高等教育“十二五”规划教材

# 软件体系结构

## Software Architecture

王小刚　黎　扬　周　宁　编著

北京交通大学出版社
http://www.bjtup.com.cn

高等教育“十二五”规划教材

# 软件体系结构

王小刚　黎　扬　周　宁　编著

北京交通大学出版社

·北京·

## 内 容 简 介

本书系统阐述了软件体系结构的概念、思想，以及建模和描述手段，介绍了软件体系结构的重要模式及应用。指导如何设计出高质量的符合需求的软件体系结构，给出了评估软件体系结构的方法，论述了当前流行技术的软件体系结构。全书强调思想和理念，重视实用和实践。

全书共分9章，第1章介绍软件体系结构的起源、概念、作用和发展等；第2章给出了软件体系结构模式（风格）的分类，列举了经典的风格实例，并通过例子阐述了使用方法；第3章是关于软件体系结构描述，包括IEEE 1471标准、软件体系结构文档、软件体系结构描述语言、动态软件体系结构及描述；第4章讲述软件体系结构的设计思想、原则、方法及流程，强调要关注的问题，说明软件体系结构恢复和重构的概念及意义，给出了重构模式；第5章阐述软件体系结构与软件质量的密切关系，以及实现质量属性的体系结构策略；第6章论述软件体系结构评估的意义和方法，重点是ATAM评估方法；第7章简单介绍了面向特定领域软件体系结构和软件产品线；第8章说明了基于Java和.NET的分布式体系结构及组成要素；第9章介绍了几种主流技术软件的体系结构及关键技术，包括SOA、Android、云计算等。

本书可作为高等院校软件工程专业及其他计算机类专业的软件体系结构教材，也可作为软件工程相关技术人员的参考用书。



**图书在版编目（CIP）数据**

软件体系结构 / 王小刚，黎扬，周宁编著. —北京：北京交通大学出版社，2014.4
（高等教育“十二五”规划教材）
ISBN 978-7-5121-1881-2

Ⅰ.① 软… Ⅱ.① 王… ② 黎… ③ 周… Ⅲ.① 软件-系统结构-高等学校-教材 Ⅳ.① TP311.5

中国版本图书馆CIP数据核字（2014）第064415号

责任编辑：郭碧云
出版发行：北京交通大学出版社　　电话：010-51686414
地　　址：北京市海淀区高梁桥斜街44号　邮编：100044
印 刷 者：北京瑞达方舟印务有限公司
经　　销：全国新华书店
开　　本：185×260　印张：12.25　字数：306千字
版　　次：2014年4月第1版　2014年4月第1次印刷
书　　号：ISBN 978-7-5121-1881-2/TP·784
印　　数：1～1 000册　定价：29.00元

本书如有质量问题，请向北京交通大学出版社质监组反映。对您的意见和批评，我们表示欢迎和感谢。
投诉电话：010-51686043，51686008；传真：010-62225406；E-mail：press@bjtu.edu.cn。

# 前　言

软件体系结构将系统描述为计算组件及组件之间的交互，同时，软件体系结构也是在一些重要方面所做出决策的集合。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。体系结构涉及的问题包括总体组织和全局控制，通信协议，同步，数据存取，给设计元素分配特定功能，设计元素的组织、规模和性能，在各设计方案间进行选择等。

软件体系结构研究已经成为计算机科学的一个重要研究方向和独立学科分支。软件体系结构研究的主要内容涉及软件体系结构描述、软件体系结构风格、基于软件体系结构的软件开发模型、软件体系结构评估和软件体系结构的形式化方法等。解决好软件的重用、质量和维护问题，是研究软件体系结构的根本目的。

在工程实践中，对于大型复杂软件，如果没有设计出合适的软件体系结构，后续工作将很难进行。如果设计了错误的软件体系结构，那返工的代价必然庞大，甚至可能导致项目彻底失败和取消。

对于软件工程及其他计算机类专业的学生来说，必须理解软件体系结构的基本概念和理论，掌握主流的软件体系结构风格，熟悉软件体系结构的设计过程和设计方法，明确基于体系结构的软件开发流程。引导学生认识到系统的质量很大程度上受软件体系结构制约，提高学生宏观分析问题的能力和软件总体设计的能力，加强对软件体系结构的整体认知、分析和处理能力，增强创新能力，为开发大型软件打下基础。本书正是为此目的而编写的教材。

本书是作者多年教学经验的总结，同时作者也参考、整理和引用了国内外一些公开的专著、教材、论文、报告及网络资源的部分内容，以介绍对软件体系结构知识的公共理解、成熟表述，并准确说明软件体系结构的研究进展。非常感谢这些作品为本书提供的丰富营养，恕不一一列举，详见主要参考文献。

本书由兰州交通大学王小刚、黎扬、周宁三位老师编著而成，其中第 1、3、4、9 章由王小刚编著，第 2、5、6、8 章由黎扬编著，第 7 章由周宁编著。王小刚总体负责、规划和统稿。

由于作者水平有限，编写时间仓促，且软件体系结构研究和应用领域很多，诸多概念尚难统一，学科又在快速发展，因此书中难免有不妥及错误之处，敬请各位读者不吝赐教！

作　者

**2014 年 2 月**

# 目　录

# 第1章

# 软件体系结构概览

我们谈到交响乐的“架构（architecture）”，反过来，又将架构（architecture）称为“凝固的音乐”。

——Deryck Cooke，*The Language of Music*（音乐的语言）

什么是体系结构？如果你问五个不同的人，可能会得到五种不同的答案。

——Ivar Jacobson，《AOSD 中文版》

## 1.1 复杂性——我们的敌人

先从一个著名的例子——“瓦萨”号战舰（图 1-1）说起。这个承载了瑞典海军强国梦想的巨大战舰，经过著名设计师的设计，云集众多的木匠、锯工、铁匠、制绳匠、玻璃匠、画师、制箱艺人、木雕师傅及其他行业的工匠，历时三年建成。1628 年，“瓦萨”号停泊在王宫之下准备首航。

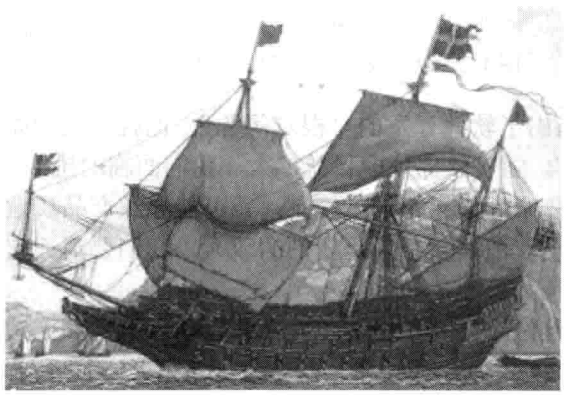

图 1-1 “瓦萨”号战舰

“启动了！”人群中一片欢呼。但是离岸还没来得及扬帆远航的“瓦萨”号在一阵大风浪过后，开始倾斜，接着又慢慢恢复平衡，但随即再一次朝右舷倾斜。岸上的人们都惊得目瞪口呆。“瓦萨”号的下层甲板在慢慢进水，舰体开始晃动下沉。“瓦萨”号就在众目睽睽之下沉没了，50 多名船员成了陪葬品。

运用现代科技我们了解到，“瓦萨”号是上部太重，正如船长所说：“一股轻风就足够了”。那么到底是谁的过错呢？海军重臣弗莱明在做稳定性试验时在场，他明知道战舰有问题而没有阻止，有罪！国王古斯塔夫二世急于拥有火炮越多越好的战舰，是他首肯了“瓦萨”号的规模，并急于求成，有责！

最后，战舰设计师，这位资深的荷兰造船师亨瑞克·慧贝特松经验丰富，曾经建造过多艘名舰，但却没有设计过双层战舰。这样一个庞大的工程涉及众多人员，从设计人员到施工人员，从国王到工程师，要考虑方方面面的问题，如结构、可靠性、规模、质量，任何地方考虑不周全都会导致问题。

注：资料来自百度百科。

我们的软件项目，有像国王、海军大臣这样的管理者，有战舰设计师这样的设计人员，也有像木匠、锯工等各种各样的实施人员；要使用多种工具和技术，经过很多复杂的工序和过程；既要考虑功能，又要考虑可靠性、规模等各种质量因素，如图 1-2 所示。

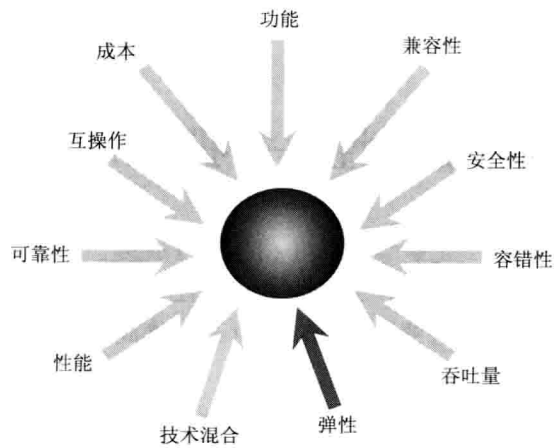


图 1-2 软件设计中要考虑的因素（一部分）

《人月神话》一书指出了软件开发的“根本问题”，也就是无论如何回避不掉的问题，首当其冲的就是“复杂性”。随着软件规模日益扩大，我们面临的挑战不是速度、成本和性能，而是复杂性的问题了。复杂性如果不加以控制，就可能导致产品质量低下、延迟交付、超预算，或者项目可能被完全取消。复杂性是我们的敌人，我们要消灭它，但是并没有轻松的解决方案可消除复杂性。

一股脑儿地将复杂性统统展开，就会使软件开发陷入混乱和无法控制之中。先进行软件体系结构设计（划分子系统、层、模块等，确定设计策略），后进行详细设计和编码实现，运用“分而治之”的理念，利于控制复杂性，可保证开发出高质量的软件。

所以，克服软件危机，控制软件开发的复杂性问题，体系结构设计有着非常重要的作用。

## 1.2 什么是软件体系结构

软件体系结构也叫软件架构，其定义有很多，也难以统一，如同坐在不同位置看舞台一样，各有自己的角度。温昱在《软件架构设计》中将软件体系结构概念分为两大流派：组成派和决策派，可以认为各种定义围绕组成和决策展开。

注：软件体系结构与软件架构的英文都是 Software Architecture，两者都使用一样的定义。如 IEEE 的定

义：一个系统的基础组织，包含各个构件、构件互相之间的关系，以及构件与环境的关系，还有指导其设计和演化的原则。

对软件开发者和软件应用者来说，倾向于使用“软件架构”，在一定程度上使用“软件体系结构”。大家对软件架构的设计人员——“架构师”广泛地认同。对于学术界，普遍使用“软件体系结构”，对架构师关注很少。Software Architecture 是一个实践性非常强的领域，统计表明理论和实践的鸿沟还是存在的。

体系结构的中文定义完全符合 IEEE 等的定义，强调整体与部分、部分与部分的关系，研究系统构成的方法学，提倡多角度研究系统。其次，从学科地位讲，作为一门独立的软件子学科，与硬件学科（计算机组织与体系结构）直接对应。

从工程实践的需要看，软件架构更能体现系统构成与相关技术。RUP 过程或软件生产线关注的软件架构并不注重原理及表示，而是由结构和技术相结合的形成框架。

总体来说，如果强调方法论，应使用软件体系结构。强调软件开发实践，应使用软件架构。

因为侧重点不同，本书根据需要采用软件体系结构和软件架构两个术语。

### 1.2.1 组成视角

从组成的视角看，软件体系结构=组件+交互。

Mary Shaw 在《软件体系结构：一门初露端倪学科的展望》中，为“软件体系结构”给出了非常简明的定义：

软件体系结构将系统描述为计算组件及组件之间的交互（The architecture of a software system defines that system in terms of computational components and interactions among those components）。

一般认为，组件（构件）是指语义完整、语法正确和有可重用价值的单位软件，是软件重用过程中可以明确辨识的系统；结构上，它是语义描述、通信接口和实现代码的复合体。

对于组件的定义有狭义和广义之分。狭义的组件是指和 CORBA、DCOM、EJB 等相关的专有的组件概念。

广义的“组件”是广泛意义上的元素之意。Mary Shaw 的定义“计算组件”是泛指，计算组件可以进一步细分为处理组件、数据组件、连接组件等。总之，广义“组件”指子系统、框架（Framework）、模块、类等不同粒度的软件单元，它们可以担负不同的计算职责。

上述定义是“组成派”软件体系结构概念的典型代表，有如下两个显著特点：

① 关注体系结构实践中的客体——软件，以软件本身为描述对象；

② 分析了软件的组成，即软件由承担不同计算任务的组件组成，这些组件通过相互交互完成更高层次的计算。

下面是更多地侧重于“组成派”的定义。

Barry Boehm：软件体系结构包括系统构件、连接件和约束的集合，反映系统需求说明的集合，以及原理的集合。原理用以说明构件、连接件和约束所定义的系统如何在实现时满足不同涉众的需要。

Bass：软件体系结构包括一个或一组软件构件、软件构件外部的可见特性及其相互关系。其中，“软件外部的可见特性”是指软件构件提供的服务、性能、特性、错误处理、共享资源

使用等。

Dewayne Perry 和 Alexander Wolf：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合连接起来。

IEEE：软件体系结构是以组件、组件之间的关系、组件与环境之间的关系为内容的某一系统的基本组织结构，以及指导上述内容设计与演化的原理。

### 1.2.2 决策视角

下面来看看 RUP（Rational Unified Process，Rational 统一过程）中的软件体系结构定义。软件体系结构包含关于以下问题的重要决策：

① 软件系统的组织；

② 选择组成系统的结构元素和它们之间的接口，以及当这些元素相互协作时所体现的行为；

③ 如何组合这些元素，使它们逐渐合成为更大的子系统；

④ 用于指导这个系统组织的体系结构风格，即这些元素与它们的接口、协作和组合；

⑤ 软件体系结构并不仅仅注重软件本身的结构和行为，还注重其他特性，如使用、功能性、性能、弹性、重用、可理解性、经济和技术的限制及权衡，以及美学等。

上述定义看似冗长，但其核心思想非常明确：软件体系结构是在一些重要方面所做出的决策的集合。该定义是“决策派”软件体系结构概念的典型代表，有如下两个显著特点：

① 关注体系结构实践中的主体——人，以人的决策为描述对象；

② 归纳了体系结构决策的类型，指出体系结构决策不仅包括关于软件系统的组织、元素、子系统和体系结构风格等几类决策，还包括关于众多非功能需求的决策。

下面给出更多决策派的定义。

Mary Shaw 和 David Garlan：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。体系结构问题包括总体组织和全局控制、通信协议、同步、数据存取，给设计元素分配特定功能，设计元素的组织、规模和性能，在各设计方案间进行选择等。

Woods：软件体系结构是一系列设计决策，如果做了不正确的决策，你的项目可能最终会被取消。

Booch、Rumbaugh 和 Jvacobson：软件体系结构是一系列重要决策的集合，这些决策与以下内容有关：软件的组织，构成系统的结构元素及其接口的选择，这些元素在相互协作中明确表现出的行为，这些结构元素和行为元素进一步组合所构成的更大规模的系统，以及指导这一组织——包括这些元素及其接口、协作和组合——体系结构风格。

## 1.3 软件体系结构核心模型

综合软件体系结构的概念，体系结构的核心模型由 5 种元素组成：构件（Component）、连接件（Connector）、配置（Configuration）、端口（Port）和角色（Role）。其中构件、连接

件和配置是最基本的元素。

① 构件是具有某种功能的可重用的软件模板单元，表示系统中主要的计算元素和数据存储。构件有两种：复合构件和原子构件。复合构件由其他复合构件和原子构件通过连接而成；原子构件是不可再分的构件，底层由实现该构件的类组成。这种构件的划分提供了体系结构的分层表示能力，有助于简化体系结构的设计。

② 连接件表示构件之间的交互，简单的连接件如管道（Pipe）、远程过程调用（Remote Procedure Call，RPC）、事件广播（Event Broadcast）等，更为复杂的交互如客户—服务器通信协议、数据库和应用之间的 SQL 连接等。

③ 组件作为一个封装的实体，只能通过其接口与外部环境交互，组件的接口由一组端口组成，每个端口表示组件和外部环境的交互点。通过不同的端口类型，一个组件可以提供多重接口。一个端口可以非常简单，如过程调用，也可以表示更为复杂的接口（包含一些约束），如必须以某种顺序调用的一组过程调用。

④ 连接件作为建模软件体系结构的主要实体，同样也有接口。连接件的接口由一组角色组成，连接件的每一个角色定义了该连接件所表示的交互的参与者，二元连接件有两个角色，如 RPC 的角色为调用者（caller）和被调用者（callee），管道的角色是数据源（soure）和接收端（sink），消息传递连接件的角色是发送者（sender）和接受者（receiver）。有的连接件有多于两个的角色，如事件广播有一个事件发布者角色和任意多个事件接收者角色。软件体系结构的核心模型如图 1-3 所示。

⑤ 配置表示构件和连接件的拓扑逻辑及约束。

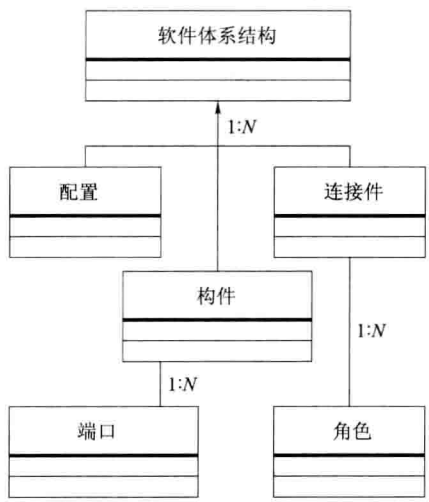


图 1-3　软件体系结构核心模型

## 1.4　软件体系结构起源

计算机软件的历史开始于 20 世纪 50 年代，历史非常短暂。相比之下，建筑工程则从石器时代就开始了，人类在几千年的建筑设计实践中积累了大量的经验和教训。独特的建筑风

格和恰当的建筑模式，可以使一个建筑独一无二。图 1-4 是中美洲古代玛雅建筑，Chichen-Itza 大金字塔，九个巨大的石级堆垒而上，91 级台阶（象征着四季的天数），塔顶的神殿耸入云霄。所有的数字都如日历般严谨，风格雄浑。难以想象这是石器时代的建筑物。

图 1-4 玛雅城邦遗址

英国首相丘吉尔说，我们构造建筑物，然后建筑物塑造我们（We shape our buildings, and afterwards our buildings shape us）。自从有了建筑以来，建筑与人类的关系就一直是建筑设计师必须面对的核心问题。

计算机软件的历史虽然非常短暂，但先前人类活动总结出的经验依然可以在这个新兴的领域得到应用。几乎所有的软件设计理念都可以在浩如烟海的建筑学历史中找到更为遥远的历史回响。例如，可以沿袭古罗马建筑师 Vitruvius 的设计原则，作为一些系统活动的参考：持久（Durability）、实用（Utility）和美（Beauty）。类似地，在建筑学界，现代主义建筑流派的开创人之一 Louis Sullivan 也认为形式应当服从于功能（Forms follows function），软件设计界也有很多人认为功能是最为重要的，形式必须服从于功能。

软件与人类的关系是架构师必须面对的重要问题，也是自从软件进入历史舞台之后就出现的问题。

20 世纪 60 年代的软件危机使得人们开始重视软件工程的研究。起初，人们把软件设计的重点放在数据结构和算法的选择上，随着软件系统规模越来越大、越来越复杂，整个系统的结构和规格说明显得越来越重要。软件危机的程度日益加剧，现有的软件工程方法对此显得力不从心。对于大规模的复杂软件系统来说，对总体的系统结构设计和规格说明比起对算法和数据结构的选择要重要得多。在这种背景下，人们认识到软件体系结构的重要性，并认为对软件体系结构系统进行深入的研究，将会成为提高软件生产率和解决软件维护问题的新的最有希望的途径。

自从软件系统首次被分成许多模块，模块之间有相互作用，组合起来有整体的属性，就有了体系结构。好的开发者常常会使用一些体系结构模式作为软件系统结构设计策略，但他们并没有规范地、明确地表达出来，这样就无法与别人交流。软件体系结构是设计抽

象的进一步发展，满足了更好地理解软件系统，更方便地开发更大规模、更复杂的软件系统的需要。

事实上，软件总是有体系结构的，不存在没有体系结构的软件。体系结构（Architecture）一词在英文里就是“建筑”的意思。把软件比作一座楼房，从整体上讲，是因为它有基础、主体和装饰，即操作系统之上的基础设施软件、实现计算逻辑的主体应用程序、方便使用的用户界面程序。从细节上来看，每一个程序也是有结构的。早期的结构化程序就是以语句组成模块，模块的聚集和嵌套形成层层调用的程序结构，也就是体系结构。结构化程序的程序（表达）结构和（计算的）逻辑结构的一致性，及自顶向下开发方法自然而然地形成了体系结构。由于结构化程序设计时代程序规模不大，通过强调结构化程序设计方法学，自顶向下、逐步求精，并注意模块的耦合性就可以得到相对良好的结构，所以，并未特别研究软件体系结构。

可以打个简单的比喻，结构化程序设计时代是以砖、瓦、灰、沙、石、预制梁、柱、屋面板盖平房和小楼，而面向对象的程序设计时代以整面墙、整间房、一层楼梯的预制件盖高楼大厦。构件怎样搭配才合理？体系结构怎样构造容易？重要构件有了更改后，如何保证整栋高楼不倒？每种应用领域需要什么构件（医院、工厂、旅馆）？有哪些实用、美观、强度和造价合理的构件骨架使建造出来的建筑（即体系结构）更能满足用户的需求？如同土木工程进入到现代建筑学一样，软件也从传统的软件工程进入到现代面向对象的软件工程，研究整个软件系统的体系结构，寻求建构最快、成本最低、质量最好的构造过程。

软件体系结构虽脱胎于软件工程，但其形成同时借鉴了计算机体系结构和网络体系结构中很多宝贵的思想和方法，近些年软件体系结构研究已完全独立于软件工程的研究，成为计算机科学的一个最新的研究方向和独立学科分支。软件体系结构研究的主要内容涉及软件体系结构描述、软件体系结构风格、软件体系结构评价和软件体系结构的形式化方法等。解决好软件的重用、质量和维护问题，是研究软件体系结构的根本目的。

## 1.5 软件体系结构与软件框架

框架是可以通过某种回调机制进行扩展的软件系统或子系统的半产品。为了提高软件开发的“起点”，以加快开发速度，提高产品质量，基于框架开发已经成为一种普遍现象。这里的框架指的是诸如 Spring、Struts、Swing、MFC 之类的框架。了解相关领域或业务领域的框架已经成为软件架构师的必修课。

框架是软件半成品，这是它和其他所有软件组件的本质区别。软件重用中存在的一个内在矛盾是：“重用概率”大小和“重用带来的价值”之间的矛盾。软件单元的粒度越大，则重用带来的价值就越大，但重用概率越小；反之，粒度小的软件单元被重用的概率越大，带来的价值就越小。框架的智慧就在于此：为了追求重用所带来的价值最大化，将容易变化的部分封装成扩展点，并辅以回调机制将它们纳入框架的控制范围之内，从而在兼顾定制开销的同时使被重用的设计成果最多。

软件框架是一种特殊的软件，由实际的代码构建而成，是软件系统、子系统的半成品。

软件框架为具体的解决方案提供了基础，提供了基础服务和可扩展点，同时软件框架也建立了一些约束，开发人员在此基础上进行特定业务功能的定制开发。例如，在 J2EE 企业级应用程序开发中，经常使用 Struts+Spring+Hibernate 来搭建一个基本的项目结构，在没有其他特殊系统需求的前提下，这就是一个软件框架。

软件体系结构是引导如何设计软件框架的重要决策。它决定了软件系统如何划分，在一定程度上描述了被划分的各个部分之间的静态、动态关系。软件体系结构的决策体现在软件系统的框架中。

下面是《面向模式的软件体系结构（第一卷）》中为框架所下的定义：

框架是一个可实例化的、部分完成的软件系统或子系统，它为一组系统或子系统定义了体系结构，并提供了构造系统的基本构造块，还为实现特定功能定义了可调整点。在面向对象环境中，框架由抽象类和具体类组成。（A framework is a partially complete software (sub-) system that is intended to be instantiated. It defines the architecture for a family of (sub-) systems and provides the basic building blocks to create them. It also defines the places where adaptations for specific functionality should be made. In an object-oriented environment a framework consists of abstract and concrete classes.）

（1）总结一下，软件框架与软件体系结构的区别如下。

① 框架是一种特殊的软件，它并不能提供完整的解决方案，而是为构建解决方案提供良好的基础。框架中的服务可以被最终应用系统直接使用，而框架中的扩展点是供开发人员定制的“可变化点”。

② 软件体系结构不是软件，而是关于软件如何设计的重要决策。涉及如何将软件系统分解成不同的部分、各部分之间的静态结构关系和动态交互关系等。

（2）软件框架与软件体系结构的关系如下。

① 为了尽早验证体系结构设计，或者出于支持产品线开发的目的，可以将关键的通用机制甚至整个体系结构以框架的方式进行实现。

② 业界可能存在大量可供重用的框架，这些框架或者已经实现了软件体系结构所需的重要体系结构机制，或者为未来系统的某个子系统提供了可扩展的半成品，所以最终的软件体系结构可以借助这些框架来扩展。

③ 框架也有体系结构。

## 1.6 软件体系结构生命周期模型

软件体系结构对于大型软件系统的成功是至关重要的。选择不合适的体系结构会导致一系列灾难性的后果，这是应该努力避免的。因此，建立软件体系结构生命周期这一概念，基于它来建立形式化推理系统与相关原则。在一个生命周期中，软件体系结构包含创建、演化、解析等过程。

软件体系结构生命周期模型用于描述软件体系结构会经历的所有阶段。该描述独立于某个项目的特定体系结构，用于指导软件体系结构遵从形式化理论基础与工程原则。

软件体系结构生命周期模型由以下几个阶段构成，如图 1-5 所示。

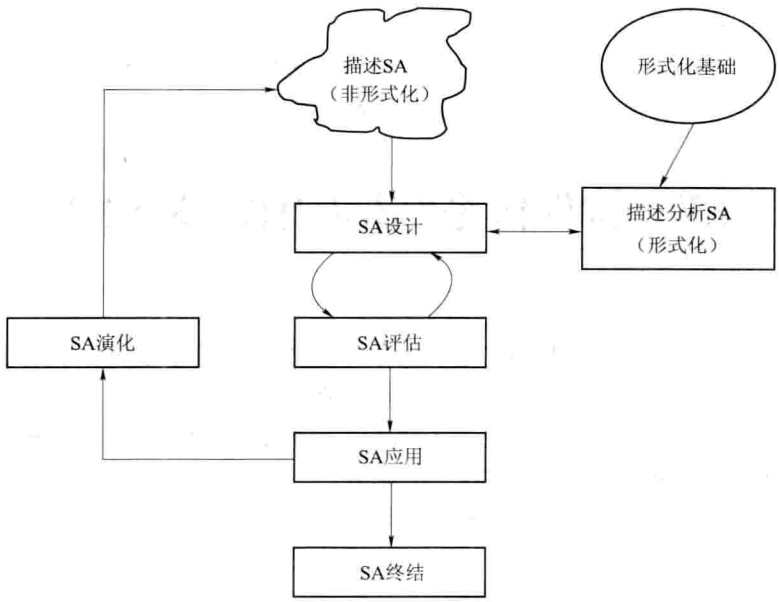


图 1-5 软件体系结构生命周期

1）非形式化描述软件体系结构

一种软件体系结构在其产生时，其思想通常是简单的，并常常由软件设计师用非形式化的自然语言表示概念、原则。例如，客户机—服务器体系结构就是为适应分布式系统的要求，从主从式演变而来的一种软件体系结构。尽管该阶段的描述常是用自然语言描述的，但是该阶段的工作却是创造性和开拓性的。

2）形式化描述与分析软件体系结构

这一阶段通过运用合适的形式化数学理论模型，对第 1 阶段的体系结构的非形式化描述进行规范定义，从而得到软件体系结构的形式化规范描述，以使软件体系结构的描述精确、无歧义，并进而分析软件体系结构的性质，如无死锁性、安全性、活性等。分析软件体系结构的性质有利于在系统设计时选择合适的软件体系结构，从而对软件体系结构的选择起指导作用，避免盲目选择。

3）软件体系结构评估

虽然形式化分析功能强大，但还不能处理体系结构的所有问题。在实际过程中，评估是一个关键性步骤，它由项目风险承担者参与并指出结构中不满意的地方，使用额外的模型来检查必需的质量属性是否与之前声明的条件一致。该阶段用于决定当前结构是否投入应用。

4）软件体系结构应用

在该阶段中，改良的软件体系结构会被应用到系统的设计中去，初始框架组织体系结构元素也正是基于该体系结构。

5）软件体系结构演化

需求、技术、环境与部署的修改可能导致体系结构的修改，称作“软件体系结构演化”。体系结构会被设计并且验证保证其在新情况下的适应性。

6）软件体系结构终止

如果软件体系结构在一系列改进和修改之后变得难以理解，且不满足蓝图的需要，则应该放弃它。该体系结构的生命将终止，而新的体系结构浮出水面。

## 1.7 软件体系结构的作用和意义

（1）体系结构是风险承担者进行交流的手段。

软件体系结构代表了系统公共的高层次抽象。这样，系统的大部分有关人员能把它作为建立一个互相理解的基础，形成统一认识，互相交流。

体系结构提供了一种共同语言来表达各种关注和协商，进而对大型复杂系统能进行理智的管理。这对项目最终的质量和使用有极大的影响。

另外，需求分析到软件实现天然存在一条鸿沟，体系结构的设计正好可以在它们之间架起一座桥梁。

（2）体系结构是早期设计决策的体现。

软件体系结构体现了早期的一组设计决策，这些决策比后续的开发、设计、编码和维护工作重要得多，对系统生命周期的影响也大得多。早期决策的正确性最难以保证，而且这些决策也难以改变，影响也最大。

软件体系结构的决策明确了对系统实现的约束条件。进而可根据系统划分确定任务划分，从而可决定开发和维护组织的结构。另外，系统的质量在很大程度上受制于体系结构的设计，相应地，也可以通过体系结构预测软件的质量。

软件体系结构的设计有助于原型的设计和开发，一旦有了体系结构的设计，就可以构建原型，并不断地循序渐进。

在对新项目组成员介绍开发的系统时，可以通过介绍体系结构使项目成员迅速进入角色。

（3）软件体系结构是可传递和可复用的模型。

软件体系结构级的复用，意味着体系结构的决策能在具有相似需求的多个系统中发生影响，这比代码级的复用有更大的好处。

软件体系结构使得能够组合大量支撑产品和服务。在一条产品线上共享软件体系结构将带给开发过程一套核心的知识和资产集，还可以显著减少开发和维护代码的成本，使生产、文档制作、培训和市场推广等工作有序化。

## 1.8 软件体系结构的现状及发展方向

自 20 世纪 90 年代后期以来，软件体系结构的研究成为一个热点。广大软件工作者已经认识到软件体系结构研究的重大意义及其对软件系统设计开发的重要性，开展了很多研究和实践工作。目前，软件体系结构尚处在迅速发展之中，越来越多的研究人员正在把注意力投向软件体系结构的研究。