第3章 软件质量工程体系

质量管理体系作为持续发展议程中经济增长的构成要素之基础，着实担纲了举足轻重的角色，然而，相较于近年来广受重视的环境整合和社会公平等较为热门的议题，却经常被忽视。

——摘自《ISO 9001：2015 未年 25 年的质量管理标准》

从最早通过简单的手工检验进行质量控制，发展到以统计学为基础的控制理论和控制技术，及后来的质量保证手段、全面质量管理思想等，质量的管理水平在不断地提高。但是，如果不能系统地建立一套有效的管理体系，这些质量的控制技术、预防措施、评审活动等不能真正发挥作用。软件开发是以个人智力为基础的、有组织的团队性活动，这使软件质量变为一项复杂系统工程问题，我们必须用系统方法研究它。

借助系统工程学、管理学等理论，把质量控制、质量保证和质量管理有效地集成在一起，形成现代软件质量工程体系，是当今质量管理的发展趋势，也是真正改善软件质量的最彻底、最有效的方法。

工程的概念在传统领域应用有相当长一段历史，从一千多年前的水利工程到后期的铁路、公路、建筑等工程，它一直被使用着，而且逐渐形成一套科学体系，即系统工程学。系统工程学是系统学和系统方法论在工程领域的应用，是组织管理系统的规划、研究、设计、制造和使用的科学方法，并用定量和定性相结合的系统方法处理大型复杂系统的问题，内容涉及工程项目的计划、时间周期管理、成本管理和风险管理等。软件工程学发展比较迟，在 20 世纪 60—70 年代，不得不借助传统工程项目的管理经验和实践，解决软件出现的危机，避免软件项目开发经常出现延期、开发经费远大于预算、软件质量差等各种糟糕情况。系统工程学的理论与软件工程的理论有着紧密的联系，可以说系统工程理论是软件工程理论的基础。

软件质量管理的困难性，主要是由软件特点——规模大、软件内部构成复杂、难以度量等造成的，我们是否也可以引进系统工程方法克服这些困难、获得更有效的软件质量管理呢？回答是肯定的。软件质量所存在的问题不仅是管理的问题，而且是工程的问题，需要系统地解决问题。

3.1 系统工程学的思想

进行大型项目或复杂问题的实施和解决，一般需要按照系统工程学的理论进行，即将整个项目或问题作为一个系统，用系统论的思想和系统方法论的技术分析、规划、设计和实施，以保证项目或问题的解决方案和计划得到更为有效、彻底地执行。

系统工程学是以系统论的思想和系统方法论为基础，借助控制论、运筹学、统计学、信息处理和计算机技术，研究复杂系统的构成和子系统的相互作用，对系统的构成要素、组织结构、信息流动和控制机制等进行分析，并建立相关的数学模型或逻辑模型，从而掌握该系统随时间推移而产生的行为模式。系统工程学把系统的行为模式看成是由系统内部的信息反馈机制决定的。通过建立系统工程学模型，可以研究系统的结构、功能和行为之间的动态关系，以便寻求较优的系统结构和功能。

系统分析在系统工程学中占据着相当重要的位置。把一个项目或问题看作一个系统，以系统的方法去完整地、全面地分析对象，而不是零星地处理问题。这就要求人们必须考虑影响系统的各种因素，而且了解这些因素动态的、变化的规律及相互之间存在的关系。

系统工程主要是沿着逻辑推理的路径，去解决那些原本靠直觉判断处理的问题。根据实践经验，可以将系统分析过程概括为如图 3-1 所示的逻辑结构。它包括 5 个环节：问题定义、分析问题、预测未来变化、建模和计算、方案评估，整个过程可归纳成问题说明、解决方案的策划和评估结果 3 个阶段。问题说明阶段的工作成果是提出目标，确定评价指标和约束条件；解决方案策划阶段提出各种备选方案并预计一旦实施后可能产生的结果；最后的评估阶段是将各个备选方案的评价比较结果提供给决策者，作为判断抉择的依据。

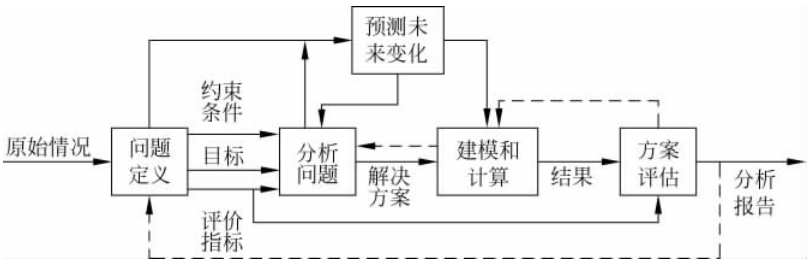

图 3-1　系统分析过程逻辑结构

1. 问题说明阶段

需要分析研究质量问题的来源、产生过程、约束条件和影响因素等，完成质量问题的定义，包括质量问题解决的目标(如将缺陷率降低 30%)、评价解决问题方案的具体指标，形成问题分析报告，主要包括以下两项内容。

- 问题性质，包括问题的结构、影响范围、形成过程和未来可能发展的势态。
- 问题条件，问题解决所需的资源，依赖于问题的性质。

问题说明阶段的工作决定着今后的分析过程，如问题的解决方案、构造的模型和某种结果是否可行等。所以，问题分析报告很重要，一定要将问题性质分析清楚，不能只看到问题的表面现象，应该追溯到问题的根源。为了保证这一阶段的成果，需要对问题性质和问题条件是否匹配做检验、审查，从而使工作任务和所需资源相当，达到一个相对平衡状态，而不会形成一头重一头轻的结果。例如，任务太重而缺乏资源，目标是不可能达到的。

2. 解决方案的策划

解决方案的策划是指方案提出和筛选的过程。策划方案是为了达到所提出的目标，一般要具体问题具体分析。通常，良好的解决方案应具备以下特性。

- 适应性。目标经过修正甚至变动较大的情况下，原来方案仍能适用。这在不确定因素影响大的情况下尤为重要。

- 可靠性。可靠性是指系统在任何时候正常工作的可能性,即使系统出错、失效也能迅速恢复正常。
- 可操作性,即方案实施的可行性。决策者支持与否是关键,不可能得到支持的方案必须取消。

总之,进行良好系统分析是取得良好解决方案的基础。在系统分析过程中自始至终要意识到,需要而且有可能发现新的更好的解决方案。

3. 评估和比较备选方案

工程问题不是数学问题,一般不会只有一个解决方案,而是可以找出多个解决方案,然后根据一定的评估准则,选出更优或最优的方案。根据评选的方法(如"成本-效益分析""成本-利润分析"法等)以及问题定义时确定的评价指标,对不同解决方案运行的结果进行评估分析,选择最为可行的一种或两种方案报给决策者。

3.2 软件质量工程体系的构成

建立软件质量工程体系(SQES)之前,应先了解传统的质量管理体系,然后基于这个质量管理体系,结合系统工程、软件工程等学科,建立现代的软件质量工程体系。

传统的质量管理体系能够帮助组织提高顾客满意度,鼓励组织分析顾客要求,规定相关的过程,并使其持续受控,从而能够持续提供满足顾客要求的产品。质量管理体系能提供持续改进的框架,以增加顾客和其他相关方满意的机会,也就是使质量管理过程成为一个持续改进的过程,这也是系统工程学的一个基本目标——有良好的反馈机制,即通过设定顾客满意度作为管理体系的质量目标,顾客的需求则是系统的约束条件,对系统中的资源再分配、质量功能进行调节等,以便寻求质量管理体系越来越优化的结构和功能。为了使组织有效地运行这个持续改进的过程,必须识别和管理许多相互关联和相互作用的过程。由国际标准ISO 9000或国内标准GB/T 19000所表述的、以过程为基础的质量管理体系模式如图3-2所示。

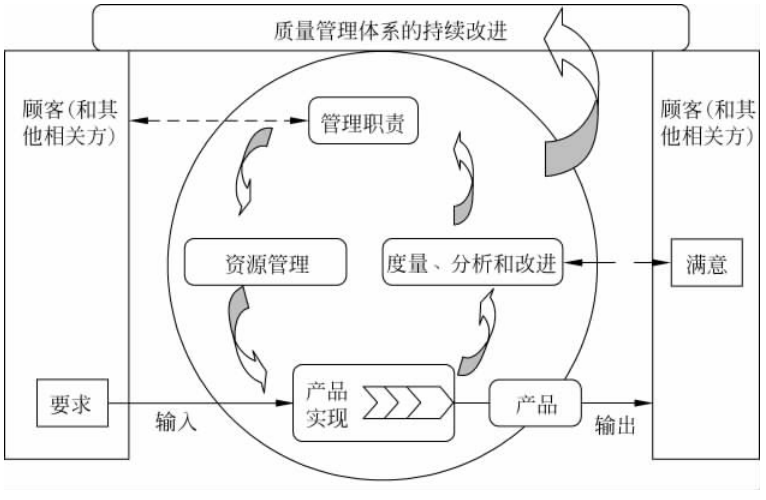

图3-2 以过程为基础的质量管理体系模式

3.2.1 通用的软件质量工程体系

上面简单地分析了软件质量的管理体系，而软件质量工程体系需要从工程的视角考虑软件工程特点，即软件的开发流程、开发技术、项目管理等特点，如：

- 明确相关的质量标准，建立组织特定的质量目标；
- 明确产品自身质量属性中需要特别关注的质量特性；
- 确定实现质量目标必需的过程和职责；
- 了解软件技术现状及其发展趋势；
- 确定和提供实现质量目标必需的资源；
- 确定防止不合格并消除产生原因的措施；
- 应用这些度量方法确定每个过程的有效性和效率；
- 建立和应用持续改进软件开发流程。

运用系统科学，将软件质量管理视作一个系统，关注系统的输入、输出和外部环境，不断收集软件产品和过程的质量信息及其反馈，然后进行调控和优化。虽然在图 3-2 中，其输入只有客户及其相关方的需求，但在软件质量工程体系中，要关注软件项目的上下文，包括项目影响因素（如团队、预算、资源、进度、风险等）、软件产品自身特点（如行业、规模、采用的技术框架等）、软件工程环境（如组织、文化、软件开发的基础设施等）和团队已经掌握的软件研发方法和技术等。其输出是产品，对顾客有价值的功能特性或服务，让顾客及其相关方满意，同时还要考虑企业自身可持续的发展，如团队的发展、经验的积累，包括及时减少或消除技术债务、提升产品的易维护性等，最终实现对软件质量进行全面、综合的系统性管理。

讨论软件质量工程体系构成之前，先讨论软件质量管理的层次性。项目的质量管理依赖于组织的质量方针、软件质量标准和规范以及与之配套的培训体系、技术、工具、模板等，事先定义质量标准与规范，建立良好的流程和培训体系，提供成熟的质量管理技术、工具和各种文档模板等，都能预防软件缺陷的产生，从这个意义上看，项目质量管理更多地体现了“软件质量保证”，不过，有些技术、工具是为了质量控制。软件质量管理最终需要落实到项目上，因为软件产品的交付是由项目团队完成的。在团队这一层更多体现质量控制，如研发团队中主要的质量活动是软件测试——属于事后检查，归为质量控制。但在项目中，也会进行过程评审、缺陷分析等活动，这些活动可以看作“质量保证”。正如 CMMI 提倡的组织、团队和个人的 3 个层次，这里将软件质量工程体系也分为组织层次、项目层次和质量工程基础设施 3 个层次，如图 3-3 所示。

1）组织层次

软件质量保证主导的层次。在软件质量方针的指导下，选择或参考国际、国内的软件质量标准和规范，建立组织的软件质量工程规范，建立良好的研发流程及其各种评审活动准则，以及如何做好缺陷预防。软件质量工程规范规定了一系列的质量活动，这些活动必须遵守相应的流程，约束软件开发人员的行为模式，创建所需的软件质量管理技术、工具、检查表和模板等。在执行质量标准或规范过程中，首先要对全员进行培训，过程中要进行常规性的评审，及时发现问题以求改进。持续改进是软件质量管理一个永恒的主题，可以参考相应的改进模型、国内外标准等，从而不断改进已建立的软件质量工程规范。

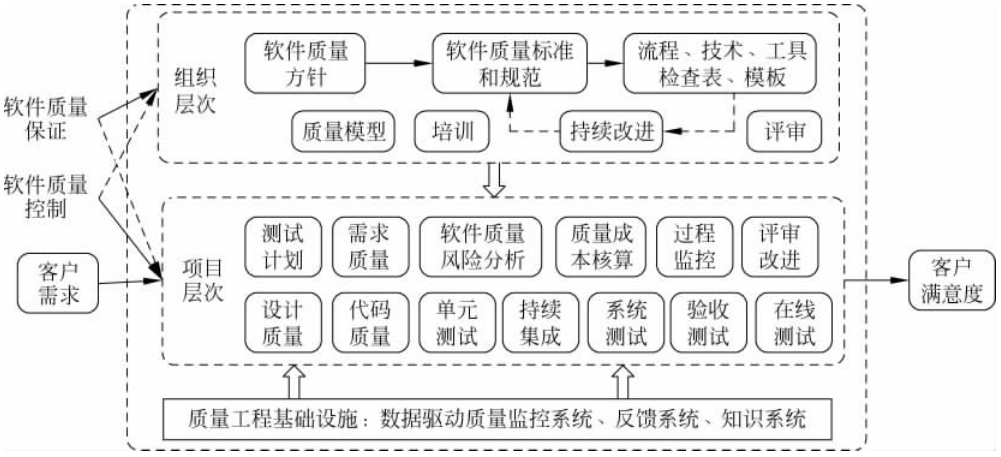

图 3-3　软件质量工程体系的层次

2）项目层次

软件质量控制主导的层次。在组织层次的软件质量保证的工作基础上，加强项目的质量控制，依据软件质量工程规范、质量模型，做好软件测试、软件质量风险控制、过程监控等工作，通过对影响质量各种因素的分析，了解可能存在的质量风险，加以回避、控制。过程监控重点是检查活动或阶段的入口与出口准则，确保研发活动遵守相应的工程规范。在目前的软件工程实践中，软件测试是软件质量控制的主要活动之一，覆盖全生命周期，包括单元测试、集成测试、系统测试、验收测试等；软件质量更依赖于其构建的过程，包括需求定义、设计和代码的质量，这其中也包括对需求、设计和代码等的分析与评审。

3）质量工程基础设施

质量工程基础设施相当于软件质量管理平台，在这平台上更好地实现软件质量的管理。从软件产品及其研发过程看，也具有传统的建筑工程、土木工程或机械工程等所没有的一些特性，如持续迭代、代码重构与复用、版本控制及其相关的配置管理等。在软件质量工程中如何适应快速迭代的需求、如何及时获取用户的反馈，是需要特别考虑的一些方面，即建立软件质量工程的基础设施。如今软件技术发展很快，包括容器技术、云计算、数据可视化技术等，可以利用这些技术改善软件研发质量管理的实时性、可视化(透明)和效率。同时，借助这样的平台，研发人员可以及时获得用户的反馈，及时改进产品的质量，可以随时随地获得需求质量、设计质量、代码质量和测试等相关知识。

我们可以借助软件质量工程体系，揭示软件质量方针、质量标准、质量策划、质量保证和质量控制之间的关系，使软件开发人员或质量管理人员有章可循，清楚组织的质量工作框架和自己在框架中的位置，然后与团队其他成员协同做好质量工作，交付高质量产品。

质量目标相对抽象，从组织上看，需要依据产品质量模型，综合考虑软件质量影响因素及其之间的关系，对质量目标进行分解，形成相对明确、具体的质量指标。每个测试项目会根据这些质量指标的优先级，进行适当的筛选，建立项目的质量目标，从而指导质量计划、测试计划的制订。例如，确定了软件质量指标及其影响因素，就可以针对性地制订相应的对策，消除或降低某个因素的消极影响，或者提高某个因素的积极影响，从而提高软件质量。在整个软件研发过程中，执行质量计划或测试计划，并根据上下文变化与反馈，调整质量计

划或测试计划，直至交付客户满意的软件产品或软件服务。

如果不能定量地确定软件产品质量属性、软件研发过程质量属性，就无法定义可验证的质量目标，质量目标是否实现也就无法验证。从系统方法论说，系统工程学是系统的结构方法、功能方法和历史方法的统一。也就是说，软件质量工程体系，不仅体现在对质量管理系统的结构划分、结构分析、质量功能展开上，而且基于质量的历史数据可以建立质量的预测或评估模型，包括软件的可靠性评估模型，从而实现软件质量的可预见性。这样，我们能够更有效地控制和管理软件质量风险。软件质量目标的定义与验证、质量预测模型等都建立在软件度量的基础上，软件质量度量在软件质量工程体系中扮演着重要角色，软件本身就是数字化产品，容易实现其度量及其管理。基于软件度量，我们能够有效地实施、控制并优化质量管理体系，形成优化的质量结构和组织功能体系，从而提升软件开发过程的质量控制、管理能力，不断改进软件开发过程，按质按量完成软件项目，以实现软件质量目标。

概括起来，从系统工程的角度描述质量管理体系如图 3-4 所示。软件质量工程体系的思想是从系统工程学、软件工程理论出发，沿着逻辑推理的路径，对软件质量的客户需求、影响软件的质量因素、质量功能结构、问题根源等进行分析，以建立积极的质量文化、构造软件质量模型，基于这些模型研究相应的软件质量标准和软件质量管理规范，并配以相对应的质量分析技术、工具等，把质量控制、质量保证和质量管理有效地集成在一起，降低质量成本和质量风险，从而系统地解决软件质量问题，形成现代软件质量工程体系。

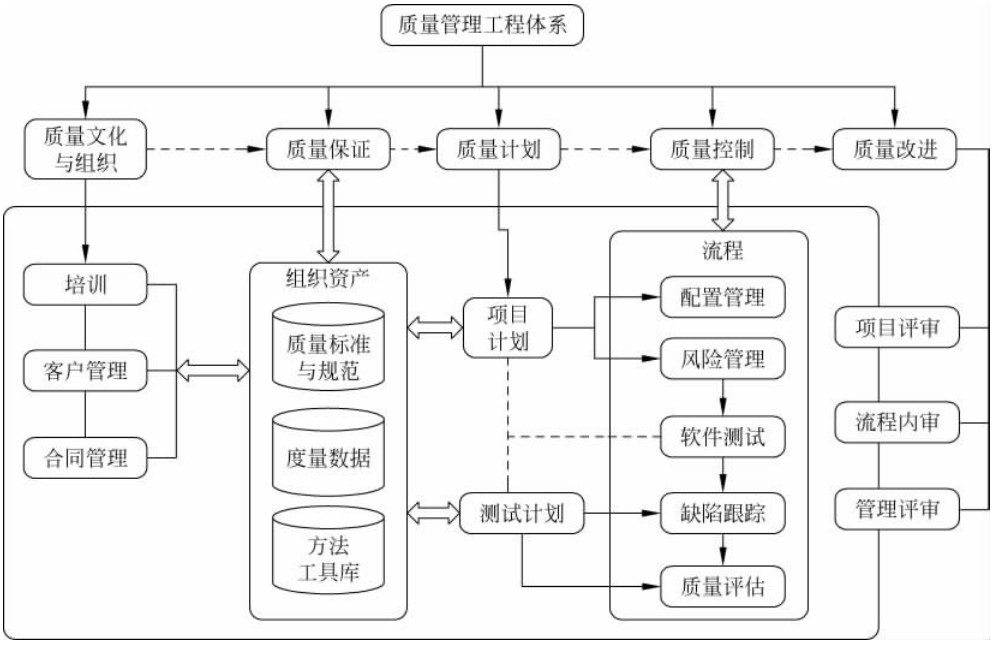

图 3-4　通用的软件质量工程体系及其构成

3.2.2　软件质量工程体系和管理体系的关系

简单地说，软件质量管理体系侧重管理，更通用、更抽象些；而软件质量管理工程体系，更侧重工程实践，涉及软件产品的研发与运维的各个环节，对质量管理落地实施提供具体的

工程实践指导。

质量管理体系集中在管理方面,其核心是管理组织、文化和流程;其基础是现有的软件质量标准、方法和工具。质量管理体系主要强调两个方面的内容,具体如下。

- 体系中的上层建筑:质量文化、上层领导的重视及对全面质量的承诺、有效的沟通和持续的改进等。
- 体系中的运行基础:软件质量管理组织(SQA小组)、软件质量标准(ISO 9000系列、CMMI等)、质量管理的流程、质量管理方法(抽样调查或问卷调查等)和质量管理工具(排列图、因果图、散布图等)。

这里论述的软件质量管理工程体系,着重从软件工程、系统工程学的角度构建质量、管理质量,在有限的资源情况下,获得最好的质量效益。其主要内容表现在:

- 将软件质量视为一个系统,深入了解软件质量的构成和结构,建立软件质量模型,应用于软件工程中。
- 软件质量策划,如同项目计划,定义软件质量管理要实现的目标、范围和方法。多数情况下,实际工作中的软件质量计划体现在软件测试计划中。
- 质量成本的分析,如何降低由低质量造成的成本。
- 软件质量风险的识别、分析与防范,这是贯穿软件生命周期的工作,与软件测试结合起来,基于风险的测试策略是最常用的策略。
- 软件质量度量,为质量管理提供客观的信息,清楚质量现状和改进的效果。

软件质量工程体系作为系统对象,具有系统的一些其他特性,包括可控性、目标性、柔性等。提出"软件质量工程"的概念,不仅能增强人们在质量管理中的"工程意识",考虑利益相关者、风险、成本、进度等,而且会促进人们采用系统工程的方法解决问题。在软件质量评价中,会采用定性和定量结合的方法:在质量目标管理、过程管理中采用统计工具、控制技术等;在软件质量成本管理中,采用指标分析、费用估算和偏差分析法。

软件质量管理不仅可以自成体系,还是软件工程理论的一部分。软件质量管理在自成体系中遵守系统学原理、采用系统方法论。同时软件质量工程体系作为软件工程理论的一部分,可以看作是软件工程体系的一个子系统,与时间管理、成本管理、风险管理并存,并相互作用。在一个组织中,软件质量管理体系占有核心的位置,对其他各项管理工作具有指导和约束作用。时间管理、资源管理、成本管理都要服从质量管理。但在一个项目中,质量管理又是项目管理的一部分,项目总体目标中包含质量目标、进度目标、成本目标等。质量、时间、资源和成本之间相互制约、相互影响,要在这些项目的关键目标上力求平衡。

3.2.3 根据上下文构建自己的软件质量工程体系

软件质量工程体系(SQES)处在不同行业、不同公司的软件研发环境中,一般会受行业特点、公司文化和组织行为模式等的影响。这种影响有多大,会产生怎样的后果?这里所说的行业特点、公司文化和组织行为模式等,可以理解为建立软件质量工程体系的上下文,上下文因素可能不局限在这几个方面,还可能包括以下内容。

- 组织规模:大型、中小型企业。
- 企业性质:国企、民企或外企等。

- 研发流程：如敏捷开发、精益开发、瀑布模型等。
- 产品类型：如性命攸关系统、使命攸关系统、一般商业系统。
- 团队能力、地域等。

上面我们已建立了一个通用的 SQES，如图 3-4 所示。如何根据上下文建立适合自己组织的 SQES？综合考虑这些上下文因素，分析哪些因素会比较显著地影响 SQES，哪些因素影响很弱，分析哪些因素会产生积极影响，哪些因素会产生负面影响。例如，大型组织比较复杂，更需要严格的、全面的制度和流程提供质量保证；而小型企业可以更灵活些，更多地依赖团队沟通、协作提高质量。技术能力强的团队，可以加强软件质量管理的基础设施的建设，通过持续集成、实时监控系统更好地控制质量、快速提供质量反馈。

在工业界，一方面强调构建高质量的需求、设计和代码，加强业务需求评审、系统需求评审、架构设计评审、组件设计评审等；另一方面通过构建软件测试体系保证质量、控制质量。图 3-5 展示了 IBM 公司的软件质量工程体系，分为以下 6 个层次。

（1）建立适合自己的、先进的软件开发流程。

（2）加强阶段性成果的评审，保证产品质量。

（3）综合运用测试方法，采用合适的测试技术，提升软件测试的效率和质量。

（4）抓好每一个测试环节，让缺陷无处藏身。

（5）不仅覆盖各种软件质量特性的验证，而且覆盖业务、软件维护等所需的测试。

（6）部署所需的测试平台，善于使用各种测试工具。

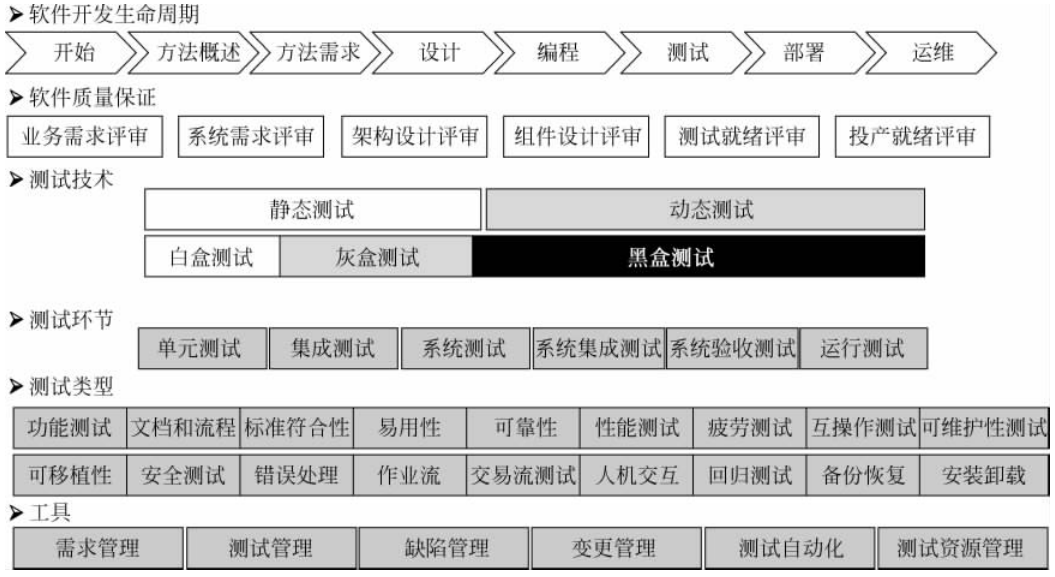

图 3-5　IBM 公司的软件质量工程体系

埃森哲（Accenture）公司根据测试左移的质量方针构建自己的软件质量工程体系，如图 3-6 所示。整个软件开发生命周期分为前期和后期。前期重心放在“缺陷预防”，推动软件自身的构建质量升级，基于测试准则分析为早期阶段提供建设性的输入，推荐更优秀的研发实践。后期重心放在“软件测试”上，驱动团队尽早发现缺陷，并根据缺陷分析、测试过程评估和测试结果分析，不断提高测试覆盖率，更好地保证测试的充分性。在研发

过程中,将质量融入测试的每一个环节,强调计划性、规范性和可管理性,具体体现在如下方面。

- 计划性:质量管理计划、需求计划、环境计划、测试计划等。
- 规范性:研发过程标准化、测试过程标准化、准则标准化、测试入口准则、测试退出准则、测试服务水平 SLAs。
- 可管理性:需求跟踪矩阵、变更管理、版本控制、发布管理、风险管理、测试管理、测试度量集成、持续评估、及时总结和报告等。

同时,强调借助工程技术更好地支撑软件质量的建设,包括需求和设计转化、低耦合架构、持续集成、自动部署、自动化测试等。

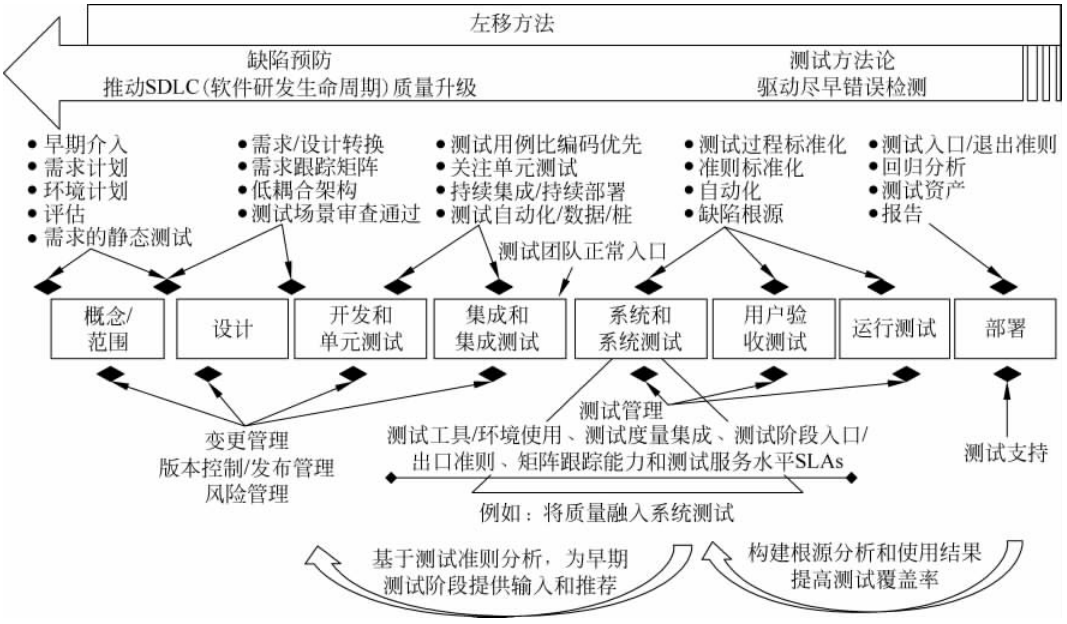

图 3-6　埃森哲公司的软件质量工程体系

3.3　软件质量工程环境

软件质量工程需要组织内良好的环境支撑,包括软环境和硬环境。软环境,主要指良好的组织质量文化并建立相应的质量机构服务于软件质量的保证与管理。良好的组织文化,实际上已在第 2 章做了充分介绍,主要有:

- 以顾客为中心的质量管理模式,只做对顾客有价值的工作。
- 缺陷预防,包括缺陷根本原因分析,建立相关的质量风险检查表。
- 全面质量管理,包括全员、全过程的质量管理。
- 零缺陷管理,指第一次就把事情做对、第一次就把工作做好。
- 六西格玛质量思想,指高要求、严要求,精益求精。

在组织上建立相应的质量管理机构,目前主要有下列 3 类机构。

(1) 软件质量保证(SQA)部门,如质量部、质量中心、SQA 工作组等,督促全体研发人员遵守已定义或引入的质量标准和已建立的软件过程规范,负责软件研发过程的规范性检查和审计,评审软件研发阶段性文档,帮助团队提升质量水平。部分软件公司没有 SQA 这样的机构,由软件测试部门或项目管理部门代替行使职责。

(2) 软件测试部门,如测试部、测试中心、测试组等,负责软件系统测试、验收测试,参与需求评审、设计评审和代码评审、单元测试、集成测试等,提供测试报告或软件质量评估报告。

(3) 软件过程改进组(SEPG),负责或协调软件过程的定义与改进。部分软件公司没有这样的机构,由 SQA 部门代替行使职责。

在硬件环境上,要建立软件质量管理(支撑云)平台,能够自动收集质量度量的相关数据,可以实时地展示软件质量状态,让软件质量管理具有良好的可视性和可追踪性。例如,实时收集软件缺陷数据,及时呈现缺陷的分布情况和趋势,随时掌控软件质量的状态,这就要求我们具有良好的缺陷跟踪与管理平台,即部署像 Jira、MantisBT 等这样的工具,构建缺陷管理平台。我们还可以采用代码静态分析工具(如 Findbugs、Checkstyle、PMD 等)、测试覆盖率分析工具(如 JaCoCo、JCover、Cobertura 等)、质量(缺陷)数据统计呈现工具(如 SonarQube)等进行更广、更深的质量管理。例如,SonarQube 可以度量缺陷、安全性漏洞、代码坏味道和覆盖率,如图 3-7 所示。达到质量要求,标示“通过”(passed)呈绿色(A 级)。如果达不到质量要求,则呈现淡绿(B 级)、黄色(C 级)、橘黄色(D 级)、红色(E 级),标示着质量问题越来越严重。它还可以把代码规模、复杂度等度量集成到一起,通过一个页面统一呈现。如果像图 3-7 中有 4 个 Bugs,还可以单击链接仔细查看。

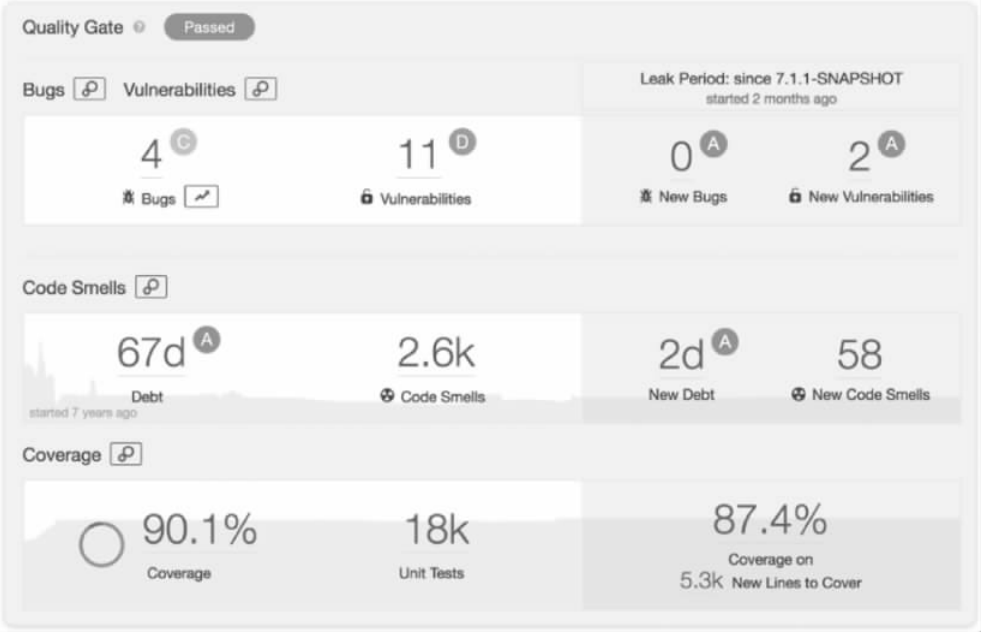

图 3-7 SonarQube 质量 dashboard

这部分内容还将在第 7 章和第 8 章进行更深入的讲解。从未来看,可以建成由 DevOps 工具链构成的软件工程环境,包括支持软件质量工程体系的软硬件环境。

3.4 依据质量标准有章可循

经过数十年的发展,软件行业形成的标准体系类别多、分工细,错综复杂。为了保证软件研发各项活动的科学性、一致性,需要采用相应的标准或规范指导工作,遵守业界的标准,更好地保证软件产品的质量。

质量标准体系是为了帮助某个行业和某类产品的生产开发过程而建立的文件体系,它包括全面管理所要求的操作规范、产品规格、活动流程,还包括强制执行的制度和指导性的思想和要求。从质量标准体系划分看,既可以横向分类,也可以纵向分类。

- 从纵向看,分为产品质量管理标准和过程质量管理标准。
- 从横向看,分为通用标准和各个行业的质量标准。
- 从范围看,分为国际标准和国内标准。

这里只讨论通用的软件质量标准体系,而不介绍其他行业的质量标准体系。

对于软件质量标准体系,主要分为软件产品、过程、技术、工具、人员和材料资源、数据、风险等几大类质量管理标准。对于每类质量标准,又可以进一步分为原理、要素标准和指南3个子类。

- 原理标准,描述各个原理级的关键组织标准。
- 要素标准,原理标准中的各个要素的详细性能要求,必须执行。
- 指南和补充,如何把原理或要素标准应用于特定场合而提供指导性文件。

在这大类基础上,有一个通用标准,主要描述软件工程领域所涉及的术语、组织框架和参考信息等。

例如,软件产品标准的结构如表3-1所示,软件过程标准的结构如表3-2所示。

表3-1 软件产品标准的结构

	产品特性		软件产品	产品文档	功能规格
原理	9126-1				
要素标准	TR 9126-2/3/4	15026	12119	9127	TR 14143-1/2
指南				18019	TR 14143-3/4/5

表中,9126指软件产品质量的模型和度量,15026:1998指系统和软件完整性级别,9127:1998指客户软件包的用户文档和覆盖信息,12119:1994指软件包质量需求和测试,14143-1:1998指软件测量功能性规格测量。

表3-2 软件过程标准的结构

	软件过程						系统过程
原理	12207/AMD1的过程结果						ISO/IEC 15288
要素标准	12207/14764	TR 15846	TR 16326	15939	14598	15910	15288标准部分
指南	TR 15271	ISO 9000-3	TR 9294			18019	15288指南

表中,12207:1995指软件生存周期过程,14764指软件维护,TR15846指软件配置管理,TR16326指软件工程项目管理,15939指软件测量过程,14598指软件产品评价,15910指软件用户文档过程,15271指ISO/IEC 12207使用指南,15288指系统生存周期过程。

3.4.1 标准的层次

根据制订软件工程标准的机构和标准适用的范围,可将其分为5个级别,即国际标准、

国家标准、行业标准、企业(机构)规范和项目规范。很多标准的原始状态可能是项目标准或企业标准,随着行业的发展与推进,这个标准的权威性可能促使它发展成为行业、国家或国际标准,因此这里所说的层次也具有一定的相对性。

1. 国际标准

一般由国际机构制订和公布供各国参考的标准为国际标准。这类机构包括国际标准化组织(International Standards Organization, ISO)、国际电工委员会(International Electrotechnical Commission, IEC)、电气和电子工程师协会(Institute of Electrical and Electronics Engineers,IEEE)等。ISO 具有广泛的代表性和权威性,所公布的标准也具有国际影响力。20 年纪 60 年代初,ISO 建立了"计算机与信息处理技术委员会"——ISO/TC97,专门负责与计算机有关的标准化工作。

ISO/IEC JTC1/SC7 发布的标准

ISO/IEC JTC1/SC7 是 ISO/IEC 第一联合技术委员会的第七分技术委员会的编号,成立于 1987 年,1991 年正式命名为"软件工程分技术委员会",2000 年更名为"软件和系统工程分技术委员"会。从成立至今,ISO/IEC JTC1/SC7 按照 ISO/IEC 严格的标准制订程序,正式制订出版了 180 多个 ISO/IEC 标准。这些标准中有几十个直接和软件质量相关,而且对质量管理工作具有很高的指导意义和参考价值。

2. 国家标准

由政府或国家级的机构制订或批准的适用于本国范围的标准。例如:GB(GuoBiao)——中华人民共和国国家技术监督局,是我国的最高标准化机构,它所公布实施的标准简称"国标"(GB)。ANSI(American National Standards Institute)——美国国家标准学会。它是美国民间标准化组织的领导机构,在美国甚至全球都具有一定权威性。它所公布的标准都冠有 ANSI 字样。FIPS 有(NBS)(Federal Information Processing Standards (National Bureau of Standards))——美国商务部国家标准局联邦信息处理标准。它所公布的标准均冠有 FIPS 字样,如 1987 年发表的 *FIPS PUB* 132-87 *Guideline for validation and verification plan of computer software* 软件确认与验证计划指南。其他的还有 BS(British Standard)——英国国家标准,JIS(Japanese Industrial Standard)——日本工业标准。

3. 行业标准

行业标准是由一些行业机构、学术团体或国防机构制订,并适用于某个业务领域的标准。例如:

- IEEE(Institute of Electrical and Electronics Engineers)——美国电气与电子工程师学会。该学会专门成立了软件标准技术委员会(SESS),积极开展软件标准化活动,取得了显著成果,受到了软件界的关注。IEEE 通过的标准通常会报请 ANSI 审批,使其具有国家标准的性质。因此,我们看到 IEEE 公布的标准常冠有 ANSI 字样,例如,ANSI/IEEE Str 828-1983 软件配置管理计划标准。
- GJB——中华人民共和国国家军用标准。这是由我国国防科学技术工业委员会批准,适合于国防部门和军队使用的标准。例如,1988 年发布实施的《GJB 473-88 军用软件开发规范》。
- DOD-STD(Department Of Defense Standards)——美国国防部标准。
- MIL-S(Military-Standards)——美国军用标准。

另外,我国的一些经济部门(如信息产业部、经贸委等)也开展了软件标准化工作,制订和公布了一些适应于本部门工作需要的规范。在制订这些规范的时候大都参考了国际标准或国家标准,对各自行业所属企业的软件工程工作起了强有力的推动作用。

4. 企业规范

一些大型企业或公司,由于软件工程工作的需要,制订适用于本部门的规范,如美国IBM公司通用产品部(General Products Division)1984年制订的《程序设计开发指南》。

5. 项目规范

项目规范是为一些科研生产项目需要而由组织制订一些具体项目的操作规范,此种规范制订的目标很明确,即为该项任务专用,如《计算机集成制造系统(CIMS)的软件工程规范》。项目规范虽然最初的适用范围小,但如果它能成功地指导一个项目成功运作并可以重复使用,也有可能发展成为行业的规范或标准。

3.4.2 ISO主要软件质量标准

ISO 9001是ISO 9000族标准体系之一,即设计、开发、生产、安装和服务的质量保证模式,这一套标准中包含了高效的质量保证系统必须体现的20条需求。

(1) 管理职责。
(2) 质量系统。
(3) 合同复审。
(4) 设计控制。
(5) 文档和数据控制。
(6) 采购。
(7) 对客户提供产品控制。
(8) 产品标识和可跟踪性。
(9) 过程控制。
(10) 审查和测试。
(11) 审查、度量和测试设备的控制。
(12) 审查和测试状态。
(13) 对不符合标准产品的控制。
(14) 改正和预防行动。
(15) 处理、存储、包装、保存和交付。
(16) 质量记录控制。
(17) 内部质量审计。
(18) 培训。
(19) 服务。
(20) 统计技术。

ISO 9000自1987年诞生以来,历经了4次正式改版。

(1) 第1次改版发生在1994年,它沿用了质量保证的概念,传统制造业的烙印仍较明显。

(2) 第2次改版是在2000年,不论是从理念、结构还是内涵,这都是一次重大的变化。

标准引入了“以顾客为关注焦点”“过程方法”等基本理念，从系统的角度实现了从质量保证到质量管理的升华，也淡化了原有制造业的痕迹，具备了更强的适用性。

(3) 第 3 次改版是在 2008 年，一次“编辑性修改”，并未发生显著变化。

(4) 第 4 次是 2015 版本，这次改版在结构、质量手册和风险等方面都发生了变化。

ISO 9001：2015 版本发生了很大的变化，下面列出几个最重要的变化。

(1) 强调建立适合各组织具体要求的管理体系。

(2) 要求组织高层能参与其中并承担责任，使质量与更广泛的业务策略相适应。

(3) 风险防范意识贯穿整个标准，使整个管理体系适用于风险预防，并鼓励持续改进。

(4) 对文件的规定性要求更少：组织机构可以自行决定需要记录什么信息，应该采用什么文件格式。

(5) 通过使用通用的新高层结构(Annex SL)，与其他主要管理体系标准保持一致。

ISO 9001：2015 要求：更多地理解外部环境、解决风险以及高级管理层更大的“质量领导力”责任，这与管理体系和产品/服务质量之间的关联环节息息相关。同时，新标准更加注重内部利益相关方的直接参与，或者说是对组织管理体系的设计、实施、架构和绩效的监督，从而确保质量管理体系是组织业务流程中的一个不可或缺的组成部分。新标准要求组织的质量管理体系(QMS)与组织的业务流程整合与统一。

(1) 赋予最高管理者更积极的角色，促使高层管理者更大程度地领导和参与组织的质量管理体系。

(2) 引入“基于风险的思维”的理念，该理念引导组织将资源重点分配到处理关键和主要风险，以及可能带来重大机会的领域。

(3) 新增“组织环境”的要素并作为整个质量管理体系的核心与基础。

(4) 统一的标准结构适用于所有的管理体系标准，有利于整合与兼容不同的管理体系。

因为 ISO 9001 标准适用于所有的工程行业，所以为了在软件过程的使用中帮助解释该标准，专门开发了一个 ISO 指南的子集，即 ISO 9000-3。由于软件行业的特殊性，ISO 9001 在软件行业中应用时一般会配合 ISO 9000-3 作为实施指南。需要参照 ISO 9000-3 的主要原因是软件不存在明显的生产阶段，故软件开发、供应和维护过程不同于大多数其他类型的工业产品。例如，软件不会“耗损”，所以设计阶段的质量活动对产品最终质量显得尤其重要。

ISO 9000-3 其实是 ISO 质量管理和质量保证标准在软件开发、供应和维护中的使用指南，并不作为质量体系注册/认证时的评估准则，主要考虑软件行业的特殊性制订。参照 ISO 9001《质量体系设计、开发、生产、安装和服务的质量保证模式》，并引用 ISO 8402《质量管理和质量保证术语》，使得 ISO 9000 系列标准应用范围得以拓展。ISO 9000-3 核心内容主要如下。

(1) 合同评审。

(2) 需方需求规格说明。

(3) 开发计划。

(4) 质量计划。

(5) 设计和实现。

(6) 测试和确认。

(7) 验收。

(8) 复制、交付和安装。

(9) 维护。

当我们展望未来时,确保“质量管理”被视为“不只是 ISO 9001 认证”是很重要的,它真的可以帮助组织“实现长期的成功”。这是“推广质量”最广泛的意义,并且鼓励组织看待质量管理超越于仅仅符合一套要求。这可以通过提供联结激发对于诸如 ISO 9004 及其他 ISO 管理体系标准的运用实现。

3.4.3 IEEE 相关的软件质量标准

IEEE 系统软件工程标准由软件工程技术委员会(Technical Committee on Software Engineering,TCSE)之下的软件工程标准工作小组(Software Engineering Standards Subcommittee,SESS)积极创立。

所有的标准依对象导向的观念进行分类,并假设软件工程的工作执行都是透过项目的方式完成,即每一个项目都会与“顾客”互动,它也会耗用某些“资源”以执行某些“流程”,而交付特定的“产品”。根据这种理念,软件工程标准汇编便环绕在顾客标准、资源与技术标准、流程标准及产品标准等 4 种对象上,而每一种标准之下又再细分为需求标准(standards)、建议惯例(recommended practices)及指南(guides)。

1. 顾客标准

- 软件获得(software acquisition);
- 软件安全(software safety);
- 软件需求(system requirements);
- 软件开发流程(software life cycle processes)。

2. 流程标准

- 软件质量保证(software quality assurance);
- 软件配置管理(software configuration management);
- 软件单元测试(software unit testing);
- 软件验证与确认(software verification and validation);
- 软件维护(software maintenance);
- 软件项目管理(software project management);
- 软件生命周期流程(software life cycle processes)。

3. 产品标准

- 可靠性度量(measures to produce reliable software);
- 软件质量度量(software quality metrics);
- 软件用户文档(software user documentation)。

4. 资源与技术标准

- 软件测试文件(software test documentation);
- 软件需求规格(software requirements specifications);
- 软件设计描述(software design descriptions);
- 再用链接库的运作概念(concept of operations for interoperating reuse libraries);

- 辅助工具的评估与选择(evaluation and selection of CASE tools)。

3.4.4 IEEE 730-2014：SQA 流程

《IEEE 730-2002 软件质量保证计划》已被《IEEE 730-2014 软件质量保证流程》取代，IEEE 730-2014 规定了启动、规划、控制和执行软件开发或维护项目的软件质量保证流程的要求。该标准与 ISO/IEC/IEEE 12207：2008 的“软件生命周期过程”和 ISO/IEC/IEEE 15289：2011 的“信息内容要求”相协调。IEEE 730-2014 规定了 SQA 的主要任务，其结构如图 3-8 所示。

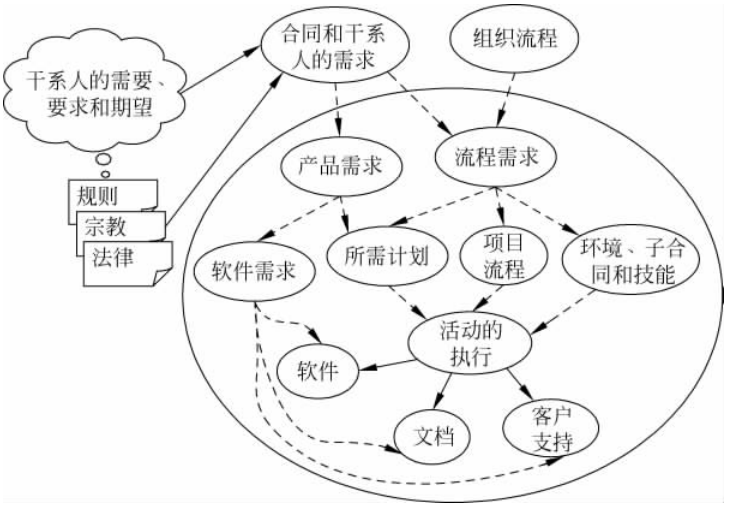

图 3-8 IEEE 730-2014 标准结构示意图

1. SQA 流程实施中的任务

- 建立 SQA 流程；
- 协调相关软件流程；
- 规划 SQA 活动；
- 执行 SQA 计划；
- 管理 SQA 记录；
- 评估和确保组织的客观性。

2. 产品保证期内的任务

- 评估一致性计划；
- 评估产品的一致性；
- 评估产品的可接受性；
- 评估产品支持；
- 测量产品。

3. 过程保证中的任务

- 评估生命周期过程；
- 评估环境；

- 评估转包商流程；
- 衡量流程；
- 评估员工技能和知识。

3.4.5 IEEE 1012-2016：验证与确认

IEEE 1012-2016是IEEE系统、软件和硬件验证和确认(V&V)标准，针对不同的集成层次，明确软件研发生命周期过程中V&V的要求。V&V流程包括产品的分析、评估、审查、检查、评估和测试，用于确定给定活动的开发产品是否符合该活动的要求，以及产品是否满足其预期用途和用户需求，其范围包括系统、软件和硬件以及它们的接口。本标准适用于正在开发、维护或复用的系统、软件(还包括固件和微代码)和硬件(包括商业离岸的遗留传统)，包括相应的文档。

(1) IEEE这样定义“验证”：“它是用来评价某一系统或某一组件的过程，判断给定阶段的产品是否满足该阶段开始时施加的条件。”也就是说，验证活动在很大程度上是一种普通的测试活动，要求验证每个开发阶段(例如软件某项需求或多项需求的实现)是否符合先前阶段定义的需求，而且开发人员应严格坚持审计跟踪。

(2) IEEE这样定义“确认”：“它是开发过程中间或结束时对某一系统或某一组件进行评价的过程，以确定它是否满足规定的需求”。换句话说，需要确认已经实现的组件是否按照规格说明书进行工作。

IEEE 1012-2016的结构分为三部分——软件技术流程、硬件技术流程和系统技术流程，前两者支持后者，如图3-9所示。在软件技术流程中，除了贯穿软件研发生命周期的流程——从概念开发、需求分析、架构设计到软件验收、软件安装之外，还有采购流程、供应流程、测试计划、集成/系统/验收测试、验证与确认流程。

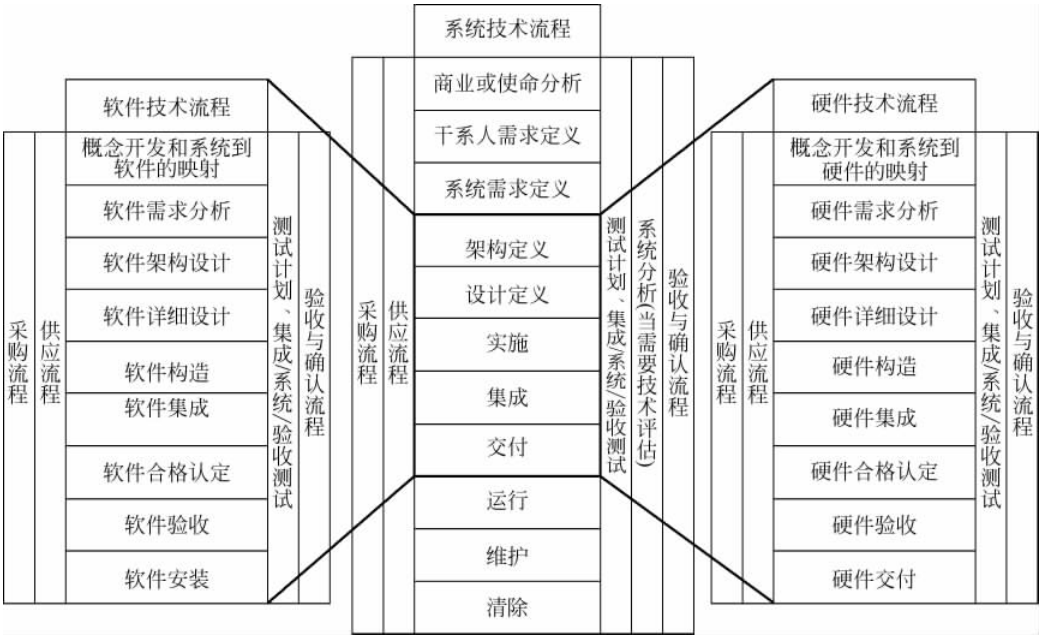

图3-9 IEEE 1012-2016的结构示意图

3.4.6 IEEE 1028-2008：评审与审计

IEEE Std 1028 对评审做了较为详尽的标准化工作。评审是对软件元素或者项目状态的一种评估手段，以确定其是否与计划的结果保持一致，并使其得到改进。一般来说，评审包括管理评审、技术评审、审查、走查和审计，如表 3-3 所示。

表 3-3 不同评审形式的说明

类 别	目 的	参 与 人	备 注
管理评审 (management review)	监控进展是否与需求相符，判定计划和进度表的状态及需求在系统中的分配；或评价为达到与目的相符所采用的管理途径的有效性；它们由对本系统负有直接责任的管理人员实行	决策制订者 评审领导人 记录员 管理人员 其他小组成员(可选) 技术人员 客户或用户代表(可选)	
技术评审 (technical review)	评价软件产品，由认定的小组人员决定对预期使用的适宜性，并标识与规格说明和标准的偏差。可能还要推荐各种替换(方案物)，以便考核	决策制订者 评审领导人 记录员 技术人员 管理人员(可选) 其他小组成员(可选) 客户和用户代表(可选)	
审查 (inspection)	查出并标识软件产品的反常，验证软件产品是否满足规格说明；验证软件产品是否满足指定的质量属性；验证软件产品是否与用到的规章、标准、指南、计划、规程相符；标识与标准和规格说明的偏差；收集软件工程数据，用收集到的软件工程数据改善审查过程本身，以及相应的支持文档	审查领导人 记录员 读者 作者 审查员	评审的所有参与者都是审查员，管理地位比审查小组所有成员都高的人不应参与
走查 (walk through)	找出反常；改善产品；考虑替换物的实现；评价与标准和规格说明的相符性	走查领导人 记录员 作者 小组成员	
审计 (auditor)	就用到的规章、标准、指南、计划和规程对软件产品和过程独立地提供评价	审计领导人 记录员 作者 项目发起人 审计组织	审计员应将观察到的不相符处和相符处记入文档

3.4.7 CMMI 质量框架

CMMI(Capability Maturity Model Integration，软件能力成熟度模型集成模型)是实现一个组织的集成化过程改进框架，集成了 3 个过程改进模型：软件(SW-CMM)、系统工程

(EIA/IS 731)以及集成化产品和过程开发(IPD CMM)。CMMI模型将整个软件改进过程分为5个成熟度等级,这5个等级定义了一个有序的尺度,用以衡量组织软件过程成熟度和评价其软件过程能力。CMMI模型中最基本的概念是"过程域",在软件和系统工程之间实现了较高的集成性,产生了一些非常具有通用性的工程过程域。CMMI的模型构件主要有以下3类。

(1) 需要的构件,即"目标",表示某个过程域想要达到的最终状态,其实现则表示项目和过程控制已经达到了某种规定的程度。针对单一过程域的目标,称为特定目标;可适用于所有过程域的目标则称为共性目标。

(2) 期望的构件,即"实践",代表了达到目标所"期望的"手段。CMMI模型中每个实践都恰好映射到一个目标。当然,只要能够实现模型中规定的目标,组织可以采用其他经过认证的手段作为"替代的"实践,而不一定非要采用模型中规定的实践。因此,实践只是模型中期望的构件,而不是需要的构件。同样,针对单一过程域的实践,称之为特定实践;可以用于所有过程域的实践则称为共性实践。

(3) 提供信息的构件有10种,分别是目的、介绍性说明、引用、名字、实践与目标关系表、注释、典型工作产品、子实践、学科扩充以及共性实践的详尽描述。这些构件为需要构件和期望构件提供了有益的补充。

CMMI的思想是一切从顾客需求出发,从整个组织层面上实施过程质量管理,实现从需求管理到项目计划、项目控制、软件获取、质量保证、配置管理的软件过程全面质量管理,正符合了TQM的基本原则。因此,它的意义不仅是对软件开发的过程进行控制,它还是一种高效的管理方法,有助于企业最大程度地降低成本,提高质量和用户满意度。总的来说,SQA是通过协调、审查、促进和跟踪以获取有用信息,形成分析结果以指导软件过程。

1. 提出软件质量需求

软件质量保证部门在新项目的需求分析阶段就开始介入,对形成的软件需求进行分析与评价,并提出可能存在的诸如安全性、可靠性、可扩展性和易用性等问题,再根据软件本身特性、规模及将来的运行环境等进行综合评定,确定软件要满足的质量要求,并记录下来形成正式文档,尽可能地做到对软件周期各个阶段的测量确定一个定量或定性的标准,作为以后各阶段评审的标准和依据。

2. 确定开发方案

经过需求分析阶段深入详细的工作,软件质量保证组与开发部门共同研究确定软件开发方法,选择软件开发所使用的开发工具。各种开发工具都各有所长,各有侧重,根据要开发的软件本身的一些特性和功能要求,同时还要考虑将来的维护,综合平衡考虑软件过程的各个阶段。例如,Delphi的数据库功能强大,C比较适合编写底层的程序等,某些情况下可能会同时使用多个开发工具进行混合编程。

3. 阶段评审

阶段评审就是利用在需求分析阶段所选择与制订的标准、规范以及安排的计划,对软件工程各个阶段的进展、完成质量及出现的问题进行正式技术评审,确保过程按计划执行,遵守相应的标准与规范执行,并形成报告。如果发现有不符合的问题,遵循逐级解决的原则进行处理,将处理结果通知所有相关人员,记录解决的过程及结果,以作为日后改进时的重要参考资料。

4. 测试管理

测试管理的好坏，直接关系到测试实施的效果。SQA 必须从宏观上制订并监督执行软件测试策略和测试计划，形成测试完成的标准以作为审查时的依据及制订测试策略时的参考，并组织测试人员制订更详细的测试计划与案例，促使测试有效地进行。

5. 文档化管理

提到文档化管理，首先想让读者思考一个问题：一个好的组织机构是如何一天比一天、一年比一年变好的？读者可能会说：经验丰富，管理有方，技术过硬。且不说技术，经验与管理意味着他们做了些什么？不是一个好的管理方式拿来一用就是适合的，而是管理体系本身具有自我完善的特性，也就是不断地自我改进。这好像与文档化管理没有直接关系，其实不然。一个组织要使改进成为可能，首先工作要有合理的依据、步骤、方法和解决问题的原则，同时这些过程所产生的数据必须记录在案，以供总结经验时参考、工作时作为依据及评价时作为标准，不断改进并产生新的文档。同样的，SQA 的工作也遵循相同的规则，生成软件文档并对文档的改变进行控制也是 SQA 一项非常重要的工作。不一定所有文档内容都是自己独立形成的，如 ISO 或 GB 等标准与规范也可以列入文档的管理范围内，成为自己文档体系的一部分。

6. 验证产品与相应文档和标准的一致性

SQA 人员会依据已经形成的相关文档对应其所处的阶段，对过程进行审查，检查执行情况形成报告，对不符合的问题依据逐级解决的原则做相应的处理，同时对问题的处理过程和结果通知与该问题相关的所有人，并跟踪至问题彻底解决，以确保软件开发过程与相应文档要求一致。

7. 建立测量机制

通过以上活动可定性评估项目工作，可以了解工作做到什么程度，达到要求的大致情况，存在哪些不足，最终项目是否可以如期按质按量地完成等。如果客户想知道完成的具体量化的数字，如质量情况占项目的要求百分比，项目进展情况占完成整个项目（包括质量要求与项目阶段）比例是多少等，可能就很难准确回答了。因此，建立软件质量要素的测量机制非常重要，能让 SQA 人员和领导层了解各种指标的量化信息，方便对进度进行精确控制，调整资源，紧急时可以做出准确的决策。

8. 记录并生成报告

记录并生成报告属于文档化管理的范畴，前述的文档化管理是对所有阶段、所有人的要求，在这里单独提出来，是作为 SQA 本身活动的一个部分，而且也是 SQA 工作结果的体现之一。

3.4.8 软件过程改进标准

软件过程评估与改进标准主要有两个：SPICE 和 CMMI。其中，SPICE（Software Process Improvement and Capability Determination，软件过程改进和能力确定）上升为国际标准——ISO/IEC 15504 信息技术—过程评估，这是一套用于计算机软件开发过程和相关业务管理功能的技术标准文档，但正在被新的国际标准所代替，例如：

- “ISO/IEC 15504-1：2004 信息技术—过程评估—概念和术语”被 ISO/IEC 33001：2015 代替。

- “ISO/IEC 15504-2：2003 信息技术—过程评估—实施过程与评估要求”被 ISO/IEC 33002：2015、ISO/IEC 33003：2015、ISO/IEC 33004：2015、ISO/IEC 33020：2015 等标准代替。
- “ISO/IEC 15504-3：2004 信息技术—过程评估—为过程评估实施提供指导”被 ISO/IEC TS 33030：2017 标准所代替。
- “ISO/IEC TR 15504-7：2008 信息技术—过程评估—组织成熟度评估”被 ISO/IEC 33001：2015、ISO/IEC 33002：2015、ISO/IEC 33003：2015、ISO/IEC 33004：2015、ISO/IEC TR 33014：2013 等标准代替。

截至目前，ISO/IEC 15504 的第 4～6、8～10 部分依旧有效，但最终也会被 ISO/IEC 330 系列标准所代替。ISO/IEC 15504 的第 4～6、8～10 包含的主要内容如下。

- 过程改进和过程能力确定的使用指南。
- 软件生命周期的示例过程评估模型。
- 系统生命周期的示例过程评估模型。
- IT 服务管理的示例过程评估模型。
- 目标过程简介。
- 安全扩展。

SPICE 过程能力度量框架考虑了评估的过程在执行中的前后关系，涵盖过程参考模型、过程评估模型等，构成一套完整的结构，对实施评估做了详细的要求，并验证过程评估一致性，使其能够保证评估结果的客观、公正、一致和可重复，保证被评估过程具有代表性。过程评估建立在二维模型之上。

(1) 过程维由外部的过程参考模型(PRM)提供，PRM 用来定义一个过程集合，过程由陈述过程的目的和结果表征。

(2) 能力维由测量框架组成，包括 6 个过程能力级别和与其相连的过程属性，评估输出称为过程剖面，由每个过程评估获得的分数的集合构成，同时也包括该过程达到的能力等级。

这个框架可用于组织的计划、管理、监督、控制和改进采办、供应、开发运行、产品和服务的演变和支持。

CMMI 流程改进基本上可归纳为 3 大步：确定流程改进的总体框架、细化框架内的要求和明确流程改进的度量方法与标准。

1. 确定流程改进的总体框架

这部分内容包括 CMM 流程改进的总体方案、总体策略、总体目标、阶段性目标，还包括目标流程的确定，流程改进与项目生命周期的关系，度量体系需要涉及的部分与总体标准，流程中权责分配表及体系文件管理。

- 策略：要求组织明确自己是采取自顶向下或自底向上、一步走或分步到达的方式来执行流程改进过程。
- 目标：分为总体目标与阶段性目标。总体目标，要求组织在较长的时间内，明确组织在实施 CMM 流程改进后可以达到的预定的运行状态。阶段性目标可以理解为大目标下的小里程碑。只有每个小里程碑可以按预期完成，大的目标才有可能达到预期要求。

- 目标流程：一般地，组织的流程相对较多，实施 CMM 为考虑平稳过渡，不可能一次完成所有流程的改进。部分重点流程可能完全被确立为改进流程，而部分流程可能本来与 CMM 的要求相近，只需要做小部分调整。还有一部分流程虽然与 CMM 要求不一致，但其具有很好的独立性，它的存在本身不影响其他流程，同时改进的必要性不大。所以，准备实施 CMM 流程改进时需要考虑改进的目标流程及改进的程度。
- 度量方案与标准：依 CMM 的关键过程域的要求，结合组织的具体情况确定大的度量方案与标准。这与目标是有区别的，目标只是从高层管理者的层面上看希望达到的状态，而度量标准却是衡量流程改进的测量参考数据。两者之间有关系却又有不同的侧重点。
- 体系文件：体系文件就是支持体系所需要的所有的文档的集合。为什么需要对其进行管理呢？原因很简单：保持文档的完整性和有效性。

2. 细化框架内的要求

细化工作主要包括详细的过程定义与描述，详细的度量和过程监控方法，以及整个过程涉及内容的详细有效的文档描述。

有关过程的定义，有标准的模板，内容涉及：

- 目的——定义本过程的目的。
- 角色——本过程中涉及的角色及其职责。
- 入口准则——什么条件会触发本过程的启动。
- 输入——文档、资源和数据。
- 过程步骤——本过程有关的处理步骤。
- 输出——文档、资源和数据。
- 出口准则——什么条件会触发本过程的结束。

对于度量、过程监控方法、工具技术和方法、差距分析、过程改进历史和相关过程，必要时也可以对其进行详细的描述。

3. 明确流程改进的度量方法与标准

度量就是依据度量的目标采集过程相关数据，对数据进行有效的分析以用来评价、跟踪和指导流程，如图 3-10 所示。

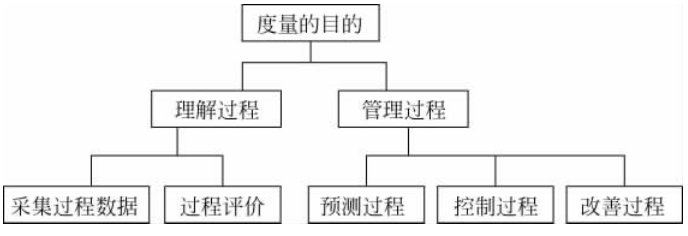

图 3-10　流程改进的度量过程

通常采用的是 GQM(Goal-Question-Measurement，目标问题度量)法，如图 3-11 所示。这种方法通常需要先采集代码缺陷、进度数据、跟踪数据。例如，需要对软件缺陷进行度量设计时，需要考虑：

- 有效地消除和记录缺陷;
- 有效地分析缺陷、有效的缺陷度量评审;
- 有效地缺陷评审报告。

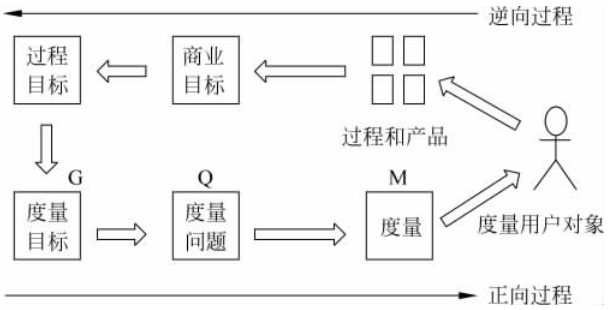

图 3-11 GQM 示意图

3.5 软件质量成本

质量成本(Quality Cost)是追求软件高质量及与此相关的活动所带来的一切成本,这好比“管理是有成本的”。质量与生产率之间有着内在的联系,高生产率必须以质量合格为前提。从短期效益看,追求高质量会延长软件开发时间并且增大费用,似乎降低了生产率。从长期效益看,高质量将保证软件开发的全过程更加规范流畅,大大降低了软件的维护代价,实质上是提高了生产率,同时可获得很好的信誉。下面介绍质量成本的概念,然后对软件质量成本进行分析,以帮助改进质量、提高软件开发的生产率。

3.5.1 质量成本

质量的成本属性,可以追溯到日本质量大师田口玄一的质量思想。田口玄一在多年研究和实践的基础上,创造性地提出了关于质量的定义:所谓质量,是指产品上市后给社会带来的损失。田口把产品质量与给社会带来的损失联系在一起,即质量好的产品就是上市后给社会带来损失小的产品。这个定义保存了满足社会需要的中心内容,在本质上它与 ISO 9000:2000 给出的质量定义是一致的。田口关于质量的定义最有价值之处是引入了质量损失的概念,开辟了定量研究质量的道路。日本的众多企业就是用田口的质量管理方法进行质量管理的。

真正的“质量成本”概念,是由美国质量大师费根堡姆在 1945 年美国电子工程师杂志上提出来的,1951 年在他所著的 *Total Quality Control*《全面质量控制》一书中作了说明,1956 年在《哈佛经营周刊》杂志上再次作了详尽的解释。他主张把质量预防费用和检验费用与因产品不符合要求所造成的厂内损失和厂外损失一起加以考虑,并形成质量成本报告,帮助企业管理者了解质量问题对企业经济效益的影响,成为进行质量决策的依据。

质量成本是为保证满意的质量而发生的费用与没有达到满意的质量所造成损失的总和,包括保证费用和损失费用,这是 ISO 8402—1994 所给出的标准定义,即质量成本可以分为质量保证成本和损失成本。

- 保证成本：为保证满意的质量而发生的费用。
- 损失成本：没有达到满意的质量所造成损失。

有的专家建议将质量成本分为预防成本、评价成本(或称评估性成本)和失效成本(补救性成本)。

- 预防成本：预防产生质量问题(软件缺陷)的费用，是企业的计划性支出，专门用来确保在软件产品交付和服务的各个环节(需求分析、设计、测试、维护等)不出现失误，如质量管理人员投入、制订质量计划、持续的质量改善工作、市场调查、教育与和培训等费用。
- 评价成本：是指在交付和服务环节上，为评定软件产品或服务是否符合质量要求而进行的试验、软件测试和质量评估等所必需的支出，如软件规格说明书审查、系统设计的技术审计、测试设备、内部产品审核、供货商评估与审核报告等。
- 失效成本：分为内部失效成果和外部失效成果，如果在软件发布之前发现质量问题而要求重做、修改和问题分析所带来的成本属内部失效成本，包括修正软件缺陷、返工、回归测试、重新设计和重新构造软件，以及因产品或服务不合要求导致延误带来的失效成本。软件发布之后给用户带来的失效成本是外部成本，包括现场维护、处理客户的投诉、产品更新或出紧急补丁包件、恢复用户数据等。外部失效成本比内部失效成本要大得多。

质量预防成本和质量评价成本之和就是质量保证成本；失效成本就是劣质成本(COPQ)，在 3.5.2 节做详细介绍。

质量成本设置的目的是为了质量成本的优化，要使预防成本、评价成本和失效成本的总和最小，使质量成本各要素之间保持合理的结构。其观点建立在这样的认识基础上——质量越高，所花费的成本会越高。为维持或提高质量水平，避免产生过高的故障成本，必须付出一定的预防和保证费用，并使两者之间取得有效平衡。例如，评审可以发现软件产品的问题，检查与评估进行得越细致、彻底，客户发现的质量缺陷就越少，但是评审并不能改进被评估的产品或服务的质量，不能杜绝有质量缺陷的产品。所以，加大预防成本能够降低评价成本，同样能够减少补救性支出。

质量成本将质量与企业经济效益直接联系起来，质量得以用货币语言来表达，质量语言和货币语言形成对话，从一个务虚的概念转换成一个务实的概念，使企业管理层对质量及其管理的意义和作用有了新的认识，更容易树立质量至上的理念，进一步加大质量管理力度，使企业立于不败之地。这是质量成本对社会经济发展的重大贡献。

20 世纪 90 年代以后，质量专家又将质量成本分为符合性成本和非符合性成本。符合性成本是指现有的过程没有故障而能满足顾客所有明示或隐含的需求所发生的费用；非符合性成本是指现有过程的故障所发生的费用。明确了研究质量成本的目的就是为了降低非符合性成本，这就提高了质量成本的实用性。

质量成本得到了西方国家的普遍重视，后来朱兰、克劳士比等质量大师又对此概念做了补充和完善。特别是随着朱兰博士“矿中黄金”理论的提出，更使建立在这一基础之上的质量成本理论日趋完善。质量大师朱兰曾在《朱兰论质量策划》一书中对质量下过两个定义：产品特征和没有缺陷。

(1) 产品特征越好，客户就觉得质量越高，产品就越容易销售，同时较高的质量通常有

较高的成本。

(2) 没有缺陷或缺陷较少,质量越高。质量是靠差异性表示的,即实际表现与客户的期望所存在的差异被用来衡量质量的程度。每一个缺陷都会带来修复这个缺陷的成本,就这个意义上的质量说,有较高的质量通常有较少的成本。

他给出了这两种概念的对比,如表3-4所示。

表3-4 产品特征和没有缺陷的对比

对比	产品特征	没有缺陷
	较高的质量,使公司能	较高的质量,使公司做到
1	提高客户的满意度	降低差错率
2	增强产品可销性	减少返工和废料
3	符合竞争	减少现场失职和保证费
4	提高市场份额	减少检验和试验费
5	提高销售收入	减少客户的不满意度
6	获得优惠价格	提高产量和能力
7	对销售额有很大影响	改进交货绩效
8	较高的质量通常有较高的成本	有较高的质量通常有较少的成本

3.5.2 劣质成本PONC和COPQ

追溯质量管理理论演变和发展的过程,可以了解到:质量成本和劣质成本都与质量密切相关,是质量管理的重要方法。质量成本概念是劣质成本概念的基础;劣质成本概念是对质量成本概念的继承和发展。下面,介绍劣质成本的两个重要概念:PONC和COPQ。

- PONC,即"不符合要求的代价"(Price of Nonconformance)或称"劣质成本",是指由于缺乏质量而造成的人力、财力、物力以及时间成本的浪费。PONC是在"零缺陷"质量管理中,为了更有效地衡量质量成本而引入的一个重要概念。
- COPQ,即"不良成本"(Cost of Poor Quality)或称"劣质成本"。COPQ指所有由过程、产品和服务中的质量缺陷引起的费用。COPQ则是六西格玛质量管理中的一个重要概念,用于有效地衡量质量成本、质量改进过程在经营效益上的表现。

虽然PONC和COPQ在定义上有些差异,但从思想或基本内容上看,PONC和COPQ是一致的,可以说是同一个概念,只不过由不同的学派(零缺陷、六西格玛等)提出来的。

关于劣质成本,除了田口玄一,还有其他质量大师的论述。例如,朱兰认为"每一项任务都能毫无缺陷地执行,就不会发生的成本";克劳士比认为"第一次没有把事情做对而产生的所有费用都应为劣质成本"。按照质量损失的要素,劣质成本可分为如下3类。

(1) 故障成本,包括质量成本中的外部故障成本、内部故障成本,需采取返工、返修、纠正等补救措施所花费的成本。

(2) 过程成本,包括非增值成本(非增值的预防成本和鉴定成本)、低效率过程成本(如多余的操作、重复的作业、低效或无效的服务和管理)、机会损失成本(指如果没有缺陷就不会发生的费用,或者可以减少的费用,由于没有努力去采取措施而导致增加的费用)。

(3) 损失成本,包括顾客损失成本(指顾客在使用产品或服务的过程中,给顾客所造成的各种额外的费用及负担,它的增加或超过顾客的承受能力,就会失去忠诚的顾客而使企业

蒙受损失)、信誉损失成本(指过程损害了企业信誉,造成顾客流失以及市场份额降低的损失)。

从劣质成本的要素构成,可看出“故障”“过程”“机会”和“顾客”是其关注点。这为降低成本指明了清晰的思路和途径,是对质量成本的突破和完善。

对于软件,劣质成本又有哪些呢?很容易就能列出一些:

- 验证缺陷;
- 回退到原来位置/版本(Roll Back);
- 代码完成后功能修改、测试用例修改;
- 缺陷报告质量低;
- 回归测试和不断的重复测试;
- 错误的开发环境或测试环境;
- 为修正客户发现的问题,紧急发布程序补丁。

由于劣质软件产品,对于软件企业,同样会产生一些间接的、不可低估的甚至巨大的成本——订单/合同丢失、企业市场机会损失、企业信誉损失等所造成的成本。即使对于可以统计的财务数据的项目,劣质成本还是非常可观的,吞噬着企业的利润。例如,在制造业中,在那些没有建立完善的质量管理体系的企业,劣质成本占总成本的20%~25%;如果企业成功实施了“零缺陷”和“六西格玛”质量管理,劣质成本占总成本的10%~15%,甚至更低6%~8%。在软件业中,劣质成本占软件开发成本的40%~80%。

曾经对一个国际性的软件公司做了一个调查,选择了10个项目,即“开发人员修正缺陷、测试人员验证缺陷、返工、设计或代码完成后的需求变化、不清楚或无效的缺陷报告、代码完成后补充的测试用例、由于缺陷修复后所做的回归测试、测试环境设置错误、产品发布后遗漏的缺陷验证、为产品发布后遗漏的缺陷出补丁包等”,统计结果表明,劣质成本竟高达45.86%,如果考虑开发周期的延长导致市场销售时机的错过、销售额的下降和产品发布后遗漏的缺陷导致客户的抱怨而影响销售额等,劣质成本肯定超过50%。这就是软件业的质量现状,所以在软件业更需要大力推进质量的改进。

许多质量成本支出是隐性的,很难通过常规的质量成本评估系统进行测定。即使被发现,其中很大一部分也会被认为是企业的正常经营支出。多数质量成本系统无法检测的隐性成本主要集中在客户补救成本、信誉损失成本和客户不满成本3方面。虽然这些成本不能通过常规质量成本系统确定,但在成本构成中却占有相当大的比重。现有与未来客户是否购买产品就与这些成本有关。消除了外部问题因素,这些支出也随之消失。因此,消除企业外部补救支出尤其重要。

英国有句格言:“每个人家中的门后都有一个骷髅”,这样的“骷髅”也存在我们的企业中,就是平时不为人注意的“隐形工厂”,不断造成损耗和浪费。在表面用于生产和销售的成本消耗背后,存在着惊人的浪费。每家企业、每个组织都有一个巨大的“隐形工厂”。正如克劳士比所说“在大多数组织中,他们在垃圾箱中浪费的金钱比他们废品桶中的损失更多”。质量成本犹如海洋中的冰山,看到的直接成本(如收费的账单出错、失败的项目、现场运行故障)是冰山一角,大部分的间接的质量成本是看不见的,好比隐藏在海平面以下的冰山主体,给企业带来的损失是巨大的,如图3-12所示。

所以,PONC和COPQ的思想基础就是“质量提高了,劣质成本就低了,企业成本也会降低”,即费根堡姆指出的,质量与成本之间的关系是“和”不是“差”。质量成本和劣质成本

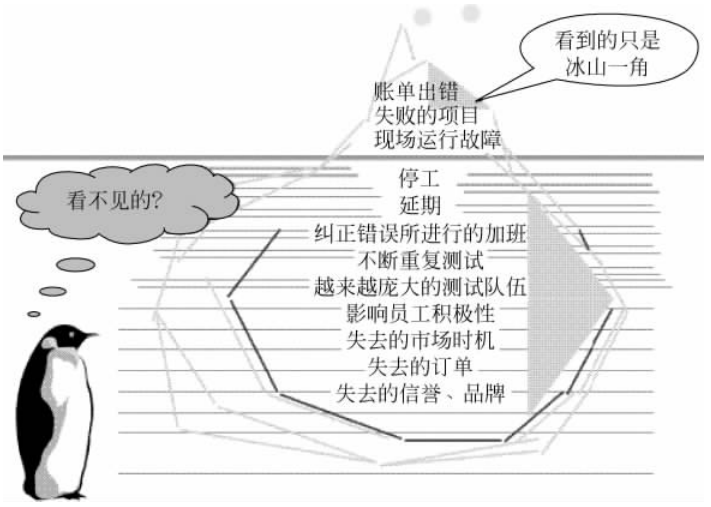

图 3-12　质量成本——海洋中的冰山

是两种质量观的体现,质量成本反映了传统的质量观,劣质成本(PONC 和 COPQ)代表着现代质量观。

质量成本依据财务部门提供的现成数据和企业财务活动结果,对各类质量成本的变化趋势进行分析,研究改进的途径,而对这些数据所反应的实质及数据背后所隐藏着的潜力分析较少,同时对现有数据之外的问题研究也很少见,从而造成企业对质量成本关注不够。

劣质成本引导我们对财务数据进行深层次的研究和分析,透过现有数据看到潜在和隐含的问题,揭示质量和成本之间的关系及其规律。劣质成本让我们透过数据追溯过程,对过程的各个环节、各项作业进行分析,判断其有效性和增值性。例如,对质量特性值的偏离状况,即使不超出规格限制,也要分析偏离的幅度,提高质量水平。它还通过对时间、资源、程序、环境和管理等众多因素的剖析,研究这些因素的影响及可能造成的损失,把握各种可能实施质量改进和降低成本的机会。

概括地说,质量成本的立足点是企业的成本,目的是将企业的成本降低,而对顾客利益方面的考虑则很少。劣质成本立足点是客户满意度,目的是减少质量损失,继承了质量成本的有效成分,扩展和延伸了质量成本的内涵和功能,使质量和成本、效益更加紧密地融合,把质量管理推进到新的阶段。

本章小结

现代软件质量工程体系继承了全面质量管理思想,要求组织中每个人都承担质量的责任,将质量融于整个软件开发生命周期,从需求分析、设计、编程、测试到维护,所有软件开发流程中都包含有质量控制、保证和改进的流程。

软件质量工程体系,涵盖质量计划、质量风险管理、质量成本控制和质量计划的实施等内容。质量计划的制订受质量文化影响、质量方针指导,通过对影响质量各种因素的分析,了解可能存在的质量风险,从而加以回避和控制。通过对软件产品、过程的测量和质量的度量,不断改进软件开发过程,以达到软件质量预先设定的目标。