**《个体软件过程》读后感**

由于目前在测试岗位上，选取了本书的第12章-19章来作为切入点。以下为书目内容摘记。

* 质量的定义

引用《质量·软件·管理》一书中的定义，【所谓质量就是符合“需求”】本书的假定在于需求已经明晰的前提下，只考虑编码部分的缺陷的应对处理和分析预防方法。

* 缺陷与BUG的异同

本书的观点，BUG的范围大于缺陷，BUG是一般的为符合需求之处，而缺陷是潜在的风险点，可能在之后的使用中产生巨大负面影响的BUG。

* 缺陷类型

由于已经排除了需求不明导致的缺陷，本书将缺陷分为10个大类，读者可以根据具体情况设置细分类。

* 记录缺陷

本书主要推荐了“日期”，“编号”，“类型”，“引入阶段”，“排除阶段”，“修改时间”，“修复缺陷”7个项目。其中修复缺陷是指修复另一编号缺陷时产生了此缺陷。

* 缺陷查找技术

编译、代码自查。

* 发现和修复缺陷的开销

开发流程越往后，开销越大。

* 代码复查

IN：需求文档，设计文档，程序清单，编码标准

处理：复查规程、修复缺陷、覆盖率复查、程序逻辑复查、命名和和类型复查、变量检查、语法检查

OUT：修改文件清单，时间日志，缺陷记录

* 其他缺陷排除手段

同行评审

* 代码复查表

各种语言各不相同，主要涵盖为：完整性、引用、初始化、输入输出格式、命名、逻辑控制、指针、文件读取和关闭等

* 缺陷预测

D\_plan为预估缺陷数，D\_n为此前开发的第n个模块检出的缺陷数，N\_n为此前开发的第n个模块的代码千行数。

* 设计缺陷

起因多为对需求的不理解或对环境不熟悉导致的，也有一些是设计时欠深思熟虑。用标准化方式例如数据流程图，伪代码等方式来表达设计有助于提前排除设计缺陷。

* 产品质量

本书的核心就是开发人员要重视自查，并应该仔细检查。并建议开发前先建立软件原型并测试以发现设计缺陷。

* 过程质量

本书将过程中的缺陷分为两类，仅影响一个模块的缺陷，和影响多个模块的缺陷。本书认为提高质量应遵循

1. 尽量提高自己代码的质量
2. 完整检查各个模块接口的交互
3. 检查需求，确保背完全的正确理解、设计和实现
4. 代码走查之后也全面覆盖的单元测试
5. 全面覆盖的集成测试
6. 全面覆盖的系统测试

PSP讨论的内容仅包含第一条，但这条是基础，是非常重要的前提。本书强调A/FR(质检过失比)的作用。

* PSP过程脚本

IN：空白PSP项目计划总结表、代码复查表、过去开发程序规模和效率统计、时间日志、缺陷日志

处理：计划、设计、编码、代码复查、编译、测试、后置处理

OUT：设计文档、源程序清单、完成的项目计划总结表、本次时间日志、缺陷日志

* PSP项目计划总结表

表头、Minute/LOC、LOC/HOUR、Defects/LOC、过程效益、A/FR、LOC、开发阶段时间计划、实际、累计、累计百分比、缺陷估计、缺陷实际、缺陷累计、缺陷累计百分比、排除缺陷估计、排除缺陷实际、排除缺陷累计，排除缺陷百分比

**总结与发散**

本书主要专注于编码开发过程中，个人的管理方法。将其核心思想萃取出来可以归纳为，检查->记录->分析->为之后项目作为预测的依据。书中介绍了一些对缺陷引入的趋势的分析计算方法，例如缺陷预测的算法等。但是作为对比目前的开发技术书中的思想还有一些出入，比如本书认为A/FR的算法中使用检查的时间除以编译的时间，事实上得益于现在SDK和IDE以及硬件技术的发展，编译程序的时间通常已经被程序员所忽略，因而这些值已经很难计算。此外，在当前程序规模庞大，项目任务繁重的环境之下，认真记录下这些分析所需的数据也是一件很有挑战的工作。

本书编写于上世纪90年代，虽然当时的技术原因，有部分指标可能很难在当今实施，但是本书提供的思想对于各个团队中开发者还是非常有指导意义的。例如代码走查和同行评审（有些团队是交叉审阅）目前还是被大多数软件开发团队所使用的。而且有一些现象作者也做了比较客观的阐述，比如为了提高A/FR而单纯的投入检查代码的时间是没有意义的，应为到了一定的程度，即使投入更多的时间来检查也已经很难发现缺陷了。项目管理的本来目标之一也是控制成本停留在合理范围之内。

综上，本书虽然有一定程度的落后于时代，但是其指导思想还是有非同一般的价值的。此次由于时间紧张，学生只读了12到19章，接下去会逐渐将其他章节看完，并继续阅读配套的《团队开发过程》，理解其思想和方法，并努力应用于我的工作之中去。