读书笔记

- 目标
- 必要条件
- 效用函数
- 约束，尤其是预算（也许并非金钱成本）
- 决策的设计树

UNTIL（“ 足够好 ”）or（来不及了）

DO另一个设计（以提升效用函数）

UNTIL设计完成

WHILE设计方案仍然可行，

做出下一个设计决策

END WHILE

回溯设计树

找到一条之前未探索过的路径

END UNTIL

END DO

采用最优设计方案

END UNTIL

理想的设计过程模型

第 2 章

工程师怎样进行设计思维 —— 理性模型

……因为设计的理论即一般的搜索理论……对象是巨大的组合空间。

—— Herbert Simon(1969),《The Sciences of the Artificial》

模型概览

工程师们对于设计过程似乎有一个清晰但通常来说也是隐含的模型。这是一个关于有序过程的有序模型，也就是工程师的构思过程。我可以举一个海滨小屋设计（在第21章给出其草图）的例子来说明这是怎么回事。

目标。首先从主要目标或目的开始：“某人想要建立一个海滨小屋，以享用面向大海的一块海滨场地的风浪。”

必要条件。和主要目标相关的是一组必要条件或者说是次要目的：“海滨小屋应该加固，以抵御飓风来袭；它应该具备至少14个人躺卧和就座的空间；它应该为宾客提供令人难忘的视野”，等等。

效用函数。人们会根据一些效用或有用性函数来为若干必要条件依其重要性加权，以对设计进行优化。到目前为止，我知道的情况是，在大多数设计师的想象中，所有的项是由线性相加的方式组合起来的，但在单独构思每一个有用性函数时，则并非使用线性方式，而是以渐近曲线的方式趋于饱和。举个例子，必要条件之一是更大的窗户面积，这是在小屋设计中所需要考虑的问题。但是由窗户面积每平方英尺的额外增加所带来的效用是递减的。对于电源插座的数量来说，这也一样成立。窗户面积以及插座数量的总效用，看起来却仅仅是每个项的简单之和。

约束。每种设计以及每种优化都是受到一些约束限制的。其中有一些约束是二元的，只有满足或不满足的结果 —— “这所小屋必须位于海滨场地的边界线并再向后退至少10英尺”。其他约束则更有弹性，不过在接近限额时所付出的代价会急剧增加，比如日程表就是这样一类约束 —— 主人可能急切地要求该海滨小屋在温暖气候来临之前完工。有些约束是简单的，比如退后尺寸的限额。另一些则在不经意间隐藏着令人生畏的复杂性 —— “该小屋必须满足所有的建

筑法规”。

资源分配、预算和关键预算。许多约束的形式是固定资源在各个设计要素之间的分配。最常见的是一揽子成本的预算。但是，此类约束绝不仅仅只有这么一种，而且在特定的项目中，总预算约束也并不一定就是最大限度地决定了设计师注意力的约束。例如，在海滨小屋的楼层规划中，占支配地位的定量因素是临海建筑距离的英尺数（甚至要精确到英寸）。在计算机体系结构的设计中，关键预算可能是控制寄存器或指令格式所占用的比特数，或总内存带宽的用量。而当人们解决软件的“千年虫”问题时，日程表上的工作天数成为了可分配资源中的关键项。

设计树。这么一来，按照理性模型的思路，设计师们形成设计决策。然后，在由于该决策而缩编后的设计空间中，他又形成另一决策。[1] 在每一个节点处，他都可以选取一条或多条路径，因此设计的过程可以认为是一种对于以树型结构组织的设计空间的系统化探索。

在这样一个模型中，设计在概念上（至少在概念上）是个简单的过程。人们对以树型结构组织的设计空间进行搜索，以可行性约束为依据对每种方案进行检验，从而优化效用函数。搜索算法是众所周知的，并且可以清晰地描述。

这种清晰性仅存在于对所有路径进行的穷举搜索中，其目的在于寻找一个真正的最优解。设计师们通常只去寻找一个“足够好”的最低限度满足解。[2] 许多工程师似乎采用了某种深度优先策略进行近似估算，并在每个节点上选择最有前途或最有吸引力的方案，并采用探索到底的办法来达成目的。如果遇到死胡同，他们会采用回溯的办法并尝试另一条路径。预感、经验、连贯性和审美旨趣引导着每一次的方案选择。[3]

该模型的构思从何而来

将设计过程建模为一种系统化的、按部就班的过程的观念，似乎肇端于德国机械工程社团。Pahl和Beitz在他们7次修改其稿的伟大论著中阐发了目前被最广泛地接受的观点。[4] 他们对达·芬奇（1452—1519）的《Notebooks》中关于设计备选方案的系统化搜索过程施以实践观点的考察，而并非只泛泛阅读那显式写出的陈述。

Herbert Simon在其著作《The Sciences of the Artificial》（1969, 1981, 1996）中独立地提出设计就是一个搜索过程的主张。他提出的模型及相关讨论远比这里的要复杂。Simon乐观地认为设计过程就是搜索人工智能意义下的合适标的（只要有足够的处理能力到位），他也投身于严格化理性设计模型的筹划，因为这样一种模型对于设计过程自动化而言乃是不可或缺的先驱力量。他的模型仍然有影响力——即使到了今天，我们已经认识到，其原始设计中的“险恶问题”[5] 可以说是在人工智能中最没前途的候选之一。

在软件工程领域，Winston Royce对于因为采用“先写了再说”的方法而造成的大型软件系统的项目失败而深感震惊，于是独立地引介了一种由7个步骤组成的瀑布模型，以将流程加以整顿，如第3章的第1插图所示。事实的情况是，Royce是把他的瀑布模型当做一个假想的批

评对象提出来的，但是有很多人已经引用并追随这个假想的批评对象，他提出的更为复杂精妙的模型反而被晾在一边儿了。我在年轻的时候也犯过那样的错误，并在之后公开地为其忏悔。[6] 即使有那么一点儿讽刺的味道，Royce的7步模型仍然必须看做是设计的理性模型的基础性表述之一。

Royce强调，他的7个步骤是彼此泾渭分明的，需要分别地规划并各有专人负责。其中确有重叠的部分，但这部分被仔细地限定在一定范围之内：

各个步骤的顺序安排乃是基于以下的概念：每前进一步，设计就变得更加详尽，在（邻接的）前一步和后一步之间有一定的重叠，但是在序列中距离较远的步骤就不太会有什么重叠之处了……我们拥有一种有用的退路，这往往可以将早期工作中仍然可资利用的以及得到保留的部分尽可能地最大化。[7]

设计空间可以表达为树型结构的观念，是在Simon的著作中隐含地提出的。这个观念在Gerry Blaauw和我合著的《Computer Architecture》一书中有具体的描述和图解。[8] 在该书中，我们将处理器架构的设计方案以严格的层阶架构形式组织在一个巨大的树型结构中，以83个链接子树来表示。有关闹钟的设计树可以作为一个简单的例子，如图2-1所示。其中，人们可以看到两种分别以开放和封闭的根型表示的分支。第一种，如“ 闹铃 ”节点所示，表示的是细分单元，每一个分支都是一种特定的设计属性，且必须指定其值。此即所谓属性分支。而备选方案分支，由“ 铃声 ”节点所示，则枚举了所有的备选方案，人们必须从中选择适当的方案。

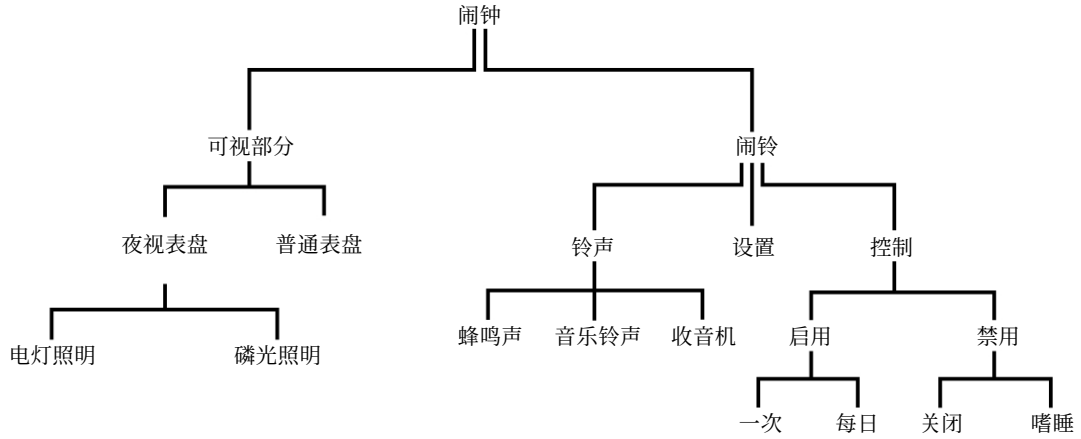

图2-1　闹钟的设计树（部分），选自Blaauw和Brooks(1997),
所著的《Computer Architecture》的图1-12和图1-14

理性模型有哪些长处

与“ 先开始编码再说，先开始构建再说 ”的行为相比，任何将设计过程系统化的工作都可以视为一种长足的进步。它为设计项目的规划提供了清晰的步骤。它为日程规划和进度评估定义了明确的阶段里程碑。它为项目组织和人员配备指明了方向。它改进了设计团队的内部沟通。

而在设计团队和其项目经理之间以及项目经理和其他利益攸关者之间而言，它对于沟通的改进尤为显著。新手很容易就可以上手。掌握了它，新手在面对分派给他的第一个设计任务时，就知道从何入手了。

理性模型在特定的情形下会体现出更多的长处。在项目早期就给出目标的显式陈述、相关的必要条件以及约束说明，这有助于避免让团队陷于举棋不定的局面，也促使团队形成关于项目宗旨的统一认识。在开始编码或正式的制图工作开始之前做好整体的设计过程规划，就能够规避大量麻烦，也避免让许多努力付之东流。将设计过程打造成对于设计空间的系统化搜索，可以拓宽设计师个人的眼界，并把他们的视界提升到远远超过其先前的个人经验的程度。

不过，理性模型太过简化了，即使是Simon洋洋洒洒、高度成熟的版本也不免于此。因此，我们必须对其缺陷加以审查。

注释

1. 按照Simon(1981)《The Sciences of the Artificial》的习惯，在整本书中我采用“man”作为一个一般性的名词加以使用，两种性别都包括在其指代的对象中，同样“he”（他）、“him”（他的——形容词用法）和“his”（他的——名词用法）也一律作为兼具两性的代词。我觉得继续使用符合传统的，把女性和男性平等地置于这些一般性的代词指代之中的做法十分亲切，这好过生硬地使用一些画蛇添足的，因而也是分散人们注意力的噱头。

2. 寻找“最低限度满足解”就是找到足够好的解，而并不一定是优化解（Simon(1969),《The Sciences of the Artificial》）。

3. 但是参见Akin(2008)的“设计中的变量与不变量”，它从DTRS7协议中发现证据表明建筑架构设计师们往往会在各个层次中横向地搜索若干个备选方案，而工程设计师们则主张从初始解决方案的提案出发，展开深度优先搜索。

4. Pahl和Beitz(1984),《Engineering Design》。

5. Rittel和Webber(1973)的“规划的一般理论中的困境”，它正式地定义了这个词。它也在以下的词条中有着详尽的讨论：http://en.wikipedia.org/wiki/Wicked_problem。

6. Brooks(1995),《人月神话》，265。

7. Royce(1970),《大型软件系统开发的管理》，329。

8. Blaauw和Brooks(1997),《Computer Architecture》。