復旦大學

软件过程管理在管理信息系统项目中的实践和应用

曹 华

目录

软件过程管理在管理信息系统项目中的实践和应用

摘要

计算机的诞生，给人类的科技文明带来了新的冲击，这是一场翻天覆地的革命，计算机从此被利用在人类的各种生产活动中。

而随着软件应用的扩展，软件的功能也越来越复杂，仅仅依靠少数人已经无法解决复杂的软件开发问题。

面向过程的软件开发理论及实践得到了人们的重视。只有成熟的软件开发过程，才有能力保证软件的质量。

现今，为世界所普遍接受并且效果显著的软件过程管理理论学说主要是 ISO 开发的 ISO 9000 系列标准和 SEI 开发的 CMM。

如何在软件开发过程中应用软件过程管理的方式、方法，是人们普遍关注的问题。

本文针对某具体的管理信息系统（CCMIS）开发过程，介绍了与该过程相关的主要软件过程管理知识，包括需求分析、设计、代码编写、测试等各个开发阶段所涉及的软件过程管理技术、方法，以及 PSP、SCM 等相关技术理论。并在介绍各种技术、方法的基础上，对 CCMIS 的开发过程进行了分析和讨论，就上述的各个方面分别做了较为深入的分析和阐述，从而有针对性的找出了 CCMIS 开发过程中的不足之处，并就此提出了相应的改进方案。

最后，对 CCMIS 开发过程中尚未在本文中深入涉及的问题进行了探讨，展望 CCMIS 开发过程改进前景。

通过本文的阐述，可以对类似 CCMIS 的小型软件项目的软件开发过程理论有一个认识，并对在类似项目中的实践应用有所了解。

关键词：软件过程管理，ISO，CMM，KPA，PSP

Managing the Software Process in Practice & Application for a Management Information System

Abstract

The birth of computer, brought an impact to the science & technology of human being. It was a great revolution. Computers have been used in all kinds of human activities.

With the development of software applications, functions of software become more and more complicated. A complex software development could not be resolved by only a few persons.。

The theories & practices of process oriented software development have been taken into consideration. Only a mature developing process could be the assurance of the software quality.

At present, the most effective & commonly accepted technologies of managing the software process are ISO 9000 standards by ISO and CMM by SEI.

How to apply technologies of managing the software process in software development? It is a focus concerned by many software developers.

For a certain management information system (CCMIS), this essay introduces some basic knowledge & technologies for the development process of that, including the knowledge & technologies for software requirement, software design, coding, testing, PSP & SCM. After that, there are some analysis & discussion about the development process of CCMIS. Through the analysis & discussion, we could see some weakness of the process, and some measures for the process improvement are given.

Finally, there are discussions about some topics to be taken into consideration.

Through the essay, a general idea of technologies & practices of managing the software process for such kind of small software project like CCMIS could be clear.

Key words: Managing the software process, ISO, CMM, KPA, PSP

第一章 软件过程管理简述

1.1 软件过程管理历史简述

1946 年，人类历史上第一台计算机 ENIAC 在美国诞生，从此以后，人类进入了计算机时代。计算机的诞生，给人类的科技文明带来了新的冲击，这是一场翻天覆地的革命，计算机从此被利用在人类的各种生产活动中。

计算机发展初期，由于科技水平所限，硬件设备昂贵，计算机应用的主要成本是硬件问题，人们工作的相当一部分精力是集中在如何节省硬件资源。随着科技水平的飞速发展，硬件的成本在大幅下降，而同时性能不断提高。与此同时，计算机应用中的另外一个问题却日益凸现出来，那就是软件。

计算机应用初期，软件是及其简单的，其功能主要就是进行简单计算。其时，软件开发没有什么规范可言，软件的质量也主要依赖少数“英雄”,“英雄”的个人行为对软件的成败有着举足轻重的作用。而随着软件应用的扩展，软件的功能也越来越复杂，仅仅依靠少数人已经无法解决复杂的软件开发问题。而且，即使是大批精英进行软件开发，也往往遭受失败的挫折。最典型的案例就是 IBM（INTERNATIONAL BUSINESS MACHINES，国际商业机器公司）在 1963 至 1966 年间开发的 IBM/360 操作系统。该项目在开发期每年投入五千万美元的资金，共投入工作量五千人年，最多时，有一千人左右同时参与该项目，写出了上百万行源程序。但是，最终，该项目却以失败告终。最近的案例如 1999 年美国大力神运载火箭发射失败，2003 年美国、加拿大部分地区大面积停电事故，均与软件错误有关。除此以外，各种中小软件开发失败的情况更是数不胜数。

时至今日，软件开发成本已经超过了硬件成本，在计算机的应用中占据了主要地位。而在诸多的失败教训面前，人们开始重新对软件进行审视，如何保证软件开发的质量，包括成本、进度，成为人们关注的重点。

在人们对软件开发进行思索的同时，工业界中成功实施的生产原则给人们带来了另外一种思路。经过比较，人们认为工业界和软件开发尽管有巨大差异，但在工业生产中成功适用的过程统计控制概念及其原则也同样适用于软件开发。

犹如平地一声惊雷，上述观点在受到关注的同时迅速被人们所接受，并且

诸多专家、机构开始投身于过程控制在软件开发中的应用的理论、实践研究中。美国国防部（DoD），赞助 SEI（Software Engineering Institute，Carnegie Mellon University，卡耐基•梅隆大学软件工程研究院），由此最终产生了 CMM。此外，欧洲委员会、英国国防部（MoD）、欧洲空间局、ISO（International Organization for Standardization，国际标准化组织），都投入了软件开发过程管理的研究中。

1.2 软件过程管理研究现状

1.2.1 过程概述

何谓“过程”，对此的定义各有不同。根据 Sami Zahram 在《软件过程改进》（*Software Process Improvement*）一书中所述，过程应当有三方面的特性：第一，过程的定义；第二，过程的学习；第三，过程的执行。这三者缺一不可。

通常，传统的企业文化是以产品为中心，但是，新的理论实践则强调以过程为中心。以过程为中心，将会得到如下益处：

✧ 协调组织活动，为共同目标而努力

✧ 通过衡量个人工作对过程的贡献，提供度量的基准

✧ 增强过程结果的一致性与可重复性，增强组织活动的一致性与可重复性

然而，“过程就像人的习惯，养成一种习惯很难，但改变习惯也不容易”（Watts S. Humphrey）。以过程为中心就是要形成成熟过程。这里所谓“成熟”即意味着实用、合理并经过充分考虑。一个成熟的过程应该具有明确的定义、有相应的培训、具有强制性与可服从性，并且能够不断改进。只有成熟的软件开发过程，才有能力保证软件的质量。

随着软件过程管理的研究，出现了诸多学说、理论。现今，为世界所普遍接受并且效果显著的主要是 ISO 开发的 ISO 9000 系列标准和 SEI 开发的 CMM。

1.2.2 ISO 9000 标准简述

为了在合同范围内控制购买者和供应商之间的关系，可以采用 ISO 9000 系列标准，其中，ISO 9001 是与软件开发和维护关系最密切的标准。ISO9001 完整的标题是“质量系统－设计/开发、生产、安装和服务的质量保证模型”。

ISO 系列标准强调如下质量概念：

✧ 组织应该达到和维持产品质量及相应服务，以不断满足购买者明确或隐

含的要求。

- ✧ 组织应该对预期的产品能达到和维持的质量具备信心。
- ✧ 组织应该是购买者相信，即发布的产品及相应的服务将拥有预期的质量，当合同需要是，这样的条款可被要求包括在协议中，并加以证明。

ISO 9001 与下述的 CMM 有共同之处，都强调了过程和文档化。但是，两者不同之处也很多，尤其是在持续过程改进，覆盖面以及详细程度上，ISO 9001 与 CMM 相比，就软件过程改进而言有诸多不足。因此，CMM 比 ISO 9001 更适合软件过程。

1.2.3 CMM 简介

CMM（Capability Maturity Model，能力成熟度模型），来源于工业界以及政府机构的软件过程评估和反馈意见，在美国国防部的支持下，由 SEI 开发出来的。最初的 CMM 是针对软件过程的，但不久人们发现 CMM 中的一般性概念对其它领域也是适用的。于是，在 1996 年，M. Konrad 等人将 CMM 划分为五类，我们通常所说 CMM 即为其中的软件 CMM（Capability Maturity Model for Software，SW-CMM），本文如果未作特殊说明，所说 CMM 均指软件 CMM。

CMM 的“正式”定义如下：

“CMM：一种将软件组织对于软件过程的定义、实现、度量、控制以及改进划分为不同阶段的方法”（M. Paulk 等，1994）。

对于软件开发和维护，CMM 提供了一个基本的过程管理和质量改进的概念。CMM 为我们提供了一个规划和框架，可以指引企业向过程化发展，帮助企业改进过程，逐步达到成熟从而获得成功。

对于客户而言，过程能力高意味着：

- ✧ 开发组织可以更好的响应客户以及市场的要求。
- ✧ 整个产品生命周期的费用会降低
- ✧ 提高最终用户的满意程度

对于开发组织而言，过程能力高意味着：

- ✧ 更低的开发及维护费用。
- ✧ 更短的生命周期及开发间隔。
- ✧ 由于对项目风险有更有效的分析和评估，从而保证费用与进度能够更好的符合预期的目标。
- ✧ 获得更好的设计质量与产品质量。

CMM 将过程成熟度分为如下五个级别，级别越高，说明过程越成熟。这

五个级别及基本特性如下：

CMM 等级	过程特点	过程特征
第五级：优化级	连续改进	过程改进已制度化
第四级：已管理级	可预测	产品及过程的质量是可控的
第三级：已定义级	标准化与集成	软件工程以及管理过程已定义并集成在一起
第二级：可重复级	制度化	已建立项目管理系统；实施情况可重复
第一级：初始级	混乱	没有统一过程；实施不可预测

表 1.2.2-1

CMM 体系中，每个级别都是由多个 KPA（Key Process Area，关键过程域）组成。实现了各个 KPA 中所包含的所有目标，就会达到相应的 CMM 级别，从而完善软件开发过程。CMM 中的 KPA 有着一些公共特征，这些都是在进行过程设计、制定相应文档是必须考虑的内容。关于 KPA 各个公共特征及解释，可以参考下表：

公共特征	定义与解释
实施约定	政策说明：为了强调组织约定与过程间的关系 领导：为了保证关键过程域的制度化，指定领导角色或必要的负责人
实施能力	组织结构：用于支持关键过程域的特殊组织结构 资源域投入：例如实现改进所需的特殊技能、工具及其它投入
实施活动	活动、角色与规程对于实现关键过程域以及达到最终目标是必不可少的 制定计划与规程，具体实施，对实施情况加以跟踪，并采取必要的纠正
度量分析	判断与过程有关的状态时基本的度量实践是必需的 其它度量措施也有利于提高管理有效性，增强软件产品的质量和功能
验证实现	高层管理者的监控是定期进行的，可以在适当的抽象层次对软件过程及时了解 项目管理的监控：通过定期的以及事件驱动的方式对项目管理进行监控 软件质量保证：以质量保证组或其它独立组进行的评审/审核形式进行

表 1.2.2-2

CMM 五个级别中共包含 18 个 KPA，关于具体的 KPA 在此不作赘述，读者可以参考相应资料进行了解。

CMM 的重点是软件企业如何实行软件工程，改进软件过程。除了 CMM 之外，SEI 还提出了 PSP（Personal Software Press，个人软件过程）和 TSP（Team Software Press，团队软件过程）。PSP 和 TSP 分别针对个人和团队的软件开发过程进行了规范，以利改进个人和团队的软件过程。这两者与 CMM 并不矛盾，双方有很多共同之处。PSP 与 TSP 可以视为 CMM 的“低级”形式，并为实现 CMM 的应用提供了更加具体的方式。

1.3 软件过程管理发展前景

经过不断的发展和实践，软件过程管理的理论与实践也有了进一步的发展。现在，SEI 已经推出了 CMMI（Capability Maturity Model Integration），将原有的 CMM（不是 SW-CMM）各个方面更好的结合到一起。

有一种观点，认为软件工业经历三次特别明显的发展浪潮。第一次是以瀑布式生命周期和结构化的方法为特征；第二次是过程成熟运动；第三次是预期的软件工业化。目前，我们正处于软件成熟度浪潮的中心，而随着时代的发展，必将达到软件工业的理想，即软件工业化，以大量生产统一的优质产品为特征。

第二章 CCMIS 项目简介及相关软件技术

2.1 CCMIS 项目简介

CMA CGM（CMA CGM SHIPPING CO., LTD，法国达飞轮船有限公司），于 1978 年成立于法国马赛（MARSEILLE）。CMA CGM 成立初期仅仅是一家经营地中海航线的支线船公司，时至今日，经过二十余年的大力发展，CMA CGM 现在已经跃居为全球第五大班轮公司，航线服务遍及世界主要国家、港口，成为真正的全球承运人（Global Carrier）。法国达飞轮船（中国）有限公司（CMA CGM (CHINA) SHIPPING CO., LTD），于 1992 年 10 月于中国上海成立，经过十余年的发展，已经由当初的办事处成为遍及具有全国各地三十余个分公司、办事处的有相当规模的航运公司。（下文若无特别说明，所涉及公司、业务或部门均指达飞中国）。现今，随着我国经济的飞速发展，众多的业务越来越集中到中国，达飞中国现业务量已占达飞集团全球业务总量的三分之一以上。

随着达飞中国业务的飞速发展，自然也对管理、信息等诸多方面提出了新的要求。原有的老旧信息系统 AIMS（AGENT INFORMATION MANAGEMENT SYSTEM）已无法满足各方面的需求，因此，达飞中国管理层决定由 IT 部门自行组织开发一套全新的信息系统－CCMIS（CMA CGM CHINA MANAGEMENT INFORMATION SYSTEM），以取代原来的系统，满足现今业务发展的需求。

就 CCMIS 的功能而言，可以看作面向两类用户：一是具体业务人员；二是管理层。业务人员使用 CCMIS 进行具体的航运业务操作，比如单证、财务、箱管（集装箱管理）、码头业务等等；而管理层则可以从中获取所需信息，以利更好的了解公司业务或调整管理方式。

简单而言，CCMIS 可以看作一个数据管理系统，因为无论是单证、财务、集装箱管理还是码头业务，都需将各个业务数据准确记录，关键是如何使数据的输入能够快捷、方便，并且最大程度的预防错误。另外，还应考虑到在数据输入后，如何对其进行有效统计，提取所需要的数据信息，满足业务需求。

CCMIS 设计时，根据公司业务，划分为如下几个主要模块：

- ✧ 箱管
- ✧ 码头

- ✧ 财务
- ✧ 单证
- ✧ 公共信息数据

如上文所述，CCMIS 主要的功能就是数据管理，因此数据的输入是其主要功能之一。由于班轮公司业务的特性，无论是箱管还是码头，也无论单证还是财务，每天都有大量的数据输入，而且，数据的准确率和及时性是有很高要求的，数据的不准确和延误，往往会给公司带来经济损失。在这种情况下，对 CCMIS 的输入界面就有相当高的要求，必须有友好快捷的人机界面，以利数据输入；另外，输入过程中，要尽可能减少鼠标的使用，仅仅使用键盘，这样也可以大幅增加输入效率。原来的 AIMS 在这个方面实现的就比较好。因为原来的 AIMS 是基于 DOS 界面开发，所以尽管支持鼠标，但是同时可以完全实现用键盘操作，这对大量的数据输入无疑是十分有利的。因此，在对 CCMIS 进行需求分析时，该要求就被明确提出，但是，由于各种原因，这个要求在开发中，尤其是开发初期，并没有很好的实现，相关的具体问题，将在本文稍后进行阐述。至于数据的管理和统计功能，主要是根据管理层和业务人员的要求，加以实现的。

CCMIS 开发由公司一位副总全面负责，相当于项目经理，但是并不负责任何开发相关的事宜，而主要是领导和协调各方工作。CCMIS 就是由该项目经理带领 IT 开发人员以及部分业务部门人员（主要是进行需求分析或测试）共同进行，开发工具为 Sybase。

笔者作为箱管部的工作人员，参与了 CCMIS 的总体设计，而且参与了 CCMIS 中 CTS（Container Tracking System，即集装箱管理模块）从需求分析到测试的开发过程。本文即以 CTS 为主，CCMIS 其它模块为辅，说明软件过程管理在开发中的应用的。另外，笔者尽管参与了 CTS 的开发过程，但没有进行直接的代码编写，而是作为业务人员与 IT 开发人员沟通的桥梁，重点在需求、设计、测试等非代码编写阶段进行工作的。

现代的班轮运输，多采用集装箱化的方式进行，就是将货物装入集装箱内，再装船运输。采用集装箱化运输，可以更有效的保障货物，减少货损。另外，由于采用了标准的集装箱，也大大提高了码头的装卸效率。既然是集装箱化运输，集装箱在整个业务中，就扮演着重要的角色。

与集装箱相关的问题主要有两个方面：一是如何很好的利用现有集装箱；二是成本控制。对现有集装箱的利用主要是要加速集装箱流转速度，减少闲置箱。但是，由于集装箱数量庞大（以 CMA CGM 为例，全球集装箱保有量共有 40 万 TEU 左右），因此，如何利用好每一个集装箱就是面临的问题。对于集装

箱的成本控制，往往是人们容易忽略的，因为集装箱的成本与船舶相比很低，但是，因为集装箱数量庞大，聚沙成塔，集腋成裘，所产生的造箱、租箱、修箱等成本也是不容忽视的。箱管工作人员对 CTS 的要求主要就是能够准确的记录每个箱子的每一步动态，并提供方便快捷的数据输入和统计查询。

2.2 相关软件技术

下面，就在本文中将着重讨论的 CCMIS 开发过程中涉及到的软件技术作简单概括。

- ✧ 软件需求
  常用需求分析方法及软件需求规格说明书模板。
- ✧ 软件设计
  设计过程及设计模式的应用。
- ✧ 代码编写
  代码编写规范及代码审查。
- ✧ 软件测试
  常见软件测试技术－白盒、黑盒等。
- ✧ PSP－个人软件过程
  PSP 的基本概念和效用。
- ✧ SCM－软件配置管理
  SCM 主要功能－变更管理。

第三章 CCMIS 软件过程分析

3.1 对软件开发过程的认同及影响

也许很多读者看到本节标题觉得奇怪或者不可思议，的确，看起来软件开发是一个技术过程，而对开发过程的认同更趋向于主观因素，这两者间有什么关系？其实不然，下面，笔者就 CCMIS 开发过程中的一些对开发过程的认同问题及其不良影响进行简单阐述，希望大家以此为鉴。

时至今日，软件开发过程中，各种变更已经是家常便饭，已经被软件开发人员所认同。人们关注的焦点不应该是如果消除变更，而是如何对变更进行控制，因为变更是无法消除的。有一句话说得很好：“只有变更是不变的!”，这可以说是软件开发过程中关于变更的真谛。

然而，理论和实践总是有所差距的，变更这个人所共知的问题，在 CCMIS 开发过程中却不被 CCMIS 开发人员所认同。开发人员对业务部门工作人员甚至提出了这样的要求，一旦需求确定后，不得做任何变更。姑且不说当时的需求分析过程是否合理，就是合理的需求分析过程，也不意味着必然会得到最终确定的需求方案。而开发人员对开发过程中的变更采取抵制态度，以需求已经确定为由，拒绝进行更改。在此，仅举一简单实例，在开发 CTS 过程中，即使提出进行简单的数据输入界面更改，以利于快捷的输入，也被开发人员所拒绝。由于上述情况的影响，业务人员和开发人员不能进行良好的沟通，双方都采取了相互抵制的态度，结果给开发带来了巨大的不良影响。而且，开发人员所拒绝的变更，最终被证明无法满足实际需要，不得不按照业务部门的需求重新进行更改，但其时已不是开发初期，结果，造成了大量的人力、时间等浪费。

无论科学技术如何发展，人还是会犯错误，软件开发过程中，由于种种原因而引入缺陷更是不可避免。这是每个开发人员都会遇到的问题。如何正确面对缺陷，也应当引起每个开发人员的思索。尽管人们都承认自己会犯错误，但是人们往往趋向于逃避自己所犯的错误，或为自己的失误开脱，这是人的天性使然。然而，在软件开发过程中，每个开发人员都应直面自己所引入的缺陷，切不可采用鸵鸟政策，或者拒不承认，这样，将无法解决任何问题。

CCMIS 开发过程中，部分开发人员对缺陷的认识不够，就采取了逃避或者拒不承认的态度。当进行测试，发现缺陷时，部分人员往往不肯承认自己的失误，反而推卸责任，在向项目经理汇报时，声称所产生的数据错误是由于业务人员计算机知识不足，而操作错误而导致的。而当时由于项目经理与业务部门相关人员之间未能进行有效沟通，也就未能督促开发人员进行修改，导致缺陷在相当一段时期内存在，结果造成的数据错误，给业务部门增加了三倍左右的工作量来对数据进行修复，造成了相当的人力、时间浪费。

根据统计，绝大多数软件项目的开发周期都超过初始的预期，因为人们往往趋向于低估所可能遇到的困难。然而，人不是万能的，实际状况往往超出人们的预期，在软件开发过程中，开发人员所能做的就是尽量准确的对软件开发进行预估，而不可能准确的预报软件开发的过程。同时，由于开发人员的自身情况限制，对具体业务的了解往往不及普通职员。此时，万万不可断章取义，关门造车，凭自己主观臆测具体业务进行开发，这样，常常造成业务与开发出来的产品脱节，无法正常使用。

CCMIS 开发过程中，部分开发人员就因为过分低估困难，对业务部门的需求随意承诺，结果却无法如期或正确实现，而业务部门的人员及工作安排却如期进行，造成业务与 CCMIS 脱节，导致了大量的额外工作。而且，部分开发人员对具体业务不够了解，在未与业务部门进行有效沟通的情况下，就按自己的臆测进行开发，结果造成了不良后果。在 CCMIS 开发过程中，就曾出现某开发人员在对码头进箱这个过程进行分析时，凭自己的主观臆测，在未与相关业务部门进行沟通的情况下，自行开发，结果实现的功能与实际业务所需大相径庭。

上述由于对开发过程认同不一而引起的问题，经过业务部门向项目经理反映，由项目经理组织开发人员与业务人员开会进行交流、沟通，在 CCMIS 开发中后期已经消除或者大大减少了。但是，所造成的影响和损失却是无法挽回的。

亡羊补牢，未为晚也，总结经验教训，发现上述各种思想观念问题可以形容为一句老话：“统一思想，实事求是。”然而，口号或理论与实践总是有着或多或少的差距。许多看上去显而易见的道理，却不是很容易在实践中实施的。在今后的软件开发过程中，人们都应当牢记“统一思想，实事求是”这句话，这个指导思想不是任何技术层面的问题，但是影响却是及其深远的。

3.2 适当工具与软件开发过程

3.2.1 软件开发工具简述

随着软件技术的发展，越来越多的软件工具涌现出来。正确运用各种工具，无论是软件的分析设计、编码或者测试，都可以达到事半功倍的效果。所以，选择正确的工具是十分重要的。

不同的开发工具运用因各个软件项目的不同而不同。常见的软件开发 IDE（Integrated Developing Environment，集成开发环境）有 Microsoft 的 Visual Studio，Borland 的 JBuilder，Sun 的 Sun ONE Studio 等等，常见的软件设计工具有 Rational 的 Rose 等，常见的软件配置管理工具有 Rational 的 ClearCase 等。在当今软件开发中，有一种建模语言受到了软件设计人员的广泛青睐，这就是 UML（Unified Modeling Language，统一建模语言）。

复杂的软件项目的开发过程是一项工程，必须按工程学的方法组织软件的生产与管理，必须经过一系列的软件生命周期阶段，这一认识促使了软件工程学的诞生。编程仍然是重要的，但是更具有决定意义的是系统建模。只有在分析和设计阶段建立了良好的系统模型，才有可能保证软件开发的正确实施。正是由于这一原因，许多在编程领域首先出现的新方法和新技术，总是很快地被拓展到软件生命周期的分析与设计阶段。UML 正是由于上述思想理论的指导，经过多方努力研究，并经过一段时期的发展，而最终形成的一种建模语言。“UML 是吸收多种方法的成果、凝结许多组织和个人智慧的产物。UML 是一种用于对软件密集型系统进行可视化、详述、构造和文档化的建模语言，主要适用于分析与设计阶段的系统建模。”（《UML 参考手册》译者序）。

的确，UML 给软件开发人员提供了一种标准的建模语言，通过 UML，可以对软件进行完整而又详细的建模，同时，由于 UML 是标准、统一的，利用 UML 进行交流不会造成歧义，更有利于人们之间的沟通，建模语言的标准化将为软件开发商及其用户带来诸多便利。由于 UML 各种优点，许多公司的设计工具都支持 UML，比如 Rational 的 Rose，Borland 的 Together 等等，UML 也为越来越多的软件开发人员所认同和接受，并广泛应用到软件开发实践中去。

3.2.2 CCMIS 开发过程与软件开发工具

尽管有各种软件开发工具，但遗憾的是，CCMIS 开发过程中，除了 Sybase 的 IDE 之外，没有任何额外的工具被使用。这大大降低了 CCMIS 的开发效率。假使在软件的分析和设计时，使用 UML 进行，则必然可以使所有参与人员对 CCMIS 有一个总体上的观感。也许有的读者认为 UML 是主要针对开发人员，不是对所有参与者都适用。其实不然。UML 不仅针对开发人员，而且，对于软

件的最终用户也是很有益处的。如果在 CCMIS 分析设计时，使用 UML，则可以使业务人员对 CCMIS 完成后的情况有一个整体的认识，而且，甚至可以对某些细节进行辨析。例如，UML 的用例图（Use Case Diagram）可以使业务人员清晰的看到该用例是否正确反映了自己所需要的功能。

不仅在软件设计阶段，在编码阶段及测试阶段，CCMIS 的开发人员也没有采用任何工具，这不能不说是软件开发过程的一个不足。尽管，没有使用工具并不会直接给软件开发过程引入缺陷，但是，如果正确的使用工具，却可以大大降低缺陷率，从而提供软件开发质量和效率。

3.3 需求过程分析及改进

3.3.1 需求分析的历史发展及现状

现有的软件项目大多不能达到客户的预期，要么不能实现客户目标，要么不能对客户提供有效支持，其原因所在，就是没有完整、详尽的需求分析。

随着软件工程学的发展，需求分析也越来越为人们所重视。根据调查，软件项目中有百分之四十到六十的错误是在需求分析阶段引入的根源。而且，众所周知，缺陷越早发现，更改缺陷所花费的成本就越小。有人提出了一个 1:10:100 的定律：针对同一个缺陷，在软件开发早期的需求分析阶段进行更改花费成本如果为 1，在软件开发中期进行更改所花费成本就为 10，如果在开发过程后期或者软件发布后进行更改，则花费成本为 100。由此可见，如何减少需求分析时所引入的缺陷，并且对所引入的缺陷能够及时发现并进行更改是至关重要的。

正确的进行需求分析，得到合适的 SRS（Software Requirement Specification，软件需求规格说明书），是需求分析的主要目标。但是，SRS 只可能是高质量的，却不可能是完美的。在进行需求分析时，不能幻想得到一个完美的 SRS，这是不现实的。即使是高质量的 SRS，也是非常难以得到的，因此，软件项目的需求分析又常常被人们认为是开发过程中最为困难的阶段之一。

3.3.2 常用的需求分析方法及常见的 SRS 模板

为了能够进行正确的需求分析，得到合适的 SRS，许多专家、学者付出了艰辛的劳动，对需求分析进行研究，提出了各种需求分析的方法，并提供了不

同的 SRS 模板。软件开发人员可以根据 SRS 模板来简捷的获得合适的 SRS，从而节省大量的精力。

一般来说，需求包括两个方面的内容：功能性需求和非功能性需求（质量需求）。功能性需求描述了系统所应具有的外部行为，也就是系统各种功能；非功能性需求则描述了系统执行预期功能的性能指标，也就是系统执行功能的好坏程度。

与需求相关的主要活动是需求分析、规格以及需求变更管理。

一般来说，需求分析过程可以按照如下步骤：

- ✧ 准备需求采集和分析
- ✧ 采集需求
- ✧ 分析需求
- ✧ 准备 SRS
- ✧ 评审
- ✧ 用户确认

准备需求采集和分析，主要是对商业、技术背景进行了解，并且确定用户比较熟悉的工具和方法，以决定如何进行需求的采集。在对需求进行采集的时候，应当采用各种合适的工具和方法。不要奢求用户会明确的描述他们的需求，这是不切实际的。有时，用户自己也无法确定自己的需求，这就更需要开发人员与用户进行全面的交流、沟通，帮助用户了解并确定自己的真实需要。

应当采取各种方式进行准备活动和需求采集。开发人员可以深入用户的工作环境，了解用户的工作流程，与用户进行交谈，了解用户的业务状况，而且，应当多次召开联合会议，进行更加深入的需求收集和分析。在得到需求分析之后，为了做出正确的需求分析，应该对得到的需求和 SRS 进行反复论证和分析，以确保需求和 SRS 的质量。

收集需求时，有许多的工具可供使用。问卷即是其中之一。当需要从用户处收集标准信息或者需要得到明确解答，只有两种相反答案的时候，调查问卷是非常有用的。

无论是何种系统，也无论数据的存储方式如何，说明系统所存储的数据以及进出系统的数据都是非常重要的。对系统数据进行良好的说明，是正确描述需求的重要方面之一。对系统数据的说明，通常可以采用如下四种形式：

- ✧ 数据模型
- ✧ 数据词典
- ✧ 数据表达式
- ✧ 虚拟窗口

1. 数据模型说明了系统所要存储的数据以及数据之间的关系。如下图所示，给出了简单的集装箱与堆场的数据模型：

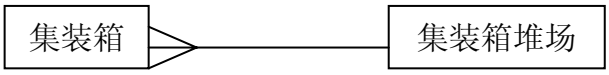

图 3.3.2-1

数据模型是通过框图来描述产品内、外的数据。利用框图表述各类数据类型，并通过连线表现其间的关系。就数据模型特点而言，能够比较精确的说明数据，但是，掌握数据模型却需要一定的时间和精力。因此，比较适合专家使用，对于一般用户是不太适合的。

2．数据词典是一个系统组织的、叙述性的数据说明。下面是一个数据词典的例子：

类型：集装箱
集装箱是班轮运输中容纳货物的容器……
属性：
箱号：文本，11 位字符
类型：文本，4 位字符……

图 3.3.2-2

数据词典的优点在于能够说明几乎所有细节及特殊情况，但是其缺点也是显而易见的，编写数据词典要花费大量的时间。而且，很难确定数据词典已经完备。

3．数据表达式，也称为正则表达式，使用简短而且可以体现数据结构的表达式来对数据进行说明，如下例：

集装箱＝箱号+类型+状态+……

数据表达式是描述序列的简洁公式，适合与描述复合数据，但是，不够直观，而且复杂的表达式往往难于理解。

4．虚拟窗口是理想化的屏幕图像，形同真是屏幕，但不具备功能或菜单。使用虚拟窗口在于可以直观的表示数据，便于用户理解。如下例：

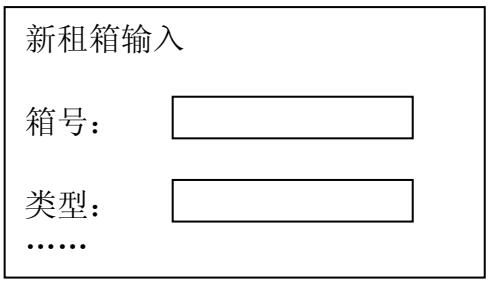

图 3.2.3-3

虚拟窗口的优点在于提供了简化的屏幕图像，并且表明了真实数据，借助与虚拟窗口，用户可以很好的理解系统数据，并发现特殊情况。而且，开发用户界面时，虚拟窗口也是很好的中间步骤。但是，如果对虚拟窗口不够熟悉，就会在虚拟窗口的设计上占用过多、不合理的时间。

有了对系统数据的描述，接下来关注一下功能性需求，在对功能性需求分析中，大多功能可以采用用例（Use Case，用况）来表述。利用 UML 中的用例图（Use Case Diagram），可以清楚的描述系统的任务。而且，用例对于用户来说，也是比较容易理解和接受的，而且用例是被广泛应用的，标准化程度也较高。但是，用例几乎没有提及任务所使用的数据，因此，对用例的这个缺点应当引起足够的重视。

UML 除用例图之外，UML 的其它各种图在需求分析中也是非常合适的工具。UML 中的各类动态图可以对功能性需求的细节进行合适的描述，而静态图的恰当使用，可以使各类数据、对象之间的关系更加清晰明了的表述出来。

除了功能性需求之外，软件的需求还包括非功能性需求（质量需求）。与功能性需求相比，非功能性需求更容易被用户忽略，或者表达不够清晰。此时，开发人员更加需要与用户的耐心沟通，仔细分析，挖掘出隐藏在深处的需求。

为了能够更好的进行需求分析，得到高质量的 SRS，多位专家、学者及机构对需求分析进行了研究，并提供了 SRS 模板工具。通过使用 SRS 模板，开发人员可以根据 SRS 模板中的列表来进行需求分析，不会遗漏重要事项，从而减少了许多繁琐的工作，可以节省大量精力，并且容易得到高质量的 SRS。常用的 SRS 模板有 IEEE830 模板和 Volere8 模板，相比之下，IEEE830 更趋向于对 SRS 进行描述，而 Volere8 则趋向于一个单纯的 SRS 模板。

3.3.3 对 CCMIS 开发需求过程的分析及改进

✧ 需求分析准备及前期工作

纵观 CCMIS 开发全过程，可以说需求分析阶段是相对来说最为符合软件工程规范的一个开发阶段了，从后文我们可以看出，相对来说，CCMIS 开发中对需求分析的重视程度远远超过了例如设计、编码、测试等等。不过，对需求的重视并非来源于开发人员自身，而是来自于项目经理。前文曾经介绍，CCMIS 的项目经理是由公司一位副总担任，该项目经理相对比较熟悉业务，因此也就比较重视业务中的需求。而且，由于项目经理身份特殊（副总），因此在资源调配方面有天然的优势，当 CCMIS 项目开发需要时，可以迅速调集所需资源（主要是人力），而不会有很大阻力。

给予项目经理相应的权利，这在项目开发过程中是至关重要的，只有这样，项目经理才能根据需要合理进行协同，从而保证项目开发的正常进行。因为对于不直接参与项目的人员或者部门来说，往往容易产生抵制情绪，为了防止项目的进行而影响本来的工作，对项目的进行采取消极的配合态度。为了最大限度的消除这种不利影响，应当赋予项目经理相应的权利，使其能够充分调用各类所需资源（包括人力、物力），以尽可能推动项目的进行。CCMIS 项目就此点而言，是非常有利的。由副总亲任项目经理，就自然拥有了许多权利，可以充分发挥项目经理的作用。

项目经理出于对需求的高度重视，亲自参与并组织了对需求的完整分析。

为了做好需求分析，项目经理提前召集开发人员、业务部门领导及部分业务骨干进行会议，要求全体人员对此提高认识程度，认真对待，并在散会后仔细思考，争取把需求分析的完整、详细、透彻。此后，就需求问题，项目经理还分别召集开发人员和各个业务部门领导、部分业务骨干进行多次会议，详细讨论各方需求。为了做到让开发人员充分了解业务需求，除了与业务人员进行会议之外，项目经理还亲自带领开发人员深入各个业务部门考察工作情况。而且，不仅本公司各个业务部门，与本公司业务相关的许多工作单位，项目经理也带领开发人员进行考察，以利用开发人员充分了解业务状况。最后，利用各地分公司和办事处业务人员出差的机会，召集不同地区的业务人员与开发人员会面，以便充分挖掘 CCMIS 的需求分析内涵。

✧ 开发人员的需求分析工作

然而，尽管做了许多工作，但是 CCMIS 的需求分析最终并不令人满意。

由于开发人员对业务了解不足，过分轻视需求分析，将业务过于简单化，

因此，开发人员在需求分析过程中没有投入足够的精力，其后果就是最终的需求分析根本无法满足实际需要。

在需求分析过程中，开发人员没有对业务人员的要求进行完整、详细的分析，也很少进行细致的记录。由于开发人员主观认为业务过程是非常简单的，这种先入为主的思想影响了开发人员对业务过程的分析，结果就无法做出良好的需求分析。开发人员往往只是在业务人员只是对工作进行了很简单的介绍是，就自以为对该工作有了足够的了解，然后自己进行臆测。另一方面，由于大多业务人员并不具备相应的软件开发背景，因此业务人员只能通过对业务的描述来反映自己对软件的要求。此时，仔细聆听业务人员对自己工作的描述，并且加以仔细分析是非常重要的。而且，众所周知，在需求分析时，往往用户自己也不知道自己的确切需求，这就更需要开发人员进行细致的分析并采用恰当的询问方式来获知隐藏的用户的真正需求。

CCMIS 开发人员在需求分析过程中，一方面过于轻视业务过程，对业务人员对工作的描述没有进行细致、完整的记录，同时进行详尽的分析；另一方面，开发人员往往要求业务人员直接说出对软件的要求，这对于业务人员来说，是很难做到的。业务人员往往只能表示工作是如何进行的，却很难表述对软件的功能性需求；至于非功能性需求，例如操作的便捷，界面的人性化，更是无从提及。此时，开发人员就根据业务人员的介绍，按照开发人员心目中的业务模式来向业务人员重新表示对软件的需求。而由于业务人员没有相关的软件知识背景，也就常常认为开发人员所表述的与自己的需要一致。结果，最后开发出来的产品其实与业务需要相差千里。

另外，在前文中，有过介绍，CCMIS 开发过程中没有采用任何工具。在进行需求分析时，开发人员与业务人员之间的沟通完全是通过语言交流。但是，单纯语言的表述是容易产生歧义和理解差异的，这也是实际完成的产品功能与业务需求相差甚远的原因之一。试想，如果采用了适当的工具，例如 UML，使用标准化的、无歧义的图文并茂的表示方式来表达需求，那么当开发人员重述自己对需求的理解时，能够有这样的工具使业务人员清楚了解自己所要表达的内容，那么，业务人员也可以及时对需求进行更改，而不是看到开发出的产品后，再去更改需求。后期的更改，正如 1:10:100 原则所述，造成了大量时间、人力的浪费。另外，如果采用了适当的 SRS 模板，也可以使开发人员对需求的关键所在能够有清晰的认识，根据 SRS 的描述，开发人员就可以对需求有相对比较准确、详尽的了解。然而，CCMIS 开发人员由于对需求及业务的轻视，重视程度不足，他们不仅没有采用任何 SRS 模板，甚至完整的 SRS 都没有，而只是简单的记录了会议过程中业务人员的一些要求，就进行 CCMIS 的开发了，其

结果是可想而知的。

开发人员对业务需求的忽视在软件开发后期有所改善，尽管如此，却已为时过晚。需求分析的错误给 CCMIS 的开发带来的后果是及其严重的。如上所述，许多需求分析的缺陷是在产品开发出来或者是原型开发出来后才被发现的。结果，有些缺陷按照实际业务需求进行了修改，因而导致了大量额外工作；更有甚者，还有一些缺陷根本无法修改，只好采用另外的补丁程序或者增加特殊业务流程的方法来进行解决，给业务部门人员造成了相当的不便。而且，问题还没有解决，随着 CCMIS 的使用，越来越多的由于需求分析而引入的缺陷凸现出来，更严重的是这些缺陷往往无法在软件中进行修改。由 CCMIS 缺陷而导致的不仅仅是软件问题，更加使业务人员对由于软件缺陷而增加的大量额外工作产生了抵制情绪，从而影响了正常的业务流转。业务人员常常戏言“一般的软件设计都是为了适应业务需求，而我们却是业务流程来适应软件设计，CCMIS 如何设计，我们的业务就要随之而作相应改变。”

相比之下，由于认识到上述问题，加之笔者对需求有足够的重视程度，因此，笔者所参与的 CTS 部分的需求分析相对比较完整和正确。通过对 CTS 的需求进行简单的分析，能够相对较为完整的向开发人员表述了业务需要，包括功能性需求及非功能性需求。而且，由于笔者对业务和软件开发都有所了解，因此能够较为恰当的对需求进行表述，没有过多的歧义产生，同时，对开发人员的表述也能够理解关键所在，并且，对 CTS 的需求进行了反复的分析论证。因此 CTS 部分的需求分析相对而言是比较完整和详尽的，其引入的缺陷也大大少于 CCMIS 其它模块。

✧ 改进方案简述

针对上述需求分析中所存在的问题，应当促使开发人员首先改变对需求分析的错误认识，另外，利用适当的分析方法和模板，提高效率并减少误解，最后，应当建立良好的变更控制机制，以处理各种变更。在此，推荐使用跟踪矩阵进行需求的跟踪。因为项目的基本目标是构建能够满足客户需求的软件系统，因此需要检测软件是否满足所有需求，为了能够进行有效检测，需求跟踪是非常重要的。

支持跟踪功能的最简单方式是进行从需求单元到设计单元、从设计单元到编码单元，从编码单元到测试用例的映射。而跟踪矩阵就是用来维护这种映射的。但是，跟踪矩阵必须及时维护更新，一个没有及时更新的跟踪矩阵是无效的甚至是有害无益的。

- CMM 中主要相关 KPA 及目标
  - KPA：CMM 第二级/需求管理
  - 目标：控制软件需求；为软件工程和管理活动建立基线，并使软件计划、产品和活动与需求保持一致；在客户和将处理客户需求的软件项目之间建立对客户需求的共同理解。
  - KPA：CMM 第二级/软件配置管理
  - 目标：软件配置管理活动纳入计划；选定的软件工作产品被标识、受控制和便于利用；被标识的软件工作产品的更改是可控的；受影响的小组和个人知道软件基线的状态和内容；在整个软件生命周期中建立并且维护软件产品的完整性。软件配置管理与软件需求相关的一个重要功能就是对需求的变更进行控制。
  - KPA：CMM 第三级/软件产品工程
  - 目标：软件工程任务被定义、集成并遵照执行，以生产软件；软件工作产品相互间保持一致；有效和高效的生产正确、一致的软件产品。需求分析和制定需求规格过程就是该 KPA 中的一项要求。

3.4 设计过程分析及改进

3.4.1 软件设计简述

现代软件项目越来越复杂，因此，软件的设计也越来越困难。没有正确的软件设计，就没有后续步骤的代码编写，测试等等，更不用说高质量的产品了。而软件设计，不意味着任何事情都要从头做起，如果能够复用以前的解决方案，那么既可以大大提高效率，减少在为设计而花费的精力，而且，由于复用以前的解决方案，引入的缺陷也大大减少。人们经过长期的研究、分析，将以往的解决方案归纳为不同的模式，每种模式解决特定的设计问题，一旦懂得了模式，许多设计的决策自然而然就产生了。而 Erich Gamma，Richard Helm，Ralph Johnson，John Vlissides（常被称为“四人帮”，Gang of Four）四人采集众家所长，最后将设计模式总结为 23 种，几乎所有的设计都包含在这 23 种模式，或者几种模式的组合之中。详尽资料，请参考由四人共同编著的《设计模式－可复用面向对象软件的基础》（*Design Patterns – Elements of Reusable Object-Oriented Software*）一书。利用这 23 种设计模式可以快速、便捷的进行软件设计，快速的得到软件设计的解决方案。

除利用设计模式进行软件设计以外，软件设计的过程（其实，软件开发的其它阶段也是如此）往往要反复进行评审，尽可能减少缺陷的引入。前文曾经提及，缺陷越早发现，修改缺陷所付出的代价就越小。

3.4.2 CCMIS 设计分析及改进

✧ CCMIS 设计过程浅析

CCMIS 由于是功能相对比较简单，不很复杂的软件系统，因此，设计问题相对来说也比较容易。开发人员并没有采用任何设计模式的思路，仅仅是根据直观的功能需求，进行了大致的设计，然后就投入到编码阶段了。或者说，开发人员仅仅是根据自己头脑中展现的一种“模式”来进行设计的。当然，并非开发人员的思路不可采用，但是，自己思考出的“模式”，通常无法与经过多人努力总结，千锤百炼出的那 23 种设计模式相提并论，其严谨程度和对问题的解决能力都要大打折扣。而且，开发人员也没有对自己的设计方案进行任何反复的评审，刚刚进行粗糙的设计，就匆匆进入编码阶段了。

设计的匆忙与粗糙主要是由于下列原因造成的。

一是开发人员自身的素质问题，几乎没有人真正对设计模式有所了解，甚至都不清楚设计模式的作用，因此，更不用说在开发中应用了。二是由于项目经理没有软件知识背景，在领导众人进行需求分析之后，就要求开发人员尽快进行“开发”，项目经理心目中的软件开发就是代码的编写。因此，开发人员也就没有时间对设计进行反复的评审。

因为 CCMIS 功能相对比较简单，因此尽管设计进行的匆忙、粗糙，却并没有引入过多和致命的缺陷，这与需求分析阶段引入的缺陷是完全不同的。需求分析中引入的缺陷，多数是由业务人员发现的。设计阶段引入的一些缺陷，例如一些数据共享的问题，还有一些人机界面的问题，往往是开发人员到了后期，自己都觉得不太合适，“看不下去”了，然后进行修改的。

✧ 改进方案简述

首先，就软件开发人员而言，应该提高自身的素质，对设计有方法和模式有基本的认识。其次，应该采取一个针对该项目而言相适应的设计过程，比如从概要设计开始至详细设计过程的规范确定，及过程的裁剪问题。

✧ CMM 中主要相关 KPA 及目标

✧ KPA：CMM 第三级/组织过程定义

- ✧ 目标：开发并维护组织的一个标准、适用的软件过程；与软件项目使用组织标准软件过程相关的信息被收集、评审并可用。

标准过程和裁剪是 CMM 第三级中组织过程定义 KPA 的要求。使用为确定项目过程而定的标准过程和裁剪是 CMM 第三级中集成软件管理 KPA 的要求。使用过程还是 CMM 第三级中软件产品工程的要求，它要求工程活动遵照定义好的过程执行。

3.5 编码过程分析及改进

3.5.1 软件编码规范及编码审查

现代软件项目越来越庞大，功能也越来越复杂，往往需要大批人员的共同协作，进行软件的开发。传统的手工作坊式的软件开发已经不适应发展需要了。由于大批开发人员共同协作，编码的标准化、规范化，就尤为突出和重要了。

一般来说，编码要求有规范的格式，适当的注释，而且易读性要好。这样，有利于大家共同协作。下面是一些通常的编码规范要求：

- ✧ 源程序的文件头部说明必须包含文件名称、软件版权、功能说明、系统版本、开发人员、开发时间和修改记录等几个部分
- ✧ 源程序的修改记录说明必须包含修改日期、修改人员、修改说明，每次修改为一条记录
- ✧ 正确的注释
- ✧ 恰当的空行
- ✧ 整齐的缩进
- ✧ 适当的对齐
- ✧ 长行要拆分
- ✧ 命名要规则，便于理解和记忆

除上述这些通常的编码规范之外，根据每种不同语言的特点，往往还有不同的符合各自语言特征的规范。

代码编写完毕后，并非就是万事大吉。开发人员应当对自己的编码进行复审，然后在开发人员之间进行同行评审（Peer Review）。

其实，不仅仅是在编码阶段，在开发的任何阶段，开发人员自行对自己的工作进行复审，然后进行同行评审都是很有作用的，可以大大提高开发质量，清除缺陷。根据统计，复审和同行评审可以发现相当数量的缺陷，有实验表明，

良好的复审和同行评审可以发现百分之六十一的缺陷，而对于一些即使通过测试也难于发现的缺陷，复审和同行评审更是有着巨大的优势来清楚缺陷。

3.5.2 CCMIS 编码过程分析

参照前述的编码规范和审查方式，CCMIS 的编码相对很不完善，开发人员在编码规范方面只是做到了最基本的格式比较整齐。开发人员除在 Sybase 的 IDE 中进行代码编写时，有适当的缩进、空行等格式外，没有自行添加任何的注释。至于代码说明，更改情况、日期等等就更是一片空白。也许因为开发人员认为 CCMIS 十分简单，无需任何说明和注释。其后果是大家可想而知的。而且，开发人员从来不对自己编写好的代码进行复审，更不用说同行评审了。只要编译器通过，开发人员就不对编写好的代码进行另外的检查，编码是否有缺陷，完全依赖于编译器和业务人员在使用时发现（测试的问题将在后文进行介绍和分析）。由于缺乏良好的编码习惯，带来的后果也是非常严重的。

由于开发人员不对编写好的代码进行复审，引入的缺陷往往在业务人员使用中才表现出来。造成正常的业务中断，等待开发人员进行修改，造成不必要的时间浪费，尤其在有紧急业务事件需要处理时，更是给业务人员的工作带来了极大的不便。而且，由于没有添加适当的注释，开发人员在听取了业务人员发现的缺陷后，需要修改代码时，往往无法确定缺陷所在，而是浏览相应功能全部代码来寻找相关的代码部分，再来查找缺陷。这样查找缺陷，经常花费大量的时间。开发人员在修改缺陷后，同样不对代码进行复审，只要编译通过，就宣布使用，因此，清除缺陷的效率是很低的。而且，由于没有适当注释，对缺陷代码的修改只是局部修改，没有将涉及到的代码全部进行修改，结果常常就是开发人员刚刚宣布缺陷已经修改完毕，业务人员就立刻发现了新的缺陷。于是，新的一轮查找又开始了，又要再次花费时间和精力进行修改。与没有适当注释同样的道理，没有对每次修改进行恰当的记录，也导致了缺陷的清除无法准确进行。开发人员经常在修改代码之后，由于业务人员发现新的缺陷，当对代码进行再次检查时，发现前次修改不正确，应当恢复修改前的代码，对另外的部分进行修改。然而，此时没有对前次修改的完整记录，因此开发人员只好根据记忆来恢复对代码的修改。当恢复不完全时，就又引入了新的缺陷。

根据不完全统计，几乎所有的代码缺陷修改都存在上述问题，而且，对一个缺陷的修改至少要反复两次，一般在三次至四次。如果新引入的缺陷凸现较早，该段被修改代码还处于开发人员的记忆所及范围之内，再次修改的速度就会快一点；如果新引入的缺陷经过一段时间才被发现，开发人员的记忆已经力

所不及，则他们往往要阅读大量的代码段，然后定位缺陷发生的代码段，才能进行修改。在后一种情况下，业务人员只好忍受有缺陷的系统，利用额外的工作来弥补由于系统缺陷而带来的不足，直至缺陷修改完毕；或者，将相关业务延迟至缺陷修改完毕。这种情况直到 CCMIS 开发后期，也没有丝毫改善。甚至，某开发人员还声称自己几天没有进行和开发有关的工作，“早就忘了”曾经要求实现某业务功能的代码段位置和当时编码是如何实现的了。

✧ 改进方案简述

对于代码编写所存在的问题，首先是要使开发人员明确代码编写规范的目的，不是为了“好看”或者让别人“感觉好”，而是确确实实为开发带来方便。良好的代码编写风格，比如适当的缩进、空行，尤其是准确的注释，有助于对代码的正确理解，增强代码的易读性，并在修改时提供便利，这才是规范编码的最终目的，不应该为了规范而规范，舍本逐末。其次，采用代码复审的方法，不要认为这是浪费时间，应该将代码复审看作是代码编写工作的一部分，并认真执行。还有就是同行评审，不要惧怕自己的缺陷暴露在他人面前，这样，才会更有效的消除缺陷。

✧ CMM 中主要相关 KPA 及目标

- ✧ KPA：CMM 第三级/同行评审
- ✧ 目标：同行评审纳入计划；尽早高效的使软件工作产品的缺陷被识别和清除。

同行评审可以在相当程度上发现各种缺陷，从而大大提高软件质量。而对于编码的同行评审，更是清除编码缺陷的重要手段之一。

3.6 测试过程分析及改进

3.6.1 软件测试发展及现状

测试是软件开发工程中重要的环节之一。有句话可以形象的说明测试的重要性：“没有测试，就没有质量，没有质量，就没有项目，没有项目，就没有业务，没有业务，就没有利润。”

一般来说，测试阶段花费的时间比软件开发的任何其它阶段都要多。然而，对软件测试却往往存在许多误解。

很多开发人员就简单的把软件测试等同于“Debugging”，还有许多开发人员认为测试的目的就是证明软件是正确的。这些都是错误的观点。

软件测试应该看作是对“Debugging”的支持。测试的目的是显示软件的缺陷，而“Debugging”的目的是发现造成缺陷的原因并修改之。另外，从实施的角度来说，测试与“Debugging”也有许多不同：

- ✧ 测试是严格的，可预计的，有计划进行的；“Debugging”则是相对比较随意的
- ✧ 许多与测试相关的工作是不需要软件设计背景知识的;不具有软件设计知识是无法进行“Debugging”的
- ✧ 测试可以由非开发人员进行；“Debugging”必须由开发人员之下
- ✧ 测试可以实现某种自动化；“Debugging”不可能有任何自动化

至于利用测试来证明软件的正确，早在 1969 年，Edsger Wybe Dijkstra 就指出“软件测试可以用于证明存在的错误，而决不能用于证明不存在错误”。

一般来说，软件测试可以根据不同标准分类如下：

- ✧ 按类型区分：功能测试，即黑盒测试，以功能需求为基础，不需要了解程序内部结构；结构测试，即白盒测试，检查程序的内部设计，需要详细了解程序结构
- ✧ 按方法区分：静态测试；动态测试
- ✧ 按级别区分：单元测试；集成测试；系统测试；验收和安装测试

无论何种测试，在测试前都应该制定详尽的测试计划。

对于白盒测试，一个基本原则就是要保证程序的所有条件及分支都至少执行一次。但是，由于白盒测试常常由编写代码的开发人员设计（因为他们最了解程序的结构），就会由于开发人员的经验而造成偏见，因为人们往往趋向于逃避自己的错误。另外，如何保证程序的所有条件、分支，以及其各种组合都进行了测试，也是一个难于解决的问题。尽管如此，白盒测试发现的缺陷数字仍居于各种测试技术之首。有统计显示，在大型项目中，通过完整的路径和参数测试，可以找出 72.9%的缺陷。

对于黑盒测试，一般存在两个典型问题，一是要求明确的需求，另一个是只能涵盖一小部分的可能测试条件。因此，如何正确选择测试用例，是进行黑盒测试时的重点所在。

一般情况下，单元测试常常采用白盒测试的方式，集成测试可以看作是连接白盒测试和黑盒测试的桥梁，集成测试主要测试软件的功能。其后，进行系统测试，最后是验收和安装测试。

尽管根据软件过程管理的理论和实践，软件测试并非是查找和清除各种缺

陷的最有效方法，比如复审和同行评审对查找和清除缺陷就有很高的效率，但是，软件测试仍然是软件过程的重要部分，不容忽视。

3.6.2 CCMIS 软件测试分析

✧ 测试计划

CCMIS 的测试过程是完全由开发人员组织和安排的。万幸的是，开发人员对测试的重视程度与对需求及编码规范相比还是相当高的。这主要是因为项目经理要求在开发人员认为测试通过后，如果出现问题，开发人员应对此负责，因此开发人员才对测试比较重视。

尽管开发人员对测试比较重视，可是由于开发人员自身原因，没有任何完整、详尽的测试计划曾被制定。然而，测试过程中，开发人员没有对任何模块进行过单元测试，没有采取白盒测试的方法来测试程序的结构。前文曾述，白盒测试发现缺陷的数字是各种测试技术首屈一指的。然而，由于开发人员没有进行任何的白盒测试与代码审查，因此，大量的缺陷就被带入了下一步的黑盒测试（功能测试）。

✧ 测试执行

CCMIS 开发过程中唯一进行的测试就是在某一部分开发“完成”后，进行功能测试。而且，测试并不是由开发人员进行，而是由开发人员指定业务人员进行测试。就以笔者所参与的 CTS 为例，当开发人员编码完毕，某功能“完美实现”后，就要求业务人员进行测试。如前文曾述，对编码没有任何事先的审查，只要编译通过，就进行所谓的“测试”。而且，开发人员往往只使用一个数据就进行功能测试。笔者曾就此问题提出置疑，但开发人员认为只要一个数据就足够了，不必进行多个数据的测试，而且认为业务人员，不应对开发人员的事情进行置喙。只要一个数据能够正确的输入、输出，开发人员就认为该功能测试通过，然后进行下一步的开发。这种“测试”方法带来的恶果就是常常表面上测试通过，当进行下一次测试或实际使用时，采用另外数据时，缺陷就显现出来。以某次输入集装箱箱号为例，当输入某一箱号时，程序正常运行，当输入其它箱号时，就会出现错误，严重时，程序崩溃。最后，经过反复检查，才发现是程序对箱号的检验算法有缺陷，没有全面考虑到各种箱号的可能字母与数字组合的范围。

测试时，黑盒测试能够涵盖的可能条件本来就很少，加之此前没有进行任何白盒测试，又及开发人员只采用极少量的数据进行测试，因此，更是有大量

的缺陷隐藏在程序当中，无法被及时发现。当这些缺陷暴露出来之时，往往开发人员以及进行了下一步的开发，甚至是软件已经投入使用了。此时，开发人员对所暴露的缺陷需要花费大量的时间去查找，工作效率相应的也极为低下，也给业务人员的正常工作带来大量的不便。

✧ 压力测试

众所周知，系统测试时应当包括压力或者容量测试，针对数据库相关的软件更是如此。然而，CCMIS 在进行“测试”时，开发人员根本没有考虑程序承受压力的问题。由于箱管部门每天处理大量的集装箱数据，因此笔者对程序所能承受压力问题比较关心。然而，由于开发人员认为业务人员对开发过程不应过多参与，笔者只好组织部门内业务人员自行进行一些相当简单压力测试，然后再向开发人员说明问题所在。当只有一人使用 CTS 的时候，程序表现还算正常，各个功能能够使用，当有两个人同时使用 CTS，速度大大降低，当三个人或以上同时使用 CTS，其速度已经令人无法忍受了，甚至出现程序崩溃的现象。当笔者向开发人员展示测试情况的时候，开发人员才认可对大量数据容量的考虑是正确的，然后进行修改。一个偶然的机会，从另外的开发人员处得知，原来的 CTS 所用数据库，开发人员贪图方便省事，竟然没有建立任何索引，因此，当仅有一人使用时，一切似乎正常，多人使用时，速度就大大下降，甚至程序崩溃。幸好，最终 CTS 开发人员接受了其他开发人员的建议，对数据库建立了索引，基本上解决了上述问题。

数据库中多个进程对同一数据的更改是一个非常值得注意的问题，和数据库管理系统有关的书籍对此多进行专门讨论。CTS 开发人员对这个问题却没有足够的重视，更没有在测试时考虑该问题。于是，同前文所述缺陷一样，也是由业务人员自行进行测试的。当多个用户对同一个集装箱记录进行操作时，系统往往产生死锁，或者多个用户的对数据的更改都被记录，而不是按照设计时的业务操作逻辑进行更新，由此经常产生莫名其妙的，不符合业务流程的数据记录。当就此问题向开发人员提出时，开发人员经过一段时间的努力，却无法轻易清除该缺陷。最后，开发人员竟然要求部门业务人员使用 CTS 时，事先约定，不允许多人同时对一个集装箱的记录进行操作。这样，业务人员在使用 CTS 时，就要不断互相询问，以防有可能操作同一个集装箱记录。至今，该缺陷也没有解决。

除箱管部所用 CTS 之外，订舱部门所用 CCMIS 部分也出现了与程序所能承受压力有关的类似缺陷。开发人员没有对程序进行压力测试，只是简单的测试了功能。结果，当多人使用订舱系统，尤其是进行查询时，就会出现死锁，

只好退出系统，再次进入后查询。

✧ 不良测试的后果

由于 CCMIS 开发人员没有采用正确的测试方法，因此测试没有起到应有的效果，无法有效的发现程序的缺陷。结果，大量的缺陷在使用时才暴露出来，给开发人员和业务人员都造成了极大的不便。

✧ 改进方案简述

首先，对测试的目的有正确的观念，测试不是“挑毛病”，而是增进软件质量的重要手段。其次，对了解各种测试方法并适当采用，不要“嫌麻烦”或者对“发现缺陷”有恐惧心理。在此基础上，还应当在测试之前建立完整的测试计划和足够的测试用例，依此来实行测试。这样，才会达到测试的真正目的，减少缺陷，增强软件质量。另外，如果可能，应该建立专门的测试小组来进行测试。因为程序员天生不能有效测试自己开发的程序，这个问题在软件开发组织中普遍存在。程序员不仅对自身工作中存在的错误具有内疚感，而且以及完成的创造性工作也会使他们产生一定偏见。程序员并不一定是不想面对自己的错误，而是根本看不到自己的错误，因此，最后有独立的测试小组来进行测试。

3.7 PSP 与开发过程

CMM 为软件工程提供了有效的实行方法和实践方式，为软件过程改进提供了具有指导意义的框架。然而，CMM 主要的焦点是在软件企业，对于小型的软件开发团队以及个人软件开发行为，CMM 没有过多的涉及。为填补个人及团队的软件开发过程管理及改进，SEI 提出了 PSP（Personal Software Press，个人软件过程）和 TSP（Team Software Press，团队软件过程）。PSP 和 TSP 可以看作是 CMM 的“低级”表现形式，为个人和团队如何实行软件过程管理及改进，提供了具体的实施方法，为软件企业实行 CMM 打下基础。

“一个软件工程师的任务就是要在预定的时间中交付预定质量的产品”（Watts S. Humphrey），PSP 的目标就是帮助个人成为更好的软件工程师。PSP 可以应用于多个方面，例如个人工作的管理，技能的提高，更好的计划，业绩的度量（纵向，本人的不同阶段，不可用于不同人员之间的度量），等等。

类似于 CMM，PSP 也分为不同的级别，级别越高，说明个人软件开发过程越成熟。PSP 的各个级别及特征如下表：

PSP 级别		PSP 各级别特征
PSP0：个人过程基线	PSP0	时间记录；缺陷记录；缺陷类型标准
	PSP0.1	编码规范；规模度量；过程改进计划
PSP1：个人计划过程	PSP1	规模预估；测试报告
	PSP1.1	任务计划；进程计划
PSP2：个人质量管理	PSP2	代码复审；设计复审
	PSP2.1	设计模板
PSP3：个人过程循环	PSP3	过程改进循环

表 3.7-1

不仅如 CMM 一样分为不同级别，PSP 也包含许多 KPA，而这些 KPA 与 CMM 的 KPA 有相当一部分是重叠相交的。简单描述，PSP 可以用下面的一个公式来表示：

时间控制 + 缺陷控制 = PSP

时间在 PSP 中是非常重要的，而且，PSP 的起点可以说就是对时间的记录。只有对时间有了一个详尽的记录，才有可能对时间进行控制，从而实行 PSP。PSP 的另外一个重要记录，就是缺陷。同样，对各个引入的缺陷有了详尽的记录，才可能对缺陷进行总结，并进一步控制。

在时间记录和缺陷记录的基础上，采取各种方法对时间和缺陷进行控制，并不断提高，就可以较好的实行 PSP 了。此处，着重强调一下代码复审。个人代码复审几乎是最有效的发现和清除代码中缺陷的方法了。代码复审就是研究源代码，并从中发现错误。代码复审最好是在源程序编码完成后并在编译和测试之前进行。在刚完成编码时是最容易发现缺陷的，因为这时程序员还可能记住本人的意图。代码复审更高效的原因是因为代码复审时所看到的是缺陷本身而不是缺陷的表象。在编译和测试阶段，发现和清除缺陷需要经历从发现缺陷表象到确定缺陷的过程，所以时间花费就大得多，而代码复审可节省这些时间。当然，代码复审也有缺点，主要就是耗时太久并且难于恰当的进行。

为了配合 PSP 的实行，SEI 提供了一系列表格，以便 PSP 中的记录。根据这些表格记录的内容，可以看出，就 PSP 的实行来说，与其说是一种“技术”，不如说是“习惯”。

然而，PSP 往往不为人们所认同，尤其是对 PSP 的持续实行，让人们无法理解，更不用说养成“习惯”了。许多人认为 PSP 实行时所填写的大量表格是一种浪费，这是完全不对的。首先，要对记录有全面的认识。要知道，大量的填写表格是为了对工作过程进行总结，不是“为了记录而记录”，记录的目的是为了完整的记录工作，以便总结。其次，记录一定要准确，详尽，否则，对大量记录的总结就无法得出正确的结论。当持续进行一段时间的记录之后，通过对记录的分析，常常可以看到某些关键所在，对工作的提高可以有的放矢，改进效率和质量。

在对记录进行总结时，特别值得注意的是，PSP 的记录可以用来针对同一个人的不同时间段评估其工作情况；但决不可以用来进行横向比较，据此对不同人员的工作进行评估。而且，要向所有人员解释 PSP 的目的及其主旨思想，否则，工作人员往往为防止真实的记录对自己造成不利影响，而伪造记录，这样，PSP 就失去了实施的意义。

PSP 在实行之前，需要经过一定的培训，通过培训，使人们对 PSP 的真正目的和作用有所了解，对于如何根据自身特点实行 PSP 就会有一个认识。而且，只有在经过培训之后，PSP 的实行才有可能显现其真实功效，而不是流于形式，仅仅是填写大量的表格，为了表现 PSP 而实施，最后不得不终止。PSP 的关键是通过大量数据的记录进行总结和挖掘，以提高自己的工作质量和效率。

就 CCMIS 的特点来说，如果能够实行 PSP，将会大幅提高开发人员的工作效率和质量。因为 CCMIS 是相对很小的项目，个人的工作质量对项目的整体影响是很大的，往往“牵一发而动全身”。没有 PSP 及正确的软件开发过程，CCMIS 开发人员对自己的工作质量和效率都无法有效进行控制。往往是项目经理由于业务繁忙，没有过问时，开发进程没有什么很快的进展，一旦项目经理对开发进度过问，开发人员就加班加点进行开发，只是为了完成项目经理要求的进度，其质量可想而知。接下去，就是对缺陷的修复，结果，看起来是按项目经理要求的计划完成了开发进度，但根本是无效产出，当开发人员认为已经将缺陷清除“完毕”，又过去相当一段时间了，然后就是对下一段开发进度的“追赶”，如此恶性循环。

其实，PSP 的理念不仅仅可以用于软件开发，在许多工作中，都可以应用 PSP。笔者就曾经利用 PSP 的方法，对工作进行了三个星期的记录，总结之后，对工作中的部分方面进行了更改，结果工作效率有了相当的提高。

3.8 SCM－软件配置管理

3.8.1 SCM（软件配置管理）概述

软件开发过程中，产生很多产品，包括各种文档、程序、数据和手册等。这些产品都是易于改变的。甚至，开发过程中，需求也可能变更。为避免软件项目在发生变更时失控，正确控制和管理变更是非常重要的。SCM（Software Configuration Management，软件配置管理）就是软件项目管理中专门对发生的变更进行关注及控制的部分。

SCM 的一个基本目标就是控制变更活动，管理变化，SCM 的功能主要有以下几项：

- ✧ 显示软件项目的状态
- ✧ 表明程序的最新版本
- ✧ 处理更改请求
- ✧ 撤消更改
- ✧ 防止未授权的更改
- ✧ 变更之间的跟踪
- ✧ 显示相关变更
- ✧ 采集当前源程序、文档和其它信息

实现 SCM 的上述功能，通常可以采用以下几个方面的机制：

- ✧ 命名规则和文件组织

  根据一些标准的命名规则为文件命名，并对文件进行有计划的组织，有助于迅速查找所需文件。正确的文件命名有助于在不参考其内容的情况下，理解其本质。
- ✧ 状态信息维护

  状态信息的维护，有助于对软件项目的状态进行快速了解和识别。
- ✧ 版本控制

  版本控制是 SCM 的关键使用，而且有很多工具对此进行支持。没有版本控制，则许多 SCM 的功能无法实现，版本控制有助于对经过多次更改后的旧版本的保留。
- ✧ 变更跟踪
- ✧ 访问控制
- ✧ 协调规程
- ✧ 修改记录

SCM 非常重要，应当作为一个独立的过程，应当制定详尽的计划，然后执

行。许多人幻想开发过程中没有变更，这是毫不现实的。从软件项目开始起，就与变更相伴，在软件开发过程中，“唯一不变的就是变更”，这个意识应当深入人心。

人们往往趋向于消除变更。人们希望完成需求分析后，需求就被永久确定，再也没有改变。同样，设计完成后，也被期望能够永久确定。人们对变更的认同往往只局限于代码的修改。这对于良好的软件开发过程是十分不利的。如果抱有消除变更这种心态，人们就容易逃避现实，不愿承认变更，因此就不会计划任何 SCM，当产生变更时，没有足够的措施来解决问题。与其采用“鸵鸟政策”逃避事实，不如事先正视变更问题，详尽计划 SCM，当变更发生时，进行正当的控制，防止变更失控。

3.8.2 CCMIS 开发过程中软件配置管理

CCMIS 尽管是小型的软件项目，但 SCM 也是必不可少的，因为变更随处可见。然而，CCMIS 开发人员没有进行任何 SCM 准备。

前文曾述，由于开发人员对需求没有足够的重视，因此开发之后，业务人员在测试或者使用中提出的需求与开发人员开发时认为的需求有很大不同，造成大量的更改。开发人员对此是没有丝毫准备的，当然，由此引起的更改已经不仅仅是 SCM 的问题，而且就笔者个人观点，也不是 SCM 可以解决的了。另外，CCMIS 开发人员对变更持否定态度，认为变更是不应该存在的，无论业务人员因为何种原因提出变更，开发人员都认为是不应该的，是多出来的“麻烦”。然而，由于交流沟通的问题，开发人员开发出的产品与开发前向业务人员介绍的总有很大区别（至少，业务人员认为如此），因此，即使经过业务人员的确认，之后再进行更改的情况也屡屡发生。业务人员曾就软件界面问题提出更改，然而，开发人员认为这是完全不应该的，是业务人员的错误，是“找麻烦”。并且，在业务人员提出几次更改之后，开发人员向项目经理进行“投诉”，认为业务人员的更改要求影响了他们的正常工作，影响了正常的开发进程。开发人员就曾“正告”某些业务人员，如果再有任何更改，他们将不予理会，不会再考虑实际使用问题，其后果由业务人员自行承担。

即使对采取的变更，CCMIS 开发人员也没有采取任何正确的措施来管理。无论是需求变更、设计更改还是代码的修改，开发人员都没有任何记录，也没有对旧版本有任何保留。其实，对几个需求变更的几个版本进行分析，往往有利于发掘真正的需求。业务人员常常没有任何软件开发经验，因此自己也不容易对需求有详尽的描述，这样，就造成了需求的多次更改，加之开发人员对需

求的重视不够，没有深入了解，需求的变更就更多了。因为开发人员对需求更改没有记录，因此就单纯按照业务人员要求进行更改，就难于符合真正的业务需要，而且，有时，需要撤消更改时，开发人员也难以恢复到原来的需求。

同样，开发人员对代码的修改也没有任何的记录，而且也不对旧版本进行保留。由于代码修改有相当部分是开发人员编码时引入的缺陷，因此，开发人员没有对业务人员有类似于“不得更改”的要求。但是，由于没有对代码的修改采取任何记录，当进行下一次修改，尤其是与前次修改相关时，由于无法了解上次修改情况，只能依靠开发人员的记忆来进行相关修改。更为严重的是，当要撤消某次更改时，由于既没有修改记录，又没有保留旧版本，也就只能依靠开发人员记忆来恢复，取消修改。在这种情况下，几乎每次对代码的修改和撤消修改都会带来大量的问题。

标准的命名规则和有效的文件组织也是 SCM 的重要机制。由于 CCMIS 开发人员各自为政，因此没有统一的命名规则，只是每个开发人员对自己的工作有一个标准的命名规则。但是，由于开发中，每个开发人员负责的部分相对比较独立，因此，开发人员之间没有统一的命名规则并没有造成过多的麻烦。但是，某些开发人员贪图方便，在开发后期，即使是本人的工作，也不遵从自己的命名规则。另外，开发人员对文件的组织也非常混乱。就笔者了解，某些开发人员把所有相关的文件都放入一个目录中，就算是进行文件组织了，甚至子目录都没有。由于没有标准命名规则和有效的文件组织，在开发中，当需要进行查询时，效率就十分低下了。有几次，开发人员甚至向笔者索要一些早期由于开发需要，大家共享的文件，因为开发人员声称那些所需的文件“找不到了”。

3.9 相关文档问题

软件开发过程中，会有许多文档产生，包括需求相关、设计相关、源代码、使用手册等等。这些文档的准确与及时更新对于软件开发是非常重要的。本节将特别对 CCMIS 使用手册的情况进行简单阐述。

尽管业务人员曾经多次提出，而 CCMIS 开发人员也曾承诺编写并提供 CCMIS 使用手册，然而，不知何种原因，也许是 CCMIS 开发人员对此不够重视，也许认为没有必要，或者希望 CCMIS 全面投入使用而且缺陷清除完毕再着手该项工作，迄今为止，没有任何使用手册提供给 CCMIS 的使用者。

众所周知，软件的使用手册是非常必要的。用户经常要参考使用手册来了解如何使用软件产品。通过对使用手册的学习，用户可以加深对软件的认识，

清楚软件的各种功能，了解软件的使用方法，从而提高效率。而且，用户如果通过使用手册来了解软件，也减轻了由于向开发人员寻求技术支持而给开发人员带来的负担。

CCMIS 开发人员没有向业务人员提高任何使用手册，因此，业务人员在使用中只好依靠自己摸索，或者向开发人员咨询。然而，开发人员与业务人员相比毕竟人数较少，无法提供足够的支持，CCMIS 的使用大多要依靠业务人员自行摸索，由于这个原因，业务人员的工作效率在相当长的时期内非常低下，比如快捷键的使用、不同窗口间的跳转，业务人员只好摸索使用。而且，有些业务功能由于没有使用手册的介绍，业务人员认为 CCMIS 没有提供，只有手工进行业务操作；或者，向开发人员提出该业务功能的需求，而造成开发人员的误解。

缺乏使用手册的另一个后果是当业务人员向开发人员咨询时，开发人员本人经常记忆不清，因此，他们就需要重新查看源代码，然后向业务人员回答。这给双方都带来了大量的不便，增加了许多不必要的麻烦。

使用手册的编写在技术上应该不存在很大的问题，关键在于对它的认知程度和开发人员的责任心。缺少使用手册或者使用手册编写不完整，多是因为开发人员认为手册不是很必要，或者编写很麻烦，因此不应花费时间和精力在使用手册的编写上。因此才导致了使用手册的姗姗来迟，这是急需纠正的错误态度。

第四章 CCMIS 开发过程改进方案总结

通过上文的阐述，根据 CCMIS 开发过程中存在的问题，本章将针对其改进方案进行一个系统化、结构化的总结。

4.1 人员培训

任何软件开发过程都是由人来完成的，所以软件开发过程是以人为本的。可以说，人员素质将直接决定软件产品的质量。

此处，培训的人员不仅仅针对开发人员，应当包括所有开发的参与者，例如，项目经理、软件产品的最终用户等等。同时，培训也不仅仅是针对开发人员的某项技能的培训，还应该包括使所有人员对软件开发过程树立正确认识的培训。

通过适当的培训，可以使开发人员和业务人员对软件开发有一个正确的认识，这样，在开发过程中，双方的配合将会更加默契，可以避免许多误会，从而大大提高效率和降低缺陷的引入。而且，通过培训，也有助于使开发人员了解更多、更好的软件开发理念、工具，从而为提高软件开发的效率和质量打下良好的基础。

对 CCMIS 开发过程的各种改进，归根到底都是以对人员的素质要求为基础的，因此，对人员的培训就是项目开发的基石。然而，这也是相当困难的一个环节，如何对人员进行培训，进行何种培训，进行培训的成本如何，都是作为 CCMIS 项目管理者需要进行考虑的。

就 CCMIS 的特点而言，其对人员的培训计划采用“短平快”的方式是最适合的。由于时间和资金的限制，很难对参与人员进行完整、系统的培训，因此，培训的主要目的就在于使参与的人员能够“符合开发需要”就可以了。这样，不需要占用过多的人力、物力，只要针对 CCMIS 的开发需要有的放矢，对人员进行合适的培训即可。

例如，针对 CCMIS 开发过程进行改进，并强调以过程为中心。短期内不一定要所有相关人员对以过程为中心的理论有深刻理解和体会，只要他们对此有所了解，并能够按照改进后的过程进行实施就可以了。全面的学习和培训可在

今后的适当时机进行。

再如，针对需求过程中使用 SRS 模板的问题，对相关人员的培训大可不必进行完整、系统的理论学习，只要找到适当的模板，并且学会使用就可以了。

而且，对人员的培训也不一定全部集中在开始阶段，完全可以在开发过程中视需要而定。例如对 UML 的使用，就可以在需要使用 UML 时，进行适当的培训，以使相关人员能够适应开发的需要。

在进行适当的培训之后，无论是项目经理还是开发人员抑或业务人员，所有的参与者对 CCMIS 的开发有一个较为全面、正确的认识，而且开发人员的理念和思想也有了转变，这样，就可以一步建立正确、适用于 CCMIS 的开发方式，并改进其开发过程了。

4.2 确立以过程为中心的开发方式

CCMIS 原开发过程仍然是以产品为中心的，当然这和参与者的认识有关。经过培训之后，所有人对软件的开发应该有一个良好的、正确的认识，此时，应当确立以过程为中心的开发方式。

过程是需要不断进行改进的，姑且不论过程是否正确，但是，在观念上首先应当有所改变，不应当秉承传统的以产品为中心的开发方式，为了产品进行“救火”式的开发，没有良好的开发过程；或者，有过程而不严格遵守，从而产生不可预测的后果。

确立了以过程为中心的开发方式之后，应当针对 CCMIS 的具体情况进行剪裁，建立一个适当的开发过程，当这个开发过程建立之后，任何人、任何时候都不可以随意对这个开发过程进行更改，一定要严格遵守这个开发过程，从而真正实施以过程为中心的开发方式。

在确立以过程为中心的开发方式时，也许会遇到相当的阻力，因为，在大多数人看来，过程远远不及产品那么“实在”，有点无法把握的感觉，因此不容易接受以过程为中心的观念。因此，要通过适当的培训，使人们转变对软件开发的认识，对其有一个整体上的改观。

也许在实施初期，建立的开发过程会有诸多不足，但是，即便如此，也要坚持以过程为中心，而不能随意更改过程，当确实需要时，针对过程进行系统的分析和改进，而不是抛弃过程。

4.3 开发过程的裁剪

“No silver bullet”（没有银弹），读者们应该都听说过。软件开发也是如此，没有普适的开发过程。因此，针对每个具体软件项目，都要对开发过程进行裁剪，以适应该项目的需求。CMM 包括 18 个 KPA，涉及到了对开发过程的各种不同要求，但我们应该采用“拿来主义”，而不是生搬硬套，照单全收。

CCMIS 是一个很小的软件开发项目，资金少、时间短、人员不多、技术难度较小，针对 CCMIS 的这样的特点，我们只需针对其关键过程，向 CMM 中相应 KPA 的目标靠近就可以了。CMM 对开发过程的要求，尤其是第四级、第五级，相对来说，更适合大型软件企业。要达到 CMM 第四级、第五级，需要大量的时间和资金支持，这对于小型软件企业或者开发团队是未必合适的。现实中，还是有许多类似于 CCMIS 这样，由企业内部人员针对某一具体目标进行开发的软件项目。对于该类开发团队来说，首要问题是能够在开发的产品质量令人满意的情况下，使其尽快投入使用，而不是针对各类软件开发均能承担开发任务。因此，CCMIS 的开发过程更注重在容易引入缺陷的环节进行改进，而不是将 CMM 作为实现目标。根据这一原则裁剪出的 CCMIS 开发过程，将更适合 CCMIS 开发的需要。

4.4 软件开发工具的应用

也许严格来说，软件开发工具的应用和软件开发过程管理关系不是很大。但是，就 CCMIS 的开发过程而言，这的的确确是一个急需改进的方面。由于没有使用适当的工具，无论是开发人员和业务人员之间的沟通，还是开发人员自身的开发过程，都受到了很大的影响。

就工具而言，建议在软件的设计中使用 UML，用标准化的表述方式来描述软件的设计开发，这样可以减少许多误解和歧义，避免许多不必要的麻烦。另外，使用其它一些工具，也可以使开发的效率大大提高并降低缺陷的引入。

当然，开发工具应用的目的是为了开发，而不是为了应用而应用，因此，开发人员应当根据实际需要进行调整，不是盲目的追求使用各种工具，而是采用最适合开发需要的工具。

CCMIS 是一个很小型的软件项目，如果使用过多的工具，反而过于累赘，降低开发效率，因此，除非必要，不需使用大量的工具，而是选择最有必要采

用的工具即可。

4.5 若干主要开发阶段的过程改进

4.5.1 需求过程改进

CCMIS 需求过程的改进，首要问题是所有人员，尤其是开发人员对需求的重视要有所改观。经过适当的培训，所有人员对需求都会有一个正确的认识态度，认识到需求的重要性及可能带来的相应不良后果。在对需求有了一个正确认识的基础上，进行需求收集和分析。

现行的需求收集和方法很多，不必拘泥于某一方法或形式，可以采集众家之长，从而确定最适合的方法。但是，无论何种方式，都应该注意，需求的收集不可以遗漏，要全面、完整的收集各方面的需求，否则，无论后期工作如何努力，结果往往会南辕北辙。

在采集需求之前，应做好各类准备工作，了解需求的业务背景，清楚业务人员的操作习惯和方法，并确定采集方案。

CCMIS 的需求采集可采用集中与分散相结合的方式。

所谓“集中”，就是项目经理、开发人员及业务人员通过会议共同进行讨论、研究，当然，在会议召开之前，各相关人员应当对会议的内容、目的、议题有清楚的了解，并对自己在会议中将要讨论的内容有所准备。“集中”采集需求的方式可以使相关人员在相对较短的时间内讨论较多的议题，并且各方人员可以共同集思广益；但是，这种方式要求各个参与人员一起集中，对日常的工作流程有一定影响，尤其是业务人员，往往要牺牲工作时间来配合。

所谓“分散”，就是开发人员利用零星时间，与业务人员交流，了解业务人员的需求。采用“分散”的方式，不需各方人员集中，可以灵活进行，对日常工作影响较小；但是，该方式不利于信息的集中，往往会导致不同业务人员提出不相符甚至矛盾的需求，从而给开发人员确立真正的需求带来不便。

采集需求的手段也多种多样。当对需求有一定了解时，可以采用问卷的方式进行采集，尤其是针对某些选择和是非类的需求时，问卷相当有用。开发人员制定好问卷后，发放给相关业务人员，根据他们的答案，对 CCMIS 的需求进行采集。

在收集了一定需求的基础上，开发人员可向用户提供一个原型，是用户对 CCMIS 开发后的情况有一个了解，并收集用户的反馈，对需求进行改进。

另外，在与用户进行交流时，开发人员不应使用模糊不清的语言，应采用无歧义的描述，适当的时候，使用 UML 等工具对 CCMIS 进行描述，是用户与开发人员的交流实现“标准化”。对于 CCMIS 中的系统数据，应针对不同人员进行描述。对于开发人员，可采用数据模型、数据词典等方式进行描述，而针对业务人员也需要了解的系统数据，则适合采用虚拟窗口的方式进行描述，使业务人员有直观、清晰的认识。

需求分析的目标是确定从收集来的信息所获得的需求是完整、正确、一致和无二义性的。在对 CCMIS 进行需求分析时，开发人员应当采用“换位”的思维方式，考虑业务人员的真正需求，应该明白业务人员对软件开发的理解有一定局限，因此，开发人员应当深层次的挖掘隐藏在表面以下的真实需求，而不是仅仅满足于表面上业务人员提出的需求，这是远远不够的。为了更好的进行分析，开发人员在可能的情况下，应当尽力了解业务人员的工作，只有这样，才会对需求做出相对正确的分析。

在对需求进行了正确收集分析后，应当编写需求规格说明书（SRS），这是需求过程的最主要的产品。由于需求规格说明书的编写实际上是相当繁复的，因此，应当采用现有的 SRS 模板，可以综合各种模板的长处，编写 CCMIS 开发的 SRS。通过使用 SRS 模板，可以避免许多可能的遗漏，避免需求阐述的不完整。

当 SRS 编写完毕之后，再由用户进行最终确认，当各方对 SRS 均认可时，需求过程才算告一段落。

软件需求过程基本是贯穿软件开发过程始终的，除了前期的收集、分析之外，还要对需求的变更及实现进行跟踪和控制。

对需求的变更进行管理，可建立一种模板，对 CCMIS 每项需求变更进行记录。模板中包括变更请求号、日期、变更说明、影响分析、进度影响、工作量影响等等。除变更模板外，还应进行变更记录，对所有需求变更情况进行汇总，并记录该变更状态（打开/关闭）。通过上述方式，可以对需求的变更进行详细的跟踪分析，并进行管理。

对功能性需求的实现进行跟踪，建议使用跟踪矩阵。跟踪矩阵由电子表格来实现，其中包括功能性需求编号、需求描述、使用数据、实现情况、测试用例等等。通过跟踪矩阵，可以清晰的观察 CCMIS 开发过程中功能性需求实现的情况。

对于需求变更记录、变更汇总及跟踪矩阵，存在着至关重要的问题，就是对它们的维护，如果没有及时进行维护，它们就失去了意义，并且会带来负面影响。因此，在使用它们的同时，还应及时对它们进行更新，以使其反映最新、

最准确的信息。

软件需求对于一个软件项目来说，重要性是不言而喻的。对于类似 CCMIS 这样开发技术难度相对较小的小型软件项目来说，其需求过程更是占据整个软件开发过程的很大比重，因此，对于 CCMIS 的开发来说，软件需求过程应该给予相当的重视。

4.5.2 设计过程改进

就 CCMIS 开发过程而言，其设计过程主要应该在两个方面进行改进。

一是模式的应用，如果复用现有的设计模式，可以减少许多重复设计的问题，并且可以使设计相对比较完善。当然，不是为了应用模式而应用模式，因为 23 种主要设计模式毕竟是经历众多锤炼，经过时间考验的，因此，如果能够正确应用，将大大简便设计过程，并且可以减少引入的缺陷。然而，对设计模式的应用，又对开发人员提出了新的要求，就是他们要对设计模式有一定的认识。这需要通过培训和学习来实现这一目标。

第二方面就是对设计的复审及同行评审。由于 CCMIS 是很小型的项目，开发人员较少，因此对相互间的开发模块内容也有相当了解，因此，开发人员之间进行同行评审就具备了有利条件。当 CCMIS 开发人员对自己负责的模块设计完毕时，首先应自行进行复审，然后再进行同行评审。其实，不仅仅是在设计过程中，在其它过程中，复审和同行评审也是非常必要的。不过，在设计过程和编码过程中，复审和同行评审的重要性更为突出。因此，这在开发过程中是必不可少的。

4.5.3 编码过程改进

代码编写差不多是软件开发中最“古老”的过程了。而且，对于代码编写方式大家也是见仁见智，有众多方式。但是，就软件工程的角度而言，代码的规范性、易读性、易理解性、易修改性是不可或缺的。

CCMIS 开发过程中，针对编码过程的问题，一是开发人员注重代码的规范编写，不能为因为贪图一时方便，随意编写，没有固定格式，没有注释，此类代码为将来的编辑、修改埋藏下了隐患。

CCMIS 开发的编码过程，应当严格遵守编码规范。源程序文件头部的说明必须完整，有正确的注释，每一次修改，都记录在案。修改记录中应对该次修改有完整、详尽的说明。除上述记录外，对缺陷的引入和清除也应明确记录，

并对此有详细的描述。

CCMIS 编码过程中另外一个非常重要的问题就是对代码的复审和同行评审。开发人员对代码的自行复审和开发人员之间进行的同行评审是提高代码质量、发现缺陷的重要手段之一。对代码的复审和同行评审可以检查出相当的缺陷，尤其是一些即使进行测试也难于发觉的缺陷，往往可以通过复审和同行评审得以清除。因此，CCMIS 的编码过程中，当开发人员对某段代码编写完毕时，应立即进行复审。此时，由于开发人员对该段代码及其功能有完整记忆印象，因此，进行复审往往是最有效果的。除开发人员自己对代码进行复审之外，由 CCMIS 的其他开发人员对自己的代码进行同行评审也是清除缺陷的有力手段。因此，CCMIS 编码过程中，要对代码的复审和同行评审给予高度重视并实施

4.5.4 测试过程改进

一个正确的测试过程应当包括测试计划的订立、测试用例的确立、执行测试、得到测试结果等等。人们也许都会很重视测试，但是，却往往对测试的作用没有正确的认识，测试不是为了证明软件“没有问题”，而是为了发现缺陷。

CCMIS 的测试过程也应当从对测试的认识入手，应当使开发人员对测试的目的有正确的认识，明确测试是为了发现缺陷，而不是表明软件产品的“完美无缺”。明确了测试的目的之后，应当制定完整的测试计划，在测试中应用多种测试方法，并确立多个测试用例，而不是简单的使用几个用例后即宣告测试结束，并且严格按照制定的测试计划进行测试，不可按照开发人员的“兴致”，随意进行测试。

针对白盒测试，确保所有逻辑分支都被执行，对各种条件组合都进行了测试。对于黑盒测试，应建立足够多的适当测试用例，尤其是各种边界值，应进行严格测试，以最大可能的清除缺陷。另外，如前曾述，针对 CCMIS 大数据量的特点，应进行多次压力测试，以检验 CCMIS 的性能。

为了更好的改进测试过程，在情况允许的情况下，应当由专门的测试人员进行测试，是开发人员和测试人员分离开来。针对 CCMIS 的具体情况而言，也许这个要求太过“奢侈”，不过，这的确是改进测试的一个良好途径。

4.6 个人软件过程 PSP 的实施

PSP 对于开发人员个人而言是非常有效的提高开发效率、质量并降低缺陷

引入的方法之一。对于类似 CCMIS 的小型软件项目而言，相对来说，也许开发人员个体的重要性要超过大型软件项目中开发人员个体的重要性。那么，如何尽可能的提高开发人员的技能和效率，是至关重要的。PSP 的实施将会给 CCMIS 的开发过程带来前所未有的思维方式和改进方法。通过 PSP 的实施，开发人员可以更加清楚的了解自身，并根据本人情况对开发中的具体问题进行更改。

然而，PSP 的实施需要投入相当的精力，因此，在确立 CCMIS 软件开发政策时，应当把 PSP 的实施作为一条原则，进行规定。另外，为了防止 PSP 流于形式，更重要的是使开发人员对 PSP 有一个正确的认识，明确 PSP 的真正目的。

首先要对 CCMIS 开发人员进行适当培训，PSP 起步并不复杂，因此，培训也不需花费很大精力。当开发人员对 PSP 有了一定认识后，即可初步实施。由 CCMIS 开发人员根据 SEI 的表格，自行记录开发过程中的时间、缺陷，当有了一定记录是，就对时间、缺陷进行分析，从而实现对时间和缺陷的控制。

实施 PSP 的关键问题在初始阶段，CCMIS 开发人员往往认为 PSP 花费过多精力来进行各种记录，而且，由于没有进行记录的习惯，常常遗忘。正如前文曾述，PSP 可以说不是“技术”问题，而是“习惯”问题，因此，只有当所有人员都从内心认识到 PSP 的意义所在，并自发自觉的实施 PSP 时，其功用才会体现，并发挥其最大效力。

4.7 软件配置管理 SCM

CCMIS 开发过程中软件配置管理（SCM）的主要问题是开发人员怕“麻烦”，结果却导致了更多的“麻烦”。

要明确变更的不可避免性，不要存在侥幸心理，不要采取鸵鸟政策，妄图逃避变更，这是 CCMIS 开发人员应当转变的观念。

由于变更的不可避免，因此必须对变更进行跟踪和控制，这也是 SCM 存在的原因。对于 CCMIS 这样的小型项目，建立一个单独机构来控制管理变更有些不现实，但是，SCM 还是要实行的。

实施 SCM 首先是确定 SCM 计划和构建配置管理。为配置管理制定计划包括确定配置项、制定用于控制和实现对这些配置项的变更的规则。配置项的一些典型例子包括 SRS、设计文档、源代码、测试计划、测试数据、项目使用的标准、培训文档、用户手册、合同文档等等。

各个开发人员应当对所有文档、代码进行有效的组织、归档、分类，并且实施版本控制，对各种更改记录在案，尤为重要的是，对与更改的文档或代码

有关的其它文档或代码进行及时更新，在正确的位置储存和正确修改，否则，错误的记录不如没有记录。

不使用工具进行 SCM 是非常繁琐、复杂而且容易出错的。现有各个软件公司提供的多种SCM工具可供选择，如Microsoft出品的Visual Source Safe（VSS），Borland 出品的 StarTeam 及 Rational 出品的 ClearCase 等。经过权衡，CCMIS 开发小组准备考虑采用 ClearCase 作为 SCM 工具。ClearCase 功能十分强大，但是，这也意味着开发人员需经过努力学习，才能掌握其使用方法，发挥其功用。

由于 CCMIS 项目自身客观条件限制，SCM 也许无法独立完整的实施，也不能有专门人员对 SCM 进行管理和控制，但是，开发人员应该在尽可能的范围内，实行 SCM，而不是不予理睬或不予考虑。

4.8 其它

除上述问题外，CCMIS 开发中，还有一些细节需要改进，例如前文曾述，缺少使用手册而造成的影响。以及其它一些文档相关的问题。这些看起来不是什么大事，但是，一个良好的软件开发过程和得到的软件产品应当尽可能为用户提供方便，因此，不应该因为这些是“小”事情，就不予重视。

第五章 结论

5.1 CCMIS 软件开发过程概要

通过前文对 CCMIS 开发过程的介绍与分析，我们可以看出，CCMIS 开发过程与现有的软件工程的标准和方式还相去甚远，没有采用有效的软件过程管理方法进行软件开发。

CCMIS 是一个比较小的软件项目，但是，“麻雀虽小，五脏俱全”，在 CCMIS 的开发中，不可避免的涉及到软件过程的各个方面，因此，软件过程管理也是不容忽视的。如上文所述，由于开发人员主观认识不足，过于轻视该项目的开发，另外，由于开发人员自身技术水平的限制，因此，CCMIS 的开发过程可以说是历尽艰辛，而且，至今，CCMIS 还未能完全投入使用。

CCMIS 的开发过程，除在项目启动时，以及需求分析过程的个别阶段之外，还是秉承一种手工作坊式的软件开发流程。开发人员各自为政，没有统一的项目规划和开发计划，没有确定的软件开发过程（或者说该过程是由开发人员自行随意确定的，而不是一个规范的开发过程），由此，CCMIS 的质量和进度都无法得到有效保证。

CCMIS 的开发过程不是面向过程的，并非基于正确的过程来保证软件项目，而仅仅定位于可以得到一个最终结果－CCMIS。固然，软件项目的开发最终是要得到软件产品，但是，仅仅面向产品的软件开发过程常常无法符合软件工程学的开发过程规范，由此，也难于得到质量、进度及成本都符合要求的软件产品。面向过程的开发，并非不注重最终产品的获得，而是通过规范的开发过程，可以对软件的开发有正确的预期，使软件的开发可控，并得到期望的产品。而单纯面向产品的开发过程，往往是不可控的，无法进行正确预期的开发过程，也就难以得到期望的产品。但是，以面向过程的软件开发却不是很容易让人们接受，因为该开发面向的不是“实实在在”的产品，是有些“抽象”的过程。

CCMIS 开发过程的问题，可以说是许多类似的小型开发项目都存在的。

首先，因为项目“小”，因此，开发人员对项目重视程度不够，认为没有什么难度，因而没有谨慎的对待开发过程的每一个阶段。其次，该类小型项目往往时间要求比较紧迫，因此项目经理不断询问开发人员的进度。此种情况下，

开发人员感受到一定的压力，为了向项目经理呈现开发的进度“符合要求”，即使存在确定的开发流程，开发人员也往往“从简”，由此也为产品的质量埋下了隐患。最后，也是关键所在，开发人员囿于自身的技术水平所限，知识比较陈旧，没有基于过程开发的观念和经验，因此无法建立规范有序的软件开发过程。软件开发技术不断更替，日新月异，这种情况下，开发人员应当保持自身知识的持续更新，才不会落后于时代。就笔者所了解的情况，CCMIS 开发人员开发观念基本上还是基于以产品为目标，“个人英雄”式的开发形式，对软件开发过程没有确定的认识，也没有相关的概念。因此，CCMIS 开发人员对开发过程没有投入足够的精力，没有确定一个规范的开发过程，造成了开发过程混乱，无法得到符合要求的软件产品的后果。

由于上述三个问题，导致了 CCMIS 开发过程趋于无序化，缺乏规范的开发过程来遵循，其后果是质量、进度都无法预期和控制的软件产品。

5.2 CCMIS 软件开发过程改进方案概要

首先，软件开发过程都是由人作为主体来实行的，因此 CCMIS 开发过程首要解决的问题就是开发人员自身的技术水平。应当通过学习或者培训，使开发人员的自身素质有大幅度的飞跃。在开发人员自身的技术水平有了提升之后，他们对开发过程的观念和看法必然与当前有所不同。在开发人员自身的观念发生转变之后，才有可能正视软件开发过程，从而为软件开发打下良好的基础。

当软件开发人员有正确的开发过程理念时，在开发前对相关的业务人员进行一定的宣传，使业务人员对项目有整体上的大致了解，这样，在开发人员与业务人员沟通时，会减少许多不必要的困扰。

在各方人员对项目都有一定了解的基础上，确立以过程为中心的开发方式，制定规范的开发过程，对项目的开发过程进行标准定义和剪裁，并把遵循开发过程作为一种制度来确定。无论何种情况，都要遵循预先制定的开发过程并不得随意更改。

剪裁后的开发过程应当详尽、规范，并适合 CCMIS 项目的自身特点。而且，该过程包括软件开发的各个阶段和各个方面，从需求分析到测试结束，从编码规范到配置管理，都应当涵盖其中。

通过以上改进，应当可以确立一个良好的软件开发过程，从而进行有效的软件项目开发，确保产品质量和进度，控制开发成本，实现既定目标。

5.3 CCMIS 软件开发过程遗留问题

前文对 CCMIS 的软件开发过程进行了分析，并着重针对其开发过程的不足之处提出了一些问题。以下，是一些前文未曾涉及或讨论不够详尽，但在将来的开发过程中准备加以考虑并给予一定重视的方面。

首先，是对开发人员的培训。其实，培训计划本身就是 CMM 第三级的一个 KPA，其目标就是将培训活动纳入计划；提高关于执行软件管理和技术的人员所需要的开发技巧和知识的培训；培育个人的技能和知识，使其有效的履行各自的角色。软件工程如何通过培训来提高人员的素质，是规范软件开发过程的一个重要手段。没有培训，就难以使相关人员的主观思想与软件过程管理的主旨相一致；没有培训，人员的技术水平素质就难以得到进一步的提高，从而对限制了软件开发过程的改进。因此，在今后的软件项目中，对人员的培训要作为一个重点来处理。没有对开发人员的培训，就无从谈起实施 PSP、TSP 乃至 CMM 等等软件开发过程技术。而且，为促进各种开发工具的运用，也需要对开发人员进行适当的培训。

管理学中有一个“木桶理论”，相信大家都很熟悉，就是一个木桶的装水的容积并非取决于最长的那块木板，而是取决于最短的那块木板。同样，对于一个软件开发团队，尤其是面向过程的软件开发团队。“最长的木板”或者说“个人英雄”所起的作用将越来越小，而且，对于越来越复杂的软件开发，无论多长的“木板”，所起的作用都是有限的。现代的软件开发，需要的是面向过程的团队的集体行为。因此，无论从个人观念还是技术角度，对人员的适当培训都是必不可少的。

其次，对 CCMIS 开发过程的剪裁。对开发过程的剪裁，看似容易，其实不然。针对 CCMIS 的特点，可以对开发过程进行，但是，如何剪裁出最适合的开发过程，绝非轻而易举。对于与 CCMIS 相类似的软件开发项目，如何进行剪裁，怎样能花费最小的代价找到最合适的开发过程，是值得反复思考的。

再次，TSP 的实行。CCMIS 是一个小型项目，而且现有状况是开发人员各自为战，彼此之间没有很多配合。如果某个项目需要开发人员共同努力，或者 CCMIS 在将来的发展需求迫使开发人员进行配合，TSP 的应用对其开发过程是大有裨益的。与 PSP 类似，TSP 主要是针对团队软件开发的过程规范和相应准则的。通过实行 TSP，可以针对一个软件开发团队确立规范的开发过程，从而保证团队开发项目的质量和进度。

最后，SEPG（Software Engineering Process Group，软件工程过程组）的成

立。SEPG 通常由专家组成，专门负责软件开发过程的监督、改进及实行。SEPG 的工作焦点就是软件的开发过程，因此是非常重要的，SEPG 通常要由技术和经验都很丰富的专家组成，以利于确立、改进软件开发过程。对于 CCMIS 开发集体类似的小型开发团队，是否组建 SEPG 是要经过一定权衡的，并非一定有 SEPG 才是有利的，通常，只有在大型软件开发企业中，SEPG 才是必不可少的。

软件工程的理论和实践随着时代的发展越来越受到人们的关注，而软件开发过程管理也越来越为人们所重视。软件过程管理的方法和实施也不断的在发展。本文中，笔者仅就本人的一些知识，CCMIS 的开发过程进行了简单分析，并对相关知识进行了简要介绍。由于笔者水平所限，缺点和不足在所难免，敬请各位读者提出宝贵意见。

谢谢！

参考文献

Watts S. Humphrey，《软件过程管理》，清华大学出版社，2003

Watts S. Humphrey，*A Discipline for Software Engineering*，人民邮电出版社，2002

Sami Zahran，《软件过程改进》，机械工业出版社，2002

Pankaj Jalote，《CMM 实践应用－Infosys 公司的软件项目执行过程》，电子工业出版社，2002

Soren Lauesen，《软件需求》，电子工业出版社，2002

Karl E. Wiegers，《软件需求》，机械工业出版社，2000

Erich Gamma，Richard Helm，Ralph Johnson，John Vlissides，《设计模式－可复用面向对象软件的基础》，机械工业出版社，2000

Ian Sommerville，Softerware Engineering，6th Edition，机械工业出版社，2003

风浪，“用项目管理法实施 SCM”，《程序员》，2003 年第 02 期

陈雷，“软件测试实践之测试计划”，《程序员》，2004 年第 09 期

潘加宇，“打开需求之门－－在中国实践用例技术的感悟”，《非程序员》，第 41 期（2004 年 9 月）

致谢

短短两年的软件工程硕士（Master of Software Engineering，MSE）学习就要结束了。作为中国首批 MSE，我们要走的路还很长。

由于是首批 MSE，因此无论对学校、老师、还是我们自己，都是一个尝试的过程，其中的艰辛与甘甜，我们曾亲身体味。

完成这篇文章，要感谢李旻老师的指导，感谢李旻老师的审阅和指正。

两年的学习过程，还要特别感谢邓冰老师，感谢他将我带入软件工程的圣殿。

同样要感谢各位为我们辛苦授课的老师，在我们学习过程中给予的各种无私指教。

还要感谢和我同组的各位同学，我们共同奋斗两年，一起上课学习、完成作业，往事仿佛就在昨天。祝愿我们的情谊地久天长。

最后，感谢我的家人及所有支持我的朋友们，感谢你们在我两年学习生活中对我的鼎力支持。