软件成本估算

- 预测一个软件开发过程所需要的资源

目标

- 软件成本计算和软件报价的基本原理以及在它们之间存在的复杂关系。
- 对软件生产率评估的度量。
- 在对软件成本和进度安排进行评估时应当使用一系列不同的技术。
- 用于算法成本估算的COCOMO 2模型的原理

内容

- 生产率
- 估算技术
- 算法成本建模
- 项目的工期和人员配备

估算的基本问题

- 完成一项活动需要多少工作量？ 人·天
- 完成一项活动需要多少时间？ 小时、天、月、年
- 一项活动的总成本是多少？ 元
- 项目估算和进度安排是交叉进行的管理活动

软件成本构成

- 硬件和软件成本
- 差旅费和培训费用
- 工作成本 (大多数项目中是主要成本)
  - 项目中投入的工程师的工资
  - 社会和保险费用
- 工作成本还要计入
  - 办公场所、供热和照明费用
  - 网络和通信费用
  - 共用设施的费用 (e.g 图书馆,员工餐厅, etc.)

成本和报价

- 估算是为了得到承包商开发一个软件的成本
- 在成本和报价之间没有一个简单的关系

  价格＝成本＋利润
- 广泛的因素包括机构的、经济的和商务上的考虑会影响报价

影响软件报价的因素

因素	描述
市场机遇	开发机构为了便于进入一个新的软件市场，可能会提出一个低报价。
成本估算的不确定性	如果机构对成本估算没有把握，它会提高报价，在正常利润之上增加某些意外费用。
合同条款	客户可能愿意让开发者保有程序源代码的所有权，以便可以在未来的开发项目中复用。这时的要价就会比移交源代码的情况低得多。
需求易变性 大公司通常会主动提高价钱；但小公司不一定	如果需求很有可能发生变化，机构就会降低报价以赢得合同，当得到合同之后，一旦需求有所变更，机构就会乘机抬高报价。
经济状况	当开发者处于资金短缺阶段时，为得到合同而作出较低报价，少获点利甚至收支相抵总比破产强。或者行情不好，没什么生意，又不能让员工闲着

程序员的生产率

- 软件开发工程师生产软件和相关文档的效率的度量
- 不是面向质量的，虽然质量保证是生产率评估的一个因素
- 本质上，我们是想度量单位时间里生产的有用功能

生产率的度量

- 面向规模的度量

这种方法是根据活动输出的量来衡量。可能是提交的源代码行数，目标代码说明书的长度或者系统文档的页数等。

- 面向功能的度量

这种方法是看移交软件总的功能有多少。功能点和对象点方法是最常用的方法。

度量的问题

- 估算软件规模
- 估算总的程序员人月数
- 估算分包商生产率，并且合并到总的估算中

代码行

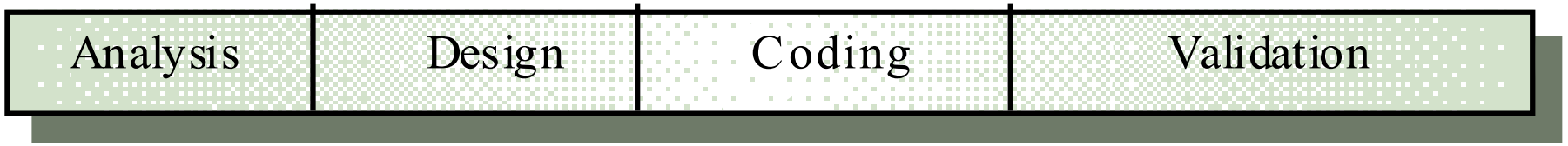
- 什么是一个代码行?
  - 当程序打印在卡片上时代开始就使用的度量方法，一行一张卡片
  - 代码行如何跟语句对应起来，一个语句可能占几行，一行也可能有几个语句。
- 什么程序应该算作系统的一部分?
- 假设在系统大小和源代码大小之间有着线性关系

生产率比较的问题

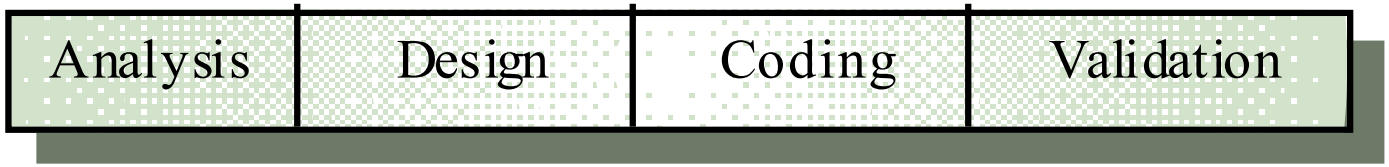
- 编程语言越低级，计算出来的生产率越高
  - 同样的功能，低级语言需要更多代码
- 程序员的代码越冗长，计算出来的生产率越高
  - 生产率计算是基于程序员所写的代码量的

高级和低级语言

Low-level language



High-level language



系统开发时间

	Analysis	Design	Coding	Testing	Documentation
Assembly code	3 weeks	5 weeks	8 weeks	10 weeks	2 weeks
High-level language	3 weeks	5 weeks	8 weeks	6 weeks	2 weeks

	Size	Effort	Productivity
Assembly code	5000 lines	28 weeks	714 lines/month
High-level language	1500 lines	20 weeks	300 lines/month

功能点

1.功能点分类
2.为每个类别赋予权值
3.乘积和

- 基于程序特性的组合
  - 外部输入输出
  - 用户交互
  - 外部接口
  - 系统使用的文件
- 每一项都有一个复杂性权值（3～15）
- 功能点计数就是将每项功能点数乘以权值，然后求和的结果。

功能点

- 功能点计数由项目复杂性修正
- 根据给定语言的每功能点平均代码行数，功能点计数可以用来计算代码行数
  - LOC = AVC * FPs
  - AVC 是基于语言的因子，汇编语言200-300， 4GL则2-40
- 功能点计数是非常主观的，依赖于估算者。
  - 自动功能点计数是不可能的。

对象点

4GL：第四代编程语言，面向问题，只需要告诉计算机“做什么”
JAVA属于3GL

- 当使用高级语言（特别是4GL）开发时，作为另一个功能相关方法，对象点方法可以代替功能点方法
- 对象点不同于对象类
- 程序的对象点是下列内容的加权估算：
  - 独立的显示屏幕数
  - 生成的报表数
  - 为辅助4GL代码而必须开发的3GL模块数

对象点估算

- 对象点估算比功能点容易，因为它只关心屏幕数、报告数和3GL模块数
- 可以在开发过程的很早期使用，这时要估计系统的代码行还很困难。

生产率估算

- 实时嵌入式系统, 40-160 LOC/人月
- 系统程序, 150-400 LOC/人月
- 商业应用, 200-800 LOC/人月
- 在对象点方法中，随支持工具和开发者能力不同，生产率在 4～50对象点/人月

影响生产率的因素

因素	描述
应用领域经验 先前经验	应用领域知识是有效的软件开发的根本。已经对领域有充分了解的工程人员很可能是生产率最高的。
过程质量	所用的开发过程对生产率有极大的影响。
项目规模 合作成本	项目规模越大，团队之间的交流和沟通就越花时间，真正有用的时间就会减少，因而，个人生产率就会降低。
技术支持	好的支持技术和CASE工具、配置管理系统等有助于提高生产率。
工作环境	一个好的工作环境有助于提高生产率。

质量和生产率

- 所有基于单位时间内产量的做法都有问题，因为没有考虑质量
- 以质量为代价，通常都能提高生产率
- 还不清楚生产率和质量之间的关系
- 变更不断的情况下，使用代码行生产率计算是没有意义的。

通常考量生产率，质量不容易考量

估算技术

- 没有一个简单的方法可以精确估算软件开发所需的资源
  - 初始估算是基于用户需求定义中不充分的信息
  - 软件可能是运行于不熟悉的计算机系统或者使用新的技术
  - 项目中的开发人员不了解
- 项目成本估算可能是自己设定的
  - 用来确定项目预算和调整软件产品以保证预算不被突破

估算技术

- 算法成本建模
- 专家判定
- 类比估计
- Parkinson定律
- 根据客户预算报价

算法成本建模

- 所建立的模型利用有关历史信息来估计需要的工作量，用到的历史信息是有关某些软件度量（例如规模）与项目成本之间的关系
- 本章后续讨论

专家判定

- 多位软件开发和应用领域方面的专家用他们的经验预测软件成本。反复进行直到达成一致。
- 优点：相对廉价的估算方法。如果专家具有相似系统的直接经验的话，可以比较精确。
- 缺点：如果没有专家的话，将非常不精确。

类比估计

- 将项目和一个同样应用领域里的类似项目作比较，从而计算出成本。
- 优点: 如果项目数据具备的话比较精确
- 缺点: 如果没有可比的项目，则不可能应用。需要系统性维护的成本数据库

Parkinson定律

- 工作占满所有可用的时间。成本决定于所有可用的资源而不是客观的估算。
- 优点: 不会超支
- 缺点: 系统经常完不成

根据客户预算报价

- 将客户对项目的预算作为软件的成本
- 优点: 得到了合同
- 缺点: 成本无法精确反映工作所需。

自上而下和自下而上的估算

- 以上的方法都可以是自上而下或自下而上的
- 自上而下
  - 从系统层面开始，评估系统的总体功能以及如何通过子系统间的交互完成的。
- 自下而上
  - 从组件层面开始，估算每个组件所需的工作量。将这些工作量加在一起得到最后的估算。

自上而下：
适合体系结构、组件识别未知的情况；
容易低估解决底层技术难题的成本；

自下而上：
适合体系结构、组件识别尤其是详细设计已知的情况；
容易低估系统级的集成、配置管理、文档的成本

自上而下的估算

- 无需系统体系结构以及组件的知识就可应用
- 成本考虑集成、配置管理和文档等
- 可能会低估解决低层技术难题的成本

自下而上的估算

- 当体系结构和组件识别已知的情况下使用
- 如果系统已经得到详细设计时是比较精确的方法
- 可能会低估系统级活动的成本，如集成和文档等

估算方法

- 每个方法都有优缺点
- 估算应该基于多种方法
- 如果得到的结果相差甚远，意味着信息不够充分
- 为了得到更精确的估算，需要额外采取行动
- 有时候根据客户预算报价是唯一可用的方法。

基于经验的估算

- 从根本上说估算是基于经验的
- 然而，一些新的方法和技术导致基于不精确的经验的估算
  - 面向对象开发而非面向功能的开发
  - 客户机/服务器系统而非基于主机的系统
  - 使用现成的软件组件
  - 基于组件的可复用的软件工程
  - 使用CASE工具和程序生成器

根据客户预算报价

- 这个方法看似不道德的而且是不实事求是的
- 然而，当详细信息缺乏的时候，它可能是唯一合适的策略
- 项目成本基于开发大纲而定，成本是开发的限制因素。
- 开发商和客户之间通过协商建立详细的项目描述，或者使用一个进化式开发方法。

算法成本建模

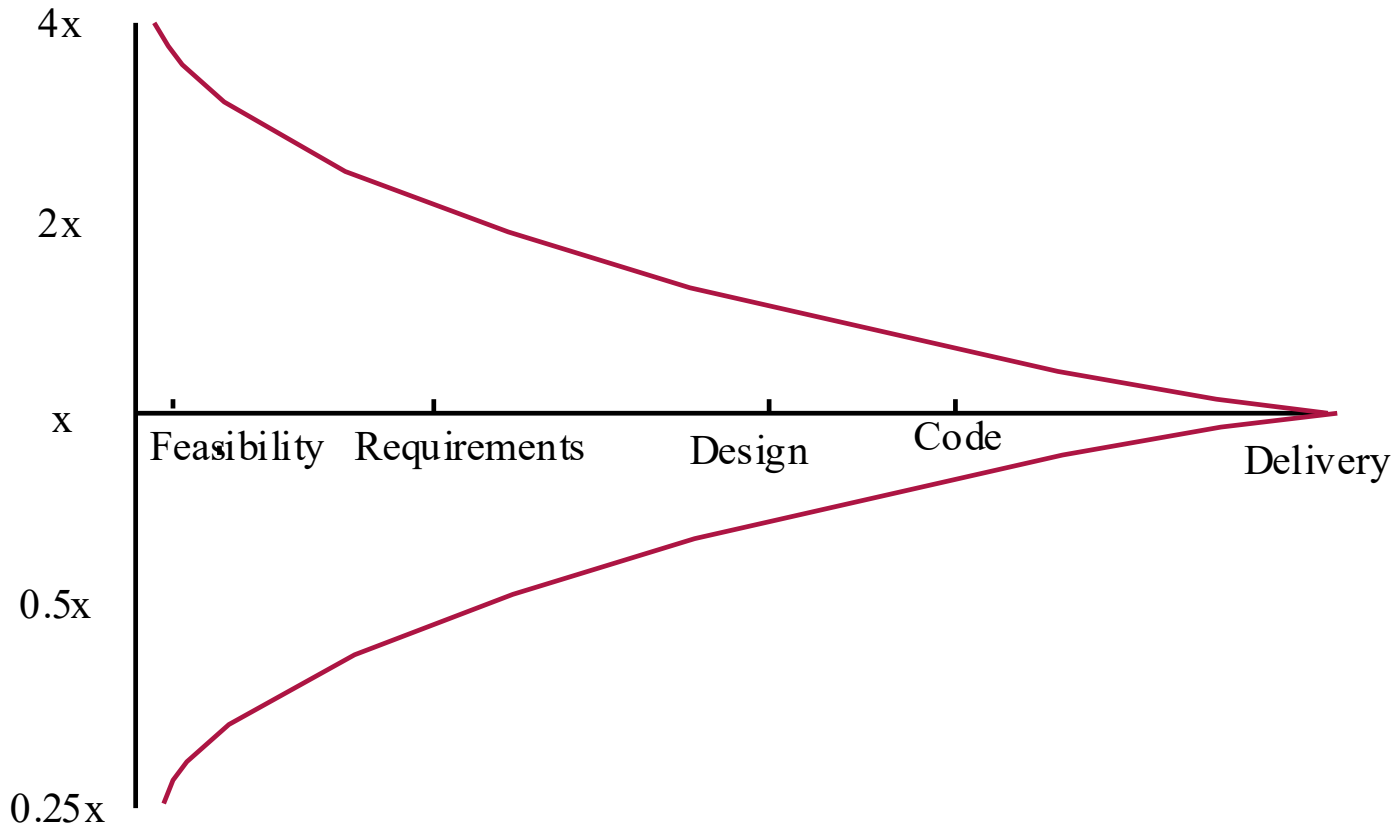
- 通过对项目规模、程序员数量以及其他过程和产品因素的估算，用一个数学公式来估算项目的成本。
  - 工作量 = A × Size^{B} × M

    Size为项目规模
  - A 是反映机构情况的常数因子，B 反映了大型项目工作量的非线性因子，M 是一个乘数因子，反映了不同过程、产品及开发特征的混合因素。
- 最常用的产品特性是代码规模
- 大多数模型都是相似的，只是A，B和M的值不同

估算的精度

- 只有系统完成后才能精确知道软件的规模
- 一些因素影响最后的规模
  - 使用现货产品和组件
  - 编程语言
  - 系统的分布
- 随着开发过程的进展，代码规模的估算变得更加精确

估算的不确定性


4x
2x
x
0.5x
0.25x
Feasibility
Requirements
Design
Code
Delivery

COCOMO模型

- 是一个基于项目经验的经验模型
- 是一个独立的模型，不限于特定的软件开发商
- 有较长的历史，初始版本发布于1981年(COCOMO-81)，不断改进得到了 COCOMO 2
- COCOMO 2考虑了不同的软件开发方法

COCOMO 81

项目复杂度	公式	描述
简单	$PM = 2.4\ (KDSI)^{1.05} \times M$	对应用理解充分，由小团队开发
一般	$PM = 3.0\ (KDSI)^{1.12} \times M$	项目较复杂，团队成员对该系统的相关系统可能有些经验
嵌入式	$PM = 3.6\ (KDSI)^{1.20} \times M$	项目复杂，软件只是复杂系统的一部分，该复杂系统是由软件、硬件、规则和操作规程紧密结合而成的

COCOMO 2 的分层

- COCOMO 2 是一个3层模型，随着开发进展允许有日益具体的估算
- 早期原型层
  - 规模估算是基于对象点的，使用一个简单的规模/生产率计算公式估算所需工作量
- 早期设计层
  - 估算基于功能点，然后把这些功能点转换成源代码行数
- 后体系结构层
  - 估算基于源代码行数

早期原型层

- 支持原型开发项目和大量复用的项目
- 基于开发者每月对象点生产率的标准估算
- 考虑了CASE工具的使用
- 估算公式：
  - PM = (NOP × (1 - %reuse/100)) / PROD
  - PM 是每人月工作量, NOP 对象点数， PROD是生产率，％reuse是所期望的复用率

对象点生产率

开发者的经验和能力	非常低	低	一般	高	非常高
CASE 成熟度和能力	非常低	低	一般	高	非常高
PROD (NOP/month)	4	7	13	25	50

早期设计层

- 需求确定后就可以做早期设计层的估算
- 基于算法模型的标准公式
  - $PM = A \times Size^{B} \times M + PM_m$ 其中
  - $M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$
  - $PM_m = (ASLOC \times (AT/100)) / ATPROD$
  - A = 2.5, Size用KLOC表示, B根据项目的新颖程度、开发灵活性、风险管理方法和过程成熟度的不同从1.1～1.24不等

乘法因子

- 乘法因子反映了开发者的能力、非功能需求、对开发平台的熟悉程度等。
  - RCPX – 产品的可靠性和复杂性
  - RUSE – 要求的复用性
  - PDIF – 平台的难度
  - PREX – 个人经验
  - PERS – 个人能力
  - SCED – 要求的进度
  - FCIL – 团队支撑设施
- PM_m 反映了自动生产代码的总量

后体系结构层

- 使用和早期设计层同样的公式
- 对代码行数的估算要考虑以下修正因素
  - 需求的易变性。对需求变更的返工工作量进行估算。
  - 可能的复用程度。复用是非线性的，其相应的成本不能通过简单的减少单位时间内的LOC
  - ESLOC = ASLOC×(AA + SU +0.4DM + 0.3CM +0.3IM)/100
    - » ESLOC是新的代码行数。ASLOC是必须修改的复用代码行数。DM是设计修改的百分比。CM是代码修改的百分比。IM是集成复用软件所需的最初费用的百分比。
    - » SU 是一个反映理解软件难易的因子, AA反映为确定软件是否可复用所做的初始评估费用。

指数因子

- 依赖于5个比例因子 (见下一页). 它们的和除以100再加上1.01就是该指数项的取值了
- 举例
  - 先例的援引 – 新项目 - 取值为“低”（4）
  - 开发的灵活性– 没有客户介入– 取值为“非常高”（1）
  - 体系结构/风险解决 – 没有风险分析 –“非常低”（5）
  - 团队凝聚力 – 新团队 – “一般”（3）
  - 过程成熟度 – 有一些过程控制 – “一般”（3）
- 所以比例因子是 1.17

指数比例因子

比例因子	解释
先例的援引	反映机构对这个类型项目的经验。“非常低”代表无经验，“非常高”代表机构对该领域已经彻底了解了
开发的灵活性	反映开发过程中的灵活性程度。“非常低”代表使用事先规定的过程，“非常高”代表客户只给出了总的目标
体系结构/风险解决	反映风险分析完成的程度，“非常低”代表无分析，“非常高”代表完全和彻底的风险分析
团队凝聚力	反映开发团队成员相互了解的程度和协作的程度。“非常低”代表交互非常困难，“非常高”代表这是一个团结高效的团队，没有沟通障碍。
过程成熟度	反映机构的过程成熟度。该值的计算依赖于CMM的成熟度调查问卷，对它的估算可以用5减去CMM过程成熟度等级得到

乘数因子

- 产品属性
  - 关于所要开发的软件产品的要求特性
- 计算机属性
  - 硬件平台的约束
- 人员属性
  - 考虑开发人员的经验和能力
- 项目属性
  - 关于软件开发项目的特征

项目成本形成因素

产品属性			
RELY	要求的系统可靠性	DATA	所用的数据库规模
CPLX	系统模块的复杂性	RUSE	所需的复用组件百分比
DOCU	所需文档编制的范围		
计算机属性			
TIME	执行时间约束	STOR	内存约束
PVOL	开发平台的易变性		
人员属性			
ACAP	项目分析的能力	PCAP	程序员能力
PCON	人员的连续性	AEXP	分析人员的项目领域经验
PEXP	程序员的项目领域经验	LTEX	语言和工具经验
项目属性			
TOOL	软件工具的使用	SITE	办公地点的分散程度以及办公地点之间的通信质量
SCED	开发进度压缩		

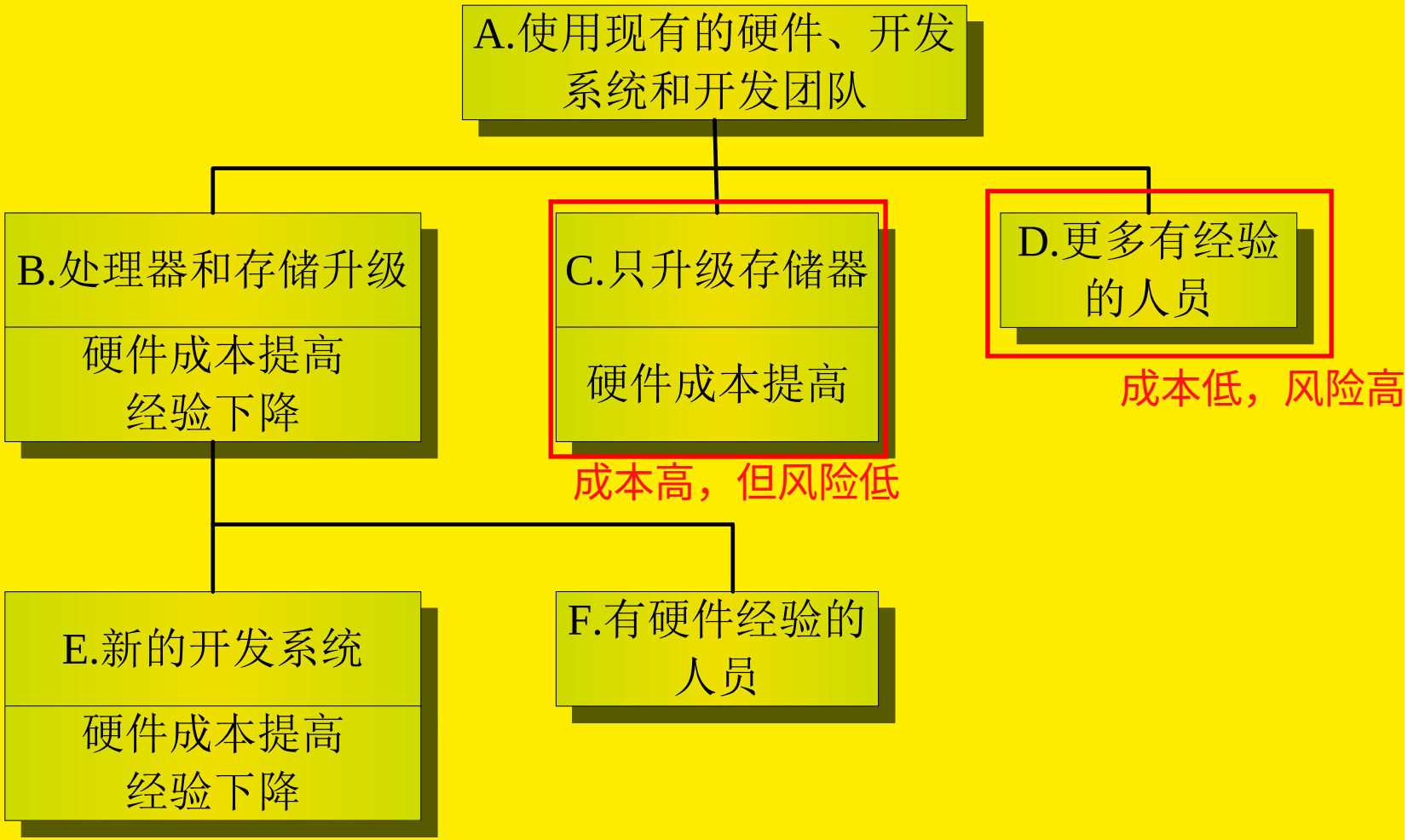
成本形成因素对工作量估算的影响

指数值 系统规模(包括复用和需求易变性因素) **不考虑成本形成因素的初始 COCOMO 估算**	1.17 128, 000 DSI **730 人月**
可靠性 复杂性 内存约束 工具使用 进度 **修正后的 COCOMO 估算**	非常高,因子取值 = 1.39 非常高, 因子取值 = 1.3 高,因子取值 = 1.21 低,因子取值 = 1.12 加快,因子取值 = 1.29 **2306 人月**
可靠性 复杂性 内存约束 工具使用 进度 **修正后的 COCOMO 估算**	非常低,因子取值 = 0.75 非常低,因子取值 = 0.75 无, 因子取值 = 1 非常高,因子取值 = 0.72 正常, 因子取值 = 1 **295 人月**

项目计划

- 算法成本模型为项目计划提供了基础，因为不同的算法成本模型提供了可供比较的不同的策略，以减少项目成本
- 举例：嵌入式太空船控制系统
  - 必须可靠
  - 必须最小化重量 (芯片的数量)
  - 可靠性和计算机约束的乘法因子 > 1
- 成本构成
  - 目标硬件
  - 开发平台
  - 所需工作量

管理选项


A.使用现有的硬件、开发系统和开发团队
B.处理器和存储升级
硬件成本提高
经验下降
C.只升级存储器
硬件成本提高
成本高，但风险低
D.更多有经验的人员
成本低，风险高
E.新的开发系统
硬件成本提高
经验下降
F.有硬件经验的人员

管理选项的成本

选项	RELY	STOR	TIME	TOOLS	LTEX	总工作量	软件成本	硬件成本	总成本
A	1.39	1.06	1.11	0.86	1	63	949393	100000	1049393
B	1.39	1	1	1.12	1.22	88	1313550	120000	1402025
C	1.39	1	1.11	0.86	1	60	895653	105000	1000653
D	*1.39*	*1.06*	*1.11*	*0.86*	*0.84*	*51*	*769008*	*100000*	*897490*
E	1.39	1	1	0.72	1.22	56	844425	220000	1044159
F	1.39	1	1	1.12	0.84	57	851180	120000	1002706

选项的选择

- 选项D(使用更多有经验的人员)似乎是最好的选择
  - 然而，它有个高风险因数，因为有经验的人员难以找到
- 选项C(只升级存储器)节省成本较少但是风险很低
- 总之，这个模型显示了软件开发中人员经验的重要性。

项目工期和人员配备

- 在估算工作量的同时，管理者必须估计一个项目的开发周期，以及人员要在什么时候到位
- 项目所需的日历时间可以用COCOMO 2公式计算
  - $TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))}$
  - PM 是工作量计算值，B是上面所计算的指数项（对于早期原型模型，B=1)。通过这个计算预计了项目的名义进度。
- 所需时间不依赖于项目中的工作人员数

人员要求

- 要求的人员不能单纯用所需工作量除以开发时间来计算。合作成本，管理成本
- 项目不同时期，投入的工作人员数量是变化的
- 项目中的人员越多，往往要求的总工作量越大
- 草率的配置人员往往导致项目进度的拖延

要点

- 影响生产率的因素包括个人能力、领域经验、开发过程、项目规模、工具支持和工作环境。
- 应使用多种软件成本估算技术，如果结果相差较大，则说明估算信息不充分。
- 软件通常是先有合同金额，然后再据此调整相应的功能。

要点

- 算法成本估算是困难的，因为需要估计将要完成产品的特征。
- COCOMO模型预测成本时，要考虑项目、产品、人员和硬件的因素。
- 算法成本模型支持量化的选项分析。
- 完成一个项目所需要的时间和参加项目的人数不成比例。