软件工程概述作业

2154312 郑博远

1. 随着软件的普及，由于程序错误所带来的公众风险已经变得越来越重要。请查阅以下事件描述，并回答问题：

   （1） 导致事故发生的原因是什么？

   （2） 在软件开发过程中应该强调什么事项以便更好地防止类似问题的发生？

   美国航空公司飞机失事的事件描述：

   达拉斯 8 月 23 日电——航空公司今天声称，去年 12 月在哥伦比亚失事的美国航空公司喷气式飞机的机长输入了一条错误的单字母计算机指令，正是这条指令使飞机撞倒了山上。这次失 事致使机上 163 人中除 4 人生还外，其余全部丧生。

   美国调查人员总结说，显然这架波音 757 飞机的机长认为他已经输入了目的地 Cali 的坐标。但是，在大多数南美洲的航空图上，Cali 单字母编码与波哥大（Bogota）的编码相同，而波哥 大位于相反方向的 132 英里处。

   据美国航空公司的首席飞行员和飞行副总裁 Cecil Ewell 的一封信中说，波哥大的坐标引导飞机撞到了山上。Ewell 说，在大多数计算机数据库中，波哥大和 Cali 的编码是不同的。

   美国航空公司的发言人 John Hotard 确认，Ewell 的信首先是在《达拉斯早间新闻》中报道，本周交到了所有航空飞行员的手中以警告他们这种编码的问题。美国航空公司的发现也促使联 邦航空局向所有的航空公司发布公告，警告他们有些计算机的数据库与航空图存在不一致。

   计算机错误还不是引起这次失事原因的最终结论，哥伦比亚调查人员也在检查飞行员训练和航空交通管制的因素。

   Ewell 谈到，当他们把喷气式飞机的导航计算机与失事计算机的信息相比较时，美国航空公司的调查人员发现了计算机错误。数据表明，错误持续了 66 秒钟未被检测到，而同时机组人员匆忙遵守交通管制的指令采取更直接的途径到达 Cali 机场。3 分钟后，当飞机仍在下降而机组人员设法解决飞机为什么已经转向时，飞经坠毁了。

   Ewell 说这次失事告诉了飞行员两个重要的教训：“首先，不管你去过南美或任何其他地方多少次，比如落基山区，你绝对不能假设任何情况。其次，飞行员必须明白他们不能让自动驾驶设备承担飞行的责任。”

答：

（1）导致事故发生的原因是波哥大与目的地 Cali 的航空图上的单字母编码相同，但它们位于相反的方向。机长输入了错误的单字母计算机指令，导致飞机偏离航线最终撞山。

（2）首先，在重要的决策操作场景中，必须保证软件对用户输入进行错误检查。例如，确认驾驶目的地时，应该对指令进行验证以确保目的地正确性，避免因为输入错误而导致严重后果；

其次，应当设计直观易懂的用户界面，从而减少用户输入错误的可能性。例如，在确认飞行目的地时，选择合适的可视化展示与输入验证，以确保用户的正确输入；

再者，软件开发过程中应当进行单元测试、集成测试与系统测试等全面的软件测试。对于材料中飞机驾驶此类涉及安全的重要软件，在软件测试阶段更应当覆盖多样的场景，力求通过详尽的测试，发现并修复潜在的错误。

2. **为什么已经投入使用的软件会不断被修改？这些修改会带来什么副作用？有哪些软件工程措施可以防范修改带来的不利影响？**

答：

演化性是软件的本质特性，软件需随使用环境与需求的变化而被修改。首先，用户可能在使用中提出新需求，既有需求也可能发生新变化。其次，软件在使用过程中可能会出现各种问题和缺陷，需要进行修复。此外，操作系统升级、硬件设备更换会导致软件运行的环境发生变化，因此需要对软件进行修改以适应新的环境要求。最后，随着安全威胁的不断演变，软件需要不断进行安全更新与修复，以修复新的安全漏洞、应对各种攻击方式。

对软件进行修改可能会带来如下副作用：首先，修改可能会引入新的bug，导致软件稳定性与可靠性下降。其次，过多的修改可能会导致软件代码的复杂度增加，降低代码的可读性与可维护性。此外，频繁的修改可能会影响软件的稳定性，导致软件出现意外的崩溃故障。

下列软件工程措施能够防范修改带来的不利影响：首先，在进行软件设计时，就需要考虑到后续可能的修改需求，采用模块化和面向对象的设计原则，以提高软件的灵活性与可扩展性。其次，还需进行严格的软件测试，以确保修改后不会引入新的缺陷。此外，使用Git、SVN等版本控制系统对软件进行管理，记录每次修改的内容和原因，以便追溯与回滚修改。最后，还需要定期进行代码审查，从而发现并纠正潜在的问题，提高代码的质量与稳定性。

3. **很多开发人员认为技术质量就等同于产品质量，请举出一个具有很高技术质量的产品实例，而这个产品的客户并不认同这个产品。从这个实例中，你认为什么是高质量的软件产品？**

答：

以 Windows Vista 系统为例。在技术上，Windows Vista 较 Windows XP 进行了大量的升级，也引入了许多新的功能。例如，Vista 更换了系统内核，将内核由 NT 5.1 变为 NT 6.0，带来了全面增强的安全性与更为合理的系统机制； Vista 还优化了用户交互的图形界面，引入了 Aero UI；

此外，还有革新任务管理器、构建WDM音频系统等一系列技术上的改进。

尽管 Windows Vista 有较高的技术质量，但并未得到客户的广泛认同。由于“一刀切”地更改为Windows NT 6.0 内核，许多驱动和软件没有 Vista 版本，给用户带来了糟糕的兼容性体验。此外，许多用户对 Windows Vista 的性能和稳定性表示不满，抱怨它的系统资源占用过高、应用程序的运行速度慢。不仅如此，Windows Vista 的硬件要求较高，在其诞生的 2006 年左右，主流的 PC 配置难以通畅地运行 Vista，很大程度上影响了用户体验。

从上述例子不难得出，即使一个产品在技术上表现出色，但若无法满足用户需求、提供良好体验，那么仍然难以得到客户认同。因此，高质量的软件产品不仅需要具备先进的技术功能，更需要考虑用户在实际使用过程中的需求与体验。

4. **某学生使用 JAVA 语言开发了一个简单的计算器程序，可以实现加、减、乘、除四种操作，具体代码如下见附件。请阅读和分析该程序，从正确性、可靠性、可维护性、可复用性、可扩展性等方面对该程序的质量进行具体评价。**

```
package version1;

import java.io.BufferedReader;
import java.io.InputStreamReader;

/**
 *
 * 计算器程序控制台版本
 *
 */
public class Calculator {
    public static void main(String[] args)throws
        Exception{ String A;
        String B;
        String oper;
        String result="";

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.println("请输入数字A:");
        A=br.readLine();

        System.out.println("请输入运算符号(+、-、×、/)： ");
        oper=br.readLine();

        System.out.println("请输入数字B:");
        B=br.readLine();


        if(oper.equals("+")){ result=String.valueOf(Double.parseDouble(A)+Double.pars
            eDouble(B));
        }
```

```
        if(oper.equals("-")){
            result=String.valueOf(Double.parseDouble(A)-Double.parseDouble(B));
        }

        if(oper.equals("*")){ result=String.valueOf(Double.parseDouble(A)*Double.pars
            eDouble(B));
        }

        if(oper.equals("/")){ result=String.valueOf(Double.parseDouble(A)/Double.pars
            eDouble(B));
        }


        System.out.println("------------------------------------------------------ ");
        System.out.println("运算的结果是: "+A+oper+B+"="+result);
        System.out.println("------------------------------------------------------ ");
    }

}
```

答：正确性：程序实现了基本的加、减、乘、除四种运算功能，并且能够正确输出计算结果。

可靠性：没有对异常情况进行处理，程序在某些情况下可能会崩溃或给出错误结果。例如，没有确保输入的 A、B 为数字；在输入 0 作为除数时，可能导致程序错误等。

可维护性：该程序的代码结构相对简单，容易理解和修改。

可复用性：该程序的代码被放在一个单独的类中，可以独立地被其他程序引用和调用。但程序中的计算逻辑和输入输出逻辑没有进行有效分离，且没有提供接口或抽象层，所以在其他程序中复用该功能可能需要进行较大的修改。

可拓展性：如果需要添加新的运算功能（如平方、开根号等），需要在程序中添加新的条件分支。这种硬编码的方式使得程序的可扩展性较差，不利于后续的功能扩展和维护。