

NPU BUS

软件测试说明

编写人员： 苏畅、王泽堃

校对人员： 刘龙涛

编写日期： 2019年7月10日

文档版本： 1.0版

**目录**

[1.引言 3](#_Toc13728147)

[1.1标识 3](#_Toc13728148)

[1.2系统概述 3](#_Toc13728149)

[1.3文档概述 4](#_Toc13728150)

[2.引用文件 4](#_Toc13728151)

[《计算机软件文档编制规范GBT85672006》 4](#_Toc13728152)

[《西北工业大学学生手册》 4](#_Toc13728153)

[3测试准备 4](#_Toc13728154)

[3.x(测试的项目唯一标识符) 4](#_Toc13728155)

[3.x.1硬件准备 6](#_Toc13728156)

[3.x.2软件准备 8](#_Toc13728157)

[3.x.3其他测试前准备 10](#_Toc13728158)

[4测试说明 10](#_Toc13728159)

[4.x(测试的项目唯一标识符) 11](#_Toc13728160)

[4.x.y(测试用例的项目唯一标识符) 11](#_Toc13728161)

[4.x.y.1涉及的需求 12](#_Toc13728162)

[4.x.y.2先决条件 13](#_Toc13728163)

[4.x.y.3测试方法 13](#_Toc13728164)

[4.x.y.4结果分析 16](#_Toc13728165)

[4.x.y.5评价结果的准则 16](#_Toc13728166)

[4.x.y.6测试结果 18](#_Toc13728167)

[4.x.y.7假设和约束 18](#_Toc13728168)

[5需求的可追踪性 19](#_Toc13728169)

[6注解 20](#_Toc13728170)

# 1.引言

## 1.1标识

中文名称：软件测试说明

英文名称：“ Software Testing Document（STD）”

文档版本：“0.1”。

文档编号：NPU-BUS- STD-0.1

## 1.2系统概述

本文档适用于“西北工业大学校车管理开发”项目（以下简称“NPUBUS项目”）的开发过程。NPUBUS项目，由本小组负责实施，该项目标识号为“NPU-BUS”，其软件产品版本号为“1.0”，包括三个内部版本，分别是0.1版、0.2版和0.3版。

项目内容为：

本产品是一款校车服务平台，分为管理层面和用户层面。目的是为了解决近年来，随着

新老校区间的课程安排调度以及每年的学生数量不断增加，乘坐校车在新老校区来往的人数也相应有所增多，对校车工作方的校车调度和发车数量控制有了更高的要求。鉴于校车工作方的数据来源仅仅是根据个人的经验去判断，尚未实现数字化规范化程序化，导致经常有学生等待数量及校车数量之间出现了矛盾，资源未能充分利用好、学生和老师权利不得已正常落实的情况等现实问题。这个平台可以实现给校车工作方提供准确的人数大数据，校车工作方能根据实时情况智能调度学校班车；能够让学生免去苦苦等待校车却上不了校车的情况。

## 1.3文档概述

本文阐述了软件的开发需求，用于指导项目的开发人员进行设计、开发和测试，以及客户对项目情况进行更具体的了解，是整个项目后续工作的基础。

它描述了系统展现给用户的行为和执行的操作等。它包括产品必须遵从的标准、规范和合约；外部界面的具体细节；性能要求；设计或实现的约束条件及质量属性。所谓约束是指对开发人员在软件产品设计和构造上的限制。质量属性是通过多种角度对产品的特点进行描述，从而反映产品功能。用户解决问题或达到目标所需条件或权能(Capability)。 (2)系统或系统部件要满足合同、标准、规范或其它正式规定文档所需具有的条件或权能。 (3)一种反映上面或所述条件或权能的文档说明。它包括功能性非功能性需求，非功能性需求对设计和实现提出了限制，比如性能要求，质量标准，或者设计限制。

# 2.引用文件

# 《计算机软件文档编制规范GBT85672006》

# 《西北工业大学学生手册》

《软件开发计划（SDP）.Docx》

《软件配置管理计划（SCMP）.Docx》

《结构设计说明.docx》

《可行性分析报告（FAR）.Docx》

# 3测试准备

本章应分以下几条，(若适用)应包括用“警告”或“注意”标记的安全提示和保密性与私密性考虑。

## 3.x(测试的项目唯一标识符)

本条应用项目唯一标识符标识一个测试并提供简要说明，应分为以下几条。当所需信息与前面为另一测试所指出的信息重复时，此处可作引用而无需重复。

**内建关建字**

|  |  |  |
| --- | --- | --- |
| as | break | case |
| catch | class | const |
| continue | default | delete |
| do | else | extends |
| typeof | finally | for |
| function | if | implements |
| import | in | instanceof |
| interface | internal | is |
| native | new | null |
| package | private | protected |
| public | return | super |
| switch | this | throw |
| to | true | try |
| use | var | void |
| while | with | false |

**应避免合用的**关键字

|  |  |  |
| --- | --- | --- |
| each | get | set |
| namespace | include | dynamic |
| final | native | override |
| static |  |  |

（5） 尽量避免使用保留字保留关键字

|  |  |  |
| --- | --- | --- |
| abstract | boolean | byte |
| cast | char | debugger |
| double | enum | export |
| float | goto | intrinsic |
| long | prototype | short |
| synchronized | throws | to |
| transient | type | virtual |
| volatile | dim | mx |

**常见的**标识符命名错误

|  |  |  |
| --- | --- | --- |
| 合法标识符 | 非法标识符 | 注释 |
| fromNo12 | from#12 | 标识符中不能使用#符号 |
| my\_Boolean | my-Boolean | 标识符中不能使用“-”符号，应使用“\_”代替 |
| Obj2 | 2ndObj | 标识符不能使用数字开头 |
| myclass | class | “class”是内建关键字 |
| jack\_rose | jack&rose | 符号“&”不能在标识符中使用 |
| GUI | G.U.I | 标识符内部不能出现“.”分隔符 |
| mybreak() | break() | “break”是内建关建字不能用于自定义函数名 |

## 3.x.1硬件准备

本条应描述为进行测试工作需要做的硬件准备过程。有关这些过程可以引用已出版的操作手册。(若适用)应提供以下内容：

a.要使用的特定硬件，用名字和(若适用)编号标识；

b.任何用于连接硬件的开关设置和电缆；

c.说明硬件、互联控制和数据路径的一个或多个图示；

d.使硬件处于就绪状态的分步指令。

需要能够运行Windows 7/8/10操作系统的计算机，安装并配置软件Microsoft Office Access数据库管理系统。

最低配置:

操作系统: Windows 7 64 Bit

处理器: Intel Core 2 Duo 3.00Ghz

内存: 4 GB RAM

图形: GTX 650 Ti 1GB|HD 7850 1GB|Intel UHD Graphics 620

DirectX 版本: 11

存储空间: 需要 60 GB 可用空间

附注事项: 6GB Memory if using integrated GPU

推荐配置:

操作系统: Windows 10 64 Bit

处理器: Intel i5-6600 | Ryzen 5 2600X

内存: 8 GB RAM

图形: GTX 970 | R9 Fury X 4GB VRAM

DirectX 版本: 11

存储空间: 需要 60 GB 可用空间

最低配置:

操作系统: macOS 10.14.4

处理器: 2GHz Intel Core i5

内存: 8 GB RAM

图形: 2GB Nvidia 680MX, 2GB AMD R9 M290, 1.5GB Intel Iris Graphics 540 or better (see notes for more details)

存储空间: 需要 52 GB 可用空间

附注事项: The game is officially supported on the following Macs. To check your Mac model and when it was released, select About This Mac from the Apple menu on your menu bar.

\* All 13" MacBook Pros released since 2016

\* All 15" MacBook Pros released since Late 2013 with a 2GB graphics card or better

\* All 21.5" iMacs released since 2017

\* All 27" iMacs released since Late 2012 with a 2GB graphics card or better

\* All 27" iMac Pros released since 2017

\* All Mac Pros released since Late 2013

最低配置:

操作系统: Ubuntu 18.04 64-bit

处理器: 3.4GHz Intel Core i3-4130

内存: 6 GB RAM

图形: 2GB AMD R9 285 (GCN 3rd Generation), 2GB Nvidia GTX 680

存储空间: 需要 60 GB 可用空间

附注事项:

\* Requires Vulkan

\* Nvidia requires 418.56 or newer drivers.

\* AMD requires Mesa 19.0.1. ​

\* AMD GCN 3rd Gen GPU's include the R9 285, 380, 380X, Fury, Nano, Fury X.

\* Intel GPUs are not supported at time of release.

\* Other modern drivers and distributions are expected to work but are not officially supported.

## 3.x.2软件准备

本条应描述为测试准备被测项和其他有关软件，包括用于测试的数据的必要过程。有关这些过程，可以引用已出版的软件手册。(若适用)应提供下述信息：

a.测试中要使用的特定软件；

b.被测项的存储媒体(如磁带、盘)；

c.任何相关软件(如模拟器、测试驱动程序、数据库)的存储媒体；

d.加载软件的指令，包括所需的顺序；

e.多个测试用例共同使用的软件初始化指令。

开源测试管理工具：Bugfree、Bugzilla、TestLink、mantis 开源功能自动化测试工具：Watir、Selenium、MaxQ、WebInject

开源性能自动化测试工具：Jmeter、OpenSTA、DBMonster、TPTEST、Web Application Load Simulator

[TestDirector]：企业级测试管理工具，也是业界第一个基于Web的测试管理系统。

[Quality Center]：基于Web的测试管理工具，可以组织和管理应用程序测试流程的所有阶段，包括指定测试需求、计划测试、执行测试和跟踪缺陷。

[QuickTest Professional]：用于创建功能和回归测试。

[LoadRunner]：预测系统行为和性能的负载测试工具。

其他工具与自动化测试框架：Rational Functional Tester、Borland Silk系列工具、WinRunner、Robot等。

国内免费软件测试工具有：AutoRunner和TestCenter。

PRQA公司成立于1986年,总部在英国。PRQA被世界范围内的高级软件开发人员、行业专家、标准团体认可为编程标准专家。其一直致力于通过静态分析来自动化地检查编程标准的遵循并发现软件的缺陷。

  PRQA的主要产品包括： QAC/QAC++，QA.MISRA C/QA.MISRA C++，QA Verify。

**QAC/QAC++是用于代码规则检查的自动化工具，**代码审查主要检查代码和设计的一致性，代码对标准的遵循、可读性，代码的逻辑表达的正确性，代码结构的合理性等方面；可以发现违背程序编写标准的问题，程序中不安全、不明确和模糊的部分，找出程序中不可移植部分、违背程序编程风格的问题，包括变量检查、命名和类型审查、程序逻辑审查、程序语法检查和程序结构检查等内容。

**MISRA Compliance 模块为QAC可选组件**，执行MISRA 2004准则检查，在MISRA Compliance 模块的帮助下，分析源代码查找不符合MISRA的结构。QA C的警告信息直接通过HTML连接到被分析的源代码，同时也与MISRA相关规则参考信息连接。这些参考信息包括MISRA兼容代码中解释性的例子和标准描述。

**QA Verify**是集代码静态分析和质量管理于一体的软件代码质量管理平台，利于客户端-服务器的架构方式，无缝地集成到日常的开发流程中，最大化地发挥团队协作部署和管理的效率。

**PRQA参与和遵从以下标准指定：**

-ISO C委员会成员

-ISO C++决策成员

-MISRA C创始人

-MISRA C++创始人

**提供的产品和服务的领域包括：**

-航空航天、汽车、国防、半导体、金融、工业应用、医疗设备、通信

## 3.x.3其他测试前准备

本条应描述进行测试前所需的其他人员活动、准备或过程。

# 4测试说明

本章应分为以下几条。(若适用)应包括用“警告”或“注意”标记的安全提示和保密性与私密性考虑。

JAVA反射API介绍

在JDK中，主要由以下类来实现Java反射机制，这些类（除了第一个）都位于java.lang.reflect包中

Class类：代表一个类，位于java.lang包下。

Field类：代表类的成员变量（成员变量也称为类的属性）。

Method类：代表类的方法。

Constructor类：代表类的构造方法。

Array类：提供了动态创建数组，以及访问数组的元素的静态方法。

## 4.x(测试的项目唯一标识符)

本条应用项目唯一标识符标识一个测试，并分为以下几条。当所需信息与以前提供的信息重复时，此处可作引用而无需重复。

遥控器输入接口（YKQSR\_JK）

 温度传感器输入接口（CGQSR\_JK）

 控制工作电机组输出接口（KZSC\_JK）

public class Welcome{

public static void main(String args[]){

int $ = 3 ;

int $abc = 5 ;

int \_123 = 5 ;

int n = 10;//java内部采用Unicode字符集

//int 123abc = 1;//标识符不能以数字开头

//int abc# = 1 ;//标识符不能包含除了以字母、数字、下划线、$之外的其他字符

//int class = 3;//标识符不能使用Java关键字

}

}

### 4.x.y(测试用例的项目唯一标识符)

本条应用项目唯一标识符标识一个测试用例，说明其目的并提供简要描述。下述各条提供测试用例的详细说明。

测试用例可以分为基本事件、备选事件和异常事件。设计基本事件的用例，应该参照用例规约（或设计规格说明书），根据关联的功能、操作按路径分析法设计测试用例。而对孤立的功能则直接按功能设计测试用例。基本事件的测试用例应包含所有需要实现的需求功能，覆盖率达100%。

设计备选事件和异常事件的用例，则要复杂和困难得多。例如，字典的代码是唯一的，不允许重复。测试需要验证：字典新增程序中已存在有关字典代码的约束，若出现代码重复必须报错，并且报错文字正确。往往在设计编码阶段形成的文档对备选事件和异常事件分析描述不够详尽。而测试本身则要求验证全部非基本事件，并同时尽量发现其中的软件缺陷。

可以采用软件测试常用的基该方法：等价类划分法、边界值分析法、错误推测法、因果图法、逻辑覆盖法等设计测试用例。视软件的不同性质采用不同的方法。如何灵活运用各种基该方法来设计完整的测试用例，并最终实现暴露隐藏的缺陷，全凭测试设计人员的丰富经验和精心设计。

### 4.x.y.1涉及的需求

本条应标识测试用例所涉及的CSCI需求或系统需求(此信息亦可在5.a中提供)。

早期的软件开发过程中软件规模都很小、复杂程度低，软件开发的过程混乱无序、相当随意，测试的含义比较狭窄，开发人员将测试等同于“调试”，目的是纠正软件中已经知道的故障，常常由开发人员自己完成这部分的工作。对测试的投入极少，测试介入也晚，常常是等到形成代码，产品已经基本完成时才进行测试。到了上世纪80年代初期，软件和IT行业进入了大发展，软件趋向大型化、高复杂度，软件的质量越来越重要。这个时候，一些软件测试的基础理论和实用技术开始形成，并且人们开始为软件开发设计了各种流程和管理方法，软件开发的方式也逐渐由混乱无序的开发过程过渡到结构化的开发过程，以结构化分析与设计、结构化评审、结构化程序设计以及结构化测试为特征。人们还将“质量”的概念融入其中，软件测试定义发生了改变，测试不单纯是一个发现错误的过程，而且将测试作为软件质量保证(SQA)的主要职能，包含软件质量评价的内容，Bill Hetzel在《软件测试完全指南》(Complete Guide of Software Testing)一书中指出：“测试是以评价一个程序或者系统属性为目标的任何一种活动。测试是对软件质量的度量。”这个定义至今仍被引用。软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题。

### 4.x.y.2先决条件

本条应标识执行测试用例前必须建立的先决条件，(若适用)应讨论以下内容：

a.软、硬件配置；

b.测试开始之前需设置或重置的标志、初始断点、指针、控制参数或初始数据；

c.运行测试用例所需的预置硬件条件或电气状态；

d.计时度量所用的初始条件；

e.模拟环境的条件；

f.测试用例特有的其他特殊条件。

### 4.x.y.3测试方法

1、按是否查看程序内部结构分为：

（1）黑盒测试（black-box testing）：只关心输入和输出的结果

（2）白盒测试（white-box testing）：去研究里面的源代码和程序结构

2、按是否运行程序分为：

（1）静态测试（static testing）：是指不实际运行被测软件，而只是静态地检查程序代码、界面或文档可能存在的错误的过程。

静态测试包括：

对于代码测试，主要是测试代码是否符合相应的标准和规范。

对于界面测试，主要测试软件的实际界面与需求中的说明是否相符。

对于文档测试，主要测试用户手册和需求说明是否真正符合用户的实际需求。

（5）动态测试（dynamic testing），是指实际运行被测程序，输入相应的测试数据，检查输出结果和预期结果是否相符的过程

3、按阶段划分：

（1）单元测试（unit testing），是指对软件中的最小可测试单元进行检查和验证。

桩模块（stud）是指模拟被测模块所调用的模块，驱动模块（driver）是指模拟被测模块的上级模块，驱动模块用来接收测试数据，启动被测模块并输出结果。

（2）集成测试（integration testing），是单元测试的下一阶段，是指将通过测试的单元模块组装成系统或子系统，再进行测试，重点测试不同模块的接口部门。

集成测试就是用来检查各个单元模块结合到一起能否协同配合，正常运行。

（3）系统测试（system testing），指的是将整个软件系统看做一个整体进行测试，包括对功能、性能，以及软件所运行的软硬件环境进行测试。

系统测试的主要依据是《系统需求规格说明书》文档。

（4）验收测试（acceptance testing），指的是在系统测试的后期，以用户测试为主，或有测试人员等质量保障人员共同参与的测试，它也是软件正式交给用户使用的最后一道工序。

验收测试又分为a测试和beta测试，其中a测试指的是由用户、 测试人员、开发人员等共同参与的内部测试，而beta测试指的是内测后的公测，即完全交给最终用户测试。

4、黑盒测试分为功能测试和性能测试：

1)功能测试（function testing），是黑盒测试的一方面，它检查实际软件的功能是否符合用户的需求。

包括逻辑功能测试（logic function testing）

界面测试（UI testing）UI=User Interface

易用性测试（usability testing）：是指从软件使用的合理性和方便性等角度对软件系统进行检查，来发现软件中不方便用户使用的地方。

兼容性测试（compatibility testing）：包括硬件兼容性测试和软件兼容性测试

2)性能测试（performance testing）

软件的性能主要有时间性能和空间性能两种

时间性能：主要指软件的一个具体事务的响应时间（respond time）。

空间性能：主要指软件运行时所消耗的系统资源。

软件性能测试分为：

一般性能测试：指的是让被测系统在正常的软硬件环境下运行，不向其施加任何压力的性能测试。

稳定性测试也叫可靠性测试（reliability testing）：是指连续运行被测系统检查系统运行时的稳定程度。

负载测试（load testing）：是指让被测系统在其能忍受的压力的极限范围之内连续运行，来测试系统的稳定性。

压力测试（stress testing）：是指持续不断的给被测系统增加压力，直到将被测系统压垮为止，用来测试系统所能承受的最大压力。(Validate the system or software can allowed the biggest stress.)

5、其他测试类型：

回归测试（regression testing）是指对软件的新的版本测试时，重复执行上一个版本测试时的用例。(When a new build or release is deployed, repeat all the test cases which has executed in the last build or release.)

冒烟测试（smoke testing），是指在对一个新版本进行大规模的测试之前，先验证一下软件的基本功能是否实现，是否具备可测性。(validate the major function is deployed or not in software of system when a new build or release is implement.)

随机测试（random testing），是指测试中所有的输入数据都是随机生成的，其目的是模拟用户的真实操作，并发现一些边缘性的错误。(means or all the test data is random, to validate the some edge bugs.)

软件测试包括的步骤：

1.制定测试计划；

2.设计测试用例、

3.实施测试（首先要搭建测试用环境）、

4.管理测试时发现的BUG、

5.测试完后（测试完，并且发现的BUG修正完）

6.要做测试报告（这样，该测试过程就算结束了，每种类型（单元测试、集成测试、系统测试、验证测试）的测试都是如此）；

### 4.x.y.4结果分析

通过稳定性测试，发现系统在进行稳定性测试前后，性能没有明显的变化，而且系统的资源使用情况也没有明显的增加，说明系统的稳定性良好。

从稳定性（疲劳）测试的结果看，系统表现良好，在30小时50用户的持续压力下，运行稳定，尽管在测试过程中进行应用操作比较缓慢，但并没有出现访问失败和发生错误的现象。

从服务器的性能表征上看，整个系统性能表现都比较平稳，相对来说：应用服务器的CPU和数据库服务器的IO操作的表现比较活跃。应用服务器的cpu利用率较高，在70％左右，在正式环境中，建议也要尽量使用较高的CPU配置；数据库服务器的IO操作比较明显，因为测试用例操作对存储设备的交互很多，也是体现的正常的用例特点，在后期，IO提升，分析是由于回滚段资源紧张导致，这种情况在正式运行中，可以通过对数据库结构的合理管理规划来避免；内存方面应用服务器和数据库服务器的可用内存都很平稳，表明没有明显的内存泄漏的情况发生。

从整体上看，系统在稳定性方面表现良好，可以在正式环境中稳定运行。

### 4.x.y.5评价结果的准则

具体地讲，测试一般要达到下列目标：  
1、确保产品完成了它所承诺或公布的功能，并且所有用户可以访问到的功能都有明确的书面说明------在某种意义上与ISO9001是同一种思想。  
产品缺少明确的书面文档，是厂商一种短期行为的表现，也是一种不负责任的表现。所谓短期行为，是指缺少明确的书面文档既不利于产品最后的顺利交付，容易与用户发生矛盾，影响厂商的声誉和将来与用户的合作关系；同时也不利于产品的后期维护，也使厂商支出超额的用户培训和技术支持费用。从长期利益看，这是很不划算的。领测认为接触过的软件产品，很少有向方正这样大大的产品、薄薄的文档。  
当然，书面文档的编写和维护工作对于使用快速原型法(RAD)开发的项目是最为重要的、最为困难，也是最容易被忽略的。  
最后，书面文档的不健全甚至不正确，也是测试工作中遇到的最大和最头痛的问题，它的直接后果是测试效率低下、测试目标不明确、测试范围不充分，从而导致最终测试的作用不能充分发挥、测试效果不理想。  
2、 确保产品满足性能和效率的要求  
使用起来系统运行效率低(性能低)、或用户界面不友好、用户操作不方便(效率低)的产品不能说是一个有竞争力的产品。  
用户最关心的不是你的技术有多先进、功能有多强大，而是他能从这些技术、这些功能中得到多少好处。也就是说，用户关心的是他能从中取出多少，而不是你已经放进去多少。  
3、 确保产品是健壮的和适应用户环境的  
健壮性即稳定性，是产品质量的基本要求，尤其对于一个用于事务关键或时间关键的工作环境中。  
另外就是不能假设用户的环境(某些项目可能除外)，如：报业用户许多配置是比较低的，而且是和某些第三方产品同时使用的。  
  
测试的原则---Good Enough  
对于相对复杂的产品或系统来说，zero-bug是一种理想，good-enough是我们的原则。  
Good-enough原则就是一种权衡投入/产出比的原则：不充分的测试是不负责任的；过分的测试是一种资源的浪费，同样也是一种不负责任的表现。我们的操作困难在于：如何界定什么样的测试是不充分的， 什么样的测试是过分的。目前状况唯一可用的答案是：制定最低测试通过标准和测试内容，然后具体问题具体分析。最明显的例子就是FIT3.0中文报版的产品测试。  
  
测试的规律----木桶原理和80-20原则

### 4.x.y.6测试结果

功能测试结论：

本次测试基本通过，遗留的问题将在以后解决，可以进入下一阶段项目目标。 建议：

1．在系统功能满足要求的情况下，增加界面的友好性。

2．在需求不是很明确的情况下，最好能让测试人员也加入到和客户的沟通中，所得测试用

例能更好地满足客户的要求。 性能测试结论：

本结论是根据在真实生产环境下做的性能测试结果得出： 1. ITMP系统的后台性能完全满足7种交换机的处理速度；

2.从两张结果表中可以得出，7种机型的IO时间平均百分比为57.7％，再由于同网元的

工单是串行施工，抛开在工单在队列中的等待时间，Tips系统处理工单的时间只占用了很少一部分时间。可以看出目前系统主要耗时的地方在于同交换机的交互等待时间； 3.根据测试得出的值和市电经验值的比较，Tips系统也能满足日后更快交换机的需求；

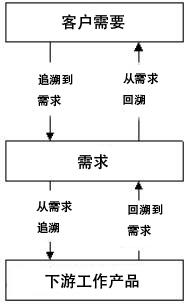
建议：

根据测试的结果和经验，建议客户能给出一份各种情景下的重连次数和指令执行超时时限的建议值，为上线的配置做参考

### 4.x.y.7假设和约束

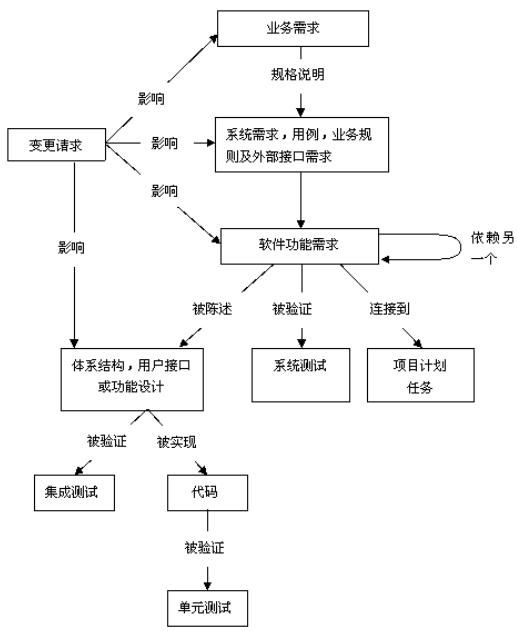
本条应标识所做的任何假设，以及在描述测试用例中由于系统或测试条件而引人的约束或限制，如时间、接口、设备、人员与数据库/数据文件的限制。如果对指定的限制和参数放弃或例外得到批准的话，应对它们加以标识，并且本条应指出它们对测试用例的影响与冲击。

# 5需求的可追踪性



上图说明了四类需求跟踪能力链。客户需求可向前追溯到需求，这样就能区分出开发过程中或开发结束后由于需求变更受到影响的需求。这也确保了需求规格说明书包括所有客户需求。同样，可以从需求回溯相应的客户需求，确认每个软件需求的源头。如果用使用实例的形式 来描述客户需求，图的上半部分就是使用实例和功能性需求之间的跟踪情况。图的下半部分指出：由于开发过程中系统需求转变为软件需求、设计、编写等，所以通过定义单个需求和特定的产品元素之间的（联系）链可从需求向前追溯。这种联系链使你知道每个需求对应的产品部件，从而确保产品部件满足每个需求。第四类 联系链是从产品部件回溯到需求，使你知道每个部件存在的原因。绝大多数项目不包括与用户需求直接相关的代码，但对于开发者却要知道为什么写这一行代码。如果不能把设计元素、代码段 或测试回溯到一个需求，你可能有一个“画蛇添足的程序”。然而，若这些孤立的元素表明了一个正当的功能，则说明需求规格说明书漏掉了一项需求。

跟踪能力联系链记录了单个需求之间的父层、互连、依赖的关系。当某个需求变更（被删除或修改）后，这种信息能够确保正确的变更传播，并将相应的任务作出正确的调整。下图说明了许多能在项目中定义的直接跟踪能力联系链。一个项目不必拥有所有种类的跟踪能力联系链，要根据具体的情况调整。



# 6注解

本文档依据《国标 GB/T 8567-2006 计算机软件文档编制规范》制定，属于

技术文档，仅限于实验室相关人员阅读。

本文档给出了本项目中文档的编号规则，所有项目文档都需按照本文档中制

订的规则进行编号，从而实现统一规范有效的文档管理。

1. 文件均采用 A4 纸幅面。文件的名称应简明准确，一般不超过 20 个汉字。

2. 文件的内容应表达准确、清楚、简明、严谨。

3. 同一文件中术语、符号、代号应统一。表达同一术语的概念应前后一致。

采用的术语尚无标准规定时且容易产生不同理解的，应给出定义或说明。

4. 文件中的缩略词(语)应采用有关标准或专业委员会认定的缩略词(语)，

自定缩略词(语)应简明，能反映主题。缩略词(语)在文件中首次出现时应做说明。

5. 文件中引用的标准和文件应是现行有效。

6. 文件中应采用国务院正式公布、实施的简化汉字。

7.关键字注解：

7.1. CSCI 是计算机软件配置项(Computer Software Configuration Item) 简称，在软件设计文档中经常用到。软件生存周期各个阶段活动的产物经审批后 即可称之为软件配置项，软件配置项包括：（1）与合同、过程、计划和产品有关 的文档和资料； （2）源代码、目标代码和可执行代码； （3）相关产品，包括软件工具、库内的可重用软件、外购软件及顾客提供的软 件等

7.2. IRS 是《接口需求规格说明》的简称

7.3. 基线是软件文档或源码（或其他产出物）的一个稳定版本，它是进一步开发 的基础。

7.4. 项目背景：本系统的软件实现计划采用 NET 框架，SQLServer 数据库，因此 具有很强的应用性、可维护性、可扩展性以及重复使用的能力，可以很方便进行 版本控制、数据库操作等复杂度比较高的控制。本项目的成果提供了一个校车预 约系统，学生和教师可以根据自己时间预约校车；管理员可以通过此平台提供的 环境对校车的调度情况进行动态管理。NET 程序设计，开发 WEB 应用程序，从而 完善课程的网络畅通程度，其中采用的 B／S 模式，让客户端从繁重的数据计算 的劳动中解脱出来，C／S 模式又提高了软件的可控与安全性，这种三层模式的 数据库访问与管理方式让我们数据库的管理与访问变得更加高效快捷。