[**迭代开发和增量开发**](http://blog.csdn.net/l12345678/article/details/5642851)

“迭代”和“增量”是敏捷软件开发中的两个重要概念。弄清楚“迭代”和“增量”以及其依据，我们就可以在实际的操作中有章法可循。

**为什么要迭代？**

我们为什么要进行迭代开发呢？您一定遇到过这样情况：   
“我们知道想要什么。但你能估算出构建它需要多长时间吗？”   
“在启动开发之前，我们必须将这些需求明确下来。”   
“客户不知道他们想要什么”   
“客户时常改变想法”

“我 虽然不知道客户想要什么，但我却知道怎么得到它。”怎么得到客户想要的东西呢？——迭代！我们不指望我们所构建的软件正是客户（或用户）所想要的，但我们 可以先构建后修改，通过多次迭代找到真正适合客户（用户）的软件。当然，我们必须保证我们初次确定的方案是正确的、行得通的，那么后继的迭代就是反复求精 的过程，是不断地对备选方案改进并选择更优方案的过程，是以更优方案取代之前勉强行得通的方案的过程。



**迭代的依据**

迭代开发的过程就是对软件功能不断细化的过程，所以，迭代的依据就是“功能细化原则”：必要性–>灵活性–>安全性–>舒适性/趣味性。

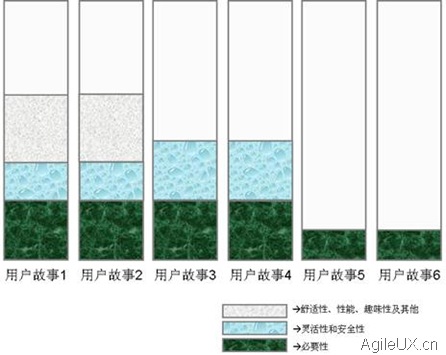
必要性：能支持用户完成任务的最少功能特性；

灵活性：支持用户使用多种方式完成任务或者支持用户做出额外选择的功能特性；

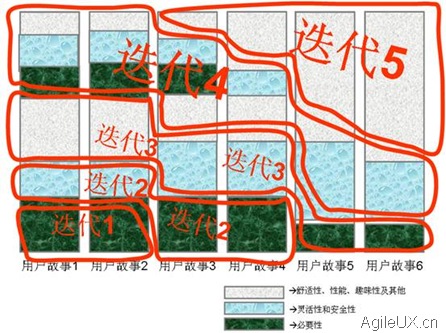
安全性：为了避免用户犯错，确保用户在软件使用过程中所做操作安全的功能特性；

舒适性/趣味性：就是可以使用户更简单、更快捷、更有趣地完成任务的功能特性。

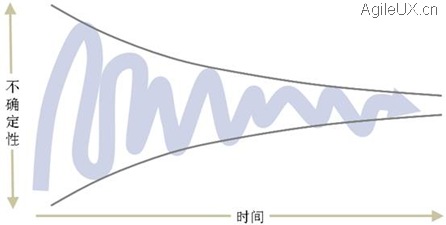
每个迭代可能包含多个用户故事，但是在同一个迭代中，我们对每个用户故事开发的完善程度是不一样的。如下图所示：

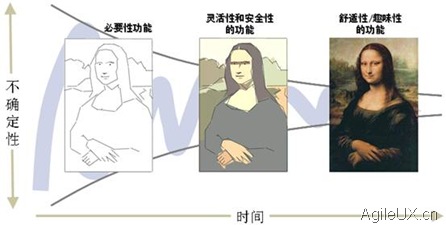


随 着软件开发工作的深入进行，我们会在每个用户故事中发现更多的功能可能是舒适性/趣味性方面的功能，也可能是必要的功能。或者，在软件开发的过程中，竞争 对手的软件产品有了新的功能、市场销售情况有了新的反馈、用户有新的需求等，我们需要不断地丰富、细化我们的软件所支持的用户故事，增加/改善新的功能。 经过多次迭代，我们就可以完成所有的功能，从多个层次（必要性、灵活性、安全性、舒适性/趣味性）满足用户需求、支持用户故事。



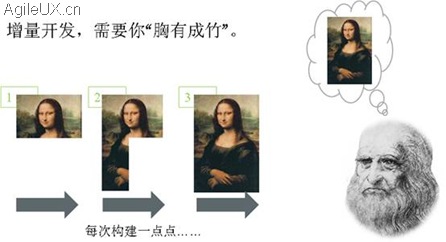
在迭代过程中，功能的不确定性逐渐减小，我们对功能的描述越来越明确。





**为什么要增量？**

不论是哪种类型的软件，其业务目标或用户（用户目标）一旦确定下来，我们都会为此准备好“理想”的解决方案和实现手段。但是，项目工期是有限的，资金预算 也是有数的，人手也不可能无限增加。为什么项目工期总是很短？资金紧张？人手不够？因为我们“理想”的解决方案是需要很大的代价的！并且，“理想”的解决 方案也有很大的风险：在这么漫长的“完美”解决方案实现过程中，市场情况、用户需要等外部因素都会发生改变；不及时发布、不从市场/用户处得到及时的反 馈，我们“理想”的解放方案是否真的完美也就无法得到验证。如果说迭代开发是为了应对软件产品内部的不确定因素的话，那么，增量地发布软件产品，为的就是 应对软件产品之外的其他不确定因素。



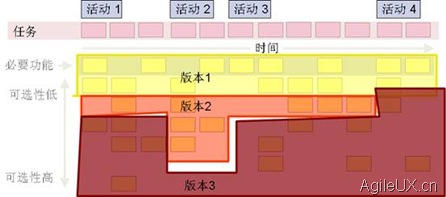
**增量的依据**

既然增量地发布软件产品是为了应对软件产品之外的不确定因素，那么，我们确定增量发布版本的过程，也就是项目风险控制的过程。在确定版本计划的时候，我们 采用什么样的尺度呢？考虑的太粗，我们的版本规划就不会太准确，在项目实施的过程中，就会存在较大的风险；“那我们就多考虑点”，要想考虑的很周到，我们 必定会在规划上花太多的时间。这其中的“度”在哪里呢？我们首先并且只会想到的就是对功能的优先级进行排序，然后看情况，到项目工期截止的时候，我们的功 能完成多少我们就交付多少。Yes，我们大多数的软件项目就是这么死的，都是在产品发布日的时候给它开追悼会！

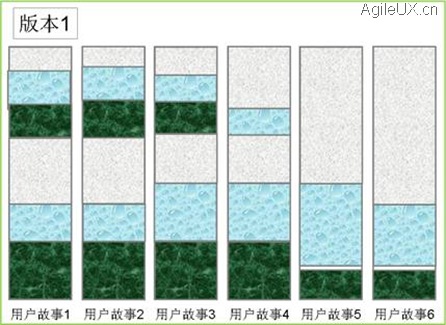
按 重要程度办事，有错吗？让我们打个比方吧，我们要造一辆轿车。先对轿车的功能物件排序：发动机、车底盘、传动轴、车轮子、刹车、方向盘……。车迷朋友们还 会列出更长、更详细的按优先级排序的轿车功能清单。有半年的工期，我们头一个月造了个发动机，不错很好很强大的发动机，第二个月，我们不但按计划把车底盘 搞定了，还有1周的时间，我们可以提前把传动轴也弄弄……（黑底白字的电视屏幕上淡入又淡出几个字“4个月后……”）还有1个月就要交付我们的轿车了，我 们的车轮子怎么上不好啊？方向盘也不转，对了，还有挡风玻璃也没弄，车门还没把手，转向灯能亮，但是它们前后左右4个一起亮……马上，半年的交付工期到 了。我们“预想”的完美轿车出厂了：绝美的流线型、强劲的引擎动力、4轮驱动、XYZ安全系统，可是、可是、、、昨天说安装的挡风玻璃怎么没安啊，好办： 在领导验收之前还来得及蒙一块塑料布！想必，这么“拉风”的轿车，定会被老板拍死、被客户拍死。

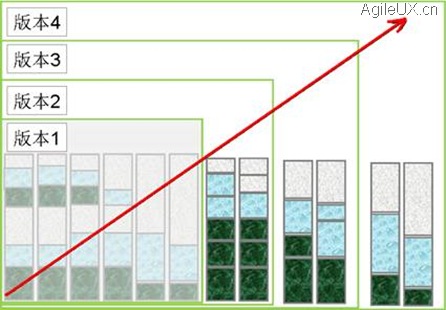
现 在，有点明白了吧，来感觉了？对，我们可以按照重要程度来做事情，但是，在全部必要的功能全部实现之前，你我所实现的再重要的功能都无人买单，无法体现其 价值。这也是在做软件需求工作时，普遍存在的问题：功能考虑的很多、优先级排的挺有水平、对每个功能的描述也很详尽，但是，各个功能各自为战、不成体系， 甚至还缺少许多必要的功能，所以，我们软件产品的用户就因此疯掉了，由此也引发了众多无聊的忧国忧民的砖家们来“反思”这样的话题：“科技创新是否真正改 善了人类的生活”。（善哉，善哉，我又不小心提到“砖家”这么不吉利的人物了）

规划 版本时，在第一个版本中，我们必须实现所有的“必要性”功能，否则，我们的软件是无法体现出价值的。在之后的每个版本中，我们都要参考“功能细化原则”， 使得我们的软件产品的所有功能都达到相同的用户体验水平。（关于用户体验的“境界”，我们会在UCD思想中作详细的介绍）如果每个版本中的各功能的用户体 验“境界”不一致，很容易出现“用塑料布当挡风玻璃的豪华奔驰轿车”。我们可以用如下图所示的“版本地图”的形式来展示软件产品增量发布的依据——版本计 划。



当然，我们也会在完成前一个软件版本后，发现新的市场/用户需求，新增用户故事。增量版本发布的过程，如下图所示：

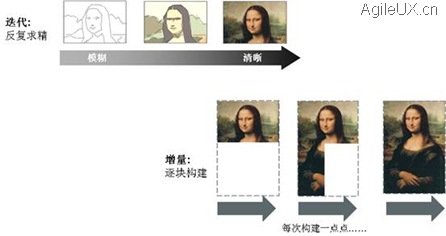




**小结：迭代vs.增量**

要想比较彻底地理解“迭代”和“增量”，我们最好将其对比一下。

迭代，就是在实现软件的每一功能时反复求精的过程，是提升软件质量的过程，是从模糊到清晰的过程；而增量，则是强调软件在发布不同的版本时，每次都多发布一点点，是软件功能数量渐增地发布的过程。二者的对比如下图所示：



# 2.2.6 阶段化开发：增量和迭代

《软件工程：第4版》理论阐述循序渐进，善于揭示各知识点之间的内在联系，并通过大量实例和工程实践深化和丰富理论知识；选材与时俱进，反映了最新的业界动态，尤其是建模和敏捷方法方面的重要进展。第2章为过程和生命周期的建模概述。本节说的是阶段化开发：增量和迭代。

AD：

**2.2.6  阶段化开发：增量和迭代**

在早期的软件开发中，客户愿意为软件系统的最后完成等待很长时间。有时，从编写需求文档到系统交付使用会经过若干年，称为循环周期（cycle time）。但是，今天的商业环境不会再容许长时间的拖延。软件使产品在市场上引人注目，而客户总是期待着更好的质量和新的功能。例如，1996年，惠普 公司80%的收入来自过去两年开发的产品，因而，他们开发了新的过程模型来帮助缩短循环周期。

一种缩短循环周期的方法是使用阶段化开发，如图2-8所示。使用这种方法设计系统时使其能够一部分一部分地交付，从而在系统其余部分正在开发的同 时，用户已经获得了一部分功能。因此，通常会有两个系统在并行运行：产品系统和开发系统。运行系统（operational system）或产品系统（production system）是当前正在被客户和用户使用的系统，而开发系统（development system）是准备用来替换现行产品系统的下一个版本。通常，用它们的发布代号表示一个系统：开发人员构建发布1，进行测试，然后把它交给用户作为第一 个可运行的发布。然后，当用户使用发布1的时候，开发人员正在构建发布2。从而，开发人员总是在开发发布n+1，而与此同时发布n总是正在运行的。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20100429/140812976.jpg> |
|  |

开发人员可以用多种方法决定如何将开发组织为发布。增量开发（incremental development）和迭代开发（iterative development）是两种最常用的方法。在增量开发中，需求文档中指定的系统按功能划分为子系统。定义发布时首先定义一个小的功能子系统，然后在每 一个新的发布中增加新功能。图2-9的上半部分显示了增量开发是如何在每一个新的发布中逐步增加功能直到构造全部功能的。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20100429/140837932.jpg> |
|  |

而迭代开发是在一开始就提交一个完整的系统，然后在每一个新的发布中改变每个子系统的功能。图2-9的下半部分说明一个迭代开发的3个发布。

为了理解增量开发和迭代开发之间的区别，我们来看一个用于文字处理的软件包。假设这个软件包要具有3种类型的功能：创建文本、组织文本（即剪切和粘 贴）以及格式化文本（例如使用不同的字体大小和类型等）。要使用增量开发模型构建这样一个系统，我们可能在发布1中仅提供创建功能，然后在发布2中提供创 建和组织功能，最后在发布3中提供创建、组织和格式化功能。但是，使用迭代开发方法时，我们要在发布1中提供简单的3种类型的功能。例如：可以创建文本， 然后剪切并粘贴文本，但是剪切和粘贴功能可能不够灵活快捷。在下一次迭代（即发布2）中，提供相同的功能，但是系统的功能增强了：剪切和粘贴功能变得方便 和快捷。每一个发布都在前一个发布的基础上进行了某些改进。

实际上，许多组织都将迭代开发和增量开发方法结合起来使用。一个新的发布版本可能包含新的功能，并且对已有功能做了改进。这种形式的阶段化开发方法是人们想要的，原因如下。

(1) 即使还缺少某些功能，但在早期的发布中就可开始进行培训。培训过程可以使开发人员观察某些功能是如何执行的，并为后面的发布提供了改进的建议。这样，开发人员能够很好地对用户的反馈做出反应。

(2) 可以及早为那些以前从未提供的功能开拓市场。

(3) 当运行系统出现未预料到的问题时，经常性的发布可以使开发人员能全面、快速地修复这些问题。

(4) 针对不同的发布版本，开发团队将重点放在不同的专业领域技术上。例如，一个发布可以利用用户界面专家的专业知识将系统从命令驱动的界面改为指向-点击式（point-and-click）的图形用户界面，另外一个发布可集中于改进系统性能。

*开发/测试是基于PDT核心组研发代表和其他成员协商确定的渐增'Builds'。'Builds' 可以通过检查‘分解和分配’的结果和基于功能选择的子集来确定。'Builds'也可以基于特定的子系统。 'Builds' 是渐增的，也就是说，每一个'Build'是以前一个'Build'为基础。因此第一个'Build'可能只包括一些关键功能，然后一些渐增的'Builds'逐渐累积功能，最后一个'Builds'将执行所有的功能。每个'Build'由'Build名称'和'Build号'来识别。这使得团队能够清晰的识别和定义每个'Build'、进行优先级排序、安排软硬件开发进度，以便各个'Build'需要的部件和编程设计能够及时的组合到一起。这种方法最主要的好处就是，团队不必等待要开发的所有部分都完成，他们可以组装、测试、调测其中的子集，然后在每个渐增的'Build'中逐渐加上越来越多的部分。*

*'Builds'受计划的软硬件单元的可获得性的约束。下表能够明确'Builds'需要的软硬件单元的可获得性。这些是基于‘计划’的，因此，它们会在开发过程中随开发计划的变化而变化；PDT要确保'Builds'和它们的测试能满足变化了的计划要求。*