WORK LOG 5

Shandong University

March 29, 2020

高德琛

Contents

1 Problem 3-6

设预估人月为 E_e，预估工作量为 S_e：

$$\begin{aligned} E_e &= 5.25 * S_e^{.91} \\ &= 5.25 * (20000)^{.91} \\ &= 43062 \end{aligned}$$

设估算值 S_e 以及真实值大小 S_t，有：

$$S_t = \frac{S_e}{1-k}$$

设原估值 E_e 与实际需要时间 E_t：

$$\begin{aligned} E_t &= 5.25 S_t^{.91} \\ &= 5.25(\frac{S_e}{1-k})^{.91} \\ &= (\frac{1}{1-k})^{.91} * 5.25 S_e^{.91} \\ &= (\frac{1}{1-k})^{.91} * E_e \end{aligned}$$

当 $k = 0.1$，有 $E_t \approx 47395$，则增加了 $47395 - 43062 \approx 4333$ 人月。

2 Problem 3-7

首先我们定义工具使用分级以及经验级别。

工具使用分级	定义
0	编辑、编码、调试
1	简短的前端、后端、CASE，较少的集成
2	基本的生命周期工具，适中的集成
3	强有力、成熟的生命周期工具，适中的集成
4	强有力、成熟的、主动的生命周期工具，很好的继承了过程、方法、复用

表 **1:** 工具使用分级

经验分级	经验时长范围
0	0-3 月
1	3-5 月
2	5-9 月
3	9-12 月
4	12-24 月

表 **2:** 开发经验分级

开发经验等级	CASE 工具等级	生产率因子
0	0	4
1	1	7
2	2	13
3	3	25
4	4	50

表 **3:** 生产率估算

名称	需求项	复杂性	权值
需求和产品设计	文档	适中	5
产品原型设计	原型图	简单	3
后端架构设计	文档、代码	适中	5
模型设计	文档	困难	8
单元测试	文档	适中	6
集成测试	文档	适中	6

表 **4:** 需求项评估

由表 4可得 $NOP = 33$，由表 3得产生率为 13，最终算得工作量约 2.54。

3 Problem 3-11

我们将风险分级为重大风险、较大风险、一般风险和低风险四类，从 1 开始标以递增等级。

Risk	Exposure (days)	Mitigating Actions	Level
深度学习模型训练滞后	(.40)(15) = 6	新增成员处理模型	2
服务器开销超过预算	(.50)(10) = 5	第一版减少部分功能	3
数据平台对接滞后	(.50)(4) = 2	专人负责数据平台对接	4
Scrum 引入失败	(0.60)(20) = 12	细分阶段引入	1
项目需求无法确认	(.20)(10) = 2	敏捷开发允许需求多次变更	4
项目状态及项目流程未及时确认	(.50)(8) = 4	管理人员与工具培训	3
深度学习模型的评估不满足需求	(.50)(20) = 10	第一版加入 BASELINE 方案	1
人员对接滞后	(.40)(10) = 4	分工需细化，工具需培训	2
前端渲染效率问题	(.70)(10) = 7	前端负责人需挑选培训	2
推荐系统数据集不足	(.10)(20) = 2	非深度学习方案	4

表 **5:** Risk Exposure & Mitigating Actions

4 Porblem 3-12

题中给出了三种该工作量评估方法的不适用场景：

- 不同语言代码量可能不同
- 编程开始之前无法进行评估
- 程序员为了增加工作量而堆积无效代码

尽管代码量和应用点是一种量化工作任务的手段，但是先然这三种场景的不足都是无法克服的。

不同语言代码量的确往往不同，例如汇编语言编写程序语句贴近底层，若实现同样的需求，其编码规模往往比高级语言大数倍。如果按照代码量衡量程序员工作量，那么汇编程序员工资也会高上数倍。

第二点体道编程工作开始前无法进行评估，因为代码才是最终呈现的成果，编程工作开始之前的文档、计划、会议、头脑风暴对于用户都是不可见的。这一部分的工作也是最难以衡量的，大概只能借由人的主观视角加以评估，很难以做到客观、无争议。

再者，考虑到软件开发的基础仍旧是“人”，那么软件开发当中一定会体现出“人”的缺点。比如说在代码量作为评判因素的情况下，程序员完全可以大量堆积无效代码。无效代码大量堆砌，会对后续的阅读、调试、维护带来极大的时间成本。

此外的量化手段，例如工作时间、编码时间同样也会产生不合理之处，例如在不加班的情况下，同一部门的工作者往往有着相同的工作时间，同时这种方案没有考虑到工作者的劳动效率不同。如果按照额外的加班时间进一步的评估，根据劳动法，本身就已经有关于加班费的支付手段；而且变相鼓励加班也是不可取的。

5 Problem 4-1

我认为作为产品的参与者，相关的开发者、需求方、用户都需要担责。尽管人身、财产损失在很多情况下都是不可预料、难以避免的。

例如说，黑客通过社交软件盗取了用户数据，其中可能涉及用户真实信息、财产信息，危害用户人身、财产安全。在这个例子当中，黑客作为犯罪活动的执行者，必然负有责任。而社交软件之所以会被黑客破解盗取数据，则是因为开发者未曾注意到软件层面的漏洞；若该软件试图继续运营，那么必然需要调查漏洞来源，并对开发层面的疏忽进行评估，同时对起他用户负责，通过更安全的系统使用户得以信任该系统。而需求方，作为软件验收者，由于种种原因并没有发现系统中潜在的安全问题。尽管寻找软件漏洞对需求方而言有些苛刻。但是需求方作为开发者与市场的中间人，负责着对质量的合格性查验，并将合格产品送入市场，故而需求方应当对市场、用户负责。