|  |
| --- |
| 北京航空航天大学 |
| 测试规格说明书 |
| **Spring Framwork** |
|  |
| |  |  | | --- | --- | | 王昕 SY1506415 | 王旭辰 SY1506413 | | 林森 SY1506416 | 李勃 SY1506402 | |
|  |

**版本变更历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 提交日期 | 编制人 | 说明 |
| 1.0 | 2016/5/11 | 全体成员 | 测试需求规格说明书 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1 范围 1](#_Toc450768245)

[1.1 标识 1](#_Toc450768246)

[1.2 编写目的 1](#_Toc450768247)

[1.3 测试职责 1](#_Toc450768248)

[1.3.1 职责 1](#_Toc450768249)

[1.3.2 职责划分 1](#_Toc450768250)

[1.4 文档概述 2](#_Toc450768251)

[2测试计划 2](#_Toc450768252)

[2.1测试目标 2](#_Toc450768253)

[2.2测试方法 2](#_Toc450768254)

[2.2.1单元测试 2](#_Toc450768255)

[2.2.2集成测试 3](#_Toc450768256)

[2.3测试工具 3](#_Toc450768257)

[2.3.1Juint插件 3](#_Toc450768258)

[3 测试规范及对照表 5](#_Toc450768259)

[3.1 编写测试计划 5](#_Toc450768260)

[3.2 设计测试用例 6](#_Toc450768261)

[3.3 测试用例与需求用例对照表 6](#_Toc450768262)

[4.容器的基本功能 6](#_Toc450768263)

[4.1容器的初始化 6](#_Toc450768264)

[4.2验证转换器 7](#_Toc450768265)

[4.3获取组件信息 9](#_Toc450768266)

[4.4验证依赖关系是否正确处理 10](#_Toc450768267)

[4.5依赖关系下组件信息获取验证 11](#_Toc450768268)

[5容器的扩展功能 12](#_Toc450768269)

[参考文献 12](#_Toc450768270)

# 1 范围

## 1.1 标识

Spring版本号： Spring-Framework 3.2

模 块 名 称： 用注解管理容器，注解定义Java组件类，根据注解类生成XML文件

测试规格版本：初稿

## 1.2 编写目的

本文档主要明确了本次测试的主要内容和测试小组各成员的分工，并对测试用例进行了初步的设计。测试技术和策略等问题不在本文档描述范围内。

## 1.3 测试职责

### 1.3.1 职责

测试是软件开发过程中的重要组成部分，肩负着如下责任：

* 编写合理的测试计划，并与需求分析相对应；
* 编写覆盖率高的测试用例；
* 针对测试需求进行相关测试技术的研究；
* 认真仔细地实施测试工作，并提交测试报告；
* 撰写工作进度日志。

### 1.3.2 职责划分

|  |  |
| --- | --- |
| 成员 | 主要职责 |
| 王昕 | 整合组员的测试成果，编写测试需求规格说明书 |
| 王旭辰 | 容器基本模块测试用例编写 |
| 林森 | 容器扩展模块测试用例的编写 |
| 李勃 | 容器的基本模块和扩展模块的集成测试 |

## 1.4 文档概述

本文档主要说明了测试阶段的工作内容。首先叙述了本组对测试工作的分析过程，给出了测试用例的需求用例与对应的测试用例的对照表，对每个测试用例按照RUCM4test的标准进行了说明。最后，本文对“非功能性需求”的“服务器性能测试”从效率的角度做了详细描述，给出了详细的测试过程。

# 2测试计划

## 2.1测试目标

通过测试来发现Spring框架所能提供给开发人员使用的功能是否符合《软件需求规格说明书》，并且在测试的过程中要尽可能的发现软件在现有阶段存在的问题，确保在软件交付客户使用之后出现的问题在可接受的范围之内。

## 2.2测试方法

### 2.2.1单元测试

单元元测试是针对软件设计的最小单位——程序模块，进行正确性检验的测试工作。其目的在于发现每个程序模块内部可能存在的差错、

　　单元测试也是程序员的一项基本职责，程序员必须对自己所编写的代码保持认真负责的态度，这是也程序员的基本职业素质之一。同时单元测试能力也是程序员的一项基本能力，能力的高低直接影响到程序员的工作效率与软件的质量。

在编码的过程中作单元测试，其花费是最小的，而回报却特别优厚的。在编码的过程中考虑测试问题，得到的将是更优质的代码，因为在这时您对代码应该做些什么 了解得最清楚。如果不这样做，而是一直等到某个模块崩溃了，到那时您可能已经忘记了代码是怎样工作的。即使是在强大的工作压力下，您也还必须重新把它弄清 楚，这又要花费许多时间。进一步说，这样做出的更正往往不会那么彻底，可能更脆弱，因为您唤回的理解可能不那么完全。

　　合格的代码应该具备以下性质：正确性、清晰性、规范性、一致性、高效性等(根据优先级别排序)。

　　1. 正确性是指代码逻辑必须正确，能够实现预期的功能。

　　2. 清晰性是指代码必须简明、易懂，注释准确没有歧义。

3. 规范性是指代码必须符合企业或部门所定义的共同规范包括命名规则，代码风格等等。

　　4. 一致性是指代码必须在命名上(如：相同功能的变量尽量采用相同的标示符)、风格上都保持统一。

　　5. 高效性是指代码不但要满足以上性质，而且需要尽可能降低代码的执行时间。

### 2.2.2集成测试

集成测试,又称为组装测试,就是将软件产品中各个模块组装起来,检查其接口是否存在问题,以及组装后的整体功能、性能表现。在开展集成测试之前,我们进行了深入的单元测试(当然,实际工作中大多公司不会做单元测试,仅有程序员各自检查自己的代码),从个体来讲,可能解决了很多的缺陷,但所有的个体组合起来,就可能出现各种各样的问题。1+1<2 的问题,此刻尤为突出。在单元测试阶段,我们无法发现资源争用、接口调用、时钟延迟等等问题。假如有两个模块A、B,一个数据在A 处处理时因代码问题,延迟了0.01 秒,然后流转到B 模块处理,又延迟了0.01 秒,那么对于个体来讲,可能0.01 秒算不了什么,但当数据流转的环节增加时,相应的延迟时间也在不断的增加,最终的累加数据可能带来非常严重的后果,二战期间的爱国者导弹时间,就给了我们深刻的教训。每个环节的缺陷在最终被放大后,可能会引起软件的失效。所以,单元测试阶段的成果并不能保证集成测试没有问题。采用科学有效的集成测试方法,在软件生产活动中是非常有必要的。

## 2.3测试工具

### 2.3.1Juint插件

JUnit是由 Erich Gamma 和 Kent Beck 编写的一个回归测试框架（regression testing framework）,供Java开发人员编写单元测试之用。

1、概述

　　Junit测试是程序员测试，即所谓白盒测试，因为程序员知道被测试的软件如何（How）完成功能和完成什么样（What）的功能。

　　Junit本质上是一套框架，即开发者制定了一套条条框框，遵循这此条条框框要求编写测试代码，如继承某个类，实现某个接口，就可以用Junit进行自动测试了。

　　由于Junit相对独立于所编写的代码，可以测试代码的编写可以先于实现代码的编写，XP 中推崇的 test first design的实现有了现成的手段：用Junit写测试代码，写实现代码，运行测试，测试失败，修改实现代码，再运行测试，直到测试成功。以后对代码的修改和优化，运行测试成功，则修改成功。

　　Java 下的 team 开发，采用 cvs(版本控制) + ant(项目管理) + junit(集成测试) 的模式时，通过对ant的配置，可以很简单地实现测试自动化。

对不同性质的被测对象，如Class，Jsp，Servlet，Ejb等，Junit有不同的使用技巧，以后慢慢地分别讲叙。以下以Class测试为例讲解，除非特殊说明。

junit中的assert方法全部放在Assert类中，总结一下junit类中assert方法的分类。

1.assertTrue/False([String message,]boolean condition);

用来查看变量是是否为false或true，如果assertFalse()查看的变量的值是false则测试成功，如果是true则失败，assertTrue()与之相反；

2.fail([String message,]);

直接用来抛出错误。

3.assertEquals([String message,]Object expected,Object actual);

判断是否相等，可以指定输出错误信息。

第一个参数是期望值，第二个参数是实际的值。

这个方法对各个变量有多种实现

4.assertNotNull/Null([String message,]Object obj);

判读一个对象是否非空(非空)。

5.assertSame/NotSame([String message,]Object expected,Object actual);

判断两个对象是否指向同一个对象。看内存地址。

7.failNotSame/failNotEquals(String message, Object expected, Object actual)

当不指向同一个内存地址或者不相等的时候，输出错误信息。

注意信息是必须的，而且这个输出是格式化过的。

# 3 测试规范及对照表

## 3.1 编写测试计划

在进行软件测试前，周密的计划和合理的安排是必不可少的。测试计划的制定是要建立在充分理解需求的基础上的，是测试的起始步骤和重要环

|  |  |
| --- | --- |
| 过程要点 | 详细说明 |
| 启动条件 | 需求分析文档终稿的完成； |
| 工作内容 | 根据项目的需求分析文档，测试文档中应该至少包括以下关键内容：   * 根据已有的需求分析文档终稿，明确相应的测试用例，包括四大部分的测试用例模块。 * 具体设计各大模块的测试用例，设计完成后进行评审工作。 * 根据评审后的测试用例编写测试代码 * 部署环境，具体测试各模块功能，记录测试数据及结果。 * 结合需求文档终稿，对测试结果进行分析。 * 完成整个测试需求文档，由评审组完成最终评审。   测试计划编写完毕后，必须提交给项目组全体成员，并由项目组组中各成员联合评审。 |
| 退出标准 | * 测试获得预期成果。 * 测试由评审组评审通过. |
| 责任人 | 全体成员 |

## 3.2 设计测试用例

一个测试用例描述了针对某个目标对程序进行测试所采用的一组实际输入、程序执行条件、测试步骤和预期的输出，以核实某个程序或其中的特定路径是否满足特定需求。在需求分析文档确立基线以后，需编写项目的测试用例，具体的任务和责任人如下：

|  |  |
| --- | --- |
| 过程要点 | 详细说明 |
| 输入条件 | 测试需求明确，测试计划明确 |
| 工作内容 | 根据每一步测试计划编写全部的测试用例 |
| 退出标准 | 测试用例需要覆盖大部分测试需求 |
| 责任人 | 全体组员 |

## 3.3 测试用例与需求用例对照表

本次测试中，小组挑选了各模块中核心及具有代表性需求用例进行测试用例的设计。测试用例与需求用例的对应表如下：

|  |  |  |
| --- | --- | --- |
| 模块 | 需求用例 | 测试用例 |
| 容器的基本模块 | 容器初始化 | 容器工厂初始化运行测试 |
| Handler转换器运行结果验证 | Handler转换器运行结果验证测试 |
| 验证是否正确获得组件名称信息 | 获得工厂内组件名称信息的验证测试 |
| 验证依赖关系获取 | 验证依赖正确获取测试 |
| 容器的扩展模块 | 解析配置文件 | 解析配置文件测试 |

# 4.容器的基本功能

## 4.1容器的初始化

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 容器工厂初始化运行测试 | |
| Brief Description | 验证容器的工厂初始化功能是否成功 | |
| Precondition | 容器初始化开始 | |
| Tester | 测试员 | |
| Dependency | None | |
| Test Setup | Name | 工厂初始化 |
| Description | 打开所设置的单元测试函数，运行单元测试函数 |
| Basic Flow  (Test Setup) | Steps | |
| 1 | 测试员启动操作系统中的eclipse； |
| 2 | 测试员利用单元测试testfactory.java进行测试； |
| Postcondition  (Test Oracle) | 运行测试函数，等待测试函数执行 |
| Basic Flow  (Test Sequence) | Steps | |
| 1 | 测试员为工厂初始化的测试设置一个测试函数，用来测试初始化过程中转换器存储区的正确运行 |
| 2 | 测试员运行测试函数； |
| 3 | 测试员测试默认Handler转换器能否成功创建 |
|  | 系统VALIDATES THAT默认Handler转换器成功创建 |
| 4 | 测试员测试命名的Handler转换器能否覆盖刚才创建的默认转换器 |
| 5 | 利用程序插桩技术，VALIDATES THAT创建成功后输出语句； |
| 6 | 测试员利用程序插桩技术测试缓存区，VALIDATES THAT创建成功； |
| 7 | 测试员利用程序插桩技术测试服务定义信息的存储区，VALIDATES THAT创建成功； |
| Postcondition  (Test Oracle) | 在工厂初始化测试成功 |
| Specific Alternative Flows  (Test Sequence) | RFS 4-7 | |
| 1 | 使用默认的Handler转换器继续完成创建工作 |
| 2 | RESUME STEP 5 |
| Postcondition  (Test Sequence) | 在工厂初始化测试成功 |

图1 容器工厂初始化测试用例规格说明

## 4.2验证转换器

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | Handler转换器运行结果验证测试 | |
| Brief Description | 验证容器的Handler转换器能否成功得到注解类中的 | |
| Precondition | 服务定义信息的存储区创建成功 | |
| Tester | 测试员 | |
| Dependency | None | |
| Test Setup | Name | Handler转换器 |
| Description | 利用Handler转换器获得注解相关的信息 |
| Basic Flow  (Test Setup) | Steps | |
| 1 | 测试员启动操作系统中的eclipse； |
| 2 | 测试员利用单元测试testhandler.java进行测试； |
| Postcondition  (Test Oracle) | 运行测试函数，等待测试函数执行 |
| Basic Flow  (Test Sequence) | Steps | |
| 1 | 测试员为Handler转换器的测试设置一个测试函数，用来测试注解信息能否正确获得 |
| 2 | 测试员运行测试函数； |
| 3 | 测试员利用程序插桩技术测试服务定义信息的存储区； |
| 4 | 测试员VALIDATES THAT创建成功； |
| 5 | 测试员通过类名称获得的包名并VALIDATES THAT与实际是否相符； |
| 6 | 测试员利用插桩方法VALIDATES THAT 包名正确； |
| 7 | 测试员通过插桩方法判断获得组件的名称类型是否正确，VALIDATES THAT 名称类型正确； |
| 8 | 测试员VALIDATES THAT 名称类型正确； |
| 9 | Do 测试员获得并验证方法与实际是否相符； |
| 10 | 测试员 VALIDATES THAT 名称类型正确； |
| 11 | UNTIL 注解的方法不存在 |
| 12 | Do 测试员获得并验证注解与实际是否相符； |
| 13 | 测试员 VALIDATES THAT 名称类型正确； |
| 14 | UNTIL 注解的参数不存在 |
| Postcondition  (Test Oracle) | Handler转换器运行成功 |
| Specific Alternative Flows  (Test Sequence) | RFS 4 | |
| 1 | 系统VALIDATES THAT服务定义信息的存储区创建失败 |
| Postcondition  (Test Sequence) | Handler转换器测试失败 |
| Specific Alternative Flows  (Test Sequence) | RFS 5 | |
| 1 | 测试员 VALIDATES THAT 包名错误 |
| 2 | RESUME STEP 5 |
| Postcondition  (Test Sequence) | Handler转换器测试失败 |
| Specific Alternative Flows  (Test Sequence) | RFS 7 | |
| 1 | 测试员 VALIDATES THAT 方法名错误 |
| 2 | RESUME STEP 7 |
| Postcondition  (Test Sequence) | Handler转换器测试失败 |

图2 容器Handler转换器测试用例规格说明

## 4.3获取组件信息

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 获得工厂内组件名称信息的验证测试 | |
| Brief Description | 将数据存入工厂后，从工厂提取信息 | |
| Precondition | 工厂的初始化以及handler转换器的测试成功 | |
| Tester | 测试员 | |
| Dependency | None | |
| Test Setup | Name | 获得工厂内信息 |
| Description | 利用初始化后的工厂获得组件名称信息 |
| Basic Flow  (Test Setup) | Steps | |
| 1 | 测试员启动操作系统中的eclipse； |
| 2 | 测试员利用单元测试testdata.java进行测试； |
| Postcondition  (Test Oracle) | 运行测试函数，等待测试函数执行 |
| Basic Flow  (Test Sequence) | Steps | |
| 1 | 测试员为工厂数据获得的测试设置一个测试函数，用来测试组件信息能否正确获得 |
| 2 | 测试员运行测试函数； |
| 3 | 测试员利用程序插桩技术测试工厂初始化成功，并且handler转换器成功获得组件信息，并存储； |
| 4 | 测试员VALIDATES THAT工厂初始化成功并且handler转换器运行成功； |
| 5 | 测试员从服务信息存储区中获得要找的组件信息 |
| 6 | 测试员VALIDATES THAT 组件信息在缓存中； |
| 7 | 测试员判断从缓存中获得的组件名称是否正确 |
| Postcondition  (Test Oracle) | 获得了所寻找组件的相关信息 |
| Specific Alternative Flows  (Test Sequence1) | RFS 3 | |
| 1 | 系统VALIDATES THAT初始化失败或者handler转换器获取信息失败 |
| Postcondition  (Test Sequence) | 工厂中获得组件信息测试失败 |
| Specific Alternative Flows  (Test Sequence2) | RFS 5 | |
| 1 | 测试员 VALIDATES THAT 组件信息不在缓存中 |
| 2 | 测试员判断根据服务信息创建的组件是否成功创建 |
| 3 | 测试员 VALIDATES THAT 组件信息成功创建 |
| 4 | 测试员将组件信息存入缓存中是否成功 |
| 5 | 测试员 VALIDATES THAT 存入缓存成功 |
| 6 | 测试员判断从缓存中获得的组件名称是否正确 |
| Postcondition  (Test Sequence) | 获得了所寻找组件的相关信息 |
| Specific Alternative Flows  (Test Sequence) | RFS Test Sequence2-2 | |
| 1 | 测试员 VALIDATES THAT 创建失败 |
| 2 | RESUME Test Sequence2 STEP 2 |
| Postcondition  (Test Sequence) | 工厂中获得组件信息测试失败 |
| Specific Alternative Flows  (Test Sequence) | RFS Test Sequence2-4 | |
| 1 | 测试员 VALIDATES THAT 存入缓存失败 |
| 2 | RESUME Test Sequence2 STEP 4 |
| Postcondition  (Test Sequence) | 工厂中获得组件信息测试失败 |

图3 容器工厂获取组件名称测试用例规格说明

## 4.4验证依赖关系是否正确处理

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 容器出现组件依赖关系handler转换器处理运行测试 | |
| Brief Description | 验证容器的工厂在出现依赖关系时能否正在运行 | |
| Precondition | None | |
| Tester | 测试员 | |
| Dependency | None | |
| Test Setup | Name | 组件依赖关系handler转换器处理 |
| Description | 打开所设置的单元测试函数，运行单元测试函数 |
| Basic Flow  (Test Setup) | Steps | |
| 1 | 测试员启动操作系统中的eclipse； |
| 2 | 测试员利用单元测试testdependence.java进行测试； |
| Postcondition  (Test Oracle) | 运行测试函数，等待测试函数执行 |
| Basic Flow  (Test Sequence) | Steps | |
| 1 | 测试员为容器的依赖测试设置一个测试函数，用来测试初始化过程中出现依赖关系时的正确运行 |
| 2 | 测试员运行测试函数； |
| 3 | 测试员测试在利用handler转换器获得组件信息时遇见依赖关系时，先处理依赖关系组件 |
| 4 | 测试员VALIDATES THAT依赖组件先被处理 |
| 5 | 测试员测试当依赖组件处理后，能否正确回到原组件 |
| 6 | 测试员VALIDATES THAT原组件被处理； |
| Postcondition  (Test Oracle) | 依赖关系组件handler转换器处理测试成功 |
| Specific Alternative Flows  (Test Sequence1) | RFS 3 | |
| 1 | 测试员VALIDATES THAT原组件被处理 |
| Postcondition  (Test Sequence) | 依赖关系组件handler转换器处理测试失败 |
| Specific Alternative Flows  (Test Sequence2) | RFS 5 | |
| 1 | 测试员VALIDATES THAT处理直接结束 |
| Postcondition  (Test Sequence) | 依赖关系组件handler转换器处理测试失败 |

图4 容器出现组件依赖关系handler转换器处理测试用例规格说明

## 4.5依赖关系下组件信息获取验证

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 容器出现组件依赖关系获得组件信息处理运行测试 | |
| Brief Description | 验证容器的工厂在出现依赖关系时能否正在运行 | |
| Precondition | None | |
| Tester | 测试员 | |
| Dependency | None | |
| Test Setup | Name | 组件依赖关系获得组件信息处理 |
| Description | 打开所设置的单元测试函数，运行单元测试函数 |
| Basic Flow  (Test Setup) | Steps | |
| 1 | 测试员启动操作系统中的eclipse； |
| 2 | 测试员利用单元测试testdependence2.java进行测试； |
| Postcondition  (Test Oracle) | 运行测试函数，等待测试函数执行 |
| Basic Flow  (Test Sequence) | Steps | |
| 1 | 测试员为容器的依赖测试设置一个测试函数，用来测试组件出现依赖关系时的获得组件信息的正确运行 |
| 2 | 测试员运行测试函数； |
| 3 | 测试员测试在利用工厂获得组件存储信息提供者时遇见依赖关系时，先处理依赖关系组件 |
| 4 | 测试员VALIDATES THAT依赖组件先被处理 |
| 5 | 测试员测试当依赖组件处理后，能否正确回到原组件 |
| 6 | 测试员VALIDATES THAT原组件被处理； |
| Postcondition  (Test Oracle) | 依赖关系组件获得组件信息处理测试成功 |
| Specific Alternative Flows  (Test Sequence1) | RFS 3 | |
| 1 | 测试员VALIDATES THAT原组件被处理 |
| Postcondition  (Test Sequence) | 依赖关系组件获得组件信息处理测试失败 |
| Specific Alternative Flows  (Test Sequence2) | RFS 5 | |
| 1 | 测试员VALIDATES THAT处理直接结束 |
| Postcondition  (Test Sequence) | 依赖关系组件获得组件信息处理测试失败 |

图4 容器出现组件依赖关系handler转换器处理测试用例规格说明

# 5容器的扩展功能

## 5.1解析配置文件

|  |  |  |
| --- | --- | --- |
| **Test Case Specification** | | |
| Name | 解析配置文件功能测试 | |
| Brief Description | 验证解析配置文件功能是否完善 | |
| Precondition |  | |
| Tester | 测试员 | |
| Dependency | None | |
| Test Setup | Name | 解析功能测试准备 |
| Description | 运行解析功能测试代码的准备工作 |
| Basic Flow  (Test Setup) | Steps | |
| 1 | 测试员将配置文件置于测试代码相同的文件夹下； |
| 2 | 测试员启动操作系统中的eclipse并导入测试代码至工程中；； |
| Postcondition  (Test Oracle) | 代码导入成功，无报错 |
| Basic Flow  (Test Sequence) | Steps | |
| 1 | 测试员编写一个测试bean，该测试bean与配置文件对应（包含了基本类型属性，容器类属性，内部bean引用，外部bean引用，带参数的构造函数，不带参数的构造函数； |
| 2 | 测试员实例化一个解析器，使用解析器读取配置文件 |
| 3 | 测试员使用容器实例化一个步骤1编写的测试bean |
|  | 测试员打印该测试bean的基本类型属性值 |
| 4 | 测试员打印该测试bean的容器类属性的值 |
| 5 | 测试员用容器实例化该测试bean的容器类，并打印其  属性的值 |
| 6 | 测试员用容器实例化该测试bean的外部引用类 |
| 7 | 测试员用容器实例化该测试bean的内部引用类 |
| 8 | 测试员运行测试代码，检查各显示属性值与配置文件中定义是否相同 |
| Postcondition  (Test Oracle) | 各显示属性值与配置文件中定义相同 |
| Specific Alternative Flows  (Test Sequence) | RFS 8 | |
| 1 | 各显示属性值与配置文件中定义不相同 |
| 2 | 检查不相同属性对应的解析功能的代码实现 |
| 3 | 修改对应部分代码 |
| 4 | RESUME STEP 8 |
| Postcondition  (Test Sequence) | 各显示属性值与配置文件中定义相同 |

# 6参考文献

[1] http://junit.org/junit4/

[2] http://huihai.iteye.com/blog/1986568

[3] 肖丁、吴建林等. 软件工程模型与方法[M]. 北京邮电大学出版社. 2008-03.

[4] http://jingyan.baidu.com/article/fcb5aff7e1c277edaa4a71ee.html

[5] 蔡建平. 软件测试实验指导教程[M]. 清华大学出版社. 2009-11.