




第4章 总体设计

• 哈尔滨工业大学 •
电气工程及自动化学院


主讲：刘晓胜 博士
 联系方式：0451-86402387
 Email: liuxsh2004@126.com



 哈尔滨工业大学 电力电子与电力传动实验室
 HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.1 总体设计的任务


- 总体设计又称为概要设计或初步设计；
- 开发阶段的开始：“怎样做？”
- 基本目的：“概括地说，系统如何实现？”
- 确定系统中每个程序由哪些模块组成以及这些模块相互间的关系。



 哈尔滨工业大学 电力电子与电力传动实验室
 HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.1 总体设计的任务


- 在总体设计阶段，应划分出组成系统的物理元素：
 - ◆ 划分物理元素：
 - 物理元素内容包括：程序/文件/数据库/人工过程和文档等
 - 物理元素分解程度：黑盒子级；
 - ◆ 确定软件结构：
 - 整个软件的程序组成；


 哈尔滨工业大学 电力电子与电力传动实验室
 HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.1 总体设计的任务

- 每个程序的模块组成；
- 模块之间的关系。
- 总体设计的必要性
 - 全局性分析，选最佳方案和最合理的软件结构；
- 总体设计分两个阶段：
 - 系统设计：确定系统的具体实现方案
 - 结构设计：确定软件结构；


 哈尔滨工业大学 电力电子与电力传动实验室
 HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.2 总体设计的过程

- 实施总体设计的过程如下
 - 1. 设想供选择的方案
 - 2. 选取合理的方案
 - 3. 推荐最佳方案
 - 4. 功能分解


 哈尔滨工业大学 电力电子与电力传动实验室
 HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.2 总体设计的过程

- 5. 设计软件结构
- 6. 数据库设计
- 7. 制定测试计划
- 8. 书写文档
- 9. 审查和复审


 哈尔滨工业大学 电力电子与电力传动实验室
 HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.2 总体设计的过程

1. 设想供选择的方案

- 出发点：需求分析阶段得出的数据流图；
- 常用方法：数据流图中的处理分组；
- 设想、列出方案，但不评价；

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.2 总体设计的过程

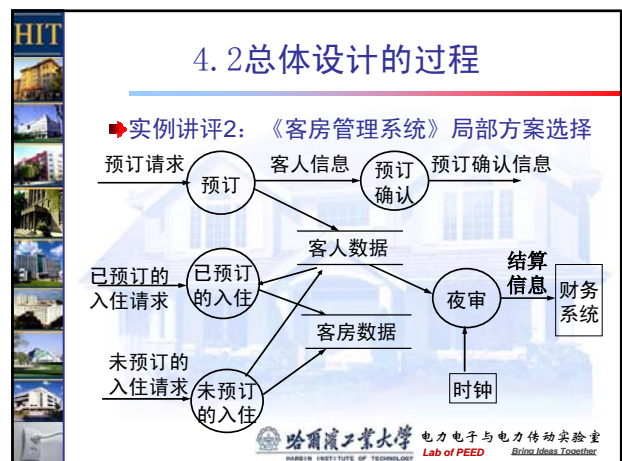
[1] 输入数据类型的选择

- unsigned para1, para2, output;
- unsigned para1, para2; int output;
- int para1, para2, output;
- int para1, para2; long int output;

[2] 试题数据保存方法的选择

- 选用数组临时保存st_para1[100], st_para2[100], st_output[100];
- 选用Delphi自带数据库;

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.2 总体设计的过程

[1] 预定请求类型的选择

- 只接受电话预订;
- 只接受网上预订;
- 只接受上门预定;
- 接受上述三种方案的任意组合预定;

[2] 夜审时间与餐费列入方法的确定

- 中午12点;
- 早晨8点;
- 餐费列入住宿费;
- ...

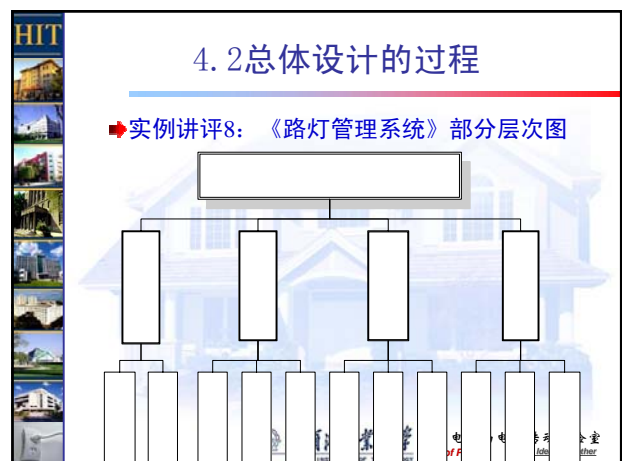
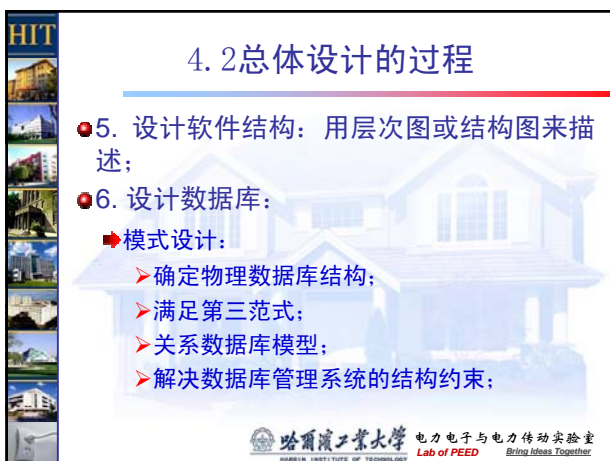
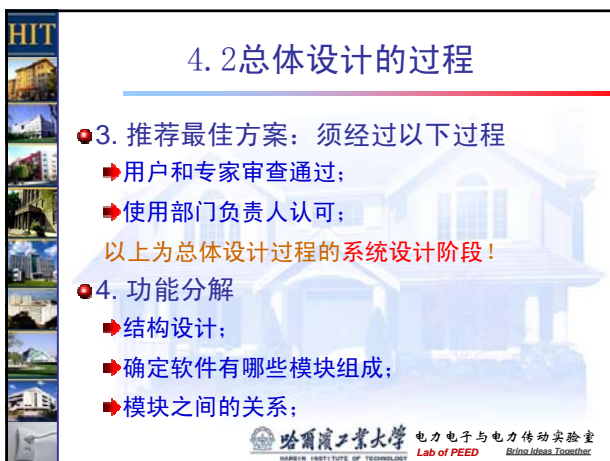
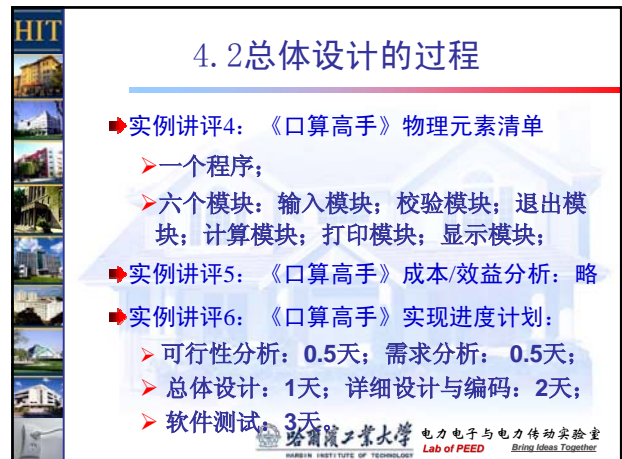
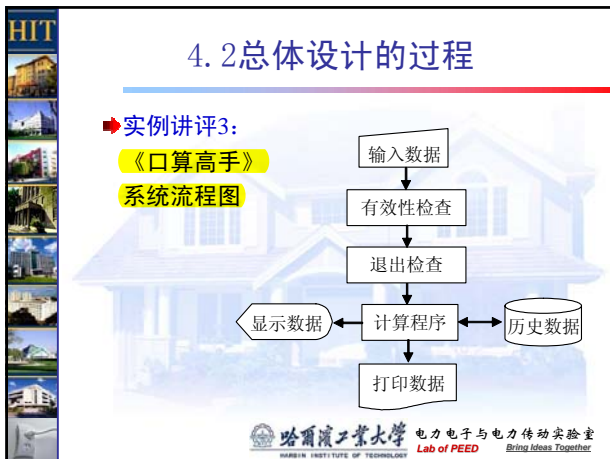
哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.2 总体设计的过程

2. 选取合理的方案：系统分析员提供


- 系统流程图;
- 组成系统的物理元素清单;
- 成本/效益分析;
- 实现这个系统的进度计划;

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.2 总体设计的过程

- 实例讲评9: 《路灯管理系统》数据库结构实例
- 实例讲评10: 国家标准《数据库设计说明书》



Lamps	No	Controller	Route	MAC	Address	Date	X	Y	Status	Sign
1	815	13573279275	2	1	即墨市湘江二路	2005-4-7	845	562	开灯	0
2	816	13573279275	1	2	即墨市湘江二路	2005-4-7	888	585	开灯	0
3	817	13573279275	2	3		2005-4-7	927	593	开灯	0
4	818	13573279275	1	4		2005-4-7	971	603	开灯	0
5	819	13573279275	2	5		2005-4-7	1012	616	开灯	0
6	820	13573279275	1	6		2005-4-7	1081	631	开灯	0

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.2 总体设计的过程

- 子模式设计: 用户使用的数据视图: 非物理数据库直接反映的数据
- 完整性和安全性设计:
 - 内容完整;
 - 使用安全性;
 - 操作安全性;
- 优化: 模式和子模式的优化: 利于存取。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.2 总体设计的过程

7. 制定测试计划

- 测试方法选择: 白盒测试/黑盒测试;
- 测试内容设计: 模块测试/功能测试/性能测试 /...
- 测试条件: 人员/设备/...
- 测试用例设计:
- 测试人员安排:
- 测试时间进度:
- 实例讲评11: 国家标准《测试计划》

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.2 总体设计的过程

8. 书写文档:

- 系统说明:
 - 系统流程图: 系统构成方案;
 - 组成的物理元素清单;
 - 成本/效益分析;
 - 最佳方案概述;
 - 精化的数据流程图;
 - 软件结构: 层次图或结构图;
 - 模块算法: IPO等工具;

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.2 总体设计的过程

- 用户手册: 修改/更正初步用户手册
- 实例讲评12: 国家标准《用户手册》
- 测试计划:
 - 测试策略
 - 测试方案
 - 预期测试结果
 - 测试进度计划

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.2 总体设计的过程

- 详细的实现计划
- 数据库设计结果
- 9. 审查和复审:
 - 实例讲评13: 国家标准《概要设计说明书》
 - 实例讲评14: 《路灯管理系统》总体设计报告

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

1. 模块化设计

- 把大型软件按照规定的原则划分为一个个较小的、相对独立但又相关的模块的设计方法，叫做**模块化设计** (modular design)。
- 模块** (module) 是数据说明和可执行语句等程序对象的集合，每个模块单独命名并且可以通过名字对模块进行访问。
- 实现模块化设计的重要指导思想是**功能分解、信息隐藏和模块独立性**。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

- 模块是由一定功能的可以用名词调用的程序语句集合，如：
 - C的子程序
 - 独立的汇编程序
 - COBOL的段和节
 - Pascal 过程
 - FORTTRAN的子程序
 - 汇编的宏

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

分解

设函数 $C(x)$ 定义问题 x 的复杂程度，函数 $E(x)$ 确定解决问题 x 所需要的工作量（时间）。对于两个问题 P_1 和 P_2 ，如果 $C(P_1) > C(P_2)$ ，显然 $E(P_1) > E(P_2)$ 。

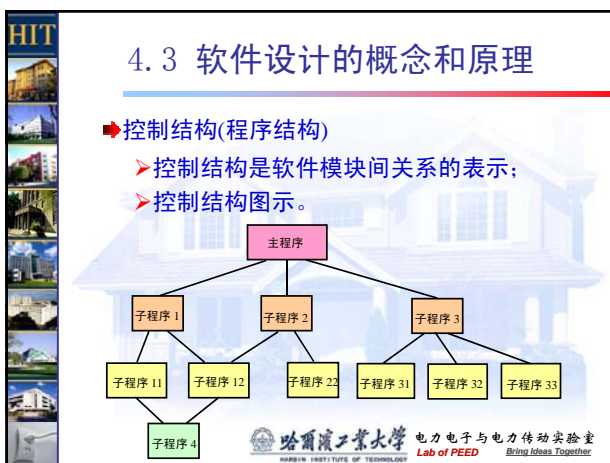
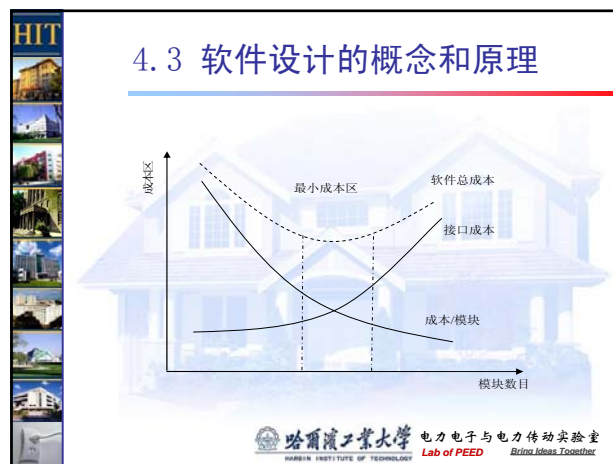
根据人类解决一般问题的经验，如果一个问题由 P_1 和 P_2 两个问题组合而成，那么它的复杂程度大于分别考虑每个问题时的复杂程度之和，即

$$C(P_1 + P_2) > C(P_1) + C(P_2)$$

综上所述，可得到下面的不等式

$$E(P_1 + P_2) > E(P_1) + E(P_2)$$

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together



4.3 软件设计的概念和原理

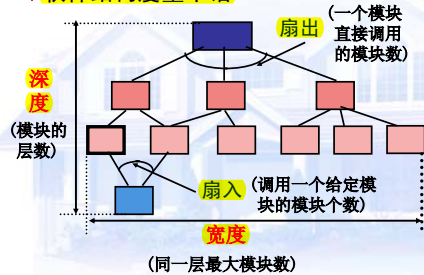
控制结构的层次规则

- 只有一个顶层 (0层) 模块
- 除0层外任一模块都会在其邻层存在一模块与它有关
- 同层模块间不发生联系

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

软件结构度量术语



哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

实例讲评15: 软件结构度量术语实例

```

Max_Value(array)      Disp(result_value)
{ ... Disp(Max_temp); ... }  { ... }
Min_Value(array)      main()
{ ... Disp(Min_temp); ... }  { ...
Ave_Value(array)      Max_Value(a[3][4]);
{ ... }               Min_Value(a[3][4]);
                     Ave_Value(a[3][4]);
                     Disp("The result is Ok!");
                     .... }
    
```

扇入: 2 深度: 3
扇出: 3 宽度: 4

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

SA与SD的关系

结构化分析的结果	结构化设计的工具
数据流程图	初始结构图
生存周期字典的数据部分	设计数据字典
伪码 实现方面	伪码
实体关系图	数据库设计
事务框图	分层、细化事务模型

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

2.抽象

- 抽象的含义: 反映本质特征, 忽略细节
- 多层次抽象
 - 最高层: 使用问题环境语言, 概括问题解法;
 - 较低抽象层: 更过程化的方法, 面向问题, 面向实现的解法;
 - 最底层: 直接实现的方式, 叙述问题解法。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

3.信息隐藏(Information Hiding)

- 模块内部的数据与过程, 应该对不需要了解这些数据与过程的模块隐藏起来。只有那些为了完成软件的总体功能而必需在模块间交换的信息, 才允许在模块间进行传递。
- 信息隐藏原理: 使一个模块内部包含的信息对于不需要这些信息的模块来说, 是不能访问的。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

- “隐蔽”意味着有效的模块化可以通过定义一组独立的模块而实现, 这些独立的模块彼此间仅仅交换那些为了完成系统功能而必须交换的信息。这一指导思想的目的为了提高模块的独立性, 即当修改或维护模块时减少把一个模块的错误扩散到其他模块中去的机会。
- 局部化: 指把一些关系紧密的软件元素物理地放得彼此靠近。Eg. 局部变量

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

实例讲评16: 隐藏与局部化

```

main()
{
    ...int a[3][4], R_A;
    Max_Value();
    Min_Value();
    Ave_Value();
    Disp("The result is Ok!");
    ...
    Disp(result_value)
    {
        ...
        Max_Value()
        {
            ...
            R_A=a[i][j]...
        }
        Min_Value()
        {
            ... R_A=a[i][j]...
        }
        Ave_Value(array)
        {
            ...
        }
    }
}
    
```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

4. 模块独立性 (module independence)

概括了把软件划分为模块时要遵守的准则, 也是判断模块构造是否合理**的标准**。一般地, 坚持模块的独立性是获得**良好设计的关键**。

两个定性度量标准——**内聚和耦合**

耦合用于衡量**不同模块**彼此间互相**依赖** (连接) 的**紧密程度**

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

内聚用于衡量一个模块**内部**各个**元素**间彼此结合的**紧密程度**。

模块独立的概念: 模块化、信息隐藏和局部化的直接结果。

- 完成特定功能
- 模块之间关系简单

需要模块独立的原因:

- 易开发
- 易测试

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

耦合强度取决于模块**接口**的**复杂程度**、通过接口的**数据**等;

应该追求尽可能**松散耦合**的系统: 影响系统的可理解性、可测试性、可靠性和可维护性;

耦合的七种类型

非直接耦合	数据耦合	特征耦合	控制耦合	外部耦合	公共耦合	内容耦合
-------	------	------	------	------	------	------

低 | 耦合性 | 高

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

耦合强度:

- 非直接耦合/无耦合: 最低
- 数据耦合: 低耦合/可以只包括该耦合;
- 控制耦合: 中耦合/通常模块分解可以用数据耦合;
- 公共环境耦合: 全程变量、共享通信区、内存公共覆盖区、存储介质上文件或设备等; /复杂程度随耦合模块个数变化: /一读一取: 属松散耦合; 既读又取: 介于数据耦合与控制耦合之间;

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

内容耦合: 最高程度耦合

无直接耦合: 两个模块没有直接关系 (模块1和模块2), 模块独立性最强。

```

graph TD
    M1[模块1] --- M3[模块3]
    M1 --- M4[模块4]
    M2[模块2]
    
```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

实例讲评17: 无直接耦合示例

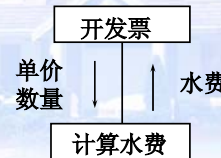
```
main()
{
    ...int a[3][4];
    Max_Value(a[3][4]);
    Min_Value(a[3][4]);
    Ave_Value(a[3][4]);
    Disp("The result is Ok!");
    ...
}
Disp(result_value)
{
    ...
}
```

```
Max_Value(array)
{
    ...
}
Min_Value(array)
{
    ...
}
Ave_Value(array)
{
    ...
}
```

4.3 软件设计的概念和原理

数据耦合: 一模块调用另一模块时, 被调用模块的输入、输出都是简单的数据(若干参数)。属松散耦合。

实例讲评18: 数据耦合示例



4.3 软件设计的概念和原理

实例讲评18: 数据耦合程序示例

```
main()
{
    ...int W_P_Price; // W_P_Price: 水电单价;
    int W_sum, W_Price; // W_sum: 用水量; W_Price: 水费
    ...
    W_Price = Cal_W_Price(W_P_Price, W_Sum);
    ...
}
int Cal_W_Price(TW_P_Price, TW_Sum) // 计算水电费函数:
{
    ... int TW_Price; ...
    TW_Price = TW_P_Price * TW_Sum;
    ...
    return(TW_Price);
}
```

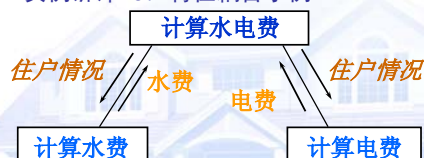
4.3 软件设计的概念和原理

特征耦合

- 也称标记耦合(复合型耦合);
- 如两个模块通过传递数据结构(不是简单数据, 而是记录、数组等)加以联系, 或都与一个数据结构有关系, 则称这两个模块间存在标记耦合。

4.3 软件设计的概念和原理

实例讲评19: 特征耦合示例



“住户情况”是一个数据结构, 图中模块都与此数据结构有关。
“计算水费”和“计算电费”本无关, 由于引用了此数据结构产生依赖关系, 它们之间也是标记耦合。

4.3 软件设计的概念和原理

实例讲评19: 特征耦合示例

```
struct User
{
    char User_name[20]; // User_name: 用户名;
    int User_ID; // User_ID: 用户号;
    int W_Quan; // W_Quan: 用水量;
    int P_Quan; // P_Quan: 用电量;
    ...
}
Users[3] = { {"Zhang San", 10001, 123, 35, ...},
             {"Li Si", 10002, 102, 38, ...},
             {"Wang Wu", 10003, 138, 36, ...} };
```


4.3 软件设计的概念和原理

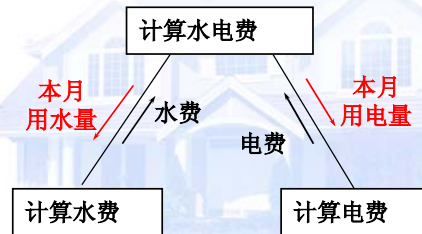
实例讲评19: 特征耦合示例

```
main()
{...int W_Price,P_Price; // W_Price:水价,P_Price:电价
  unsigned C_W,C_P; // C_W:水标志,C_P:电标志
  ...
  W_Price=Cal_WP (Users[1]. User_ID, C_W);
  P_Price=Cal_WP (Users[1]. User_ID, C_P);
  ...
}
int Cal_WP (int TW_ User_ID, unsigned C_Sort)
//计算水电费函数;
{... return(W_Price);
... return(P_Price);
}
```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

将标记耦合修改为数据耦合举例



哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

实例讲评20: 将标记耦合修改为数据耦合示例

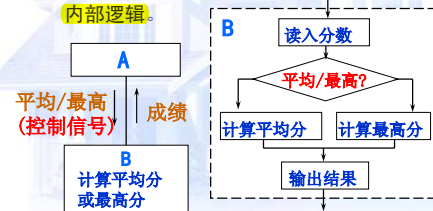
```
main()
{...int W_Price,P_Price; ...
  W_Price=Cal_W (W_Quan[i]);
  P_Price=Cal_P (P_Quan [i]);
  ...
}
int Cal_W (int WQuan) //计算水费函数;
{... return(W_Price);}
int Cal_P(int WQuan) //计算电费函数;
{... return(P_Price);}
```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

控制耦合

模块向下属模块传递的信息（开关量、标志等控制被调用模块决策的变量）控制了被调用模块的内部逻辑。



哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

实例讲评21: 控制耦合示例

```
main()
{...int C_Kind; //计算种类标志
  ...
  C_Kind=1;
  Max_P=C_MA (C_Kind);
  C_Kind=0;
  Ave_P= C_MA (C_Kind);
  ...
}
int C_MA (int TC_ Kind)
{... if (TC_ Kind >=1
  { ...
    return(Max_V);
  }
  else
  {...
    return(Ave_V);
  }
} //计算最大值和平均值函数
```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

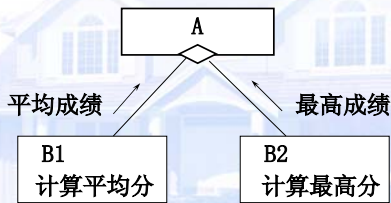
去除模块间控制耦合的方法

- 控制耦合增加了理解和编程的复杂性，调用模块必须知道被调模块的内部逻辑，增加了相互依赖
- (1) 将被调用模块内的判定上移到调用模块中进行
- (2) 被调用模块分解成若干单一功能模块

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

改控制耦合为数据耦合举例



4.3 软件设计的概念和原理

实例讲评22：改控制耦合为数据耦合示例

```
main()
{
    int C_Sort;
    int Max_V, Ave_V;
    ...
    if (C_Sort>=1
    { Max_V=Cal_Max_V; }
    else
    { Ave_V =Cal_Ave_V; }
    ...
}
```

```
//计算最高分函数:
int Cal_Max_V(void)
{
    ...
    return(Max_V);
}

//计算平均分函数:
int Cal_Ave_V(void)
{
    ...
    return(Ave_V);
}
```

4.3 软件设计的概念和原理

外部耦合

- 一组模块均与同一外部环境关联(例如, I/O 模块与特定的设备、格式和通信协议相关), 它们之间便存在外部耦合。
- 外部耦合必不可少, 但这种模块数目应尽量少。

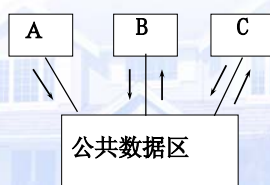
4.3 软件设计的概念和原理

公共耦合(公共数据区耦合)

- 一组模块引用同一个公用数据区(也称全局数据区、公共数据环境)。
- 公共数据区指: 全局数据结构、共享通讯区、内存公共覆盖区等。

4.3 软件设计的概念和原理

实例讲评23：公共耦合示例



模块A、B、C间存在错综复杂的联系

4.3 软件设计的概念和原理

实例讲评23：公共耦合示例

```
main()
{
    ...
    unsigned W_Flag;
    unsigned P_Flag, A_Flag;
    ...
    Max_V=Cal_Max_V;
    Ave_V =Cal_Ave_V;
    Min_V=Cal_Min_V; ...
}
```

```
//计算最高分函数:
int Cal_Max_V(void)
{
    ... W_Flag=1;...
}

//计算平均分函数:
int Cal_Ave_V(void)
{
    ... P_Flag=0;...
}

//计算最低分函数:
int Cal_Min_V(void)
{
    ... P_Flag=2;...
}
```

HIT

4.3 软件设计的概念和原理

公共耦合存在的问题

(1) 软件可理解性降低

(2) 诊断错误困难

(3) 软件可维护性差,

(4) 软件可靠性差

(公共数据区及全程变量无保护措施)

慎用公共数据区和全程变量!!!

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

4.3 软件设计的概念和原理

内容耦合

A

B

A

B

一模块直接访问另一模块的内部信息(程序代码或数据)

模块代码重叠

多入口模块

Entry1

.....

Entry1

.....

最不好的耦合形式 !!!

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

4.3 软件设计的概念和原理

耦合设计原则

尽量使用数据耦合

少用控制耦合

限制公共环境耦合范围

完全不用内容耦合

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

4.3 软件设计的概念和原理

内聚概念：衡量一个模块内部各元素彼此结合的紧密程度。

简单地说，理想内聚的模块只做一件事。

设计时应该力求做到高内聚，通常中等程度的内聚也是可以采用的，而且效果和高内聚相差不多。但是，坚决不要使用低内聚。

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

4.3 软件设计的概念和原理

内聚分类：七种类型

偶然内聚

逻辑内聚

时间内聚

过程内聚

通讯内聚

顺序内聚

功能内聚

低

内聚

高

低内聚：偶然内聚：出现错误的概率比其他类型的模块要高/0分；

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

4.3 软件设计的概念和原理

偶然内聚(巧合内聚)

实例讲评24：偶然内聚示例

例：

A

B

C

M

MOVE

0

TO

R

READ

FILE

F

MOVE

S

TO

T

模块M中的三个语句没有任何联系

缺点：可理解性差，可修改性差

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

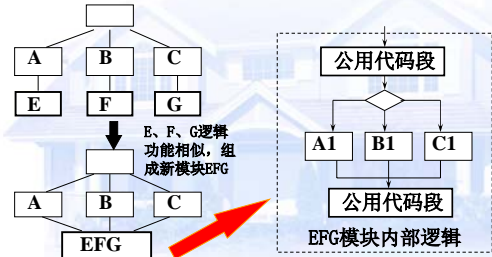
4.3 软件设计的概念和原理

逻辑内聚:

- 把几种相关功能（逻辑上相似的功能）组合在一模块内，每次调用由传给模块的参数确定执行哪种功能。
- 修改困难/1分。

4.3 软件设计的概念和原理

实例讲评25: 逻辑内聚示例



缺点: 增强了耦合程度(控制耦合), 不易修改, 效率低

4.3 软件设计的概念和原理

时间内聚

- 经典内聚, 比逻辑内聚好一些/3分。模块完成的功能必须在同一时间内执行, 这些功能只因时间因素关联在一起。

实例讲评25: 时间内聚示例

- 例如: 初始化系统模块、系统结束模块、紧急故障处理模块等均是时间性聚合模块。

4.3 软件设计的概念和原理

实例讲评25: 时间内聚: 初始化模块

```
void main( void )
{
    unsigned char count0,count1=0;
    uchar i;
    devices_init();
    SEI();
    Flag=0;
    for(;;)
    {
        ... //主循环体;
    } //主循环体结束;
} //主程序结束;
```

//初始化部分: 时间内聚;

4.3 软件设计的概念和原理

过程内聚 (顺序性组合)

- 过程内聚: 程序流程图作为工具设计软件时得到模块/5分
- 模块内各处理成分相关, 且必须以特定次序执行
- 属中内聚;

4.3 软件设计的概念和原理

实例讲评26: 过程内聚: 定时器与中断标志

```
#pragma interrupt_handler INT0_interrupt:iv_INT0
void INT0_interrupt( void )
{
    WDR();
    GICR |= (1<<INTF0); //清中断标志;
    GICR &= ~(1<<INT0); //立刻关中断---进入中断
    TIFR = (1<<TOIE0); //清Timer0溢出标志;
    TIMSK = (1<<TOIE0); //置Timer0溢出允许;
    ...
    GICR |= (1<<INT0);
}
```

顺序内聚

4.3 软件设计的概念和原理

实例讲评27: 过程内聚

读入成绩单 → 审查成绩单 → 统计成绩 → 打印成绩

读入并审查成绩单 统计并打印成绩单

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

通信内聚

- 模块内各部分使用相同的输入数据，或产生相同的输出结果
- 通信内聚: 7分
- 属中内聚:

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

实例讲评28: 通信内聚示例 1

职工工资记录 → 产生工资报表 → 职工工资报表

职工工资记录 → 计算平均工资 → 平均工资

产生职工工资报表并计算平均工资模块

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

实例讲评28: 通信内聚示例 2

```

...
if (Uart_RcvFlag==1) //串口接收成功;
{
    switch (rx_buffer[3]) //判断接收缓冲区第3个字节数据;
    {
        case 0: {TCNT0=0xFF; sendchar(0x00); break; }
        case 1: {TCNT0=0xE0; sendchar(0x01); break; }
        case 2: {TCNT0=0xC1; sendchar(0x02); break; }
        case 3: {TCNT0=0xA2; sendchar(0x03); break; }
    }
}
...

```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

顺序内聚

- 数据流图作为工具设计软件时得到的模块 / 9分;
- 信息内聚;
- 模块完成多个功能，各功能都在同一数据结构上操作，每一功能有唯一入口。
- 属高内聚;

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

实例讲评29: 顺序内聚示例

```

...
SoftMasterTXBuffer[1]=0x01;
SoftMasterTXBuffer[2]=0x02;
SoftMasterTXBuffer[3]=0xaa;
SoftMasterTXBuffer[4]=0x01;
SoftMasterTXBuffer[5]=0x02;
SoftMasterTXBuffer[6]=0x03;
SoftMasterTXBuffer[7]=0x04;
SoftMasterTXBuffer[8]=0x05;
for(i=0;i<9;i++)
{
    sendchar(SoftMasterTXBuffer[i]); //顺序串行发送;
}
...

```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.3 软件设计的概念和原理

功能内聚

- 最高内聚/理想内聚只作一件事/10分
- 模块仅包括为完成某个功能所必须的所有成分。
- 模块所有成分共同完成一个功能，缺一不可。
- 内聚性最强；
- 属高内聚；

4.3 软件设计的概念和原理

实例讲评30：功能内聚示例

```
void devices_init(void)
{
    CLI();           //disable all interrupts
    IOPort_Init();   //IO口初始化;
    adc_init();       //ADC初始化;
    uart0_init();     //串口初始化;
    timer0_init();    //定时器0初始化;
    timer1_init();    //定时器0初始化;

    //all peripherals are now initialized
}
```

4.3 软件设计的概念和原理

内聚设计原则：

- 力求高内聚；
 - 中等内聚也可以采用；
 - 低内聚不要用。
- 与耦合关系：高内聚意味松耦合。
- 实践表明，内聚更重要，应该把更多注意力集中到提高模块的内聚程度上。

4.4 总体设计工具

1. 启发式规则

- (1) 改进软件结构, 提高模块独立性
- (2) 深度、宽度、扇出和扇入应适中
- (3) 模块的作用域应该在控制域之内
- (4) 力争降低模块接口的复杂程度
- (5) 设计单入口、单出口的模块
- (6) 模块功能应该可以预测

4.4 总体设计工具

注：在软件开发过程中既要充分重视和利用这些启发式规则，又要从实际情况出发避免生搬硬套。

- ❖ 改善软件结构，提高模块独立性；
- ❖ 模块规模应适中；
- ❖ 深度、宽度、扇出和扇入都应适当
- ❖ 深度：软件结构中控制的层数；

4.4 总体设计工具

实例讲评31：深度为3的示例

```
...
if 条件1 then
{
    if 条件2 then
    {
        if 条件3 then{ .....}
        else{...}
    }
    else{.....}
}
else{....}
endif
```


4.4 总体设计工具

实例讲评32: 深度为1的示例

```

if 条件1 then {...}
else {...}
endif

if 条件2 then {...}
else {...}
endif

if 条件3 then {...}
else {...}
endif

```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

实例讲评32: 深度为4的示例

```

function calcul_price()
{
...
calcul_volume(x, y, z);
...
}
function
calcul_volume(int
a,int b,int c)
{...
calcul_area(a,b);
...
}

function calcul_area(int m,
int n)
{...
calcul_length(d);
...
}

function calcul_length(int u)
{...}

```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

- 宽度: 软件结构内同一个层次上的模块总数的最大值
- 扇出: 一个模块直接控制/调用的模块数。平均扇出为3或4
- 扇入: 一个模块扇入表明有多少个上级模块直接调用它

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

- 模块的作用域应该在控制域之内
 - 作用域: 受该模块内一个判断影响的所有模块集合。
 - 控制域: 该模块本身以及所有直接或间接从属于它的模块的集合。
 - 作用域应是控制域的子集。
 - 改变作用域与控制域的方法: 判断点上移/作用域对象下移。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

```

graph TD
    R[R] --> A1[A1]
    R --> A2[A2]
    R --> AN[AN]
    A2 --> B1[B1]
    A2 --> B2[B2]
    A2 --> BM[BM]
    B2 --> C1[C1]
    B2 --> C2[C2]

```

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

- 力争降低模块接口的复杂程度:
 - 实例讲评33: 一元二次方程求解函数


```

Quad_root(a, b, c, root1, root2)
Quad_root(table, x)

```
- 接口复杂或不一致: 紧耦合或低内聚) 重新分析模块独立性
- 设计单入口单出口的模块: 易理解;
- 模块功能应用可以预测;

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

2. 总体设计的图形工具

层次图：层次图（也称H图）是在总体设计阶段最常使用的图形工具之一，它常用于描绘软件的层次结构。**层次图中的每个方框代表一个模块，方框间的连线表示模块间的调用关系。**

- 矩形代表一个模块；
- 连线表示调用关系；
- 适于在自顶向下设计软件的过程中使用
- 与层次方框图类似；

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

实例讲评34：层次图举例

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

HIPO图实际上由H图和IPO图两部分组成。
 IPO: 输入/处理/输出图的缩写。
 为了使HIPO图具有可跟踪性，在H图里除了最顶层的方框之外，每个方框都加了编号；
 和H图中的每个方框相对应，有一张IPO图描述这个方框代表的模块的处理过程。IPO图能够方便地描述数据输入、数据输出和数据输出之间的关系。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

IPO基本形式是：在左边的框中列出有关的输入数据，在中间的框中列出主要的处理——处理框中列出的处理次序，暗示了执行的次序，在右边的框中列出产生的输出数据。
 另外，还用类似向量符号的粗大箭头清楚地指出数据通信的情况。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

实例讲评35：HIPO层次图举例

图 4.2 带编号的层次图 (H 图)

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.4 总体设计工具

实例讲评36：IPO图举例

图 4.3 IPO 图的例子

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

HIT

4.4 总体设计工具

◆结构化设计(SD)方法概述

(1) 首先研究、分析和审查数据流图，从软件的需求规格说明中弄清数据流加工的过程。

(2) 然后根据数据流图决定问题的类型，即确定是变换型还是事务型。针对两种不同的类型分别进行分析处理。

(3) 由数据流图推导出系统的初始结构图。

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

(4) 利用一些试探性原则来改进系统的初始结构图，直到得到符合要求的结构图为止。

(5) 修改和补充数据词典。

(6) 制定测试计划。

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

◆SD方法在概要设计中的主要表达工具约定:

不加区分的数据

数据信息

控制信息

编辑学生记录

学号

学生数据

无此学生

读学生记录

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

◆结构图(SC Structure Chart): 四种模块

传入模块

传出模块

变换模块

协调模块

(a)

(b)

(c)

(d)

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

◆SC中的选择调用

A根据内部判断决定是否调用B

A按另一判定结果选择调用C或D

A

B

C

D

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

◆SC中的循环调用

A

B

C

A根据内在的循环重复调用B、C等模块

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

17

HIT

4.4 总体设计工具

SC设计优化

(1) 在不考虑时间因素的前提下开发并精化软件结构;

(2) 在详细设计阶段选出最耗费时间的那些模块, 仔细地设计它们的处理过程, 以求提高效率;

(3) 使用高级程序设计语言编写程序;

(4) 必要时重新设计或用依赖于机器的语言重写上述大量占用资源的模块的代码, 以求提高效率。

(5) 在软件中孤立出那些大量占用处理和资源的

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

实例讲评37: 医院管理系统SC实例

医院管理系统

门诊管理

药库管理

药房管理

病房管理

财务管理

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

实例讲评38: 酒店管理信息系统功能结构图SC实例

H M I S

收银管理系统

收银管理系统

收银管理系统

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

实例讲评39: 零售管理信息系统功能结构图SC实例

T M M I S

系统维护

P O S 系统

零售实时系统

商品进销管理

商品批发管理

商品库存管理

商品及商品帐管理

顾客管理

连锁店管理

财务管理

人事工资管理

计划统计管理

经理查询

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.4 总体设计工具

实例讲评40: 产生最佳解功能结构图SC实例

产生最佳解

得到好输入

计算最佳解

输出结果

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.5 面向数据流的设计方法

SD以数据流图为基础, 它定义了把DFD变换成软件结构的不同映射方法。

DFD (问题结构)

映射

软件系统的结构 (程序结构)

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

18

4.5 面向数据流的设计方法

- 系统结构特征可归纳为两种典型形式
 - 变换型结构
 - 事务型结构
- 数据流图可分为两种类型
 - 变换型数据流
 - 事务型数据流

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.5 面向数据流的设计方法

系统结构基本模型

变换型结构

输入 → 变换中心 → 输出

特征：由输入、变换中心和输出三部分组成

事务型结构

接受路径 → 事务中心 → 动作路径

特征：具有在多种事务中选择执行某类事物的能力

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.5 面向数据流的设计方法

数据流图基本模型

变换型数据流结构

传入部分 → 传入 → 变换中心 → 变换 → 传出 → 传出部分

事务型数据流结构

接受部分 → 接受 → 事务中心 → 事务分析 → 动作1 → 动作2 → 动作3

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.5 面向数据流的设计方法

变换型数据流举例

传入部分：输入信息 → 物理输入 → 格式检查

变换中心：正确信息 → 处理

传出部分：结果 → 显示 → 数据 → 逻辑输出 → 物理输出

特点：具有明确的传入、变换(或称主加工)和传出界面的DFD

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.5 面向数据流的设计方法

事务型数据流图举例

数据流图示例：A → I → B → L → E → O → H, I → C → M → F → O, I → D → N → G → O

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.5 面向数据流的设计方法

大型系统DFD中, 变换型和事务型结构往往共存

数据流图示例：事务中心(T)连接到多个变换型结构(传入、变换、传出)

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

HIT

4.5 面向数据流的设计方法

面向数据流设计方法的设计步骤

(1) 精化DFD。

(2) 确定DFD类型。

(3) 把DFD映射到系统模块结构设计出模块结构的上层。

(4) 基于DFD逐步分解高层模块设计出下层模块。

(5) 根据模块独立性原理，精化模块结构。

(6) 模块接口描述。

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.5 面向数据流的设计方法

精化数据流程图

“事务”

“变换”

流类型

区分事务中心和数据接收路径

区分输入和输出分支

映射成事务结构

映射成变换结构

事务分析

变换分析

用启发式设计规则精化软件结构

导出接口描述和全程数据结构

复查

详细设计

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.5 面向数据流的设计方法

SD方法的两种映射过渡方法

变换分析

变换型DFD

初始SC

事务分析

事务型DFD

初始SC

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.5 面向数据流的设计方法

初始的SC

主模块

输入模块

主加工模块

输出模块

事务控制模块

接受模块

动作发送模块

动作1模块

动作2模块

动作3模块

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.5 面向数据流的设计方法

变换分析设计方法

(1) 区分传入、变换中心、传出部分，在DFD上标明分界线。

(2) 第一级分解(建立初始SC框架)，设计顶层和第一层模块。

(3) 第二级分解(分解SC各分支)，自顶向下分解，设计出每个分支的中、下层模块。

哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

HIT

4.5 面向数据流的设计方法

变换中心

传入部分

传出部分

变换中心

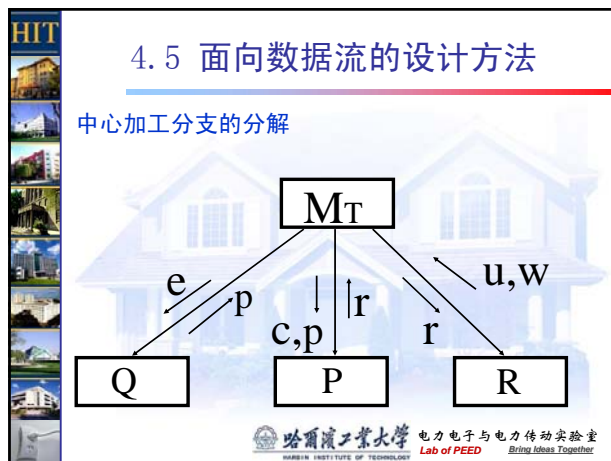
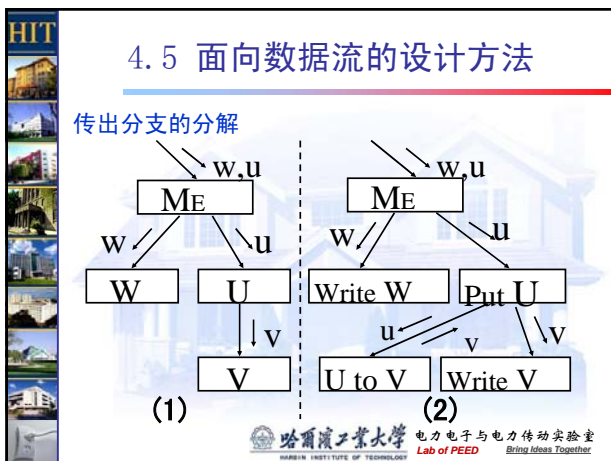
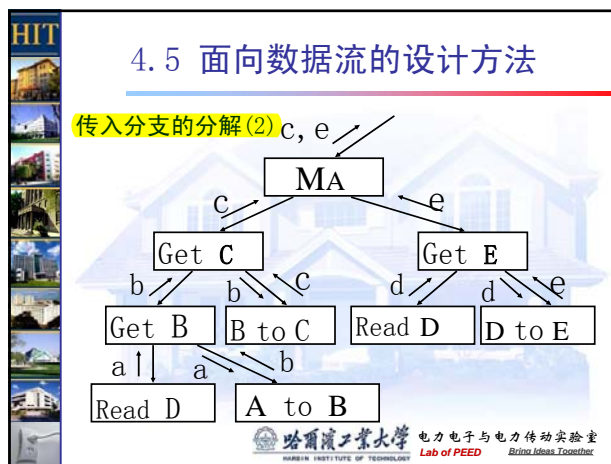
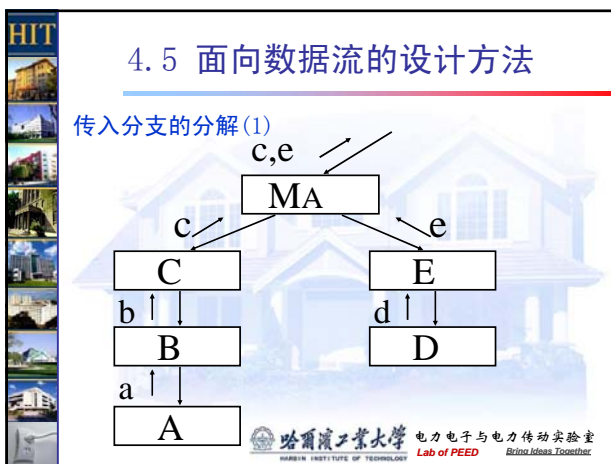
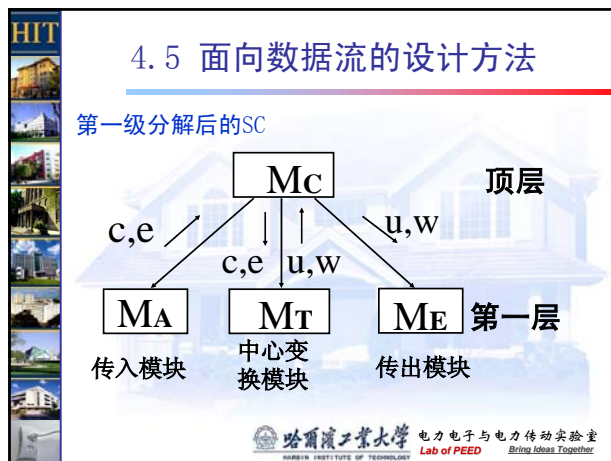
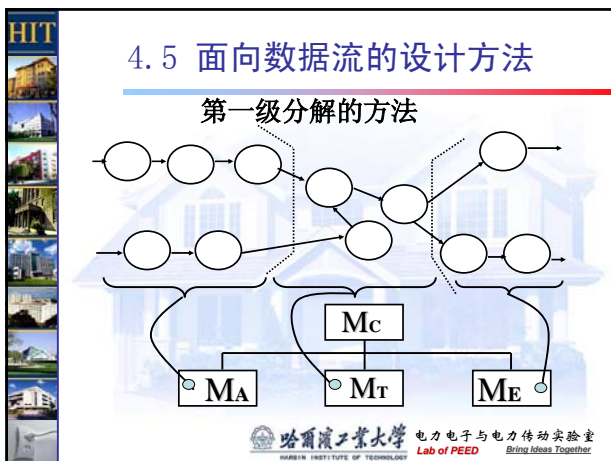
哈尔滨工业大学

电力电子与电力传动实验室

Lab of PEED

Bring Ideas Together

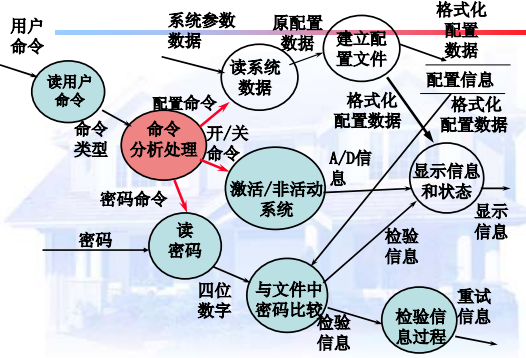
20



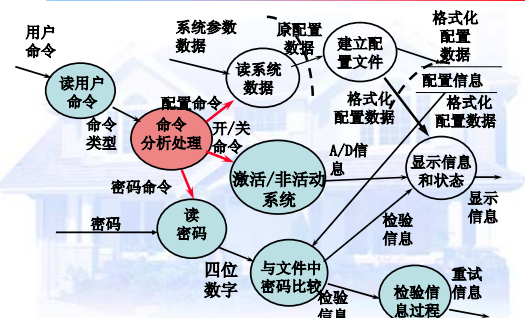
A vertical strip of nine images. The top image shows the word 'HIT' in large, bold, blue letters. Below it are eight images of different buildings, likely part of the HIT project. The buildings vary in style, from modern glass-fronted structures to more traditional brick buildings. The bottom image shows a white remote control with a silver button, pointing towards the right.

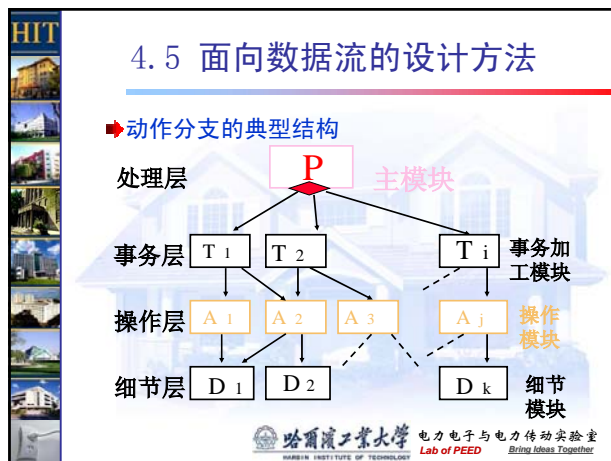
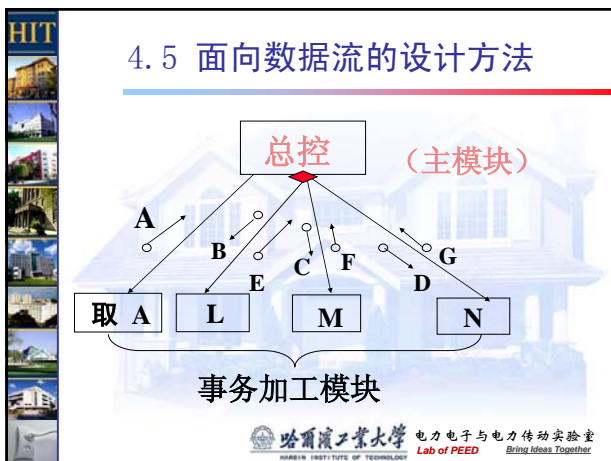
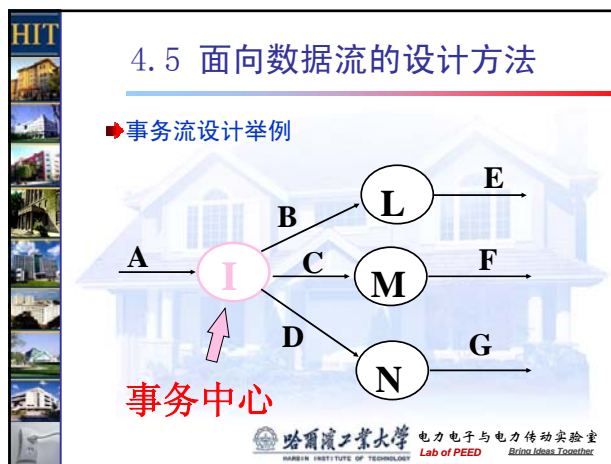
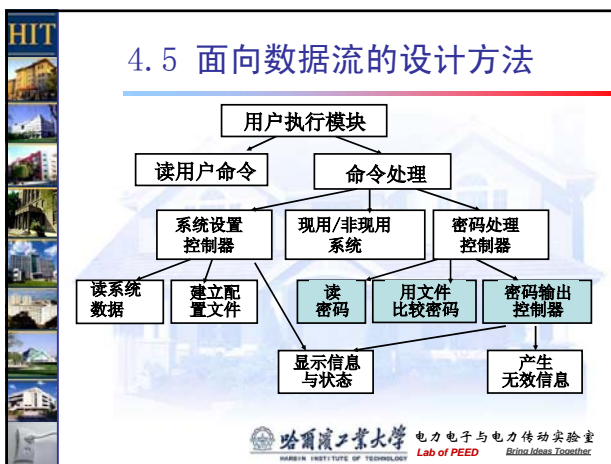
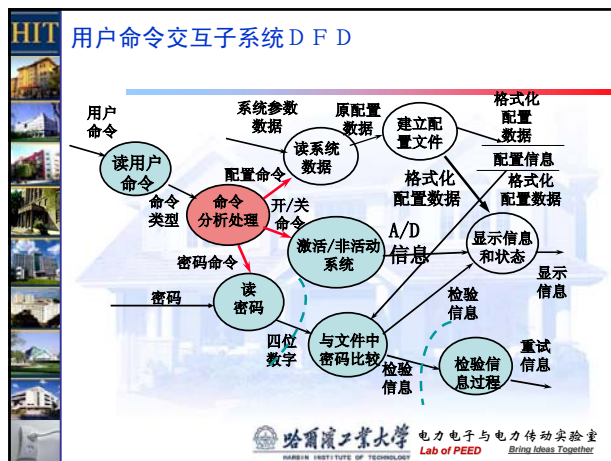
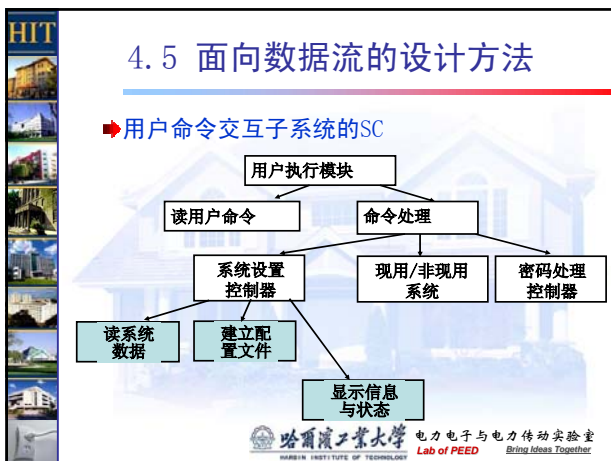
➡任何情况下都可使用变换分析方法设计软件结构，但如数据流具有明显的事务特点时（有一个明显的事务中心），以采用事务分析方法为宜。

- (1) 在DFD上确定事务中心、接收部分和发送部分。
- (2) 画出SC框架，把DFD上的三部分分别映射为事务控制模块、接收模块和动作发送模块。
- (3) 分解细化接收分支和发送分支，完成初始SC。

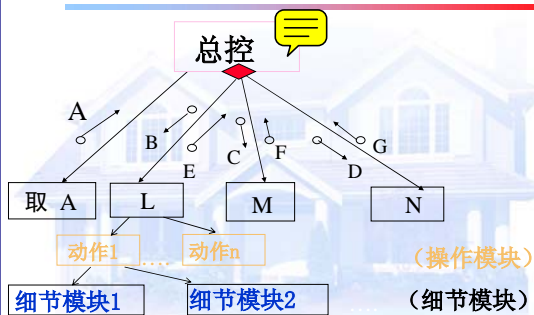


```
graph TD; A[用户执行模块] --> B[读用户命令]; B --> C[命令处理]; C --> D[系统设置控制器]; C --> E[现用/非现用系统]; C --> F[密码处理控制器];
```



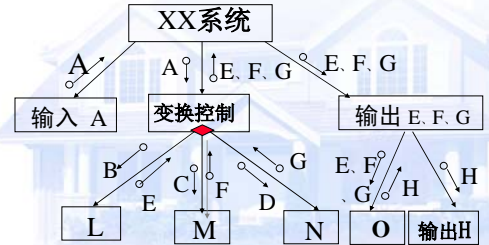


4.5 面向数据流的设计方法



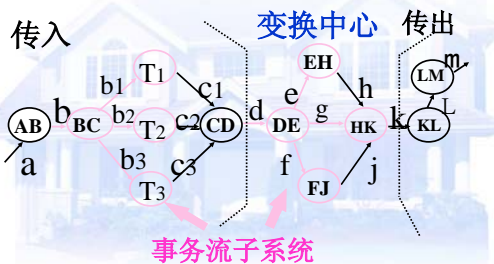
4.5 面向数据流的设计方法

事务流设计举例（另一种画法）

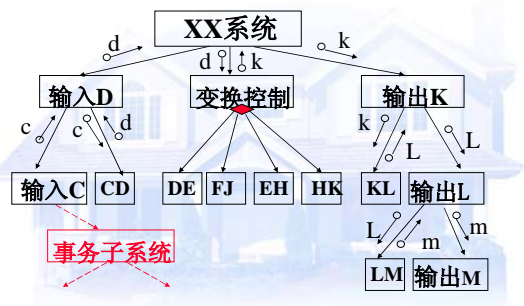


4.5 面向数据流的设计方法

混合流设计举例



4.5 面向数据流的设计方法



4.6 体系结构设计优化

- 将初始SC根据模块独立性原则进行精化，对模块进行合并、分解修改、调整，得到高内聚、低耦合模块，得到易于实现、易于测试和易于维护的软件结构，产生设计文档的最终SC。

4.6 体系结构设计优化

- 改进软件结构设计的指导原则
 - (1) 模块功能的完善化
 - (2) 消除重复功能
 - (3) 将模块的影响限制在模块的控制范围内
 - (4) 深度、宽度、扇出和扇入适中
 - (5) 模块大小适中
 - (6) 降低模块接口的复杂性
 - (7) 模块功能可预测
 - (8) 避免模块的病态连接
 - (9) 根据设计约束和可移植性要对软件打包

4.6 体系结构设计优化

- (1) 模块功能的完善化
完整的模块应包括三部分:
 - ➡ 执行规定功能部分
 - ➡ 出错处理部分
 - ➡ 需返回给调用者数据时, 返回是否正确结束标志。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.6 体系结构设计优化

- (2) 消除重复功能

改进前
Q1、Q2功能相似

改进方法1:
将Q1、Q2合并为Q'
不可取

改进方法2:
将Q1、Q2的公共部分分离出来

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.6 体系结构设计优化

- (3) 将模块影响限制在模块的控制范围内

如果模块C作出的决策影响了模块L, L超出了C的控制范围

模块C的控制范围: C、D、E、F、G、H

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.6 体系结构设计优化

- (4) 减少高扇出争取高扇入

避免平铺结构

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.6 体系结构设计优化

- 增加中间层降低扇出

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

4.6 体系结构设计优化

- (5) 模块大小适中
模块过大: 可理解程度下降
模块过小: 开销大于有效操作
系统接口复杂
- (6) 降低模块接口的复杂性
接口传递信息应简单且和模块功能一致。
- (7) 模块功能可预测
➡ 模块看成黑盒子, 相同输入产生相同输出, 其功能为可预测的。

哈尔滨工业大学 电力电子与电力传动实验室
HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

HIT

4.6 体系结构设计优化

模块带有内部状态其功能可能是不可预测的。难理解、难测试、难维护。

防止模块功能过分局限。功能单一的模块具有高内聚。但如任意限制局部数据结构的大小，过分限制控制流中可做的选择或外部接口的模式，模块功能就过分局限，使用范围过分狭窄，缺乏灵活性和可扩充性。

哈尔滨工业大学 电力电子与电力传动实验室

HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

HIT

4.6 体系结构设计优化

(8) 避免模块的病态连接：
防止指向模块中间的分支或引用（针对内容耦合）

(9) 根据设计约束和可移植性需求对软件打包
打包指用来为特定环境组装软件的技术

哈尔滨工业大学 电力电子与电力传动实验室

HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

HIT

4.7 总体设计报告撰写与相关标准

国家标准对总体设计报告的要求

实例讲评40：路灯管理系统总体设计报告

哈尔滨工业大学 电力电子与电力传动实验室

HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

HIT

本章结束

哈尔滨工业大学 电力电子与电力传动实验室

HARBIN INSTITUTE OF TECHNOLOGY Lab of PEED Bring Ideas Together

26