RUP 的十大要素

作者：**Leslee Probasco(开发经理)**
译者：沈斌（domybest313@hotmail.com）

为了有效的应用 Rational Unified Process(我们通常亲切的把它称为 RUP),首先要理解它的关键目标,并且弄清楚每一个目标为什么重要,他们是怎么样结合在一起，共同帮助你的开发团队满足涉众需求，生产出优质产品的。

首要的是抓住要点

有天晚上，我的邻居 Randy 过来求助。他正在为周末野营和徒步旅行作准备，但是不知道带些什么东西才好。他知道，我经常领导和参加野外旅行，而且我能够很快的决定在有限的包裹里塞些什么东西，他还记得我曾经给他提过，我有一张我拥有的所有设备和衣服的清单。“那么，我可以借那张清单吗？”他问道。

“当然，但是恐怕帮助不大。”我解释道。你看，在我的外出设备清单中有好几百项，涉及很多种类型的外出，从背包攀登到滑雪,旅行时间从几天的短途旅行到很多天的远征探险。我知道，如果没有相应的指南，Randy 将会陷入冗长的清单之中，以致弄不清，就他相对简单的外出而言，什么才是他真正需要的。

始于要素，逐步递增

因此，我提出看一下 Randy 在他的鼓鼓囊囊的包里面都已经装了那些东西。我们可以看以看，他是否可以少带些什么以减轻负担，或者是还有什么该带的却没有带。过了一会儿，我已经能肯定，他真正缺少的不是别的，而是对野外旅行的理解，也就是说，抓不住野外旅行的要点。

我拿出一张空白的纸，列出以下十个项目：[1]

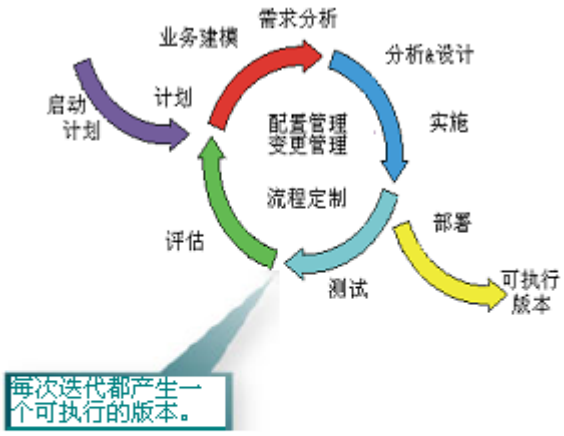
1. 地图(Map)
2. 指南针(Compass)
3. 太阳镜和防晒油(Sunglasses & sunscreen)
4. 额外的衣服(Extra clothing)
5. 额外的食物和水(Extra food & water)
6. 头上戴的小灯(Headlamp)
7. 急救箱(First-aid kit)
8. 打火机(Fire-starter)
9. 火柴(Matches)
10. 刀子(Knife)

“你看，Randy。这就是你真正需要的。如果你从这十大要素出发，那么，无论遇到什么旅行，再来考虑还需要增加哪些内容就变得容易多了。”多年前，我第一次登山时，靠的就是这张清单，现在我仍然使用它，无论我准备的旅行时那种类型、要去多长时间。每一项的膨胀或者压缩取决于旅行本身。始于简短的清单，然后需要时再扩展，这是一种方式;始于冗长的清单，然后再来决定不采用什么，这是另一种方式。但是两种方式相比，前者显然要容易得多。

把这一课应用到ＲＵＰ中

当我帮助项目组就ＲＵＰ的很多元素进行排序时，常常听到这样的问题：“我怎样对所有这些内容进行排序？而且决定在我的项目里究竟需要哪些要素？”“ＲＵＰ包括这么多的信息。它一定是针对大项目的－－我真的能在我的小项目使用它吗？”

我断定，这些人真正需要的是“RUP 的十大要素”，就像我给我的朋友 Randy 的简单的清单一样。这个 RUP 的清单，可以作为任何项目的符合情理的起点，无论小项目、中型项目还是大型项目。这个



对那些不熟悉RUP的人，这张图表现了RUP的增量和迭代的特点。RUP的内容覆盖整个软件开发生命周期，包括工件、指南、团队成员角色以及活动。

迭代式开发循环模型

列表会聚焦在被我称之为“精华或要素”的东西上，可能是 RUP 的，也可能是任何有效软件过程的。

在所有成员领悟到提交合格产品所需要的关键过程元素之前，项目往往陷入某个特定主题的细节的沼泽中。然后，当项目拖后时，大家就会怪罪以前被过分强调的某些活动，或者是怪罪大家不理解其用处的某些活动，“嘿，我早就告诉你需求管理（或者是 Use Cases、收集项目度量数据、使用配置管理、使用缺陷跟踪工具、召开项目状态会议里面的一个或几个）会放慢我们的进度！你不信！”

有一个“精华或要素”列表让团队成员采用一种更系统、更全面的方式来思考和执行整个软件开发过程。一旦一个过程框架或“构架”到位了，团队成员就能更有效的面对和处理单个的问题域（大部分时间我得承认，需求管理应该在列表的顶部）。同样，一开始就标识显然的问题以及相关的风险，并且确定处理他们的优先级，也是很重要的，这样，团队才能在早期就根据需要采取相应的解决或缓解对策。

ＲＵＰ的十大要素

那么，在 RUP 的十大要素中应该包括哪些内容呢？下面是我的意见：

1. 开发前景
2. 达成计划
3. 标识和减小风险
4. 分配和跟踪任务。。
5. 检查商业理由
6. 设计组件构架
7. 对产品进行增量式的构建和测试
8. 验证和评价结果
9. 管理和控制变化
10. 提供用户支持

让我们逐一的审视这些要素，看一看它们什么地方适合ＲＵＰ，找出它们能够成为十大要素的理由。

1. 开发一个前景

有一个清晰的前景是开发一个满足涉众真正需求的产品的关键。

前景抓住了ＲＵＰ需求流程的要点：分析问题，理解涉众需求，定义系统，当需求变化时管理需求。

前景给更详细的技术需求提供了一个高层的、有时候是合同式的基础。正像这个术语隐含的那样，它是软件项目的一个清晰的、通常是高层的视图，能被过程中任何决策者或者实施者借用。它捕获了非常高层的需求和设计约束，让前景的读者能理解将要开发的系统。它还提供了项目审批流程的输入，因此就与商业理由密切相关。最后，由于前景构成了“项目是什么？”和“为什么要进行这个项目？”，所以可以把前景作为验证将来决策的方式之一。

对前景的陈述应该能回答以下问题，需要的话这些问题还可以分成更小、更详细的问题：

- 关键术语是什么？（词汇表）
- 我们尝试解决的问题是什么？（问题陈述）
- 涉众是谁？用户是谁？他们各自的需求是什么？
- 产品的特性是什么？
- 功能性需求是什么？（Ｕｓｅ　Ｃａｓｅｓ）
- 非功能性需求是什么？
- 设计约束是什么？

2. 达成计划

“产品的质量只会和产品的计划一样好。”[2]

在ＲＵＰ中，软件开发计划（ＳＤＰ）综合了管理项目所需的各种信息，也许会包括一些在先启阶段开发的单独的内容。SDP 必须在整个项目中被维护和更新。

ＳＤＰ定义了项目时间表（包括项目计划和迭代计划）和资源需求（资源和工具），可以根据项目进度表来跟踪项目进展。同时也指导了其他过程内容（原文：process components）的计划：项目组织、需求管理计划、配置管理计划、问题解决计划、QA 计划、测试计划、评估计划以及产品验收计划。

在较简单的项目中，对这些计划的陈述可能只有一两句话。比如，配置管理计划可以简单的这样陈述：每天结束时，项目目录的内容将会被压缩成 ZIP 包，拷贝到一个 ZIP 磁盘中，加上日期和版本标签，放到中央档案柜中。

软件开发计划的格式远远没有计划活动本身以及驱动这些活动的思想重要。正如Dwight D.Eisenhower所说：“plan什么也不是，planning才是一切。”

“达成计划”—和列表中第 3、4、5、8 条一起—抓住了 RUP 中项目管理流程的要点。项目管理流程包括以下活动：构思项目、评估项目规模和风险、监测与控制项目、计划和评估每个迭代和阶段。

3. 标识和减小风险

RUP 的要点之一是在项目早期就标识并处理最大的风险。项目组标识的每一个风险都应该有一个相应的缓解或解决计划。风险列表应该既作为项目活动的计划工具，又作为确定迭代的基础。

4. 分配和跟踪任务。。

有一点在任何项目中都是重要的，即连续的分析来源于正在进行的活动和进化的产品的客观数据。在RUP中，定期的项目状态评估提供了讲述、交流和解决管理问题、技术问题以及项目风险的机制。团队一旦发现了这些障碍物（篱笆），他们就把所有这些问题都指定一个负责人，并指定解决日期。进度应该定期跟踪，如有必要，更新应该被发布。（原文：updates should be issued as necessary。）

这些项目“快照”突出了需要引起管理注意的问题。随着时间的变化/虽然周期可能会变化（原文：While the period may vary。），定期的评估使经理能捕获项目的历史，并且消除任何限制进度的障碍或瓶颈。

5. 检查商业理由

商业理由从商业的角度提供了必要的信息，以决定一个项目是否值得投资。商业理由还可以帮助开发一个实现项目前景所需的经济计划。它提供了进行项目的理由，并建立经济约束。当项目继续时，分析人员用商业理由来正确的估算投资回报率(ROI，即return on investment)。

商业理由应该给项目创建一个简短但是引人注目的理由，而不是深入研究问题的细节，以使所有项目成员容易理解和记住它。在关键里程碑处，经理应该回顾商业理由，计算实际的花费、预计的回报，决定项目是否继续进行。

6. 设计组件构架

在 RUP 中，软件系统的构架是指一个系统关键部件的组织或结构，部件之间通过接口交互，而部件是由一些更小的部件和接口组成的。即主要的部分是什么？他们又是怎样结合在一起的？

RUP 提供了一种设计、开发、验证构架的很系统的方法。在分析和设计流程中包括以下步骤：定义候选构架、精化构架、分析行为（用例分析）、设计组件。

要陈述和讨论软件构架，你必须先创建一个构架表示方式，以便描述构架的重要方面。在 RUP 中，构架表示由软件构架文档捕获，它给构架提供了多个视图。每个视图都描述了某一组涉众所关心的正在进行的系统的某个方面。涉众有最终用户、设计人员、经理、系统工程师、系统管理员，等等。这个文档使系统构架师和其他项目组成员能就与构架相关的重大决策进行有效的交流。

7. 对产品进行增量式的构建和测试

在 RUP 中实现和测试流程的要点是在整个项目生命周期中增量的编码、构建、测试系统组件，在先启之后每个迭代结束时生成可执行版本。在精化阶段后期，已经有了一个可用于评估的构架原型；如有必

要，它可以包括一个用户界面原型。然后，在构建阶段的每次迭代中，组件不断的被集成到可执行、经过测试的版本中，不断地向最终产品进化。动态及时的配置管理和复审活动也是这个基本过程元素（原文：essential process element）的关键。

8. 验证和评价结果

顾名思义，RUP 的迭代评估捕获了迭代的结果。评估决定了迭代满足评价标准的程度，还包括学到的教训和实施的过程改进。

根据项目的规模和风险以及迭代的特点，评估可以是对演示及其结果的一条简单的纪录，也可能是一个完整的、正式的测试复审记录。

这儿的关键是既关注过程问题又关注产品问题。越早发现问题，就越没有问题。（原文：The sooner you fall behind, the more time you will have to catch up.）

9. 管理和控制变化

RUP 的配置和变更管理流程的要点是当变化发生时管理和控制项目的规模，并且贯穿整个生命周期。其目的是考虑所有的涉众需求，*尽可能*的满足，同时仍能*及时的*交付合格的产品。

用户拿到产品的第一个原型后（往往在这之前就会要求变更），他们会要求变更。重要的是，变更的提出和管理过程始终保持一致。

在 RUP 中，变更请求通常用于记录和跟踪缺陷和增强功能的要求，或者对产品提出的任何其他类型的变更请求。变更请求提供了相应的手段来评估一个变更的潜在影响，同时记录就这些变更所作出的决策。他们也帮助确保所有的项目组成员都能理解变更的潜在影响。

10. 提供用户支持

在 RUP 中，部署流程的要点是包装和交付产品，同时交付有助于最终用户学习、使用和维护产品的任何必要的材料。

项目组至少要给用户提供一个用户指南（也许是通过联机帮助的方式提供），可能还有一个安装指南和版本发布说明。

根据产品的复杂度，用户也许还需要相应的培训材料。最后，通过一个材料清单（BOM 表，即 Bill of Materials）清楚地记录应该和产品一起交付哪些材料。

关于需求

有人看了我的要素清单后，可能会非常不同意我的选择。例如，他会问，需求在哪儿呢？他们不重要吗？我会告诉他我为什么没有把它们包括进来。有时，我会问一个项目组（特别是内部项目的项目组）：“你们的需求是什么？”，而得到的回答却是：“我们的确没有什么需求。”

刚开始我对此非常惊讶（我有军方的宇航开发背景）。他们怎么会没有需求呢？当我进一步询问时，我发现，对他们来说，需求意味着一套外部提出的强制性的陈述，要求他们必须怎么样，否则项目验收就不能通过。但是他们的确没有得到这样的陈述。尤其是当项目组陷入了边研究边开发的境地时，产品需求从头到尾都在演化。

因此，我接着问他们另外一个问题：“好的，那么你们的产品的前景是什么呢？”。这时他们的眼睛亮了起来。然后，我们非常顺利的就第一个要素（“开发一个前景”）中列出的问题进行了沟通，需求也自然而然的流动着（原文：and the requirements just flow naturally.）。

也许只有对于按照有明确需求的合同工作的项目组，在要素列表中加入“满足需求”才是有用的。请记住，我的清单仅仅意味着进行进一步讨论的一个起点。

总结：十大要素的应用

那么，发现了 RUP 的十大要素之后，怎样才能让它给我的职业生涯带来根本的变化呢？这儿有一些建议，能帮助我们对付各种规模的项目。

对于非常小的项目

首先，如果谁来问我，在一个非常小的、没有经验的项目组（才学了 RUP）中，如何使用 RUP 和 Rational 开发工具来构造一个简单的产品，我会与他分享十大要素列表，以使项目组不被 RUP 的细节和 Rational Suites 的功能压垮。
实际上，即使没有任何自动化工具也可以实施十大要素。管理一个小项目，一个项目笔记本，就已经是一个非常好的起点，可以把它分成十个部分，每一部分专用于十大要素中的一个要素。（我还发现及时贴[原文：Post-It Notes]对于小项目变更请求的管理和跟踪以及确定变更的优先级非常有用。）

对于增长的项目

当然，当一个项目的规模和复杂度增长时，以上这些应用十大要素的简单方法很快就变得不可操作，而对自动化工具的需求就变得比较明显了。然而，我还是愿意鼓励项目的领导者刚开始时应用十大要素和 RUP 的“最佳实践”，需要时再逐步增加支持工具，而不是一下子就尝试使用全套 Rational Suites。

对于成熟的项目团队

对成熟的项目团队而言，可能已经在采用某种软件过程和使用 CASE 工具，十大要素可以提供一种快速评估方法，用来评估关键过程元素的平衡性，标识他们并确定改进的优先级。

对于所有的项目

当然，各个项目都不太一样，有些项目似乎并不真正需要所有的要素。在这些情况下，重要的是考虑：如果你的团队忽视某个要素后会发生什么问题。举例如下：

- 没有前景？你会迷失方向，走很多弯路，把力气浪费在毫无结果的努力上。
- 没有计划？你将无法跟踪进度。
- 没有风险列表？你的项目会陷入“专注于错误的问题上”的危险里面，可能一下子被一个没有检测的地雷击倒，并为此付出五个月的代价。
- 没有问题列表？没有定期的问题分析和解决，小问题会演变成大问题。
- 没有商业理由？你在冒浪费时间和金钱的风险。项目最终要么超支，要么被取消。
- 没有构架？在出现交流、同步和数据存取问题时，你可能无法处理。（原文：You may be unable to handle communication,synchronization, and data access issues as they arise.）你也可能在伸缩性和性能上有问题。
- 没有产品（原型）？你将不能有效的测试（原文：You won't be able to test effectively），并且会失去客户的信任。
- 没有评估？你将没有办法掌握实际情况与项目目标、预算和最后期限之间的距离。
- 没有变更请求？你将无法估计变更的潜在影响，无法就互相冲突的需求确定优先级，无法在实施变更时通知整个项目组（原文：and update the whole team when changes are implemented.）。
- 没有用户支持？用户将不能最有效的使用产品，技术支持人员也会淹没在大量支持请求中。

现在你知道了，不懂得十大要素是一件非常冒险的事情。我鼓励你把它们作为项目组的一个起点。决定哪些是你们想要的，哪些是不要的，哪些是要修改的。然后，再决定还有哪些其他因素是你们项目（无论项目大小）成功(保证项目组及时的、不超预算的交付产品，并且真正满足涉众的真正需求)的关键因素。

其他要素

其他组织也出版了类似的软件工程要素列表。IEEE 软件杂志 1997 年 3/4 月号上有一篇 Steve McConnell 写的文章，“软件的十大要素”。软件项目经理网络也有一个“16 个关键软件实践”的列表，在www.spmn.com上可以找到。SEI-CMM 的 KPAs（Key Process Areas,关键过程域）也可以作为一种要素列表（见 www.sei.cum.edu）。

(1)参见“The Freedom of the Hills”,第 6 版，Mountaineers of Seattle 出版，1997 年。
(2)参见“They Write the Right Stuff”,Charles Fishman,Fast Company, 第 6 期，第 95 页，1996 年 12 月。