第2章　Web设计进程

创建一个好的Web站点极具挑战性，从外观设计到数据库集成，那么多不同的部分都会留下很多犯错误的空间。为了减少Web项目失败的风险，我们需要有一个进程模型来指导开发过程。不幸的是，很多Web设计者采用了一种可能被称为NIKE的开发方法——他们只是做，而很少考虑前景和计划。这种建设网站的过程是不符合方法学的；站点的目标定义得很松散，整个进程依靠的是直觉，没有严格的过程定义而缺乏可预见性。以这种方法开发的站点像植物一样，它们自然地生长，偶尔会变成美丽的花卉，但更多的情况却是一团乱草。复杂的Web设计需要认真地计划。应该采用适当的进程模型或方法学来指导Web的设计和开发。

2.1　进程需求

与20世纪60年代相似的“软件危机”今天也同样出现在Web开发中。几年前，大多数Web站点只不过是数字化的小册子，或被人称为“小册子软件”。开发那样的站点并不需要大量的计划——经常是简单开发好界面后，再用内容填充就足够了。今天的站点变得更复杂更庞大。随着电子商务和动态网页的引入。站点已经从“小册子软件”变为完全的软件应用。然而，很多开发者仍然还没有采用一种健壮的开发方法，继续依赖于特定的方法。

> 注意　“软件危机”指的是在一段时间内，在软件开发领域，硬件能力的提高容许构建更复杂的程序。然而，开发和维护新的程序极具挑战性。因为过去几乎不采用什么方法学，结果很多项目失败，导致专家宣布“软件危机”发生。结构化的开发方法和“自顶而下”的设计方法的引入可以解决这种危机。

Web开发实践的危机广泛存在。不像过去内部的客户机/服务器项目会得到保密，现在很多失败的Web项目都会受到大家的关注。始终处于开发状态或者即将需要开发的网页的数目显示了Web站点设计的糟糕。不幸的是，黄黑色的仍在建设的标志以及动画的手提钻很少被删除。如果从它们的内容和最后修改日期来判断，一些站点好像几年来一直处于建设之中。一些半死的站点，被陈旧的内容和风格陈旧的网页所充斥，过时的技术、断开的链接体以及运行不当的脚本，这些站点就像在线的鬼城一样。不要把这些错误简单地归结为疏忽和印刷错误。断开的链接体是一种严重的错误，想像一下如果某个软件存在功能缺失的菜单，情况又会怎样。

站点出现这些问题的原因多种多样。一些站点逐渐恶化，是因为站点设计者感到厌倦逐渐退出。另外一些站点则是因为站点被废弃或资金被撤离。还有一些站点未能完成，则是因为站点太复杂拖垮了设计者。有时候，开发者未能很好地理解他们使用的工具，或未能充分地理解媒体的限制。项目失败的原因多种多样，很多网站的倒闭则是因为项目太冒风险。

2.2　特别的Web进程

通常Web站点的开发方法很简单：实现网站，用浏览器完成外观测试并发布。这与软件小项

目开发中的先开发后测试的过程非常相似。并不令人惊讶的是，用非正规的方法开发站点会遇到很多问题。今天的Web开发太快以至于仅分为两个阶段：实现和发布。值得注意的是，很多Web开发工具鼓励这种在线的设计方法。一些工具鼓励开发者直接从构建用户界面着手，并逐渐用自动化工具增加功能。另外一些则先写代码再添加用户界面。勿须质疑的是，考虑到Web的时间需求，Web开发的速度非常重要。然而发布一个蹩脚而设计拙劣的站点也会成为问题，尤其是当用户对站点中出现的问题感到沮丧时。

在软件行业，很多专业人员倾向于认为非正规的方法仅适合于小项目，通常只有一个程序员，并且将来的软件维护工作量非常小。经常用这种缺乏计划的方法开发程序会产生糟糕的编程逻辑——所谓的“意大利细面条式的代码”。这种代码极其难以维护，因为除了初始的设计者外，没人能够解开这个疙瘩。即使是原来的程序员也会随时间忘记这些代码的意义。Web站点存在同样的问题。生存周期很短的小型Web站点，通常由一个人设计时很少采用方法学来进行。观察这些站点的HTML网页、JavaScript和导航结构，会发现“意大利细面条式的代码”也被采用，不过增加了一道被称为“杂乱的HTML标记色拉”的副食。

在Web开发项目中计划能够解决一些问题。不幸的是，对特别的Web进程来说，计划仅仅局限于简短的交流，简单而又不完全的潜在内容的收集，或者一些仓促写就的流程图，花在计划上的时间与花在实现上的时间相比之下微忽其微。当然，也可能计划得太多而患上“分析麻痹症”，这阻碍了网站的建设，但这种情况并不普遍。一定要记住，计划的时间量正比于计划的复杂性，应付项目管理挑战的关键在于创建一个正规的进程，借助它进行计划、测试，并用结构化的方式配置站点。

2.3 基本的Web进程模型

为了减轻站点建设的困难，应该采用一种进程模型来描述Web站点开发中涉及到的不同阶段。通过使用指导准则、文档和确定开发步骤以确保每一步顺利完成。理想的Web进程模型能帮助用户解决站点的复杂性，减小项目失败的风险，处理项目中几乎不可避免的改变，在开发过程中快速发布站点并得到及时的反馈信息。当然，理想的进程模型也应该容易学习和操作，这是一个相当苛刻的要求，任何一个单一的模型都不能适应项目的所有特定需求。

Web站点开发中使用的最基本的进程模型为大多数人所熟悉，至少在思想上是这样的，因为它是可演绎的。基本模型从大的背景开始逐渐向特定的步骤细化，以便完成整个站点的设计。在软件工程中，这个模型称为瀑布模型，或者有时称为软件生存周期模型。因为它描述了软件生存周期的不同阶段，各个阶段逐步进行直到结束。模型的第一阶段是计划阶段，接着是设计、实现、测试，最后为维护阶段。每个阶段都有独特的步骤，但相连阶段的边界并不明显。进一步说，每一阶段并不总是有一个固定的目标。有时候，前一阶段可能会因为项目中未曾预料的改变而修改。步骤的实际数目和名称因人而异，但瀑布模型的思想如图2-1所示。

注意 尽管Web的开发模型可能是相同的，但仍有很多Web开发者认为他们创造了新的模型。他们在自己的站点上把它作为没确定归属的专利介绍，而实际上并没有新的东西。

无论是五个步骤还是七个步骤，或者名称是复杂还是简单，需要记住，真正重要的是模型能否加速站点的开发或提高最终结果的质量。

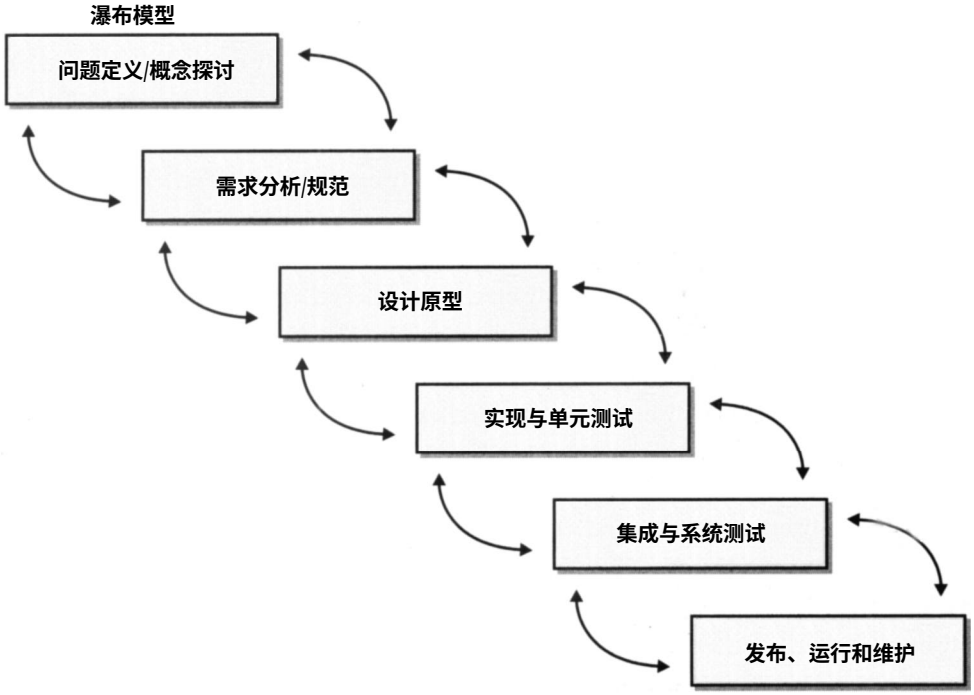

图2-1 瀑布模型

单纯的瀑布模型的好处在于它使得用户能够在前端计划一切，这也同时是它的最大弱点。在Web项目里，完成一个项目需要做些什么具有非常大的不确定性，尤其是当 Web开发者没有太多的经验时。另一个与这个模型相关的问题是，像软件开发一样， Web开发的每一过程也相互重叠，并且前后相互影响，而且经常不得不重复。不幸的是，瀑布模型过于严格。如果发生过多的改变，可能要求开发者停止项目，并重复原来的过程。简言之，这一过程不能很好地适应变化。然而对于Web站点来说，瀑布模型因为便于理解和实施，仍然继续使用。进一步说，进程模型中每一阶段的分离有利于管理，因为它们便于监理和一步一步地实施。

2.3.1 修正瀑布模型

瀑布模型的一个重要方面在于它要求在前端做计划。然而由于在进程中需要有所有的步骤，开发者早期仓促地略过每一步骤，结果事后不得不重复，或者继续建造有缺陷的站点。进程也过于严格，不支持进一步的探索，导致不必要的风险。一种可能的提高方法是在瀑布模型的早期阶段花更多时间，并重复几次，在进入设计与实现阶段后反复挖掘目标和需求。因为进程的周期性，经常发现伴有涡旋的修正瀑布模型更加自然。当你解决带有很高的不确定性的项目时，图2-2显示的带有涡旋的修正瀑布模型是个很好的主意。

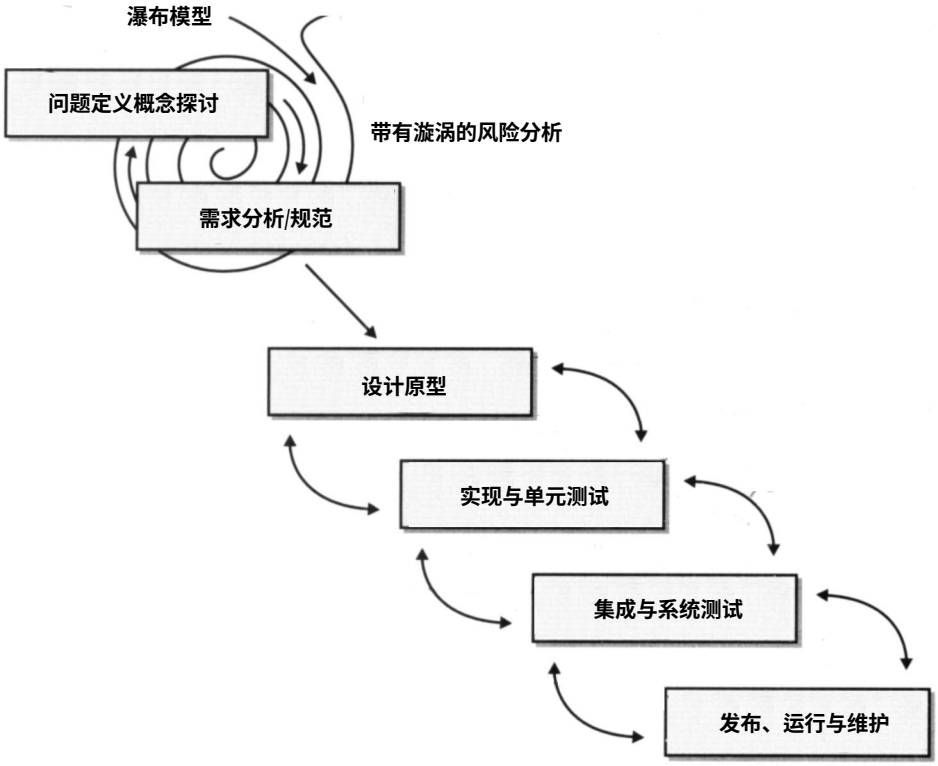

图2-2 带有旋涡的瀑布模型

2.3.2 联合应用开发模型

最后提到的软件开发进程模型对站点开发很有意义，它就是联合应用开发设计，简称JAD(Joint Application Development)，也称为进化原型法，原型系统通过一系列的演变得到最后的形式。原型不是创建用来测试理论的一个模拟站点，而是为用户创建的。用户直接的反馈信息将用来指导产生新版本的站点，反复如此，直到系统最终定型。 JAD 的基本概念如图 2-3所示。

JAD的很多方面非常适合Web开发，尤其是当非常难以确定项目的规范说明书时。与瀑布开发模型相比，JAD模型是渐进的，因此它也显得很快。然而 JAD也有着严重的缺陷。首先，让用户看到尚未完成的站点会危害开发者和用户之间的关系。即使让用户参与指导项目，我们也一定要记住用户不是设计者，如第 1章所指出的，这种Web设计指导准则会因为用户提出不合理要求而偏离轨道。以 JAD运作的Web项目很难作出预算，因为修正的版本数很难预测。如果用户变幻无常，成本会螺旋上升而失控。记住，隐藏在 JAD之后的核心概念就是在得到正确的设计之前反复试验。撇下它的缺陷不说， JAD在Web开发中有自己的位置，尤其在软件维护项目之中。然而，最初的JAD项目开发最好让有经验的开发者来完成 —— 尤其是那些能与用户很好沟通的设计者。

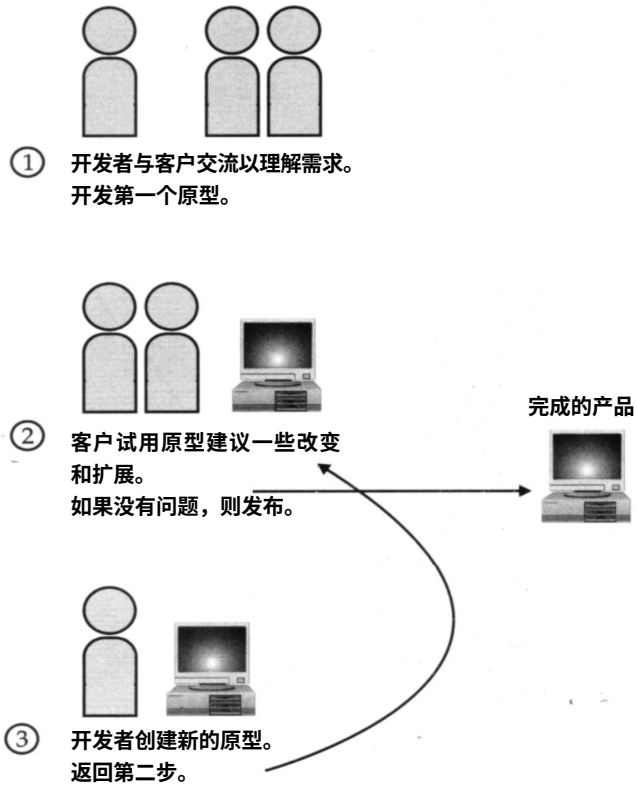

图2-3 实践中的联合应用开发

其他一些指导Web开发的方法也讨论过。还存在一些能为用户服务的方法。记住，建设站点是先标识好等待解决的问题或达到的目标，并以一致和启发的方法得到结果。站点的开发应该是挑剔的和深思熟虑的，而不是随便的和被动的。一个挑剔的方法并不意味着完全抛弃机遇和灵感，相反，它提供了机会。设计者不应该把站点工程的一些概念当作限制的因素，而应该把它们当作指导准则。

2.4 Web站点项目的途径

理论上，Web站点工程很有道理，但在实践中，它是否奏效呢？答案是完全奏效。然而，由于Web是一个新兴的领域，站点开发很少一致，如显著的时间限制以及项目不断改变的特性。开发者应该小心地继续。为了指导开发，在项目之初，就应该采用某个进程模型。如果站点是崭新的或增加起来非常困难，就应该采用瀑布模型或带有涡旋的修正瀑布模型。如果项目是有关维护的，比较简单并有很多未知的因素，那么联合应用开发方法就比较适合。不考虑项目本身，第一步通常都是相同的：那就是确定项目的整体目标。

2.5 目标和问题

很多站点因为缺少清晰的目标而最终失败。在 Web开发的前几年，很多公司创建站点的目的

很单纯，就是为了显示公司有站点。不知何故，没有站点的公司会被认为缺乏改革精神，不能作为市场领导者。有站点的竞争者会被认为危险。很多时候，站点很少赢利，而建设的原因仅是为了显示公司的存在。随着Web站点的盛行，创建站点的原因越来越明显。今天，站点的目标变得更重要，而且显示的方式越来越清晰。然而并不能因此主观地认为理念统治了站点——很多的站点项目开发纯属出于乐趣或因为察觉受到了危险，而不是为了解决真实的问题。

为Web站点提出一个目标并不很困难；问题在于提炼目标。警惕诸如“为用户提供更好的服务”或“通过开发在线市场而赚更多的钱”这样非常模糊的目标。这些可以作为项目的合理想法或一般目标，但目标需要更详细的陈述。好的目标陈述可能有以下的内容：

- 建立支持用户的站点，通过提供全天候的用户问题答复，提高用户满意程度，从而降低25%的电话技术支持。
- 创建一个在线的汽车零部件商店，每个月直接向用户销售一万美元的零部件。
- 建立一个日本餐馆站点，通知用户有关时间、菜单、氛围以及价格等信息，以鼓励用户通过电话预定或访问站点。

记住，以上三个目标的陈述中，有两个目标是可度量的。这在做实际的财政预算时，非常便于确认项目的成功与失败。第三个站点目标的陈述无法度量。这么做非常冒风险，因为不能让别人确信站点是成功的，或以某种方式度量站点。在餐馆信息站点中，站点访问者的预定数目或用优待券来度量站点访问人数会很有帮助。考虑以下修正的目标：

- 建立一个日本餐馆站点，每个月通知300个潜在用户有关时间、菜单、氛围以及价格的信息，以鼓励用户通过电话预定或访问站点。

简单的增加特定的访问人数会让目标更起作用。通过规定期待的访问人数，餐馆拥有者可以在效率相同的调查速度下比较通过印刷品或收音机做广告和运营站点之间的优劣。

2.5.1 集体讨论

一般来说，提出目标是非常简单的事情。最重要的问题是让目标的陈述简明和可现实。在很多Web项目中，有在站点中包容一切的倾向。记住，一个站点不可能满足所有人的所有需求；在心里一定要有特定的用户和特定的任务。为了确定目标，集体讨论是必要的。集体讨论的目的是尽可能提出有关站点的想法。在集体讨论时，一张黑板是非常有用的，可以用来快速记下和修改有关站点的想法。

通常，集体会议会偏题，因为参与者离题太远或讨论了太多的关于站点的哲学问题。在这种情况下，最好集中于大家一致感兴趣的问题。通过让大家讨论在站点中不希望见到的特性，以确定关于站点的一般设计哲学。让参与讨论者意识到他们不希望站点太慢，或难于使用，这些通常很容易。一旦集体统一了一致目标，哪怕这些只不过是一些关于站点不应该太慢的看法，未来的探讨以及关于站点应该做些什么的陈述都会更加顺利。

> **注意** 在指导重做某个站点的项目时，一定要注意不要召开集体会议来斥责既存站点中出现的问题，除非项目中的参与者与站点没有任何关系。一个万无一失的防止项目偏离的方法是让站点的原来设计者来为自己遭到的批评进行辩护。记住，人们不得不设计站点，所以组织一支积极的设计队伍是非常重要的。

2.5.2 缩小目标

在集体会议中，所有想法都是很了不起的。会议的要点是挖掘各种各样的被称为“愿望清单”的想法。“愿望清单”就是描述各种不考虑价格、可行性、可应用性的有关站点的想法的文档。然而，最终的“愿望清单”会缩小为有关站点的合理和合适的部分。这对有着各种各样可能目标的站点来说非常具有挑战性。例如，考虑某个公司的站点，包括产品信息、投资者信息、新闻稿、职位申请以及技术支持等部分。每一个负责相关部分的人都会认为他们的那部分是最重要的。每个人都希望把他们的那部分的链接体放在主页上。在那么多方案中权衡确实是非常困难的。

另外一种缩小目标的方法是使用小的纸片或 3×5的卡片。把每一个想法写在卡片上并堆起来。接着再让每一个人依据重要性，每次从卡片堆里抽出一张。当然，一定要限制从卡片堆里抽出的卡片数量。以这种方式非常有希望挑出重要的想法。不幸的是，依靠集体，这种运作很可能失败——尤其是当参与者在各自的领域扮演着很重要的作用时。

2.6 访问者

缩小目标的最好方法是始终考虑访问者。集体会议所需要的与用户所想得到的并不一定一致。首先要做的是精确描述站点的访问者以及访问站点的理由。然而，不要寻找“美国在线”上的某个名为Joe Enduser或者某个偶然使用56k调制解调器访问你的站点的访问者。对大多数站点来说，这种用户是无法标识的，而大多数用户在心中都有一个特定目标。首先考虑一下你的用户是些什么样的人，并向他们问以下问题：

他们在什么地方？
他们有多大年纪？
他们的性别是什么？
他们说什么语言？
他们技术熟练程度如何？
他们采用什么样的因特网接入设备？
他们使用什么样的计算机？
他们可能使用什么样的浏览器？
接着，考虑用户访问站点时干什么：
他们怎么访问到站点的？
用户想在站点上实现什么样的目标？
他们在什么时候访问站点？
他们会在站点上呆多长时间？
他们从哪个网页退出站点？
他们什么时候会再返回站点，是否可能？

当你可以从这些问题的回答中描述出用户的特性时，你应该能够快速地确定访问你的站点的可能不只是具有单一目标的单一类型用户。对大多数站点来说，有很多类型的用户，每种类

型的用户都有着不同的特征和目标。

用户特征

最好的了解用户的方法是和他交谈。如果可能，你应该直接和你的用户交流以验证关于用户特征和想法的设想。一个调查可能是合适的，但现场交流更可能提供了探索超越预先确定的问题的可能 。不幸的是，现场交流或调查用户非常消耗时间，不可能考虑每一种特征或想法的用户类型。从用户调查、现场交流或有关用户的一般思考中，你应该建立综合但详细的关于用户的特征。考虑至少三种类型的用户。对大多数站点来说，这三类用户大致对应于没有经验的用户、有Web经验但不经常访问站点的用户、能力强经常访问站点的用户。大多数站点有三类用户，能力中等但不经常访问站点的用户群是规模最庞大的一群。确信分配恰当的比率给相应类型的用户群，从而以合适的份量考虑他们。现在开始命名你的用户。在每交流完一个真实的用户后，你可能想命名他，或采用一般的名字如名叫 Bob初学者、名叫Irene 一般用户和名叫Paul超级用户。接着利用前面章节的问题描述出各种综合用户的概况。尽量把每一个答案对归类。这样，如果交流的用户中有快速接入设备的很少，大多数的接入设备很慢，我们就把后者当作更为一般的情形。第3章将更详细地讨论一般的用户特征和个性化的特征。

一旦完成了对一般站点访问者的特征的描述，就应该开始设计访问方案。当名叫 Bob初学者访问站点时，他会干些什么？什么是他希望完成的任务？什么是他的目标？方案设计有助于你集中于每一个用户实际希望完成的任务。从这种实践中，可能发现你的目标陈述与用户想做的不一致。如果是这样，可能仍需要反复做设计。返回到你的初始步骤之中，根据新的信息修改你的目标。

2.7 需求

根据站点的目标和访问者的类型，站点需求应该有所不同。需要什么样类型的内容？站点应该具有什么样的外观？应该需要开发什么样的程序？为了满足访问者应该需要多少服务器？用户在带宽、屏幕尺寸、浏览器等方面有什么样的限制？等等。需求会显示站点的成本和潜在的实现问题。需求会显示需要多少开发者并显示缺少什么样的内容。如果需求相对未来的回报来说过多，就应该重新回到目标阶段，并重新定义访问者类型。进程的前三个阶段可能要重复多次，直到形成站点规划或规范说明书。

2.8 站点规划

一旦讨论了目标、访问者和站点需求，并形成了文档。就应该起草一个正式的站点规划。站点规划应该包括以下部分：

1) 简短的目标陈述。应该包括对站点整体目标的简短讨论和对站点成功与否的基本度量。

2) 详细的目标讨论。应该详细讨论并提供可度量的站点目标，以验证站点的收益。

3) 用户方案讨论。讨论用户各种各样的访问方案。首先从用户怎样访问站点开始，直到达到目标。本节包括对度量方法的讨论，如下载量、每次访问的网页数、填写的窗体数，以及诸

如此类的有关详细目标讨论的内容。

4) 内容需求。应该提供一个有关文本、图像以及站点中需要的其他媒体的清单。一个显示需求的内容、形式、存在性以及潜在的拥有者和创造者的矩阵是有用的，因为这个矩阵能显示哪些内容非常重要。简单的矩阵如表2-1所示。

5) 技术需求。提供站点所使用的技术的综述，如HTML、JavaScript、CGI、Java、插入件等等。技术需求应该与用户的能力直接相关。更多的技术上的考虑在第13章中可以找到。

表2-1 内容矩阵

内容名称	描 述	内容类型	内容格式	是否存在	所有者
管家机器人的新闻发布	“今日机器人”杂志探访的新的第7个系列的管家机器人新闻发布	文本	Microsoft Word	是	Jennfier Tuggle
软件协议窗体	简短的关于使用实验机器人的智能软件所承担的法律责任的描述	文本	纸张	是	Lohn P.Lawyer
手持超级计算机的屏幕快照	新Demo Company的Cray-9000掌上型电脑的图片	图像	GIF	否	Pascal Wirth
来自总裁的欢迎词	来自总裁的欢迎用户使用站点的简短介绍信	文本	Microsoft Word	否	总裁的执行助理

6) 外观需求。应该给出用户界面设计的轮廓。应该详细地给出站点与既存的宣传材料之间的关系，并给出用户在图形以及诸如屏幕尺寸、颜色深度、带宽等多媒体应用方面的限制。本小节也应该给出特定的字体和颜色使用的细节，但很多站点外观的细节应该留在开发过程中确定。

7) 发布需求。指出发布需求，特别是主机方面的考虑。该小节应该包括多少用户访问站点、一个典型网页会消耗多少页以及典型页的尺寸。即使那些仅是一些设想，也可能借此进一步对服务器和发布站点的带宽需求进行简短的分析。

8) 站点结构图。该小节应该给出站点的结构或详细描述站点不同部分的流程图。应该根据前面小节分析的各种各样的用户方案，给出每部分适当的标题和一般想法。站点每部分的组织也是重要的，应该始终进行提炼。站点结构的选择在第4章中会讨论，但一般来说，站点结构图应该与图2-4相似。

9) 资源。站点应该给出运行时需要的各种资源。度量方法可以用简单的人-小时法表达，并且应该与四种资源相关：内容、技术、外观设计和管理。

10) 时间表。应该结合前面章节描述的典型瀑布模型和前面叙述的资源，显示项目的进展程度。

11) 财政预算。财政预算主要决定于资源需求和发布需求。然而，市场成本和内容版权也应该包括在财政预算中。

站点规划的实际组织和内容由设计者决定。记住，规划的目的是在从事该项目的各种各样的人们之间交流站点的目标。不要忽略站点规划的撰写，尽管它很令人沮丧。如果没有这种文档，你就只能以一种进化或JAD的方式开发项目。进一步说，没有规范说明书，你不可能从投资商中获得任何现实的投标。然而，一个完成的计划并不允许你立刻着手实现。一旦规范书形成，

就应该审视一次。完成的规范说明书可能暴露不现实的估计，从而让你陷入反复修改初始目标和定义访问者特征的过程之中。如果不是，就可以进一步，沿瀑布模型开始原型设计阶段。

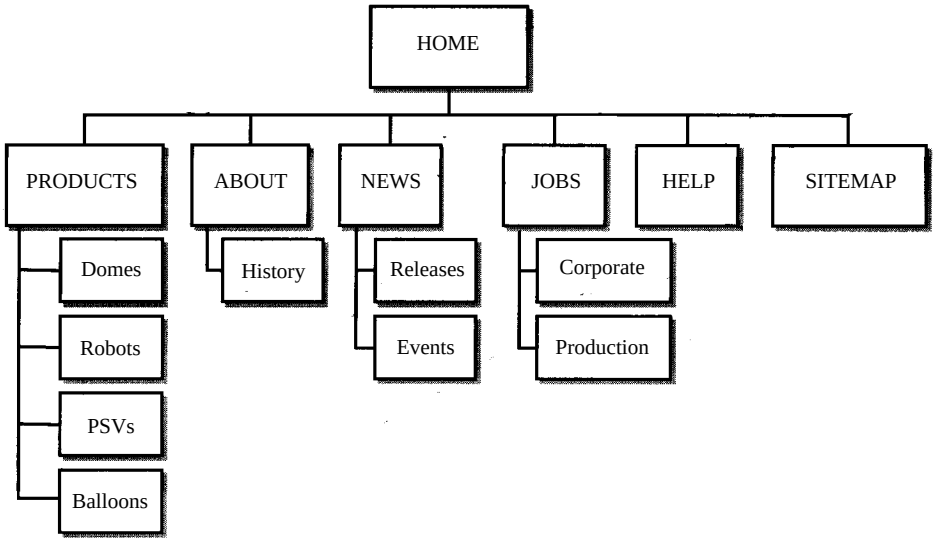

图2-4 典型的站点图

2.9 分割的设计阶段

原型设计阶段对大多数Web设计人员来说是最有趣的，因为它开始让项目成型。在这个阶段，应该同时开发外观和技术上的原型系统。无论如何，在建立原型系统之前，应该尽可能收集更多的内容。内容本身会影响站点并帮助指导它的形成。如果内容的基调非常严肃而外观却非常有趣和随意，站点对用户来说会非常奇怪。预先看内容有助于统一技术和内容。同时，也应该考虑到收集内容是站点设计中最慢的一个方面。 Web项目的参与者很积极地参与集体会议，但一旦需要他们在内容方面上做工作时，却很难找到他们。缺少内容在 Web项目中绝对是个大问题。应该预先准备好处理这个问题。

建议：在设计之前尽可能收集站点的内容。

2.9.1 块的组合

设计应该自顶而下进行。首先考虑一下用户是如何访问站点，又如何结束访问站点的。在大多数情况下，这意味着首先设计主页，再设计子页，最后是内容网页。

规则：外观设计应该采用自顶而下的方式，从主页到子页，最后是内容网页。

首先在纸上创建以块形式存在的模拟网页，如图 2-5所示。

块的组合允许设计者专注于对象的类型，以及不必在网页上考虑精确的位置和细节的组织。块分区法能有助于设计者创建网页模板，它会使后来的实现更加容易。确信你所创建的合成块

是满足Web浏览器窗口的约束条件的。浏览器边界的影响是一个非常重要的因素。网页块的组合效果图在http://www.Webdesignref.com./chapter2/中可以找到。一旦网页块的组合设计好后，其他类型的网页也可以按同样的方式设计。如果完整的方案已经很详细，应确信到每个网页块的路径是合乎逻辑的。这样，就可以继续下一阶段。

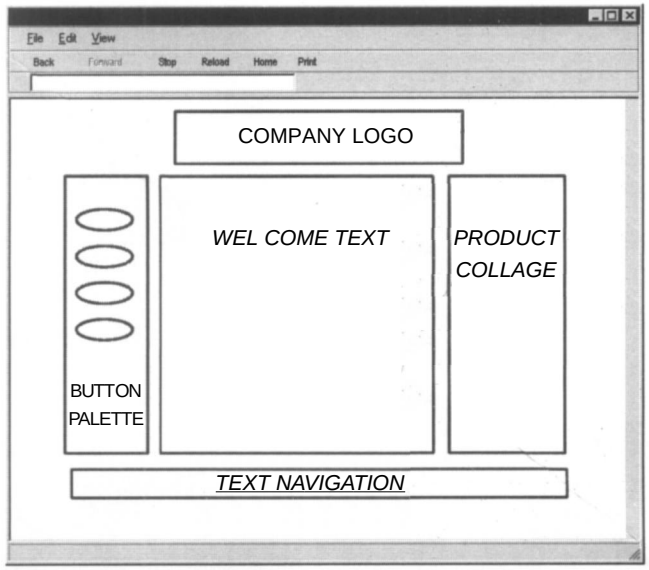

图2-5 主页的块组合

2.9.2 屏幕和纸张的组合图

下一设计阶段就是纸张或屏幕原型阶段。在这个阶段，设计者应画出草图，或实现站点某个典型网页的组合。无论是在纸上或屏幕上，一定要确定考虑了浏览器窗口或者屏幕的尺寸。块组合阶段使用的带有浏览器窗口轮廓的纸张可以用来做草图。

建议：在开发外观组合图时，一定要考虑浏览器窗口的边界效果。

画出各种各样效果的按钮、标题以及特征网页的草图，如果可能，在网页中提供一些文本说明，即一些缩写文本，或者真实文本。

注意 很多设计者在屏幕组合中仅使用一些类似文本的东西。尽管这种方法更有利于设计者专注于设计网页的要素，但如果可以得到真实的文本，应尽量使用它，这样有助于模仿最终的效果。

组合阶段提供了很大的创造空间，但应该提醒设计者在Web的可能限制下进行创作，并且同时考虑Web规范说明书中的外观需求，考虑到文件的大小、颜色支持情况以及HTML样式可能受限制，这可避免设计者提出外观极好但无法实现的网页。尤其是要防止为了追求艺术性而重新设计站点结构的行为。记住，站点规划会清楚地说明外观需求，包括市场营销的限制。第1章已

讨论了在形式、功能、目标和内容之间进行平衡的困难，当设计者企图在 Web技术、用户能力和站点需求等约束条件下满足自己的创造欲望时，这种平衡的困难会变得更加明显。典型的网页组合图如图2-6所示。

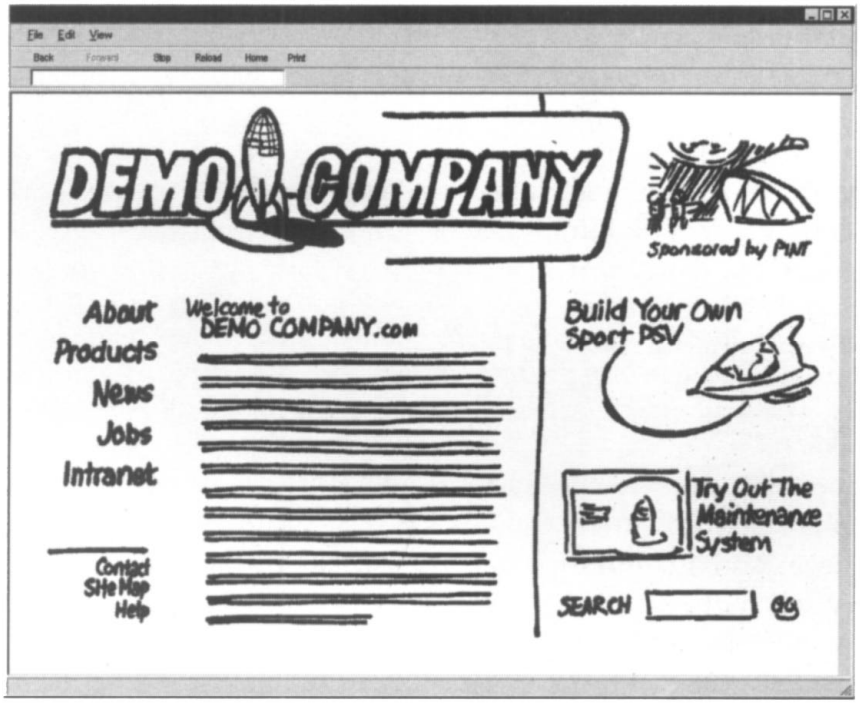

图2-6 演示唱片公司站点主页的纸上组合图

在创建原型时，应创建一个显示整个屏幕包括按钮、图像和文本的单个图像。把这个图像保存为GIF或JPEG的图像格式，并用浏览器加载测试在典型环境下它的效果。在这个阶段，应该尽量避免完全用HTML实现网页。你有可能不得不废弃设计，再次实现会很浪费。

一旦纸上或屏幕上的原型完成，就应该和用户一起测试。问一些用户，为了完成特定任务哪些部分会被点击，哪些按钮会被选择。确信不只是向一个用户显示原型系统，因为个体的品位是决定原型的接受程度的重要因素。如果用户对网页有太多的负面评价，应该考虑重新返回原来的设计阶段。在原型阶段，不能太关注你的作品。如果是这样，站点就不会以用户为中心，而是以开发者为中心。记住以下的设计规则：

规则：和原型系统保持距离。倾听用户并提炼的设计。

一旦提出了一个可以接受的主页设计，就继续进行子页设计和内容网页设计。一个典型的子页组合图如图2-7所示。

对于高度交互的站点，可能必须为特定的任务如购买或下载的每一步骤设计原型网页。这种类型的网页必须设计得更加详细，应该包括如窗体输入框的标题等其他非常有用的细节。一个交互性更强的纸上组合物的例子如图 2-8所示。

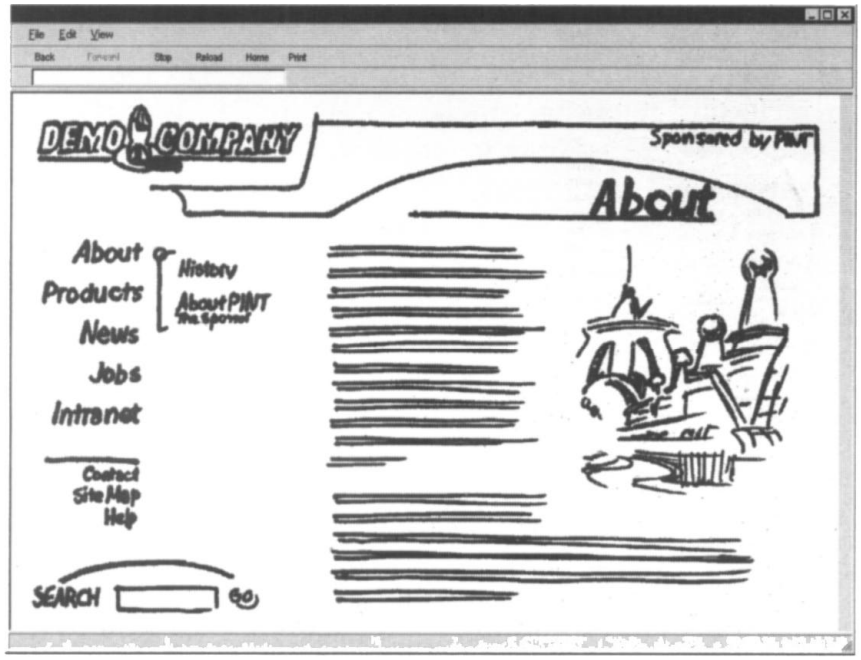

图2-7 Demo Company子页的纸上组合图

1. 用户浏览产品细节并决定购买

2. 初始的排序信息收集

3. 排序的位置

图2-8 电子商务的纸上组合图

尽管不是所有的站点都需要技术原型，高度交互的站点不应该只考虑界面原型，应该包括显示诸如数据库查询、个性化、电子商务等如何运作的概念原型。不幸的是，大多数情况下，

直到界面完成时才会考虑技术原型，这通常会导致工作的重复。

2.9.3 模拟站点

在完成所有原型后，就到了创建模拟站点或 alpha站点的时候。模拟站点的实现始于把组合体拆分再用HTML把网页组合在一起，也可能用层叠样式单。尽量用模板，这样整个站点的组合会快一点。尽量采用符号代替文本，除非有些地方需要用真实的文本测试方案。一旦模拟站点组合好，站点应该完全可以浏览，尽管虽然包含的只是一些空的内容和空洞的交互。一个模拟站点的例子可以在http://www.Webdesignref.com/mocksite中找到。在这个意义上，让一些用户来试一试这个模拟站点是个好主意。看看站点是否易于浏览以及响应是否快。如果用户在完成一些任务时存在困难，最好考虑废弃设计并返回软件开发的前一阶段。一般来说，这种情况不会发生，除非站点超安全设计或站点很少接受用户的反馈信息。

2.10 Beta版站点实现

一旦模拟站点可以接受，就可以着手实现真实的站点。真实的内容应该放在网页上，后端构件和交互的要素也应该与最后的外观设计集成为一个整体。这里要讨论的技术和实现问题太多，第10章到第13章会单独讨论这些问题。尽管实现看起来是项目最消耗时间的方面，但如果所有的构件已经收集好并且已经实现好原型，实际的站点实现做起来会比较快。

2.11 测试

对大多数程序员来说，测试可能是Web开发过程中最不喜欢的一个方面。在完成了所有艰苦的任务如规范说明书、设计和实现后，大多数程序员只是准备好发布。应该防止这种冲动。对积极的用户综合印象来说，测试是非常关键的。在发布后，不要促使用户测试站点。如果他们在实现好的站点上遇到一个问题，他们是不会原谅的。一定要记住以下设计规则：

规则：站点一定会存在一些问题，必须好好测试你的站点。

不幸的是，Web站点的测试经常就是简单的归为用浏览器快速地访问站点，或者检查站点的链接体。问题一定会在站点中存在，无论是什么样的。不幸的是，大多数开发者认为只要站点的外观看起来没有问题，它就是没有问题的。记住第 1章中所说的，站点设计不只是仅仅包括外观设计：必须同时测试站点的其他方面，就像以下规则所综述的：

规则：测试应该涉及站点的各个方面，包括内容、外观、功能和目标。

附录B详细讨论了站点的评估和测试，尤其是针对完成的站点。这里简单综述一下Web的测试。

1. 外观可接受度测试

外观可接受度测试可以保证Web的外观与设想的一致。浏览站点的每个网页，确信它们在样式、颜色和风格上一致。用不同的浏览器和分辨率或与真实用户一致的浏览环境来浏览网页。用浏览器快速地访问站点，看看样式是否一致。在浏览站点的同时，注意寻找样式不规则的地方。外观可接受度测试可能要求每一个网页都打印出来。记住不要专注于为在线消费设计的打

印测试网页。

2. 功能测试

既然大多数网页的基本功能就是在屏幕上显示自己，功能测试和外观测试在某种意义上经常重叠。然而，大多数站点至少包括导航这样的基本功能。确信检查了站点上每个链接体并校正了每个断开的链接体。断开的链接体应该当作一个非常严重的错误。确信测试了诸如窗体这样的交互元素。通常采用现实情况和极端情况这两种测试条件。通过输入明显的错误来测试窗体。记住，用户不会按照你所想的行事，尽量考虑一些不可预料的情况。

3. 内容验证

站点的内容细节很重要。确信所有的内容都是合适的且词语的用法保持一致。检查诸如产品名、版权日期和商标等细节。一定要记住检查单词的拼法。客户和用户会仅因为一个印刷错误而认为整个站点很糟糕。这种重要性怎么强调也不过分。进行测试的最好办法是把每一个网页打印出来并认真阅读每一行。

4. 系统和浏览器兼容性测试

在开发时可能就考虑到了系统和浏览器的限制，但测试时一定要验证。在浏览站点的时候，一定要用用户会使用的同样类型的系统和浏览器。不幸的是，大多数情况下测试用的系统比用户采用的系统一般要强大些。项目规划应该有详细的浏览器需求，一定要让站点用指定的浏览器访问时非常顺利。

5. 发送测试

检查一下站点发送得是否充分。在用户的真实条件下浏览站点。如果站点是为“美国在线”的调制解调器用户设计的，建立一个“美国在线”用户账号，用调制解调器测试站点发送速度。为了模拟站点流量，考虑用测试软件创建虚拟用户点击站点。这会模拟出站点的响应速度。测试时一定要用真实的服务器和大致相同的系统。一定不能低估站点发送的影响。在设计规范说明书时，如果对此考虑得不够充分，整个项目可能偏离轨道。关于发送条件的更多信息，请参看第14章。

用户接受程度测试

用户接受程度测试应该在站点看起来正常后进行。对于软件，这种测试通常成为 Beta测试。让用户使用站点，并做最后一次的评价。不要直到明显的错误被校正后再进行这种测试，就像下面规则所说的。

规则：用户测试是最重要的测试形式，不要在最后才进行。

因为用户测试最接近真实的用户，所以它是最重要的测试形式。如果问题没被发现，你可能不立刻纠正错误。如果问题不是很大，你仍可发布站点，而迟些时候再纠正。然而，如果发现了严重的问题，最好延期发布直到问题被 纠正。

2.12 发布和以后的问题

如果站点准备发布，不要放松 —— 你还没有发布。实际上，你的工作仅仅开始。现在是观察站点实际运作情况的时候了。站点是否符合用户的期望？站点的开发目标是否已达到？是否还

需要其他的校正？底线是站点必须继续运行。新的特征可能需要，为了适应技术更新而升级是不可避免的。为了满足市场需要而改变外观也非常可能。这样就开始了被成为维护的持续开发过程。一旦瀑布模型的最后阶段完成，就应该重新返回起始阶段，正如以下规则所述：

规则：站点开发是一个持续的过程：规划、设计、开发和发布，如此周而复始。

2.13 欢迎来到真实世界

尽管站点的开发过程看起来非常直观，但它并不是一直都很顺利。对真实世界来说，存在太多的变化。例如，考虑为某个人如老板或客户设计站点。如果某个人为创建某个站点掏钱，你可能需要把他的想法掺杂进去，而不管需求是否与用户的想法一致。一定要劝说别人在做决定时考虑用户的想法。一定要显示设计理论的优越而不是鼓吹规则。做好举一些你的想法的例子的准备。然而，必须做好接受你的想法被别人抛弃的准备。

注意 有经验的设计者会准备很多站点组合以备讨论。与为顾客准备的发型书相同，用户所想的是很难用词语描述。

大多数站点项目容易存在争端问题。不要指望任何人都同意。公司的部门会争夺控制权，这种情况经常出现在技术部门和市场部门之间。会有数不清的自称 Web专家的人给出建议，这会激起更多麻烦。如果某人的兄弟的朋友声称用微软的 FrontPage自动化工具在一小时内建好一个站点，不要感到惊讶。解决争端的唯一方法是耐心并尽量地说服别人。没有一个适当而清晰的规范说明书，开发者会发现自己处于一个容易受到攻击的不稳定位置。

一定要记住，遵循某个进程模型的目的是减少 Web项目中出现的问题。然而，一个进程模型不能考虑现实世界中的所有问题，尤其是关于人的问题。经验是处理很多问题的唯一老师。 Web项目中缺少经验的开发者会被鼓励摸滚打爬，从而在克服障碍时获得经验。

2.14 小结

建立一个现代的站点非常具有挑战性，所以站点开发者应该采用一种方法学或进程模型来指导开发过程，从而有希望减小风险和管理的复杂性，并且改进最终结果。诸如修正瀑布模型的软件工程进程模型在大多数 Web项目中很容易应用。然而，有时候，由于缺乏项目管理经验或清晰的目标，应该采用一种原型驱动方法或联合应用方法。通常，基于原型的方法更适合站点的自然特性，减少不必要的风险，在开发出合适的站点之前反复进化。在站点开发早期阶段进行规划，可以减小风险并提高最后的开发质量。应该撰写一个包括站点目标、访问者、任务分析、内容需求、站点结构、技术需求和管理考虑的内容设计文档。设计文档应该指导站点的实现。在站点实现时，应该利用块结构图、纸张模拟图、情节说明图板、甚至模拟站点来 减小后来重新设计站点的可能性。在规划详细且原型实现好后，实现就应该很快速，而需要重复的工作量很小。然而，一旦完成，不要急于实现在线访问 —— 充分的测试是必要的。长时间的维护是必要的，持续的警惕也是必要的，否则你的站点会开始恶化。