黑盒测试用例自动生成系统的设计与实现

数学与信息学院　软件工程专业
123012016049　胡劲元　指导教师 林岭

【摘要】软件测试是软件生命周期至关重要的一步，尤其是测试用例生成的自动化更是可以大大提高测试效率从而降低测试成本。黑盒测试是软件测试后期最主要的测试方法之一，其用例的设计生成更是重中之重。使用基于C++语言的开发工具QT,在边界值和等价类划分等原理的指导下,利用自定义构造的模型设计实现了黑盒测试用例自动生成系统，设计实现多参数多因素全组合排列的递归调用算法，并加入了基于正交试验法原理生成用例的功能。复现了系统由定义问题到设计再到实现的全过程。

【关键词】软件测试；黑盒测试；自动生成测试用例；QT开发；正交试验法

目录

0 引言

随着当今互联网与电子行业高速发展，市场对于软件的需求量急剧增加，从而导致以指数函数式爆炸增加的软件产品涌入市场，此时，这成千上万的软件产品的质量优劣便与软件价值大小紧密相连。通过对于软件工程原理的把握，在软件生命周期中，软件测试是严格把控软件质量的“重要出口”，一个成功的测试用例是能够发现系统中迄今为止尚未发现的错误的测试用例，所以测试用例的成功设计便是软件测试的核心要点，而测试用例的生成更是软件测试工作的重难点与突破点，一直以来，大量研究人员也都致力于软件测试用例的生成工作[1][2]。

但是目前来看，对于软件测试用例的设计生成，我们主要还是靠测试人员手工完成，这就要求测试人员有着较高的专业素养与专业水平[3]，同时还要有一定的经验去支撑他所拥有的专业技能。这样便使得我们软件测试过程有很大的模糊性与盲目性，从而使我们测试工作效率十分低下，测试成本占开发成本比例严重上升，软件的质量也难以保证，造成软件性价比很不可观。因此，如何能在软件测试过程中自动生成测试用例便很有其研究探索的实践性与必要性。

1 黑盒测试

1.1 黑盒测试定义及作用

黑盒测试[4]是软件测试最主要的测试方法之一，不同于白盒测试[5]对于程序模块检查的结构测试，黑盒测试是基于功能的。黑盒测试不言而喻便是如同检查一个不透明的盒子，无需去了解或者研究软件系统内部源代码的具体设计或实现[6]，也完全不需要考虑内部构成与逻辑结构。测试人员只需要知道软件的用途和功能而无需明白软件功能的运行方式[7]。也就是说测试人员只需要按照软件规格说明书，在软件的接口处进行一些参数的输入，就能根据所得到的输出结果[8]判断测试结果，而正因为黑盒测试无需了解系统编程与实现问题，只是从功能与界面去测试软件系统得出测试结果，从而黑盒测试在软件测试中后期的使用非常广泛且有效，甚至未参与系统设计实现的工作人员也可以利用黑盒测试完成测试工作。常见的黑盒测试方法主要有等价类划分、边界值分析、错误分析和因果图等等。

黑盒测试主要是用于功能测试，力图发现以下错误：功能正确性的错误、功能的完善性错误、界面错误、数据结构或外部数据库访问错误、性能错误、数据输入输出差距错误、系统初始化、终止错误等等

1.2 黑盒测试方案设计

1.2.1 测试功能的有效性及合适的测试输入类型

黑盒测试主要是对于软件系统的功能性测试，那么能否准确无误的测试出系统子功能有效性的测试方案便是衡量一个方案成功与否的重要标准。在方案设计过程中，我们要从测试功能完整性、有效性、正确性等方面来考虑方案的设计与实现，同时也要注意方案的可行性与有效操作性。

我们在软件测试过程中，是通过输入测试用例以观察输出结果从而得出测试结果的，那么所输入的测试用例的参数类型的确定也是黑盒测试方案设计的重中之重。我们应考虑到单类型输入和多类组合型输入，并确定所得到的测试用例对于系统的可反应性。一个成功的黑盒测试方案是所有有效参数类型的有组织有逻辑的排列组合，而这个排列组合是具有高度准确性与全覆盖性的。

1.2.2 系统对于特定输入值是否敏感及如何划分数据边界

所谓对于数据敏感，就是系统有能力从外部输入的大量看似杂乱无章的数据之中识别出特殊的个例，以便于进行后续的操作或处理。而我们在设计黑盒测试方案的时候，应当充分考虑到所测试系统对于数据输入的处理机制，了解到系统所敏感的特定输入群，这样会有助于我们设计出行之有效的测试方案。黑盒测试方法中，对于边界值的分析有着系统的解释。那么如何正确划分数据边界，对于设计出高效的黑盒测试方案十分关键，只有数据边界的划分准确无误，我们在依靠边界进行输入数据微调的时候就能对相应的输出结果进行更准确的把控，从而使得测试方案具有更高的性价比。

1.2.3 测试用例的代表性

我们在设计黑盒测试方案的时候，原则上应当避免没有逻辑的穷举测试用例的方式。首先，穷举的方式并不能保证我们可以毫无遗漏的列举所有可能的测试用例，并且伴随着穷举测试用例数量增加的同时还有测试用例的重复率和无效率。其次，在软件测试的过程中我们还需要时刻注意到软件测试的经济学[9]问题。因此，我们在设计黑盒测试用例方案的时候，可以利用等价类划分和边界值分析法原理的相结合，在所有的测试用例中，选取具有代表性的测试用例[10]。除上述两原理之外，我们还可以利用用途非常广泛的正交实验法[11]来辅助系统设计测试用例，在资源、时间有限的条件下尽可能做到更高效、覆盖面更广以及更完整的软件测试。

1.3 黑盒测试方法分类

目前来说，黑盒测试主要包括等价类划分、边界值分析、错误推断法和因果图示法这四种常见的方法，以下是对本系统采用的两种方法及加入的正交试验法进行的分类说明。

1.3.1 等价类划分

等价划分是把程序的输入域划分为若干个数据类，由此导出测试用例。等价类划分的方法主要针对于设计出能够发现若干种程序漏洞或错误的测试用例，这样就可以减少我们在设计黑盒测试用例方案的时候所必须的用例设计。

我们把可能的输入数据划分成若干个类，作出下述设定：每一类中的数据都是属于检测同一种错误的，而在这一类中，我们可以找到一组典型的数据，它与其它数据的作用域以及作用相同，同时它是系统最为敏感的一组，那么我们就可以在每一类数据中找出一组典型数据作为测试数据，它们是具有代表性的，同时由于系统的高度敏感他们更容易发现系统程序中的错误漏洞。

1.3.2 边界值分析

在长期以来的软件测试工作中，大量经验表明我们在处理边界情况的时候是程序发生错误的高发期，因为许多程序在处理循环边界值的时候或者是在有出现下标、数据结构等等边界的时候很容易发生错误，因此如果我们尝试设计出运行在边界值附近的黑盒测试用例方案，那么我们更容易在系统相对稳定不易崩溃的情况下，获得更准确真实的错误反馈，得到更加完整的测试结果。而基于边界值分析法的原理，是选取略大于最大值+max、最大值 max、略小于最大值-max、正常值 nom、略大于最小值+min、最小值 min 和略小于最小值-min 进行测试[12][13]，结果也往往对于程序的健壮性有更好的把控。

1.3.3 正交试验法

长期以来我们在一般软件黑盒测试用例设计生成并投入到实际测试的过程当中，往往做了大量繁琐、成本高却效率差的工作：实际测试工作展开的时候，我们面对的不仅仅只有一个系统输入参数同时每一个参数的取值也几乎不可能是唯一值甚至数量十分庞大，如果按照这样的参数及其取值水平设计测试用例的话将会在一个项目中人力物力财力的苛刻限制下使程序测试员寸步难行。正交试验法便是现如今在各领域均有所建树的新兴试验原理，其主要就是探索研究多因素多水平参数组合的设计方法，在软件测试用例设计工作中，其基于正交性从全覆盖的测试用例中挑选尽可能适量、高效且覆盖广的测试用例原理极大减少了软件测试用例的繁杂性，同时予以程序测试人员工作极高的效率以及准确率[14]。

正交试验法是在概率论与数理统计的基础上借助正交表在大量参与试验的数据中挑选适量极具代表性的水平点来设计试验方案的，经过这样设计出来的试验方案是及其科学严谨的[15][16]，因而在很多科研领域都有不可忽略的利用价值。正交表规格整齐，第一行、列分别是所有参数因素的排列与按照正交试验法设计出来的所有方案标号，并且每一列不同的数字出现频率相等，而余下的每一行的排列均是对应一个设计方案。

基于正交试验法设计生成软件测试用例的时候，通过输入数据参数提取各个因素的不同取值作为此因素的水平，根据各因素水平的加权值合理、适量的挑选出一定数量的因素水平，尽可能保证在因素水平挑选的准确性与覆盖性上要做足功夫，因为这关乎于软件测试用例设计的覆盖完整性与高效性。通过前述得出的数据构造出参数因素分析表并与正交表相匹配对应，生成系统所需测试用例。

2 系统可行性研究

2.1 编写目的与背景

随着互联网的普遍与电子社会的到来，市面上如雨后春笋般涌出来的软件的测试和维护问题越来越影响着这些软件的质量，那么如何高效准确的进行软件测试便成为了一个长期以来一直困扰着软件测试人员的问题。目前我们使用的测试方法大多都还是人力穷举测试用例，这样首先就是很难做到万无一失其次大大增加了软件测试成本。而当电子自动化时代与我们越来越近的时候，很容易就会想到能不能设计实现一个在软件测试过程中能够自动生成测试用例的系统，这样就可以尽可能的做到软件测试的准确性、高效性还有对成本的极致缩小。

而基于黑盒测试几个传统测试方法的用例自动生成系统对于现阶段我们软件测试工作的帮助已经渐渐式微，随着市面上的软件测试负荷越来越重，市场上也新兴出一些来源于传统却优于传统的优秀测试用例设计的方法原理。在本系统的设计实现中，除全组合与边界值测试用例设计生成之外还将加入开发基于正交实验法原理的测试用例设计与生成，而针对于正交实验法原理的说明与实现将在本文下述的功能实现部分进行具体阐述。

2.2 系统实现前提要求与目标

首先要了解有关黑盒测试的相关知识，阅读大量文献掌握黑盒测试的定义、原理和方法，查询阅读相关正交实验法原理定义和实现的论文文献与论坛进行深入了解学习。同时通过学习《软件工程导论》学习软件从无到有的全过程，清楚在软件生命周期内每个阶段应该做的工作，尽可能为以后的具体设计和实现系统做好基础准备。然后学习基于 C++语言程序开发的 QT 工具，了解各部件功能，熟悉开发流程，同时尝试设计系统核心算法并时刻修改以契合系统要求和开发工具限制。

根据黑盒测试的特性，黑盒测试用例自动生成系统无需考虑需要检测的程序内部代码和逻辑结构，在基于黑盒测试常用的等价类划分和边界值分析等方法原理下，由系统外部的用户进行接口参数输入，随后根据所设计实现的系统功能界面的具体功能键位，由系统判断响应并调用相关内部响应函数算法进行数据群的处理并最终输出对应该检测程序所自动生成的测试用例。同时，在系统处理结果输出界面对数据群所经历的相关功能处理予以说明并汇总，便于用户根据用例生成结果对所需测试的程序进行更强针对性的测试工作。

2.3 可行性分析

我们使用基于 C++语言的 QT 开发工具来搭建系统外部终端，同时使用信号槽的方式将相应的操作响应连接起来就可以达成系统的基本功能要求。同时利用 QT 的添加动作、添加资源等组件丰富系统包容性与多面性，使得系统功能更加完善。在现有计算机性能的限制下，系统的设计实现与运行基本不会出现内存超过负荷或者性能不够用的情况，而对于开发人员数量质量要求也不会特别高，有一定的项目

开发理论基础和程序设计经验就可以做出行之有效的系统。同时，由于系统目标明确功能凝练，在一定的时间限制内也是可以完成对于此项目系统的开发。而系统属于本科生毕业设计，严格来说属于学术成果并不用于任何商业途径因此不存在经济效益以及对经济收益的要求，对于经济可行方面完全可以不用深入研究，只需注意系统开发的周期和个人精力的投入尽可能将系统做到精益求精。本系统是作为本科生关于软件工程黑盒测试的毕业设计，没有签订任何法律合同文件，所以不存在任何合同责任。同时，系统设计实现成果属于学术成果，属于个人，并没有挪用他人成果，不存在侵犯专利权或者版权问题。

综上所述，从技术、经济、法律等方面可行性研究分析来看，系统可行，可以立即投入设计实现。

3 系统需求分析

3.1 需求分析目的

对系统进行需求分析是在对可行性研究分析的基础上，对系统的需求细节和实现方式进行更为详细的阐述。可行性研究分析旨在评估系统是否值得开发实现，意在确定系统能否设计实现。而需求分析旨在搞清楚系统要做到什么来保证后续开发的系统产品能够真正达到预想效果。其完整性与准确性直接关乎系统最终的走向，影响系统开发质量，因此在需求分析过程中需求的描述要尽可能的准确、无二义性、具体并且具有一定的现实可操作性，同时也要随着系统开发进度的推进对需求分析部分进行不断深入完善。

3.2 系统总体概述

3.2.1 目标

本系统的主要目的是开发一款基于软件测试中黑盒测试的用例自动生成系统。用户首先通过系统的登陆界面进行账号密码的录入以登录系统的内部功能界面，在系统内部读入用户所输入的账号密码并进行检验判断之后决定是否调用功能界面弹出函数以实现登陆界面到功能界面的跳转。

成功登录系统之后，用户可以根据自己所要进行测试的程序块，将相关的程序接口所能匹配接受的接口输入值的参数个数及其可取值域等相关数据群通过系统功能界面的输入端进行录入，录入完成后用户根据自己的需求进行主观选择基于不同生成方法的系统功能键位进行不同的功能生成，此时系统的内置键位响应函数将会根据输入的数据群进行分析处理，调用用户输入数据群预处理功能函数对系统所读入的用户输入数据群进行分析处理并得到能作为参数直接传到各响应功能函数内部之中而且可以被各响应功能函数识别处理的功能数据群，随后连接调用用户所选择的对应功能响应函数生成合适的测试用例群到输出端，同时在输出端对系统响应用户操作所执行调用并产生作用的相关功能处理情况予以说明并汇总，形成简要汇总报告之后一同打印在输出端，以便于用户后续的测试工作具有更强的针对性和高度的逻辑性。

根据系统的开发计划，系统目标主要包括以下几个方面：

（1）设计实现用户登录系统界面并做合理输入信息提示；

（2）设计实现简明的用户操作界面以及合理的输入端模块接口；

（3）设计安排合理的功能处理界面并保证功能选择的具体性和无二义性；

（4）实现生成结果的高度可视化和易理解化；

（5）实现系统功能的完善化和具体化。

3.2.2 运行环境

在 windows 系统上使用基于 C++语言开发的 QT 开发环境，对于硬件网络没有要求。QT5 环境下，多使用信号槽特性功能函数实现用户操作与系统内部算法的连接响应，同时添加功能与资源组件对用户选择功能操作实现可视化与具体化。

3.3 外部接口需求

追求尽量简洁的界面，争取直观的传递给用户尽量多的讯息，功能界面应该具体准确，不使用容易造成二义性的组件以增强系统工作的高效性。而对于系统从外部读取用户输入的输入模块接口，要通过系统使用者根据自己所测程序块的接口情况具体分析出输入测试参数个数及其可取值域等相关数据群再传到此系统中。因此要合理利用 QT 开发工具的相应功能组件配合系统内部运行算法实现系统对用户外部各种操作的准确获取以及快速响应。

3.4 系统结构图

针对于本系统的系统结构如图 1 所示：

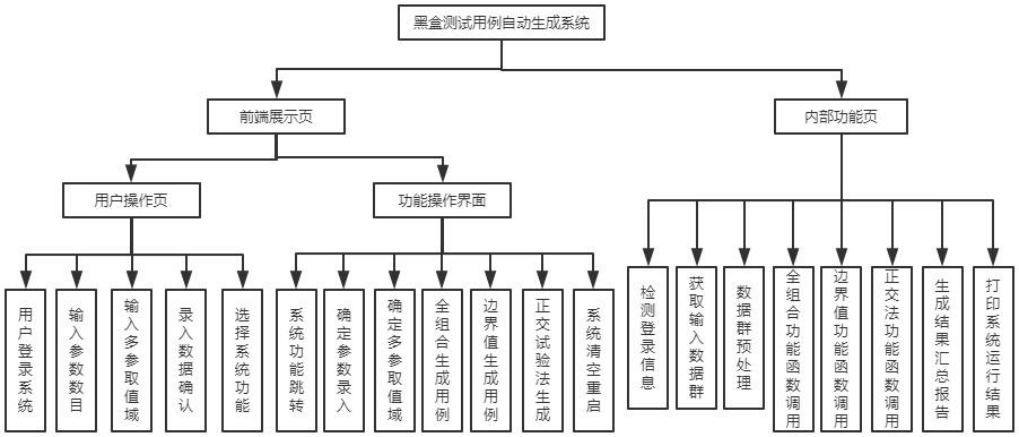

图 1 黑盒测试用例自动生成系统结构图

3.5 系统功能需求

针对于本系统的功能描述流程如图 2 所示：

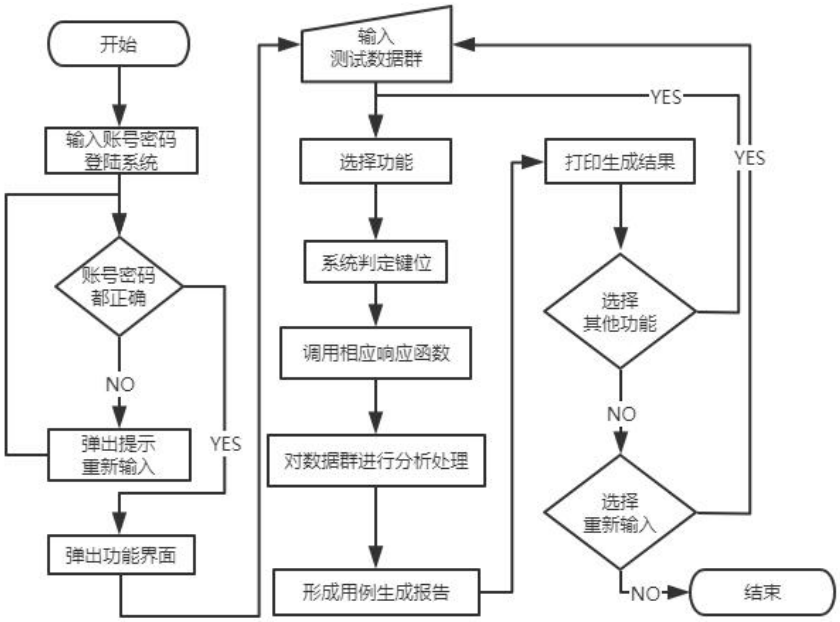

图 2 黑盒测试用例自动生成系统功能流程图

对上述系统功能流程图描述如下：

系统启动时需要用户输入正确的账号密码登录进本系统的核心功能界面，若出现账号或密码的任意错误系统均弹出相应提示框对用户进行输入提示。

用户登录进系统之后，在所弹出的功能界面要设置合理的输入输出模块以及系统各个功能的对应键位，同时对用户的输入进行相应提示。此时用户要根据自己所需测试的程序块特性选择适量、正确的测试参数在系统的输入模块进行录入，随后用户根据自身主观选择想要实现的功能键位。

系统在调用不同键位响应功能之前要先对用户录入的数据群进行分析预处理，得到系统功能函数可以识别并计算生成的功能数据群，而系统调用功能响应函数得到生成结果之后要对测试用例生成结果、系统响应用户操作所执行调用并产生作用的相关功能处理情况进行整理形成报告并打印在系统的输出模块中，此时用户还可选择是否使用其他功能或重新录入新的数据群。

4 系统设计与实现

4.1 界面

4.1.1 界面设计

本系统界面设计包括登陆系统界面和功能实现界面两部分。

对于登陆系统界面，设计在空白窗口界面添加两个文本框输入栏，用以输入进入系统所需要的账号与密码，同时输入栏下部添加一个登陆系统的按钮。设定两个输入栏在空白未输入任何数据的情况下，提示系统用户合适的输入规则。

相比于登陆界面，功能实现界面的设计便会稍微复杂但却全面一些：首先设置一个文本输入栏提供给用户进行所测参数个数的录入以实现多参数多因素的动态功能处理，随后设置一个大的文本输入块用以输入本系统运行处理所需要的数型数据群，并在输入模块上部添加相应的文字用以提示用户合乎规则的待处理数据群，同时在两处输入板块旁边的相应位置均添加两个确认按钮用以实现系统对数据的预处理及保存。接着在上述整体输入板块下面位置添加 4 个功能键位按钮，以此用以运行不同功能函数：黑盒测试用例全组合生成、黑盒测试用例健壮型边界值生成、黑盒测试用例基于正交实验法生成以及清空重启功能系统。最后在界面窗口右边整块设置一个结果输出栏，用以输出相应功能键的结果，同时在结果输出栏上面以文字表明模块组件的作用。

4.1.2 界面交互与实现

在基于 C++语言的 QT 程序开发工具中，可以很好的解决界面设计与交互问题。

首先在 QT 中创建一个 mainwindow 类作为系统的初始窗口用于作为系统的登陆界面，并对其 windowTitle 命名为“黑盒测试用例自动生成系统”。在登陆系统界面放置两个 QLineEdit 组件分别命名为 Line1 和 Line2，其分别作为输入账号与输入密码的组件，除了在两者 placeholderText 属性中添加相应对于用户还未进行任何输入的情况下的登陆系统的输入规则提示以外，同时在 Line2 的 echoMode 属性中选择 Password 实现隐藏密码的功能，并用鼠标在 mainwindow.ui 文件界面中拖动两个组件直到位置合适。然后在两个 QLineEdit 组件下面的合适位置添加 QPushButton 组件命名为登陆系统。

同时在 mainwindow.h 的 mainwindow 类声明中添加针对于登陆系统功能组件的 on_pushButton_1_clicked()响应函数用于实现界面登录和与功能实现界面的交互。即在此函数中创建一个新的 Dialog 类用以用户在点击登陆系统按钮的时候弹出一个新的对话框界面用于承载后续的功能实现界面，并同样对此界面的 windowTitle 命名为“黑盒测试用例自动生成系统”，针对此函数关于系统登录成功与否的判断与具体实现将在 4.2.2 中的功能定义及部分实现简述模块予以具体阐述，本部分不

做任何详细说明。

紧接前述，在所创建的 Dialog 中，进行功能界面的具体实现。首先直接进入 Dialog 类的界面文件 dialog.ui，添加 1 个新的 QLineEdit 组件作为功能实现界面的数据群参数数目输入栏并以鼠标拖动至合适位置，再于其下方添加 1 个新的 QTextEdit 组件作为数据群具体输入栏。同时在两个不同意义输入块周围的合适位置各添加 1 个 QPushButton 组件用以用户确定输入开放系统功能，并对上述不同类型数据的输入板块以 label 组件予以文字说明输入规则。

随后在整体输入板块的下端添加四个 QPushButton 组件分别命名为全组合生成、正交试验法生成、健壮型边界值生成和清空重启用以后续实现相应的功能处理。最后在前述输入板块以及功能板块组成的用户功能请求模块的右边，添加一个 QTextBrowser 组件用来作为整个系统的处理结束以后的结果输出栏并用鼠标拉伸至合适尺寸，同时在其上端同样添加一个 label 组件用来说明 QTextBrowser 组件的作用。

至此，关于此黑盒测试用例自动生成系统所有相关界面的设计实现与交互实现阐述完毕。具体实现系统的登录界面与功能界面分别如图 3、图 4 所示：

图 3 用户登录界面

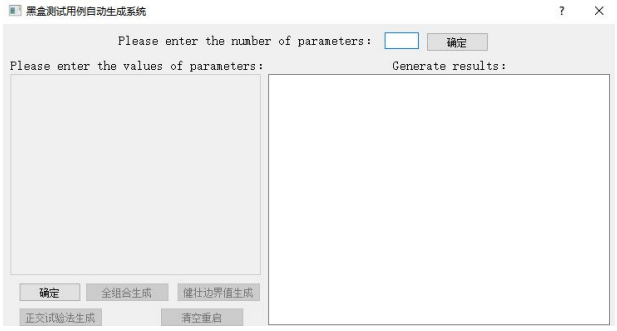

图 4 黑盒测试用例自动生成系统功能实现界面

4.2 功能

4.2.1 功能设计描述

基于前述的此黑盒测试用例自动生成系统的界面设计与交互，下面对关于此系统的所有功能设计说明如下。

首先，对于系统登录界面的具体响应功能分析出主要就是通过 Line1 和 Line2 中用户所输入的数据判断系统是否登陆成功，同时以弹窗形式弹出相应的处理结果对用户进行提示。

接着对于系统功能实现界面的相应功能相对而言更加具体更加完善：设置第一个功能作为预处理功能即通过以 QLineEdit 组件和 QTextEdit 组件组成的输入栏模块中的具体数据群，得到系统下一步功能处理所不可或缺的功能数据群；设置第二个功能用来实现对于前述功能界面的 4 个 QPushButton 组件其中之一的测试用例全组合生成功能；第三、第四个功能分别用来实现上述 5 个 QPushButton 组件中的健壮型边界值用例生成、基于正交试验法用例生成和清空重启功能。除此之外，考虑到程序的健壮性，还要实现在确定输入之前对所有的功能按键进行锁定关闭防止系统阻塞崩溃，在运行功能的时候也要对输入模块进行锁定关闭使其不能编辑，而此功能可以使用各组件指针指向 setEnabled(false)函数进行实现。

4.2.2 功能定义及部分实现简述

(1) 数据结构设计与定义：

① param <int>： 输入参数的个数；

② str_context <QString>： 接收 QTextEdit 所录入的具体多参数多因素的数据群；

③ vec <QList<QList<QString>>>：存放经分割算法处理后功能数据群；

④ new_list <QList<QString>>： 系统生成结果存放表；

⑤ ZJ <class>：自定义正交试验法的正交表存放结构体；

(2) 功能函数设计与定义：

① get_num()：用户输入数据群预处理函数（其中包含分割算法）；

② Blackboxtestcombine()：黑盒测试全组合用例生成功能函数；

③ Blackboxtestboundary()：黑盒测试健壮型边界值用例生成功能函数；

④ Blackboxtestorthogonal()：黑盒测试基于正交试验法的用例生成功能函数；

⑤ get()：多参数多因素全组合排列的递归调用算法；

⑥ Restart()：黑盒测试用力自动生成系统清空重启功能函数；

(3) 用户登录与界面交互实现：

前述部分 4.1.2 中提到了 mainwindow 类的声明中所添加的针对于登陆系统功能组件的 on_pushButton_1_clicked()响应函数，而对于其关于系统登录成功与否的判断与具体实现便是通过对于 mainwindow.ui 中所添加的两个 QLineEdit 组件中用户根据其提示的输入规则所输入的具体数据进行处理判断。其中包含了当账号为空、账号错误即不存在、密码为空、密码错误、账号正确与密码正确的六种判定条件的两两组合，即通过 if-else 语句对两个 QLineEdit 组件中用户所输入的数据进行判定。同时对于判定结果，则是通过 QMessageBox 头文件中的提示框组件予以相对应的响应提示，而当两个正确判定条件即账号正确、密码正确同时成立的时候，便如前文所述的创建一个 Dialog 对话框类，同时将 on_pushButton_1_clicked()函数作为信号槽机制中的信号发出，同时所创建的 Dialog 类作为信号接收者接收函数的信号之后立即调用 Dialog 类自身的针对于此函数信号的 on_pushButton_1_clicked()函数作为响应函数，这样便实现了由系统登录成功之后跳转功能处理界面的交互，而再每次判定之后，无论是否成功进入系统，均调用 QLineEdit 组件的 clear()函数对输入栏进行清空。

(4) 录入数据群读入数据分割算法（内含系统程序避阻塞代码实现）：

QString str_context：存放输入的多参多因素数据群；

QList<QString> temp_list：存放按参数分割后的多因素数据群；

QList<QString>temp2_list：存放按因素再次分割上述 temp_list 的数据群；

QList<QList<QString>> vec：存放分割完成后得到的功能数据群；

算法具体实现如图 5 所示：

```
void Dialog::on_paramB_clicked()
{
    QString str = ui->lineEdit->text();

    if(str == NULL)
    {
        QMessageBox::warning(this, tr("输入错误"), tr("参数不能为空"));
        return;
    }
    param = ui->lineEdit->text().toInt();//健壮性

    cout << param << endl;
    QMessageBox::information(this, tr("黑盒测试用例自动生成系统"), tr("参数数目录入成功，请继续！"));

    ui->textEdit->setEnabled(true);
    ui->paramB->setEnabled(false);
}

void Dialog::on_textButton_clicked()
{
    str_context = ui->textEdit->toPlainText();
    if(str_context == NULL)
    {
        QMessageBox::warning(this, tr("输入错误"), tr("文本不能为空!"));
        return;
    }
    QList<QString> temp_list = str_context.split("\n");//健壮性

    for(int j=0; j<temp_list.size(); j++)
    {
      QList<QString> temp2_list = temp_list[j].remove(QRegExp("\\s")).split(",",QString::SkipEmptyParts);
      vec.append(temp2_list);
    }
}
```

图 5 黑盒测试用例自动生成系统-录入数据群读入数据分割算法代码实现

(5) 多参多因素全排递归调用算法：

int i：对参数数目 param 越界判定值；

QString str：暂存每次递归生成的排列项；

QList<QList<QString>> list：经分割算法处理过后的功能数据群；

int param：用户输入的参数数目；

int *n：生成排列项计数；

算法具体实现如图 6 所示：

```
void Dialog::get(int i,QString str,QList<QList<QString>> list,int param,int *n)
{
    if(i == param)
    {
        //qDebug()<<str<<endl;
        this->new_list.append(str);
        str = "";
        *n = *n+1;
        return ;
    }
    for(int k=0; k<list[i].size();k++)
    {
        QString temp = str;
        str+= list[i].at(k) + " ";

        get(i+1,str,list,param,n);

        str= temp;
    }
}
```

图 6 黑盒测试用例自动生成系统-多参多因素全排递归调用算法代码实现

(6) 自定义结构体存放正交表实现代码如下：

int zj_k：对用户输入参数数目的存放；

int zj_n：用户输入数据群由正交实验法生成用例的个数存放；

int zj_m：用户批量输入的参数取值域数目存放；

int **zj_chart：正交试验法正交表的存放；

具体实现代码如图 7 所示：

```
class ZJ{
public:
    int zj_m;  //取值个数
    int zj_n;  //用例个数
    int zj_k;  //参数个数
    int **zj_chart;

    ZJ(int k,int m,int n,int **chart){
    zj_m = m;
    zj_n = n;
    zj_k = k;

    zj_chart = new int*[zj_n];

    for(int i=0; i<zj_n; i++)
    {
        zj_chart[i] = new int[zj_k];
    }
    for(int i=0; i<zj_n; i++)
    {
        for(int j=0; j<zj_k; j++)
        {
            zj_chart[i][j] = chart[i][j];
        }
    }

    }

};
```

图 7 黑盒测试用例自动生成系统-自定义结构体存放正交表实现代码

（7）边界值生成用例代码实现如下（含数型类型判定算法）：

bool VerNumber(QString str)：对用户输入数据群是否含数型类型的判定函数；

QString str：作为判定函数的传参字符串；

QList<QList<QString>> new_vec：复制数据预处理功能后所得功能数据群再判定出数型类型后将其边界值加入新的容器 QList<QList<QString>> new_vec 中存放；

int *n：边界值生成用例项计数；

此功能实现时，经数据处理后调用上述的递归算法进行用例因素组合；

具体实现代码如图 8 所示：

```
bool VerNumber(QString str)
{
    std::string temp = str.toStdString();
    for (int i = 0; i < str.length(); i++)
    {
        if (temp[i]<'0' || temp[i]>'9')
        {
            return false;
        }
    }
    return true;
}
Blackboxtestboundary()
{
    ui->textBrowser->setText("");
    QList<QList<QString>> new_vec = vec2;
    for(int i=0; i<vec2.size(); i++)
    {
        if(VerNumber(vec2[i].at(0)))
        {
            for(int k=0; k<vec2[i].size(); k++)
            {
                new_vec[i].append(QString::number(vec2[i].at(k).toDouble()+1));
                new_vec[i].append(QString::number(vec2[i].at(k).toDouble()-1));
            }
            //ui->textBrowser->append("true");
        }
    }
QString str = "";
int *n=new int;
*n = 0;
new_list.clear();
get(0,str,new_vec,param,n);
}
```

图 8 黑盒测试用例自动生成系统-边界值生成用例代码实现

（8）多参多因素输入匹配正交表生成用例代码实现如下：

int k：存放系统读入用户输入参数数目;

bool flag：各参数取值域数目判定；

int **chart：匹配正交表存放；

具体实现代码如图 9、图 10 所示：

```
Blackboxtestorthogonal()
{
    int k = vec2.size();//参数
    int countm = vec2[0].size();
    bool flag = false;
    QList<int> num_size;

    for(int i=1; i<k; i++)
    {
       if(countm != vec2[i].size())
       {
           flag = true;
           break;
       }
    }
    if(flag)
        {
            int m = vec2[0].size();
            int n = k*(m-1)+1;
            for(int i=0; i<M.size(); i++)
            {
                for(int j=0;j<k;j++)
                {
                    if(num_size[j] == M[i])//
                    {
                        if(chart[i][j]<M[i])//
                        tr += vec2[j].at(chart[i][j])+" ";
                        else str += vec2[j].at(0)+" ";
                    }
                }
            }
       }
```

图 9 多参多因素输入匹配正交表生成用例代码实现(1)

```
    else
    {
        if((k==K) && (m<=M))
        {
            for(int i=0; i<n; i++)
            {
                QString str = "";
                for(int j=0; j<k; j++)
                {
                 if(chart[i][j]>=m)
                 {
                     str += vec2[j].at(0)+" ";
                 }
                 else{
                 str += vec2[j].at(a1_chart[i][j])+" ";
                 }
                }
                ui->textBrowser->append(str);

            }
        }
    }
}
```

图 10 多参多因素输入匹配正交表生成用例代码实现(2)

5 系统运行测试

5.1 测试方法与测试环境

自己作为使用系统的用户体验本系统。

在装有 windows 系统的 PC 端电脑上使用 QT 工具进行系统运行测试。

5.2 测试用例

表 1-黑盒测试用例自动生成系统运行测试用例

测试序号	前置条件	实际操作	预期结果	测试结果
1	用户登录系统	输入空账号或错误账号	提示登陆失败	通过
2	用户登录系统	输入空密码或错误密码	提示登录失败	通过
3	用户登录系统	输入正确账号密码	登录成功跳转功能界面	通过
4	用户输入数据	输入空参数点击确定	系统提示报错	通过
5	用户输入数据	输入非空参数点击确定	解除文本编辑区锁定	通过
6	用户输入数据	输入空参数群并确定	系统提示报错	通过
7	用户输入数据	输入非空参数群并确定	解除功能键位锁定 再次锁定文本编辑区	通过
8	用户选择功能	点击全组合生成键	输出区打印生成报告	通过
9	用户选择功能	点击边界值生成键	输出区打印生成报告	通过
10	用户选择功能	点击正交试验法生成键	输出区打印生成报告	通过
11	用户退出系统	点击清空重启键	关闭功能面跳转登录面	通过

5.3 全组合生成测试

成功登陆系统之后进行以下数据输入并生成全组合结果如图 11 所示：

注：由于程序添加 setEnabled(false)函数，因此在用户点击确定键后会将输入栏关闭以实现程序运行时输入数据文本不可编辑，因此下述所有截图中输入栏呈灰色态。

参数数目 param：3；

参数一取值：阿里，腾讯，百度；

参数二取值：A,B,C,D；

参数三取值：1.2，3，4，5；

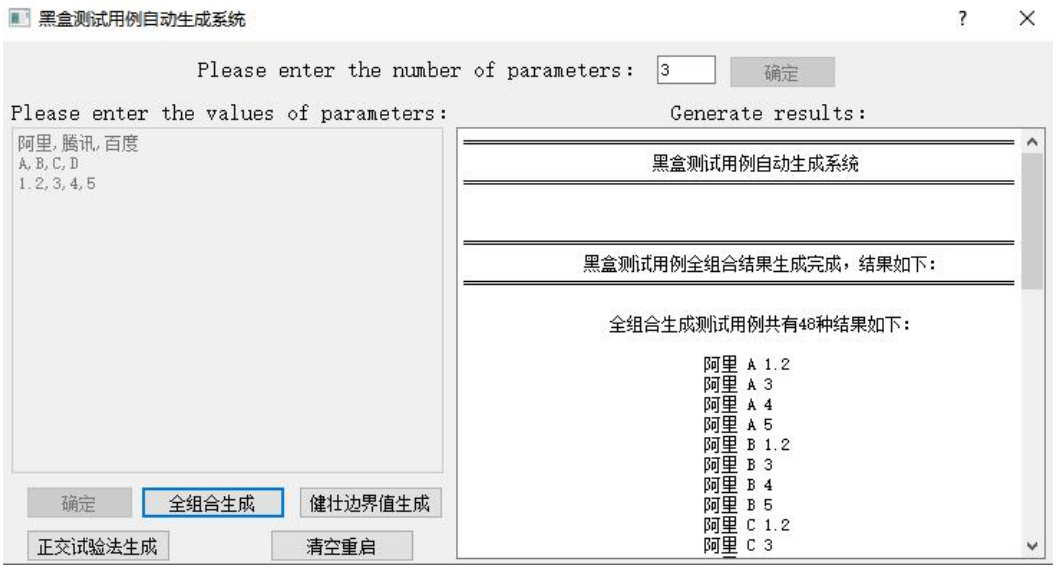

图 11 黑盒测试用例自动生成系统运行实例-全组合生成（部分用例）

5.4 健壮型边界值生成测试

成功登陆系统之后进行以下数据输入并生成边界值结果如图 12 所示：

参数数目 param：4；

参数一取值：AA,BB,CC；

参数二取值：11,22,33,44；

参数三取值：a,b；

参数四取值：苹果,华为,三星；

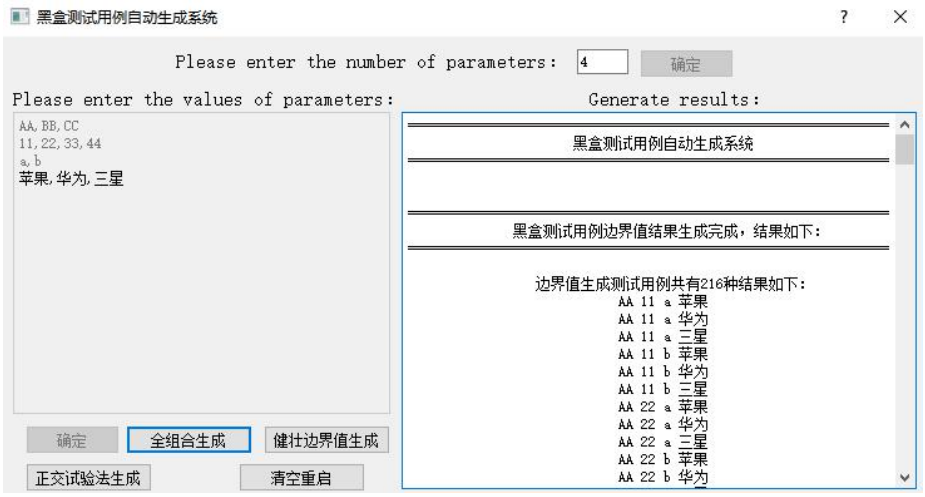

图 12 黑盒测试用例自动生成系统运行实例-边界值生成（部分用例）

5.5 正交试验法生成测试

成功登陆系统之后进行以下数据输入并生成基于正交试验法结果如图 13 所示：

参数数目 param：8；

参数一取值：11,22；

参数二取值：aa,cc；

参数三取值：AA,bb；

参数四取值：鼠标,键盘；

参数五取值：P,I,L；

参数六取值：1,2,3,4,6,8；

参数七取值：9,8,7,5,1,6；

参数八取值：1,j,y,b,f,g；

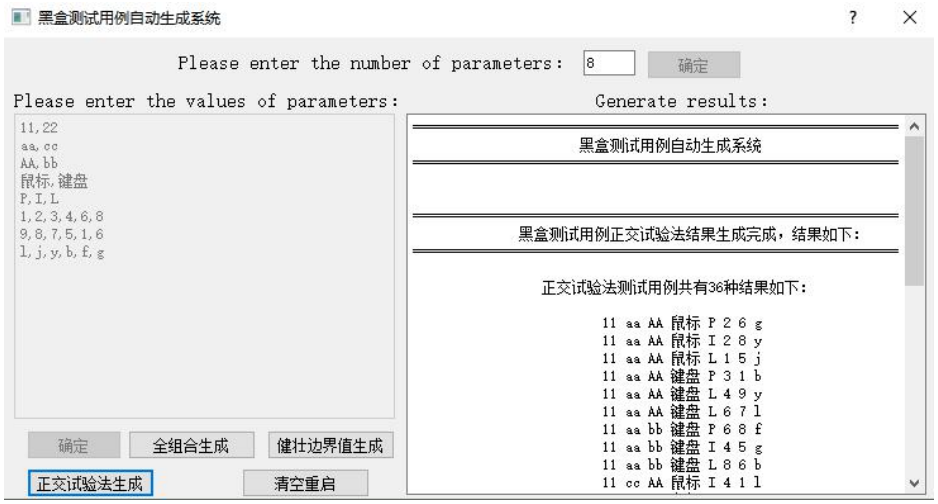

图 13 黑盒测试用例自动生成系统运行实例-正交试验法生成（部分用例）

至此，对本黑盒测试用例自动生成系统的所有功能具体实现及测试均以说明完毕。

结束语

本系统中，实现了黑盒测试的几种传统用例设计生成方法，并解决实现了多参数多因素录入数据的动态处理。除此之外，还开发加入了基于正交试验法的用例设计生成方法，深切体会到这个原理在软件测试用例设计生成工作中极大的优势以及对于软件测试人员工作的极大辅助性与便利性。而在极简化设计用例的同时也反思现如今软件质量测试的重要性已经是重中之重了。但在本系统中的所实现的正交试验法仍存在例如正交表不够完整以及算法兼容性不够高等不足，而系统中在对于多参多因素全排递归算法的与分割读入算法的健壮性比较可观，是本系统基于传统的创新点所在。

同时，我体会到了一个项目系统从无到有的不易与困难，哪怕是一个再简单的项目，也应该要有足够的重视度去实施与执行。拿这次毕业设计来说，我在一开始便敲定了开发工具与开发语言，这样我的项目进度便就变得针对性特别强，一切系统的构建都会显得自然而然。而如果我一开始就选错了开发工具，那么等待我的就是毫无头绪的焦虑。选择 QT 开发工具是我在综合自己的知识储备以及项目经验之下选定的，接着对于此工具的开发教程和运行机制进行系统学习之后便开始投入系统开发。但是一个项目开发的过程肯定是坎坷不断的，无数的组件与函数类的未知查询花了我很多的时间去适应学习，在不断的尝试不断的系统崩溃之后，我深刻明白了项目开发的细节问题是何其重要，而在系统实现功能中所加入的正交试验法更是需要投入大量的精力去学习理解掌握，这对于我个人的专业水准的提升是显而易见的，而尝试开发过程本身便就是一个不断出错不断改错努力提升自我的过程吧。

而在本次毕业设计的实现前期，由于我个人疏忽并未跟导师即使沟通项目细节和进度，导致项目开发浪费了很多时间与精力。在此，我向导师林岭老师致以歉意，并诚挚感谢导师对我的耐心教导！

参考文献

[1]DeMillo R. Experimental results from an automatic test case generator[J]. ACM Transactions on Software Engineering Methodology, 1993 ,2(2):109 - 175.

[2]DeMillo R. Constraint based automatic test data generation[J]. IEEE Transactions on Software Engineering, 1991, 17(9):900- 910.
[3]孙淑香,侯秀萍,蔡文涛. 基于树型模型的黑盒测试用例自动生成[J]. 计算机技术与发展,2009(2):77-80.
[4]梁金宏，刘威，赵利民，廉诗阳. 软件系统黑盒测试方法研究[J]. 电子世界，2016(22):139-140.
[5]余慧敏,徐白,周楷林,李芳芳. 动态软件测试中的白盒测试和黑盒测试探讨[J]. 电子测试,2018(8):59-59.
[6]龚昌.浅谈白盒测试与黑盒测试在软件测试中的应用[J].信息与电脑.2011(1).
[7]杨海成. 航天型号软件工程[M]. 中国宇航出版社，2011: 100-102.
[8]张小松，王钰，曹跃.软件测试[M].机械工业出版社,2014:35-37.
[9]Myers GJ, Badgett T, Thomas T M, et al. The Art of Soft-ware Testing[M].王峰,陈杰译.北京 :机械工业出版社, 2006.
[10]Chen T Y. A new heuristic for test suite reduction[J]. Information and Software Technology, 1998,40(5 - 6): 347-354.
[11]陈磊,简炜. 计算机实现基于正交试验的测试用例自动生成[J]. 信息安全与技术,2011,5.
[12]孟祥丰. 软件的黑盒测试方法解析与优化[J]. 物联网技术. 2011(8): 69-71.
[13]董晓霞. 相邻因素组合测试用例集的最优生成方法[J]. 计算机学报，2007，30(2)：200-210.
[14]陈策,王凤琴,赵绍才. 正交试验及其在软件测试用例设计中的应用[J]. 计算机应用与软件,2008,25:158-160.
[15]吴翊等. 应用数理统计[M]. 国防科技大学出版社,1995.
[16]徐仲安等. 正交试验设计法简介[M]. 科技情报开发与经济,2002,12(5):148-150.

The Design and Implement of Black Box Testing Case Automatic Generation System

Hu Jinyuan 123012016049 Advisor: Lin Ling
Major in Software Engineering College of Mathematics and Informatics

【Abstract】Software testing is a crucial step in the software life cycle, especially the automation of test case generation can greatly improve test efficiency and reduce test cost. Black Box Testing is one of the most important testing methods in software testing, its use case design generation is the most important. Using QT development tool based on c + + language, the boundary value and the equivalence class division, under the guidance of the principle of using custom structure model design to achieve the automatic generating system of black box test case design to realize multi-parameter multi-factor all combination permutation algorithm of recursive calls, and joined the generated on the basis of the principle of orthogonal test the function of the use case. The whole process of the system from definition to design to implementation is reproduced.

【 Keywords 】 Software Testing; Black Box Testing; Automatically Generate Test Cases; QT Development; Orthogonal Test