**SOFT ENGINEERING**

# l5 Project Mangement

目录

**[SOFT ENGINEERING 1](#_Toc28038)**

**[l5 Project Mangement 1](#_Toc15739)**

[1.About Measurement 2](#_Toc17790)

[Usual Metrics: Size and Complexity 2](#_Toc11203)

[2.Measurement under White Box: 2](#_Toc3129)

[Cyclomatic Complexity
2](#_Toc18035)

[3.Measurement under Black Box: 3](#_Toc1184)

[Storey Points (Size Estimation) 3](#_Toc17527)

[4.Software Laws: Patents, Copyright, Contract, Privacy 4](#_Toc4875)

[Review 6](#_Toc17785)

**[1. 我们如何衡量复杂性？ 6](#_Toc14944)**

**[2. 为什么我们使用黑盒选项？ 6](#_Toc17157)**

**[3. 什么是专利？ 6](#_Toc3897)**

**[4. 专利和版权之间有什么区别？ 6](#_Toc25507)**

**[5. 从社交网络中我们学到了什么有关合同？ 7](#_Toc14602)**

**[Code Inspection Checklist (Java) 7](#_Toc2006)**

**[1. 是否根据命名约定使用了描述性的变量和常量名称？ 7](#_Toc6557)**

**[1. 是否存在变量名称相似而令人困惑的情况？ 7](#_Toc26673)**

**[1. 每个变量是否都正确初始化了？ 7](#_Toc13673)**

**[1. 是否有任何非局部变量可以改为局部变量？ 7](#_Toc5333)**

**[1. 是否有应该命名为常量的文字常量？ 7](#_Toc2947)**

**[1. 是否有应该是常量的变量？ 8](#_Toc27157)**

**[1. 是否根据命名约定使用了描述性的方法名？ 8](#_Toc16957)**

**[1. 在使用每个方法参数之前是否进行了检查？ 8](#_Toc27417)**

**[1. 对于每个方法，在每个方法返回点是否返回了正确的值？ 8](#_Toc30137)**

**[2. 比较运算符（即 < <= > >= ==）是否正确。 8](#_Toc15027)**

[控制流（if、switch、for、while） 8](#_Toc8170)

**[1. 是否有 if 结构带有悬挂的 else 子句？ 9](#_Toc27588)**

**[1. 每个方法调用中参数的顺序是否与方法的声明一致？ 9](#_Toc3557)**

**[1. 单位中的值是否一致？ 9](#_Toc28133)**

[注释 9](#_Toc13422)

**[1. 注释是否有助于你理解代码？ 9](#_Toc16427)**

1.About Measurement  
2.Measurement under White Box:  
 Lines of code  
 Cyclomatic Complexity  
3.Measurement under Black Box:  
 Planning Poker  
4.Software Laws:  
 Patents, Copyright, Contract, Privacy

## 1.About Measurement

Usual Metrics: Size and Complexity

After development ...
How much effort will it require for maintenance?
Where should we direct testing effort?
How much effort was required for development?
Metrics are based upon source code (“white box”)
Before development has started ...
How much programming effort will module X require?
What will be the estimated cost of the final product?
Metrics are based upon requirements / specification (“black box”)

## 2.Measurement under White Box:

**Number of lines in a file (or a group of files)**

Easy to compute  
Easy to understand and interpret  
Often sufficient for an approximate measure of size  
Widely used (perhaps the most widely used) metric  
Comments  
What is a line?  
Blank lines  
Not all “lines” are equal  
Ignores logical/ architectural complexity  
Highly language-specific

**Cyclomatic Complexity**

Calculated from the control flow graph:   
V(G) = E – N + 2P  
E – number of edges;   
N – number of nodes;  
P – number of procedures (usually 1)  
Number of independent paths through the code  
Independent path – any path that introduces at least one new statement/condition

## 3.Measurement under Black Box:

**Storey Points (Size Estimation)**

An informal, agile unit of “size measurement”  
 Usually an estimate from 1-10  
Derive an estimate from the whole team at sprint planning meetings  
Based on the idea of the “Wisdom of the Crowds”  
 The collective estimate of groups (i.e., of effort required for a story) is better than the estimate of an individual

**Planning Poker**

**Team Velocity**

Number of (estimated) story points implemented per sprint.
Can be derived from previous sprints.
e.g., Average points implemented from previous x sprints. 
Can be used to estimate:
Time required to complete project.
Target number of stories that can be completed in a sprint.

## 4.Software Laws: Patents, Copyright, Contract, Privacy

**Patent Law**  
A government license giving a right for a set period, especially to exclude others from making, using, or selling an invention  
  
Granted by the government  
to stop others exploiting your invention  
Lasts 20 Year   
  
Inventions Must   
be new  
be an inventive step (not an obvious improvement)   
capable of industrial application

**Copyright**

Creator has exclusive rights to perform, copy, adapt their work.   
Everyone else must get Permission (and possibly pay)  
"literary, dramatic, musical and artistic works" includes software  
Automatically owned (not granted)   
Lasts 70 years after authors death (lots of exceptions)   
  
This affects software in 2 different ways:   
Illegal Copies of Applications (Piracy) !  
Using someone else's code/UI design/etc. in your application   
(Not the "idea" but the actual "stuff" (code, design, documents) created by someone else)

**Copyright Theft?**

NO:
Get permission (obtain a licence)
Be within "fair use" (e.g. for study or review)
Use "open source" software
Create something similar yourself, independently
"Obvious" code can't be copywrited

YES:  
Displaying an image from another page   
Using code found on the internet  
Copying Windows 95 for your friends

**Contract Law**  
Employer contracts usually force an employee to:
Not work for anyone else
Hand over any ideas (Intellectual Property)
Not disclose company secrets (Non-disclosure-agreements)
(even after you stop working for them)

**Data Protection**   
*8 Principles of Data Protection:*
Any company storing "personal data" must make sure it is:
fairly and lawfully processed (consent, contractual and legal obligations, public interest, ...)
processed for limited purposes;
adequate, relevant and not excessive;
accurate and, where necessary, kept up to date;
not kept longer than necessary;
processed in accordance with the data subject's rights;
secure;  
not transferred to countries without adequate protection

UK : Data Protection Act
EU : Data Protection Directive
US : a "patchwork" of state and national laws

## Review

How can we measure complexity?  
Why do we use black box options?  
What is a patent   
What is the difference between patent and copyright?  
What do we learn about contract from Social Network?

1. 我们如何衡量复杂性？

复杂性可以通过多种方式来衡量，包括代码行数、算法复杂度、问题的范围和变化等因素。

1. 为什么我们使用黑盒选项？

我们使用黑盒选项是因为它们提供了一种简单的方式来处理复杂性，而无需了解内部的具体实现细节。这对于简化问题、提高效率和保护知识产权都是有利的。

1. 什么是专利？

专利是政府授予的一种权利，赋予发明人在一定时间内独占性的权利，以防止他人制造、使用或销售该发明。

1. 专利和版权之间有什么区别？

专利主要用于保护发明或创新性的产品、方法或流程，而版权主要用于保护原创的文学、艺术和音乐作品。专利提供了对发明的独占权，而版权提供了对作品的独占权。

1. 从社交网络中我们学到了什么有关合同？

从社交网络中我们学到了合同的重要性和约束力。合同可以确保各方之间的权利和责任得到明确界定，从而有效地管理商业关系并防止纠纷的发生。

# Code Inspection Checklist (Java)

变量和常量声明

1. 是否根据命名约定使用了描述性的变量和常量名称？

描述性的名称使代码更易于阅读、理解和维护。

1. 是否存在变量名称相似而令人困惑的情况？

如果是这样的话，在打字时容易输错一个名称而实际指的是另一个。如果这些变量是相同类型的，编译器就不会检测到你的错误。

1. 每个变量是否都正确初始化了？

编译器要求所有变量都进行初始化，但它们是否初始化为正确的值呢？

1. 是否有任何非局部变量可以改为局部变量？

如果类范围的变量（即实例变量）只被单个方法使用，并且该方法不必在调用之间记住其值，那么该变量可以改为方法的局部变量。不必要的非局部变量会使类过于复杂，从而更加难以阅读、理解和维护。

1. 是否有应该命名为常量的文字常量？

将文字常量的出现替换为命名常量，例如，将0.175这样的文字常量替换为命名常量（例如，将其声明为final int VAT\_RATE = 0.175）。如果增值税率发生变化，你只需在程序中更改常量的值，而不是每个0.175的出现。使用命名常量，你可以确保在所有情况下都使用了正确的值。

1. 是否有应该是常量的变量？

如果一个类有一个值不打算更改的变量，你应该将其设为常量，以防止意外更改。然后你可以依赖编译器来检测到试图修改的行为。

方法

1. 是否根据命名约定使用了描述性的方法名？

描述性的名称使代码更易于阅读、理解和维护。

1. 在使用每个方法参数之前是否进行了检查？

在使用方法参数之前检查其值是一个良好的做法。例如，当设置一个名称时，检查是否为 null 字符串，如果参数为 null，则不更改对象的状态。

1. 对于每个方法，在每个方法返回点是否返回了正确的值？

运算符

1. 对于每个具有多个运算符的表达式，对于评估顺序和优先级的假设是否正确？使用括号消除歧义。
2. 比较运算符（即 < <= > >= ==）是否正确。
3. 每个布尔表达式是否正确？一般来说，在编写布尔表达式时，使用等号运算符（而不是赋值运算符）。

控制流（if、switch、for、while）

1. 对于每个循环，是否使用了最合适的结构？在不事先知道需要进行多少次迭代的情况下，请使用 while 循环；在其他情况下，请使用 for 循环。
2. 所有循环都会终止吗？
3. 对于循环中的多个退出点，每个退出点是否必要并且正确处理？
4. 每个 switch 语句是否都有一个默认情况？

通常情况下，请为每个 switch 语句提供一个默认标签。

1. 缺少 break 语句（在 switch 结构中）是否正确，并且标有注释？

通常情况下，请在 case 语句后跟上 break。

1. 是否有 if 结构带有悬挂的 else 子句？

避免使用悬挂的 else 语句。

接口

1. 每个方法调用中参数的顺序是否与方法的声明一致？

例如，如果一个方法有两个相同类型的参数，编译器无法检测到当你调用方法时实际参数是否按正确顺序给出。

1. 单位中的值是否一致？

回想一下在第一讲中提到的火星气候轨道飞行器系统。一个组件向另一个组件提供了公制值，后者将其解释为英制单位（没有进行任何转换）。

注释

1. 每个类和方法是否都有适当的头部注释？
2. 每个变量或常量是否都有注释？
3. 每个类和方法的基本行为是否用简明的英语表达出来？
4. 注释和代码是否一致？

一旦代码发生变化，描述代码以前如何工作的注释就变得毫无价值。

1. 注释是否有助于你理解代码？

注释应该为源代码增加价值。

输入/输出

1. 打印或显示的任何文本中是否存在拼写或语法错误？
2. 是否检查了错误的输入数据？

以下是文档1(EquivalencePartition.pdf)的关键内容：  
总结：  
等价划分是一种黑盒测试方法，用于高效覆盖所有可能的输出情况，提高软件质量和可靠性。在generategrading组件中，该方法通过识别输入输出的等价类别，制定测试用例来执行测试。以课程作业评分系统为例，根据考试和课程作业分数生成等级，划分为'A'到'D'，超出范围则生成错误消息'FM'。  
测试人员通过识别和制定等价划分，为系统测试提供了重要依据。他们考虑了有效和无效输入，以及对应的输出情况，从而确保组件在所有可能情况下都能正确运行。  
测试用例的生成是关键，通过一对一方法，测试人员详细展示了测试用例的输入、计算总分和期望输出。这些测试用例涵盖了有效输入、无效输入和有效输出，以验证评分系统的准确性和可靠性。  
尽管测试用例可能涉及不同的输入值组合和分数计算，但测试所有案例的完整性和正确性至关重要。即使某些案例在实际实现中可能无法执行，仍需考虑其完整性。  
总之，通过等价划分和制定详细的测试用例，测试人员能够更全面地测试系统，确保在各种输入情况下，系统都能稳定、可靠地运行。这为课程作业评分系统提供了有力的保障，提升了软件的质量和用户体验。