项目管理规范-RUP 管理实施

第一部分：项目阶段

第二部分：核心工作流程

第三部分：角色划分

第四部分：目前实施项目规范的考虑

概述

软件开发的产品质量水平，是一个由来已久的话题。而提高软件企业的产品质量水平，必须改进软件产品的开发过程。但是这里没有什么百试百灵的灵丹妙药，我们必须根据本企业的实际情况，参考国内外先进企业的经验，总结出一种适合本企业的软件开发模式。

此规范是基于 CMM 模型规范，以 RUP 软件工程过程为蓝本，由我本人根据项目实际情况而选择修改，从而使之适应当前应用级系统设计开发的需要。

本文主要以 RUP 的软件工程框架为主，省略复杂概念部分。着眼点放在控制软件产品开发流程上，由于人员配置与软件分工现行状况的限制，对其中的部分细节进行了合并可省略从而适应目前国内软件开发所要求。

Rational Unified Process（简称 RUP）是一套软件工程过程（在下面介绍）。

在 RUP 过程中，我们可以看到它非常强调一点：循环。

现在我们做的每一个项目都存在不断变化的问题。用户需求变化、系统设计变化（可能是需求变化也可能是存在了技术问题）、编码变化（由测试与复审等环节引发的）等问题困扰着项目进行。解决这些问题的方法就是不断的循环。

这个规范是我根据自己的观点整理编写而成的，有不足之处请指教。

RUP 简介

Rational Unified Process（简称 RUP）是一套软件工程过程，主要由 Ivar Jacobson 的 The Objectory Approch 和 The Rational Approch 发展而来。同时，它又是文档化的软件工程产品，所有 RUP 的实施细节及方法导引均以 Web 文档的方式集成在一张光盘上，由 Rational 公司开发、维护并销售，当前版本是 RUP2000。RUP 又是一套软件工程方法的框架，各个组织可根据自身的实际情况，以及项目规模对 RUP 进行裁剪和修改，以制定出合乎需要的软件工程过程。

RUP 吸收了多种开发模型的优点，具有很好的可操作性和实用性、从它一推出市场，凭借 Booch、Ivar Jacobson、以及 Rumbaugh 在业界的领导地位、以及与统一建模语言（Unified Model Language，以下简称 UML）的良好集成、多种 CASE 工具的支持、不断的升级与维护，迅速得到业界广泛的认同，越来越多的组织以它作为软件开发模型框架

在 RUP 中，软件开发生命周期根据时间和 RUP 的核心工作流划分为二维空间。

如上图所示，时间维从组织管理的角度描述整个软件开发生命周期，是 RUP 的动态组成部分。它可进一步描述为周期（Cycle）、阶段（phase）、迭代(Iteration)。

核心工作流从技术角度描述 RUP 的静态组成部分，它可进一步描述为行为（activities）、工作流（workflow）、产品（artifact）、工人（worker）。

图中的阴影部分描述了不同的工作流，在不同的时间段内工作量的不同。值得注意的是，几乎所有的工作流，在所有的时间段内均有工作量，只是大小不同而已。这与 Waterfall process 有明显的不同。

RUP 采用 Use Case 的概念，把要开发的系统根据各功能使用的情况划分多个 Use Case，并采用迭代的思想把系统的风险分布在四个阶段，风险越大的迭代越要放在靠前的阶段做，使软件产品的风险不断降低；而不是像传统软件工程那样越往开发的后期问题越多。所以 RUP 的思想一推出就受到软件企业的欢迎。按照 RUP 的开发模式一般可以达到 CMM2、3 级的水平。当然，理解和掌握 RUP 需要一个相对较长的过程。

1. 项目阶段

从管理的观点来说，软件生命周期随着时间分为四个依次进行的阶段，每个阶段的结束都有一个主要里程碑；实质上，每个阶段就是两个主要里程碑之间的时间跨度。在每个阶段结束时进行评估，以确定是否实现了此阶段的目标。良好的评估可使项目顺利进入下一阶段。

1.1. 计划阶段

在进度和工作量方面，所有阶段都各不相同。尽管不同的项目有很大的不同，但一个中等规模项目的典型初始开发周期应该预先考虑到工作量和进度间的分配：

先启 精化 构建 产品化

工作量 ~5% 20% 65% 10%

进度 10% 30% 50% 10%

可表示为下图

对于演进周期，先启和精化阶段就小得多了。能够自动完成某些构建工作的工具将会缓解此现象，并使得构建阶段比先启阶段和精化阶段的总和还要小很多。

通过这四个阶段就是一个开发周期；每次经过这四个阶段就会产生一代软件。除非项目“死亡”，否则通过重复同样的先启阶段、精化阶段、构建阶段和产品化阶段的顺序，产品将演进为下一代产品，但每一次的侧重点都将放在不同的阶段上。这些随后的周期称为演进周期。 随着产品经历了几个周期，新一代产品随之产生。

1.2. 先启阶段

1.2.1. 目标

先启阶段的基本目标是实现项目的生命周期目标中所有相关因素（如客户等）之间的并行 先启阶段主要对新的开发工作具有重大意义，新工作中的重要业务风险和需求风险问题必

须在项目继续进行之前得到解决。对于重点是扩展现有系统的项目来说，先启阶段较短，但重点仍然是确保项目值得进行而且可以进行。

先启阶段的主要目标包括：

• 建立项目的软件规模和边界条件，包括运作前景、验收标准以及希望软件中包括和不包括的内容。

• 识别系统的关键用例（也就是将造成重要设计折衷操作的主要部分）。

• 评估整个项目的总体成本和进度（以及对即将进行的精化阶段进行更详细的评估）

• 评估潜在风险（不可预测性的来源）

• 准备项目的支持环境。

1.2.2. 核心活动

• 明确地说明项目规模。这涉及了解环境以及最重要的需求和约束，以便于可以得出最终产品的验收标准。

• 计划和准备商业理由。评估风险管理、人员配备、项目计划和成本/进度/收益率折衷的备选方案。

• 综合考虑备选构架，评估设计和自制/外购/复用方面的折衷，从而估算出成本、进度和资源。此处的目标在于通过对一些概念的证实来证明可行性。该证明可采用可模拟需求的模型形式或用于探索被认为高风险区域的初始原型。先启阶段的原型设计工作应该限制在确信解决方案可行就可以了。该解决方案在精化和构建阶段实现。

• 准备项目的环境，评估项目和组织，选择工具，决定流程中要改进的部分。

1.2.3. 里程碑：生命周期目标

生命周期目标里程碑评估项目的基本可行性。

先启阶段末是第一个重要的项目里程碑，即生命周期目标里程碑。此时，检查项目的生命周期目标，并决定继续进行项目还是取消项目。

1.2.3.1 评估标准

• 规模定义和成本／进度估算中，所有相关因素（如客户等）可并行

• 对是否已经获得正确的需求集达成一致意见，并且对这些需求的理解是共同的。

• 对成本／进度估算、优先级、风险和开发流程是否合适达成一致意见。

• 已经确定所有风险并且有针对每个风险的减轻风险策略。

如果项目无法达到该里程碑，则它可能中途失败或需要进行相当多的重新考虑。

1.2.3.2 提供的文档及模型

核心文档及模型（按照重要性排序） 里程碑状态

前景 已经对核心项目的需求、关键功能和主要约束进行了记录。

商业理由 已经确定并得到了批准。

风险列表 已经确定了最初的项目风险。

软件开发计划 已经确定了最初阶段及其持续时间和目标。软件开发计划中的资源估算（特别是时间、人员和开发环境成本）必须与商业理由一致。

资源估算可以涵盖整个项目直到交付所需的资源，也可以只包括进行精化阶段所需的资源此时，整个项目所需的资源估算应该看作是大致的“粗略估计”。该估算在每个阶段和每次迭代中都会更新，并且随着每次迭代变得更加准确。 根据项目的需要，可能在某种条件下完成了一个或多个附带的“计划”工件。此外，附带的“指南”工件通常也至少完成了“草稿”。

迭代计划 第一个精化迭代的迭代计划已经完成并经过了复审。

软件验收计划 完成复审并确定了基线；随着其他需求的发现，将对其在随后的迭代中进行改进。

项目专用模板 已使用文档模板制作了文档工件。

用例建模指南 确定了基线。

工具 选择了支持项目的所有工具。安装了对先启阶段的工作必要的工具。

词汇表 已经定义了重要的术语；完成了词汇表的复审。

用例模型（主角，用例） 已经确定了重要的主角和用例，只为最关键的用例简要说明了事件流。

领域模型（也叫做业务对象模型） 已经对系统中使用的核心概念进行了记录和复审。在核心概念之间存在特定关系的情况下，已用作对词汇表的补充。

原型 概念原型的一个或多个证据，以支持前景和商业理由、解决非常具体的风险。

1.3. 精化阶段

1.3.1. 目标

精化阶段的目标是建立系统构架的基线，以便为构建阶段的主要设计和实施工作提供一个稳定的基础。构架是基于对大多数重要需求（对系统构架有很大影响的需求）的考虑和风险评估发展而来的。构架的稳定性是通过一个或多个构架原型进行评估的。 精化阶段的主要目标包括：

确保构架、需求和计划足够稳定，充分减少风险，从而能够有预见性地确定完成开发所需的成本和进度。对大多数项目来说，通过此里程碑也就相当于从简单快速的低风险运作转移到高成本、高风险的运作，并且在组织结构方面面临许多不利因素。

处理在构架方面具有重要意义的所有项目风险

建立一个已确定基线的构架，它是通过处理构架方面重要的场景得到的，这些场景通常可以显示项目的最大技术风险。

制作产品质量构件的演进式原型，也可能同时制作一个或多个可放弃的探索性原型，以减小特定风险，例如：

设计/需求折衷

构件复用

产品可行性或向客户和最终用户进行演示。

证明已建立基线的构架将在适当时间、以合理的成本支持系统需求。

建立支持环境。

为了实现这个主要目标，建立项目的支持环境也同等重要。这包括创建开发案例、创建模板和指南、安装工具。

1.3.2. 核心活动

• 快速确定构架、确认构架并为构架建立基线。

• 根据此阶段获得的新信息改进前景，对推动构架和计划决策的最关键用例建立可靠的了解。

• 为构建阶段创建详细的迭代计划并为其建立基线。

• 改进开发案例，定位开发环境，包括流程和支持构建团队所需的工具和自动化支持。

• 改进构架并选择构件。 评估潜在构件，充分了解自制/外购/复用决策，以便有把握地确定构建阶段的成本和进度。集成了所选构架构件，并按主要场景进行了评估。通过这些活动得到的经验有可能导致重新设计构架、考虑替代设计或重新考虑需求。

1.3.3. 里程碑：生命周期构架

生命周期构架里程碑为系统构架建立管理基线，并使项目团队能够在构建阶段调整规模。

精化阶段末是第二个重要的项目里程碑，即生命周期构架里程碑。此时，您检查详细的系统目标和规模、选择的构架以及主要风险的解决方案。

1.3.3.1 评估标准

• 产品前景和需求是稳定的。

• 构架是稳定的。

• 可执行原型表明已经找到了主要的风险元素，并且得到妥善解决。

• 构建阶段的迭代计划足够详细和真实，可以保证工作继续进行。

• 构建阶段的迭代计划由可靠的估算支持。

• 所有客户方人员一致认为，如果在当前构架环境中执行当前计划来开发完整的系统，则当前的前景可以实现。

• 实际的资源耗费与计划的耗费相比是可以接受的。

如果项目无法达到该里程碑，则它可能中途失败或需要进行相当多的重新考虑。

1.3.3.2 提供的文档及模型

核心文档及模型（按照重要性排序） 里程碑状态

原型 已经创建了一个或多个可执行构架原型，以探索关键功能和构架上的重要场景。

风险列表 已经进行了更新和复审。 新的风险可能是构架方面的，主要与处理非功能性需求有关。

项目专用模板 已使用文档模板制作了文档工件。

工具 已经安装了用于支持精化阶段工作的工具。

软件构架文档 编写完成并确定了基线，如果系统是分布式的或必须处理并行问题，则包括构架上重要用例的详细说明（用例视图）、关键机制和设计元素的标识（逻辑视图），以及（部署模型的）进程视图和部署视图的定义。

设计模型（和所有组成部分） 制作完成并确定了基线。已经定义了构架方面重要场景的用例实现，并将所需行为分配给了适当的设计元素。 已经确定了构件并充分理解了自制/外购/复用决策，以便有把握地确定构建阶段的成本和进度。集成了所选构架构件，并按主要场景进行了评估。通过这些活动得到的经验有可能导致重新设计构架、考虑替代设计或重新考虑需求。

数据模型 制作完成并确定了基线。已经确定并复审了主要的数据模型元素（例如重要实体、关系和表）。

实施模型（以及所有组成工件，包括构件） 已经创建了最初结构，确定了主要构件并设计了原型。

前景 已经根据此阶段获得的新信息进行了改进，对推动构架和计划决策的最关键用例建立了可靠的了解。

软件开发计划 已经进行了更新和扩展，以便涵盖构建阶段和产品化阶段。

指南，如设计指南和编程指南。 使用指南对工作进行了支持。

迭代计划 已经完成并复审了构建阶段的迭代计划。

用例模型 用例模型（大约完成 80%）- 已经在用例模型调查中确定了所有用例、确定了所有主角并编写了大部分用例说明（需求分析）。

补充规约 已经对包括非功能性需求在内的补充需求进行了记录和复审。

可选 里程碑状态

商业理由 如果构架调查不涵盖变更基本项目假设的问题，则已经对商业理由进行了更新。

分析模型 可能作为正式工件进行了开发；进行了经常但不正式的维护，正演进为设计模型的早期版本。

培训材料 用户手册与其他培训材料。根据用例进行了初步起草。 如果系统具有复杂的用户界面，可能需要培训材料。

1.4. 构建阶段

1.4.1. 目标

构建阶段的目标是阐明剩余的需求，并基于已建立基线的构架完成系统开发。构建阶段从某种意义上来说是一个制造过程，在此过程中，重点在于管理资源和控制操作，以便优化成本、进度和质量。从这种意义上说，从先启和精化阶段到构建和产品化阶段，管理上的思维定势经历了从知识产权开发到可部署产品开发的转变。 构建阶段的主要目标包括：

- 通过优化资源和避免不必要的报废和返工，使开发成本降到最低。

• 快速达到足够好的质量
• 快速完成有用的版本（Alpha 版、Beta 版和其他测试发布版）
• 完成所有所需功能的分析、开发和测试。
• 迭代式、递增式地开发随时可以发布到用户群的完整产品。这意味着描述剩余的用例和其他需求，充实设计，完成实施，并测试软件。
• 确定软件、场地和用户是否已经为部署应用程序作好准备。
• 开发团队的工作实现某种程度的并行。 即使是较小的项目，也通常包括可以相互独立开发的构件，从而使各团队之间实现自然的并行（资源允许）。这种并行性可较大幅度地加速开发活动；但同时也增加了资源管理和工作流程同步的复杂程度。如果要实现任何重要的并行，强壮的构架至关重要。

1.4.2. 核心活动

• 资源管理，控制和流程优化
• 完成构件开发并根据已定义的评估标准进行测试
• 根据前景的验收标准对产品发布版进行评估。

1.4.3. 里程碑：最初操作性能

最初操作性能里程碑确定产品是否已经可以部署到 Beta 测试环境。

在最初操作性能里程碑，产品随时可以移交给产品化团队。此时，已开发了所有功能，并完成了所有 Alpha 测试（如果有测试）。除了软件之外，用户手册也已经完成，而且有对当前发布版的说明。

1.4.3.1 评估标准

构建阶段的评估标准涉及到对以下问题的回答：

• 该产品发布版是否足够稳定和成熟，可部署在用户群中？
• 是否已准备好将产品发布到用户群？
• 实际的资源耗费与计划的相比是否仍可以接受？

如果项目无法达到该里程碑，产品化可能要推迟一个发布版。

1.4.3.2 提供的文档及模型

核心文档及模型（按照重要性排序） 里程碑状态

“系统” 可执行系统本身随时可以进行“Beta”测试。

部署计划 已开发最初版本、进行了复审并建立了基线。

实施模型（以及所有组成部分，包括构件） 对在精化阶段创建的模型进行了扩展；构建阶段末期完成所有构件的创建。

测试模型（和所有组成部分） 为验证构建阶段所创建的可执行发布版而设计并开发的测试。

培训材料 用户手册与其他培训材料。根据用例进行了初步起草。 如果系统具有复杂的用户界面，可能需要培训材料。

迭代计划 已经完成并复审了产品化阶段的迭代计划。
设计模型（和所有组成部分） 已经用新设计元素进行了更新，这些设计元素是在完成所有需求期间确定的。
项目专用模板 已使用文档模板制作了文档模板。
工具 已经安装了用于支持构建阶段工作的工具。
数据模型 已经用支持持续实施所需的所有元素（例如，表、索引、对象关系型映射等）进行了更新
可选 里程碑状态
补充规约 已经用构建阶段发现的新需求（如果有）进行了更新。
用例模型（主角，用例） 已经用构建阶段发现的新用例（如果有）进行了更新。

1.5. 产品化阶段

1.5.1. 目标

产品化阶段的重点是确保最终用户可以使用软件。产品化阶段可跨越几个迭代，包括测试处于发布准备中的产品和基于用户反馈进行较小的调整。在生命周期中的该点处，用户反馈应主要侧重于调整产品、配置、安装和可用性问题，所有较大的结构上的问题应该在项目生命周期的早期阶段就已得到解决。 在产品化阶段生命周期结束时，目标应该已经实现，项目应处于将结束的状态。某些情况下，当前生命周期的结束可能是同一产品另一生命周期的开始，从而导致产生产品的下一代或下一版本。对于其他项目，产品化阶段结束时可能就将文档与模型完全交付给第三方，第三方负责已交付系统的操作、维护和扩展。
根据产品的种类，产品化阶段可能非常简单，也可能非常复杂。例如，发布现有桌面产品的新发布版可能十分简单，而替换一个国家的航空交通管制系统可能就非常复杂。
产品化阶段的迭代期间所进行的活动取决于目标。例如，在进行调试时，实施和测试通常就足够了。 但是，如果要添加新功能，迭代类似于构建阶段中的迭代，需要进行分析设计。

当基线已经足够完善，可以部署到最终用户领域中时，则进入产品化阶段。通常，这要求系统的某个可用部分已经达到了可接受的质量级别并完成用户文档，从而向用户的转移可以为所有方面都带来积极的结果。
产品化阶段的主要目标是：

- 进行 Beta 测试，按用户的期望确认新系统
- Beta 测试和相对于正在替换的遗留系统的并行操作
- 转换操作数据库
- 培训用户和维护人员
- 市场营销、进行分发和向销售人员进行新产品介绍
- 与部署相关的工程，例如接入、商业包装和生产、销售介绍、现场人员培训
- 调整活动，如进行调试、性能或可用性的增强

• 根据产品的完整前景和验收标准，对部署基线进行的评估
• 实现用户的自我支持能力
• 在用户之间达成共识，即部署基线已完成
• 在用户之间达成共识，即部署基线与前景的评估标准一致

1.5.2. 核心活动

• 执行部署计划
• 对最终用户支持材料定稿
• 在开发现场测试可交付产品
• 制作产品发布版
• 获得用户反馈
• 基于反馈调整产品
• 使最终用户可以使用产品

1.5.3. 里程碑：产品发布

产品化阶段末是第四个重要的项目里程碑，即产品发布里程碑。此时，您确定是否达到目标，以及是否应该开始另一个开发周期。有时候，该里程碑可能与下一周期的先启阶段末重合。产品发布里程碑是项目验收复审成功完成的结果。

1.5.3.1 评估标准

产品化阶段的主要评估标准涉及到对以下问题的回答：

• 用户是否满意？
• 实际的资源耗费与计划的耗费相比是否可以接受？

在产品发布里程碑处，发布后的维护周期同时开始。这涉及开始一个新的周期，或某个其他的维护发布版。

1.5.3.2 提供的文档及模型

核心文档及模型（按照重要性排序） 里程碑状态

产品工作版本 已按照产品需求完成。客户应该可以使用最终产品。

发布说明 完成。

安装产品与模型 完成。

培训材料 完成，以确保客户自己可以使用和维护产品。

最终用户支持材料 完成，以确保客户自己可以使用和维护产品。

可选 里程碑状态

测试模型 在客户想要进行现场测试的情况下，可以提供测试模型。

2. 核心工作流程

软件工程中的工作流程分为两部分：核心工作流程与核心支持工作流程 核心工作流程（在项目中的流程）

• 业务需求建模
• 分析设计
• 实施
• 测试
• 部署

核心支持工作流程（在组织中的流程）

• 环境
• 项目管理
• 配置与变更管理

2.1. 业务需求建模

2.1.1. 目的

业务建模的目的在于：

• 了解目标组织（将要在其中部署系统的组织）的结构及机制。
• 了解目标组织中当前存在的问题并确定改进的可能性。
• 确保客户、最终用户和开发人员就目标组织达成共识。
• 导出支持目标组织所需的系统需求。

为实现这些目标，业务建模工作流程说明了如何拟定新目标组织的前景，并基于该前景来确定该组织在业务用例模型和业务对象模型中的流程、角色以及职责。

作为对这些模型的补充，还编写了以下文档：

• 补充业务规约
• 词汇表

2.1.2. 业务建模工作流程

2.1.3. 提供的文档与模型

•商业逻辑建模（USE CASE）（ROSE）
•业务需求说明书（MS WORD）
•专业词汇表（英汉对照）（MS WORD）
•风险说明（MS WORD）
•复审说明书

2.1.4. 文档模板

参见项目管理规范目录下业务需求文档模板子目录

2.2. 分析设计

2.2.1. 目的

分析设计的目的在于：

• 将业务需求转换为未来系统的设计。

• 逐步开发强壮的系统构架。

• 使设计适合于实施环境，为提高性能而进行设计。

2.2.2. 分析设计工作流程

2.2.3. 提供的文档与模型

•系统总体设计报告（MS WORD）

•系统设计模型 DOMAIN MODEL（ROSE）

•系统设计模型 DESIGN MODEL （ROSE）

•数据库设计模型 （POWER DESIGNER）

•数据字典（MS WORD）

•系统详细设计报告（MS WORD）

•工作量化书（MS WORD）

2.2.4. 文档模板

参见项目管理规范目录下分析设计文档模板子目录

2.3. 实施

2.3.1. 目的

实施的目的包括：

• 对照实施子系统的分层结构定义代码结构、

• 以构件（源文件、二进制文件、可执行文件以及其他文件等）的方式实施类和对象、

• 对已开发的构件按单元来测试，并且

• 将各实施员（或团队）完成的结果集成到可执行系统中。

实施工作流程的范围仅限于如何对各个类进行单元测试。系统测试和集成测试将在测试工作流程中进行说明。

测试的目的在于：

• 核实对象之间的交互。

• 核实软件的所有构件是否正确集成。

• 核实所有需求是否已经正确实施。

• 确定缺陷并确保在部署软件之前将缺陷解决。

2.3.2. 实施工作流程

2.3.3. 提供的文档与模型

•实施总结书（MS WORD）

•实施模型（ROSE）

•系统集成书（MS WORD）

•代码审核意见书（MS WORD）

•源代码（MS WORD）
•用户使用手册（MS WORD）
•错误解决记录手册（MS WORD）
•构件及其说明
2.3.4. 文档模板
参见项目管理规范目录下实施文档模板子目录
2.4. 项目管理
2.4.1. 目的
本部分的目标是，通过提供一些项目管理的环境，使这个任务更加容易完成。它虽然不是成功的秘诀，但它介绍了可以显著提高成功交付软件可能性的项目管理方法。
项目管理的目的是：
• 为对软件密集型项目进行管理提供框架。
• 为项目的计划、人员配备、执行和监测提供实用的准则。
• 为管理风险提供框架。
该工作流程主要侧重于迭代式开发流程的以下重要方面：
• 风险管理
• 计划迭代式项目，贯穿生命周期并针对特定的迭代
• 监测迭代式项目的进度、指标
2.4.2. 项目管理工作流程

2.4.3. 提供的文档和模板
•风险管理计划（MS EXCEL）
•工作计划书（MS EXCEL）
•风险列表（MS EXCEL）
•迭代计划（MS EXCEL）
•问题解决计划（MS EXCEL）
•测试计划书（MS EXCEL）
•系统集成计划书（MS EXCEL）
•子系统集成计划书（MS EXCEL）
•工作单（MS EXCEL）
•产品验收计划（MS EXCEL）
•评测计划（MS EXCEL）
•项目计划复审意见书（MS WORD）
•开发总结（MS WORD）
2.4.4. 文档模板

参见项目管理规范目录下项目管理文档模板子目录

2.5. 部署

2.5.1. 目的

部署工作流程用来描述那些为确保最终用户可以正常使用软件产品而进行的活动。

部署工作流程描述了两种产品部署的模式：

• 自定义安装

• 通过 Internet 使用软件

在每个实例中，都强调要在开发场所对产品进行测试，并在产品最终发布之前进行 Beta 测试。

尽管部署活动主要集中于产品化阶段，但在较早的一些阶段中也会有一些为部署进行计划和准备的活动。

2.5.2. 提供的文档和模板

•部署计划

•安装文档

•发布说明

3. 角色划分

角色是抽象的职责定义，它定义的是所执行的一组活动和所拥有的一组文档与模型。

角色通常由一个人或作为团队相互协作的多个人来实现。

项目团队成员通常要履行许多不同的角色职能；就象一个人可以担任许多职务，一个人也可以担任许多不同的角色。

角色并不代表个人，而是说明个人在业务中应该如何表现以及他们应该承担的责任。

分析员角色集

分析员角色集用于组织主要从事需求获取和研究的各种角色。

角色

• 业务流程分析员

• 业务设计员

• 业务模型复审员

• 需求复审员

• 系统分析员

• 用户界面设计员

3.1.1. 业务流程分析员

业务流程分析员通过概括和界定作为建模对象的组织来领导和协调业务用例建模。例如，确定存在哪些业务主角和业务用例，他们之间如何进行交互。

人员配备

担任业务流程分析员的人员应该善于简化工作，并且具有良好的沟通技巧。担任此角色的人员中必须要有具备业务领域知识的人才，但这种知识并不是所有人都必备的。

业务流程分析员应该准备好开展以下工作：

- 评估将在其中部署项目最终产品的目标组织的情况。
- 了解客户与用户的需求、策略和目标。
- 协调目标组织的建模工作。
- 在必要时对业务工程工作进行讨论和协调。
- 对目标组织中所建议的任何变更进行成本效益分析。

3.1.2. 业务设计员

业务设计员通过描述一个或几个业务用例的工作流程来详细说明组织中某一部分的规约。他指定实现业务用例所需的业务角色及业务实体，并且将业务用例的行为分配给这些业务角色及业务实体。业务设计员定义一个或几个业务角色和业务实体的责任、操作、属性和关系。

人员配备

担任业务设计员的人员应该善于协调，并且具有良好的沟通技巧。他最好具有业务领域的知识，但这并不是担任此角色的所有人都必需的。业务设计员需要熟悉用于获取业务模型的工具。

3.1.3. 业务模型复审员

业务模型复审员参与对业务用例模型和业务对象模型的正式复审。

人员配备

在大多数情况下，担任业务模型复审员的人员都需要具备业务领域的基本知识，或者对将用来实现业务自动化的技术具备基本的知识。业务模型复审员应该具备的另一种技能是详细了解所应用的业务工程技术。

3.1.4. 系统分析员

系统分析员通过概括系统的功能和界定系统来领导和协调需求获取及用例建模。例如，确定存在哪些主角和用例，以及他们之间如何交互。

人员配备

担任系统分析员的人员应该善于协调，并且具有良好的沟通技巧。担任此角色的人员中必须要有具备业务和技术领域知识的人才，但这些知识并不是所有人都必须具备的。

3.1.5. 用户界面设计员

用户界面设计员通过以下方法领导和协调用户界面的原型设计和正式设计：

- 分析对用户界面的需求，包括可用性需求；
- 构建用户界面原型；

• 邀请用户界面的最终用户参与可用性复审和使用测试会议；

• 对用户界面的最终实施方案（由设计员和实施员等其他开发人员创建）进行复审并提供相应的反馈。

人员配备

用户界面设计员不应实施用户界面。用户界面设计员的工作重点和时间都应集中在用户界面的设计和“可视化成形”，原因如下：

• 用户界面设计员所需的技能通常需要为当前的项目和应用程序类型（可能具有独特的可用性需求）而加以改进和优化，这需要投入时间并集中工作重点。

• 应该限制因“一心二用”而带来的风险，即用户界面设计员不应该因为实施方面的考虑（相对于可用性方面的考虑而言）而受到过多的影响。

3.2. 开发角色集

开发人员角色集用于组织主要从事软件设计与开发的各种角色。

角色

• 构架设计师

• 构架复审员

• 代码复审员

• 数据库设计员

• 系统设计员

• 设计复审员

• 实施员

• 集成员

3.2.1. 构架设计师

构架设计师负责在整个项目中对技术活动和工件进行领导和协调。构架设计师要确立每个构架视图的整体结构：视图的详细组织结构、元素的分组以及这些主要分组之间的接口。因此，与其他角色相比，构架设计师的见解重在广度，而不是深度。

人员配备

构架设计师必须多才多艺、成熟练达、洞察力强、经验丰富。这样，他才能在无法获得完整信息的情况下迅速领会问题并根据经验作出审慎的判断。更准确地说，构架设计师（或者构架团队的成员）必须兼具以下技能：

• 经验：既包括在问题领域的经验（通过彻底了解需求），也包括在软件工程领域的经验。对于一个构架团队，这些素质要求可由各团队成员来分别承担，但其中至少要有一名构架设计师能够把握项目的全局。

• 领导才能：能够推动各个团队的技术进展，并能在压力下作出关键性的决策然后将其贯彻到底。要提高效率，构架设计师和项目经理必须紧密协作。构架设计师主要负责解决技

术问题，项目经理主要负责解决行政管理问题。构架设计师必须有权在技术问题上作出决定。

• 沟通：能够赢得他人的信任，以对其进行说服、激励和指导。构架设计师不能靠命令进行领导，而必须要赢得项目中其他人员的赞同。为了提高效率，构架设计师必须赢得项目团队、项目经理、客户、用户群体以及管理团队的尊敬。

• 以目标为中心、积极主动，不懈地追求成效。构架设计师是推动项目发展的技术动力，而不是空想家。在其职业生涯中，成功的构架设计师一直都要在捉摸不定和承受压力的情况下作出折衷决定。构架设计师只有将注意力集中在该做的事情上，才能在项目中取得成功。

从专业角度看，构架设计师必须具备系统设计员的所有能力。

团队。如果项目较大，需要组建一个构架团队，则应尽量广聚贤才，使该团队既拥有广泛的经验，又对软件工程流程具有一致的认识。构架团队不应该是由各团队、领域或承包商的代表组成的委员会。软件构架设计是一项长期的工作，始终都需要配备专职人员。

3.2.2. 构架复审员

一般而言，构架复审员负责计划并执行对软件构架的正式复审。

人员配备

构架复审员角色的人员配备要求与构架设计师的人员配备要求相同，但前者更加注重于技术问题。虽然对领导才能、成熟程度、实用主义及注重结果这些方面的重视程度稍低，但这些方面仍然重要：复审员可能会发现构架方面的缺陷，并且有可能会因为影响项目的进度而不受欢迎。尽管如此，最好还是在问题可以解决的时候及早提出关键性的问题，而不是盲目地追随进度，致使项目团队步入歧途。构架复审员需要根据成本对风险加以权衡，并对影响项目成功的概括性问题保持一定的敏感性。构架复审员还需是善于说服的沟通者他应该能够提出并讨论对他人来说比较敏感的问题。

3.2.3. 代码复审员

代码复审员负责确保源代码的质量，并且计划和执行源代码复审。在复审活动中，代码复审员还负责有关返工的任何反馈意见。

3.2.4. 数据库设计人员

数据库设计员定义表、索引、视图、约束条件、触发器、存储过程、表空间或存储参数，以及其他在存储、检索和删除永久性对象时所需的数据库专用结构。相关信息记录在数据模型中。

人员配备

数据库设计员必须在以下方面具有扎实的应用知识：

• 数据库和面向对象的分析设计技术

• 系统构架，包括数据库和系统性能调整，以及硬件和网络负载平衡

• 数据库管理

• 了解实施语言和环境

3.2.5. 系统设计员

设计员定义一个或几个类的职责、操作、属性及关系，并确定应如何根据实施环境对它们加以调整。此外，设计员可能要负责一个或多个设计包或设计子系统，其中包括设计包或子系统所拥有的所有类。

人员配备

设计员必须在以下方面具有扎实的应用知识：

• 用例建模技术。

• 系统需求。

• 软件设计技术，包括：

o 面向对象的分析设计技术。

o 统一建模语言。

• 实施系统时将利用的技术。

3.2.6. 设计复审员

设计复审员计划并进行设计模型的正式复审。

人员配备

设计复审员的人员配备要求与构架设计师的人员配备要求相同，但前者更加侧重于技术问题。虽然对领导才能、成熟程度、实用主义及注重结果这些方面的重视程度稍低，但这些方面仍然重要：复审员可能会发现设计方面的缺陷，并且有可能会因为影响项目的进度而不受欢迎。尽管如此，最好还是在问题可以解决的时候及早提出关键性的问题，而不是盲目地追随进度，致使项目团队步入歧途。设计复审员需要根据风险对成本加以权衡，并对影响项目成功的概括性问题保持一定的敏感性。设计复审员还需是一个善说服的沟通者，他应该能够提出并讨论对他人来说比较敏感的问题。

从技术知识的观点来看，设计复审员应该具有与设计员相同经验。

3.2.7. 实施员（程序员）

实施员负责按照项目所采用的标准来进行构件开发与测试，以便将构件集成到更大的子系统中。如果必须创建驱动程序或桩模块等测试构件来支持测试，那么实施员还要负责开发和测试这些测试构件及相应的子系统。

人员配备

实施员应具备的相应技能和知识包括：

• 了解系统或所测试的应用程序

• 熟悉测试及测试自动化工具

• 编程技能

建议负责实施子系统的实施员同时应负责该子系统所包含的构件。

3.2.8. 集成员

实施员将经测试的构件交付到集成工作区，由集成员在集成工作区将构件组合起来，生成一个工作版本。集成员还负责制定集成计划。集成在子系统和系统级别进行，每次集成均有独立的集成工作区。正如经测试的构件从实施员的专用开发工作区交付到子系统集成工作区一样，已集成的实施子系统也从子系统集成工作区交付到系统集成工作区。

人员配备

有时，担任集成员的个人还可以担任实施员或测试员。例如，如果项目较小，或者是在子系统级别上进行集成，就可以让同一个人兼任集成员和测试员，以做到有效地利用人力资源。实际上，对于子系统级别的集成（和测试），一个人就可以兼任实施员、集成员和测试员的角色。但是，对于系统级别的集成，建议应由独立的团队来执行集成和测试。

3.3. 测试员角色集

测试员角色集用于组织主要从事软件测试的各种角色。

角色

- 测试设计员
- 测试员

3.3.1. 测试设计员

测试设计员是测试中的主要角色。该角色负责对测试进行计划、设计、实施和评估，包括

- 生成测试计划和测试模型
- 执行测试过程
- 评估测试范围和测试结果，以及测试的有效性
- 生成测试评估摘要

人员配备

测试设计员应具备的相应技能和知识包括：

- 了解系统或所测试的应用程序
- 了解测试及测试自动化工具
- 具备诊断和解决问题的技能
- 编程技能（最好具备）

3.3.2. 测试员

测试员负责执行测试，其职责包括：

- 设置和执行测试
- 评估测试执行过程并修改错误

人员配备

测试员应具备的知识和技能可能会因为他们所执行的测试类型和/或测试阶段的不同而有所

差异。例如，在执行性能测试或集成阶段的测试时，需要更高级的技能。在执行功能测试或系统测试阶段的测试时，则不需要太高级的技能。

以下是测试员所需知识和技能的一些标准：

高级测试员：

- 了解系统或所测试的应用程序
- 了解联网和系统构架
- 了解测试及测试自动化工具
- 具备诊断和解决问题的技能
- 编程技能（必备）

初级测试员：

- 了解系统或所测试的应用程序
- 了解测试及测试自动化工具
- 具备诊断和解决问题的技能
- 编程技能（最好具备）

3.4. 经理角色集

经理角色集用于组织主要从事软件工程流程的管理与配置的各种角色。

角色

- 变更控制经理
- 配置经理
- 部署经理
- 流程工程师
- 项目经理
- 项目复审员

3.4.1. 变更控制经理

变更控制经理这一角色负责对变更控制过程进行监督。此角色通常由配置（或变更）控制委员会 (CCB) 来担任，该委员会应该由有关各方（包括客户、开发人员和用户）的代表组成。在小型项目中，项目经理或软件构架设计师一人即可承担此角色。

3.4.2. 配置经理

配置经理负责为产品开发团队提供全面的配置管理 (CM) 基础设施和环境。CM 的作用是支持产品开发行为，使开发人员和集成员有适当工作区来构建和测试其工件，并且使所有工件均可根据需要包含在部署单元中。配置经理还必须确保 CM 环境有利于进行产品复审、更改和缺陷跟踪等活动。配置经理还负责撰写 CM 计划并汇报基于“变更请求”的进度统计信息。

3.4.3. 部署经理

部署经理负责制定向用户群体发布产品的计划，并将其纳入布署计划中。

3.4.4. 流程工程师

流程工程师对软件开发流程本身负责。其职责包括在项目开始前配置流程，并在开发工作过程中不断改进流程。

人员配备

担任流程工程师的人员需要具有广博的软件开发知识。良好的沟通技巧是对担任此角色的人员的基本要求。

3.4.5. 项目经理

项目经理负责分配资源，确定优先级，协调与客户和用户之间的沟通。总而言之，就是尽量使项目团队一直集中于正确的目标。项目经理还要建立一套工作方法，以确保项目工件的完整性和质量。

3.4.6. 项目复审员

项目复审员负责在项目生命周期中的主要复审点处评估项目计划工件和项目评估工件。在这些复审点发生的是非常重要的复审事件，因为在它们所标志的时间点处，如果计划不够充分或者进展无法令人满意，项目很可能会就此取消。

人员配备

项目复审员应具有多年的业务（包括合同制订及谈判）、技术和软件项目管理的经验，并具有业务管理级别的决策能力。项目复审员还应对风险管理原理具有出色的理解力，并且善于在信息不全或不明确的环境下进行评估。