outline

- 编码
- 配置管理
- 质量管理
- 软件度量

项目案例

- 人物

小王：软件项目负责人

老王：公司技术总监

项目案例

- 在项目策划阶段的碰头会上
  - 老王问小王项目开发估计需要多少时间，多少成本？
  - 小王回答说“时间估计不会太长，成本也在一个可接受的范围之内”，老王显然对这种回答不满意，他希望能够得到一个较为准确定量性的描述。
  - 经过一番考虑后，小王回答说“时间7—8个月，成本需40—45万”，老王显然对这种回答也不满意，况且用户要求在6个月内完成项目。于是他进一步问道“你是如何得到这组数据”，小王显然没有准备，也没有充分的依据，于是他哑口无言。

项目案例

- 在制定项目计划时
  - 小王不知如何预测项目可能所需的工作量？
  - 小王不知如何预测项目可能所需的成本？
  - 小王不知所制定的计划是否可行和科学？
  - 因此，小王尽管制定了软件开发计划，但对于该计划能否得到有效的实施、实施能否遵循计划执行没有足够的信心

项目案例

- 项目已进展了2个月，各个方面进展尚可，在某周的碰头会上，老王继续向小王发问
  - “目前软件质量如何？”，小王回答道“不错”
  - 老王对这种回答不满意，他希望能够得到一个较为准确定量性的描述，但是小王又没有办法给他一个更加确切的答复，实际上连他自己也没有办法说清楚目前软件产品的质量情况，因为他只有直观的、定性了解。

概述

- 任何工程化的工作都需要度量
  - 准确了解工程的实施情况
- 软件工程需要定量、科学的描述
  - 实施前：估算成本和工作量，辅助制定软件项目的计划
  - 实施过程中：提供软件开发的可视性，跟踪和控制软件项目的开发，评估软件开发质量，进行质量控制
  - 实施完成后：对项目的实施情况进行评估，为后续项目积累经验数据
- 定量、科学的描述有助于获取软件项目以及所开发的软件的某种可视性，促进软件项目的管理
- 定量的信息描述必须在软件项目开发过程中采集

基本概念

- 对事物属性的**定性**描述
  - 个子很高, 软件的成本很高
- 对事物属性的**定量**描述
  - 个子有1.9米，软件成本是 23.5万

基本概念

- 软件度量（Metrics）是指对软件产品、软件开发过程及软件开发项目的定量描述。
- 软件度量三维度
  - 产品度量：软件开发过程中所生成的各种文档和程序
  - 过程度量：与软件开发有关的各种活动，如软件设计等
  - 项目度量：度量项目规模、项目成本、项目进度等

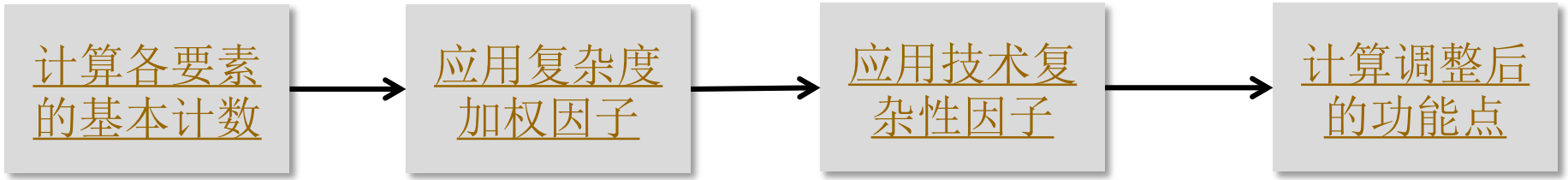
产品度量

- 需求模型度量
- 设计模型度量
- 程序代码度量
- 测试度量
- 维护度量

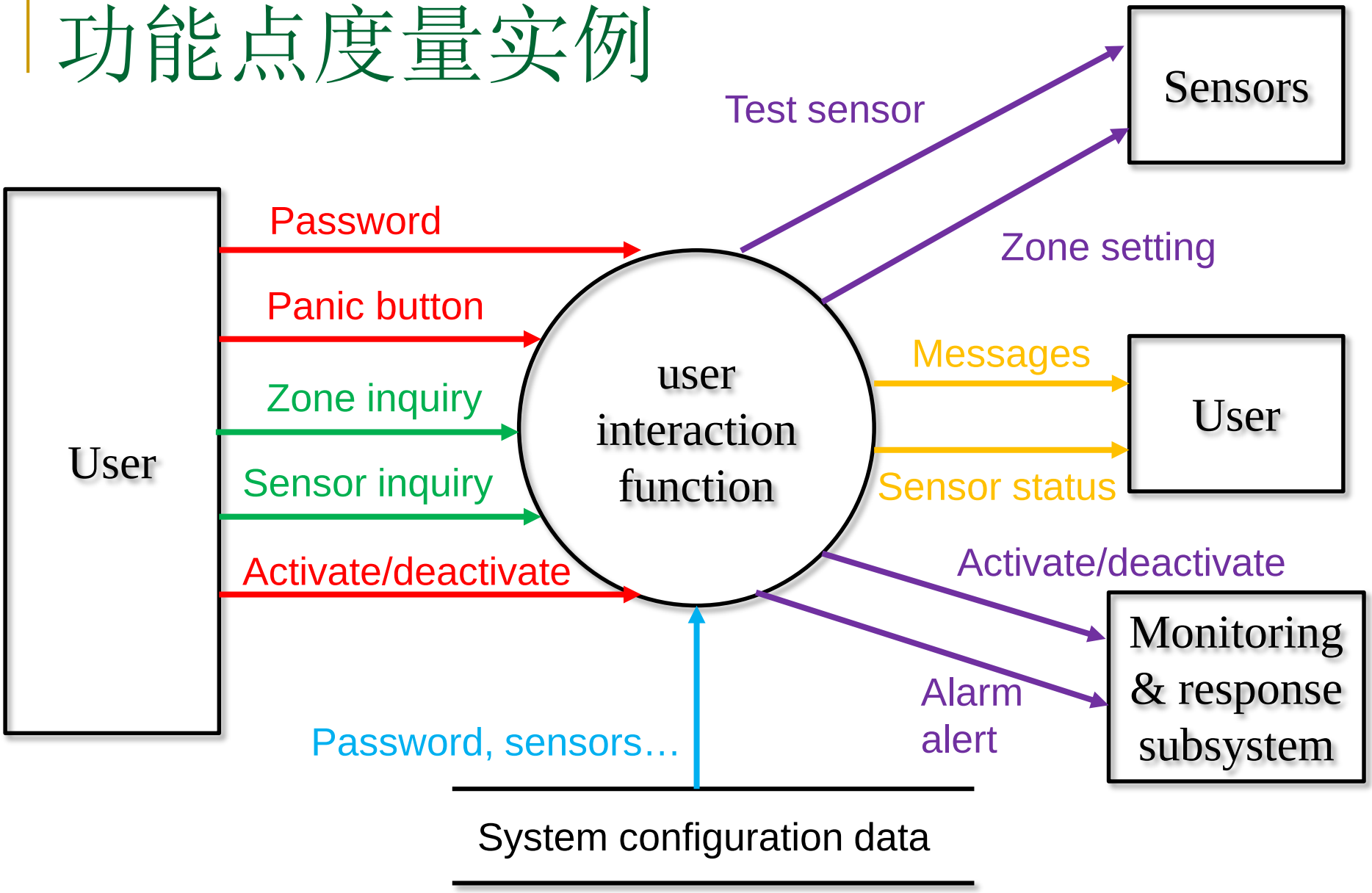
需求模型度量

- 检测需求模型，预测将来系统的规模
- 面向功能的度量
  - **功能点度量**（function point, FP）：度量系统提供的功能
  - 基于系统功能的一种规模估算方法
  - IBM的工程师Allan Albrech提出
  - 目前主要基于经验公式

功能点度量基本步骤



功能点度量实例


Sensors
Test sensor
Zone setting
User
Password
Panic button
Zone inquiry
Sensor inquiry
Activate/deactivate
user
interaction
function
Messages
Sensor status
User
Activate/deactivate
Monitoring
& response
subsystem
Alarm
alert
Password, sensors…
System configuration data

功能点度量实例

信息量	计数		加权因子*wf* 简单	平均	复杂		
EIs	3	×	**3**	**4**	**6**	=	9
EOs	2	×	**4**	**5**	**7**	=	8
EQs	2	×	**3**	**4**	**6**	=	6
ILFs	1	×	**7**	**10**	**15**	=	7
EIFs	4	×	**5**	**7**	**10**	=	20
未调整的功能点数UFP							50

功能点度量实例

- 假设该实例是一个中等复杂度的产品
- $\sum(F_i)$=46

FP = UFP×TCF=50×[0.65+(0.01 ×46)]=56

设计模型度量

- Card软件设计复杂性度量
- Fenton简单形态度量
- OOD度量
- ……

Card软件设计复杂性度量

- 结构复杂性

  $S(i)=f_{out}^{2}(i)$ ， 其中 $f_{out}(i)$为模块i的扇出

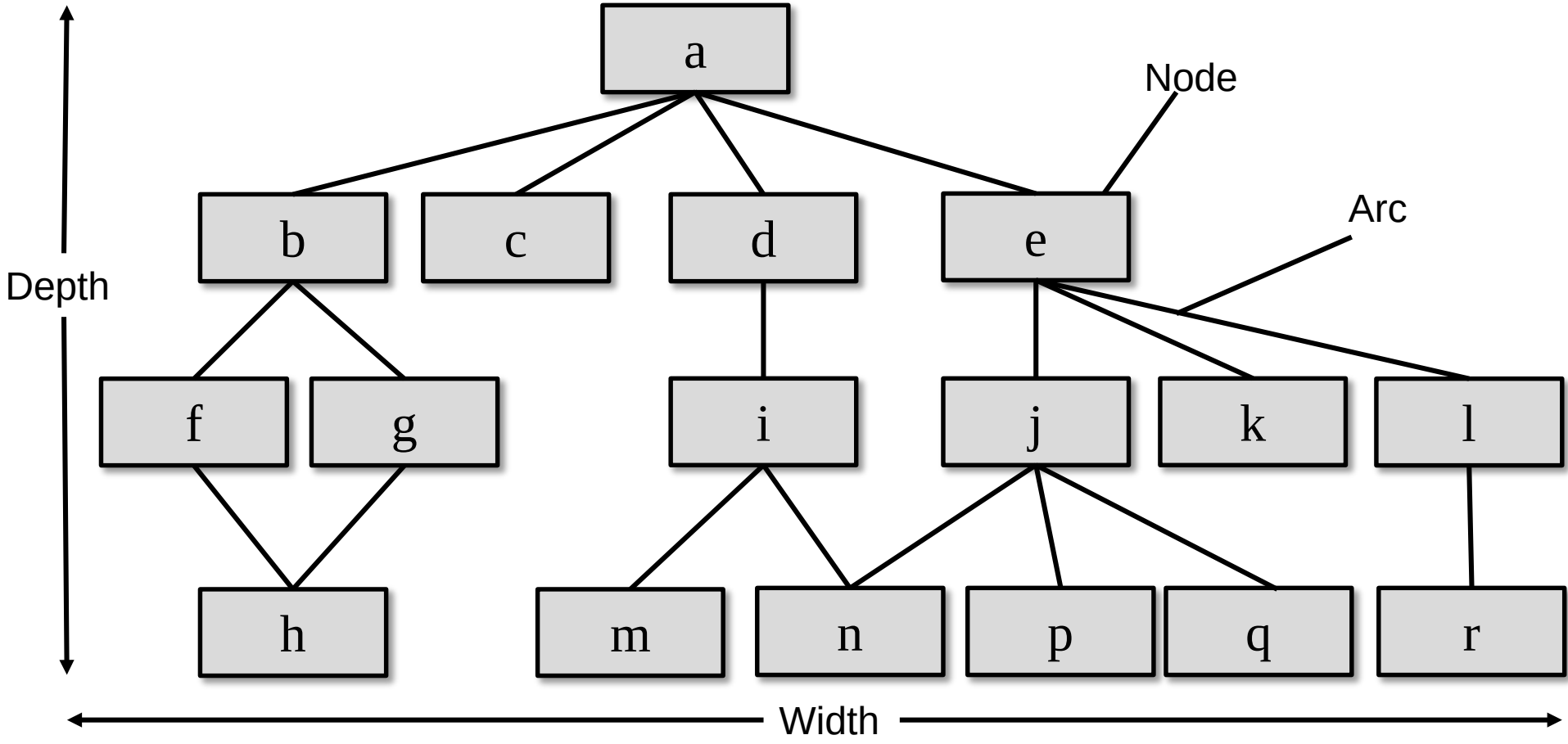
- 数据复杂性

  $D(i)=\dfrac{v(i)}{[f_{out}(i)+1]}$， 其中 $v(i)$为模块i的输入、输出变量数

- 系统复杂性

  $C(i)=S(i)+D(i)$

Fenton简单形态度量



- Size = $n + a$ = 17+18 =35
- Depth = 4
- Width = 6

- $r = a/n$ = 18/17 = 1.06

r为该体系结构中模块连接紧密程度的度量值

OOD度量---MOOD Metrics Suite

- Method inheritance factor (MIF)

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

- 其中，TC是所有类的数量，C_i是其中的一个类

$$M_a(C_i) = M_d(C_i) + M_i(C_i)$$

- $M_a(C_i)$为和C_i中能够被调用的方法的数量
- $M_d(C_i)$为类C_i中申明的方法的数量
- $M_i(C_i)$为类C_i中继承的方法的数量

OOD度量---MOOD Metrics Suite

- Coupling factor (CF)

$$\frac{\sum_{i=1}^{TC}\left[\sum_{j=1}^{TC} is\_client(C_i, C_j)\right]}{TC^2 - TC}$$

其中，函数 $is\_client(C_c, C_s) = \begin{cases} 1 & iff\ C_c \Rightarrow C_s \wedge C_c \neq C_s \\ 0 & otherwise \end{cases}$

源代码度量---Halstead方法

- Halstead方法根据程序中运算符和操作数的总数来度量程序的复杂程度。
- 令N_1为程序中运算符出现的总次数，N_2为操作数出现的总次数，程序长度N定义为：

$$N=N_1+N_2$$

- 程序中使用的不同运算符(包括关键字)的个数n_1，以及不同操作数(变量和常数)的个数n_2。预测程序长度的公式如下：

$$H = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

- 多次验证都表明，程序的预测长度H和实际程序长度N非常接近。
- 预测程序中包含错误的个数的公式如下：

$$E = N \log_2 (n_1+n_2)/3000$$

代码行（Lines of Code）

- 代码行技术是比较简单的定量估算软件规模的方法。
- 依据以往开发类似产品的经验和历史数据，估计实现一个功能所需要的源程序行数。
- 当有以往开发类似产品的历史数据可供参考时，估计出的数值还是比较准确的。把实现每个功能所需要的源程序行数累加起来，就可得到实现整个软件所需要的源程序行数。

估算方法：

- 由多名有经验的软件工程师分别做出估计。
- 每个人都估计程序的最小规模(a)、最大规模(b)和最可能的规模(m)，
- 分别算出这3种规模的平均值、和之后，再用下式计算程序规模的估计值：

$$L = \frac{\bar{a} + 4\bar{m} + \bar{b}}{6}$$

单位：

LOC或KLOC。

LOC & FP

Programming language	LOC per FP			
	Avg.	Median	Low	High
Ada	154	-	104	205
ASP	56	50	32	106
Assembler	337	315	91	694
C	148	107	22	704
C++	59	53	20	178
C#	58	59	51	704
FORTRAN	90	118	35	-
HTML	43	42	35	53
Java	55	53	9	214
JSP	59	-	-	-
Javascript	54	55	45	63