设计

对于一个伟大的软件而言，最重要的是在正确的时机，推出正确的产品。也就是说，你必须知道如何准时推出软件，而且能够抓住顾客内心深处的需要，这就愈能够体会到顾客的内心，这个软件就愈伟大。软件的设计——每一位团队成员都必须参与——这表示团队整体对功能需求的了解程度。总而言之，软件设计的第一要诀是：将全团队中最好的想法组织起来，去满足顾客内心最深处的需要。

在设计过程中，最困难的是让每一个人最好的想法或建议、最棒的创意或灵感表现出来。但是这样做的代价是绝对值得的。想想看：只有两三个人做产品设计的话，加起来的智商最高只有230，或是250，有10个人的话，结合的智商也许可以到1 300。算一下你的团队有多少人，你能结合的智商愈高，做出伟大软件的潜力就愈高。在设计或开发过程中的每一个阶段，让创意充分发挥是非常重要的。一个伟大的教堂，只需要一位伟大的建筑师，但是软件的设计却需要上百人或上千人的智能来为它创造价值。

然而，让每个人都充分发挥创意而又不牵绊住其中的天才，是一件相当复杂的工作，也是领导者所面临最大的挑战，当然这不是软件业中惟一的挑战。教育可说是领导

者或管理者最主要的角色：带领团队做案例研讨，带领大家思考如何解决一切的疑难，让每一件事都在该做的时候做好。

法则 19

Go for greatness
追求卓越

如何让组员生活在美的环境中，陶冶他们对美的素养？让组员阅读什么样的书籍或给他们什么挑战，可以加强他个人的成长？管理者应该培养组员对美学的概念，可以把千百年来人类的文明当作美学的来源，而重点则放在软件的设计。人类对美的感受是非常复杂的，几个世纪以来不断有人研究这个主题。因此，不妨利用前人探索的经验当作基础，以下我将引用一些前人的理论，来引导本章的讨论。

除了美学之外，我们也可以利用历史的角度：暂时忘掉这个信息时代，想想看历史上有什么人物或事迹（时间较近的排在前面）令你觉得很崇拜或很感动，改变了你的想法或态度？你团队的工作是什么？两者有没有类似的地方？这个历史故事能不能激发人们的潜能，而开启另一扇创意之门？这个创意是不是完全没有人想到过，而且会导向一个没有人想象过的未来？你可以思考这一类的问题，用历史的眼光来分析一下你的工作，也许会有意想不到的结果。

法则 20

State your theme

设定主题

有几件伟大的著作影响了我对伟大软件的观念：乔治·桑塔亚那（George Santayana）1896年的经典名作《美感》（The Sense of Beauty），以及较近的乔治·史汀尼（George Stiny）与詹姆士·吉普（James Gip）于1978年合著的《美学法则》（Algorithmic Aesthetics，University of California Press出版）。尤其是史汀尼的著作，对于我们现在所讨论的软件美学有很大的关系。史汀尼对美学的定义如下：

> 美学所关心的问题是如何描述、诠释和评价一件艺术作品(work of art)，以及如何创造一件美的艺术作品。

在德惠克·帕克（DeWitt H. Parker）于1926年所著的《艺术分析》（The Analysis of Art，Yale University Press出版）中提到六项美学的准则（Criteria of esthetic form），很适合作为分析软件设计之用，而我要求我的团队把它们当作设计时的标准。

> 德惠克·帕克的六项美学的准则是：统一、主题、变奏、演进、平衡、层级。

根据帕克的看法，**统一性（unity）**是伟大艺术作品最主要的原则。而我也曾经在许多件伟大的软件作品中看

到这项特质。如果我们将帕克对统一性的解释套用在软件设计上，我们可以说，一件具有统一性的软件，每一个小组件对于整体的美感价值都非常重要，而且每一个美感元素都应该出现在作品中，缺一不可，而且不应该出现的东西、对美感没有贡献的东西，绝对不要出现。因此，软件设计应该是把顾客想要的东西全部纳入，而顾客不要的东西统统排除，由于软件中所有的东西都是需要的，顾客对于软件的使用不会被干扰，而去注意不必要的东西。所以，追求软件的统一性，是软件设计的首要目标。也许你在别的领域（也许是创造一件艺术作品）的工作中，也曾经由于直觉或别的理论而运用统一性，而我认为帕克的六项原则对于软件的设计非常有用。

软件的**主题**（theme）会主宰设计的基础观点，也是软件价值的主要根源。因此，你必须在团队中明确地传播这个主题，让开发人员和行销人员对主题有非常透彻的了解才行。软件的主题事实上是目标的同义词。目标愈明确，造成的的冲击就愈大，因为你可以将模糊降到最低，而目标在每个人心目中造成的感受与解读会更一致，整个设计过程就愈平顺。但是主题决定之后，你还得注意与主题无关的部分都要删除掉，即使开发人员认为这一部分很重要，

或者这是你一贯的信念，都还是得忍痛牺牲。

产品的销售信息会由主题衍生。精明的观察家只要看到主题就能抓住信息。信息只是补充说明主题的意义，如果主题模糊或是不只一个，再好的销售信息也没用（大家都明白这个浅显的道理：产品若没有鲜明而惟一的形象，广告再多也没用）。产品的主题根源于你的对市场的观念，以我Visual C++的例子来说，我们对市场的观念是：大家都觉得C++太难学，于是我们设计的主题是：让使用者容易学习C++，我们的信息自然就是：Visual C++把C++变容易了。

重点是产品的功能特色不能像是一袋子随便抓过来的东西，应该把与主题无关的东西都删掉，而且你的目标也必须符合统一性（unity of purpose）才行，这一点是与主题互为一体的两面。将资金投注在这个目标上，让所有的人都完全明白这个目标，并且为这个目标努力，做得到这些的话，你的产品就会完全包含这个目标。

专心致力于主题

我不知道以下的故事是真是假，但我很喜欢它，而且经常说给我的组员听。

有一家电子表格公司做了一项研究，结果发现到使用者大约每打20个数字就会想用这些数字画张图表。这项研究发现，大多数的人在用电子表格时都有这样的行为模式。

所以他们研究再研究，开发再开发，全心全意地努力让产品有这样的功能：可以利用很少量的数字，很容易地产生图表。那些用电子表格绘图的使用者看到这项功能特色都非常高兴，认为只要有这个功能就值得买下这套软件。

另一个类似的故事（我也不知道是真是假）是一个做家庭财务软件包的团队。他们研究过市场之后发现：这项产品一定要让顾客立即得到好处，否则就卖不出去，而且消费者不会再买升级版。他们决定要让任何一位完全陌生的使用者，都能够在打开包装后十分钟之内得到结果，从安装、操作、输入、输出，一定要又简单又快。

所以他们派人到软件商店去，征得顾客的同意，跟着他们回家观察他们的使用状况，巨细靡遗地记录下来：包括他们拆开包装之后会先看什么东西，在安装和执行软件的过程中会遇到什么困难等等之类的，

以提供产品改进的依据。分析研究之后，他们找到了数十种加强顾客满意度的机会，然后他们在往后的几个版本中逐步实现这个目标——十分钟内的满意。产品终于成功了！

变奏（variation）是将主题稍加变化润饰后，重新表现一遍。在主题表现过之后，为了持续吸引使用者的注意力和兴趣，变奏是以另外一种方式来诠释主题，加强使用者对主题的理解和欣赏，使他对主题留下深刻的印象。

演进（evolution）的意思是用前一部分来决定后一部分，就像是学习的过程应该是先入门后进阶一般，由浅入深的变化会让人更容易接受，更喜欢学习。如果软件作品的前后能够如此呼应，通常会有满意的结果。

平衡（balance）是对软件中各项组件都不偏废或过度强调。例如，正好相反的两个对象，应该给予相同份量的说明。

层级（hierarchy）是指软件作品中的各个元素，依照它们的重要性与大小给予合理程度的比重。层级与平衡的概念很接近，层级可以说是建立与衡量平衡的方法。如果主题是在层级的最顶端，则以下各层级的同层组件都应该彼此平衡，同层级的组件对主题的支持力也应该相等，

愈近顶端的层级对主题的支持力量愈大，以此类推。

美的特质

1975年，盖·瑟西罗（Guy Sircello）在他的著作《美学新主张》（A New Theory of Beauty）中，提出了一个有关于美感认知的学说，相当有趣。我们不谈他对物体特性的质与量的详细划分，他提到一些有关美学感受的理论，非常深得我心。瑟西罗认为我们之所以感受到美，是因为这个物体有一种以上的特质很美；瑟西罗进一步解释说，惟有一项特质在作品中被特别强调，才会是有美感的特质，而一个物体惟有包含一项以上的美感特质，才会被人们感受到它的美。足够的美感特质不一定保证让整个物体显得美，但是一个没有美感特质的物体绝对美不起来。

瑟西罗的学说或多或少解释了为什么有一些软件叫好不叫座，这些失败的软件什么都有，想要大小通吃，有一大串的功能特色就是没有主题，结果没有一项特色能够使产品鹤立鸡群，就无法吸引顾客的注意，当然注定要失败。

法则 21

Minimize dependencies
不要倚赖不确定的事

尽量减少团队需要而又无法控制的事情(dependency)。项目开始的时候，决定允许倚赖的事情愈少，最后就会愈顺利。一般来说，程序设计的效率是不会太高的（也许是由于不熟练或是错虫太难抓），即使你努力加班在时限内完成了自己份内的事，别人也可能无法做到这样。

因此，在设计时就得考虑这种必要而又不确定的事，要知道这种依存性有可能会吃掉大量的成本，只有在很重要或是非不得已的地方才允许这种倚赖，让成员们明白其严重的后果，尽量配合协助，并事先评估倚赖之事失去控制的可能性，以及会有什么影响。

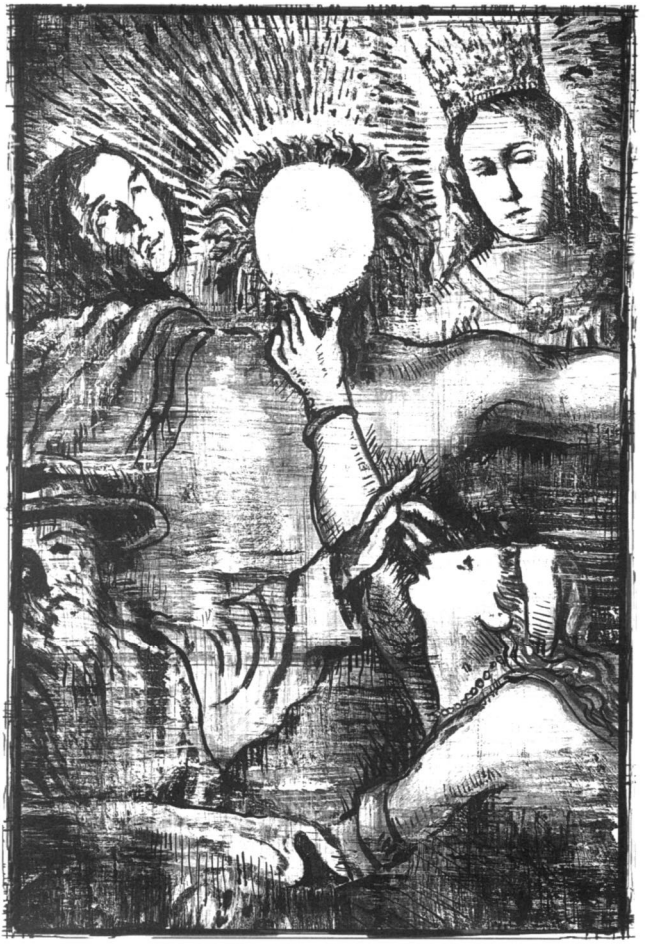
法则 22

Propitiate the gods

平息顾客的愠怒

在项目进展的过程中，总有一些依存性或外在的不确定性因素可能会拖垮你。你必须在其中找出最有可能绊住你的因素，事先研究好万一发生问题时你该采取什么步骤。

先检视一下产品的功能特色，其中有没有不明确的、或是对顾客满意度没有太大意义的部分，在时间来不及时可以牺牲掉这些。不错，可能会有顾客不满意你没有做这项特色，但是只要你能如期推出新版，那顾客就不会放弃你，并期待在下一版中见到他要的特色。



平息顾客的愠怒

法则 23

Portability is for canoes.

软件的可移植性

对于大部分的软件厂商而言，做到跨平台（multi-platform）的支持是相当困难的。即使不考虑每增加一种支持平台所增加的开发成本，在品保方面所增加的工作负担也是呈指数增长，再优秀的品保管理也无法真正解决这个问题。最好的办法是要求系统软件厂商提供工具支持，然后慎重小心地决定你要支持的平台，数目愈少愈好。

但千万别选错了，那会是你的致命伤。

法则 24

Design time at design time

在设计时将时间因素考虑在内

在设计时就将时间因素考虑在内，千万不要先设计好再决定要花多少时间才能做出来。时间是你最大的限制条件。

产品设计的目标一定要完成才能推出，你不能把做到一半的软件给顾客使用。开发人员和管理者在做产品设计时很容易忘记考虑时间因素。正确的做法恰好相反，你应该在设计阶段就把时间当成关键因素，当你在考虑替代方案时，时间短的加分，时间长的减分。通常只要把时间因素纳入设计时的考量重点，你就能够缩减开发的时间。

如果你的设计并不一定要产品如期完成？别傻了，还是如期完成最重要，比什么伟大的理想（可惜实现不了）重要太多了。