软 件 工 程 技 术 丛 书

Rational软件公司
首席科学家Grady Booch做序

软件架构师职业导读

The Software Architect's Profession An Introduction

(美) Marc T. Sewell
Laura M. Sewell 著
蒋 慧 等译

PH PTR
机械工业出版社
China Machine Press

The Software Architect's Profession
An Introduction

软件架构师职业导读

架构无论是用砖头和钢铁还是用计算机代码建造的，整个过程总是从设计师和客户开始。他们一起达成共识——一个规划，设计师用这个规划来指导整个投标阶段、构造阶段和实现阶段。帕台农神庙和纽约帝国大厦都是按照架构设计而建筑的。但是，长期以来软件业却一直在没有架构的情况下建造信息摩天大厦。

成功的基于软件的技术总是首先设计，然后才构建。谁进行设计呢？软件架构师。现实世界对软件架构师有着巨大的需求——但目前还没有建立真正的软件架构职业。许多职业软件人都采用了“软件架构师”这一庄严称号，但很少有人真正完成这一经典角色所负的职责。Marc T. Sewell（世界软件架构师协会的主席）和Laura M. Sewell一起，将软件架构与传统建筑业相比较，对架构的本质和如何定义软件架构师，以及这一职业该如何走向成熟进行了深入研究。

本书内容

- 弥补客户与技术人员之间的隔阂
- 区分软件开发业内的不同职业，定义软件工程师和软件架构师的角色和责任
- 讨论标识软件架构师的职业特点和学术倾向
- 概述架构的各个阶段
- 阐述在理解和确认软件构造与设计的过程中客户的关键作用

无论你是首席信息官（CIO）、首席执行官（CEO）、IT管理人员，还是专业软件人员、学生，你都生存在软件架构中，你的整个世界都深刻地受到架构设计的影响。本书提供了一幅简单的认知图，它将改变你对软件架构、构造以及与我们日常工作和生活密切相关的信息结构的看法。

本书没有任何专业术语。它是一本澄清概念的书，能让任何人理解软件制造过程，它首次为这个飞速发展的行业中许多概念做了精确的定义。

封面设计 陈子平

ISBN 7-111-11498-1

9 787111 114987

HZ BOOKS

华章图书

网上购书：www.china-pub.com

北京市西城区百万庄南街1号 100037

读者服务热线：(010)68995259，68995264

读者服务信箱：hzedu@hzbook.com

ISBN 7-111-11498-1/TP · 2756

定价：19.00 元

软件架构师职业导读

The Software Architect's Profession

An Introduction

(美) Marc T. Sewell
Laura M. Sewell 著

蒋 慧 等译

机械工业出版社
China Machine Press

本书将软件制造业与传统建筑业相比较，对软件架构的本质、软件架构师的定义，以及这一职业该如何走向成熟进行了深入的研究。其主要内容包括软件工程师和软件“建筑者”的角色和责任，架构师的职业特点，设计软件架构的各个阶段等。是软件架构师必读的一本好书。

本书版权登记号：图字：01-2003-0287

图书在版编目 (CIP) 数据

软件架构师职业导读 / 休厄尔 (Sewell, M. T.), 休厄尔 (Sewell, L. M.), 著；蒋慧等译. - 北京：机械工业出版社，2003

(软件工程技术丛书)

书名原文：The Software Architect's Profession, An Introduction

ISBN 7-111-11498-1

Ⅰ. 软… Ⅱ. ① 休… ② 休… ③ 蒋… Ⅲ. 软件设计 Ⅳ. TP311.5

中国版本图书馆CIP数据核字 (2003) 第014683号

机械工业出版社 (北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑：张金梅 王高翔

北京昌平奔腾印刷厂印刷 · 新华书店北京发行所发行

2003年4月第1版第1次印刷

787mm × 1092mm 1/16 · 9.75印张

印数：0 001- 4 000 册

定价：19.00 元

序

安德利亚·帕拉迪奥(Andrea Palladio)是历史上最具影响力的建筑设计师。他于1570年在威尼斯出版的奠基性作品——*The Four Books of Architecture*（建筑学四本书），对文艺复兴时期的建筑产生了迅疾而深远的影响。帕拉迪奥的天才在于他具有把罗马建筑风格的精华萃取出来的能力，并在此基础上为整个建筑领域建立起一套广泛的术语和概念，明白而清晰地给出一系列实用的规则，帮助建筑设计师们创造出具有实用而美观结构（structure）的建筑物。正如帕拉迪奥所说，“有三点……是每座建筑都需要考虑的，没有它们，任何大型建筑都不值得称赞——这就是实用（或方便）、强度、美观。”帕拉迪奥醉心于他的职业，并通过他的著作与人们分享他对这一职业的热忱与理解。

Marc和Laura同样对软件架构师（software architect）这一新兴职业充满热忱。在本书中，我们将分享他们在这一行业的资深经历。从伊姆贺特普到李·柯贝伊，从贝聿铭到克里斯多佛·亚历山大，作者把软件架构师与建筑设计师的历史相比较，对软件架构师的角色、他不能做什么，以及能做什么都给出了清晰的陈述。

软件是最终的建筑材料，具有无限的适应性和不可损毁性。经济的发展引导我们不断建筑更复杂的系统，而由于人类在处理复杂性事物时所具有的与生俱来的局限性，所以常常进行抽象。因而，软件工程的历史就是抽象的历史。在向更高层次抽象的过程中，面向对象取代了功能分解。然后，出现设计模式，以便为开发团队提供充足的术语来描述一个由社会对象组成的机制。架构的框架比设计模式更广泛，它表达设计中所有那些为塑造整个系统的结构和行

为而做出的公共并且重要的决策。从而，是否具有很强的架构视图则成为预测复杂系统成功与否的一个关键。因此，软件架构师的重要性在不断加强。这一行业还处于成长的阶段，但在本书中，Marc和Laura却把它当成一个真正的职业、一个能从中学习到知识，并且值得从事的职业。

如果你已经是一名软件架构师，本书将从更广泛的角度帮助你更高效地完成你的角色职责。如果你正希望成为一名软件架构师，那么本书通过从职业的角度到深入的技术活动，对架构意义、架构设计过程，以及这一职业的功能进行解释，帮助你完成这一转变。

作为职业软件人，我们都寻求使用一种有效而经济的过程来建造一个能够工作的有用的产品。所以软件架构师将具有很大的需求，以及难以置信的回报。软件架构师在他的全盛时期将构造出优美的系统。

祝愿你也能构建出优美的系统。

Grady Booch

Rational软件公司首席科学家

前　言

当某种转变悄然发生时，如果正好处在转变的正确方向上将非常令人振奋，而如果处于转变的错误方向上则令人沮丧，因为这样会觉得一切都不顺。有人会顺应改变，而有的人则不。为了适应转变，人们需要改变观察问题的方法，改变被心理学家称之为的心理定向（mental set）。这就是本书的目的所在：为了改变人们观察软件设计和构造的方法，向读者提供一种全新的认知指南。

本书不是一本技术类书籍。本书所要说明的是：在构建一幢建筑物和构建一个软件系统之间存在着惊人的相似性，而这个类比就是帮助我们完成转变的工具。如果我们深入研究那些熟悉的软件设计及构造的名词术语和实践，那么将会把具有技术背景的读者拖进出他们自己的经验建立起来的习惯的泥潭中——这反而会阻碍转变。

相反，我们将更多地谈及架构和构造的建立，然后把主题引回到软件业，从而展示这一类比所呈现的明白无误的道理。希望读者能从一个崭新的角度（而不是从他们自己软件构建经历的角度）来真正了解架构和构造，了解它们的历史、角色和过程。认识架构和构造的经典形式及其产生的独特样板，与所熟悉的软件构建背景结合起来，从而促进转变的发生。这样就能以可预测的和可靠的方式构建软件。

这一类比是完成这一转变的工具，不要被它那看似简单的外表所愚弄。简单不等于肤浅，简单的背后也有深奥的道理。建筑物是很复杂的，它的构造是困难的，但所有人都了解建筑物的构造过程以及参与者的角色分工。软件构造缺乏的就是角色和过程的明确定

义，因此给软件业角色进行明确定义正是改变视角的关键。

本书是为广大的技术和非技术人员写的。对软件项目的客户、软件业内人士、学生，以及所有对软件系统感兴趣的人都会有所帮助。客户尤其重要，因为是他们而不是学术界或职业软件人在推动着这一转变。

在20世纪90年代，客户和雇主开始使用架构方法进行软件构建。是他们把架构师（architect）这一头衔授予职业软件人，进而描述这一工作，并成立了架构部门。

但即使是这些客户和雇主也没有同时认识到架构师的明确定义。他们发明了CTO头衔，这种人负责技术、企业架构和软件战略等方面的工作。他们把这个头衔的名称搞错了，实际上，CTO就是担任了首席架构师的角色。

客户对这个类比有一个自然、直观的理解。他们从心理上能理解和管理软件构建。简而言之，他们认为软件构造就是先进行设计，然后在该设计的要求下仔细地构建。

我们希望这本书至少能使读者在这一工具的帮助下，从一个全新的角度深入地理解架构。

Marc Sewell和Laura Sewell

marcandlaura@wwisa.org

architect / ˈɑːkɪtɛkt / *n.* & *v.* M16. [Fr. *architecte* f. It. *architetto,* or their source L *architectus* f. Gk *arkhitekton,* f. *arkhi-* **ARCHI-** + *tekton* builder.] **A** *n.* **1** A designer of buildings, who prepares plans, and superintends construction. M16. **b** In full *naval architect.* A designer of ships etc. M19. **2** A designer of any complex structure. L16. **3** A person who plans, devises, or contrives the achievement of a desired result. L16.

THE NEW SHORTER OXFORD ENGLISH DICTIONARY
ON HISTORICAL PRINCIPLES

EDITED BY
LESLEY BROWN

CLARENDON PRESS - OXFORD

1993

c In full software architect. A designer of software based technology, who prepares plans, and superintends construction

（全职的软件架构师是基于软件技术的设计人员，负责准备规划和监督构造）

目　录

第1章　一个简单的类比

“所有巧妙的想法都已经被人们想到了；惟一需要做的事情就是把它们再重新思考一遍。”

——歌德

具有讽刺意味的是，那些简单而易于掌握的思想却具有促进改变思想和认识的力量。纵观人类的历史，我们看到，简单的概念引发了世界观的剧变。弗洛伊德断言，潜意识的存在是行为的动机。哥白尼1530年的“天体运行论”证明太阳是宇宙的中心，而一个掉到牛顿头上的苹果成为把他引向万有引力定律的催化剂。

这些都是非常著名的例子，但是简单的经历也会在我们的日常生活中发生。有多少次我们找不到问题的答案直到我们突然看到并意识到那些从一开始就摆在我们鼻尖下的事情？有时，最明显的事情反而像梦境一样模糊。一次又一次，人们都被愚弄了，直到某个人大喊：“皇帝没穿衣服！”

一个范例是一个思想的模型，是对看似熟悉的事物的一种不同的认识方式。建立一种新的范例通常会需要对最基本的、简单的假设重新审视和提出挑战。这并不是说需要爬到已有思想的叶子上或分枝上，而是深入到表层的下面，找到最关键的根——那棵树的根，或者是了解那棵树的最正确的方式。

1.1　完美的类比

在建筑业构造和软件业构造之间存在着一个完美而深刻的类

比。这个类比很简单，但我们从前却一直未能注意到。软件业从一开始就一直饱受可靠性、可预见性、成本和可用性方面危机的困扰。当然这一行业不断地在进步，但引用英国博物学家David Attenborough的话，却仍然是：

“他们经过一系列的蜕变，但始终没有彻底地转化。”[2]

如果在所有的层次都应用这一类比，就能促使软件业发生转变，从令人失望和沮丧的现状中摆脱出来。这一类比并不是新鲜事物，但却从来没有被深入地研究和探讨过。人们经常用类比在不同事物之间建立联系，但又往往因为它过于简单而放弃它。事实上，这一类比对所有实际的情形都适用，无论是简单还是复杂的情形。无论结构是用砖头和钢铁还是计算机代码建造的，其中的角色和过程都是类似的。

软件构造和建筑构造之间的类比远不止是一个学术性的习题——它是一个不可或缺的工具，一个模板。它是一个混乱产业中的试金石，为客户、职业软件人、管理人员以及我们信息系统的“居民们”提供透明度和可预见性。有了这个类比，即使是没有技术背景的人们也能理解和确认软件的构造过程。

1.2　认知图

当我们到一个陌生城市时，如果没有地图或方向的话，很难四处行走。但不久我们就会形成自己的认知图（Cognitive Map），能开车到任何地方而无需有意识地去思考了。

同样，通过体验和观察，我们能从直觉上了解建筑是如何构造的——无论是最简单的房子还是巨大的钢化玻璃建筑。甚至是儿童对这一过程都有自己的认识，知道建筑师、科学家、工程师和建筑工人之间角色的不同。有了这一类比，人们对软件的构造就有了相应的意向。任何人都将知道整个过程是如何开始，如何进展，什么时候能“入住”，等等诸如此类的问题。人们知道所有相关的角色，

并将看到，和物理意义的建筑一样，即使软件已交付使用，也可以被继续修改、翻新。

我们知道建筑不仅是居住的场所，还方便和促进我们进行各种活动。厨房用于烹饪，卧室用于休息，办公室用于工作，阳台用于夏日小憩，博物馆用于存放宝贵的文物，洗衣房可以清洗和烘干衣服等等。建筑可以非常实用化，非常没有个性化，比如就像一块超大的石头一样的高中学校，也可以是一种具有无法言表的美感的结构。建筑可以表达文化，甚至可能带有某种幽默感，比如街边竖立的热狗雕塑。

有了这个类比，人们相应就能理解，软件系统是我们大脑的延续，它为我们进行各种活动提供“场所”。它提供了通信的场所，或者只是简单的闲聊之所。它提供了图书馆、商店、艺术设计室，以及进行审计和管理的地方。这些地方可以是非常实用而没有个性的，也可以具有无以言表的美感。它可以像一个小茅屋一样简陋，也可以像拥有精致细节的古西班牙宫殿一样复杂。有了这个认知图，人们可以认识到信息系统完全类似于物理建筑，并可采用类似的思维方式来理解信息系统。

在认知图的指引下，这一类比就能说明我们是如何构造软件的。把建筑构造的模板覆盖到我们的软件构造经验之上，就开始了一个深刻的变革。

1.3 类比揭示出被遗漏的要素——架构

有了这个类比，软件构造中被遗漏的知识领域立刻就显现出来，这一知识领域组成人类历史上最古老的职业。但在软件构造中，架构和架构师的角色大都被忽略了，需要加以强调和确定。软件业已经深受没有真正的架构设计之苦了。

现在到了发生变革的时候了，阿尔文·托夫勒把历史写成一系

3

列变革的浪潮。第一次浪潮来自农业的出现，人类结束游牧生活方式，得以形成稳定的村落，并最终繁荣发展成为现代社会。第二次浪潮起源于工业革命。机器代替肉体的人和牲畜，金钱代替土地成为衡量财富的标准。现在我们处于第二次浪潮和第三次浪潮之间的斗争中。在第三次浪潮——信息时代中，知识就是力量。

值得注意的是，正式的软件架构师职业出现在第三次浪潮的顶峰时期，与正式的建筑设计师职业出现于第二次浪潮顶峰时期完全类似。从一种深刻的角度来说，这是一种完美的逻辑。从农业社会到工业社会，需要有新的复杂的物理结构，让人们可以从事和进行更多的由变革所产生的活动，此时，有自己一整套名词术语和标准的职业设计师就应运而生，可靠地满足了这一需求。

第三次浪潮的变革也需要新的复杂软件结构，使人们能够从事和进行由工业时代到信息时代转变所必需的活动。需要有职业的软件架构师以可靠的方式设计这些软件结构，大量的新问题需要解决，需要设计出新的结构类型。我们不再盲目地在没有职业设计师和设计蓝图的情况下构建复杂的软件结构了。

1.4　类比使角色和意图之间的关系明晰

当IBM的CEO想建一个新的总部时，他会找谁？一个“建筑科学家”？一个“建筑工程师”？还是一位“精神领袖”？当然不是。他会找贝聿铭——一位设计师。如果教皇从来没有雇过米开朗基罗的话，将会发生什么？还会有圣彼得教堂吗？实际上，教堂的设计和建造都是极端困难的，最后有太多的设计师都经手了圣彼得教堂，如果没有建筑设计师、没有建筑规划的话，很难想像还会有这座如此瑰丽的教堂吗？

遗憾的是，业界的黄页上还没有软件架构师的清单，没有关于软件架构师，以及客户和软件构造人员的角色的明确定义。因为设计角色几乎被遗忘，从而产生了一个真空。这个真空被程序员、工

程师和计算机科学家所填补，他们没有设计的训练和经验，也没有必要具有架构的观点。尽管如此，他们却把设计看成是自己的份内之事，从而造成角色模糊、责任混乱的结果。 4

Bass、Clements和Kazman合著的*Software Architecture in Practice*（《实用软件架构》）通过一张题为“架构师”的图表来说明，这本书表面上是写给软件架构师关于软件架构内容的。但在前言中，作者提到：

> “我们毕竟是软件工程师。”[3]

在这本书中，在整个软件业界，职业软件人等同地自称架构师或工程师是非常普遍的。但在这一类比下，头衔、技能和角色最终将成为关注的焦点。我们不仅使客户被这些歧义所迷惑，也使我们自己迷惑。

想像一下，如果你在没有真正的建筑设计师、没有设计蓝图，甚至工程师和建筑工人连明确的角色分工都没有的情况下，构建自己梦中的家。换言之，想像一下自己采用一般软件构建的过程来构建自己梦中的家。你可能在一开始时就雇一位建筑顾问或项目经理，然后，他们会带来一批人完成需求收集的过程。他们将使用一种昂贵的需求管理包，你会告诉他们你所有关于家的要求，包括会花多少钱、要建在哪。你的要求会非常具体，因为自己已经仔细研究过了：

伍尔坎餐厅式壁炉

核桃木色的木地板

在大客厅装上高的加保险的天花板

主卧室是茶色大理石

外面是红砖

起居室大小18×24英尺

阳台是石板地

四个加热和空调带

一种柔和的艺术外观和感觉

这张清单可以一直写下去，包括墙纸和地毯的颜色，窗和门框的牌子等。

项目经理会雇一队建筑专家和建筑工程师，跟着是开会，完成白板，把工作被分派给各个小组，直到所有需求都有人负责完成。可能卫生间、厨房和卧室需要首先完工，让客户能省点钱早点搬进来，同时也向客户展示房子其他部分的观感。

5

建筑工人的角色将根据他们的技能、经验和所携带的工具箱来决定。参与布线的人可能还要安装车库门，铺瓷砖的人如果有类似经验的话还得设计厨房，用最新的“铜管结构”做管道的人还必须负责房顶，因为那里是管道的通风口。毕竟，他们都是“建筑工程师”。

这个规划被加入到分阶段完成的房间，专门设计，直到完成所有的房间，有一个建筑专家负责各个房间相互交接的地方。

任何人都能看出这是一个产生混乱的方法，虽然能造好一个房子，但会是什么样的房子呢？如果它没有完成预定功能，将由谁来负责？毕竟，已经完成了所有需求，房子具有客户要求的所有条目，但它能否正常工作？至于艺术的外观和感觉，在危险的时候谁还能想到艺术？

1.5 一切从客户和建筑设计师开始

有了这个类比，显然，所有这些离散需求即使堆起来有5英尺高，也无法与一个协调的房屋整体设计草图等同。这种整体设计的过程都从客户和建筑设计师开始。在建筑的构造中，客户不仅仅是一个风险承担人，他起着积极的关键的作用。想像一下米开朗基罗把教皇作为一位风险承担人或一个最终用户，客户拥有整个结构（无论是实质性地还是象征性），他要和建筑设计师一起对最终要完

成的目标达成共识，并用草图记录下来。然后客户将不断确认整个构造过程。

建筑设计师和建筑规划形成连结客户或用户与建筑技术人员之间的桥梁。建筑设计师通常都是客户的代言人，在整个项目过程中首先要代表客户，其次才是介于两者之间。设计师立足于两个不同的世界——客户的世界和建筑技术人员的世界。他理解这两 6
个世界，从而在艰难的构造阶段保持设计的完好无缺。

例如，如果没有建筑设计师或代言人，木匠可能会对客户说就目前的空间和角度，没法做他想要的弧形楼梯。客户只能想到会有以下几种可能：

- 也许这是真的。
- 也许木匠意识到他缺乏相应的技巧或工具来做一个弧形楼梯，但他不愿意承认这一点。
- 也许木匠认为不行，但实际却行。
- 也许木匠对这项工作的提的价钱有点低，他觉得对目前的工钱来说弧形楼梯有点贵了。
- 也许有一份钱更多的工作在等着他，而弧形楼梯太费时。

木匠可能会用许多技术名词甚至几何理论来支持他的观点，但是，除非客户也是一位有经验的木匠，否则他无法知道木匠说的是否属实。客户解决这些“也许”的惟一办法就是再找一个木匠或一位建筑顾问进行确认。但这两种方法的费用、工程的延期以及所带来的后果代价都是昂贵的。

现在，想像同一情景，只是项目中有一位训练有素的职业建筑设计师。那么，木匠不可能对一个简单的楼梯提出疑问，除非真的不可行或者超出他的能力之外。不过，首先建筑设计师也不可能设

计出一个无法建造的楼梯。建筑设计师知道到底是木匠缺乏技能还是真的有什么问题。无论哪种情况，都将由建筑设计师来决定，无需把客户卷入其中。

但是，在软件领域，一切都由软件开发人员说了算，客户很少能够对设计、过程、成本和产出加以确认。前景？技术战略用途？设计的统一与优雅？美观的图形？客户惟一能做的就是希望软件的结构稳固，不要把资源耗尽。不过一旦接受前面提到的类比，那么架构师的经典角色就会根植于软件业界人士的大脑中，接着就是明晰角色和职责，再扩展到把前景变成现实的开发人员。

7

1.6　通过类比，每个词汇都有意义

一旦接受这一类比，软件业也将有一个有意义的词典，映射出建筑构造所使用的标准词汇。工程师（engineer）就是工程师，不是架构师（architect），程序员只负责写代码，这是软件构造的组成模块，而不是创建架构。我们通常只在使用常规名词来描述完全不同事物时才会产生迷惑。

有了这一类比，很明显架构（architecture）一词与设计（design）是同义词。同时，架构还指的是架构及其实用的领域。蓝图（blueprint）不是架构，它不是一个工程元素，也不是一组技术元素。在软件业中，架构一词被人们滥用，使人迷惑。这一类比使人们理解这些构造元素词汇的意义，就像建筑业一样。他们使用事物的真实名词，如托梁、斜削接头、电路箱、管道、子墙、循环器、接地电源插头等等。这些都不是架构。

在软件业，词汇是人们争执和讨论的主题，这些讨论目前非常热，因为所讨论的词汇往往没有明确的、可预测的过程和产出。

下面是一个真实的故事：

Meg是一个大的农产品生产公司的经理，最近他被委派去监

督一个新的销售、仓储、运输和付账软件系统的开发，这项工作令他高兴。他雇了一个大的顾问公司来开发这个系统，他们又雇了一些转包商（子承包商）。

作为公司的项目经理，Meg与顾问公司一位有着许多软件开发项目经验的资深技术项目经理并肩作战。这位项目经理使用一个项目管理工具开发了一个复杂的项目计划并制定了精确的时间进度表。尽管如此，支出却令人担心地不断上涨。

Meg以为自己没有足够的技术背景理解问题所在。需求文档非常细致，项目计划已经极尽详尽，但她却无法控制。

她决定和项目的两位最高层技术人员举行一个会议，这两个人在业界都非常有名，他们能够告诉她问题出在哪里。但是，这两位技术专家非但没有给她任何答案，反而针对分析层设计和底层设计这个两个名词的意义热烈地争论了将近1个小时。一个主张这两者之间的区别很小，同一层次的设计应该存在于同一文档中。而另一个则固执地坚持这两者之间有着完全不同的含义，需要放到两个文档中。

这一激烈的讨论在Meg心中产生巨大的恐慌，因为她一直都以为“收容所是由居住在里面的人运作的。”她奇怪的是，当程序员都已经开始写代码了，为什么这两个人还在争论设计文档的问题。那他们在构建什么呢?

技术负责人们显然是聪明的，她也很聪明，那为什么会出现这种情况？她完全熟悉整个业务，对所需要的产品也有清晰的心理想像。她付给这两个人每人每天2 000美金，他们非但无法达成一个一致的设计，甚至连设计到底是什么也不能形成一致的看法，而整个项目却已经开始3个多月了！

Meg离开了会议室，她想尽量保持自己的镇静。

在这个没有任何结果的情景中，关于词汇的争论贯穿整个过程。如果过程像建筑构造中的过程一样可预见，那么文档到底叫什么其实并不重要。应该有一个蓝图描绘对设计的共同理解，至于蓝图的

名称则无关紧要：对于搞建筑的人来说，房子到底是否具有西南部的风格、架构和设计并不重要，过程和角色是一样的，因为房子自己会说话。

1.7　通过类比，过程变得可以预见

“首先设计，然后小心地构建”，这听起来像在夸大一件显而易见的事物，但应用于软件构造就远不是这么回事了。建筑构造中设计和构造过程有一个逻辑顺序，软件业应该把这一顺序作为参考的模板来使用：

整个过程的开始是客户和建筑设计师关于架构规划达成共识，一旦设计阶段完成，客户将能够通过蓝图、3D模型等形式看到建筑及其整个组织。然后，所有在构造过程中担当角色的人，包括工程师、督察员、建筑工人、内部设计人员、管道工、电工和木匠等，他们的意图各自不同，但都将使用同一架构规划。

建筑设计师的规划或蓝图是高层设计，但在构造的整个过程中，建筑工人也进行设计。例如，架构规划可以表明电插座和照明设备放置的位置，电话线路的数量，甚至包括整个电路必须没有接合的需求等。规划甚至可以细到牌子、风格，乃至照明设备、插座和开关的零件号码等。

然后，主电工将设计线路盒和总体布线方案。事实上，所有设计师没有特别提到的细节都由电工设计的，包括电线的类型、电路断电器等等。这些都是低层设计，并且不仅仅限于此。

然后电工的助手要设计电线穿过椽子以及在墙内布线的方式。就像主电工必须负责所有设计师没有特别提到的细节一样，电工的助手也必须设计所有主电工没有特别强调的细节。每一层设计都由比它更高一层的设计人员监督。

建筑构造的这个有序过程是根深蒂固的，无需再做出说明，它是行业的标准，所有参与的人都知道应该这样。这不会影响整个过程的灵活性或复杂性，因为它实际上使建筑工人不必被过程、角色和职责所困扰，他们反而能灵活发挥。

科学家和工程师的角色同样清晰。科学家进行研究，使建筑材料、工具和方法继续发展，工程师和设计师以及建筑工人一起确保按照设计建造结构，满足客户的需求。

1.8 类比使复杂性和灵活性有了秩序

软件失败和效率低下的根本原因不能归咎于软件技术的复杂本质，而是因为我们人类不能像构造建筑物那样构造软件系统。这一类比在软件业中只限于用来说明观点或提出问题，却从来没有被更深入地使用过。人们认为它过于简单，但复杂往往使人产生混乱。

批评家迅速地拒绝这一类比，因为当人们想起建筑时，便会觉得它过于复杂。关于建筑的方法、工具和材料有太多的科学研究。一片简单的胶合板是多年来对薄板、木头、胶和制造方法的不断实验和提高的结果，电工有自己的行话和名词术语，管道工把复杂的铜管管道看成一件艺术品，事实上也的确如此。生活在一个房子里，欣赏它并且保持它的整洁都不需要特殊的训练，然而对于最小结构所包含的系统，其背后都有大量研究和开发的支持。摩天大厦则又是另外一件完全不同的事情了。

如果说建筑技术很简单是有悖逻辑的，人类的工程和设计的辉煌成果横跨多个世纪，从金字塔到大教堂到西尔斯大厦。建筑的构造看似简单只是因为有一个系统而有序的过程、清楚的角色、清晰的规划来管理复杂性。

同样，不要假定这一类比限制了灵活性，因为可以从建筑构造中继承很大的灵活性。很多情况下都无需雇用建筑设计师。例如，

当结构非常简单，或者建筑工人决定采用杂志上的蓝图时，一个有经验的建筑工人也许能在只有蓝图情况下建车库和辅助的房间设施，这些往往取决于建筑工人的技能和经验，以及当时的任务。但是，即使是在这些情况下，设计仍是第一步，在客户确认后，承包商进行投标，最后才是破土动工。这时，建筑工人的角色和技能才是可预见的。

建筑和软件随着时间的推移都会有所改进和有所提高，同时也存在着灵活性。可能客户开始时只能支付基本的房屋规划，把一些大的元素如车库、客房、游泳池等留到以后再做。但至关重要的是，必须在架构设计中预先考虑这些架构元素。

建筑设计师有多种方式来规划这些元素，每个元素在场地规划中有适当的空间；必须知道和符合建筑编码和变化；必须对楼层加以规划，确保将来能包容这些元素；还必须考虑一些设施的安装。如果没有这些步骤，那么将来就无法增加车库、游泳池或其他任何设施，甚至需要把已有结构进行部分或全部重新设计。

软件业与这种有组织的方法正好相反。你是否遇到过设计与构造并行进行的项目？是否遇到过真正能描绘信息系统项目广度、组织和行为的高层设计根本不存在，或不完全，或没有得到遵守的情况？是否遇到过客户对最终提交给他们的结果根本不理解甚至惊讶的情况？是否遇到过在整个规划和最初的构造中根本就不考虑未来对灵活性和扩展的需要的情况？

11

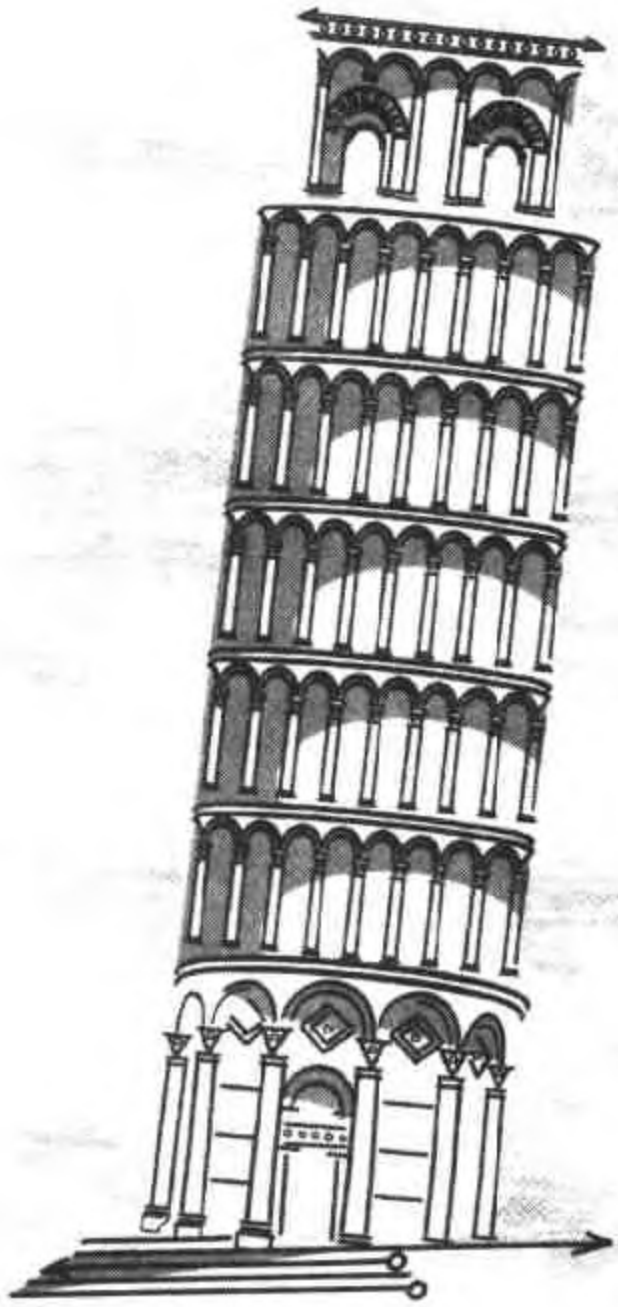
1.9　结论

这一类比的自然力量不仅会引导人们建立一种关于软件架构的新职业，它还具有深远的连锁反应。它通过明晰职业软件人的角色和职责，引入可预见的有序过程，使名词术语标准化，从而转变软件构造的方式。更令人惊讶的是，这一类比还将使客户和软件系统

的使用者们找到他们作为信息技术系统的所有者和管理者的正确位置，就像居住的人必然要监督他们自己房子的构造和对其进行维护一样。

尾注

1. Johann Wolfgang von Goethe, *Proverbs in Prose*, translated by Norbert Guterman, in *Familiar Quotations*, 16th ed , ed Justin Kaplan (Boston: Little, Brown & Co., 1992), 351.
2. David Attenborough, in *The New Shorter Oxford English Dictionary*, Thumb Index ed. (Oxford: Oxford University Press, 1993), 3369.
3. Len Bass, Paul Clements, and Rick Kazman, *Software Architecture in Practice* (Reading, Mass.: Addison-Wesley, 1998), xi



第2章　没有软件架构师的世界

"软件缺乏可用性以及糟糕的程序设计是软件业不愿公开的羞耻。"[1]

——Mitchell Kapor, 1990

如果把软件和信息技术系统都想像成物理建筑，建筑的大小与它们的成本相对应，那么"软件危机"就是最应被人们关注的事情。显而易见，那将是一个到处都是空房和废弃的没有完成的建筑的情景，其中甚至还有摩天大楼，它们毫无用处地被遗弃在路边。当然也有投入使用的建筑，但其中却有空房间、死角和不开心的住户。

一边面对这种充斥着毫无用处的建筑和极端昂贵的失败，那些要为此负责的公司和工作人员一边还在建造着类似的建筑。他们挥舞着新的工具，审慎地说着新的令人恐慌的行话，因为他们和他们的客户都抱有天真的想法，希望绝对全能的理想架构这次能走向成功。

但不幸的是，尽管有越来越多的复杂的工具和方法，软件的构造人员还在继续生产出达不到期望、甚至完全失败的产品。而那些可怜的、健忘的客户还在不断为这些工作付钱，他们还经常问自己："到底什么是正确的方法呢？"

2.1　软件业的悖论

开发出失败软件的软件人员有很重的负担。他们会因为没有做出

满足或超出客户期望的产品而否定自己，同时，还必须忍受自己所创造的成品根本没有用处而带来的痛苦。在这一点上，他们生活中存在着令人震惊的悖论：一方面，他们生存的产业处于革新的漩涡中心，从心脏起搏器到运货卡车，软件被嵌入到任何东西中。同时，软件又必须紧跟人类通信模式的技术进步和改变，对于许多行业来说，决策是孤注一掷的，要么最快要么死亡。在商业的达尔文理论中，信息系统不仅仅是跟踪和操作数据的手段，它还是关键的成功因素。

另一方面，是职业软件人员活在一个完全不同的现实世界中，超出媒体和一般大众的视线之外。他们生存在所谓的“软件危机”中，未曾出现的2000年（Y2K）危机在大众的思想意识中打上了“千年热”的烙印。然而真正的软件危机对于大众而言却是视而不见的，但决不是假想的警钟。

软件行业对“软件危机”一词实在是太熟悉不过了。它最早出现在1968年，因为解决它的办法实在难以捉摸，所以各种书籍和文章中不断提到它，有的书整本地记录了软件失败、混乱和全面否定。例如Robert Glass的*Computing Calamities*《计算灾难》、Alan Cooper的*The Inmates are Running the Asylum*《同行操纵避难所》以及Robert L. Glass的*Software Runaways*《软件大逃亡》。

软件开发的权威著作*Object Solutions: Managing the Object-Oriented Project*（《面向对象项目的解决方案》即将由机械工业出版社出版）作者Grady Booch承认：

> 我本想给这本书起名为“项目陷入危机（Projects in Crisis）”，因为看起来所有复杂度合理的有趣项目都处于危机中，无论它具有什么样的技术基础……[2]

软件行业的每个人都有自己的关于项目失败的经历，尽管似乎没人知道到底如何追究责任，但肯定存在严重的错误。Standish Group调查发现，由财富500强企业开发的一个软件项目成功并按

时完成的比例只有23.6%。在完成之前被取消的比例更高，估计是40%。剩下的要么是非常困难，要么即使成功，也超出了预算和时间。

2.2 臭名昭著的软件

统计数据有助于分析软件失败的广度和特征，但细节往往是问题所在。对软件开发失败的案例研究能够暴露细节，但由于人们一般不愿意张扬自己的弱点，所以这一点很难做到。而且由于软件是抽象的，你不可能看着它一块一块地像建筑物那样被拆卸。

相反，失败的代码和文档被安静地归入公司的系统中，公司使用性能低下的系统蹒跚前行。由于这些公司和它们的软件承包商的努力掩饰，股东和公众对这种情况依然一无所知。

幸运的是，美国政府来自于大众，代表大众，也服务于大众。因此，虽然美国公民承担了几次软件大崩溃的损失，还好，那些应该受到责备的公仆和承包商在国会的听证会上无处可藏。他们详细的证词已经让人们洞察到没有软件架构师的世界是什么模样。

政府的项目在资源方面从不节省，尽管在技术和最好的蓝筹股承包商身上花费巨资，但是项目还是失败了。

2.3 联邦航空管理局

联邦航空管理局（FAA）于1982年开始对整个航空交通控制系统进行大规模修补，项目预期大约需要11年完成。开始是一个4年的设计竞争，最终IBM取胜。目标是对整个空中交通控制系统进行现代化，以开发单个空中交通控制器工作站为中心，控制器的雷达显示器将用一个基于大型机的网络所取代，可靠性需求非常高，而且网络必须保证在99.9999%的时间里都是正常运转的，不能把系统停止来修复错误。工作人员为此项目付出了巨大的努力，但仍出现

了许多问题。

在失败的项目中，原因往往是不完整和常常改变的客户需求，但没有比这个项目更精确的需求了，据说规格说明文档大约有20英尺高。

很难准确地描述IBM和FAA在开发过程所采用的密集和严格的管理：会议协议、验收以及质量保证措施都被严格地统一管理，在开发最紧张的时候，有超过2000名IBM员工同时在开发，再加上FAA的雇员和一些子承包商也在工作。

在推迟很多次后，IBM交出了一个新的工作站，但没有实现过，因为存在许多基本问题。尽管收集了所有的需求，但是空中交通控制器本身并不是根据设计和原型开发出来的，原因在于人们认为这样会阻碍开发出真正的新技术解决方案。所以，最终的结果是工作站软件把人类对空中交通控制器的注意力从跟踪飞机发现不安全时刻这一任务中移开。

例如，控制器不仅没有把按钮变成触摸式来产生一个飞机向量线，相反却变成要输入16次按键，而且，电子生成的飞机跟踪曲线总是和控制器重叠在一起。

问题还不止这些，这些只是主要的例子。此外，项目的意图太雄心勃勃，总想一次进行多项创新，却没有采取“封装和强制”的策略来实现这些创新。

为什么会出现这种情况呢？为什么一群杰出的人员、一个名副其实的智囊团，也会犯如此低级的错误呢？

空中交通控制器联合小组抱怨缺少来自控制器的输入，情况的确如此。控制器的任务是视觉、声音、空间关系、决策和感知模式的复杂结合，没有详细地咨询各方意见就改变控制器的软件是有悖常理的。当然，软件开发人员询问了有关控制器的需求，但是显然

他们从来没有真正达成一致的理解，而且尽管有一屋子的需求文档，显而易见的事实是，无论在构造之前或构造过程中，从来没有机会对控制器的功能进行过确认。 18

1977年，总审计局出版了一个事后调查分析报告，详细披露了他们对FAA现代化的问题分析和建议，这个报告名为“空中控制系统：FAA系统现代化需要一个完整和加强的架构”（强调架构一词）。

从FAA的需求和规格说明书来看，它的最终目的是想建造一个信息摩天大楼，但是没有架构师，也没有生成一个客户可以确认和理解的架构设计。许多批评家认为这个项目“不断变化的需求”是导致失败的原因，但是需求不断变化的原因却在于没有正确地开始设计。没有架构规则，没有人具有架构的知识，这些原因使软件业和FAA（以及美国公民）遭受了重大的损失。

虽然FAA是从设计竞争开始整个项目的，这点值得赞赏。但是在四年的设计阶段中它却没有做出一个合理的样板。在设计阶段提出这么多高风险的创新，确实需要合理的样板来加以确认和验证。FAA和IBM在架构的设计阶段中不得要领，这个阶段应该由客户和“住户”（即使用者）来验证设计是否正确。他们是否满意？他们能否理解整个设计？如果不能，又该如何以他们能够理解的方式为设计建模、描绘和制定标准？

雇一组技术人员和建筑工人在纽约的东34大街建一栋120层的摩天大楼，这和上面的情景也是类似的。需求可能会像FAA的需求一样堆满一屋子，大量严格的管理控制会规范到每一分一秒的活动，但是仅仅这样就能保证最后的大楼像帝国大厦一样吗？

2.4 美国国税局税务系统现代化

据统计，IRS TSM（Internal Revenue Service Tax System

Modernization，美国国税局税务系统现代化）的失败花去了美国纳税人的33亿美元。其他的，包括德勤公司，花去了大约80亿美元。换句话说，这两种情形要花去一个中等大小的美国城市，如佐治亚州的肯尼索（Kennesaw）或新泽西州的威霍肯（Weehawken）几百年的税收。

IRS TSM是以一大堆好主意开始的，这包括它的中心意图是实现一个光扫描仪来取代手工数据输入。表面上人们很难否决这一技术的逻辑，但实际上，在投资了几百万研究扫描仪之后，他们甚至连1040EZ都无法实现。扫描仪对字符的识别精度只有95%，而且一次只能扫描纳税单的一面。

19

在处理成百万纳税申报单的时候，5%的误差将是灾难性的，即使只有一个工作单出错。同时，纳税单的背面也非常重要。在巨额投资之后，IRS的法律委员会决定扫描纳税单的背面也是完成纳税人申报单的必要手续。只是由于他们事先没有去发现这些问题，由此给整个设计增加了负担，但却没人对这一设计观点做出解释。

扫描仪技术的不合理应用只是问题的一部分。而且IRS非常紧缺的“技术资本”并非技术专家，而是这类复杂项目所需要的商业经理。组织那些拥有开发软件的技术人员的根据是（或者说甚至根本没有组织技术人员）：

> ……许多大的甚至无关的项目都被集中到一起，组成一个庞然大物，并被IRS叫作TSM。[3]

美国审计总署（GAO）和美国国家科学院（NAS）的事后分析报告提出，应该立即雇佣一位高素质的首席信息官（CIO）——当然同时还要有一个技术领导小组——但他们最先提议的是需要一个架构设计：

> 尽管已经进行了将近10年，但TSM的架构设计几乎没有形成或者说被一堆细节所淹没。IRS必须开发一个完整、精确的架

构定义，以供将近2 000名IRS开发人员和承包商使用

架构不仅仅是一组有许多文字和图片的文档，它还包括在系统被建造之前或建立样板过程中，分析系统各部分所需要的关键信息。这种分析对于大型系统最终的成功是至关重要的……

任何事物都无法取代一个充分的架构定义和系统模型。如果IRS不尽快解决这一问题，那么某些巨额投资将永远无法产生显著的效益。[4]

美国国家科学院和审计总署还认为，尽管IRS开发了许多软件规划文档，但仍然没有一个清晰的架构，而且

……没有一个文档能够精确地描述出整个需求和规格说明，从而允许一个外部技术专家以及IRS内部专家介入，来理解整个TSM的系统结构。需要有这样的文档，可作为所有设计和开发决策的基础，因此，它是TSM成功的关键。[5]

他们建议成立一个独立的首席架构师办公室，来开发和设计系统的草图，然后在继续TSM开发之前，由IRS和私营独立部门进行复查。再强调一下，与FAA以及所有成百上千脆弱的软件系统一样，开发人员没有架构规划作为开发的指导。

或者，正如John Ruskin于1880年深刻地指出：

我相信，失败更多是来自对要做的事情的误解，而很少是因为手段以及工作人员缺乏耐心。[6]

把软件的失败或使用困难归咎于“需求的不断变化”，只是对缺少真正架构的一种委婉说法。当客户和开发人员看到所开发的系统后，意识到有错误，于是在事后的工作中不断努力改变，想把现实的结构尽量向他们预想的方向靠拢。

把软件的失败归咎于管理不善也同样错误的。没有架构，软件业缺少的将是设计。即使是最好的管理人员也无法从不良或不充分

的设计中生出令人满意的结果。而且，管理也不能弥补因缺乏标准化构造过程和含糊不清的角色，以及令人迷惑的名词术语所带来的损失。只有有经验的架构师才能在客户的参与下，设计出“真正要做的事物”。MBA称之为业务流程再造（business process reengineering），我们称之为架构。

Frederick P. Brooks在他的经典著作《人月神话》的二十周年纪念版中总结道：

> 今天我更加确信，概念集成是生产质量的中心，而拥有一个系统架构师则是通往概念集成的最为重要的一步。这些原则不仅仅限于软件系统，也适用于任何复杂结构的设计，无论是一台计算机，一架飞机，还是一个战略防御倡议，一个全球定位系统。[7]

21

2.5 结论

在一个没有建立起软件架构师职业的世界里，随着时间的推移，不断产生出众多失败或半失败的系统，FAA和IRS只是其中的代表。所浪费的金钱和智力资源令人惊叹，甚至有更大的危害：软件危机已经阻碍了信息时代的繁荣。

阿尔文·托夫勒称之为第三次浪潮，但是无论叫什么，这些现象都表明我们的世界正向信息技术转变，没有真正的架构师，这种转变不可能实现。

尾注

1. Mitchell Kapor, "A Software Design Manifesto," *Bringing Design to Software*, ed. Terry Winograd (Reading, Mass. ACM Press/Addison-Wesley, 1996), 3.
2. Grady Booch, *Object Solutions: Managing the Object-Oriented Project* (Menlo Park, Calif : Addison-Wesley, 1996), 119

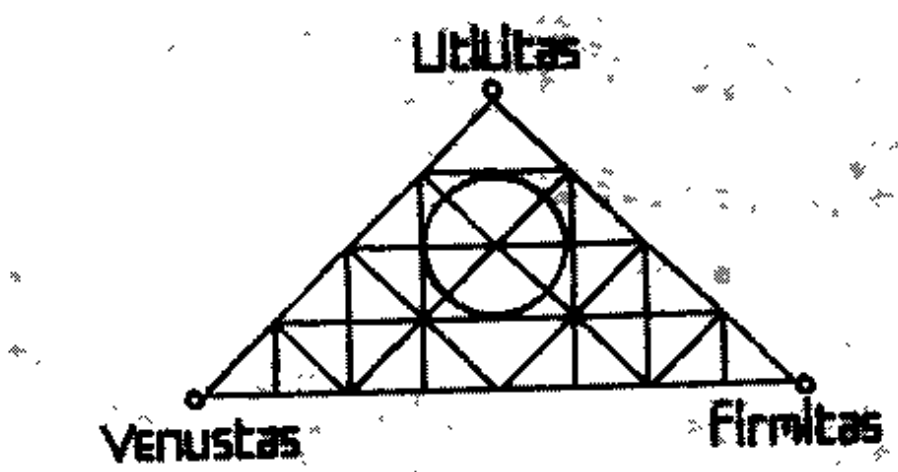
3. National Academy of Sciences, "Continued Review of the Tax System Modernization of the Internal Revenue Service," 1996.

4. Ibid.

5 Ibid.

6. John Ruskin, *The Seven Lamps of Architecture* (New York: Dover Publications, 1989), 1.

7. Frederick P. Brooks, *The Mythical Man-Month*, Anniversary ed. (Reading, Mass.. Addison-Wesley, 1995), 257.


Utilitas
Venustas
Firmitas

第3章　什么是架构

“有架构的城镇才令人愉快。”[1]

“电话有架构，帕台农神庙也有架构。熟悉自己房子的架构其实很容易！”[2]

“架构是人类创建自身世界的第一表现形式。人类以逼真自然的方式来创建架构，并接受自然规律的检验，这些规律支配着自然界，也支配着人类世界。重力定律，静止的和运动的定律，这些规律共同产生了一条归谬法则：所有东西必须合成一体，否则就会坍塌。”[3]

“架构和各种‘风格’无关。路易十四、路易十五、路易十六或哥特式的风格对于架构而言就像是妇女头顶的羽毛：有时很漂亮，尽管不总是如此，此外不再是其他什么了。”[4]

——李·柯贝伊

软件业长期以来一直没有架构师职业，确立这一职业的第一步是去理解架构的本质特征，因为它一直存在于人类的历史中。每个词汇应该具有清楚的意义，但不幸的是，在信息技术领域，词汇、头衔和角色都是混乱和令人迷惑的。任何人都可以自称架构师，“蓝图”把过程和活动详尽地罗列其中，而不是设计，许多公用的词汇都有多种意思。在一个精确性是绝对重要的领域出现这种情形是非常可笑的，但是软件业内人士已经越来越经常地抛开架构师、设计人员、架构、风格和模型等词汇的通常含意而滥用它们。

对于信息技术的业内人士以及客户来说，了解架构是什么至关

重要。架构远不止是“技术架构”（与管道和布线相近）、领域的映射或一组协议。为了使前面的类比更有意义，从而使软件架构师的职业成为现实，我们必须提出并尝试解答以下问题：架构的本质特征是什么？架构是否总是存在？只有在理解这些后，我们才能在基于软件技术的世界中使用架构的真正概念。

例如，架构只是应用在结构上的艺术风格吗？架构是设计吗？建筑架构在我们身边比比皆是，我们可以不去欣赏艺术和音乐，但必须依赖建筑架构，而且我们都在某种程度上了解它，尽管架构（architecture）一词是概念性的，很难下精确定义。许多书都只是为了提出和研究这个问题：“什么是架构？”一个惊人的事实是，架构和有文字的历史一样久远，而问题的答案却永远也只是一个大略，就像柏拉图在墙上的影子一样。

架构这一宏大的概念是与应用活动以及诸如建筑、软件结构、船只等架构产品分离开来的，意识到这一点非常有用。艺术这种完全不可定义的概念也类似，人们只能在绘画、雕刻和表演中找到艺术的物理形式。

> 架构实际并不存在，只有架构的作品存在。架构存在于头脑中，完成架构的作品的人奉献出架构的灵魂——一个根本就不知道风格、技术和方法为何物的灵魂。它只是等待着展现它的载体出现。架构又确实存在，它是不可测事物的化身。[5]
>
> Louis Kahn

关于架构有众多解释和书籍，但是：

> 到一堆书籍中去寻找架构就像到屠户的店中去寻找一头绵羊，要找的东西的确在那儿，但却需要以一种特殊的方式来寻找。[6]
>
> Paul Shepheard

为了达到我们关于软件的目标，迈出切实的第一步，就是了解

架构不是一个只与建筑设计有关的狭隘概念，架构从来都不是这样的。事实上，无论是直接的还是间接的，架构一词在过去一直有着比现在更为广泛的意思。在现实中，建筑架构和软件架构的关系不仅仅是一个能说明问题的类比，因为架构较为宽泛，足以把诸如信息技术系统的结构作为它自身的组成部分。建筑架构和软件架构是同一架构领域的树干上的不同分枝。两者都是架构，软件架构不应该只被看成是一个类似架构的东西，它就是架构。 26

3.1 技术——各种架构的共同基础

维特鲁威是一个生活在罗马帝国时代的建筑师，他后来成为建筑师的开山鼻祖，是*De Architectura*（架构十册）的作者。许多西方架构理论都源自他的大才。维特鲁威认为架构适用于三类：建筑、机械和计时器（指日晷）。他提到的机械是罗马用来摧毁城墙的武器。米开朗基罗后来也是意大利城邦佛罗伦萨的现代武器装备的设计师。

所以，即使最早的架构概念也是非常宽泛的，侧重于技术。可以把架构看成知识的组成部分，是一个设计规则，可应用于多种技术分支。架构的技术性是所有分支（如建筑、机械、计时器、轮船以及现在的软件）的坚强基础。在这个关键的方面，软件架构在意义上是与维特鲁威的观点相一致的，相信维特鲁威也会认可这一类比。

3.2 难以定义的事物有多种定义

每个人都喜欢深究难以定义的事物。John Ruskin于1874年把架构定义成（更抽象一些）对抗力（抑制力）的各种形式，而歌德在1829年把建筑结构称为“凝固的音乐”。当代维特鲁威派的Sallust写道：“每个人都是他自己幸福的架构师”；Milton提到人们是自己幸福的架构师；诗人罗伯特·白朗宁写道：

在那片梦想的遥远土地上，

每个人都是他自己的架构师。

Red Cotton Nightcap Country（1873）

至于架构的定义，可以引用*The New Shorter Oxford English Dictionary*（《牛津新编简明英语词典》）：

architect /' a: kit ɛ kt / n. & v. M16.|法语：architecte. f. It. architetto, 或者他们的词源L architectus f. Gk arkhitekton, f. arkhi-ARCHI- + tekton建筑者。| A. n. 1建筑的设计者，准备规划，
27 监督构造。M16. b naval architect。船只的设计人员等。M19. 2任何复杂结构的设计人员。L16. 3规划、设计或策划来实现某个需要的结果的人。L16. 3一种建筑风格；构造或组织、结构的模式，方式或风格。E17.B计算机或基于计算机系统的概念结构和逻辑组织。M20.4建筑的过程或动作，构造。arch. E17.

1 Marine architecture, naval architecture 轮船等的设计和建筑。[7]

所以，无论所设计结构的特性和意图是什么，架构就是架构。与软件的类比不是诗歌上为了押韵而破格，也不是一种方便的理论说明。多个世纪以来，人们已经接受了广义的架构含义，甚至牛津词典的作者也认为软件架构是架构的子集。

3.3 Utilitas、Venustas、Firmitas

维特鲁威写道：架构是由功能、美感和结构这些元素组成。这三个元素所组合的“三和弦”成为自古罗马时代以来的架构基础，简洁而优雅，也足以成为软件架构的理论基础。James O'Gorman关于维特鲁威三和弦令人信服地写道：

建筑设计师以几何的方式思考，我们也必须如此。把维特鲁威三和弦想像成一个等边三角形，其中每一个元素是一个角，每个角都是离散的，但组合起来形成一个更大的整体。这个用等边三角形表示的更大整体，就是架构的作品。[8]

Utilitas代表对结构的需要——结构的功能。三角形的这一边是客户和居住者，无论动机如何，他们都有尚待满足的需求，这可以通过建筑或软件的构造来实现。有的客户把他们的需求看成一个需要解决的问题，例如一个不能正常运转的订单输入系统，效率极低，妨碍了客户服务；有的客户把他们的需求看成是一个增加效益或市场份额的机会，例如一条航线，通过更好的里程回报系统可以获得更大的竞争力；而另外一些客户可能把他们的需求看成是一种为他们所服务的客户、学生或公民提供更优质服务的方法。Utilitas是一个建筑项目的理由或愿望，是此建筑项目将提供服务的功能。这就是客户的角色。

Venustas是设计。设计的产生是为了满足客户的功能需求，代表了系统和材料的结构和艺术布置。这就是架构师的角色和职责。 28

Firmitas代表构造的手段：构造的材料和构造的保障。没有一个稳固、建筑良好的结构，就无法满足客户的需求，也不可能实现设计。Firmitas是建筑工人的角色。

在这个三角形内部，有一些辅助的角色，如工程师，顾问和管理人员。这些角色是必要的附属品，从属于功能、设计和构造主三角。例如工程师通常受雇于架构师来验证结构元素，也有可能受雇于建筑者，就如何构造某种元素提供专家意见。

维特鲁威三和弦被完美地用于解释软件构造，是对在创建结构时的基本角色和职责的一个简洁的认知图。

3.4 设计的神话

通过功能、设计和构造这三元素，各种结构被概念化并被实现，但结果可能非常荒谬也可能异常理想。伟大的设计是那些既符合居住者的需求，在美学上又令人愉悦的产品。这已经是众多书籍的主题，其研究领域已经自成体系。

伟大的设计是建筑设计师存在的理由，但在软件领域，这一点却很少被人们提起。有的职业软件人认为系统的设计根本不需要用文档来记录，只要启动构造的过程就可以了，让设计永远是隐含的。还有的人只把软件架构和设计看成是一个阶段，而不是一个关键的活动。正如Peter Freeman的解释：

> 一般来说，把时间和资源集中投入到分析和设计活动中是明智的，因为在这些活动上花费的1美元比得上在以后阶段花费的10美元甚至100美元。这样做的理由已经被众多文献重复，并成为一个简单的事实：因为在许多情况下，如果能够仔细地理解问题并计划出解决方案，将有助于避免在构造或运行中出现一些严重的、并且难于修复的错误。而且，软件有许多关键的特性，如可靠性，是无法在构造阶段加到系统上的，它们必须在设计时就被加在系统上。[9]

我们希望在降低软件项目令人惊讶的高失败率的同时，确立起一种真正的软件架构职业，以便走向光辉的设计。我们还希望《纽约时报》开设设计竞赛、设计评委论坛和软件架构批评家论坛栏目。伟大的设计来自于设计师对客户世界的理解，来自于他对架构的洞察和构造技巧，不是来自委员会或建筑工人工作的堆积，也不是来自于那些忽略设计或仅把设计看成是开发阶段一个小插曲的那些人。

29

3.5　圣彼得大教堂的教训：和谐与统一

经典的建筑架构有许多意义深远的经验（或教训）值得软件架构师学习。圣彼得大教堂非常著名和宏伟，但当相互冲突的视角效果要求施加到设计上时，它也深受其苦。

米开朗基罗的发明创造力和活力简直是超越常人的，他被教皇指定为设计师，设计一个献给圣彼得的教堂。圣彼得是耶稣的门徒

之一，耶稣曾对他说："在这块石头上我将建立自己的教堂"。米开朗基罗此时已经72岁，而究其一生，他真正想做的只是成为一个雕刻家。但他的客户——教皇，却总让他做一些诸如绘画、建筑和纪念碑之类的事情。

米开朗基罗之前曾经想把西斯庭教堂的设计从拉斐尔的手中夺过来，但没有成功，现在他要努力致力于圣彼德大教堂的设计，直到他89岁去世。后来的教皇们和设计师们都禁不住把自己的想法和理念添加在后续工程中，结果破坏了整个工程的视觉感受，使设计质量严重滑坡。

这里，李·柯贝伊在谈到罗马建筑和圣彼得教堂的教训时，谈及架构统一与和谐的灵魂：

> 建筑规模是相当可观的。用石头构建这样一个大教堂，是一个很少有人敢做的绝活……雅典楼层尖拱的安排通常与古罗马圆形剧场同源，具有相同的高度。整体的模式是一个完整的统一体，它把最宏伟和最华丽的元素组合到一起：门廊、圆柱、方形体、圆筒体、穹顶。壁线的装饰非常豪放，线条粗而简单。整体的设计已经上升为一个统一体，独具风格而又浑然一体，人们的眼睛会把它们当成同一个事物。米开朗基罗完成了门廊以及穹顶的圆柱，剩下的部分落入野蛮人的手中，一切都被毁了。人类失去了人类智慧所能完成的最完美作品。如果让米开朗基罗看到了灾难性的结果，那真是悲剧。[10]

野蛮人之一是贝尔尼尼，正如李·柯贝伊的解释：

> 贝尔尼尼的柱廊本身很美，但繁琐而笨拙，建筑的正面本身很美，但与穹顶没有任何联系。建筑的真正目的在于穹顶，但却被隐藏起来了！穹顶应该与拱点有着某种恰当的关系：但它们也被隐藏起来了。门廊成了一块固体，它仅仅是一个门面。[11]

……愚蠢而没有头脑的教皇解雇了米开朗基罗，可悲的人谋杀了圣彼得教堂的里里外外。如今的圣彼得大教堂看上去十分愚蠢，就像一个富有而性急的红衣主教，缺乏……一切。真是巨大的损失！超越普通的激情和智慧——这才是永久的东西，但已经令人悲痛地变成一个“大概”，一个“似乎”，一个“可能”，一个“我不能确定”。这是多么悲惨的失败！[12]

罗马的教训是给聪明人的，给那些具有理解力和鉴赏力，懂得拒绝和会验证的人们。罗马应该为不完善的教育受到诅咒，把建筑学生送到罗马学习就是使他们荒废一生。[13]

李·柯贝伊对简洁设计的热情非常具有影响力，这方面对于软件架构师来说有很多经验和教训需要吸取。软件的抽象的、不可见的特性掩盖了糟糕的设计，使客户和最终用户无法了解真相，而这种不可见性因为软件业内广泛地缺乏明晰的草图而加剧。软件系统的用户与建筑物中的住户相类似，经历着与糟糕或不一致设计相对应的“正确”和“错误”。正如一个没有经验的人也许会认为圣彼得大教堂很美，同样，一个没有做过研究的新用户也会对一个笨拙的系统感到满意。

3.6　无以言表的品质

每个人对于城镇，建筑物或荒野的认知标准都有一个核心的品质，这是他生活和精神的准则。这种品质是客观和准确的，但无法用语言表达出来。

我们一生对于这种品质的追求可以说人人皆此，是任何人人生经历的症结所在。我们追求最富活力的时刻和最富活力的状态。[14]

——克里斯多佛·亚历山大

无论是软件用户还是建筑物的住户都有过那种满意和不满意

的感觉，并且通常很难确切地表达原因。“我就是喜欢它”是一个完美的让人能够接受的答案，是一个高度的肯定。克里斯多佛·亚历山大，一位来自加州伯克利的卓越的建筑设计师，在*The Timeless Way of Building*（建筑的永恒方式）一书中把这种无法用语言表达的情形称为“无以言表的品质（quality without a name）”。它是“什么是好的架构”以及“什么是好的设计”这两个问题的核心。

无以言表的品质通常只能体验，无法言表。它可以是一幢建筑，一个人，一个软件应用程序，一部音乐作品，一个城镇，甚至任何东西的一部分。亚历山大先生思想的信条之一是基于模式（pattern）：我们的生活是事件的模式，被一遍遍地重复；一个城市按照在其中 31
居住的人们的模式不断演化；一幢建筑是设计模式的集合体，这些模式或者方便居住者的生活，或者妨碍他们的生活。好的架构，好的设计都必须与我们行为与感知的单个模式以及总体模式相符合。

考虑两个人类社会生活模式。一方面，一些希腊村庄的街道上都有一条石灰水刷的道，4～5英尺宽，每幢房子的外面都有，使人们可以把他们的椅子放到街道上，这样街道一半是自家的，一半是公用的，从而扩大他们的生活空间。

另一方面，洛杉矶的咖啡馆都是室内的，远离人行道，这是为了避免食物受到污染。

但这两种模式都有用途。前者是允许人们为街道的生活做出贡献，成为街道的一部分，这是通过标记出范围来实现的。后者是为了人们的健康，通过确保人们不吃到粘染了脏东西的食物来保证。但前者是开放而主动性的，后者是封闭而被动性的。[15]

亚历山大先生进一步解释了希腊模式的石灰道能够自我延续。居民们每年都自愿地刷新它，因为这个模式与他们联系在一起，并且有价值。另一方面，室内的咖啡馆只能通过法律来维持：

在春天的日子里人们喜欢呆在户外，喝着自己的啤酒或咖啡，看着世界在自己的身边走过。而在公共健康的法律束缚下，人们会感到在咖啡馆里仿佛是被囚禁。这种情况是一种自我毁灭，不仅仅因为它会随着支撑它的法律的消失而改变，而且更微妙的原因在于，它不断地产生室内的冲突，积蓄压力……当得不到满足时，压力不久就会越积越大，最后爆发出来，以另一种破坏或拒绝合作的形式表现出来。[16]

这是克里斯多佛·亚历山大有关架构模式的一个非常好的例子，完美地展示了自然和人类的审美观念，两者神秘而客观地引导人们具有或多或少的活力感。这些内在的支配力量存在于伟大设计的核心，对软件架构职业具有深远的重要影响。

3.7　结论

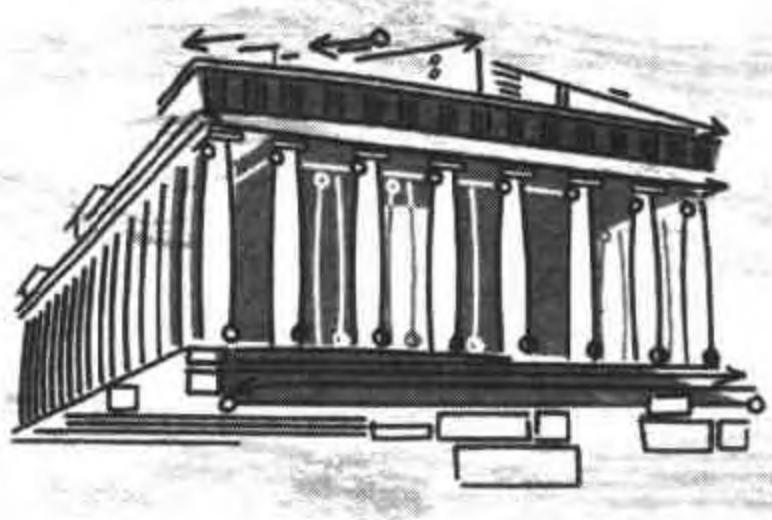
什么是软件，而并非是由人类行为模式和感知模式构成的房间、建筑、城镇呢？有的应用程序使用起来非常方便，具有自我支持的能力；有的应用程序会把使用者气得想把机器扔出窗外；有的软件会激发出游廊的神奇感觉、使人忘却时间，如同具有无比的活力。软件也遵循与以往建筑结构相同的审美原则，同样要求应用好的设计来使其不流于一般。

我们必须看到，只有架构的产品是各不相同的：建筑、软件、船只、机器、钟表甚至我们的命运。材料的差异也是巨大的：木头、钢铁、计算机代码，而这些材料也推动完全不同的技术和行业的发展。但是架构的所有产品都属于设计师，是他们创建出结构来扩展和增强人类的活动本身。这些架构产品集中于技术，并通过utilitas、venustas和firmitas三角形的相互依赖而得到实现。

软件架构师职业一直被软件构造所遗漏，需要确立起来。只有一个与经典建筑设计概念一致的真正的软件架构学科，才能把软件提升到与过去的伟大设计传统同等的地位。

尾注

1. Le Corbusier, *Toward a New Architecture* (Mineola, N.Y.: Dover Publications, 1986), 15
2. Ibid., 15
3. Ibid., 73.
4. Ibid , 37.
5. Louis Kahn, in *Louis Kahn: Writings, Lectures, Interviews*, ed. Alessandra Latour (New York: Rizzoli International Publications, 1991), 168
6. Paul Shepheard, *What Is Architecture? An Essay on Landscapes, Buildings, and Machines* (Cambridge, Mass.: The MIT Press, 1994), 25
7. Lesley Brown, editor, *The New Shorter Oxford English Dictionary on Historic Principles* (Oxford: Clarendon Press, 1993)
8. James O'Gorman, ABC *of Architecture* (Philadelphia, Pa.· University of Pennsylvania Press, 1998), 10.
9. Peter Freeman, *Software Perspectives: The System Is the Message* (Reading, Mass.: Addison-Wesley Publishing, 1987), 148.
10. Le Corbusier, Ibid., 170.
12. Ibid., 171.
13. Ibid., 172
14. Ibid., 173
14. Christopher Alexander, *The Timeless Way of Building* (New York: Oxford University Press, 1979), ix.
15. Ibid., 119.
16. Ibid., 120.



第4章 建筑简史

“过去是序幕的开始。”[1]

——威廉·莎士比亚

4.1 希腊的理想

“建筑”（architecture）一词源于古希腊，该词出现之后，希腊人很快就制定出了条款，这就是Ictinus建筑公司和Callicrates建筑公司合同中的第一个惩罚性条款，这两个人都是帕台农神庙的官方建筑师。此外，他们同时还是并不出名的多利安式建筑的建筑师。因此，人们认为是雕刻家Pheidias将帕台农神庙提升到伟大艺术的顶峰。无论如何，帕台农神庙证明，人类走向现在或将来顶峰的前进道路并不平坦。从技术上来说，毫无疑问，我们现在远远超越了我们的祖先，但是在艺术成就上却完全是另一回事了。大量的文献都著述了帕台农神庙，每一篇著作都非常肯定地论证了该建筑在大约公元前450年就达到了其尖峰之颠。

即使是爱挑剔的李·柯贝伊——他总是对事物持激烈的观点，或者说他从不放过别人的缺点——也这样说：

> 没有任何一个地方也没有任何一个时期曾有过帕台农神庙的辉煌。这种情况一度也会发生在事物的最佳状态；当一个人受到他高贵思想的激发，将这些思想塑造成光与影的永恒雕塑。帕台农神庙的塑造是真确无误的，同时也是其他塑造不可比拟的。严格地说，前人远远超过了我们，或说超过了人类正常的能力。在

这里，凝固着对感觉生理学和建筑本身所包含的思想（数学上的思索）最完整的忠实：我们由于感官感受而凝神，在自己的魂灵中心神情荡漾，我们触摸着和谐的轴心。没有宗教教条的介入，没有抽象的描述，也没有自然主义的表像，除了将事物间正确的关系以简单的形式表达出来之外，没有任何杂质参杂其间。

两千年来，那些曾经看过帕台农神庙的人都认为这里是"建筑"（Architecture）最伟大最辉煌的时刻 [2]

在我们这样的技术时代，帕台农神庙的建筑向我们讲述了很多卓越的用户界面。由于人类视觉的失真，那些看起来是由直线和起支撑作用的垂直圆柱所组成的完全是矩形的庙宇，实际上是由许多巧妙的曲线组成的。巨大的多利安式圆柱在中间凸起，这样肉眼看起来就是直的，而不是向里凹的。圆柱向内倾斜2.375英寸，伸向顶部。如果每个圆柱向内的轴心都向上延伸，那么它们最终将在天空中1.5英里处交合。庙宇的柱座，或叫做基石，缓缓的向上凸起直至建筑物正面末端大约15英寸处和侧翼中间位置的28英寸处。如果将这个巨大弯曲的半径延伸成一个巨大的弓形，那将会形成一个3.5英里的圆。

作为一个整体，在平台基础上向上攀升的曲线的一般效果或可被比作是在西瓜的表面切割一个矩形。[1]

——William Bell Dinsmoore

这些并不仅仅是对几何学的简单实践。它们是很多对角度和曲线变化进行精确规划的例子中的两个，这些精确规划的角度与曲线的变化形成了帕台农神庙统一而整齐的线条。延伸的轴是规划中的一部分。庙址的选取经过周密考虑后定在雅典卫城的后面。轴心从位于比雷埃夫斯的大海中，穿过帕台农神庙，直到蓬特利卡斯山。

整个建筑，可以说是建立在主观基础上，而非客观基础上的。它的主旨不是要达到数学上的精确，而是为了要适应于参

观者的眼睛。对人类来说，曲线远比直线更令人心神愉悦，同时，并非严格的正确性也赋予了这个坚固的、大理石建筑意志的品质，甚至是生命的品质。[4]

——Percy Gardner

作为一个真正的民主国家，古希腊的建筑师需要接受来自公众的监督。由于花的是人民的钱，人们就可以向雅典政府抱怨这座神庙的开销！伯里克利是一个精明的市长，也是一个很富有的人，他说："好吧，我会用自己的钱来修建帕台农神庙，我会把自己的名字刻在上面。"雅典的公民们马上就做出了让步，同意让伯里克利使用公众资金，而且要用多少就用多少。而他也正是这么做了，并给我们上了一堂不朽的实践课。

4.2 建筑师：无名的工匠与超级明星

建筑师常常不得不在政治和自我之间做抉择，即使他们有时会被一些贵族恩主娇养。伊姆贺特普在塞加拉设计了埃及第一座金字塔，这是在Zoser国王的资助下完成的，但是，这些建筑的设计经常归功于宝贵的资源，虽然所有这些工作都是由这个贫穷的建筑师所完成的。不过，天道仍在，伊姆贺特普本人后来被视作了神。

伊姆贺特普看起来是第一个超级建筑师，虽然直到现在有些人仍然拒绝相信一个凡人会设计出金字塔，相反，还认为是外星人设计了金字塔。上帝，外星人，抑或只是一个建筑设计师——我们永远不得而知——但是建筑师却是不一而足，从长久以来倍受尊敬的伟大艺术家到从社会底层涌现的无名工匠，很多很多。

我们不知道印度建筑师和建筑工人的名字，是他们发明并领导了从坚硬的石头上刻出完整的庙宇。建筑者将石头分割成250 × 160英尺的石片，这是Kailasha神龛成型的初步。从这些石片中，他们

雕刻出复杂的中楣、栋梁、雕像和浅浮雕。内里都是手工雕刻的，用成千上万的人物图形和动物图形来装饰。然后，在环绕庙宇的墙面的三个相邻面上，再雕刻出一系列的小庙宇和僧侣。有很多这样的庙宇，Kailasha是其中最受瞩目的一个。

还有让人很难以置信的是坐落在菩提加雅的佛教寺庙，是这里的寺庙尤其具有完美的哥特式建筑的弓形结构。它们的历史可以追溯到公元1世纪。在西方，没有人能够对此做出解释。

这里简单描述的伟大的建筑遗产并非在所有国家都存在。16世纪以前，中国的建筑是很少的，因为很多早期的杰作都是取材于木头。除此之外，建筑那时在中国是一个备受冷落的艺术。建筑师远不如制陶师受到尊敬。

37

首先，中国建筑缺少像其他伟大国家古建筑的三个机制：世袭贵族制、强有力的宗教制度或神权制以及一个富足强大的中央政府。[5]

——Will Durant

欧洲的中世纪受到强有力的神权制和贵族制统治。大教堂就是这种统治的结果。创造这些奇迹的建筑师，就像那个时候的艺术家一样，一直默默无闻，直到文艺复兴后，他们才又再次名声昭显。米开朗基罗当然是名人，就像帕拉迪奥一样，后者是惟一成为一个名词术语的建筑师——帕拉迪奥主义。同时，也成为一个形容词：帕拉迪奥式窗户修饰了大大小小的博物馆，旧式学校以及郊区的居民区，直到现在它们看起来还是一如400年前文艺复兴时期的威尼斯那样充满了生命力。

还有一些无名的建筑殉难者。譬如，有一个不幸的建筑师被法国国王的情妇砍去了双手，仅只为了防止他再为她的情敌修建同样漂亮的城堡。

4.3 现代建筑：它的兴起和没落

阿尔文·托夫勒在《第三次浪潮》一书中将历史描述成一连串浪潮撞击下的变化。第一次浪潮发生在一万年前，由于人类在农业上的发明创造导致了文明的革命。这一次浪潮将人类社会从分散的游牧部落生活改变成以饲养牲口和捕猎来给养村民的乡村生活，并最终演变成现代社会。在第一次浪潮中，当居所越来越先进，房屋也从驱寒避雨的庇护所延伸出更多的意义时，建筑诞生了。

第二次浪潮是工业革命，目前仍在进行。工业革命始于1600年前，自那时起人和动物不再是动力和生产的惟一资源。从家庭模式到通信、政治、商业和经济，机器和能源改变了生活的方方面面。

当一切都在向美学的角度演变时，建筑的摩登时代开始了，并与第二次浪潮同步前进。正如Jonathan Hale在*The Old Way of Seeing*（《视觉的旧方式》）中所言，“正交”一词过去的意思是“成直角”，从1828年才开始具有它现在的意义。但是某些快乐——按自己的意愿去做事的快乐——消失了。男人们不再穿有褶边的衣服，或是羽毛，也不再穿五颜六色的紧身衣。现在甚至女人们穿的都是颜色灰暗的衣服。正确性被放在灵感之上，严肃性被也被过分强调。

38

（1820～1840）这个时期的关键特征，看起来好像是灵魂已经开始有了自我意识。[6]

——爱默生

同样，建筑也开始有自我意识，开始严格而精确地计算。过去那些伟大的建筑遗产是由直觉几何图形、式样和调节线来设计完成的，现在已被严格的函数所取代。建筑开始有自我意识，那些令人赏心悦目、质量上乘但却没有名字的建筑大多都消失了。

“永恒的方法”和“视觉的旧方式”让步于新兴的摩登时代，建筑物不再仅仅是漂亮的建筑，更成为社会现状的表白。其损失远不止这些。

著名的“黄金分割”比例从伊姆贺特普一直到殖民时代的木匠，对建筑师和建筑工人起着指导作用。这种视觉的方式，这种特有的几何学，现在全都荡然无存了：

> 一个鸡蛋，一朵苹果的花蕊，一张人脸，一个海贝壳——所有一切都体现出了黄金分割比例。基奥普斯金字塔（胡夫金字塔）可能就是其建筑上最生动的表现。同样，帕台农神庙的外观也遵循了黄金分割，查奇斯大教堂更是充分展现了黄金分割的协调性。在其他杰出方式中，黄金分割比例也叫神圣比例。[7]
>
> 黄金分割的比例是这样的：a和b的关系相当于b和c的关系，a加b同时等于c。[8]
>
> 糖枫树的分割比例和奥黛丽·赫本的脸的分割比例相同，她或许看上去不象树，但是她(的分割法)却和树是相同的。[9]
>
> ——Jonathan Hale

这种指导原则的缺失导致人和建筑间一定程度上的不协调。在此之前，建筑中使用的分割比例与人的空间尺度不仅是非常协调的，并成为人类空间的延伸。现代的建筑形式却割离了人类潜意识中的和谐。在建筑中，我们不再有“家”的感觉。

当即使是在农村，贸易都已取代了易货制度时，“大”时代开始了。房屋的设计是为了表现“尊贵”、“家庭的幸福”，以及“力量”的展示，而曾经的令人愉悦的生动性和远古比例的神秘都已一去不复返了。

随着工业革命不断向前发展，房屋也变得越发的专业化。实际上，人们开始需要全然一新的建筑：从批量生产成衣、小零件和报纸的工厂，到储存各种货物的杂货店，到美国中西部的谷仓、消防

站、类似巴黎市里的火车站，以及各种办公大楼。新的建筑材料和批量生产都需要复杂的工程和新的建筑方法。可采用的建筑形式成指数地增长，对越来越大的房屋的需求也日渐膨胀。公众安全成了观注的焦点、从而导致了房屋范围和标准的革命性的变化。

职业化、执照、认证和规章的时代就此诞生，并且从未停止过。在19世纪中期之前，还没有任何建筑学校，没有任何学位课程，只有在职培训。建筑师都是自我封号，通常他们是些泥瓦匠或木匠，这些人通常在设计上有天赋和喜好。著名的建筑设计师经常是从艺术开始的。美国建筑学院诞生于19世纪50年代，随后，在整个工业化国家，各种建筑学校纷纷开始建立。

4.4 建筑设计师是社会的思想家

20世纪的建筑主要将重心放在新的领域上——城市。建筑设计师们成为城市规划者和社会批评家。根据乌托邦社会主义的精神，很多人认为如果建筑的条件充分，那将完美地体现人类的自然本性。传统的家以前常被寄托乡情，但是当居住者象笼子里的动物那样居住在批量生产的、由钢筋混凝土铸成的、犹如迷宫般的街区单元中时，包括我们的朋友李·柯贝伊在内的建筑师都认为，人们的眼睛需要适应这一切。

但是人们的眼睛从来都不能适应钢筋混凝土的房屋或是立体式的生活　一个个装了玻璃的房屋被嘲弄为文件档案柜。Norman Mailer曾经尖刻地说，建筑师在人脑中建立的是一片死亡之地，使那些不知如何排解他们内心孤独的人产生很深的疏远感。Tom Wolfe写了*From Bauhaus to Our House*（《从鲍豪斯建筑学派到我们的房子》）一书，可以称得上是“解构主义的一次卓越的解构”。很多受现代主义启发所构建的建筑已经被拆毁，其他的也将步其后尘。法兰克·劳埃德海上保险协会位于Johnson Wax 的总部屋顶已经渗露。

现代主义者的实践并没有为人们造就精神和美学上的典范建筑设计师。没有对建筑师化复杂为简洁表示感谢，没有为他们对社会重建所做的英雄事迹拍手称赞，更没有对他们的创造性的建筑技术交口称赞，建筑师们就这样被轻意地忽视了。自从他们的美学改革之后，公众的品位和鉴赏力再没有实质性的提高。尽管还有很多很好的现代建筑，有很多建筑师对公众的
40 需求表现出异常的敏锐，但悲剧仍然出现了。缺乏可靠性的设计规范意味着同类型的建筑仍在被修筑，但被伪装得看起来不再乏味，不再受冷落，不再千篇一律。向后现代主义转变就要开始了……。[10]

——Paul-Alan Johnson

4.5　架构与第三次浪潮

目前，我们正处于第二次和第三次文明浪潮相互碰撞的混乱之中。我们正从一个以都市为中心、被工业化和批量生产所驱动的社会向一个基于分散的信息和想像力的社会发展。除了少数几个城市之外，例如纽约和芝加哥，大部分城市已经是一片凋敝之气。第三次浪潮正在卫星城和家庭型办公室中蓬勃发展。有时在转变之间时，会很难看清现实，但事实是，一切都改变了。

伟大的新艺术不再被悬挂在画廊的墙上受人顶礼膜拜。它们常常被收藏起来，成为第二次浪潮中那些日渐消亡的精英们自恋、虚无的表现。现在在设计革新中常常可以发现流行艺术的影子。电影、电视、网址、时装和广告在视觉上都是引人入胜的，并且更是令人惊骇的多产。

美学的品质已经民主化。一直到20世纪80年代之前，在都市上层建筑之外，要想找到设计精美的家具、服装、设备、瓷砖，甚或是一杯美味的咖啡，都是非常困难的，对普通百姓而言，可以说是力所不能及的。而现在，设计精美的商品可以在任何商场或沃尔马

超级市场中找到，那些旧派商品于是乎就常常相形见绌了。在堪萨斯州，卡车司机背着名牌背包，上面画着莎士比亚和詹姆斯·乔伊斯的肖像，坐在星巴克咖啡店啜饮咖啡，这都是常有的事。

但具有讽刺意味的是，在品质和信息飞速发展的时候，建筑又回退到古典形式了。请注意在电影和商业中作为背景的房屋，它们基本上都是那种既温暖，又坚固，充满了战前风情的样子：有竖框的窗户、壁炉、精致修剪的院落、石砖小路、木地板，有很多的狗和各种各样的花。这些都是古老的、充满人性的建筑。在现代主义里都是没有的。郊区住宅区到处都是传统的砖瓦房，有着帕拉迪奥式的窗户和舒适的僻静角落。

一些学者说我们现在处于历史的边缘。在建筑上我们无疑是在独创能力的尽头。看起来我们更可能正处在进入第三浪潮的过渡阶段。当架构的一个分支——建筑架构日渐消亡，在第二次浪潮中的竞争力也日渐殆尽，软件架构——架构的另一个分支——出现了，

并为迎接信息时代的挑战做着准备。

4.6 结论

历史明示现在，告诉了我们可以从建筑业中学习到很多经验教训。所有的架构都应该学习设计的历史和技巧，但我们不应只重现历史或是做一些后现代的假设。我们的民族特性应伴随我们自己的学科和理论铺展开来。这一切都将建立在第三次浪潮的现实基础之上，而并不应再追随着第二次浪潮的模式和解决方式。软件架构正在兴起，它将在成功演绎第三次浪潮的进程中起到重要作用。

尾注

1. William Shakespeare, *The Tempest*, in *Familiar Quotations*, 16th edition, ed. Justin Kaplan (Boston Little, Brown, & Co., 1992), 224.

2 Le Corbusier, *Toward a New Architecture* (Mineola, N.Y.: Dover

Publications, 1986), 219–221.

3 William Bell Dinsmoor, "The Design and Building Techniques of the Parthenon," 1951, *The Parthenon*, ed , Vincent J. Bruno (New York W.W. Norton & Co., 1974), 184.

4 Percy Gardner, *Grammar of Greek* Art in William Bell Dinsmoor, "The Design and Building Techniques of the Parthenon," 1951, *The Parthenon*, ed , Vincent J Bruno (New York: W.W. Norton & Co., 1974), 178.

5 Will Durant, *The Story of Civilization: Our Oriental Heritage* (New York: Simon & Schuster, 1954), 741.

6 Ralph Waldo Emerson, in *The Old Way of Seeing*, Jonathan Hale (Boston: Houghton Mifflin Co , 1994), 42.

7. Jonathan Hale, *The Old Way of Seeing* (Boston: Houghton Mifflin Co., 1994), 61.

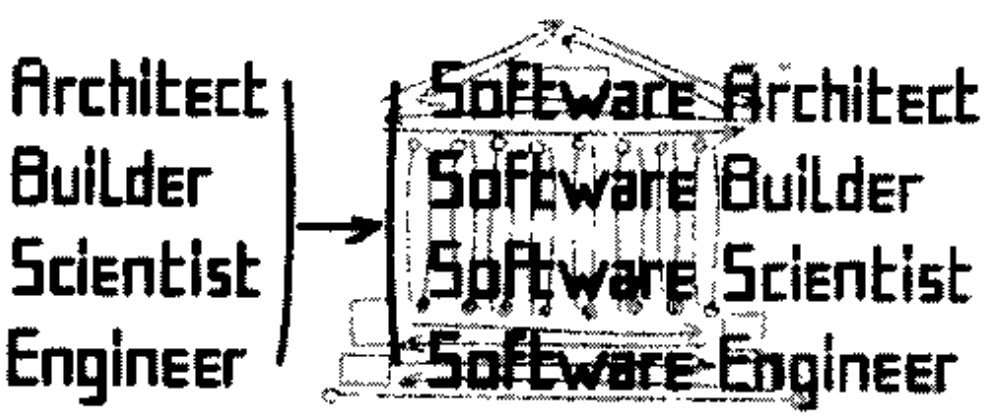
8 Jonathan Hale, *The Old Way of Seeing* (Boston: Houghton Mifflin Co., 1994), 61.

9. Jonathan Hale, *The Old Way of Seeing* (Boston: Houghton Mifflin Co., 1994), 69.

10. Paul Alan Johnson, *The Theory of Architecture: Concepts, Themes, &*

 Practices (New York: Van Nostrand Reinhold, 1994), xiv.

-


Architect
Builder
Scientist
Engineer
Software Architect
Software Builder
Software Scientist
Software Engineer

第 5 章　软件构造过程中的角色

要想使架构师的重要作用能够被理解，就必须把架构及用户想像的产品定义与它的实现区分开来。[1]

——Frederick P. Brooks, Jr.

5.1　建筑设计师、建筑工人、工程师、科学家

建筑设计师、建筑工人、工程师、科学家都是泾渭分明的职业，这些职业都和我们日常生活相关，大家也大概都知道他们做什么，他们的产品是什么，使用什么样的工具，他们应负什么样的责任，以及他们受到什么样的职业培训。我们甚至从直觉上就可知道这些专业应该具有什么样的禀赋才干以及自己将要选择其中某个作为一生的职业。事实上，这正是职业课程所教授的内容。

现在，让我们在这几个名词——建筑师、建筑工人、工程师、科学家——前面都加上一个词“软件”或“计算机”时，原本很清楚的概念变得模糊了。人们通常无法将软件构造中的工作头衔和工作角色区分开来，即使可能，也是一种随意行为。这就是问题的所在，也是软件业一直陷于困境的原因。如果你去询问首席执行官、首席信息官，甚至是专业软件人士自己，到底软件工程师、架构师、程序员和计算机科学家之间的确切区别，以及他们的工作、作用和责任，恐怕他们自己都没有一个统一的答案。答案要么晦涩难懂，要么有相同之处，要么大相径庭，因为这些头衔或者这些角色的标准的有效性太少了。那么，不可避免的结果是进程和结果没有可预测性。

这种对角色的定义、区别和责任定义的缺乏就是软件失败的关键根源。除非这种根源得到解决，否则还会继续失败。首先要解决的是建筑设计师、建筑工人、工程师和科学家这几个名词的意义能得到普遍的接纳，同时还要抵制目前软件业中对这些名词的滥用。

在建筑和软件构造之间有一个类比，为头衔的区分和角色、技术和架构过程的可预测性提供框架、认知图和模板。只有在这种透明度下，我们才能将软件业从品质、成本、美学和责任的危机中解救出来。

5.2 指导原则

每一种职业都需要有一个简单的组织主题，或者说指导原则，以定义该职业，从而形成它的使命和任务。例如，医生救死扶伤，老师传授知识，厨师掌厨烹饪。这些基本的指导原则在软件业中已然消失殆尽。我们接纳软件工程师，他们有时是架构师，有时又是可以完成构建任务的计算机工程师，而有时他们又是建筑者，常常在没有任何蓝图或范本的情况下，策划、设计或者建筑。换句话说，在软件业我们的角色是自我定义的，我们的头衔与我们的角色、技术、所受的训练以及经验是不相吻合的。

特别是软件工程师，目前正在对他们做出明确的定义，并建立确切的专业。但是他们自己以及其他人在不明白其指导原则的情况下，仍然继续使用“科学家（scientist）、建筑者（builder）、工程师（engineer）和架构师（architect）”这样的头衔。结果导致这些词不再有效，所有试图定义这个职业的努力都陷入一片迷雾之中。

但是回顾职业课程，建筑师、科学家、工程师和建筑工人，所有这些传统职业的指导原则都是非常清楚的，可以用以下的方式传授给学生：

建筑设计师说：是我决定建筑物应该是什么样的。

科学家说：我的工作是研究、发明，进一步挖掘知识。

工程师说：由我来确定建筑物是否修建得合理。

建筑工人说：我的工作是修建房舍。

计算机科学、软件工程或者程序设计的各种职业不是由软件架构师来定义的。不过，我们可以定义自己的职业，告诉其他软件工作者在我们的计划中，我们是如何看待他们的工作性质的，我们又打算如何利用他们的技术。通过定义和完成软件架构师的职责，通过我们的计划及其实现，其他软件职业的分级将会非常清晰。由于我们都对与此类比相一致的“架构驱动软件构造（architecture-driven software construction）”负责，所以客户和我们自己将对这个过程进行确认。

软件构造的客户和消费者是软件业这一转变的最终仲裁者和催化剂，这一事实决非夸大其辞。他们将雇佣实现这一方法的架构师，以及其他担任以往工程师、科学家和建筑工人角色的专业软件人员。客户和雇主将创造这些职业，并对这些工作做出定义描述。客户将决定是否需要架构师来设计、是否需要工程师来使强度和力度最大化，是否需要程序员来创建代码模块。其实，在很大程度上，他们已经这样做了。

作为架构师，我们需要从一开始就向客户解释这一类比，解释架构驱动软件构造，同时还要强调，与Bass等人不一样，我们毕竟不是“软件工程师。”我们是软件架构师。 47

5.3 软件架构师决定结构的外观和功能

软件架构师设计基于软件的技术结构，并且是构建过程中设计的管理者。软件架构涵盖了软件系统外观、感觉、功能以及所有的软件系统组织形式。同时它还决定了软件系统如何与现存的软件结构协同工作。软件架构包含了大大小小的软件结构，还有建立在软

件基础上的产品和已嵌入的软件要素。

软件架构指的是软件架构师的工作产品以及他们的实践。它是venustas，是构建程序的设计方面。就像修建房子一样，软件架构将技术和功能同人类的美学观点和生产率相结合，来决定居住者对结构的感觉和看法。软件架构要满足人类的需求，解决人类的各种问题，同时通过设计促进人类的活动。

5.4　软件工程师使结构合理

所有的工程师都尽量通过数学和科学的精确性，来最大限度地提高结构、机器、系统或化学公式的强度和力度。他们的任务就是确保建筑能稳固，桥墩可以承受飓风，机器在压力下也可以正常运转，软件可以处理加诸给它的各种问题。

软件应有怎样的外观、如何运转、需要什么样的配置，这些都是软件架构师的工作。对应地，由软件工程师评估、量化，并提出建议确保该软件的效力和性能。要完成一个预定的安装需要占用多少已知的硬件？在一定环境下，如何使响应时间最优化？这些都是软件工程师应该回答的问题。他们设计工程解决方案，就像结构工程师设计一流的新方法，能使桥梁和建筑稳固。

设计工程的元素和解决方法是架构和工程相互重叠的领域。工程师不断找到出色的解决方法，建筑师将工程元素应用于建筑的外观设计中。作为一个非常好的例子，乔治·华盛顿大桥最初的结构设计是从南康涅狄格州开始，外体覆盖一层大理石。但当纯钢结构的大桥竖起后，它看起来是那么壮观雄伟，于是就这样保留了下来，没有再裹上那层石头外衣。

48

李·柯贝伊曾经详细评论过20世纪建筑设计的断层现象，他相信那个时代最好的设计作品——从汽车、谷物升降机到桥梁——都是工程师的杰作。

> 受到经济学法则的启发和数学计算的制约，工程师使我们和大自然法规相契合、他们实现了和谐 [2]

建筑的范畴从古教堂的屋顶一直到摩天大厦，体现了建筑学和工程之间很强的相互依赖、互相依靠的关系。由于建筑总是依赖于技术和工程来实现它的设计，从而使得工程和建筑之间存在这种联系 但在维特鲁威三和弦中，工程是firmitas的一部分 建筑学是venustas。尽管工程会影响到美学，建筑学会对工程产生影响，这两者的任务和组织原则是不一样的，而这两个学科的技能和主体也是不尽相同的。

软件工程一直在试图给自己一个定义，但没有类比和维特鲁威三和弦的组织概念来指导完成这一任务，从而导致身份（定义）危机。一方面，他们希望被看作是真正的工程师，可以获得认证和行业许可证。另一方面，他们又很难将自己与架构师和程序员区别开来。

在*After the Goldrush: Creating a Profession of Software Engineering*（《在淘金热之后：创建软件工程的职业》）一书中，Steve McConnell提出了一个非常好的问题："什么是软件工程？"但是，一直到书的结尾，他也没有给出一个结论性的答案。

> 我们需要在几个领域继续努力——建立广泛的本科教育，颁发职业软件工程师执照，设立软件工程认证程序，将最佳准则应用到这一行业中。软件工程领域至少需要10～15年的时间来建立一个真正的软件工程职业。[3]

McConnell和其他软件工程师，创立了计算机科学家的又一分支，却又把软件工程师看成其他任何一种：设计时是架构师，编码时是程序员，更多时候，又是应该得到证书的真正的工程师。

如果真是这样，软件工程师可以担当所有这些角色，那末他们就不应该再戴有"软件工程师"的头衔，而应该换成像"架构工程

师（archigeer）”或“工程架构师（engitect）”这样更贴切的头衔。在其他某些时候，他们的头衔还是基本正确的。Steve McConnell这样写道：

49

> 工程的能力决定一个系统在多大程度上可以成功建立，使用起来有多么简单方便，运行起来有多么快，包含多少错误，以及如何同其他的系统协调合作。[4]

目前，如果把这个类比用作组织原则，那末这个表述就更进一步的描述了软件架构师的角色，但有一点错误，即软件工程师不能决定系统使用起来有多么容易。架构师设计系统的功能和可用性，即它如何被使用的，而工程师对可用性进行测试，从而使架构师的设计有效可行。

以上的摘引表明软件工程师的工作远比他们真正的任务简单的多，他们相信自己几乎可以负责一切，这就是迷雾产生的原因。达到最优力度、强度和可靠性是他们的任务，也是他们应该保证的。有一些专业人士称自己为“软件工程师”，而实际上他们是熟练的架构师，或介于二者之间。这样当然很好，他们意识到这二者之间有不同的角色和责任之分，可以充当任一角色。

软件工程师正极力探讨认证和执照（许可证）的问题，作为获得职业可信度和标准化的途径。这个或许是走向正确方向的一个动力，但他们首先应该将自己和其他的专业人士区分开来，对自己的角色做出明确的定义。架构师 - 工程师 - 建筑者三者合一的想法太不切实际。现在不正是承认这三者的角色都太专业，太复杂，从而无法合为一个职业的时候吗？软件架构师或其他任何相信这个类比的人都会说，软件工程师是工程师，这个名词和角色都是非常清楚的：软件工程师的任务是使结构合理。

还有就是，当软件工程师把自己称做“架构师”时，他们通常暗指在技术层面上的设计，而非真正的架构层面。工程的世界观来

自系统的技术层面，工程师用“架构（architecture）”一词来描述设计，这个设计是与横梁、承重墙以及其他结构系统类似的工程元素的设计。

工程师通常看不到客户、整个企业或领域的传统架构优势，所以当他们使用“架构（architecture）”和“架构师（architect）”来描述他们的工作时，仅仅引用了这两个词，却没有表现出它们的真正意义，也就没有完成真正的架构工作，更没有完成他们传统的角色任务。他们把软件看作是工程师，因此很大程度上是作为工程师在工作着，但又坚持说他们所做的是架构（architecture）。 50

现在的工程师总想把架构师、工程师、建筑者三者集于一身，实在有些自不量力。他们承担太多的责任，其结果是不得不为软件的失败承受太多指责。在软件杂志或关于软件工程的网站上，经常可以看到他们规劝同行，在进行更可靠的软件开发时，要接受进一步的训练，要更严格。他们的挫败感在以下的言论中可以得到印证：“如果我们能从错误中更进一步学习的话……”，或是“我们要更系统地应用更高的标准……”。

这一切是多么令人沮丧，令人灰心！在其他任何人类的工作领域，难道需要同行之间穿着苦行僧的衣服，像这样彼此告戒吗？当然，建筑需要艺术级的工程方可矗立，然而结构工程师没有迫在眉睫的危机。无论结构是精是简，建筑总能使用。

软件失败在以惊人的速度发生着，在寻求对这一问题的解答过程中，软件工程师不断告戒：不要相信“银弹”（Silver bullet）。他们认为，简单的答案不能回答技术的复杂性。他们宁愿为终端客户创建更多更详尽的需求以及更多更可靠的技术，耐心地追求着项目的成功。所有的努力都依赖一个前提，那就是完美的工程能解决这一问题。

Alan Cooper在他的*The Inmates Are Running the Asylum*（《同行

操纵避难所》）一书中雄辩地记录了上述问题，并假设只靠工程就能解决软件危机。以下的摘录足以说明问题：

> 高科技产业一不留神就将程序员和工程师放在了主导位置，于是他们很难使用的工程知识成了主宰。企业行政人员尽管看起来是，但实际上不是控制高科产业的人。真正的主导者是工程师。在我们迫不及待地要接受硅谷芯片带来的诸多好处时，我们已然放弃了应尽的责任。我们让同行正在操纵避难所。
>
> 当同行正在操纵避难所时，他们很难看清楚让他们饱受磨难的问题的本质所在。我们照镜子时，很容易找到自己的优点而忽略瑕疵。当软件产品的创造者检查自己的作品时，他们总是忽略它是多么的糟糕。相反，他们只看到它令人敬畏的力量和可用性。他们看到的是该产品在特性和功能上是多么丰富。他们却没有看到这个产品使用起来是多么的令人烦恼，要学习如何使用它又是多么耗功费时，也没有看到它使每天必须使用它的人在不断减少。[5]

51

工程师们的确存在一个根本问题，就是他们不能超越自己的思维定势和世界观，从而看不到更广阔的天地。他们的假设根深蒂固：软件构造只需要完美的工程来支撑。尽管对架构方法和设计的强调在不断加强，软件工程师却没有完成作为架构师的角色任务，这一类比的指导原则被遗忘，软件失败的根源正是这种遗忘，而不是糟糕的工程。一直以来，我们试图建立详尽的、复杂的软件结构，它却没有维特鲁威三和弦的组织力量。结果当然是所建造的建筑要么坍塌，要么倾斜，要么就是使居住者疯狂。

软件工程师认为可靠的软件工程实践非常重要，这是绝对正确的。仅仅这个任务就已经让他们忙得马不停蹄了。当他们的职业、角色在完成这一任务中而不是在架构和建筑中被定义，并被有条理地组织起来时，他们能很出色完成任务。软件工程师将使软件大厦稳固的矗立，但是决定结构能否像帝国大厦那样坚固的人是架构师。

而“开发人员”（developers）则来完成修建工作。

5.5 “开发人员”建造结构

生命是如此短暂，要学的技能又是如此之多……

——乔叟

目前急需一个更好的名词取代“开发人员”来表示软件结构工程中的建造人员。开发就是展开，揭示和让人知晓。影像通过和化学处理药品的接触展开，从而形成电影。孩子们慢慢长大成人，但是房屋不会成长，软件也不会。它们是建造出来的。家和软件随着时间慢慢发展。家会有新的面貌：更高级的家具、不同的油漆色彩，甚至还会添加新的东西和池塘。软件也会以同样的方式发展。但是，这并不能将“构建者”变为“开发人员”。

程序员也是构建者，但是并不是所有的构建者都是程序员，所以程序员一词并不适合来描述软件构建者。建筑业有很多的专家，比如，水管工、电工、泥瓦工、木匠、劳工、细木工和油漆工。在软件构建中也一样，有数据库管理员、测试人员、网络程序员，以及擅长使用多种专业工具的程序员。这些头衔都非常准确，反映了他们所要做的工作，但应该由开发职业来区分所有的定义，同时再创造一个名词来描述它们作为维特鲁威三和弦的一 52
个分支fermitas。

如果这个类比能得到认可，“开发人员”就需要改变他们的头衔。在建造建筑物时，就像规划住宅区和办公花园，开发人员是承办者，主管整个房地产开发的策划。实际修建人的头衔是按照他们特定的工作来称呼的。他们的头衔反映了他们的技能和角色，在软件业中也应这样。

关于“开发人员”与程序员这两个名词之间的混淆，由于人力资源部将头衔作为奖惩的一个奇特形式，从而使这种混淆更为

严重。如果你获得加薪，他们会认为你的业绩或资历还应该给你更夸张的头衔，即使这个头衔与你的工作毫无关系。随着你的头衔不断改变，他们甚至还准备将数字化等级制度应用到程序设计头衔中，而在所有的头衔中，排名第一的当然是架构师。这就好比是有才干的水管工人从他或她的修理水管的工作中被升级，最终被提为建筑师。

所以，在软件构造的其他分支中所存在的不严格的头衔制，使软件的构建者们饱受其苦，承受同样的后果。不管他们的头衔是怎样的，他们的角色只是依照架构规划来构建软件。他们用代码——软件的修建材料，来构建软件，“开发人员”编写和汇编符合计划的代码，完成底层的设计，也就是说，架构师并不给出具体的细节。

和建筑物的建筑一样，软件开发人员也有不同的特长、技能、才干和经验。在团体的一端，是那些在监督下做着日常工作的人；另一端则是知名的技工和革新者。需要记住的是，尽管软件构造和建筑构造之间的这个类比从角色、头衔和过程上无疑非常正确，但两者的历史背景却大相径庭，不应该对我们作为软件专业人员的思考造成影响。

在建筑业，建筑商业（building trades）的声望在上个世纪就已经衰落了，一同衰落的还有很多房屋建筑师。现在到处都是大规模建造的房屋（例如McStick和McMansion式房屋等等），特殊的艺术要求和小规模的建筑在建筑商业中已不多见了。建筑材料也在改变，特别是在北美，建筑材料已经从石头和坚硬的木头变为聚苯乙烯泡沫塑料，2×4s，OSB干胶板和房屋提前装修。砖成了“陶瓷涂料”，你可以用一根手指就戳穿“粉刷过”的墙。由于对建筑者的技术和才能的要求很低，他们的社会地位也就相应下降了。

53

另一方面，请注意当许多人抱怨建筑越来越缺少个性化和艺术性时，人们却能用很少的钱买到更多的房子，允许低收入阶层拥有

自己的房屋。所以，某种意义上说，对低价位房屋的需求，和对房屋艺术性和个性化要求的降低，使得建筑工人可以建造出廉价的房屋。另一方面，屋主们对建筑的结构选择知道的不多，所以就不知道对房屋的要求应该包含什么。这两个因素相互作用，同软件业的情形一样。

注意，不要将目前对建筑工人的观点影响到“开发人员”身上去。软件构造的角色和过程与建筑构建是十分雷同的，但是与这两个职业相关的声望却是大不一样的。传统意义上来说，建筑者是工匠，是他们修建了古老、神奇而又令人快乐的城镇，而不是城市规划者们完成了这些工作。建筑师正是从这些头衔中延伸出来的。是那些拥有计算能力和天赋的几何能力的建筑者们在上个世纪被人们淡忘了。

这并不等于再没有好的工匠了，当然还有，有时他们被建筑师弄的灰了心，必须弥补他们设计方面的失误。例如，在一个有桁条和椿柱的房子里，一个著名的建筑师完成了他的设计，但却没有注意到木框架梁柱结构的压力外壳中的电力问题。房屋的巨大框架暴露在内部，像一个贝壳一样封存在填满泡沫的镶嵌板里。地板是厚重的舌榫木，楼下的天花板也是同样材料。因此，没有合适外用的夹墙和夹地板可以走电线（或是水管管道——这又是另一回事了）。

惟一能隐藏电线的方法是穿过内墙或者是将电线绕着木梁走明线，然后再顺着椿柱做电线盒走暗线。那么，做这个工作的电工就成了主管，他从一个技工的角度来解决这个问题。于是，不仅电线完全看不到了，就连闭路盒也成了艺术品，每条电线都非常完美地、对称一致地隐藏在线盒中，在拐角处也是那么的精确无误。毫无疑问，这个电工在电力系统方面是个非常出色的设计者。

像这样有才干、有创造性的“开发人员”在软件领域随处可见，他们做了很多独一无二的有艺术性的工作。他们在克里斯多佛·亚历山大的工作基础上，把古老的模式语言应用在他们的工作中（令 54

人惊讶的是，最初是由亚历山大自己开始的）。“开发人员”对亚历山大研究并描述的语言加以改进，并将它们应用到软件的设计中。

在程序设计和开发中，模式的变化是很繁杂的，但它不是架构。开发人员的任务就是设计并修建技术层面上的、底层的构件和系统。亚历山大模式将架构后发式描述成是随着岁月不断演化，现在已经深植于我们的脑子里。在某种意义上，模式是非常老套的：大家最常见的像什么“楼梯井”（staircase well）、“起居圆”（sitting circle）、“儿童洞穴”（child caves）、“有格架的小路”（trellised walk）等等。

在建筑架构中，建筑师常将这些模式应用于家庭房屋的设计。“开发人员”一直采用这样的模式，并将它们应用于程序设计代码的内部技术工作中。这就好像是水管工人将建筑模式应用到他们的工作中，来发现安装管道的新途径。这是个好事情，开发人员希望使用低层模式，但是不能因此就称呼他们为“软件架构师”或“架构师”。我们必须注意不能将构造模式与建筑师所采取的设计模式混淆。

建筑师和建筑工人之间相互依赖，为彼此工作的改进提供了一个积极的动力。建筑师挑战建筑工人的勇气，挖掘他们的潜能。相反，建筑工人挑战并扩大建筑师的设计潜力。建筑工人从传统上一直是艺术家和革新者，他们非常仔细地使坚硬的石头看起来如水般流畅、还将人类整个发展史刻在木头上。

当出现架构师——建筑者或工程师——建筑者的双重角色混合时，通常会吸引具有不同能力、脾气和性格的人。如果没有开发人员融合逻辑、节俭、好奇和独创性的才能，那么软件架构师的全部设计技能就只是设计McStick式软件，或者，最好也只是McMansion式的软件。

5.6　计算机科学家推进知识

计算机科学家的工作是研究、发明、调查，并进一步促进计

算机软件和硬件知识的推广。科学家提出并不断修正理论，通过不断的试验和经验来建立事实。他们用这种办法，在学术界、政府以及进行新产品研究和开发的私人企业中推广知识，这就是科学家所做的事。然而，在计算机世界，计算机科学用来表达其他的意思。

55

计算机科学一度用于学术界中进行计算的部门，这种情况作为普遍规则在许多部门一直持续着，即使学生和教授所涉及的不只是科学。在60和70年代，“计算机科学”是个非常贴切的名称，知识的界限很窄，所有和科学领域相关的东西都需要探索和研究。从这一层面来说，若在当时的情况下，我们都是科学家。

不过，现在这一角色和头衔应该与这个类比一致。如果这些名词仅仅意味着它们本身的意义，那么很显然，科学部门应该生产科学家，工程部门应该生产工程师，架构部门应该生产架构师，而程序部门应该生产的是程序员。计算领域需要有专长的人，个人所从事的职业与本人所得的学位息息相关。

一旦“计算机科学”如上所言，那么它可以完全将重心放在进一步开发知识上。同时，一旦工程、构造和架构被作为知识的组成成份来定义，那么计算机科学将可以有的放矢的指导它的研究、工具和学习。

5.7 客户的角色

……自己才是真实的……[7]

——威廉·莎士比亚

客户的角色是utilitas，是维特鲁威三和弦的一个边，代表构造项目的需求或愿望。客户是这个三角形中的一条边，在设计和构造阶段起着举足轻重的作用，当然在构造结束之后同样也非常重要。只要客户和居住者拥有结构，他们就受到它的影响。

> 软件不只是用户交互的工具，它还是空间生成器，使用户在其间生存。软件设计就像建筑：当建筑师设计一个房子或一个办公大楼，就指定了它的结构。更重要的是，居住者的生活模式被定型，人们更多的是被看作是建筑物的居住者，而非使用者。[8]
>
> ——Terry Winograd

或者，就像温斯顿·邱吉尔所言：

> 我们塑造我们的建筑，因此，建筑也塑造我们。[9]

客户在设计过程中是一个积极的参与者，他们与建筑师不断反复协作，不断改进设计，一直到它成为大家都接受的最终产品。客户的角色就是得到他们真正想要的和需要的建筑，在理解的基础上坚持该修建什么样的建筑。

56

客户不应被建筑师如泉水喷涌般的行话术语，或是光滑的图纸、不断的电话声，抑或是时下流行的建筑式样所吓倒。这个类比永远有效，对于建筑的构造也没什么不同。

> 如果他们在某个地方听说厚玻璃板大型落地窗很不错，他们会接受这个观点，并视其为来自于智者的至理名言，而不是他们自己想当然，即使觉得坐在一个有小玻璃窗的房间里会更舒服，他们还是会说他们是多么的喜欢那样的落地窗。但是，架构师的时尚品位是如此地具有影响力，人们于是无视于自己的亲身感受，仍然相信大型落地窗效果更好。他们已经丧失对自身判断力的信心。他们将这个权力拱手让给了设计师，他们是如此彻底地放弃了他们的模式语言，以至他们对架构师言听计从。[10]
>
> ——克里斯多佛·亚历山大

除了要得到他们真正想要和需要的东西外，客户还应该坚持得

到能真实表现即将修建的东西的规划。客户应该能够使规划、构造过程和最终结果生效。这一切的完成其实与修建建筑的过程是完全一致的。客户无须知道如何修建房屋，如何使用工具，或某些技术名词有何意义。但是客户能够使进程依照规划生效，以确保正在修建的东西与规划是一致的。

最后，在软件业从失败项目走向明晰和成功这一转型中，客户承担了至关重要的角色。如果客户接受这个类比，把它当作一个强有力又很简单的工具来使用，同时还坚持雇佣真正意义上的架构师、科学家、工程师和修建者，那么软件业将会成功。只有在对客户需求的回应中，我们才能看到知识的程序等级、调查、工具、培训、出版物和知识体在各个不同的领域相协调。客户是，也将永远是架构驱动软件构造的推动力量。

归究到底，软件业的专业人员可以就这个类比展开辩论，分析词汇，并出版以自我为参考的冗长文章和缺乏知识的电子邮件。他们可以做所有的一切，但是如果他们没有将他们的软件架构的理念出售给客户，那么他们就得对这一切保持沉默。软件业需要客户来掌控这个权力，我们相信这个类比将会赋予他们掌控的工具。这一切不是专业软件人员所为，而是在自由市场上客户和消费者每天所做的成千上万的决定，才使得这些理念得到不断的改进，有了长久的价值。

57

非技术人员——客户，住户，消费者或只是一般百姓——都有这样一种想法：软件的好坏只有根据技术、工程的标准来评判：有效，可靠，快捷（还有甚至神秘）。克里斯多佛·亚历山大受其在建筑模式方面工作的启发，曾经多少有些诧异地发现“开发人员”一词。他在评判由此而来的软件中将他所认为的缺点如下写道：

> 那些使用亚历山大模式或其他方式开发这些程序的人在做更好的工作吗？这些程序更好了吗？他们是否得到更好的结果，更有效率、更快捷或更有深度吗？使用它们时人们感到更具有

灵活性吗？按照通常的非物质标准，这些程序所取得的成就，那些使用这些程序的人和受到这些程序影响的人，是不是更好，是不是更具有高品位，是不是更有深度呢？

我在这里可以说是有巨大的缺陷，我不是程序员，我不知道如何判断程序的好坏……[11]

首先，高品位的、有精神感召力的和有深度的这三个词很少能从低层“终端用户”的嘴里听到，但是我们可以对亚历山大先生以及所有同样迷惑的人说：有了类比在手中，你可以评判软件。你可以立即知道它是如何被设计、构造和如何运行的。你可以通过操纵它的空间，以及这些空间所带给你的感受来判断它的简易性。你会不会觉得厨房让你感觉很快乐，就好像你可以一整天都下厨？一个软件的空间可不可以让你非常愉悦，使你忘记时间的流逝，感觉电脑就是你灵魂的一个延伸呢？

那就是“不可言表的品质”，无论是居住在房间中或是软件的架构中，无论有没有技术，任何人都可以感受它。

5.8　定义而非限制

架构师、工程师、科学家和建筑者，这些角色之间是相互依赖又相互作用的。每一个角色都受到其他几个角色的制约或授权。一个设计如果超越了科学、建筑，或是工程专家的技能，那么它就不可能被完成。同样，先进的科学、工程和修建技术要求设计以完整、统一、和谐的标准来满足客户的需求。

58

由于这个原因，这些职业接受这个类比是为了定义这些角色而不是限制它们。接受这些经典角色的定义、过程的可预见性和专业词汇将使“venustas、utilitas和firmitas”回归它们的本意。就像摄象机镜头要聚焦一样，每个职业人员都将知道他们的职业组织原则，聚焦他们的努力。只有到这个时候，我们才会拨开重重迷雾，走向成功。每一个职业都需要软件业的支撑，我们需要在能够改进整体

职业之前，先不断改进每一个个体职业。当这一切得到解决后，存在于软件业的危机就会解除，信息时代将硕果累累。

这种职业的精确性需要把研究、书、工具和方法目标化到架构过程、专业词汇和标准集上去，并成为职业的标志。同样，对建筑师和工程师来说他们所需要的方法、工具和材料也是按职业精确刻画的。我们不必再相互借用工具，借用方法和词汇，不必再处心积虑地将别的职业成分徒劳无功地套用到我们的职业中来，以适应我们的需求。

怎样强调职业的精确区分都不为过。很多非技术人员都发现软件业的专业人士们不理解工具就是工具。像某些编程技术或语言一样，工具常常被看作是设计和风格的同义词。这就好比一个建筑师对家的断言："我们把最先进的空气枪技术作为平台。"或是"我们将使用PVC管道架构来把家建起来。"

同时，我们的客户和住户受到这些技术性的firmitas 的困扰，而又不知道如何提出venustas问题。

5.9 对构造角色的举例说明

当客户决定要修建一个家时，他们的脑子里常常充满了极其主观的愿望、幻想和要求。一些客户非常懂技术，他们希望在窗户设计、电器设备和建筑方法上都采用最新的技术。其他人则在晨曦中坐在空旷地上，以期找到房屋承泽晨光的最佳方位。

不过，这是个不会亏本的赌博，很少有客户会对他们未来的排污系统赞美有加——如果他们有一点点的考虑过。不过，就我们的目的而言，这个排污系统是对建筑职业角色之间差别的很好的说明。同时，这也是个不言而喻的道理：如果整个系统失败的话，客户所有期望的特点将以失败告终。

像软件一样，排污系统可以对毫无经验的人形成威胁（呀！我

们的水和垃圾都放在空地上吗）。但是在一块土地同意被作为居住地出售之前，一个工程公司应该签署合同，以完成灵敏度测试和排污计划。

工程师要验证所提议的排污地区，并对该块土地的可行性进行证实。由于需要锄耕机来完成这个工程任务，那么就可以利用排污建筑公司。他们可以雇佣工程师，或是与他们签约。

反过来，在选址或决定房屋的大小之前，建筑师需要把排污地区考虑进去。如果这个排污场地经考证，只适合建造四个卧室的房屋，那么就不可以在这片地上建六个卧室的房子，除非将这块地扩大。或者，如果建筑师想将这个房子建在最初选址的地方，那么整个排污场地就得搬迁。工程师就需要重新评估要完成这个建筑所需要的技术，还要对改造进行证实并获得批准。

有趣的事就开始了。有这么一些致力于排污系统的专业公司。重型设备的操作人员挖壕沟，并将混凝土的槽放进壕沟内。建筑工人将管道放进混凝土槽，并将管道在槽内相互连接。电工来安放泵。

这一切完成后，由工程师来检查和测试整个系统，证明建造的一切与工程规划是一致的。

请注意，建筑师在此之前已经将这块排污场地的大概轮廓画在蓝图上了。客户同意并运行这个规划。反过来，市政工程师同意并运行由排污工程师所完成的一切细节。而这个排污工程师的规划是由排污建筑公司来执行的。

当一个房舍修建完毕，客户会毫无知觉的使用排污系统，他们只是看着房屋旁边的青草在蓬勃生长。

5.10　结论

在这个例子中，架构师并不想实施一个排污的系统工程。工程

师也与房屋的规划或设计无关，即使这个工程的结果影响了这些决定。当建筑工人修建这个系统时，操作重型机器或负责管道的人既不是建筑师也不是工程师。而客户是不知道如何建立排污系统以使整个进程行之有效的。

这些就是建筑中的不同角色。每一个职业都是角色分明，有着自己的专长和技能，然而彼此间又是相互依赖，对彼此的贡献又是 60
相互理解的。只有在这种情况下，建筑中所存在的固有的困难和变化才能够得到解决。只有在这种情况下，才可能有一个可以预测的进程和结果。

架构师、工程师、科学家、修建者——就连孩子们都知道软件是怎样构造的。

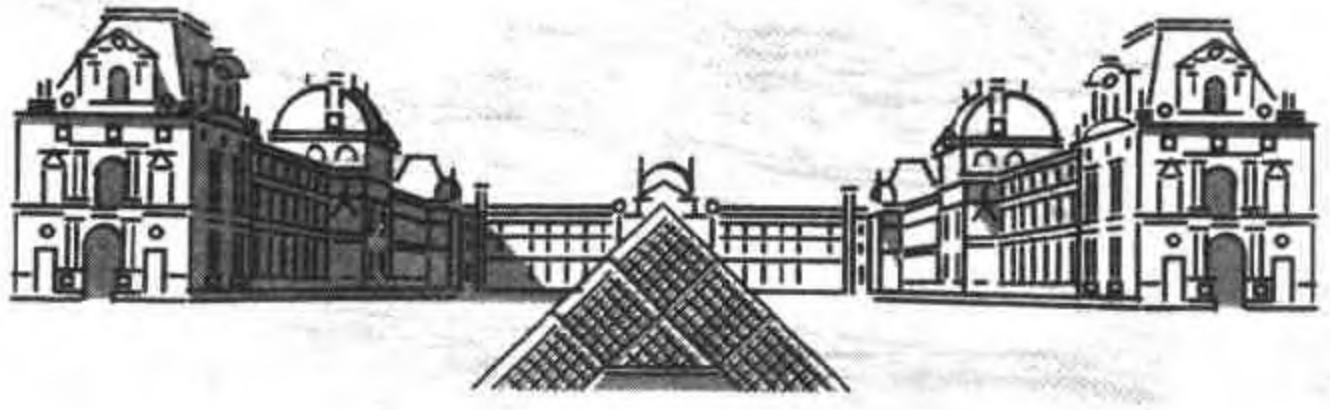
尾注

1. Frederick P. Brooks, *The Mythical Man-Month*, Anniversary ed. (Reading, Mass.: Addison-Wesley, 1995), 257.
2. Le Corbusier, *Toward a New Architecture* (Mineola, N.Y.: Dover Publications, 1986), 1.
3. Steve McConnell, *After the Goldrush: Creating a True Profession of Software Engineering* (Redmond, Wash.: Microsoft Press, 1999), 155.
4. Ibid., 59.
5. Alan Cooper, *The Inmates Are Running the Asylum* (Indianapolis, Ind.: SAMS, 1999), 15.
6. Geoffrey Chaucer, *The Parliament of Fowls* [1380–1386], in *Familiar Quotations* 16th edition, ed. Justin Kaplan (Boston: Little, Brown, & Co., 1992), 128.
7. William Shakespeare, *Hamlet*, in *Familiar Quotations*, 16th edition, ed. Justin Kaplan (Boston: Little, Brown, & Co., 1992), 194.
8. Terry Winograd, *Bringing Design to Software* (Reading, Mass.: Addison-Wesley, 1996), xvii.
9. Winston Churchill, *Time Magazine*, 1960, in *The Theory of Architecture: Concepts, Themes, & Practices*, Paul-Alan Johnson (New York: Van Nostrand Reinhold, 1994), 269.

10 Christopher Alexander, *The Timeless Way of Building* (New York: Oxford University Press, 1979), 233.

11. Christopher Alexander, in *Patterns of Software: Tales from the Software Community*, Richard P Gabriel (New York: Oxford University Press, 1996), vi.

61



第6章　软件架构师的角色

“架构是一种手段，它的核心功能是影响到人们的喜好。”[1]

——James M.Fitch

“最重要的行为是把某个人的想法在产品架构中实现，这个人负责把用户对产品的所有需求进行概念集成。”[2]

——Frederic P. Brooks, Jr.

虽然千百年来，建筑设计师的形象和特征发生了巨大的变化，但是无论他们是无名的工匠还是超级明星，他们扮演的角色都是一样的，而且始终如一。建筑设计师的角色，可以生动地用utilitas、venustas、firmitas构成的维特鲁威三和弦来表征，这已是千百年来不变的真理，并且将继续指导我们建立起软件架构师这个职业。

架构师的最基本角色就是设计。架构师的设计是一个创造过程，有时还具有神秘色彩。其最高成果就是提交一个结构构造的规划，无论它是一个建筑、或一台机器、或一艘轮船、或一个软件系统或软件产品。所谓设计，就是通过建立结构——firmitas，把客户需求合为一体——utilitas，实现venustas。

而架构师的核心角色是沟通。架构师是催化剂，牢固地深入两个领域：客户的领域和建设者的领域。架构师必须做好沟通工作，他要深入客户领域以获得最佳设计，然后再通过规划或蓝图把设计信息足够详尽地传达给建设者。

因此，架构师的基本角色就是面向客户的利益需求做结构设计。这听起来简单。但对架构师来说怎么才能完成这个角色？他怎么才能完成一个好的的设计？作为一个架构师，他又如何扮演把设计转化成实现的指导角色？

6.1　架构师的角色始于客户

架构师的角色就是为满足客户的需求而做结构设计。所谓满足客户需求，即理解客户需求和通过设计满足需求，这既是一门艺术，也是一门科学，需要架构师经过多年的培训和设计经历才能体验到。设计过程绝不是纸上谈兵式的需求搜集练习，即往往理想化地认为软件系统的客户和用户能够完全清楚地估计他们的需求，知道应有的设计要素，而且准确地把它们用名词术语表达出来。

如果我们准确地知道什么是需求，知道如何把它们综合到一个完美的设计中去——一个软件系统中或我们梦想中的家园——那就不需要架构师了。这犹如让居民能简单地把他们对房屋的构想直接传达给建筑工人，并在最终获得了与其构想一样的房屋结构。众所周知，我们现在的软件就是这样打造的，这也正是软件业失败的原因。实际上，这个假设的前提是根本不成立的。

6.2　架构师是客户的代言人，是设计的领导者

现在绝大多数关于软件架构的书在讨论客户问题上都是失败的。虽然有些书涉及到终端用户、风险承担人等，但忽略了架构师与客户关系这个关键问题。就构架这个学科的原旨而言，独立的架构师与其客户间扮演最重要的角色，已经形成了设计与构造过程的里程碑。架构师不仅是、而且必须是客户的代言人。

客户可以是形形色色的人，如一个委员会，特别是当公共建设项目中把各种软件项目都集中在一起的时候。在一些机构和实践中，采取多层次的架构师——通常是副架构师向主架构师负责。无论架

构师的身份是什么，他们的职责都是向客户负责，根据客户的需要做出所有决定。

要想让前面这句话成为现实，通常需要架构师具有独立于建筑者的专业技能，特别是在复杂而且投入大的项目中，这里充斥着大量矛盾的需求。当然有些既擅长架构又擅长设计的杰出人才，但他们常会出于效费的考虑而改变设计初衷。抄近路的做法，往往会破坏构架的纯洁性。“无以言表的品质”虽然不至于弄错，但当设计并非构架的主要考虑因素时就显得脆弱并走向失败。

下面举例说明架构师作为客户代言人和设计领导者的关键角色。

Robert A.M.Stern是美国一位著名的建筑设计师，把门廊和阳台描绘成无比优美的空间。像所有优秀的建筑设计师一样，他非常精于此道。他的美国梦幻之家样板的蓝图，是1994年为生活杂志设计的，起先阳台是设计成由外观涂白的红杉木圆柱支撑，柱体呈锥状，由下向上逐渐变细显得很雅致，与前门厅一致。

但是样板房的建造工人很快发现锥状柱子每根要花1 000美元，而一根上下一般粗的柱子只需不过几百美元。价格差距怎么这么大？他犹豫再三，最终选择了后者，这样省下了好几千美元。这样做纯粹是为了降低成本。

幸运的是，由Robert Stern最后决定所有设计。这些柱子全用错了，他把它们全拔了出来，这使建筑工人和木匠们惊惶失色。合适的柱子被立了起来，设计的完整性和无与伦比的美学效果被复原。

今天房子的主人们及他们的孩子们、他们的客人们，坐在这个阳台上，度过南方闷热的夏夜，周围点起蜡烛，谈话到清晨。他们说不出为什么对空间感到这么舒适，至少他们感到这12个古典大柱子非常安全，就好像融于一体一样，而并非刻意所为。总之，他们

就是感到空间很舒适。

如果Robert Stern，或是面对同样情况的其他建筑设计师，是为建筑公司服务的话，那么这些栩栩如生的柱子将不复存在，它们将被便宜的柱子所取代，因为建筑设计师不得不服从公司管理层的决定。今天的房主和居住者将不再拥有这些优美的柱子。这真的要归功于设计、经费和所属关系之间的内部利益较量。

65

要成为真正的代言人，还需要建筑设计师掌握更广泛的技能，以便可以从更大的选择范围中精选出最好的设计方案。但是如果预先用固定的技术、工具、方法把建筑设计师束缚住，这样只会缩小适用于客户的解决办法、创新性和设计策略范围，那么他又如何能够完成其代言人的角色呢?

在建筑设计师的职业生涯中还会形成个人的构架风格和偏好，它们会对客户产生影响。但对头脑清晰的建筑设计师来说则是锦上添花，而不受限于他的技能、教育和经历。一个泥瓦匠永远也成不了建筑师，你给他一把凿子，他给你凿出的都像是石头。

那么，一个架构师——客户的代言人，是如何完成其角色的呢？也就是说，他怎么做出最好的设计呢?

6.3　倾听的艺术

建筑师们把大部分时间花在架构前面的工作上：倾听客户和其他将使用建筑的人及居住者的意见。倾听是一门艺术。我们总会遇到这样一些人，他们总是自己说，而不倾听别人，他们不愿意也不可能进入我们的领域。他们总是回避一个事实，就是他们没有耐心，总以自己的想法为中心。他们不倾听我们，甚至鹦鹉学舌般地把我们的话还给我们。

如果架构师不能倾听客户并进入客户的领域，那他的设计将达不到客户的期望。架构师通过倾听，了解客户的资源和需求、特殊

的困难、偏好、企业的心理和商务氛围等等。对于结构的使用者也需要应用相同的的倾听技巧，从而考虑整个规划的现实性。通过倾听，架构师将了解如何将客户与组织结构统一起来，知道如何通过架构来应用解决方案。

除了所涉及的企业外，架构师还需要了解客户全部的背景和环境。企业关键的成功因素是非常重要的，架构师需要通过倾听来理解雇主和职员们是如何被激励和考核的——这通常与一般的行为准则有很大不同。

66

一个架构师的设计如果不能把上述这些因素有效考虑进去，那么他会背离客户和用户的要求，这就好像建的房子不符合居住者的行为方式及其品味习惯。

6.4 观察的艺术

除了倾听，架构师还必须会观察。一个被雇来设计软件，跟踪与控制大型生产评估的架构师应当熟悉设备的整个生命期，并向所有可能的用户咨询。从而架构师掌握隐藏在客户、管理人员和职员们所陈述需求背后的现实情况。

架构师必须“通过四处观察来进行设计”。FAA和IBM犯了个关键错误，没有让软件设计人员进入空中交通控制工作站。其实，如果让架构师那双受过训练的眼睛看到空中交通控制工作站的实际情况，如果让架构师穿着控制员的制式鞋在工作站里走上一圈，他们肯定会发现控制员的注意力时刻都不能离开雷达显示屏的事实，他们就不会偏离空中交通控制的固有特征。也许项目经理担心真实的观察会影响架构师的想像力，但最后，缺乏观察将降低设计的可行性。

6.5 策略的艺术

设计过程开始于架构师最初的想法，而后随着对实际情况了解

的不断加深，否定初始的想法，建立起新的想法，进行进一步精化。当架构师掌握所有该知道的内容时，设计开始与客户结合，并形成整体。其中非常重要的部分来自架构师的策略。架构师要想如何策略地把设计与客户和用户的喜好一致起来。从这个角度来说，软件架构师远不只是一个把需求打包到软件结构的角色，他利用自己的技术、重组织和调整来发现机会，从而具有改进企业的潜质。

在策略层面的观察与思考能力是每一个架构师需要努力具备的。这样架构师就能跳出狭窄的牢笼，建立起改善企业各个方面的创新性软件，或者建立改善用户应用的软件产品。这种能力是可以培养的，但不同的人情况也各不相同。正如贝聿铭的天分超越了大部分其他建筑设计师一样。但这并不意味着软件架构师没有硕士文凭和设计天分就不能建立很好的、符合客户需求的软件系统或基于软件的产品。以此类推到建筑业架构上，虽然只有几个闻名遐尔的设计师，但所有设计师都努力把自己的策略优势投入到卓越的设计上，让客户满意。

67

这样，架构师的角色就是要尽可能地整合客户与企业知识，在倾听——观察——设计——倾听——观察——设计的循环中点燃智慧的火花，按客户的喜好策略地运用技术。这就是设计的目标，无论设计一个庞大的公司软件系统，还是以软件为基础的应用系统，还是介于两者间的任何系统。只有当架构师做到下面的事，就能很好地履行其职责：

- 精通倾听、询问、观察的艺术。
- 掌握足够的客户领域知识，如银行、政府、教育、卫生健康、零售或赛马等等。
- 建立对客户企业的战略观，而不仅仅是战术或操作层面上的。
- 拥有广泛的技术知识，这样能在架构规划中进行全方位的策略选择。

- 与客户与建设者进行有效的沟通
- 监控、审查、维护客户的观点与设计

6.6 巴黎的金字塔

建筑设计师既要接受成功设计所带来的赞美，也必须承受失败设计所引发争议的煎熬。

贝聿铭使巴黎的卢浮宫焕然一新（增加了几个金字塔）的故事，是一个如何应用职业特点才能应对严峻挑战的研究案例，他完美地体现了复杂的设计师角色。

故事起自新上台的社会党，密特朗宣读总统令，决定由贝聿铭设计伟大项目——卢浮宫工程（一个奔放的法国建筑，让人意识到高卢文艺的复兴）。社会党人非常热衷于传统的骄傲，密特朗总统相信自己有这个权利。卢浮宫成了首批不拿出去投标的项目。因此， 68
起初公众对此一无所知。

从某种程度说，贝聿铭的客户不只是密特朗总统，而且还有法国人民、法国的历史，甚至是卢浮宫本身，这个建筑已经为其名声所累。虽然巴黎人把卢浮宫看作是他们最伟大历史遗产之一，但是参观博物馆的游客中只有三分之一是法国人，更只有十分之一的人是巴黎人。观光客发现他们花在卢浮宫游玩的时间平均只有1.5小时——只是在纽约大都市博物馆一半的时间，而他们的期望很快就消失了。

“当密特朗总统最初要求我做这项工程时，我真不敢相信。这对法国是多么重要的事，让一个美国人来做，这简直不可思议。我告诉总统我对此深感荣幸，但我不能立刻答应。我请求他能否给我四个月的时间，不是考虑是否接受——其实我已决定我要做——但确实想看一看我能不能做好。”[3]

贝聿铭没有把这事告诉他的同事，只告诉他的妻子——Eileen。他秘密地到巴黎做了三次旅行，研究卢浮宫的建筑风格和周围环境，他不停地走好几个小时，沉思、分析、策划。贝聿铭总结了卢浮宫存在几个大问题：

“建造卢浮宫的目的在于让国王居住，这样它永远也不可能真正成为一个博物馆……。因为你很容易在里面走失——你不知道东西都在哪儿，而且没有厕所、餐馆等等服务性设施，虽然你的确有时间参观卢浮宫。

当你想建造一个博物馆，其50%的展示空间需要与其他50%的辅助空间相匹配——古典精致的图书馆、餐馆、礼堂、演讲厅、公众接待区、厕所以及其他设施，而这一切卢浮宫都不具备。很少很小的厕所，实际上，我记得很清楚，当我经常去那里参观的时候，通常我不想在卢浮宫久留。而当游客离开卢浮宫时，很多的人就不会再来了。卢浮宫失去了很多游客。

卢浮宫博物馆仅仅成了卢浮宫的一个“房客”，仅此而已。它沿着塞纳河蜿蜒而居，大约800米长，要从一头走到另一头必须要上下楼梯。结果，像我这样的很多游客到卢浮宫，只看到25%的内容，而错过了其他部分。”[4]

69

游客最经常问的问题是：“我们怎么到那儿？”

卢浮宫呈巨大的U字形状，拥有巨大的卡尔·拿破仑庭院。庄严的庭院中由沙砾铺成停车场，租用给卢浮宫的另一个“房客”——法国内阁。现在对贝聿铭来说就很清楚了，庭院是地理中心。显然，法国内阁需要让出庭院，空出这条美丽的长廊。

贝聿铭起初没有想到金字塔，但他建议在庭院中央建立一个新入口通向地下大厅。密特朗总统欣然接受。

但法国人民不高兴了。当宣布贝聿铭作为首席设计师时，民众一片哗然。排外的法国人已经在平民文化上输给了美国麦当劳，他

们对选择美国人来改变他们民族珍品感到沮丧，虽然贝聿铭已经为华盛顿的国家艺术馆做出了成功的设计，而且他还是华裔美国人。他说：

> 我从不为自己是个华裔美国人而感到受伤害。大家看到，历史对法国是多么重要，而我希望让法国人相信我来自一个具有悠久历史的国度，我不会对此掉以轻心而会严肃对待。[5]

当然，法国人还是很尊敬密特朗总统，他们刚刚选出的：他们的自由、平等和友善的领袖。

回到曼哈顿，贝聿铭带着精选出来的助手，组成小组开始设计一个占地5英亩的地下石灰建筑，包括一个礼堂、储藏室、会议厅、咨询台、豪华咖啡厅、书店和一个输送艺术品的电动车道。

新入口起着轴心的作用，由此分别向三个大厅辐射，就像三个翅膀。游客只需走不到100英尺就能到达某个翅膀。第四个大厅通向时髦的购物区（这太资本主义：引起一场巴黎人新的论战）。新的大卢浮宫拥有165间房子，使之成为世界上最大的博物馆。这样，馆长就可以展出那些多年来被遗忘在阴暗角落里的70 000件艺术品。

金字塔被公开后，法国人感到震惊。他们一直把卡尔·拿破仑庭院看作一片庄严的土地（现在不过是白天的停车场，晚上的散步场所）。他们的态度更多的是一种反射性的，而不是理智的。大概是法国人对几乎所有事物一开始时都会说“不”字这种传统的一种延续。

如果法国人认真想一想，他们可能会意识到卢浮宫已经有800年的历史，而每个时代都留下了印记。最初，这里不过是一片森林和地牢，而后拆了建起辉煌的皇宫。16世纪，它经历了文艺复兴，两代路易国王为它扩展了大片的土地。1793年，其中一部分被用作博物馆，在拿破仑三世登基前已经给它增加了很多附

属品。

“当Eugenie皇后问道：‘但这是什么风格？既不像路易十四，也不像路易十五，也不像路易十六’，歌剧院设计师Charles Garnier自信地说：‘陛下，这是拿破仑三世的风格，我们还有什么可抱怨的呢？’”[6]

——Elizabeth de Farcy 与Frederic Morvan

就历史而言，卢浮宫的结构从来就不是固定不变的，领导人无所顾及地在它上面刻上时代的烙印。如果他们害怕的话，卢浮宫仍然只是个巨大的中世纪森林。历史事实最终使法国人不再反对对他们的宝贵遗产做任何改变。

贝聿铭发现不得不破坏庄严的庭院。他说：

“卡尔·拿破仑庭院是地理中心、公众都要来这儿。当你来这儿的时候，你要做什么？你可以进入地下空间吗，就像地铁一样？不，你需要更大的空间。它能让你意识到这个时代的特征。这个空间容量必须大，它必须有灯光，而且有入口标志。你看到它就会说：‘啊，这是入口！’”[7]

贝聿铭的70级金字塔入口是根据埃及大金字塔模式设计的。

“卢浮宫金字塔是卓越的技术成就。它高70英尺，坐落在100英尺的方形基座上，由793块钻石形和三角形玻璃嵌在一起构成，精确地建筑在铝制框架上，由93.5吨的大梁和不锈钢骨架支撑着。”[8]

——Elizabeth de Farcy 与Frederic Morvan

71

在卢浮宫金字塔的四周，有三个小“金字塔”和三个喷水池。在购物区有一个玻璃做的倒金字塔延伸至地下，就像倒挂在地面上一样。其他诸如楼梯、窗户、天花板等结构都采用相同的技术方法和细节，都在破土动工之前仔细设计和测试出来。

贝聿铭相信没有一个固体的附加建筑能替代旧的卢浮宫景貌，但玻璃结构可以反射景貌而不抢眼。而且，巴黎本来就充满了金字塔、几何体等抽象建筑。最后，金字塔座落在卢浮宫北边的外围，贝聿铭坚持采用金字塔形状只是因为它是正确的设计，而不是模仿巴黎的其他建筑。

1665年，贝尔尼尼想用豪华的意大利风格刷新卢浮宫这一皇家宫殿正面，法国人对此反映强烈，迫使路易十四重新召集设计师们再做规划，甚至在动工后还在修改规划。最后，几个法国设计师取代了贝尔尼尼，国王也把精力集中到凡尔赛宫的建筑上去。但密特朗总统公开向贝聿铭保证：

在贝尔尼尼身上发生的事情决不会在你身上重演。[9]

贝聿铭于1984年1月23日向顾问委员会提出了设计方案，这是一个具有历史意义的时刻。贝聿铭回忆道：

他们一个又一个地站起来反驳我的项目计划，我的翻译被吓坏了，在那儿发抖。她已经不可能在我捍卫自己意见的时候为我翻译了。[10]

贝聿铭和他的小组沮丧地来到小酒馆借酒消愁。幸运的是，委员会并没有被赋予权力，而密特朗总统的态度是对设计无条件的支持。贝聿铭总结道：

只要把一个人搞定就行了。[11]

但事情没那么简单。当拿出金字塔方案的时候，法国人大声惊呼："这太可怕了！"卢浮宫油画馆馆长把它比作是"俗气的钻石"。博物馆总监因此愤然辞职。一家名气很大的杂志把它的反射特点比做是达拉斯的蹩脚建筑。*Toutle Paris*掀起了反贝聿铭的大众情绪，刊登大幅标题质问："把金字塔淋个落汤鸡？"贝聿铭的女儿亲眼目睹妇女向她父亲脚下吐口水。

贝聿铭保持镇定，当然，最后他胜利了。这项工程的一位年轻设计员Mihai Radu，这样总结道：

“我从未感到他失去信心或表现沮丧。他认识到让人们理解他的作品是他份内工作的一部分。他是一个高品质的人。我看到贝聿铭先生在任何时候都保持微笑，即便在那种最艰难的时候他还在微笑。”[12]

从1983年～1989年是第一阶段，即设计与最初建设期。贝聿铭描述道：

我们浪费了两年时间。我不得不去应对媒体。我事先没有完全的心理准备。我的法语功底不能充分地表达我的目的……18个月里只有苦恼　看到的全都是人们的示威　他们并不是真正地严肃对待这件事，我好像要崩溃了。确实，我已经快到那个地步了。结果，在第一阶段我们花了六年时间，只有四年投入到架构和建筑上。[13]

当设计师深入到架构的重要细节中去的时候，他还需要精通如何选择正确的技术和供应商来实现设计。贝聿铭在这方面很聪明：

法国政府在这方面是坦诚的，那就是所有的东西都要用法国的。但法国承包商真是难以合作。就拿玻璃问题说吧，你看，玻璃要求很光洁，而它们又得很厚，大约3/4英寸厚。如果不纯净，玻璃就会呈现绿色和深绿色。但卢浮宫不能被颜色挡住，而且遏色的石头不能与绿色的玻璃相配，所以绿色不能被采用。结果，当我要求一位法国企业来做玻璃时，他们告诉我：“不行，我们做不了”。最后他们说：“这样，你造1000座金字塔，我们就给你做”。我没有向总统汇报此事，我去找了家德国公司，我问他们：“你们能做吗”，他们说：“可以”。而后，法国公司也说：“这样的话，我们可以做。”[14]

建筑设计师需要知道何时保持传统，何时进行创新。创新必然

要带来挑战，建筑设计师必须能对风险做出估计，并决定在项目中做多少创新。建筑设计师必须确认结构能经得起推敲。在这方面，甚至采用金字塔这么清晰简明的结构都会招来狂风暴雨。但这个问题的解决最终证明了贝聿铭的丰富经验和解决困难的能力。

很长一段时间大家争论一个问题，那就是“你能把玻璃弄干净吗”。结果有不同的方法。他们从加拿大雇了些印度人，他们甚至想用机器人！我们居然要用机器人！最后，我们让人站到顶上，从金字塔尖引根绳子下来，人挂在绳子上洗刷玻璃。结果我们只用了一天，清洗工作就做好了。不过，这个问题确实争执了很久，法国人为了反对金字塔计划竟然什么理由都提。清洗金字塔的事耗费了我们很大的精力。[15]

73

第一阶段我主要设计和构造金字塔670 000平方英尺的新地下广场。第二阶段主要把法国内阁办公室占据的广阔长廊区转变成三层展厅，足有370 000平英尺。有了这么大的空间，卢浮宫成了世界上最大的博物馆。

建筑对设计师最大的挑战就是在历史的围墙间创立新的空间，包括被用作停车场的三个内部庭院。贝聿铭把停车场变成露天的雕塑区，他要处理的难点在于满足油画收藏区的客户需求。

法国的油画收藏可能是世界收藏品中最重要的部分。不是可能，实际上就是，没有油画，就没人去那儿。法国管理员希望它们在楼上，这样能在阳光下欣赏。他们非常坚持要有阳光，于是他们要了顶楼。要爬到顶楼，你得直上直下75英尺，这样的话，很多人就不会到那里去了。结果，他们就会失去很多观光客。我了解到大约只会有10%～15%的人上到顶上，这太令人悲哀了。这样，我就建议用电梯。其实，我不愿用电梯，因为这是19世纪的东西，如果不是特别的理由就不要加进来。这艰难的仗终于打赢了。现在，没人反对它的必要性，因为在这样的垂直高度间往返实在是太困难了。[16]

贝聿铭在让客户满意的构架设计方面表现了卓越才能，甚至面对挑剔的法国人，都能以非凡的方式解决旧博物馆的照明问题。而后他又充分地考虑了所有影响设计的因素，包括法国管理员的文化背景。他的创造性根植于他的实际解决问题的态度。

油画的照明是非常重要的——在这之前，这个问题一直未得到充分的关注。我们在美国的展览馆就没有很好的日光照明……。自然光很明亮。照进来时天花板最亮，地板其次。而你所希望照亮的是那些挂画的墙，但它们很暗。这样，我们就做了个研究。我认为画廊的一个突破点在于它的照明。解放方案是：把天花板做成三层。第一层是玻璃的，阳光能通过紫外线过滤后射进来。玻璃下面是一个蛋形的篓子。通过方位的精确计算以使阳光不能直射进来。（要是那时我们把它做成个可移动的就更好了。因为那样光线整年都会很好，而且不受季节影响。不幸的是，富有经验的法国管理员说，“这起不到好效果。”结果他们获胜了，后来我们把它做成固定式的，不需要动它。）阳光反射到墙上，而不是直射到地上。这样墙就接收到阳光，变得很亮了。我确信就此不再有反对意见了。而保守的法国人——确实非常保守——终于接受了它，现在他们称这是世界上最好的。他们非常喜欢。[17]

74

经过设计与构造，贝聿铭成了客户领域和设计领域的头。他对自己的设计充满自信，他即便在面对公众压力的情况下也不屈服，不放弃自己的观点。他一脚站在客户世界，一脚站在建筑工人的世界。他有力地控制着设计。他证明了设计师是真正的领导角色。他说：

就在卢浮宫建成200年后，1993年11月Richelieu Wing开放了。它与拿破仑庭院浑然一体。现在，卢浮宫正如我们设计的那样正常运转。如果你现在去那儿，你就会明白为什么要在那里建金字塔。当整个博物馆竣工时，整个规划被证明是正确的。你

可以乘三套电梯到中间层，而你仍在地下，你可以进入卢浮宫的三个翼。第四个翼通向购物区、停车场、公共汽车站。过去你看到的所有公共汽车现在都在地下了。这对大城市来说是一大贡献。在这一层，我们建礼堂、旅馆、接待区等。我们还有大型书店、商店、会议厅和会议中心——这里具备一切。下面一层是流通层，有一个卡车道连接所有地下单元，还有一个巨大的储藏库能存放卢浮宫几乎所有的收藏品。现在卢浮宫的物品都可以回到家里来了。[18]

6.7 结论

博物馆开放的那天，法国人对新金字塔做出最终的反应：

……金字塔揭幕的那天，惊喜的巴黎人驱车前往，他们不得不承认这个摩天架构确实给卢浮宫的周围添色不少。他们把目光聚焦在庭院和透过玻璃得到的新景观。[19]

——Alec Lobrano 75

在尖刻而严厉的批评之后——他们爱上这个建筑。这个故事告诉我们，一旦涉及到文化，要做一些变化就会受到来自多个方面的抵制，即使现状已经是非常不舒适、不灵活和低效益的。我们退一步说，建一个玻璃的现代金字塔确实是疯了，但充满自信的贝聿铭和他光芒四射的设计征服了民众，甚至先前的批评者也交口称赞。现在，如果没有位于中心的建筑之光，人们都不敢想像卢浮宫会是什么样。

贝聿铭和金字塔的故事卓越超群。他非凡的作品成为设计师角色、设计的内在挑战和构造过程的典范。这个故事给软件架构师们上了一堂生动的课，他们只有像贝聿铭那样具有非凡的分析与设计能力，激励自己去实现自己的角色，才能创造奇迹。

贝聿铭的故事提出了这样一个问题：如果卢浮宫是一个新软件

系统，没有优秀的架构师结果将是什么样？要解决人口问题，需要在卢浮宫的侧面或后面建一个大型广场。要是这样的话，技术上能“满足要求”吗？这不类似于现在很多网站不按客户业务需要，给人家设了众多“前门（Storefront）”？

当然，有时快速网站在战术上确实是一个好的解决方案，特别是在时间就是市场，时间就是效益的背景下。但是对架构师来说，他要站在整个企业的大背景下做最终的、紧密一致的策略设计。而一旦脱离更大的背景把制作网站的团队拉在一起，只能做出零碎的设计，就好象在卢浮宫前面放了一个大盒子。

现在，像盒子式的卢浮宫入口，即便被设计成与古老的建筑相匹配，也还会让观光客交了钱却什么也看不到。它只能说是技术上符合使用要求。建一些浴池凉亭可以满足上厕所的需要，还可以避免一大堆意见。

但是那些博物馆低层次的“终端用户”，即少数观光客仍然会对参观提不起兴致，甚至提出异议。没有优秀的架构师，虽然没有那么多争议和责备，但是最后也没有荣耀。

贝聿铭给我们展示了架构师的角色，囊括了所有要素和规则，如收集需求、现场分析、项目规划、工程实施、子项目承包等等。但是当架构师设计出的作品超出本身各部分总和时，他就可能是搜集了过多的元素了。这就是真实的设计，这才是构架。

76

尾注

1. James M. Fitch, “The Aesthetics of Function,” in *People and Buildings*. ed Robert Gutman (New York: Basic Books, 1972), 9.
2. Frederick P. Brooks, *The Mythical Man-Month*, Anniversary ed. (Reading, Mass.: Addison-Wesley, 1995), 256.
3. I. M Pei, in *The Mandarin of Modernism*, Michael T. Cannell (New York: Crown Publishing, 1995).

4. I. M. Pei, quoted from the MIT Technology Day Address. Transcribed by Pei, Cobb, Freed & Partners and published on the Internet at www.algonet.sepw (no copyright).

5 Michael T Cannell, Ibid.

6. Elisabeth de Farcy and Frederic Morvan, editors, *The Louvre* (New York: Alfred A. Knopf, Inc., 1995), 83.

7. Michael T Cannell, Ibid.

8. Elisabeth de Farcy, Ibid., 86.

9. Francois Mitterand, quoted in Michael T. Cannell, Ibid.

10. Michael T. Cannell, Ibid

11. Ibid.

12. Ibid.

13. I. M. Pei, MIT Technology Day Address, Ibid.

14. Ibid.

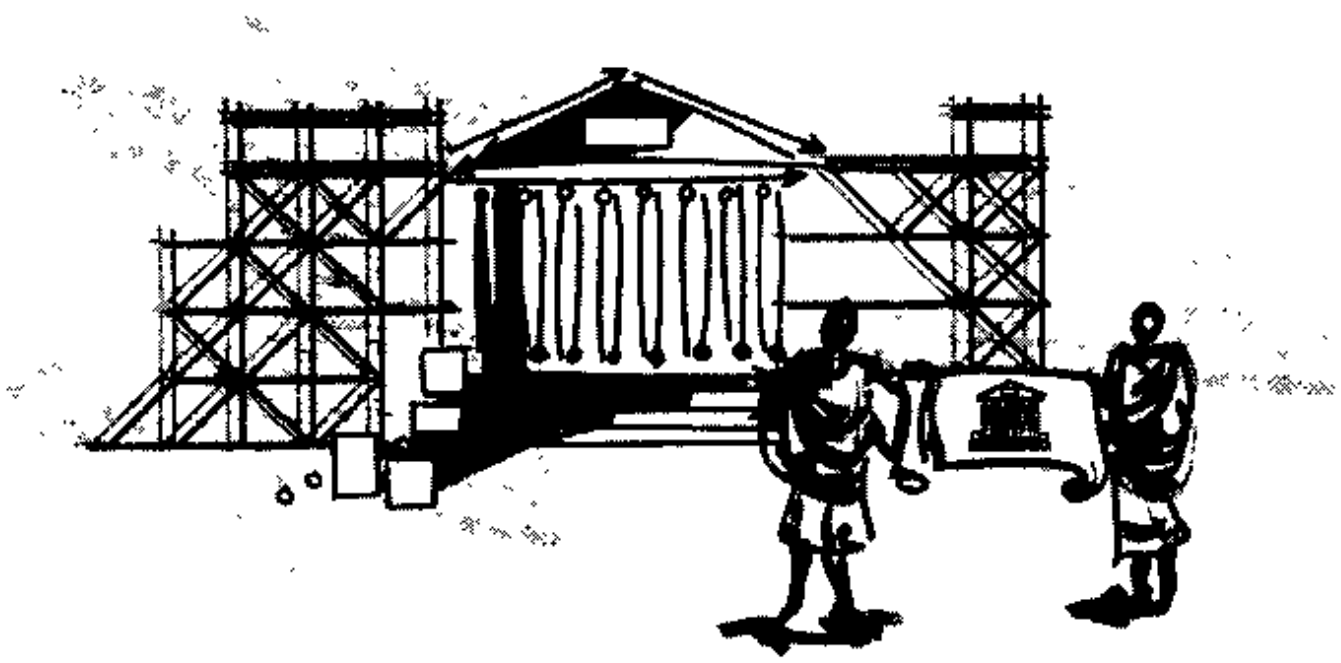
15. Ibid.

16. Ibid.

17. Ibid

18. Ibid.

19. Alec Lobrano, *Irreverent Guide to Paris* (Foster City, Calif.: IDG Books Worldwide, Inc., 2000), 99.



第7章　架构驱动的软件——构造阶段

“排序和简化是掌握和支配问题的第一步——真正的敌人就是无知。”[1]

——Thomas Mann

“我弄清了一小部分，我就掌握了这一小部分。”[2]

——Sir William Schwenck Gilbert

值得关注的是，本章的标题是关于软件构造的阶段，不是软件架构的阶段。该过程的要点是构造什么东西，而不只是设计它。当然，构造过程是由构架来驱动的，这是因为最初的工作是在架构师与客户间进行的，而后的工作都是遵照架构规划来的。

7.1　两个大致阶段

所有项目构建都可以分为两个阶段：设计或架构阶段，以及构造阶段。当然，还可以根据项目的大小和复杂度来划分成几个部分。例如，开发大型居住工程，需要对道路、子单元、设施、公众娱乐场所做大的设计布局，如网球场、游泳池、公园和篮球场。

构造阶段将按照架构建成所有东西，诸如建公路、安装设施、清洗场地，并可能还要建公众娱乐场所。

同样的道理，每栋房子都有自己的设计与架构阶段——通常会有不同的客户、建筑设计师、建筑工人。但是，开发者都必须事先

规划好建筑材料以确定足以满足建房需要。并且大多数情况下各种房子的大小、质量、风格设计规则都是一样的，甚至包括他们对木料的选择，这些都被认为是整体开发计划的一部分。在一些城市，建筑设计师审查委员会给出设计指南。在社区里，设计指南和合同会搞得非常仔细——细到规定好花盆的用料限制（如要求只许用自然的陶瓷制品）。甚至有一个加州社区还规定禁止灌溉系统把水流到人行道上。

即使在被出售前，房子的设计阶段还会受到其周边环境的影响，在这一点上，建筑设计师能做的不多。有时，房主只有三种房型选择——正如他们所说，就这样吧。

这完全像基于软件的技术结构一样。例如，一家大银行想重新设计一套软件，需要在每一个结构展开工作前，比如做一个贷款应用程序，要对其整个领域和范畴做出明确的定义和界定。全局性的设计问题，例如风险管理，都必须作为公约，每个具体应用都必须遵守。

对建造房屋来说其基本框架都是一样的。首先需要布置好基础设施，确定好各个单元间的关系，规划好各个结构体的大小和数量，以及外观、质量等，“公约”根据需要不断细化——如果需要的话，直到花盆的布置。

灵活性是无止境的。在纽约有一个社区都是劳埃德海上保险协会之家。虽然每套都不一样，但总体风格都是一样的。可以说，独具风格的优美建筑彼此相邻，但它们总能浑然一体。另一方面，在

80

佐治亚州的一个已经规划好的社区中，忘了给出可供户主选择篱笆的类型，结果带来了一连串的后果——白桩子加雪松，招来一片埋怨，太恐怖了。

关键在于每项工程，无论是砌砖造房还是计算机编码，都有其特定的设计和构造阶段。按照每个项目的范围界定，这其中有各种嵌套层和中间层要设计和构造。与这个例子相似的是，在尚未考虑

到更大的背景情况下，矗起篱笆是使各单元最佳比邻的方式。

7.2 架构阶段以及一些警示

是的，架构师就是负责设计的。但是他们需要了解在架构各阶段、完成设计蓝图或完成建模之外更多的东西。客户、架构师和建造者——在他们的视野中都有一个共同的目的：成功地完成建筑。其惟一的关键点就是要建造个东西让人们使用。

7.3 设计是不可交付的

由于软件业并不是伴随架构职业的发展而发展的，也不是同建筑业清晰的角色和内在的过程一同进行的，所以有时我们的关注点会有所偏离，甚至把设计和需求文档看作是可交付的。其结果，为了按期完成项目，可能浪费大量的时间，而这样又会招致一些批评意见，如认为在架构上花费太大，浪费了时间。

在建立架构方面，客户既热衷于要做出正确的设计，又非常渴望尽早动工。计划的关键在于能否在建造前充分地进行交流，而在当时情况下我们又需要尽快地完成构造。对此，我们可以应用诸如面向对象的设计技术、确定对象及其属性、行为都是设计的一部分，这样做就可以提高执行的效率。

完成了的建筑是可交付的。在此之前所有环节都只是一个阶段、一个步骤或一个里程碑。同样需要注意的是，在软件业里容易犯优先顺序倒置的事。

阶段是重要的、有时对建造的投资是随着特定阶段而定的，如文档设计、建立框架、封装、安墙和获得居住证书等。这对建筑业和软件业来说都是一样的。进程需要按目标测量。但在户主入住以后，房子的设计蓝图总是被束之高阁。在那里它们毫无用处——上面落满灰尘并长了霉，直到孩子们翻箱捣柜地找杂物时

才能被碰到。

考虑到这些以及后续的告诫，下面给出了前四个架构阶段。整个阶段列表是对架构和构造各阶段的放大，它们由美国建筑师联合会总结并实践的

第一阶段：预设计阶段

在这个初始阶段，设计师聆听并掌握客户的需要和期望，以及项目的广度和大小。客户的设计要点、陈述出来的需求和优先选择在这里得到评估。架构师仔细研究项目的背景——项目只是企业全局中的一个部分。使用者的需要和文化，以及他们的行为方式，这些都需要通过聆听、观察、阅读、询问来获得。正如客户要解决的问题是被限定的一样，客户的资源包括现有的财政和智力资本也都是被限定的。架构师由此开始制定策略——通览现有的技术、组织管理和过程的变化来确定可能的解决方案。简单地说，架构师要问：怎样把各种形式的技术综合起来，以便能最好地实现客户的需求？其结果是一个设计思路开始成型。这就是通过架构师和客户的合作、描绘草图、交谈和精炼他们的理解，直到出现一个共同的视图。在大量反反复复的交换意见基础上，建立起一个广域的预算和目标规划。

第二阶段：领域分析阶段

架构师的任务是理解所建系统所在的领域，制定领域文档，尽可能详尽地掌握客户的期望与需求。系统所应具有的行为（为使用者提供的服务）被描述出来。架构师评估客户的业务和技术环境，以及它们与项目广度之间的相互影响。在这里，领域名词术语和概念被准确定义。

软件架构师可能在某一具体领域具有专长，也可以请该领域的专家。这点与建筑业类似，有的设计师在某类建筑上是行家，如建造医院的设计专家或综合性住宅的设计专家，或者其合作者具有这

方面的知识。 82

第三阶段：概要设计阶段

在该阶段，描述领域特征和技术结构的架构级设计已经就绪，设计系统的外观与感觉——用户界面风格。如果需要，还可以建立原型。由于架构师和客户共同工作来确保所有环节都正确，因此，在这点上各阶段间的界线变得模糊了。在这里要做移植与风险评估。

第四阶段：设计开发阶段

现在，架构师继续深入细化和精炼设计，走向最后的设计。所有领域和技术设计图纸，经客户根据自身需要确认其期望是否被满足后，形成终稿。

7.4 设计阶段不是线性的

阶段列表、次序列表及其他各种列表的完成足以让大家心满意足了，但是我们却处于危险之中。“嗨，真好。”我们这样想着。“这就是项目的心脏，我完成了，这就是我需要做的，我可以提交定单了。”

但本章所述的设计阶段并不是线性的，甚至没有一个路线图。事实上，这里列出的设计阶段对架构师的活动指导有多少，客户获得的期望值就有多少。客户需要知道他们在参与设计步骤中担当了多么活跃的角色，而对各阶段的描述就此提供了一个逻辑和认知上的框架。对架构师和客户来说，这些设计阶段证实了我们何时完成工作，而不是告诉我们如何去开展工作。

由于设计是一个合作的、重复的、甚至神秘的过程，所以前面所述都是真实的。它充满了捉摸不定的现象、障碍和细节。新想法

总是在出乎预料的时候冒出来。

客户可能首先对建筑设计师说这样的话，“我想要在院子里铺石板地板，就像我们在缅因州的海岸边渡过的夜提时光。”当然，如果建筑设计师告诉客户，这要等到下一阶段做细化时再考虑，那就可笑了。建筑设计师应当把这个信息看作是磨上的谷物，因为它说明了客户的期望和品味。

在同一个会上客户还可能提及一本关于立体派风格建筑的书，
83
或关于对高大花板的爱好。建筑设计师需要聆听，并让客户不断说出自己的想法和爱好。然后，建筑设计师可以把这些想法和爱好带出客户的视野，提出设计概念。现在开始做最初的草图，在客户和建筑设计师之间形成相互反馈的循环。有些设计要点将从一开始就不断细化，而其他设计要点将一直保持模糊，直到最后的设计阶段才开始清晰。

客户抛出了一大块混杂一体的事实，要求架构具有技术性、而且经常需要有发明创新。这就需要在设计完成之前使创新工程化并建立样板。这可能会带来更大的失望、意外和连锁决策。

在建立架构过程中，花在工程和发明上的工作量是高度可变的。就像是贝聿铭的金字塔，需要精细的文档、工程、建模和研究。而第十个金字塔就省事多了。

其关键点在于能在设计阶段获取发明，并且发明应当是可管理的。尽管FAA项目在设计阶段花了四年的时间，但没有对发明建立样板。其结果甚至让项目人员都难以想像他们到底有没有完成最后的设计？如果这些发明不成功那将怎么办？他们所称的“设计”实际只是一个细化方法。

因此，在这个循环中，通过架构师对领域的理解，聆听客户的需要和期望，完成必要的发明和工程，相互合作不断反馈，最后一个共同的蓝图终于产生了，双方达成共识。

第五阶段：生成项目文档阶段

架构师关注那些确实对构造系统有用的需求。构造过程、团队成员的角色和构造次序都在这里被文档化。构造指南、用户界面风格指南、测试指南都被明确写出。架构师根据需要确定使用的工具和方法。那些向构造系统的人所提供的细节都在这个阶段完成。

架构师制定文档的详尽程度是根据项目的需要和建筑者的问题来决定的。如果架构师与某一建筑工人存在工作关系，并且知道他将为这个项目工作，那么架构师就可以按照他的需要裁剪计划。对于建筑者来说，他可能并不像其他人那样在高层架构设计上要求那么详尽。 84

另一方面，架构师可能需要根据具体情况做出很详细的低层次设计说明。例如，当建造一个横梁与柱子结构的房子时，一种称为“支撑外壳”的特殊模式被用作外墙。这就不需要在外结构上建立子墙了。在这种情况下，除非采取专门的隐藏技术，否则电线将暴露在外面。其结果，布线问题成了建筑设计师的重要问题，这是因为裸露的电线会影响整个架构设计。针对具体情况，决定在什么时候在什么方面制定多细的文档是架构师的职责。

第六阶段：人员安排与合约签订阶段

架构师为确定系统的实际建筑者提供支持。对于外包的项目，向外面的承包商发出投标书，并对潜在的参与者进行评估。架构师参与确定合约的细节并做出预算，经过排序比较后签署合同。

至此，严格的架构阶段宣告结束。但一般地说，架构师还将在后续的构造阶段中继续参与项目。

7.5 构建阶段

在这点上，焦点从设计阶段转向构造阶段。在这里，结构变成

了现实。

第七阶段：构造阶段

在构造过程中架构师的监督职能是确保初始视图被理解并得到执行。架构师根据对构造过程规定的灵活度和改变量来审查构造级的设计。架构师进行设计审查，分析问题，并改变需求。他也设计那些被接受的改变，评估其对整个设计和开销的影响，并对这些改变进行排序。架构师根据客户的需要和要求参与测试和认证审查。

第八阶段：构造后阶段

85

架构师在项目转出和移植到新系统方面为客户提供支持。如果需要，架构师将参与系统操作员和应用人员的培训。如果需要，架构师还将在证书发布和后续维护过程中提供支持。

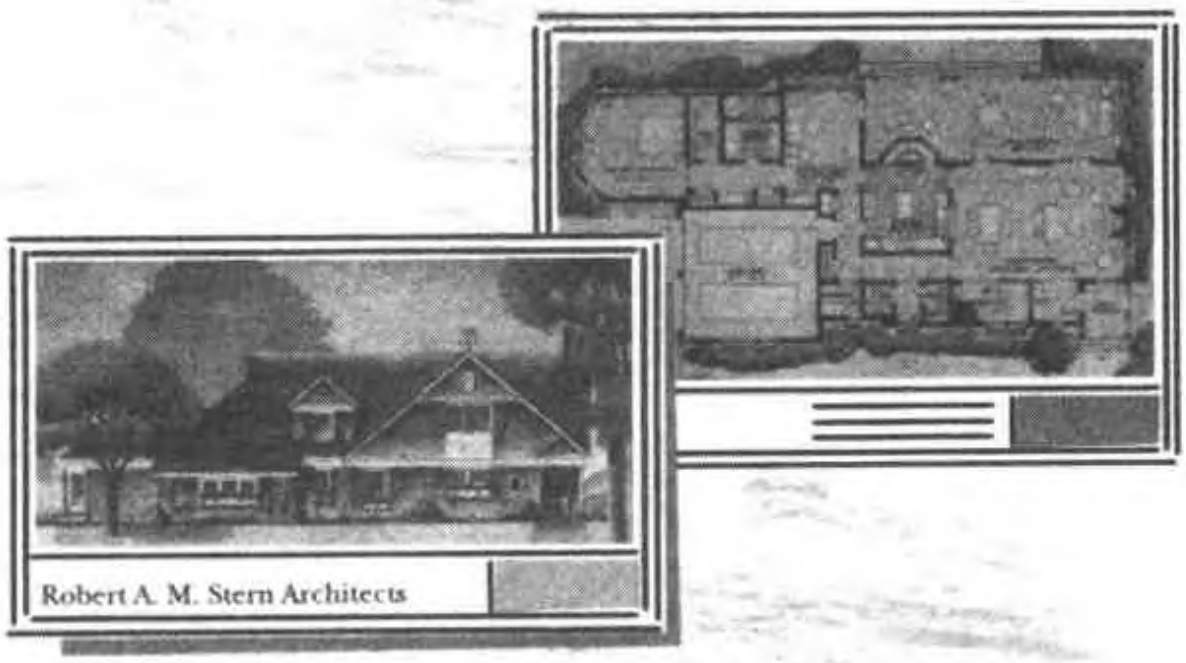
7.6 结论：聚会阶段

当构造全部完工时，架构师和客户聚会一堂，回忆试验和成功的往事（甚至最好的构造项目都充满着难点）。他们在墨西哥饭店为项目相关的建筑者、雇员和顾客举行了一个大型晚会，还有圣母玛利亚乐队的伴奏。那些曾经不停抱怨说这事不成的人现在站在那儿吸着烟，哑口无言。

尾注

1. Thomas Mann, *The Magic Mountain*, translated by H. T. Lowe-Porteo, in *Familiar Quotations*, 16th edition, ed. Justin Kaplan (Boston Little, Brown & Co., 1992), 629.
2. Sir William Schwenck Gilbert, *The Mikado*, in *Familiar Quotations*, 16th edition, ed. Justin Kaplan (Boston: Little, Brown & Co., 1992), 530.

86


Robert A. M. Stern Architects

第8章 架构计划

“计划是生成器。”[1]

“没有计划，就不会有目标和表达的重要性，也不会有韵律，不会有质量，不会有连贯。没有计划，我们就会有不对称感，无序感，觉得空荡荡的，觉得过于随意。而这一切对人类来说都是那么的无法忍受。

一个计划需要最丰富的想像力。同时，它还需要有最严格的纪律。计划决定一切；它是决定性的时刻。计划不是像麦当娜的脸那样是一个漂亮的东西，可以被画下来；它是极端抽象的。计划只不过是一个代数几何，是一个看上去很干瘪的东西。数学家的工作依然是人类灵魂世界中最高级的活动。”[2]

——李·柯贝伊

8.1 架构计划的特征

建筑设计师的计划最基本的目的是想在两个方向上进行交流：与客户和与建筑工人进行交流。计划让客户看到、明白并同意要修建的东西，当建造完建筑后，计划进一步证实被完成的建筑与设计是一致的。反过来，建筑工人把计划作为他们的工作资料。计划是他们工作的基准。

计划、模型或者是设计图本身就可以说明问题。它应该不需要任何的推论或解释，就能独立成立，尽管如何最好地实现计划中每一项足解释的主题，并且因建筑人员的不同而大不相同。但是，不

管在低级设计上会有怎样的不同，计划都应该经受得住建筑师的考验，在不同的建筑工人的手中，都能产生同样的视觉效果。

计划简单与否受制于要修建的结构，无论怎样都要对客户和建筑工人讲清楚。修建完毕后的建筑将准确地反映出计划的好坏。

在软件架构中，和建筑建造一样，计划是从内部延伸出来的。意思是说，结构的所有外部要素都是由客户和使用者的内部需求决定的。这些内部需求包括修建建筑的目的、数目、大小、房间的类型、所希望的技巧和细节的程度，以及外观和感觉。

同样，对某种特殊的技术（例如，美国国税局的扫描程序）或某种方法、语言、工具的喜恶不应该成为计划的决定因素。客户与使用者的内部需求和期望应该是架构师的灵感和动力，既使对有技术偏好或技能有限的架构师来说，需求和期望也不能成为强制力量。

8.2　好的架构师，好的计划

一个既没有全面的技术和知识，又不能完全理解客户世界观的名义上的软件架构师是不能完成客户代言人这一角色的。在这种情况下，客户不可能对正确的多种解决方案进行选择。

这种早已形成的观点在软件业经常被遗忘了。设计和计划常常被分派给专业软件人士，却没有把设计才能或经验的因素考虑进去。可以这么说，有一个不完善的计划总好过一点计划没有，但现在却丢掉这么多的机会！比起只把软件各元素组合起来满足已提出的要求，架构计划就要好的多了。

例如，在美国国税局中（IRS），一个熟练的架构师除了要有效地处理交税形式、付款和罚款外，应该看得更远。架构师应该了解到比所有专家和顾问辛苦积累的需求更多的内容。架构师的观点应该涵盖所有的范畴——美国税收体制，甚至整个联邦政府。这些前

后关系规定了国税局的行为：鼓励、不鼓励、问题和解决办法。一个可靠的架构计划处理这些全局因素，并解决体制问题。

在国税局中，熟练的架构师应该已经设计一种基于软件基础上的解决方法，通过把设计变更融入这个庞大系统来解决国税局 90
的困境和官僚体制。客户可以和架构师一起决定提出的计划是否可行。

没有这种关键性的设计技巧和架构计划来正确地驱动架构过程，其结果是螺旋式失败的开始。最后交付期限的不断加压，对有一个好计划的项目来说，将是健康的推动力，但对于坏计划或根本就没有架构计划的项目来说，只会招致更多的麻烦。

8.3 究竟为什么需要计划

究竟应不应该有蓝图，这个问题在软件业遇到了一些阻力。对于非技术人员，他们感到有些震惊，但在软件业中，没有蓝图却是可以被接受的一种尝试。在这种情况下的设计是匆匆忙忙完成的，资料文献被看作是无用的东西。一些软件业内人士用很深奥的术语来证明这种对待蓝图的态度，他们说软件的设计必须保持在形而上学的范畴内，或者只能在结构建立后再绘制出来。一些人认为，如果让软件结构在运行过程中自己展示出来，那么会显得更有技巧，这与爱斯基摩的雕刻家削磨象牙、从原材料上有机的表现出动物的形状、是非常类似的。

这种看似深奥的理论在周围软件结构不断失败的情况中显得非常愚蠢，建筑设计师常处理一些非常晦涩难懂的文档，但总是能在设计真正的建筑时，看到理论和实践之间的差别。他们已经获得建立理论的权利，因为他们的工作成果是这么的值得信赖。建筑的失败是极其少的，其中的一个例子发生在意大利的一个旅游胜地——比萨，它的一个钟楼很神秘地停止运转了。

8.4 计划的层次

计划的组织结构就是一个关于软件架构蓝图的共享知识体。从这个角度来说，与其他所有方面一样，建造建筑物的类比指明了这条道路。

在真正开始绘制蓝图和制作模型前，建筑师要提交一个提议，详细讲述项目的目的、目标、要求、预算和安排。在软件架构中，这常常被称作商业计划。无论在哪种情况下，这都是utilitas的书面简述，在范围广度、资源利用、时间安排上，客户需要有一个关于
[91] 建筑的结构和描述。在软件架构中，提议通常包括用例（use cases），以便简单描述软件的主要功能或行为，同时还包括主要的设计要点和关键类别。

一旦客户接受了提议，就可以绘制实际的计划了。计划的关键是交流和确保客户的期望得到实现。根据方案的要求，现行计划的格式会不尽相同。如果二维绘制图不能满足要求，还可以做三维的模型。

所谓计划事实上是一套绘图或描画，它有两个方向的交流：与客户和与建筑者。计划根据细节和复杂性具有不同的层次。最不详细的在最上端，依次往下就越来越详尽，底部是最详细的。通常说来，顶端的描画是最不详细的，是给客户看的，但在指导建筑过程上这个描画就不够了。所以，最详细的描画当然是给建筑者看的。

最上面的页总是表示整个项目的广度，很容易在脑海里领会这个类比。如果项目是要建造一个有10个小区1 200户人家的社区，那么计划的最上页将会描绘整个开发地区的鸟瞰图，10个小区的边界都会在开发地区界线内。

在这张总览下面，会有10页纸，每一页纸是每一个小区的鸟瞰

图，同时每一个建筑都被区分开来，并做了标号。

接下来是工程草图，标明开发区的公用管道和输送管道应该放置的方位，其间会指出排雨下水道和地沟的修建位置，同时该草图还应包括对道路修建的具体方案。

然后是对公用区、活动区、池塘、野生动物保护区、自行车道和网球场地的户外设计，在这些区域的建筑都有各自的草图，比如用横截面表明自行车道铺设所使用的材料，详细的池塘设计，以及在保护区内建立观赏平台的板材设计。

最后就是每一个住房的设计图，这些设计图也应该首先有这片土地的鸟瞰图，并将房屋的地点画在上面，然后是房屋的正面图，背面图以及侧面图，所有的都应该展现所有靠近它的人对它的感受。

接下来是客户和住户最爱挑剔的布局图。整个空间由主要场地和次要场地组成。例如，一个工厂，应该有主要的工作场所，然后是一系列的次要场所，譬如餐厅、礼堂、浴室和储藏室。然后是更详细的图，其中包括截面图和详细的建筑图。

在布局图之后是规格说明书，其间列出了需要使用的材料：设备的品牌；电器和管道材料的标准；要使用的砖、瓦或木头的类型；还有其他对建筑设计师很重要的材料，以及为了完成与计划要修建的建筑相一致所需要的一切材料。在重要的草图上，规格说明书用数字被标记出来。

建筑设计师决定计划的设计等级。不管他画得怎样，在建筑过程中都要非常仔细、准确地表现出来。建筑草图中所没有具体画出的地方将由建筑工人来完成。这个层面的细节会由于建筑设计师的不同而不同，也会因为建筑师与建筑工人的关系不同而不同。例如，在某些情况下，工程师可以决定椽子的样子，或者一个熟练的泥瓦匠可以决定拱门和拱顶石该是什么样子。

8.5 结论

和建造建筑物一样，架构计划或蓝图的完成对软件架构师而言，也是至关重要的。设计图的重点在于交流。它确保了架构师和客户之间的相互理解，彼此之间能够共享观点，从而使设计满足客户的需求。

计划在每一个领域都有涉足。它既和客户交流也和建筑者交流，为他们提供足够的细节来想像和按照计划建造建筑。

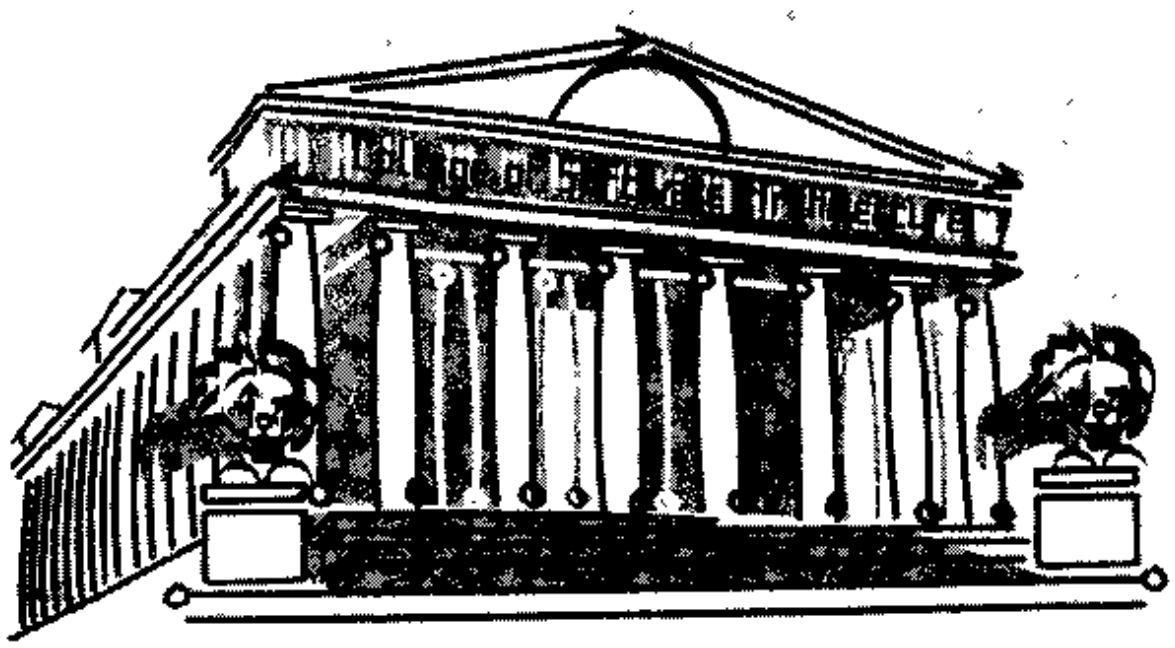
必须制定蓝图。这句话是如此的简单，但在软件业却不能被完全理解和接受。设计是必须要文档化的，因为如果你不知道你在向何方去，任何一条道路都将引领你到达不同的目的地。

尾注

1. Le Corbusier, *Toward a New Architecture* (Mineola, N.Y.: Dover Publications, 1986), 2.

93

2. Ibid., 48.



第9章　软件架构师的教育

“当你忘记你所学过的一切时，教育是仅剩的东西。”[1]

——无名氏

“是教育形成了大众的思维：就好像如果小枝是弯曲的，那么整颗树就会倾斜。”[2]

——Alexander Pope

“软件设计师应该被训练成和架构师、计算机科学家差不多的人。软件设计师应该有很好的技术基础，而不是用写产品质量代码的能力来评估他们。”[3]

——Mitchell Kapor

9.1　第二次浪潮教育，第三次浪潮需要

教育软件架构师的挑战存在于更大的社会环境中：第二次和第三次文明浪潮的冲突。对年轻人统一的、广泛的教育始于第二次浪潮。早在工业革命前，教育还只是一种特权，不是权力。正式的学习教育对大部分人来说是不必要的，在当时的农业社会中，他们不需要学知识来帮助他们进行体力劳动。

当机器取代了体力劳动，人们离开农场迁入都市或郊区时，这一切彻底改变了。

在整个第二次浪潮时代中，工厂和办公室的工作逐渐变得

95

更有重复性，更专业化，也更有时间紧迫感，雇主们雇佣的工人都是听话而且守时的，愿意做一些机械性的工作。在学校培养着同样的人，公司对此也非常欣赏。[4]

——阿尔文·托夫勒

今天，尽管很多公立和私立学校也很希望进入信息时代，但是它们仍遗留着第二次浪潮的习惯和思维方式。他们庞大的官僚体制和教学课程固守不变，不愿接受为培养第三次浪潮的学生而应该有的变化。通过严格的管制，不允许老师充分展示他们的优点和个人特质。

当学校进行改良时，经常是在错误的方向上做着改变，没有强调学生应该在读、写、算这些重要的基本功上多下功夫。科技指导经常是由图书管理员或者是媒体中心的管理员来负责，而其实他们远不如学生们懂的多，这些学生回到家就上互联网，通过即时短信同时和好几个朋友聊天。

遵守纪律和注重老师所讲细节的学生所受到的奖励与真正掌握知识的学生受到的奖励一样多，有时甚至更多。遵守规章制度同学生的本身智力、解决问题的能力和创造力一样，都可以使学生升级。诚然，规章制度是很重要，但是对这方面的强调已经有些偏颇了。目前，有可能成绩一般，但很听话的学生可以得到A，而那些聪明淘气的学生得到C，然后他们回家接着上网。渐渐有人都发现有很多的领袖都曾是得B的学生，有些甚至还被学校退了学。有一个俗语正在四处流传：A是笨蛋，B是领袖。特德·特纳最近在哈佛大学的一次公开演讲时，指责该校多年前拒绝了他的入校申请。

很多领域是永远要求学生们能遵从已有的答案，不要成为冒险家。我们总是希望会计和物理学家能给我们十分准确的答案，当然不希望他们做什么冒险的事。但是信息时代越来越需要那些与众不同，标新立异的思想家，这些人都是非常出名的

叛逆者。一些拒绝随大流的人被称为野鸭子。特别是一些公立学校不仅没有培养这样的思想家，而且还对他们进行打击。

软件工业在不断地发生变化，学校无法很快适应这一行业的需要。尽管软件架构正在崛起，目前却还没有软件架构师的学位课程。他们必须从有关软件工程、计算机科学和程序设计的书本中汲取知识，然后再将这些知识应用到软件架构中。各种书籍和工具都是面 96
向软件构造的，很少有专门适合架构和设计的。

9.2 另一次危机

这个问题比刚才讨论的问题要严重得多。软件架构师一般都是具备计算机科学或软件工程的知识，然后再慢慢发展为架构师的。但是，学习计算机科学和软件工程课程的学生人数上没有变化，而对这些专业毕业的学生的需求量却在不断增加，这些专业的学生还远远不够。在很多地方，有一半的信息技术职位空缺。诚然，学生们对计算机很感兴趣，他们当然很想得到信息技术产业的高薪，但是他们抱怨说计算机科学或计算机工程不是他们喜欢的专业。这种情况在女性中尤其明显。

相反，学生们对更有威望的学科更感兴趣，譬如，海洋生物学或者心理学。有数量惊人的学生认为他们可以参加一些有关计算机科学或者软件工程的课程，但是对在信息技术的高薪领域中有市场的专业更感兴趣。在他们心里，不同的IT行业之间没有什么差别，他们没有看到掌握一种特长或广泛了解科技知识是有必要的。他们认为以自己的计算机基础知识和对某些课程的一知半解，是可以谋到一份不错的差事的。然而当开始求职不久，他们便发现自己不能在IT业中找到适合自己的位置，自己的那些技术知识太单薄，不能完成现实任务。

其他人报名参加了技术程序管理的研究生班。这将使他们走上

项目管理和首席信息官的职位。当这些学生在安全岗位，远离工程和计算机科学，因为坦白地说，他们很害怕这两门学科，但是同时，他们也想进入技术部门。

所有这些学生都是软件架构职业的主要人选。他们受到技术和软件的吸引，但是并没把自己当作是科学家或工程师。

9.3　做我们想做的

职业的选择其实是一个人的天资、兴趣、性情和期望的奇特混合。在职业选择上，我们有些人比其他人目的明确，但是请回头想想：你是如何选择了你现在的工作。我们根据印象、经历、所认识的人和想做的事，在感兴趣的几个职业中选定了现在的职业。

97

一些年过半百的人到现在还在自问长大后想做什么，但是基本上，学生们都对未来的工作有一定明确的目标，认为自己将来会成为一个特殊的角色——或许拎着办公包在华尔街漫步，或给成绩优秀的学生授课。但不幸的是，计算机科学家的形象对很多学生来说感觉不是太好。迪尔波特，一个卡通人物，没有对此做一点帮助，但是，在他身上还是有真实成分的。

9.4　计算机科学家的特征是什么

与其他人或事物相比而言，计算机科学家、程序员和软件工程师更愿意和详细的信息打交道。他们都是逻辑思维很强的思想家，擅长数学。他们是天生的逻辑问题解决专家，做这一切是受到成就感和控制力的驱动，而并非需要得到社会的承认或接纳。人们一般认为他们和工程师差不多。

当然，对这种形象感兴趣的学生是那些也对计算机科学感兴趣的学生。无视学生的兴趣、能力和特点，试图用高薪、奖学金和导师等等来诱惑学生的方法是不会成功的，也不应该成功。对商业、

社会福利救济工作或者对艺术感兴趣的学生是不适合计算机科学的，尽管他们会很努力地去尝试。

为了能吸引那些参加了一些计算机科学或软件工程课程、多少希望在信息技术部门工作，但最终选择了其他职业的学生，需要对结构做些调整。大学和学院应该设立学位课程，这些课程要清楚反映在IT业中正在出现的不同工种，其中包括软件架构。当学生对构造不感兴趣时，他们没有必要成为建筑者或者科学家，即软件工程师、程序员或计算机科学家。

这个类比仍然非常正确。在建筑业，建筑设计师、工程师、建筑工人、工程管理人员、熟练工人、房地产经纪人等等，都有专门针对他们的教育课程，就连建筑科学家都会深入学习建筑的材料和方法。如果有针对架构师、软件工程师、程序员和计算机科学家的独立学位课程，学生们就会根据各自的兴趣，特点和能力，选择某个专业。

在传统意义上，建筑设计师对设计情有独钟。他们学习工程和建筑科学一直到能设计出切实可行的建筑物，但他们的世界观和对事物的看法都受到架构和设计的影响。这就是建筑设计师。请设想一下：给学生上的学位课程是建筑科学和结构工程，却希望毕业后 98
他们能自觉自愿地成为熟练的设计师，这难道不荒唐吗？然而这却在我们当今软件业中普遍存在。

软件工程师和程序员，换句话说，就是建筑者，把设计工作看作是他们获得架构师资格的必经阶段。为什么不是呢？现在并没有设立有关软件架构师的任何学位课程，任何人都可以把自己称做架构师。只不过，他们经常会失败，然后会毫不犹豫地被迫回到他们最初的工程和程序设计工作上。这就是他们的形象，这就是他们的技术，这就是他们的世界观。

架构师一般没有太多科学和数学方面的天赋，他们更多地倾向

于设计、交流、商业和应用技术。软件架构师的程序课程将会吸引新类型的学生进入信息科技领域，不过现在由于不适应计算机科学和软件工程，很多学生正在流失。

9.5 架构教育

普遍的正式教育和职业的专业化是最近才出现的现象。在过去，注册建筑设计师不亲自去修建城堡和教堂，工程师也不去证明自己计划的可行性。事实上，直到19世纪晚期，才出现正式的建筑设计学校。

> 整个19世纪，有成千上万的男人（后来还有女人）称自己是建筑设计师，并在不同层面上实践着他们的技能……。只有一小部分人……在作学徒时接受过培训，时间的长短和内容因人而异。[5]
>
> ——Dana Cuff

为了适应第一次浪潮向第二次浪潮的转变，不可避免地建立了更正式的建筑职业。建筑的范围以及范畴的变化要求明确一个标准的、共享的建筑知识体。突然之间，不仅需要更大的建筑，还需要形状各异的建筑：专门的工厂、早期的摩天大楼、公寓大楼、火车站、百货商场和办公建筑——所有的建筑都表现出自身的特征。

安全成了大众关心的问题，建筑和工程的职业人员要接受学校的课程培训和职业实践，通过旁听和合作的方式，建筑设计师和工程师才能进一步掌握技能，这就像律师事务所所实行的制度一样。在建筑业，大学课程都是按照巴黎的Ecole des 美术艺术设计学院（Beaux Arts）的模式来建立的，一些知名的建筑师在那里研究本国的教育体制，并在回国后对这些体制做出改变或产生影响。

受到客户需求的驱使，建筑业的学位课程迅速增加。建筑学在那时是跨学科的，它涵盖了建筑、工程、制图、绘画和历史等多门课程，不过首先是建筑设计课程。在现代主义出现后，建筑设计师把自己看作是社会变迁的媒介，社会科学在课程中脱颖而出。悲哀的是，当社会主义思潮在学术界站稳脚跟后，对社会科学的强调竟然逐渐遮蔽了设计这一角色。

软件架构师也是在相似的基础上出现的——通过设立符合自身类型的课程和设立基于软件构造商业的学徒培训。学院和大学目前还没有设立软件架构的学位课程，虽然私营和国营单位对设计和架构的兴趣日渐高涨，但各学校还无法在课程中增加相应的内容来体现这一趋势。从这个方面来说，学校教育已远远地落在现实世界的后面。是雇主（客户）把软件架构师的头衔添加到他们的组织图中，从而引起了这个职业的崛起。是这些客户一直在定义软件架构师的角色和职责，而并非学术界。

学校没有对这个趋势做出反应，相反他们把职业软件人都教成建筑者。计算机课程是融合了科学、程序设计、硬件和软件工程的边缘学科。但是即使是这些领域也会不准确。比如说，软件工程是不应该与传统意义上的工程混淆的。它在本质上不是一个工程职业。软件工程师是没有执照的。如果软件结构失败或者有什么结构缺陷，可以说，他们是不会受到指责的。软件工程师不是传统工程师职业的一部分、不能将他们与程序员，工程管理人员或架构师从一般意义上来区别。他们有时会同时担当这所有的角色。

与一般的软件业一样，由于缺乏明晰和对角色的区分，教育专业软件人员的机构也在蒙受损失。能不能说由于这些机构不断地以这种范例来培训学生，从而加剧了，甚至根本是他们导致了这种不明晰呢？

9.6　设立软件架构教育

促进和发展软件架构学课程的任务将落在现在的软件架构师身上。因为，目前的计算机科学和软件工程的教授们没有专业知识或经验来做这件事。商业学校和建筑学校也许会建立同盟　不要认为计算机科学部门或者软件工程部门会开设软件架构学。计算机科学家的主观因素会使他们看不到这个新的学科，也不会接受这个类比。他们仍然相信是软件工程师和程序员逐渐生成为架构师。

建筑学校也提出了同样很尖刻的问题：学校究竟应该有什么样的课程？如何定义架构学？设计——作为一种才能，如何被传授？

9.7　设计能否传授

一旦定义了软件架构，并简要描述出它的主干知识，就会建立起软件架构学教育的正式课程。但是传授设计的古老过程也会同时出现。设计既是一种才能又是一种技巧。就像所有职业的从业者一样，架构师总是会有不同的天赋。卓越的设计是需要付出一辈子的努力的，但是训练是从学校开始的。本质上来说，架构设计者是一流的问题解决者，无论客户的私人企业有什么限制和资源，他们都会按要求提供出色的解决方案。

传统上，设计一直是通过身体力行来传授，而不是通过讲课的方式。法国美术艺术设计学院（the Beaux – Arts Institute）创立了工作室（atelier）方法，并且被传承下来，作为所有建筑设计指导的基本。譬如，esquisse 是一个紧凑的9小时课程，要求学建筑的学生对某个特殊的问题独立设计出解决方案。由于会将他们的设计相互做比较，所以这个工作室的气氛是竞争性的。建筑设计师约瑟夫·艾舍力克就是通过此方法学习设计的：

设计宣布前的1周，只告诉我们设计的名字（比如："纪念

安提俄克圣杯的建筑”，或是“滑雪俱乐部”）。于是我们就尽可能多地到图书馆查阅资料，以找到手边所有有关这个设计的资料。这通常意味着我们查阅的某种特殊类型的建筑范例都是已经建造了的或是设计好的。[6]

101

这就好像是在为特定软件解决方案做设计前，要先学习此领域知识一样。在9小时的会议开始时，对所要解决的现实问题、各种条件（例如，对场地的描述）、可利用的资源以及客户的需求等都要一一介绍。

初期要解决的首要问题是，准确找出你正在解决的是什么样的问题——更确切地说，是计划问题、流通问题、位置规划问题、还是内部问题……最初我们只是关注制定计划的人想些什么，但是后来，由于很大程度上存在着激烈的竞争，我们就必须不仅要猜测制定计划的人在想什么，还要揣摩观察我们的评审委员会是如何看待计划制定人的观点。[7]

在对问题进行初步分析后，学生们开始绘制一些可能会解决问题以及满足条件的小型结构草图。然后将它们反复修改成一些可行的草图，最后形成惟一的最终草图。最终的草图将会用墨水绘制，然后再交给导师。这个学生在接下来的4～5个星期里要不停地忙于他的设计，要向独立评审委员会逐步阐明他的设计思想并为他的设计辩护。

这个esquisse课程模拟了设计时的现实压力和约束力，加强了智力训练。

……如果在解决问题时有一个粗略的计划作为限制，这对不明确、不严谨的思考是一种有利而持久的纠正。[8]

——John Harbeson

Cret这样评论道：

> 为了改进一个糟糕的规划，如果从一开始就给学生正确的解法，那么他会花费更大的力气……（如果学生被允许）一起工作，可以不必留着原始草图，过不了多久（他们会）拿出同样的草图，而这个草图要么是班上最出色的学生的作品，要么是导师曾指出的最好的作品。[9]

在工作室中的强化设计会议——专家研讨会（charrettes）——先前给出计划最后期限，这是建筑学校文化传统的一部分。由于学生的设计是要相互对比的，因此氛围是竞争性的，但是，有时会持续好几天的专家研讨会也是需要经验的。学生或建筑设计师相互帮助，可以及时得到有关彼此的技术的反馈。每个人都开始意识到光有设计知识是不够的。天赋、判断力和承诺是每个人天生就有的，只是深浅程度不同。在工作室中，学生对自己的强项和弱点有了了解，也知道他们还有多少路要走——这些道理在课堂上是学不到的。

102

9.8 结论

软件架构师——设计者——有其职业的特征，他们在兴趣，资质和特点上与建筑者是不一样的。客户和雇主已经看到了其间的不同，也了解了对架构师的需求，因此从基层推动了软件架构师职业的建立。这些客户创立并简单描述了这个角色。现在需要的是建立学位课程来培养软件架构师，同时建立独立的学科。软件架构师可以在建筑设计界找到范例，学习优秀设计的严密性，其间包括天赋和技巧。在学校里，架构师将了解到要成为一个好的架构师所需付出的一切。离开学校，他们会认识到还有更多的东西需要他们用一辈子的时间来学习。

尾注

1. Anonymous, in *Familiar Quotations*, 16th edition, ed. Justin Kaplan (Boston: Little, Brown & Co., 1992), 783

2. Alexander Pope, *Moral Essays* (1731–1735), in *Familiar Quotations*, 16th edition, ed Justin Kaplan (Boston Little, Brown & Co , 1992), 300.

3 Mitchell Kapor. "A Software Design Manifesto," *Bringing Design to Software*, ed. Terry Winograd (Reading, Mass.: ACM Press/ Addison-Wesley, 1996), 9.

4. Alvin Toffler, *The Third Wave* (New York: Bantam Books, 1981), 384.

5. Dana Cuff, *Architecture· The Story of Practice* (Cambridge, Mass.: The MIT Press, 1995), 26.

6. Joseph Esherick, "Architectural Education in the Thirties and Seventies· A Personal View," in *The Architect*, ed. Spiro Kostoff (New York: Oxford University Press, 1977), 258.

7 Ibid., 259.

8 John Harbeson, in Joseph Esherick, Ibid., 259.

9 Paul Phillippe Cret, in Joseph Esherick, Ibid., 260.


Medical Doctor
Attorney at Law
Architect
Software Architect

第 10 章　架构师职业宣言

“某一范围的人群在工作与生活中逐步形成了某些特定习惯，这些习惯不断发展演变，就形成了文化。”[1]

——Dana Cuff

建筑设计师职业始于1857年纽约那个寒冷的冬天，那是一个动荡的时代，世界处于工业时代的前夕。13个实力派人物——他们都是校长、资本家、知识分子和建筑师——从马车上走下来，为了同一个目标聚集到一起：将建筑行业提升到一个职业的高度，建立独立的建筑职业。在此之前，任何人都可以给自己冠以建筑师的头衔，人们也的确是这样做的。

就像140多年后的现在，我们准备开始建立软件架构职业，那时的人们很可能和我们现在一样有很多相同的忧虑和疑问。什么是职业架构师？如何最好地培训他们？他们的主流知识是什么？我们如何回答这些问题？

140多年前的社会正处于转型阶段。金钱代替土地成为新的权力形式，被新兴的资产阶级所掌控。在社会前进过程中，人们开始经历令人目眩的变革。到1920年，报纸和杂志中出现了缓解压力和现代生活所存在的问题的文章——财富所引起的善与恶的斗争。现实世界由于摩天大厦的矗立而改变。为了满足这些需求，建筑设计师作为一种职业出现了。

同样，我们现在也处于转型期。信息技术正作为新的流通形式出现，新的财富都聚敛在新的企业家手中。我们现在或许会认为工业时代人们的行为有些怪异，其实我们现在也同样身处于改革的阵

痛中，只不过我们没有物理上的摩天大楼，我们的摩天大楼是无形的信息结构和网络，而且规模越来越大。

建筑设计师职业的建立是为了满足工业革命的种种需要。建筑的广度和功能无论从质还是量上，都有了很大的提高，同时，为了设计新式建筑，并确保这些建筑是按标准设计的，是合理安全的，对这种正式职业的需求也提高了。建立在软件基础上的信息技术结构也可以说是同种情况，由于很多相同的原因，现在正是到了把软件架构师作为一个正式职业来宣讲的时候了。

10.1　职业是什么

职业是一个具体的行业，或者说事业，它需要某类特定的知识。职业同时还指的是从事那个行业的全体人员。他们已经建立起培训和教育的方法、职业标准、现成角色，以及相似的产品和过程，在特定知识领域中，他们是公认的权威。

目前，软件架构一词被不恰当的使用，软件架构师们有着不同的角色、技术、知识和经验。他们现在必须从别的学科中攫取知识，将学到的知识和技术相结合。很明显，软件架构实际上还不是一个正式的职业，它的建立需要软件架构师和客户的驱动。软件架构还需要有关键性的支持和行动来驱动各大院校开设学位课程。

106

软件架构师是设计者，当前面临的问题是要设计他们自己的职业。与所有的建筑工程一样，这一切都要从utilitas开始：需求是什么？在称呼自己职业者之前，我们需要什么来定义和设计？

10.2　客户的期望

在消费者需求驱动的自由市场经济中，客户或雇主很自然地要求出现软件架构。这个类比给他们产生了一种直觉，架构师在结构建立前先对它进行设计，然后对所有的设计决定进行监督。这个概

念很快就被接受，得到客户在架构上提供投资的支持。

客户对这些职位做了很多职能描述，并决定谁是最合适的人选。那些受雇佣的人同时认为，他们就是架构师了。这种没有任何学校教育或理论支持的开发，其影响是深远的。在很多行业中的企业，无论大小都会应用到软件架构：IBM、星巴克、Sapient、雅芳、西门子等等，不胜枚举。比尔·盖茨最近被晋升为（或自己宣布是）微软首席软件架构师。

尽管如此，对于把软件架构称作是已确定的职业，仍存在很多的偏差。当客户雇佣软件架构师时，他们需要知道通常意义上的软件架构师的角色、技能、过程、产品和能力水平。如果对职业化水平缺乏共同的理解，架构师就必须重新设置客户的预期。一个职业架构师不应该让客户说："上个星期在这儿的架构师认为这个工作比你描述的要少（或多，或者不同）。"

软件架构师是在软件工程、程序设计、系统设计和程序管理的工作中获取他们的头衔的——也就是说，要涉足很多不同的领域。他们的背景在很大程度上决定他们的设计决策和观点。技术很高的人经常更倾向于技术架构，不会太多的注意全球的、领域的和客户的问题。但是情况并非总是这样，关键一点是：客户能从架构师那里得到什么，这应该是可预测的。

一个恰当的例子就是，有一个技术学院现在正在提供信息技术的硕士学位课程，这个学院不要求学生预先有任何计算机经验。学位课程一般对大学毕业生比较有吸引力，他们在学校中已经深知他们的学位并非如希望的那样有用，因此他们现在对技术方面的工作更感兴趣。学院给他们安排了少量课程来学如何使用通用应用软件以及一些联网技术。

107

有人曾经询问这个计划的主管：学生毕业后的就业选择是什么。他回答说他们的工作是局域网管理员、程序员、桌面帮助协调员、

测试员等等。还有人问：这些毕业生是否可以担当软件架构师的角色，他说："那当然。他们可以胜任这个职务"。

客户需要知道软件架构师是否确实具有设计高质量软件的技术能力。所有的软件架构师都不再需要使用建筑设计师所使用过的方法，和建筑设计师一样，软件架构师也有不同的专长。但是对于架构的角色、进程和产品应该让客户有可预测性。如果软件架构的职业被提升到能有一个标准的角色，同时还建立起软件架构的学位课程，那么这个可预测性就会成为现实了。

不幸的是，由于客户对软件开发结果的期望被破坏了，以至很多客户现在都采用短期的方式。也就是，小的软件应用程序一次一个人地开发，然后再以组件形式相加整合。想想类比，就很清楚了，这就像在一片空地上先修浴室和厨房，然后再修房屋其他的东西。是的，我们可以说每一个房间都有自己的作用，但是任何人都可以看出这种方式所存在的问题和局限性。

只有当客户知道他们能从软件架构师那里得到非常专业的、合格的、大规模的设计，并且这一切都是在合理的、易于管理、能得到验证的构造过程中进行，这种短期的方式才会结束。

10.3　标准知识体

软件架构的实践是软件架构师知识的表现。由于这个原因，我们不能得到确定的结果，也不能保证与客户的期望保持一致，除非我们开发出一种标准的共享知识体。目前，这个知识体是有限的，但仍在继续发展。客户是对软件架构师提高要求的动力，这些软件架构师一点点地，通过一个一个工程的实现来逐步建立知识库。

知识库是从客户和架构师每天所做的成千上万的决策中日积月累起来的。和其他领域一样，知识库也是通过实践、培训、理论、

讨论和著述得到延续。我们所需要的是一个充满了个案研究、设计、理论、专业词汇、技术、实践、工具、技巧和沟通的知识库。 108

这个被共享的知识体将被纳入到一个真正的职业所具有的基础结构之中。同样，这个基础结构需要建立在一个普遍接受的假设基础上。比如，需要接受的软件架构就是架构，它不是软件工程或是计算机科学。这个类比是它的关键所在。如果这个类比不被接受，也就是说，如果软件架构仍然被看作是构造的一部分，那么软件架构师这个职业就永远不会被树立起来。

没有这个类比，专业软件人士们就会继续在语义上不停地辩论。比方说，继续辩论风格和模式两个词。如果接受了这个类比，架构的知识体也建立起来了，那么这些辩论就毫无意义了。当建筑物到处都在坍塌或倾斜或是要整体毁掉的情况时，一群建筑设计师们还在为是叫维多利亚式风格还是维多利亚模式在进行辩论，请想想这是怎样的一种荒唐情形？建筑设计师不需要对这些意义进行讨论，因为无论是维多利亚式风格，还是维多利亚式模式，维多利亚式原素，抑或是维多利亚式范例——不管你想怎样称呼它——它都以稳固的、可预测的、精美的、有形的形式表现了出来。如果专业人士们对实用的知识体、进程和结果有一个统一的认识，那么称呼就变得无关紧要。架构的作品说明了一切。

10.4 教育

当然，在软件架构师职业尚未被充分定义，知识体还没有被概括出来时，要想建立软件架构的学位课程是很难的，但我们现在应该开始着手做了。现在有这么多的软件架构师和客户，他们积累了足够的经验，可以开始确定软件架构师的技能和经验。我们期望某个有魄力的学校或大学看到并抓住设立最初的专业课程的机会。这就是事物如何在已划分的第三次浪潮中前进的。别期望由13个男女组成的团体在纽约碰头，尽管这也许也会出现令人感兴趣的结果。

很明显，学位课程首先要从设计中精炼出来，然后是商业、计算机科学、工程和通信。架构师所受的教育应该与软件工程师、程序员或计算机科学家所受的教育不一样，强迫学生接受这些职业的课程训练不会产生架构师。尽管建筑设计师、建筑工人和工程师之间有天壤之别，但是建筑设计师在成为建筑设计师之前，是不会成为建筑工人或工程师的。在建筑设计师熟悉相关专业的时候，他们的角色是设计、并与这些人员相互协作，并进行监督，但并不直接参加具体建造。

109

由于毫无先例可以参照，所以目前的软件架构实践者应该帮助各大院校建立课程，这个职业一直处于缺失状态，现在需要将这个真空填补。当前的教授们都没有当过架构师，他们常常是把精力都投诸于自己的模式中，即使接受了架构驱动构造的前提，对他们来说，面对当前课程和学生的日常需要，也很难将其发展成架构。

一旦教育课程建立起来，知识体将不仅通过新毕业生的工作成果来得到扩展，同时也会从适合软件架构的教育研究和出版物中得到扩展。

10.5　身份鉴定

应该建立身份鉴定的方法。也就是说，要建立一种架构师能被承认为架构师的方法。人类的天性是渴求秩序和稳定性，因此常有人建议说应该建立某种身份鉴定过程，就是把认证打印到真正的架构师的“额头上”。由于一些非常简单的原因，这种身份鉴定的方法并不可取。

首先，在初始阶段，身份鉴定远比证明要重要得多。就此而言，如何判断一个人到底是架构师，还是医生、会计师？如果能确定这个人已经掌握了共享知识体，并达到该职业的所有标准，那么鉴定就变为可能了。由于在这点上架构师共享知识体和职业标准的界定

尚不分明，不可能鉴定一个人是否已经成为真正的架构师。

其次，谁来担当身份鉴定专家呢？又由哪些少数人可以来定义这个职业，决定谁有资格进入这个职业呢？他们的权威又是如何建立的？为什么每个人都得接受他们的鉴定呢？

谁是架构师？——惟有这个职业正式化了，这个身份鉴定的问题才会得到解答。像所有出色的建筑物一样，这个职业需要从里至外地进行设计，而不是按照人为的从外向里的强加顺序来进行。就明晰和精练而言，每天自由市场上所做出的成千上万的个体决定比起一个委员会所做的宣言要远远好得多。

目前，客户是提供主要身份鉴定方法的人，他们通过理解软件如何被设计和构造，以及通过鉴定和雇佣他们所信任的架构师来进行鉴定。客户已经接受了架构师的角色，并很需要这个角色。公司 110
的架构部门以及独立的实践者目前在全世界范围内出现。软件架构的职业确实是一场民主运动，它的定义和鉴定方法也确实应该成为这场运动的灵魂。

10.6 职业道德准则与标准

职业道德准则与标准的意义远比简单的宣誓深远得多。这个准则与某个人履行他的职业头衔有关，也与在软件架构师、客户和其他专业人士之间建立起的信任有关。从美国医学协会到比利时牧羊犬俱乐部，道德准则应用于所有的职业中。这个准则代表了理想和道德的肯定声明——是对职业化主义的肯定。这个准则一般都是非常简单的，但是它向客户和合作者确保他们所打交道的专业人士是对专业组织和其标准负责。如果有任何不符合职业道德的行为发生，通常可以通过一个既定的投诉过程进行投诉。

由于其道德的透明度，职业道德的准则是持久的。新开业医生所立誓约在2000年后本质上依然没有改变。在美国建筑联合会诫条

中有这样三句话：规则——操行的一般原则；道德标准——更明确的行为准则；行为法则——强制行为。违反规定的行为会受到美国建筑师联合会国家职业道德委员会的惩戒。

如果软件架构师这个职业能更好地定义，人们的期望也固定下来，软件架构师的道德准则就会得到实现，但是现在就应该启动这个过程。不管是如何成为软件架构师的，所有的软件架构师都希望能担当这个权威角色，相信他们是客户的代言人，并有能力为他们的客户解决问题和按照客户的品味应用技术。

当然，即使是在非常尖端的职业中，也不可能一直保持高质量。质量确实是客户接受与否的关键。病人决定某个医生是伟大的，还是普普通通的，还是一无是处的，对某个医生的看法会因为病人的不同而有很大的出入。在架构业，客户必须对处于未知领域的设计做出判断。

知道软件架构可以解决公司的信息需求，这还不够，客户的期望远远高于这个。在需求的领域，这个软件是不是能提供战略优势？它会随着公司的成长而成长吗？对于一直“生活”在这个系统中的被雇佣者——它是安逸快乐的源泉呢，还是会增添更多的烦恼呢？它的外观和感觉会提高使用者的生产力和满足感吗？当这个软件面对其他企业和公众时，公司会以此为傲吗？

111

在建筑设计中也会碰到同样的问题。这个类比真是很完美。就像在建筑中的所有情况一样，没有任何道德准则可以保证天赋或是品味。但是准则可以使建筑设计师忠于行为的最高标准，会为客户尽心尽力地工作。

10.7　从何开始

为软件架构设计和建立职业的工作目前已在启动中，这个情况

现在已经很明了。但是，早在软件架构被提升成为一个正式职业之前，架构驱动软件构造就存在了。在这种构造方式的核心环节上，客户会坚持的与软件架构师能够采用的核心思想和方法会出现严重的分歧，将加速软件架构的职业化进程：

- 软件架构师接受与建筑业的类比，这个类比确确实实是正确的。只有在这个类比的帮助下，客户和使用者才能理解设计和构造过程，并使之有效。只有在这个类比下，建筑师和建筑工人才会角色分明。软件架构师是venustas——不是utilitas，也不是 firmitas。他们或许会帮助建造，但是在构造过程中，他们的角色是设计的监护人。

- 软件架构师是客户的代言人，他们为满足客户的全部需求进行设计。他们进入客户的世界来了解领域、需求和要解决的问题。设计是一个与客户相互协作的过程。

- 软件架构师不应受限于有限的知识和很少的经验，他们应当拥有很广泛的知识和技术。

- 软件架构师首先要对设计有信心，然后很细心地把它绘制出来。他们所提出架构计划，以使客户和建筑者之间建立清晰的沟通。

- 软件架构师捍卫他们的工作成果，同时也欢迎所有独立的架构评论。

- 软件架构师渴望承担真正的经典架构师角色，并为某个直到现在才在软件业被提及的目标努力奋斗，对未来也是同样关键的目标，即一个真正伟大的设计。

从这些方面来说，我们可以斗胆把自己叫做架构师，并期待着那令人心满意足的时刻：当所有人站在一起——utilitas、venustas 和 firmitas——看着曾经在一起建造的软件现在这样强壮而健康地

112

运行着，并为使用者带来快乐。

尾注

1. Dana Cuff, *Architecture The Story of Practice* (Cambridge, Mass.: The MIT Press, 1995), 5.

索　引

索引中的页码为英文原书的页码，与书中边栏的页码一致。

B

C

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W