

# 软件体系结构

# Software Architecture

张莉

北京航空航天大学

软件工程研究所

*[lily@buaa.edu.cn](mailto:lily@buaa.edu.cn)*



# 自我介绍

张莉 教授，博士生导师

- 主要研究方向：软件工程、软件体系结构、需求工程、系统建模、经验软件工程。

**我们便可以互为师生。**

“孔子曰：‘三人行，必有我师。’是故弟子不必不如师，师不必贤于弟子。闻道有先后，术业有专攻，如是而已”

———韩愈《师说》

- 教学内容 } 学什么?
- 参考资料 }
- 学习方法 } 如何学?
- 考试方法 } 学习的结果?
- 每年的教学内容侧重点不同，作业可能不同

# 什么是软件体系结构？



- 你的背景
  - 你开发过的最大项目（代码、模块、人数）
  - 你对软件体系结构的理解
- 你的期望
  - 你期望的收获

- 一个形象的比喻：

开发一个具有一定规模和复杂性的软件系统和编写一个简单的程序大不一样。其间的差别，借用G. Booch的比喻，如同建造一座大厦和搭一个狗窝的差别。

# Architecting a dog house



Can be built by one person  
Requires  
Minimal modeling  
Simple process  
Simple tools



# Architecting a house

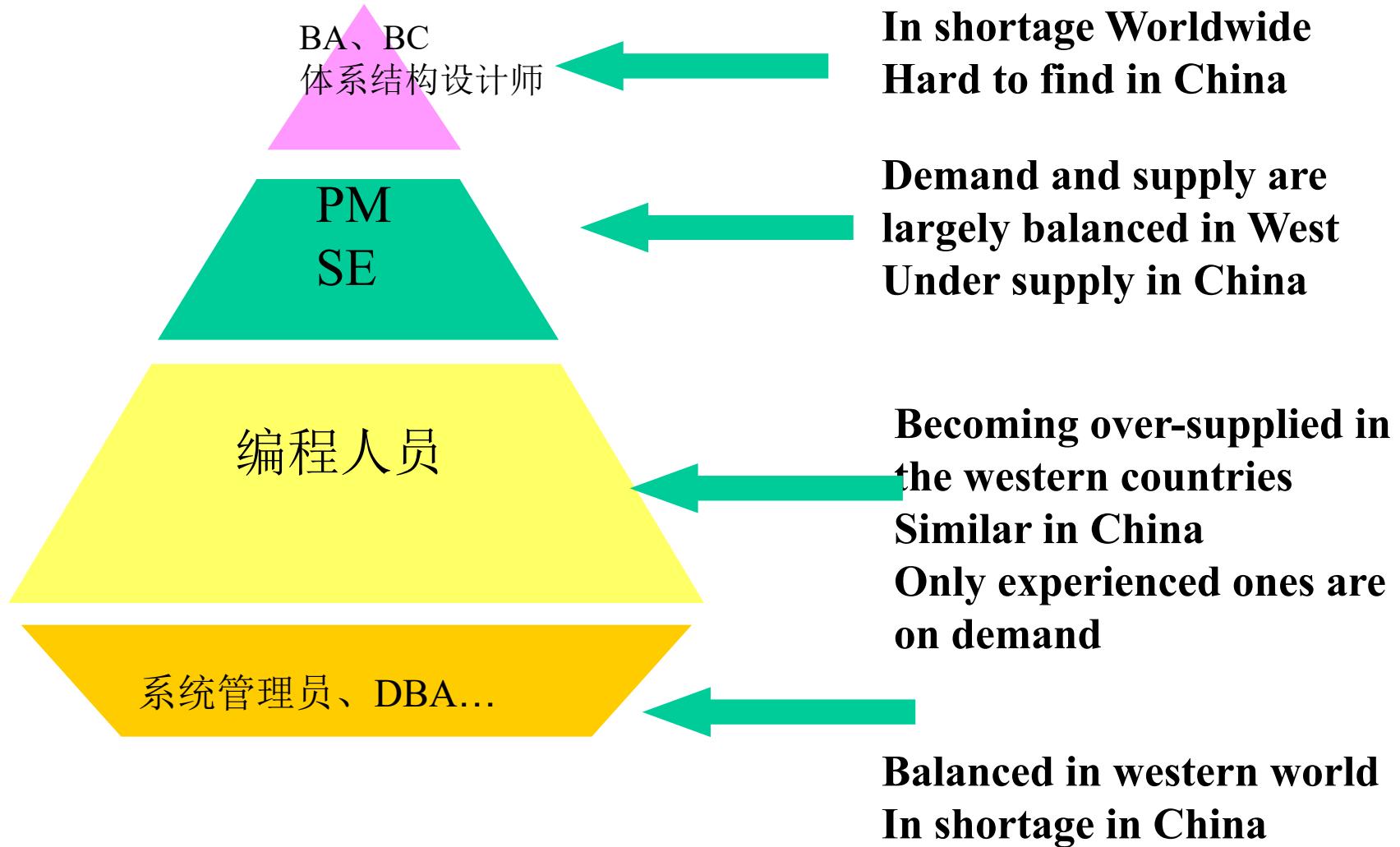


Built most efficiently and timely by a team  
Requires

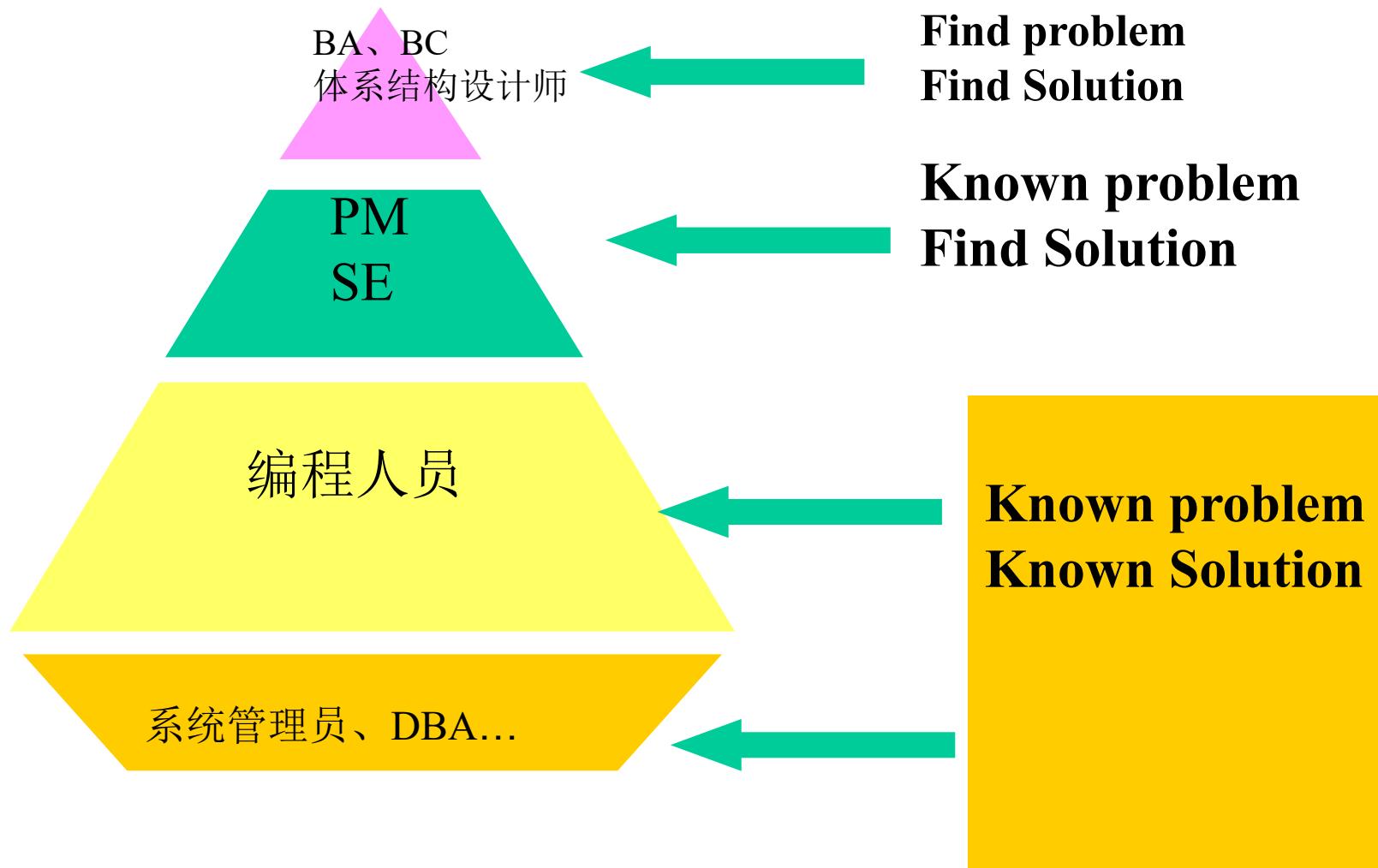
- Modeling
- Well-defined process
- Power tools

# Architecting a high rise





# IT 行业不同职业的差别



- 布卢姆的认知分类法由六种不同认知层次的思维水平所组成(Bloom 1956)。
  - 1、识记 (knowledge)。最底层次的认知水平。要求学生回忆信息，识别事实、定义和规则。
  - 2、理解 (comprehension)。要求学生能够改变交流的形式，能够转述或重新组织读过和讲过的知识。
  - 3、应用 (application)。要求学生应用所知的事实、原则和归纳于新的环境。
  - 4、分析 (analysis)。要求学生能够将问题分成几部分，并能就各个部分之间建立联系。
  - 5、综合(synthesis)。要求学生综合各种要素和各个部分以形成一个整体，构建出对一个问题的独特新颖的回答。
  - 6、评价(evaluation)。这是最高级的认知水平。要求学生能够按照一定的标准对不同的方法、思想、人物或产品的价值做出判断。

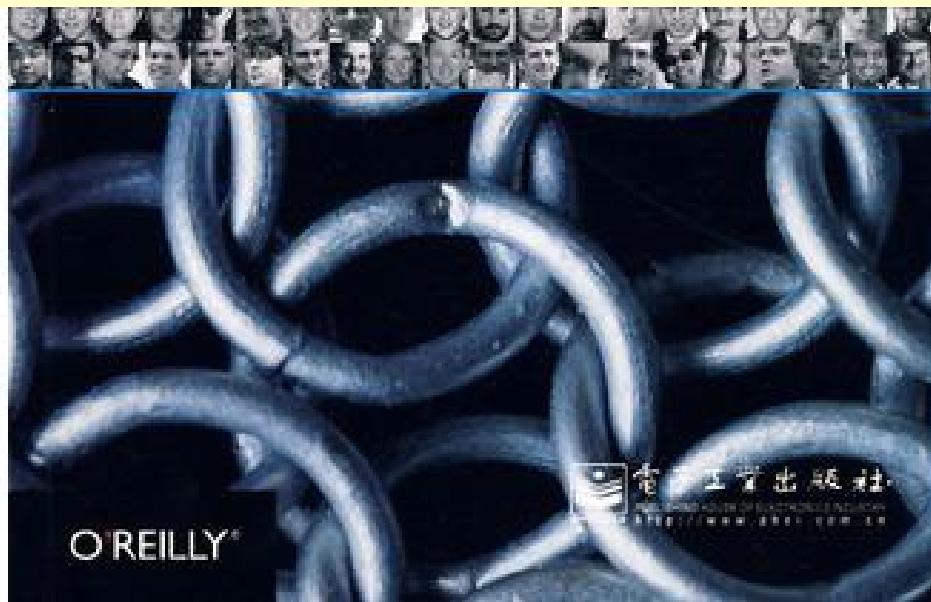
- 总体上鼓励大家：
  - Find Problem, Find Solution, 去创新
- 毕业生的把握：
  - 本科生： Known Problem, Known Solution
  - 硕士生： Known Problem, Find Solution
  - 博士生： Find Problem, Find Solution
- 这种能力是可以通过训练得到的！

最近，还看到一个兔子理论。

- 华罗庚：比喻导师和研究生的关系是：
  - 导师负责给研究生指出兔子在哪里，并指导学生学会打兔子的本领。反之，研究生则是从导师那里了解到兔子的位置、大小、肥瘦，并采用从导师那里学到的打兔子本领擒获一只兔子（就是做完论文）。
- 有人由此衍生出本科生、硕士生、博士生、博士后之区别的“兔子理论”
  - 本科生：学习捡“死”兔子。
  - 硕士生：学习打一只在视野中奔跑的活兔子。
  - 博士生：学习打一只看不到的活兔子。
  - 博士后阶段：（兔子在哪里？）



- 软件架构师应该知道的97件事（领导力、技能、思维模式、沟通、博弈……）



- 系统架构师是一个最终确认和评估系统需求，给出开发规范，搭建系统实现的核心构架，并澄清技术细节、扫清主要难点的技术人员。主要着眼于系统的“技术实现”。因此他/她应该是特定的开发平台、语言、工具的大师，对常见应用场景能马上给出最恰当的解决方案，同时要对所属的开发团队有足够的了解，能够评估自己的团队实现特定的功能需求需要的代价。
- 系统架构师负责设计系统整体架构，从需求到设计的每个细节都要考虑到，把握整个项目，使设计的项目尽量效率高，开发容易，维护方便，升级简单等。

- 软件系统架构师综合的知识能力包括9个方面，即：
- 1、战略规划能力。
- 2、业务流程建模能力。
- 3、信息数据结构能力。
- 4、技术架构选择和实现能力。
- 5、应用系统架构的解决和实现能力。
- 6、基础IT知识及基础设施、资源调配能力。
- 7、信息安全技术支持与管理保障能力。
- 8、IT审计、治理与基本需求分析、获取能力。
- 9、面向软件系统可靠性与系统生命周期的质量保障服务能力。

- 作为系统架构师，必须成为所在开发团队的技术路线指导者；具有很强的系统思维的能力；需要从大量互相冲突的系统方法和工具中区分出哪些是有效的，哪些是无效的。架构师应当是一个成熟的、丰富的、有经验的、有良好教育的、学习快捷、善沟通和决策能力强的人。丰富是指他必须具有业务领域方面的工作知识，知识来源于经验或者教育。他必须广泛了解各种技术并精通一种特定技术，至少了解计算机通用技术以便确定那种技术最优，或组织团队开展技术评估。优秀的架构师能考虑并评估所有可用来解决问题的总体技术方案。需要良好的书面和口头沟通技巧，一般通过可视化模型和小组讨论来沟通指导团队确保开发人员按照架构建造系统。

- 一、系统架构相关的知识和经验。
- 二、很强的自学能力、分析能力、解决问题的能力。
- 三、写作、沟通表达、培训。

- 一般来讲，系统架构师应该拥有以下几方面的能力：
  - 1：具备 8 年以上软件行业工作经验；
  - 2：具备 4 年以上 C/S 或 B/S 体系结构软件产品开发及架构和设计经验；
  - 3：具备 3 年以上的代码编写工作经验；
  - 4：具备丰富的大中型开发项目的总体规划、方案设计及技术队伍管理经验；
  - 5：对相关的技术标准有深刻的认识，对软件工程标准规范有良好的把握；
  - 6：对 .Net/JAVA 技术及整个解决方案有深刻的理解及熟练的应用，并且精通WebService/J2EE 架构和设计模式，并在此基础上设计产品框架；
  - 7：具有面向对象分析、设计、开发能力（OOA、OOD、OOP），精通 UML 和 ROSE，熟练使用 Rational Rose、PowerDesigner 等工具进行设计开发；
  - 8：精通大型数据库如 Oracle、Sql Server 等的开发；
  - 9：对计算机系统、网络和安全、应用系统架构等有全面的认识，熟悉项目管理理论，并有实践基础；
  - 10：在应用系统开发平台和项目管理上有深厚的基础，有大中型应用系统开发和实施的成功案例；
  - 11：良好的团队意识和协作精神，有较强的内外沟通能力。

## 一、软件体系结构概述

- 什么是软件体系结构,为什么研究软件体系结构?
- 软件体系结构的重要性,软件体系结构研究领域
- 一个实例

## 二、软件体系结构建模与描述

- 软件体系结构模型及建模 (MDA、UML)
- 软件体系结构描述语言
- 软件体系结构文档(documentation)

## 三、创建软件体系结构

- 理解质量属性
- 实现质量属性: 软件体系结构风格、软件设计模式、软件体系结构设计的原子操作、软件体系结构实现战术
- 案例分析 (OS, 等)

## 四、软件体系结构分析和评估

- 软件体系结构分析
- 软件体系结构评审

## 五、面向领域的软件体系结构

- 特定领域的软件体系结构
- 软件产品线

## 六、网络环境下的软件体系结构

- SOA
- 网构软件
- 网络化软件

可能有变动----根据教学过程调整

- Reference books:

- <http://www.Course.buaa.edu.cn>
- "**Software Architecture in Practice**" Len Bass,  
**Paul Clements, Rick Kazman(1st Version)**
- "**Software Architecture in Practice**" Len Bass,  
**Paul Clements, Rick Kazman(2nd Version)**
  - 软件构架实践（第二版）
  - 软件架构实践（第三版，英文印影版，2013, 2016）

- Documenting Software Architectures, Paul等（中译本：软件架构编档）
- Software Product Lines, Practices and Patterns, Paul Clements, Linda Northrop
  - (软件产品线实践与模式，张莉等译)
- Evaluating Software Architectures, Paul等
  - 软件架构评估
- Design patterns-- Elements of Reusable Object Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
- 《恰如其分的软件架构》 **George Fairbanks, 2016年**
- 还有要求大家阅读的论文。

- 你能学到什么？
- 你不能学到什么？
- 你可能不能完全听懂，这时你要学会提问。
  - 具体的问题，而不是抽象的问题。
  - 给出一个例子，我们来分析。
- 强调观念的更新和思路
  - 设计在整个软件生命周期中的重要性
  - 学会如何设计一个复杂系统，学会在系统层面思考问题，学会多问为什么
- 积累靠自己

# 学习方法

课堂上

授课—老师讲

讨论问题—同学讲

回答问题—同学讲、互相讨论

示范作业—同学讲

你的参与是课程成功的基础

目标：知识变成你的能力

请大家关闭手机！

- 32学时： 1-16周
- 每周四下午 2 :00-3:50。
- 地点
  - 新主楼A209
- 助教：田家豪 刘一帆
  - 地点：G312
  - 请大家按时提交作业、高质量完成ppt
  - <http://judge.sei.buaa.edu.cn/>

- 2014年被研究生院选中全程听课
  - 整体评价很好：内容、讲课
  - 问题：
    - 老师不点名，存在学生缺课情况 ——后来就点名了
    - 作业太少，太容易 ——越来越难了
- 2016年，有学生反应，考试简单了。
- 2017年，有同学建议可否设计一个系统
- 今年继续整改措施：
  - 不定期抽查点名，计入最后成绩，一次扣5分。  
(请假需要导师签字的假条)
  - 加强作业，希望大家都有收获，设置奖励分数。

成绩：作业、课堂表现+期末考试

成绩组成：

作业 70 (20+20+30) + 期末 20 + 课堂表现10.

作业：如何通过群智的方式开发一个群智软件开发环境。

从需求开始，到设计。分三步：

# 群智软件开发环境

1) 第一步，调研。“群智软件开发环境”。我给大家一篇综述文章。之后，你想要的群智软件是什么样的--用户体验的角度。分组完成，

提交：1. 综述，还是选一个主题？还是精读2-5篇文章，形成报告；报告含三个部分：综述/读论文总结、想要的群智软件是什么样的、你们组是如何采用群智的。互评，写批注，打分。老师分配一人2份其他组的。

2) 看互评，看其他组的报告。形成需求--开发者角度。感受自己的需求。组内同样努力采用群智，形成的报告：需求（给出优先级排序）、自己组是如何展开群智的。

互评，老师分配一人2份其他组的。写批注，打分。

# 群智软件开发环境

3) 软件体系结构设计：看互评，看其他组的报告。形成自己组的需求，给出需求优先级。分组，各组考虑如何群智。形成设计方案。

每个组的三个文档：

- 1) 你想要的群智软件是什么样的；
- 2) 软件开发需求，对设计师有意义的；
- 3) 设计方案。

----这三个文档可以一直更新，提交。

我们保留所有的版本，给成绩；看反思情况。

- 如果选课人数太多的话，可能还需要一些题目：
  - 读论文：给定主题
  - 分析现有的体系结构
  - 分析系统

- 给分规则：
  - 分组；
  - 每次作业，群智的方式，每个同学的贡献，你们是如何考虑贡献计算的。
    - 依据：难度、规模、质量综合权衡。
  - 上课的活跃度：
    - Ppt的讲解（清楚、易懂、重点突出、回答问题有分析）
    - 上课提问（包括听同学ppt时的提问）
  - 作业互评
    - 写批注：认真程度、理解能力、理由表述
    - 打分：合理性。和批注的一致性。
  - 反思和改进；成果被采用的情况。

		乙	
		1	0
甲	1	B-	A
	0	C	B+
		B-	C
		A	B+

# 囚徒困境

		乙	
		1坦白	0抵赖
甲		1坦白	5年
1坦白	5年	5年	10年
0抵赖	10年	释放	1年
		释放	1年

乙

1坦白

0抵赖

甲  
1坦白  
0抵赖

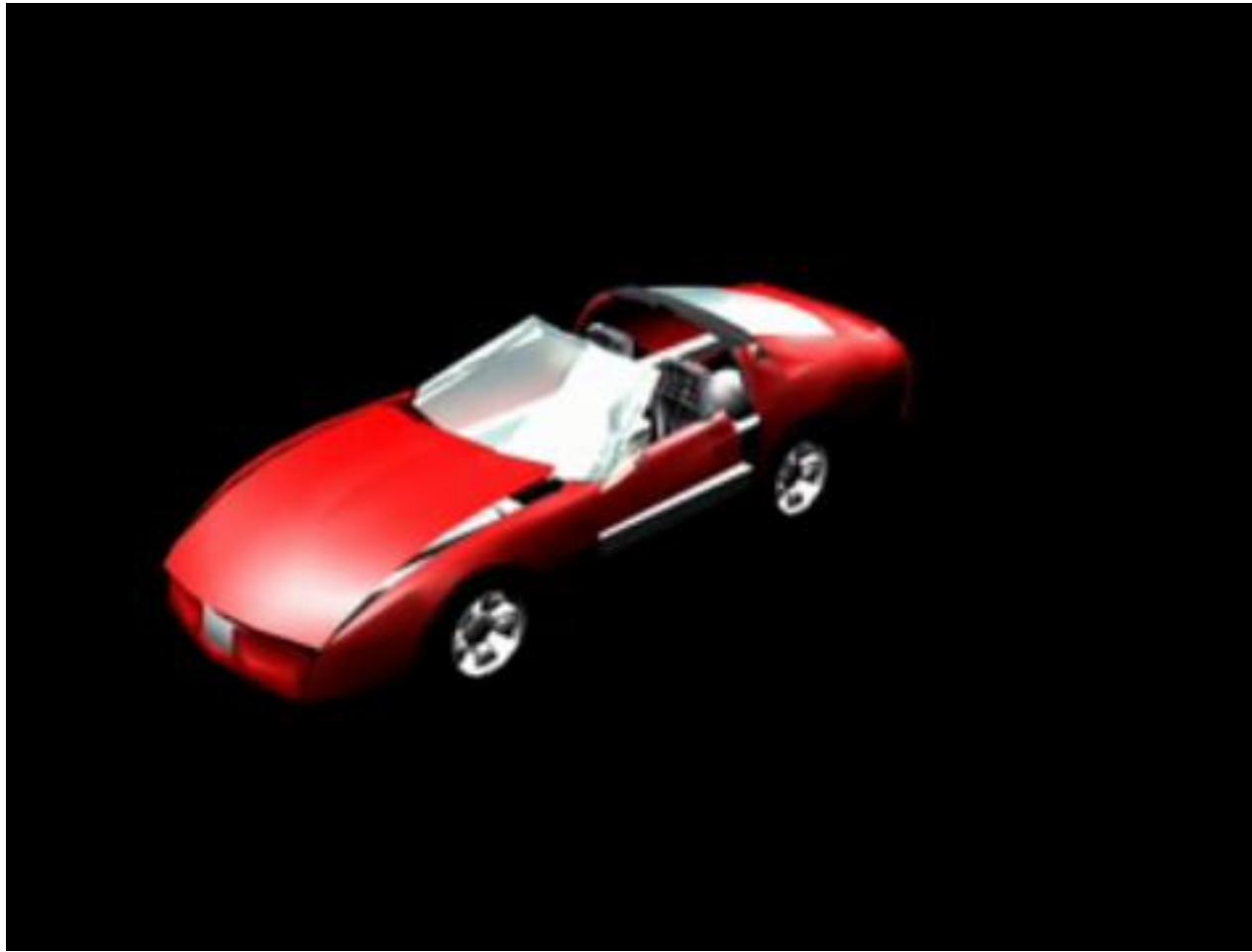
	8年	释放
1坦白	8年	10年
0抵赖	10年	1年
	释放	1年

# 第一部分 软件体系结构概述



- 什么是软件体系结构（粗略的）
- 为什么研究软件体系结构？
- 对软件体系结构的进一步认识
- 软件体系结构的重要性
- 什么是软件体系结构
- 软件体系结构的研究领域
- 一个实例

# 什么是软件体系结构？



# 什么是软件体系结构？

- Dewayne Perry和Alexander Wolf于1992年正式提出软件体系结构的概念：软件体系结构是具有一定形式的结构化元素。
  - 结构化元素包括：进程元素、数据元素和连接元素。
- Mary Shaw和David Garlan于1993年提出：
  - 软件体系结构是软件设计过程中的一个层次，这一层次超越计算过程中的算法设计和数据结构设计。
  - 设计和说明总体系统结构作为一个新问题正式提出来了。
  - 结构方面的因素包括总体组织和全局控制结构，通讯协议，同步，数据访问，功能分配，物理分布，设计元素的组合，性能优化以及各设计方案之间的取舍。

- Shaw 1995, 在第一届软件体系结构国际讨论会上
  - Shaw 将软件体系结构分为:
    1. 在结构模型中, 软件体系结构是由一些组件, 组件之间的联系以及其它方面组成。
      - 配置(configuration), 风格(style)
      - 约束(constraints), 语义(semantics)
      - 分析(analyses), 属性(properties)
      - 基本原理(rationale), 需求(requirements), stakeholder's need
    2. 框架模型: 侧重面向领域的体系结构的设计和分析方法
    3. 动态模型: 强调系统的行为质量
    4. 过程模型: 关注系统结构的构建及其步骤

# 什么是软件体系结构？（续）

- **Dewayne Perry**和**David Galan**于1995年IEEE软件工程学报上定义：
  - ◆ 软件体系结构是一个程序/系统各部件(components)的结构、它们之间的相互关系、进行设计的原则和随时间演进的指导方针。
- 南加州大学软件工程中心的Barry Boehm等**1995**指出：
  - 一个软件体系结构包括：(1)一个软件和系统部件、各种连接器以及约束(constraints)的集合；(2)一个系统需求说明的集合(风险承担者(stakeholder)的需求陈述)；(3)一个基本原理用以说明这一部件，连接器以及约束定义的系统可以满足系统stakeholders的需求陈述。

# 什么是软件体系结构？（续）

- 1997年，**Bass, Clements**和Kazman在《实用软件体系结构》一书中的定义：一个程序或计算机系统的软件体系结构包括一个或一组软件部件、软件部件的外部可见特性及其相互关系。
  - 多个结构（structures）
  - “外部可见特性”是指软件部件提供的服务、性能、特性、错误处理、共享资源使用等。
  - 这一定义强调软件体系结构必须从系统中抽象出某些信息。

- Bass, Celments, and Kazman *Software Architecture in Practice, Addison-Wesley 2003*

“一个程序或计算系统的软件体系结构是指该系统的各种**结构**，它包含软件**组件**，这些组件的**外部**的可见属性，以及它们之间的**关系**。”

- ANSI/IEEE标准1471-2000对大规模软件系统体系结构的描述建议

“软件体系结构被定义为系统的基本组织结构包括构件、构件之间的关系、环境以及管理系统设计和演化的原则。”

- Booch & Rumbaugh & Jacobson 定义
  - 一个软件系统的体系结构要对以下一系列重要问题作出决定：
    - 确定软件系统的组织方案 -- Organization
    - 选择构成该系统的结构元素及其界面 --- Element Interface
    - 确定由这些元素的合作所获得的行为 ---- Behavior
    - 确定由这些结构与行为元素逐步至规模更大的系统的组合方案  
----- Structure
    - 指导组织这些元素及其界面以及它们之间的合作与组合的体系结构风格。  
----- Style

# 什么是软件体系结构？（续）

对软件体系结构定义的总结分析：

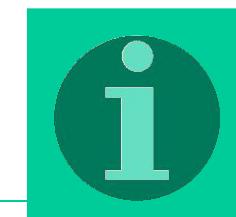
- 软件体系结构定义了软件部件(Component)，包括部件间交互的定义，特别强调省略和部件相互关系无关的内容信息(content information)。
- 软件体系结构并不说明部件的具体实现是什么、部件相互关系的具体体现是什么。
- 每一个软件系统都有自身的体系结构，即由软件部件及其相互关系组成。
- 软件体系结构中每一部件的行为是体系结构的一部分，反映部件间如何进行交互。
- 软件体系结构的基本元素是部件，部件的描述信息包括：
  - (1) **计算功能**：部件所实现的整体功能；
  - (2) **额外功能特性**：描述部件的执行效率、处理能力、环境假设和整体特性；
  - (3) **结构特性**：描述部件如何与其他部件集成在一起，以构成系统信息；
  - (4) **家族特性**：描述了相同或相关部件之间的关系。

# 什么是软件体系结构？（续）

表1：普通部件及其支持的相互作用

部件类型	部件支持的相互作用类型
模块(Module)	过程调用、数据共享
对象(Object)	方法调用
过滤器(Filter)	数据流
过程(Process)	消息传递、远过程调用、通讯协议、同步
数据文件(Data file)	读/写
数据库(Database)	模式(Schema)、查询语言
文档(Document)	共享表示假设

- 软件体系结构
- 软件体系结构是具有一定形式的结构化元素，即构件的集合，包括处理构件、数据构件和连接构件。处理构件负责对数据进行加工，数据构件是被加工的信息，连接构件把体系结构的不同部分组组合连接起来。这一定义注重区分处理构件、数据构件和连接构件，这一方法在其他的定义和方法中基本上得到保持。
- 一、软件体系结构的定义
- 虽然软件体系结构已经在软件工程领域中有着广泛的应用，但迄今为止还没有一个被大家所公认的定义。许多专家学者从不同角度和不同侧面对软件体系结构进行了刻画，较为典型的定义有：
  - . . . . .
  - . . . . .



## Software architecture

**From Wikipedia, the free encyclopedia**

Jump to: [navigation](#), [search](#)

The **software architecture** of a program or computing system is the structure or structures of the [system](#), which comprise software components, the externally visible properties of those components, and the relationships between them. The term also refers to documentation of a system's software architecture. Documenting software architecture facilitates communication between [stakeholders](#), documents early decisions about high-level design, and allows reuse of design components and patterns between projects.<sup>[1]</sup>

# 第一部分 软件体系结构概述

- 什么是软件体系结构（粗略的）
-  – 为什么研究软件体系结构？
- 对软件体系结构的进一步认识
- 软件体系结构的重要性
- 什么是软件体系结构
- 软件体系结构的研究领域
- 一个实例

## 1、软件开发当前的问题



—规模；  
—质量；  
—进度；  
—新技术、软件资产、  
需求.....  
怎么解决？

## 1. 计算机应用的深入及影响

### a. 软件组成的复杂化

(新开发的代码，集成软件资产，源代码和二进制文件，静态库，动态库，不同语种、不同平台的集成)

### b. 程序规模变化 (1000行 -3000万行的程序代码)

### c. 平台的多样化

(多种操作系统、应用程序的混合、多厂商联合开发)

- d. 物理的分布（由集中到分布，由此带来技术上的考虑）
- e. 协议（各种通讯协议的混合）
- f. 功能分配（出现了多层的应用程序模式）
- g. 产品需求的更改（需求的变化，市场变化）
- h. 算法和数据结构（退居次要地位，特别是硬件性能的提高）

如何有效的考虑软件组织，而且是从高层进行抽象？适应将来的负载？

## 2. 软件市场的新趋势

——零时间（开发周期短）

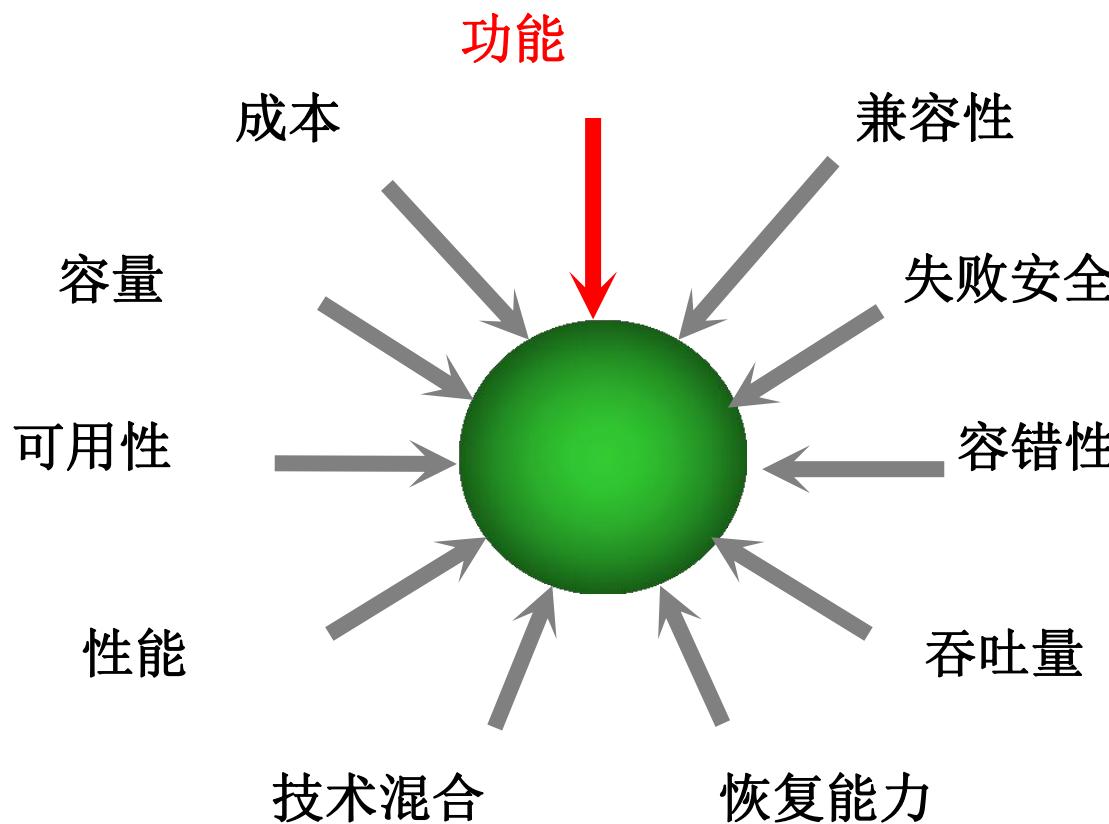
——零bug（软件质量）

——开发方法的革新

(硬件、机械和汽车产业的启示，

基于构件的软件开发要求考虑软件的体系结构)

- 软件系统的开发已变得越来越复杂



- The challenge over the next 20 years will not be speed or cost or performance; it will be a question of complexity.

Bill Raduchel, Chief Strategy Officer, Sun Microsystems

- Our enemy is complexity, and it's our goal to kill it.

Jan Baan

- IBM360系统之父 F. Brooks: 复杂性是软件的固有属性
- 软件复杂性不能通过任何方法彻底消除，只能采取某些手段加以控制

- ☞ 技术创新
- ☞ 软件开发过程改进
- ☞ 可视化建模和模型驱动的开发
- ☞ 提高软件设计的抽象层次

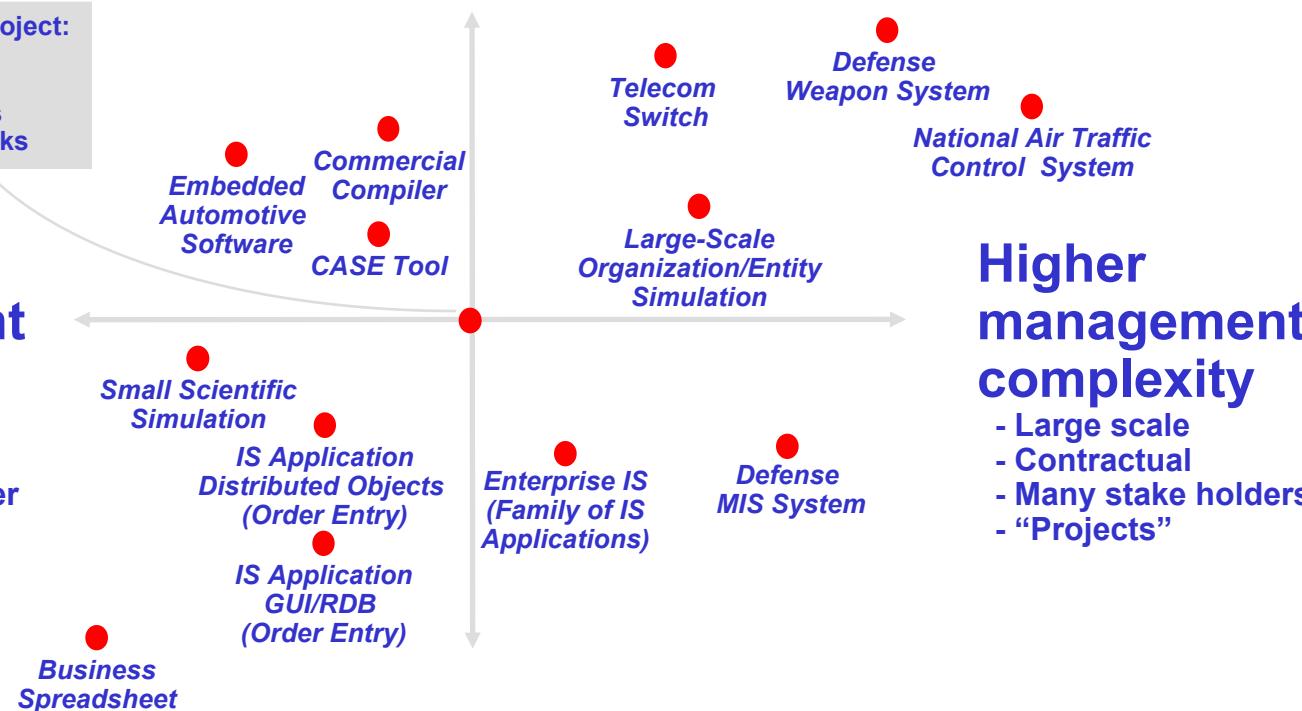
## Higher technical complexity

- Embedded, real-time, distributed, fault-tolerant
- Custom, unprecedented, architecture reengineering
- High performance

An average software project:  
- 5-10 people  
- 10-15 month duration  
- 3-5 external interfaces  
- Some unknowns & risks

## Lower management complexity

- Small scale
- Informal
- Single stakeholder
- “Products”



## Lower technical complexity

- Mostly 4GL, or component-based
- Application reengineering
- Interactive performance

- 美国空中交通管制系统ATC(Air Traffic Control)
  - 需求:
    - 美国的航空业发达，商用飞机、私人飞机和军用飞机量远远高于其他国家；
    - 系统要求很高的实时性、安全性和分布性
  - 对系统软件体系结构的要求
    - 极高的可用行和高性能
    - 开发性、可更改性、不断提高自系统的能力和互操作性
  - 系统
    - 架构在广域网上，综合了20世纪90年代计算机领域的各种技术
    - 系统规模

- 系统规模

- 总长度：Ada程序 > 100万行；
- 系统分布在数百台计算机上，并嵌入了各种新的、复杂的硬件设备。
- 所有这些都必须对不可预见的实时事件随时作出响应。

- 例子：其核心部分——初始阶段系统ISSS

- 要求每个中途中心站能够支持最多20台控制台，控制台使用处理器的CPU是IBM RS/6000；
- 要求每个中途中心站能够管理400到2440架飞机，每个场站需16-40台雷达，一个中途中心需要60—90个控制位置。
- 该系统花费**10多亿美元**开发，运行**5年后**再次请专家对它进行评审，以便决定是改进它还是抛弃它。

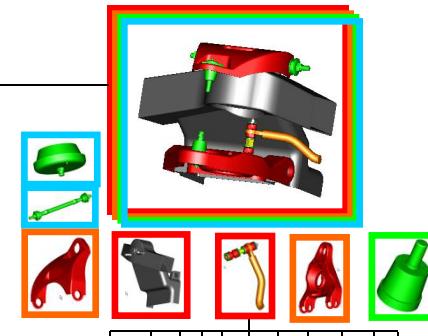
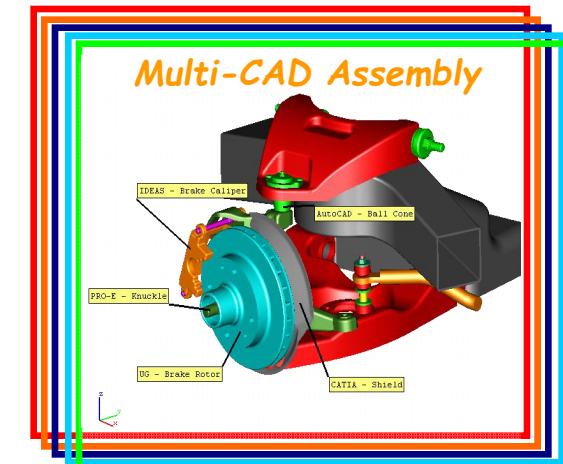
- 软件系统的复杂性来自多个方面 (Zuse, 1991)
  - 软件所描述问题域的复杂性
  - 软件系统本身多状态、多元素及其交互的复杂性
  - 人对现实事物的认知与其被认知的方式根本不同所造成的翻译过程的复杂性
  - 人在构件软件过程中的认知的复杂性
  - 软件开发过程中用户、开发者等涉众之间交互的复杂性
  - 软件项目组织管理的复杂性
  - . . . . .

## • 我们面临的困境

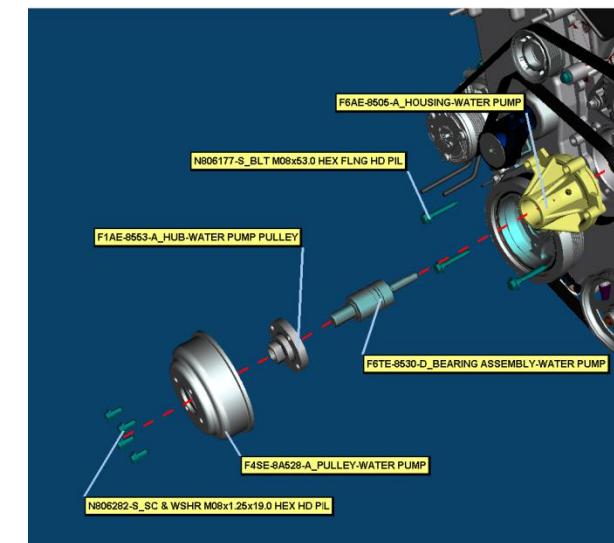
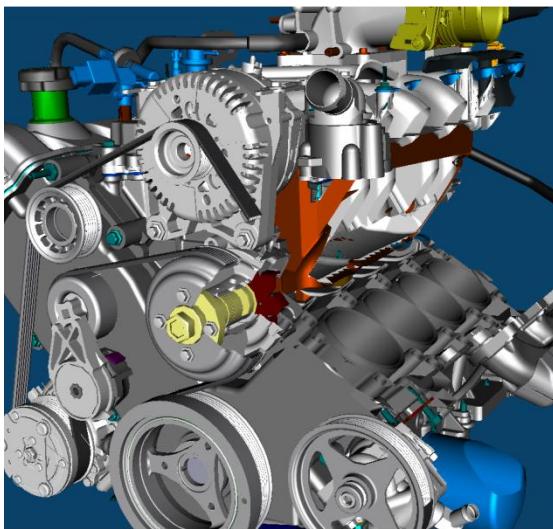
- 软件的抽象性和不可见性
- 软件开发过程的不可见性



软件：  
光盘 + 一堆文档？



- 又如：建筑、房屋装修



- 带来的问题：
  - 需求中供需双方的交流问题
  - 方案设计和评审（难以交流、用户参与困难）
  - 开发进度的控制、问题的及时发现
  - 管理无法预见
- 尤其是随着软件规模的增大，  
不再是：算法 + 数据结构 时，传统的流程图不再够用。



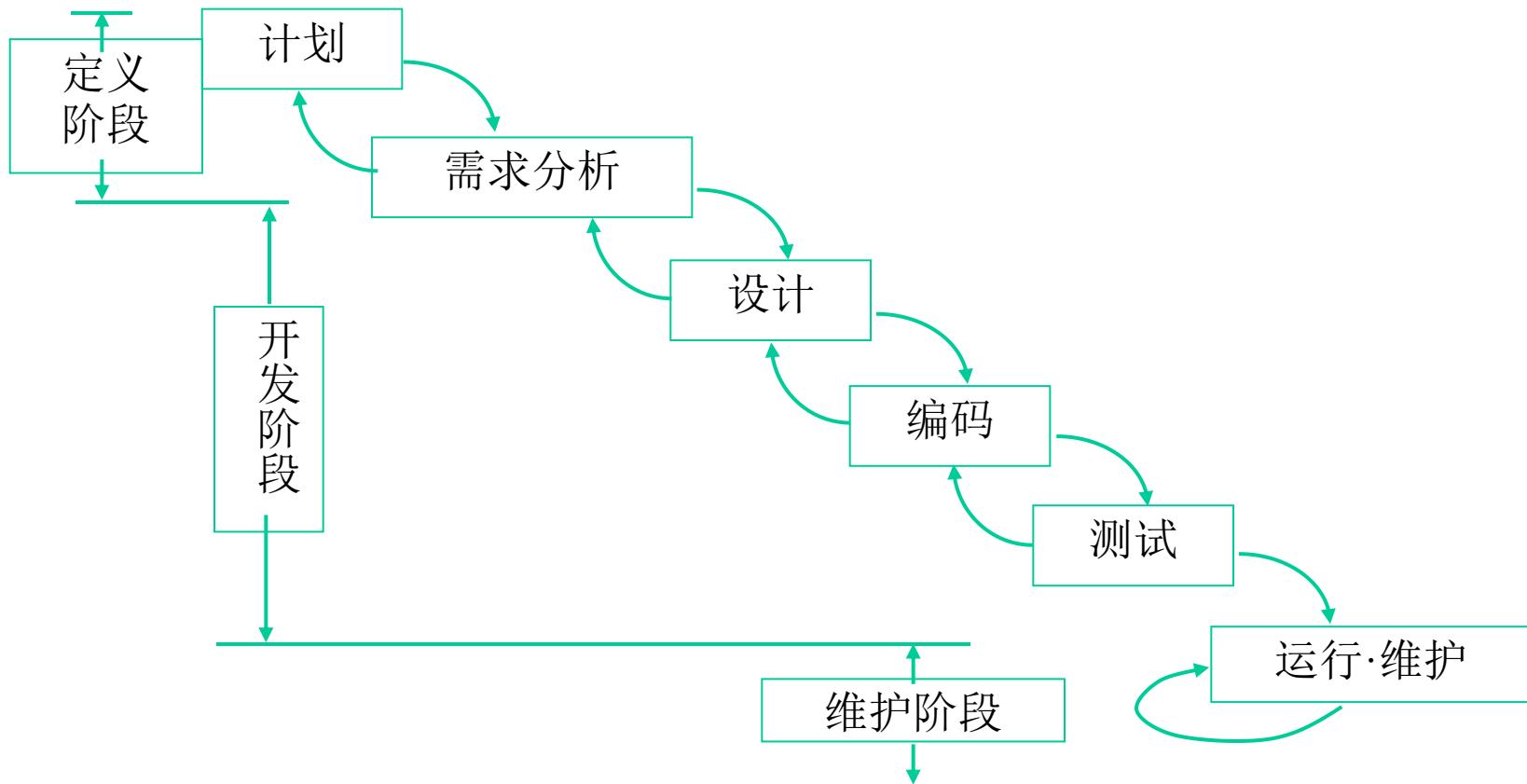
- 需求分析
  - 软件项目中**40%~60%**的问题都是在软件需求分析阶段埋下的“祸根” .(Leffingwell 1997)
- 软件测试、维护
  - 软件开发费用的**40%~60%**，还是**80%**

- 改变思想，更新观点，寻求全生命周期的综合集成、整体优化。
  - 软件开发的目标：
    - 质量
    - 成本
    - 时间

# 质量保证

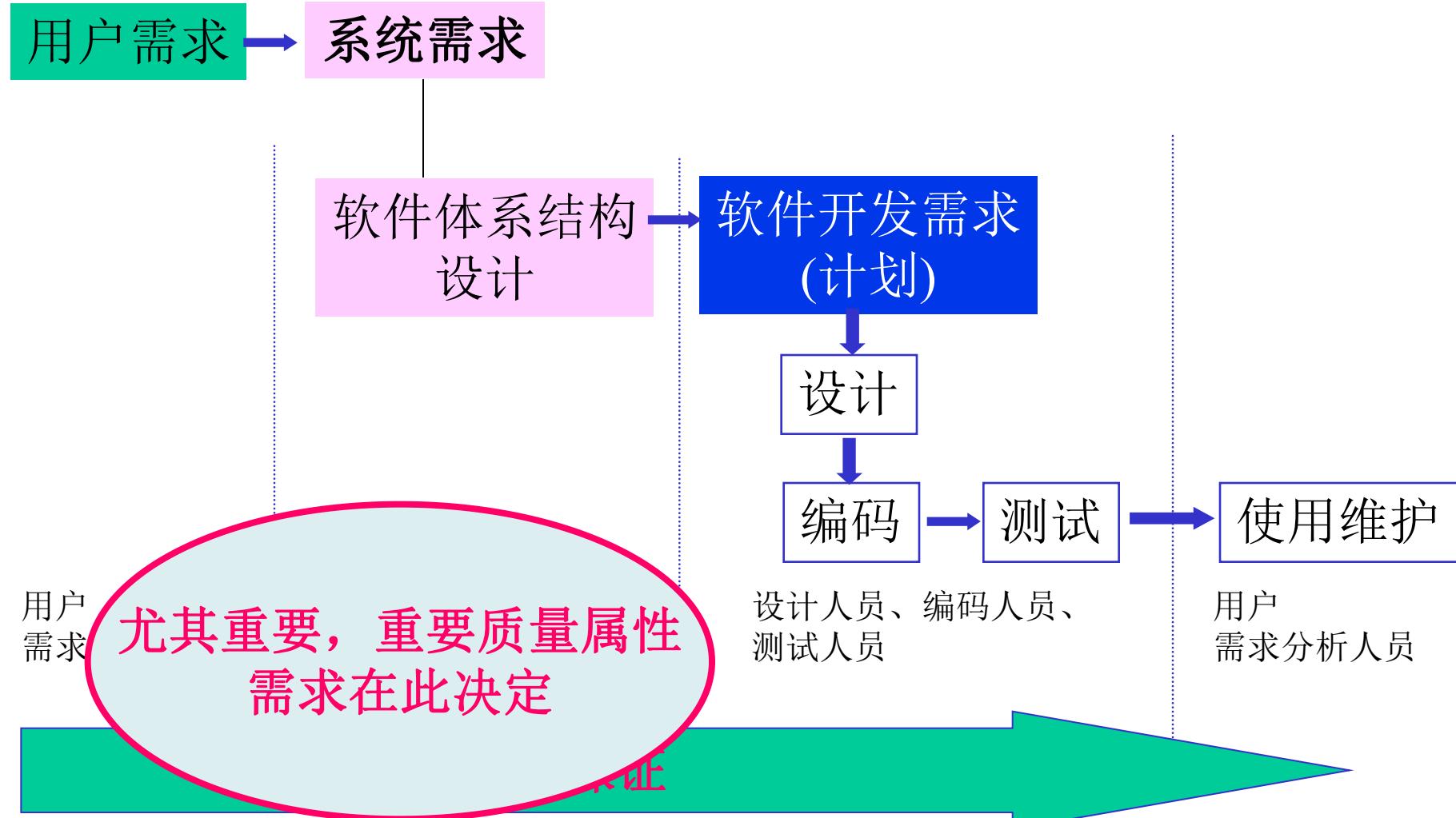


- 我们能做什么？
- 需求是谁写的？质量谁定义？
- 一个故事。
  - 隐含的需求，软件质量的定义。



## 区分两个概念

- 用户需求、系统需求和开发需求



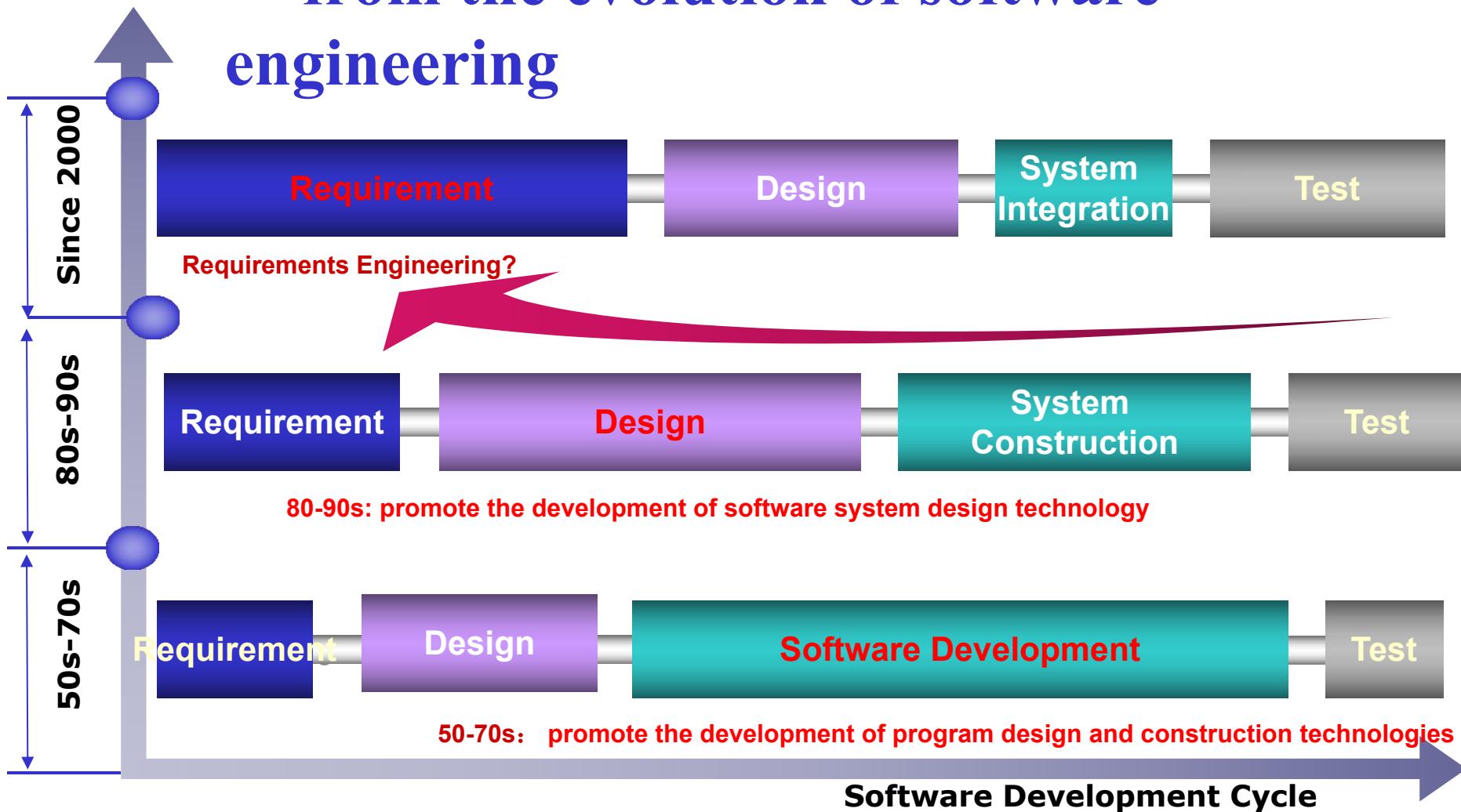
- 成本
  - 软件项目中40%~60%的问题都是在软件需求分析阶段埋下的“祸根”.(Leffingwell 1997)
  - 软件测试、维护
    - 软件开发费用的40%~60%，还是80%
  - 为什么?
    - 一直以来重编码，轻设计。重体力劳动，轻脑力劳动。
    - 早期埋下祸根。缺乏整体规划
  - 怎么办?
    - 考虑全生命周期的费用，增加前期投入，降低整体成本。

## • 时间——进度

- 磨刀不误砍柴功。
- 抓设计、重评审，可以提高进度和质量，大大缩短系统开发和调试时间。**早期评审正在成为保证软件质量的重要技术手段。**
- **重用：**设计重用和代码重用—基于构件的软件开发：体系结构和构件是核心
- 体系结构是并行开发和外包的基础。
- **增量式开发：****总体规划、分步实施。**有利于占领市场、也有利于用户反馈，降低风险。成功的基础是良好的体系结构。

- 20世纪60年代末—70年代中期：
  - 高级语言应用，结构化程序设计技术，支持工具
- 20世纪70年代中期—80年代：
  - 计算机辅助软件工程（CASE），软件工程环境。
- 20世纪80年代中期—90年代：
  - 出现面向对象语言和方法，并成为主流的软件开发技术
  - 开展软件过程及软件过程改善的研究
  - 软件构件技术、软件体系结构技术

# More important, more time and higher cost ---from the evolution of software engineering



## • Software Architecture

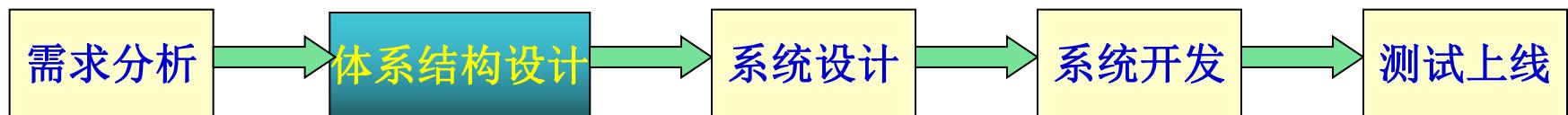
- 1992和1993年，以【1】【2】两篇文章的发表为主要标志，软件体系结构作为软件工程中一个新兴的研究领域逐渐引起了人们的广泛关注。
- 【1】 Dewayne E. Perry and Alexander L. Wolf. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 17(4):40–52, October 1992.
- 【2】 David Garlan and Mary Shaw. An introduction to software architecture. In Advances in Software Engineering and Knowledge Engineering, volume 1. World Scientific Publishing Co., 1993.

- 软件体系结构是对软件设计层次的再一次提升

	设计	软件体系结构设计
设计重点	算法 + 数据结构	系统整体组织结构
主要概念	过程、函数、过程调用	构件、连接件、端口、角色、配置
关注特性	单个构件的接口和行为特性	系统全局质量属性

- 软件体系结构在软件开发中的作用越来越受到开发组织重视
- 以软件体系结构为中心的开发方法逐渐被广泛采纳

- 随着计算机应用的逐步广泛和深入，随着信息技术的突飞猛进的发展，各行各业对于大规模、密集型软件应用系统的需求越来越普遍，对软件开发的周期要求越来越短，对于软件质量保证的要求越来越高，基于软件体系结构的软件开发方法是满足这些苛刻要求的必经途径。



- 处于软件系统建设的上游
- 需要全面考虑多方面的因素
- 对于同一个问题，可以有多种设计结果
- 是在各种制约条件下取得的较好折衷方案
- 科学+ 经验+ 艺术
- “系统架构”往往被滥用

- **软件架构—巨大的知识海洋**

- 门槛相对较高、职业生涯非常长

- 相对独立于技术的新陈代谢

- 适合于喜欢学习的人

- **不断学习、增加积累、注重经验**

- 注意学习方法论、框架

- 不断增加各种系统架构的知识

- 经验积累非常重要

- **在与高手和同行合作中提高水平**

- 与高手的合作是最佳途径

- 同行之间的交流也非常有效

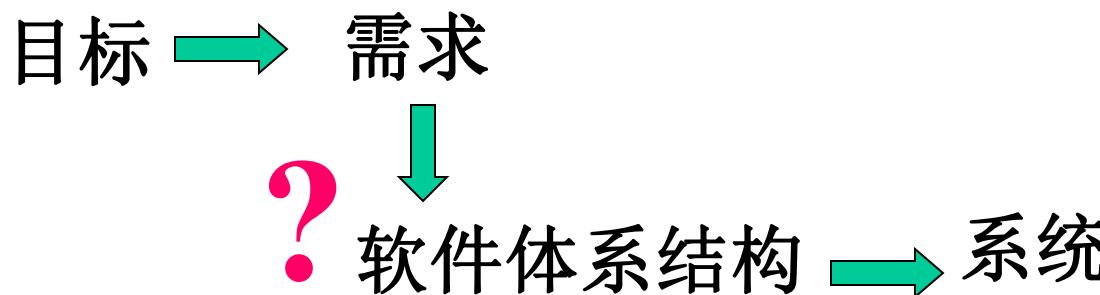
- 在每一个项目中进行创新

# 第一部分 软件体系结构概述



- 什么是软件体系结构（粗略的）
- 为什么研究软件体系结构？
- 对软件体系结构的进一步认识
- 软件体系结构的重要性
- 什么是软件体系结构
- 软件体系结构的研究领域
- 一个实例

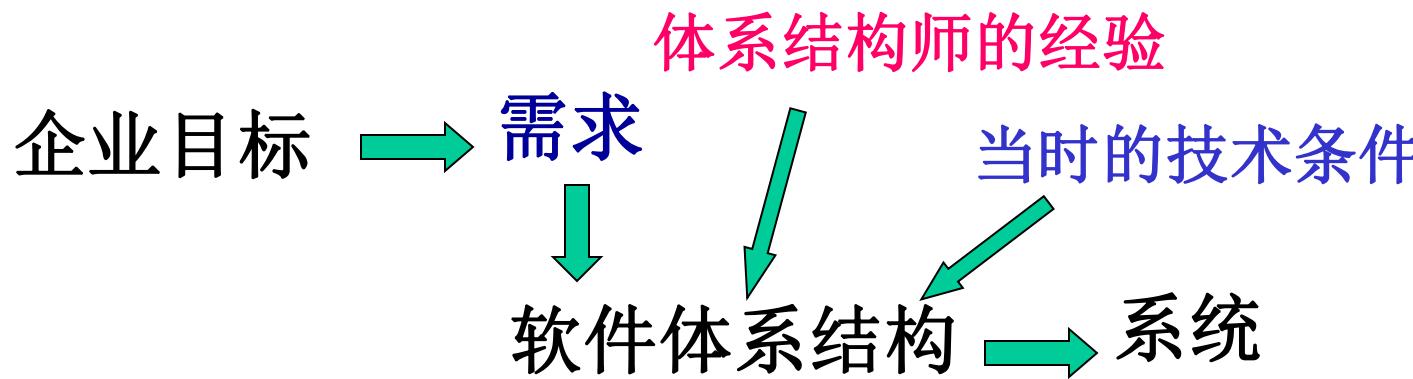
- 影响软件体系结构的因素：



一个例子.

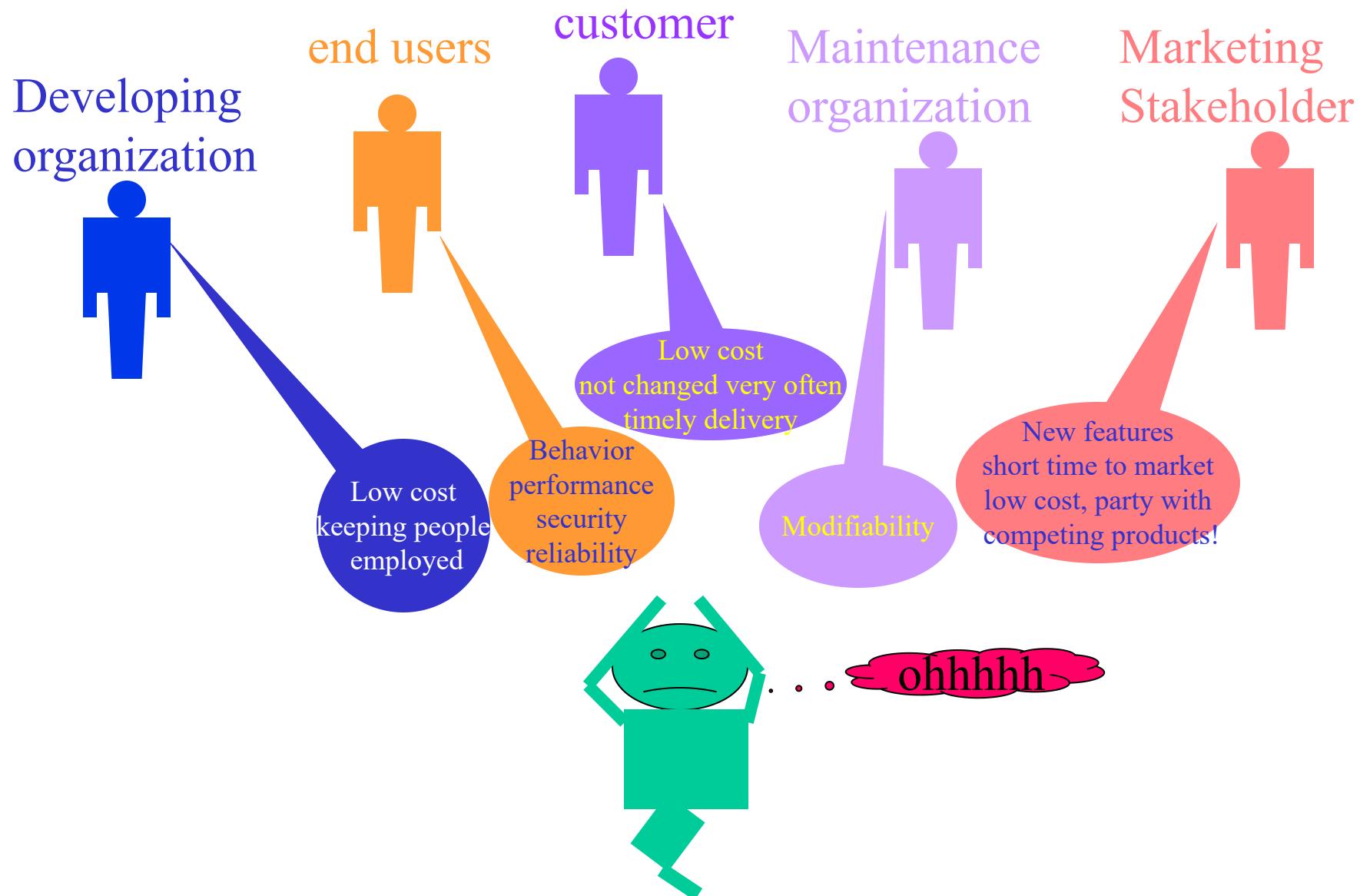
- Goal: in the 1620s, Sweden and Poland were at war. The king of Sweden determined to put a swift end to it, so commissioned a new warship, the like of which had never been seen before: 70 meters long, able to carry 300 soldiers, and with an astonishing 64 heavy guns mounted on two gun decks.
- Architect Henrik Hybertsson: with an impeccable reputation
  - He had to balance many concerns:
    - Swift time to development was critical
    - then performance, functionality, safety, reliability, and cost.
  - He was also responsible to a variety of stakeholders
    - The real customer: the king
    - also responsible to the crew that would sail his creation
  - His experience: a single-gun-deck ship, --> technical environment of the day
- Result: August 10, 1628, first sail, never back. --- “bad proportioned”

- 从这个古老的故事我们得到什么启发?
  - Enterprise goals beget requirements, which beget an architecture, which begets a system.
  - The architecture flows from the architect's experience and the technical environment of the day.

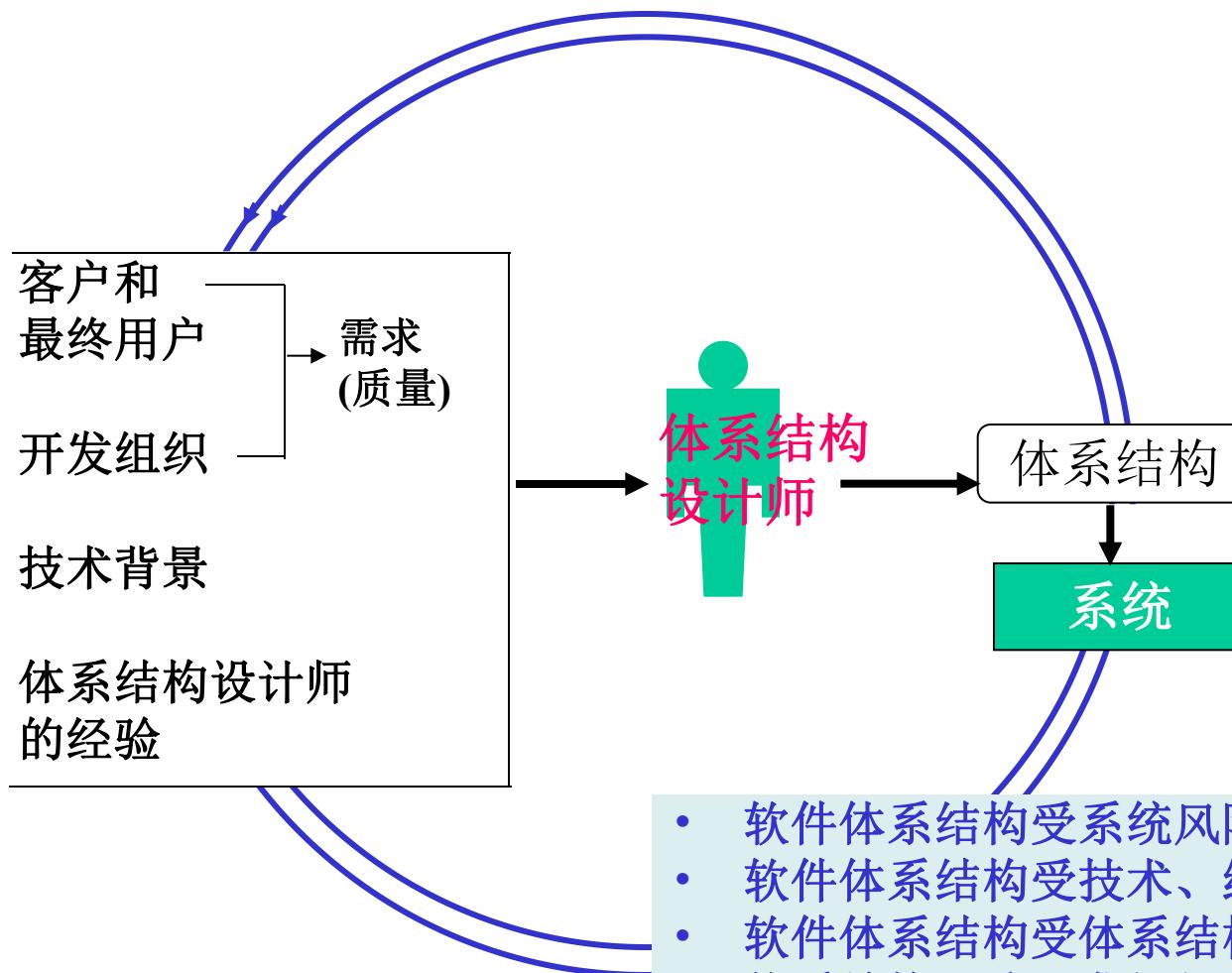


- 合理切实可行的系统结构是保证系统成功运行的首要因素。
  - Vasa战舰设计精良，但是船体比例严重失调，体系结构存在致命的缺陷
- 设计师在无前人设计、建造过配备如此多武器装备舰船的情况下，既要执行国王的命令，又要对未来的船员负责，还有考虑舰艇的性能、功能、安全性、可靠性、造价等多项指标。（见下页）
  - 勉强答应不可能的需求
- 吸取的教训：在构建任何一个应用系统之前，应对所用体系结构进行评价，以减少开发系统的风险。
- 建立软件体系结构的业务周期ABC。

# Architectures are influenced by system stakeholders



# Architecture Business Cycle



- 软件体系结构受系统风险承担者的影响
- 软件体系结构受技术、组织方面因素的影响
- 软件体系结构受体系结构设计师的影响
- 体系结构影响开发组织的结构
- 体系结构影响开发组织的企业目标
- 体系结构影响后续系统的客户需求
- 该系统的创建过程影响体系结构设计师的经验.
- 少量系统将影响，并实际改变软件工程文化

- Technical: 1986– center DB
- Len Bass : 1986– center DB
- Tim Berners-Lee, a researcher with CERN, the European Laboratory for Particle Physics
  - 1989 “Information management: A Proposal”
  - 1990 World Wide Web
    - to promote interaction among CERN researchers within the constraints of a heterogeneous computing environment.

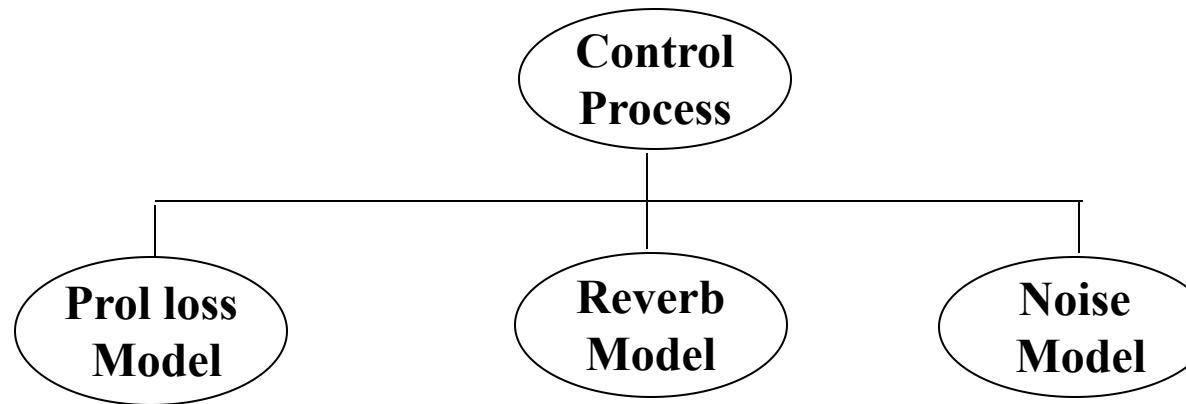
# • 什么是软件体系结构

- 软件体系结构的定义、作用
- 软件体系结构的组成

- 软件体系结构是什么，不是什么
  - 软件体系结构是高层设计。
    - True enough, in the sense that a horse is a mammal.
  - 软件体系结构是系统的总体结构
  - Architecture is the structure of the components of a program or system, their interrelationships, and principles and guidelines governing their design and evolution over time.
  - 软件体系结构是构件和连接件（components and connectors）
  - 软件体系结构是构件、连接件和约束（constraints）。

# Why that is not enough

A system description for an underwater acoustic simulation



- **What we can tell from this diagram?**
  - The system consists of four components
  - Three of the components might have more in common with each other because they are positioned next to each other
  - All of the components apparently have some sort of relationship with each other, since the diagram is fully connected.
- **Is this an architecture? If it is, what can we not tell from the diagram?**

- **what can we not tell from the diagram?**
  - What is the nature of the components?
    - What is the significance of the separation? Do the components run on separate processors? Do they run at separate time? Do the components consist of processes, programs, or both? Do the components represent ways in which the project labor will be divided, or do they convey a sense of runtime separation? Are they modules, objects, tasks, functions, processes, distributed programs, or something else?
  - What is the significance of the links?
    - Do the links means that the components communicate with each other, control each other, sent data to each other, use each other, invoke each other, synchronize with each other, or some combination of these or other relations? What are the mechanisms for the communication?
  - What is the significance of the layout?
    - Why is CP on a separate level? Dose it call the other three components, and are the others not allowed to call it? Or was there simply not room enough to put all four components on the same row in the diagram?
  - How does the architecture operate at runtime?
    - How do data and control flow through the system?

# 第一部分 软件体系结构概述



- 什么是软件体系结构（粗略的）
- 为什么研究软件体系结构？
- 对软件体系结构的进一步认识
- **软件体系结构的重要性**
- 什么是软件体系结构
- 软件体系结构的研究领域
- 一个实例

## 1. 在风险承担和早期设计中的作用

- 交流作用
- 早期决策
- 实现

## 2. 在软件开发各阶段的作用

## 3. 是系统分析和设计的高层复用

## 在风险承担和早期设计中的作用

1. 软件体系结构是系统开发中不同参与者进行交流和信息传播的媒介。  
——— 软件体系结构建模技术
2. 软件体系结构代表了早期的设计决策成果。早期的决策最难处理、最难于改变、影响范围也最大。  
——— 软件体系结构评审技术等
3. 软件体系结构可以作为一种可变换的模型。  
—— MDA

- 1、软件体系结构对风险承担者进行交流的手段
  - 客户：（一般）软件系统投资方。关心：方案不超预算、按期完成。
  - 用户：软件系统的使用者。关心是否可用、是否可靠。
  - 项目经理：负责软件系统开发的最高领导。关心提出体系结构是否规范、可控并使团队中的各小组独立、协调地实现目标。
  - 程序员：具体编程人员。关心：体系结构是否可实现；
  - 测试人员：测试系统的功能和性能是否达到设计目标。关心：系统的易测试性。
  - 维护人员：系统的维护、升级、移植等。关心系统的可修改性。
  - . . . . .

# 软件体系结构的作用(重要性)

体系结构是大家表述自己观点，协商寻找一个合理方案的研究对象。



## • 2 软件体系结构是早期设计决策的产品—决策点

- “软件体系结构体现了对系统作出的最早的设计决策。这些早期决策的正确性最难保证，而且这些决策也最难改变，它们的影响也最为深远。”
- 要保证早期决策正确是困难的，需要风险承担者的各方使用统一的概念、术语、对用户需求进行反复彻底的讨论和分析，对初步得到的体系结构进行合理性验证、对性能和功能进行评估，在进一步修改——反复迭代的螺旋式设计过程。
- 软件体系结构既规定了所开发应用系统的结构，也决定项目开发组的结构。
  - 对于大型系统，一般采用结构化分解方法，确定软件体系结构中构件的功能和组成、各构件间的关系、接口定义，以及整体配置的约束和必须遵循的规则。——对应于任务的分解

- 后期，对体系结构的修改是困难的。
  - 一旦对系统的构件、连接件和配置原则达成一致意见，要对其进行修改是困难的。
- 我们将对系统的修改分为三类：
  - 对本地构件的修改（局部修改）
  - 非本地构件的修改——非局部修改：可能会对较大范围的开发产生影响
  - 最坏的情况，是对体系结构的修改，如对构件间交互的基本方式进行修改、系统划分的修改，可能会引起系统的全面改动。
- 体系结构设计就应考虑体系结构变更的问题，考虑在运行时体系结构可以变更。

### • 3 体系结构对应用系统的影响

- 软件体系结构制约着应用系统的质量属性，对整个软件的生命周期影响深远。直接影响着系统设计、开发、维护、升级、演化等后续活动。
  - “在大型软件系统中，质量属性更多的是由系统结构和功能划分来实现的，而不再仅仅是选用的算法和数据结构。”
- 软件体系结构是一种可重用的模型
  - Product lines share a common architecture。软件产品线是产业化的重要思想。提高质量、降低成本、缩短上市时间。
  - Systems can be built by using large, externally developed components——使得基于构件的开发成为可能。
    - 由于COM/DCOM、CORBA、JavaBeans的广泛应用
    - 大量构件的面市
    - 特定领域软件体系结构的研制成功
  - 提供高层次和早期的重用
  - 体系结构培训。

1. 在风险承担和早期设计中的作用
2. 在软件开发各阶段的作用
  - ① 项目规划阶段
  - ② 需求分析阶段
  - ③ 项目设计阶段
  - ④ 项目实施阶段
  - ⑤ 测试与系统评估阶段
  - ⑥ 维护与升级阶段
3. 是系统分析和设计的高层复用

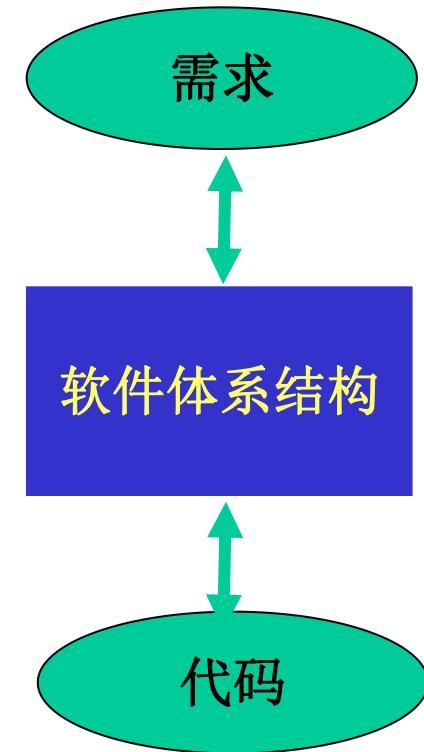
## 1. 在风险承担和早期设计中的作用

- 交流作用
- 早期决策
- 实现

## 2. 在软件开发各阶段的作用

## 3. 是系统分析和设计的高层复用

- 在需求和代码之间起到桥梁作用。理想情况，体系结构表示为整个系统提供了易驾驭的、知识性的指导。
- **易理解：**抽象，简化对大型系统的理解，考虑重要问题。
- **重用：**支持多种重用，构件、体系结构的重用。
- **构造：**定义构件和构件之间关系，为开发提供蓝图。
- **演化：**能够指出系统演化的方向
- **分析：**包括对系统连接性检查、体系结构风格约束的一致性检查、质量属性一致性检查、依赖性分析，对于构建在特殊体系结构风格上的体系结构领域的特定分析等。
- **管理：**经验证明，成功的项目规划往往将合理可行的体系结构看成是软件开发过程的关键里程碑。某个软件体系结构的关键性评估，将导致对需求、实施策略和潜在危机的清晰而深入的认识。
- **交流：**是风险承担者之间进行的手段。



- MDA—模型驱动的软件体系结构
- 由OMG定义的软件开发框架
- MDA开发生命周期
  - 平台独立模型 (Platform Independent Model, **PIM**) : 具有高抽象层次、独立于任何实现技术。
  - 平台相关模型 (Platform Specific Model, **PSM**) : 为某种特定实现技术定做的模型。一个PIM可以转换成多个PSM
  - 代码: 由PSM变换成代码。

# 第一部分 软件体系结构概述



- 什么是软件体系结构（粗略的）
- 为什么研究软件体系结构？
- 对软件体系结构的进一步认识
- 软件体系结构的重要性
- 什么是软件体系结构
- 软件体系结构的研究领域
- 一个实例

- Bass, Celments, and Kazman *Software Architecture in Practice*, Addison-Wesley 1997
  - 一个程序或计算系统的软件体系结构是指该系统的各种**结构**，它包含软件**组件**，这些组件的**外部**的可见属性，以及它们之间的**关系**。
- Uml process
  - 一个软件系统的体系结构要对以下一系列重要问题作出决定：
    - 确定软件系统的**组织**方案 -- Organization
    - 选择构成该系统的结构**元素**及其**界面** --- Element Interface
    - 确定由这些元素的**合作**所获得的**行为** ----- Behavior
    - 确定由这些**结构**与行为元素逐步至规模更大的系统的**组合**方案 ----- Structure Composition
    - 指导组织这些元素及其界面以及它们之间的合作与组合的体系结构**风格**. ----- Style

- Bass, Celments, and Kazman

一个程序或计算系统的软件体系结构是指该系统的一种或多种**结构**，它由软件**组件**，这些组件的**外部**的可见属性，以及它们之间的**关系**组成。

- 1) 体系结构定义了组成构件、构件之间的关系。忽略了与之无关的信息。
- 2) 明确指出系统能够并且经常包含一个以上的结构。
  - Static & Dynamic
- 3) 说明任何系统都有一个体系结构。
- 4) 每个构件的行为是体系结构的组成部分。构件作为可以被观察或者从其他构件的角度可以被认识的部分而存在。
- 5) 与系统的体系结构的是好是坏没有直接联系。体系结构可能满足/也可能不满足系统的行为、性能和生命周期的需求。

- 一个软件系统的体系结构要对以下一系列重要问题作出决定：
  - 选择构成该系统的**结构元素**及其**界面** -- Component/Interface
  - 确定软件系统的**组织**方案 -- Organization
  - 确定由这些元素的**合作**所获得的**行为** ----- Behavior
  - 确定由这些**结构**与行为元素逐步至规模更大的系统的**组合**方案 ----- Structure Composition/Integration
  - 指导组织这些元素及其界面以及它们之间的合作与组合的**体系结构风格**. ----- Style

- 系统架构师是一个最终确认和评估系统需求，给出开发规范，搭建系统实现的核心构架，并澄清技术细节、扫清主要难点的技术人员。主要着眼于系统的“技术实现”。因此他/她应该是特定的开发平台、语言、工具的大师，对常见应用场景能马上给出最恰当的解决方案，同时要对所属的开发团队有足够的了解，能够评估自己的团队实现特定的功能需求需要的代价。
- 系统架构师负责设计系统整体架构，从需求到设计的每个细节都要考虑到，把握整个项目，使设计的项目尽量效率高，开发容易，维护方便，升级简单等。

# Structures of Software Architecture

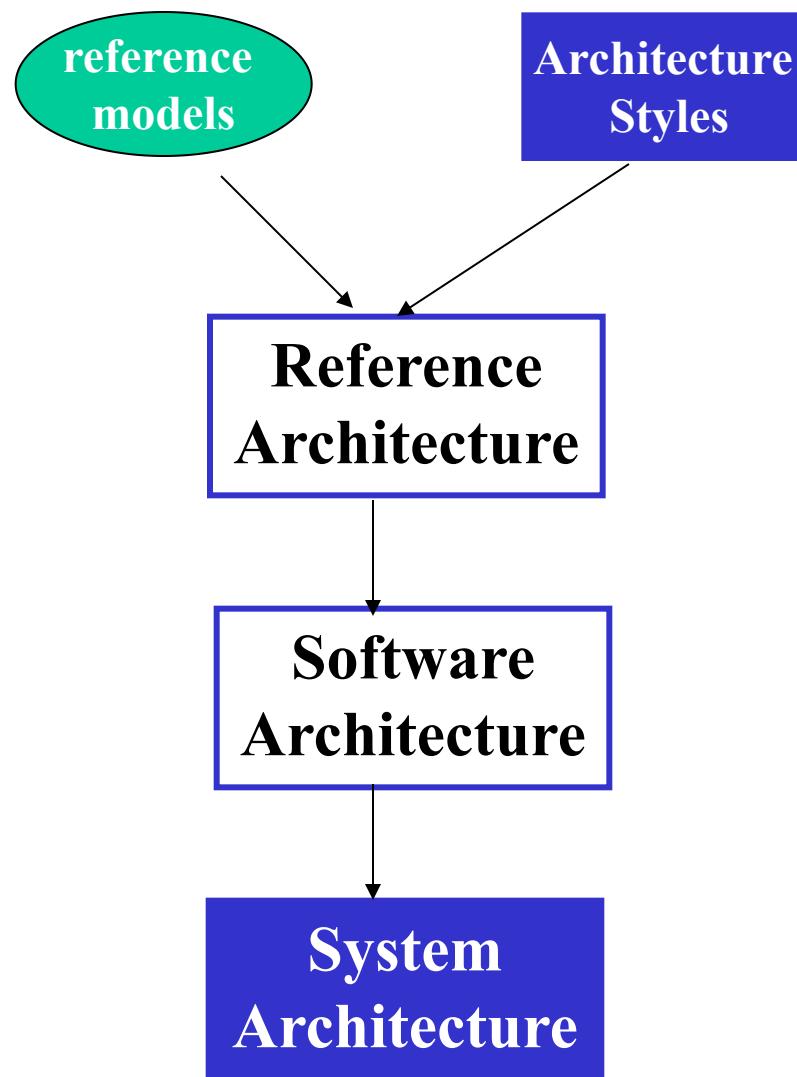
- **Module structure**
- **Conceptual, or logical, structure -----function reqs**
- **Process structure, or coordination structure**
- **Physical structure**
- **Uses structure ---for incremental build**
- **Calls structure**
- **Data flow**
- **Control flow**
- **Class structure**

软件结构	单元	连接所表示的关系	适于
模块	任务划分	父子模块关系; 共享秘密关系	资源配置、项目结构与规划；信息隐藏、封装；配置控制
概念结构	功能	共享数据关系	理解问题空间
进程结构	程序	并发执行关系；可并行 执行关系；排斥关系； 先后关系	调度分析；性能分析
物理结构	硬件	通信关系	性能、可用性和安全性分析
使用结构	程序	要求正确存在	实现基础；实现扩展
调用结构	程序	带参数调用	性能概貌；消除瓶颈
数据流结构	功能任务	可发送数据	功能需求的可跟踪性
控制流结构	系统状态或模式	按连接所标注的事件或条件变迁	带有定时信息，可作为自动模型模拟、时序和功能行为验证的基础
类结构	对象	实例关系； 共享方法关系	在按面向对象设计开发的系统中， 可通过使用模板快速实现。

# 区分几个概念

- **An architecture style is a description of component types and a pattern of their runtime control and /or data transfer.**
  - Example: client-server is an architecture style. Countless architectures are of the client-server style as we have defined, but these architectures are different from each other. An architecture style is not an architecture, but it still conveys a useful image of the system.
- **A reference model is a division of functionality together with data flow between the pieces.**
  - a standard decomposition of a known problem into parts that cooperatively solve the problem.
- **A reference architecture is a reference model mapped onto software components(that will cooperatively implement the functionality defined in the reference model) and the data flows between the components.**

# The relationship between them



- 技术架构

- 信息系统
  - 应用架构
  - 程序架构
  - 数据架构
  - 技术架构
  - 流程架构



## • 1、应用架构

- 应用架构指的是在信息系统里，所有的程序如何分组、分层地构建为整个信息系统，包括各层级间的定位与关系、层级内部各组件的定位与关系。

## • 2、程序架构

- 指的是在信息系统里，各个程序自身的内部结构。

## • 3、数据架构

- 数据架构指的是在信息系统里，所有数据从产生、处理、传递到应用分析等如何组织与分布，以及它们之间的关系，包括各种各样的数据（数据资源）与各种各样的程序（逻辑资源）之间的关系。

## • 4、技术架构

- 一个信息系统，自底向上，通常由物理设施（服务器、存储器、网络等）、系统软件（操作系统、数据库）、中间件（各种软件平台）、应用系统组成。技术架构指的是在信息系统里，这些组成部分的定位与相互关系。

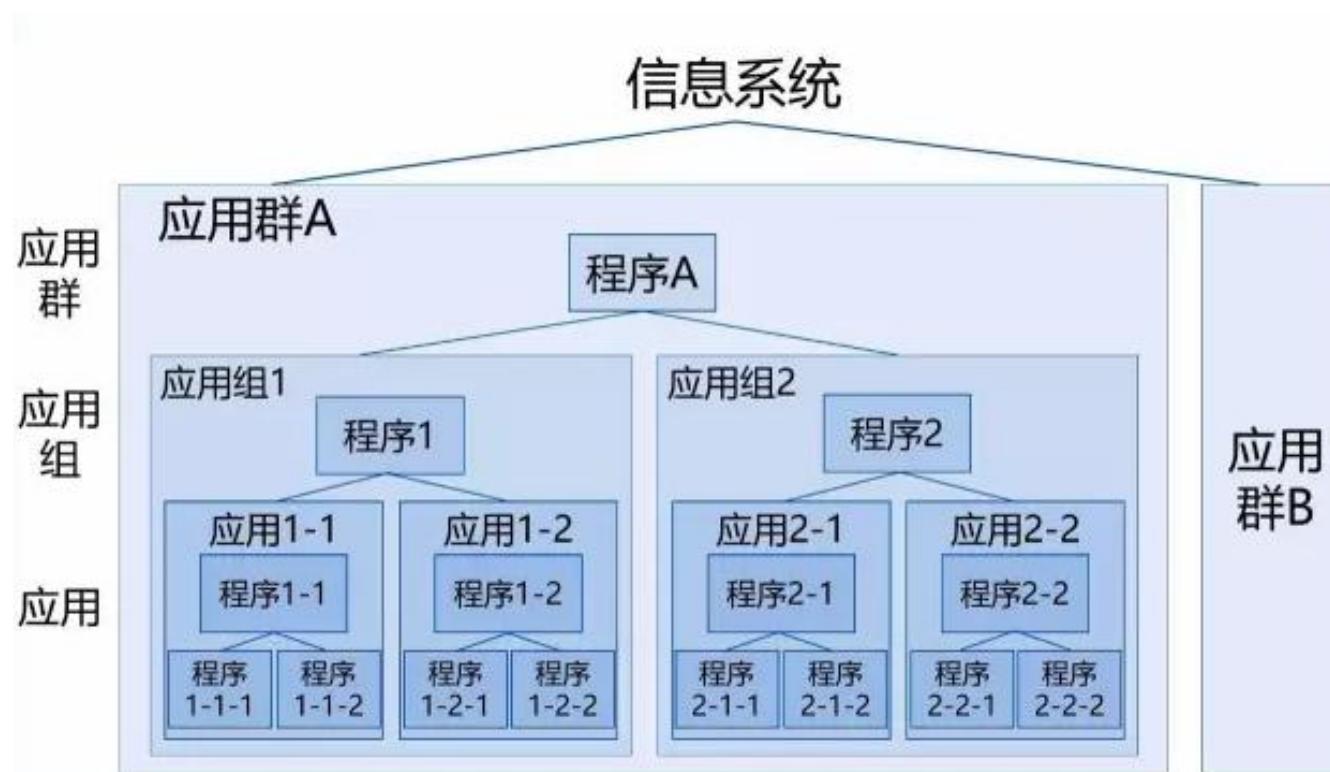
## • 5、流程架构

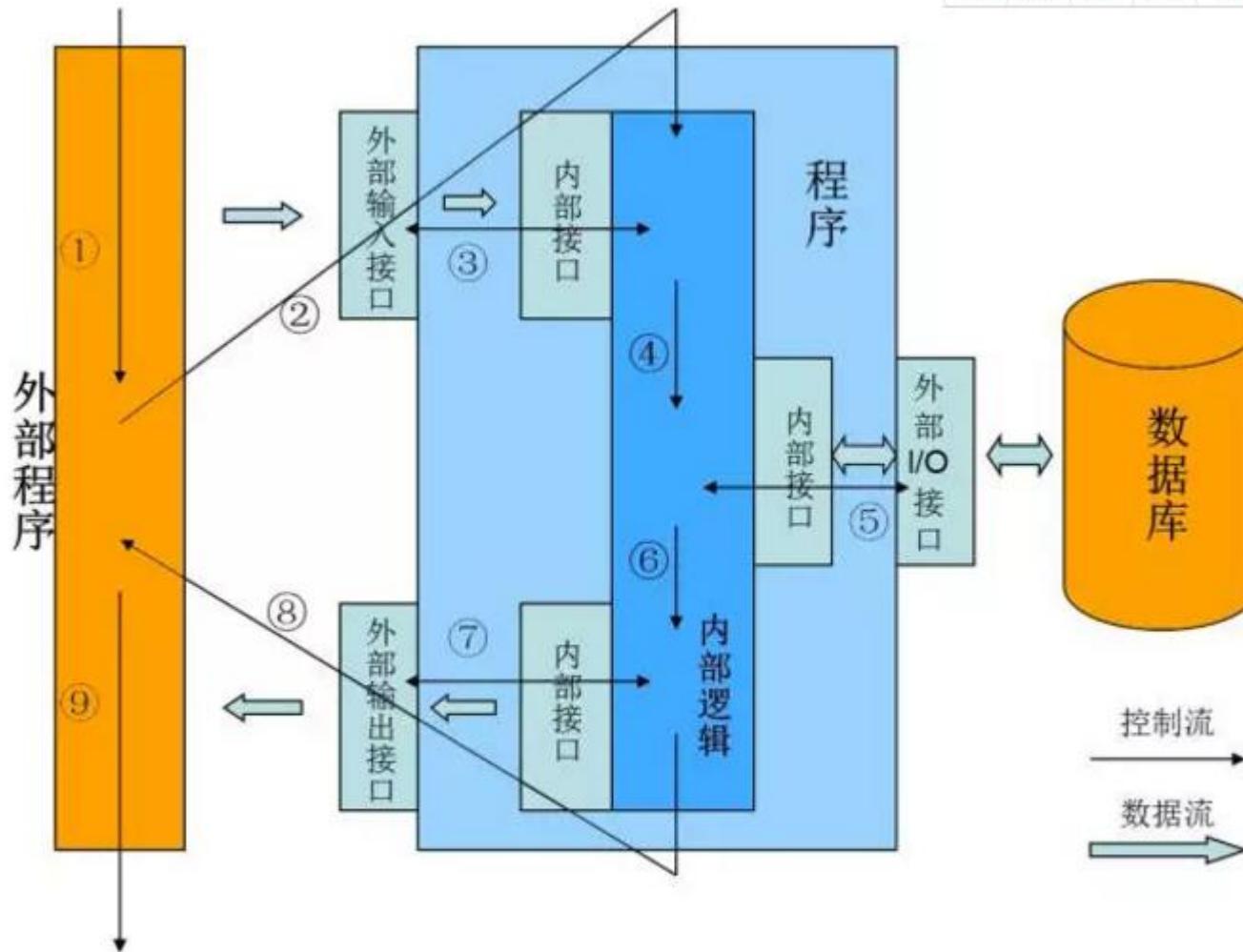
- 随着信息系统的发展，信息系统已经几乎可以承担银行中原来由人工完成的全部工作与管理。流程架构指的是，如何在信息系统里构建流程，一方面更好地完成已知的工作任务；另一方面，能够适应业务发展，快速定制流程，满足新业务的需要。

## • 6、基础设施架构

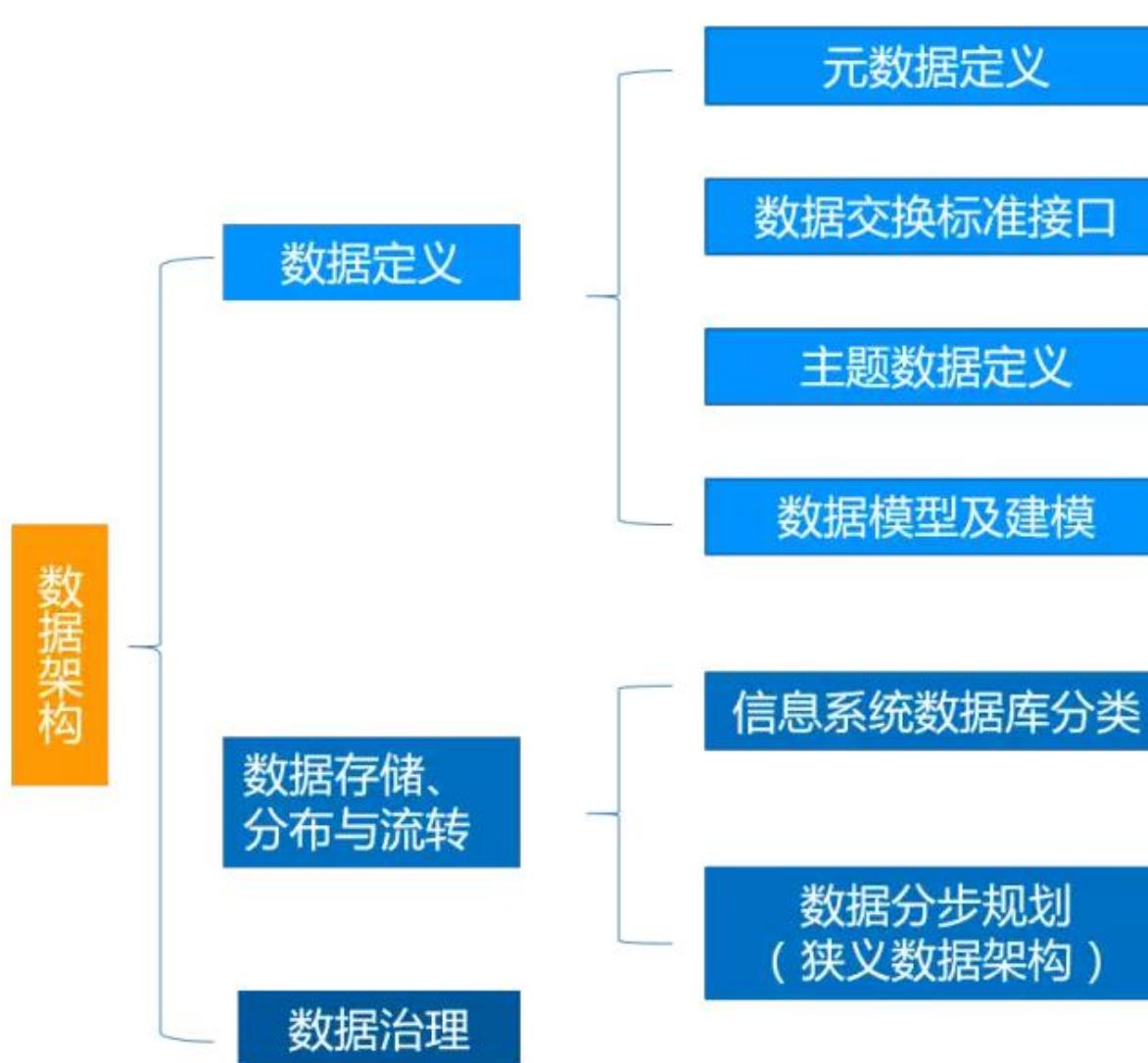
- 基础设施架构指的如何准确地理解应用与设备松耦合、与地域松耦合的概念，再根据应用需要部署各类物理设备。

- 应用架构：面向应用逻辑，功能追踪。
  - 通常 是树形结构



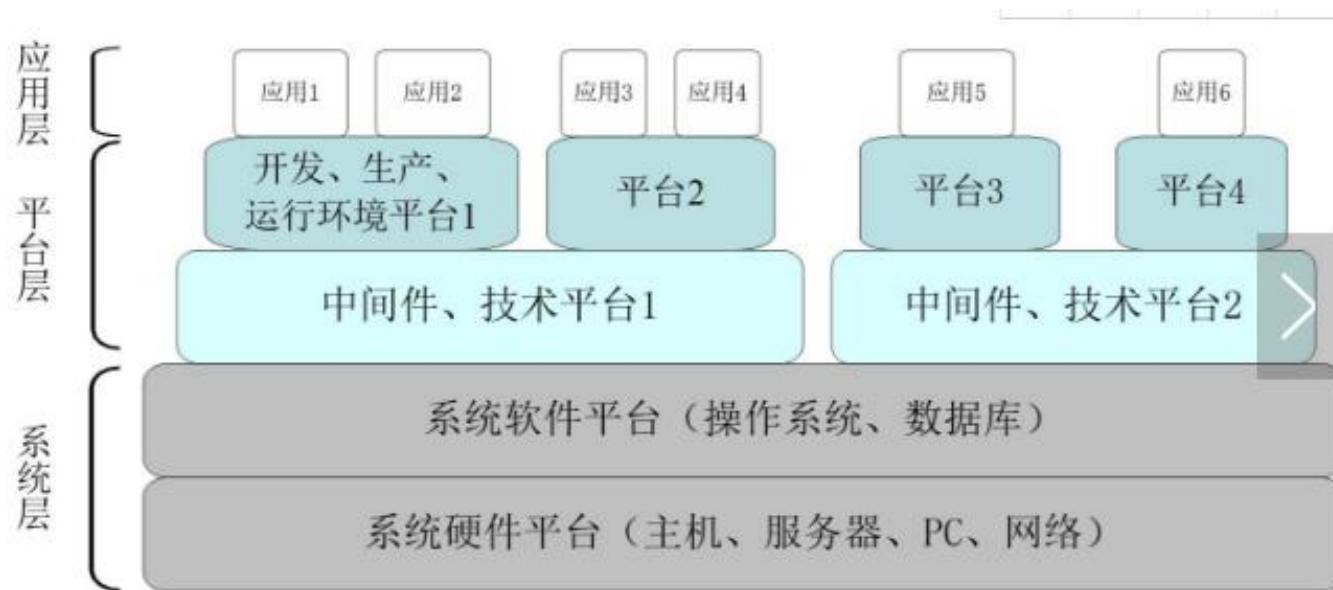


# 数据架构



# 技术架构

信息系统架构的其中一个维度，是系统的技术架构。本文所说的技术架构，指的是在一个可以独立部署的应用系统里，应用、应用平台、基础设施之间的关系。也就是说，技术架构探讨的是应用系统的纵向架构，而此前论述的应用架构探讨的是系统的横向架构。



图：系统的技术架构

# 第一部分 软件体系结构概述



- 什么是软件体系结构（粗略的）
- 为什么研究软件体系结构？
- 对软件体系结构的进一步认识
- 软件体系结构的重要性
- 什么是软件体系结构
- **软件体系结构的研究领域**
- 一个实例

- 软件体系结构概念被正式提出至今已经历了10多年的发展
  - 文章、书籍
  - 国际会议和研讨会
  - 研究团体
  - 课程、认证、标准、方法、工具等不胜枚举



- 软件体系结构模型的研究
  - 如4+1模型，结构模型、动态模型
  - 建模技术的研究
- 软件体系结构描述的研究
  - ADL语言的研究，相应的工具研究
  - 形式化机制
- 软件体系结构设计的研究
  - 体系结构设计方法
  - 体系结构风格 等

- 软件体系结构分析与验证
  - 如何将软件的非功能特性转化为体系结构需求，如何分析体系结构满足期望的需求属性等
  - ATAM等方法
- 基于体系结构的软件开发方法
  - 引入体系结构的开发过程
  - 中间件技术集成
  - 基于体系结构的程序框架自动生成技术
  - 软件产品线
  - MDA, MDE

- 面向特定领域的软件体系结构研究
- 软件体系结构自身的演化、逆向构造等
- 自适应体系结构、智能体系结构
- 系统重构

- 学术界：

- 更关注体系结构的完整的理论模型
- 体系结构的分析和评估
- 严格的符号建模和强有力地分析技术（ADL）
- 为特定目标提供解决方案

- 工业界

- 多视图建模
- 更强调实用性而非精确性（采用成熟的UML）
- 将体系结构视为开发蓝图

- 首要问题：如何对软件体系结构建模。通常，根据不同的侧重点，分为5类：
  - 结构模型：
  - 框架模型
  - 动态模型
  - 过程模型
  - 功能模型

# 关于UML的思考

## 静态建模制

类图  
(classdagram)

用例图  
(usecasedagram)

对象图  
(objectdagram)

构图  
(compositdagram)

部署图  
(deploymetedagram)

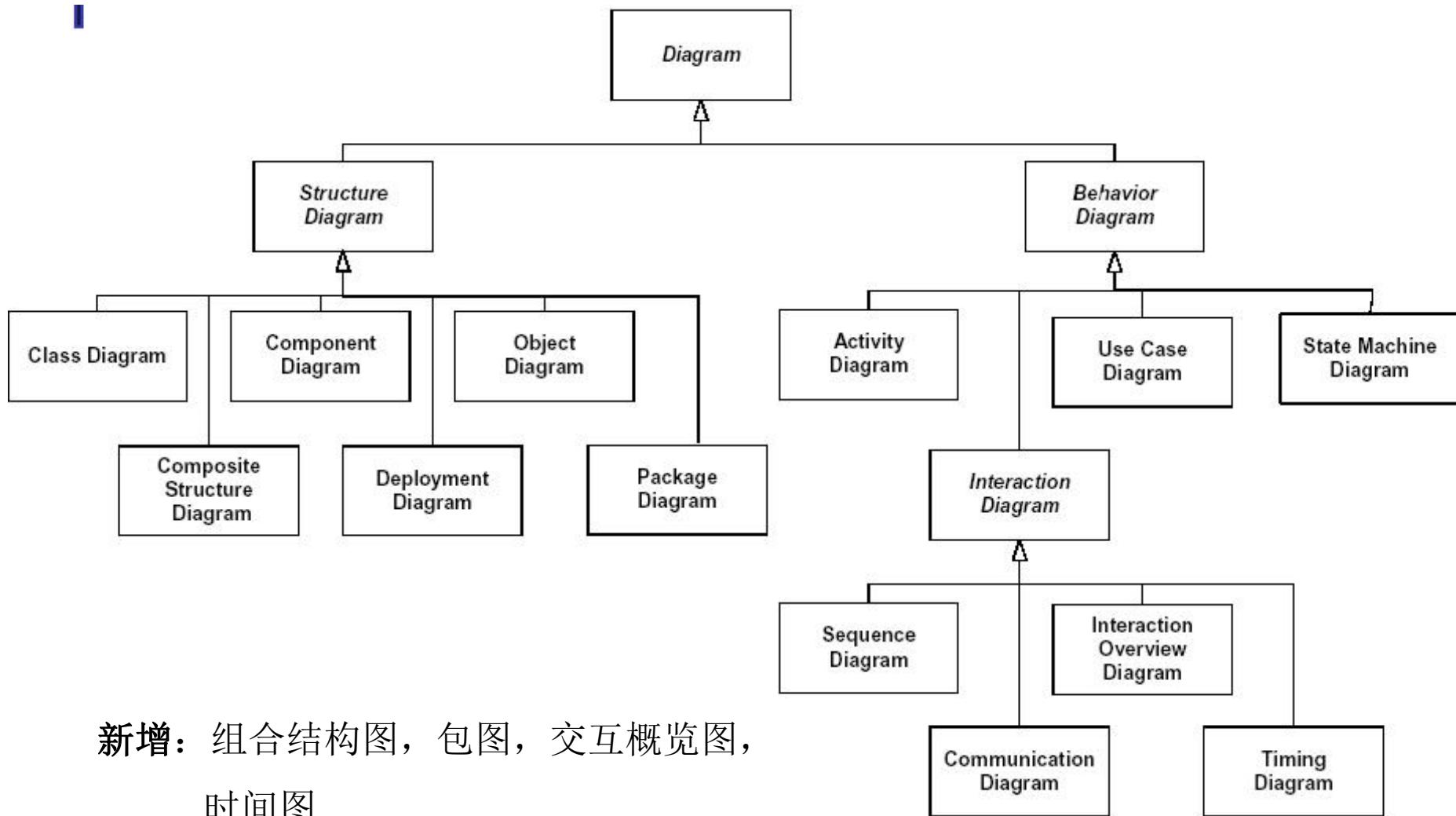
## 动态建模制

活图  
(activitydagram)

顺序图  
(sequencedagram)

会话图  
(collaborationdagram)

状态图  
(statedagram)



新增：组合结构图，包图，交互概览图，  
时间图

# 基于体系结构的软件开发

- 系统需求工程（由软件工程师、领域专家）
- 选择、开发和选用特定系统的软件模式（体系结构设计空间）
- 体系结构表示与设计（采用适当的表示模型和设计）
- 基于体系结构的分析、评估（进行软件需求属性分析）
- 按照体系结构实现软件系统（选择开发构件、构件组装）
- 验证软件系统与软件体系结构的一致性（检验）

## 软件体系结构相关的研究团体/机构及领军人物

领军人物	研究团体/机构	主要研究内容及成果
<u>Barry Boehm</u>	Center for Software Engineering at University of Southern California (CSE USC)	<p>该中心由Barry Boehm于1993年创建，主要从事大型系统设计和开发过程、通用以及特定领域软件体系结构、软件工程工具和环境的研究及教学工作。在软件体系结构方面，该研究组侧重于研究软件体系结构在整个软件生命周期中的作用，包括：软件体系结构设计、建模和细化、视图集成和一致性检查、软件体系结构描述语言与UML的集成、特定领域软件体系结构等。</p> <p>网址： <a href="http://sunset.usc.edu/about_center/index.html">http://sunset.usc.edu/about_center/index.html</a></p>
Richard N. Taylor	Institute of Software Research (ISR) at University of California, Irvine	<p>该课题组主要研究C2风格的软件体系结构的设计、描述、动态演化以及支持工具的开发，定义了软件体系结构描述语言C2、xADL，并在ADL的分析、比较以及与UML的融合方面做了大量研究工作。</p> <p>网址： <a href="http://www.isr.uci.edu/architecture/index.html">http://www.isr.uci.edu/architecture/index.html</a></p>

领军人物	研究团体/机构	主要研究内容及成果
Len Bass, Rick Kazman, Paul Clements	Software Engineering Institute, Carnegie Mellon University	<p>该研究组重点关注对软件体系结构开发和评审技术、以及软件产品线技术的研究，提出了多种软件体系结构评审方法，如SAAM<sup>1</sup>、ATAM、CBAM等。主要关心非功能属性（质量属性）对体系结构的影响，提出了属性驱动的软件体系结构设计方法ADD (Attribute Driven Design)、Quality Attribute Workshop (QAW)。这一研究组出版了大量关于软件体系结构的书籍和著作。目前，该研究组正在致力于将其软件体系结构方面的研究成果集成到一些较流行的软件开发过程方法当中，如RUP、XP等。</p> <p>网址：<a href="http://www.sei.cmu.edu/publications/publications.html">http://www.sei.cmu.edu/publications/publications.html</a></p>
David Garlan, Mary Shaw	Composable Systems Group, School of Computer Science, Carnegie Mellon University	<p>Garlan 和Shaw对软件体系结构研究有非常重要的推动作用，他们的合著” <b>Software Architecture: Perspectives on an Emerging Discipline</b>”成为软件体系结构研究领域被引用最多的著作之一。该研究组在软件体系结构研究方面处于较为领先地位，其研究内容极为广泛，包括：软件体系结构基础理论、描述语言(Aesop、Wright、Unicon、ACME)及支持工具(Aesop、AcmeStudio)、软件体系结构风格形式化理论、软件体系结构风格整理和分类、软件体系结构学科教育和培训、动态软件体系结构、软件体系结构细化、以及软件体系结构通用文档化技术等。</p> <p>网址：<a href="http://www-2.cs.cmu.edu/~able/">http://www-2.cs.cmu.edu/~able/</a></p>

领军人物	研究团体/机构	主要研究内容及成果
David Luckham	Program Analysis and Verification Group, Department of Electrical Engineering, Stanford University	该研究组主要侧重基于事件的、复杂、并发、分布式系统的软件体系结构设计和描述研究，定义了软件体系结构描述语言Rapide。
唐稚松	中国科学院软件研究所计算机科学重点实验室	在软件体系结构方面的主要研究成果包括软件体系结构描述语言XYZ/ADL和软件体系结构细化方法等。
杨芙清, 梅宏	北京大学信息科学技术学院软件工程研究所	该研究机构从事的研究方向主要包括软件工程基础理论、软件复用技术、面向对象方法与技术等。在软件体系结构方面的主要研究成果包括基于消息总线的软件体系结构风格及描述语言，基于软件体系结构的可复用构件制作和组装技术，基于体系结构、面向构件的软件开发方法ABC及支持工具(ABCTool、PKUAS)，以及运行时软件体系结构等。 网址： <a href="http://sei.pku.edu.cn">http://sei.pku.edu.cn</a>

# 第一部分 软件体系结构概述

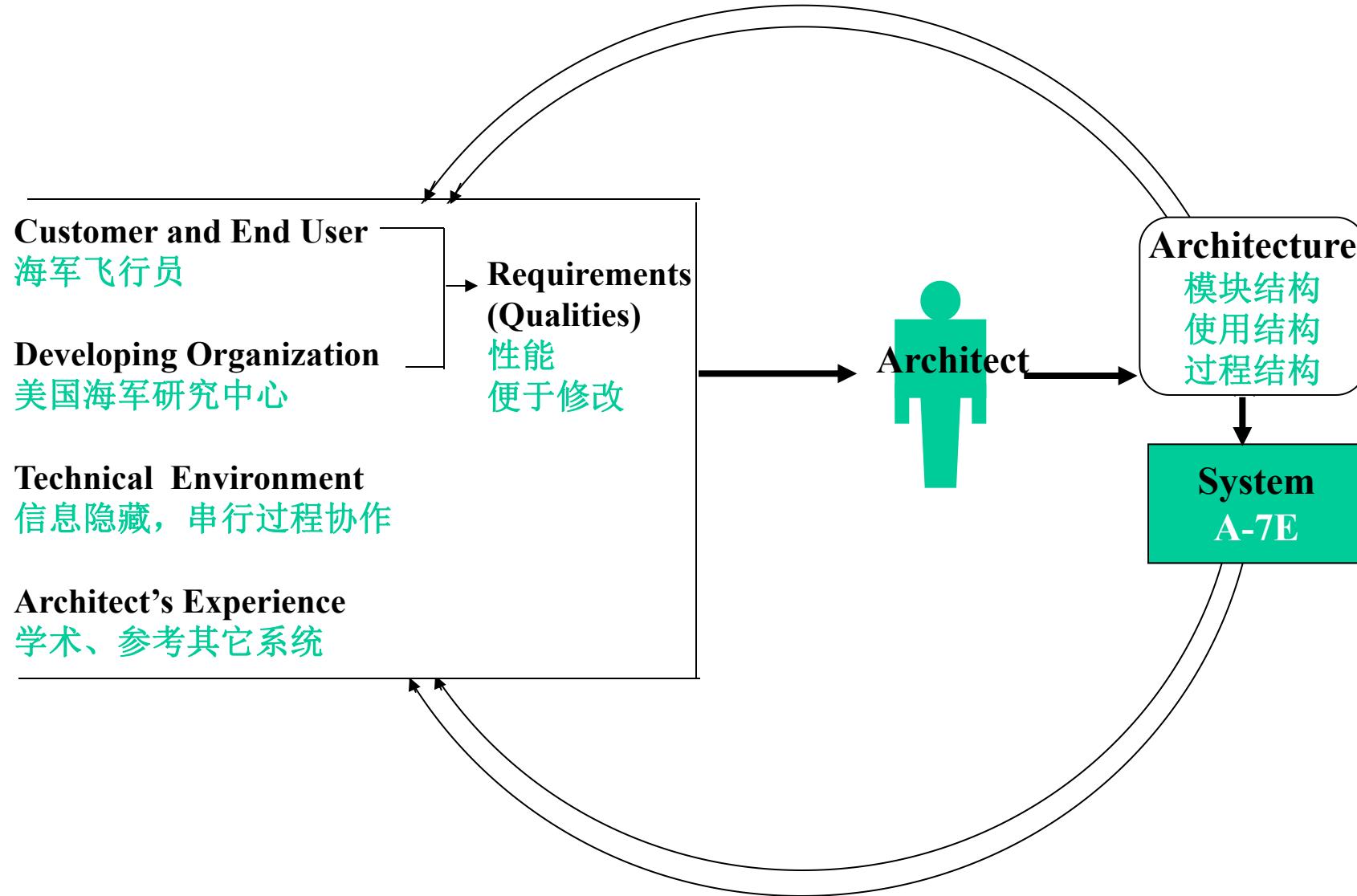
- 什么是软件体系结构（粗略的）
- 为什么研究软件体系结构？
- 对软件体系结构的进一步认识
- 软件体系结构的重要性
- 什么是软件体系结构
- 软件体系结构的研究领域
- 一个实例



**A case study  
in utilizing architectural structure**

**A-7E项目**

- 海军需求：需求降低成本（20世纪70年代）
- 通过令人信服的案例向人们展示当时未广泛应用的软件设计和实现技术：
  - 设计中遵循信息隐藏的原则
- A-7E项目的指导思想是留下一个完整的工程模型，把相关的文档、设计方案、代码、方法和原则公之于众，供相关人员模仿使用。

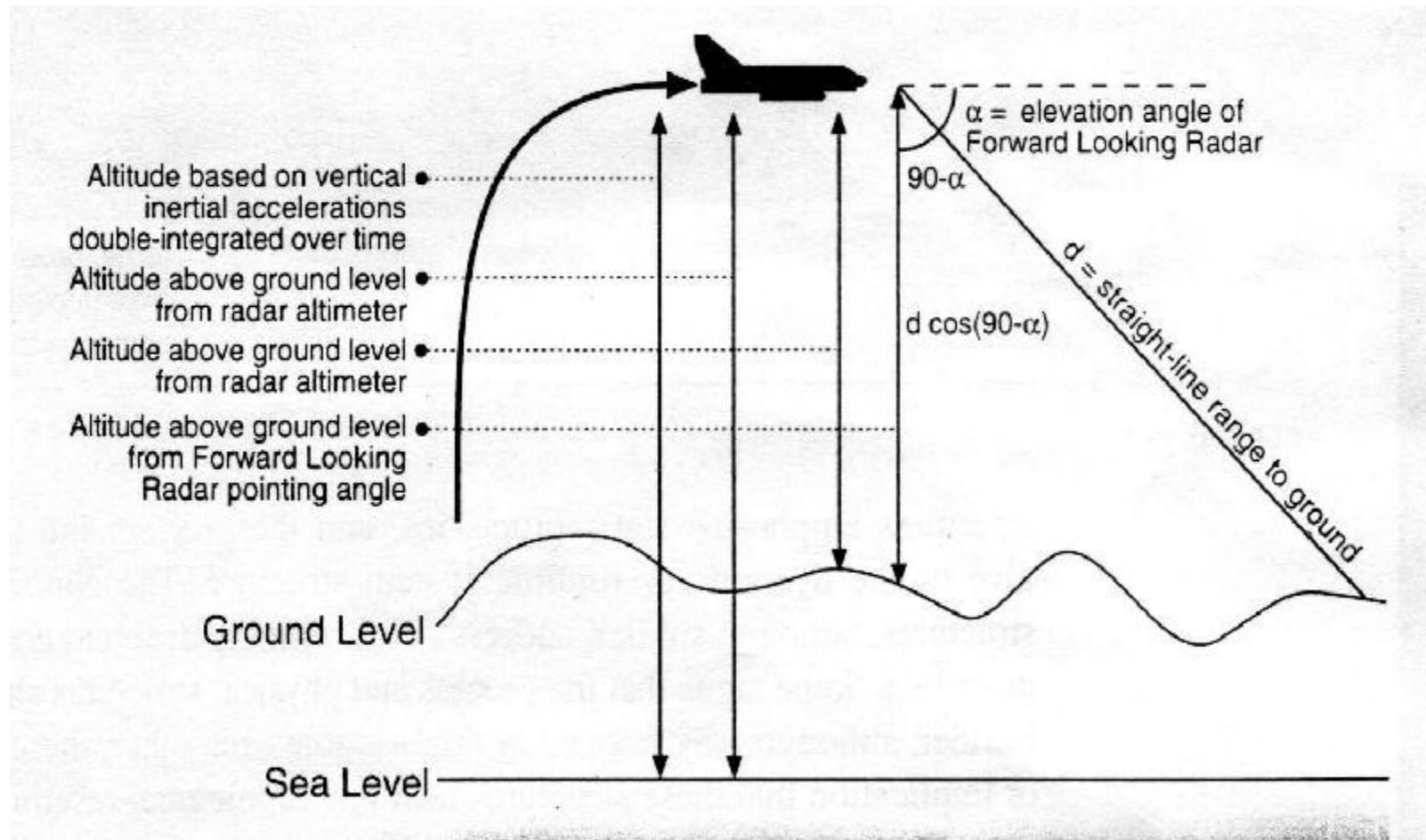


# An A-7E Corsair II



**FIGURE 3.2** An A-7E Corsair II. Used with permission and under copyright of Squadron/Signal Publications, Inc.

# Requirements



结构	构件	构件间的关系	影响
模块结构	模块 (任务分配)	父子模块关系; 共享秘密关系	更改的难易程度
使用结构	过程 (Procedures)	要求正确存在	实现子集和 渐进开发的能力
进程结构	进程Processes; thread of procedures	同步关系; 共享CPU; 互斥	可调度性; 通过并发执行实现 性能目标

- 模块设计：基于信息隐藏原则（封装）
- 模块划分的具体目标：
  - 每个模块的结构应足够简单。能够充分理解
  - 应该能够在无需了解其他模块的具体实现，并且不影响其他模块的功能的情况下修改某个模块的内部实现
  - 对设计进行修改的容易程度应该与该修改可能发生的程度有合理的对应关系
  - 应该能够把要对软件系统做的比较大的改动分解成对各个模块的一组独立的修改

## 目标

## 如何实现

易于使用

信息隐藏

理解未来可能的改动

正式的评审过程，以吸取相关领域专家的经验

为各开发小组分配任务，  
使各小组之间的交流最  
少

按层次结构组织各模块，每个开发小组负责  
开发一个二级模块及其所有子模块。

## 硬件隐藏模块

### --Extended Computer Module

Data Module  
Input/Output Module  
Computer State Module  
Parallelism Control Module  
Program Module  
Virtual Memory Module  
Interrupt Handler Module  
Timer Module

### --Device Interface Module

Air Data Computer Module  
Angle of Attack Sensor  
Audible Signal Device  
Computer Fail Device  
Doppler Radar Set Module  
Flight Information Display  
Forward Looking Radar  
Head-up Display Module  
Inertial Measurement Unit  
Input-Output Representation  
Master Function Switch  
:  
:  
Weapon Characteristics  
Weapon Release System

## 行为隐藏模块

### --Function Driving Module

Air Data Computer Module  
Audible Signal Module  
Computer Module  
Information Display Module  
Intelligent Radar Module  
Head-up Display Module  
Measurement Module

### --Application Data Type Module

Numeric Data Type Module  
State Transition Event Module

### --Data Banker Module

Singular Values Module  
Complex Event Module

### --Filter Behavior Module

### --Physical Models Module

Aircraft Motion Module  
Earth Characteristics Module  
Human Factors Module  
Target Behavior Module  
Weapon Behavior Module

### --Software Utility Module

Power-Up Initialization Module  
Numerical Algorithms Module

### --System Generation Module

System Generation Parameter Module  
Support Software Module

- 1、引言
- 2、接口概述
  - 访问过程表
  - 事件信号
- 3、本地类型字典（数据类型）
- 4、字典（术语）
- 5、不期望事件字典
- 6、系统生成参数
- 7、接口设计问题
- 8、实现备忘
- 9、假设列表
  - 基本假设
  - 关于不期望事件的假设
- 10、精度、时间和效率指南

区分：

**Use、interaction、call relation.**

## 目标

增量式逐步实现和  
测试系统的各个功能

提高可移植性

生成规模合理的  
使用关系指南

## 何如实现

为程序员创建“被允许使用”结构，限制每个  
程序员所能使用的过程

以层次结构描述使用结构

在适当的情况下，将使用关系定义为模块间  
的关系

# Layered Architecture of A-7E

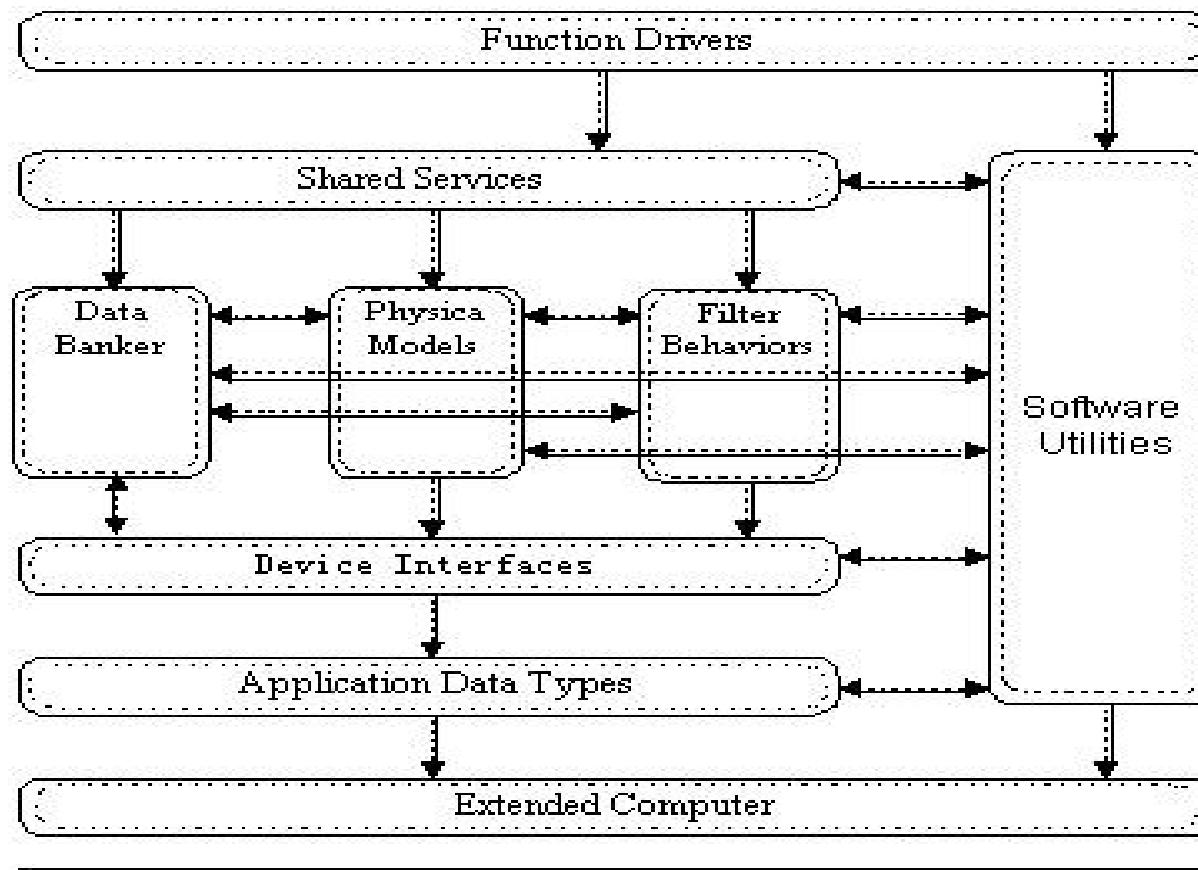


FIGURE The layered architecture that emerges from the A-7E uses structure.

## 目标

实现从输入到输入的映射

满足实时需求

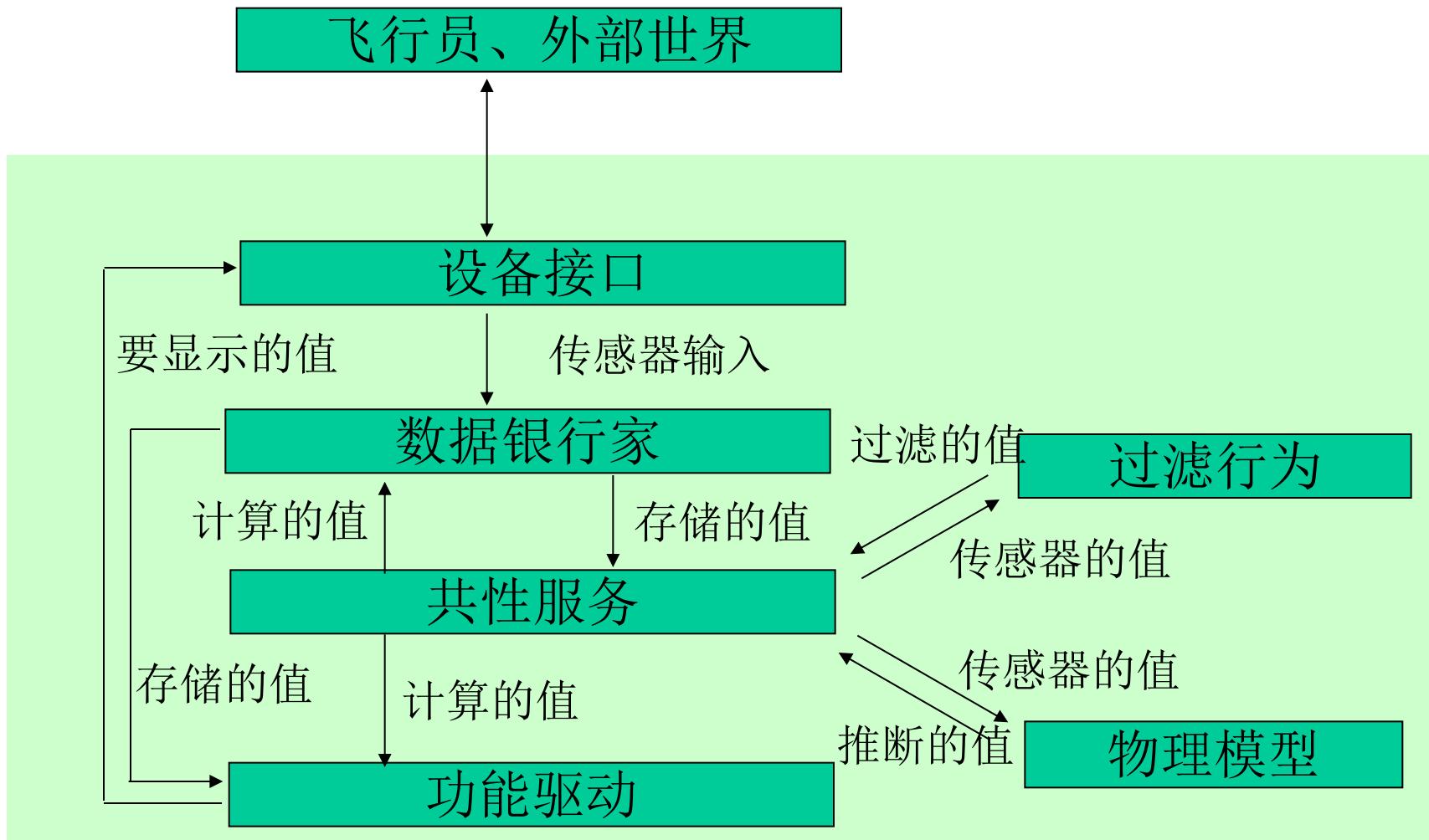
及时提供运算量大的计算结果

## 如何实现

每个进程都作为一个采样、输入、计算、输出的周期来实现

通过进程结构确定进程，实现进程的非在线调度

后台计算，并在需要时将最新结果返回。



- 粗粒度数据流图

## 二、软件体系结构建模与描述

- 软件体系结构模型及建模
- 软件体系结构描述
- 软件体系结构文档 (documentation)

## 三、创建软件体系结构（设计）

- 理解质量属性
- 实现质量属性：软件体系结构风格、软件体系结构设计的原子操作、软件体系结构实现战术
- 案例分析

## 四、软件体系结构分析和评估

- 软件体系结构分析
- 软件体系结构评审

## 五、面向领域的软件体系结构

- 特定领域的软件体系结构
- 软件产品线

## 六、网络环境下的软件体系结构

- SOA
- 网构软件
- 网络化软件
- 服务架构

- Course.buaa.edu.cn

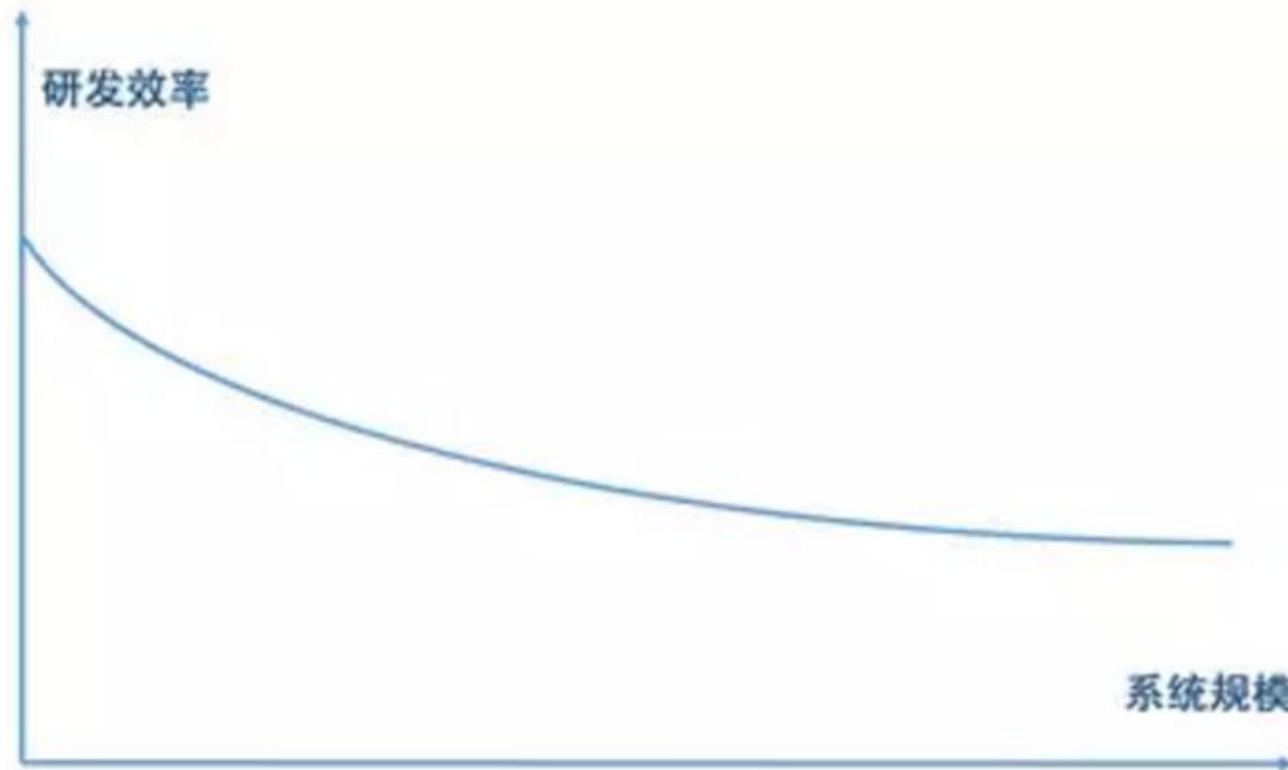


图1 系统规模与研发效率的关系