> 计算机软件工程课程设计第五次实验报告
>
> **实验五 工作量估算、风险管理、需求获取**
>
> 学号：201705130112
>
> 姓名：张烜

1. 实验目的

1. 练习工作量估算。
2. 练习细化项目风险管理。
3. 需求引发：持续项目沟通，响应变化。
4. 完善之前的项目文档，跟踪项目进展

2. 实验内容

2. 1 练习工作量估算：

2.1.1 ch3 习题6（P98）

> 一个大型的政府机构希望与一家软件开发公司就一个顶目签订一份合同，该项目包含20000行代码。Hardand软件公司使用 Walston和 Felix的估算技术来确定编写这么多代码所需的时间以及这么长的时间里所需的人员数。 Hardand公司进行的估算将需要多少人月？如果政府的项日规模估算低了10%（即20000行代码只表示了实际规模的90%），还将要增加多少额外的人月？总的来讲，如果政府的规模估算低了%，则人员估算必须改变多少?

IBM模型是1977年IBM公司的Walston和Felix提出的。其中估算工作量的公式如下：E=5.2×L^0.91，L是源代码行数（以千行计），E是工作量（以人月计）

所以可以直接计算得出原本以及变动后的工作量为

$$E_t = 5.25 * S_t^{0.91} = 5.25 * 20000^{0.91} = 43062$$

$$E_t^{'} = 5.25 * (20000 * \frac{1}{0.9})^{0.91} = 49394$$

所以会额外增加4332人月的工作量

把E_t 与 $E_t^{'}$ 相除得到一个只与k相关的比例关系

$$E_t / E_t^{'} = (1/(1-k))^{0.91}$$

所以可以得到对应的人员变动比例为$(1/(1-k))^{0.91} - 1$

2.1.2 习题12（小组讨论）

很多项目经理根据过去项目中程序员的生产率来计划项目的进度，生产率通常根据单位时间的单位规
模来测量。例如，一个组织机构可能每天生产300行代码或每月生产1200个应用点。用这种方法测量生产率合适吗？根据下列事项讨论生产率的测度：

用不同的语言实现同样的设计，可能产生的代码行数不同

在实现开始之前不能用基于代码行的生产率进行测量。

程序员可能为了达到生产率的目标而堆积代码。

2.1.3 估算自己初始工作量

> 参考书3.7（P94)皮卡地里电视广告销售系统按COCOMOII的工作量模型应用例子，估算自己项目的初始工作量。

2.2 风险管理

2.3.1 分析风险

ch3 习题11分析自己项目中可能存在的风险。并进一步细化风险管理（做出风险分级及应对预案）。

在软件开发过程中可能会遇到很多风险，于是进行了简单的调研

> 1. 功能无限蔓延；
> 2. 需求镀金或者开发人员镀金；
> 3. 质量不定；
> 4. 计划过于乐观；
> 5. 设计欠佳；
> 6. 银弹综合症；
> 7. 研发导向的开发；
> 8. 人员薄弱；
> 9. 签约商失败；
> 10. 研发人员和客户的摩擦；

那么应该如何防范风险呢?

防范风险前，先了解工程师对公司的期许理想的情况下，我们都不希望让一个优秀的程序员离开团队，希望程序员能与公司一同成长、长久共事。所以我们可以先了解对于大部分程序员来说，对公司的期许是什么。

问题一：

> 开发团队因故无法完成或交付任务。

这种状况其实还可以细分为完成一半还是全没完成，以及程序员还有办法联络到还是无法联络到。因此在做管理的当下，一定要记得掌握一些基本的原则。

预防方法:

- 代码一定要用Git 管理，并且定期请工程师Commit 代码，而且一定要写Commit Log。如果是做到一半的开发，至少会大概知道程序员写到哪。

- 数据库定期备份，虽然程序员有时候会做自动备份，但是有时候翻脸不认人的时候，还是有数据存在自己电脑最实在。
- 所有的服务器帐号密码一定要有列表，如果交接后，请全数变更。这样比较不怕程序员消失，就无法进入服务器进行管理与备份。
- 最关键也最重要的就是要有技术文件，但是很多人其实并不了解要做哪些文件才算齐全，不过至少有张图让你了解你们用了几台Server，大概系统的架构长怎样，API规划的文件是怎样，这些基本的理解，最好还是要有一些文件去做纪录和呈述。

问题二

> 数据库发生问题或是不翼而飞：前阵子发生的血淋淋案例就是Gitlab 的工程师不小心删除服务器的数据库，这些状况都让不少代码与数据付之一炬。

预防方法：

- 数据库备份其实也是一门学问，除了现在有很多云端服务会提供自动备份硬盘，建议还是可以定期一个月手动异地备份一次。
- 进一步请工程师使用Docker 进行管理，Docker 除了单纯的程式与资料备份外，能够更快地还原整体开发环境。
- 转移Schema 前一定要进行测试，很多数据的毁损与遗失，往往发生在schema 改变的当下，也因此，每次转移前的备份，决定是否要停机转移等等，都是需要谨慎思考的问题。

我认为作为一家创业公司的创始人，最好能够自己稍作了解，或是跟着走一趟，毕竟数据销毁的事，对很多IT 公司来说，应该就是命脉了。

问题三：

> 开发时间过久才发现大家想的不一样。

这问题有两种，一种是工程师的能力与原本评估有落差，另一点则是沟通不善。沟通不到位较为简单处理。但以评估落差来说，对于找外包或是再找其他工程师的方案，其实各有不同的恼人问题，对外包来说，麻烦的是如果头款已经支付，很难做到一半停下来，而换其他工程师的话，熟悉代码可能要好久，这都是普遍可以见到的状况。

预防方法：

- 如果是沟通不良的问题，团队可以用两到三周的时间作为一个循环，让工程团队定期做一个简单的Demo，每一次的工作都不宜开出一个太大的项目组，有别于以往长期项目组的思维，凡事要做到尽善尽美的思维一定要改掉。反过来说，每一个项目开发组慢慢建立，从主要功能到辅助功能分批完成，可以有阶段性的产出并且经历测试，这点非常重要。
- 针对选择招聘程序员或是外包团队，很多人会问说怎么可以确认工程师的水平？当然可以做reference check 或是code review，也许有些帮助。

2.3.2 Ch4 p1

开发者和需求提出者应该都负责任,开发人员要对需求审核,需求提出者也应该确定各种因素之后再提需求