# 07-05-Software

Created on 20220605.

Last modified on 2022 年 6 月 5 日.

# 目录

# Chapter 1 Introduction

软件工程

已经完成的题目 2018, 上半年，上午。2020, 下半年, 上午。

# Chapter 2　软件系统结构

## 2.1　软件重建工程的理论与技术研究

## 2.2　软件规格说明的形式方法与 CASE 工具研究

# Chapter 3　SoftwareDesign

## 3.1　软件工程与结构化开发

不同模块中相同程序块提取出来，块内语句没有关系，这属于巧合内聚，影响模块之间的耦合关系。

支持共同代码拥有和共同对系统负责是 XP 中的代码共享（代码集体所有权）

### 3.1.1　设计各阶段

接口设计主要基于需求分析阶段的数据流图

架构设计，定义软件的主要结构元素及之间的关系

设计软件模块结构时，减少高扇出结构可以改进设计质量，将相似功能模块合并不能。

结构化分析输出包括：数据流图，数据字典，加工逻辑。不包括结构图。

测试时，路径覆盖数独立的路径数量，语句覆盖在路径覆盖基础上减掉没有语句的路径。

### 3.1.2　极限编程（XP）的 12 个最佳实践

1. 现场客户（ On-site Customer ）　　XP：要求至少有一名实际的客户代表在整个项目开发周期在现场负责确定需求、回答团队问题以及编写功能验收测试。　　评述：现场用户可以从一定程度上解决项目团队与客户沟通不畅的问题，但是对于国内用户来讲，目前阶段还不能保证有一定技术层次的客户常驻开发现场。解决问题的方法有两种：一是可以采用在客户那里现场开发的方式；二是采用有效的沟通方式。　　项目：首先，我们在项目合同签署前，向客户进行项目开发方法论的介绍，使得客户清楚项目开发的阶段、各个阶段要发布的成果以及需要客户提供的支持等；其次，由项目经理每周向客户汇报项目的进展情况，提供目前发布版本的位置，并提示客户系统相应的反馈与支持。2. 代码规范（ Code Standards ）　　XP：强调通过指定严格的代码规范来进行沟通，尽可能减少不必要的文档。　　评述：XP 对于代码规范的实践，具有双重含义：一是希望通过建立统一的代码规范，来加强开发人员之间的沟通，同时为代码走查提供了一定的标准；二是希望减少项目开发过程中的文档，XP 认为代码是最好的文档。对于目前国内的大多数项目团队来说，建立有效的代码规范，加强团队内代码的统一性，是理所当然的；但

是，认为代码可以代替文档却是不可取的，因为代码的可读性与规范的文档相比合适由一定的差距。同时，如果没有统一的代码规范，代码全体拥有就无从谈起。 项目：在项目实施初期，就由项目的技术经理建立代码规范，并将其作为代码审查的标准。3. 每周 40 小时工作制（40-hour Week） XP：要求项目团队人员每周工作时间不能超过 40 小时，加班不得连续超过两周，否则反而会影响生产率。 评述：该实践充分体现了 XP 的”以人为本”的原则。但是，如果要真正的实施下去，对于项目进度和工作量合理安排的要求就比较高。 项目：由于项目的工期比较充裕，因此，很幸运的是我们并没有违反该实践。4. 计划博弈（Planning Game） XP：要求结合项目进展和技术情况，确定下一阶段要开发与发布的系统范围。 评述：项目的计划在建立起来以后，需要根据项目的进展来进行调整，一成不变的计划是不存在。因此，项目团队需要控制风险、预见变化，从而制定有效、可行的项目计划。 项目：在系统实现前，我们首先按照需求的优先级做了迭代周期的划分，将高风险的需求优先实现；同时，项目团队每天早晨参加一个 15 分钟的项目会议，确定当天以及目前迭代周期中每个成员要完成的任务。5. 系统隐喻（System Metaphor） XP：通过隐喻来描述系统如何运作、新的功能以何种方式加入到系统。它通常包含了一些可以参照和比较的类和设计模式。XP 不需要事先进行详细的架构设计。

评述：XP 在系统实现初期不需要进行详细的架构设计，而是在迭代周期中不断的细化架构。对于小型的系统或者架构设计的分析会推迟整个项目的计划的情况下，逐步细化系统架构倒是可以的；但是，对于大型系统或者是希望采用新架构的系统，就需要在项目初期进行相信的系统架构设计，并在第一个迭代周期中进行验证，同时在后续迭代周期中逐步进行细化。 项目：开发团队在设计初期，决定参照 STRUTS 框架，结合项目的情况，构建了针对工作流程处理的项目框架。首先，团队决定在第一个迭代周期实现配件申请的工作流程，在实际项目开发中验证了基本的程序框架；而后，又在其它迭代周期中，对框架逐渐精化。6. 简单设计（Simple Design）

XP：认为代码的设计应该尽可能的简单，只要满足当前功能的要求，不多也不少。 评述：传统的软件开发过程，对于设计是自顶而下的，强调设计先行，在代码开始编写之前，要有一个完美的设计模型。它的前提是需求不变化，或者很少变化；而 XP 认为需求是会经常变化的，因此设计不能一蹴而就，而应该是一项持续进行的过程。 Kent Beck 认为对于 XP 来说，简单设计应该满足以下几个原则： A. 成功执行所有的测试； B. 不包含重复的代码； C. 向所有的开发人员清晰地描述编码以及其内在关系； D. 尽可能包含最少的类与方法。
对于国内大部分的软件开发组织来说，应该首先确定一个灵活的系统架构，而后在每个迭代周期的设计阶段可以采用 XP 的简单设计原则，将设计进行到底。 项目：在项目的系统架构经过验证后的迭代周期内，我们始终坚持简单设计的原则，并按照 Kent Beck 的四项原则来进行有效的验证。对于新的迭代周期中出现需要修改既有设计与代码的情况，首先对原有系统进行”代码重构”，而后再增加新的功能。7. 测试驱动（Test-driven） XP：强调”测试先行”。在编码开始之前，首先将测试写好，而后再进行编码，直至所有的测试都得以通过。 评述：RUP 与 XP 对测试都是非常的重视，只是两者对于测试在整个项目开发周期内首先出现的位置处理不同。XP 是一项测试驱动的软件开发过程，它认为测试先行使得开发人员对自己的代码有足够的信心，同

时也有勇气进行代码重构。测试应该实现一定的自动化，同时能够清晰的给出测试成功或者失败的结果。在这方面，xUnit 测试框架做了很多的工作，因此很多实施 XP 的团队，都采用它们进行测试工作。　　项目：我们在项目初期就对 JUNIT 进行了一定的研究工作，在项目编码中，采用 JBUILDER6 提供的测试框架进行测试类的编写。但是，不是对所有的方法与用例都编写，而只是针对关键方法类、重要业务逻辑处理类等进行。详细的关于 JUNIT 测试框架的使用，请参见我的同事撰写的另一篇文章 (http://www-900.ibm.com/developerWorks/cn/java/l-junit/index.shtml) 8. 代码重构（Refactoring）　　XP：强调代码重构在其中的作用，认为开发人员应该经常进行重构，通常有两个关键点应该进行重构：对于一个功能实现和实现后。　　评述：代码重构是指在不改变系统行为的前提下，重新调整、优化系统的内部结构以减少复杂性、消除冗余、增加灵活性和提高性能。重构不是 XP 所特有的行为，在任何的开发过程中都可能并且应该发生。在使用代码重构的时候要注意，不要过分的依赖重构，甚至轻视设计，否则，对于大中型的系统而言，将设计推迟或者干脆不作设计，会造成一场灾难。　　项目：我们在项目中将 JREFACTORY 工具部署到 JBuilder 中进行代码的重构，重构的时间是在各个迭代周期的前后。代码重构在项目中的作用是改善既有设计，而不是代替设计。9. 成对编程（Pair Programming）　　XP：认为在项目中采用成对编程比独自编程更加有效。成对编程是由两个开发人员在同一台电脑上共同编写解决同一问题的代码，通常一个人负责写编码，而另一个负责保证代码的正确性与可读性。
评述：其实，成对编程是一种非正式的同级评审（Peer Review）。它要求成对编程的两个开发人员在性格和技能上应该相互匹配，目前在国内还不是十分适合推广。成对编程只是加强开发人员沟通与评审的一种方式，而非唯一的方式。具体的方式可以结合项目的情况进行。　　项目：我们在项目中并没有采用成对编程的实践，而是在项目实施的各个阶段，加强了走查以及同级评审的力度。需求获取、设计与分析都有多人参与，在成果提交后，交叉进行走查；而在编码阶段，开发人员之间也要在每个迭代周期后进行同时评审。10. 集体代码所有制（Collective Ownership）

XP：认为开发小组的每个成员都有更改代码的权利，所有的人对于全部代码负责。
评论：代码全体拥有并不意味者开发人员可以互相推委责任，而是强调所有的人都要负责。如果一个开发人员的代码有错误，另外一个开发人员也可以进行 BUG 的修复。在目前，国内的软件开发组织，可以在一定程度上实施该实践，但是同时需要注意一定要有严格的代码控制管理。

项目：我们在项目开发初期，首先向开发团队进行”代码全体拥有”的教育，同时要求开发人员不仅要了解系统的架构、自己的代码，同时也要了解其它开发人员的工作以及代码情况。这个实践与同级评审有一定的互补作用，从而保证人员的变动不会对项目的进度造成很大的影响。

在项目执行中，有一个开发人员由于参加培训，缺席项目执行一周，由于实行了”代码全体拥有”的实践，其它的开发人员成功地分担了该成员的测试与开发任务，从而保证项目的如期交付。11. 持续集成（Continuous Integration）　　XP：提倡在一天中集成系统多次，而且随着需求的改变，要不断的进行回归测试。因为，这样可以使得团队保持一个较高的开发速度，同时避免了一次系统集成的恶梦。　　评述：持续集成也不是 XP 专有的最佳实践，著名的微软公司就有每日集成（Daily Build）的成功实践。但是，要注意的是，持续集成也需要良好的软件配置变更

管理系统的有效支持。 项目：使用 VSS 作为软件配置管理系统，坚持每天进行一次的系统集成，将已经完成的功能有效地结合起来，进行测试。12. 小型发布（Small Release） XP：强调在非常短的周期内以递增的方式发布新版本，从而可以很容易地估计每个迭代周期的进度，便于控制工作量和风险；同时，也可以及时处理用户的反馈。 评论：小型发布突出体现了敏捷方法的优点。RUP 强调迭代式的开发，对于系统的发布并没有作出过多的规定。用户在提交需求后，只有在部署时才能看到真正的系统，这样就不利于迅速获得用户的反馈。如果能够保证测试先行、代码重构、持续集成等最佳实践，实现小型发布也不是一件困难的事情，在有条件的组织可以考虑使用。 项目：项目在筹备阶段就配置了一台测试与发布服务器，在项目实施过程中，平均每两周（一个迭代周期结束后）进行一个小型发布；用户在发布后两个工作日内，向项目小组提交”用户接收测试报告”，由项目经理评估测试报告，将有效的 BUG 提交至 Rational Clear Case，并分配给相应的开发人员。项目小组应该在下一个迭代周期结束前修复所有用户提交的问题。

### 3.1.3 E-R 图

父图与子图平衡：父某个加工的输入输出流，与子的，在数量和名称上相同；父的一个流对于子图一个或多个流，这些组合起来正好是父的。

# Chapter 4　系统软件

## 4.1　操作系统

### 4.1.1　进程管理

前驱图

进程的同步与异步；互斥与共享。单缓冲区，多缓冲区，同步与互斥问题。

#### PV 操作

临界资源：进程间需要互斥方式进行共享的资源。临界区：访问临界资源的那段代码。信号量：一种变量，如 P(s) 中的 s P 操作，S-=1；S<0, 阻塞；V 操作，S+=1;s<=0, 唤醒。

不死锁：资源数 = 并发进程数 *（进程资源数-1）+1

死锁的条件：互斥保持、等待不剥夺环路等待

死锁避免：有序资源分配；银行家算法：银行放贷的思想，评估无能力偿还就不分配。需求量不超过资源数时接纳该进程；进程可分期请求资源；资源不满足时，推迟分配。

### 4.1.2　存储管理

CPU 中的寄存器，cache（按内容存取），内存（主存），外存，越来容量越大，速度越慢。

#### Cache

提高性能依据的是程序的局部性原理。读取时系统的平均周期 = 命中率 *cache 周期时间 +（1-命中率）* 主存周期时间

直接映像，全相联映像，组相联映像

#### 内存

随机存取存储器 RAM，只读存储器 ROM

AC000H 到 C7FFFH 共多少 K 个地址单元?$C7FFFH+1H-AC000H)/2^{10} = (1C000H)/4/16/16 = 28*4 = 112$; 内存地址按字（16bit）编址，28 片存储芯片组成，每片 16k 个存储单元，则每个存储单元存多少位？112K * 16bit = 28*16 * x ; x = 4bit

磁盘存取时间 = 寻道时间 + 等待时间（平均定位时间 + 转动延迟时间）

总线内部系统：数据，地址，控制外部

系统可靠性：串联：$\lambda = \sum \lambda_i$ 并联：$\mu = \frac{\lambda}{\sum_{j=1}^{n} \frac{1}{j}}$ 模冗余系统，m 个子系统同时计算，结果利用表决器屏蔽错误。

申请内存时的分配方法：首次适应法，选择地址最小的足够大的块；最佳适应法，选择块最小的满足的块。（缺点是导致碎片小）最差适应法，选择块最大的满足的块，增大碎片。循环首次适应，内存块首地址每次查找时移动一个。

#### 页式存储

页表：用户程序的所在页号，与内存中的块号（物理地址）之间关系表。逻辑地址前 k 位为页号，根据页表查找块号，块号和逻辑地址剩余的位组成物理地址。

#### 段式存储

有效地址 = 段号 + 位移量。根据段号查找段长和基址，基址 + 位移量得到物理地址。

各段长度可变，内存碎片浪费大，多程序共享内存。

#### 页面置换算法

最优（OPT）算法随机算法

先进先出 FIFO 算法，可能产生抖动，抖动意思是增加资源反而效率更低。

最近最少使用 LRU 算法，不会抖动

### 4.1.3 文件管理

#### 索引文件结构

索引节点，直接盘块与一级、二级、三级盘块地址复合组成。

#### 文件和树形目录结构

绝对路径，从根开始的路径，盘符开始；相对路径。例如固定电话的区号。

#### 空闲存储空间管理

空闲区表法（空闲文件目录）空闲链表法位示图法【重点】成组链接法

### 4.1.4 设备管理

虚拟设备与 Spooling 技术：例如 4 个人要打印，打印设备先将打印内容存放到缓冲区，顺次打印缓冲区的内容。

#### IO 控制方式

程序控制：需要主动问程序是否完成。

程序中断：程序运行完发送中断让系统处理。

DMA，DMA 不需要 CPU 执行程序指令。专门的控制器。主存与外设直接成块传送。

无条件传送、程序查询、

### 4.1.5 微内核操作系统

只实现客户进程、进程服务器、终端服务器、文件服务器等基本功能；图形系统、文件系统、设备驱动、通讯功能放在内核之外。

分为用户态、核心态，核心态响应客户进程的请求，回答文件服务器。

嵌入式操作系统的特点：（1）微型化，从性能和成本角度考虑，希望占用的资源和系统代码量少；（2）可定制，从减少成本和缩短研发周期考虑，要求嵌入式操作系统能运行在不同的微处理器平台上，能针对硬件变化进行结构与功能上的配置，以满足不同应用的需求；（3）实时性，嵌入式操作系统主要应用于过程控制、数据采集、传输通信、多媒体信息及关键要害领域需要迅速响应的场合，所以对实时性要求较高；（4）可靠性，系统构件、模块和体系结构必须达到应有的可靠性，对关键要害应用还要提供容错和防故障措施；（5）易移植性，为了提高系统的易移植性，通常采用硬件抽象层和板级支撑包的底层设计技术。

## 4.2 编辑系统

## 4.3 实时系统

# Chapter 5　应用软件

## 5.1　并行与分布式软件

ParallelAndDistribution

### 5.1.1　并行编译技术

### 5.1.2　并行调试技术

### 5.1.3　并行与分布式软件、工具与环境

### 5.1.4　可视化并行程序设计环境

# Chapter 6 Else

## 6.1 标准化和知识产权

商标权是可以续住来长期拥有的知识产权。