对象框架构造软件--一种可视化编程示例
(3)调试并运行应用程序

战德臣 教授

开发/编辑状态下的应用程序



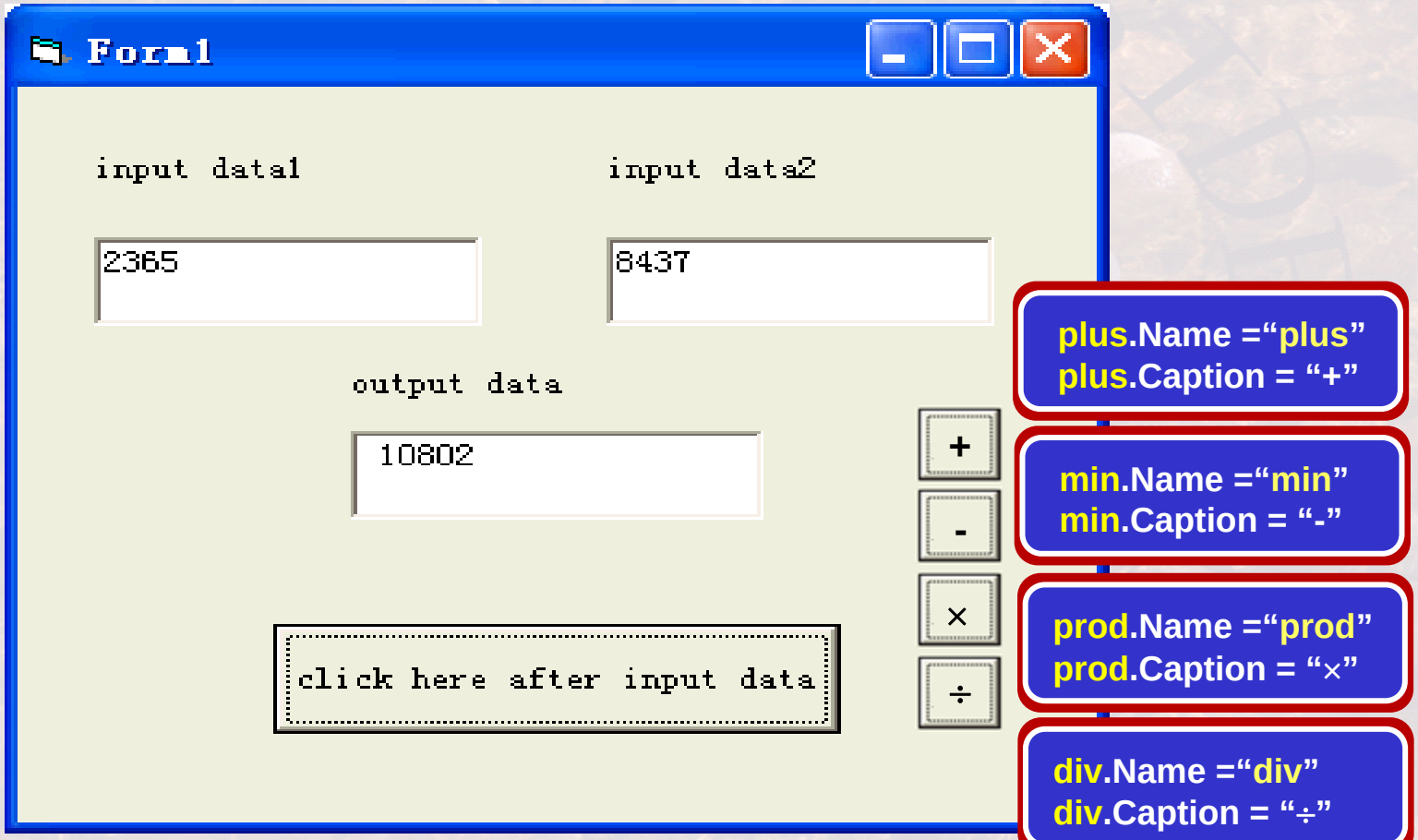
运行状态下的应用程序



基于对象框架构造软件--一种可视化编程示例
(4)再进行构造，以扩展功能

战德臣 教授

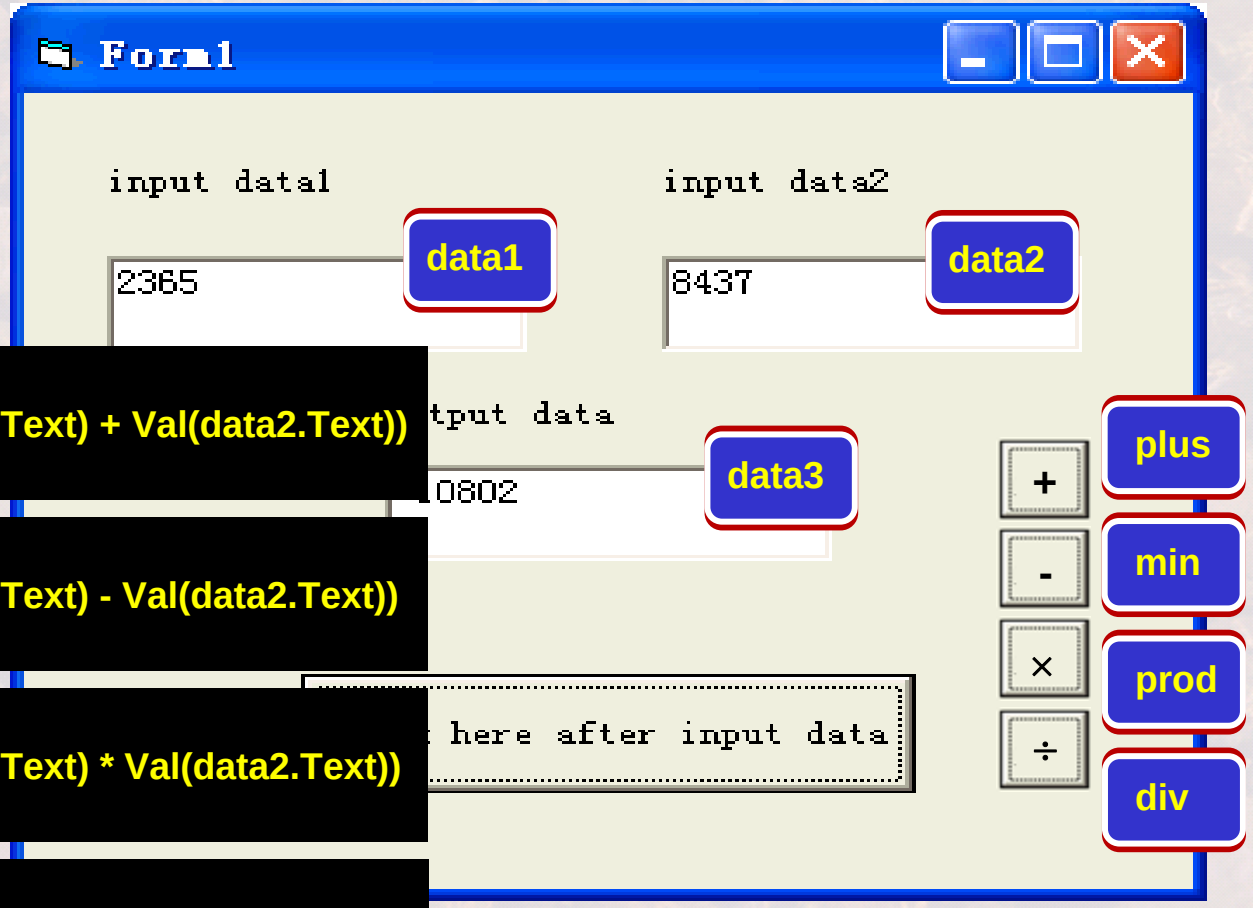
再增加一些功能，可以执行加减乘除运算



基于对象框架构造软件--一种可视化编程示例
(4)再进行构造，以扩展功能

战德臣 教授

再增加一些功能，可以执行加减乘除运算




```
Private Sub plus_Click()
    data3.Text = Str$( Val(data1.Text) + Val(data2.Text))
End Sub
```

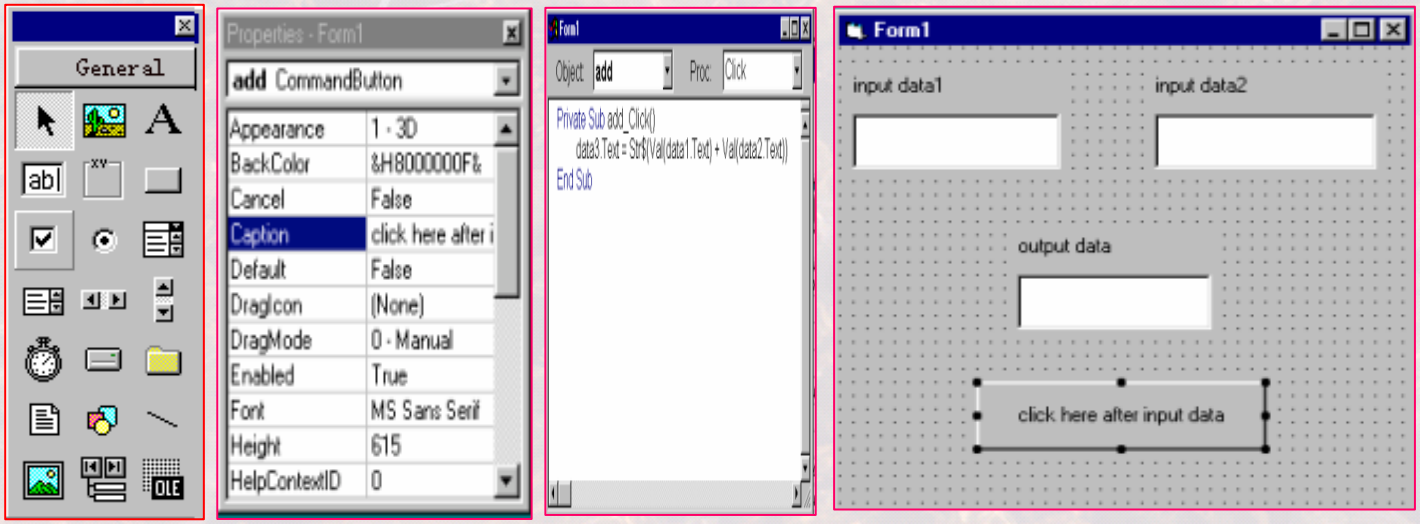
```
Private Sub min_Click()
    data3.Text = Str$( Val(data1.Text) - Val(data2.Text))
End Sub
```

```
Private Sub prod_Click()
    data3.Text = Str$( Val(data1.Text) * Val(data2.Text))
End Sub
```

```
Private Sub div_Click()
    data3.Text = Str$( Val(data1.Text) / Val(data2.Text))
End Sub
```

基于对象框架构造软件--一种可视化编程示例
(5)小结

战德臣 教授



用面向对象思维构造对象框架

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

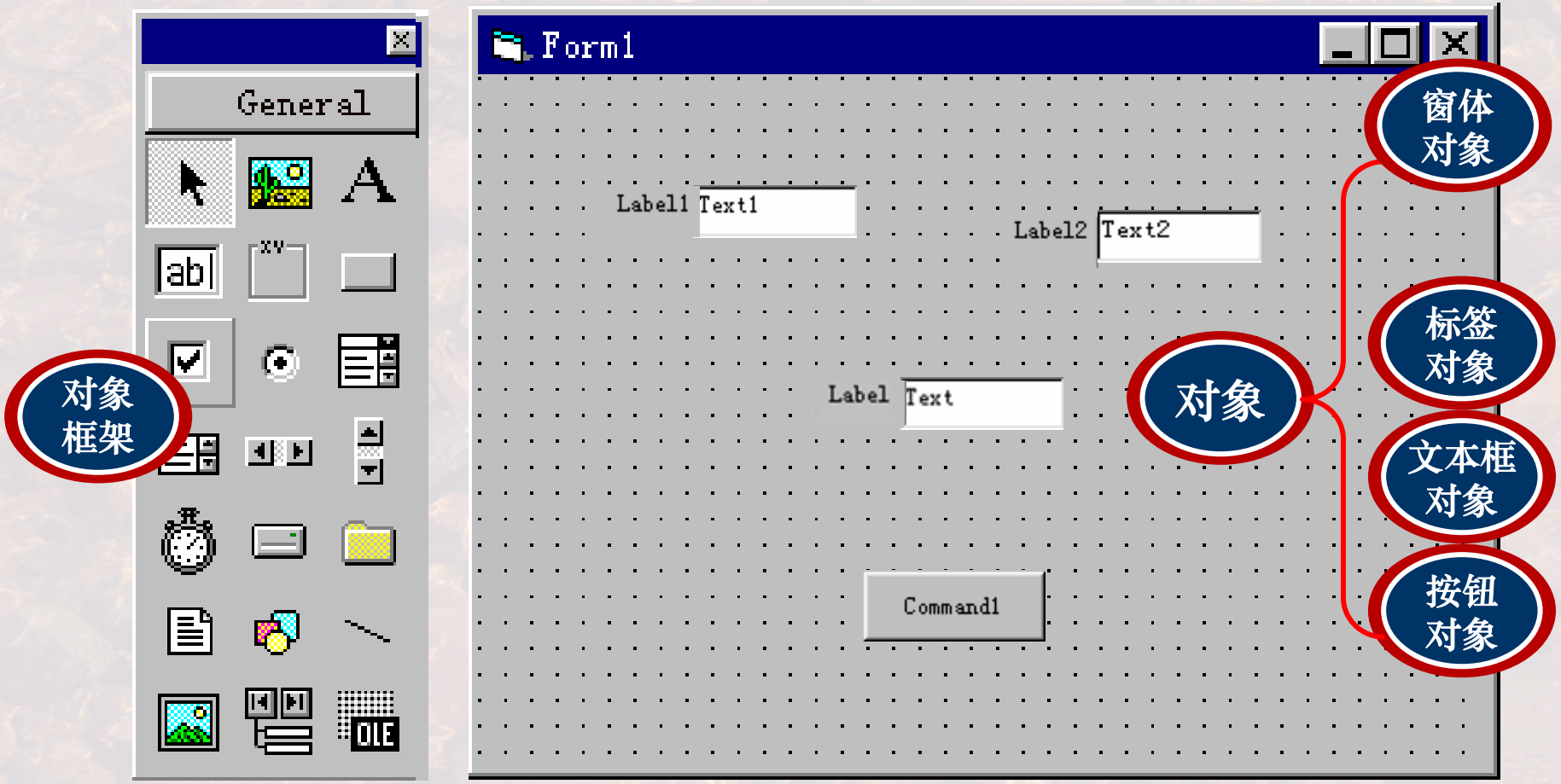
Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

用面向对象思维构造对象框架

(1)基于对象框架的应用程序构造?

战德臣 教授

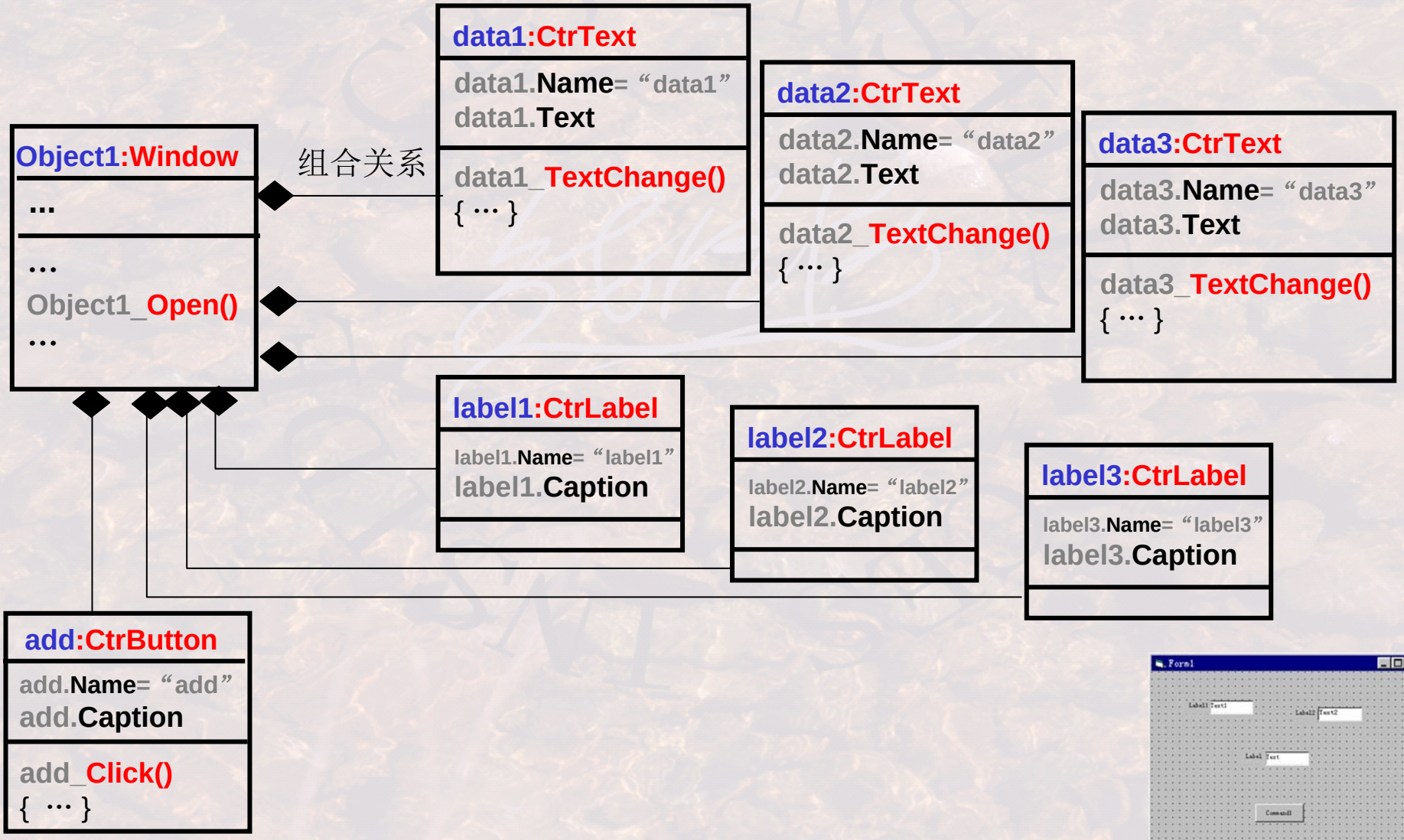
应用程序就是一系列对象，及其属性和函数的集合。



用面向对象思维构造对象框架
(1)基于对象框架的应用程序构造?

战德臣 教授

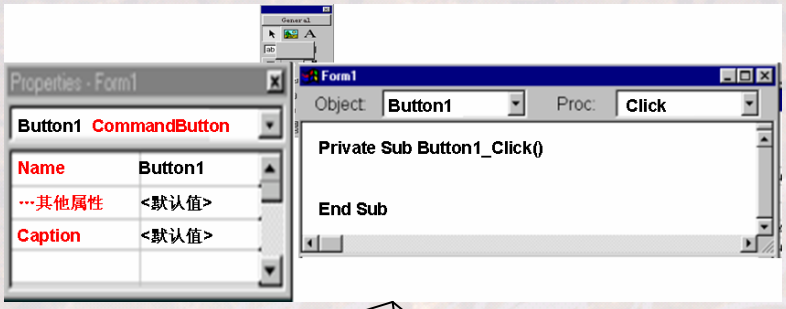
窗体对象与控件对象的关系示意(对象图)



用面向对象思维构造对象框架
(2)什么是对象框架?

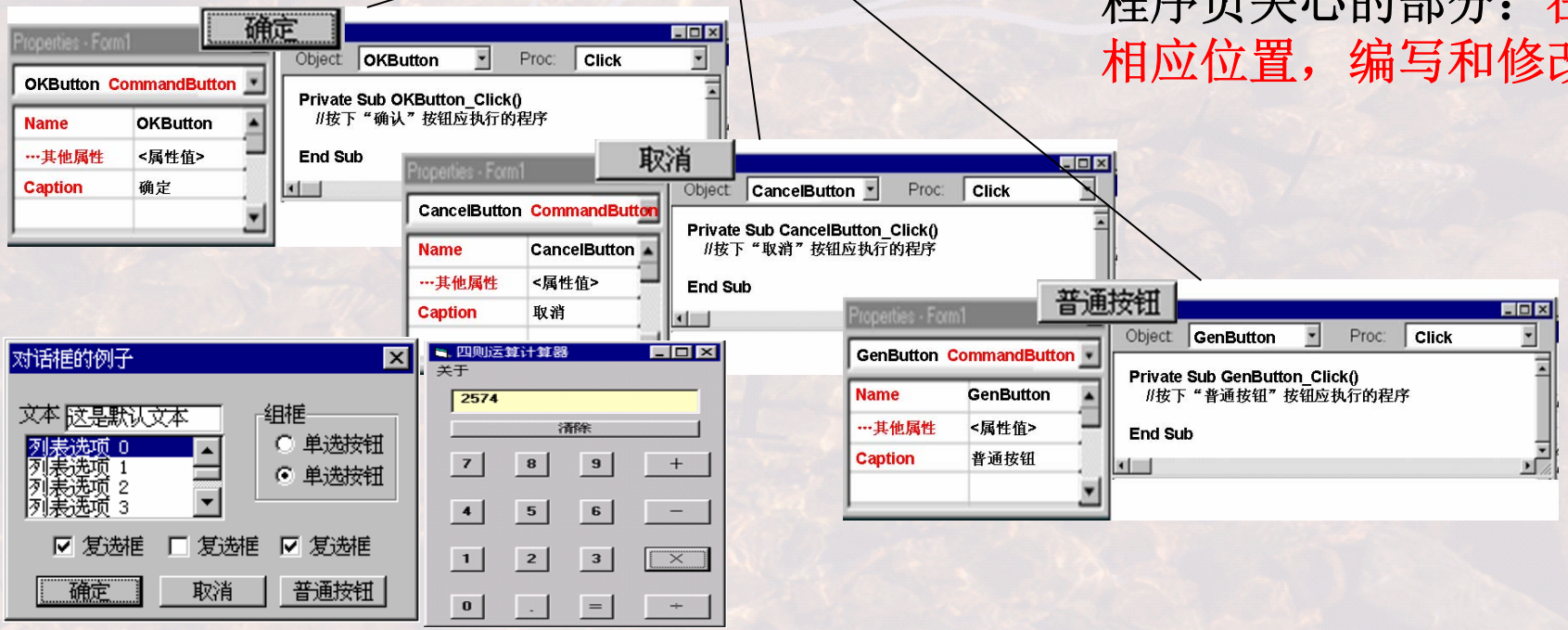
战德臣 教授

将同类别对象的共性内容做成一个对象框架



程序员无需关心的部分：
事先做好，直接使用
程序员可能关心的部分：
留出相应位置，允许程序员编写和修改

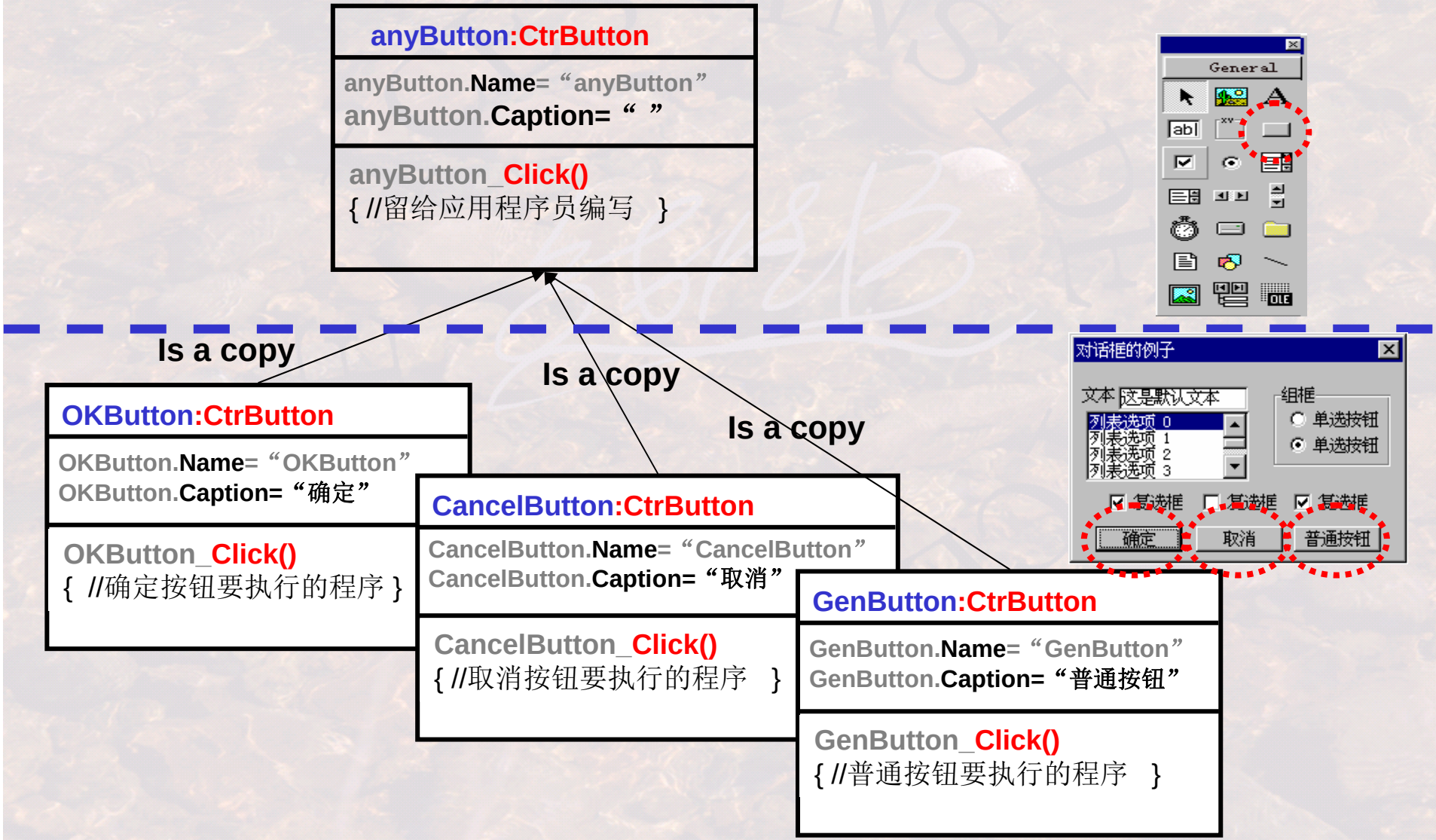
程序员关心的部分：在相应位置，编写和修改



用面向对象思维构造对象框架
(2)什么是对象框架?

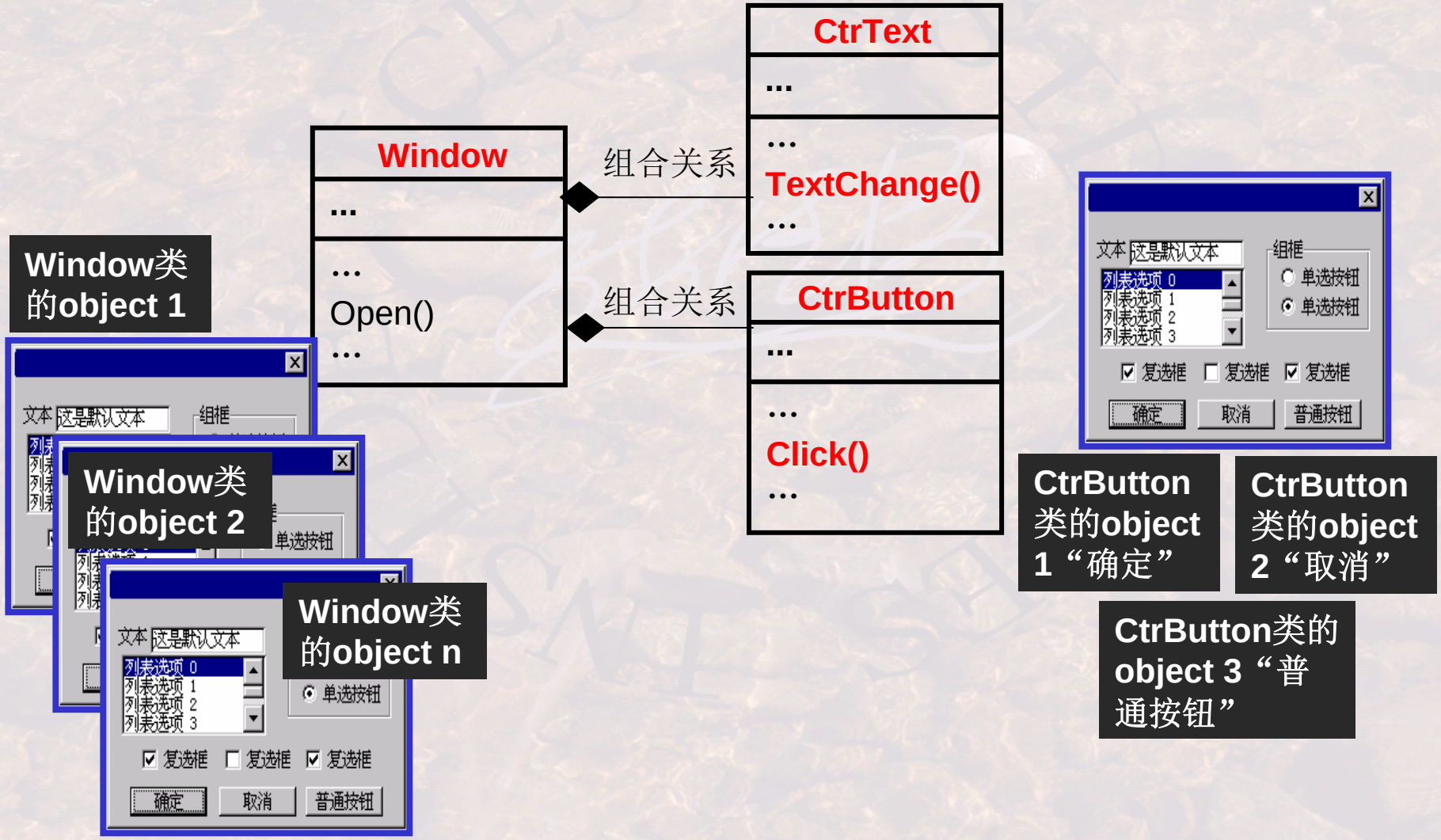
战德臣 教授

将同类别对象的共性内容做成一个对象框架



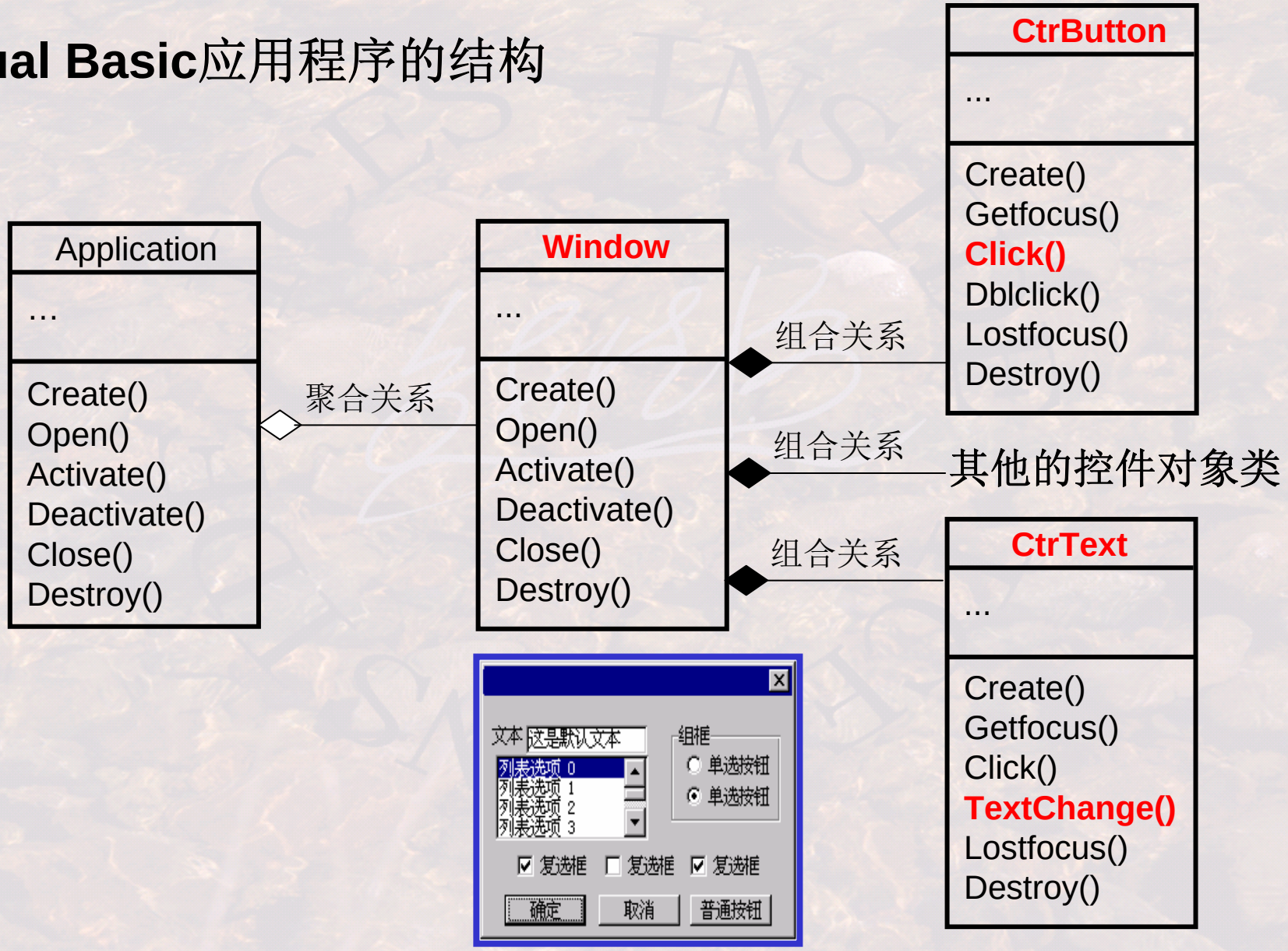
用面向对象思维构造对象框架
(3)由对象到类的表达?

窗体对象类与控件对象类的关系示意(类图)



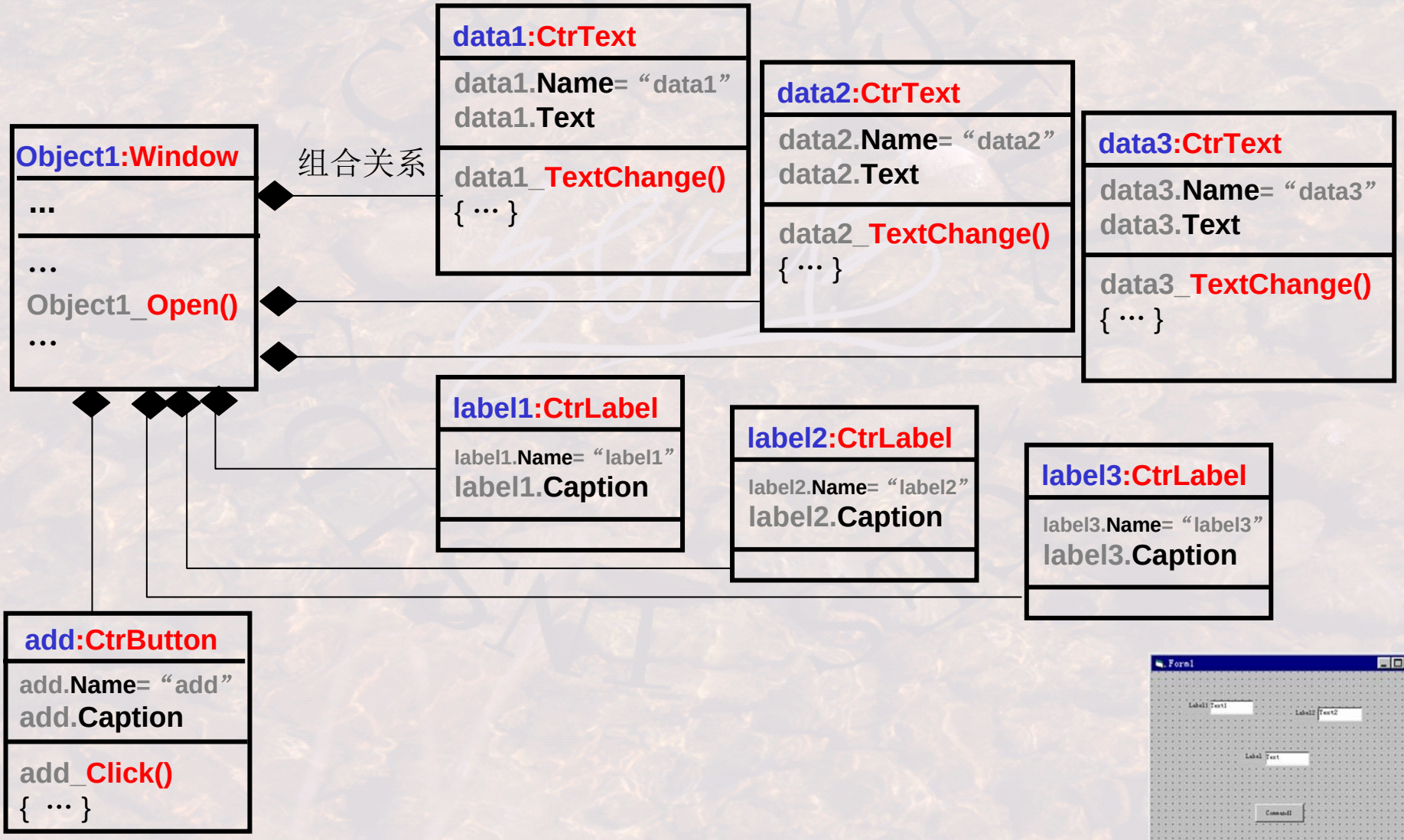
战德臣 教授

Visual Basic应用程序的结构



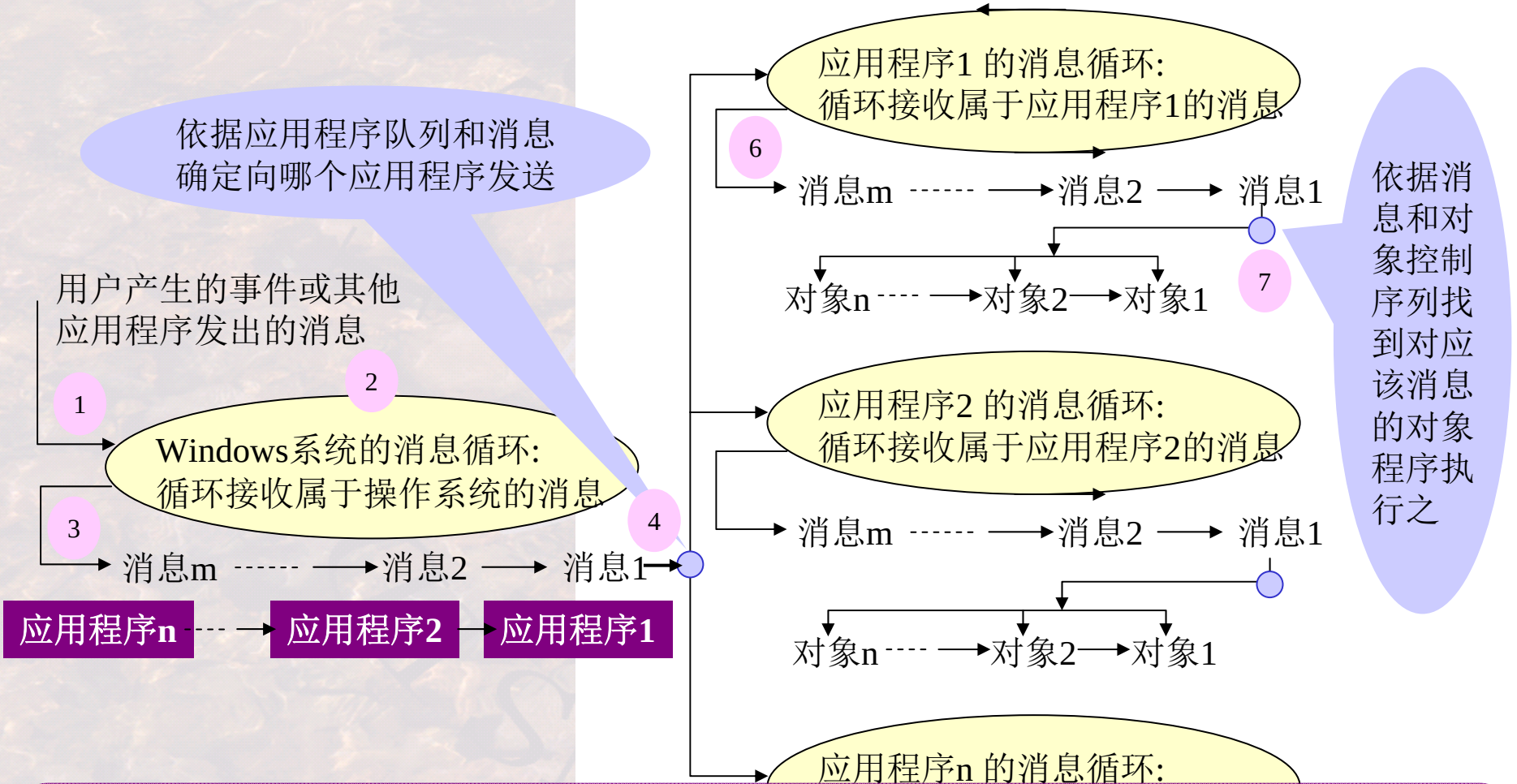
战德臣 教授

Visual Basic应用程序的结构—实例-对象图



用面向对象思维构造对象框架
(4)一种对象交互的支撑技术—消息循环?

战德臣 教授



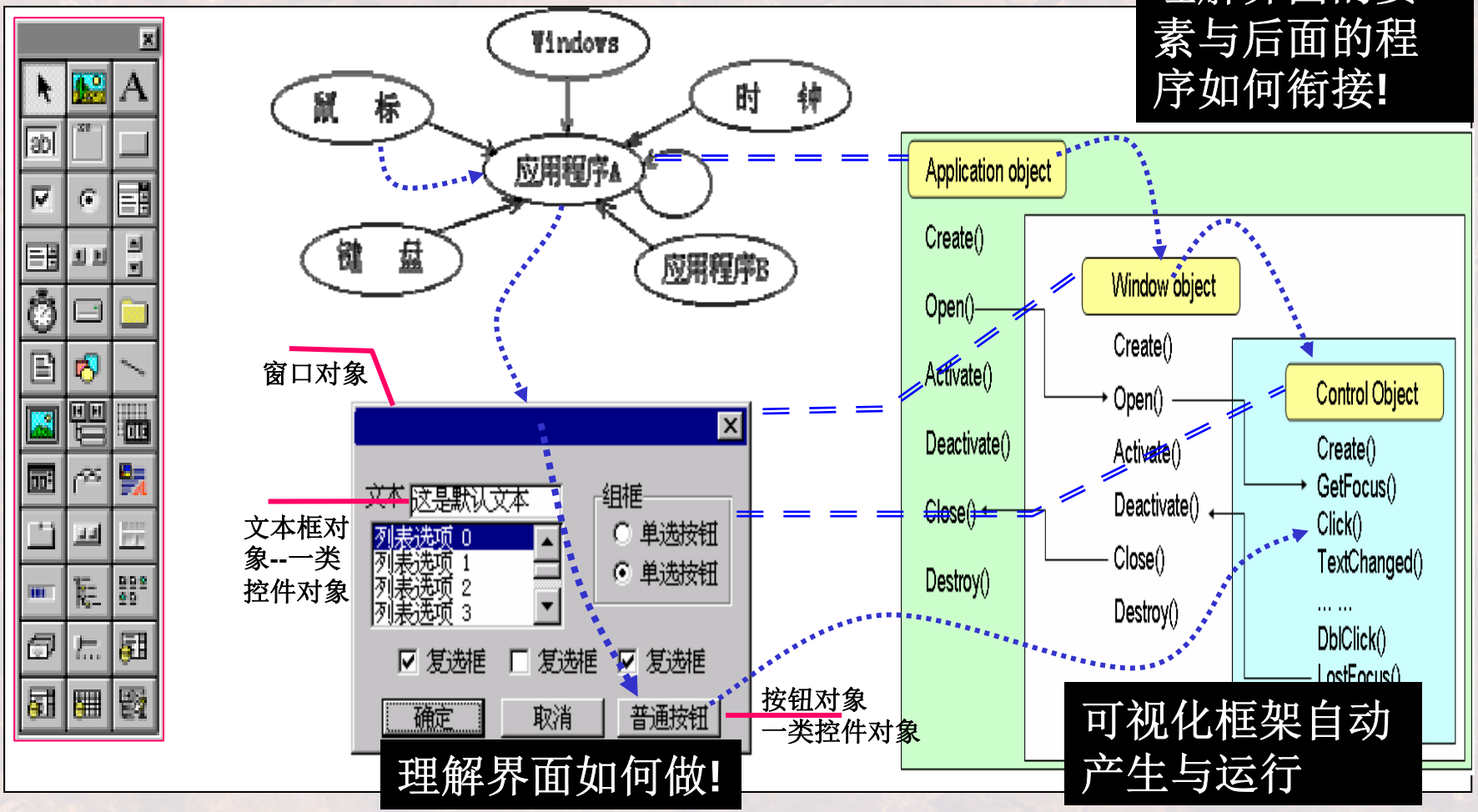
消息循环是一个消息管理程序。它能够接收各种消息，并将消息传递给各种相关对象，调用相关对象的事件驱动程序予以处理。是对象之间调用关系的一种处理机制。

用面向对象思维构造对象框架

(5)Visual Basic的对象组织及其调用关系?

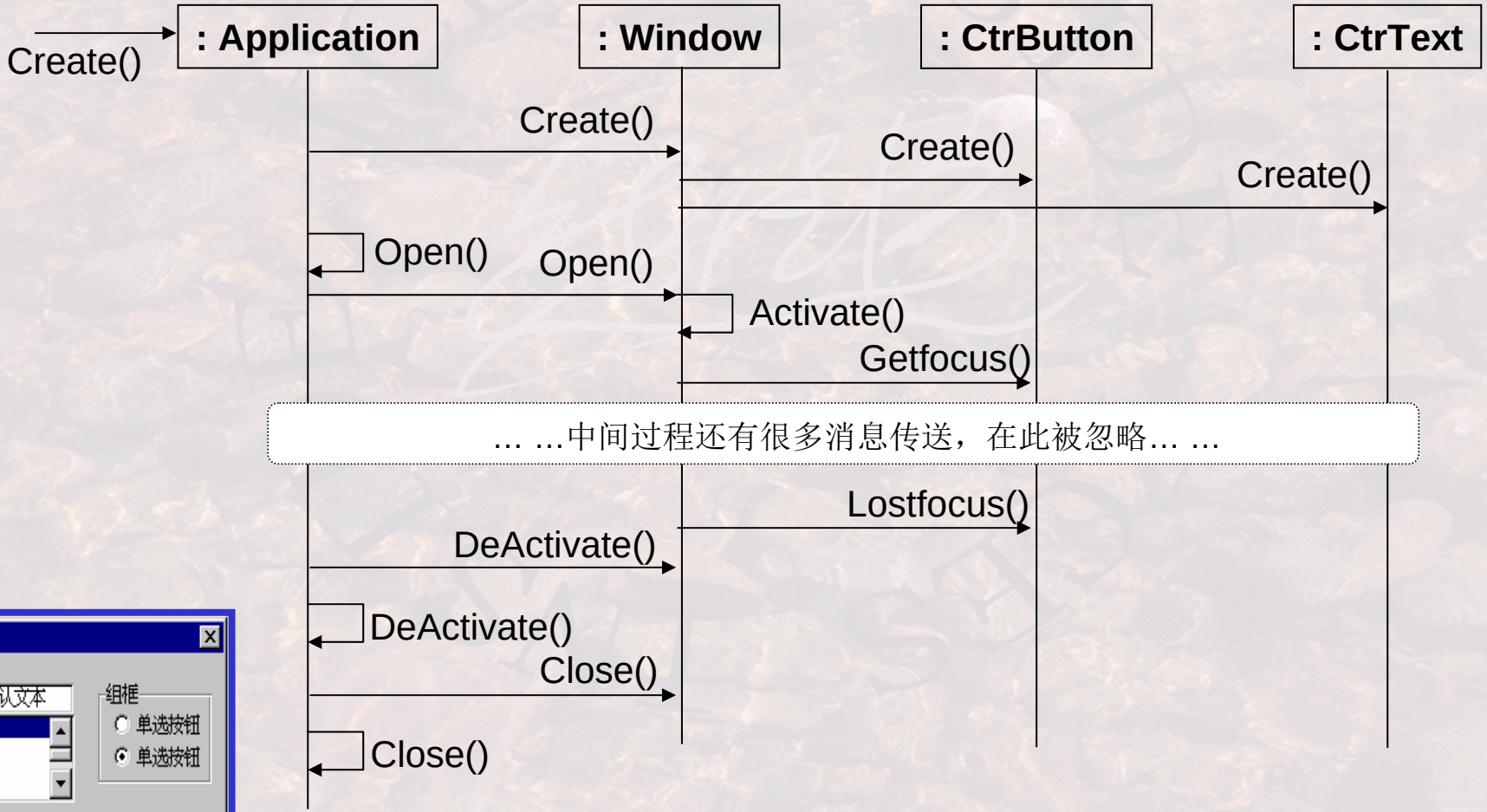
战德臣 教授

将不同类别对象之间的关系做成一个结构框架。结构框架是按照某一种体系结构(Architecture)实现的，用于连接、装配各种对象形成系统的一套程序



战德臣 教授

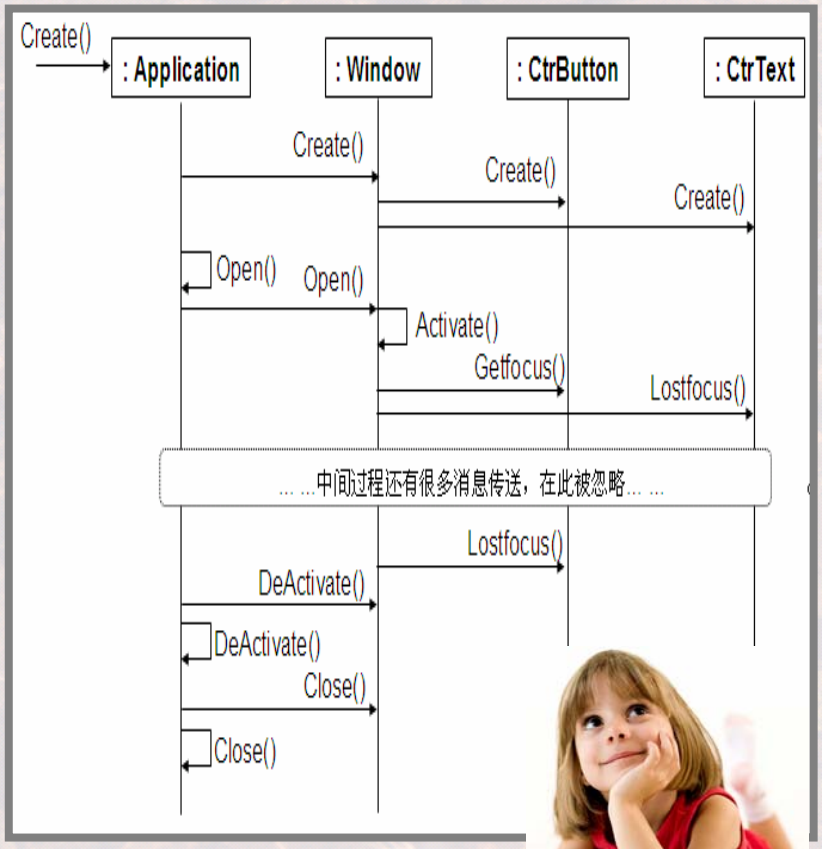
Visual Basic各类对象的消息交互关系示意



战德臣 教授

软件模式(Software Pattern)

由面向对象的程序构造，到(可视化的)对象框架的构造

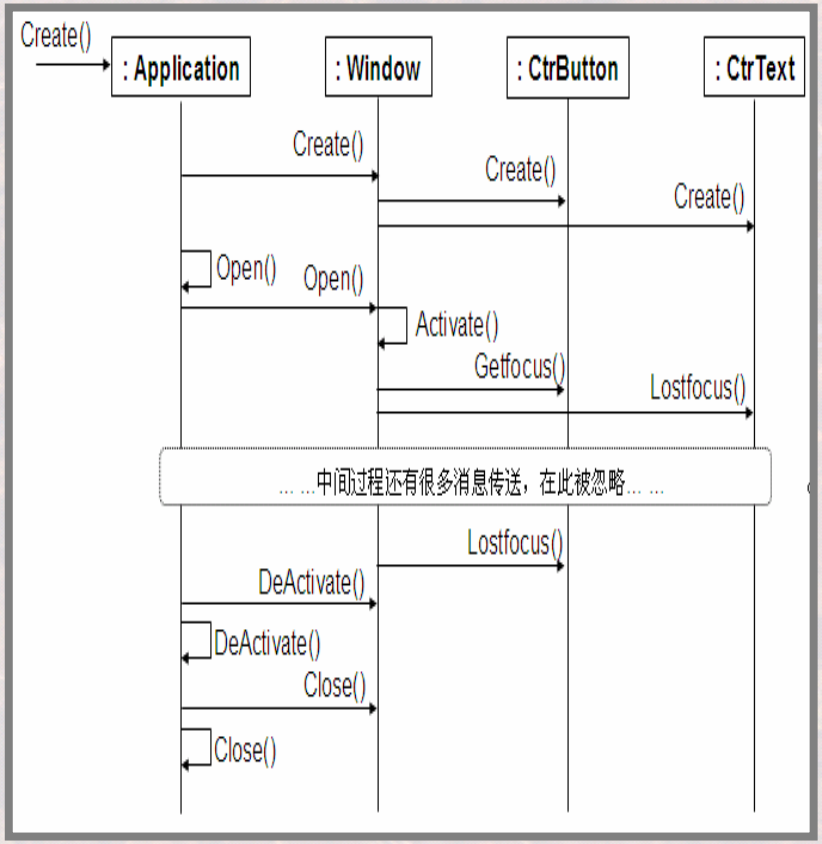


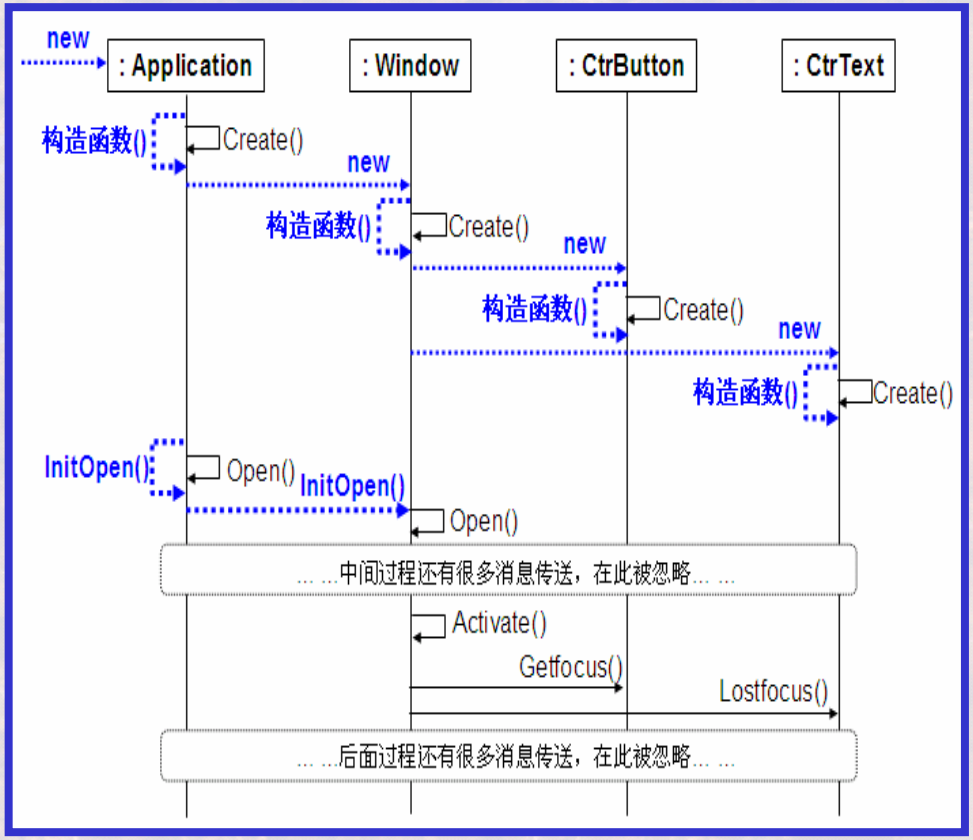
为什么没有 **anobject = new someCLASS;**

战德臣 教授

软件模式(Software Pattern)

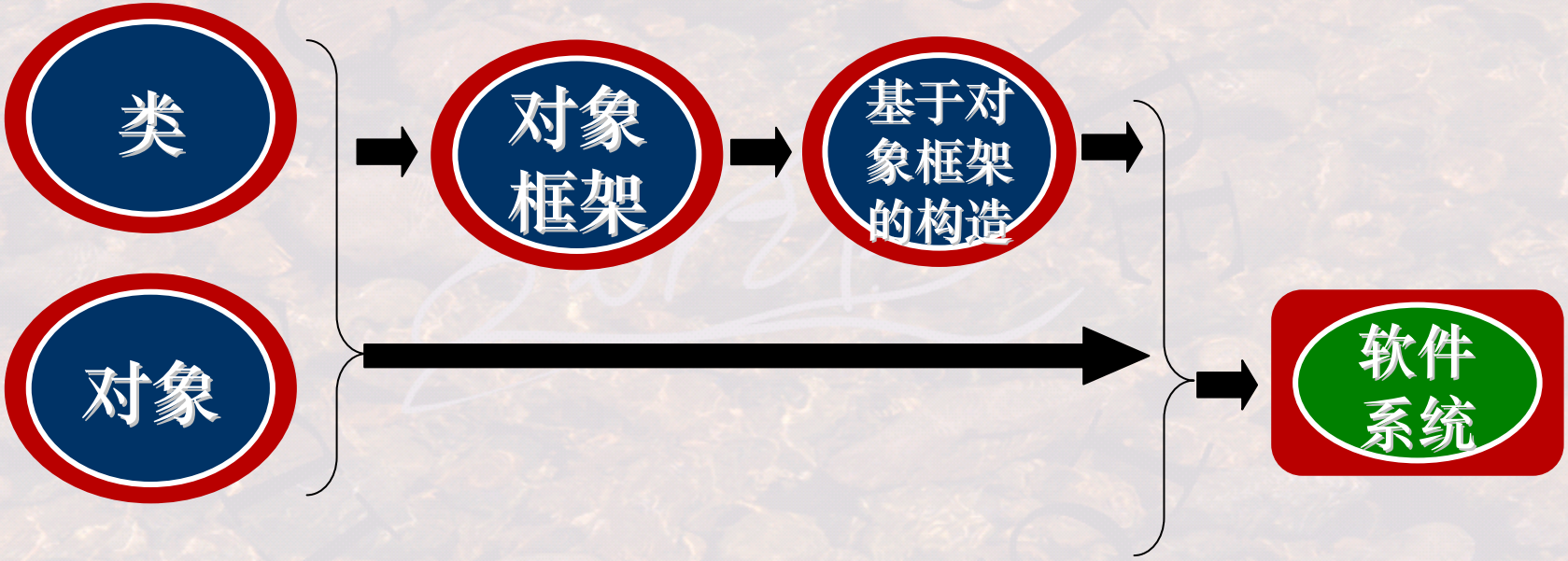
由面向对象的程序构造，到(可视化的)对象框架的构造





用面向对象思维构造对象框架
(8)小结

战德臣 教授



面向对象的程序设计

软件工程

UML

基于组件/构件的软件系统构造

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

基于组件/构件的软件系统构造
(1)为什么需要基于组件/构件的开发?

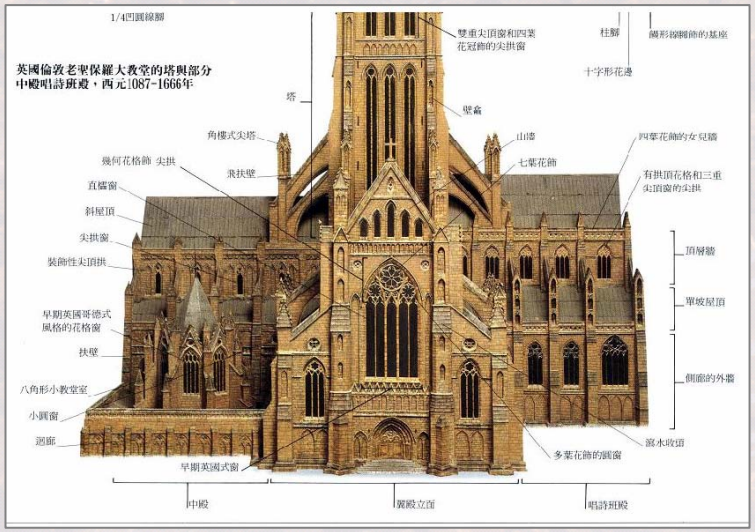
战德臣 教授

基于组件的软件构造

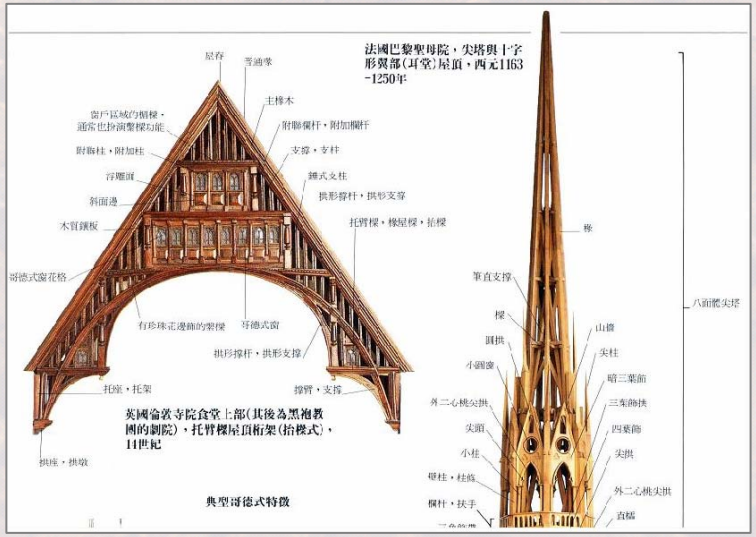
- 是解决复杂环境下软件规模与复杂性的一种手段
- 是通过分工实现标准化-流程化进而解决批量化软件生产的手段
- 使软件工程步向成熟发展的轨道

战德臣 教授

建筑中的构件与结构



- 基本的“构件”是什么？
- 这些“构件”怎样连接在一起？ “连接件”

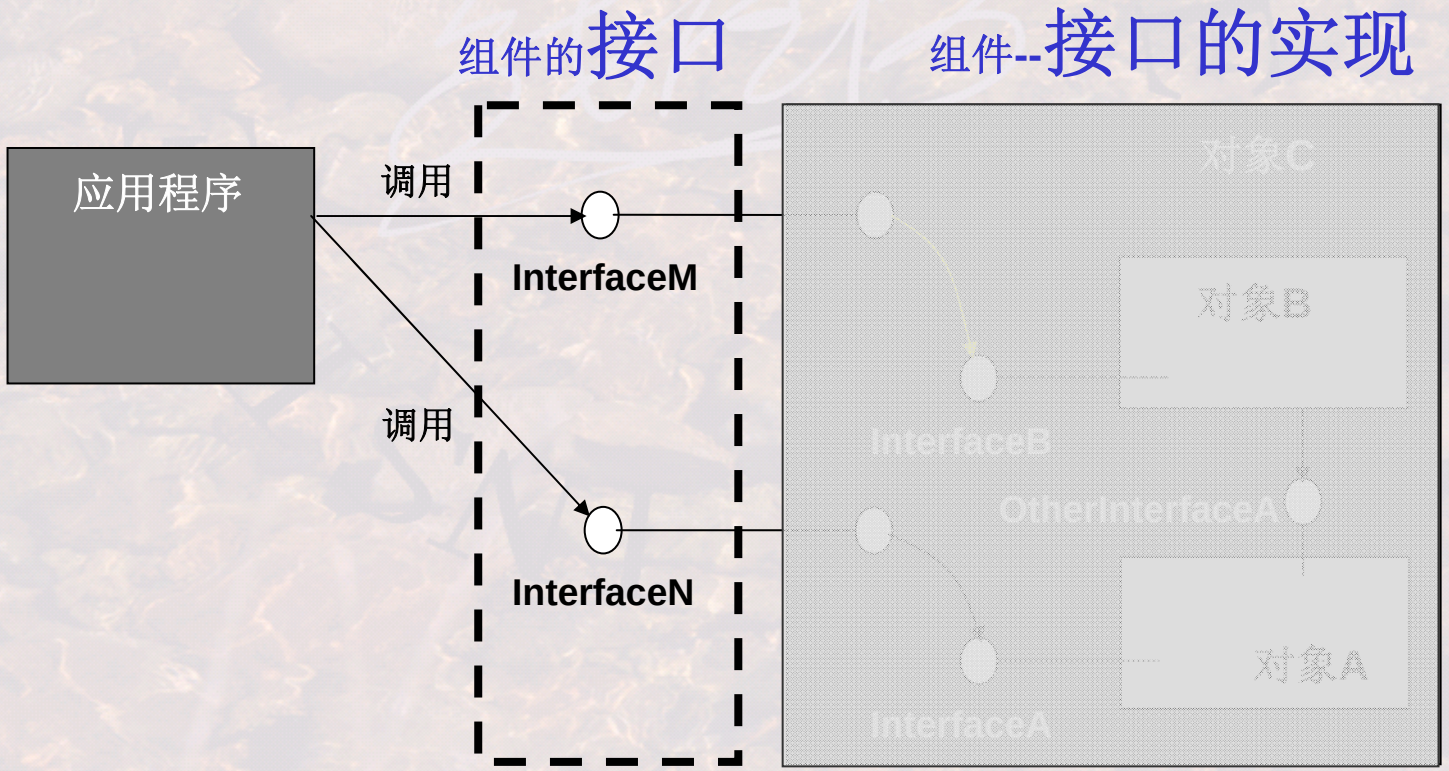


- “构件”如何来建造？
- “连接件”如何来建造和连接？

(2)什么是组件/构件？

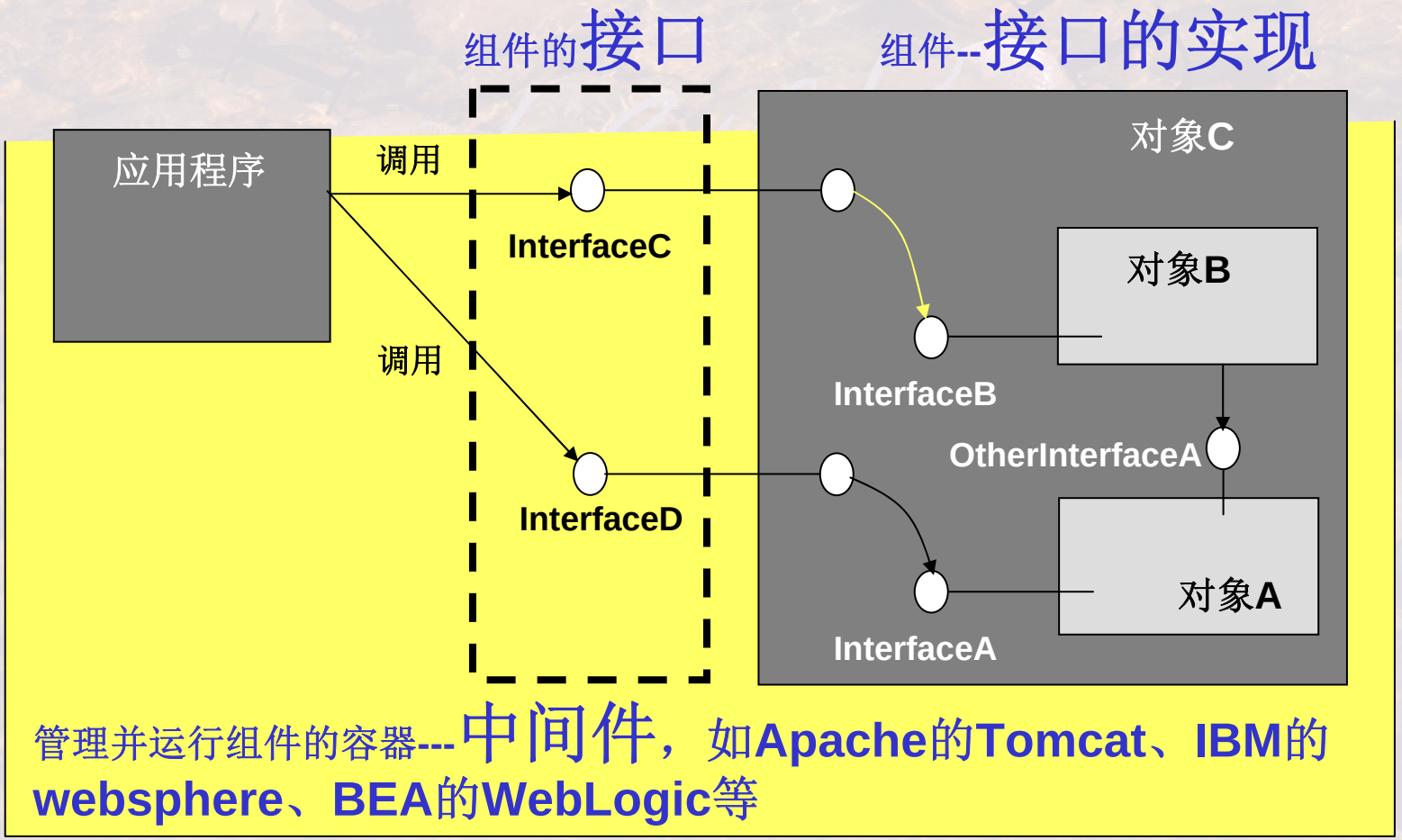
ICES 战德臣 教授

- **组件/构件**：可独立发布的机器级程序文件；将若干对象及其内部关系进行封装所形成的一个可执行程序文件；
- **“接口与实现分离”**---“接口”是对象或者组件的通信协议；“实现”是对象或者组件的内部构造细节。
- **解耦**—尽可能地消除软件之间或者软件的不同部分之间的联系



基于组件/构件的软件系统构造
(2)什么是组件/构件？

- 基于复用的软件开发 == 用可重复使用的软件组件开发应用程序
- 面向复用的软件开发 == 开发可重复使用的软件组件

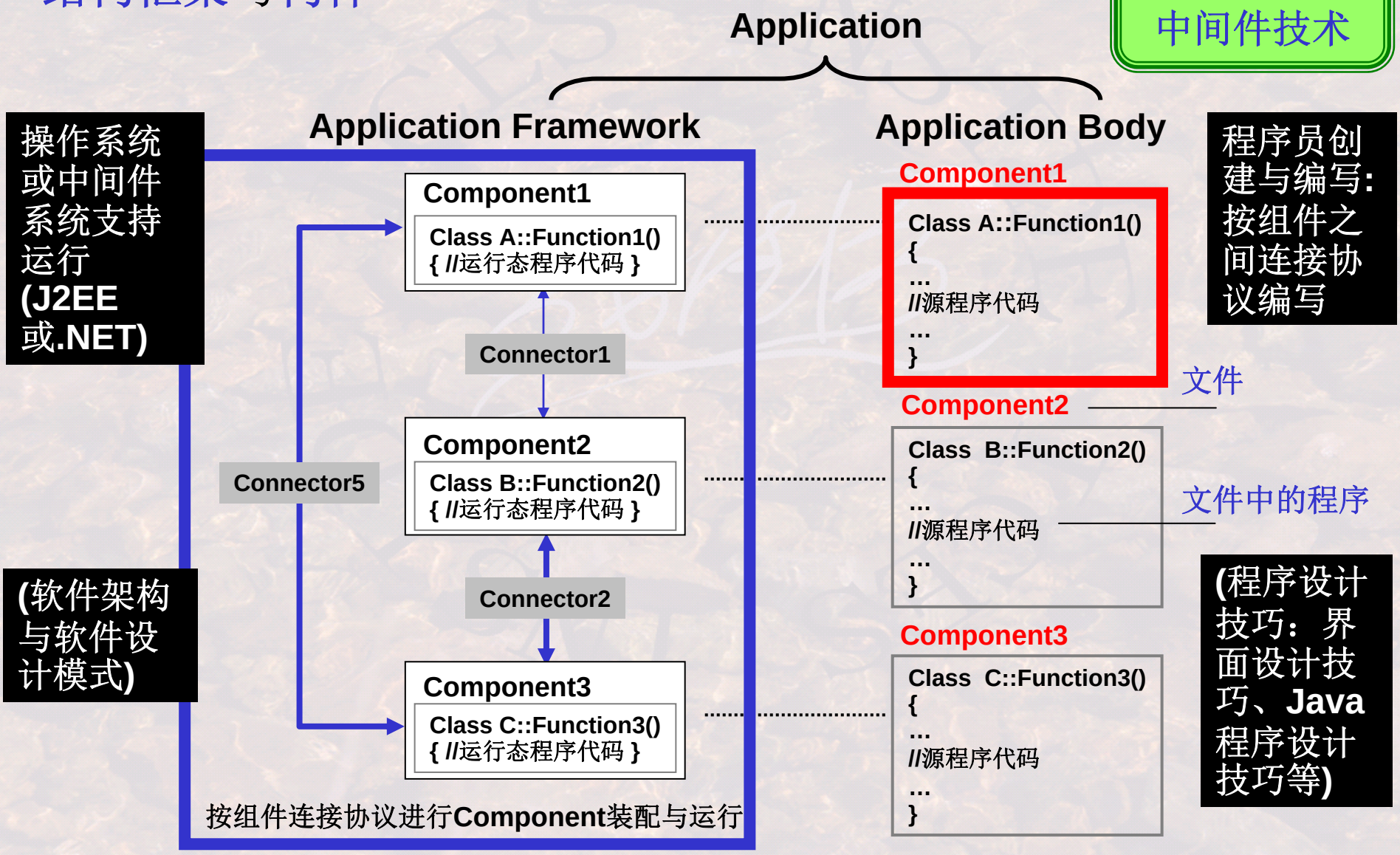


基于组件/构件的软件系统构造
(3)一般意义的结构框架与组件?

战德臣 教授

结构框架与构件

软件复用与中间件技术



基于组件/构件的软件系统构造
(4)一种典型的结构框架：组件及其之间的关系

战德臣 教授

界面

连接件(Connector)
程序员按连接约定书写；结构框架按连接约定给予连接实现

业务处理程序

```
case "入库单": In-Bill-Transfer(bill);
case "出库单": Out-Bill-Transfer(bill);
case "XX单据": XX-Bill-Transfer(bill);

In-Bill-Transfer(bill){
    account = account + transAccount;
}

Out-Bill-Transfer(bill){
    account = account - transAccount;
}
```

连接件(Connector)

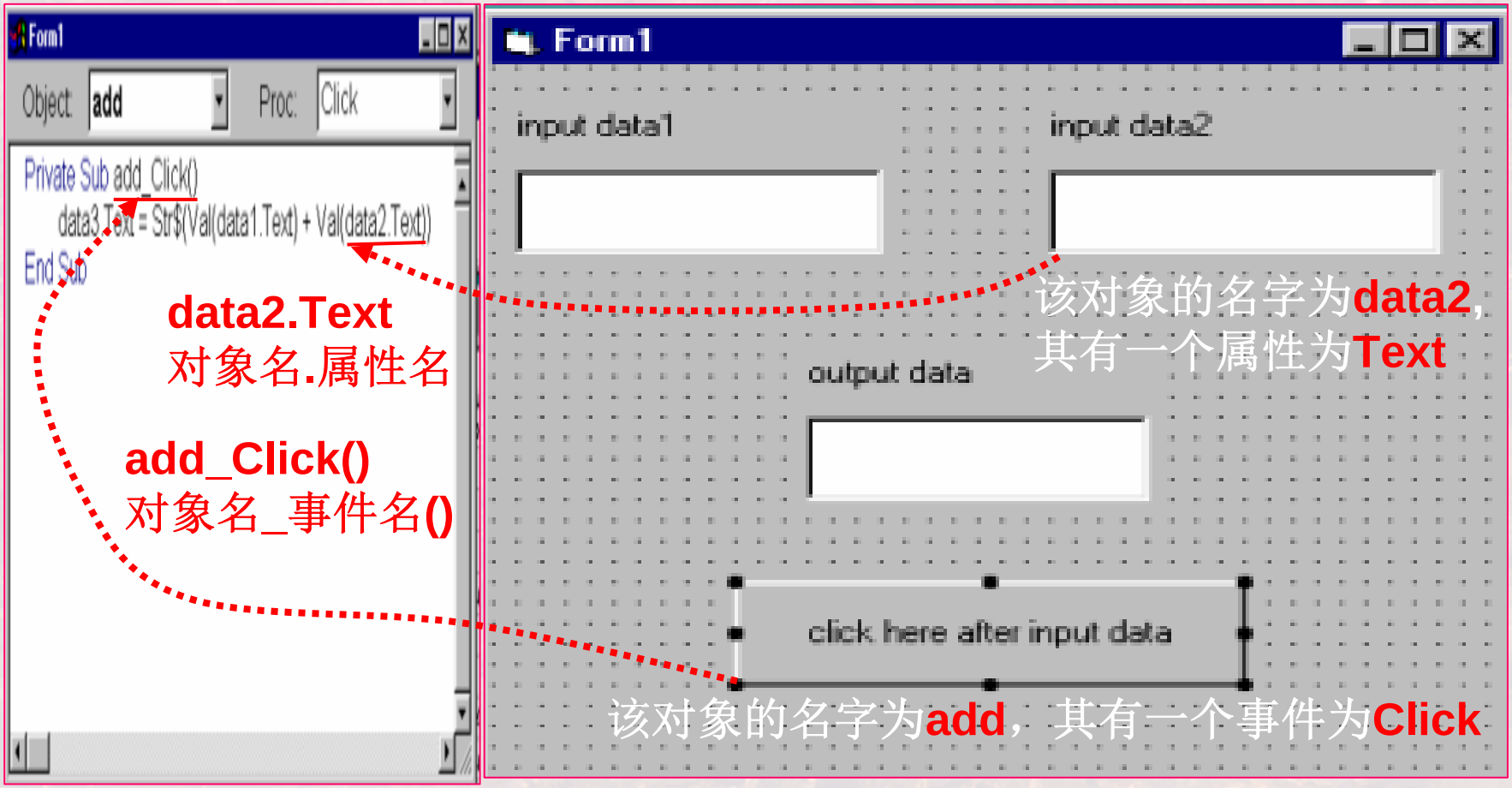
数据结构与数据存取

库存流水账

日期	物资	入/出	数量	单价	金额
2012.06.01	**M1**	**入**	**200**	**1,100.0000**	**220,000.00**
2012.06.02	M1	出	80	1,100.0000	88,000.00
2012.06.03	M1	出	80	1,100.0000	88,000.00
2012.06.05	**M1**	**入**	**200**	**900.0000**	**180,000.00**
2012.06.07	M1	出	80	933.3333	74,666.66
2012.06.09	M1	出	80	933.3333	74,666.66
2012.06.11	M1	出	80	933.3333	74,666.68

战德臣 教授

不同组件的连接示意：事件驱动程序 vs. 用户界面



业务处理程序 与 用户界面程序紧耦合在一起

战德臣 教授

不同组件的连接示意：事件驱动程序 vs. 用户界面

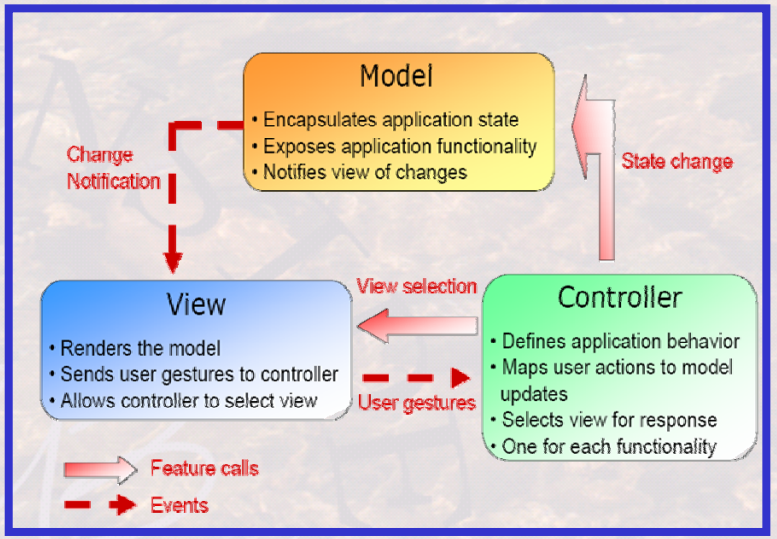

```
public class BillModel(Bill bill) {
    private BillModel billModel = null;
    ... ...
    transfer(bill){
        ... ...
        BillType = bill.billType;
        swich(BillType){
            case "入库单": In-Bill-Transfer(bill);
            case "出库单": Out-Bill-Transfer(bill);
            case "XX单据": XX-Bill-Transfer(bill);
        }
        ... ...
    }

    In-Bill-Transfer(bill){
        ... ...
        account = account + transAccount;
        ......
    }

    Out-Bill-Transfer(bill){
        ... ...
        account = account - transAccount;
        ... ...
    }
    ... ...
}
```

处理业务的各种程序，类及其中的函数

业务处理程序 与 用户界面 是分离的，但相互之间有耦合关系

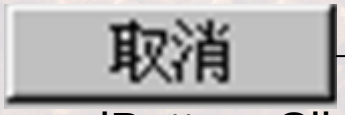
战德臣 教授

MVC框架: Model-View-Controller

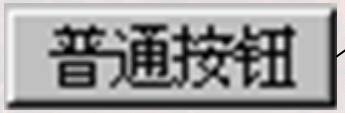


View

OKButton_Click



CancelButton_Click



GenButton_Click

界面要素及其操作

Controller

```
Case: OKButton_Click
      Process1();
Case: CancelButton_Click
      Process2();
Case: GenButton_Click
      Process3();
```

界面要素及其操作 与
业务处理程序 的 映射

Model

```
Process1()
{ … }

Process2()
{ … }

Process3()
{ … }
```

业务处理程序

战德臣 教授

MVC框架: Model-View-Controller

View：界面

(a)用户界面组件: 用户界面及其背后的程序示意图

```
public class BillController{
    private BillController billController = null;
    ... ...
    public void iAccount(Bill bill) {
        try {
            this.billModel.transfer(bill);
            putMessage("过账成功！");
        } catch (Exception e) {
            e.printStachTrace();
            putError("过账失败："+e.getMessage());
    }
    ... ...
}
```

将用户界面的"过账"按钮 与 过账的处理程序 BillModel.transfer() 连接起来

(b)控制组件: 连接用户界面组件与业务逻辑组件的程序

```
public class BillModel(Bill bill) {
    private BillModel billModel = null;
    ... ...
    transfer(bill){
        ... ...
        BillType = bill.billType;
        swich(BillType){
            case "入库单": In-Bill-Transfer(bill);
            case "出库单": Out-Bill-Transfer(bill);
            case "XX单据": XX-Bill-Transfer(bill);
        }
        ... ...
    }

    In-Bill-Transfer(bill){
        ... ...
        account = account + transAccount;
        ------
    }

    Out-Bill-Transfer(bill){
        ... ...
        account = account - transAccount;
        ... ...
    }
    ... ...
}
```

按不同单据类别进行过账处理的程序类 BillModel

类中的函数 Transfer() 识别单据类别并调用相应的过账函数

入库过账的函数 In-Bill-Transfer()

出库过账的函数 Out-Bill-Transfer()

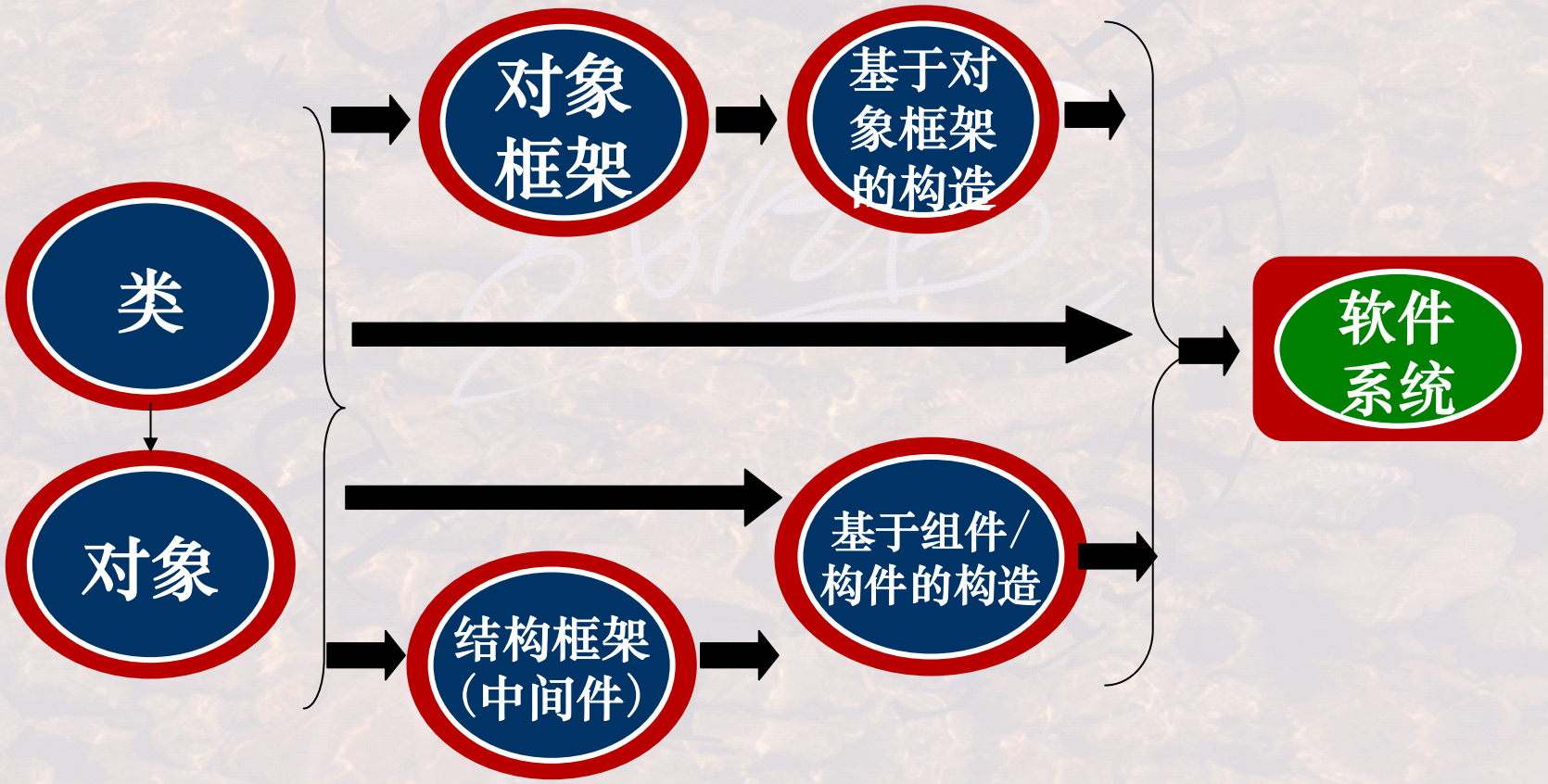
(c)业务逻辑组件: 处理各种

Model：程序

Controller：界面要素与程序的映射

业务处理程序 与 用户界面 是分离的，相互之间通过Controller联系和映射

基于组件/构件的软件系统构造
(8)小结



面向服务的软件系统构造方法

战德臣

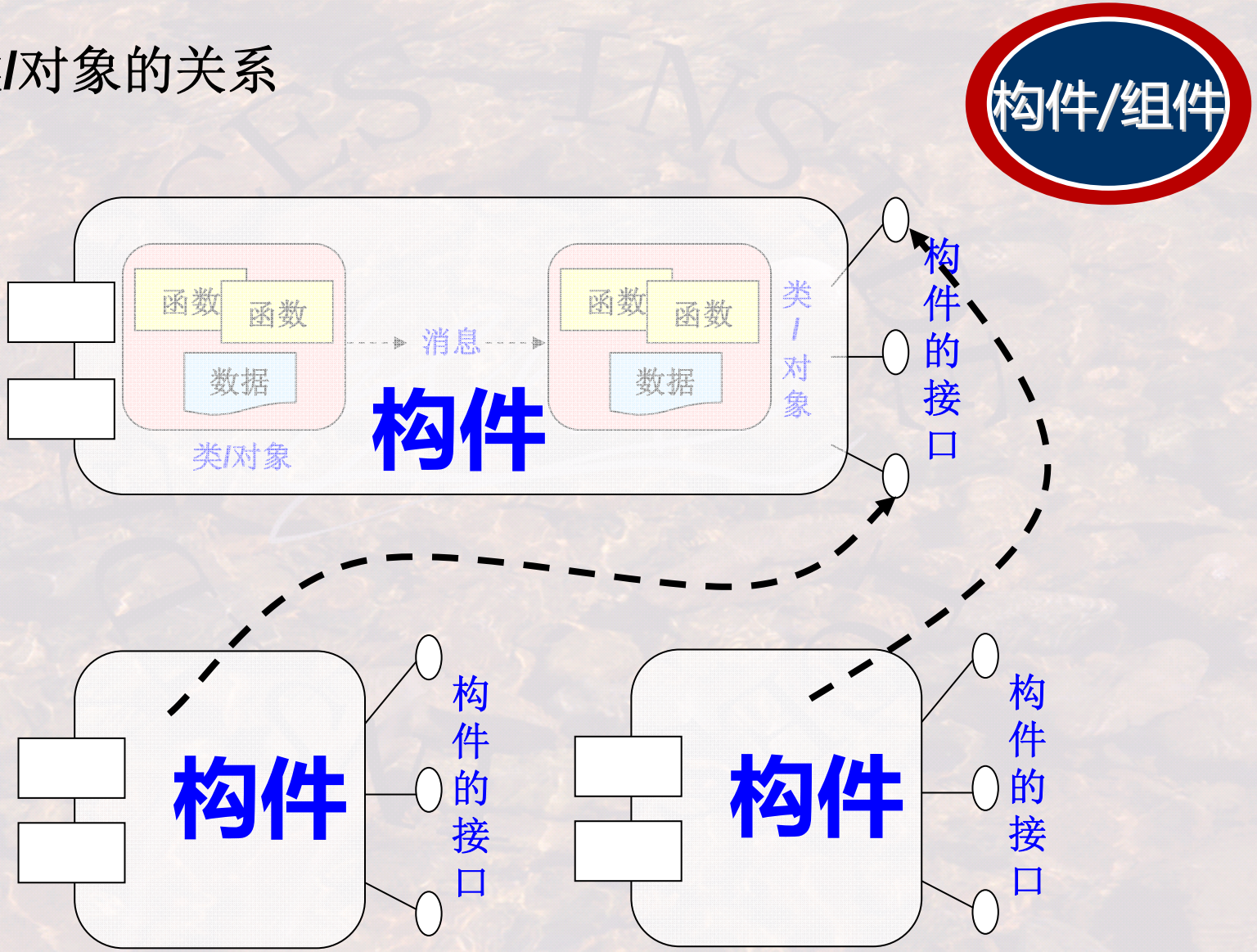
哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

战德臣 教授

构件与类/对象的关系



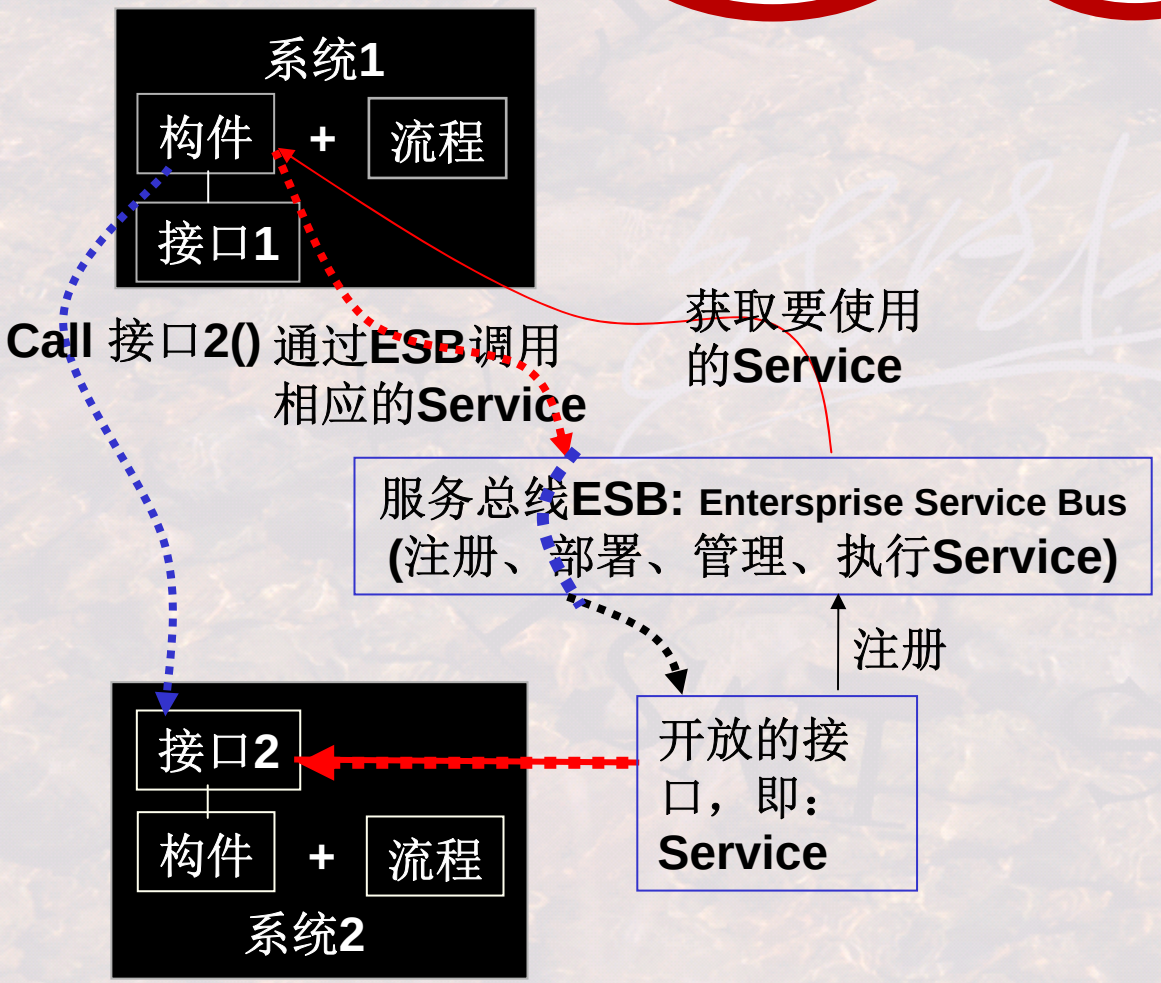
面向服务的软件系统构造方法
(2)由“构件”到“服务”?

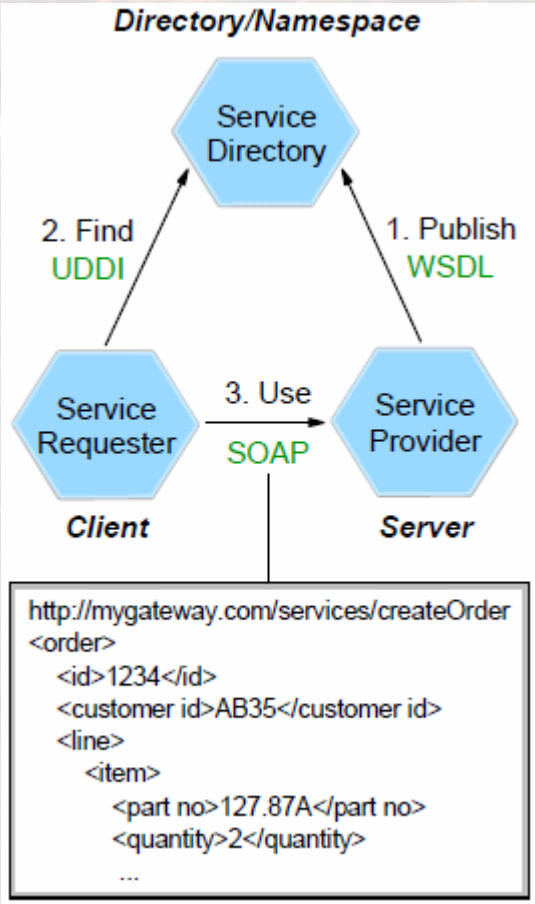
战德臣 教授

Interface

Web Service

服务计算

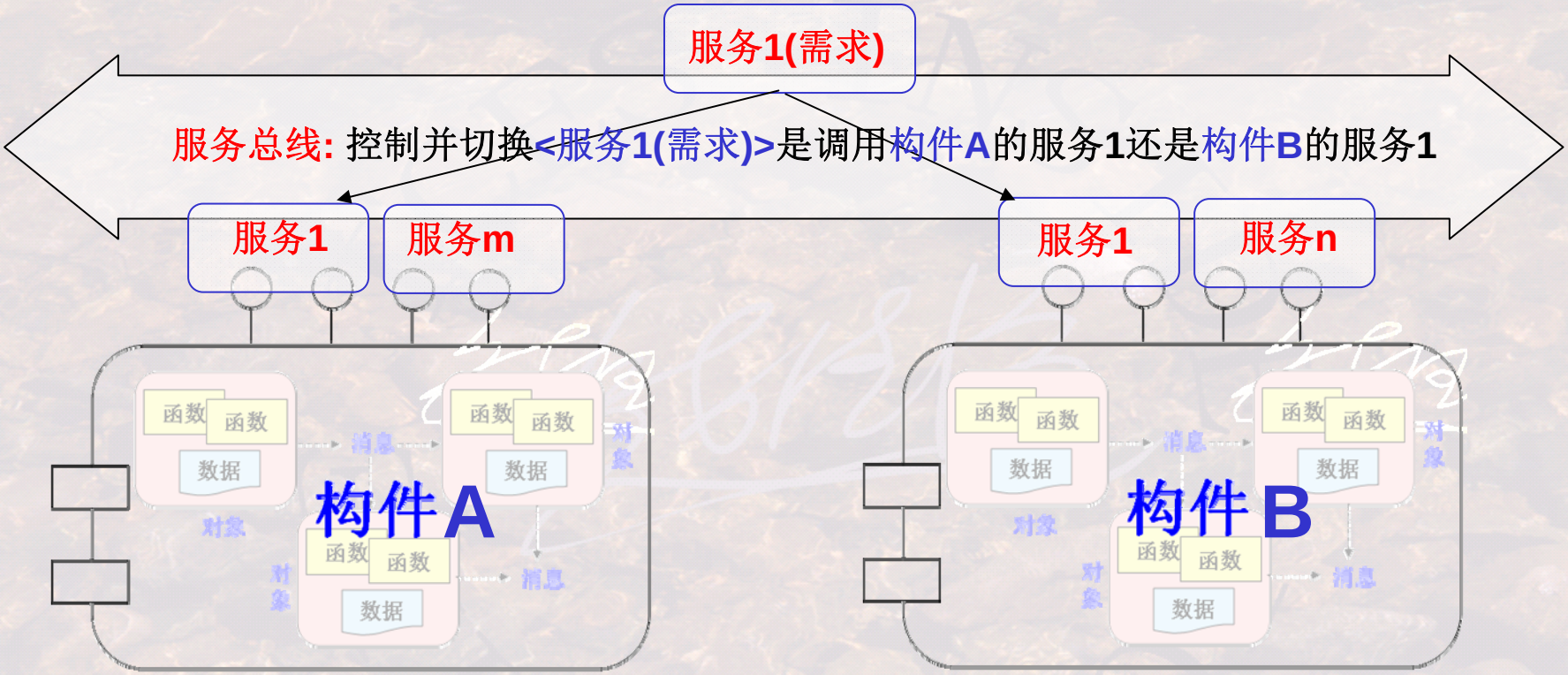




Web Service

面向服务的软件系统构造方法
(3)面向服务的软件系统构造?

战德臣 教授



系统 = 服务 + 服务总线

服务= 构件的公共标准的接口

系统 = 对象 + 消息

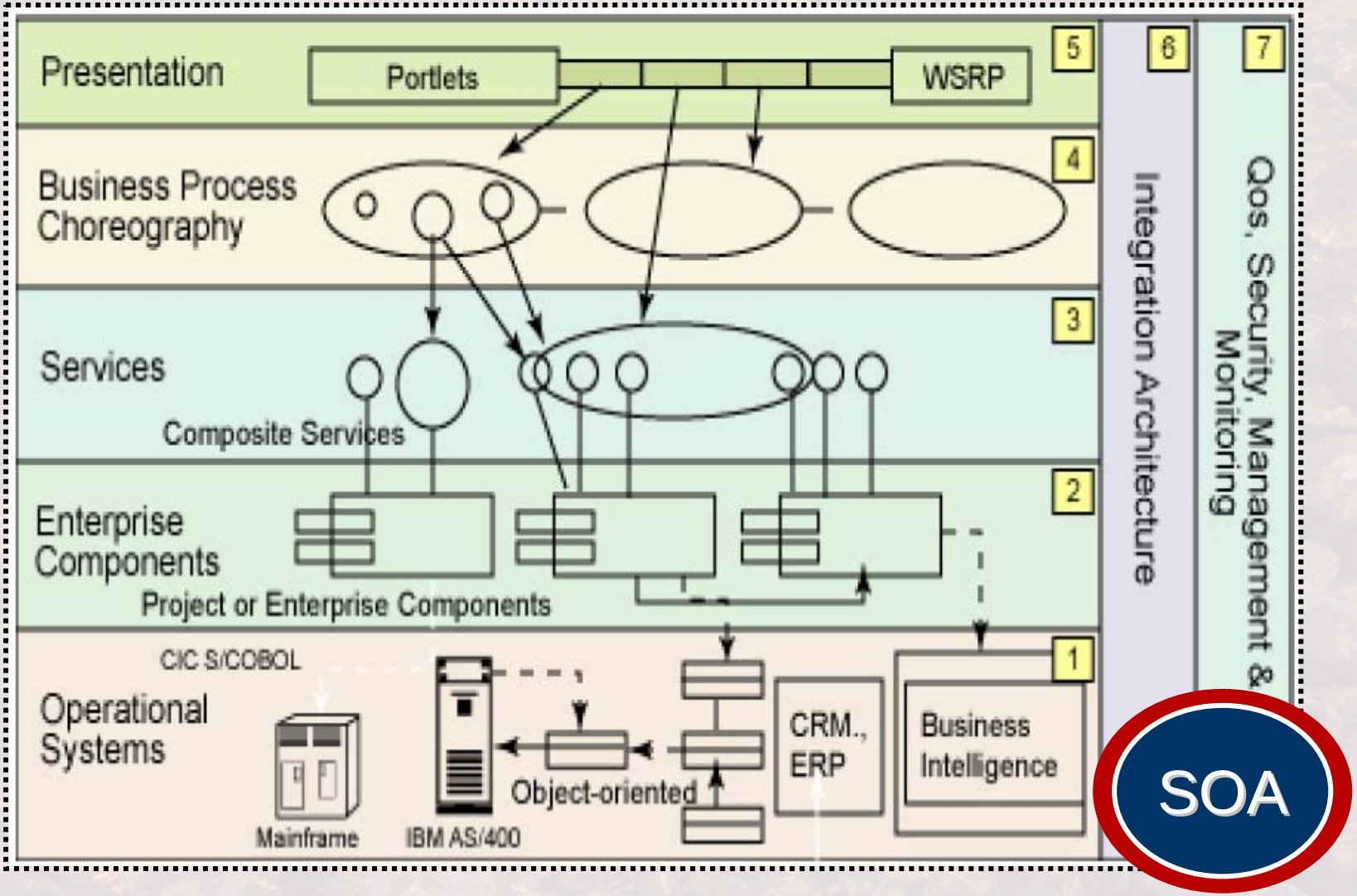
系统 = 构件 + 连接件

城镇与城市的构建

如：各建筑之间的互连互通?

战德臣 教授

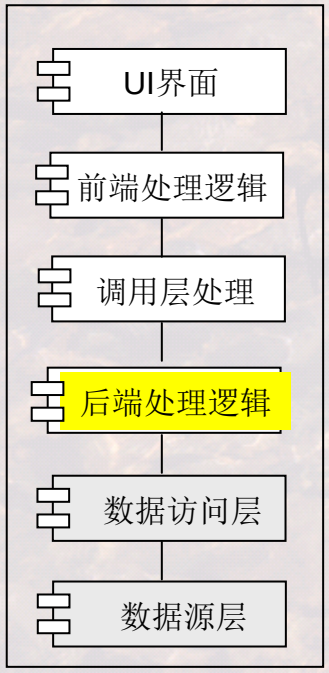
面向服务的体系结构: SOA—Service Oriented Architecture



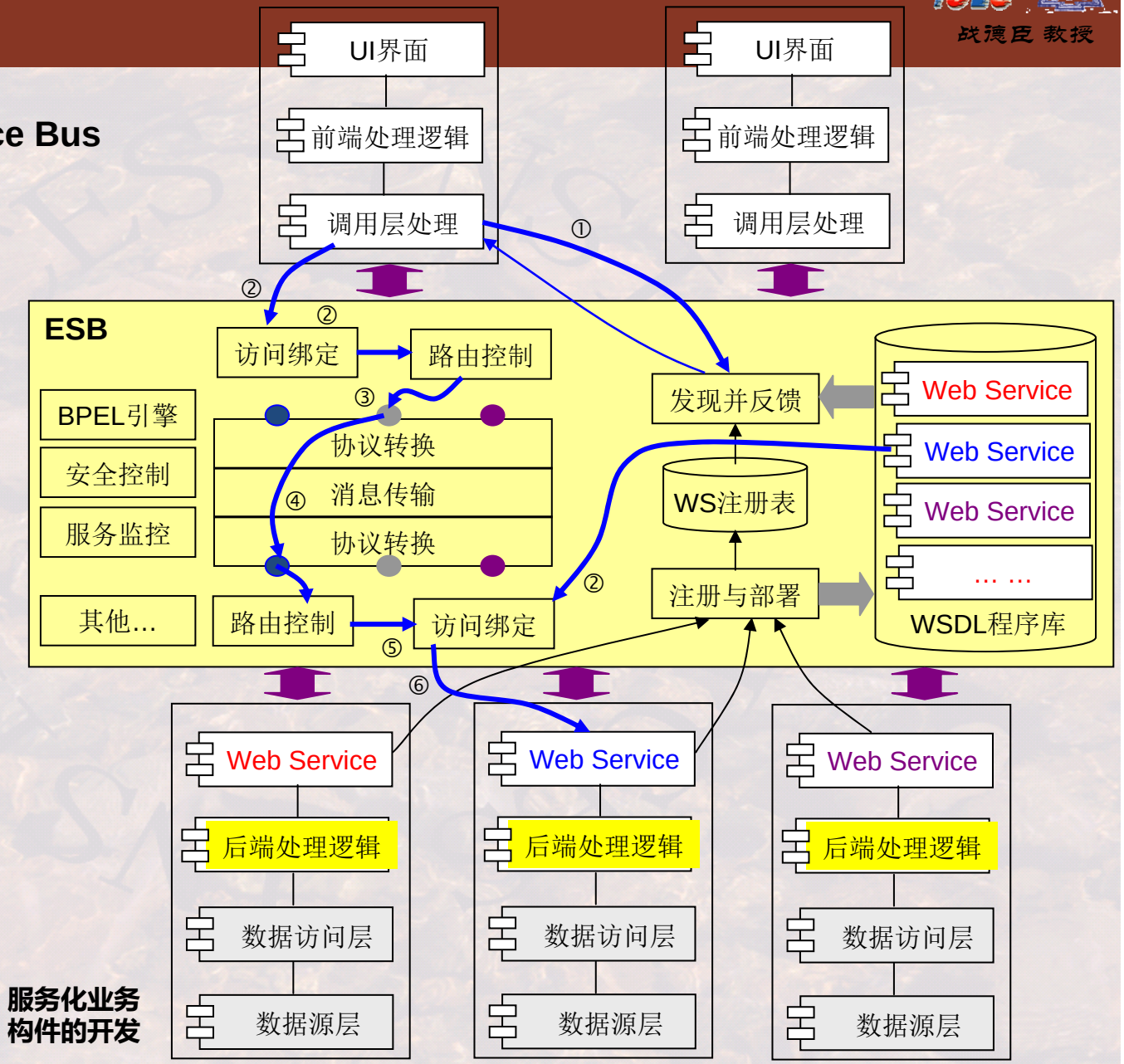
面向服务的软件系统构造方法
(4)SOA & ESB?

ICES 战德臣 教授

ESB: Enterprise Service Bus

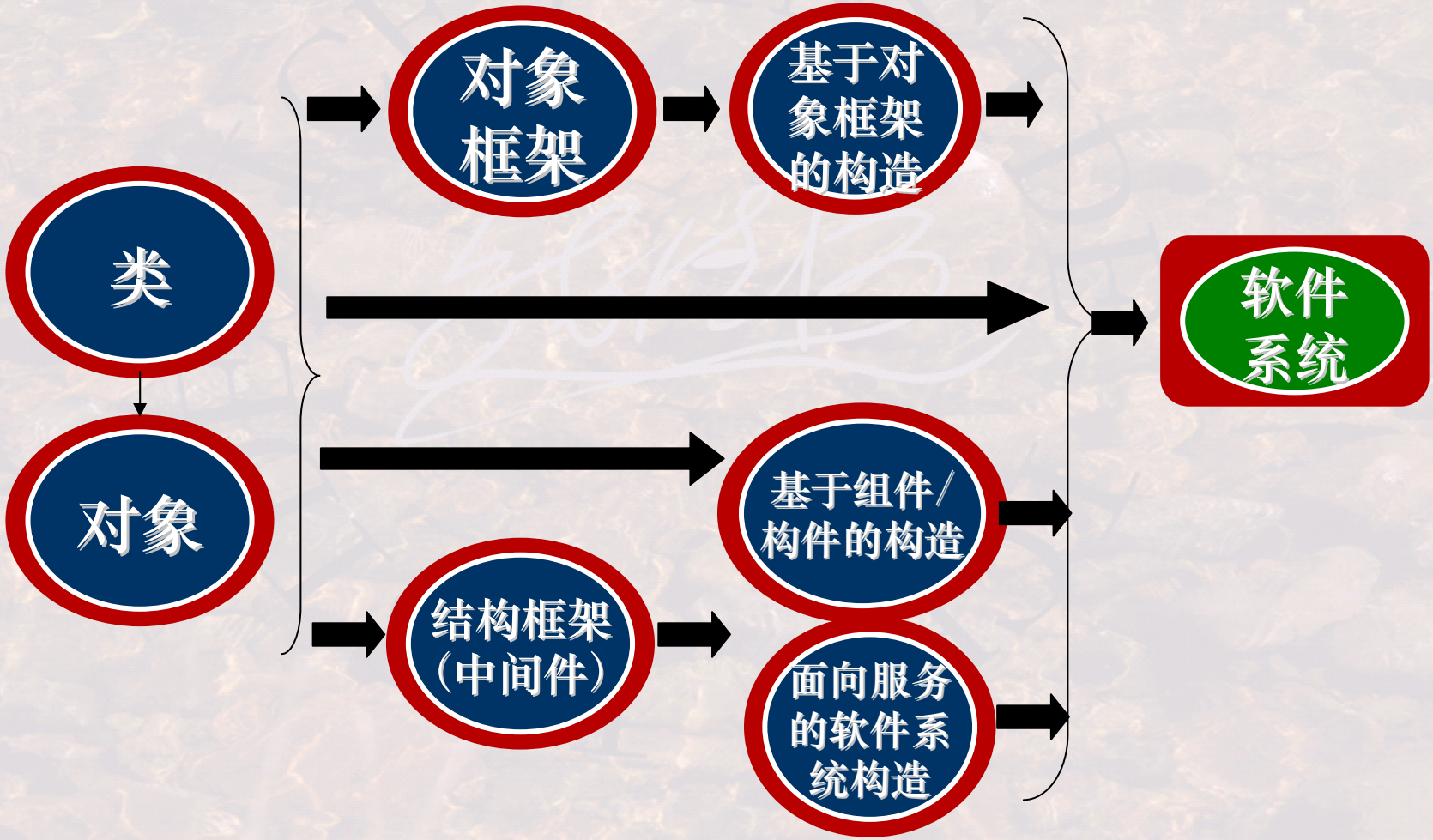


传统业务构件的开发



服务化业务构件的开发

面向服务的软件系统构造方法
(6)小结



软件系统构造方法的演变

战德臣

哈尔滨工业大学　教授.博士生导师

教育部大学计算机课程教学指导委员会委员

Research Center on Intelligent Computing for Enterprises & Services, Harbin Institute of Technology

软件系统构造方法的演变
(1)结构化程序构造?

战德臣 教授

60's, 70's年代的软件: 结构化程序

编程语言：**C语言，Fortran语言**

简单的平房

主函数()

数据结构与数据

算法（的实现）

```
#include <stdio.h>
#define  n  4
main(){
  int   D[n][n], S[n], Sum, I, j, K, L, Dtemp, Found;
   S[0]=0;  Sum=0;   D[0][1]=2;  D[0][2]=6;  D[0][3]=5;  D[1][0]=2;  D[1][2]=4;
   D[1][3]=4;  D[2][0]=6;  D[2][1]=4;  D[2][3]=2;  D[3][0]=5;  D[3][1]=4;  D[3][2]=2;
   I=1;                    //注意程序是从 0 开始，流程图是从 1 开始的，下同.
   do {                    //I 从 1 到 n-1 循环
      K=1;  Dtemp=10000;
      do{                  //K 从 1 到 n-1 循环
         L=0;  Found=0;
         do{                  //L 从 1 到 I 循环
            if(S[L]==K)
            {  Found=1;  break;  }
            else L++;
          } while(L<I);       //L 从 1 到 I 循环
         if(Found==0 && D[K][S[I-1]]<Dtemp)
         {  j=K;  Dtemp=D[K][S[I-1]];  }
         K++;
      } while(K<n);          //K 从 1 到 n-1 循环
      S[I]=j;  I=I+1;  Sum=Sum+D[I][j];
   } while(I<n);          //I 从 1 到 n-1 循环
  Sum=Sum+D[I][j];
  for(j=0;j<n;j++){  printf("%d,",S[j]);  }  //输出城市编号序列
  printf("\n");                              //换行
  printf("Total Length:%d",Sum);             //输出总距离
}
```

函数f1(){ … }

函数f2(){ … }

系统 = 算法 + 数据结构(1960's)

系统 = 子程序/函数 + 函数调用 (1980's)

软件系统构造方法的演变
(2)面向对象的程序构造?

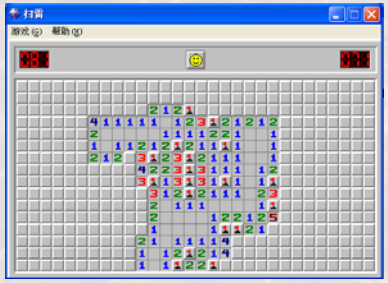
战德臣 教授

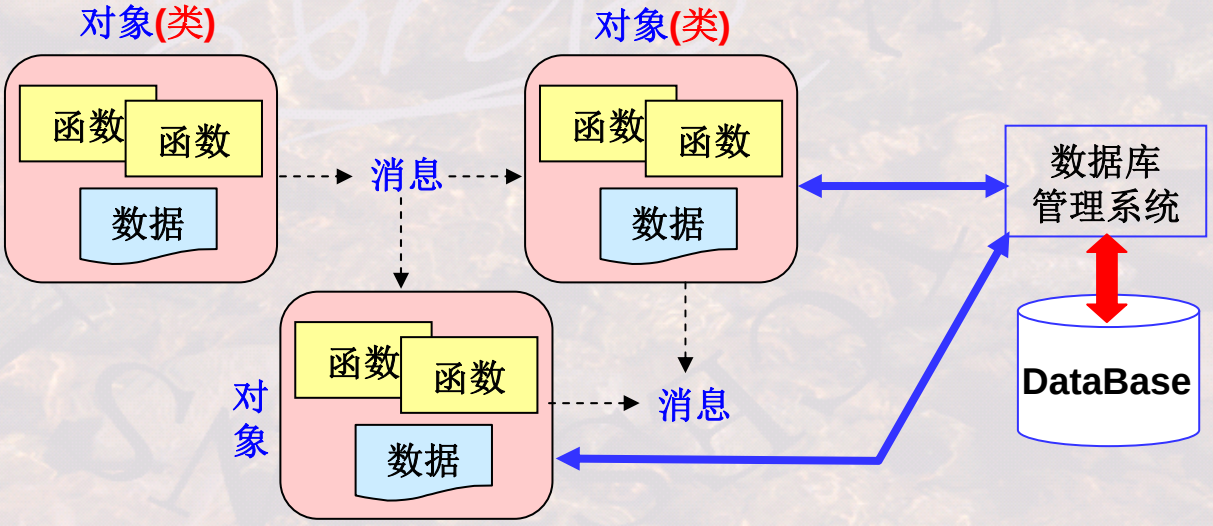
80's年代的软件: 面向对象的程序

编程语言：**C++**(83)，**Java**(95)，**Visual** 系列语言**(**90)等

复杂的特色建筑

如：巨大拱顶和圆顶如何建设?





对象 = 函数 + 数据结构

系统 = 对象 + 消息 **(1980's)**

软件系统构造方法的演变
(3)构件化的程序构造?

战德臣 教授

90's年代的软件: 构件化系统

编程语言：**Visual系列语言；Windows操作系统**

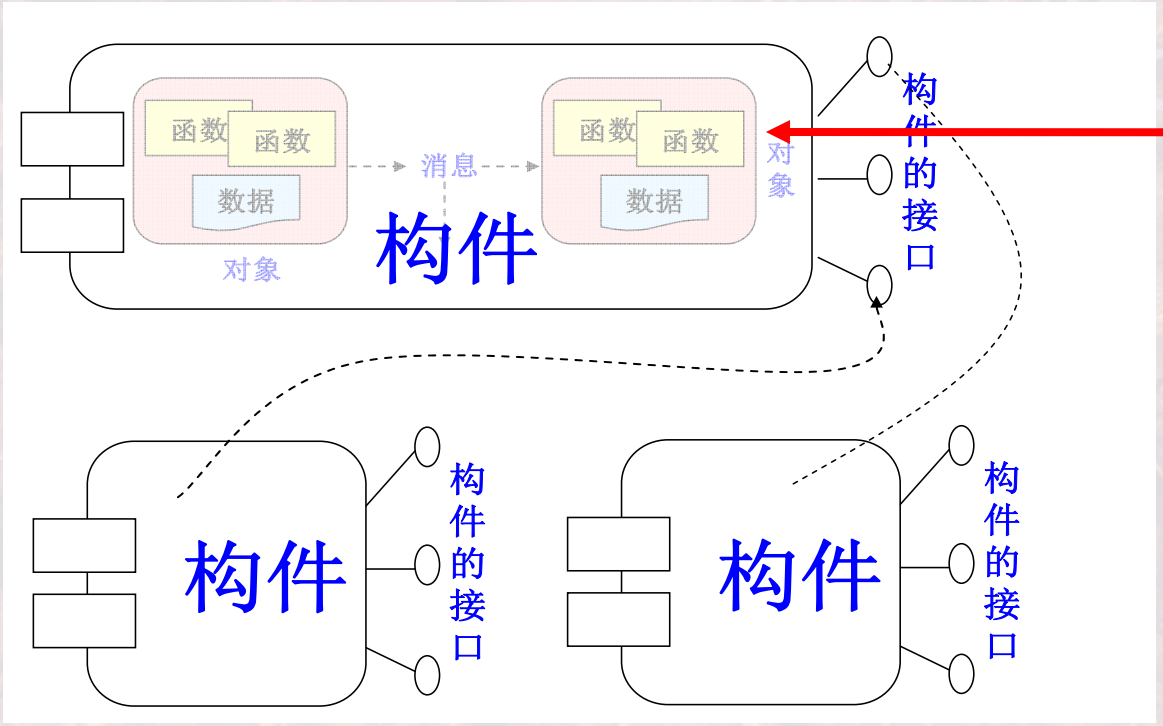
构件 = 对象 + 消息

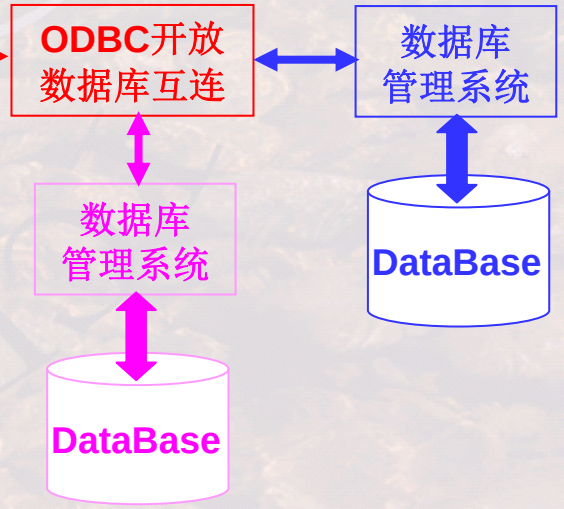
构件 = 实现体 + 接口

系统 = 构件 + 连接件 **(1990's)**

现代化复杂的高楼

如：地基、承重、结构怎样?





软件系统构造方法的演变
(3)构件化的程序构造?

战德臣 教授

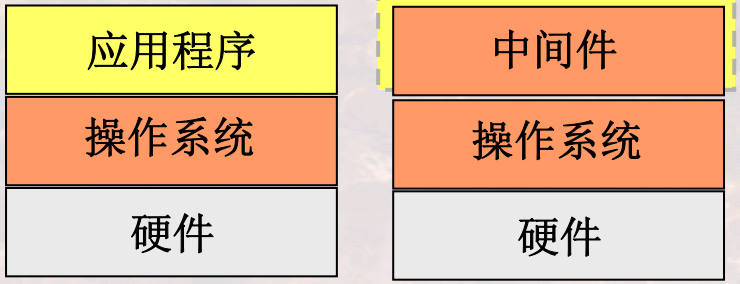
90's年代的软件: 构件化系统

基于体系结构与设计模式的软件开发

- **J2EE: Springs, Rails**等
- **.Net**
- **CORBA**

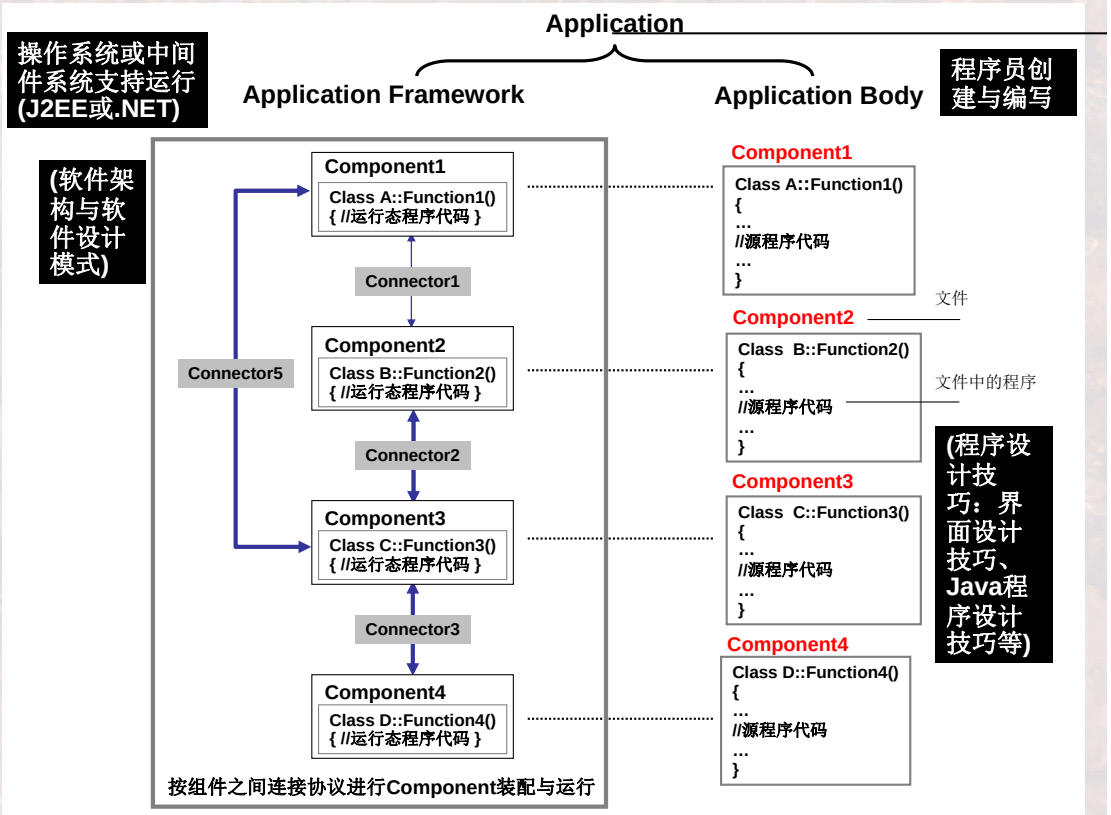
中间件予以支撑。

- **Web logic**(BEA)
- **Web Sphere** (IBM)
- **Tomcat**(开源)
- …



现代化复杂的高楼

如：地基、承重、结构怎样?



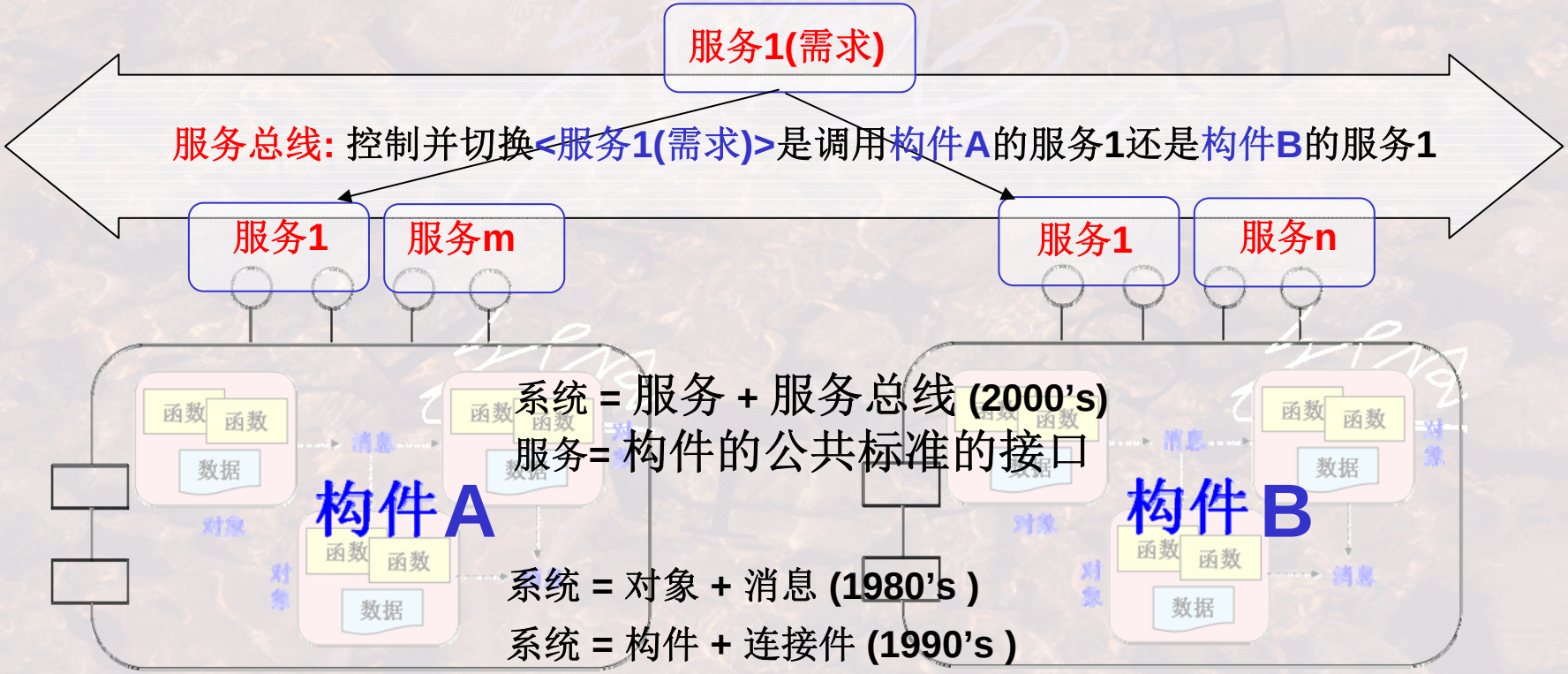
软件系统构造方法的演变
(4)服务化的程序构造?

战德臣 教授

2000's年代的软件: 面向服务的体系结构SOA ➔ 基于Internet的软件开发与云计算体系结构

城镇与城市的构建

如：各建筑之间的互连互通?



软件系统构造方法的演变
(5)软件系统构造方法的演变过程?

战德臣 教授

简单的平房

结构化程序：
算法+数据结构
子程序+函数调用

复杂的特色建筑

如：巨大拱顶和圆顶如何建设?

对象化程序：
对象+消息

现代化复杂的高楼

如：地基、承重、结构怎样?

构件化程序:
构件+体系结构

城镇与城市的构建

如：各建筑之间的互连互通?

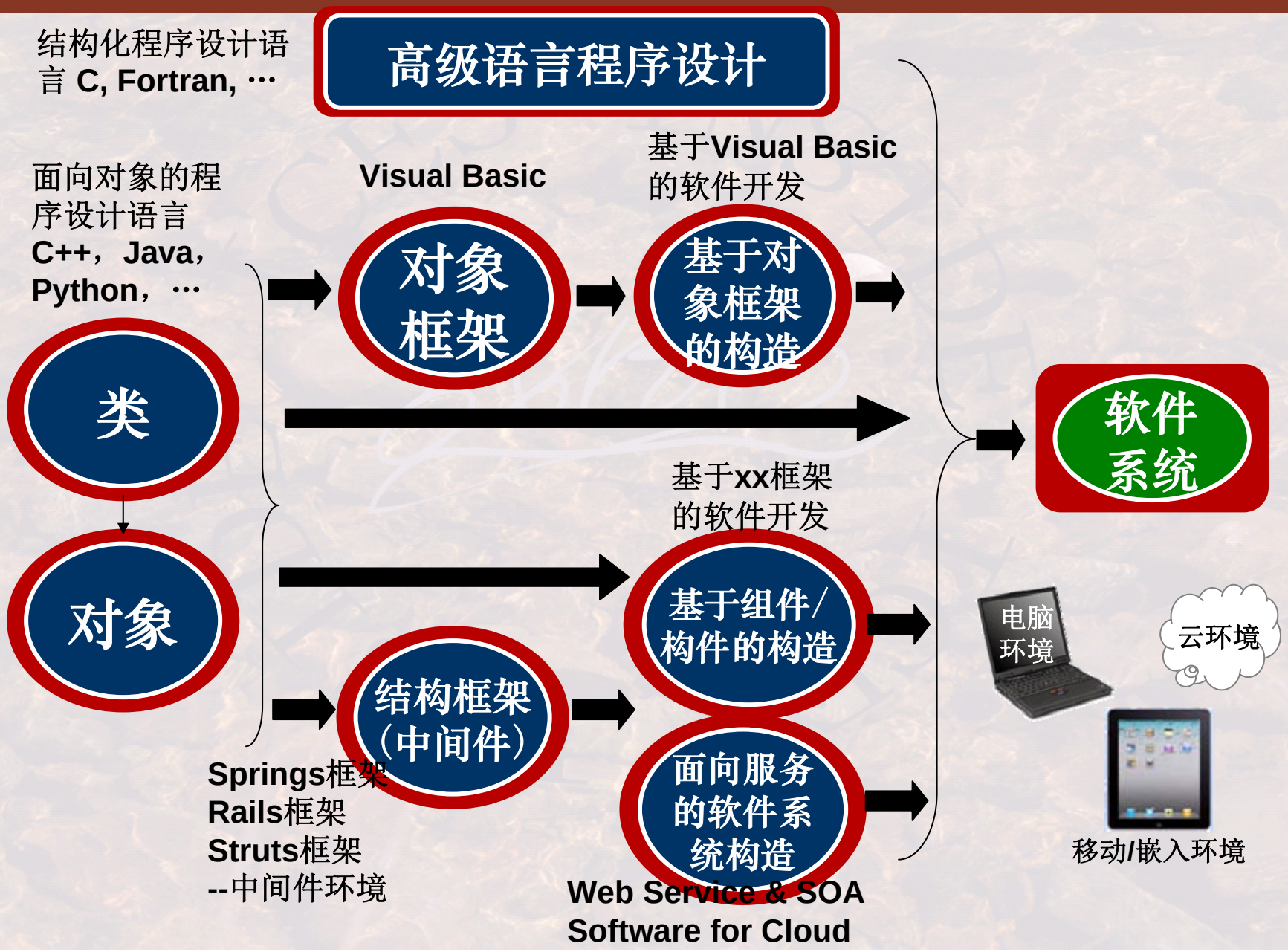
服务化程序：
服务+服务总线

不能倒塌是第一位的

软件的可靠性、可用性等

软件系统构造方法的演变
(6)软件系统构造方法一览

战德臣 教授



软件系统构造方法的演变
(7)软件系统构造能力的发展路线图

战德臣 教授

高级语言程序设计与算法

基于框架的(大型)系统开发

设计新的开发框架供别人使用

大型软件的架构/模式设计和实现