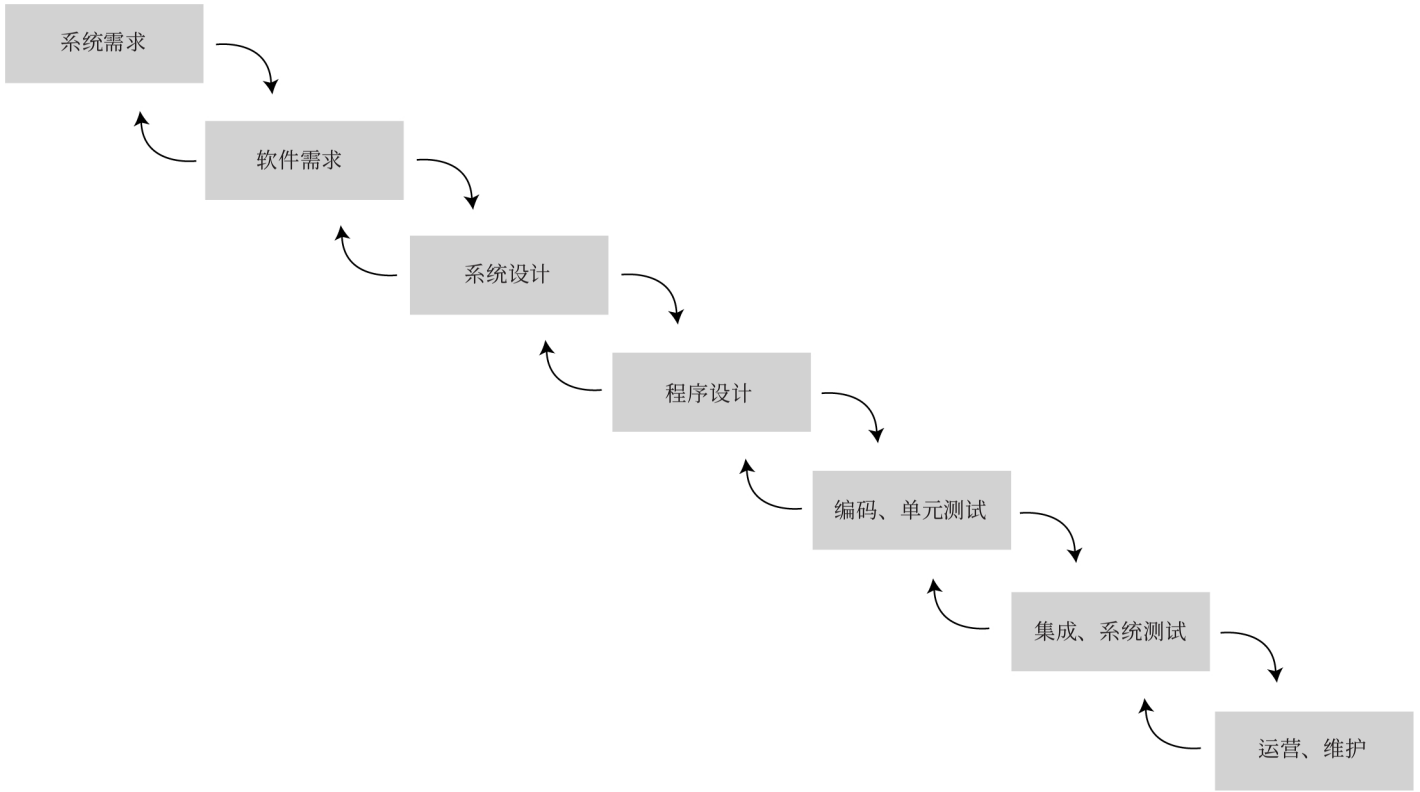
读书笔记



软件开发的瀑布模型

参照Royce(1970),“ Managing the development of large software systems ”和Boehm(1988),“ A spiral model of software development and enhancement ”

第 3 章

理性模型有哪些缺陷

现实情况是，设计师只把理性模型视为一种理想化的东西。它以某种方式描述了在我们的认识中设计过程应该如何运作，但在现实生活中，并不是那么一回事。

事实上，不是每个工程师都会大方地承认在他的心目中有这么一个很天真很理想的模型。但我认为我们中的大多数人都有这样的想法，我自己心中的这种想法持续了很长时间。因此，让我们对理性模型进行仔细彻底的剖析，以确切地了解它究竟在哪些方面脱离了现实。

我们在初始阶段并不真正地知道目标是什么

理性模型最严重的缺陷在于，设计师们往往只有一个模糊不清的、不完整的既定目标，或者说是主要目的。在此情形之下：

设计中最困难的部分在于决定要设计什么。

在我还是学生的时候，有一个暑假里去替一家很大的军火商打工，在那里我被指定去做设计和构建一个小型数据库系统的工作，用以跟踪某个雷达子系统的上万张图纸以及其中每一张图纸的更新状态。

过了几个星期，我做出来了一个能运行起来的版本。我自豪地向我的客户演示了一个输出报告的样例。

“做得不错，这的确是我想要的，不过你可否把这里改一下？那样我们就可以……”

在接下来的数个星期，每天早上我都给客户演示输出报告，每次都是顺应了前一天提出的要求之后的修订结果。每天早上，他都会对产品报告研习一番，然后使用一成不变的、彬彬有礼的口头禅提出另一项系统修订的要求。

系统本身很简单（是在打孔卡片机上实现的），而且那些修订在概念上看起来也是平淡无奇的。就算是最影响全局的变化也只是将图纸列表按照内部等级排序或缩进，而等级是用卡片上单独一个0～9的数字来表示的。其他的改进包括多级局部汇总 —— 当然有例外情况要处理 —— 以及自动地为不同的值得注意的值标注上星号。

有那么一阵子，我很是愤愤不平：“为什么他不可以就想要的内容下定决心？为什么他不能把想要的对我一口气说完，而偏偏要每天挤一点出来呢？”

然后，我一点点地认识到，我为客户提供的最有用的服务，乃是帮助他决定什么是他真正想要的。

那么，如今的软件工程原则要复杂得多了。我们认识到，快速原型是一种进行精准的需求配置的必要工具。不仅整个设计过程是迭代的，就连设计目标的设定过程本身也是迭代的。

软件工程领域的复杂化不仅没有停止的势头，甚至连明显的放缓也看不到，在汗牛充栋的文献资料中，“产品需求”仿佛是给定设计过程的常规假设前提。不过，我要提出一点异议，那就是，在初期就能了解整个产品需求属于相当罕见的例外，而远非常态：

设计师的主要任务乃是帮助客户发现他们想要的设计。

至少在软件工程领域，快速原型的概念有其地位及其公认的价值，但在计算机（体系结构）设计或是建筑架构设计中，它的地位与在软件工程中并不总是相当。但无论如何，在目标迭代的方面，我在这些设计领域都看到了相同的现象。越来越多的设计师们为计算机构建模拟器，为建筑构建虚拟环境演练，作为快速原型，以促成目标的收敛。目标的迭代必须作为设计过程的固有组成部分加以考虑。

我们通常不知晓设计树的样子——一边设计一边探索

在复杂结构的初始设计，如计算机、操作系统、航天飞机以及建筑等，每一项主要设计工作在如下方面都有足够的新意：

- 目标
- 必要条件和效用函数
- 约束
- 可用的加工技术

这些步骤中，设计师很少有机会能坐下来先验地绘制出一个设计树来。

此外，在高技术领域的设计中，甚至很少有设计师能够拥有足够的知识以绘制出该领域中基本的决策树来。设计项目往往会进行两年以上。设计师在此期间会得到升迁，从而脱离一线的设计工作。这样导致的后果就是，很少有设计师会在其职业生涯中将其一线工作深入到参与上百个项目的程度。这意味着，对于设计个人来说他就失去了探索其设计科目的所有分支的宝贵机会。因为这是工程领域的设计师的特点，和科学家大相径庭的是，他们很少会去触碰那些不能一眼看出是通往解决方案的备选途径。[1]

相反地，设计师们会一边做着设计，一边进行设计树的探索——做出某个决策，然后查看由它启发或否决的备选方案，继而依此做出排在下一个的设计决策。

（设计树上的）节点实际上不是设计决策，而是设计暂定方案

事实上，特定的设计树自身只是在树形结构中搜索的简化模型。如图2-1所示，有并列的属性分支，也有备选分支。在一个分支中的各个备选方案彼此紧密联系——或彼此相斥或相辅相成或平分秋色。我们在《Computer Architecture》一书中给出的大块头设计树其实还是过分简化了；那样的一个设计树中所展示出来的“计算机众生相”对于阐明决策之间的联系乃是必不可少的。[2]

这意味着，在设计树的每一个节点处，设计师所要面对的不仅仅是为单独一个设计决策准备的若干简单备选方案，而是为多个设计暂定方案准备的备选方案了。

此外，设计树中的决策排列顺序事关重大，可以参见Parnas在其经典论文“Designing software for ease of extension and contraction”中所阐述的真知灼见。[3]

以树型结构表示的设计模型，其复杂性带来的组合爆炸是人们思维中的难以承受之重。（这情形就像是国际象棋中的棋子移动所构造出来的状态空间树。）该困境在第16章会有进一步的探讨。

有用性函数无法以增量方式求值

理性模型的假定是，设计是对于设计树的搜索，并且在每个节点处人们可以对若干下一级分支的有用性函数求值。

事实上，除非探索到所有分支的所有叶节点的程度，否则人们就很难做到这一点，因为大量的有用性指标（比如性能、成本等）对于随后的设计细节有着强烈的依赖。因此，虽然对有用性函数的求值在原则上是可行的，但是在实践上，人们就会在这里再次遭遇组合爆炸。

那么，设计师该怎么做？估算！理所当然，正式的也好，非正式的也罢，都要做估算。在求精的步骤中，人们必须对设计树进行剪枝。

经验。很多辅助信息都能够促进该过程中的直觉判断。其中之一是经验，无论是直接还是间接的：“OS/360的设计师们将OS/360操作系统中在整个系统范围内共享的控制块的格式细节暴露出来，这后来成为了维护工程师的噩梦。我们会将其封装为对象。”“宝来B5000系列在很久以前就探讨过基于叙词的计算机体系结构。由于本质性能损失实在太大，我们不打算继续深入设计子树了。”当然，工艺方面的权衡早已日新月异，但是上面的例子仍然很好地说明了经验教训。研习设计史的最有力的原因是去了解怎么样的设计方案是行不通的以及为什么这些设计方案行不通。

简单估算量。设计师们经常在进行设计树探索的早期就例行地采用简单估算量。建筑师们在得知目标预算以后，会粗略地估算一个平均到每平方英尺的成本，得出一个每平方英尺的目标，并使用它进行设计树的剪枝。计算机架构设计师们则会根据指令组合来对计算机性能做粗略的初步估算。

当然，这样做的危险在于，粗略的估算有可能会将本来可行但由于某个特定的估算量所采用的估算方法而看上去不可行的分支剪掉。我见过一个建筑师，他以过高的成本为理由，把一个早已指定的房顶结构之下的一堵墙壁给取消了，纯粹基于例行的平方英尺估算量他就作了这样的决定。而实际情况是，为增加的空间而付出的成本主要在于房顶，但那个是已经计算在内的了，所以这么一来边际成本会非常低。

将欲免费取之，必先无偿予之。

必要条件及其权重在持续变化

Donald Schön，已故麻省理工学院的都市研究与教育教授、设计理论家如是说：

（当设计师）按初始状况进行设计改造的时候，状况本身会“抵触”，而他只能就这种状况反弹做出回应。

在健康的设计过程中，这种状况交互是自反的。在回应状况反弹时，设计师会将问题的构造、行动的策略以及现象的模型纳入行动的考量，在每一步的推进中都隐含了这些考量。[4]

简而言之，在对于权衡的沉思中，一种对于整体设计问题的新理解逐渐浮现，即它是诸多因素以错综复杂、彼此牵制而又彼此交互的方式组合的结果。由此，对于诸项必要条件的权重计算方法就发生了变化。客户方——如果有——也逐渐地接受了这种理解，以此为出发点来形成对他将得到的成果的期望以及他将如何使用这个成果的预见。

例如，在我们的房屋改造设计中（详见第22章），一个在原始项目中看似简单的问题，在设计推进的过程中突显出来，原因就在于我和我的妻子将用例场景应用到了原始设计时的一个发问：“来参加会议的客人们该将他们脱下的外套搁在什么地方呢？”这个看起来权重不高的必要条件产生的影响规模很大，结果是把主卧从房间的一端迁移到了另一端。

此外，对于那些必须进行分块加工的设计，比如建筑和计算机的设计，设计师们从建造者处一点一滴地学习到有关设计和加工是如何交互的理解。大量的必要条件和约束条件被变更和改进。加工工艺也会有演进的过程，这对于计算机设计而言就是老生常谈的事了。

由于许多必要条件（如速度）是以性价比为权重的，这就会导致另一种现象的发生。随着设计向前推进，人们会发现在只需负担极少的边际成本的前提下，就可以增加某些特定的有用性的机会。在此情形下，在原始的必要条件清单中根本不存在的项目就会被添加进来，而这往往会使其后的设计变更中要求保留的预算余地被挤占。

例如，只有北卡罗来纳大学的西特森厅在设计、建造和投入使用的过程中，计算机科学系作为该建筑的用户，才学会如何在由楼下大堂、楼上大堂、学院会议室、讲演厅和走廊的成套空间内，将所有这些漂亮地组合成一个能够举办多至125人参加的会议的基础设施，同时把因其施工而对大楼内其他工作的影响减至最低。这个成功也可谓有着各种机缘巧合，因为在最初的建筑方案中并未考虑该厅所拥有的功能。然而，这是价值颇高的特色：任何对于西特森厅的

未来修订肯定会把保留这些功能作为目标。

约束在持续变化

即使设计目标固定而且已知，所有的必要条件皆已枚举清楚，设计树已经刻画成很精确的样子，并且有用性函数也有着明确无误的定义，设计过程也仍然会是迭代的，因为约束在持续变化。

通常情况下是环境变了 —— 市政厅会通过令人沮丧的规定给设计投下新的阴霾；电气规范每年都会更新，本来计划要用的芯片被供应商召回了，等等。一切都在不断变化，即使在我们的设计向前推进的过程中也并未留步。

约束也会由于设计过程中甚至加工过程中的新发现而发生变化 —— 建筑工人碰到了无法凿穿的岩层，分析结果表明芯片的冷却问题成为了新近的约束，等等。

并非所有的约束变化都是增长型的。约束也经常消弭于无形。如果这种约束变化是偶发的，而不是人为的，熟练的设计师就能利用这样的新机遇，发挥其设计的灵活性，以绕过该约束。

并非所有的设计都有灵活性。更为常见的是，当我们深入一个设计过程时，就意识不到原来某个约束已经消失不见，也想不起来由于它的消失而早已排除的设计备选方案了。

重要的是要在设计过程的一开始就明确地列出已知的约束，作为架构师所谓的设计任务书的组成部分。设计任务书是一个文档，需要与客户共同完成，它规定了目标、必要条件以及约束。本书的网站给出了一个设计任务书的示例。设计任务书和正式需求描述文档不是一回事，后者通常是具有合同约束力的、定义某个设计方案的接受标准的文档。

将约束明确列出，是把丑话说在前面，这就可以避免日后突然爆发令人不快的局面。这同时也是在设计师的脑海中烙下对于这些约束的印象，从根本上提高某一约束消失时被设计师发现的可能性。

我们都是围绕着约束来做设计的，该过程要求对于设计空间中少有人问津的犄角旮旯有着很多创新和探索的精神。这是设计之乐的部分所在，这也是设计之难的大部分所在。

在设计空间之外的约束变化。然而，有时，设计的突破性进展来自于完全跳出设计空间的囚笼，在那里将设计的约束加以消除。在设计厢房的时候（见第22章），我努力了很久均未果，就为了一个令人心情灰暗的靠后尺寸需求约束以及音乐室的必要条件（要放置两架三角钢琴、一架管风琴以及一个正方形的空间以容纳弦乐八重奏乐队，加上一英尺宽的教学之用的余地）。如图3-1所示，这是设计过程的一次迭代以及其约束。

这个设计过程中遇到的棘手问题最终是在设计空间之外得到了彻底解决 —— 我从邻居处买下了另外五英尺的地皮。这比向市政厅申请靠后尺寸变更 —— 另一种设计空间之外的解决途径 —— 来得可能更经济，并且肯定更富效率。它同样给设计方案的其他部分带来了解放，对于F书房的西北角的定位贡献尤其明显（见图3-2）。

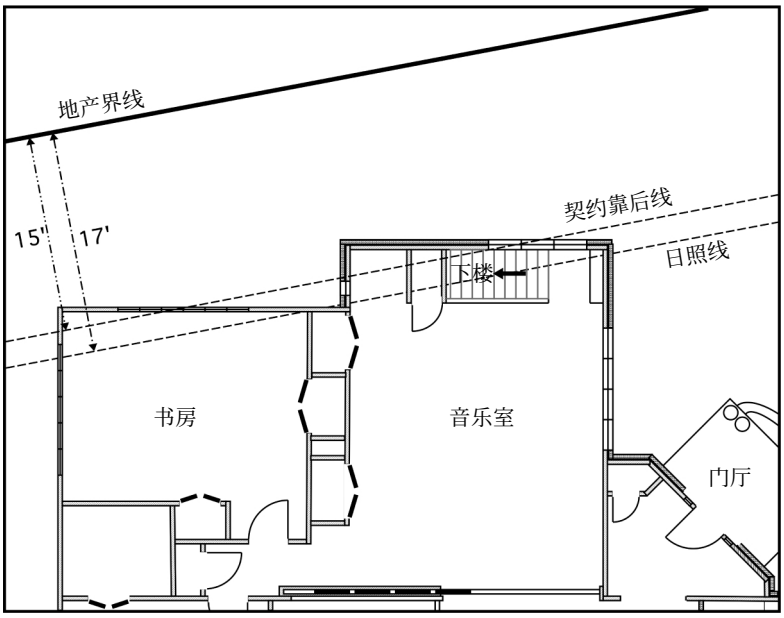


图3-1　依约束进行的设计

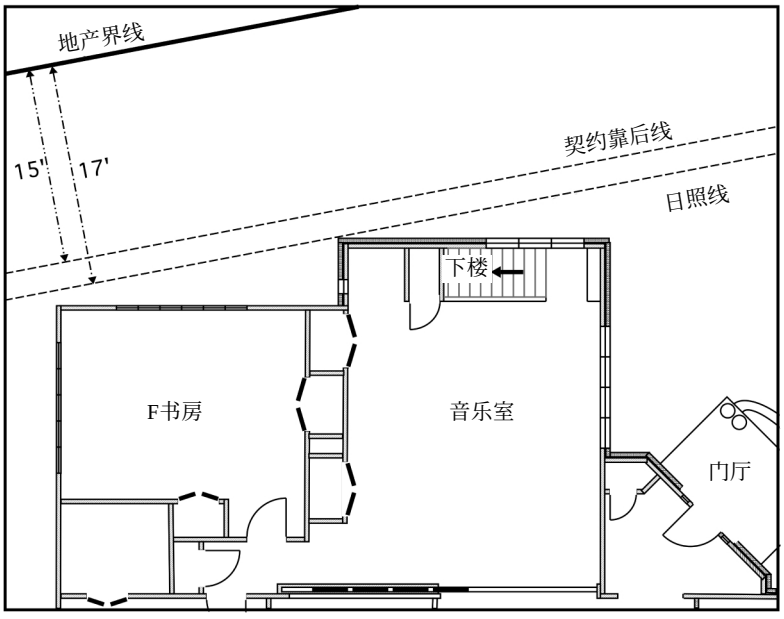


图3-2　约束被放松了

将设计任务书中的已知约束明确列出的好处，在此处也有体现。设计师们可以定期地检视这个清单，自问：“现在有些东西已经变化了，这个约束能够去掉了吗？能不能通过在设计空间之外想出办法来规避它呢？”

对于理性模型的其他批评

理性模型是一种自然的思维模型。理性模型，如上所述、如上所评，似乎看上去相当幼稚。但它是人们能够自然而然地想到的一种思维模型。其思维自然程度可以从Simon版本、瀑布模型版本以及Pahl和Beitz版本分别独立地提出而得到强烈的印证。然而，从最早的时候开始，设计界就有了对于理性模型有说服力的批评。[5, 6, 7]

设计师们根本不那样做事。也许对理性模型最具解构性的批评 —— 尽管也许亦是最难以证明的 —— 就是经验最丰富的设计师根本不那样做事。虽然已经发表出来的批评只是偶尔才会有“皇帝没有穿衣服耶！”这样指出该模型并未反映出专业实践做法的呛声，但是人们还是可以感觉到不厌精细的分析背后的这种压倒性的主张。[8]

Nigel Cross，其绅士般的言论，也许是最具张力的不同意见。引经据典之下，他毫不讳言地说：

有关问题求解的习惯思维，往往看起来和专家级设计师们的行为背道而驰。不过，设计活动和使用习惯思维进行问题求解的活动有着相当多的不同之处……我们必须在从其他领域引入设计行为模型时倍加小心。对于设计活动的实证研究经常会发现，“有直感力的”设计能力乃是最有成果的，也是和设计的内禀性质最密切相关的。不过，设计理论的有些方面就是企图针对设计行为开发出反直觉的模型和处方来。[9]

又及，

在绝大多数设计过程的模型中都忽视了性质上处于同等地位的设计推理。在有着公论的关于设计过程的模型中，比如说由德意志工程师协会颁布的模型中（VDI，1987）……就主张设计活动应该分成一系列的阶段进行……在实践中，设计活动似乎是以在子解域和子问题域的区间振荡的方式进行，同时也是将问题分解，尔后合并所有的子解决方案的过程。[10]

我发现，这个争鸣意见本身以及其实证材料都很具说服力。此处提及的振荡的确可以说是我所有设计经验的特点所在。“外套在哪里摆放？”这样的需求发掘了我们房屋设计过程的深层次内容就是个典型例证。

Royce对于瀑布模型的批评。Royce在他的论文原稿中描述了瀑布模型，以便他能够演示其不足之处。[11] 他的基本观点在于，尽管在毗邻方块之间有反向箭头表示逆向的工作流，但是该模型仍然行不通。不过，他的对策仅仅是采取了容许逆工作流箭头指回两个前向方块的模型罢了。治标不治本。

Schön归纳的批评小结。

[Simon]发现在专业知识和现实世界的要求之间有着一道鸿沟……Simon提出……采用科学化设计的方式弥补这个鸿沟，他的科学化方法只能应用于对从实践中总结的、良好构建的问题。

如果这种所谓的技术化理性模型未能考虑到实践能力的“发散”情景，用了还不如不用。那么，还是让我们转而追寻一种基于实践的认识论，它隐含在艺术的、直观的实践过程中，而这样的过程已经被一些实践者用以应对不确定性、不稳定性、单一性和价值观冲突。[12]

但是，尽管有这些缺陷和批评，理性模型仍然不屈不挠地存在

通常来讲，某种理论或技术的原始创意提出人都比后继的追随者更了解其承诺所在、其局限所依以及其正确的应用范围。由于天资有限、热忱有余，他们的一腔热情却导致了思维僵化、应用偏差和过分简化等。

遗憾的是，理性模型的诸多应用亦是如此。近至2006年的文献，设计研究员Kees Dorst亦不得不承认，

尽管从彼时到现在已经有了长足的进步，但是Simon所著的探讨问题求解以及具有病态结构的问题本质的原始著作，仍然在设计方法论领域有着难以忽视的影响。基于Simon提出的概念框架的理性问题求解范式，在该领域仍然占主导地位。[13]

诚哉斯言！在软件工程领域，我们仍然太过盲从瀑布模型——理性模型在我们领域的衍生品。

德意志工程师协会标准VDI-2221。德意志工程师协会于1986年采纳了理性模型，本质上如同Pahl和Beitz所介绍的那样，作为德国机械工程业界的官方标准。[14] 我见过很多由于这场运动引发的思想僵化。但Pahl自己一直在尽力设法澄清如下：

在VDI准则2221-2223以及Pahl和Beitz（2004）中给出的过程并非“按部就班”式的，它只能被视为有明确目的的基础行动的指导性准则。在实际行动中有用的解决方案可能或是选择一种迭代途径（即那种带有“前进或回退”步骤的途径），或在采用更高信息层次上的反复途径。15

美国国防部标准2167A。类似地，美国国防部于1985年将瀑布模型正式纳入其标准2167A。[16] 直至1994年，在Barry Boehm的领导下，他们才开放了其他模型的准入门槛。

那又如何？我们的设计过程模型真的那么事关紧要吗

为什么就过程模型讲了这么多？我们或是别人用来进行设计过程的思维真的会影响我们的设计本身吗？我认为的确是这样的。

并非所有的设计思想家都同意我的观点。剑桥大学教授Ken Wallace与Pahl和Beitz著作的所有三个版本的英文译者，相信一个主要的进步会是某种让人能够轻而易举地理解和沟通的模型。他指出这一点对于设计的初学者来说是多么重要。Pahl和Beitz的模型为新手做设计准备了一个入手的空间，这样他们就不会徒陷彷徨。“我把Pahl和Beitz的图（他们书中的图1-6）放在

我的讲义上，并解释了一下。在我紧接着下一张幻灯片上就写道：‘ 但现实中设计师是不那样工作的。’” [17]

不过，我担心是否有更年轻的、个人设计经验更少的教育工作者总是会这样说。

苏珊娜 · 罗伯逊和詹姆斯 · 罗伯逊夫妇有着国际化咨询的实战经验，并且著有需求规划的重要作品，他们认为理性模型的不足之处并不值得大惊小怪。“ 更了解设计的人也更有智慧。” [18]

无论如何，我相信我们带有缺陷的模型以及对其的盲从，将导致臃肿、笨重、功能过多的产品以及时间表、预算和性能的灾难。

右脑型的设计师。绝大多数设计师是右脑型的，是视觉 - 空间导向的。事实上，我在考察未来设计师的天赋时，用于投石问路的问题就是：“ 下一个11月在什么地方？ ” 当听我说话的人显出莫名其妙的表情时，我会进一步地解释，“ 你有一个日历的空间思维模型吗？很多人都有的。如果你也有，能给我描述一下吗？ ”几乎每一个有竞争力的候选人都会有这样一个模型，但这些模型彼此大相径庭。

类似地，软件设计团队总是在他们共用的白板上画草图，而不是写文字和代码。而建筑师则把在描图纸上用的美工笔视为一个不可或缺的沟通工具，但是在独立思考时则用得更多。

因为我们设计师是空间思维导向的人，我们的过程模型在脑海中是以图表的形式深深植根的，无论其具体形式是Pahl和Beitz的垂直式矩形，还是Simon的树型结构，抑或是Royce绘制和批评的瀑布状图形。这些图表在潜意识层面大大地影响着我们的思维。因此，我认为一种先天不足的过程模型会以我们不能完全知晓、只有一知半解的方式阻碍我们前进。

一个由于采用了理性模型而造成的明显损害就是我们无法对接班人进行恰当的教育。我们教给他们连我们自己都不遵循的工作模式。结果，在他们形成自己在现实世界中采用的工作模式的过程中，我们就没能提供有用的帮助。

我认为，对于更资深的，尤其是那些有在业界设计经验的教师来说，情况就会大不一样。我们很清醒地认识到，这些模型是有意简化过，以帮助我们解决真实生活中遇到的问题，后者往往复杂得令人生畏。因此，我们在教给学生的时候也会提出“ 这只是地图，而不是真实的地形 ” 的警告，因为只有模型是不够的，即使在可以适用的场合，它们也仍然有失精确。

在软件工程实践领域，还可以很容易地发现另一种不利因素 —— 理性模型，无论以何种面目出现，都会导向一种先验的设计需求描述。它导致我们盲目相信这种需求真的是可以预先制订的。它也导致我们在对于项目一无所知的基础之上就彼此签订了合同。一种更加现实的过程模型将使得设计工作更富效率，并省却许多客户纠纷和返工努力。第4章和第5章将阐述需求问题。

瀑布模型是错误的、有害的，我们必须发展并摒弃之。

注释

1. 工程师需要的是最低限度满足解，而科学家则需要的是发现，这往往可以通过在更大范围里探索而求得。

2. Blaauw和Brooks(1997),《Computer Architecture》，26-27，79-80。

3. Parnas(1979),“为简化可伸缩性软件而进行的软件设计”，明确地将设计过程作为树型结构的遍历来处理。他强烈主张使设计尽可能地灵活。他敦促人们设计的灵活性是重要的目标之一。在软件工程领域，面向对象的设计也好，敏捷开发方法也好，都将此作为根本目标之一。

4. Schön(1983),《The Reflective Practitioner》，79。

5. 出乎意料的是，我发现对于Pahl和Beitz进行理性模型的构造方法，以及对于Simon对此的大部分构造方法都少有批评。Pahl和Beitz自己倒是意识到了该模型的不足之处：在他们著作的后续版本里，他们的模型（在第2版、第3版英文版中的图3-3、图4-3）包括了越来越多的迭代步骤（Pahl和Beitz(1984, 1996, 2007),《Engineering Design》）。Simon三个版本的《The Sciences of the Artificial》并未反映出计划中模型的变化，尽管他于2000年11月在和我的私人谈话中曾经透露，他自己对该模型的认识已经有所改变，但是他还没有机会去就此重新思考和改写原著。

Visser(2006),《The Cognitive Artifacts of Designing》，该书中的9.2节是精彩的一节，“Simon在其更新近工作中的微妙位移”，它考察了Simon在其更近期发表的论文中体现出来的思想演化。

Visser与我一样吃惊地发现，这些思想演化并未反映到《The Sciences of the Artificial》的更新版本中。

6. Holt(1985),“设计还是问题求解”:

有关工程设计存在两种截然不同的解释。一种是“问题求解”解释，这在很多大专院校中比较普及，它强调使用标准化技术对结构化的、有明确定义的问题求解，这种解释可以追溯到“裸”系统思维。另一种是“创新设计”解释，它将分析思维及系统思维和人为因素结合在工程设计中，以创设和利用各种机会来更好地为社会服务。本论文旨在探讨“问题求解”解释在应付很多现实世界的任务时会遇到的种种限制。

7. 如果说Cross的批评是基于实证的，那么Schön的批评则针对理性模型的深层哲学。他说，理性模型，正如Simon所阐明的那样，是一种更加普适的他称为技术理性的哲学思维方式的自然外延，他认为后者继承了现在已经不再有市场的实证哲学的衣钵。他发现只立足于这种深层哲学本身对于理解设计的要求而言是完全不够的，尽管它已经被制度化地引入了最专业化的设计课程中：

从技术理性的角度来看，专业设计是个问题求解的过程。问题……通过从可用的手段中找到最符合既定目标的选择而得以解决。但是由于过分强调了问题求解这回事，我们忽视了问题本身的要求，以及定义所作决定、最终达到的目标，以及选择可能的途由等这些过程。在现实世界的实践中，问题在实践者那里往往并不像乍看之下那样表现。问题必须由带来问题的情形

材料构造出来，而后者往往令人费解、麻烦不断而且难以捉摸……实践者必须得就此费些力气才行。他必须得把一开始完全不显山露水的情形搞个水落石出……像这样的情形才被专业人员越来越多地视作他们工作的核心……技术理性取决于如何看待目标这回事。

8. 一个生动的例子是Seymour Cray在1995年所言节录：“我应该算是个科学家，不过我在作决定时使用直觉更胜过考虑逻辑。”http://www.cwhonors.org/archives/histories/Cray.pdf，访问于2009年9月14日。

9. Cross(2006),《Designerly Ways of Knowing》，27

10. Cross(2006),《Designerly Ways of Knowing》，57。Dorst(1995),“描述设计活动的图表比较”，有一个特别好的关于Simon和Schön的比较讨论。他们的杂志文章在Cross(2006),《Analysing Design Activity》中重印。Dorst还展示了在代尔夫特Ⅱ协议中，Schön的模型和观察所得的设计师行为更精确地符合。

11. Royce(1970),“大型软件系统开发的管理。”

12. Schön(1983),《The Reflective Practitioner》，45-49。

13. Dorst(2006),“设计问题和设计悖论。”

14. VDI(1986),《VDI-2221：Systematic Approach to the Design of Technical Systems and Products》。

15. Pahl(2005),“VADEMECUM-设计方法论之开发和应用中的建议。”

16. DoD-STD-2167A企图修正此问题，但遗憾的是它将瀑布图表放到了一个突出位置，从而一切都还是基本保持不变。MIL-STD-498取代了2167A，并着手处理了模型问题。DoD自采纳了工业标准IEEE/EIA 12207.0、IEEE/EIA 12207.1和IEEE/EIA 12207.2之后才取代了498。

17. 个人通信（2008）。

18. 个人通信（2008）。