融 RONG 管理社区
Product & Project Management

软件研发质量管理体系建设白皮书

V1.0

Software R&D Quality Management System Construction White Paper

融管理社区专家团队撰写：

宋　涛 / 孙正亮 / 陈晓鹏 / 刘　哲 / 刘　冉 / 周航宇 / 周　玲

何　凡 / 郑蒙正 / 郭智杰 / 卢腾飞 / 陈　磊 / 徐东伟

（排名依照章节承担顺序）

目录

前言

前言

本白皮书的背景和意义

软件在现代社会中发挥着越来越重要的作用，但软件质量问题也越来越受到人们的关注。尤其是在软件开发的过程中，可能存在的质量问题会导致交付产品质量下降、项目超预算和延迟交付等问题。在这种背景下，建立研发质量管理体系是非常重要的。

本白皮书的主要目的是指导团队如何建立软件研发质量管理体系，帮助软件开发团队提高交付质量，降低缺陷逃逸率和不良质量成本。具体来说，本白皮书为保证软件研发质量而建立的一套完整的流程、方法、实践等的整体内容，覆盖了管理职能、资源管理、产品发布、质量与改进四个方面。

通过阅读本白皮书，读者将会了解到建立软件研发质量管理体系的重要性，以及如何建立一套覆盖全面的、可以有效实施的质量管理流程和方法，以提高整体软件研发质量。

预期读者

本白皮书的目标读者是质量总监、质量经理、软件开发人员、软件测试人员、项目管理人员、其他质量保证工作相关人员，以及所有对质量关心的组织成员。

1. 需要参与质量管理体系建设和管理的管理人员和技术人员：包括 CTO、研发总监、产品总监、质量总监、质量经理、质量保证工程师、测试经理、项目经理、开发人员、产品经理以及测试人员等。本白皮书将为他们提供相关的指导和实践案例，帮助他们实现高质量软件开发和管理。

2. 具有软件开发和管理背景的专业人士：这些人士对于质量建设和保障的技术层面有深入的理解和实践经验，他们需要提升通过组织层面质量管理体系的建设来推动软件质量和持续改进的能力，以便能够更好地满足客户的需求。

3. 软件开发和管理领域的研究人员：白皮书对于软件开发和管理领域的研究人员也有一定的参考价值。他们可以从本白皮书对实战的指导中获得启发。

4. 已经或准备进入软件工程领域的学生：白皮书也适用于存在学习软件工程相关领域的学生，帮助他们更好地理解质量工程、质量管理体系建设和管理以及软件开发和项目管理的相关概念和实践案例。

总之，本白皮书的预期读者包括从事软件开发、项目管理、质量管理、软件测试等相关工作的各类人员，以及对于软件工程领域有所关注和了解的学生和研究人员。

策划和组织本白皮书的团队

宋涛、孙正亮、陈晓鹏、刘哲、刘冉、周航宇、周玲、何凡、郑蒙正、郭智杰、卢腾飞、陈磊、徐东伟共同完成了本次白皮书的撰写（排名按照章节承担顺序）。

第一章

质量管理体系概述

第一章 质量管理体系概述

1.1 质量管理体系的定义

全面质量管理的创始人，费根堡姆，将质量管理体系定义为“一个协调组织中人们的质量保持和质量改进努力的有效体系，该体系是为了用最经济的水平生产出客户完全满意的产品。”ISO9000 给出的定义是指“在组织建立的对质量进行指挥和控制的管理体系（To direct and control an organization with regard to Quality）”。所以说质量管理体系首先是一种管理体系，与人力资源管理体系、财务管理体系类似，是各种要求、资源、治理等有机组成的一个整体，是系统化的，是全局的而不是局部的。

其次，质量管理体系是组织内部建立的，是协调组织内部的人、资源，为组织能够持续满足客户需求、提高客户满意度而服务的。它是将资源与过程结合，以流程管理方法进行的系统管理。企业根据自身特点，选用若干体系要素加以组合，形成系统。质量管理体系一般包括管理职责、资源保障、价值创造以及度量分析与改进四大要素（见图 1-1，来自 ISO9001）。其中价值创造（又称产品工程）涵盖了从确定顾客需求、设计研制、生产、检验、销售、交付之前全过程的策划、实施、监控、纠正与改进活动的要求，是企业实现客户价值的最核心要素，也是企业流程优化的重中之重。

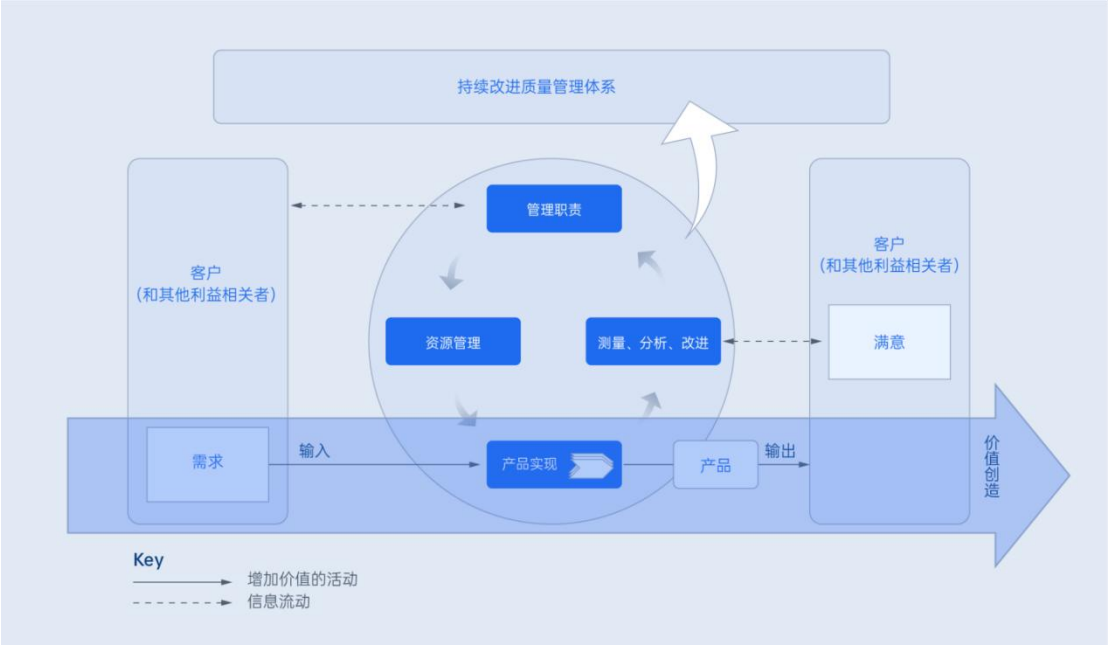


图 1-1 质量管理体系

再次，质量管理体系是为实现质量方针和目标而构建的，与组织的使命和战略方向是一致，如同环境管理体系是为环境服务的，职业健康安全管理体系是为职业健康和安全服务的，质量管理体系是为实现质量目标所必需的，是组织其他目标，如收入、盈利等的有效支持与保障。

最后，质量管理体系采用的手段是管理。所谓管理，是“用来指挥和控制组织的相互协作的活动，达成既定的组织目标”。因此，需要公司自上而下地推进，全员参与，并制度化、标准化，从而形成企业独特的文化。

1.2 质量管理体系的原则

质量管理体系的有效实施，因不同组织、不同行业的属性不同，在具体实施方法上会有所不同。但从“道法术器”的基本原理层面，它有着质量管理体系所必备的基本“原则”。

作为“原则”，它是一种基本的信念，理论或规则，对完成某项工作的方式具有重大影响。而“质量管理原则”，类似软件开发的敏捷宣言一样，是一整套公认的基本信念、规范、规则和价值观，可以用作质量管理的基础。国际标准化组织的 ISO9000 标准族定义了质量管理的 7 大原则（在 ISO 9001 质量管理体系 2008 年版及以前的版本中，是用的质量管理八项原则）。而在 ISO 9001:2015 质量管理中，删减了“系统管理方法”一项，它们是由 ISO/TC176 工作组的国际专家开发和更新的（ISO/ TC176 负责制定和维护 ISO 的质量管理标准），分别是原则 1 – 以客户为中心、原则 2 – 领导作用、原则 3 – 全员参与、原则 4 – 流程方法、原则 5 – 持续改进、原则 6 – 循证决策和原则 7 – 关系管理。

1.2.1 原则 1 以客户为中心

质量管理的主要重点是满足客户需要并努力超越客户期望。只有当组织能做到吸引客户，并维持、提振客户和其他利益相关方的信心时，才可以获得持续的成功。

以客户为中心，可以提高客户价值，增加客户满意度和忠诚度，从而维持长久的业务关系。通过组织的客户关注，提升了组织声誉，从而进一步扩大客户群体，增加组织的收入和市场份额。

以客户为中心，可以让企业通过与客户全方位地互动，为客户创造更多价值。而这些互动，包括了客户的沟通，高层的互访，客户需求的讨论，售后的反馈，客户的申诉等。每一次互动，提供了了解客户和其他利益相关方当前和未来需求的机会，保障了组织的持续成功。

针对 B 端客户，常见的客户聚焦的行动包括了如何识别组织的直接或间接客户，了解客

户当前和未来的需要和期望（Needs vs. Want），将组织的管理目标与客户需要和期望联系起来，并充分沟通客户需要和期望。同时，需要组织从产品的规划、设计、开发、生产到产品的交付和支持及服务，实现制度化、流程化，以满足客户的需要和期望，并能够度量和监控客户满意度，采取适当的措施，以积极、主动地管理客户关系，确定并采取可能影响客户满意度的利益相关方的需要和期望。

1.2.2 原则 2 领导作用

企业各级领导者需要建立统一、一致的使命、愿景和战略方向。企业领导者应将质量作为组织重要战略之一，落实在行动中，并努力创造条件，激励企业全员参与实现组织的质量战略目标。

领导的积极参与和重视，能够提高实现组织质量目标的效率和效果。企业领导者身体力行地实践质量管理的理念，能够让全体员工“听其言、观其行”，自觉实践领导的意图，使质量战略的实施，达到事半功倍的效果。

建立统一目标，能够使组织调配其战略、政策、流程和资源。通过组织的制度及流程协调资源，改变“人治”的管理模式，改善组织各层级和职能部门之间的沟通，降低沟通成本。同时，创造条件，激励员工积极参与质量目标建设，则能够发展与改善组织及其员工交付预期结果的能力，集中优势，有效实现其目标。

不同企业，其管理风格各不相同，但归纳总结起来，质量管理成功的企业，如丰田、摩托罗拉、宝马等，都能够做到：及时、准确地传达组织的使命、愿景、战略、政策和流程；对组织各层级的行为做出规范，创建并维持公平的环境，建立共享的价值和道德规范，并培育信任和诚信的文化。

企业应该鼓励整个组织对质量的承诺，各级领导起到模范带头作用。

企业的领导作用还包括为员工提供所需的资源、培训。并充分授权、激发、鼓励、认可员工的贡献。

1.2.3 原则 3 全员参与

拥有具备竞争力的、高度授权、高度投入的团队是整个组织持续创造和交付客户价值的关键所在。通过认可每一个人的贡献，并建立授权机制和完善的培养体系，这可以提高每一个个体的能力，从而促使每个人都参与实现组织的质量目标。

全员参与，重要的是尊重每一个个体都是“不一样的烟火”，要使所有人都参与进来。

这样，组织的全体成员能够更好地理解组织的质量目标，激发实现该目标的动力。通过全员参与，能够促进人们积极参与质量改善活动，从个人方面，可以提高个人主动性和创造力，有益于个人职业发展。对组织而言，能够在组织中增强信任与合作，提高员工对共享价值的关注，有利于整个组织文化的形成。

要实现全员参与，组织的管理风格及人事制度可以从几个方面强调：

- 沟通透明：让组织的每一位成员了解，作为个体，对质量目标实现的重要性及其贡献的价值；
- 文化倡导：在组织中倡导“协作”而不是“背锅”的文化，鼓励员工谨慎冒险精神，赞赏、认可员工的贡献；
- 工作氛围：创建公开透明的工作氛围，鼓励知识和经验的共享。如有可能，实施员工满意度调查，及时沟通调查结果并采取改进措施；
- 评价体系：让员工能对个体目标作出自我评价，并自驱改进，并对其进步及时正向反馈。

1.2.4 原则 4 过程（Process）方法

“过程（Process）”，在某种意义上，被某些自媒体、宣传媒介妖魔化，被定性为外资企业在国内发展不顺的绊脚石。然而，作为工业革命的重要产出物，我们需要意识到，只有将企业经营活动理解为相互关联、相互依赖的过程，以流程管理作为中枢神经，才能促使企业各部门如同一部精密钟表的零部件一样，相互协调、密切合作，成为一个连贯的，高效的系统。只有这样，企业交付的产品或服务质量才能更稳定，从而提振干系人对未来预期的信心，而不必担心出现飘忽不定的大起大落。

企业的经营活动是由一系列相互关联、相互依赖的过程活动构成的一个有机整体。而质量管理体系则是描述这些企业活动过程，包括但不限于必要的体系文档，过程交互，资源管理，度量系统，人员配置等。所以，只有理解质量管理体系的内涵，系统化地管理该体系，组织才可以优化其经营活动，提高组织产出。

应用流程管理的方法，就是促使组织对满足内、外客户需求的理解保持一致性，企业经营、生产和研发过程强调对客户的增值而不是部门利益保护。流程管理的方法需要根据客观事实、数据及信息对经营、生产和研发过程持续改进，确保流程与时俱进，提高组织的适配性和敏捷性。通过流程管理的方法，可以最大化地利用资源，优化企业经营表现，打穿部门墙，降低部门间的内耗。而流程体系化，将企业主要的客户增值流程通过质量管理体系显性表达，能够聚焦于重要的和关键的客户增值流程及其改进机会，使各

经营活动之间顺畅衔接，保障了企业交付的一致性及稳定性，减少了内卷，提振了企业干系人对企业经营的效果及效率的信心。

需要强调的是，流程管理方法，不是简单地把经营活动文档化。实际上，流程管理更像产品的设计，需要高屋建瓴的系统化思维，从架构设计到详细设计、各流程域、流程模块要环环相扣，如同齿轮一样，协调运作。流程体系的建设，可按下列五步法进行：

（1） 行动前，了解组织目前的能力和资源约束、管理干系人的期望。

（2） 确定流程体系的目标并识别达成该目标所需要的必要流程。

（3） 确定流程之间的依存关系。

（4） 系统化管理流程及其相互关联的流程，对单个流程的修改要分析其对整个系统的影响。

（5） 建立流程管理的职责及授权和问责制度，保障流程体系能够制度化、标准化，并落地执行。通常，在传统 B 端行业，如汽车行业，这部分职责落在质量管理部门。

除此之外，在实施流程管理方法过程中，我们还需要：

- 确保信息的透明、通畅。对整个系统进行监控、分析和绩效评估；
- 针对可能影响流程和整体质量管理体系输出成果的风险进行管理。

1.2.5 原则 5 持续改进

成功的组织是非常关注持续改进的。只有持续改进与优化，组织才能保持当前的绩效水平，对内部和外部环境的变化及时做出反应并创造新的机遇。

持续改进是组织自我愈合、自我改进的重要原则。如质量原则 QMP 4 所言的流程化管理，只有通过持续改进，才能保障流程性能改进，组织能力提升和客户满意度增强。如果想要进行持续改进，就需要了解持续改进的方法和技巧，培养组织及员工追根溯源的精神，定义并执行纠正、预防措施的能力。同时，增强了对内部和外部的风险、机遇的预期和反应能力，以及组织的创新能力。

持续改进，可以是自上而下的改进，也可以是自下而上的改进。通常，系统性的重大改进或对组织影响深远、范围广泛的改进，都需要自上而下的领导力支持，以确保改进的效果。但不管何种改进，都需要组织在各个级别建立改进目标，倡导组织持续改进的文

化；都需要定义和部署持续改进相关的流程，遵循项目管理的方法，在组织中实施改进项目。同时，从技能上，为确保人们有能力成功地促进和完成改进项目，应在各个层次上教育和培训员工如何应用基本的持续改进工具和方法，例如，质量管理基本工具、精益工具、六西格玛工具等，来实现改进目标。

在改进文化成熟的公司中，常见的持续改进的方法通常遵循一下原则：

（1） 识别改进的机会（problem statement）。

（2） 得到利益相关方的支持，成立改进项目。

（3） 跟踪、评审和审计改进项目的计划、实施、完成和交付物，确保改进的效果。通常，改进过程的实施可以按照 DMAIC[1]的小迭代模式快速推进。

（4） 如果改进效果显著，将改进事项应用到新的商品、服务和流程中，并将之制度化，确保改进成果能够保持。

（5） 认可并表扬改进项目团队。

1.2.6 原则 6 循证决策

基于数据和信息的分析和评估，能够让我们的决策更有可能产生预期的结果。我们都知道，决策是一个复杂的过程。它的复杂性在于决策时太多的不确定性因素。这些因素通常涉及多种类型和来源的输入，以及对输入的主观解读。所以，了解事物的深层因果关系及可能的、预期之外的后果，对决策是非常重要的。通过事实、证据和数据分析，推理预演可能各种方案，可以有效提高我们对决策的客观性和决策信心。

循证决策，可以帮助我们改善决策的流程和效率、有效度量流程的绩效及组织取得既定目标的能力。循证决策，还可以提高我们的运营效率，通过对历史经验数据的分析，沉淀知识，增强组织自我审查、挑战各种决策选项的能力。

要支持循证决策，组织需要利用系统化的管理手段，制定、测量和监控与组织管理绩效相关的关键指标，收集、萃取数据，并能够最大程度地做到数据的透明，促使全员可以有效方便地访问数据，确保人员有能力根据需要分析和评估数据。循证决策需要数据，但并不是完全依赖数据。由于数据在收集过程中可能被污染，我们需要确保数据和信息足够准确、可靠和安全。要使用适当的方法，比如假设检验、帕累托图（Pareto）、散

[1] DMAIC：是六西格玛管理中流程改善的重要工具，指定义（Define）、测量（Measure）、分析（Analyze）、改进（Improve）、控制（Control）五个阶段构成的过程改进方法，一般用于对现有流程的改进，包括制造过程、服务过程以及工作过程等等。

点图等图表、回归预测等，全面分析评估数据和信息。同时，根据证据做出决定并采取行动，需要结合过去的经验和直觉而不是完全放弃。

1.2.7 原则 7 关系管理

为了获得持续的成功，组织需要管理与利益相关方（例如供应商）的关系。利益相关方是组织生态环境的重要组成部分，包括了供应商、合作伙伴、战略联盟、加盟商、投资伙伴、社区等。他们的绩效会直接影响组织的绩效。当组织管理与所有相关方的关系，优化其对自己绩效的影响，则更有可能获得持续的成功。其中，对供应商和合作伙伴的关系管理尤为重要。

管理完善的供应链，可提供稳定的商品和服务。通过对供应商和合作伙伴的关系管理，可以有效识别增强组织绩效。与利益相关方理解彼此的目标和价值，甚至通过共享资源、技能以及管理与质量相关的风险，增强为利益相关方创造价值的能力。

要对利益相关方进行管理，首先，组织需要确定谁是利益相关方及其与自己的关系。常见的利益相关方包括供应商、合作伙伴、客户、投资者、员工以及整个社会。其次，我们需要确定需要管理利益相关方关系的优先级。同这些利益相关方建立长期战略与短期利益的平衡关系，与其共享信息、专业知识和资源。最后，需要度量利益相关方的绩效并酌情向其提供绩效反馈，建立改进计划，与供应商、合作伙伴和其他相关方共同进步。当然，对供应商和合作伙伴的成就及进步也要及时鼓励和认可。

1.3 质量管理体系的目的和意义

1.3.1 质量管理体系重要性

质量管理体系为广大企业完善管理、提高产品和服务质量提供了科学指南，同时为企业走向国际市场找到了“共同语言”。

近些年，各行各业将加快推进国际标准化进程，“标准贯彻”变得更加迫切。毋庸置疑，“标准贯彻”不是万金油，不能包治百病，但通过质量管理体系的建立及其符合国际标准的选取：

1. 增强了企业全体员工的质量意识与管理意识，明确了各项管理的职责和工作的程序，促使企业的管理工作由“人治”转向“法治”，真正做到了凡事有人负责、凡事有章可循、凡事有据可查、凡事有人监督，实现了“以预防为主”。

2. 规范了企业的作业程序，明确了各部门和全体员工的职责和权限，预防并控制了不合格的发生，降低了企业质量管理成本。

3. 通过定期组织质量检查、质量审核活动，能够及时发现和找出经营管理活动、服务质量方面存在的问题和薄弱环节，并进行有效纠正，提高了企业整体经营管理水平和质量监控能力，为企业实施全面的科学管理奠定了基础。

4. 贯彻了“以人为本”的原则，全面提高了员工的业务技能和综合素质，为企业长远发展打下了坚实的基础。

5. 围绕“让客户满意”及时认真地处理客户投诉或意见，不断满足客户需求与期望，赢得客户信任，提高客户满意度，提升企业的社会形象和市场竞争力。

1.3.2 质量管理体系的收益

质量管理体系的作用，如果企业能够真正踏踏实实将其落地执行，将会得益良多，大大节省成本，提高效率。

1. 提高组织的形象和信誉

当顾客看到您被认可的认证机构认证，他们会明白，您已经实施了一个以满足顾客要求和改进为关注焦点的体系。他们会更加相信您的承诺。

2. 提高顾客满意

质量管理体系的一项主要原则是识别并满足顾客要求和需要，以提高顾客满意为关注焦点。顾客满意度提高了顾客的回头率也就相应提高。

3. 全面整合的过程

质量管理体系原则要求过程方法，您不仅看到组织中的各个过程，也看到这些过程的相互作用，这样就更容易识别到组织内需要改进和节约资源的领域。

4. 使用基于证据的决策

确保在证据充分的基础上作出决策，是质量管理体系成功的关键。确保决策有充分的证据支持，就可以把资源用到刀刃上，以最好的效果纠正问题、改进组织的效率和效益。

5. 创造持续改进的企业文化

以持续改进作为质量管理体系的主要输出，您在时间、资金和其他资源上的节省可见日益增加。将持续改进作为企业文化，可将员工的工作重点放到对他们直接负责的过程的改进。

6. 员工参与

在某个过程中工作的员工，是改进该过程的最好帮手。调整员工的工作重点，让他们不仅管理，而且还参与改进过程，他们会更加与组织休戚与共。

1.3.3 软件企业质量管理体系的作用

建立好的研发质量管理体系对于企业的软件开发和研发工作非常重要。

1. 提供高质量的软件产品

企业最终要提供的是能够满足客户需求的软件产品，在软件开发过程中，建立好的研发质量管理体系，通过各种规范和标准将研发过程的每个环节予以衡量和标识，从而掌握产品开发并控制系统开发风险，能够在代码、需求、文档、测试等方面综合考虑，确保软件质量（参阅 ISO25010 中有关软件质量的定义）达到客户预期。强调高品质的开发过程，具有在项目中提高软件产品质量标准，降低产品缺陷率的能力。在市场竞争激烈的今天，这一点对于企业的成功至关重要。

2. 提高软件开发效率和降低成本

通过研发质量管理体系着重的质量内建（Build-In Quality）、模块化和重用性，高质量的研发成果相关的潜在危险和维护成本将降低，从而系统性的降低整个的开发成本。在长期的软件开发过程中，缺陷率更低的软件需要更少的维护工作，节约开发成本，同时提高了开发的质量。

3. 提高客户满意度和信任度

通过完善的研发质量管理体系，提供高品质的软件产品，让客户对企业的开发和研发能力产生信任，提高客户对企业品牌和产品的满意度，为企业长期的发展壮大奠定良好的基础。从而推动企业的长期稳定发展。

4. 强制自我评估和改进

建立研发质量管理体系过程中，普遍会针对流程文档和内部标准和规程进行自我评估、调整和改进。研发质量管理体系要求及时发现研发中的问题，能够更快地提高研发过程

的质量，为最终产品提供更好的品质保证。长期的一个处于不断成长和改进状态的研发质量管理体系能够有力地支持研发过程。

研发质量管理体系的建立对于企业来说具有非常高的实践价值。通过建立研发质量管理体系，能够确保软件开发的高品质，提高开发效率和降低整个研发过程的成本，并且提升公司在市场中的竞争力。

第二章

确立软件质量管理体系建设思路

第二章 确立软件质量管理体系建设思路

2.1 分析现状问题

分析现状问题是指对当前软件研发过程的质量状况进行全面的分析和评估，找出存在的问题和不足之处。以下是分析现状问题的一些要点：

1. 收集数据

收集软件研发过程中的各种数据，包括缺陷率、任务完成时间、代码质量等。这些数据可以反映软件研发过程的质量状况，为后续的问题分析提供依据。

2. 分析问题

对收集到的数据进行深入分析，找出软件研发过程中存在的问题和不足之处。可以采取多种分析方法，如趋势分析、因果分析、鱼骨图分析等，以便更好地挖掘问题的本质和原因。

3. 归纳总结

将分析结果进行归纳和总结，找出主要的问题和瓶颈环节。针对不同的问题，可以采取不同的解决方案和方法，如改进流程、优化工具、提升技能等，以便更好地提升软件研发过程的质量。

需要注意的是，分析现状问题需要全面、客观、准确地进行，以确保分析结果的真实性和有效性。同时，需要根据实际情况不断调整和完善，以便更好地适应软件研发过程的变化和需求。

2.2 确定建设范围

确定建设范围是指在进行软件研发质量管理体系建设时，明确建设的范围和目标，以便有针对性地进行建设工作。以下是确定建设范围的一些要点：

1. 明确建设目标

确定软件研发质量管理体系建设的目标，如提高软件质量、降低缺陷率、优化研发流程

等。建设范围应该围绕这些目标进行规划和设计。

2. 确定建设范围

根据建设目标，确定软件研发质量管理体系的建设范围，包括需要涵盖的部门、团队、流程等。建设范围应该根据实际需求进行合理划分，避免过于宽泛或过于狭窄。

3. 确定建设内容

根据建设范围，确定软件研发质量管理体系的建设内容，包括需要建立的质量管理流程、质量保证措施、质量评估标准等。建设内容应该与建设目标相匹配，以便有效实现建设目标。

4. 确定建设周期

根据建设范围和建设内容，确定软件研发质量管理体系的建设周期，包括建设开始时间、建设结束时间、建设周期内的具体计划等。建设周期应该合理安排，以便在保证建设质量的同时，尽快实现建设目标。

需要注意的是，确定建设范围是软件研发质量管理体系建设的重要环节，应该根据实际需求进行合理规划，确保建设工作的有效性和可行性。同时，在建设过程中应该不断调整和完善，以便更好地适应软件研发过程的变化和需求。

2.3 获取管理支持

获取管理支持是指在进行软件研发质量管理体系建设时，获取高层管理者的支持和参与，以确保建设工作的顺利推进和成功实施。以下是获取管理支持的一些要点：

1. 明确管理支持的重要性

向高层管理者介绍软件研发质量管理体系建设的重要性，包括提高软件质量、降低风险、提升组织能力等。使高层管理者认识到他们在建设工作中的重要性和作用。

2. 建立沟通渠道

与高层管理者建立有效的沟通渠道，及时汇报建设进展、解决问题、获取支持等。保持良好的沟通，使高层管理者能够了解建设工作的具体进展和困难，并提供必要的支持和帮助。

3. 展示成果和效益

在建设过程中，及时展示取得的成果和效益，如降低缺陷率、提高软件质量、提升客户满意度等。让高层管理者看到建设工作的实际效果，以便更好地获得他们的支持和认可。

4. 培训和管理层参与

对高层管理者进行软件研发质量管理的培训，使他们了解建设工作的具体内容和流程。同时，邀请高层管理者参与建设工作中的重要活动和会议，以便更好地推动建设工作的开展。

5. 定期汇报和监控

定期向高层管理者汇报建设工作的进展情况、存在的问题和解决方案等。同时，建立监控机制，及时发现和解决问题，确保建设工作的顺利进行。

需要注意的是，获取管理支持是软件研发质量管理体系建设的关键环节，应该高度重视并与高层管理者保持有效的沟通和合作。同时，在建设过程中应该不断调整和完善，以便更好地适应软件研发过程的变化和需求。

第三章

制定质量政策和质量目标

第三章 制定质量政策和质量目标

3.1 质量政策制定

质量政策是对组织质量理念的高层次诠释和承诺，指导质量管理体系建设的方向。制定质量政策需要注意以下实践步骤：

1. 调研分析阶段

通过调查和访谈，分析现有质量问题和需求改进的方面，确定质量政策需要重点关注的领域。

2. 政策草稿阶段

起草质量政策的内容框架，明确质量政策需要表达的理念、原则和价值取向。内容上要简洁明确，易于传播。

3. 征求意见阶段

将质量政策草稿发布，在公司内部广泛征求员工和管理层的意见建议，吸收整合有价值的想法。

4. 评审修改阶段

组织评审会，由公司高层和质量管理团队组成的专家进行评审，提出修改意见，进一步优化完善。

5. 发布实施阶段

质量政策经过评审后，由最高管理者发布和批准，并在公司范围内传达贯彻。通过培训、议事等使员工理解质量政策。

6. 持续评估阶段

实施一段时间后，评估质量政策的实际成效，如产生的影响和改变，根据情况对政策进行动态调整和更新。

质量政策的样式应该简短和原则性。这主要基于以下考虑：

- 质量政策定位为高层方针，其作用是阐释和导向，而不应过于具体；
- 简洁的语句可以增强可传播性和可记忆性；
- 原则性指导可以适应组织的变化和长期发展；
- 具体执行要求应在质量目标和质量管理流程中展开；

但是，过于空泛的质量政策也会显得形式化和宣传式，为使质量政策兼具指导性和可操作性，建议：

- 在简洁原则的前提下，增加某些具体的管理诉求；
- 结合组织实际，使政策内容更具针对性；
- 在政策中连接到质量目标，形成闭环；
- 不时复核政策，确保与组织发展方向一致。

如果能在简洁和具体之间找到平衡，使质量政策既有原则性又有可操作性，就能够使之成为真正有价值的指导文件。

质量政策示例 1：A 公司质量政策

我们公司立志成为最值得信赖的软件解决方案提供商，致力于为客户提供高质量、高效率的软件产品和服务。

为达成这一愿景，我们公司坚持以下质量理念：

- 以客户为中心，倾听客户声音，持续优化客户体验；
- 培育质量优先文化，使质量意识深入员工日常工作的方方面面；
- 加强过程管控，落实质量管理体系，持续改进软件生命周期中的各环节；
- 强化预防理念，从源头减少缺陷的产生；
- 开展持续的质量改进，确保产品和服务持续进步和超越客户期望。

我们全体员工将致力于贯彻这些质量理念，以赢得客户信任和满意度。公司管理层也将提供全力支持和资源保障，以实现质量政策中阐述的承诺。

质量政策示例 2：B 公司质量政策

我们公司立志成为业界公认的高质量软件提供商。为实现这一目标，我们将遵循以下质量方针：

- 以客户为中心，深入理解客户需求，提供可靠、高效的软件解决方案；
- 培育全员质量文化，营造积极主动的质量改进氛围；
- 强化过程管控，持续优化全生命周期质量管理流程；
- 加强质量数据监测和分析，及时发现和解决质量问题；
- 开展质量培训和交流，提升全员质量管理意识；
- 加大质量管理基础设施建设投入；
- 鼓励创新，运用新工具、新技术提高质量效率；
- 与客户和合作方保持高度的质量共识。

我们全体员工承诺恪守质量政策，与客户和合作伙伴一起，持续提升产品和服务的质量水平。

3.2 质量目标设定

3.2.1 概述

质量目标的设定是质量管理的关键环节之一，合理的质量目标将指导质量管理实践的开展。质量目标设定分为长期和短期。长期质量目标是组织对产品质量的承诺，也是组织持续改进产品质量，传播质量文化，提高客户满意度的指南针。短期质量目标是组织在年内需要达成的具体质量目标，短期质量目标可以是组织级的复合指标，也可以是项目级别的项目交付质量目标。

组织质量目标，需要分解落实到具体的战略行动或项目，并清晰确认责任人/单位，确保目标的逐步，有序顺利实现。

设定质量目标时，需要注意以下几点：

1. 质量目标要具体、可测量

质量目标不能停留在语义模糊的阶段，要设定具体的、可测量的目标指标，如“降低产品缺陷率到 0.5%”。可测量的目标便于衡量进度和考核达成情况。

2. 质量目标要与业务目标相联动

质量目标要紧密服务于业务目标，比如降低质量成本、提升用户满意度等。要进行业务影响分析，评估质量目标对业务结果的促进作用。

3. 重点关注客户需求

质量目标的制定要以客户需求为导向，比如提高产品易用性、用户体验等。可以通过客户调研、质量数据分析等确定客户关注的质量要点。

4. 考虑现有资源约束条件

在设定质量目标时，要考虑现有的资源条件、能力限制，不要设定超出可实现范围的目标。目标要与现状匹配，同时也要有适当的提升空间。设定质量目标时，需要考虑对应的质量指标的数据收集的可行性和采集成本，如有必要，实施改进项目，实现数据采集的数字化手段。

5. 质量目标要层层分解

高层的质量目标需要分解为各部门、团队的质量目标，还要分解为每个人员的质量目标，形成全员参与的目标体系。

6. 开展质量目标评审

设定的质量目标需要提交评审，由质量管理团队、相关部门主管进行评审，确保质量目标合理、可实现。还可以组织专题评审会进行讨论。

7. 持续监控和评估目标达成情况

质量目标达成情况要定期进行监控，如果出现需要调整的情况，要及时更新和调整质量目标，做到动态管理。

3.2.2 常见质量目标指标

1. 产品完成度

（1） 需求通过率（已通过需求/已计划需求），体现需求的完成度，也可以统计为测试用例通过数/计划的测试用例总数计算（用测试用例计算隐含了默认用例覆盖是完全的这样的前提条件）。

（2） 功能点通过率（已通过功能点/已测试功能点），同上，当需求规模比较大时，功能点统计会更有价值，难点在于，需求功能点需要有额外的过程进行确认，一般在测试分析阶段统计拆分功能点。

（3） 风险规避情况（已规避风险/已预估风险），产品已知风险的应对情况，需要风险分析过程的支持。

（4） 需求稳定性（需求变更数/需求总数），衡量一个周期内需求的稳定性。

2. 产品质量

（1） 测试通过率（已执行测试数/已计划测试数），比较直观的数据，通过测试的通过率来衡量产品质量。

（2） 缺陷密度（缺陷总数/千行代码数），缺陷密度对于产品质量而言是非常直观有价值的，但由于千行代码数这一度量并不多用，对测试而言也可能获取存在难度，所以经常可以转化为（缺陷总数/功能点数）*100%、（缺陷总数/对应模块）（缺陷分布率）、（缺陷总数/交付需求数）*100%等计算方式，但是具体选择哪一种计算方式还需要依据团队相关实践来决定。

（3） 缺陷严重级别分布（对应严重级别缺陷数/缺陷总数），缺陷的数量并不能总是体现出产品实际质量，比如最严重级缺陷过多显然是一个问题，所以缺陷统计应该体现数量和严重级别的二维分布。

（4） 缺陷类型分布（对应类型缺陷数/缺陷总数），通过对应缺陷类型分布比例来衡量软件某一方面的质量。

（5） 缺陷模块分布（对应模块缺陷数/缺陷总数），通过对应缺陷模块分布比例来衡量软件某个模块的质量。

（6） 缺陷修复率（已修复缺陷数/缺陷总数），缺陷已被修复的比例统计。

3. 测试完成度

（1） 用例覆盖率（已设计用例数/计划设计用例数），用于监控测试设计的进度情况。计划设计用例数这一数字比较模糊可能来自估算。可以采用自下而上的方式收集：即让模块测试负责人进行局部数据收集，再汇总统计。

（2） 测试执行率（已执行的测试数/计划执行的测试数），测试已被执行的情况，用于测试进度跟踪。执行率并不关注测试失败情况，进一步细化可以展开统计测试通过、失败、阻塞和未执行的比率。

（3） 测试通过率（已通过测试数/计划执行的测试数），测试通过比率。

4. 研发过程质量

（1） 缺陷生存周期（缺陷生存总时长/缺陷数），通过统计缺陷从打开到关闭的平均时长，衡量研发团队的缺陷修复能力。

（2） 测试用例命中率（缺陷数量/用例数量），通过用例发现的缺陷数量统计，衡量测试设计的有效性。

（3） 二次故障率（缺陷二次重开数量/缺陷总数量），缺陷多次重开会造成缺陷修复周期拉长，说明 1. 开发修复缺陷能力存好 2. 测试团队缺陷压量存好 3. 开发理试之间沟通效率存好，需要具体分析。

（4） 缺陷有效率（有效缺陷数量/缺陷总数量），测试团队提交的缺陷有多少比重是有效缺陷，应该就具体缺陷失效原因进行分析。

（5） 缺陷移除率（缺陷当阶段移除数量/当阶段引入缺陷数量），用干统计研发等阶段的缺陷数量和在当阶段被解决的比例，实际是测试尽量介入的体现，比如需求中的缺陷需要在当阶段通过需求评审等手段探测并解决，此数据统计起来有一定难度。

5. 计划偏离度量

（1） 工作量偏离（（实际工作量-计划工作量）/计划工作量），用于衡量计划的合理度，是否有大量计划外工作未被纳入估算当中。

（2）工作进度偏离（已超出计划进度的时间），通过统计工作进度的偏离来揭示项目时间风险。

（3）预算使用比例（已花费测试预算（人/天）/计划总测试预算），用于计算测试预算的花费情况。

（4）问题等待时间（具体问题等待解决时间），等待时间通常难以被计划，通过计算等待时间可以帮助衡量项目瓶颈所在，并为后续项目组织提供思路。可细化为需求等待时间，测试阻塞时间等。

6. 产品质量趋势

（1） 缺陷到达率（缺陷数量/时间周期），周期性的缺陷接出数量，比如月缺陷到达率，周缺陷到达率，通过持续时间的到达率监控，可以体现项目产品的趋势。

（2） 缺陷收敛度（缺陷遗留数量/时间周期），通过统计遗留缺陷随时间推移的趋势，判断后续产品质量的走向。理想情况下，单迭代周期内缺陷数量经过集中爆发后，应呈持续走低态势。

（3） 缺陷引入率（新增缺陷数量/新增千行代码数），用以衡量产品的增量和修改对于质量的影响，也为后续产品的更新迭代提供参考指标。

第四章

建立质量型组织

第四章 建立质量型组织

4.1 质量型组织设计

质量型组织设计是指在一个组织内部建立一种质量文化，并通过合理的组织结构和职责分工，确保软件研发过程的质量。以下是质量型组织设计的一些要点：

1. 质量型组织结构的确定

根据软件研发规模和组织规模，设计合适的组织结构，明确各级别之间的职责和关系。例如，可以设立质量管理部门或质量保证小组，负责软件研发质量管理体系的建立和实施。

2. 质量职责的分配

将质量职责分配给各个部门或岗位，确保每个部门或岗位都能够承担相应的质量责任。例如，开发部门负责编码和测试，质量管理部门负责质量保证和质量控制。

3. 质量流程的设计

设计合理的质量流程，包括代码审查、测试、缺陷管理等，确保每个环节的质量得到有效控制。同时，需要明确每个环节的责任人和流程执行的标准，以便在具体实施过程中进行统一的指导和要求。

4. 质量工具的配备

根据需要配备相应的质量工具，例如自动化测试工具、代码审查工具、缺陷跟踪工具等。同时，需要明确每个工具的责任人和使用方法，确保工具的有效使用和顺利运行。

5. 质量计划的制定

制定软件研发的质量计划，明确软件研发的质量目标、质量标准、质量保证措施等。同时，需要明确质量计划的执行人和执行时间，确保计划的有效实施。

需要注意的是，质量组织设计需要根据具体的组织规模、研发流程和质量要求进行具体的设计和实施。同时，需要根据实际情况不断调整和完善，以确保软件研发过程的质量得到有效控制。

4.2 质量角色定义

质量角色定义是指在一个组织内部明确与软件研发质量相关的各个角色及其职责和权力。以下是质量角色定义的一些要点：

1. 质量经理

质量经理是软件研发质量管理体系的核心角色，负责制定和执行软件研发的质量策略，监督和管理软件研发过程的质量，确保软件产品的质量符合要求。

2. 质量工程师

质量工程师是负责软件研发质量的技术专家，负责软件研发过程中的质量保证和控制，包括缺陷预防、缺陷检测、缺陷分析和修复等。

3. 需求分析工程师

负责收集、分析和管理项目需求，与产品经理和研发团队紧密合作，确保需求的准确性和一致性。

4. 产品经理

产品经理是负责软件研发需求质量的技术专家，确保将客户需求，准确，清晰，完整地转化为组织内部技术需求。典型的质量相关的行动包括产品定义评审，产品走查，产品验收等。

5. 软件架构师

架构设计是软件质量的基石。软件架构师是负责软件产品架构的质量的技术专家，确保软件产品的架构设计的合理性，平衡软件产品成本，产品架构的生命周期，技术实现手段与软件质量属性，如可用性，易用性，安全性等之间的平衡。

6. 开发工程师

开发工程师是软件研发的主要实施者之一，负责按照软件研发的质量要求进行编码和测试，确保软件产品的质量。

7. 测试工程师

测试工程师是负责软件测试的专业人员，负责制定测试计划、设计测试用例、执行测试、反馈问题和跟踪缺陷等。

8. 配置管理员

配置管理员是负责软件配置管理的专业人员，负责配置管理过程，包括版本控制、变更管理、构建和发布管理，确保软件配置的正确性、可追溯性和一致性。

9. 项目经理

项目经理是软件研发项目的管理者，负责协调和管理整个项目的进度和质量，确保项目按照预定的时间和质量要求完成。

需要注意的是，质量角色定义需要根据具体的组织结构和研发流程进行具体的设置和分配。同时，需要根据实际情况不断调整和完善，以确保软件研发质量的全面管理和有效控制。

4.3 质量关系协调

质量关系协调是指在一个组织内部协调与软件研发质量相关的各个关系方之间的关系，以确保软件研发过程的顺利进行和软件质量的提高。以下是质量关系协调的一些要点：

1. 内部关系协调

内部关系协调主要指在组织内部协调各个部门之间的关系，包括开发部门、测试部门、质量管理部门等。质量管理部门可以作为各部门的桥梁和纽带，协调各部门之间的合作和沟通，确保软件研发过程的质量得到全面管理和控制。

2. 外部关系协调

外部关系协调主要指与组织外部的相关方之间的关系协调，包括客户、供应商、第三方机构等。在与客户的协调中，需要明确客户的质量要求和期望，并确保软件研发过程的质量能够满足客户的要求。在与供应商的协调中，需要确保供应商提供的产品和服务符合质量要求，并协调供应商与组织内部的相关方之间的沟通和合作。在与第三方机构的协调中，需要确保第三方机构的评估和审核符合质量要求，并协调第三方机构与组织内部的相关方之间的沟通和合作。

3. 质量问题协调

质量问题协调主要指在软件研发过程中协调解决质量问题的方式和流程，包括缺陷管理、问题跟踪、质量反馈等。在缺陷管理中，需要确保缺陷得到及时报告和反馈，并协调相关方解决和处理缺陷。在问题跟踪中，需要跟踪和监控质量问题，并及时向相关方反馈问题和建议解决方案。在质量反馈中，需要收集和分析质量反馈信息，并协调相关方改进和优化软件研发过程的质量。

需要注意的是，质量关系协调需要根据具体的组织结构和研发流程进行具体的协调和管理。同时，需要根据实际情况不断调整和完善，以确保软件研发过程的顺利进行和软件质量的提高。

4.4 质量管理组织架构

4.4.1 质量管理团队组织方式

质量团队的组织方式通常是建立一个专门的质量管理团队，负责质量管理活动和流程的执行。这个团队通常由一些经验丰富的专业人员组成，他们在质量管理领域具有专业知识和技能，以下是一些常见的质量团队组织方式。

1. 质量管理部门

建立一个独立的质量管理部门，该部门负责整个组织的质量管理活动。这个部门通常由质量经理或质量总监领导，负责质量管理策略的制定、质量目标的设定以及质量管理过程的规划和执行。

2. 组织级质量管理小组

组织内部设立的一个专门的组织级虚拟质量管理小组，通常由质量管理部门牵头，由各相关部门代表组成，他们具备所需质量管理知识和技能，共同制定和推动质量管理策略、指导项目团队进行质量管理活动，以及监督和评估质量绩效。比如按 CMMI 标准设立的 EPG 小组，甚至是为特殊的目的成立的质量问题解决小组都属于此类。

3. 项目级质量负责人

在每个项目中指定一位质量负责人，负责项目级质量管理工作。这些负责人通常是具备质量管理背景和经验的人员，他们负责制定和执行符合组织级质量要求的项目级质量计划、监控项目级质量目标的实现，并协助项目团队进行质量审核和改进活动。项目级质量负责人也有可能虚线汇报给组织级质量管理小组或者质量管理部门中的成员，形成质量管理的层级关系。

4. 组织级质量管理委员会

这是由组织级最高领导者组成的组织级虚拟委员会，该委员会确保：

- 对组织质量管理体系的有效性承担责任；
- 确保制定质量管理体系的质量方针和质量目标，并与组织环境和战略方向相一致；
- 确保质量管理体系要求融入组织的业务过程；
- 促进使用过程方法和基于风险的思维；
- 确保获得质量管理体系所需的资源；
- 沟通有效的质量管理和符合质量管理体系要求的重要性；
- 确保实现质量管理体系的预期结果；
- 促使、指导和支持员工努力提高质量管理体系的有效性；
- 推动改进；
- 支持其他管理者履行其相关领域的职责。

组织级质量管理委员的具体开展形式可能是单独的会，由质量部门最高领导牵头，也有可能与其他最高领导层会议，例如总经理办公会，事业部运营会议等合并。

这些组织方式的选择取决于组织的规模、结构和需求。不同的组织可能采用不同的方式或结合多种方式来建立和组织质量团队，以确保质量管理活动得到有效执行和协调。重要的是要建立明确的角色和责任分工，以确保质量团队能够有效地推动质量管理工作并实现持续的质量改进。

4.4.2 团队角色

不同的质量团队组织方式，其角色也有一定的差异性，以下给出一些角色建议。

1. 质量管理部门

质量经理/质量总监：负责领导和管理质量管理部门，制定质量管理策略和目标，监督

质量管理活动的执行，并与组织高层管理层沟通和协调。

质量管理专员/质量管理工程师：协助质量经理进行质量管理工作，包括制定和更新质量管理计划、指导和培训团队成员，监控和评估质量绩效，收集和分析质量数据等。

2. 质量管理小组

质量小组负责人：领导和协调质量管理小组的工作，包括制定质量管理策略和目标，分配任务，指导团队成员，确保质量管理活动的执行和进展。

质量管理专家：提供专业的质量管理知识和经验，参与制定和改进质量管理过程和方法，指导和培训团队成员，协助质量问题的解决和质量改进活动的推动。

3. 质量管理代表

项目质量管理代表：作为项目团队的一员，负责项目的质量管理工作。制定和执行质量管理计划，监督和检查质量控制活动，收集和报告质量指标，协助质量审核和改进活动，确保项目团队按照质量管理要求进行工作。

功能团队质量管理代表：作为功能团队的一员（实线或者虚线），负责功能团队内部的质量活动，培训辅导组织新人对流程的理解与答疑解惑，监督审核团队成员组织级工作流程符合性，宣传组织质量文化，培养团队成员的质量意识，推动组织和产品质量的持续改进。

4. 质量管理委员会

- 委员会主席：主持质量管理委员会的会议，制定议程，确保会议的有效进行，促进成员之间的合作和沟通；

- 部门代表：来自各个部门的代表，负责将部门的质量管理需求和问题提供给委员会讨论，分享和推广最佳实践，参与质量管理政策和流程的制定，协调解决组织层面的质量问题。

在团队组织方式下，质量团队角色成员有可能存在跨职能合作，目的是共同推动质量管理和质量改进。同时团队角色需要根据组织的具体需求和特点，进一步调整和补充角色和责任分工，以适应项目和组织的实际情况。

第五章

定义符合质量管理体系要求的流程

第五章 定义符合质量管理体系要求的流程

5.1 流程体系设计

设计质量管理流程体系时，需要注意以下实践方面：

1. 流程范围确定

根据组织情况和质量管理需求，确定流程体系需要覆盖的范围，是否包括需求、设计、编码、测试、发布等全部软件生命周期过程，以及需规划的辅助性质量保证流程。

2. 现状流程分析

通过流程绘制、观察、访谈等手段，调研当前的质量管理流程情况，找出存在的问题和改进点。

3. 标杆流程研究

研究行业内优秀公司的质量管理流程模式，借鉴其中可行的元素。同时，参考质量管理标准中对流程的规定。

4. 流程架构设计

根据研究结果，设计流程体系的总体框架和结构，确定关键流程和关联关系。采用流程图或模型描述流程架构。

5. 细节流程设计

在架构基础上，详细设计每个流程的具体内容和步骤、输入与输出、职责分配、资源需求等。制定流程规范文件。

6. 流程评审优化

组织进行流程评审，由流程专家和相关角色负责人参与，提出优化意见。并根据反馈进行迭代优化。

7. 流程支撑系统设计

考虑在流程设计时融入信息系统支持，如质量数据收集和分析系统。提高流程执行效率。

8. 流程改进计划制定

根据设计结果，制定未来流程优化改进和持续改进的具体行动计划。

9. 流程培训计划准备

规划对员工开展流程培训的范围、方式、时间等，确保员工理解并能贯彻新的流程。

5.2 流程管理实施

为了有效实施质量管理流程，需要注意以下实际操作方面：

1. 制定实施计划

针对不同流程，确定实施的范围、阶段、时间表、资源及工作分解，形成详细的实施计划。

2. 组织实施小组

由相关部门负责人及员工组成实施小组，明确小组成员的角色和责任。

3. 开展培训

对涉众员工进行流程培训，确保他们理解新的流程要求。

4. 完善支持系统

配套建设或改进相关的 IT 系统，提供流程执行所需的系统支持。

5. 流程规范发布

通过手册、在线 Wiki 等方式，发布流程规范，便于员工查询。

6. 组织内部分享交流

通过召开流程说明会等方式，引导员工就新流程进行讨论和经验分享。

7. 规范测试运行

在试运行阶段，监督员工按规范执行，记录问题，及时优化。

8. 正式实施

通过部门会议等方式正式宣布启动新流程，并监督落实。

9. 结果评估

实施后适当时间，评估实施效果，是否达到质量和效率提升的预期。

10. 持续改进

根据评估结果，继续优化流程，使之成为组织的一种习惯。

5.3 流程体系优化

在质量管理流程正式实施后，还需要建立常态化的流程优化和改进机制，具体实践包括：

1. 优化机会识别

通过员工建议、客户反馈、质量数据分析、流程评审等方式，识别流程存在的问题点和优化机会。

2. 优化方案提出

对应优化点，由流程管理团队和相关角色提出具体的优化建议方案。方案需要论证优化的意义和效果。

3. 定量和定性效果分析

评估方案的效果影响，包括对质量、效率、成本等的定量和定性影响。明确优化的价值。

4. 管理层评审

由高层管理者对优化方案进行评审，确定是否有必要投入资源进行优化。

5. 优化实施

对获批准的优化方案，制定具体实施计划，安排流程调整和员工培训等工作，将优化落到实处。

6. 效果跟踪验证

优化实施后，监测流程指标变化，以验证优化效果是否达到预期。持续跟踪效果。

7. 标准化和推广

将经验证的优化方案，进一步扩展推广到其他相关流程，并形成标准化的最佳实践。

8. 持续优化文化培育

通过持续的优化实践，培育员工的流程改进意识，使之成为组织文化的一部分。

5.4 典型质量管理体系流程

5.4.1 需求管理

- 明确用户需求和产品功能，并确保需求和功能能够被追踪和满足。
- 设计并维护需求跟踪矩阵和需求变更控制。
- 需求评审内容和需求分工。

1. 需求质量管理的定义

需求质量管理是指通过管理和控制产品需求，确保最终产品或服务能够满足用户的需求和期望。

2. 需求质量管理的重要性

需求质量管理在整个产品开发过程中起着至关重要的作用：

（1） 提高产品质量：通过清晰、明确的需求，确保项目团队在开发过程中准确地实现需求，从而提高产品质量。

（2） 降低成本：有效的需求管理可以避免项目团队在开发过程中出现偏差，确保产品设计和开发的方向正确，从而节省纠正错误的成本。

（3） 减少风险：通过保持需求的一致性和完整性，降低项目中的成本和风险。

（4） 增强市场竞争力：提供高质量的产品可以增加客户的满意度，以及在市场上获得竞争优势。

3. 需求质量管理的流程

需求质量管理流程包括以下步骤：

（1） 需求获取

需求的收集、分析、细化、核实并组织的步骤，并将它编写成文档。这个活动包括了编写项目视图和范围文档、用户群分类、选择用户代表、建立核心队伍、确定使用实例、召开联合会议、分析用户工作流程、确定质量属性、检查问题报告和需求重用等。

（2） 需求分析

根据需求获取中得到的需求文档，分析系统实现方案。这个活动需要完成下面几个任务：

- 绘制关联图，用于定义系统与系统外部实体间的边界和接口的简单模型；
- 创建开发原型，当开发人员或用户不能明确某些需求时，开发一个系统原型，这样使得许多概念和可能发生的事更为直观明了；
- 分析可行性，在允许的成本、性能要求下，分析每项需求实施的可行性，明确每项需求实现相联系的风险，包括与其它需求的冲突，涉及各类用户的利益平衡，对外界因素的依赖和技术障碍；
- 确定需求优先级：分析方法来确定使用实例、系统特性或单项需求实现的优先级别，以优先级为基础确定产品版本将包括哪些特性或哪类需求；
- 为需求建立模型，为需求建立图形分析模型是软件需求规格说明极好的补充说明，可以为系统需求从多个角度建模；
- 编写数据字典，创建数据字典数据字典是对系统用到的所有数据项和结构的定义，以确保开发人员使用统一的数据定义；

- 应用质量功能调配，将系统特性、属性与对客户的重要性联系起来，提供了一种分析方法以明确哪些是客户最为关注的特性。

（3） 编写需求规格说明书

需求开发的最终成果是客户和开发小组对将要开发的产品达成一致协议，这一协议就是通过文档化的需求规格说明书来体现。需求规格说明书包括项目视图和范围文档说明了系统的业务需求，而使用实例文档则说明了用户需求。这个活动需要完成下面几个任务：

- 采用模版：编写软件需求规格说明书等文档定义一种标准模板，该模板为记录系统需求和各种其它与需求相关的重要信息提供了统一的结构；

- 指明需求来源：为了让所有项目风险承担者明白需求规格说明书中为何提供这些功能需求，要能追溯每项需求的来源，来源可能是一种使用实例或其它客户要求，也可能是某项更高层系统需求、业务规范、政府法规、标准或别的外部来源，这些来源应该记录在需求的跟踪能力矩阵中；

- 为每项需求注上标号：为了满足软件需求规格说明的可跟踪性和可修改性的质量标准，必须唯一确定每个软件需求，制定一种惯例来为需求规格说明书中的每项需求提供一个独立的可识别的标号或记号；

- 记录业务规范：指关于系统的操作原则，比如谁能在什么情况下采取什么动作，将这些编写成需求规格说明书中的一个独立部分或一独立的业务规范文档；

- 创建需求跟踪能力矩阵：建立一个矩阵把每项需求来源、定义与实现、测试它的设计和代码部分联系起来，这有利于需求的管理和需求变更影响范围的评估。

（4） 需求验证

需求的验证是为了确保需求说明准确、完整，表达必要的质量特点，需求将要作为系统设计和最终验证的依据，因此一定要保证它的正确性。需求验证务必确保符合完整性、正确性、灵活性、必要性、无二义性、一致性、可跟踪性及可验证性这些良好特征。这个活动需要完成下面几个任务：

- 审查需求文档，对需求文档进行正式审查是保证软件质量的有效的方法。组织一个由不同代表（如用户，分析人员，设计人员，测试人员）组成的小组，对需求规格说明书及相关模型进行仔细地检查；

- 依据需求编写测试用例，根据用户需求所要求的产品特性写出系统的功能测试用例作为系统测试依据；

- 编写用户手册，在需求开发早期即可起草一份用户手册，用它作为需求规格说明的参考并辅助需求分析；

- 确定合格的标准，需求说明中描述什么样的产品才算满足用户的要求和适合用户使用的，将合格的测试建立在使用情景描述或使用实例的基础之上。

（5） 需求管理

需求管理是一种用于组织、控制和文档化需求的系统化方法，也是一种建立和维护用户和开发组织对于改变系统功能的协议。需求开发的结果经验证批准就定义了开发工作的需求基线，这个基线在客户和开发人员之间就构筑了一个需求约定，需求管理包括在项目进展过程中维持需求约定一致性和精确性的活动。现在很多商业化的需求管理工具都能很好的支持需求管理活动。这个活动需要完成下面几个任务：

- 确定变更控制过程，建立需求跟踪矩阵（Requirement Traceability Matrix，RTM）：它可以帮助团队成员跟踪每个需求的状态，以便进行变更影响范围分析和验证。RTM通常以表格或图形形式表示，其中包含了每个需求的标识符、状态、相关任务、实现人员等信息。通过 RTM，团队成员可以快速了解每个需求的相关信息，并及时发现和解决可能出现的问题。通过需求跟踪矩阵，确定一个选择、分析和决策需求变更的过程，所有的需求变更都需遵循此流程；

- 建立软件变更控制委员会（SCCB，Software Change Control Board）：组织一个由项目风险承担者组成的小组作为变更控制委员会，由他们来评估和确定需求变更；

- 进行变更影响分析：评估需求变更对项目进度、资源、工作量和项目范围以及其它需求的影响；

- 跟踪变更影响的产品：当进行某项需求变更时，参照需求跟踪能力矩阵找到相关的其它需求、设计文档、源代码和测试用例，这些相关部分可能也需要修改；

- 建立基准和控制版本：需求文档确定一个基线，这是一致性需求在特定时刻的快照，之后的需求变更就遵循变更控制过程即可；

- 维护变更的历史记录：记录变更需求文档版本的日期以及所做的变更、原因，还包括由谁负责更新和更新的新版本号等情况；

- 跟踪每项需求的状态：这里状态包括“确定”、“已实现”、“暂缓”、“新增”、“变更”等。建立一个数据库，其中每一条记录记录一项需求。

5.4.2 软件设计和编码

在全面质量管理时代，软件研发不仅需要注重结果质量，过程质量同样重要。软件设计与编码阶段是研发过程中十分关键的环节。我们需要制定详尽的软件设计与编码规范，为研发人员提供依据，持续改进研发工作质量，指导团队成员更高效、更高质量地进行研发，最终呈现给用户体验良好、性能优异、稳定性高、安全性高的产品。

本阶段遵循以下原则：

- 预防于未然，培育质量意识，降低缺陷率和维护成本。通过严密的过程控制和规范指引，从源头上消除质量问题；
- 统一标准，提高协作效率。制定详尽的规范和标准，为研发人员设计和编码提供明确指南，最终实现设计和代码的高度一致。这可以显著提高团队协作效率；
- 追求工匠精神，打造精品代码。要求研发人员以工匠的态度对待设计和代码，追求精益求精，生成高质量的精品设计和代码。通过严谨的代码评审与检查，不断优化以产出高质量代码。

1. 软件设计与编码流程管理

本阶段流程图如图 5-1：

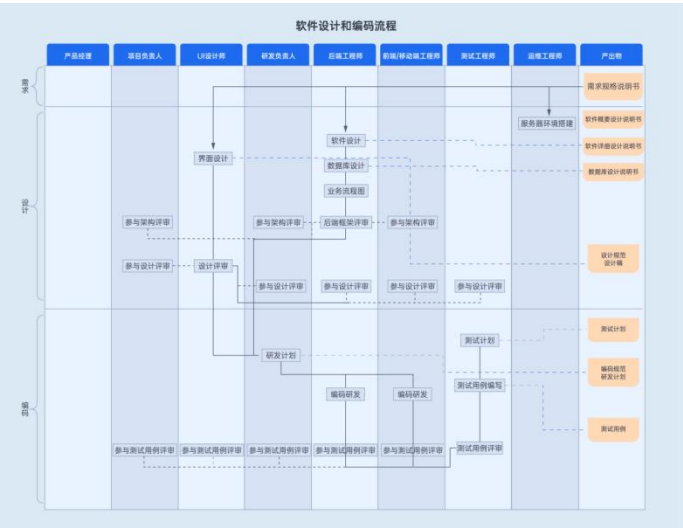


图 5-1 软件设计与编程流程图

（1） 需求获取与变更

在需求管理阶段，软件研发团队与需求提出方共同评估相关需求的技术实现可行性，确定最终需求，形成了原型设计和需求规格说明书。

当需求方提出需求变更请求时，研发团队首先需要评估技术实现的可行性，其次需要考虑对现有产品逻辑和代码架构的影响。对原有产品和代码有重大影响的变更，需要进行全面、彻底的回归测试。每次需求变更需由需求管理人员修改相关文档，并提交研发团队审核确认，最终版本文档由研发团队管控与存档。

（2） 架构设计

基本原则：在设计架构、应用程序或类时，要考虑代码的扩展性，而不仅仅是实现某些基本功能。常用的设计原则包括：SOLID 原则（单一职责原则、开放封闭原则、里氏替换原则、接口隔离原则、依赖倒置原则）。

根据需求，确定所选用的相关技术和框架，例如前端语言及框架、后端语言及框架、数据库、缓存、消息中间件等。依据产品需求规格说明书，设计对应的模块和功能流程，采用适当的设计模式，由研发负责人组织研发工程师制定架构设计图。

（3） 代码开发

搭建初始框架后，按照架构设计图中设计的模块和模式编写相应代码。项目中推荐采用配置管理平台对配置值进行集中管理。

（4） 单元测试

研发过程中，对于常用或核心的模块和算法，应编写相应的单元测试用例。研发人员在开发自身负责的功能模块时，应在交付测试前自行完成测试验证。

（5） 代码重构

研发工程师可对基础模块提出重构申请，或研发负责人在现有代码审核后提出重构要求，评估后安排研发工期。

代码重构的最佳实践可以参考《重构——改善既有代码的设计》一书。该书阐述了许多代码重构的技巧与方法，如提炼函数、内联函数、拆分变量、消除重复代码、简化条件表达式等，助于理解代码重构的具体操作与收益。

2. 代码管理

采用统一的代码管理服务器管理和维护研发项目代码。通常研发过程中，配置多个代码分支，比如开发分支、测试分支、预发布分支、发布分支，可以在开发分支上开发，相关功能在测试分支经过审核和测试确认无误后，合并到预发布分支，产品上线发布时则使用发布分支。

（1） 代码审核机制

1） 上级审核：部门上级对下属提交的代码进行抽查。不符合规范的代码将退回，要求相关人员修改后重新提交。审核时间根据开发时间制定，比如两周一次。审核结果记入代码审核记录表，作为后续绩效评估依据。

2） 代码互查：部门成员之间采取代码互查。对不规范代码可以告知相关人员进行修正，拒不修改的代码可向上级部门报告。互查时间根据开发时间制定，比如一周一次。互查结果记入代码审核记录表，作为后续绩效评估依据。

3） 审核重点：

- 查看代码逻辑是否与需求规格说明书中的功能描述一致；
- 查看代码中是否存在严重影响性能的逻辑（如构建大量对象、死循环等，滥用线程 sleep 等）；
- 边界值判断是否考虑周全（如非空判断）；
- 检查其他常见问题。

（2） 代码注释规范

为了保证相关代码逻辑清晰，方便功能扩展、代码维护和问题修复，代码注释规范如下：

1） 核心逻辑部分：相关类、方法、字段都必须添加相应注释；方法内部代码块必须添加相关逻辑描述文字。

2） 其余部分：尽可能多添加相关代码注释。

（3） 代码命名规范

为确保研发项目整体编码风格的统一，方便团队成员之间的协作和代码维护，应该有组织级的代码命名规范。组织级的代码命名规范一般都是遵循各语言通用的命名规范：如类名通常采用大驼峰命名法，JavaScript、Python、Java 方法名采用小驼峰命名法，C# 方法采用大驼峰命名法等。

具体可以参考以下规范：

- 谷歌代码风格指南；
- Vue 风格指南；
- .NET 设计指南；
- 阿里巴巴移动端开发手册；
- 阿里巴巴 Java 开发手册。

（4） 接口安全规范

为确保单个系统的内部安全，系统对外提供接口时，需遵循以下接口安全规范：

- 接口设计采用参数加密+超时处理+私钥验证+HTTPS 机制实现安全性；
- 对于无需登录验证的公开接口，后台需要做好反爬等限流机制以防止服务器过载；
- 对于包含密码、手机号、身份证等敏感信息的接口，敏感信息应进行加密处理，切勿明文传递。

（5） 接口文档规范

为减少编写文档时间，方便团队成员之间的协作开发，推荐使用 Swagger 在线文档工具。接口文档规范如下：

- 文档应包含接口功能说明、接口名称、请求路径、访问类型、请求参数说明、返回参数说明、返回错误码说明及作者信息等；
- 对涉及加解密相关的接口，密钥应通过内部邮件、U 盘等方式传递而非在接口说明中公开；
- 生产环境接口应屏蔽 Swagger 接口说明。

3. 数据管理

（1）数据管理

采用统一的代码管理服务器管理和维护项目代码。未经部门领导和公司同意，严禁将工作相关代码、资源和数据上传至外部代码仓库、网盘，严禁分享给公司外部人员。违者按违反保密协议进行处理。一般情况下，也不通过聊天工具传输代码。

根据业务流程和规则，采用规范的数据建模工具设计数据库表结构、字段属性和约束条件。通过数据建模可以制定出简洁且符合业务需求的数据库设计。数据库设计直接影响软件的性能、扩展性和可维护性。

制定数据校验机制，如唯一键约束、非空约束、外键约束等，确保数据的准确性、有效性和业务逻辑正确性。并定期对数据库数据进行校验，发现并修复不一致和异常数据。

在系统版本升级或业务需求变更时，可能会涉及到数据的迁移和不同系统的数据集成。制定数据迁移计划和方案，选择合适的工具执行迁移，并在迁移后验证数据的正确性和一致性。对不同系统的数据进行匹配、清洗和汇聚集成，实现跨系统的数据共享和使用。

运维人员根据实际需求，按固定周期（小时或天）备份生产环境数据库。定期备份数据库操作记录等。

（2） 数据库操作管理

禁止在生产数据库上直接通过数据库管理工具或 SQL 语句进行数据的增加、修改和删除等操作。如遇特殊情况，需要提前备份相关数据。操作前，需要找另外的研发人员核对相关 SQL 语句逻辑是否正确。操作完成后，需要核对相应数据，确保没有对范围外数据造成影响。

软件设计与编码是软件研发质量控制的核心部分之一。良好的设计与编码标准可以大大提高软件质量，减少维护成本，促进研发效率。本部分系统地阐释了代码、接口与数据等方面的规范与管理，为软件研发质量过程控制提供指导与依据。研发团队需要在具体项目中执行这些规范，定期评审实施效果，并根据需要进行持续改进。

5.4.3 软件测试

软件测试是指在软件开发过程中对软件系统进行验证和验证的过程。它旨在发现和纠正软件中的错误、缺陷和问题，并确保软件在发布之前达到预期的质量标准。软件测试可以涉及多个方面，包括功能测试、性能测试、安全性测试、兼容性测试等；从测试层次来看可以区分单元测试、集成测试、系统测试、验收测试等。

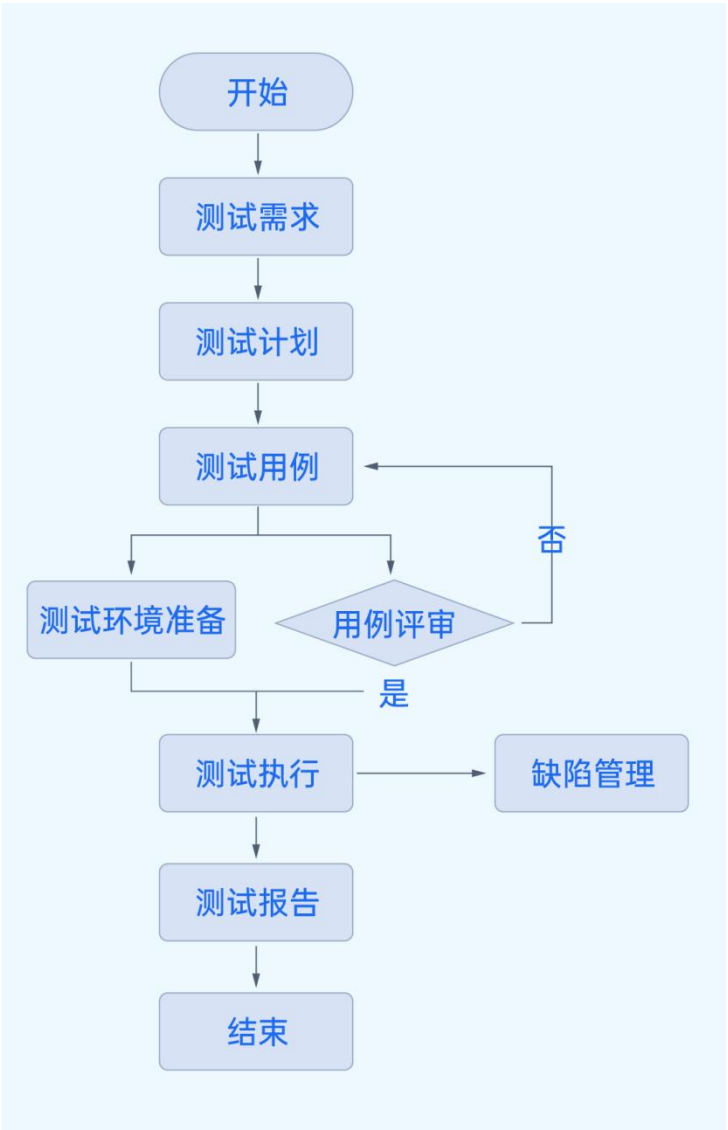


图 5-2 测试流程

1. 测试流程

软件测试生命周期是一个反复的、循环的过程，其目的是防止软件中的错误。它包括测试分析、计划、设计、设置、执行和测试结束等活动。由于软件的复杂性，如果只进行一次测试，不可能保证产品没有错误。因此，在软件测试生命周期的每个阶段都要进行多项测试。

（1） 测试需求分析

这个阶段从测试角度检查功能和非功能需求，以确定可测试的需求。测试团队与客户、解决方案架构师、技术负责人、业务分析师和其他利益相关者与质量保证团队沟通，理解客户的要求，以便根据客户的规格制定测试方案。在本阶段确保对业务需求的正确理解，识别和确定测试需求。

（2） 测试计划

这个阶段根据测试需求，明确测试的目标和范围，选择要进行的测试类型和测试方案；针对测试需求确认和分配角色和责任，以及需要的测试资源和设备。计算测试活动所需的工作量和时间计划表，并与开发人员和产品经理同步。测试计划一般包含测试所需资源，测试工作分工，研发提测节点，回归测试节点等。

（3） 测试用例的编写

测试用例的表现形式直接影响了它的可读性，可维护性。因为一套不易读，冗长繁琐且没有统一规范的测试用例会直接导致测试用例难以阅读和维护；其次还会直接影响到测试执行和结果的正确性。下面总结了三种经典的测试用例编写方法。但是由于不同的团队和不同的项目情况不同，所以没有一种最佳方法适合于所有团队。测试用例编写人员需要根据自己团队和项目的特点和情况，选择适合并且实用的测试用例编写方法，从而更好的维护和执行测试。

方法一：操作和执行步骤

这是一种常规的测试用例编写方法，通过描述操作和执行系统的步骤来描述测试用例。其优点是非常直观，而缺点是过于繁复。如果操作和执行步骤经常更改的情况下，其维护成本就非常高。所以它适合一些比较稳定，业务需求和执行步骤变更较少的项目。下面是两个实例，展示了两个申请新用户的用例。

实例：

用例 1：申请一个新账号成功

Given <理查德>准备申请一个新的账号

When 打开“注册”页面并点击“下一步”

Then “用户信息”页面成功打开

When 完成填写“用户信息”页面并点击“下一步”

Then “审核”页面成功打开

When 用户信息审核完成并点击“下一步”

Then “申请<成功>”页面成功打开

用例 2： 申请一个新账号失败

Given <尼奥>准备申请一个新的账号

When 打开“注册”页面并点击“下一步”

Then “用户信息”页面成功打开

When 完成填写“用户信息”页面并点击“下一步”

Then “审核”页面成功打开

When 用户信息审核完成并点击“下一步”

Then “申请<失败>”页面成功打开

方法二：系统或者用户行为

在行为驱动开发（BDD）出现后，越来越多的测试用例通过系统行为来编写。其优点是相对于方法一，它非常简练。所以在修改和维护时成本较低。但是它有一个前提，就是

测试执行人员必须了解功能细节的前提下，才能通过行为描述来进行测试。所以这种方法写出的测试用例多数用于自动化测试。下面是将方法一种的两个实例通过行为驱动开发重写后的实例。

实例：

```
用例 1： 申请一个新账号成功

    Given <理查德> 准备申请一个新的账号

    When  完成新账号申请流程

    Then  新账号创建<成功>



用例 2： 申请一个新账号失败

    Given <尼奥> 准备申请一个新的账号

    When  完成新账号申请流程

    Then  新账号创建<失败>
```

方法三：用代码思维编写测试用例

方法一和方法二都存在不同程度的重复，不易复用。为了改善或者解决这两个问题，从而进一步增强测试用例的可维护性，降低新增用例和维护已有用例的成本，需要使用代码思维来进行测试用例编写。其核心是尽可能抽象出通用的操作步骤或者系统行为的领域特定语言（DSL），然后通过参数化的方式来进行测试数据传递，从而复用领域特定语言。下面是通过本方法重写方法一和方法二后的实例，它主要体现了众多代码思维中的其中一种，即逻辑与数据分离。

步骤实例：

```
用例集 1： 申请一个新账号

    Given <用户>准备申请一个新的账号
```

```
    When  打开“注册”页面并点击“下一步”

    Then  “用户信息”页面成功打开

    When  完成填写“用户信息”页面并点击“下一步”

    Then  “审核”页面成功打开

    When  用户信息审核完成并点击“下一步”

    Then  “申请<结果>”页面成功打开
数据集：

    | 用户     | 结果  | 描述              |

    | 理查德  | 成功  | 他是一名管理员    |

    | 尼奥     | 失败  | 他是一名黑客      |
```

行为实例：

```
用例集 1：  申请一个新账号

    Given <用户>准备申请一个新的账号

    When  打开“注册”页面并点击“下一步”

    Then  “用户信息”页面成功打开

    When  完成填写“用户信息”页面并点击“下一步”

    Then  “审核”页面成功打开

    When  用户信息审核完成并点击“下一步”

    Then  “申请<结果>”页面成功打开
```

数据集：

用户	结果	描述
理查德	成功	他是一名管理员
尼奥	失败	他是一名黑客

（3） 用例评审

在编写测试用例之后，进行审查和验证。邀请其他团队成员、开发人员或领导者参与审查过程，以确保测试用例的准确性、完整性和可理解性。

（4） 测试环境准备

1） 确定测试环境拓扑结构：根据测试需求和系统架构，确定测试环境的拓扑结构。这涉及到服务器、客户端、数据库、网络设备和其他相关组件的布局和连接方式。

2） 环境配置和安装：根据测试环境需求和拓扑结构，配置和安装必要的软件和组件。这可能包括被测软件版本、操作系统安装、数据库配置、Web 服务器设置、网络连接等。

3） 数据准备和导入：根据测试需求，准备测试所需的数据。这可能涉及创建测试数据集、导入现有数据、生成模拟数据等。确保测试数据的完整性、准确性和合理性。

4） 第三方集成和模拟：如果软件系统需要与第三方系统进行集成，确保第三方系统在测试环境中可用。对于无法直接访问的第三方系统，可能需要使用模拟数据或模拟服务进行集成测试。

5） 环境验证和确认：在测试环境准备完成后，进行环境验证和确认。确保环境的稳定性、可用性和正确配置。执行一些基本测试，例如网络连通性、服务器访问权限、软件安装验证等。

（5） 测试执行

1） 执行测试用例：根据测试计划和测试用例，按照预定的步骤和操作顺序执行测试。记录每个测试用例的执行结果，包括输入、实际输出和预期结果。

2） 记录测试结果：在测试执行过程中，及时记录每个测试用例的执行结果，保存原始

测试日志等相关记录。这包括记录通过的测试用例、失败的测试用例以及遇到的问题和异常情况。

3） 跟踪和管理缺陷：如果在测试执行过程中发现软件缺陷，及时记录并报告给相应的负责人或缺陷跟踪系统。包括准确描述缺陷、提供复现步骤和截图，以便开发团队进行修复，验证缺陷是否修复。

4） 进行回归测试：在所有功能提交测试通过后，执行回归测试，确保验证相关功能是否正常工作，是否引入其他未知缺陷，是否具备版本发布条件。

5） 如果规划了性能或负载测试，执行性能和负载测试：根据需求和测试策略，执行性能测试和负载测试，评估软件系统在不同负载和压力下的性能和稳定性。

（6） 测试报告

测试报告是测试工作的重要成果之一，它提供了测试执行的详细结果和评估，以便项目团队和相关利益相关者了解软件质量和测试覆盖范围。测试报告的一般内容为：报告概述、测试执行数据、缺陷分析数据、测试结果和评估、性能和负载测试结果（如有）、风险评估、总结和建议，附录和支持文档（测试用例、测试数据、缺陷详情、环境配置）。

2. 测试方法与工具

（1） 单元测试

单元测试是一种集中测试最小单元（通常是函数、方法或模块）、独立性和隔离性强、功能验证和正确性验证的测试方法；由开发人员执行，旨在提高软件质量、减少错误和缺陷，并支持软件开发过程的可靠性和可维护性。常用的单元测试方法包括：白盒测试、边界值测试、参数测试；单元测试常采用自动化方式进行。

（2） 集成测试

集成测试是验证多个组件或模块在整体上正确集成和协同工作的测试方法，以确保系统的功能和接口符合预期，并检测潜在的问题和缺陷。涵盖系统的主要功能和关键路径，验证系统与外部系统或服务的集成，如数据库、网络服务等。它是软件开发过程中的重要环节，帮助确保系统的稳定性、可靠性和互操作性。常用的集成测试方法包括：自顶向下集成测试、自底向上集成测试、并发集成测试、Big Bang 集成测试、逐步集成测试等。

（3） 系统测试

系统测试的目标是测试整个软件系统，包括其各个组件、模块和子系统之间的集成和协作。它着重于验证系统的功能、用户界面、性能、安全性、可靠性、兼容性等方面，以确保系统在各种使用场景和负载条件下的正常运行。系统测试可以分为以下几个方面：

1） 功能测试：验证系统的各项功能是否符合规格和需求，包括核心功能、辅助功能和特定的用户场景。

2） 性能测试：评估系统在不同负载条件下的性能和响应能力，包括吞吐量、响应时间、资源利用率等指标。

3） 安全性测试：检查系统的安全性和保护机制，验证系统对潜在威胁的防御能力，如授权验证、数据加密、漏洞扫描等。

4） 兼容性测试：测试系统在不同操作系统、浏览器、设备和网络环境下的兼容性和可用性。通常手动在不同操作系统、浏览器和设备上进行测试。

5） 可靠性测试：验证系统的稳定性和可靠性，包括错误恢复、容错机制、日志记录等。

6） 用户界面测试：评估系统的用户界面的易用性、可访问性和一致性，确保用户能够轻松使用系统并满足其期望。

（4） 验收测试

验收测试的目标是确保软件系统满足业务需求，具备所定义的功能、性能和质量要求。它着重于从最终用户的角度对系统进行测试，以确定系统是否满足用户的实际使用需求和预期效果。验收测试通常由最终用户、客户或业务代表来执行，他们代表了最终使用软件的利益相关者。在验收测试过程中，用户执行一系列测试用例或场景，模拟真实业务操作，验证软件的功能、界面、性能、安全性等方面是否符合预期。

3. 测试用例管理

（1） 介绍

在软件测试工作中，测试用例是其最为重要的基础。一个良好的测试用例可以帮助测试人员更容易阅读、理解、修改并管理它，从而提高测试工作的质量和效率。

要编写好的测试用例，首先需要对业务需求和验收标准进行深入的分析，并确定业务需求和验收条件的正确性和合理性。然后对其进行测试分析，并完成整体测试用例的设计

和编写，其中包括功能测试用例，端到端测试用例，异常测试用例等等。在分析和设计用例的过程中，可以通过启发式测试策略模型（HTSM）这类方法来进行分析，并通过等价类、边界值、决策表、PairWise 等方法来设计测试用例。

其中的难点是如何让测试用例尽可能覆盖到验收标准，从而完成验收功能的高覆盖测试率。并且还要尽可能找到业务需求、技术架构等系统相关的各种限制，通过分析限制可以得到更多的测试用例，包含异常测试用例。

对于设计好的测试用例需要进行分类并管理，然后根据不同的分类进行分层测试。通常情况下可以将测试分为端到端测试，功能测试，集成测试，单元测试等。根据这个分类方法，可以方便进行测试分层管理，就是某些测试用例放在端到端测试类型里面，而有些测试用例则放到集成测试类型里面。

而根据测试用途还可以将某些类型的测试分类成回归测试、验收测试、健全测试和冒烟测试等。由于一个测试用例可能既属于回归测试，又属于冒烟测试，所以这种情况下就需要一个良好的测试管理系统或者管理方法来对大量的分类后的测试用例进行管理。

编写和管理测试用例是测试用例工作中工作量最大、最为繁琐的部分。其质量的高低直接影响到测试工作是不是能高效和顺利地进行并完成。所以结合产品的类型和团队的情况，选择适合自己团队的用例编写和管理方式，从而事半功倍。

（2） 测试用例分类

测试用例在来源方面可以分为两大类，即验收测试用例和探索测试用例；在执行方式方面可以分为手动测试用例和自动化测试用例；在用途方面可以分为功能测试用例和非功能测试用例。但是对测试用例进行管理，需要对不同类型的测试用例进行系统化的管理。而对于一个软件系统的测试用例体系化，需要注意以下几个重点：

1） 测试用例分类的三个难点

难点一，明确清晰的验收条件。

验收测试用例的核心就是验收条件，对于明确清晰并且细化到足够程度的验收条件，可以直接转换为或者作为测试用例来使用。但是往往很多情况下验收条件没有细化，甚至不够明确和清晰，存在歧义，从而导致很难设计出足够的用例来映射到所有验收条件，称之为 AC Mapping。

难点二，功能测试覆盖率。

对于整个软件系统来讲，由于很多验收条件都是一些分散的点，而如何通过分析业务需求和验收条件等来设计测试用例，保证测试用例对于系统功能或者端到端功能场景的全覆盖是一个难点。

难点三，功能和系统限制。

它的核心是如何通过分析验收条件、业务流程、技术架构等信息，发现某个功能或者系统级别的限制，只要突破这个它们即开始在做探索性测试，它们也可以叫探索性测试用例。无论验收测试用例还是探索性测试用例，都可以使用各种基础的测试用例方法来分析和设计测试用例，比如等价类、因果图、错误推测法或者场景法等等。所以一般情况下，可执行的功能测试用例全集等于验收测试用例与探索式测试用例之和。

2） 测试用例分类的一些注意事项

对于手动测试用例和自动化测试用例，它们只是执行方式的不同。并且大部分情况下，自动化测试用例是从手动测试用例中选出来的一部分子集。

除了必须执行的功能测试用例以外，一个软件系统还存在大量的非功能测试用例，比如安全测试、性能测试、易用性测试、兼容性测试等等。

对于验收测试用例和探索式测试用例，以及手动测试用例和自动化测试用例，都应该进行统一化管理；对于非功能测试，一般需要与功能测试用例分开进行管理。而对于测试用例本身，需要使用代码化的思维来进行编写、维护和管理，其目标就是易读、易维护、易执行和易管理。为了达到这个目标，还需要克服以下测试用例的难点：自然语言的多样性、数量大、自动和手动难统一。大部分情况下的测试用例都会使用自然语言进行描述。而自然语言的多样性和歧义，可能会让用例设计和编写者以外的手动测试执行人员或者自动化测试开发人员产生误解，从而理解错测试用例，并得到错误的测试过程和结果。其次当测试用例的数量增大以后，比如几千上万的时候，维护和管理的问题也随之困难起来，其中还包括如何统一管理自动化测试和手动测试用例，从而减少对于重叠测试用例的维护和管理。为了解决这些问题，需要用多一些方法，其中包括使用代码式思维来编写测试用例，比如用 DSL 语言、测试步骤和测试数据分离等，其次还需要强大的测试用例管理系统等等。

4. 测试用例分类与管理

编写和管理测试用例一直是一件十分繁琐又很难降低成本的工作，为了尽可能降低其成本，测试用例需要具有以下特性：易阅读、易维护、易执行、易管理。而难点也比较突出，其中包括语言的歧义性和多样性导致的不易阅读和理解；手动测试和自动化测试用例很难统一管理和统一执行；当测试数量很大的时候，如果测试用例管理系统不易用，测试用例的复用性也不高，则会导致测试用例不易维护，从而会极大的增加了其管理成

本。所以测试用例的编写需要用到代码思维，比如高内聚、低耦合、易复用、清晰的模块划分等。再加上自动和手动测试用例的统一管理系统，则可以使测试管理更加容易。

（1） 测试用例的管理方法

测试用例管理是一项繁琐的工作，所以需要明确的管理流程，下面是一个经典的测试用例管理流程（如图 5-3），项目可以根据自己的特定和需求对这个流程进行修改和定制。

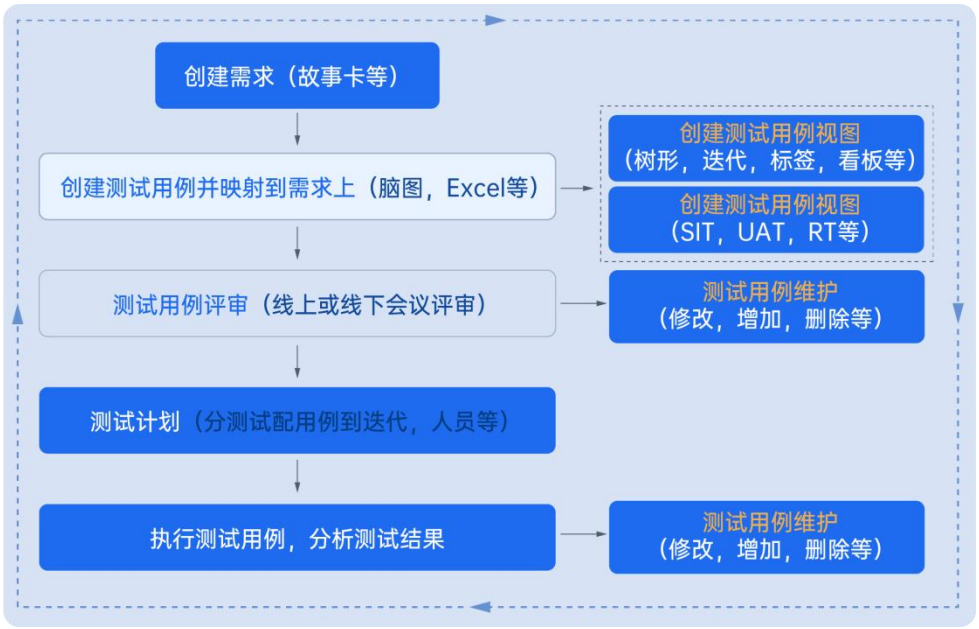
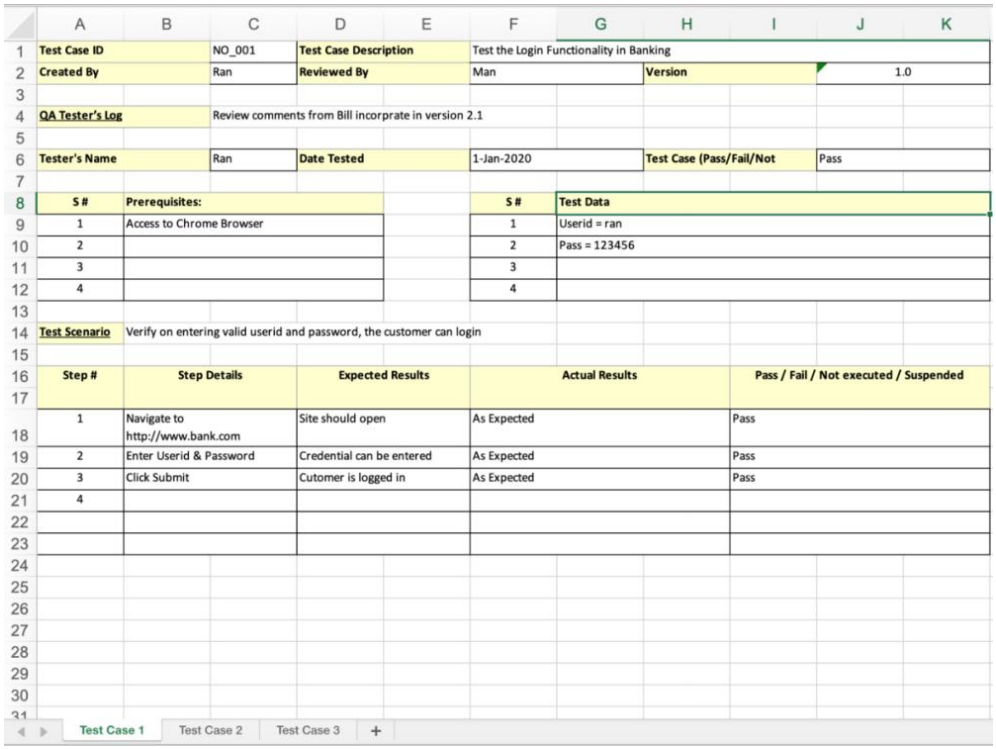


图 5-3 测试用例管理流程

其次测试用例有四种经典管理方法方法，分别是文件管理、系统管理、代码活文档和系统活文档。与编写用例一样，没有一种用例管理方法是银弹，适合所有不同的团队和不同的项目。所以了解它们的特点，再根据自己团队和项目的实际情况，选择适合的才是最佳实践。

方法一：文件管理

本方法是中小型项目中比较常见的测试用例管理方法。其优势是简单易用，而劣势是需要自己对测试用例模版进行定制，并且当测试用例过多的时候管理成本会急剧增加。其次对于本地文件模式，则很难让多人进行协作编写，但是在线文档没有这个问题。下面是一个 Excel 实例，如图 5-4：



	A	B	C	D	E	F	G	H	I	J	K
1	Test Case ID		NO_001	Test Case Description		Test the Login Functionality in Banking					
2	Created By		Ran	Reviewed By		Man		Version		1.0	
3											
4	QA Tester's Log		Review comments from Bill incorprate in version 2.1								
5											
6	Tester's Name		Ran	Date Tested		1-Jan-2020		Test Case (Pass/Fail/Not		Pass	
7											
8	S #	Prerequisites:				S #	Test Data				
9	1	Access to Chrome Browser				1	Userid = ran				
10	2					2	Pass = 123456				
11	3					3					
12	4					4					
13											
14	Test Scenario	Verify on entering valid userid and password, the customer can login									
15											
16	Step #	Step Details		Expected Results		Actual Results			Pass / Fail / Not executed / Suspended		
17											
18	1	Navigate to http://www.bank.com		Site should open		As Expected			Pass		
19	2	Enter Userid & Password		Credential can be entered		As Expected			Pass		
20	3	Click Submit		Cutomer is logged in		As Expected			Pass		
21	4										
22											
23											
24											
25											
26											
27											
28											
29											
30											
31											

Test Case 1 | Test Case 2 | Test Case 3 | +

图 5-4 Excel 管理实例图

方法二：系统管理

此类方法一般是中大型项目中最为常用的管理方法，其优势是管理系统提供了强大的管理和协作功能，如协作编写用例、协作执行用例、测试步骤管理、截图管理、测试迭代管理，以及丰富的测试用例和测试结果报表等。因此，它有一定的学习曲线，并且基本上都是界面操作，相对比较烦琐，有些修改很难跟踪，如测试步骤、测试数据的更改等。另外，这种系统一般需要一个独立服务器来部署和运行，相对成本比较高。

方法三：代码活文档，自动化测试框架和代码版本工具

本方法适合于有足够软件技术工程实践的团队和个人，因为它需要使用到代码版本管理工具、集成开发环境（IDE）、自动化测试框架、持续流水线等实践才能高效的编写、维护、执行、管理测试用例、测试日志和测试结果。本方法的优势是可以同时管理自动化测试用例和手动测试用例，并且更容易跟踪测试用例和测试数据的更改。而劣势是需要测试工程师有足够的工程技术能力来实现。

方法四：系统活文档，方法二结合方法三

本方法是将代码活文档和系统管理结合，通过测试管理系统编写和管理测试用例，然后

会自动生成代码模式的测试用例。也可以只编写代码模式的测试用例，然后自动同步到测试管理文档中。自动化测试在持续集成流水线执行，通过流水线进行展示并同步到测试管理系统中。手动测试人员执行了手动测试后，将测试结果通过测试管理系统或者在测试代码中进行记录，并最终汇总到测试管理系统的进行统一展示，从而实现了让不同人员可以一起协作分析、设计、管理和执行测试用例的工作。下面是本方法的架构设计图（图 5-5、图 5-6）。

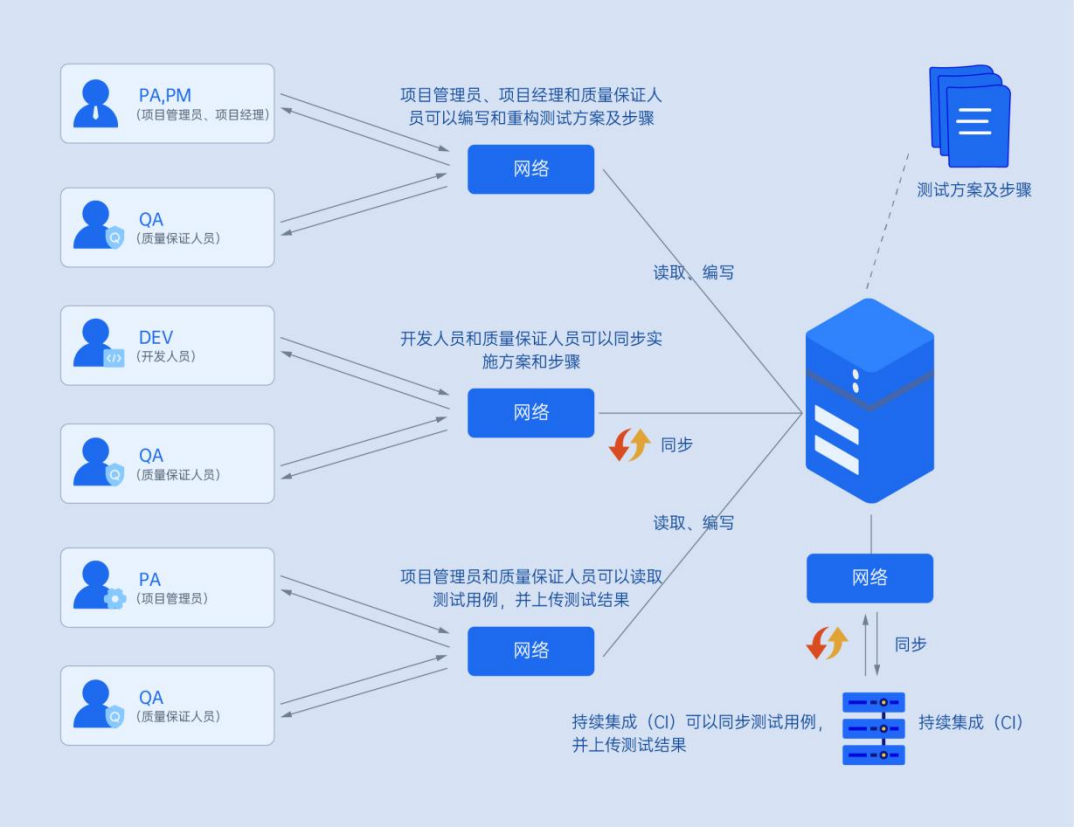


图 5-5 系统活文档架构图 1

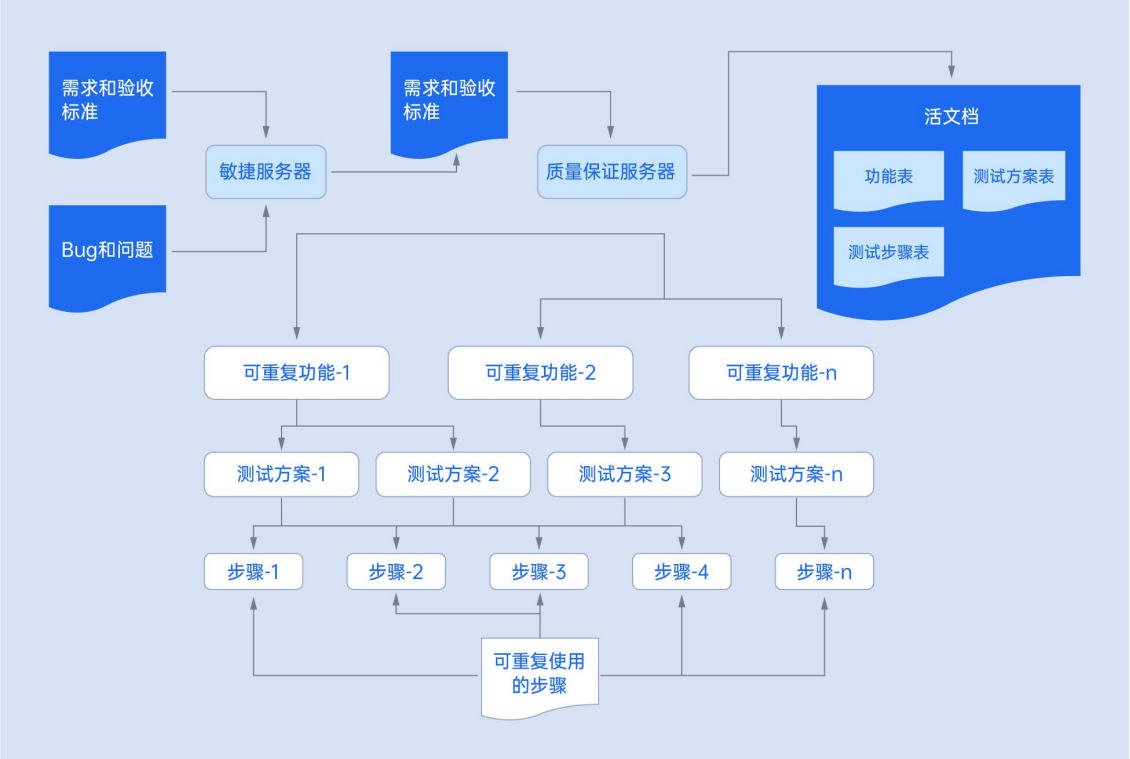


图 5-6 系统活文档架构图 2

5. 总结

测试用例是测试工作的根本，不管是手动测试还是自动化测试的成功，都十分依赖于测试用例的质量。但是只有充分的做好测试分析、设计、编写和管理才能产出一套合格甚至优秀的测试用例套件。从保证测试工作可以高效正确的进行，为产出高质量软件保驾护航。

5.4.4 缺陷管理

缺陷的定义可以追溯到 IEEE729-1983 中的描述：“从产品内部看，缺陷是软件产品开发或维护过程中存在的错误、毛病等各种问题；从产品外部看，缺陷是系统所需要实现的某种功能的失效或违背。”缺陷是产品与规定要求不相符的部分，会存在于软件产品的整个生命周期中，通过测试活动及早发现软件系统中的缺陷，并确保缺陷被有效标识、跟踪、和修改，保证软件系统能够达到要求的质量。

1. 缺陷的生命周期

缺陷的生命周期就是一个缺陷从新建到关闭的过程，这是一个典型过程（如图 5-7）。在实践应用过程中由于不同的测试管理平台，不同团队的内部实践都会有些细微调整。

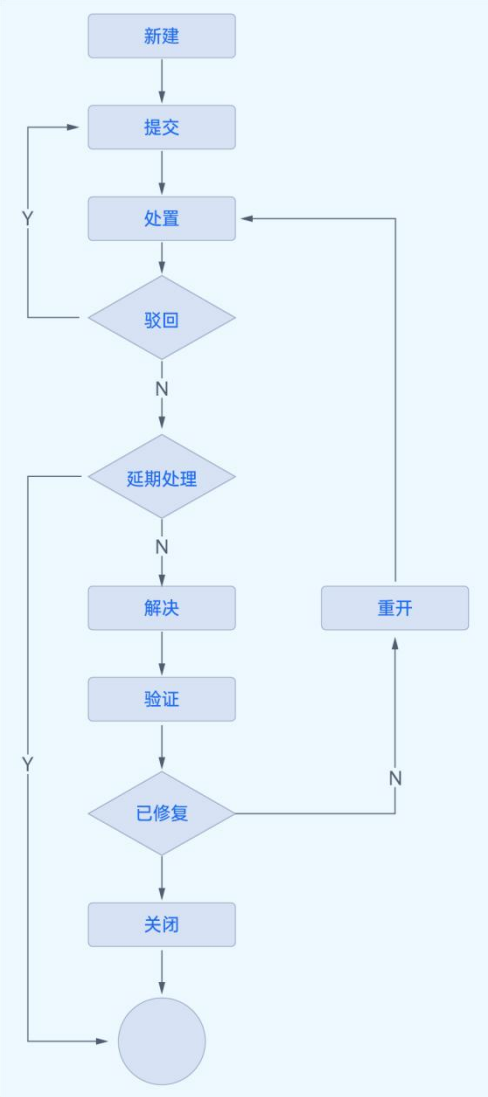


图 5-7 缺陷的生命周期

（1） 新建

缺陷发现后，由测试工程师或者其他发现缺陷的人员新建缺陷（推荐采用有缺陷管理功

能的系统来维护，也可以用电子表格维护）。缺陷新建后，提交前可以反复编辑，补充缺陷记录的信息。新建缺陷描述的要求为分类准确、叙述简洁、步骤清楚、有实例、可再现、复杂问题有据可查（截图或上传附件的形式），具体要求为：

- 单一：尽量一个报告只针对一个软件缺陷；
- 简洁：每个步骤的描述应简洁明了；
- 再现：描述重现的步骤和条件，比如具体输入参数值，以便进行回归验证。应提供截图；
- 期望结果：有期望结果描述；
- 实际结果：有实际结果描述；
- 其它信息，可依实际情况增加。

（2） 提交

测试工程师确认缺陷已经表述清楚，可以提交缺陷。提交后的缺陷状态时“已提交”。缺陷提交前必须分配一个具体的缺陷处理负责，如果测试工程师不确定谁负责，可以把缺陷分配给开发负责人或者项目负责人，由开发负责人重新分配责任人。

（3） 处置

开发工程师确认缺陷是自己负责后，开始着手处理，并修改缺陷的状态为“打开”，表示缺陷正在处理中。如果开发工程师无法复现或者认为测试工程师的描述不是一个缺陷，可以将缺陷状态改成已驳回，这样缺陷就退回给测试工程师进行再次的验证或丰富的辅助信息描述，再重新进入对应的流程。如果不需要驳回，开发工程师就需要确定是否在当前迭代完成修复，这个一般都是基于缺陷的修复难度、缺陷的影响范围等经团队负责人、需求负责人商讨后做出的判断，如果需要延期修复那么将缺陷状态修改成“延期处理”，缺陷生命周期处理结束，等待下次进入解决阶段后，进入后续处理流程。否则开发工程师完成对应缺陷的修复，在对缺陷处置完成后，需做处置记录：

- 原因：说明缺陷产生的原因，比如：设计考虑不周，边界处理不严密，逻辑判断不合理。要求描述具体简洁，以便总结经验；
- 解决方法：修改稿涉及的文件、源代码、配置、脚本等；
- 概括：缺陷是否可能存在于其他位置，或引起其他问题。

已打开的缺陷也可以修改负责人。

（4） 解决

问题解决后，填写解决处理记录，写明造成缺陷的原因和解决方案，改变缺陷状态为“已解决”。

如果开发工程师发现如下情况，可以把缺陷驳回给测试工程师：

- 缺陷不可再现；
- 与先前登记的缺陷重复；
- 不是缺陷，是测试人员理解错误；
- 缺陷轻微，且修改困难、或修改易导致更大的潜在问题。

如果按照开发计划，缺陷发生的功能不属于当前开发阶段必须完成的（需与项目负责人确认）。

（5） 验证

测试工程师对“已解决”状态的缺陷进行重新测试，测试步骤应当按照等级的可重现步骤进行。

（6） 关闭

测试工程师确认缺陷已经解决后，关闭缺陷。对于被开发工程师驳回的缺陷，测试人员需和项目负责人讨论，项目负责人同意的可以关闭，否则需驳回给开发人员。

（7） 重开

验证测试不通过的缺陷，应当驳回给开发人员，状态为“重新打开”。关闭了的缺陷再次出现时（通常因为解决缺陷的方法导致相同位置出现不同形式的缺陷时），测试人员重新打开缺陷，开发人员需要继续解决。

2. 缺陷的编写规范

缺陷描述的清楚与否，可以很好的帮助开发工程师快速定位、解决问题，而且还可以提

高测试工程师基本测试技能。因此，建立标准的缺陷描述规范是十分重要、也是十分必要的。

首先清晰的缺陷描述可以帮助开发人员快速定位、解决问题。软件测试部门中员工的水平各有不一，对于缺陷的认知、描述侧重面也会存在不同。因此，如同一个问题，由不同测试工程师描述缺陷，就有可能会存在描述不一致的问题。这就会造成让开发工程师理解不清晰，从而延误解决问题的周期。

其次标准的缺陷描述可以提高测试工程师的基本测试技能。如有新入职员工，他可以先从缺陷中查找缺陷了解公司产品的整个开发、研制中产生的问题。而标准清晰的缺陷描述可方便快速的使其尽早、尽快地融入测试部门。另外，对于缺陷的追踪验证时，由于是不同测试工程师进行验证，所以规范的缺陷描述，可以提高测试工程师验证问题的效率。

编写一个缺陷需要的必要信息有标题、详情、属性。其中标题是用简洁的话描述该缺陷，主要是让开发知道这是一个什么样的缺陷，主要从功能操作和缺陷现象上描述。详情描述便于开发重现和定位问题，需要描述在什么样的测试环境中，通过什么测试数据，用那几个步骤发现的缺陷，并且要针对缺陷站在测试工程师的角度描述这是一个什么样的缺陷，和预期结果产生了什么样的偏差，过程中尽量提供截图、视频等方式给开发直观的缺陷表述，截图或者录制视频要进来的提供全页面的图像，避免局部截图；属性重点包含验证程度、状态、优先级、提交人等等信息。

表 5-8 缺陷的严重程度及说明

严重程度	标示	含义	例子
致命	1	导致软件无法使用问题，例如整个程序崩溃，导致无法使用，测试阻塞。 1. 问题会自发的影响整个系统。 2. 用户使用正常的操作步骤，就会影响整个系统提供的服务。 3. 具有操作先后顺序的功能，已开始的步骤出现故障，导致后续步骤无法使用。	1. 系统无法访问，崩溃 2. 死循环 3. 数据库死锁 4. 因错误操作导致的程序中断 5. 数据库连接异常

			6. 主要功能缺失
严重	2	某个功能未实现或导致一个特性或导致一个特性不能运行并且没有替代方案	1. 功能与需求不相符 2. 系统接口设计错误 3. 数据库表、业务规则、缺省值没有完整性约束等 4. 主要功能错误
一般	3	错误导致了一个特性不能运行但可有一个替代方案。 功能特征设计不符合系统的需求，不影响系统的业务，并且有相应的补救方法。	1. 操作界面问题 2. 格式、显示问题 3. 删除没有二次提醒 4. 数据库大量空字段
建议	4	建设性的意见或建议。 需求文档没有规定的特性，如果实现会对系统功能或易用性有所提高。	1. 辅助性说明不清晰 2. 提醒交互信息不明确 3. UI 设计不符合规范

表 5-9 缺陷的状态及说明

缺陷状态	描述
初始状态	测试或开发人员提交一个新的缺陷，等待开发人员或项目经理分配

	修改负责人
确认	已经确定是 BUG，但是还没开始修改
激活	确认了 BUG，并开始修改 bug
已解决	缺陷已被开发人员修复，等待测试人员验证
关闭	测试人员验证已修复的缺陷

表 5-10 缺陷优先级

优先级	标示	含义
立即解决	P0	如果故障妨碍开发人员的进一步开发活动，应立即修复。如果阻塞测试，应立即修复。
高度重视	P1	必须修改，版本发布前必须修正
正常处理	P2	必须修改，不一定马上修改，但需确定在某个特定版本发布前必须修正
低优先级	P3	如果时间允许应该修改

5.4.5 配置管理

配置管理是贯穿与整个软件生命周期，为软件研发提供了一套管理办法与活动原则，它是一种 IT 管理流程，用于跟踪 IT 系统的各个配置项目，通过使用配置识别、配置控制、配置状态记录与报告、配置审计等手段，建立并维护工作产品的完整性，以此为所有过程域提供支持。配置管理是一个系统工程流程，用于跟踪和监视对软件系统配置元数据的更改，它强调了对项目的所有的相关产物及其之间的关系都要进行有效管理，从而实现管理项目中的变化，实现不同角色之间的高效协同，并且实现全部的制品过程的可追溯、可审计，从而达到持续的又快又好的交付的目标。配置管理可以提炼成三个方面的内容分别是版本控制、变更控制和过程支持。在应用层面上，配置管理主要包含了代码和制品的配置管理、环境的配置管理和应用的配置管理。

- 代码和制品的配置管理主要包含了组织采用的分支策略，使用的版本管理系统，管理制品的制品库的选择以及对于二方、三方包的依赖管理；
- 环境的配置管理主要是针对应用对外提供服务需要依赖的软硬件、外部系统级依赖等的管理；
- 应用的配置管理是对应用中的一些环境配置、服务配置、数据库配置等。

1. 配置管理及其质量要素

（1） 代码和制品的配置管理

系统的变更从代码变更开始，那么针对代码的版本控制主要依托于某种版本控制系统。这里特别需要注意的是，版本控制中所说的代码并不仅仅是业务实现代码，还包含了数据库变更代码、测试代码、构建脚本、部署脚本等。先定好一个版本控制系统后，就需要决定团队采用的分支模型，管理变更交付过程中代码仓库中各个分支的关系。针对不同的团队、不同的系统以及不同的交付模式，都应该有最合适的一个分支模型，并没有一个公认的最好的分支模型，“因地制宜”才是最优解。在分支模型的实施过程中，需要将静态代码扫码、单元测试等质量保证活动的结果加入到分支合并的一些质量门禁当中，从而防止代码腐化。

从版本控制的代码，经历了构建、测试最后交付到了制品库，完成制品的版本管理。从上面的描述中可以看出制品就是一系列的构建产物（例如二进制包等），制品更加推荐使用制品库管理系统进行管理。在现在制品晋级的理念之下，一次构建多个环境部署的方式越来越收到业界认可，制品库除了管理团队自我构建的包以外，还可以用来管理项目依赖的二方或者三方包。无论是哪一种制品，都推荐有版本控制和语意化版本命名方

式。语意化版本其实是为了能够一样就能看出来版本的变更内容做的一个版本上的定义，一般我们把版本定义为 X1.X2.X3 的三段位，其中 X1 是主版本号，当做了不兼容的修改的时候，才会升级改版本，X2 是次版本号，当做了向下兼容的功能性新增的时候升级这一位，X3 是修订号，当做了向下兼容的问题修正的时候，升级这个版本号。对于制品库中的制品，通常可以使用二进制的一些扫码工具，针对二进制包的安全性进行保证。

（2） 环境配置管理

环境配置管理伴随着软件规模的不断增大，持续交付广泛实施变得越来越重要。交付在当前工程实践领域并不再单独交付一个系统，而是交付一个可以对外提供服务的应用，这其中就包含了应用对外提供服务的硬件、操作系统、中间件、数据库等一系列基础设施。通过一些工具可以很方便的定义在那个环境中安装什么软件，启动那个服务，使用那个配置，测试过程需要保证在这些环境管理工具上的环境配置是幂等的、有效的、安全的。

（3） 应用配置管理

应用配置管理能够将一切应用程序的运行依赖参数标准化，并可以注入部署包中的服务。应用通过配置的变更管理、推送等功能实现了一次构建多次运行，通过集中管理所有应用环境中的配置，降低分布式系统中管理配置的成本，并降低因错误的配置变更造成可用性下降甚至发生故障的风险。测试工程师要验证应用配置的可用性、有效性，保证对应环境的对应配置是正确的和起效果的。

2. 配置管理的质量约束

虽然有格式各样的系统保证配置管理的过程，降低了操作成本提高了自动化程度，但是很多生产故障都和配置参数错误有关，因此对配置参数的变更需要有严格的控制过程和验证流程。配置参数的验证往往是伴随制品晋级的过程进行验证的，也是通过开发、测试、用户验收后才能进入生产环境。同时在变更过程中需要加入人工评审环境，变更的多重保证机制的存在可以从不同角度保证交付的质量，尤其是针对通过某一种开关机制可以控制的配置，其多种控制方式、多种参数参数的交叉验证更应该是不能忽略的配置管理中的质量验证点。

针对一次构建，多次部署的应用，不同环境的配置文件需要严格验证，尤其是程序内部需要调取后才能运行的参数，要充分验证不同环境对应参数的有效性和正确性，对于参数配置和应用服务的先后升级关系一定要在团队内达成共识，建立统一约束，统一操作，避免配置文件更新程序读取老配置的常见问题。配置文件验证中最重要的一个环境就密钥、密码的保密性设置，常规是通过配置文件的人工评审来保证的。

5.4.6 发布管理

1. 发布条件

软件发布前需要满足一定条件，才允许启动发布流程，其中有这几方面报告要求。

- 功能要求：是否进行充分测试活动，是否满足项目期初期制定目标；
- 性能需求：是否进行性能测试，评估性能指标是否实现；
- 安全性需求：项目实现规范性是否满足，对于安全性指标实现情况；
- 易用性及 UI 需求：需经过特定角色验证并达标。

发布项目文件、程序等需经过评审会评审通过，其中参与角色包括但不仅限于技术部门、测试部门、产品部门、业务部门等。

2. 发布前准备

一个项目或版本发布需要有相关的配套文档作为支撑。其中包括：发布说明、产品白皮书、产品说明书、技术说明书、对内培训文档、对外培训文档、用户操作手册、安装说明书、运维手册等。

- 程序版本：确定版本号并进行程序打包；对原程序进行备份；对上线过程进行模拟演练；
- 数据准备：对与历史版本和数据备份、上线验证数据准备、数据预埋；
- 风险管理：应急预案（包含版本回滚）、上线风险点评估；
- 场景设定：针对不同应用场景设定不同的验证方式和验证角色及时间。

3. 发布过程管理

- 版本号管理：新版本版本号管理；

- 验证管理：按照上线文档进行分布部署并分步对发布产品进行验证，验证过程需区分哪些功能由什么角色验证、什么时间段验证、预埋数据和验证时间；对于验证过程中出现的问题需要按照应急预案进行处理，并将问题及处理方式和结果进行记录；

- 过程文档记录：对于发布的程序进行了哪些验证；对于上线测试用例执行了哪些以及结果如何进行记录；对于无法验证的部分进行归类并阐明原因，最终形成产品上线验证报告。

4. 发布后管理

- 版本管理：新版本进行备份；

- 验证跟踪：针对产品特点，在产品发布后，有些功能需长时间跟踪验证，并对结果进行收集、反馈和处理分析；尤其对于上线验证报告中的问题，需重点进行跟进，并及时和相关人员同步处理状态和结果；

- 问题收集处理：产品发布后设定问题收集机制，经收集到的问题区分功能性、非功能性、优化性等进行问题归类，按照线上问题处理机制做出不同的处理决策，并且问题及处理结果，可用于整体项目产品发布质量的考核指标之一，也可用于回顾会上来进行全面深度分析，从而更好的提升团队产品产出质量；

- 运维：产品发布后需要运维长期的跟踪支持，以提升产品口碑，同时不断地进行优化处理提升产品质量。需要格外注重运维服务质量和效率。

5.4.7 回顾与总结

软件质量管理体系的一个重要环节就是回顾与总结，通过回顾与总结对过去一段时间内的软件工作进行总结和反思，以评估软件的质量，找出质量保证过程中的问题和不足，并通过建立改进计划、技术改进需求完成团队的自我成长。

1. 迭代回顾会中质量回顾

在项目的迭代过程中，迭代中的质量回顾和总结是在迭代回顾会中完成的，通过总结本次迭代中质量保证相关的经验、教训并进行改善，从而提高团队的交付质量。迭代回顾会中的针对质量的回顾和总结部分是为了达到如下的一些目的：

- 评估上一个迭代中的质量保证相关的流程、活动、工具方面问题和优秀实践；

- 对发现的问题进行分析，指定改进计划；

- 对优秀实践进行总结，留存到知识库。

迭代回顾会上针对质量保证的回顾可以推动内建质量的文化，提高团队的质量意识。因此在团队针对质量的回顾过程中，大家要避免相互指责、推脱责任，而是要建立一种相互支持、相互鼓励的氛围。

迭代回顾会从名字中就可以看出每个迭代结束后都会进行的一个环节，因此在回顾会的形式上、相对时间上、参会人员上以及内容环节上是相对固定并且一致的。在参会的人员上比较提倡和项目制品过程相关的所有人都要参加，有一名主持人主持会议，主持人可以是任意角色，不需要固定，但是要有轮值计划，让下一次迭代会的主持人事先就知道自己未来的角色。迭代回顾会应该在迭代完成后尽快召开，在回顾会上鼓励团队成员对质量保证环境、测试流程等任何和交付质量相关的活动进行回顾，提出问题甚至给出改进意见。对于测试工程师，应该针对本次迭代的制品过程中的质量和生产质量做出整体的回顾，针对问题进行总结，并给出改进计划，从而落实 PDCA。团队的负责人则应该关注交付过程中的协作、质量门禁的流程卡点做回顾总结，总体上管控交付过程。两周一个迭代的项目回顾会时长在 1 小时以内，回顾会不要超过 1 个小时，测试工程师的质量回顾不要超过 10 分钟为宜，针对其他迭代周期的回顾，可按需采纳会议时长。具体取决于迭代的时间长度以及团队完成的工作量。改进项可以整理成技术需求故事卡，进入团队的待办事项列表，在后续迭代中进行交付。

回顾会常用 5 段式讨论完成，分别是预设会议基调、收集数据、激发灵感、决定做什么和结束会议，具体的实施方法以及其他回顾会的套路、方法可以参加更多的敏捷实践。

2. 月度、季度、年度的质量回顾和总结

整体的质量回顾和总结一般会按照月度、季度、年度的时间阶段，虽然时间跨度不一致，但是形式、内容方面还是相对一致的，对于月度、季度、年度质量回顾和总结可以从如下几个方面进行确定：

- 目的：为了提高软件产品的交付质量，增加客户满意度，促进团队学习和成长；

- 对象：是软件项目或者软件产品在不同的阶段或者周期范围的的制品过程质量和交付质量；

- 内容：对软件项目或者产品在需求、设计、编码、测试、发布等方面的质量保证活动和质量结果进行评价，分析问题原因和影响，总结经验教训和优化建议；

- 形式：根据实际情况选择合适的方法和工具，例如文档、汇报、会议等，参与人员包括项目相关人员、客户代表等；

- 结果：形成一个回顾与总结报告或者记录，包含了项目或者产品的质量数据、缺陷分析数据、经验教训数据和改进计划数据等，如果有可转换成技术需求卡的可以讲需求卡的访问链接一并记录在回顾和总结报告中。

将月度、季度、年度的回顾和总结进行标准化会是一种更好的实践方式，这样随着团队不断的实践和积累有助于团队资产的积累并增强团队成员之间的信任。也鼓励团队成员对内容、方式给出更多的改进想法，鼓励更多的成员参与，保持会话的活跃度。总体来说这种时间跨度比较大的回顾和总结重点就是发现优秀的质量表现、挖掘质量实践，建立质量回溯机制，如图 5-11 所示。

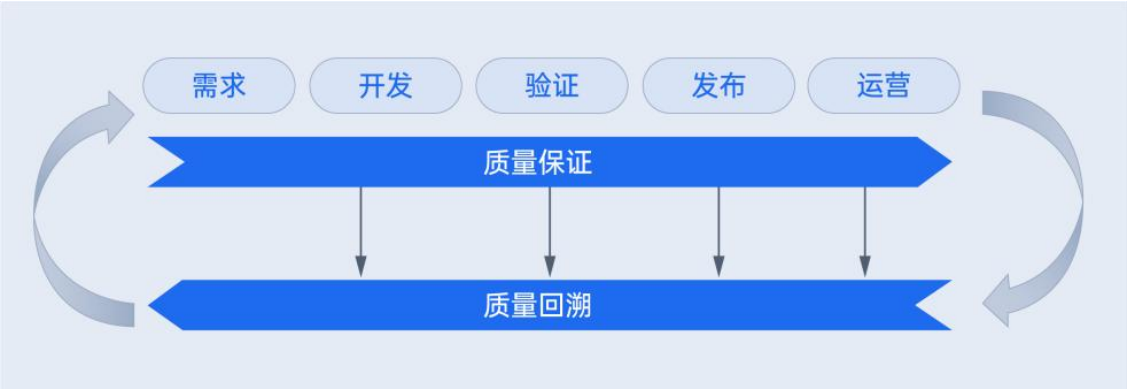


图 5-11 回顾和总结

从而通过制品过程中的质量保证和质量问题的回溯达到了质量回顾和总结的闭环。质量保证目的是质量水平的保持，质量回溯目的是质量保证水平的提高。

5.4.8 文档管理

1. 介绍

在软件研发的质量管理中涉及的文档有多种多样，包含了记录软件系统设计信息的软件开发设计类规范性文档，在过程中记录、留痕信息的研发过程文档以及记录质量结果相关文档等。规划文档管理部分的工作是确保组织能够有效地管理和控制所有相关文档，以支持业务的顺利运作和质量管理的实施，文档管理主要涉及了文档标准和流程、文档

分类和存储、文档的访问和权限约束、文档的审计和改进。

2. 文档标准和流程

在组织内部需要有适用于组织的文档标准和流程，包含文档的创建、评审、批准、发布、修订等流程，这些标准和流程需要组织内部全员推广，接受审计。

（1） 创建：确定需要创建的文档类型和内容，并制定相关的标准和要求，这里面包含了文档的命名要求、特定的文档模板、统一的文档结构、组织级的文档格式和样式等。

（2） 评审：在文档创建之后，进行文档评审以确保其准确性、合规性和可行性。评审可以由相关部门或指定的人员进行，他们将评估文档的内容、语法、技术要求、法规要求等方面。

（3） 批准：经过评审后，文档需要获得批准以确保其适用和可靠性。文档批准的过程可能需要相关领导或专门的批准人对文档进行评估并签署批准意见。

（4） 发布：批准的文档将被发布给相关人员或部门，以确保他们能够访问、使用或参考。文档发布可以通过内部网络、共享文件夹、文档管理系统等方式进行。

（5） 修订：在使用过程中，文档可能需要进行修订以反映变更、修复错误或改进内容。修订的过程应遵循变更控制的原则，包括记录变更的日期、版本号、修订内容和修订人。

（6） 注销和销毁：当文档不再需要时，可以根据规定的保留期限进行注销，并销毁。无论是电子文档还是纸质文档都需要确保文档的安全销毁，以防止机密信息泄露或不当使用。

3. 文档分类和存储

文档的分类和存储主要是通过文件管理系统提供一种方式来组织文件，通过目录、子目录等层次结构为文件建立合适的组织方式，根据项目、主题、日期等相关因素进行分类，以便于快速的查找、访问和方便的管理。该部分一些常见的方法包含文档的分类和命名、文件的索引和元数据、安全存储和备份、归档、访问控制、版本控制和物理文档的管理。

（1） 文档的分类和命名：使用目录、子目录来建立层次化的组织结构。目录可以按照项目、部门、主题、日期等不同的分类方式进行创建，以便将相关文档组织在一起。同时也需要制定一致的文件命名规则，以便于识别和查找文档。命名规则可以包括项目代码、日期、文档类型等关键信息，以确保文件名的描述性和唯一性。

（2） 文件的索引和元数据：应该为文档建立索引和元数据，从而可以提供更加丰富的搜索和筛选的能力，元数据的定义可以包括文件名、作者信息、创建的日期、修改的日期、关键字、文档内容摘要等等能够帮助搜索、筛选能力的内容，从而可以实现快速的定位和筛选所需要的文档。

（3） 安全存储和备份：文档是组织级别资产，要确保存储截止的稳定和可靠，确保文档的安全存储和备份，以防止意外丢失或损坏。这可能涉及使用网络存储、服务器存储、云存储等技术，以及定期进行数据备份和恢复测试。

- 归档：在不同行业，不同公司对过期、过时文档都应该有对应的归档处理，对不再活跃或已过时的文档进行归档和保留。归档的文档应该按照指定的保留期限进行管理，并采取适当的安全措施来保护其完整性和可访问性；

- 访问控制：设置适当的文档访问权限，以确保只有经授权的用户可以访问和修改文档。这包括定义不同用户角色和权限级别，并进行权限管理和访问审计；

- 版本控制：确保文档版本控制的实施，以防止混淆和误用旧版本的文档。这可以包括使用版本号、修订日期、修订记录和变更控制表等工具来管理文档的变更历史和版本跟踪；

- 物理文档管理：对于纸质文档，可以使用物理存储方法，如文件柜、文件夹、标签等，以及档案管理流程来进行分类、存储和维护。

4. 文档的访问和权限约束

文档的访问和权限约束是对文档的使用价值描述，确保文档的访问权限受到适当的管理和控制，以保护机密信息和遵守数据隐私法规。制定合适的权限级别和角色，并实施适当的访问控制措施，主要包含用户角色和权限级别、访问控制、目录权限等方面。

（1） 用户角色和权限级别：定义不同的用户角色和权限级别，根据用户的职责和需要确定其可访问和操作的权限范围。确保可访问的用户有对应的权限，无法访问的用户无对应权限。

（2） 访问控制：建立基于角色的访问控制规则，将用户分配到特定的角色，每个角色都有与之关联的权限。这样可以简化权限管理，并根据用户角色的变化自动更新权限，推荐使用访问控制列表（ACL）进行角色和缺陷的控制。

（3） 目录权限：除了针对单个文档的权限控制，还需要约束对整个目录以及结构层次的权限约束。这样可以实现对目录下所有文档的批量权限管理。

第六章

制定质量考核机制

第六章 制定质量考核机制

6.1 质量检查设计

组织常见的质量检查设计包括了组织内部的质量审核，项目关键里程碑的质量评审，版本发布的决策评审等活动。无论何种形式，质量检查的目的是保护客户不被劣质质量的产品所困扰，并能够及早发现，预防问题的漏出，降低组织内耗，提升组织利润率。

设计有效的质量检查机制，需要注重以下实际操作细节：

1. **检查标准制定**

根据过程的输入和输出要求，制定对应的检查标准，如代码规范检查标准、文档质量检查标准。

2. **检查表/列表设计**

将检查标准化为可操作的清单，明确检查内容、评分机制、不符项处理等。

3. **检查时机确定**

确定关键节点进行检查，如需求评审、代码提交检查、文档评审等。

4. **抽查范围确定**

根据资源情况，确定抽查的样本量和比例，如抽查 10%的代码提交。

5. **检查人员确定**

考虑相关角色的专业能力尤其是检查人员，如测试人员检查测试用例。

6. **检查结果记录**

记录检查发现的问题和改进机会，以进行跟踪和进行数据分析。

7. **质量数据收集**

收集质量检查相关的数据，如发现问题的数量、类型和次数等。

8. 改进闭环设置

根据检查结果，推动对应改进计划，形成“检查-改进”的闭环。

9. 检查效果评估

定期评估检查机制的效果，如问题发现效率和项目质量提升情况。

6.2 质量评估执行

实施质量评估需要关注以下方面：

1. 评估计划制定

明确评估的范围、对象、时间线、资源及方法。评估可面向流程执行效果、产品质量达成程度等。

2. 评估团队组建

由熟悉业务与质量管理的资深员工组成评估小组。也可外聘专业评估机构。

3. 评估标准准备

依据组织质量目标与标准，制定量化的评估指标与考核标准。

4. 评估实施

采用文档检查、过程观察、数据统计等方式，收集评估证据信息。

5. 评估报告撰写

汇总评估结果，形成评估报告。报告需量化结果，确保结论客观公正。

6. 改进机会识别

识别评估中发现的质量问题及改进机会，区分不同类别，按优先级排列。

7. **改进计划制定**

针对主要问题制定改进方案，确定负责人，考虑所需资源与时间。

8. **改进执行监督**

跟踪改进计划的执行进度，必要时采取措施，确保改进落地见效。

9. **评估效果确认**

在下一轮评估中，验证前一轮改进效果，确保问题得到纠正，达到持续改进。

6.3 质量改进闭环

建立质量改进的闭环机制需要注意以下方面：

1. **问题识别**

通过各种渠道收集质量问题、客户反馈、员工建议，质量目标的错失，客户审核改进项等，从中识别质量改进机会。

2. **根本原因分析**

对关键问题进行根本原因分析，找到问题产生的根源，而非仅治标。

3. **改进方案制定**

针对分析结果，由相关部门、团队集体制定改进方案。

4. **执行计划确定**

明确改进方案的执行步骤、时间表、资源及职责分配，制定详细计划。

5. **改进执行**

按计划组织实施改进措施，确保各项行动落到实处。

6. **效果评估**

在改进后适当时间对效果进行评估，判断问题是否得到解决。

7. 标准化和推广

将经验证的改进方案应用到更广范围，并形成规范，避免问题重复出现。

8. 持续改进文化培育

不断完善闭环流程，使质量改进成为一种组织文化和习惯。

第七章

实施质量培训和文化建设

第七章 实施质量培训和文化建设

7.1 质量培训计划

质量团队的培训是提升团队技能和知识水平的关键环节。培训内容包含且不限于以下几方面：

1. 质量管理基础

提供质量管理的基本概念和原则的培训，包括质量规划、质量控制、质量改进和度量等方面的知识。团队成员需要了解质量管理的基本流程和技术。

2. 质量管理过程培训

培训对质量管理过程的要求，包括质量规划、质量控制、质量度量和绩效评估、质量改进等方面。团队成员需要理解并熟悉这些过程的实施步骤和技巧。

3. 领域相关的培训

根据团队所在的行业和项目类型，提供领域相关的培训，例如软件开发方法、测试技术、需求工程等。这些培训可以帮助团队成员在具体的质量管理活动中应用相关的领域知识。

4. 软件配置管理培训

如果团队负责软件配置管理，需要提供相关的培训，包括配置管理的基本原理、工具和技术的使用方法、版本控制、变更管理等方面的知识。

5. 沟通和协作培训

团队成员需要具备良好的沟通和协作能力，因此可以提供沟通和协作技巧的培训，包括团队合作、有效沟通、冲突解决等方面的内容。

为了更好的组织质量团队培训，建议采取以下步骤：

1. 确定培训需求

根据团队成员的角色和职责，确定每个成员需要接受的培训内容，并结合项目和组织的

需求进行评估。

2. 制定培训计划

根据培训需求，制定详细的培训计划，包括培训内容、培训形式（例如面对面培训、在线培训、研讨会等）、培训时间和地点等。

3. 确定培训资源

确定培训所需的资源，包括培训师资、培训材料、培训设施等。可以考虑内部培训师、外部专家或培训机构的合作。

4. 组织培训活动

根据培训计划，安排培训活动的时间和地点，并发送邀请给参与培训的团队成员。确保培训活动的顺利进行，包括提供必要的培训设施和技术支持。

5. 提供培训材料

为培训参与者提供相关的培训材料，包括课程大纲、参考资料、案例研究等。这些材料可以帮助参与者更好地理解和掌握培训内容。

6. 进行培训评估

在培训结束后，进行培训效果评估，收集参与者的反馈和意见。根据评估结果，对培训进行改进和优化，以提高培训的质量和效果。

7. 持续学习和发展

质量团队应鼓励成员进行持续学习和发展，提供进一步的培训机会和资源，帮助团队成员不断提升质量管理的能力和水平。

需要注意的是，培训计划应根据组织的实际情况进行定制化，根据团队成员的角色和职责、项目特点和质量管理需求进行调整。同时，培训应注重实践和案例分析，以帮助团队成员将所学知识应用到实际工作中，并促进知识的分享和交流。

7.2 质量文化推广

构建一个积极健康的质量文化对于质量团队的成功至关重要。以下方式建议能帮助大家

构建一个良好的质量文化：

1. 高层发声

由高管在重要会议和场合发表关于质量的讲话，释放推进质量文化的信号，在整个组织中强调质量意识的重要性。这可以通过培训、宣传和激励机制来实现。质量意识的根本是每个团队成员都要理解他们的工作对最终产品质量的影响，并对提供高质量的成果负责。

2. 宣传推广

通过内部刊物、宣传海报、邮件等多种方式进行质量文化宣传。

3. 实施质量管理过程

确保团队遵循一套规范化的质量管理过程，包括质量计划、质量控制和质量改进。这些过程可以帮助团队识别和解决质量问题，并确保在项目的各个阶段都有适当的质量控制措施。

4. 鼓励团队合作

建立一个团队合作的环境，鼓励团队成员之间的沟通和协作。团队成员应该能够自由地分享知识和经验，并共同解决质量问题。这可以通过定期组织团队会议、知识分享活动和跨部门合作来实现。

5. 典型案例示范

总结和展示质量管理良好实践的正面案例，树立典型，进行横向推广。

6. 目标考核

确保团队对质量目标有清晰的理解，并提供可衡量的指标来评估和监控团队的绩效。这有助于激励团队成员努力达到高质量标准，并及时纠正任何潜在的问题。将质量目标纳入员工和部门绩效考核体系，强化质量意识。

7. 培训强化

继续深化质量管理理念培训，使员工形成共同理念。

8. 经验共享

鼓励知识共享和学习，使团队成员能够不断提升自己的技能和知识水平。这可以通过组织培训课程、工作坊和交流活动来实现。团队成员应该被鼓励参与专业社区，并将新的最佳实践引入到团队中。

9. 授权激励

对在质量管理中表现突出的员工给予授权和激励。

10. 领导力示范

管理者以身作则，在日常工作中践行质量优先。

11. 持续改进

鼓励团队成员积极参与持续改进活动。建立一个开放的反馈机制，鼓励团队成员提出改进建议并参与改进实施。这可以通过定期审查团队的绩效、收集客户反馈和组织内部审核来实现。

第八章

提供资源保障

第八章 提供资源保障

8.1 人力资源配备

8.1.1 人力资源配备要点

在质量管理体系中，需要关注人力资源的以下配备：

1. 质量管理团队配备

设立独立的质量管理部门，或在各部门设立质量管理角色，确保质量工作的推进。

2. 培养质量专业人才

通过内部培养和外部招聘，获取具备质量管理专业知识和技能的人才。

3. 质量管理岗位设计

设计质量经理、质量工程师、质量审核员等质量管理岗位，明确岗位职责。

4. 质量考核激励

建立质量目标责任制，将质量指标纳入员工绩效考核，形成质量激励。

5. 供应商质量人员交流

与重要供应商开展质量管理人员的交流，提升供应链质量意识。

6. 人力资源培训规划

根据组织质量目标，制定相应的质量管理培训规划。

7. 质量协作机制建设

通过工作组等形式，促进与其他部门的质量管理人员协作。

8. 质量人才梯队建设

关注质量人才的职业规划，建立质量管理人才梯队。

8.1.2 人员管理

要想做好质量管理，需要关注所谓的“铁三角”——人、流程、技术，而这三个维度缺一不可。本节主要讨论的就是关于“人”的维度，因为人的因素往往是这三个维度中最重要的。怎样才能进行有效的人员管理，我们可以借鉴人力资源管理领域的四个环节“选、用、育、留”来阐述如何对人员进行管理。

1. 选

“选”就是要选择合适的人员团队。需要注意的是这里提到的“合适”并不是说指要选择资深人员，因为资深人员一般成本会比较高，对于项目而言是需要考虑投入产出比的，所以一个团队不可能全都由“高手”组成。关于如何才能选到合适的人，有以下几点实践可以供读者参考：

（1） 要明确招聘需求：即要清晰地知道团队需要招什么类型、什么技能、什么背景的人。只有需求清晰了，招聘同事或猎头在寻找候选人的时候才能精准地找到相应合适的人。

（2） 引入多元化的招聘渠道：不仅仅依赖传统的招聘渠道，如招聘网站和招聘中介，还可以考虑利用社交媒体、校园招聘、员工推荐等渠道，以扩大招聘范围，吸引更多优秀的候选人。

（3） 简历筛选要注意细节、“管中窥豹”：比如我们在看简历的时候可以检查中文和英文的内容是否对应，以此推论这个候选人是否细心。因为从事质量管理工作的人如果不够细心是做不好的，如果对自己简历中的问题都熟视无睹，那就很难证明他可以做好质量管理。

（4） 进行有效的面试和评估：设计有针对性的面试问题，结合岗位要求和能力评估，进行综合性的评估。可以采用不同形式的面试，如行为面试、案例面试等，以获取更全面的了解候选人的能力和潜力。可以进行多轮次面试，并让多个面试官参与评估候选人。不同的面试官可以提供不同的观点和评估维度，有助于更全面地评估候选人的能力和适应性。

（5） 进行背景调查和参考核实：在最后决定前，进行背景调查和参考核实，联系候选

人的前雇主、直属上司或同事，了解他们的工作表现、团队合作能力和职业操守等方面的信息。

（6） 注重文化匹配：除了技能和经验，确保候选人与组织的价值观和文化相匹配。考虑候选人的工作风格、沟通方式、适应能力等因素，以确保他们能够融入并与团队协作。

（7） 态度和自学能力比当前所知更重要：我们不要因为候选人个别问题答不出，就断定该候选人不适合。现在不会不代表以后也不会，关键是候选人要有诚恳的态度、主动的意识和较好的自学能力。只要具备这些品质，假以时日，经过一段时间的锻炼后他一样可以成为合格的质量管理人员。

2. 用

“用”是指要用好这些人才，最大化的发挥每个人的能力。关于“用”以下有一些实践可以供读者参考：

（1） 适应性培训：按照所到的团队培训对应内容，为员工提供适应性培训，帮助他们快速适应组织文化、工作流程和团队合作方式。培训可以包括组织介绍、岗位培训、工作流程说明等，以帮助他们尽快上手。

（2） 要把合适的人放在合适的岗位中，人尽其才：有人天生就喜欢钻研技术，那可以安排偏技术方向的工作岗位；有人属于外向型，喜欢与人沟通，那就可以安排多与客户或者外部对接的岗位，从而可以发挥沟通能力强的优点。只有把每个人放在自己喜欢而且适合的位置，他才会有很强的主动性。而如果位置放错了，那么这个人不仅会丧失主动性，而且可能还会导致人员流失的风险。

（3） 做好绩效管理：建立明确的绩效管理体系，设定明确的目标和绩效标准。定期进行绩效评估和反馈，帮助员工了解自己的表现，并提供改进和成长的指导。

3. 育

“育”就是要考虑如何培养人员，帮助他们提升自我。关于“育”有以下一些实践可以供读者参考：

（1） 采用激励的方式而不是惩罚的方式来培养人才：使用激励方式将会收到更加明显地效果，因为惩罚的方式短期内或许有效，但可持续性不强。激励的方式可以很多种，比如可以搞些内部竞赛赢大奖之类的活动，一方面可以活跃团队气氛，另一方面也可以提高成员学习热情从而增长知识。需要注意的是在进行奖励时物质和精神奖励同样有效。

其实有时候，员工对于领导的认可和表扬可能比给它发点奖金的激励更加有效。

（2） 建立导师制度：建立导师制度，将新员工与经验丰富的员工或领导者进行配对，进行经验分享和指导。导师可以提供实践经验、指导建议和职业支持，帮助新员工更好地融入和发展。

（3） 注重对新员工的培养：一般来说新员工刚进公司，对环境比较陌生，心理上会有不安全感。这时如果能对员工多加关心和爱护、帮助员工一起设计和规划未来发展的方向，这对员工以后的成长非常重要。一方面能迅速消除新员工的陌生感，尽快融入团队；另一方面可以按照团队要求的方向去发展，一张白纸总是比涂满色彩的纸张能画出更美丽的图案。

（4） 定期评估和反馈：进行定期的绩效评估和反馈，与员工讨论他们的职业发展和培训需求。根据评估结果，制定进一步的培训计划和发展机会。

（5） 建立学习型组织文化：鼓励学习和知识共享的文化，在组织中营造一个积极的学习环境。提供资源和平台，让员工可以分享经验、学习新知识，并互相支持和合作。

4. 留

“留”就是要留住对团队有价值的人员。对团队没有价值或者是“害群之马”，当然需要从团队中移除出去。如何才能留住人才，有以下几点实践供读者参考：

（1） 要让成员了解团队风格，帮助其适应风格：每个公司、每个组织、每个团队都有自己的风格。作为团队中的一分子，只有适应这个风格，融入到这种风格，才能在团队中稳定下来。如果成员发现不能适应目前团队的氛围，那是很难能留得住人的。

（2） 主管要多与团队成员沟通：管理学有过一个研究，员工的离职，90%是和他的顶头上司有关。作为团队主管，需要经常性和下属进行沟通，不管是正式的还是非正式的沟通。沟通可以让员工的情绪得到宣泄，起到防微杜渐的效果，比如可以规定自己每周需要和三名下属进行沟通聊天，可以一起喝咖啡或者一起聚餐，这样在相对放松的环境下，员工往往会比较容易敞开心扉，也有助于他释放负面情绪，起到稳定团队的作用。

（3） 创造内部岗位轮换的机会：有些岗位做了很长一段时间后人就会疲惫，变得没有动力。这时可以考虑团队内部进行轮换。这样一方面让成员保持新鲜感，另一方面也让成员有机会接触和学习新的东西。

（4） 提供职业发展和晋升机会：为员工提供良好的职业发展和晋升机会。制定明确的晋升路径和发展计划，为他们提供挑战性的项目、培训机会和导师指导，帮助他们实现

个人和职业目标。

人员管理只要做好选、用、育、留四个环节，团队就会变得稳定，人员素质就会变得更好，而整个团队效率就会变得更高。

8.2 基础设施建设

质量管理体系需要以下基础设施建设：

1. 质量管理信息系统

选择或开发质量数据收集、分析和报告的信息系统平台。通常与项目管理信息系统 PMIS 合二为一。

2. 质量知识库建设

构建质量管理知识库，收集质量技术文件、最佳实践等。

3. 质量工具配备

配备质量控制统计工具、质量函数展开工具、质量成本核算工具等。

4. 测试环境建设

提升软件测试环境的覆盖度，配置模拟生产环境中不同软硬件版本，满足测试对环境的要求。

5. 质量管理平台建设

创建质量管理可视化展示平台，展示质量目标、进度和关键质量指标。

6. 智能质量管理应用

采用图像识别、大数据分析等新技术提升质量管理能力。

7. 质量管理专家库建设

建立内部质量管理专家库，开展经验传递。

8. 质量管理文化场馆

建设质量管理主题环境、展览等文化场馆。

8.3 资源分配保障

要确保质量管理体系建设取得进展，需要在以下方面提供资源保障：

1. 预算投入保障

在年度预算中，确保质量管理项目和基础建设的资金需求。

2. 人员配备保障

根据质量组织结构和岗位设置，逐步补充质量管理人员编制。

3. 能力建设保障

按照人员培训规划，逐步扩大培训规模，提升能力。

4. 设备投入保障

根据信息化、自动化需求，配备质量管理信息系统。

5. 激励考核保障

将质量目标完成情况纳入员工绩效考核体系，形成质量激励。

6. 组织协同保障

建立质量管理与其他部门的协作机制，争取组织资源协同支持。

7. 过程管控保障

优化质量管理各项流程，提高资源使用效率。

第九章

持续改进与评审

第九章 持续改进与评审

9.1 内部质量审核（Audit）

内部质量评审指组织内部针对质量管理体系进行的评审活动，开展内部质量评审需要注意以下要点：

1. 评审范围确定

明确评审质量管理体系的范围，如某项流程、关键岗位活动等。

2. 评审计划制定

确定评审的时间区间、资源投入、评审小组等，形成评审实施计划。

3. 评审标准拟定

根据质量管理体系要求，拟定评审的检查项和评分标准。

4. 评审小组组建

由质量管理和相关部门经验丰富的员工组成评审小组。

5. 开展评审

通过访谈、抽查、数据分析等方式，收集评审证据信息。

6. 不合格项处理

对发现的不符合项，要求相关方面限期整改。

7. 评审报告撰写

撰写评审报告，记录评审结果、发现问题及整改情况。

8. 评审效果评估

评估评审的覆盖率和效果，不断优化评审工作。

9.2 外部质量审核

外部质量考核是指由组织外部的独立机构或个人对软件研发过程和成果进行评估和考核。与内部质量评审相比，外部质量考核更注重客观性和公正性，因为外部机构或个人通常不具备组织内部的各种利益关系和主观因素干扰。

外部质量审核可以简单分类为“二方审核”或“三方审核”。二方审核通常是由甲方客户或者甲方客户委托的第三方对组织进行的审核；而“三方审核”是由独立的，经过授权的第三方机构对组织进行的审核。

以下是外部质量考核的一些要点：

1. 审核机构选择

选择具备相关经验和资质的外部机构或个人进行质量考核，确保其能够进行客观、公正的评估。

2. 审核范围确定

明确外部质量考核的范围，包括需要评估的软件研发阶段、流程、成果等。同时，需要确定考核的目标和期望的结果，以便在考核过程中进行有针对性的检查。

3. 审核标准制定

根据软件研发质量管理体系的要求，制定具体的考核检查项和评分标准。这些标准应该与软件研发过程的具体环节相对应，以便在考核过程中进行准确的评估和判断。

4. 考核实施

由外部机构或个人对软件研发过程和成果进行评估和考核，通过访谈、抽查、数据分析等方式，收集能够证明软件研发质量管理体系符合性的证据信息。

5. 不合格项处理

在考核过程中发现的不符合项，需要要求相关方面限期整改。对于严重的不符合项，可以采取现场纠正措施或者进行跟踪管理，确保不符合项得到及时有效的解决。

6. 审核报告撰写

考核报告是外部质量考核的重要成果之一，需要详细记录考核结果、发现问题及整改情况。考核报告应该清晰、准确、完整地反映考核过程和结果，以便相关方面了解软件研发质量管理体系的现状和改进方向。

7. 审核效果评估

最后，需要对外部质量审核的效果进行评估。评估的内容包括考核的覆盖率、发现问题的数量和质量、整改措施的有效性等。通过评估，可以发现外部质量考核的不足之处，并采取相应的措施进行改进和优化，提高软件研发质量管理体系的整体水平。

需要注意的是，外部质量考核通常是基于特定的目的和要求进行的，例如认证、评估、审核等。因此，在具体的实施过程中，需要根据外部机构或个人的要求和标准进行针对性的评估和考核。

9.3 持续改进机制

持续改进机制是指在一个组织内部建立一种机制，通过不断监测、评估、反馈和改进，推动软件研发质量管理体系的持续优化和提升。以下是持续改进机制的一些要点：

1. 建立持续改进的文化

在组织内部建立持续改进的文化，鼓励员工积极参与质量改进活动，并营造一种不断追求卓越的氛围。

2. 监测和分析

通过各种数据监测和分析手段，收集和分析软件研发过程中的数据，包括缺陷率、任务完成时间、代码质量等，以了解软件研发过程的问题和改进空间。

3. 评估和反馈

通过定期的评估和审核，对软件研发质量管理体系进行全面的检查和评估，发现并反馈存在的问题和不足之处，提出改进建议和方向。

4. 建立问题跟踪机制

建立问题跟踪机制，对发现的问题进行跟踪和管理，确保问题得到及时有效的解决。同

时，对解决问题的过程进行记录和总结，为将来的改进提供参考和依据。

5. 推广最佳实践

总结和推广软件研发过程中的最佳实践，将成功的经验和方法进行分享和传播，以推动在整个组织内部的经验交流和知识共享。

6. 激励和奖励

建立激励和奖励机制，对积极参与改进活动并取得明显成果的员工进行表彰和奖励，以激发员工的积极性和创造性。

7. 持续改进循环

持续改进机制是一个不断循环的过程，需要不断监测、评估、反馈和改进。通过不断优化和提升软件研发质量管理体系，可以提高软件研发的效率和质量，为组织的长期发展提供保障和支持。

附录 A

常见的质量管理体系介绍（QMS）

附录 A 常见的质量管理体系介绍（QMS）

不同的行业，价值创造流程会不一样。不存在一套流程适用所有的行业，所以，逐渐形成不同的，具有行业特色的质量管理体系。

目前,国际上的质量管理体系标准主要有 ISO9000,卓越绩效模式,QS9000,ISO/TS16949, VDA，TL9000，等。同时，针对软件行业的特殊性，有 IT 软件的 CMMI 软件能力成熟度模型，汽车软件的 ASPICE 评估模型，这些模型都是质量管理体系的有效补充。

不同的行业标准，法令法规的要求，会有各自深深的行业烙印，形成行业特色的，必须遵守的规范。体系标准会规定流程必须包含特定的，本行业需要遵守的要求，比如电信行业标准 TL9000 要求必须实现产品需求到产品设计，开发和测试的双向可追踪性。因为行业特色的存在，如果要进入到该行业，成为行业中的一员，组织需要获取本行业必要的质量体系认证，验证和确认体系的适用性和有效性。组织一旦获得了认证并拿到认证证书，就可以对外宣传（但需遵守使用原则）组织已成功获得认证。为保证认证资格的有效性，组织还需要继续实施所有质量体系的要求。认证机构也会定期对标准执行情况要进行检查（监督审核）。认证有效期一般为 3 年，三年后需要再次评价（复评），复评合格更换认证证书。

表 A-1 常见的质量管理体系及适用行业

行业	质量管理体系
任何行业	ISO9001
汽车行业（硬件相关）	IATF 16949
医疗服务行业	ISO13485
食品行业	ISO22000

服务行业	ISO20000
IT 信息技术	ISO27001
航空工业	AS9100D
电信/信息通讯	TL9000
道路交通安全	ISO39001

图 A-2 是 ISO 国际标准化组织 2018 年对各质量管理体系认证证书的最新调查结果。很明显，ISO9001 仍然是认证证书最多，行业最广的一种认证。

图 A-2 ISO 2018 认证调查

ISO9000 族标准是国际标准化组织为适应国际经济技术交流和国际贸易发展的需要而制定的质量管理和质量保证标准，现已有 170 多个国家采用。特别是 2000 年最新版 ISO9001：2000 质量标准，具有适用性广、通用性强，对与质量有关的活动进行系统控制的优势，其核心内容是“使客户满意”。贯彻 ISO9000 族标准已被众多企业所看重，成为企业证明自己产品质量、工作质量的一种“护照”。

A.1 ISO9001

ISO9001 的核心质量管理体系的架构如图 A-3 所示：

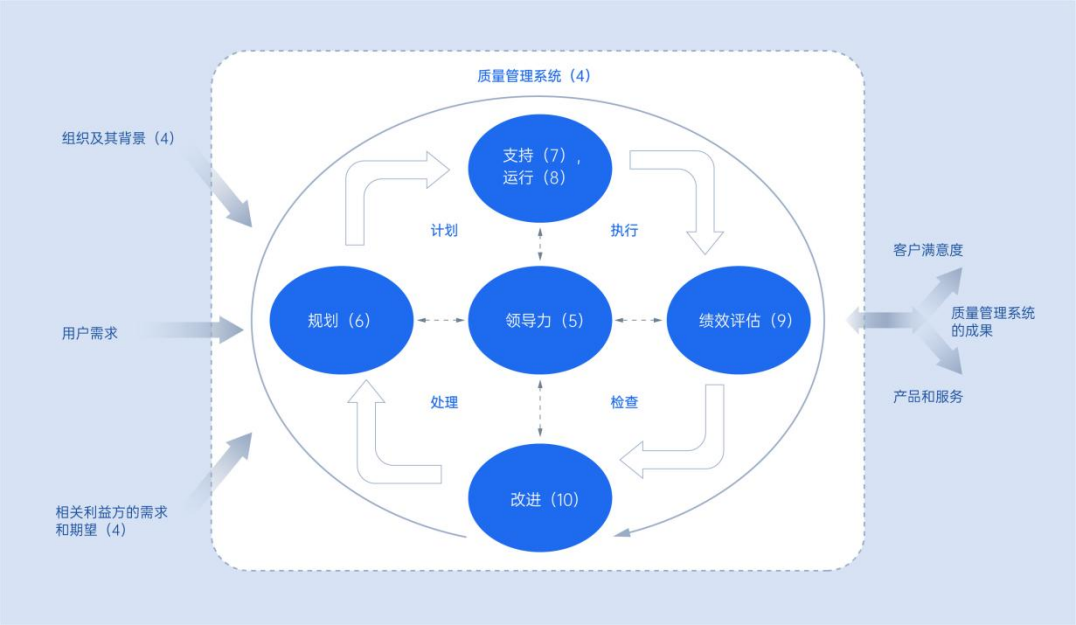


图 A-3 ISO9001 的核心质量管理体系的架构

具体来讲，ISO9001 结构分为 10 个章节。前面 3 个章节是介绍性的，后面 7 个章节包含质量管理体系的要求。以下是 7 个主要章节的内容。

- 第 4 节：组织的环境——本节讨论了理解组织的要求，以实施质量管理体系。它包括识别内部和外部因素，识别相关方及其期望，定义质量管理体系范围以及确定过程及其相互作用的要求；

- 第 5 节：领导作用——领导作用要求涵盖了最高管理者在实施质量管理体系方面发挥作用的必要性。最高管理者需要通过确保以顾客为关注焦点，确定和传达质量方针以及在整个组织中分配角色和职责来证实对质量管理体系的承诺；

- 第 6 节：策划——最高管理者还必须策划质量管理体系的持续有效。需要评估质量管理体系在组织中的风险和机遇，并且需要确定改进的质量目标并制定计划以实现这些目标；

- 第 7 节：支持——支持这一节规定质量管理体系所有资源的管理，涵盖控制所有资源的必要性，包括人力资源、建筑和基础设施，工作环境，监测和测量资源以及组织的知识。该章节还包括有关能力、意识、沟通和成文信息（过程所需的文件和记录）的控制要求；

- 第 8 节：运行——运行要求涉及策划和建立产品或服务的所有方面。本节包括对策划、产品要求评审、设计、外部供方的控制、产品或服务的提供和放行以及不合格过程输出控制的要求；

- 第 9 节：绩效评价——本节包括确保监控质量管理体系是否运行良好所需的要求，包括监视和测量过程、评估客户满意度、内部审核以及对运行中的质量管理体系进行管理评审；

- 第 10 节：改进——最后一节包括使质量管理体系持续改进的要求。这包括需要评估不合格过程并采取纠正措施。

A.2 IATF6949（TS16949）

ISO/TS16949“质量管理体系—汽车行业生产件与相关服务件的组织实施 ISO9001：2000 的特殊要求”。

作为汽车生产的两大基地之一，美国三大汽车公司（通用汽车、福特和克莱斯勒）于 1994 年开始采用 QS－9000 作为其供应商统一的质量管理体系标准；同时另一生产基地，欧洲特别是德国均各自发布了相应的质量管理体系标准，如 VDA6.1、AVSQ94、EAQF 等。因美国或欧洲的汽车零部件供应商同时向各大整车厂提供产品，这就要求其必须既要满足 QS－9000，又要满足如 VDA6.1，造成各供应商针对不同标准的重复认证，这就急需要求出台一套国际通用的汽车行业质量体系标准，以同时满足各大整车厂要求。

为了协调国际汽车质量系统规范，由世界上主要的汽车制造商及协会成立了一个专门机构，称为国际汽车工作组（International Automotive Task Force，简称 IATF）。IATF 的成员由如下 9 家整车厂：宝马（BMW Group），克莱斯勒（Chrysler LLC），戴姆勒（Daimler AG），菲亚特（Fiat Group Automobiles），福特（Ford Motor Company），通用（General Motors Corporation），标致（PSA Peugeot Citroen），雷诺（Renault）和大众（VolksWagen AG）以及 5 个国家的监督机构：美国国际汽车监督局（IAOB），意大利汽车制造商协会（ANFIA），法国车辆设备工业联盟（FIEV），英国汽车制造与

贸易商协会（SMMT）和德国汽车工业协会-质量管理中心（VDA-QMC）组成。IATF 对 3 个欧洲规范 VDA6.1（德国），AVSQ94（意大利），EAQF（法国）和 QS-9000（北美）进行了协调，在和 ISO9001：2000 版标准结合的基础上，在 ISO/TC176 的的认可下，制定出了 ISO/TS16949：2002 这个规范（2016 年之后，ISO 和 IATF 组织共同宣布，IATF16949 标准名称替代 ISO/TS16949）。

2002 年 3 月 1 日，ISO 与 IATF 公布了国际汽车质量的技术规范 ISO/TS16949，这项技术规范适用于整个汽车产业生产零部件与服务件的供应链，包括整车厂，2002 年版的 ISO/TS16949 已经生效，并展开认证工作。

在 2002 年 4 月 24 号，福特，通用和克莱斯勒三大汽车制造商在美国密歇根州底特律市召开了新闻发布会，宣布对供应厂商要采取的统一的一个质量体系规范，这个规范就是 ISO/TS16949。供应厂商如没有得到 ISO/TS16949 的认证，也将意味着失去作为一个供应商的资格。目前，法国雪铁龙（Citroen），标志（Peugeot），雷诺（Renault）和日本日产（Nissan）汽车制造商已强制要求其供应商通过 ISO/TS16949 的认证。

ISO/TS16949 是国际汽车行业的技术规范，是基于 ISO9001 的基础，加进了汽车行业的技术规范。目 ISO/TS16949 的最新版本是 2016 版，该版本是基于 ISO900:2015 发展而来，其整体的规范架构基本与 ISO9001 一致。但更着重于缺陷防范、减少在汽车零部件供应链中容易产生的质量波动和浪费，关注产品成本、在供应链中持续不断的改进、产品质量改进、生产力改进、成本的降低、强调缺点的预防、SPC 的应用、防错措施、减少变差和浪费、确保存货周转及最低库存量、质量成本、非质量的额外成本（待线时间，过多搬运）。图 A-4IATF6949 的标准目录结构，可以看出，其目录保持了与 ISO9001:2015 的完全一致。而图七则是 IATF 在 ISO900:2015 基础上增加的具体要求示例。

目　　录

图 A-4 IATF6949 标准目录结构

5.1.1.2过程效率

最高管理者应评审产品实现过程和支持过程以确保其有效性和效率。最高管理者的管理评审过程活动及结果应包括在管理评审中（见9.3）。

图 A-5 增加具体要求实例

ISO/TS16949 标准的针对性和适用性非常明确，只适用于汽车整车厂和其直接的零备件制造商，也就是说这些厂家必须是直接与生产汽车有关的，能开展加工制造活动，并通过这种活动使产品能够增值。同时，对所认证公司厂家的资格也有着严格的限定，那些只具备支持功能的单位，如设计中心，公司总部和配送中心等，或者那些为整车厂家或汽车零备件厂家制造设备和工具的厂家，都不能获得认证。对 ISO/TS16949 认证的管理是由 5 大监督机构代表 IATF 来完成的，它们采用相同的程序方法来监督 ISO/TS16949 规范的操作和实施，以在全世界形成一个标准和操作完全统一的系统。

对受审核方的要求 ISO/TS16949 认证注册，只适用于汽车整车厂和其直接的零部件制造商。这些厂家必须是直接与生产汽车有关的，具有加工制造能力，并通过这种能力的实

现使产品能够增值。要求获得 ISO/TS16949：2009 认证注册的公司，必须具备有至少连续 12 个月的生产和质量管理记录，包括内部评审和管理评审的完整记录。对于一个新设立的加工场所，如没有 12 个月的记录，也可经评审符合确认质量系统规范要求后，由认证公司可签发由 NQA 颁发的“符合性证明”。当具备了 12 个月的记录后，再进行认证审核注册。经认证获颁证书的机构，如不能继续保持质量体系的正常运转和产品质量的一致性，将有被吊销证书的风险。

而 NQA 为获得 IATF 认可的少数认证机构之一，全面开展 TS16949 的认证活动。SNQA 作为 NQA 的亚太总部，积极地为汽车行业供应商及 OEM 的提供认证。SNQA 在中国各地目前拥有近四十名 TS16949 资深审核员，能根据客户需求迅速安排现场审核。

A.3 CMMI

CMMI（Capability Maturity Model Integration For Software，软件能力成熟度模型集成）是在 CMM（Capability Maturity Model For Software，软件能力成熟度模型）的基础上发展而来的。CMMI 是由美国卡耐基梅隆大学软件工程研究所（Software Engineering Institute，SEI）组织全世界的软件过程改进和软件开发管理方面的专家历时四年而开发出来的，并在全世界推广实施的一种软件能力成熟度评估标准，主要用于指导软件开发过程的改进和进行软件开发能力的评估。所以，严格意义上来讲，CMMI 不属于质量管理体系标准，是质量管理体系之上的对软件成熟度的能力评估标准。

CMMI 模型最初是为了美国国防部衡量其软件承包商的软件产品质量及交付软件产品能力，后期，CMMI 模型从软件工程领域扩展到任何组织，行业，帮助他们开发，改进，度量组织能力及绩效。CMMI 描述的更多的行业认可的标准实践方法，聚焦于如何改进绩效，并将企业运营与商业目标对齐。随着新的工程实践，例如敏捷，CMMI 也与时俱进，与这些新兴的工程实践相结合。图 A-6 是摘自 CMMI Institute 的对企业管理的调查结果，从另一角度进一步说明为什么要引进业界最佳实践：对标和持续改进。

图 A-6 CMMI Institute 对企业管理的调查结果

CMMI 模型是由一系列业界的最佳实践方法构成。这些实践方法按照关键业务能力来分类，帮助企业解决其面临的各种挑战，比如，产品工程和产品开发，绩效提高，服务交付及管理，稳定交付，业务可塑性，管理业务风险，选择及管理供应商，质量保证，员工管理等（具体的关键业务能力参见图 A-7）。

图 A-7 业务关键能力

CMMI 模型包括开发 CMMI 模型，服务 CMMI 模型，供应商管理 CMMI 模型。其中，最常用的模型是开发 CMMI 模型。目前，CMMI 标准的最新版本是在 2018 年 3 月发布的开发 CMMI 模型 V2.0，2020 年 4 月份，CMMI 模型 V1.3 版将不再支持。CMMI 开发模型 2.0 版本共包括了 4 大类，9 个能力域，20 个实践域（CMMI1.3 版本称之为流程域，共 22 个流程域。），这 20 个实践域包括（如图 A-8）：

CMMI V1.3 过程域 ->V2.0 实践域		实践个数小计	Level 1	Level 2	Level 3	Level 4	Level 5
Causal Analysis and Resolution (CAR)	原因分析与解决方案	11	1	2	5	2	1
Configuration Management(CM)	配置管理	7	1	6			
Decision Analysis and Resolution (DAR)	决策分析与解决方案	8	2	5	1		
Estimation (EST)	估算	6	1	3	2		
Governance(GOV)	治理	7	1	4	2		
Implementation Infrastructure (II)	实施设施	6	1	2	3		
Managing Performance and Measurement(MPM)	管理性能与度量元	22	2	6	6	5	3
Monitor and Control(MC)	监督与控制	10	2	4	4		
Organizational Training (OT)	组织级培训	9	1	2	6		
Peer Review (PR)	同行评审	6	1	4	1		
Planning(PLAN)	策划	15	2	8	4	1	
Process Asset Development(PAD)	过程资产开发	11	1	3	7		
Process Management(PCM)	过程管理	12	3	2	6	1	
Process Quality Assurance (PQA)	过程质量保证	6	1	4	1	1	
Product Integration (PI)	产品集成	10	1	6	3		
Requirement Development and Management (RDM)	需求开发与管理	14	1	6	7		
Risk Management (RSKM)	风险与机会管理	8	1	2	5		
Supplier Agreement Management (SAM)	供应商合同管理	10	3	4	2	1	
Technical Solution (TS)	技术解决方案	10	1	3	6		
Verification and Validation (VV)	验证与确认	7	2	3	2		
合计		195	29	79	73	10	4

图 A-8 二十个实践域

1. 执行类别

（1） 质量保证能力（ENQ: Ensuring Quality）

- RDM：（Requirement Development & Management）需求开发和管理。需求开发的目的在于定义系统的边界和功能、非功能需求，以便涉众（客户、最终用户）和项目组对所开发的内容达成一致。而需求管理的目的是在客户和软件项目之间就需要满足的需求建立和 维护一致的约定；

- PQA：（Process Quality Assurance）过程和产品质量保证。为项目组和管理层提供项目过程和相关工作产品的客观信息。

- VV：（Verification & Validation）验证及确认。验证确保选定的工作产品满足需求规格，并确认证明产品或产品部件在实际应用下满足应用要求；

- Peer Review：同行评估，确保第二双眼睛一起保障质量。

（2） 产品工程与开发能力（EDP: Engineering and Developing Product）

- TS：（Technical Solution）技术解决方案。在开发、设计和实现满足需求的解决方案。解决方案的设计和实现等都围绕产品、产品组件和与过程有关的产品；

- PI：（Product Integration）产品集成。从产品部件组装产品，确保集成产品功能正确并交付产品。

（3） 交付与服务管理能力（DMS；Delivering and Managing Service；该能力域属于CMMI 服务模型）

- SDM：（Service Delivery Management）服务交付管理。产品如何交付给客户，开始产品售后维保期；

- STSM：（Strategic Service Management）服务管理战略。应组织战略计划需要，如何建立并维护标准服务。

（4） 供应商选择与管理能力（SMS: Selecting and Managing Supplier）

- SSS：（Supplier Source Selection）供应商选择。组织如何选择潜在供应商，如何发标书应答等实践（该实践域属于 CMMI 服务模型）；

- SAM：（Supplier Agreement Management）供应商合同管理。如果有外包或采购产品或服务的行为时，旨在对以正式协定的形式从项目之外的供方采办的产品和服务实施管理。组织如何评估，量化供应商的绩效。

2. 管理类别

（1） 计划和工作管理能力（PMW: Planning and Managing Work）

- PP：（Project Plan）项目计划。保证在正确的时间有正确的资源可用。为每个人员分配任务。协调人员。根据实际情况，调整项目；

- MC：（ Monitoring and Control）项目监督与控制。通过项目的跟踪与监控活动，及时反映项目的进度、费用、风险、规模、关键计算机资源及工作量等情况，通过对跟踪结果的分析，依据跟踪与监控策略采取有效的行动，使项目组能在既定的时间、费用、质量要求等情况下完成项目；

- EST：（Estimating）评估。对开发或收购或交付的解决方案从工作量，时间和成本进行评估，包括工作本身及所需要的资源。评估用来更好的做计划和锁定项目，降低风险；

- CONT：（Continuity）持续性。对潜在影响公司业务运营的事件或风险提前计划相应的应对措施，保障业务即使在灾难时仍然能够顺利进行（该实践域属于 CMMI 服务模型）。

（2） 业务弹性能力（MBR: Managing Business Resilience）

- RSKM：（Risk Management）风险管理。识别潜在的问题，以便策划应对风险的活动和必要时在整个项目生存周期中实施这些活动，缓解不利的影响，实现目标；

- IRP：（Incident Resolution & Prevention）服务事件解决及预防。解决产品或解决方案的事件，降低对客户的影响，提高服务效率（该实践域属于 CMMI 服务模型）。

（3） 人员管理能力（MWF: Managing the Work Force）

- OT：（Organizational Training）组织培训管理。增加组织各级人员的技能和知识，使他们能有效地执行他们的任务。

3. 支撑类别

（1） 支撑实施能力（SI: Supporting Implementation）

- CM：（Configuration Management）配置管理。建立和维护在项目的整个软件生存周期中软件项目产品的完整性；

- DAR：（Decision Analysis and Resolution）决策分析与解决。应用正式的评估过程依据指标评估候选方案，在此基础上进行决策；

- CAR：（Causal Analysis and Resolution），识别缺失的原因并进行矫正进一步的防止未来再次发生。

（2） 管理安全能力（Manage Safety）

（3） 管理信息安全能力（Manage Security）

4. 改进类别

（1） 习惯养成及坚持能力（SHP：Sustaining Habits and Persistence）

- GOV：（Governance）管理架构。提供指南、建议给组织的高层领导，如何赞助，管控流程管理相关的行动，降低流程实施的成本，提高流程与组织目标的契合度；

- II：（Implementation Infrastructure）实施基础。确保对组织价值重大的流程能够制度化，标准化并持续改进。

（2） 绩效改进（IMP：Improving Performance）

- PCM：（Process Management）流程管理。管理实施对流程和框架持续改进，支持业务目标的实现。识别并实施利益最大化的改进项目，并确保改进结果可见，可用并能制度化，从而形成组织习惯；

- PAD：（Process Asset Development）流程资产开发。创建相应的流程体系，资产，确保流程的必要更新，与时俱进；

- MPM：（Managing Performance and Measurement）绩效与度量管理。通过度量数据和分析来管理绩效，达成业务目标。开发和维持度量的能力，以便支持对管理信息的需要。作为改进、了解、控制决策。

CMMI 成熟度是分级别的。在 CMMI1.3 共有 5 个级别，代表软件团队能力成熟度的 5 个等级，数字越大，成熟度越高，高成熟度等级表示有比较强的软件综合开发能力。而 CMMI2.0 增加了一个 0 级别（见图 A-9）。0 级代表组织能力不存在的，工作可能交付，也可能无法交付，甚至有随时关闭的风险。

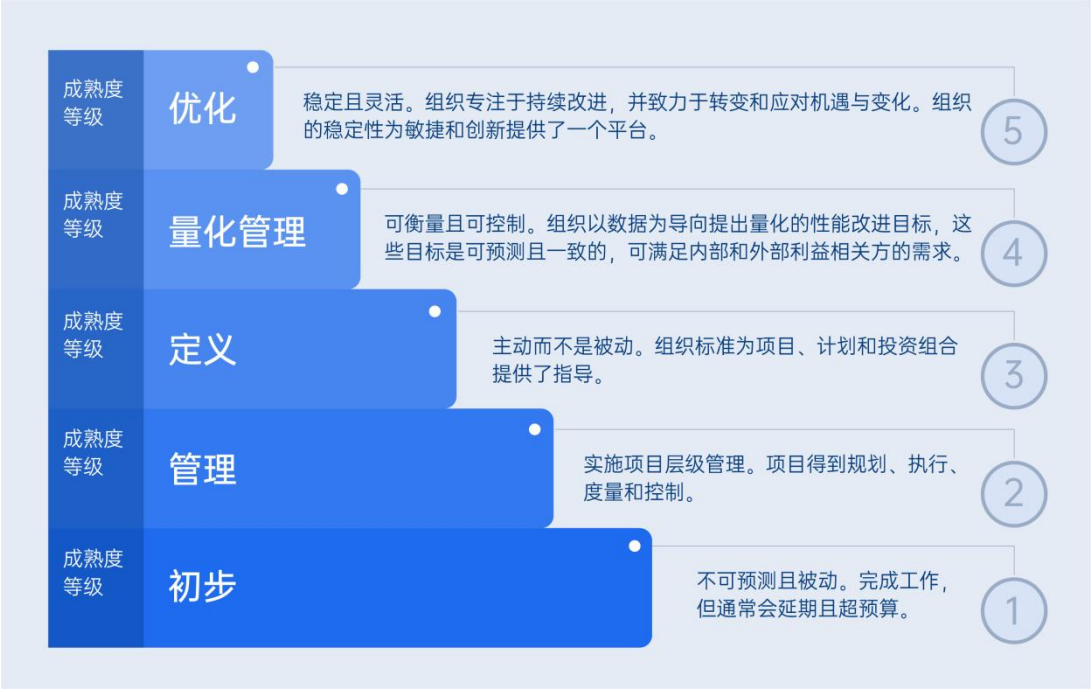


图 A-9 CMMI2.0 成熟度

- CMMI 一级，执行级。在执行级水平上，软件组织对项目的目标与要做的努力很清晰，

项目的目标可以实现。但是由于任务的完成带有很大的偶然性，软件组织无法保证在实施同类项目时仍然能够完成任务。项目实施能否成功主要取决于实施人员；

- CMMI 二级，管理级。在管理级水平上，所有第一级的要求都已经达到，另外，软件组织在项目实施上能够遵守既定的计划与流程，有资源准备，权责到人，对项目相关的实施人员进行了相应的培训，对整个流程进行监测与控制，并联合上级单位对项目与流程进行审查。二级水平的软件组织对项目有一系列管理程序，避免了软件组织完成任务的随机性，保证了软件组织实施项目的成功率；

- CMMI 三级，明确级。在明确级水平上，所有第二级的要求都已经达到，另外，软件组织能够根据自身的特殊情况及自己的标准流程，将这套管理体系与流程予以制度化。这样，软件组织不仅能够在同类项目上成功，也可以在其他项目上成功。科学管理成为软件组织的一种文化，成为软件组织的财富；

- CMMI 四级，量化级。在量化管理级水平上，所有第三级的要求都已经达到，另外，软件组织的项目管理实现了数字化。通过数字化技术来实现流程的稳定性，实现管理的精度，降低项目实施在质量上的波动；

- CMMI 五级，优化级。在优化级水平上，所有第四级的要求都已经达到，另外，软件组织能够充分利用信息资料，对软件组织在项目实施的过程中可能出现的次品予以预防。能够主动地改善流程，运用新技术，实现流程的优化。

附录 B

软件研发质量管理过程常见文档设计实践

附录 B 软件研发质量管理过程常见文档设计实践

软件开发设计文档是一种用于记录软件系统设计信息的文档。它描述了软件系统的整体结构、组成部分、模块划分、数据流、算法、接口等关键设计方面的内容。设计文档提供了开发团队成员之间的共享视野，指导开发工作的进行，并为将来的维护和扩展提供依据。在编写软件设计文档时，确保文档清晰、简洁、易读，并遵循一致的格式和规范。考虑到文档的受众，使用清晰的术语和适当的技术表达。及时更新和维护文档，以反映系统的最新状态和变更。

B.1 架构设计文档编写

1. 文档概述：在文档的开头，提供对文档的概述和目的的简要描述。说明文档的受众和所关注的重点。

2. 系统概述：介绍软件系统的整体概述，包括系统的目标、范围和主要功能。说明系统的关键业务需求和技术要求。

3. 架构目标：阐明系统架构设计的目标和原则。例如，可维护性、可扩展性、性能等方面的目标。

4. 架构视图：使用适当的图表和图示，描述系统的不同视图，如逻辑视图、物理视图、过程视图等。每个视图都应明确说明其目的、关注点和关键元素。

5. 系统组成部分：详细描述系统的主要组成部分和模块，包括每个模块的功能、职责和接口。这包括系统内部的组件和外部的集成模块。

6. 数据流和处理流程：描述数据在系统中的流动和处理过程。说明系统中的关键数据流和处理逻辑，以及数据的输入和输出。

7. 技术选择和决策：讨论系统设计中的关键技术选择和决策。解释选择的原因、优势和劣势，以及与其他备选方案的比较。

8. 接口定义：定义系统内部和外部的接口，包括函数、类、模块等的说明和使用方法。描述接口的输入输出、数据格式、通信协议等。

9. 性能和可伸缩性考虑：讨论系统设计中的性能和可伸缩性方面的考虑。包括关键性能指标、优化策略、缓存策略、并发处理等。

10. 安全性和权限设计：阐述系统设计中的安全性和权限控制机制。描述系统的安全需求和安全措施，如身份验证、访问控制等。

11. 部署和扩展计划：描述系统的部署计划和扩展方案。包括系统的部署拓扑、硬件和软件要求，以及扩展系统的策略和方法。

12. 附录和参考资料：提供附录和参考资料，如相关的图表、图示、文档、文献、第三方库和工具的使用说明等。

13. 审查和验证：在完成初稿后，进行内部审查和验证，确保文档的准确性和完整性

B.2 模块设计文档编写

1. 文档概述：在文档的开头，提供对文档的概述和目的的简要描述。说明文档的受众和所关注的重点。

2. 模块概述：对每个模块进行简要介绍，包括模块的功能、职责和关键特点。明确模块在整个系统中的作用和目的。

3. 接口定义：描述模块的接口，包括输入和输出的数据格式、参数、返回值等。明确模块与其他模块之间的交互方式和接口约定。

4. 数据结构和算法：描述模块内部使用的数据结构和算法，包括关键的数据结构定义、算法逻辑和处理流程。提供足够的细节以支持模块的实现。

5. 函数和方法说明：详细描述模块中的函数和方法，包括函数名、参数、返回值、功能和实现逻辑。提供适当的注释和说明以帮助其他开发人员理解和使用。

6. 异常处理和错误处理机制：说明模块在出现异常和错误情况下的处理方式和机制。包括异常类型、错误码、错误处理流程等。

7. 依赖关系：说明模块与其他模块之间的依赖关系，包括依赖的模块、接口和数据。确保模块的正确集成和协作。

8. 测试策略：描述模块的测试策略和方法。包括单元测试和集成测试的计划和用例设计。性能优化和扩展性考虑：讨论模块设计中的性能优化和扩展性方面的考虑。包

括关键性能指标、优化策略、缓存策略、并发处理等。

9. 安全性和权限设计：阐述模块设计中的安全性和权限控制机制。描述模块的安全需求和安全措施，如身份验证、访问控制等。

10. 附录和参考资料：提供附录和参考资料，如相关的图表、图示、文档、文献等，以帮助理解和实现模块。

11. 审查和验证：在完成初稿后，进行内部审查和验证，确保文档的准确性和完整性。

B.3 数据库设计文档编写

1. 文档概述：在文档的开头，提供对文档的概述和目的的简要描述。说明文档的受众和所关注的重点。

2. 数据库概述：对数据库进行简要介绍，包括数据库的目的、范围和主要功能。明确数据库在整个系统中的作用和目标。

3. 数据库架构：描述数据库的整体架构，包括数据库服务器的部署和拓扑结构。说明数据库服务器之间的关系和连接方式。

4. 数据库模式设计：详细描述数据库的模式，包括表、字段、索引、关系等。明确每个表的功能、属性和关系。

5. 数据表设计：为每个表提供详细的设计说明，包括表名、字段、数据类型、约束、默认值等。明确每个字段的含义和目的。

6. 关系设计：描述表之间的关系，包括主键、外键、一对一关系、一对多关系等。明确关系的约束和操作。

7. 数据完整性和约束：阐述数据库的数据完整性要求和约束条件。包括唯一性约束、非空约束、参照完整性等。

8. 视图和存储过程设计：讨论视图和存储过程的设计和使用，包括视图的目的和内容，以及存储过程的功能和参数。

9. 数据访问和安全性设计：阐述数据库的访问控制和安全性设计。包括用户权限、角色、审计等方面的考虑。

10. 数据备份和恢复策略：讨论数据库的备份和恢复策略，包括备份频率、备份存储位置和恢复流程等。

11. 性能优化和索引设计：讨论数据库性能优化的考虑和策略，包括索引的设计和优化。

12. 数据库扩展计划：描述数据库的扩展计划和策略，包括数据增长的预测、容量规划和扩展方法。

13. 附录和参考资料：提供附录和参考资料，如数据库模型图、表结构定义、索引设计等，以帮助理解和实现数据库。

14. 审查和验证：在完成初稿后，进行内部审查和验证，确保文档的准确性和完整性。

B.4 接口设计文档编写

1. 文档概述：在文档的开头，提供对文档的概述和目的的简要描述。说明文档的受众和所关注的重点。

2. 接口类型和范围：明确系统中涉及的接口类型，如应用程序接口（API）、网络接口、服务接口等。定义接口的范围和功能。

3. 接口命名和版本控制：定义接口的命名规则和版本控制策略。确保接口具有唯一的标识符，并能够支持接口的演进和兼容性。

4. 接口描述和功能说明：对每个接口进行详细描述，包括接口的功能、输入参数、返回值、异常情况处理等。说明接口的预期行为和使用方法。

5. 输入和输出数据格式：定义接口的输入和输出数据的格式，如数据类型、数据结构、数据编码等。确保接口的数据交换格式一致和可理解。

6. 接口调用示例：提供实际的接口调用示例，包括请求的数据、参数设置和响应的数据。帮助开发人员理解接口的使用方法和实际效果。

7. 接口认证和授权：讨论接口的认证和授权机制，包括身份验证、令牌管理、权限控制等。确保接口的安全性和合法性。

8. 接口错误处理和异常情况：说明接口在遇到错误和异常情况时的处理方式和返回结果。定义错误码和错误信息的规范。

9. 接口依赖和关系：描述接口与其他模块、组件或系统之间的依赖关系。确保接口的正确集成和协作。

10. 接口文档和工具支持：提供相关的接口文档和工具支持，如代码示例、SDK、文档链接等。帮助开发人员快速理解和使用接口。

11. 附录和参考资料：提供附录和参考资料，如相关的图表、图示、文档、文献等，以帮助理解和使用接口。

12. 审查和验证：在完成初稿后，进行内部审查和验证，确保文档的准确性和完整性。

B.5 数据流图文档编写

1. 文档概述：在文档的开头，提供对文档的概述和目的的简要描述。说明文档的受众和所关注的重点。

2. 数据流图类型:明确使用的数据流图类型,如数据流程图(DFD)或统一建模语言(UML)中的活动图。说明数据流图的范围和目的。

3. 系统概述：对系统进行简要介绍，包括系统的主要功能和数据流。概述系统中数据流图的位置和作用。

4. 数据流图符号和约定：定义数据流图中使用的符号和约定，包括外部实体、过程、数据流、数据存储和数据转换等。

5. 上下文级数据流图：绘制系统的上下文级数据流图，描述系统与外部实体之间的数据流动。说明外部实体和系统的交互过程。

6. 进一步细化：对上下文级数据流图进行进一步细化，逐步展开系统的各个功能模块和子过程。绘制层级逐渐增加的数据流图。

7. 过程描述和功能说明：对每个过程或功能进行详细描述，包括过程的功能、输入数据流、输出数据流和处理逻辑。说明数据的加工和转换过程。

8. 数据存储：描述数据存储的类型、内容和用途。说明数据存储与其他过程或功能之间的关系。

9. 数据流跟踪：跟踪和标识数据流在数据流图中的路径和转换过程。确保数据流在系统中的正确流动和处理。

10. 异常处理和错误流：讨论异常情况的处理方式和错误流的流向。描述错误处理和异常处理的过程和逻辑。

11. 附录和参考资料：提供附录和参考资料，如相关的图表、图示、文档、文献等，以帮助理解和使用数据流图。

12. 审查和验证：在完成初稿后，进行内部审查和验证，确保文档的准确性和完整性。

B.6 详细算法和数据结构描述

1. 算法描述：

（1） 算法目的：明确算法的目的和所要解决的问题。

（2） 输入和输出：描述算法的输入和输出，包括数据类型、数据结构和数据格式。

（3） 步骤和流程：逐步描述算法的执行步骤和流程。使用适当的结构和语法来清晰地表示每个步骤。

（4） 循环和条件：处理循环和条件语句时，确保算法的正确性和可读性。使用适当的控制结构和逻辑表达式。

（5） 变量和数据结构：描述算法中使用的变量和数据结构，包括它们的类型、命名和作用域。

（6） 时间和空间复杂度：讨论算法的时间复杂度和空间复杂度，以评估算法的效率和资源消耗。

2. 数据结构描述：

（1） 数据结构类型：明确数据结构的类型，如数组、链表、栈、队列、树等。

（2） 结构定义：描述数据结构的定义，包括字段、属性和关联的数据类型。

（3） 操作和方法：列出数据结构的操作和方法，包括增、删、改、查等。描述每个操作的功能和实现细节。

（4） 数据存储和访问：讨论数据结构的存储方式和访问方式，如索引、指针等。

（5） 使用示例：提供使用数据结构的示例代码，演示数据结构的创建、初始化和使用过程。

3. 清晰可读性：

（1） 使用适当的命名：为算法和数据结构中的变量、函数、方法和类选择清晰、具有描述性的命名。

（2） 代码注释：在关键位置添加注释，解释代码的意图、实现细节和关键步骤。

（3） 格式化和缩进：保持代码的良好格式化和缩进，以提高可读性和可理解性。

（4） 图表和图示：使用适当的图表、图示和示意图来解释算法和数据结构的逻辑和操作过程。

在完成初稿后，进行内部审查和验证。确保算法和数据结构描述的准确性、完整性和一致性。编写详细的算法和数据结构描述时，需要注重准确性、清晰性和易读性。使用合适的术语和逻辑结构来表达算法和数据结构的思想和实现细节。适当地使用图表和图示来支持文档的可视化和理解。

B.7 编码过程文档

在软件开发的编码阶段，以下是常见的文档需要维护：

1. 源代码文档：对于每个代码文件或模块，编写代码注释来解释代码的功能、输入参数、输出结果和实现细节。这些注释可以作为源代码文档的一部分。

2. 编码规范：虽然编码规范不是直接的文档，但在编码阶段需要遵守和维护团队的编码规范。编码规范确保代码风格的一致性和可读性。

3. 单元测试文档：记录软件系统的单元测试用例、预期结果和实际结果，用于验证代码的正确性和健壮性。

B.8 源代码文档编写

1. 代码注释

（1） 在关键代码块、函数、方法和类的上方添加注释，简要解释其功能和用途。

（2） 使用自然语言描述代码的行为和意图，帮助读者理解代码的用途和预期结果。

（3） 解释输入参数和返回值的含义和格式，以便其他开发人员正确使用代码。

2. 函数和方法文档

（1） 对于每个函数和方法，编写文档说明其功能、输入参数、返回值和使用方法。

（2） 使用规范的注释标记（如 Java 中的 Javadoc、Python 中的 docstring）来定义和格式化函数文档。

（3） 提供示例代码和使用说明，以帮助其他开发人员正确调用和使用函数。

3. 类和模块文档

（1） 对于每个类和模块，编写文档说明其功能、设计目的和关键属性。

（2） 解释类之间的关系和依赖关系，以帮助其他开发人员理解代码的整体结构。

（3） 提供示例代码和使用说明，以展示类和模块的正确使用方法。

4. 文件头注释

（1） 在源代码文件的顶部添加文件头注释，包括文件名、作者、创建日期和简要描述。

（2） 提供版权声明和许可证信息，确保代码的合法性和合规性。

5. 变更记录

（1） 在源代码中记录对代码进行的重要更改和修复。包括修改日期、作者、变更类型和详细描述。

（2） 使用版本控制系统的提交消息或注释，或者在代码中添加特定的注释标记（如 TODO、FIXME）来跟踪变更。

6. 示例和说明

（1） 提供实际的示例代码，演示如何使用和调用源代码的不同功能和场景。

（2） 解释示例代码的输出和期望结果，确保其他开发人员理解和验证代码的行为。

7. 异常处理

（1） 对于可能引发异常的代码块，提供注释或文档说明处理异常的方式和逻辑。

（2） 解释异常类型、错误处理和恢复机制，以帮助其他开发人员正确处理和处理异常。

8. 代码结构和命名

（1） 使用清晰、一致的命名约定和代码结构，以提高代码的可读性和可维护性。

（2） 在注释中解释和说明自定义的命名规则或约定，以帮助其他开发人员理解代码的结构。

B.9 编码规范编写

1. 清晰明确：确保编码规范的规则和指导准确、清晰明确。使用简洁明了的语言和术语，避免歧义和模棱两可的描述。

2. 统一风格：编码规范应确保代码风格的一致性。规定代码缩进、命名规则、注释规范、代码结构等方面的规则，以确保代码易读、易维护。

3. 可读性和可理解性：强调代码的可读性和可理解性。建议使用有意义的命名、注释和文档，以及清晰的代码结构和逻辑。

4. 命名规则：规定变量、函数、类和文件等的命名规则。建议使用有意义的名称，避免使用缩写和不明确的命名。

5. 缩进和格式化：规定代码缩进的方式和格式化的规则。建议使用一致的缩进风格，如空格还是制表符，并规定代码块的大括号位置和换行规则。

6. 注释规范：规定注释的格式和使用规则。建议提供函数和方法的注释，解释其功能、输入参数、返回值和使用方法。注释还可以用于解释复杂的算法和处理逻辑。

7. 异常处理：规定异常处理的标准和实践。建议捕获并处理可能出现的异常情况，以确保代码的健壮性和可靠性。

8. 最佳实践：推荐使用编程语言和平台的最佳实践。避免已知的编码陷阱和反模式，以提高代码质量和性能。

9. 文件组织：规定代码文件的组织结构和目录命名规则。建议按照功能、模块或层次进行文件组织，以便于代码的查找和维护。

10. 版本控制：规定代码版本控制的策略和实践。建议使用版本控制工具，并定义提交和分支策略。

11. 可扩展性和可维护性：考虑到未来的需求变化和代码的可扩展性。规定代码模块化、接口设计和可测试性等规则，以支持代码的持续演进和重用。

12. 静态分析和代码审查：建议使用静态代码分析工具进行代码质量检查。推荐定期进行代码审查，以确保代码符合编码规范和最佳实践。

B.10 单元测试文档编写

1. 文档概述：在文档的开头，提供对文档的概述和目的的简要描述。说明文档的受众和所关注的重点。

2. 测试目标和范围：明确单元测试的目标和范围。描述要测试的模块、函数或类的功能和行为。

3. 测试用例设计：设计全面而有针对性的测试用例。覆盖不同的测试场景、边界条件和异常情况。确保测试用例具有可重复性和独立性。

4. 输入和预期输出：对于每个测试用例，明确输入数据、参数设置和预期的输出结果。确保预期结果符合预期行为和预期值。

5. 测试环境和依赖项：描述测试环境的配置和依赖项。包括测试框架、测试数据和测试工具等。

6. 执行步骤和方法：提供测试用例的执行步骤和方法。描述如何准备测试环境、执行测试用例和记录测试结果。

7. 异常处理和错误报告：讨论在测试过程中遇到的异常情况和错误处理方式。描述如何记录和报告错误，以便开发人员进行修复。

8. 期望覆盖率：明确所需的代码覆盖率目标。根据项目的要求确定满足可靠性和质量标准的最低覆盖率。

9. 测试结果和统计：记录测试的执行结果和统计数据。包括测试通过的用例数量、失

败的用例数量、覆盖率等。

10. 维护和更新：及时更新单元测试文档，以反映代码和需求的变更。确保文档与实际代码的一致性。

B.11 测试文档分类

软件测试文档通常包括以下内容：

1. 测试计划：描述测试的范围、目标、资源需求、进度安排、测试阶段（单元测试、集成测试等）和测试策略。测试计划还确定测试的方法和技术，例如手动测试或自动化测试。

2. 测试用例：测试用例是一组明确的步骤和输入，用于验证软件的特定功能或特性。测试用例描述了预期的结果和实际结果之间的差异。

3. 缺陷报告：缺陷报告用于记录在测试过程中发现的问题或错误。它们描述了缺陷的详细信息，例如缺陷的类型、严重程度、复现步骤和环境条件。

4. 测试结果：测试结果文档记录了每个测试用例的执行结果，包括通过的用例、失败的用例和跳过的用例。测试结果文档还可以包括非功能性测试，如性能测试，安全测试，压力测试，稳定性测试等的结果。以及测试验证和确认。

5. 测试环境配置：测试环境配置文档描述了在测试期间使用的硬件、软件和网络配置。它确保测试团队使用相同的环境进行测试，并提供必要的背景信息。

6. 测试验证和确认：测试验证和确认文档用于记录测试团队和利益相关者之间的讨论和批准。它们可以包括测试计划、测试用例和测试结果的确认。

7. 测试报告：测试报告用来汇总不同测试阶段/类型的测试整体状态，包括测试的覆盖率，测试通过率，测试进展，失败测试分析以及影响测试进展的重点缺陷等信息。测试报告用于测试团队和利益相关者之间的讨论打合批准。

B.11.1 测试文档编写

测试文档的目的是确保测试过程的可追溯性和透明性。它们提供了对软件质量和可靠性的评估，帮助团队识别和解决问题，并最终改进软件的质量。

编写测试文档时需要遵循一些原则，以确保文档的质量和有效性。通常包括以下编写原则：

1. 清晰和简洁：测试文档应该清晰明了，使用简洁的语言和结构，以便读者能够轻松理解文档的内容。避免使用过于复杂的术语和技术语言，除非在必要的情况下提供适当的解释。

2. 完整和详细：测试文档应该提供足够的信息，以便读者能够理解测试的目的、步骤和预期结果。确保测试用例覆盖了所有关键的功能和场景，并描述了预期的结果和实际结果之间的差异。

3. 可追溯性：测试文档应该具有良好的追溯性，即能够追踪到测试需求、测试用例和测试结果之间的关系。确保每个测试用例都与相关的需求或功能进行关联，以便能够准确地跟踪测试覆盖和测试进度。

4. 一致性：在编写测试文档时，保持一致性非常重要。使用相同的术语、格式和结构，以便读者能够更容易地理解和比较不同部分的内容。此外，确保测试文档与其他相关文档（如需求文档）之间的一致性。

5. 可更新性：测试文档应该是可更新的，以反映软件开发过程中的变化。及时更新测试文档，以便与软件的最新版本保持一致，并反映新的需求、改进或修复。

6. 可验证性：测试文档中的信息应该是可验证的，即其他团队成员或利益相关者可以独立验证测试的结果和结论。提供清晰的步骤、输入和预期结果，以便他人能够重现测试过程并验证测试的有效性。

这些原则有助于确保测试文档的质量和可用性，提高测试的效率和准确性。根据具体的项目和团队需求，还可以根据实际情况制定其他适用的原则。

B.11.2 测试计划编写

测试计划一般包括测试目标、 测试概要、 测试范围、 重点事项、 质量目标、 资源需求、 人员组织、 测试策略、 发布提交、 测试进度和任务人员安排、 测试开始/完成/延迟/继续的标准、 风险分析。

1. 测试目标：对测试目标进行简要的描述。

2. 测试概要：摘要说明所需测试的软件、名词解释、以及提及所参考的相关文档。

3. 测试范围：测试计划所包含的测试软件需测试的范围和优先级，哪些需要重点测试、哪些无需测试或无法测试或推迟测试。

4. 重点事项：列出需要测试的软件的所有的主要功能和测试重点，这部分应该能和测试案例设计相对应和互相检查。

5. 测试沟通：包括测试解脱的沟通汇报机制，关键测试干系人分析，沟通的方式（会议，报告或者数字化工具展现）等.

6. 质量目标：制定测试软件的产品质量目标和软件测试目标。

7. 资源需求：进行测试所需要的软硬件、测试工具、必要的技术资源、培训、文档等。

8. 人员组织：需要多少人进行测试，各自的角色和责任，他们是否需要进行相关的学习和培训，什么时候他们需要开始，并将持续多长时间。

9. 测试策略：制定测试整体策略、所使用的测试技术和方法。

10. 发布提交：在按照测试计划进行测试发布后需要交付的软件产品、测试案例、测试数据及相关文档。

11. 测试进度和任务人员安排：将测试的计划合理的分配到不同的测试人员，并注意先后顺序.如果开发的Release不确定,可以给出测试的时间段.对于长期大型的测试计划，可以使用里程碑来表示进度的变化。

12. 测试开始/完成/延迟/继续的标准：制定测试开始和完成的标准；某些时候，测试计划会因某种原因（过多阻塞性的Bug）而导致延迟，问题解决后测试继续。

13. 风险分析：需要考虑测试计划中可能的风险和解决方法。

B.11.3 缺陷报告编写

缺陷报告是在软件测试过程中记录和报告发现的问题或错误的文档。它用于描述在测试过程中发现的缺陷、缺陷的详细信息以及缺陷的严重程度。

缺陷报告通常包括以下内容：

1. 标题和摘要：缺陷报告应该具有一个清晰的标题，用于概括缺陷的主要问题。摘要部分提供对缺陷的简要描述，通常包括缺陷的现象或错误行为。

2. 缺陷描述：缺陷报告应该提供对缺陷的详细描述，包括如何重现缺陷、在何种环境下出现、测试使用的软件版本、影响的范围以及对软件功能的影响。描述应该具体明确，以便开发人员能够理解问题的本质并进行修复。

3. 复现步骤：缺陷报告应该提供重现缺陷所需的详细步骤。这些步骤应该清晰明了，使开发人员能够在其开发环境中重现缺陷并进行调试。

4. 环境信息：缺陷报告应该包括相关的环境信息，例如操作系统、浏览器、设备类型等。这些信息有助于开发人员在特定环境中定位和修复缺陷。

5. 严重程度和优先级：缺陷报告应该评估缺陷的严重程度和优先级。严重程度指的是缺陷对系统功能或性能的影响程度，而优先级表示修复缺陷的紧急程度。

6. 相关附件：如果有必要，缺陷报告可以包括相关的截图、日志文件或其他附件，以提供更多的上下文和支持信息。

缺陷报告的目的是记录和报告软件中发现的问题，以便开发人员能够了解问题并进行修复。良好的缺陷报告能够提供清晰的信息和重现步骤，有助于加快缺陷修复的过程，并提高软件质量。

B.11.4 测试结果编写

测试结果文档是记录软件测试过程中执行的测试用例及其结果的文档。它提供了关于每个测试用例的执行情况、通过或失败的结果以及相关的详细信息。

测试结果文档的目的是提供对测试活动的全面概览和汇总，使团队成员和利益相关者能够了解测试的覆盖范围、执行情况和结果。它还有助于追踪测试进展、确定需要进一步调查或修复的问题，并为软件质量提供评估和决策的依据。测试结果文档通常包括以下内容：

1. 测试用例标识：每个测试用例都有一个唯一的标识符或编号，以便在文档中进行跟踪和引用。

2. 测试用例描述：对于每个测试用例，测试结果文档应该提供清晰的描述，描述了测试的目的、输入、预期结果和任何特定的测试条件。

3. 测试结果：记录每个测试用例的执行结果，包括通过、失败或跳过。对于失败的测试用例，还应该提供详细的错误信息和相关的日志。

4. 错误分类和严重程度：对于失败的测试用例，可以将错误进行分类，并指定其严重程度。常见的分类包括功能缺陷、性能问题、安全漏洞等。

5. 环境信息：记录测试用例执行时使用的测试环境的相关信息，包括操作系统、浏览器版本、设备类型等。这有助于后续的环境复现和问题定位。

6. 备注和附加信息：在测试结果文档中，可以提供任何其他相关的备注、问题描述、重要观察或附加信息，以增加文档的完整性和可读性。

B.11.5 测试环境配置编写

测试环境配置文档是记录和描述测试环境配置的文档。它提供了关于测试环境的详细信息，包括硬件、软件和网络配置，以及其他必要的设置和准备工作。

测试环境配置文档的目的是确保测试团队使用相同的环境进行测试，并提供必要的背景信息和指导，以确保测试的准确性和可重复性。它还有助于解决测试环境相关的问题，并为测试团队和开发团队提供清晰的环境配置说明。

测试环境配置文档通常包括以下内容：

1. 硬件配置：记录测试环境中使用的计算机、服务器、网络设备等硬件的详细配置信息。这包括处理器、内存、存储容量等硬件规格。

2. 软件配置：描述测试环境中所需的软件组件和版本。这包括操作系统、数据库、应用服务器、浏览器等软件的名称、版本号和安装路径。

3. 网络配置：记录测试环境中的网络拓扑和配置，包括 IP 地址、子网掩码、网关、DNS 服务器等。这对于测试涉及网络通信的功能非常重要。

4. 测试工具和框架：列出在测试环境中使用的测试工具和框架，例如自动化测试工具、性能测试工具、代码覆盖工具等。提供它们的名称、版本和安装路径。

5. 数据配置：描述测试环境中所需的数据设置和准备工作。这包括数据库配置、测试数据集、数据文件路径等信息。

6. 环境变量和配置参数：记录测试环境中的环境变量、配置参数和设置。这包括任何需要在测试环境中设置的参数或选项。

7. 其他设置和要求：在文档中提供任何其他测试环境设置和要求的相关信息，例如访

问权限、安全设置、许可证配置等。

B.11.6 测试验证和确认编写

测试验证和确认文档是用于确认和验证软件测试活动的文档。它提供了关于测试执行、结果和确认过程的信息，以确保测试的有效性和准确性。

测试验证和确认文档的目的是提供对测试活动的确认和验证的记录。它确保测试过程和结果的正确性，并提供准确的信息和证据，以支持软件质量评估和决策。这些文档对于项目团队和利益相关者了解测试活动的情况、测试结果的可信度以及进一步的测试计划和改进提供重要的参考。

测试验证和确认文档通常包括以下内容：

1. 测试执行结果：记录每个测试用例的执行结果，包括通过、失败或跳过。这些结果可以通过执行自动化测试脚本或手动测试来获得。

2. 验证测试覆盖：确认测试用例是否覆盖了软件需求或功能的各个方面。验证测试用例是否全面、恰当地涵盖了所需的测试范围。

3. 缺陷处理状态：跟踪已发现的缺陷并记录其处理状态。这包括缺陷的修复情况、验证缺陷修复的结果以及缺陷的关闭状态。

4. 验证测试环境：确认测试环境的正确配置和准备情况。验证测试环境是否符合测试要求，并确保测试环境的可用性和稳定性。

5. 确认测试数据：确认测试数据的准备和使用情况。验证测试数据是否符合需求，以及测试数据的正确性和完整性。

6. 验证测试工具和框架：确认所使用的测试工具和框架的有效性和正确配置。验证测试工具是否准确地执行测试，并提供准确的结果和报告。

7. 验证测试报告和文档：确认测试报告和文档的准确性和完整性。验证测试报告是否包含正确的测试结果、统计信息和相关的补充信息。

软件研发质量管理体系建设白皮书

RONG

管理社区

Product & Project Management

扫码加入群聊

融管理社区专家团队撰写