** 软件过程与管理实验 石家庄铁道大学信息学院**

**实验1：个人软件过程**

**本次实验内容是个人软件过程部分，通过本次实验，学生将掌握以下内容：**

1、掌握个人软件过程中的基本理念和各个台阶上对应的软件过程和活动。

2、能够使用个人软件过程的辅助工具PSP Studio和Process Dashboard来提高自己的软件开发的管理水平。

3、会用COCOMO II模型对软件工作量进行估计。

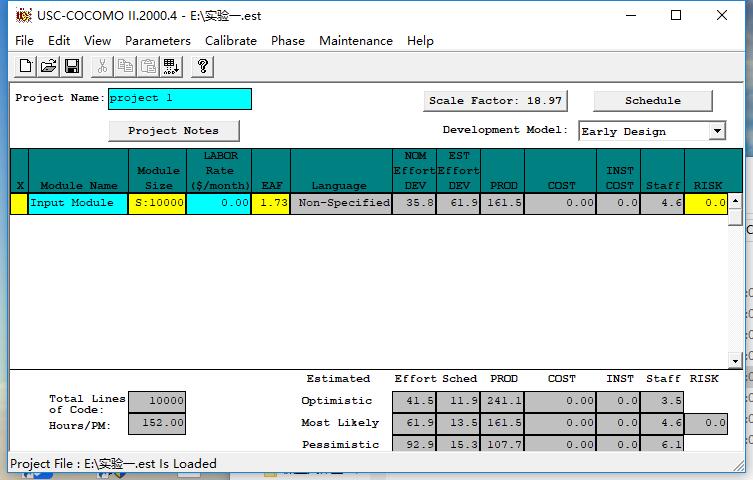
# [实验任务一]：COCOMO II 软件的使用

**实验要求：**

1. 通过自己看Manual，熟悉COCOMO II软件的使用；
2. 验证**软件工作量估计\_例子.ppt**中的例子，结果如下所示：

# **cocomo_test_revised**

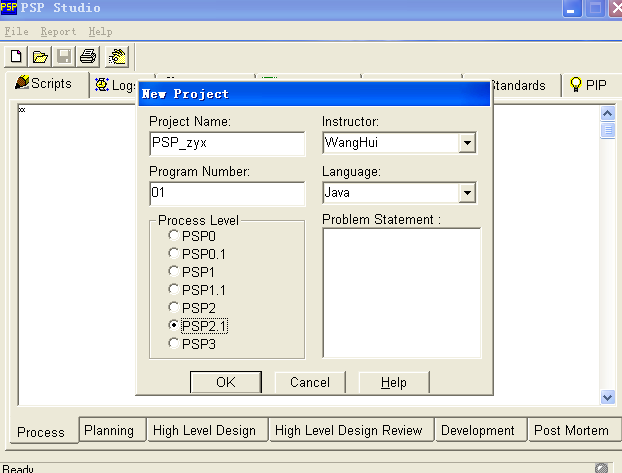
1. 提交扩展名为est的COCOMO项目文件，并将结果进行截图。



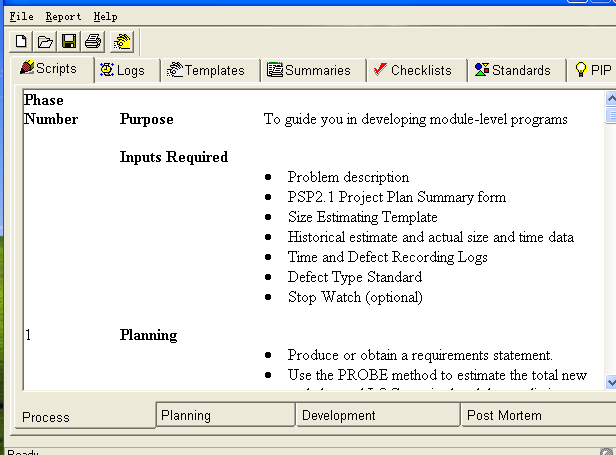
# [实验任务二]：基于PSP的个人软件过程

**实验要求：**

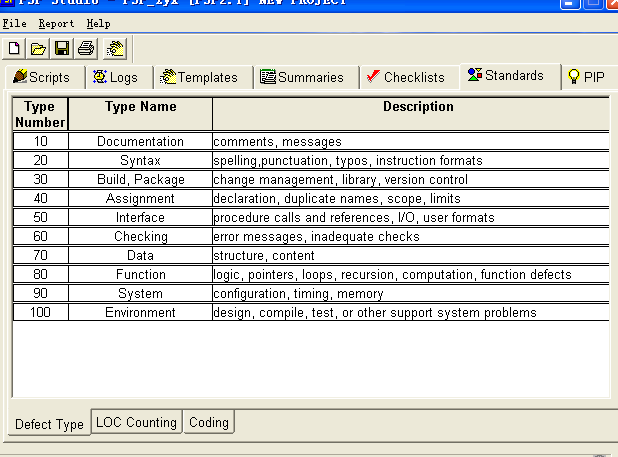
1. 通过阅读Manual，熟悉PSP Studio和Process Dashboard软件的使用；
2. 将PSP 2.1等级中的个人软件过程管理方法应用其它课程中的实训项目(或者其他实际项目)，完成该等级上对应的表、模板、日志等文档；
3. 利用PSP提供的4个模板完成系统的设计和其中一些模块的设计(登录模块除外)，也可以利用UML中与PSP中4个模板等价的图；
4. 提交对应的程序和使用说明文档。



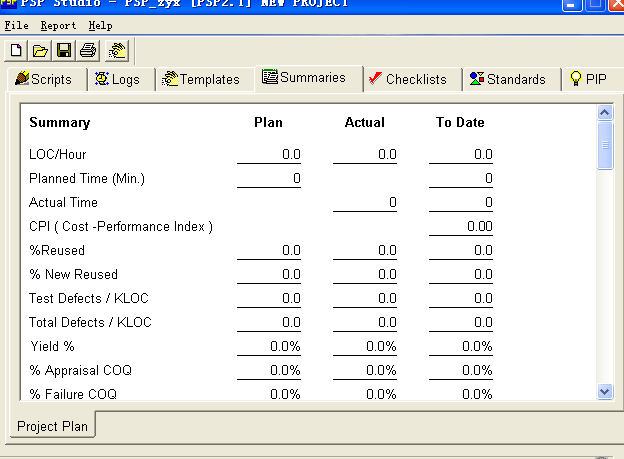
脚本表：



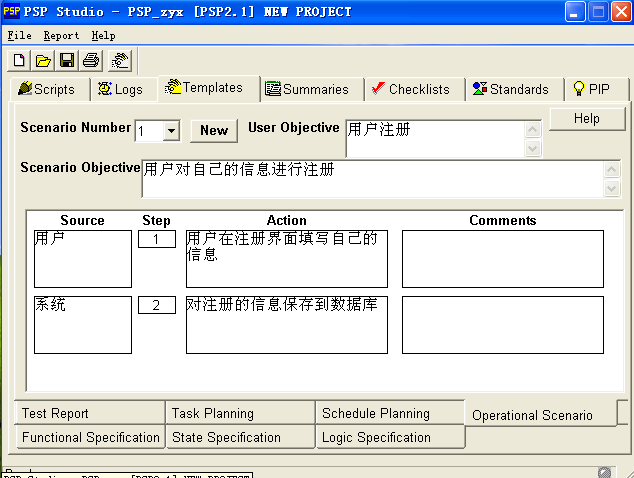
标准表：



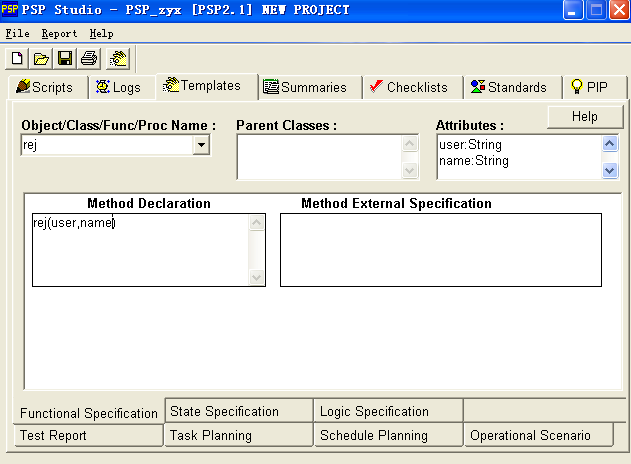
项目计划表：



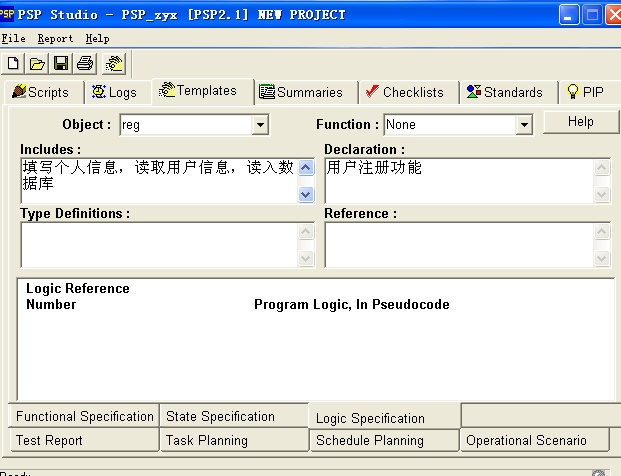
操作规格模板：



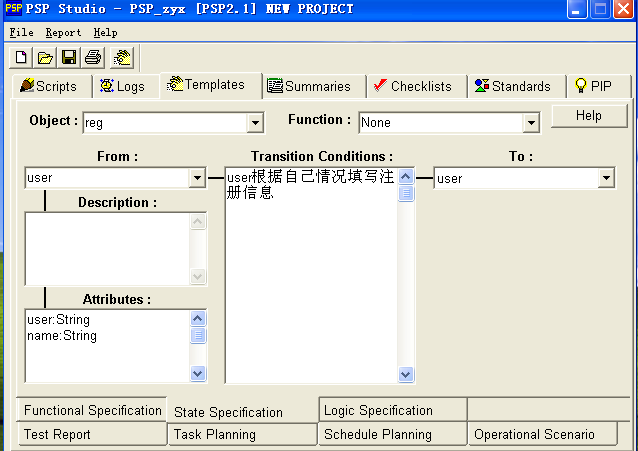
功能规格模板：



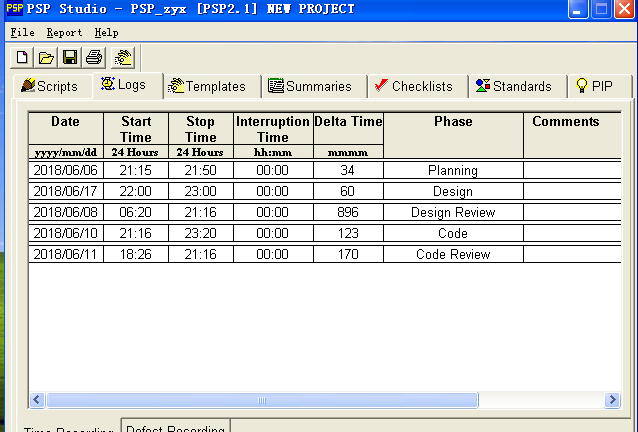
逻辑功能模板：



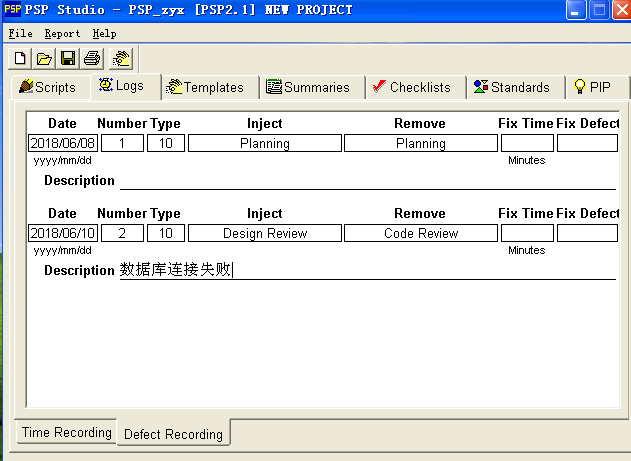
状态功能模板：



时间记录日志：



缺陷记录日志：



# [实验任务三]：代码规范和评审模板的建立

**实验要求：**

1. 根据自己的编程习惯和缺陷，建立自己的**代码规范**和**代码评审标准**，并且今后务必加以使用；
2. 需要提交**代码规范**和**代码评审标准**两个表格，使用Word即可。

**代码规范：**

|  |  |
| --- | --- |
| 目标 | Java程序的开发 |
| 程序头 | 所有程序都应从一个描述性的开头开始 |
| **注释规范** | 1、Java 文件：必须写明版权信息以及该文件的创建时间和作者；             2、类：类的目的、即类所完成的功能，以及该类创建的时间和作者名称；多人一次编辑或修改同一个类时，应在作者名称处出现多人的名称；             3、接口： 在满足类注释的基础之上，接口注释应该包含设置接口的目的、它应如何被使用以及如何不被使用。  在接口注释清楚的前提下对应的实现类可以不加注释；             4、方法注释： 对于设置 (Set 方法 ) 与获取 (Get 方法 ) 成员的方法，在成员变量已有说明的情况下， 可以不加注释；普通成员方法要求说明完成什么功能，参数含义是什么且返回值什么；另外方法的创建时间必须注释清楚，为将来的维护和阅读提供宝贵线索；             5、方法内部注释： 控制结构，代码做了些什么以及为什么这样做，处理顺序等，特别是复杂的逻辑处理部分，要尽可能的给出详细的注释；             6、参数： 参数含义、及其它任何约束或前提条件；             7、属性： 字段描述；             8、局部 ( 中间 ) 变量： 无特别意义的情况下不加注释； |
| 格式规约 | 左括号和后一个字符之间不出现空格；同样，右括号和前一个字符之间也不出现空格。  if/for/while/switch/do等保留字与左右括号之间都必须加空格。  任何运算符左右必须加一个空格。  单行字符数限制不超过120个，超出需要换行。  方法参数在定义和传入时，多个参数逗号后边必须加空格。 |
| 并发处理 | SimpleDateFormat 是线程不安全的类，一般不要定义为static变量，如果定义为 static，必须加锁，或者使用DateUtils工具类。  创建线程或线程池时请指定有意义的线程名称，方便出错时回溯。 |
| 控制语句 | 在if/else/for/while/do语句中必须使用大括号，即使只有一行代码  在一个switch块内，每个case要么通过break/return来终止  循环体中的语句要考量性能，在定义对象、变量、获取数据库连接尽量移至循环体外处理。 |
| **命名规范** | 1. 包（ Package ）的命名             Package 的名字应该采用完整的英文描述符，都是由一个小写单词组成。并且包名的前缀总是一个顶级域名，通常是 com、edu、gov、mil、net、org 等；         2、类（ Class ）的命名             类名应该是个一名词，采用大小写混合的方式，每个单词的首字母大写。尽量保证类名简洁而富于描述。使用完整单词，避免缩写词 ( 除非工程内有统一缩写规范或该缩写词被更广泛使用，像 URL ， HTML)         3、接口（ Interface ）的命名             基本与 Class 的命名规范类似。在满足 Classd 命名规则的基础之上，保证开头第一个字母为 ”I”，便于与普通的 Class区别开。其实现类名称取接口名的第二个字母到最后，且满足类名的命名规范；         4、枚举（ Enum ）的命名             基本与 Class 的命名规范类似。在满足 Classd 命名规则的基础之上，保证开头第一个字母为 ”E” ，便于与普通的 Class区别开。          5、异常（ Exception ）的命名            异常（ Exception ） 通常采用字母 e 表示异常，对于自定义的异常类，其后缀必须为 Exception         如： BusinessException         6、方法（ Method ）的命名             方法名是一个动词，采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。方法名尽可能的描述出该方法的动作行为。返回类型为 Boolean 值的方法一般由“ is ”或“ has ”来开头           7、参数（ Param ）的命名             第一个单词的首字母小写，其后单词的首字母大写。参数量名不允许以下划线或美元符号开头，虽然这在语法上是允许的。参数名应简短且富于描述。         如： public UserContext getLoginUser(String loginName);                 8、常量字段 （ Constants ）的命名             静态常量字段（ static final ） 全部采用大写字母，单词之间用下划线分隔；         如： public static final Long FEEDBACK;         public static Long USER\_STATUS; |
| 异常处理 | 不要捕获Java类库中定义的继承自RuntimeException的运行时异常类 |
| 空白区域 | 为了增加程序的可读性，写程序时应留出一定的空间。  每个程序结构之间至少有一行将他们隔开。 |
| 编码规约 | 不要通过一个类的对象来访问该类的静态变量或静态方法，直接通过类名访问即可。  Object的equals方法容易产生NPE问题，应该使用常量或者确定有值的对象来调用equals。  所有包装类对象之间值的比较全部使用equals方法。  不要尝试通过异常来进行流程控制，条件控制，因为异常的处理效率比条件分支低。  方法的参数不宜过多，当超过6个参数时，应将参数封装成一个类来使用。 |
| 缩进 | 缩进很重要！  不同级别括号之间有缩进  每一个起始括号和终止括号单独占一行  在遇到有关类、结构、函数或过程、以及枚举等等复杂程序结构的定义的时候，将它的内容缩进一层。 |
| 其他规范 | JSP文件命名：采用完整的英文描述说明 JSP 所完成的功能，尽可能包括一个生动的动词，第一个字母小写。如： viewMessage.jsp 、editUser.jsp 等。 |

**代码评审标准：**

|  |  |  |  |
| --- | --- | --- | --- |
| 评审对象 | |  | 作者 |
| 评审人 | |  | 评审工作量 |
| 程序结构组织 | | | |
| 1 | 代码结构清晰，外观整齐 | | |
| 2 | 所有的函数以及接口定义清晰 | | |
| 3 | 整个代码体系结构组合合理 | | |
| 4 | 模块中所有的数据结构都定义为局部的，并且通过定义好的函数进行访问 | | |
| 5 | 接口模块化，修改时不影响其他代码模块 | | |
| 6 | 提供了处理数据的策略 | | |
| 7 | 数据库操作、IO操作等是否正确关闭资源 | | |
| 8 | 多处使用相同代码时，应定义唯一方法或变量以供使用。 | | |
| 9 | 导入类时，如果仅使用包中的几个类，应导入具体类，而不是导入整个包。 | | |
| 10 | 检查代码和详细设计是否一致。 | | |
| 11 | 检查类中是否有无效的代码或者是无用的代码。 | | |
| 12 | 不要使用System.out.print()输出，需要进行日志处理 | | |
| 13 | 所有的文件名符合文件命名规范，见名知意 | | |
| 14 | 文件和模块分组清晰 | | |
| 15 | 所有的代码行在80字符以内 | | |
| 16 | 每个程序文件都小于2000行 | | |
| 代码组织 | | | |
| 1 | 数据库查询语句不要出现select \* | | |
| 2 | 对需要处理的字符串定义为StringBuffer ，常量定义成静态的。 | | |
| 3 | 所有的变量名都小于32字符 | | |
| 4 | 每行代码都小于80字符 | | |
| 5 | 类名、方法名、变量名的命名是否规范 | | |
| 6 | 有返回值的方法是否正确返回。Return语句应定义在方法结尾处。 | | |
| 7 | 代码排版是否规范 | | |
| 8 | 所有的行每行最多只有一句代码或一个表达式 | | |
| 9 | 复杂的表达式具备可读性 | | |
| 10 | 续行缩进 | | |
| 11 | 括号在合适的位置 | | |
| 12 | 每个顺序的小块用空行隔开 | | |
| 13 | 注释和代码对齐或接续在代码之后 | | |
| 14 | 代码的逻辑是否有问题。 | | |
| 15 | 和CSS除非必要就放到外部的单独文件中，在JSP中引用。 | | |
| 16 | 连接数据库的方法必须调用公用的方法。 | | |
| 17 | 异常要统一处理，异常处理方法是否符合项目组的约定 | | |
| 18 | 显示的文本无拼写和语法错误 | | |
| 19 | 代码与操作符的优先级无关 | | |
| 20 | 所有的表达式使用了正确的操作符 | | |
| 函数组织 | | | |
| 1 | 代码中有相关注解 | | |
| 2 | 函数的名字清晰的定义了它的目标以及函数所做的事情 | | |
| 3 | 函数中所有的部分都合理的组成函数，相关独立的语句组组成函数 | | |
| 4 | 参数遵循一个明显的顺序 | | |
| 5 | 所有的参数都被使用 | | |
| 6 | 函数的参数接口关系清晰 | | |
| 7 | 如果一个函数有返回值，在所有的出口都有返回值 | | |
| 8 | 函数的参数个数小于7个 | | |
| 9 | 使用的算法说明清楚 | | |
| 10 | 函数检查了输入数据的合法性 | | |
| 11 | 函数异常处理清楚 | | |
| 12 | 代码检查调用函数的返回值，参数和调用匹配 | | |
| 13 | 递归定义了出口 | | |
| 14 | 递归局限于一个函数 | | |
| 15 | 对接口的调用必须要有判断传入的参数不能为空和传入的参数正确。 | | |
| 数据类型与变量 | | | |
| 1 | 数据类型存在数据类型解释 | | |
| 2 | 代码避免了重新定义预先定义的数据类型 | | |
| 3 | 静态变量明确区分 | | |
| 4 | 每一个变量都初始化了 | | |
| 5 | 所有的变量都有最小的活动范围 | | |
| 6 | 所有的全局变量都描述清楚 | | |
| 7 | 所有的变量都用到了 | | |
| 条件判断 | | | |
| 1 | 条件检查和结果在代码中清晰 | | |
| 2 | If/else 使用正确 | | |
| 3 | 判断的次数降到最小 | | |
| 4 | 最常用的情况最先判断 ，所有的情况都考虑 | | |
| 5 | 判断条件使用引用时，应先判断其是否为null | | |
| 循环 | | | |
| 1 | 不要出现三层以上的for循环。如有必要必须写清楚注释 | | |
| 2 | 循环体不为空，循环体能够一次看清楚 | | |
| 3 | 注意while循环和for循环的使用情况 | | |
| 4 | 代码中不存在无穷次循环 | | |
| 5 | 循环体内的循环变量起到指示作用。 | | |
| 6 | 所有的循环边界是否正确 | | |
| 7 | 循环终止的条件清晰 | | |
| 8 | 循环条件、判断条件的边界值，数组、集合下标是否越界 | | |
| 注释 | | | |
| 1 | 类、方法、判断条件、循环、主要变量是否添加注释。 | | |
| 2 | 类名、方法名、变量名的命名是否规范。 | | |
| 3 | 源代码能够自我解释 ，每个人看到代码就能很快理解 | | |
| 4 | 注解清楚正确 ，注解为用户服务 | | |
| 5 | 类、方法、变量等注释是否符合规范，注释是否能够清晰明了 | | |
| 6 | 对if和for、while等要进行必要的注释，并且方法体都要用{ }包起来 | | |
| 7 | 代码的注释不能过于简单。要清晰明了，提交的时候必须写清楚修改的功能点和为什么要修改 | | |
| 8 | 注释是否较清晰且必要 | | |
| 9 | 函数是否已经有文档注释 |  |  |
| 总括 | | | |
| 1 | 代码直观 | | |
| 2 | 无用的代码、注解已经删除 | | |
| 3 | 除非必要，不要再循环体内定义对象 | | |
| 4 | 代码是否与详细设计一致 | | |
| 其他注意项 | | | |
| 1 | 尽量减少同名的类 | | |
| 2 | 不要在循环中反复定义创建变量 | | |
| 3 | 不要在循环中使用复杂的计算 | | |
| 4 | 在sql中注意敏感的单词要回避使用 | | |
| 5 | 代码与操作系统无关，不需要任何假设条件 | | |
| 特殊 | | | |
| 1 | 所有的数组访问在它们的边界内 | | |
| 2 | 类型转换明确指明 | | |
| 3 | 所有常量定义和使用替代代码中的数字 | | |
| 4 | 一个方法体的代码幅应该在一屏比较和合理；逻辑复杂的代码可以抽离出方法体 | | |