计算机学院 软件工程 课程实验报告

|  |  |  |
| --- | --- | --- |
| 实验题目：实验内容六 | | 学号：202100130028 |
| 日期：2024/4/8 | 班级： 计科2班 | 姓名：薛然然 |
| Email：1745047373@qq.com | | |
| 实验目的：  1. 工作量估算  2. 风险管理  3. 学习软件需求规格说明SRS文档的要求和结构 | | |
| 实验内容：  1.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。  （1） 使用不同语言进行设计，产生的代码量差异巨大。如果使用C语言或C++，甚至是Java代码量很大，但如果使用python语言或者Go，则代码量较少。因此不能简单地按照语言的行数判断生产率。  （2） 代码复用率高时，代码量也会相对减少，但是代码效率将会和可维护性将大大增加  （3） 在实现开始之前，虽然已经对工程量进行了估计，但项目的难度和具体的难点可能还未知。比如开发一个OS内核的速度显然要慢于开发一个简单的web项目，以程序员过去写代码的速度估计本项目的速度不一定合适。  （4） 在开始实现整体项目之前不可能有效估计代码行数，使用基于代码行数的生产率也就失去了意义。  （5） 当程序员没办法按时写出达到数量的代码时，可能为了完成任务而应付，导致很多无用或冗余的代码，这对系统的代码架构甚至性能来说是一个问题。  （6） 因此应当考虑兼顾生产目标、管理模式以及团队成员的业务熟练度，以及考虑在项目的不同阶段更换不同的估算侧重点。  自己项目的初始工作量估计如下：   |  |  |  |  | | --- | --- | --- | --- | | 名字 | 屏幕或报表 | 复杂性 | 权值 | | 调查与分析 | 报表 | 适中 | 5 | | 项目主题讨论及确定 | 报表 | 适中 | 5 | | 成员分工 | 报表 | 简单 | 2 | | 数据收集及项目制作 | 屏幕 | 难 | 3 |   NOPS=15  权值计算根据课本上表3-11：      小组开发人员具有低的经验和高的CASE成熟度，根据课本上表3-12，生产率估算值为（7+25）/2=16。    因此项目的估算工作量为15/16=0.94人月  2.即使你在做学生项目，在按时完成项目方面也有极大的风险。分析一个学生软件在开发项目并列出其中的风险。风险暴露是什么？你可以使用什么技术来减轻各种风险？  在我们的项目中，风险暴露指标为风险发生的概率和可能浪费的时间。  具体风险如下：技术兼容性风险，项目缺少可见性，新技术的引入，性能问题和可用性问题。   |  |  |  | | --- | --- | --- | | 风险类型 | 风险暴露（/天，概率\*浪费的时间） | 减轻风险的措施 | | 技术兼容性风险 | 0.10\*5=0.5天 | a.使用容器化技术以确保在不同环境中的应用运行一致性。  b.采用跨平台框架或技术以确保应用在不同设备和平台上的兼容性。 | | 项目缺少可见性 | 0.20\*20=4天 | a.使用持续集成/持续交付（CI/CD）工具，例如Jenkins、GitLab CI等，以自动化构建、测试和部署，并提供项目的实时状态和进度。  b.实施敏捷开发方法，如Scrum或Kanban，以确保团队成员之间的高度透明度和交流，并定期举行会议来展示进展。 | | 新技术的引入 | 0.10\*5=0.5天 | a.进行原型验证，使用快速迭代的方法验证新技术的可行性和适用性，以减少引入新技术所带来的风险。  b.建立良好的技术评估流程，并使用技术评审来确保新技术的选择和应用符合项目需求。 | | 性能问题 | 0.20\*20=4天 | a.使用性能监测工具以实时监控应用程序性能，并分析和识别潜在的瓶颈。  b.进行负载测试和压力测试，使用工具如Apache JMeter或Gatling，以评估系统在高负载下的性能表现，并优化系统架构和代码。 | | 可用性问题 | 0.25\*10=2.5天 | a. 到用户工作现场，了解目标用户使用软件的真实目的，从用户的角度、从用户的立场出发，了解如何通过软件系统替代用户的业务处理流程中，最繁琐、最容易出问题、或者是大量重复劳动的环节，让软件提高用户的工作效能和效率。  b. 通过对市场上同类竞争性产品进行分析，或者对这些产品进行实验性测试，了解这些产品的用户界面问题，从而对新系统的开发提供启发。 |   3. GB-T-9385-2008列出了编制SRS时宜考虑的事项及编制原则，即文档要求，SRS的引言部分即为SRS的结构。  文档11与其他文档的关系如下：  GB-T-9385-2008  （1）根据GB/T1.1的规定，原GB/T9385--1988版中第1章引言部分中的内容放在新版的引言部分；  （2）新版标准的范围部分重 新进行调整改写；  （3）第2章规范性引用文件删去了GB/T 8567；  （4）根据GB/T 8566和GB/T 11457的规定，术语“开发者”改为“供方”；  （5）原GB/T9385-1988版的第4章和第5章调整为新版的第4章，且名称为“SRS"的编制原则。调整后的第4章更加清晰、完善。而删去了旧版第5章中有关模型的内容；  （6）旧版标准的第6章的主要内容调整为新版标准的第5章，而提纲部分调整为新版标准的附录A，且附录A的内容扩充了一部分。  GB-T -8566-2007  （1）在结构上作了调整：将ISO/IEC 12207：1995/Amd.1的附录F调整为本标准的附录D，将Amd.1的附录G的内容调整到本标准的正文中，将Amd.1的附录H的H.1的内容调整到本标准的附录D，将H.2的内容调整到本标准的正文中；  （2）为更好地理解本标准，增加并修改了个别术语和定义；  （3）根据修订内容，对4.1.1.4.1.1.2.4.1.1.3.5.1.5.2.5.3.5.4、5.5、第6章以及图1.图C.1、图C.2的内容作了补充修改；  （4）第2 章规范性引用文件的内容作了修改调整：删去了已废止的ISO 8402（GB/T 6583），增加了GB/T 11457和ISO 13407，根据GB/T 19001的修订情况，其名称改为“质量管理体系要求”；  （5）删去了附录E.  本标准是GB/T 8566的第三次修订。本标准与GB/T 8566-2001的主要差别如下：  （1）支持过程中增加了 易用性过程；  （2）组织过程 中将培训过程改为人力资源过程，另外增加了3个过程，即资产管理过程、重用大纲管理过程和领域工程过程；  （3）增加了一个附录即附录D；  （4）增加了都分术语和定义，并对部分中文术语定名作了如下修改：  developer开发者→开发方；  maintainer维护者→维护方；  operator操作者→操作方；  migration移植→迁移：  security保密安全性→安全保密性；  change更改→变更；  trace跟踪、追溯→追踪；  release→发行发布；  test coverage测试覆盖→测试覆盖率；  qualification鉴定→合格性认定。 | | |
| 结论分析与体会：  1.代码行数作为生产率衡量标准不合适：  ——使用不同语言实现相同设计可能导致巨大的代码量差异，如C++和Python的比较。  不能在实施开始前有效使用基于代码行数的生产率测量。  程序员可能为达到生产率目标而堆积代码，影响系统的质量和可维护性。  2.项目风险分析：  ——技术兼容性风险、项目可见性不足、新技术引入风险、性能问题和可用性问题是主要风险。  ——采用容器化技术、跨平台框架、持续集成/持续交付工具、敏捷开发方法、技术原型验证、性能监测工具和用户场景分析等方式来减轻这些风险。  3.GB-T-9385-2008标准与SRS编制相关内容：  根据标准的结构和修订内容，介绍了SRS的编制原则及文档要求，包括引言部分的结构和与其他相关标准的关系。标准的修订包括术语的调整、引用文件的更新和结构的重新调整，以更好地支持软件需求规格的编制和理解。 | | |