目 录

代码检查

作者：薛丽 崔烨

摘 要： 代码检查是白盒测试的一种静态测试方法，是众多软件测试方法中发现软件缺陷最有效的方法之一。本文结合国内外学者在相关领域的研究情况，介绍代码检查相关的基本概念、过程和分析方法。

关键字：白盒测试，代码检查，静态分析，检查规则

一、 引 言

按照测试时源代码是否可见，软件测试可以分为白盒测试和黑盒测试两类。

白盒测试（结构测试），即逻辑驱动的测试，是在了解程序内部结构的基础上，对程序的逻辑结构进行检查，从中获取测试数据。白盒测试关注的是测试用例执行的程度或覆盖程序逻辑结构的程度。白盒测试一般只应用于软件开发阶段。

白盒测试，又可按照是否需要运行程序，进一步细分为了静态测试和动态测试两种。通常情况下是按照先静态后动态测试顺序来实施。其中，静态测试包括代码检查、静态结构分析、代码质量度量等测试内容。静态测试既可以由人工进行，充分发挥人的逻辑思维优势，也可以借助软件工具自动进行。

代码检查是一种对程序代码进行静态检查。传统的代码检查是通过人工阅读代码的方式，检查软件设计的正确性；用人脑模拟程序在计算机中的运行，仔细推敲、校验和核实程序每一步的执行结果，进而判断其执行逻辑、控制模型、算法和使用参数与数据的正确性。

在实践中，代码检查比动态测试更有效率，能找到更多的缺陷，通常能发现 30%～70%的逻辑设计和编码缺陷。代码检查非常耗费时间，而且需要专业知识和经验的积累。代码检查定位在编译之后和动态测试之前进行，在检查前，应准备好需求描述文档、程序设计文档、程序的源代码清单、代码编码标准和代码缺陷检查表等。

代码检查可以发现的软件问题包括：声明或引用错误、函数/方法参数错误、语句不可达错误、数组越界错误、控制流错误、界面错误和输入/输出错误等。

二、 代码检查的基本概念

1、代码检查

代码检查包括桌面检查、代码走查和代码审查等方式，主要检查代码和设计的一致性，代码对标准地遵循、可读性，代码逻辑表达的正确性，代码结构的合理性等方面；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、命名和类型检查、程序逻辑检查、程序语法检查和程序结构检查等内容。

下面对代码检查的三种具体方式进行介绍。

- 桌面检查

是一种传统的检查方法，由程序员检查自己编写的程序。程序员在程序通过编译之后对源代码代码进行分析、检验，并补充相关的文档，目的是发现程序中的错误。

- 代码走查

代码走查就是针对代码，在假想的输入情况下，逐行的浏览代码，走查代码中潜在的缺陷并记录结果的过程。

代码走查以小组会议方式进行，每小组 3-5 人。与代码审查不同的是，走查要求与会者扮演计算机的角色让测试用例沿被测程序的逻辑运行，是在模拟动态测试；而代码审查更多的是静态测试。

- 代码审查

代码审查是由一组人通过阅读、讨论和争议对程序进行静态分析的过程，以小组会的方式进行。

审查小组一般由若干程序员（包括程序代码的设计者）和代码检查人员组成。会前把设计规格说明书、控制流程图、程序文本以及要求、规范、错误检查清单交给与会者，开会时程序作者朗读解释程序，其他人则集中精力，捕捉程序在结构、功能、编码风格等方面的问题。

2、代码检查项

代码检查项即检查代码时，指定需要进行检查的内容。具体如：检查变量的交叉引用表；检查标号的交叉引用表；检查子程序、宏、函数；等价性检查；标准检查；风格检查；选择、激活路径；对照程序的规格说明，详细阅读代码，逐字逐句分析；补充文档。

检查项可以作为依据，用来编制代码规则、规范和缺陷检查表等。

3、编码规范

编码规范是程序编写过程中必须遵循的一套事先约定或者已经制度化、标准化的规则集，一般会详细的规定代码的语法规则和语法格式。

一个良好的编码规范能够带来许多好处：改善代码质量；提高开发进度；增进团队精神。对于软件开发而言，采用好的编程规范，虽然不能彻底杜绝糟糕的代码产生。但对于代码检查和将来的代码维护，仍然是意义重大的。

4、缺陷检查表

在进行人工代码检查时，使用代码缺陷检查表作为代码检查的参考依据。在软件测试项目实践中代码缺陷检查表又常被称作代码检查清单。

代码缺陷检查表中一般包括开发人员容易出错的地方和在以往的工作中遇到的典型错误。对应于不同的编程语言，代码缺陷检查表的具体内容将会有所不同。例如：对于 C/C++语言代码缺陷检查表内容有以下几部分：文件结构；文件的版式；命名规则；表达式与基本语句；常量；函数设计；内存管理；C++函数的高级特性；类的构造函数、析构函数和赋值函数；类的高级特性；其他的常见问题等。

5、代码检查规则

在代码检查中，需要依据被测软件的特点，选用适当的标准与规范。在使用测试软件进行自动化代码检查或辅助代码检查时，测试工具需要内置许多编码规范。不同编程语言，对应的检查规范有所不同。针对与 C/C++语言的规则有以下几类规则：通用规则、C++编码规则、C 编码规则、Meyers-Klaus 规则以及自定义规则。使用时，需要根据编程语言和被测程序的特点，选择适当的规则进行检查。

6、静态分析

静态分析是不执行程序，而分析程序代码的过程。源代码被静态分析器分析之后，得到的静态分析结果，通常可以表示成一棵静态语法树。其中包含了被测项目源代码的静态结构信息：基本代码成分、程序结构、语句结构、类型和模板等信息。

程序代码静态分析的结果能够给代码检查提供帮助。

三、 代码检查过程

传统的代码检查是一种静态检查程序的测试方法，通常以团队的形式来进行。检查团队由程序作者，一个负责人，一个记录员以及一些检查员组成。首先需要一系列的准备工作，包括参与者的挑选和材料的准备。然后是个人准备阶段，每个小组成员各自熟悉材料。个人准备阶段后，就是实际的检查会议。在会议上，检查小组在假想的输入下，由程序作者带领，逐行的浏览代码，评审代码中潜在的缺陷。检查小组根据发现缺陷的严重程度和类型对其进行分类，并将问题记录下来供作者修正。会议后是作者的返工，作者汇报每个缺陷，最后确认每个缺陷已经被陈述过了。图 11 为传统的代码检查过程。

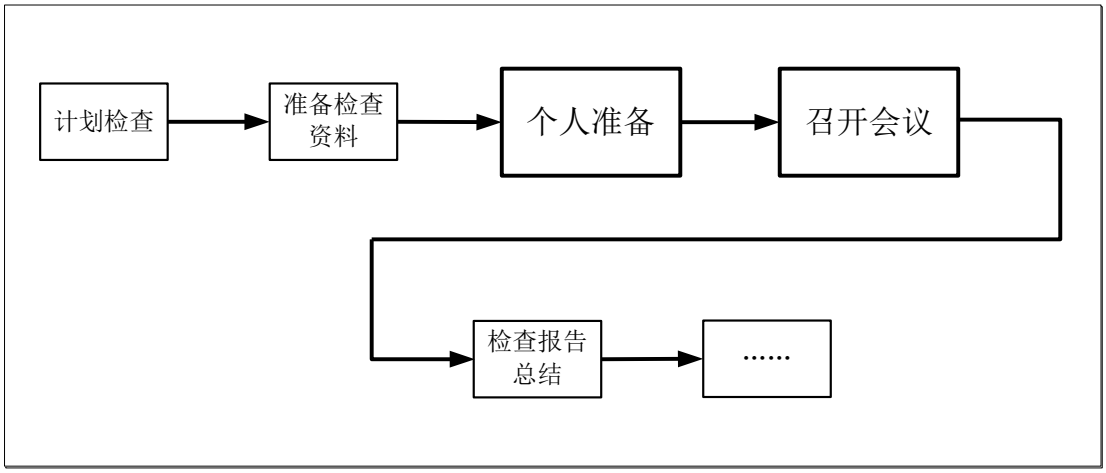


图 1 代码检查过程示意图

代码检查过程中的两个重要阶段“个人准备”和“召开会议”阶段有以下注意事项：

1、“个人准备”阶段：

会前准备阶段是检查过程的一个关键阶段，因为如果检查者没有为检查做好充分的准备，检查效果会大打折扣。如果有检查人员没有做好准备，主审员可取消其代码检查资格，甚至取消这次检查会议。

检查人员要熟悉检查内容的相关文档，了解程序背景、设计思想和编程方法，在读懂、“吃”透代码的基础上，查出尽可能多的错误。

2、“召开会议”阶段：

参与会议的检查者应具有一定的专业技能和经验，缺乏经验的检查人员必然缺乏合适的领域知识来深入理解材料；

参与会议的检查者应做充分的个人准备，没有做充分准备的检查人员不能在检查会议中做出实质性的贡献；

检查会议的速度应进行控制，如果试图在短时间内处理太多的材料，检查效果也会大打折扣。现在较为常见的代码检查速度上的建议为：汇编代码 150 行/小时，C 语言 150 行/小时，而对于 C++、Java 这种面向对象语言，代码检查速度可以提高到 200-300 行/小时。

由此可见，代码检查适合于采用工具辅助的特性有：文档处理，个人准备，会议支持，数据收集。

- 文档处理

这是工具可支持的最明显的领域。传统的检查要求分发每份文档的复印件等，而将纸质的文档替换成计算机式的文档，不只是简单的介质变更，更是提供了一种契机——提高文档的可用性和表示性的机遇。

- 个人准备

首先，自动的缺陷检测可以用来发现简单的缺陷。如果简单问题能被自动发现，检查员就能专注于更加复杂/困难的缺陷，以及那些不能被自动发现的、潜在的、可能带来更大影响的问题。另外，自动化工具应该对个人准备阶段提供更多的帮助。例如，检查员可以利用检查表以及其它支持文档，并能很容易地交叉引用它们；还有些代码辅助理解工具，可为检查员理解程序、了解程序结构提供帮助。

- 会议支持

一些成员由于某些原因，可能没有花费足够的时间来进行准备，但他们仍然参加会议并试图掩盖他们的过失。项目管理人员可以使用计算机监控的个人准备时间信息，来剔除那些没有做好个人准备的成员，或者督促他们投入更多的努力。

召开会议时，检查员通常面对的是一堆枯燥的程序代码，如果在代码之外再结合一些图、表等便于分析、理解代码的信息，相信检查会议可以进行得更加有序和高效。

- 数据收集

代码检查一个重要的部分就是度量信息的收集，用来提供反馈以改进检查过程。度量信息包括会议时间、发现的缺陷、检查花费的总时间等。根据这些数据，可以来评价每一次代码审查的质量，进而给出关于代码审查的改进建议。

通过对检查过程的部分阶段提供计算机支持，代码检查可以进行得更加有效。使用计算机来支持检查过程，可以提高效率，并增加检查过程的严格性。

四、 代码检查历史数据

代码检查中的历史数据本质是软件问题（缺陷）。按照不同的代码检查角度，存在多种对缺陷分类的方法。对过往发现的软件问题进行分析，总结出今后对于类似的代码需要按照某种规则来加以检查，这种的规则就是检查清单上的一条清单项，代码检查清单就是大量规则的集合。此外，由于软件问题总是以软件问题报告为载体形式出现，因此软件问题报告也被通俗的理解为代码检查历史数据。

下面对缺陷分类、代码检查清单和软件问题报告加以研究。

1、缺陷分类

关于缺陷分类存在以下几种常见的划分方式：

1）按缺陷出现的区域分类

这种分类方式是最常见的缺陷分类方式。按照出现区域将代码缺陷划分为变量级、属性级、函数/方法级和类级缺陷。其中，变量级、属性级和部分函数/方法级的缺陷，与传统的面向过程编程中的缺陷分类基本一致；而多数方法级缺陷和类级缺陷，则是针对面向对象技术编程特点提出的。

2）按检测内容分类

分为冲突、一致性问题两种。

冲突对应于文献[1]中的基于确定性“信念”的判定，而一致性问题则对应于基于可能性“信念”的判定。

3）按对代码的危害分类

按照对代码的危害，一般分为浪费时间和空间；语义混淆；暴露封装性，扩大使用权限；程序一致性问题；程序约束条件问题和空指针问题等。

2、代码检查清单（Checklist）

代码检查过程中，代码检查人员都会有一份代码检查清单。代码检查清单是一份为代码检查人员准备的缺陷检查表，检查表中开列所有可能与代码有关的缺陷，并注明了检查的内容、缺陷类型以及严重性。检查清单是检查代码的依据，代码检查人员根据它来发现并判断问题。

代码检查清单中会逐条列出所有应该检查的缺陷种类，以及每条缺陷的各种特征，并且根据缺陷的严重程度和类型对其进行分类。通常每一条缺陷的特征描述如下：

1）缺陷描述：该缺陷的问题描述、举例说明，以及相应的正确形式；

2）缺陷出现的区域：分别为表达式级、语句级、声明级、模板缺陷、预处理缺陷、类级缺陷以及性能缺陷。表达式级、语句级、声明级以及预处理的缺陷，主要面向过程程序中的缺陷；模板缺陷、类级缺陷，则是针对面向对象软件的特点提出的；代码冗余等归为性能缺陷；

3）缺陷对代码的危害：代码中出现某种缺陷将会造成什么样的影响。

例如，检查表中一条缺陷的特征描述如下：

问题描述：指针所指内存释放后没有将指针赋为 NULL。

举例说明：

```
char *p=(char *) malloc(100) ;
strcpy(p, "hello");
free(p); //p 所指的内存被释放，但是 p 所指的地址还是不变
…
if(p!=NULL)  //没有起到防错的作用
{
   strcpy(p, "world"); //出错
}
```

正确形式：在释放内存的同时将指针置空。

```
char *p=(char *) malloc(100) ;
strcpy(p, "hello");
free(p);
p=NULL;  //增加指针置空语句
…
if(p!=NULL)
{
   strcpy(p, "world");
}
```

出现区域：语句级。

危害：指针被 free 释放后其地址并不会自动发生改变（非 NULL），p 成为了“野”指

针，这种情况下再对 p 进行操作，很容易造成程序崩溃，后果非常严重。

而代码检查清单正是由若干条这样的缺陷特征描述构成的。

3、软件问题报告（Software Problem Report）

在软件测试过程中，对于发现的每个软件问题（缺陷），都要进行记录该错误的特征和再现步骤等信息，以便相关人员分析和处理软件问题。为了管理测试发现的软件问题，通常要采用软件问题报告数据库，将每一个发现的软件问题输入到软件问题报告数据库中，软件问题报告数据库的每一条记录称为一个软件问题报告。

软件问题报告包括头信息、简述、操作步骤和注释。

头信息包括：被测试软件名称、版本号、缺陷或错误类型、可重复性、测试平台、平台语言、缺陷或错误范围。并要求填写完整和准确。

简述是对缺陷或错误特征的简单描述，可以使用短语或短句，要求简练和准确。

操作步骤是描述该缺陷或错误出现的操作顺序，要求完整、简洁和准确。对命令、系统变量、选项要用大写字母，对控件名称等要加双引号。

注释一般是对缺陷或错误的附加描述，一般包括缺陷或错误现象的图像，包括其他建议或注释文字。

软件问题报告是软件测试过程中最重要的文档之一。它记录了软件问题发生的环境，软件问题的再现步骤以及性质的说明，而且还可以跟踪软件问题的处理过程和状态。软件问题的处理进程从一定角度反映了测试的进程和被测软件的质量状况及改善过程。

五、 代码检查规则管理的研究

1、潜在的编码规则和缺陷代码模式

潜在的编码规则（Implicit Coding Rules）和缺陷代码模式（Bug Code Pattern）是 Tomoko MATSUMURA 在文献[3,4]中针对代码检查实践，提出的两个相关的概念。

潜在的编码规则

潜在的编码规则包含以下几个特征：

1）不同于在开发启动时明确决定的“编码规范”的规则，这些规则在长期的测试/维护过程中是潜伏的，对这些规则的发现是不可预见的。

2）这些规则很少在设计文档或者特定的文档中被清楚的描述。他们通常只存在于开发人员、测试/维护人员的记忆中。换言之，是一种尚未系统化的经验积累和总结的结果。

3）不同于使用规范库的公用规则。对于特定的软件有其特定的规则，这也意味着对

于不同的软件有不同的潜在的编码规则。

4）由于违反潜在的编码规则导致的缺陷通常情况下不是那么容易发现的。其中相当多一部分只在特定的罕见的情况下发生，所以在早期要想发现这些问题是很困难的。

5）目前，还不存在好的工具或者检查清单来发现违反潜在的编码规则的代码片段，通常的检查工具（例如 PC-Lint、Purify）和通用的检查清单只能发现常见的问题。

6）为了减少违反潜在的编码规则的现象的发生，而进行重构通常很困难。要重构一个软件，准确理解代码是非常必要的，然而，老的系统太复杂，并且没有精确的文档和了解系统的专业维护人员。总之，重构过期系统的代价很大，需要冒很大的风险。

缺陷代码模式：违反潜在的编码规则的编码模式。

缺陷代码模式不是肯定会导致缺陷的发生，一段符合缺陷代码模式的代码片段，并不意味着代码片段一定就有缺陷，缺陷代码模式只是疑似存在缺陷。另一方面，因为缺陷代码模式是静态的，没有考虑到代码片段之间的动态关联。需要代码检查人员或者维护人员把符合缺陷代码模式的代码片段提出来，并判断究竟是否存在缺陷。

在软件开发过程中发现和建立缺陷代码模式有三条主要途径。其一：在进行代码检查过程中，代码检查人员发现一个软件问题的同时，根据对该问题是否具备代表性和通用性等因素的考虑，确定是否建立一个缺陷代码模式；其二：当软件失效或者发生问题，检查对应的代码部分，发现并确定是否有潜在的编码规范与之相关；其三：分析现存的代码规范和积累的大量问题报告，从中提炼出潜在的编码规则。

在文献[3,4]中还给我们介绍了一个代码缺陷检测系统的大致工作流程，如 2 所示。

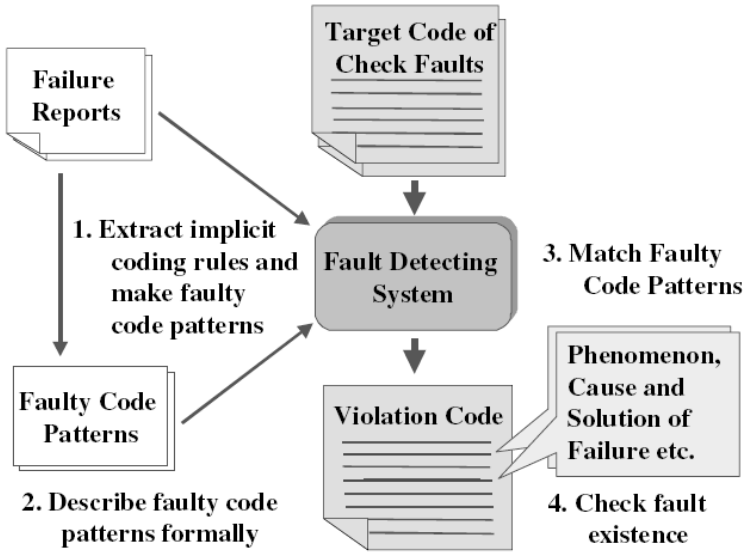


图 2　缺陷检测模型系统的代码检查流程参考图

2、C++代码检查规则类型

1）规则层次

在代码检查工作中常常可以发现这样的现象：有些规则能在所有的项目中都能发现问题，另一些规则所能发现的问题只存在于某类项目中。

根据规则的这个特点，如图 33 中所示，参考文献[2]中将代码检查规则分为两个层次：

公共规则（General checks）：用于检查在大多数情况都有可能发生的缺陷。

项目相关规则（Project specific checks）：用于在项目中检查可能的缺陷。

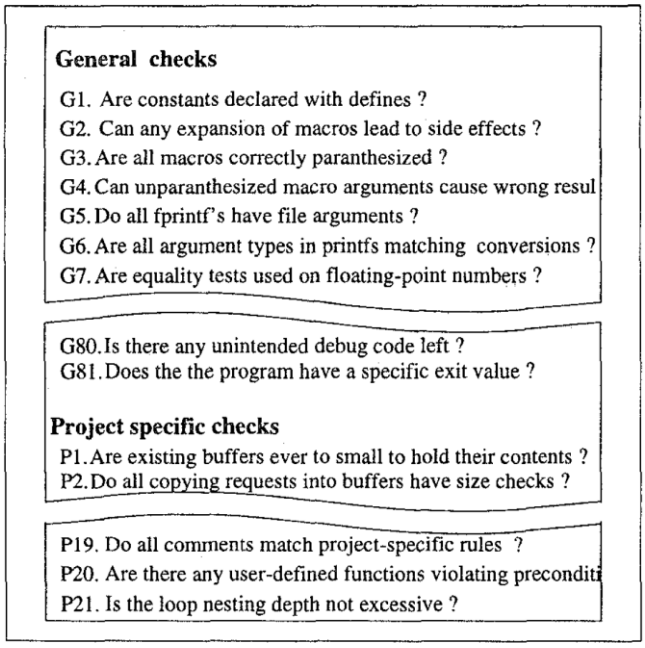


图 3　一个典型的代码检查规则清单节选图

在项目中积累了大量软件问题报告历史数据的支持下，可以从中进一步细化出与项目或开发人员相关的检查规则。

在学习任何一种计算机编程语言时，总是按照基本数据类型－>表达式－>语句－>复杂语句－>函数－>整个程序体(类)的顺序逐步学习的。事实上软件正是按照这样的顺序自下而上逐层组建起来的，代码缺陷作为软件编程写时的一种异常情况，毫不例外也是按照这样层次的构建而成。在实际测试项目的代码检查过程中，我们发现在每个层次上都有可能存在潜在代码缺陷，要找到引起软件问题的根源，要求在尽可能低的层次上找到引发缺陷的代码。正因如此，非常有必要在 C++语法的每个层次上都建立相应的检查元规则。

图 4 为一个代码检查规则体系模型图[2]，图中展示了在代码检查项目开始前，通过逐级组合各种元规则和规则形成新的检查规则，最后形成了初始的检查清单。在项目实践中，经过对缺陷代码模式的推导，进而得到扩展的检查清单。初始检查清单和扩展检查清单本

质上并没有什么区别，只是因为形成的时间不同。

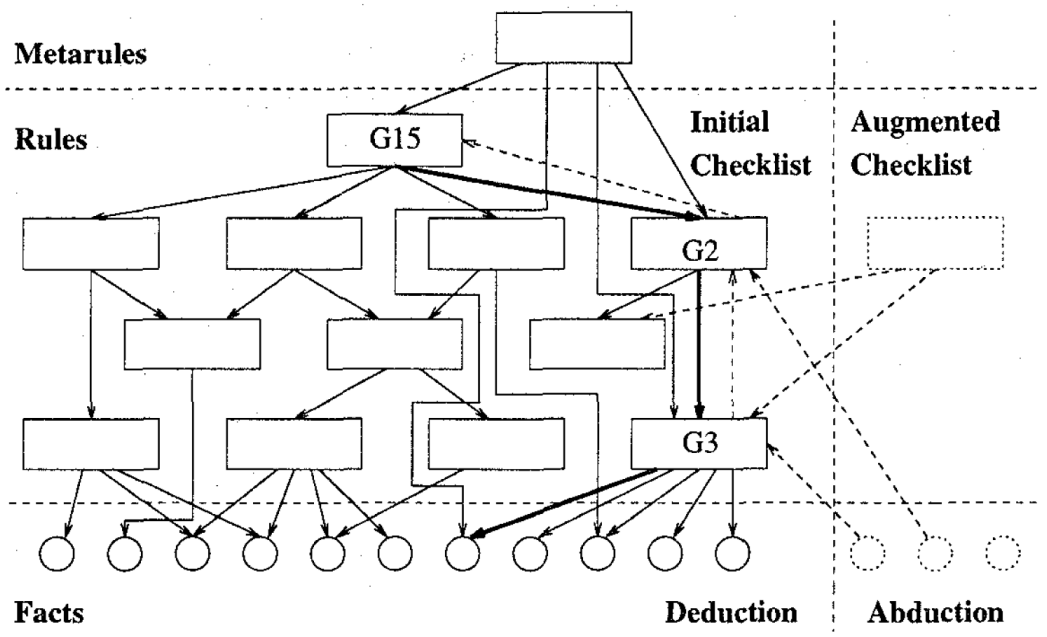


图 4　代码检查规则体系模型图

在检查代码时我们有时会想要定义一个带有否定意义的规则，如“在 AA 情况下如果没有 BB，则可能存在一个问题”。这类检查规则采用自然语言描述比较容易，但是要用代码实现起来往往并不简单，并且对这类规则的定义和维护也比较麻烦。定义组合规则，是解决这类问题一种变通的方法。

下面简单介绍一下定义组合规则的原理。如图 5 中所示定义三个规则，“满足情况 AA”对应规则 R1，“满足在 AA 情况下出现 BB”对应规则 R2， 将满足 R1 但不满足 R2（即以！符号表示）组合则对应规则 R3－“在 AA 情况下如果没有 BB，则可能存在一个问题”。

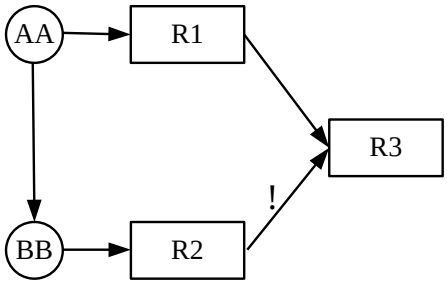


图 5　组合规则示例图

根据前面讨论，本文将代码检查的规则分类设计如下：

- 公共规则

定义针对函数体（含）以上层次的检查规则，在这些层次上出现的缺陷问题一般不容易精确到具体的代码行。

- 关键字规则

针对每个关键字定义的检查规则。由于关键字是 C++语法中一种最普通的元素，单独使用关键字规则的意义不大，一般情况需要和语句、表达式规则或者复杂语句规则配合使用。

- 语句/表达式规则

针对基本语句类型或基本表达式定义的规则，满足对应结构的表达式，则可认为符合了相应的表达式规则。语句/表达式规则中可以包含多个关键字，在同一语句/表达式规则中包含的关键字地位是平等的，与检查的先后次序无关。

- 复杂语句块规则

针对条件、开关选择等多分支语句定义的规则，通常由关键字、语句/表达式进行组合来定义复杂语句块，并在定义时可以进行嵌套，在定义复杂语句块规则加入语句或表达式和复杂语句时需要考虑检查的先后次序。

- 高级组合规则

关键字规则、语句/表达式规则和复杂语句块规则合称为普通规则。

对于难以使用普通规则定义方式定义的复杂语义，需要定义高级组合规则。定义高级组合规则可以使用上面几种规则作为基本单元，也可以嵌套使用其它组合规则。

图 6 为一个由下至上、由多个缺陷代码模式组合形成的组合规则结构图。其中{}表示某条缺陷代码模式对应的规则。

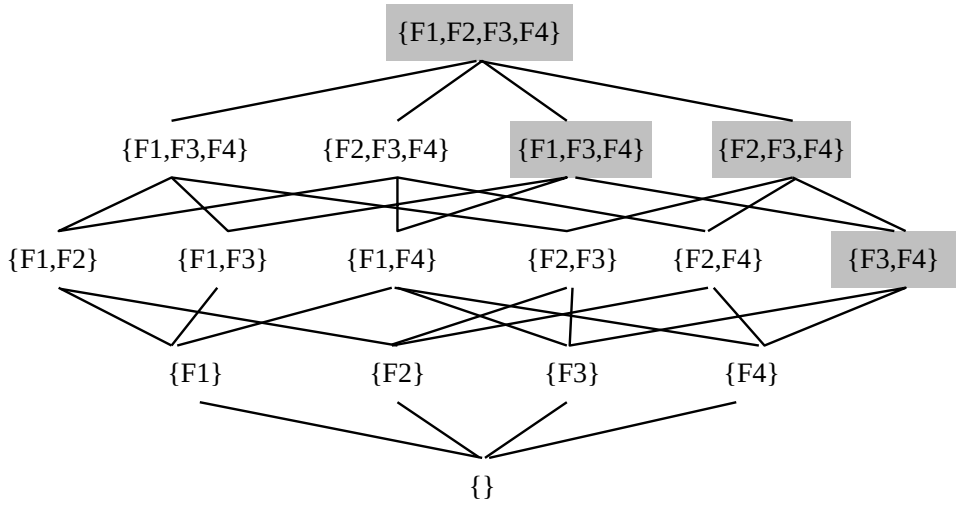


图 6　组合规则结构图

六、 代码分析方法

1、静态分析

静态分析主要对源代码进行词法分析、语法分析，提取被分析程序的静态信息，所提取的静态信息是代码缺陷检测的基础。静态分析结果主要包括三部分信息：

程序定义信息：程序定义信息包含了程序中所有的定义和声明信息，如类定义、方法和数据成员的定义、方法内局部变量的定义等。

程序结构信息：主要指方法内的控制流信息和方法间的调用关系。静态分析器分析程序的语句分支、分支间的嵌套关系和方法调用，记录方法的控制流信息和调用信息，构造语法树。

分支内的变量操作：以方法控制流程中的分支为基本单元，记录每一分支中各语句对各变量施加的操作和操作序列。

2、数据流分析

数据流分析也是一种静态代码检查方法。它是在不通过计算机运行被测程序的条件下，利用预先进行静态分析后获取的信息，检测对变量的赋值与使用操作中，是否存在不合理情况，即找出被测程序中是否存在变量在使用前未被赋值；变量在两次赋值之间未被使用；一个变量在被赋值后是否未被使用等异常情况。

数据流分析目前的主要用途大多局限在编译器的实现和优化技术方面，而在代码检查系统中实用的数据流分析技术并不多见，主要集中在某几种缺陷检测上，如赋值引用异常检测以及内存错误检测，使用方式主要是定义数据流操作的符号，使用该符号系统构造数据流表达式（由数据操作符号构成的符号串），再分析该符号串来确定是否存在代码缺陷。

数据流分析包括以下两个步骤：一是分析程序的所有逻辑路径；二是对所有逻辑路径上的所有变量，分析其所有操作序列，然后将得到的操作序列输入自动机进行分析。因此数据流分析方法不可避免的存在以下缺点：

1）信息量多，上面所述的数据流分析方法是一种穷举法。事实上一个变量在大部分路径上存在问题的几率并不高，因此穷举每个变量的所有操作序列不可避免的要分析很多正确的信息，而且信息量巨大；

2）组合爆炸，当程序复杂度增长时，该分析方法的复杂度呈几何级数增长，并且当这种组合是建立在对所有逻辑路径、所有变量的穷举基础上时，如果不能找到一个非常高效的算法，数据流分析方法将是一个非常低效的方法；

3）实用性低，上述两点导致的数据流分析的实用性降低。

为缓解这些的缺点，数据流分析过程有许多改进方法，但实现都具有一定难度。本系统中数据流分析不是重点，采取的策略是尽可能简化数据流分析的过程，或者在可能的情况下尽量避免数据流分析。

七、 代码检查方法的研究

1、基于 DOM 的模式匹配计算

对程序代码进行静态分析后，代码中所包含的大量各类语法信息，对于代码检查具有很好的指导意义。为了方便在静态分析的基础上进一步的研究，并使得静态分析结果能方便的被解释和显示，甚至能提供给其它的模块或者软件使用，需要采用一种合适的格式来保存静态分析结果。在本文中选择的是 XML 格式。

XML 即可扩展标记语言（eXtensible Markup Language）是一种具有数据描述功能、高度结构性及可验证性的语言。在 XML 文件中，可以使用标记并配合属性来描述数据，因此，XML 非常适合用于作为对象或者标准的描述语言。

对于 XML 有两种解析方式：

DOM：文档对象模型（ Document Object Model）

DOM 以一种平台和语言无关的方式表示 XML 文档的官方 W3C 标准。DOM 是以层次结构组织的结点或信息片断的集合。这个层次结构允许开发人员在树中寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能做任何工作。由于它是基于信息层次的，因而 DOM 被认为是基于树结构或基于对象的。DOM 以及广义的基于树的处理具有几个优点。

首先，由于树在内存中是持久的，因此可以修改它以便应用程序能对数据和结构做出更改。它还可以在任何时候在树中上下导航，而不是像 SAX 那样是一次性的处理。DOM 使用起来也要简单得多。

另一方面，对于特别大的文档，解析和加载整个文档可能很慢且很耗资源，因此使用其他手段来处理这样的数据会更好。这些基于事件的模型，比如 SAX。

SAX（Simple API for XML）

这种处理的优点非常类似于流媒体的优点。分析能够立即开始，而不是等待所有的数据被处理。而且，由于应用程序只是在读取数据时检查数据，因此不需要将数据存储在内存中。这对于大型文档来说是个巨大的优点。事实上，应用程序甚至不必解析整个文档；

它可以在某个条件得到满足时停止解析。一般来说，SAX 还比它的替代者 DOM 快许多。

对于需要自己编写代码来处理 XML 文档的开发人员来说，选择 DOM 还是 SAX 解析模型是一个非常重要的设计决策。

DOM 采用建立树形结构的方式访问 XML 文档，而 SAX 采用的事件模型。

DOM 解析器把 XML 文档转化为一棵包含其内容的树，并可以对树进行遍历。用 DOM 解析模型的优点是编程容易，开发人员只需要调用建树的指令，然后利用 navigation APIs 访问所需的树结点来完成任务。可以很容易的添加和修改树中的元素。然而由于使用 DOM 解析器的时候需要处理整个 XML 文档，所以对性能和内存的要求比较高，尤其是遇到很大的 XML 文件的时候。由于它的遍历能力，DOM 解析器常用于 XML 文档需要频繁的改变的服务中。

SAX 解析器采用了基于事件的模型，它在解析 XML 文档的时候可以触发一系列的事件，当发现给定的 tag 的时候，它可以激活一个回调方法，告诉该方法制定的标签已经找到。SAX 对内存的要求通常会比较低，因为它让开发人员自己来决定所要处理的 tag。特别是当开发人员只需要处理文档中所包含的部分数据时，SAX 这种扩展能力得到了更好的体现。但用 SAX 解析器的时候编码工作会比较困难，而且很难同时访问同一个文档中的多处不同数据。

通过以上对这两种解析特点的比较，不能看出，对于处理静态分析结果这种结构性很强，并且内容保持相对稳定的信息数据，选用 DOM 解析模型是非常合适。

JDOM：Java 文档对象模型 （Java Document Object Model）

JDOM 的本质是一种使用 Java 作为编程语言实现文档对象模型解析 XML 的工具，它简化与 XML 的交互并且比使用 DOM 实现更快。由于它是第一个 Java 特定模型，JDOM 一直得到大力推广和促进。正在考虑通过“Java 规范请求 JSR-102”将它最终用作“Java 标准扩展”。

虽然，JDOM 与 j2sdk 中包含的标准 DOM 都能实现对 XML 的 DOM 方式解析，但两者仍然在两个方面存在不同。首先，JDOM 仅使用具体类而不使用接口。这在某些方面简化了 API，但是也限制了灵活性。第二，JDOM 的 API 大量使用了 Collections 类，简化了那些已经熟悉这些类的 Java 开发者的使用。

JDOM 文档声明其目的是“使用 20%（或更少）的精力解决 80%（或更多）Java/XML 问题”（根据学习曲线假定为 20%）。JDOM 对于大多数 Java/XML 应用程序来说当然是有

用的，并且大多数开发者发现 API 比 DOM 容易理解得多。JDOM 还包括对程序行为的相当广泛检查以防止用户做任何在 XML 中无意义的事。然而，它仍需要充分理解 XML 以便做一些超出基本的工作（或者甚至理解某些情况下的错误）。

模式匹配

在传统的《数据结构》教材中都提到了模式匹配这个概念，通常情况下它指的是字符串的模式匹配，即子串定位，是一种重要的串运算。一个通俗的描述如下；设 s 和 t 是给定的两个串，在主串 s 中找到等于子串 t 的过程称为模式匹配。如果在 s 中找到等于 t 的子串，则称匹配成功，函数返回 t 在 s 中的首次出现的存储位置(或序号)，否则匹配失败，返回-1。其中的 t 也称为模式。

使用正则表达式的方法来分析字符串，可以看作一种很好的模式匹配实现。事实上，不少开源代码检查工具之所以使用 Perl、Python 和 Tcl 等脚本语言来实现，也是看中了内置在其中的强大的正则表达式功能。

上面阐述的这种传统的模式匹配和模式的基本元素是单个字符。如果基本元素换成 DOM 模型中 Element 元素，传统的模式匹配转变成了针对 Element 序列的模式匹配。我们称之为基于 DOM 的模式匹配。

当静态分析结果以 DOM 形式解析时，当把代码检查规则也描述成相应的 DOM 对象的 Element 序列时，检查代码对象静态语法树上是否包含代码规则对应的 Element 序列的过程，也就是进行基于 DOM 的模式匹配计算的过程。

图 7 为一个基于 DOM 的模式匹配模型示意。其中左侧的静态语法结构就是使用 DOM M 方式的静态分析结果，而右侧的检查规则就是基于 DOM 方式解析的、潜在的编码规则。

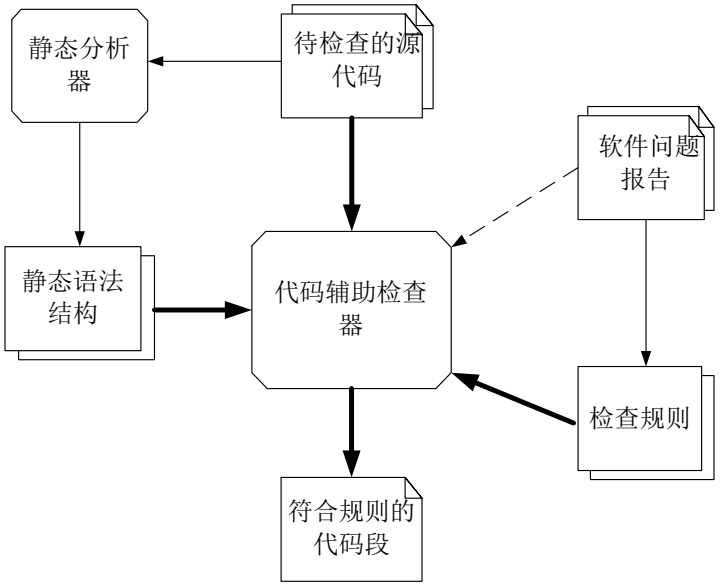


图 7　代码检查辅助工具模型数据流图

2、支持版本变更的代码检查

程序开发过程中，同一模块的设计以及所对应的实现代码总是不会一蹴而就的，随着开发工作持续开展，测试团队会定期收到开发团队提交的新版本程序代码。当版本变化很大时，将新版本认为是一个全新的模块重新进行完整的代码检查，效率影响不大。当然，工程实践中这种极端情况出现的几率并不大，通常情况只会因为改进或增强某项功能，甚至只是为了修复某个软件问题，对一个类中的某几个方法，或者一个函数中某个分支做了一些局部的改动。在这种情况下，如果对新版本的代码再重新按照静态分析、代码检查完整的周期走一遍，肯定是多数代码检查员所不希望的，而且这样做的效率也非常低的。代码检查员总是希望能使用软件把新版本代码中改变部分标示出来，并参考静态分析结果来引导检查的进行，这样会更有针对性、更有效率。

对于分析软件代码的版本差别，有不少成熟的软件工具和算法，常见的工具有：Windows 下的 diff 和 Linux 下的 comp。这些工具大多都是基于文本的，虽然也引入了一些独特高效的算法，来提高比较结果的准确性。但是，实际使用时经常发现工具会对两个不相关的函数进行比较，出现这种情况显然是对代码检查的一种误导。而对于一些对于不同版本使用了不同排版风格，或者仅在使用字符存在差异的代码版本，基于文本的比较工具有时会错误的认为两个版本是截然不同的。

正因为用于代码比较的传统工具存在这些问题，要想代码检查能更准确的引导版本变更情况下的代码检查，代码比较必须采用一个基于结构的，能够排除格式和字符集等影响

的方法。

本文参考传统的版本比较方法，给出了一种基于静态分析结果的版本更替时使用的代码检查方法。首先，分别对不同版本的代码都和新代码一样进行静态分析和深层语句分析；然后对各版本代码的静态分析结果进行比较，即比较两棵抽象语法树的结点序列。比较结点序列时，模仿 comp 工具在设置差异行阈值的方法，设置方法内分支以及分支内部语句的差异阈值，从而确定新代码相对于旧代码，类、方法、方法内部分支、以及语句行的新增/减少/改变（实践中可以处理成新增和减少的组合）等变化情况。比较的结果保存在 compareresult.xml 中，能够通过着色显示的方式来引导代码测试，按照优先检查变化代码的原则开展代码检查工作。

对于版本变化的代码，其分析过程如图 8 所示。

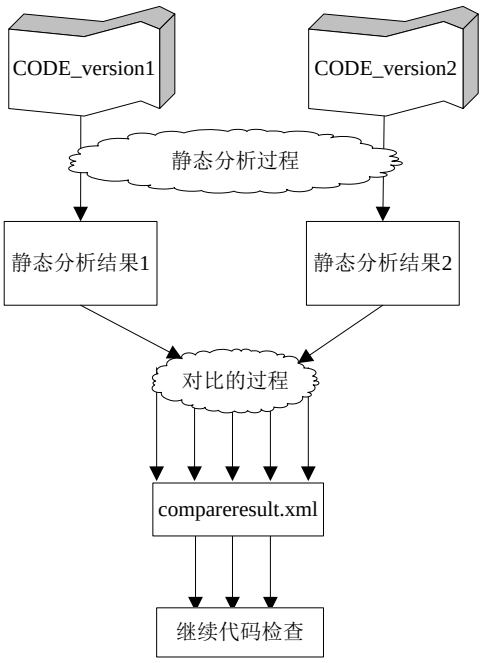


图 8　版本变化的代码检查过程示意图

由图可知，用于做版本比较的主要是静态分析结果文件结构信息。由于所使用的静态分析器尚存在少许不足，一些位于方法以外的语句由于无法进行语句行定位，使得在后来的深层语句分析中也没法分析到，当然也就不可能被包含在分析结果中。虽然，在代码检查实践中可以使用通过获取方法体外语句行文本方法，并进行即时的代码行分析这种方式作为补充，以实现代码检查样本的获取，继而能够进行合适的代码规则匹配检查。但是，对存在于这些地方的版本差异，代码检查过程中，就不可能对这些代码给出有关变更的引导信息。

总结： 本文介绍了代码检查的基本概念、过程以及代码检查规则、代码分析和检查方法，希望通过对本文的阅读，能使读者对代码检查这一静态白盒测试方法有所了解。

参考文献

[1] Dawson Engler, David Yu Chen, Seth Hallem, Andy Chou, and Benjamin Chelf, Bugs as Deviant Behavior: A General Approach to Inferring Errors in Systems Code.[J] ,ACM, 2001

[2] Fevzi Belli, Radu Crisan. Towards Automation of Checklist-Based Code-Reviews[J], 1996 IEEE

[3] Tomoko MATSUMURA, Akito MONDEN, Ken-ichi MATSUMOTO, The Detection of Faulty Code Violating Implicit Coding Rules[C], Proceedings of the 2002 International Symposium on Empirical Software Engineering (ISESE’02).

[4] Tomoko MATSUMURA, Akito MONDEN, Ken-ichi MATSUMOTO, A Method for Detecting of Faulty Code Violating Implicit Coding Rules[J], IWPSE 2002 Orlando Florida.

揭秘 QTP 保留对象机制

作者：陈能技

在 QTP 的帮助文档中，介绍了通过 Dictionary 对象来在 Action 之间共享数据的方法，就是在注册表中添加一个保留对象（reserved object），例如“GlobalDictionary”，把 Dictionary 对象作为环境变量来使用，可以在多个 Action 之间共享数据。文档中仅介绍了如何在注册表中添加这个对象的注册信息，实际上还可以设置更多的注册信息，让添加的保留对象更好用、更强大。

本文将深入探索 QTP 的保留对象机制，以及如何利用 QTP 的保留对象机制来简化测试脚本的编写工作。

一、 什么是保留对象？

在 QTP 自动化测试脚本编写过程中，我们通常会使用 QTP 提供的一些函数或对象，例如帮助文档中列出的 Utility 中的对象：

```
Crypt Object
DataTable Object
Description Object
DotNetFactory Object
DTParameter Object
DTSheet Object
Environment Object
Extern Object
LocalParameter Object
MercuryTimers Object (Collection)
MercuryTimer Object
```

```
Parameter Object
PathFinder Object
Properties Object (Collection)
QCUtil Object
RandomNumber Object
Recovery Object
Reporter Object
RepositoriesCollection Object
Repository Object
Services Object
Setting Object
TextUtil Object
TSLTest Object
XMLUtil Object
```

这些都是 QTP 的保留对象，或者称之为“内置对象”，它们封装了各种常用的函数，对于简化测试脚本的编写起到很大的作用，例如使用 DotNetFactory 这个保留对象就可以方便地访问、创建和使用.NET 的对象，下面是一个小例子，用于创建.NET 的 Form 窗体对象，显示一段时间后关闭：

```
Set var_CreateInstance =DotNetFactory.CreateInstance("System.Windows.Forms.Form",
"System.Windows.Forms")
    var_CreateInstance.Show
    wait 2
    var_CreateInstance.Close
```

二、 注册自定义保留对象

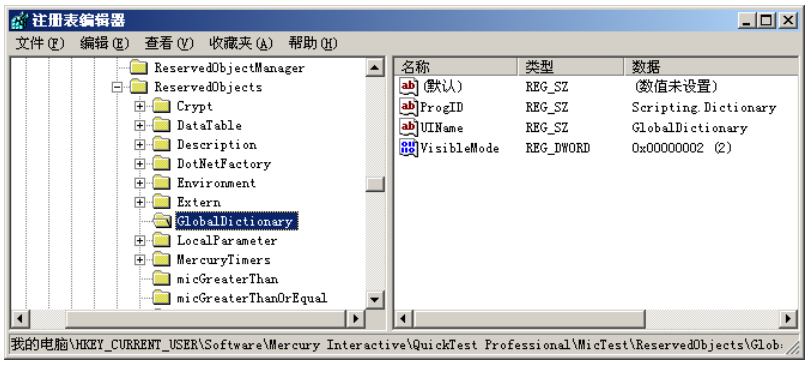
按照 QTP 的帮助文档的提示，我们可以通过修改注册表的方式来添加自定义的保留对象。例如，添加 Dictionary 保留对象的方法如下：

1、打开注册表，定位到下面注册项：

```
HKEY_CURRENT_USER\Software\Mercury
Interactive\QuickTest Professional\MicTest\ReservedObjects
```

实际上，从该注册项所包含的子项可以看到，QTP 的常用保留对象都列在这里了，例如：DataTable、DotNetFactory、Environment 等。而我们要做的是把一些自定义的保留对象添加进去。

2、添加一个名为“GlobalDictionary”的注册项，如图所示：



3、然后添加两个“REG_SZ”类型的注册项，分别是：

ProgID：准备创建的 COM 对象的 ID，在这里就是 Dictionary 对象所对应的 COM 对象名“Scripting.Dictionary”。

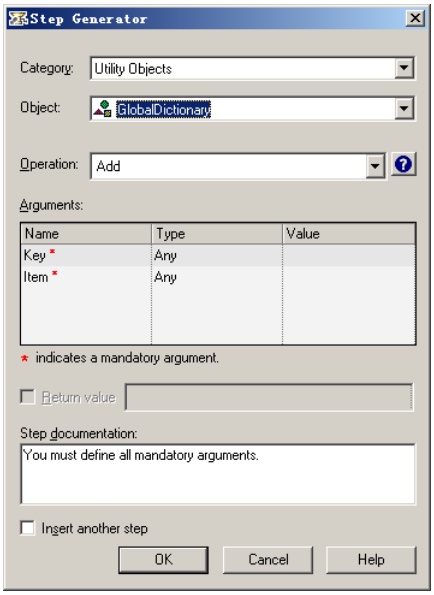
UIName：QTP 中指向保留对象的名字，在这里输入“GlobalDictionary”。

4、然后再添加一个类型为“REG_DWORD”的注册项：

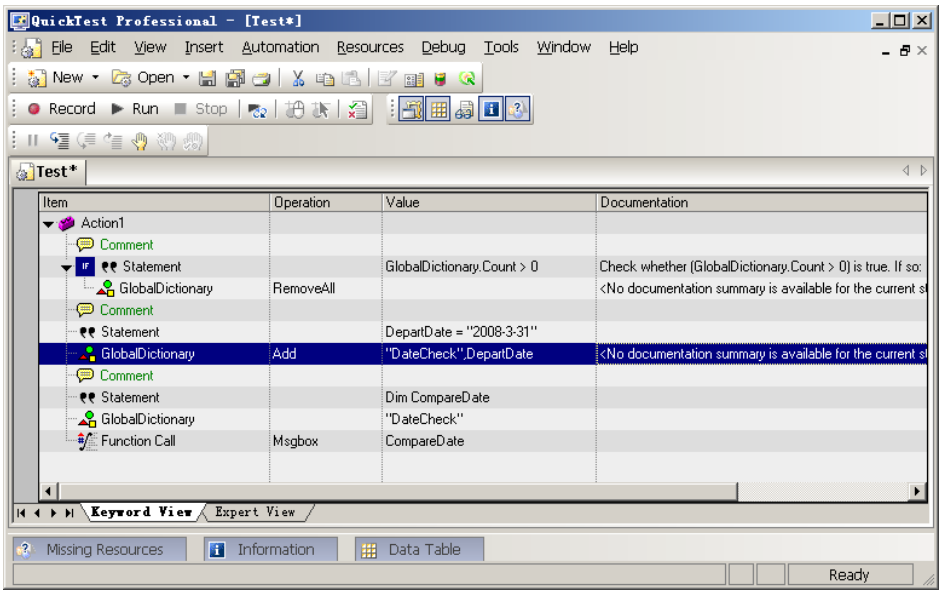
VisibleMode：设置 DWORD 值为 2，用于控制自动完成（auto-complete）和代码智能感知（intelligence）。

三、 使用自定义保留对象

做完这些设置后，重新启动 QTP，进入关键字视图，添加测试步骤，在“Step Generator”中可以选择“Utility Objects”，然后选择已注册的“GlobalDictionary”以及其中的各种方法。



例如，我们可以选择 Dictionary 对象的 Add 方法，在参数（Arguments）中输入 Key 和 Item 的值。单击“OK”，然后在关键字视图中就可以看到新添加的步骤包含的对象：



在专家视图中也可以使用该对象，并且输入对象名，加点号后可以自动弹出对象的方法和属性列表。下面是一个使用 GlobalDictionary 对象的例子：

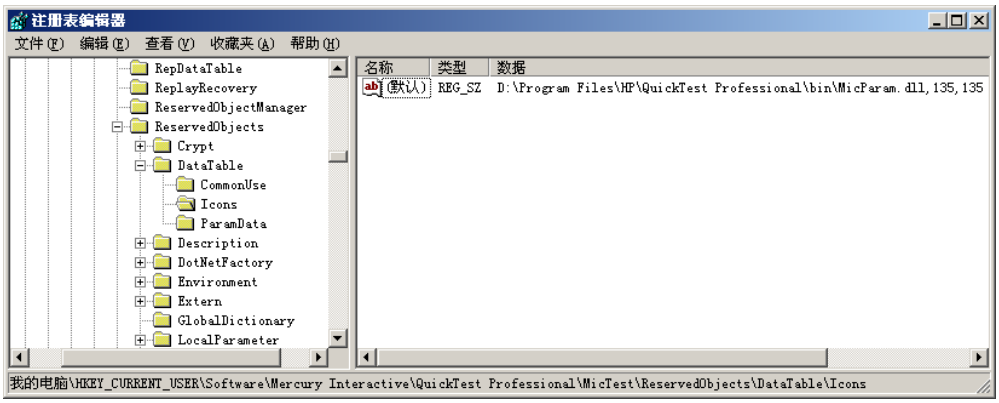
```
’使用 GlobalDictionary 前清空里面的数据
If  GlobalDictionary.Count > 0  Then
   GlobalDictionary.RemoveAll
End If
’存储一个数值
DepartDate = "2008-3-31"
GlobalDictionary.Add "DateCheck", DepartDate
```

使用 GlobalDictionary 添加数据后，可在当前 Action 使用 GlobalDictionary 中的数据，也可在另外一个 Action 中使用添加到 GlobalDictionary 的数据，下面脚本从 GlobalDictionary 中读取数据：

```
Dim CompareDate
CompareDate=GlobalDictionary("DateCheck")
Msgbox CompareDate
```

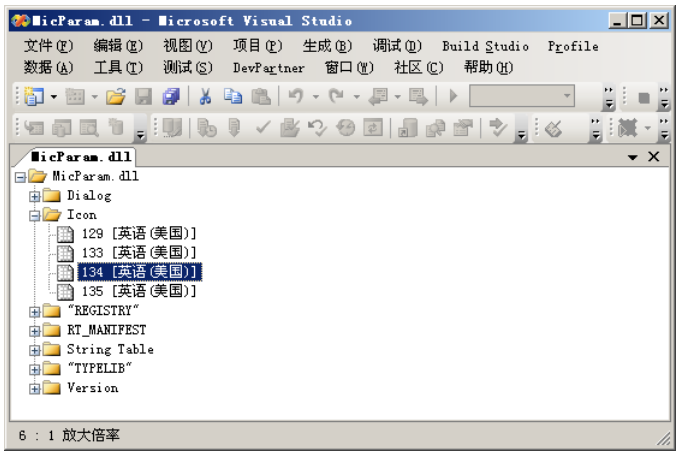
四、 探秘 QTP 保留对象注册信息

查看注册表的其它保留对象的信息可知，还有很多其它的项和值可以添加和设置，例如在 Icons 项中设置对象的图标。

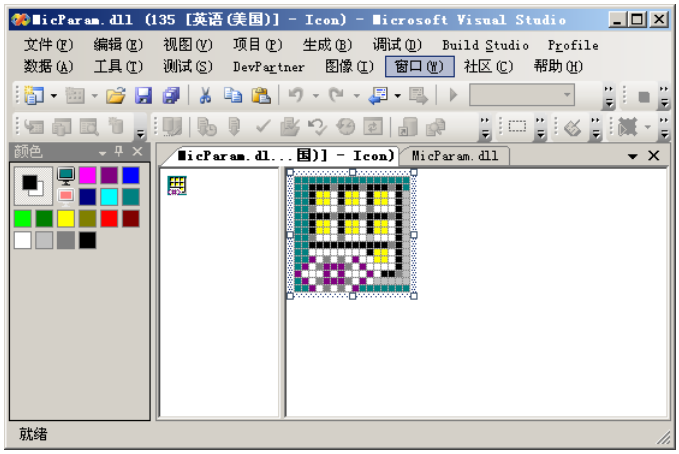


查看 DataTable 对象的 Icons 注册项可看到，图标的数据指向：D:\Program Files\HP\QuickTest Professional\bin\MicParam.dll 这个文件。

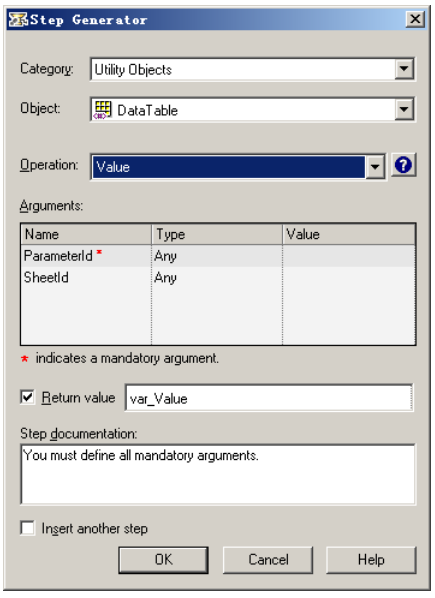
用 VisualStudio.NET 2005 打开这个 DLL 可看到 Icon 的信息：



双击可打开图标编辑器查看 Icon 的详细信息，例如打开名为“135”的 Icon，可看到如图所示的图标。



这正是我们通常在 QTP 的步骤产生器（Step Generator）或关键字视图中看到的 DataTable 对象的图标。



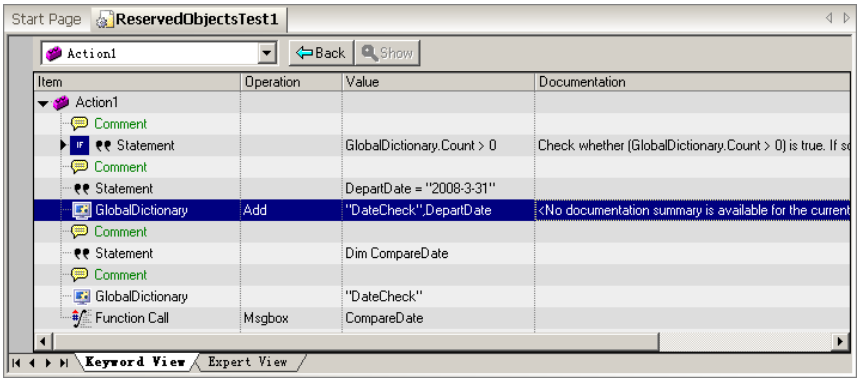
因此，我们也可以为自定义的 GlobalDictionary 对象添加一个图标，方法如下：

1、在注册表中的 GlobalDictionary 项下添加名为“Icons”的注册项。

2、修改默认值，输入包含图标资源的 DLL 所在的文件路径，例如“D:\Program Files\HP\QuickTest Professional\bin\AQTProductInfo.dll”，并且在后面添加指定哪个 Icon，完整的注册项的值为：

D:\Program Files\HP\QuickTest Professional\bin\AQTProductInfo.dll,204,204

3、重新启动 QTP 后，在关键字视图就可以看到 GlobalDictionary 对象的图标已经发生了变化，不再是默认的图标，而是指定 DLL 所包含的图标：



五、 使用自定义保留对象来简化代码编写工作

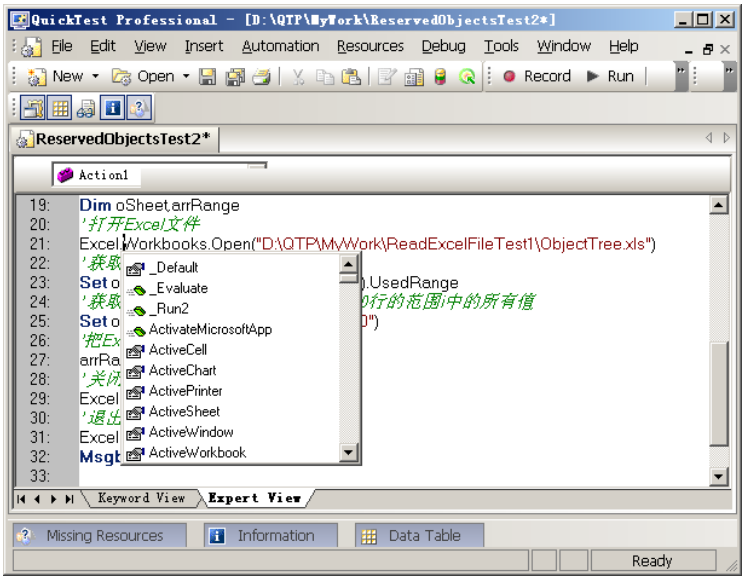
保留对象（Reserved Objects）的机制是 QTP 用于创建单件 COM 对象的内部机制，可以让脚本的编写更加简单。例如，如果不用保留对象的机制，则在 QTP 中使用 Excel 需要按下面代码来实现：

```
    Dim oSheet,arrRange
    ’ 创建 Excel 应用程序对象
    Set oExcel = CreateObject("Excel.Application")
    ’ 打开 Excel 文件
    oExcel.Workbooks.Open("D:\QTP\MyWork\ReadExcelFileTest1\ObjectTree.xls"
)
    ’ 获取表格的使用范围
    Set oSheet = oExcel.Worksheets("Tree").UsedRange
    ’ 获取从 A 列到 Z 列，从第 1 行到第 1000 行的范围中的所有值
    Set oRange = oSheet.Range("A1:Z1000")
    ’把 Excel 数据转换到数组
    arrRange = oRange.Value
    ’ 关闭工作簿
    oExcel.WorkBooks.Item(1).Close
    ’ 退出 Excel
    oExcel.Quit
    Set oExcel = Nothing
```

可看到，脚本中需要使用 CreateObject 方法，首先创建名为“Excel.Application”的 COM 对象，然后才能使用这个 Excel 中的属性和方法。而如果在注册表中添加了 Excel 对象作为 QTP 的保留对象，则可以使用下面的代码来实现：

```
    Dim oSheet,arrRange
    ’ 打开 Excel 文件
    Excel.Workbooks.Open("D:\QTP\MyWork\ReadExcelFileTest1\ObjectTree.xls"
)
    ’ 获取表格的使用范围
    Set oSheet = Excel.Worksheets("Tree").UsedRange
    ’ 获取从 A 列到 Z 列，从第 1 行到第 1000 行的范围 i 中的所有值
    Set oRange = oSheet.Range("A1:Z1000")
    ’把 Excel 数据转换到数组
    arrRange = oRange.Value
    ’ 关闭工作簿
    Excel.WorkBooks.Item(1).Close
    ’ 退出 Excel
    Excel.Quit
```

这种使用 Excel 的 COM 接口的方式不仅省去了创建 COM 对象的代码，更重要的是得到了代码智能感知（IntelliSense）的功能，可以查看到 Excel 应用程序中的 COM 对象接口的所有属性和方法，如图所示。



这将有助于提高测试脚本的编写效率，让某些 COM 对象的使用更为简单、直接，让我们的自动化测试工程师在编写测试脚本时更加高效，所要做的仅仅是在注册表中添加一

些必要的信息即可。

六、 保留对象机制解析

在掌握了自定义保留对象的创建和使用方法后，我们回过头来思考一下 QTP 的保留对象机制：

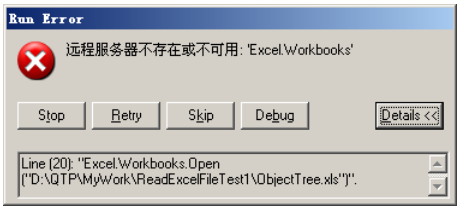
1、COM 对象是可以在 QTP 中通过 CreateObject 命令创建的对象，例如 Scripting.Dictionary、Word.Application 等。

2、保留对象就是把这些 COM 对象作为 QTP 可自动识别的、已定义的对象来使用，这些对象在 QTP 启动时被加载和创建，并且只创建一次，是单件对象。

3、创建后可以在 QTP 的脚本中直接访问和使用，并且对象可在所有 Action 中共享，类似于全局的变量，可用于替代环境变量（Environment）使用，但是作为对象拥有更丰富的“内涵”，包括各种属性和方法，

4、保留对象可用于简化和组织很多编码工作：文件 IO、写 Log、使用 Excel、Word、Outlook 等，也可以用于替代环境变量，例如用 Dictionary 对象来替代环境变量，而不需要在环境变量中定义大量的变量，而且能利用 Dictionary 对象的灵活结构来存储、编辑和访问大量的数据。

5、QTP 启动时会自动创建保留对象，例如我们把 Excel 注册到 QTP 的保留对象中，则 QTP 在启动时也会把 Excel 进程也加载，并且可以看到在 QTP 的整个运行过程中，Excel 进程都处于激活状态，可以在 Windows 任务管理器中看到 Excel 的进程，当 QTP 退出时，Excel 进程也会被关闭。因此，在这种方式下使用 Excel 要特别注意 Excel 进程不要被意外关闭，否则将出现如图所示的错误。



小结：

本文深入探索了 QTP 的保留对象的机制，介绍了如何创建和使用自定义的保留对象来简化自动化测试脚本的编写工作，提高自动化测试脚本的编写效率。

实际上，QTP 中还隐藏着不少尚未对外公开的“秘密”，有待那些充满“好奇心”的人们去挖掘！

基于协议应用的系统之性能测试方法讨论

作者：方耀 杨燕 兰海

摘　要： 在 LOADRUNNER 中使用脚本和 IP 欺骗对系统进行性能测试

关 键 词： LOADRUNNER IP 欺骗 协议

出　处： 广东亿迅科技有限公司　质量管理部

一、 前言

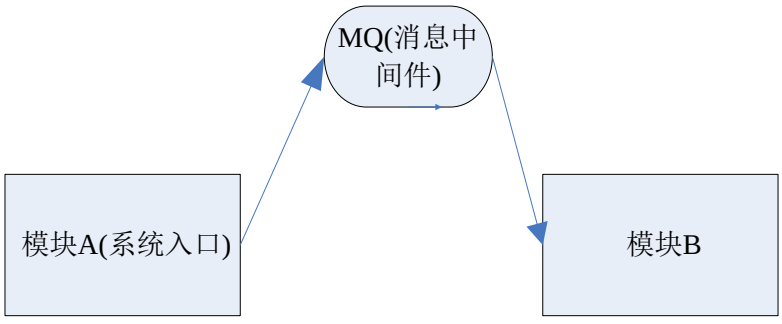
在当前信息技术的高速发展中，特别是诸如电信、金融、银行、保险等领域，对于这些领域的信息化支撑，都离不开大型系统、数据库、中间件等软件产品的支持。在这些软件产品中，对数据的采集、读取、储存等，都有严格的时效性要求。而这些软件产品之间都是通过某种特定的协议进行交互的，如 TCP、UDP、HTTP 等，为了满足特定的时效性，我们通常都要对这些软件产品的应用、通讯等进行性能测试，以期发现问题并解决问题。

本文使用常用的性能测试工具 LOADRUNNER 和解包工具 Ethereal，讨论基于协议应用的系统性能测试的方法，希望能抛砖引玉，寻找出更新更好的方法。

二、 任务引入

某大型系统（以下称 N 系统）应用到 IBM MQ 中间件，但 MQ 支持持久化和非持久化两种方式的信息传递，为了验证 N 系统在 MQ 的持久化和非持久化两种方式下的性能表现（持久化指接收消息并保存到本地，即使 MQ 非法中断，MQ 重启后，消息依然存在；非持久化指接收消息不保存在本地，只负责接收，如果 MQ 非法中断，MQ 重启后，消息丢失），需要做性能测试。

N 系统的部分模块流程图如下：



模块 A 接收从客户处发来的包，经过处理，送入 MQ 消息中间件中，然后模块 B 再从 MQ 中读取。整个系统需要承载的用户量是千万级的，而且在逐年增加，同时对这个性能测试也需要注意 1 点：需要模拟大量的并发用户发包，对模块 A 产生压力（由于业务的特殊性，模块 A 对同一个 IP 的请求时间间隔是有限制的，但对于不同的 IP，请求一样也可以）；

1. 测试分析

由于开源的客户端工具不支持多并发发包，同时，不可能部署 n 个客户端，所以必须使用 Loadrunner 来进行施压，同时，要使用 Loadrunner 的 IP 欺骗来模拟不同 ip 的客户端；

对于客户端而言，发送的包是通用的文本方式，但对于 Loadrunner 而言，发送的包是要经过处理的双字节 16 进制数据。所以要先在服务器上抓包，然后用 Ethereal 工具分析，再截取出协议包体。

然后用 Loadrunner 的 winsocket 协议来发送定制好的包。

2. 测试实施

2.1 抓包

1、用 root 用户登录后台系统服务器，执行：

snoop -c 500 -s 1500 -o test.pkt

【说明：-c 表示抓取包的个数；-s 表示包的大小；-o 表示输出的文件】

2、此时切换到另一个窗口，用客户端发送包；

3、将服务器上的 test.pkt 下载到本地；

2.2 分析包

1、使用 ethereal 打开“test.pkt”；

2、点击“ 

”，找到刚才客户端 10.17.34.219 发出的 request：

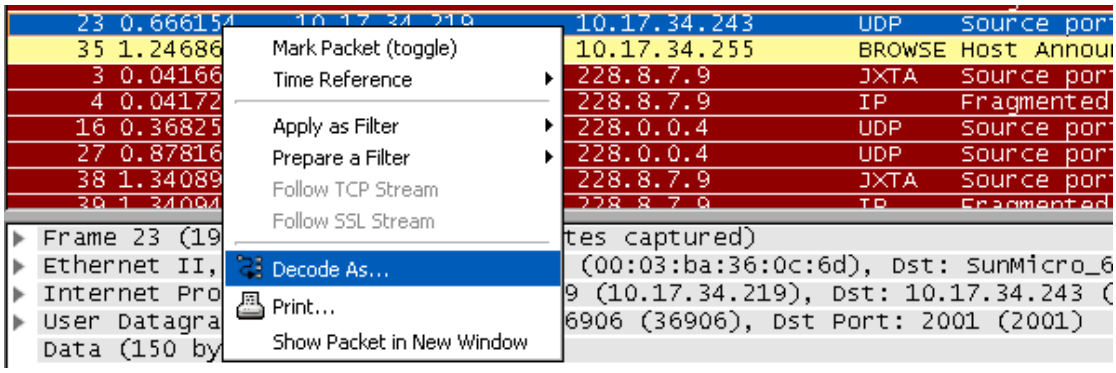


图 2.2-1

3、选择“Decode AS”

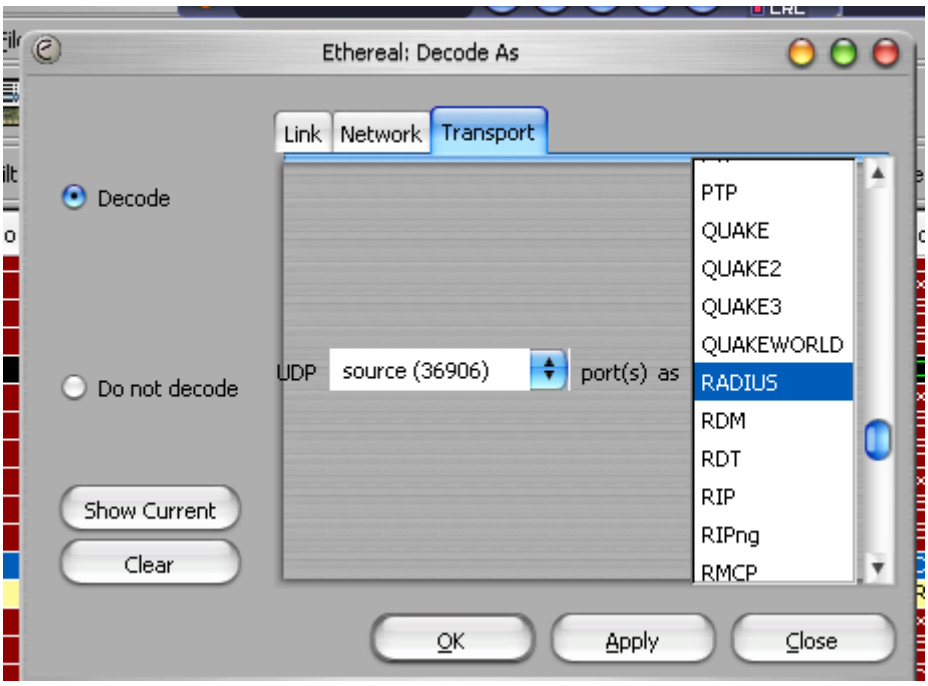


图 2.2-2

选择“RADIUS”。

4、显示出抓到的这个包的 detail：

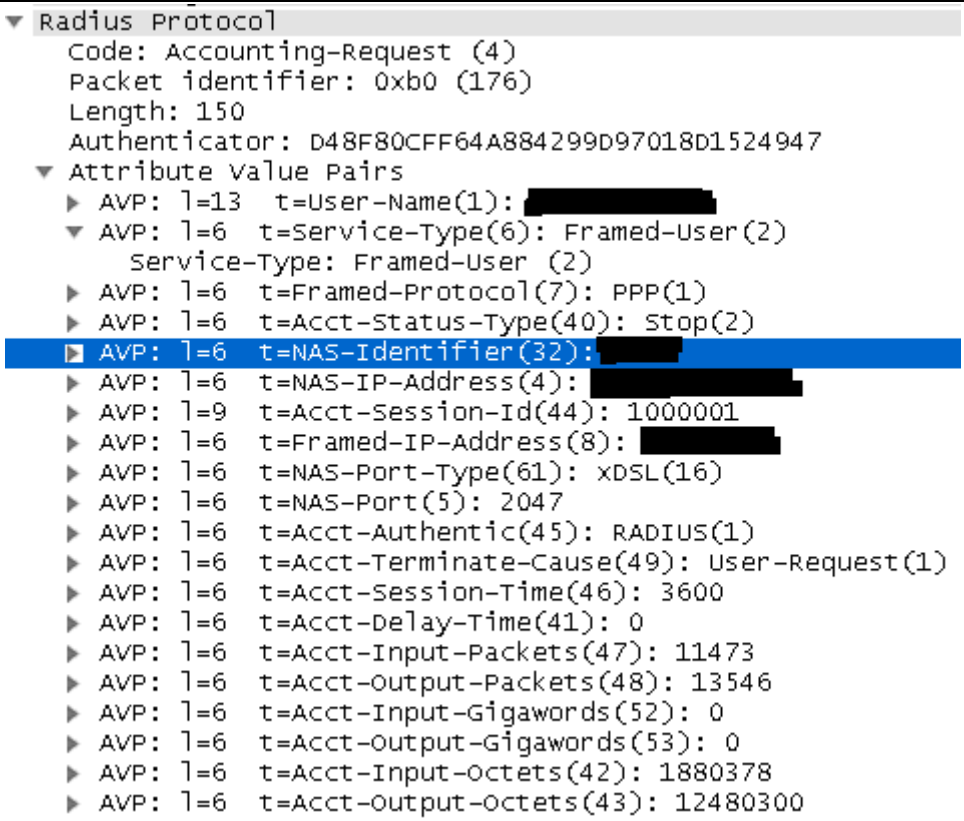


图 2.2-3

5、点击“Code:Accounting-Request(*)”,找到这个包的包头

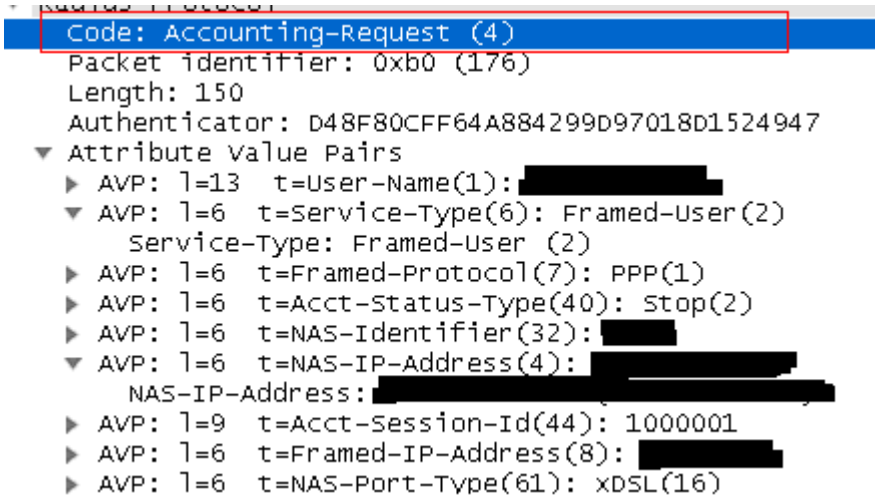


图 2.2-4

最下面的 16 进制的字节包体中会有相应的显示，包头在哪儿里

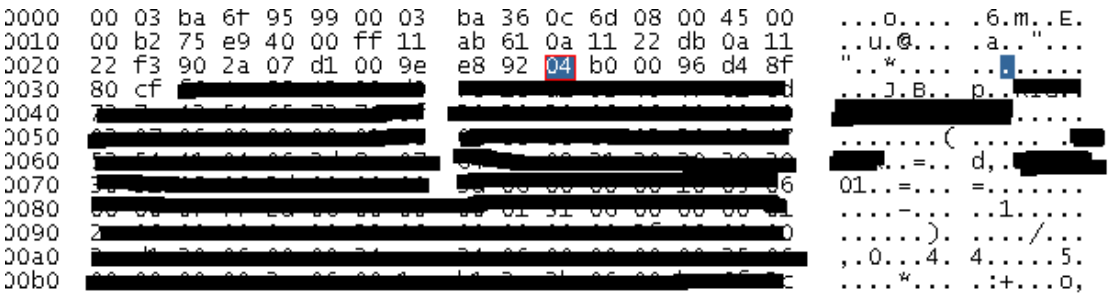


图 2.2-5

6、点击右键，选择“copy”，将这个字节包粘到 utralEdit 中

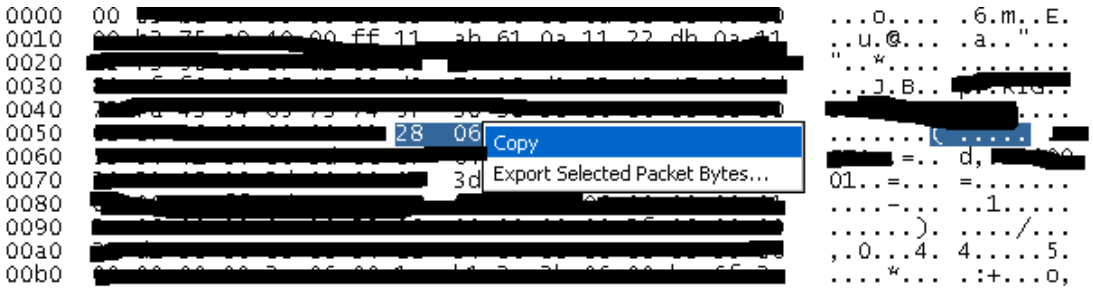


图 2.2-6

7、制作这个包体，将前面和后面的字符去掉，如下所示（在 UE 中进行列模式操作）：

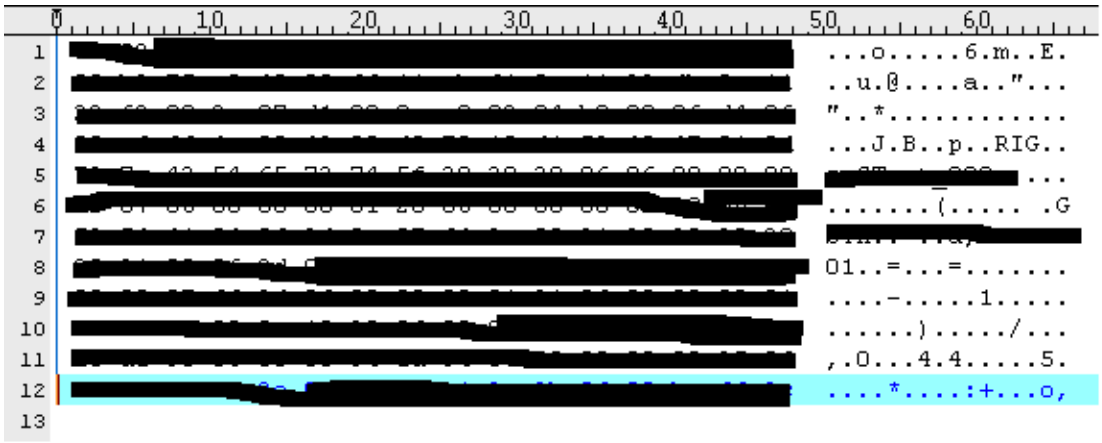


图 2.2-7

去掉 50 至 67 列的乱码部分，再将将图 2.2.3-5 中的包头去掉，如下图所示：

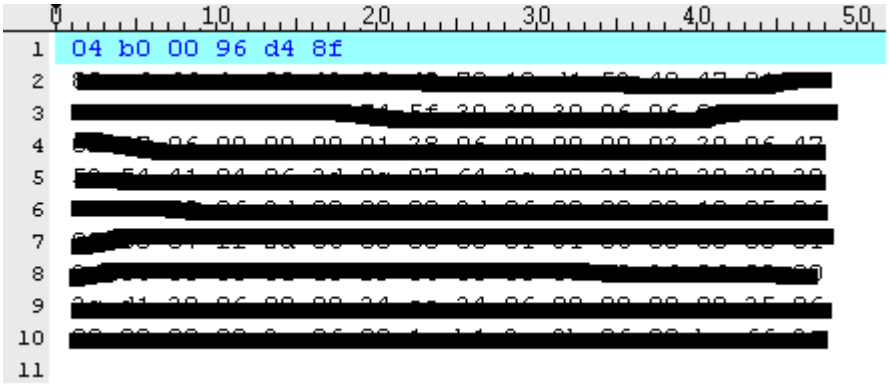


图 2.2-8

8、将空格替换为“\x”：

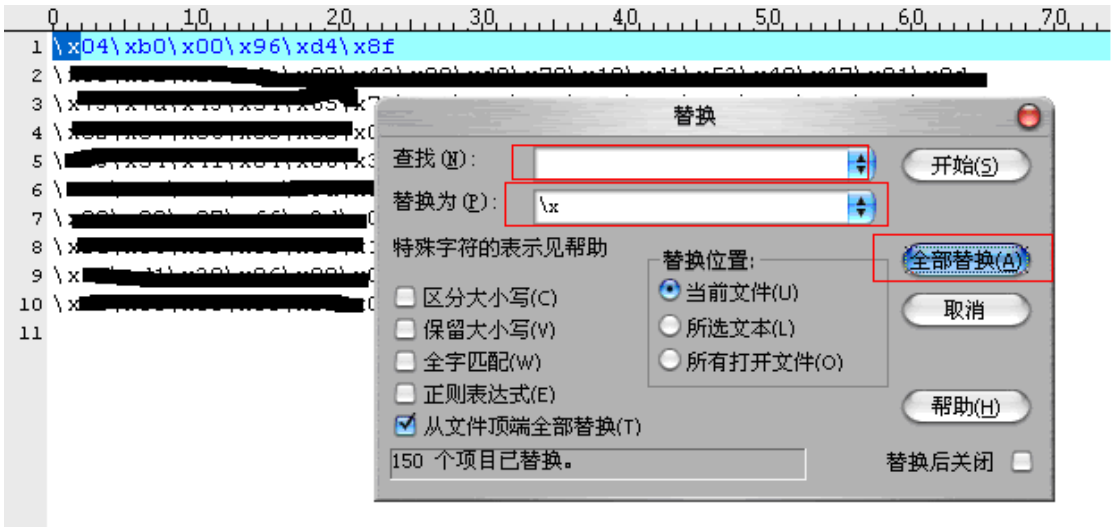
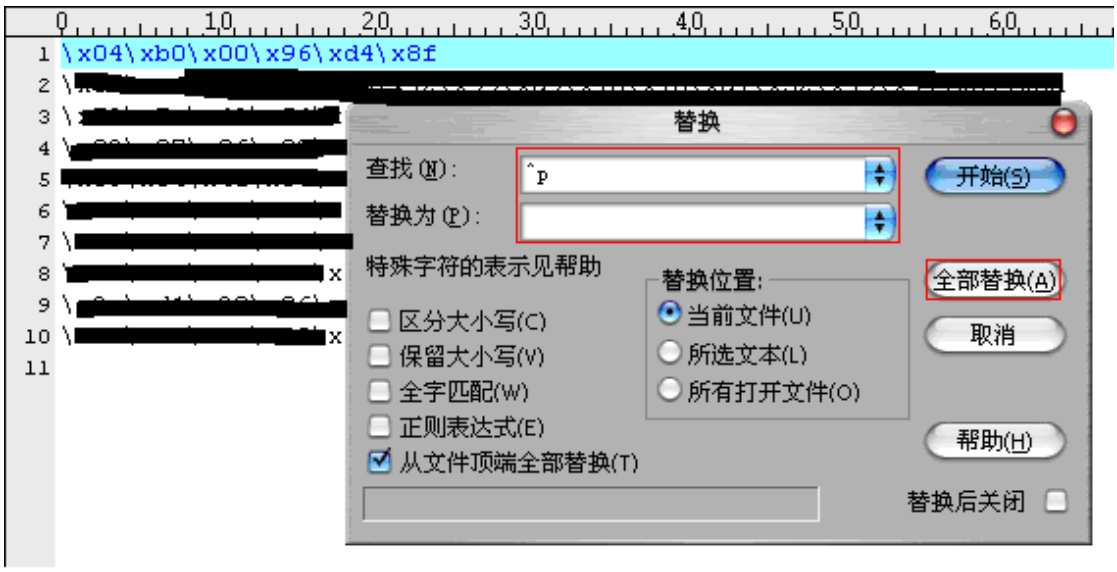


图 2.2-9

将回车去掉：



9、取得 detail 的 16 进制字节流：

\xff\xbf\xff\xff\xdf\xff\xff\xcf\xff\xfa\xff\xff\xff\xdf\xff\xff\xdf\xff\xff\xff\xff\xfd\xff\xfa\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfd\xfc\xff\xff\xfc\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfd\xff\xff\xff\xfd\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfd\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfe\xff\xff\xff\xfe\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfc\xdf\xff\xff\xff\xff\xff\xea\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xfa\xff\xff\xfc\xbf\xfa\xfb\xff\xff\xbe\xff\xfc

说明：由于涉及到商业机密信息，这里的图片进行了处理，最后得到的字节流也是处理的

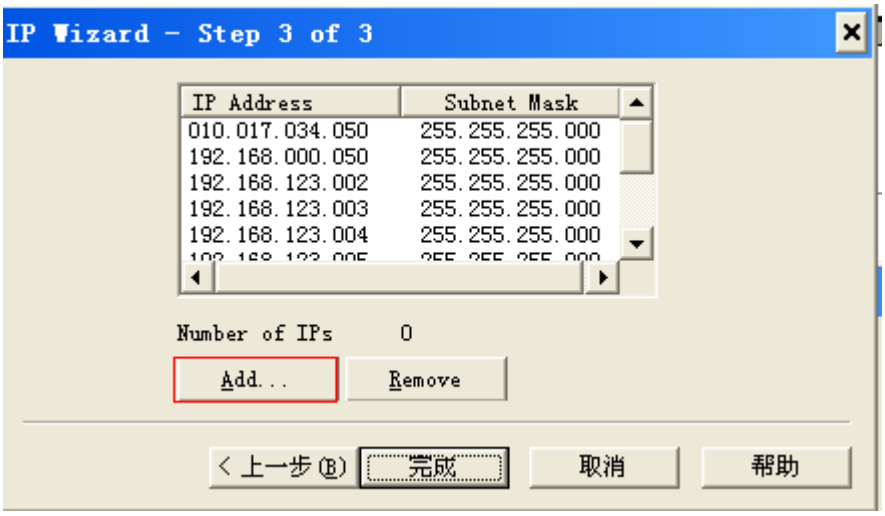
2.3 使用 LoadRunner 制作虚拟 ip

1、点击：程序－－>LR－－>tools－－>IPwizard

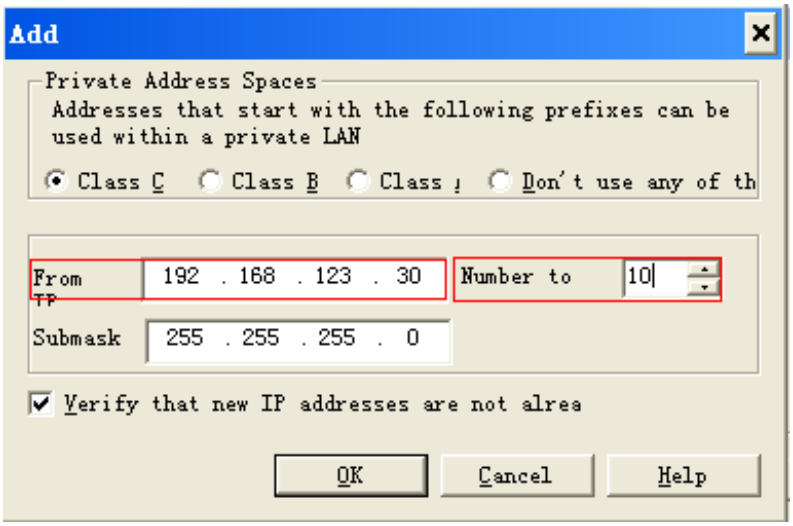
点击下一步

2、点击下一步

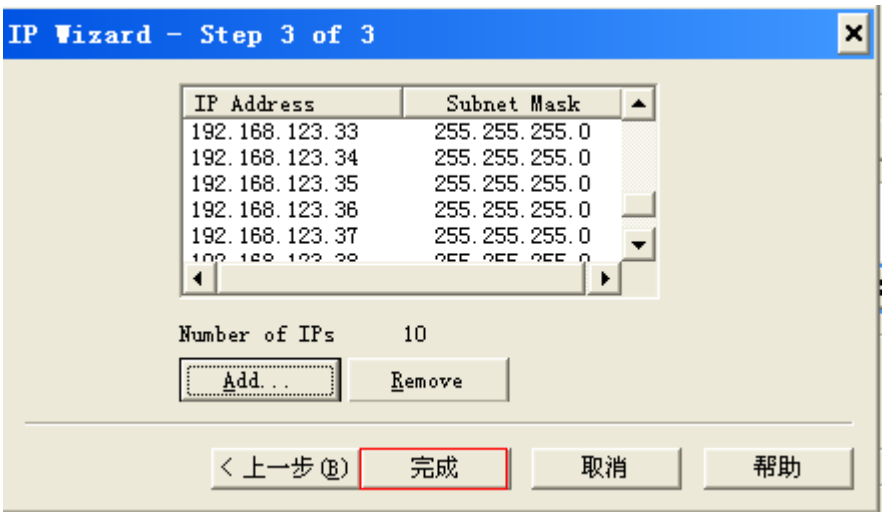
3、点击“Add”



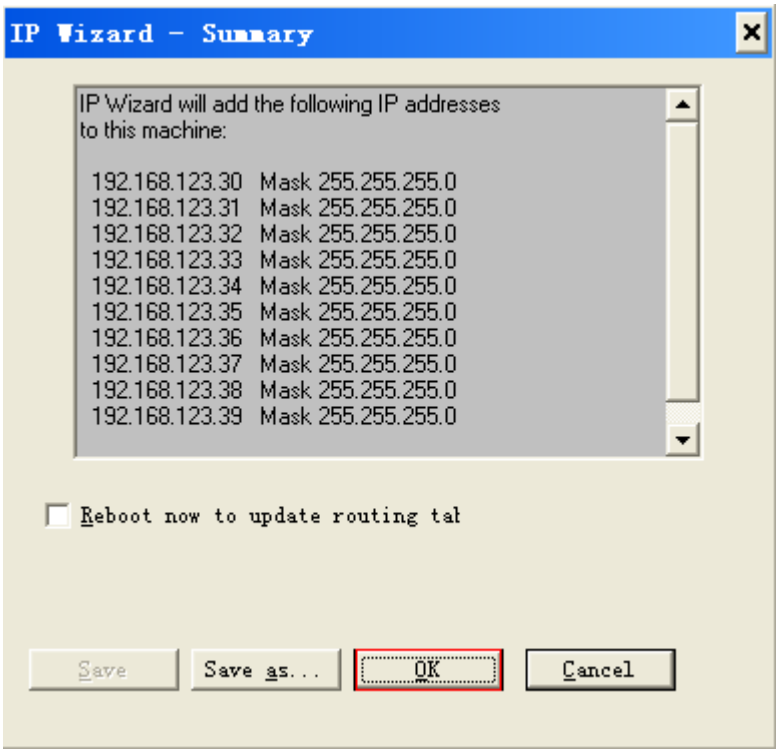
4、填入起始 ip 和需要虚拟的 ip 数量，点击 OK



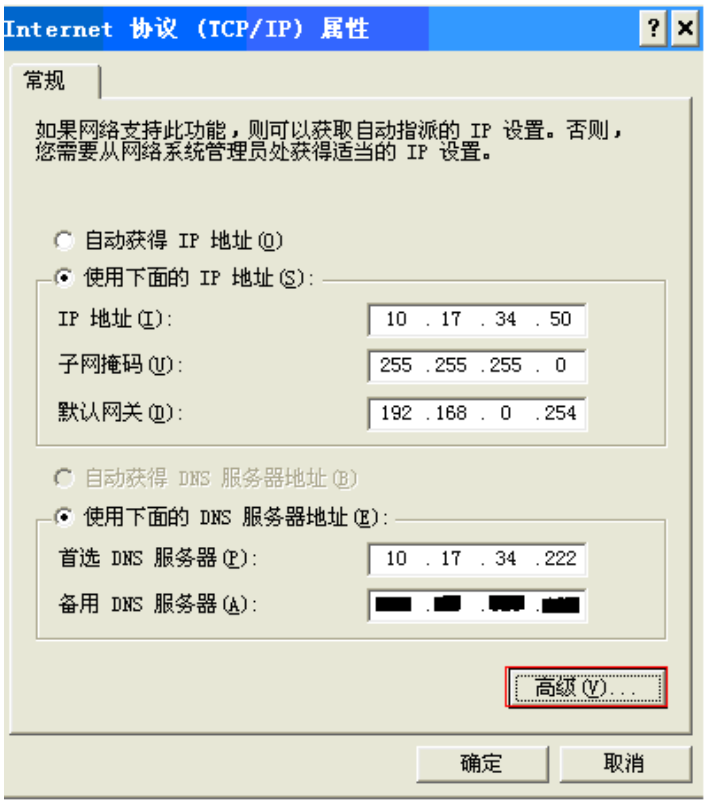
5、查看新加入的虚拟 ip，确认后点击“完成”;

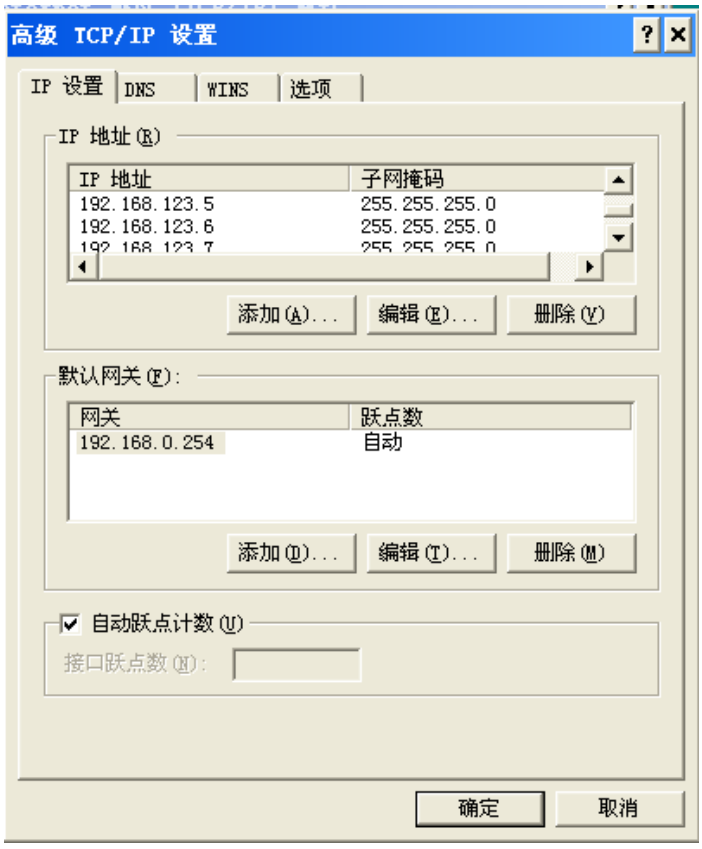

IP Wizard - Step 3 of 3
IP Address
Subnet Mask
192.168.123.33 255.255.255.0
192.168.123.34 255.255.255.0
192.168.123.35 255.255.255.0
192.168.123.36 255.255.255.0
192.168.123.37 255.255.255.0
Number of IPs 10
Add...
Remove
< 上一步(B)
完成
取消
帮助

6、点击“OK”，重启电脑。


IP Wizard - Summary
IP Wizard will add the following IP addresses
to this machine:
192.168.123.30 Mask 255.255.255.0
192.168.123.31 Mask 255.255.255.0
192.168.123.32 Mask 255.255.255.0
192.168.123.33 Mask 255.255.255.0
192.168.123.34 Mask 255.255.255.0
192.168.123.35 Mask 255.255.255.0
192.168.123.36 Mask 255.255.255.0
192.168.123.37 Mask 255.255.255.0
192.168.123.38 Mask 255.255.255.0
192.168.123.39 Mask 255.255.255.0
Reboot now to update routing tal
Save
Save as...
OK
Cancel

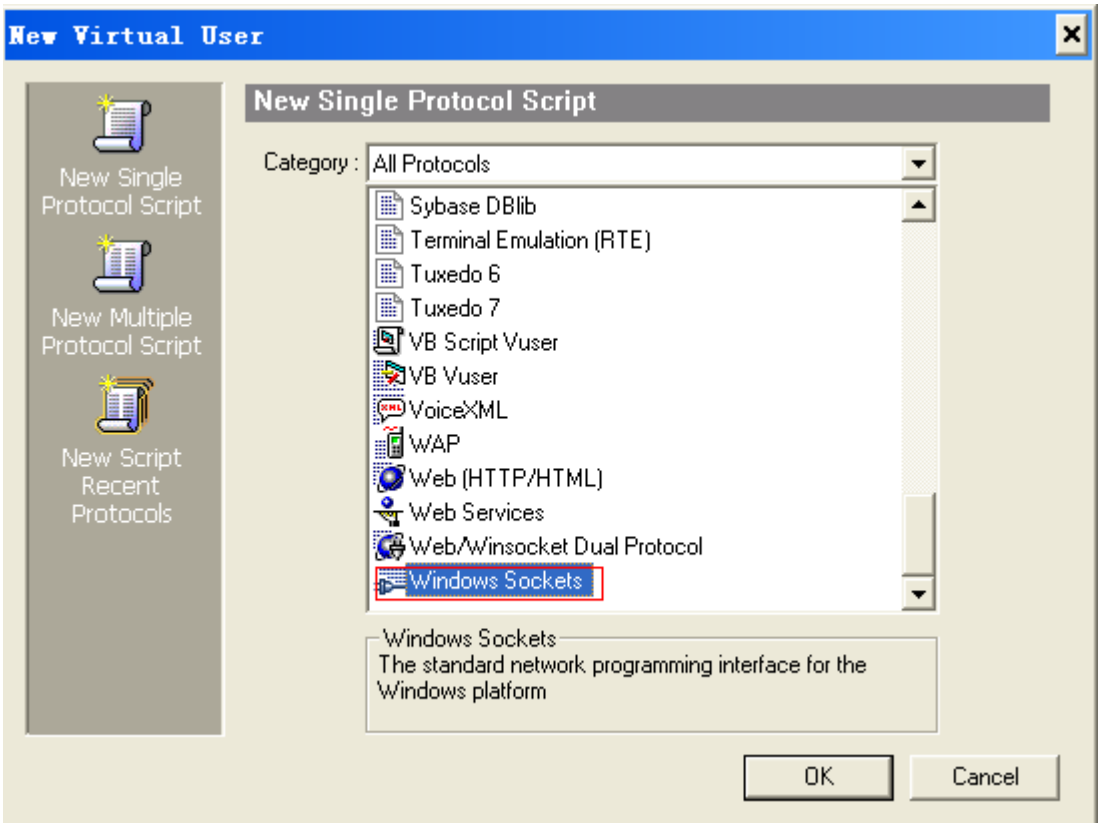
7、查看本机的 ip 设置，确定 ip 中已经有了加入的虚拟 ip



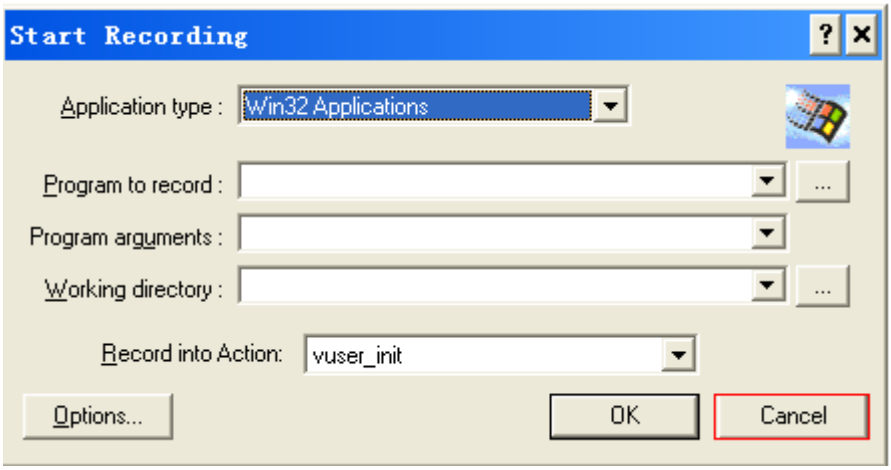


2.4 制作测试脚本

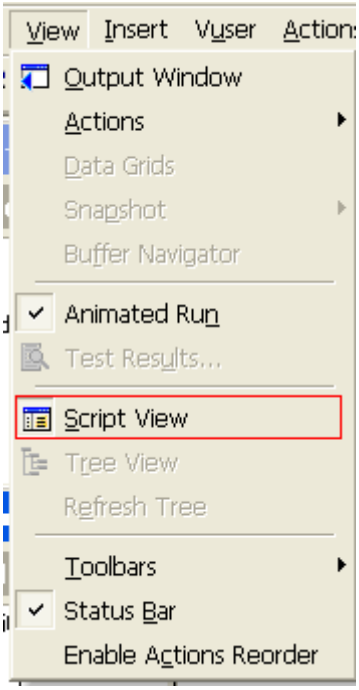
1. 打开 LR 的“Virtual User Generator”
2. New 一个 virtual user，选择“Windows Sockets”



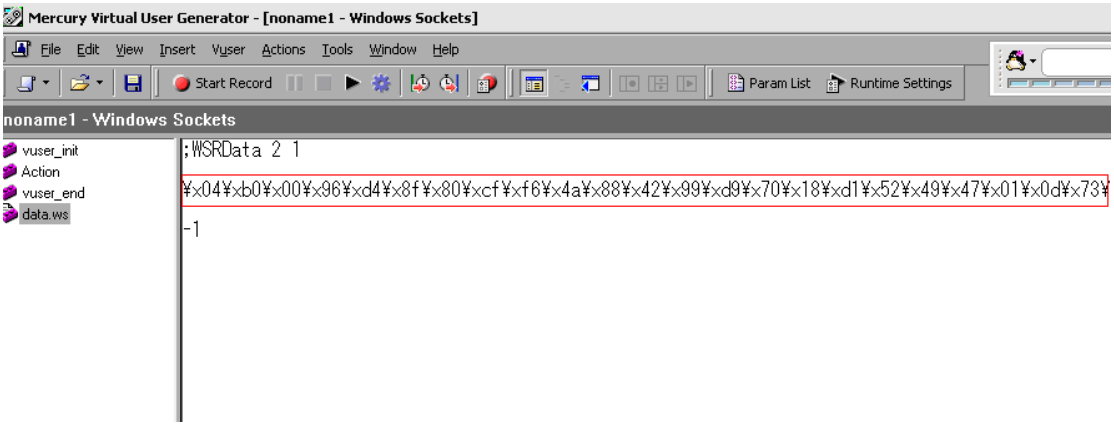
3. 选择“cancel”



4. script View 模式编辑



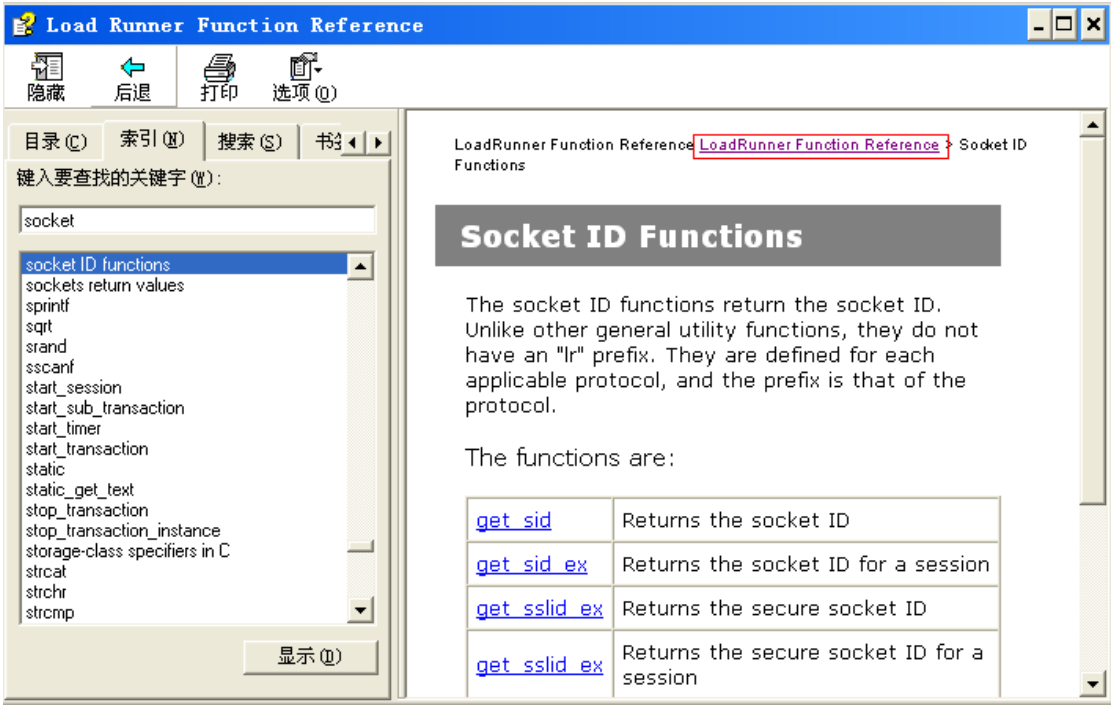
5. 将 2.3.4 中抓包所得的字节流加入到



6. 构造 action

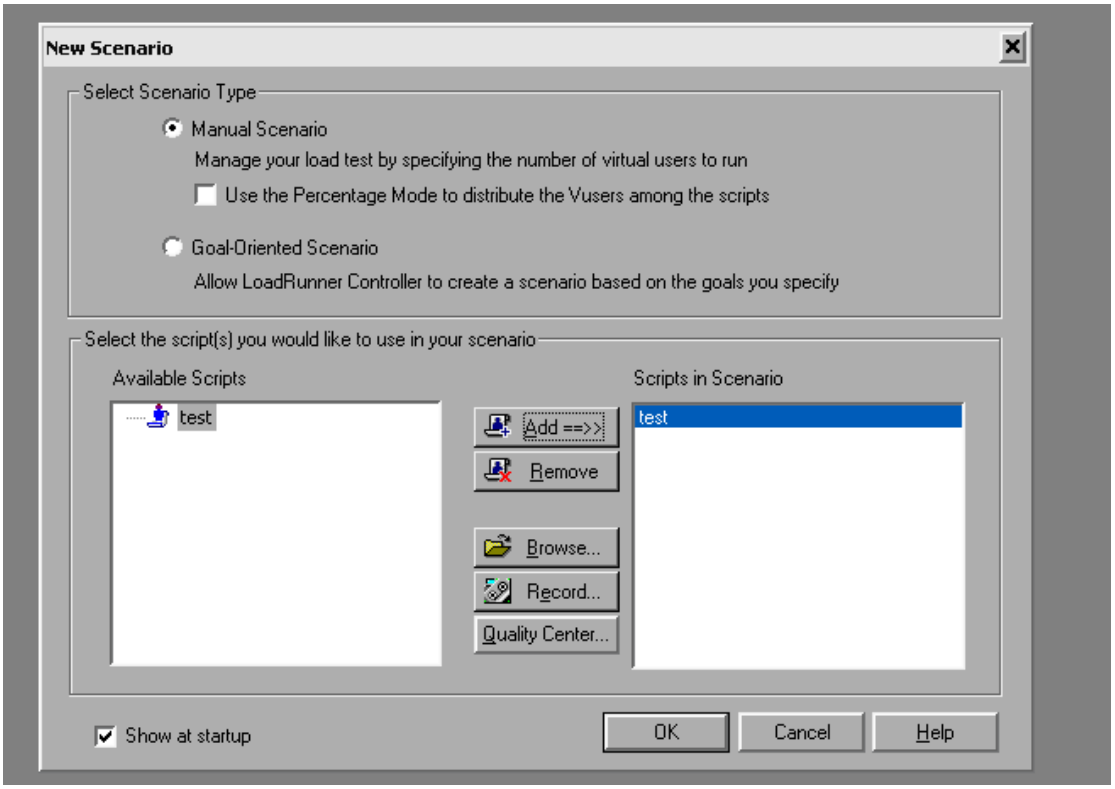
```
#include "lrs.h"

Action()
{
    lrs_create_socket("socket1","UDP","RemoteHost=10.17.34.243:8889"
,LrsLastArg);
    lrs_send("socket1", "buf0", LrsLastArg);
    lrs_close_socket("socket1");
    sleep(5);
    return 0;
}
```

7. 可以查看帮助，搜索关键字“socket”，如下：

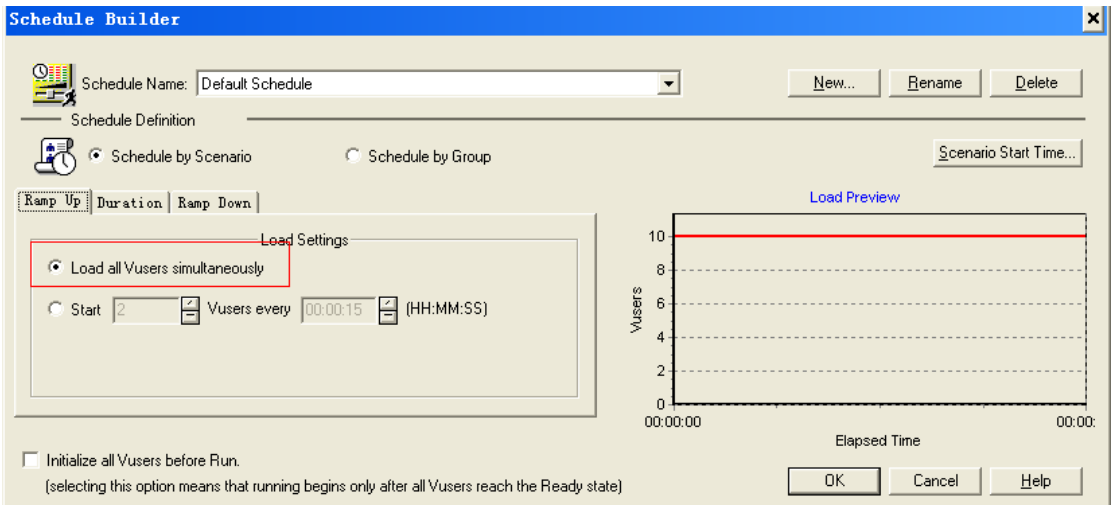


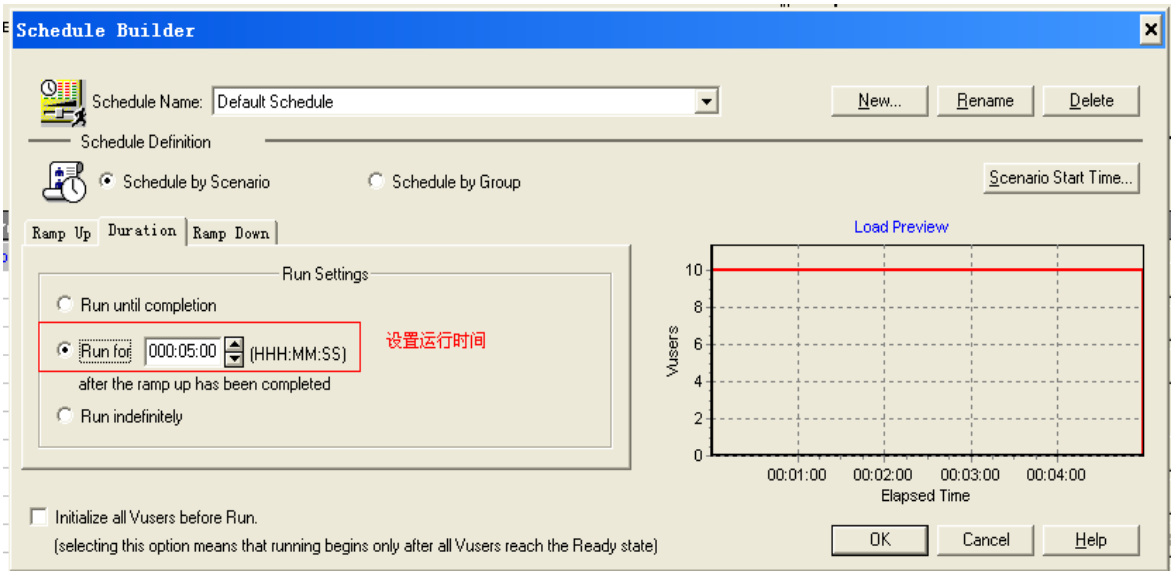
2.5 使用多用户发包

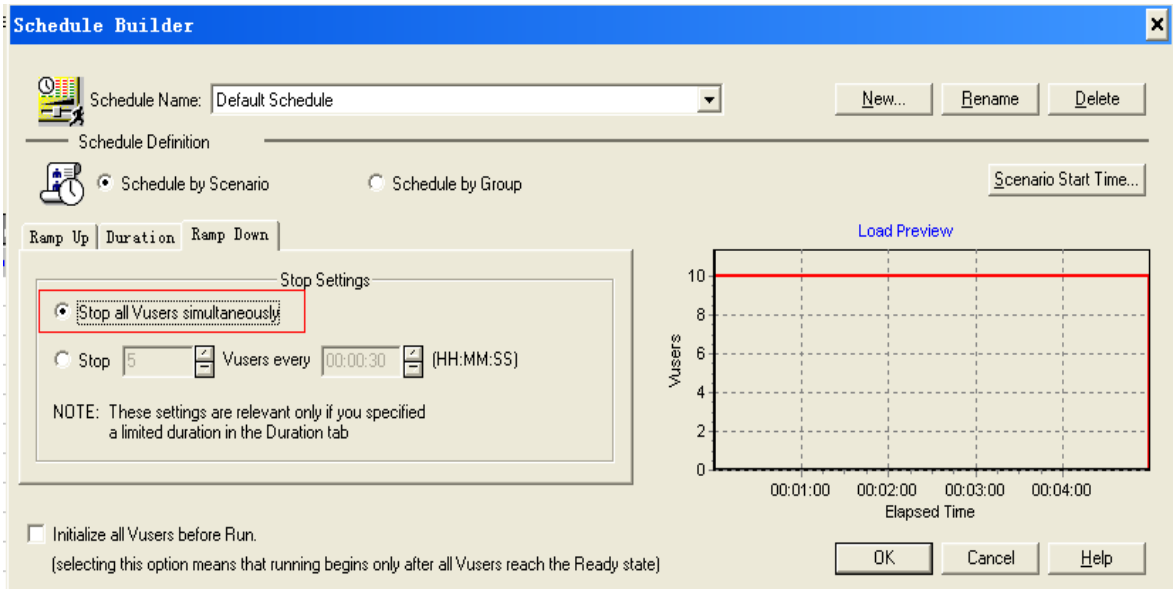
1、 打开“Controller”



2、点击 

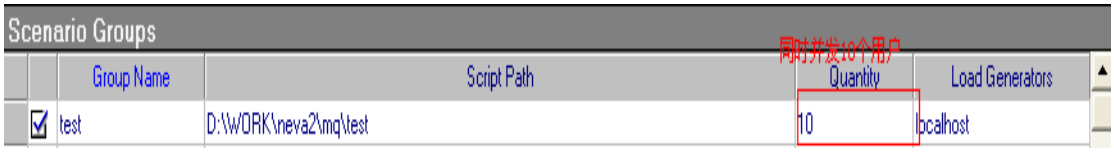






点击“OK”；

2、 设置并发用户数：



3、 然后： 

边运行边查看运行过程中指标。

三、 小结

上面的过程从思路到工具的使用，都进行了详细的描述和截图，目的是为了和大家进行交流。这里只涉及到了部分协议和脚本，实际的应用中，会有更多的情况出现，可能会应用到其他更复杂的协议和脚本，希望此篇能给大家提供一些思路和帮助。

还是对这个测试过程和方法进行一下总结吧：

1、 对于很多对同 IP 访问时间存在一些限制的系统，使用 Loadrunner 的 ip 欺骗是一种很好的方式，也是最简单的方式。因为如果自己编写代码实现，将会涉及到底层的网络编程，一般的测试人员能力可能不及；

2、 如果对于同 IP 访问时间无限制的系统，如果有能力的话，可以用 c 或 java 编写多线程程序来实现多并发，能更加灵活的实现测试需求。而用 Loadrunner 来实现的话，将涉及到 buffer 参数化的实现，会比较麻烦。

纵观上面的应用，我们应用了多种工具和方法进行测试，目的就是要逐步提高我们的测试水平，提高测试效率。些许经验，共享之，错漏难免，不吝赐教。

LoadRunner 架构

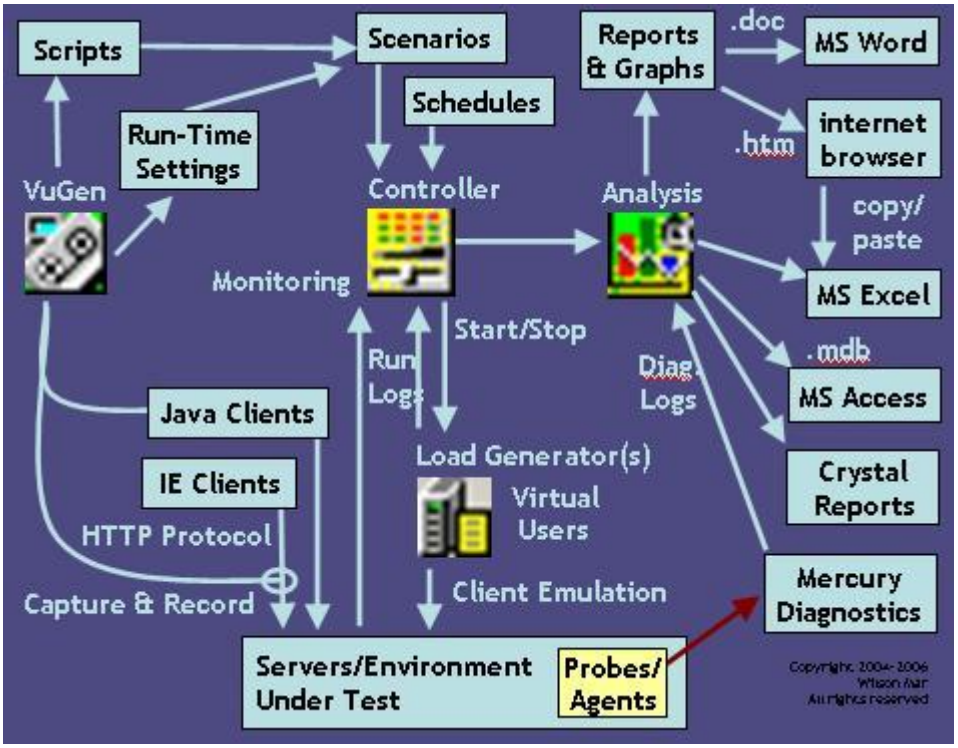
原文地址：http://www.wilsonmar.com/1loadrun.htm

译者：赵岗耀

这里简要说明使用 LoadRunner 的性能测试。这是一个同事在我的专栏上发表的有关 Vu 脚本、性能监测、性能调优和报告的文章。

一、 LoadRunner 架构(www.LoadRunner.Info)

1. 架构概要：



LoadRunner 通过建立虚拟的用户（virtual users）代替真正的用户来操作客户端软件，如 Internet Explorer 使用 HTTP 协议（HTTP Protocol）发送请求到 IIS 或 Apache 网络服务器。

来自许多虚拟用户的客户端的请求由“负载生成器（Load Generators）”所产生，以创建一个对不同服务器下的测试（servers under test）的负载。

这些负载生成器代理都通过 Mercury 的“控制器（Controller）”程序被启动（start）和停止（stop）。

控制器调用已编译“脚本（Scripts）”和相关的“运行时设定（Run-time Settings）”基

于“场景（Scenarios）”的运行来控制负载测试。

脚本使用 Mercury 的“虚拟用户脚本生成器”（命名为"V U Gen"）来制作，它生成由虚拟用户执行的 C 语言的脚本代码，捕获（capture）在互联网应用的客户端和服务器之间的网络流量。

在 Java 客户端（Java clinets），VuGen 捕获挂接在客户端的 JVM 内的调用。

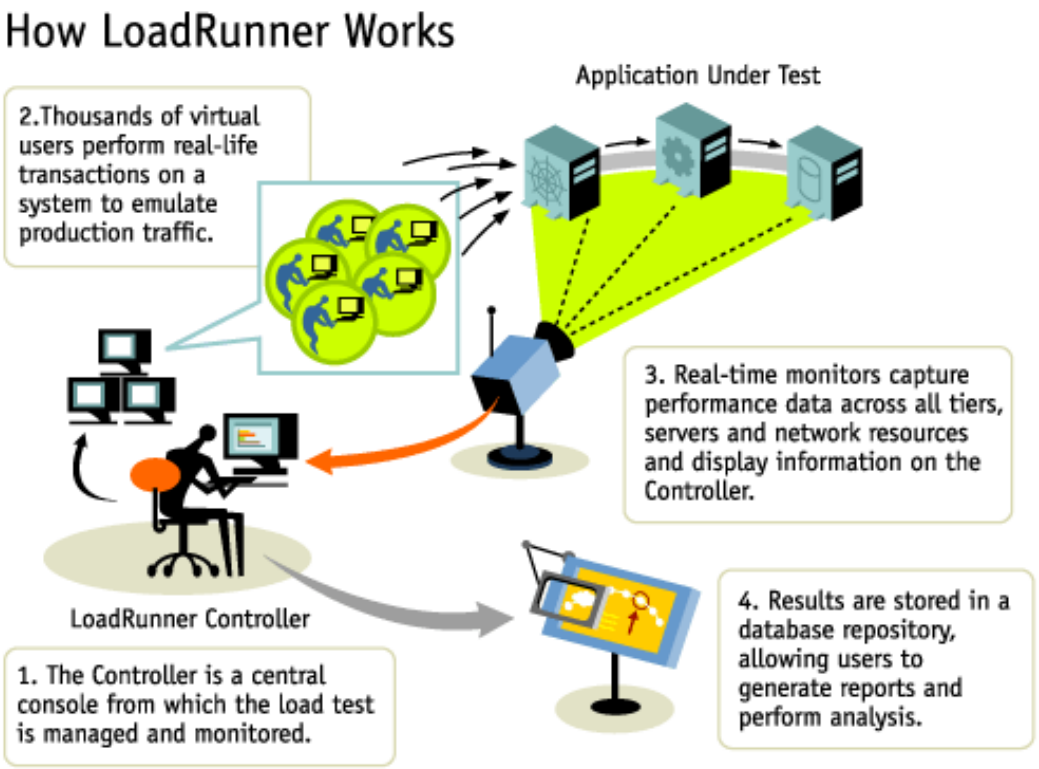
在运行时，每台机器的状态由控制器监控（monitor）。

在每次运行结束时，该控制器组合其监测日志（logs）与负载生成器所得日志，把它们提供给“分析（Analysis）”程序，然后可以创建运行结果的报告（reports）和图表（graphs）为 Microsoft Word，水晶报告（Crystal Reports），或一个 HTML 网页浏览器（browser）。

每个分析程序所生成的 HTML 的报告页，包括一个到一个文本文件中的结果的链接，微软 Excel 可以打开这个报告页执行额外的分析。

在每个运行中发生的错误储存在一个数据库中，该数据库可以使用 Microsoft Access 来读取。

2、 虚拟用户（Vusers）



不像 WinRunner 工作站，其模拟单个用户使用客户端，LoadRunner 可以模拟成千上万个虚拟用户。

负载生成器由 VuGen 脚本控制，使用与测试客户端相同的协议，发出非 GUI API 调用。但 WinRunner GUI Vusers 可以从一台机器上运行，在只有一个 GUI 用户测试的客户端，模拟按键、鼠标点击、和其他用户界面的行为，除非 LoadRunner 的终端服务管理器（Terminal Services Manager）启用终端服务器代理（Terminal Server Agent），并且登录到终端服务客户端（Terminal Services Client），管理远程计算机。

在运行时间，线程 vusers 共享一个共同的内存池。因此，线程支持每个负载生成器有很多的 Vusers。

所有负载生成器上 Vusers 的状态从“运行（Running）”开始，然后在运行脚本的初始段后转到“准备（Ready）”状态。在运行通过（passed）或失败（failed）后，Vusers 的结束状态是“完成（Finished）”。当负载生成器超载（overloaded）时，Vusers 自动变成“停止（Stopped）”状态。

不需要额外的许可（license）来监控标准 web（HTTP）服务器（Apache、IIS 和 Netscape）。

3、 产品版本

08 年 2 月可用的 9.10 版本的安装程序，扩充后是 2.31 GB。但是，英文版安装后创建的文件夹是 931MB。

版本 8.1 功能包 4 修补安装程序 LR81FP4P136.exe 大约 7,786,800 字节，作为文件版本 8.1.4.0（Build：1735）签发于 2007 年 1 月 2 日，是记录版本：1290。

版本 8.1 功能包 4 安装程序 LR81FP4.exe，大约 194,644,720 字节，作为文件版本 8.1.4.0（Build：2249），签发于 2006 年 12 月 15 日，是记录版本：1289。这需要一个到 MS.NET 2.0 客户端的升级。

版本 8.1 功能包 3 安装程序 LR81FP3.exe，大约 116,601,240 字节，作为文件版本 8.1.3.0（Build2085），签发了关于 2006 年 6 月 18 日。它安装（作为一个项目在您的开始>程序文件）Microsoft WSE（Web Services Enhancements）2.0 SP3，为运行.NET Framework 1.1 的系统部署安全策略。

8.1 版本 2005 年 10 月可用。在 VuGen 中它增加了一个“工作流程的视图（Workflow View）”，“工作流程向导（Workflow Wizard）”，和一个内存泄漏（memory leak），这些被整理成一个补丁，自 2005 年 12 月可下载。它重新命名 VuGen 的“执行日志（Execution Log）”为“回放日志（Replay Log）”。

8.0 版 2004 年 8 月可用。它在运行时设置（Runtime Settings）中增加了“其他属性

（Additional Attributes）”。它也增加了（需要额外的费用）诊断和调整的能力，使事务衰减（Transaction Breakdown）衰减横跨不同的服务器层的事务时间，这些服务器服务于各种事务层（Web 服务器、Oracle 11i＆PeopleSoft 8 应用服务器，数据库）。它分离 SQL 时间为执行、解析和达成时间。

7.8 版本功能包 1 增加了对 Windows XP 的支持。

7.8 版本 2003 年 9 月可用。

6.5 版 2000 年 6 月可用，提供了新的“TurboLoad”技术——一种全新的回放引擎，使用一个单一的操作系统线程来运行数以千计的 vusers。

6.0 版每个用户使用一个单独的线程，这比 6.5 需要将近 10 倍以上的 I/O 和 CPU 周期。

注意：在 Mercury Interactive 公司的律师要求把他们移除后，使用在这里的文件的链接被移除。在网上的 PDF 文件的页码与相同标题的纸质文件的页码，是不同的（有更多的页）。

虽然版本 9.10 现已安装在“HP”下，“Mercury”仍然保留在 Program Files\Common Files 下，它的\TDAPI\Client 文件夹中包含文件 TDCIntui.dll 和 tdclient.dll。

这些文件夹，如隐藏的文件夹 C:\Config.Msi，Macrovision 文件夹（在文件夹 Documents and Settings\All Users.WINDWS\Application Data 内），以及其他许多文件在卸载后仍保留。

卸载后，超过三千条目还留在 Windows 注册表中。

4、 应用组件的需求

LoadRunner 使用 4 个具有不同的系统需求的可执行文件

应用产品		进程外形名称	V9.0	V8.0ImgKB	文件大小
	Launcher	LRLauncherApp.exe	15,840	16,288	n/a
	Virtual User Generator	VuGen.exe	23,980	12,436	2,334,769
	Controller with On-Line Monitors	wlrun.exe	61,312	13,076	5,681,215

	Load Generator Agent	magentproc.exe	3,336	3,236	
		magentservice.exe	3,496		65,536
		mdrv.exe	-		
	Analysis	Analysisui.exe	64,460	13,132	6,058,496
	Tuning Console	protune.exe	-		3,403,833

5、 控制台程序

perl5.8.0.exe	解释程序	20,535
regtlb.exe	注册批次自动化类型库	30,720
sed.exe	GNU sed （gsed）版本 2.05	55,296
wdiff.exe	比较文本（text）文件	197,632

Alex Arbitman 的 LR 7.8 Footprints.xls 报告了运行时每进程和每线程的 Web 需求。

6、 使用 Windows 远程桌面连接

对于 LoadRunner 来说，远程桌面连接（Winodows XP 附带的终端服务的一部分）不象远程管理员那样可靠。

在测试时，为了在未连接成功的时间内保持 Windows 的远程桌面连接会话，每台机器上的终端服务应配置如下：

单击开始，指向程序（或控制面板），管理工具，并选择终端服务配置。

在文件夹树中单击连接文件夹，打开它。

右键单击 RDP-Tcp 并选择属性。

单击会话（Sessions）选项卡。

确保“不考虑用户设置（Override user settings）”选中。

设置闲置的会话限制（Idle session limit）为最高的 2 天代替默认的 2 个小时。

单击应用（Apply）。

单击确定（OK）以确认消息“配置更改已改变系统注册表;不过，正作用在 RDP-Tcp 连接上的用户会话将不会改变” 。

确保当您这样做时，不违反您的公司的安全策略。

终端服务器只允许同时存在两个连接。从一个会话上断开时，不要点击远程桌面窗口上的“X”，只要按一下开始并注销。

7、 LR 的安装和配置

HP 公司的 LoadRunner 的支持是 OpenView 支持的一部分。

其中安装 LoadRunner 时常见的问题：在 Windows 2003 ＆ XP SP2 中有一个 DEP（数据执行保护（Data Execution Prevention））功能，它阻止 VuGen 录制。进入控制面板，系统->高级选项卡，性能区域的“设置”按钮，数据执行保护标签，并新增客户端程序，或选择“只为基本的 windows 程序和服务开启 DEP”。必需重新启动。

我建议你把下载的 LoadRunner 的安装文件和补丁放到一个单独的媒体，如 CD 或 USB 驱动器中。然后标识这些文件为只读。

在调用安装程序之前，禁用您的防病毒软件（Symantec、McAfee 等）。

如果您正在运行 Intel 的芯片，通过关机后进入 BIOS 禁用 Intel 超线程技术。

病毒检测引擎可能会发现，程序 regtlb.exe（注册/未注册类型库（registers/unregisters type libraries））中包含一个“病毒”，他们称之为“Backdoor.Win32.PoeBot.15872”。 由病毒去除程序自动修复将破坏这些文件。

为了避免 LR 9.x VuGen 在 Windows XP SP2 和 Windows 2003 上的录制的问题，打开开始 >控制面板 >系统，高级选项卡，单击性能设置。在性能选项的数据执行保护选项卡上，选择“只为基本服务的 DEP”，然后重新启动机器。

LR 盒中配有两个 CD 和安装手册。单独的安装手册用于控制器（Controller）和分析（Analysis）模块。

Windows CD 自启动到初始屏幕。

通过选择“完整安装（Full install）”，你只可以安装一个单一的组件（如 VuGen），但

是，选择“自定义（Custom）”选项可以检查具体要安装的组件。

由于 v8.0 有一个奇怪的错误，你做这一点之前，首先安装负载生成器，然后返回再安装“自定义”组件。

Unix 的 CD 只能在 Unix 机器上安装负载生成器（不能安装控制器或 VuGen），因为控制器和 VuGen 只能运行在 Windows 机器上。

零填补机器命名为 t001、... t010 等。LR 控制器通过命名 t1、t2、... t10 为 t1、t10、T2 来分别机器。

7.1 程序文件的位置

该 LoadRunner 的安装程序添加文件在程序文件，Windows 文件夹，和 Windows 注册表中，这些在卸载期间不会被删除。

如果你收到“许可侵犯”的讯息，您需要获得来自 HP 的支持——一个为期一天的授权码进行安装。

如果您在一个 64 位机上安装 LoadRunner，那么，Program Files（x86）是默认的文件夹。

不同版本的 LoadRunner 将安装到不同的位置：

LoadRunner 9.1 可执行文件将安装到文件路径：

“C:\Program Files\HP\LoadRunner\bin”

LoadRunner 8.1 和 9.0 的可执行文件都将安装到文件路径

“C:\Program Files\Mercury\LoadRunner\bin”

LoadRunner 8.0“stutters”，它安装到其默认文件路径

“C:\Program Files\Mercury Interactive\Mercury LoadRunner\bin”

注意：即使 8.0 使用一个不同的文件夹，先前版本创建的文件夹仍然需要先移除后，才能安装。

LoadRunner 7.8 可执行文件将安装到文件路径

“C:\Program Files\Mercury Interactive\LoadRunner\bin”

在 Java 环境下工作时，如 KB article 11878 所建议，不是放置这些默认的安装文件夹到一个无空格路径（如 C:\LR78），只需使用相当于 DOS 8.3 的文件名：

对于 LoadRunner 8.1: C:\PROGRA~\MERCUR~1\LoadRunner\lib

对于 LoadRunner 8.0: C:\PROGRA~\MERCUR~1\MERCUR~1\lib

为了尽快到达该 LoadRunner 安装文件夹，创建一个名为“LR91”的环境变量，使您可以使用快速命令，如

cd %lr81%

为了尽快到达这个文件夹，在包含该文件夹的 cmd 的默认 C:\根文件夹中，我创建了一个批处理文件，命名为“L.bat”：

cd \Program Files\Mercury\LoadRunner\bin

pause

我给该文件在我的桌面上创建了一个快捷方式，并将它拖到 Windows start 上，这样我可以从任何地方点击进入该文件夹中。暂停命令确保命令窗口不自动消失。另外，在一个命令窗口中，我可以只需输入“L”，然后按下 Enter。

7.2 **Windows 文件夹中的文件**

在文件 wlrun.ini 中，这个特定的路径作为 M_ROOT 贮藏在[ProductEnv]下。当该文件被移到 LR Config 文件夹中时，该文件是在 C:\WINNT（或 C:\Windows）文件夹中直到 v9.10。

在安装过程中，C:\WINNT（或 C:\Windows）文件夹也保存特定的维护码（Maintenance Number：mpn），作为一个参数，命名为“LoadRunner 序列码（LoadRunner_SerialNumber）”（如 1234-1234567890），存储在 mercury.ini 文件中。

7.3 **开始（Start）菜单**

从 LR9.0 开始，安装程序把链接加到如下最常用的程序中

Start> Programs > LoadRunner

在这之前，LR 安装程序把链接加到如下最常用的程序中

Start> Programs > Mercury Interactive > LoadRunner

然而，一些被安装的程序不方便列在这儿，如

从 v7 开始，LoadRunner 通过规定安装 10 天内提供许可密钥，防止软件盗版（很象 Microsoft 对 Windows XP 开始做的一样）。Mercury 基于每部电脑上产生的主机 ID，生成其许可密钥。

对于 v7.x，生成一个 HostID 密钥（如“XCCWJU-APBE-BYDS”）按下

Start〉 Programs > Mercury LoadRunner > LoadRunner > License tab

安装之前，密钥可以从程序 licidgenerator.exe 得到，（注册后）它的 lm70.dll 从安装 CD 文件夹\lrunner\lm70.nt\bin 或\lrunner\setup\lm70.nt\bin 获得。

7.4 示例应用程序/协议

运行示例安装程序移至文件夹

C:\Program Files\Mercury Interactive\Mercury LoadRunner\

复制链接位置放在“Program to record”栏下面：

协议 Protocol	服务器 Server	客户端程序 Client Program	参数 Parameter	备注 Notes
Web	WebTours\ StartServer.bat	http://localhost:1080/ mercuryWebTours		
COM/ DCOM	（操作系统）	samples\bin\frsui.exe		
Winsock	sockfrs.exe	samples\bin\flights.exe	Winsock WinSockWeb	
ODBC	(MS Access)	samples\bin\flights.exe	ODBC_Access	
CORBA	samples\CorbaSamples\server.cmd & samples\CorbaSamples\server.bat	samples\CorbaSamples\client.cmd & samples\Corbasamples\clientrecord.cmd		
RMI	samples\RMISamples\ server.cmd & samples\RMISamples\ server.bat	samples\RMISamples\ client.cmd & samples\RMISamples\ clientrecord.cmd		

据 CPT11877.doc，对于每个具体的 LoadRunner 的版本（7.6、7.8 FP1 或 8.0），JDK 1.5 用户需要联系 Mercury 支持来获取补丁。否则，您将获得这些信息：

```
Error: Failed to find javac.exe Java Compiler in Path and JDK installation folder in registry. [MsgId: MERR-22981]
Error: Failed to compile the Actions.java file. Please add the \bin to the path and try again. [MsgId: MERR-22996]
Warning: Extension java_int.dll reports error -1 on call to function ExtPerProcessInitialize [MsgId: MWAR-10485]
Error: Thread Context: Call to service of the driver failed, reason - thread context wasn't initialized on this thread. [MsgId: MERR-10176]
```

在 VuGen 本地机器上的数据源（ODBC）中，Java 示例应用程序使用用户 DNS 表中，带有 Microsoft Access 驱动程序（*.mdb）的“flight32lr”用户数据源。

此外，示例 Java 服务器必须被预先操作来开启客户端。这是用“samples\RMISamples\server.cmd”来完成：

```
set lrpath=C:\PROGRA~1\Java\jre1.5.0_02\bin;C:\PROGRA~1\MERCUR~1\MERCUR~1\classes
set
lrclasspath=C:\PROGRA~1\MERCUR~1\MERCUR~1\classes;C:\PROGRA~1\MERCUR~1\MERCUR~1
\classes\srv;C:\PROGRA~1\Java\jre1.5.0_02\lib\rt.jar
set flightRmi=%~dp0;%~dp0RmiSamples.zip
set classpath=%lrclasspath%;%flightRmi%;C:\PROGRA~1\Java\lib\rt.jar;.;%classpath%
set path=%lrpath%;.;%path%
cd %~dp0
start java -Djava.security.policy="%~dp0RmiFlights.policy" RmiFlights.Server
```

注意我已加到默认示例中的 LoadRunner class 文件的位置。他们被预先放置在现有的 classpath 下。

请注意，在文件路径中没有空格。

Zip 文件相当于 Unix 系统中的一个 JAR 文件。

不要删除黑色命令窗口，因为 Java 服务器在内运行。

CORBA 和 RMI Java 客户端被 Windows 命令调用，来启动 java.exe 程序。这个“samples\RMISamples\client.cmd”文件中包含：

```
set lrpath=C:\PROGRA~1\Java\jdk1.5.0_02\bin;C:\PROGRA~1\MERCUR~1\MERCUR~1\classes
set
lrclasspath=C:\PROGRA~1\MERCUR~1\MERCUR~1\classes;C:\PROGRA~1\MERCUR~1\MERCUR
~1\classes\srv;C:\PROGRA~1\Java\jdk1.5.0_02\lib\rt.jar
set flightRmi=%~dp0;%~dp0RmiSamples.zip
set classpath=%lrclasspath%;%flightRmi%;C:\PROGRA~1\Java\jdk1.5.0_02\lib\rt.jar;.;%classpath%
set path=%lrpath%;.;%path%
cd %~dp0
start java RmiFlights.main
```

请注意，该 RmiFlights.main 类文件名称被传给 Java，由它来加载。

用 VuGen 录制 Java 时，不同的命令—如示例 clientRecord.cmd—需要被调用，因为

VuGen 需要在 JVM 沙箱内部被调用：

```
set flightRmi=%~dp0;%~dp0RmiSamples.zip
set classpath=%flightRmi%;%classpath%
cd %~dp0
start InvokeVugen.exe
```

JDK 的位置需要在 Windows 环境变量 PATH 中加以指明，以避免此讯息：

```
Error: Failed to find javac.exe Java Compiler in Path and JDK installation folder in registry. [MsgId: MERR-22981]
```

VuGen 的“Java Vusers”只能作为 Single Vuser 模式（不是 multi-vuser）运作。

不同于 web“开始录制”，Java VuGen 脚本在 Actions 部分引用 Java 功能。“vuser_init”和“vuser_end”活动与 Java VuScripts 内部是不相关的。

在 cjhook.ini 文件内，指定哪些 Java 类可以挂在其[EXC_SYSTEM_CL]节。在[SYSTEM_CL]节中指定的 Java 类没有被挂上。

在 LR\bin 文件夹中的 user.hooks 文件是一个普遍的格式，不能被用在内。它需要被复制。

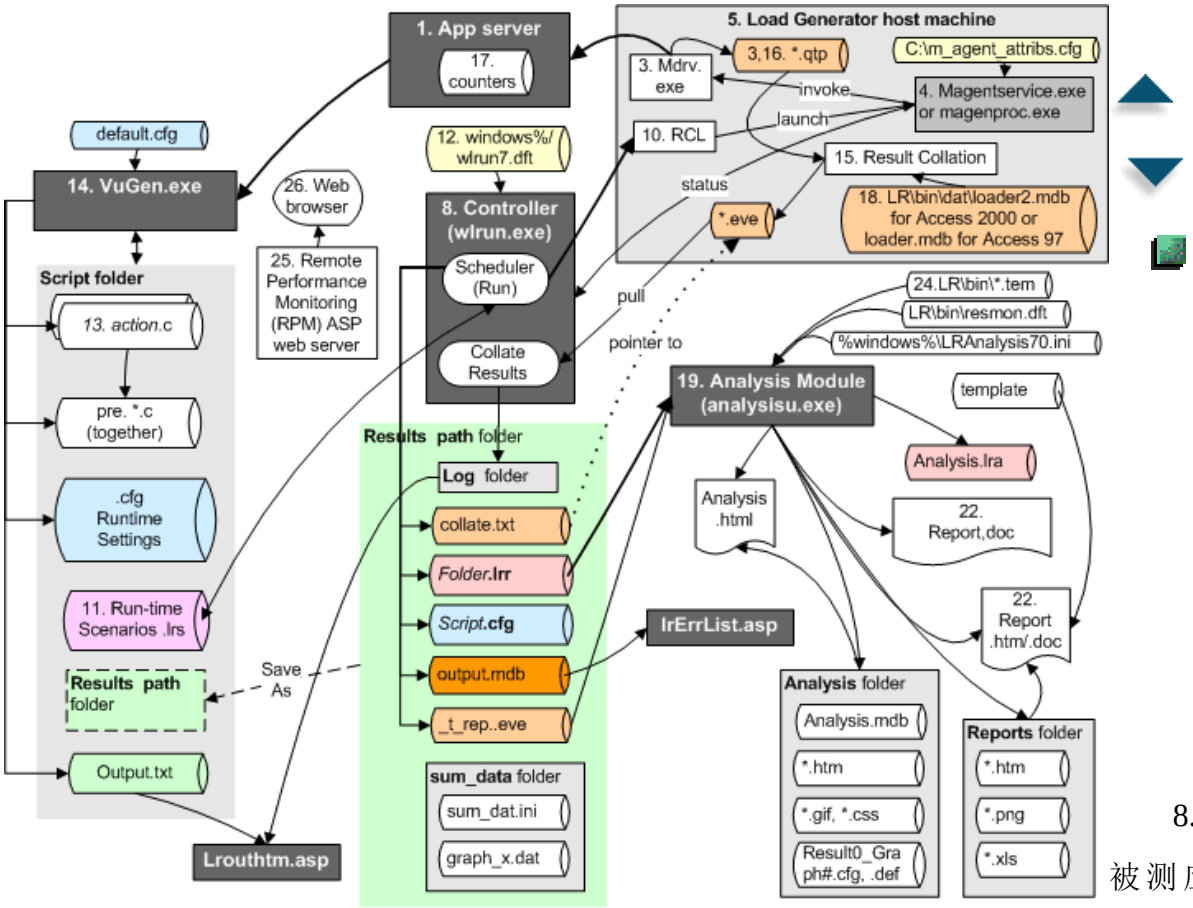
7.5 卸载

卸载 LoadRunner 时，你必须用与原始安装时所使用的相同的 Windows userID 登录！如果您使用不同的 userid，卸载将只删除包含“miuninst”文件的 dat 文件夹。

不像 Microsoft Office 应用程序，Mercury 没有程序化的单独组件，可以有选择地卸载它自己。

8、 LoadRunner 内部架构

这个 Visio 2002 文件是基于 LR8.0 的



8.1 被测应用服务器通过 2 被置于重载之下

8.2 驱动进程 mdrv.exe（多线程驱动进程（Multi-threaded Driver Process））和 r3vuser.exe，其效仿应用程序客户端，例如 Internet Explorer 网页浏览器。它执行 3 个主要动作：

Kli> cpp（C 语言预处理器（c language pre-processor））

cci（C 预编译），其创建一个带有 ci 文件的的文件，同时使用驱动程序 执行被测得协议技术。

运行可以被引用，通过从 Windows 批处理脚本调用 mdrv.exe 来“默默”运行。

Mdrv 可以自动停止加载 Vusers，因为他们和 vusers 通信并监测 Windows 负载生成器上机器的 CPU 使用率。

一个独立的 JVM 被基于 Windows 的机器上的每个基于 Java 的 Vuser 实例化。Java Vusers 在 Unix 平台上不被支持。

8.3 虚拟 Vusers 作为组被引用（虚拟用户的逻辑集合在一特定负载生成器的机器上运行相同的脚本）

8.4 通过代理（agents）（3,900K magentproc.exe）作为一项服务或作为一个过程运行

8.5 在负载生成器客户端机器上。

8.6 每个机器托管代理在一个.qtp 文件中维护一个执行日志（Execution Log）。

8.7 当日志被启用时，代理也为每个 Vuser（由 Vuser 组隔开）在结果文件夹内创建一个有序的日志文件。

8.8 在执行期间，这个文件显示 LoadRunner 控制器的机器（Controller machine）上的视图（view）> 显示输出（Show Output）窗口。

8.9 在一个预设定的延迟下，运行于控制器的机器上的调度程序（Scheduler）通知代理（经由 Windows 端口 54345 或动态的 UNIX 端口）开始测试会话场景（test session scenarios）。控制器（wlrun.exe）随请求发出一个场景文件的副本。

8.10 通过每个负载生成器的机器上的远程代理发报（Dispatcher）过程（以前称为远程命令发射器（Remote Command Launcher：RCL））, 代理被启动。

8.11 每个代理参阅场景（.lrs）定义文件，以确定哪些 Vuser 组和脚本在主机上运行。

这意味着控制器可以从 DOS 批处理（.bat）文件（最好是根驱动器下的短名称）启动：

```
REM Start Controller:
SET M_ROOT=C:\Program Files\Mercury Interactive\LoadRunner\bin
cd %M_ROOT%
wlrun.exe -TestPath D:\Dev\Dev1.lrs -port 8080 -Run -DontClose
pause Press Ctrl-Z to keep this window or
```

祈求自动包括-Run 参数和手动按下“Start Scenario”是一样的。这不是一个好主意，因为你可能要决定从预运行中整理文件或要改变输出文件夹。

这个假设条件是，该系统环境 PATH 变量被更新，以包括 LoadRunner 被安装的位置。

8.12 在 Windows OS 文件夹（Windows 2000 是 WINNT，Windows XP 是 Windows）下，该控制器用文件中的参数值被调用。Windows 文件夹被使用，是因为 LoadRunner 被设计成，在一个时间对一台机器只有一个控制器实例运行。

为了快速在几个应用程序之间转换，在控制器内部，在 LoadRunner 的 ini 文件工作

后，保存它的一份副本，然后在执行 wlrun 之前，使用记事本制作一个批处理文件来复制 ini 文件的应用特定版本，对于应用 XXX 复制活动的一个文件的例子：。

```
copy %LRDir%/config/wlrun7-XXX.ini    %LRDir%/wlrun7.ini
copy %LRDir%/config/wlrun7-XXX.dft    %LRDir%/wlrun7.dft
```

v9.0 之前：

```
copy %WinDir%/wlrun7-XXX.ini    %WinDir%/wlrun7.ini
copy %WinDir%/wlrun7-XXX.dft    %WinDir%/wlrun7.dft
```

一些您可能会想要改变的默认设置：

- 为长期运行，在 wlrun7.ini 文件[output]部分，MaxNumberOfOutputMessages=从 10000 到 100000。这限制了存储在数据库中输出信息的数量。
- MaxOutputUIRowsToShow 限制显示在控制器的输出窗口中消息/错误（行）的数量。
- 在 LoadRunner Program Files dat\protocols 的文件夹内 QTWeb.lrp 文件的[Vugen]部分，新增条目 MaxThreadPerProcess=5 限制由每一个负载生成器 mdrv.exe 进程所管理的线程的数量。

无论控制器内的值是否改变，储存在 wlrun5.ini 和 wlrun7.dft 文件中，defaultscenariodir，defaultscriptdir，defaultresultdir，和[Recent File List]的值都被更新。

8.13 每个 Vuser 获取的 actions 区块

8.14 用 LoadRunner 的 VuGen.exe 创建的 Vu 脚本定义。当这一计划被调用时，它在 Windows 文件夹中存储一个 comparamui.INI 文件，以在文件历史“[LastTablesUsed]” 和使用菜单选项 Insert>New Parameter>Dates 指定的[paramdialogdates]下保存。

VuGen 在 Windows 文件夹中存储和检索 vugen.ini 文件。当使用 Java 时，使用额外的调试选项：

```
[DynaDlg]
JavaLevel=3
```

```
[Editor]
OLDEDITOR = 1
```

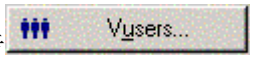
当使用 VuGen 8.1 内的 8.0 脚本时，添加到 vugen.ini：

VuGen 在 LR 文件夹 template/qtweb 打开 default.cfg 和脚本文件。

VU 脚本可以使用从外部参数文件获得的变量值来编码成脚本。

8.15 在一个运行中，执行的结果储存到一个结果文件夹中。

我更喜欢设置结果设置（Results Settings）为“为每个场景的执行自动创建一个结果路径”。这意味着，当我开始一个场景运行时，LR 将增加结果名（Results Name）的名称。例如，一个“Res11”值会自动递增到“Res12”或有时是“Res11-1”。

8.16 在每个结果文件夹中，“日志”文件夹被自动创建，每个组（group）包含一个日志文件。运行后，查看来自控制器内的日志文件，单击 

，然后在一组上按一下鼠标右键，选择“显示 Vuser 日志”。

8.17 当一个场景运行时，监测器维护每个主机上的本地计算器。

8.18 运行过后，“整理”进程获取.eve 和.lrr 的结果文件，并在结果文件夹中创建一个临时的.mdb（MS-Access）数据库。

处理大型的结果文件时，为了防止错误，使用 MSDE（微软 SQL 桌面引擎（Microsoft SQL Desktop Engine））。不要在 LoadRunner 7.8 CD 上的 Add-in 文件夹中安装它，这是过时的 SQL7。下载 MSDE 2000 Release A，其中包括了 MSDE 2000 Service Pack 3a 和 MDAC 2.7 SP1a，在任何 Windows 机器被分析器使用。提取文件和共享该文件夹。打开一个命令窗口，运行命令，例如：

```
setup SAPWD="StrongPassword" INSTANCENAME="LR" SECURITYMODE=SQL
DISABLENETWORKPROTOCOLS=0 /L*v path to log file
```

使用 Windows 资源管理器，共享数据文件夹。

然后在分析器选项>数据库选项卡上，使用 8.3 无空格名称（用 DOS 命令 DIR/x 确定）：

- 输入上述指定 SAPWD 密码。
- 逻辑存储位置：\\loadclient02\Data（您共享的文件夹）

- 物理存储位置：C：\progra～1\MICROS～1\MSSQL\Data（不是 C：\Program Files\Microsoft SQL Server\MSSQL\Data）
- 点击“测试参数”。（这需要几秒钟）

8.19 分析模块（8,320k analysisu.exe）

8.20 使用来自.mdb 数据库的数据生成分析图表和报告。

8.21 来自每个场景运行的 LoadRunner 结果文件 results_name.lrr - 也称为分析文档文件 - 被分析程序阅读来显示百分图。

8.22 默认情况下，LRReport 文件夹在测试分析师的本地机器上 My Documents 文件夹中被创建，来存储分析会话文件。

8.23 在 HTML 中，他们可以选择性地被格式化。

8.24 其格式是由一.tem template 文件控制。

8.25 可选地，对于 LoadRunner 7.8 的 Mercury 的远端性能监视（Remote Performance Monitoring：RPM）MS-IIS/ASP web 服务器，可以安装在 Windows 2000 Server 上（但不是在 Windows 2003 Server 上），使

8.26 负载测试结果被使用 Web 浏览器察看。

9、 负载生成器代理的过程与服务

在安装过程中，在用户登录设定显示屏：

- 选择“允许无用户登录的虚拟用户在这台计算机上运行”，是指 LoadRunner 的代理将作为一个名为“LoadRunner 代理服务”的系统服务运行。在窗口的执行中，这个图像名为 magentservice.exe。

为了更好的安全性，指定一个单独的服务帐户用户 ID 和密码，使他的权限可以被限制。

安装完毕后，辨别它在运行，到 Windows 的服务列表：

在 Windows 2000 上，转到开始->控制面板->管理工具->服务。

在 Windows NT4，转到开始->控制面板->服务

您还需要进入服务列表更改密码，或取消设置服务为“自动”。

- 选择“手动登录到该负载生成器”，是指 LoadRunner 代理将作为命名为 magenproc.exe 的进程来运行。这种做法意味着，在您每次开机机器，您需要从 LoadRunner 的\launch_service\bin 文件夹调用负载生成器：magentproc.exe 。

你可以在显示屏右下角的 Windows 任务栏上，通过“卫星天线”图标分辨它的运行。

为了重新启动后自动启动以获得代理，为它在程序\启动文件夹中创建快捷方式。

作为一个进程，Windows 操作系统限制来自运行的 GUI（WinRunner，QuicktTest Professional 等），或象 GUI 一样的脚本（Citrix，SAPGUI 等）的代理服务。因此，如果远程代理调度（Remote Agent Dispatcher）是作为一种服务而不是作为一个过程，被安装在负载生成器的机器上时，GUI 和 SAP Vusers 无法运行。

安装完毕后，从作为一个服务运行 LoadRunner 切换为作为一个过程运行：

cd \PROGRA~1\MERCUR~1\MERCUR~1\LAUNCH~1\bin

magentservice.exe -remove

安装 LoadRunner 作为一项服务：

magentservice.exe -install

这些命令不会引起回应信息。但他们把 m_agent_attribs.cfg 文件放到了负载生成器的 C：\根文件夹。

如果您没有管理员权限，并尝试改变用户 ID：Admin 和密码：Admin，您会看到消

ERROR: "29972:- Failed to reset launcher status call back function reason:no monikor was passed.

对 UNIX 机器，代理是通过编辑 LoadRunner 根文件夹下 dat 文件夹中 br_inch_server.cfg 文件而被配置。

9.1 确认＆确保代理准备

在控制器的负载生成器对话框中，点击“连接”按钮后，您应该可以看到“准备就绪”（对高亮显示的代理）。

如果你得到消息“无法连接到代理。超时的命令行被执行后，负载生成器没有回应”，重复这些命令。

如果这仍然没有达到“Ready”状态，转到负载生成器客户端机器上的 Windows 服务，并删掉，然后在重复上述命令之前，删除“LoadRunner 代理服务”。

为真时设置 FireWallServiceActive 为 1，为假时设置为 0。运行 bin/agent_config。

9.2 使用网络驱动器映射

如果几个负载生成器需要存取相同的物理文件，而不是每一次他们变化时，必须记得复制文件，每一个负载生成器可以使用映射驱动器参考一个共同的文件夹。但由于驱动器映射与特定的用户关联：

- 作为用户登录负载生成器，负载发生器将使用
- 打开 Windows 资源管理器，在工具下，选择映射网络驱动器，并创建一个驱动。这样可以节省时间和麻烦，使一致的驱动字母通过负载生成器，所以有些组织为特定的位置保留特定的驱动字母。
- 在服务（访问控制面板，管理任务）内，打开 LoadRunner 的服务，
- 点击“Login”标签。
- 指定负载生成器服务将使用的用户名和密码的。（如果本地网域是 userid，一圆点出现在用户名的前面） 。
- 停止并启动该服务。

10、 控制负载生成器和监测通过防火墙

防火墙的目的是通过模块化通信和只允许某些端口的通信来提高安全性，如 HTTP 和 HTTPS 交互的 80 和 443。

默认情况下，LoadRunner 的控制器使用 TCP 端口 50500 将数据发送到在 Windows 负载生成器上的 TCP 端口 54345。

负载生成器通过一个动态端口回发信息。通过 MI 收听器（MI Listener）。

为了避免乞求网络管理员打开更多的端口，在防火墙内每个负载生成器机器上，从开始>程序> ... LoadRunner> 高级设置>代理配置（Agent Configuration）（launch_service\bin\AgentConfig.exe）安装（监测防火墙机器）MoFW/RoWF 代理。选中选项“启用防火墙代理”。

它收集性能计算器数据，并跨越防火墙把它们传送到一个控制器。

LoadRunner 7.8 自动化完成，LoadRunner 7.6 及更早版本需要您进入“代理设置”，来改变安装 LoadRunner 的 dat 文件夹内 mdrv.dat 文件。[FireWall]段应改为含有“FireWallServiceActive=1”。这在工具栏的代理图标上应当引起小小的停止闪光变绿。

br_lnch_server.cfg

MoFW 通过端口 443 与 MI 收听器通信，所以你不能有任何 Web 服务器（如 Apache 的 WebTours，IIS，或 Oracle HTTP 服务器）运行在两个机器上。

Verify whether port 443 actually allows communication by running command and substituting the ip address in: 通过运行命令和取代以下 IP 地址，验证是否在端口 443 上实际上是允许通信的：

telnet 194.194.194.194 443

这应该打开 telnet 窗口。

Unix 的负载生成器使用不能被确定的动态端口。

当从控制器内定义“远程”负载生成器时，单击“Details”打开“负载生成器信息”对话框，在这里您可以按一下“Firewall”标签并选中“启用防火墙”。

11、 在 Windows 及 UNIX/Linux 上的监视器

11.1 监测 Unix 机器

在 Linux 环境下开始运行之前，检查以确保 rstatd 监测器起作用。当服务器变得太忙或重新启动时，如果 rstatd 服务结束，LoadRunner 7.8 并不试图从 rstatd 重新获取信息。工作环境退出并再一次重新初始化控制器。

11.2 用于 HTTPS/SSL 的交互的 MI 收听器

当代理由防火墙后面发送 HTTPS 交互（通过端口 443）时，它使用了“透过防火墙部件监测”和控制器进行通信，通过防火墙以外的 Mercury Interactive 的 MI 收听器（通过端口 50500），使用一个代理的象征名称。通过防火墙监测 Windows 机器使用 TCP 端口 139。

12、 VuGen 录制＆脚本

LoadRunner 的脚本代码从录制 ANSI C 语言语法中获得，在图标视图中由图标表示，除非您点击脚本视图（Script View）。

使用 LoadRunner 7.8，在 Window XP 平台，微软.NET 框架▸\runtime 内，录制运行的应用程序前：

- 在 LoadRunner \Bin 目录中，备份 trpfnc32.dll。
- 在 LoadRunner 7.8 CD 上，进到\Patches\ Trap_for_.net_patch 文件夹中。
- 复制 trpfnc32.dll 到 LoadRunner 的\Bin 目录。

12.1 文件位置

我建议你把所有文件存储到您创建的（映射成一个文件夹）驱动器盘符中。当您随着时间的推移更改机器和驱动，特别是当几个测试人员使用 LoadRunner 时，这使得控制 LoadRunner 的最初默认值更容易，也更容易保持一致。

默认情况下，在脚本开发期间俘获的.htm 和资源文件，被存储在为文件夹 x:\Program Files\Mercury Interactive\LoadRunner\scripts 下的每个脚本创建的一个数据文件夹中。

每个脚本的运行时设置存储在每个 Vuser 脚本的目录中的上述脚本的.cfg 文件中，被 VuGen.exe 和 LR 控制器两个使用。

VuGen 存储和找回 Windows 注册表 HKEY_LOCAL_MACHINE\SOFTWARE\Mercury Interactive\LoadRunner 的键 RecentScripts 的每个脚本。

VuGen 也存储和找回生成的每一个.qtp 文件的位置，Windows 注册表中的键 Test Results\Recent File List。

显示在可用脚本列表中的脚本的最大数是存放在注册表的键 HKEY_CURRENT_USER\Software\Mercury Interactive\RecentScripts\max_num_of_scripts

每个 Vuser 可以申请不同的加权（weighting）来迭代一些比其他更频繁的活动的档案。

12.2 数据文件的位置

默认情况下，数据文件（含 URLs，userids 等）创造在位于脚本的文件夹的一个脚本中。我想把它留在那儿，而非将其移动到对所有的脚本共享的公共的根目录下，因为当您保存或导入脚本时，VuGen 自动复制这些文件。另外，改变了一个公共文件的数据结构的禁用先前版本的脚本。

我喜欢随机存取数据文件，因为这是一个简单的方法来测试来自 VuGen 内部的多个值——我只是再次运行该脚本。不过，由于顺序存取被指定时，VuGen 始终使用第一个值，为了运用多个值，我要在控制器内变更数据文件，或再次运行该脚本。

在.usr 文件内： Type=General-Java 为“Java Vuser”，Type=Rmi-Java 为“RMI-Java”

12.3 浏览器

浏览器类型	浏览器版本	平台
Microsoft Internet Explorer	7.0, 6.0, 5.5, 5.01, 5.0, 4.0	HP-UX, Unix, SunOS,

Mozilla Firefox	1.5.0.6, 1.5.0.4, 1.5, 1.0.7, 1.0.6, 1.0	Win32, Windows, WinNT, WinXP
Netscape Navigator	6.2, 6.0, 4.76, 4.6, 4.5, 4.0	

12.4 运行结果文件

默认情况下，VuGen 在当前的用户名称后，创建一个新的命名用户。例如，一个用户名为“Tester”将在 C:\Documents and Settings\文件夹下，有一个用户命名为 Tester.LOADTEST 的运行结果。VuGen 自动设置 Windows 环境变温 TEMP 为% USERPROFILE%\Local Settings\Temp，使结果被写到该用户的子文件夹\Local Settings\Temp。对用户 Tester 的完整路径会是

C:\Documents and Settings\Tester.LOADTEST\Local Settings\Temp

从每个 LR 运行的结果是储存在“Results”文件夹下它自己的文件夹中。

- 当控制器的结果设置改变，一个.lrr（LoadRunner Results）文件创建在指定的目录 File>Save。在这个文件的[Data Collection]段内，FullData=1 时使能看到是否数据被保存。如果这个值是 FullData=0，LR 将不会寻求运行结果。

当场景启动时，在 UNIX 日期格式下，它被更新了开始时间，例如：

[Scenario]

Start_time=1018300521

- 在控制器机器的为运行指定的结果文件夹中，remote_results.txt（ASCII）文本文件列出为每个主机名称（IP 地址）自动生成的事件（events）（.eve）文件的文件路径，。例如：

[Hosts]

10.0.95.109=d:\temp\brr_BGf.453\netdir\d\ecom\ecom1_s3\res3\10.0.95.109_20.eve

- KB 文章 11367 提到，_t_rep.eve 事件文件中包含双 Vuser 和会合信息。有一个对于每个代理主机的 localhost_1.eve 主事件文件。它们的第一行包含：

28 11 1018300521 2 4627103 22735576 5481115"

- 在上面的例子中的 11 是开始场景的事件代码。
- 1018300521 是开始时间。

- KB 文章 22538 指出，如果这个和.lrr 文件中的开始时间不一样，当用户打开一个分析文件时，一个“致命错误（Fatal error）”会发生。
- vuser.cfg 配置文件包含 VuGen 内运行时间设定（思考时间（think time）、迭代（iterations）、日志（log）、网络（web）的设置。
- vuser.usp 日志文件中包含脚本的运行逻辑，包括活动段（actions sections）如何运行
- 图表的 off_1.def 定义文件描述在线和其他监视器。
- collate.txt 的档案列出结果及分析器整理文件的路径。
- output.mdb 输出的 MS - Access 数据库文件由分析器所创建
- offline.dat 示例数据文件监测信息。
- 日志文件夹中包含的——当启用日志时——res_dir\output.txt 消息文件，该文件在每组一个文件夹下的回放过程产生，在该组中，每个 Vuser 包含一个文件夹。
- sum_data 文件夹中包含图形概要.dat 文件。

如果 lr_end_transaction 命令不严格使用相同的名称作为其响应的 lr_start_transaction 名称，其后所有的活动将处在该事物（transaction）影响下。

如果一个 Web SSL 脚本在控制器下，而不是在 vuGen 下工作，尝试改变控制器的运行时设置参数为“WININET 代替 Socket 回放”。

为了使 VuGen 记录 nca_java_action 和其他对于 Java 对象的功能，应用服务器需要配置成能提供必要的数据。为此，编辑启动 HTML 文件，该文件在 applet 视图启用时被呼叫。修改该行：

<PARAM name="serverArgs ..... fndnam=APPS">

and add the Oracle key "record=names":

<PARAM name="serverArgs ..... fndnam=APPS record=names">

13、 控制器的场景

场景在控制器内被指定

场景概括了 Vuser 组和在运行时间负载生成器上执行的脚本。

手动场景可以分配基于分析师指定的百分比（在负载生成器中均匀分布）的脚本中 Vusers 的总数。

目标导向的场景基于指定事务的响应时间或点击次数/每秒事务（transactions-per-second）（TPS），被自动创建。测试分析师指定脚本中目标的%。

13.1 运行场景类型

这表为每种类型的运行场景总结了典型的设置

菜单	设置	A.速度 Speed	B.争夺 Contention	C.超负荷 Overload	D.时长 Longevity
Controller	# of Iterations	1 only	几次	无穷	
Run Options	输出频率：Sample once every	1 秒	10 秒	1 分钟	5 分钟
Run-time Settings	Vusers	1 only	max. licensed	# below "knee"	
	Logging	For debugging	No		
	Think Time	None			实际随机的
	错误时继续？	No	Yes		
	网络速度模拟	Maximum	No		
	浏览器（缓存）仿真	No	Yes		
	Content Checks	Yes	No		
Schedule	Ramp-up:	同时加载所有 Vusers		Yes	
	运行前初始化？	No	Yes	No	

-	间隔（秒）	4	>4	30 or more	
Tools > Options > Monitors	服务器资源监测器：Data Sampling Rate	3 seconds (Default)			5 minutes

当计划运行直到完成时，一个场景的数量是一个时间运行一次的 vusers 的数目。

当计划运行一段时间，一个场景的数量是同时运行的 vusers 的数目。指定的时间从坡道期后开始，此时，所有 vusers 已进入运行状态。该指定的时间加上获取所有 vusers 进入运行状态所花费的时间，成为一个有限时间运行的总耗费时间。

为了同时运行具体数目的 vusers，达到文件结尾时，设置脚本（在 VuGen 中）中的参数文件为终止。

13.2 场景设置

场景运行时设定使用运行逻辑脚本，另加：

- 如何快的步速启动一个新的迭代（以前的迭代结束后或在一定的时间间隔后__秒）
- 日志 － 高级追踪（AdvancedTrace）
- 思索时间（Think Time）
- 错误时杂项是否继续？lr_error_message 的失败开启事务
- 网速模拟带宽下，选择除“最大可用”以外的其他任何情况时，需要一个单独的 WAN 模拟器许可证的购买，通过客户端和服务器之间减慢，减弱，和不断变换的数据包，模仿 WAN/互联网。具有该许可，您可以选择自订，或这些额定速度：
- 14.4 Kbps（模拟调制解调器）
- 28.8 Kbps（模拟调制解调器）
- 56 Kbps（模拟调制解调器）
- 64 Kbps (ISDN)
- 128 Kbps（双 ISDN）
- 512 Kbps(DSL)
- 浏览器仿真：模拟浏览器的缓存呢？
- 协定的代理或使用 ContentCheck 应用
- 控制器保存了这些设置在结果设置对话框中的指定的.lrr 文件中。

13.3 高级的追踪

如果在脚本运行之前，运行时设置有“高级的追踪”复选框选定，这些行将出现在输出日志：

```
UTC (GMT) start date/time    : 2005-08-02 02:31:30          [MsgId: MMSG-26000]
LOCAL start date/time        : 2005-08-01 21:31:30          [MsgId: MMSG-26000]
Local daylight-Savings-Time: Yes        [MsgId: MMSG-26000]
Some of the Run-Time Settings:    [MsgId: MMSG-27142]
Run Mode: HTML        [MsgId: MMSG-26845]
Download non-HTML resources: Yes        [MsgId: MMSG-26845]
Verification checks: No        [MsgId: MMSG-26845]
Simulate a new user each iteration: Yes        [MsgId: MMSG-26845]
Non-critical item errors as warnings: Yes      [MsgId: MMSG-26845]
WinInet replay instead of Sockets: No        [MsgId: MMSG-26845]
HTTP version: 1.1        [MsgId: MMSG-26845]
Keep-Alive HTTP connections: Yes      [MsgId: MMSG-26845]
Max self Meta refresh updates: 2        [MsgId: MMSG-26844]
No proxy is used (direct connection to the Internet)      [MsgId: MMSG-27171]
DNS caching: Yes        [MsgId: MMSG-26845]
Simulate browser cache: Yes        [MsgId: MMSG-26845]
Cache URLs requiring content (e.g., HTMLs): Yes      [MsgId: MMSG-26845]
Additional URLs requiring content: None      [MsgId: MMSG-26845]
Check for newer versions every visit to the page: No        [MsgId: MMSG-26845]
```

Page download timeout (sec): 120 [MsgId: MMSG-26844]

Resource Page Timeout is a Warning: No [MsgId: MMSG-26845]

ContentCheck enabled: Yes [MsgId: MMSG-26845]

ContentCheck script-level file: "C:\...\LrwiAedScript.xml" [MsgId: MMSG-26842]

Enable Web Page Breakdown: No [MsgId: MMSG-26845]

Enable connection data points: Yes [MsgId: MMSG-26845]

Process socket after reschedule: Yes [MsgId: MMSG-26845]

Snapshot on error: No [MsgId: MMSG-26845]

Define each step as a transaction: No [MsgId: MMSG-26845]

Read beyond Content-Length: No [MsgId: MMSG-26845]

Parse HTML Content-Type: TEXT [MsgId: MMSG-26845]

Graph hits per second and HTTP status codes: Yes [MsgId: MMSG-26845]

Graph response bytes per second: Yes [MsgId: MMSG-26845]

Graph pages per second: No [MsgId: MMSG-26845]

Web recorder version ID: 5 [MsgId: MMSG-26844]

LR8.0 中的“[MsgId:MMSG-26842]”在 LR7.8 中是“[MsgId:MMSG-26844]”

LR8.0 中的“[MsgId:MMSG-26844]”在 LR7.8 中是“[MsgId:MMSG-26846]”

LR8.0 中的“[MsgId:MMSG-26845]”在 LR7.8 中是“[MsgId:MMSG-26847]”

13.4 日志文件

负载测试中一个最常见的头痛是，在一个长时间的运行中，运行紧缺硬盘空间。

如果一个事务失败，搜索负载生成器机器中包含如下文字的日志文件

xxx" ended with "Fail"

其中 xxx 是交易名称尾部少数字符。

LR vugen 存储日志（对一个“null”用户）在“output.txt”文件中。

LR 控制器为一个文件夹层级的每个 vuser 存储日志

确定生成器时，从指定的文件夹开始。

然后在该文件夹下生成器创建文件夹，名称如“brr_rf2.63”。

这些每个文件夹下是一个 netdir 文件夹，其中包含指定的有关控制器中结果的文件夹。

当日志文件的增长超出兆字节时，记事本无法打开文件。所以你需要一个文本编辑器程序，这并不尝试在显示它之前，载入整个文件到内存，例如来自 gsoft 或 hexedit 的免费编辑便笺本

13.5 控制器安排表

对于安排表 UI 一个轻微的恼火是用于选项的不同的话语是概括的。下坡道状况根本不会显示在概要网页。

群存在于一个场景中。因此，“场景安排”涉及为所有群更改设定。

模式 （Mode）	场景期间 （Scenario Duration）	负载行为 （Load Behavior）
场景安排 __按场景安排： 	在期间（Duration）标签： __运行直到完成 __运行 _____HH:MM:SS __运行无限期（永远）	内部在 Ramp Up 标签： __同时加载所有 Vusers（默认） __开始__Vusers 每___HH:MM:SS 在 Ramp Down 标签（如果运行有限的时期）： __同时停止一切 Vusers __停止__Vusers 每___HH:MM:SS
组安排  __按组安排 （对每个脚本）：	未知时期	每场景组定义

运行之前初始化所有 Vusers？

确定要指定的恰当的上坡道时间的一个方法是，设置 vusers 同时开始，然后看处理后运行 Vusers 下落的结果比率（如在一个十五秒跨度内 10 个用户） 。

13.6 控制器在线图

在运行标签，LR 最多允许显示 16 个图形。在这样安排下，我喜欢使用它们所有 16 个指标：

事务	第二	系统资源

（Transactions）	（Secondary）	（System Resources）	
运行时：运行 Vusers + ＃连接	统计误差	UNIX 的平均负载	Win 线程
事务：响应时间（秒）	-	UNIX CPU Util	Win CPU Util
总的事务/秒	每秒点击数+下载的网页数+连线+SSL	UNIX Paging	-
吞吐量（字节数）	网络延迟	UNIX 的磁盘交互	-

13.7 合并图

在这两个运行图表和分析曲线图中，我更愿合并成一个单一的图形“响应时间”和“运行 Vusers”和/或“连接数”。

这些是通常正在研究中的最重要的关系。

13.8 每秒图

我也一起合并“每秒”指标和一下图：

- “每秒点击数”- Web 服务器上的点击次数（Y 轴）作为场景经过的时间（X 轴）的一个函数。此图可以显示整个场景，或持续 60，180，600 或 3600 秒。您可以比较这个图和事务响应时间图，看看点击次数怎样影响事务性能。
- “每秒 HTTP 响应”- HTTP 状态码的数量，标示 HTTP 请求的状态，例如，在场景运行的每秒（X 轴），从 Web 服务器返回的“请求成功，”“网页没有被发现”，被状态码分组。
- 在场景运行的每秒中，来自服务器的“每秒下载的网页数”。此图帮助您以下载页面的数目的方式，评估负载 Vusers 生成的数量。像吞吐量，每秒下载的页面是在任何特定秒，Vusers 从服务器收到的，一个数据量的表示。
- 每秒的总事务（通过）
- 每秒的总事务（失败）
- 连接+SSL

LR 不记得大多数场景图设置（4 图是难懂的代码的默认值）。因此，代替刮伤的构建图，我从打开然后改变我的自定义，而不是标准的场景文件开始。

您不需要曾经看见删除图的定义。LR 收集图形数据，即使它是不显示的。

KB 26817：默认情况下，控制器在线监测器显示每个图形的 20 尺寸的一个最大值。为了增大它，到 LoadRunner 的\dat\online_graphs 路径，在控制每种类型的图形的文件中，修改 MaxDispMeasurments=的值：

描述（Description）	文件名称（File Name）
All	generalsettings.ini
系统资源图（System Resource Graphs）	online_resource_graphs.rmd
运行时图（Runtime Graphs）	online_runtime_graphs.def
事务图（Transaction Graphs）	online_transaction_graphs.def
Web 资源图（Web Resource Graphs）	online_web_graphs.def
流媒体（Streaming Media）	online_web_graphs_mms.def

默认的系统资源计算器，微软的 IIS，微软的 ASP，或 SQL Server 监视器在 LoadRunner 的/dat 文件夹，res_mon.dft 文件中被定义。其值可以从场景.lrs 文件内的[MonItemPlus]段被粘贴。

UNIX 的资源和其他一些图表不断被更新，甚至直到测试以后。所以，场景运行后立即，在图形上按鼠鼠右键，冻结显示值，锁定与其他图形相关的值。

安装提示：如果 Web 资源图是空白，通过在 LoadRunner 的\bin 文件夹中，运行 MS-DOS 批处理文件 resister_controller.bat 和 set_mon.bat，尝试重新注册.dll 文件。

活动被显示在运行时设置中的命令顺序执行。不过，事务在两个观测点之间的开始和停止，会表现同时运行，即使他们实际上是顺序执行的。一个 4 秒观察间隔是最经常的，您可以设定控制器在线图（以防止过多的 CPU 密集的图形刷新时间消耗控制器机器）。但分析报告将比在线图显示更多的粒度，- 小到 1 秒。

14、 协议图＆脚本前缀

该控制器基于 WINNT（Windows OS）文件夹内，wlrun5.ini 文件中的值，被调用。服务按照在控制器的可用图形面板中出现的顺序在这里列出。Vugen 在 C 中用前缀 lr_或在 Java 中用 lr.，为 CORBA/RMI Vusers 记录一般函数的名称。其他前缀：

被呼叫的服务	协议/产品（Protocol/Product）	模板文件夹	标头（Header）
自定义/一般	C [WebJS]	lrc	lrun.h, global.h
	Visual Basic	General-Vba	-
	VBscript 类型的脚本	General-Vb	-
	Javascript Vuser	General-Js	global.js
	Java	General-Java	-
电子商业网络资源（eBusiness）	[HTTP]/HTML, SSL, Web Services	http, web, General-Js	as_web.h
	FTP	FTP	mic_ftp.h
（操作的）[系统资源]	Microsoft (Memory, Network, System, NTDS, Objects, Paging, Disk, Queue, Processor, Thread), Windows DHCP/IAS, UNIX, SNMP, Antara Flame Thrower, SiteScope	-	-
[网络]	Winsock Proxy，网络延时	winsock WinSockWeb	lrs.h
[防火墙]	检测点防火墙-1 (no Cisco?)	-	-
[Web 服务器资源]	Apache, iPlanet/Netscape, iPlanet (SNMP), MS-IIS	-	-
[Web 应用服务	Ariba, ATG Dynamo, MS ASP, ASP.NET,	-	-

器]	Broadvision (4.5-5.4 & 5.5-6.x), ColdFusion, BEA WebLogic, Fujitsu INTERSTATE, iPlanet (NAS), Oracle9iAS HTTP Server, SilverStream, WebLogic, IBM Websphere (4.x-5.x & EPM)		
客户端/服务器[数据库服务器资源]	IBM [DB2_CLI], MSSQLServer 2000/ 7.5, ODBC, Sybase CTlib, Sybase Dblib Siebel DB2 CLI, Siebel MSSQL, Siebel Oracle	Siebel_web	lrd.h lrdtypes.h
	Oracle NCA	Oracle_NCA	orafuncs.h
[流媒体]	Media Player Client / Windows Media Server	-	mic_media.h
	Real Client, Real Server	real	lreal.h
[ERP/CRM 服务器资源]	SAPGUI, SAP Portal, SAP CCMS PeopleSoft (Tuxedo)	Sapgui, SAP_Web	as_sap.gui.h
	Siebel Web Server, Siebel Server Manager	-	lrdsiebel.h
	Baan	baan	-
分布式组件[Java性能]	J2EE, CORBA-Java	-	-
[应用组件]	Microsoft COM/DCOM, COM+	com	-
	.NET CLR	dotNet	-
Enterprise Java Beans	EJB, RMI-Java	-	-
[应用程序部署]	Citrix ICA MetaFrame XP	Citrix	ctrxfuncs.h
[中间件]	IBM WebSphere MQ (WMQ) messaging (using CommerceQuest, Inc. MQTester & MQMonitor avail. since Sep. 2003)	-	-

	MSMQ, Tuxedo, Jacada	-	-
邮件	POP3	POP3	mic_pop3.h
	SMTP (Simple Mail Transfer Protocol)	-	mic_smtp.h
	IMAP (Internet Messaging)	IMAP	mic_imap.h
	MAPI (MS Exchange)	MAPI	mic_mapo.h
关系网络	MS-DNS, Antara.net Load, SNMP	DNS	-
	LDAP (Lightweight Directory Access Protocol)	LDAP	mic_mldap.h
应用服务	Tuxedo	-	lrt.h
	MS-Content Management Systems	-	-
无线	i-Mode	I_Mode	-
	VoiceXML	voiceXML	-
	WAP	wap	-
Legacy 终端服务	327x (IBM mainframe) UNIX RTE (Remote Terminal Emulation)	rte	lrrte.h
应用交互管理	F5 BIG-IP	-	-

15、 事务 Web 请求计算器

服务（Service）	概要计算器（Summary Counter）	组件（Components）
ASP 请求	交易量： 跨越一个完整运行的消逝时间（T）的请求总数，	未发现
		拒绝
		错误
		（接受＆完成）
	管道队列长度： 当前（瞬时计数）的请求	排队（等待）
		执行
	驻留时间，秒，事务在服务器的花费时间	等待时间
		服务时间
	使用率，服务器提供服务的百分比	等待
		繁忙
	吞吐量完成率，每秒 请求数/秒 事务数/秒	错误数/秒
		（有效/秒）

它可能混淆而轻视计算器清单。因此，我开发这个表来显现一个单一服务计算器之间的差异。ASP 服务计算器为是所有其他计算器的典型，它的方方面面包括：交易量，管道队列长度，和吞吐量完成率。

括号中的标识，如（Accepted），不被系统所收集，因为他们是假设的，可以通过从总的所有其他相关的计算器减去得到。

ASP 的数据不包括驻留时间（等待时间和服务时间的总和）和使用率。

该 800 美元的 SPECweb99（1.0 版 1999 年发布）和 SPECweb99_SSL（2002 年 3 月）预定义的工作负载生成器基准中，每秒具体的硬件配置的 WWW 服务器连接的数目，能持续需要一个持续的 400 和 320 Kbps 的吞吐量，以便测量被视为的符合。

IBM 发明大师 Arun iyengar 的 Web 服务器性能的分析器

“http://www.research.ibm.com/people/i/iyengar/ton04.pdf”

免费的 Mindcraft WebStone 2.5 基准由原自 Silicon Graphics 公司的 1995 年版本加以改进，也模拟电脑上 100 个 Web 客户端的活动，使得到对 CGI 和服务器的 API 的调用，象静态的 HTML 页面一样。其运行规则目前不支持 POST，SSL，Authentication，HTTP 1.1，HTTP 1.0 keep-alives，Cookies，动态的数据库访问的工作负载量。

运行的示例测试结果使用来从网络监测公司 Paessler 的 Web 服务器压力工具

15.1 Web 性能

TPC（Transaction Performance Council）的 TPC-W Web 的电子商务基准（2000 年 7 月首次发布＆v1.8 在 2002 年 2 月发行)测量每秒处理的 Web 交互(Web Interactions processed Per Second：WIPS）的数目，该 WIPS 指“Web 交互组合”，包括购物（Shopping)（WIPSs)，浏览（browsing)（WIPSb）及订购（ordering)（WIPSo）事务，模拟了一个包括购物车功能，具有 14 个网页的零售书店。01/28/02 的最高审定的价格/效能比的结果是一个在 Dell 的服务器上，Windows 20003AS 内，使用 IIS5＆SQL2000，基于 TCO 的 24.50 美元-277.08 美元/WIPS 的范围。经审定的性能特征被 web 服务器，web 缓存，数据库服务器，和图像服务器详细描述。基准通过提供可执行的远程浏览器模拟器（Remote Browser Emulator：RBE）测算可测量性，该 REB（无客户端缓存）模拟了 2880 个不同的用户，访问具有 2 至 4 GB 的内存（30 秒非 SSL 缓存的休息时间)，各种规模（10,000 或 100,000 等具有 8 表计划的唯一的产品条目）的数据库服务器的数据库。所以 TPC-W 需要一个网络的拓扑结构，该拓扑结构支持数 100 Mbytes/秒的数据。用户考虑时间是基于一个平均 7 秒和最多 70 秒的分配。

15.2 Web（Apache/IIS）监测器

Apache Web 服务器比微软 IIS 提供更少数的监测器

- ＃繁忙的服务器 - 在繁忙状态的服务器的数量。
- ＃闲置的服务器 - 在空闲状态的服务器的数量。
- Apache 的 CPU 使用率 – CPU 被 Apache 服务器使用的时间百分比。
- 次/秒 - HTTP 请求率。
- 发送的千字节数/秒 - 从 Web 服务器发送的数据字节的速度。

LoadRunner 为这些计算器从议题 http://server /server-status？auto 获得的值

Apache 计算器的数目和描述在 LoadRunner 的\dat\monitors\apache.cfg 文件中指明。

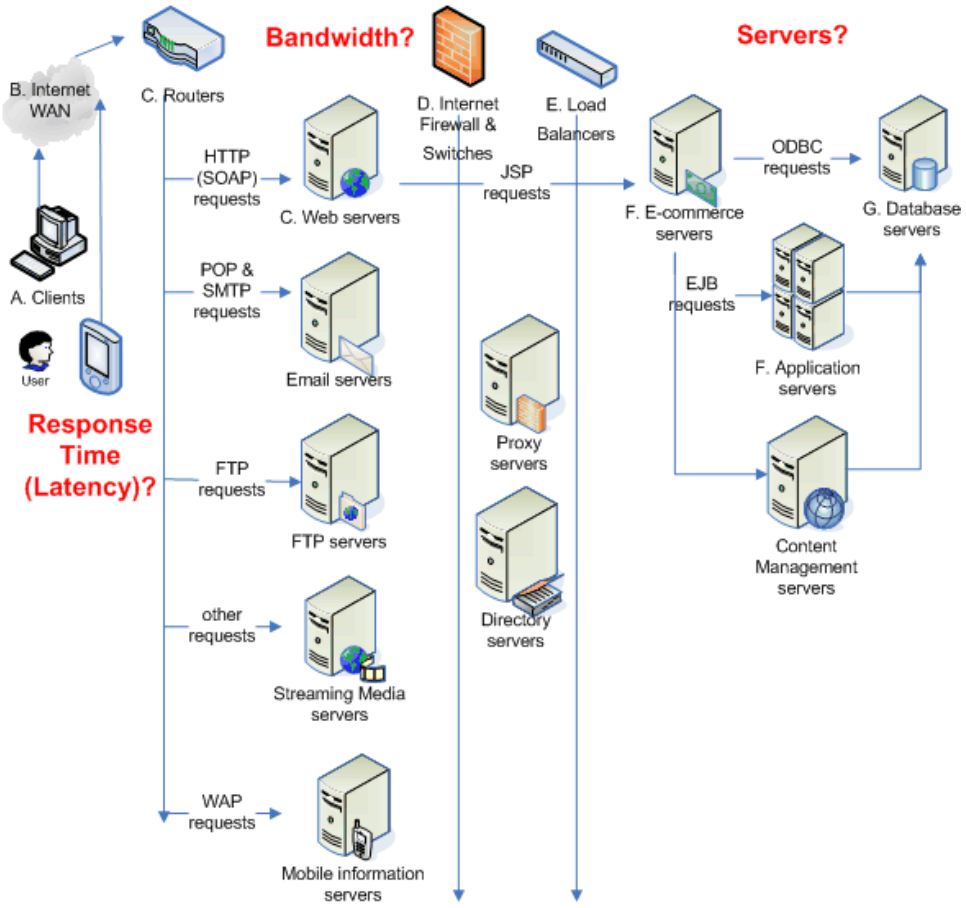
- Counters=指定计算器的数目（默认情况下是 5）。
- Counter1=指定计算器的名称，当它出现在从服务器返回的 HTML 页中时。如果这是拼写错误， LoadRunner 将发出“监测器名称：Apache.解析错误，无法找到凭证… …”
- Label1=计算器的 UI 标签。
- Description1=对计数器的描述。
- KeepPrevValue1=0 表明值是多少，1 表明比率。

如果您是使用 Apache V2，此文件应代替为 apache_2.0.x.cfg。

16、 事务响应时间

A.GUI 客户端 API	B.外部 Internet WAN	C.前向前端的 Web 服务器	D.Internet 防火墙 &路由器	E.负载平衡器	F.中间件（应用服务器）	G.数据库服务器

- 端到端响应时间是总的消费的“来回”时间，从用户发起动作时起（例如在 Web 浏览器上点击提交按钮）- 再加上通过网络，和中间件（如 IBM WebSphere MQ 或 Tuxedo 6/7）和服务器的处理的时间，- 到来自 Web 服务器的一个响应被完全收到，用户可以对服务器响应采取动作止：



- 为了找出瓶颈，需要更精细的事务数据：
- 网络和服务器的响应时间通过在客户端机器上运行非 GUI Vusers 来衡量。
- GUI 文件下载的响应时间通过比较 GUI 端到端时间与非 GUI 总时间来计算。
- GUI 客户端显示（描绘）的时间被手动得到（用秒表） 。
- 中间件到服务器的响应时间，只有通过运行 Vusers 向应用服务器发出单独许可的中间件的 API 得到。
- 单独的服务器响应时间通过运行直接连接到应用服务器（不只是一起在同一个子网中）的一台机器上的 Vusers 得到。

几个运行在事务处理时间的不同允许不同的负载，在事务分析图形中加以比较。

每项事务衡量一个或更多的活动步骤的性能。[Workbook 3-17]

在事务性能图中，“Top 时间”事务的结果指向该系统的瓶颈，在这方面他们不论系统的负载 - 明显需要多于平均的时间来完成。

按照安排的斜上和斜下坡道影响这个。

17、 网页的分项数字

每秒事务数（通过和失败）和事务响应时间从虚拟用户的用户定义的数据点得到，该虚拟用户由网页细分图详述，网页细分图祥细分析了从 Web 服务器资源监测器获得的虚拟用户的客户端事务时间。

网页的分项图形只有在运行前的分析中可用，在 LR8 控制器 Diagnostics 菜单上的 Distribution...对话框中，无论是“启用以下诊断”和“网页诊断”复选框都选中。

以下说明了网页下载时间分解图测量的默认的顺序：

协议 Protocol	度量 Metric	测量说明 Measurement Description	基础结构技术 Infrastructure Technologies
HTTP/S	[客户端时间]	由于浏览器思考时间或其他客户端相关的延迟，请求在客户机上的延时时，平均所需的时间。这不包括为 Flash 来绘画图形（需呀许多秒）的时间。	-
	[连接时间]	需要与 Web 服务器主机指定的 URL 建立一个初始连接。连接测量是沿网络问题的一个很好的指标。它还表明服务器是否是回应请求。	-
	[DNS 解析时间]	使用最接近的 DNS 服务器，解决 DNS 名称为一个 IP 地址所需要的时间。DNS 查询测量是 DNS 解析中问题，或 DNS 服务器问题的一个很好的指标。	-
	[出错时间]	从 HTTP 请求被发送时刻，直到错误信息（只是 HTTP 错误）被返回的时刻，所经过的平均时间	-
	[第一缓冲时间]	从最初的 HTTP 请求（通常是 GET），直到第一次缓冲从 Web 服务器返回时被成功接收所经过的时间。第一缓冲的测量是 Web 服务	-

		器延误，以及网络延迟的一个很好的指标。（到第一缓冲的时间）	
FTP	[FTP 认证时间]	在 FTP 服务器开始处理客户端的命令前认证客户端所需要的时间。这种测量只适用于使用 FTP（不是 HTTP/s）协议的通信。	
HTTP/S	[接收时间]	that passes until the last byte arrives from the server and the downloading is complete. The Receive measurement is a good indicator of network quality (look at the time/size ratio to calculate receive rate). This is the metric reported by: 直至最后一个字节从服务器到达，并且下载完成的耗时。接收测量是网络质量（看时间/大小比例来计算接收率）的一个很好的指标。这是度量报告： longLastByteMSecs=web_get_int_property(HTTP_INFO_DOWNLOAD_TIME);	
HTTPS	[SSL 交换信息时间]	建立一个安全套接字层连接（包括客户呼叫，服务器呼叫，客户端的公钥转移，服务器证书转让，和其他阶段）需时。	

17.1 LRAnalysis70.ini

分析模块读取和保存其设置在 Windows 目录（C:\Winnt 或对 Windows XP 的 C:\Windows）的 LRAnalysis70.ini 文件中。为了让 LoadRunner 7.8 网页分项图形显示长度大于 25 个字符的 URL,，在[WPB]段添加一个 SURLSize 入口指定为 255（最大）：

[WPB] SURLSize=255

此外，在 LoadRunner\bin\dat 目录，使用 MS Access 2000 打开 loader2.mdb 或使用 MS Access 97 打开 loader.mdb。突出 Breakdown_map 表，选择设计视图，改变“事件名称”一栏的尺寸到 255。

17.2 Web 服务器资源监测器（虚拟用户）：

第一缓冲分项图形也显示服务器和网络的时间，即每个网页组成部件相关的服务器和网络的时间（以秒计），直到从 Web 服务器返回的第一缓冲被成功地接收时的一段时间。如果为一个部件下载时间较高，该图可以用来判断是否问题与服务器或与网络相关。

吞吐量图表显示在场景运行的每秒（X 轴）的 Web 服务器上（Y 轴）的吞吐量数额。

每秒点击图表显示作为场景的经过时间（X 轴）的一个函数的 Web 服务器上的点击数（Y 轴）。

18、 测量计算

每次事务的时间越少，系统的吞吐量越大，由 Little 的法则所推导的公式表达如下：

服务器吞吐量 = ＃真正的同时使用者/（平均响应时间+思考时间）

因此，当没有思考时间使用时，虚拟用户的数量可以由 Menasce＆Almeida 所建议的这个公式计算：

＃真正同时使用者*[响应时间/（响应时间+思考时间）]

举例来说，模拟 1,000 个真正用户使用 2 秒响应时间和 20 秒思考时间：

1,000 x [2 / (2 + 20)] = 91

18.1 错误

如果您定义了一个事务在另一个事务中，在控制器的场景状态失败事务统计中，一个错误会算两次，每项事务概要下 1 次。

18.2 系统负载

通过一个比率 - 在一定期限内，执行的任务的总数（1 小时，1 天，1 周，1 个月，1 季，1 年等）来衡量。例如：每秒 300 事务。

可以有几个层次的负载：

- 每小时或每秒高峰期的数量通常用于系统限上限。
- 正常负载水平用来计划持续支持需求。
- 低负载水平用来计划资源百分比，它可以用于脱线的维修和升级。

18.3 任务分配概况

目前任务数的两维视图在时间线的连续段上执行。

风险概况同时也可能包括故障风险的一个估计和影响业务的可能性。

19、 变化的测量

平均数测量一组数的主要倾向。举例来说，10，20 和 30 的平均数是 20。这是这样计算的，所有三个数加入到一起（10+20+30=60），由加起来的项数（在这个例子中是 3）除以这个和。

19.1 标准偏差

标准偏差是在一组数据中对变化程度的一种统计度量。标准偏差告诉你单个项围绕平均数离开的程度。

运行在分析程序，（而不是在控制器）完成后，90％列的数目被显示，因为它是计算由运行期间获取的值排序的所有项计算，然后确定与前所有项的 10％的项相关的值。

CV（变化系数（Coefficient of Variation））通过代表它为一个平均数的倍数，使标准偏差数更容易理解，。

19.2 统计的意义

为了确定是否两个平均数的不同足够使单靠机会不会发生，变化的测量是必要的。为了确定随机的机会的程度，结果需要标准化成一个标准的分布。这是一种数学技术，成为 ANOVA（方差分析（Analysis of Variance）），一个对于因子和等级的任意数的 t 测试的扩展。标准分布的形态可发展成正态分布或指数伽玛分布。

如果一个平均数大于另一个平均数的 90％，这是一个很好的时机，在两个平均数之间有一个显著的差异。

但一个统计上显著的影响实际上并不一定是重要的。

20、 使用分析模块

默认情况下，分析概要（Analysis Summary）被过滤成包括考率时间。因此，每次运行后，单击全面过滤通道图标，向下滚动到“考虑时间”过滤器条件，并取消勾选框。

在控制器中定义的合并图没有输送到分析器。所以，你必须重新建立他们。

结合/合并图形，如“运行的 Vusers”和“事务响应时间”：

- 选择“叠加图”或...

在分析器工具中，将鼠标移动到“运行的 Vusers”图表，并右键单击（竖向的 X 轴）。

- 然后选择“合并图形”。
- 对于“合并的图”，选择“平均事务响应时间”（横向 Y 轴）。
- 选中单选按钮为“关联”。
- 复制[整体运行]图流行的原因在下面列出。
- [初始] - 上坡道期间，良好的识别限制。
- [上限] -限制层，所以相对较少波动的测量更为明显。
- [平坦] -上坡道期间以后。

我喜欢抹掉默认的名称“… …的副本”，并在图形的名称底部添加一个标记，如方以上括号内的词。这是为了让类似的图形保持垂直对齐。

分析器选定的颜色非常随机 - 因此，（讨厌地）就同一图形，相同的颜色可用于两种不同的线。[你会认为应该比这更精确]

因此无论如何你要手动分配颜色，可以考虑为主要指标使用一套标准的基本颜色：

主要指标（Major Metric）	**颜色（color）**	
运行的 Vusers	浅褐色	左上角
平均事务时间	绿色	
数据吞吐量	紫色	
每秒事务数	蓝色	

由 HTML 报告生成的 Excel 文件是被改变尺寸。因此，或者是创造 HTML 报告前改变一切尺寸为 1，或者在电子表格中乘以数目。

为分析器引进外部数据

从其他监测应用程序导入数据到 LoadRunner 分析器，用于统计计算，作图，和自相关的统计资料，LoadRunner 控制器收集：

- 打开一分析结果的文件。
- 选择工具->外部监测器- >导入数据。
- 选择格式，日期，时间，然后单击“Next”。（支持的文件格式是的.csv，.xls，或.log）定义您自己的导入文件的数据格式，。
- NT 性能监测器（*.csv）
- Win 2K 性能监测器（*.csv）
- 标准逗号分隔文件（*.csv）
- 标准 Microsoft Excel 文件（*.xls）
- 主从标准逗号分隔文件（*.csv）
- 主从 Microsoft Excel 文件（*.xls）
- <自定义文件格式>
- 选择这个将创建一个“新格式名称”。
- 点击“添加文件”。
- 点击“确定”添加监测器。
- 单击工具 > “导出自定义图形设置...” 在一个*.axm 文件中存储格式，使该格式在当前会话后可重复使用。它们包含该监测器的名称和 X 轴与 Y 轴的信息。

21、 IP 欺骗

使用 Web 和 WinSocket 协议测试时，默认情况下每个主机上 Vusers 使用主机的（代理人的）IP 地址。由于特定服务器保持服务特定的 IP 地址，负载测试将反映网络路由器缓存和其他的优化，而不是现实生活中的情况，此时不同的用户用不同的 IP 地址进入该服务器。测试的硬件配置均衡穿过几个（web 或数据库）服务器的“farm”的负载时，这是特别需要的。“IP 欺骗”，使每个主机“欺骗”服务器，以为这是使用许多不同的 IP 地址。

首先，内部的 IP 地址（如 192.168.1.1）可能需要由组织的网络管理员加以分配，除非您工作在一个封闭的网络。

IP 向导程序（ipwizard.exe 在 LR bin 中）是用来在每个负载生成器主机上，定义多个 IP 地址。不过，主机必须使用静态 IP 地址，而不是 DHCP 动态分配地址。

注： IP 地址包括两部分，一个 netid 和 hostid。submask 决定在哪儿地址的 netid 部分停止，在那儿 hostid 开始。

当一个主机重新开始，路由表更新成新的地址。

IP 向导程序会创建一个批处理文件（nt_routing.bat for Windows 和 unix_routing.bat for Unix 机器）和.ips 文件，该文件更新 IP 表。

在 LoadRunner 控制器使用多个 IP 寻址时，下拉菜单条目“场景”，选中“启用 IP Spoofer”（在较早版本中，为“使用多个 IP 地址”）。这使在选项对话框常规选项卡上选中了“多 IP 地址模式”，这在工具下拉式菜单上“专家模式”被选中后出现。IP 的分配应该和用户正在运行的方式一致。如果用户作为一个进程在运行，那么 IP 分配应该作为一个进程来做，如果用户作为一个线程在运行，那么 IP 分配应该作为一个线程来做

在启动或关闭 IP 欺骗后，为了设置生效，负载生成器需要断开和再次重新连接。

22、 Mercury 认证

Mercury 的 CPC（认证产品顾问）认证费用$2,500，并在 Mercury 办公室花费 8 个小时。（约 5+小时的程式和 3+小时的问题，具体根据您自己的决定） 。

这里是一个学习这个包的详细方法：

活动（Action）	小时 Hours	累积小时	我保持的时间
阅读 Windows 安装手册，下载补丁，然后从 CD 安装控制器& VuGen	2	2	2
根据说明，在 Linux 服务器上安装负载生成器客户端	1	4	2
在另一个 Windows 客户端上作为一个进程，安装负载生成器 Windows 客户	1	8	2

端			
根据说明，下载，解压缩，然后在您的服务器上安装 Mercury 旅游网站的 Apache Servlets 和 patch57（在 Program Files 文件中）。然后在读 Mercury 的解决方案之前，在学生练习本中尝试练习	4	12	2
阅读每个应用程序的帮助文件。打入并执行示例代码。	4	12	2
执行 VuGen 示例脚本	4	16	2
执行＆分析从 Mercury 支持 LoadRunner 下载的脚本	4	20	2
读在 Mercury 支持 LoadRunner 的知识库的 8130 文章	4	24	2
阅读在 Mercury 支持 LoadRunner 的讨论板的每个线程	4	28	2
阅读在讨论板和其他用户居处的每个线程	4	32	2
从“杂项下载”获得虚拟表服务器，安装它，然后编程操作	2	34	2
配置远程服务器	4	38	2

因此，如果你每星期投入不变的充实的 8 小时的研究，您将需要（36/8）5 周。

23、 用户的居处

在允许你进入他们的产品支持页面或在 408.822-5400（按 1 为支持，然后按 3 为 LoadRunner）收到您的电话之前，Mercury 需要您提供你的产品盒中的“维修号码（Maintenance Number）”（MPN）。这个数字储存在安装过程中创建的，C:\WINNT 内，mercury .ini 文件的“LoadRunner_SerialNumber”入口。这个，在结果上，是与开放源代码的范例极端相反的。

在 Mercury 网站，不使用你想保有机密的密码，（如一个您所使用来进入银行账单支付网站的密码），因为 Mercury.com 存储您的密码在一个互联网浏览器的 Cookie 的纯文本（未加密）中 - 命名为“密码”!

自 1999 年以来，LoadRunner 雅虎集团已积累了超过 1 万的信息。最高贡献者包括创办人 Mark McWhinney（portata.com），Tim Nichols，James Pully，Kevin Quinn，Steve Cheney，Michael Foley，Vishnu Reddl

在 QAForums.com 上的 LoadRunner 论坛 -由 Ilia 和 Jeff Nyman（Globaltester.com）管理。- 使用一个功能丰富的系统主持一个关于几个软件质量的主题的积极的群体。最高贡献者包括 Scott Barber

Google 群 comp.software.testing 在其他议题中有“loadrunner”思路。我已经写信 Google 要将它分离出来。

TCQAA Mercury 论坛已很少进入，但值得偶尔一看。

加拿大人 Roland Stens 的 Performance Tester.com 提供一个论坛，知识库等，使用 Wiki，RSS 等。

测试思考由 Suresh Nageswaran 等写博客。

Mercruy Interactive 的学术访问计划支持学校，例如 Santa Clara，CA，提供这个课程。

24、 RPM（远程性能监测器（Remote Performance Monitor））

RPM 的是一套“一流的”（不是.NET）微软 ASP 网页，它使网页浏览器来监测运行在远程机器上，来自一个 LoadRunner 控制器上的测试结果。

为对于 LR 7.8，一个额外的授权是没有必要的，但对于 LR 7.6 需要。

Mercury 知识库文章 22664 指出，，RPM 需要 Windows 2000 的 IIS 服务器 4.0 或 5.0，但不需要安装 Windows 2003 服务器的 IIS 6.0。

如果控制器被安装在 Windows XP 机器，RPM 服务器被安装在一个 Window 2000 Server，或安装在带有 IIS 4.0 的 Windows NT 机上，XP 控制器机器的本地安全策略“网络安全：共享和安全模式为本地帐户”应从“仅来宾”改变为“传统的”。在 RPM 的安装中：

- 在数据驱动器上指定一个根级别的文件夹（如“RPM”），而不是让 RPM 安装在与 Program Files > Mercury Interactive 下 LoadRunner 相同水平的默认的“RPM 服务器”文件夹中。
- 指定控制器机器的用户 ID 和密码。如果你采取默认的 MI_Viewer，另一个新的用户将可创建在 RPM 机器上。用户 MI_Viewer 的默认密码是 MIOrchid＃1
- 转到开始->程序- > RPM 服务器-> 远程性能监视器用户配置，在“测试配置”一节的对话框中，输入该控制器机器的机器名称，然后点击“测试机”按钮。

如果您没有管理员权限，尝试改变用户 ID：Admin 和密码：Admin，您就会看到此信息：

Run-time error -2147467259 (80004005)':

Method '~' of object '~' failed

RPM Server 安装在一台机器后，控制器不能安装在它上。控制器可以安装在该机器上以前，该 RPM 必须被卸载。

在控制面板 >行政工具 > Internet 服务管理器，您的 IIS 服务器上按鼠标右键，出现 New Web Site。在网站创建向导中，指定名称“RemoteView”，然后 Browse 以连接名称和“WWW 网页”文件夹。

为了启动新定义的网站，您需要重新启动 IIS 服务器，然后启动控制器中的一个场景，以便它显示监测信息。该控制器没有运行场景时，如果您尝试查看 Internet Explorer 上的监测器，您会被带回登录页。

使用浏览器 URL http://loadclient1/remoteview 验证网站默认用户：Admin 密码：Admin 如果响应是“HTTP 401.1 -未经授权：登录失败”或“您无权查看此页”，使用 Windows Explorer，右键点击“WWWPages”文件夹，共享选项卡，

它具有一个帮助不大的“一个适合所有的标尺”的错误讯息：

该网页无法显示正确，因为发生错误

错误描述：

是控制器没有运行，或指定的服务器上没有活动的场景，或您的 RPM 服务器配置不正确。

每个远程阅读器可以选择和自定义图表来监测。

如果没有浏览器活动达到 20 分钟，RPM 暂停。这就需要另一个用户登录。

隐藏数据测试

作者：Mitch Goldman 译者：刘英

隐藏数据测试在软件验收和确认阶段是十分必要和重要的一部分。程序的质量不仅仅通过用户界面的可视化数据来验证，而且必须包括遍历系统的所有数据。一些数据库技巧和下面我所提到的 4 步，能够帮助 QA 团队来在测试过程中很好的执行这样的测试。

隐藏数据测试是非常重要的，在 QA 团队里这也是非常容易被训练测试的。我这篇文章解释了“为什么，谁，怎样”测试隐藏数据。

一、 什么是隐藏数据？

假设一个应用程序要求用户两条信息-----用户名和密码来创建帐户。这个用户输入这两条数据后保存。最后，一个确认窗口将通过数据库中找到这条数据来显示用户名和密码给用户。为了验证所有的数据保存是否正确，一个 QA 测试人员会在这个确认窗口简单的查看下用户名和密码。如果他们成功了？假设数据库记录了第三条信息----创建日期，它可能不会出现在确认窗口，而只在存档中才出现。如果创建日期保留的不正确，而 QA 测试人员只验证屏幕上的数据，那么这个问题就不可能被发现。创建日期可能就是一个 bug，由于一个用户帐户保存了一个错误的日期到数据库中，这个问题也不可能会被引起注意，因为它被用户界面所隐藏。这只是一个简单的例子，但是它却演化出了一点：隐藏数据测试的重要性。

应用程序包括两类数据：一种是最终用户可以看到的数据，另一种就是隐藏数据。在可见的数据中的 bug 很容易被 QA 测试人员发现，因为这些问题可以通过用户界面 UI 来直接看见。而隐藏数据的 bug 却比可见的数据发现困难的多，因为他们不能在用户界面 UI 中看到，它只能通过数据库找到--------然而这些 QA 测试人员可能并没有意识到它们。

二、 谁应该测试隐藏数据？

测试隐藏数据是谁的职责？这个问题不好回答。数据的存取测试是通过应用程序的非用户互动的部分来测试的，这种测试我们通常称为“数据流测试（Data Flow Tetsting）”（Culbertson,Brown,&Cobb,2002）.这种混淆是数据流测试和一些测试学科的跨越线。软件工程知识体系（SWEBOK）（2004，IEEE 计算机协会）将它归为“基于代码”的测试（Bourque& Dupuis，2004）。一个应用程序的组成包括输入数据和输出数据，隐藏数据和可见数据，它们都能被程序单独调用，通过适当的场景来测试。因此这就要求有一定编码知识和会使用

编码工具，因此数据流测试也被认为是基于代码的测试，SWEBOK 似乎也表明隐藏数据应该是开发人员在代码编写阶段进行的单元测试。

然后，作者和软件工程专家 James Whittaker（2003）指出数据流应该也能够通过在用户输入和集中数据存取过程中通过用户界面去调用。因此这些测试可以通过用户界面进行，Whittaker 表示他们应该由 QA 测试人员在开发的测试阶段进行测试。这种测试也在 SWEBOK 关于 Bourque&Qupuis(2004)的文章中描述了作为“基于需求”的测试，是一种功能测试。这就与他们早期的关于开发人员应该执行这些测试相矛盾。因此，如果开发人员和 QA 测试人员的各自团队都认为其对方团队都将执行所有的这些测试，那么可能没有一个团队将测试他们，（或者只有一个团队中的一些人测试而不是所有的人），那么他们的测试结果中的隐藏数据的 bug 就不会被发现。

让我们用我在文章首段举的例子为例。假设创建用户帐户以“CreateUser”表示，另外两个输入---Name，Password---三个输入以 Name，Password，CreationDate 表示。那么在单元测试阶段，开发人员将会输入所有可能的字符来测试 CreatUser。他们会发现一个问题----如果 password 为空，CreationDate 输出也可能为空，而不会显示当前 CreationDate。开发人员修复了这个问题，并检查了代码。但是假设其他开发人员突然覆盖了这段代码，以至于这个 bug 在程序开发结束阶段也没有修复。QA 测试人员通过用户界面测试了一个用户帐户的创建，发现 Name 和 Password 都能正确保存，象第一段所描述的那样。由于不能测试 CreationDate，他们询问开发人员是否它已经进行过单元测试了。开发人员说是的，并且把显示他们在单元测试中关于创建用户成功结果给测试人员看，因此这个 bug 被忽略掉了，如果一个用户输入一个空 password，他们帐户 CreationDate 也将为空，这在以后可能会引起许多问题。

为了解决上面例子中的问题，QA 测试人员必须在数据库中直接验证 CreationDate 字段。事实上，我建议 QA 测试人员应该验证所有隐藏的字段。这看起来很简单，但是实际上，它很大程度上暗示着测试人员的技能和能力：他们需要知道隐藏数据的存在，隐藏数据之间怎样关联，怎样执行关于隐藏数据的测试，也需要有检测隐藏数据之间关联的方法。对于上面的四个问题要求 QA 测试人员的角色上升一个档次，因为他们的钻研超出了黑盒测试的范围，进入了一个“灰盒测试”，灰盒测试必须要求有程序内部工作机制的知识。而前两个问题涉及智力技能：知道隐藏字段在哪里？测试会引起什么变化。测试人员必须知道数据存取的性质，使用这些知识来设计测试用例（Whittaker，2003）。第三和第四个

问题需要更多的技能，使用工具来执行测试和验证测试结果。

三、 测试隐藏数据需要掌握的技能

测试隐藏数据有三个核心数据库技能是必须掌握。QA 测试人员可能不熟悉数据库，因此需要学习这些技能（Sweeny，2004）。第一个技能是了解基本的数据库设计，特别是数据库关联，这在现在是最常见的。测试人员也必须了解一些概念如：关键字段，外键关系，参照完整性，父表与子表的关系记录，数据类型，记录锁定，数据库设计图表和符号，create,retrieve,update,delete 的过程。这些技能是分析数据库设计，找到隐藏字段，和设计测试用例所必须掌握的。这些测试也能够帮助 QA 测试人员识别隐藏数据 bug 引起的原因，而不只是悬浮在症状表面。

第二个数据库技能是 QA 测试必须能书写流畅的数据库语言，结构化查询语言（SQL）。大多数应用程序是通过使用 SQL 报表与数据库通信的，在现代程序中，某些组成比如数据库视图和存储过程都是通过书写 SQL 语言来完成的（Sweeny，2004）。SQL 技能是验证测试结果所必须掌握的，通过书写 SQL 语句从数据库中检索数据，或通过分析 SQL 语句和从数据库写入和取出的数据。SQL 通常用 create,update,delete 数据来创建一个独特的测试用例。并且能准确的找出隐藏数据 bug 引起的原因。

第三个数据库技能是有使用数据库管理实用工具的实践知识和经验。这种具体应用不同于数据库到数据库，而是一个重要的技能：a）能够手动的送达 SQL 语句到数据库。b）能够查看数据库，表和字段设计。c）能够使用 SQL 语句捕捉数据从数据库中的出入。例如，MS-SQL 2000 数据库为企业伙伴提供了各自的特点：a）SQL 查询分析，b）企业管理，c）SQL 事件探查器。QA 人员需要找出支持测试下的应用程序的数据库和学着怎样使用这些功能。

四、 怎样测试隐藏数据

一旦必须掌握的技能被学习，QA 测试人员必须运用它们来完成测试隐藏字段的任务。这里有一个方法来执行任务，下面我分四个部分来详细论述：

1. 找出隐藏数据。

QA 测试不可能测试他们不知道的东西。大多数隐藏数据信息能够从设计文档中推断出来，比如数据库结构，数据流图表，需求。如果这些文档无法使用或者写的不详细，QA 人员可以做一些“侦探性的工作”来找到它们。一个方法是创建输入和输出变量表格。这个方法将程序与单独加工阶段分离开来，表格的结果将显示用户不可见的输入和输出

(Kaner, Falk, & Nguyen, 1993)。了解数据库结构和设计的技能对这一步非常有用。

2. 定义测试用例。

一旦知道了隐藏数据，QA 测试团队必须找到何时如何他们能够改变和设计测试来证明和反证所引起的变动。何时和如何可以通过第一步如果定义隐藏数据方法来找到。它可能有简单业务规则或数据流的形式，比如：“当 X 可见数据 insert/update/delete，那么 Y 隐藏数据被 insert/update/delete。这里指出对于测试需要的任何事先存在数据，来帮助后面执行该测试是非常重要的。

从这个信息来设计测试是非常麻烦的，因为依靠的测试类型正是 QA 测试团队试着完成的。比如，如果安全性是优先的，那么设计的测试确定隐藏数据不能被没有权限的用户所访问或中断。如果高容量是一个优先的，那么设计的测试可能是隐藏数据和大量的同步数据的负载和压力测试。最低限度，至少业务规则或数据流可以证明和反证一次。例如：“改变 X 可视数据的 value=A，然后验证 Y 隐藏数据是否变成 value=B”和不改变 X 可视数据的 value=A，然后验证 Y 隐藏数据是否没有变成 value=B”这是测试隐藏数据的基本方法，如果有需要的话，也提供给更复杂的数据基础。

3. 执行测试用例。

这通常通过在用户界面处理可视数据来完成，通过在数据库中提交可视和隐藏数据来引起这种变化。通过带有预先填入数据的“seed”数据库来创建一个所须的测试场景是非常必须，如上所述，在第二步中设计测试用例以指出。有时候，隐藏数据的变化可能是通过其他用户界面所触发的，比如通过时间事件或自动过程。在一些案例中，执行测试用例是通过设置触发事件来完成的。

4. 分析测试结果。

这个过程是最有技术挑战的部分，因为它直接涉及连接的数据库，而不使用应用程序的用户界面。所有的三个数据库技能都将在这一步发挥作用，QA 测试人员也必须对他们寻找的隐藏数据具有一个直觉(Whittaker, 2003)。这里有两个重要的方法来分析这个结果：A）在测试执行后发生 B）在实时测试过程中发生

在测试执行后查询数据库。一旦数据提交已经改变，可以通过检查记录来确定数据是否与第二步中所设计的测试用例的期盼结果相匹配。最简单的办法是通过送达一个 SQL 查询语句到数据库来执行它，来说明这个测试结果。最常用的语句是，一个“SELECT FROM”语句就能够检索所有的或部分记录数据。这个语句也通常包含一个过滤作用，比如一个

“WHERE”语句，以至查询出只在测试中返回的记录。

当然也有一些直接的方法可以使用，依靠数据库的可用的实用工具。比如：Microsoft SQL Server 2000 的数据库管理系统的实用工具，企业管理，提供的一些可选程序。它可以让用户从一个列表中选择一个表，然后右键来“返回所有行（return all rows）”或“返回前 X 行（return top X rows）”而不需要使用任何 SQL 语句。这个结果能够快速的扫描所有被测试的记录，然后验证上面所有的期盼结果。

B 在实时过程中追踪数据的改变。当测试的应用程序送达了一个命令给数据库，比如 SQL 语句，该数据流就可以被某些数据库实用工具所捕捉。同时，如果数据库的任何一个对象，比如查看或存储过程被调用，他们的 SQL 命令也能够被捕捉。不仅仅语句能被捕捉，而且任何常量或动态数据也会在 SQL 中被嵌入。因此“实时数据”可以对期盼结果进行验证。依靠这些实用工具，SQL 数据流能够被过滤，以至只显示期盼语句。

隐藏数据测试案例

作为上述过程的一个案例，让我们运用第一段所述的例子来深入阐述。

第一，识别隐藏数据。创建一个包含“create user account”的输入和输出变量的表格如下：

表 1：输入/输出变量

UI 事件	创建一个用户帐户	
输入变量	用户创建的帐户	
变量	数据源	隐藏/可视
Name	用户输入	可视
Password	用户输入	可视
输出变量	在帐户创建后通过窗口确认	
变量	数据源	隐藏/可视
Name	用户帐户	可视
Password	用户帐户	可视
CreationDate	自动产生（数据保存的日期/时间）	隐藏

第二，编写测试用例。使用上述信息，一些基本的测试结果可以推导如下：

表 2：测试用例

测试用例 1		
第一步：打开应用程序，指向“Create User Account”导航栏		
第二步：输入下面的数据		
输入	字段名	数据
	Login Name	John Doe
	Password	隐藏数据
第三步：点击“保存”		
第四步：期盼结果：在用户帐户表格中创建了一条新的记录，如下：		
结果	字段	数据
	UserName	John Doe
	Password	隐藏数据
	CreationDate	保存日期和时间
测试用例 2		
操作步骤与上面相同，但是数据/结果如下		
输入	字段名	数据
	Login Name	空
	Password	隐藏数据
结果	字段	数据
	UserName	空
	Password	隐藏数据
	CreationDate	保存日期和时间
测试用例 3		
操作步骤与上面相同，但是数据/结果如下		
输入	字段名	数据
	Login Name	Jane Doe
	Password	空
结果	字段	数据
	UserName	Jane Doe

	Password	空
	CreationDate	保存日期和时间

第三，执行测试用例。因为所有的测试用例都是通过用户界面所执行的，不需要什么技巧或实用工具。QA 测试人员只需要简单的输入可视的测试用例数据进入可视字段，然后点击“保存”

最后，分析测试结果。一个方法是在测试用例执行后运行 SQL 语句。这些数据如下：

表 3：SQL 语句

测试用例 1 的验证查询：
SELECT UserName,Password ,CreationDate FROM UserAcct WHERE UserName = “John Doe”
测试用例 2 的验证查询：
SELECT UserName,Password ,CreationDate FROM UserAcct WHERE UserName = “”
测试用例 3 的验证查询：
SELECT UserName,Password ,CreationDate FROM UserAcct WHERE UserName = “Jane Doe”

在测试场景中，SQL 语句的结果如下：

表 4：保存后的数据结果

测试用例 1：如果保存在 1/1/2005 10：00am		
UserName	Password	CreationDate
John Doe	隐藏数据	1/1/2005 10：00am
测试用例 2：如果保存在 1/1/2005 10：30am		
UserName	Password	CreationDate
空	隐藏数据	1/1/2005 10：30am
测试用例 3：如果保存在 1/1/2005 11：00am		
UserName	Password	CreationDate
Jane Doe	空	空

前面两个测试用例的测试结果与表格 2 的期盼结果相匹配。但是第三个测试用例却没有，创建日期为空而不是在记录创建的日期。如果创建日期数据不能直接在数据库中验证，它可能会比较容易被 QA 测试人员所跳过，因为它并没有出现在用户界面，也就不会有人

看见记录。

另外一种方法来分析测试结果是在实时情况下“追踪”SQL 语句。创建用户模型是一个“储存过程”，在数据库是一个对象。同时创建用户也会将 CreationDate 作为第三方输入变量。在每个测试执行过程中 SQL 追踪都会显示下列语句：

表 5：实时 SQL 追踪

测试用例 1：如果保存在 1/1/2005 10：00am
EXEC CreateUser ‘John Doe’, ‘HiddenData’, ‘1/1/2005 10:00am’
测试用例 2：如果保存在 1/1/2005 10：30am
EXEC CreateUser ‘’, ‘HiddenData’, ‘1/1/2005 10:30am’
测试用例 3：如果保存在 1/1/2005 11：00am
EXEC CreateUser ‘Jane Doe’, ‘’, ‘’

同样的，前面两个测试结果与期盼测试结果相匹配，但是第三个测试结果却没有。此外，追踪显示数据源问题来自于应用程序而不是存储过程。我们能够清楚得看到数据进入到存储过程的并不正确-----在测试案例 3 的测试结果中 CreationDate 的值为空。因为这些数据来自于程序本身，我们能够推倒出问题就在于此，而不是存储过程。

有时候，SQL 追踪和 SQL 语句的验证对准确定位一个 bug 是十分必要的。假设测试案例 3 中，SQL 追踪显示“1/1/2005 11：00am”已经进入存储过程，而 SQL 语句直接返回了 CreationDate =空的值。在这个案例中，这个问题定位在存储过程。应用程序让象 SQL 追踪里所看到的正确的数据进入到存储过程，但是存储过程没有保存象 SQL 语句结果一样的数据到记录中。

五、 结论

隐藏数据测试在软件验收和确认阶段是十分必要和重要的一部分。程序的质量不仅仅通过用户界面的可视化的数据测试，而且必须包括遍历系统的数据测试。所有的往返数据都应该检查，从数据输入到存储，从检索数据输出或显示（Sweeny，2004）。一些数据库技巧和这篇文章中提到的方法，能够帮助 QA 团队来在测试过程中很好的执行这样的测试。

如何攻击软件

作者：James A. Whittaker 译者：贾国莹

摘要：本文通过介绍一些方法（即所谓的“攻击”）来揭露软件设计和开发中的错误。这些攻击是手工的、探索性的测试设计，可以飞快的执行而花费很少或者根本没有开销。这些攻击是通过学习了大量实际的软件错误，将这些错误的原因和症状进行了归纳之后形成的。美国佛罗里达州技术学院的学生通过一个学年的手工精确测试，已经确定了数十种攻击软件的办法，以达到发现软件中错误的目的。这些方法已获得成功，在很短的时间内在几乎不熟悉软件的情况下，发现了大量的额外的错误。本文旨在介绍这些方法中的一部分来验证它们在发现软件错误中的实际使用情况。

概述：

是什么使优秀的软件测试人员变得更优秀？是什么直觉指导他们准确的定位错误？可以学到这些有用的本领吗？

这些问题就是本文的主题。我相信一个优秀的测试人员不仅仅靠直觉来引导，实际上，多年来很多测试人员建立了一个标准攻击的兵工厂。每一次当他们面对一个新的测试问题时，他们都要精心组织攻击，发誓一定要找到错误。尽管这些攻击策略很少能够记录下来，但是这些策略在手工测试中占有很重要的地位，是培训新的测试人员的一个非常好的途径。

我们通过学习实际的测试人员如何寻找软件错误，来开始兵工厂之旅。在本文中，我们将探讨已有攻击策略的一部分。我们的下一个挑战是开始研究自动化攻击的过程，并找到有效的解决方法。

攻击方法可以分为三类：

输入/输出攻击

数据攻击

计算攻击

每一类特定类型的攻击，可确定一些非常有趣的软件故障。接下来的部分我将描述每一类方法中的一些攻击手段，并有具体的错误示例。我验证的每一个错误都来自于微软公司的产品。但这不代表我具有反对微软的立场。事实上，微软作为全球顶级的软件公司使得它成为众矢之地。但是不要假设微软的产品就比其它公司的产品错误多。文中所描述的

攻击能够突破来自于许多厂商的几乎每一个可以想象到的平台。我的经验表明，开发人员编写的错误，存在一个相当统一的概率。而不论其应用领域，使用哪种操作系统，以及是否公布其源代码。当然，除非他们是网页开发人员，因为很少有机会需要去中断网页软件。

一、 输入/输出攻击

输入/输出攻击也就是大多数测试人员所说的“黑盒”测试，它不考虑内部的数据和计算。实际上，这是最普遍的测试类型。因为看源代码十分枯燥，又费时费力。除非你确实知道你要找什么，不然将做大量的无用功（我们将在下面的两部分讨论如何去寻找你的目标）。

输入/输出攻击有如下表所示的 10 种类型，分为输入数据，输入数据组合和输入顺序 3 个大类。下面我们将详细讨论每一个攻击类型，并举例说明。

输入/输出攻击	1、输入数据攻击	强迫所有错误信息出现
		强迫默认值被指派
		探索允许的字符集
		强迫改变输出规模
		溢出显示区域
		强迫屏幕刷新问题
	2、输入数据组合攻击	强迫非法输出
		找到不能并存的输入
	3、输入顺序攻击	强迫非法输出
		重复多次输入序列

1. 输入数据攻击

这种类型的攻击只需要使用简单的输入或者输入数据（如果是输入变量）就可以进行。我们试图发现一个简单的数据来中断应用程序，即使其它的大量数据能正常使用。有很多种方法可以用来选择数据，而不仅仅简单的找到可接受和不可接受输入之间的界线，尤其是如果你想要找到错误，并且这些错误是开发人员要修复的但是无法用事实证明是存在的。

下面是例子但是这些例子有时很难实现：

请务必确保你能看见所有的错误信息。

确定一个程序的错误确实很难，这通常意味着存在错误。有些错误信息较为简单，只要简单的暂停执行就能显示出错误提示，然后可以继续下一个输入或者将时间消耗完。然而，其它的错误信息来自于抛出的异常或者执行的异常句柄。异常句柄（或者任何集中式错误事务）是因为指针意外的发生了变化，而没有随着数据状态的改变而改变。突然，异常指针开始执行，所有的数据错误接踵而来：文件一直处于打开状态，内存一直被占用，数据可能仍然没有初始化。当控制一旦返回到主程序，就很难分清错误句柄在哪里返回，有什么遗留下来的副作用可能会在什么地方等待粗心的开发人员。例如：打不开一个文件，因为这个文件已经处于打开状态，或者你可能会使用没有正确初始化的数据等等。如果我们要确保能看到所有的错误信息并且系统一直工作正常，我们就必须为我们的用户（更不用提我们的维修商了）做大量的服务工作。

图 1 显示了一个十分有趣的软件缺陷，它是由我的学生在 Word 2000 中发现的。在 Word 2000 中错误提示会莫名其妙的出现两次。该缺陷是在用单一输入值探查法攻击错误处理例程时发现的。

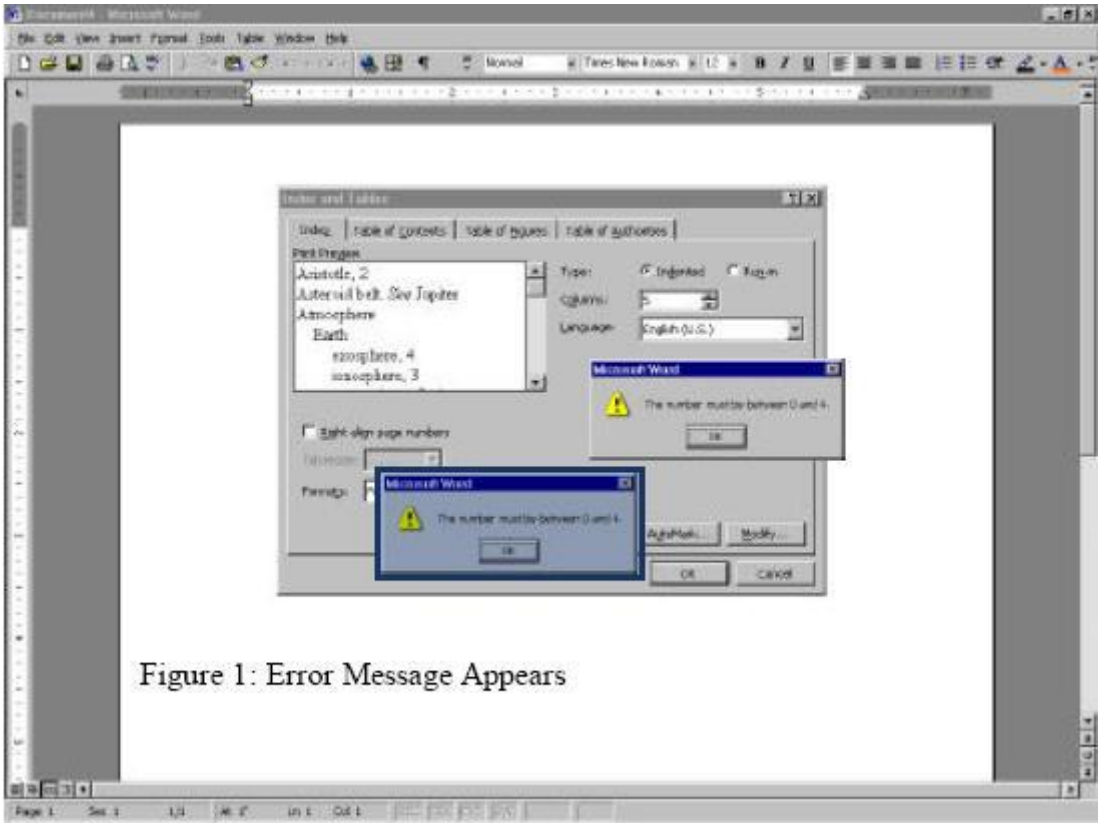
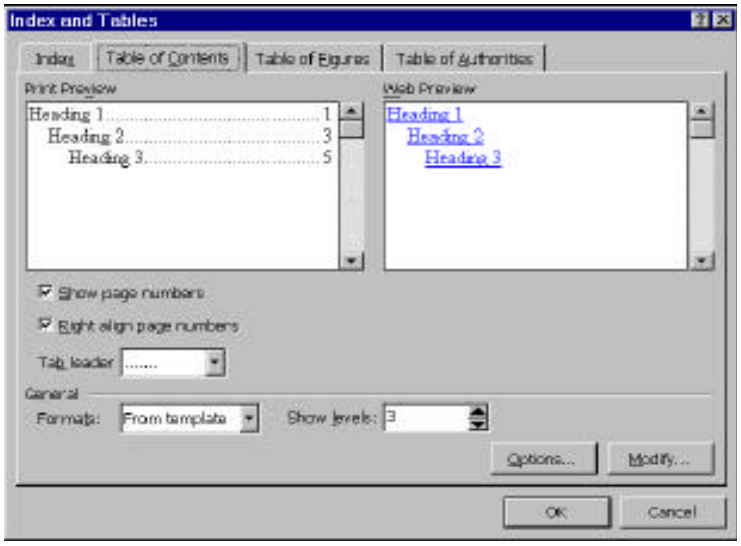


图 1

请务必迫使软件建立默认值。

开发人员经常忘记当用户输入超出范围的数据或者错误的设置参数时，要为用户建立合适的程序默认值。有时强行的默认值能做任何事情，即使优秀的开发人员也能从中获益，因为它能产生意想不到的效果。例如：在 Word 2000 如图 2 所示的对话框中有一个选项菜单，当左边无变化，在对话框重新显示之前实际上控制是无效的。比较图 2、图 3 两个对话框，注意到任何丢失的控制了吗？ 图 2:



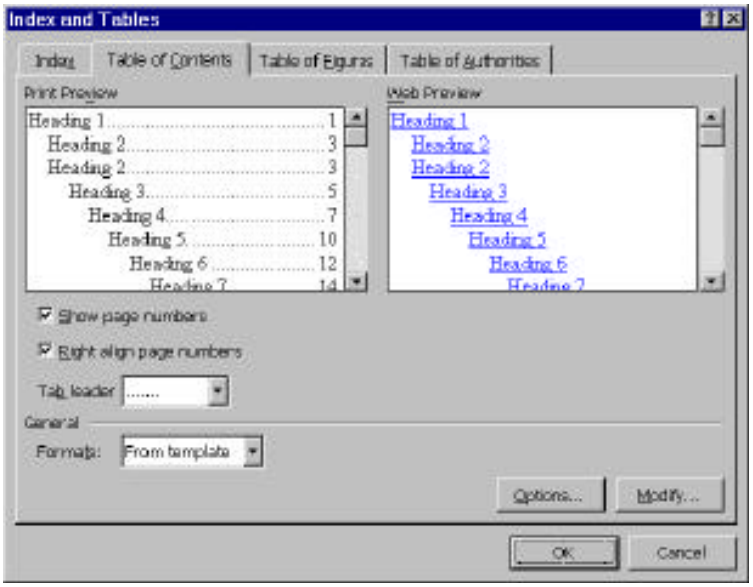


图 3

有时候强迫建立默认值需要对以前的初始值进行变化，然后变成错误的设置。这种黑盒到黑盒的改变，可以保证一旦默认值变成其它的非法数据时能够被重建。

为变量输入探索所有的字符集。

有些输入值是要慎重考虑的，特别是当你对待一些特殊字符如$,%,#,引号，以及其它在很多程序语言中代表特殊含义的字符，在作为输入时经常要特殊处理。如果开发人员错误的应用了特殊字符，这些错误的输入很可能引起程序出错。

通过用少量的输入值代替大规模的输入值的方法，迫使输出值规模发生变化，反之亦然。

着眼于处理输出值的一种使用技术少获得回报大的寻找错误的方法。该方法考虑一个输出或者标示缺陷的行为，然后试图追踪能迫使这种行为发生的输入。按照这种思路有一种常用的攻击，它通过改变输入字符串和输出字符串的长度来迫使输出区域重新计算。

一个好的理论例子是设置一个时钟，从 9：59，观察它滚动到 10：00。在第一轮其显示区域是 4 个字符长，第二轮其显示区域是 5 个字符长。通过另外一种方法，我们建立 12：59（5 个字符长），然后观察文本缩小到 1：00（4 个字符长）。开发人员往往编写的代码是通过初始的空白显示区域来运行，结果当显示区域已经有数据，而用不同规模大小的新数据来代替已有数据时常常发生错误。

例如：PowerPoint 中的“WordArt”，也就是艺术字存在一个有趣的问题。假设我们输入一个字符串，如下图所示：

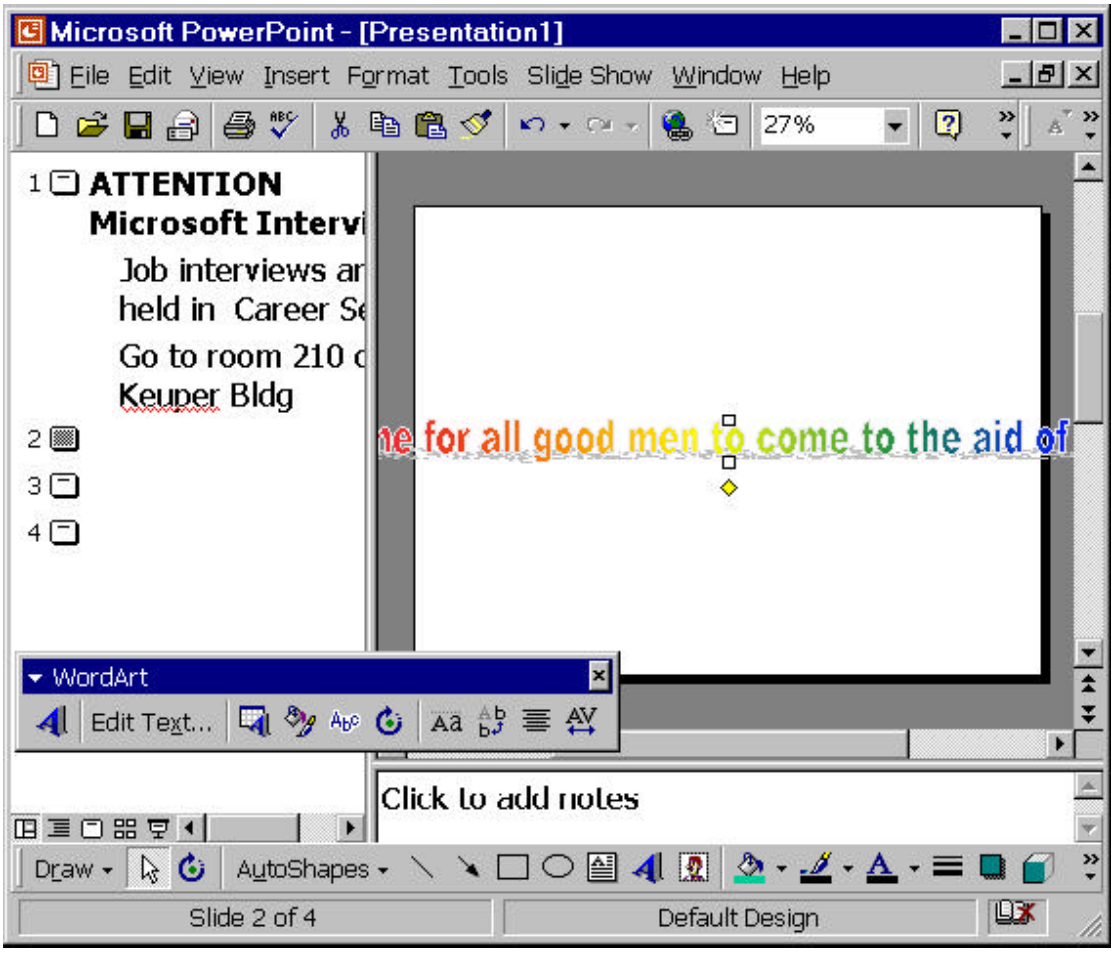

Microsoft PowerPoint - [Presentation1]
File Edit View Insert Format Tools Slide Show Window Help
27%
1 ATTENTION
Microsoft Intervi
Job interviews ar
held in Career Se
Go to room 210 c
Keuper Bldg
2
3
4
ne for all good men to come to the aid of
WordArt
Edit Text...
Click to add notes
Draw
AutoShapes
Slide 2 of 4
Default Design

图 4

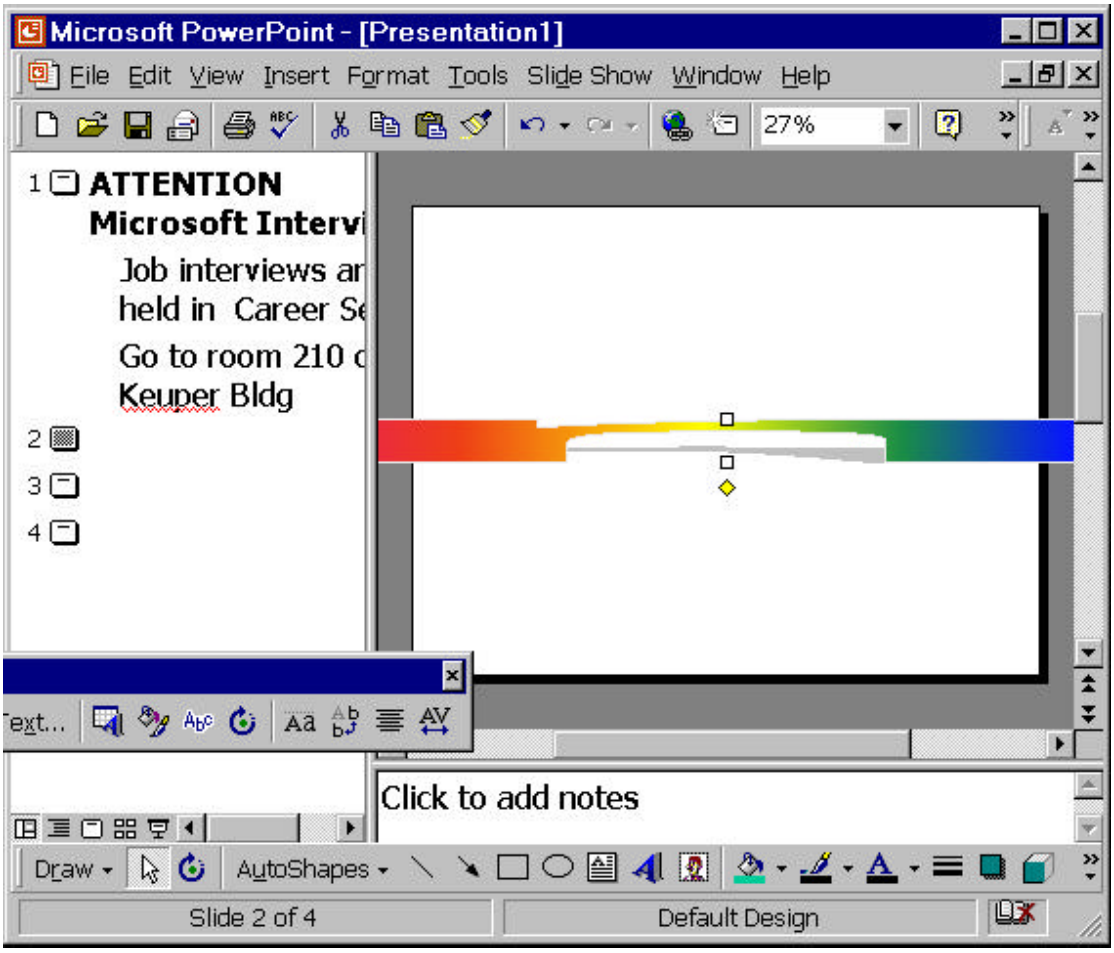


图 5

请注意整个字符串因为太长并不能全部显示出来。但实际上这并不是主要的问题。当按下“OK”按钮后会发生两件事情。第一件是计算出需要的常规输出域的大小以及我们输入文本的大小。现在让我们编辑字符串，并用一个单一字符代替它。

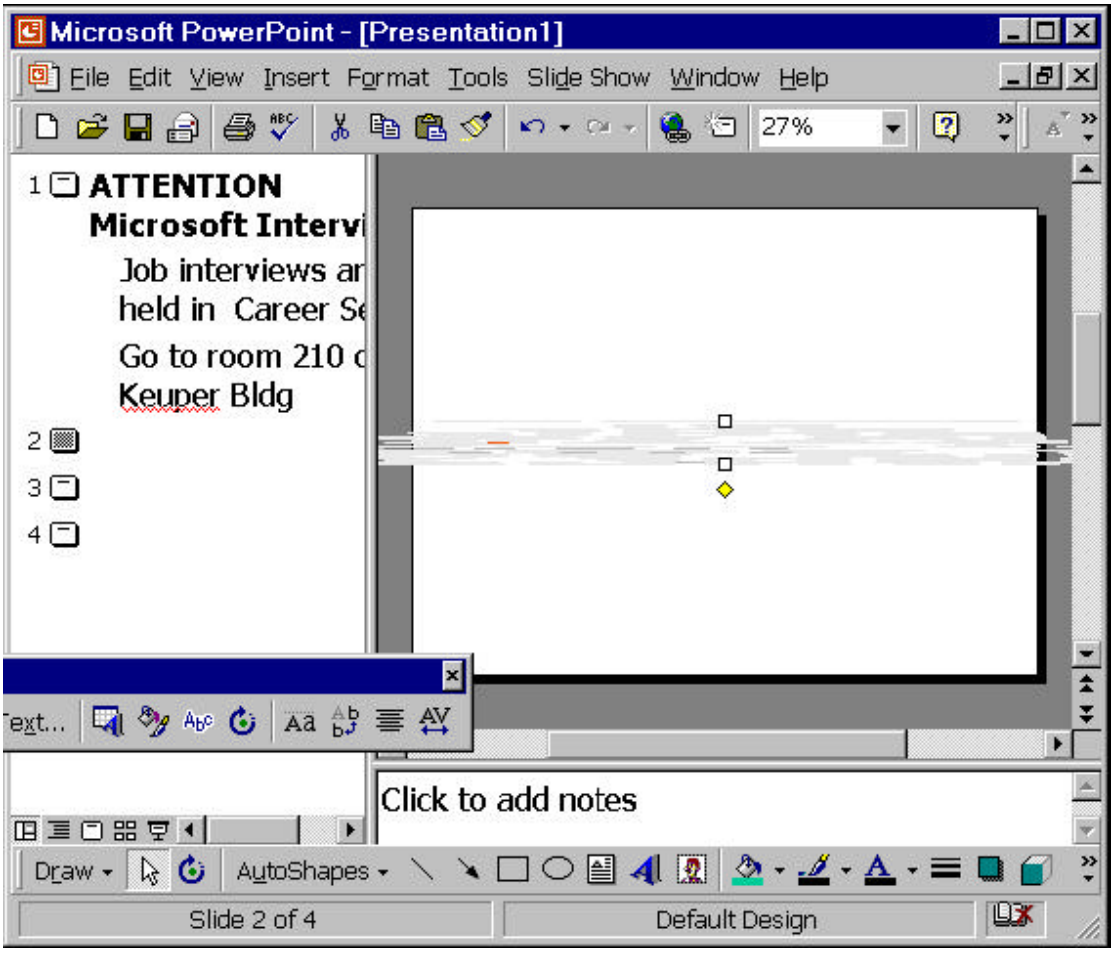


图 6

请注意显示区域仍然没有改变，尽管我们只输入了一个字符而且字体大小没有改变。我们进行下一步工作。如果我们再次编辑这个字符串，键入一个多行字符串，那么输出将更有趣。

我认为这点已经讲明白了，可以开始下一条攻击了。

确保你探索了显示区域的边界。

这是基于输出的另外一种攻击，和前一种攻击类似。然而，我们并不是要想办法使显示区域内出错，而是要将注意力集中在显示区域外部。这一次我们将不再需要重新计算显示边界而仅仅使其溢出。

再来看前面提到过的 PowerPoint 的那个例子，我们可以画一个文本框，用一个上标字符串填充它。

调整上标文字为大号字体，使得文字顶部突出被截断。

这个功能要论证的问题与下面将要研究的相关问题存在关联。

试图迫使引起刷屏错误。

这个问题是现代图形界面用户所面临的最主要问题。甚至也是开发人员所面临的最大问题：如果刷新太频繁将使应用程序运行变慢，错误的刷新可以引起任何可能的错误，从监视器问题（需要用户强迫刷新）到主要的缺陷的发生（即阻止用户完成工作）。

通常所讲的寻找刷新问题是指在屏幕周围增加，删除，移动对象。这将引起背景对象重新显示，并且如果它无法及时地、正确地显示，就表示你找到了典型的刷新缺陷。将一个对象从初始位置进行移动，并调整移动的距离，是个很不错的想法。将对象先移动一点儿距离，然后再多移动一些距离；或者先将对象移动到一倍或两倍的距离，然后再将其移动到十多倍的距离上。

继续我们上面用到的那个大字号上标文字的例子，试着每次将其在屏幕周围移动一点儿。请注意如下图所示的糟糕的刷新问题。

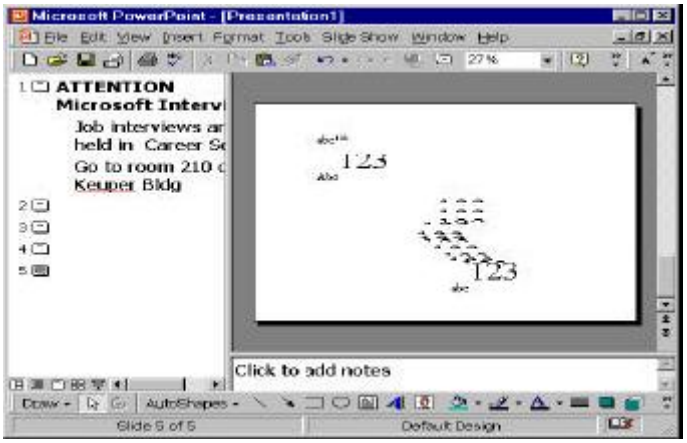


图 7

另一个要指出的 Office 2000 中与屏幕刷新相关的问题是消失的文本。 在 Word 中最恼人的就是页边距的问题。

2. 输入组合攻击

第二种类型的输入/输出缺陷是处理多输入数据时的情况，这些输入可以一起执行或者一个个执行。例如：一个 API 函数要调用两个函数，一个参数值的选择基于另一个参数值的选择。这种情况往往是数值的组合，而且由于代码的复杂逻辑关系，易造成编程错误。

找到不能并存的组合输入值。

哪些组合输入值是有问题的呢？这仍然是我们积极探索的一个重要问题。但是目前我们已经找到了一种方法，它对于决定你想要得到的一个输出尤其有效，而且可以通过该方法寻找引起输出错误的输入组合。

设法使目标应用程序产生无效的输出。

对于确实了解错误区域的测试人员来说，该方法是一种十分有效的攻击手段。例如：如果你要测试一个计算器应用程序，知道其中一些函数对它们的结果有限制范围，则想办法寻找组合的输入值来验证这些结果是一种有意义的尝试。

有时窗口自身也能够为你提供线索来说明哪些输入值是相互关联的。在这种情况下，测试人员可以通过试验数据的范围来尝试违反已知的关系，以获得有价值的线索。

3. 输入顺序的攻击

软件的输入形成了一种正式的语言。单独的输入构成了该语言的字母表，输入字符串组成了该语言的句子。一些这样的句子应当通过接口启用和禁用控制来阻止其产生，输入域和这种行为可以通过应用大量的输入字符串和所有可能的输入顺序来测试其正确性。

选择可以迫使生成错误输出的输入字符串。

这是一个很好的鉴定有问题输入序列的策略，和上面提到的用来发现有问题的输入组合的策略有异曲同工之处。例如：当我们注意到 Office 2000 中存在消失文本的错误时，我们制定了一个针对 PowerPoint 幻灯片标题文本框的攻击。下面一系列的截图演示了一个详细的由输入序列引起的文本消失的过程。

图 8

图 9

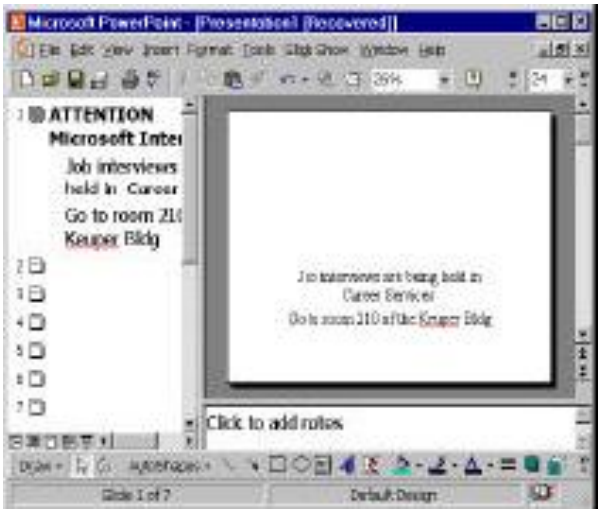


图 10

有趣的是，请注意，刚才旋转了 180 度的文本框没有显示错误。一是必须按照描述的旋转命令顺序执行：旋转 10 度（或更多），其次是 180 度。二是撤销运行的序列，或者没有纠正错误的序列，每次单击一下标题外的区域，这样标题文本将消失。

输入序列之所以可以作为胖缺陷攻击策略，是因为很多操作可以成功完成，但是遗留下来了一些能导致将来执行错误的负效应。全面彻底的输入序列探查可以使很多同类型的问题暴露出来。有时，输入序列变量的数量的多少并不能发生显著的变化，以用来发现接下来将要描述的攻击所能找到的缺陷。

多次重复输入相同的输入数据或者输入序列。

这将包括大量资源造成的影响以及要强调一个应用程序存储数据的空间，而不用暴露不良的负效应。不幸的是，大多数应用程序没有意识到自身空间和时间的局限，很多程序

开发人员习惯于假设有充足的资源可用。

在 Word 里可以找到这样的例子，Word 里的等式编辑器看起来并没有意识到它只能处理 10 层嵌套的括号。

二、 数据攻击

数据是软件的生命线。如果你的数据管理混乱，软件将最终不得不使用坏的数据，而接下来将要发生的就不是什么好事了。所以了解如何以及在什么地方建立数据值是有意义的。

实质上，数据的存储要么通过读取输入值将其储存在内部，要么储存一些内部计算的结果。所以可以通过提供输入和被动计算，我们能使数据流经被测试的应用。基于这种方式的数据攻击概述如下。

数据攻击	1、变量值攻击	迫使错误输入的数据被储存
		迫使数据值超过所允许的范围
	2、数据元素规模攻击	溢出输入缓存
		迫使过量的数值被存储
		迫使太少量的数值被存储
	3、数据访问攻击	找到替代的方法来修改相同的数据

1. 变量攻击

这种类型的攻击需要研究数据类型，以及与内部存储的数据对象有关的所允许的数值。如果能访问到资源则这方面的信息就是现成的。然而，暗含类型的信息可以通过一个小的探测性测试来确定，这时要注意提示的错误信息。

通过输入域中不同数据类型的使用找到类型匹配错误。

在程序期望得到整型值的地方输入字符（或者相似的攻击），这是种早已被证明富有成果的攻击方法。但是我们发现该类型的攻击没有前面提到的攻击方法有效，这时因为现代程序语言放松了对类型检查和类型转换的处理。

试图超出数据值所允许的范围。

存储的变量数据就像输入的变量数据一样，也是攻击的对象。

2. 数据元素规模攻击

第二种类型的数据攻击，目标是上溢和下溢数据结构。换句话说，该攻击企图找到违背预定数据对象规模限制的数据。

第一种该类型攻击的方法是缓冲区溢出。

试图溢出输入缓冲。

这里是指输入长字符串来溢出输入缓冲。该方法是黑客的最爱，因为有时候当一个应用程序销毁后它仍然可以一直执行一个进程。如果黑客在输入的长字符串尾部附加一个可执行字符串，则进程就可能执行该附加字符串。

在 Word 2000 里就有一个可以利用的缓冲区溢出缺陷。这个缺陷在查找/替代功能里，如下图所示。

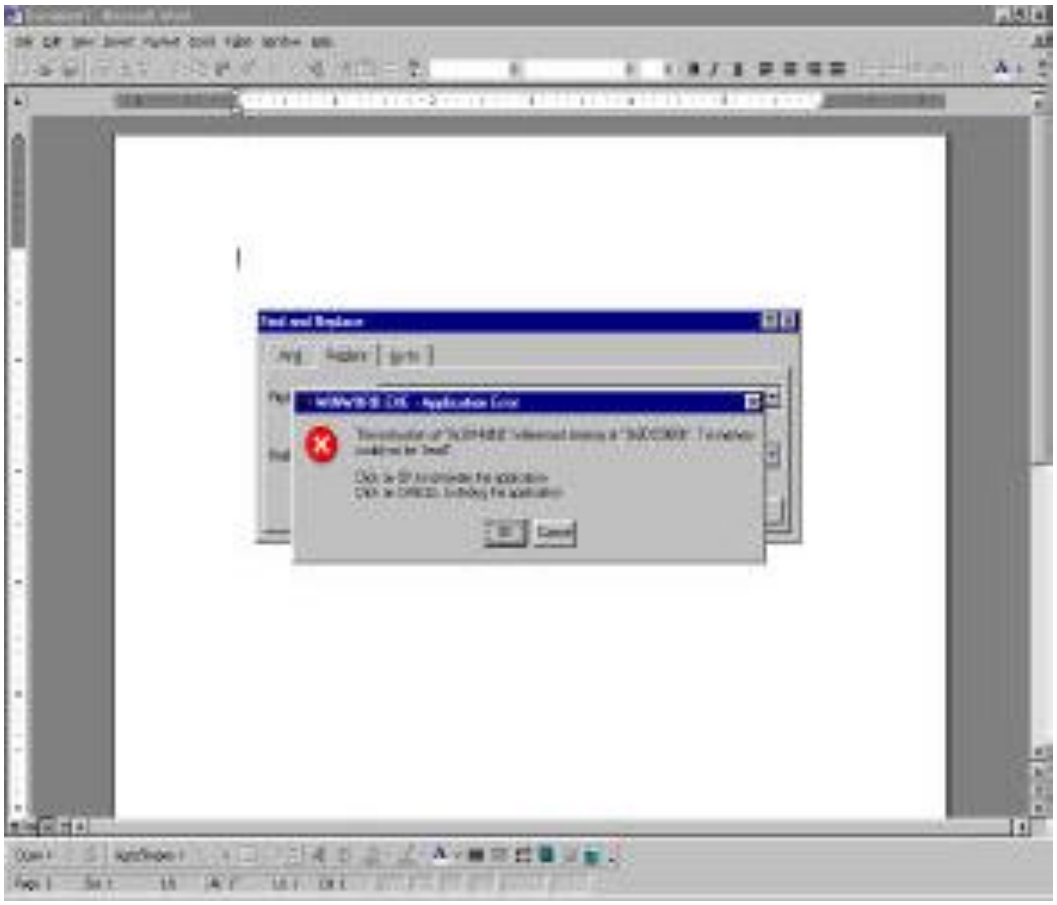


图 11

有意思的是我们注意到发现域是恰当约束的，而替代域没有。

迫使在数据结构中储存过量的数据。

复杂的数据结构，如数组、矩阵和队列，是攻击的主要对象，不但要关注其存储的数据，还要关注其存储数据的数量。

迫使一个数据结构存储较少量的数据。

当数据结构允许既可以增加信息也可以删除信息时，一种常用的成功攻击手段是使 N 连续增加或者混合的使 N-1 移除。

3. 数据访问攻击

我的朋友Alan Jorgensen喜欢用这样一句话“右手不知道左手在什么”来形容这种类型的缺陷。意思很好理解，就是说开发人员使自己为这种攻击广开方便之门：在大多数程序中有很多种方法能做到任何事情。这对于测试人员来说意味着相同的函数可以通过多种调用点来调用，而每一个调用必须要满足函数的初始条件。

一个很好的破坏缺陷的例子是由我的学生在PowerPoint中发现的，该缺陷是关于数据表格大小方面的。PowerPoint创建的表格最大尺寸是25*25。然而，我们可以创建这样一个表格，在程序的另一个位置增加行和列来破坏应用程序。也就是说右手优先已经知道可以允许出现26*26规模的表格，但左手还未意识到该规则。

三、 计算攻击

计算攻击	1、操作数攻击	强迫进行非法操作数计算
		找到非法操作数组合
	2、结果攻击	迫使一个计算结果规模太大
		迫使一个计算结果规模太小
	3、特征交互攻击	寻找不充分共享数据的特征

1. 操作符攻击

这种类型的攻击需要对一个或多个与内部计算操作数有关的数据类型和所允许的数据进行探查。如果能访问到资源，那么就可以得到这些信息。否则，测试人员必须尽最大努力来决定哪些计算要进行，以及哪种类型的数据要被使用。

试图使用一个错误的操作符使计算能够发生。

有时候输入值或者存储的数据是在合法的范围内，但是对于一些计算类型来说确实非法的。除数为零，是一个很好的例子。零是个合法的整数，但在除法运算中作为除数就是错误的。

试图找到不能并存的操作符组合。

需要多个操作数的计算不仅仅是上面提到的非法操作符计算机攻击的对象，也是潜在运算符冲突攻击的目标。

2. 结果攻击

第二种类型的计算攻击，其目标是上溢和下溢存储计算结果的数据对象。

试图迫使计算结果存储规模太大。

即使一个简单的计算如 y=x+1，在边界值附近也存在问题。如果 x 和 y 是两个字节的整数，并且 x 的值是 32768，则该计算将失败，因为计算得到的结果将会溢出内存。

试图迫使计算结果存储规模太小。

和上面提到的例子一样，在式子 y=x-1 中给 x 赋值-32767，得到的结果同样会溢出内存。

3. 特征交互攻击

这是本文讨论的最后一类的攻击，该类攻击可能是所有攻击的祖父，通过它可以将测试新手和优秀测试人员区别开来，这就是所谓的特征交互。这里所指的并不是新问题：共享相同数据空间的不同应用程序特征，或者是对数据处理的不同假设，或者是通过不良负效应的产生，两种特征的交互造成了应用程序失败。

但是哪种共享数据的特征和用哪种冲突方法来解释它仍然是测试领域一个公开讨论的问题。现在我们采用试错法，下面这个例子能很好的说明问题。

这个例子反映了在 Word 2000 中当在单一页面上合并脚注和双列时会显示出意想不到的错误。问题在于 Word 从注解的参考点计算一个脚注的页宽。这样，如果在同一页面上有两个脚注，一个脚注参考了双列的位置，另一个脚注则参考了单列的位置，单列的脚注把双列脚注推到了下一页面上。同样被推到下一页面上的是位于注解参考点和页面底部之间的任何文本。下面的截图生动说明了这个问题的特征。

图 12

图 13

文本的第二列在哪儿呢?是在第二页沿着脚注的位置。你能忍受存在如图中所示这样的文本吗？看来你暂时必须得忍受一下了，除非你能找到一种迂回工作法（在这里我个人理解作者意思是说如果你没有充足的时间来安排文档格式的话，只能先暂时忍受出现这种错误。）

结论

简单的浏览一下上面所述的二十一种攻击，就可以执行一个应用程序中很多的功能。实际上，筹划一次成功的攻击通常意味着要试验很多种可能发生的情况和追踪大量的死角地带。但不能因为这些探索中的一些尝试没有发现错误就否定其没有使用价值。首先，要考虑测试人员熟悉所使用的应用程序的时间，以及熟悉程序中所有可能要实现功能的范围所花费的时间，还有引起额外攻击的新方法。其次，成功的测试人员也是个好的消息。他们表明一个产品是可靠的：尤其是正如上文所述的那样，如果认为一套详尽精确的测试是恶意攻击的话。如果代码能经得住这些攻击，那么它就可能经受得住用户的检验。

同样，当你测试时不要低估在头脑里始终有一个具体的目标的价值。我已经看到过很多的测试人员把时间浪费在拨弄键盘或者随意进行一个 API 函数的调用以使得一些东西中断。筹划攻击策略意味着要有明确的目标，特别是基于那些能引起错误的事情的攻击，我们一定要仔细设计对所调查目标进行测试的详细过程。这样，每一个测试才能更有针对性，测试的过程才能得到有效的及时监测。

最后，要一直牢记测试是件快乐的工作。将测试比作攻击，很生动的描述了测试的特

性，就好像添加了一些调味品作为消遣一样。祝“打猎”成功！

要想对本文原作者的研究有更进一步的了解，以下文献可作为参考：

[1] J. A. Whittaker, “What is software testing. And why is it so hard,” IEEE Software, 17, 1 pp. 70-79, (2000).

[2] J. A. Whittaker and A. Jorgensen, “Why software fails,” ACM SIGSOFT Software Engineering Notes, 24, 4, (1999).

[3] J. A. Whittaker, “Stochastic software testing,” The Annals of Software Engineering, 4, pp. 115-131 (1997).

[4] J. A. Whittaker and M. G. Thomason, “A Markov chain model for statistical software testing, IEEE Transactions on SoftwareEngineering, 20, 10, pp. 812-824 (1994).

[5] J. A. Whittaker and J. H. Poore,“Markov analysis of software specifications,”ACM Transactions on Software Engineering and Methodology, 3, 1, pp. 93-106 (1993).

[6] J. A. Whittaker and M. Al-Ghafees, “Selecting software test data using black-box data flow information,” submitted to ACM Transactions on Software Engineering and Methodology.

[7] J. A. Whittaker and J. M. Voas,“Toward a more reliable theory of software reliability,”submitted to IEEE Computer.

[8] J. A. Whittaker, “Software’s invisible users,” submitted to IEEE Software.

机器与人

作者：Jonathan Kohl 译者：陈能技

我喜欢走路去上班。我喜欢风景、运动，还有走路的时候可以思考我的工作的事情。我有很多好主意都是在走路的时候想到的-远离计算机、出来呼吸新鲜的空气。我的很多主意都是由走路时的观察而触发出来的，走路可以让我思考出很多有创造性的解决方案。

但是走路会很慢，如果我停下来观察那些动物、或者欣赏山那头的日落的话，很可能会赶不上时间。因此，当我需要赶时间时，我会使用代步工具，搭公交车或者火车也需要一些步行，但是如果我需要按时参加一个会议的话，这会比全程步行快很多。我很少开车去上班，因为开车的话我需要全神贯注地注意路面情况，这样我就很少有时间思考、观察周围的事物。然而，如果其他人来开车的话，我就有更多的时间去观察。那条河我其实已经经过好几回了，但是那天当我坐火车经过的时候，我可以从一个完全不一样的角度去欣赏那条河的景色-不用分心在步行或开车。

类似地，在测试的时候，我会寻找很多工具来协助我完成测试任务。如果我需要完整地研究清楚被测试的软件的话，我会做更多的手工测试和分析工作。如果我需要更快地完成任务，或者那些不需要怎么思考和分析的、重复性的工作，我就会使用一些自动化测试。我把这种混合了手工测试和自动化测试的测试称之为“交互式的自动化测试”。与完全用自动化测试替代手工测试相比，这种形式的自动化测试更多地关注如何让自动化测试工具扩展测试人员的能力。而且它可以让我从不同的角度观察我正在测试的软件。

一、 在探索性测试中使用自动化测试

当我开始作为一名专职的自动化测试工程师时，我的目标是尽可能多地把测试任务自动化。我们认为无人值守的测试执行才是理想的测试，因此我们编写测试脚本实现那些复杂的手工测试用例。然而，有些测试用例是很难实现可靠的自动化的，不值得花费大量的精力去实现。很多时候，我们的脚本会漏掉一些 BUG，因为在编写测试脚本时不会想到那些情况，并且有些测试用例对于人工执行而言是很容易的事情，但是对于计算机而言则很容易出错。最后我们的自动化测试有时需要一定程度的人工干预。虽然这看起来有点倒退的感觉，但是我们的自动化测试脚本能可靠地运行，当我们确实需要插手协助手工测试时，我们会发现很多关于被测试软件的重要信息。我们发现我们的无人值守执行的自动化测试脚本能发现的 BUG 有限，比人工执行的少很多，并且不能像人那样报告测试结果。

有时候当我在做探索性测试时，我需要专注于软件的某个深入的区域。如果这时候的测试需要执行很多步骤的话，我会使用工具来自动化界面导航的动作。我会把这些自动化测试脚本称之为探索性测试的“测试设备”（注：原文“text fixture”，怀疑笔误，应该为“test fixture”）。一旦这些自动化的导航脚本完成了，我就会执行这些“测试设备”，观察它运行到我需要接手进行手工测试的地方，然后专注在我的手工探索测试上。

自动化的探索性测试的“测试设备”是增强手工测试的有效工具。你可以因此获得自动化测试的速度、精准度、以及重复性，并且与“好奇”的测试人员的观察、分析研究能力结合。这不仅帮助测试变得更加有效率，而且不用分心与应用程序进行交互，这样可以从另外一个角度观察应用程序工作，因为当我没有亲自操作应用程序、敲击键盘、移动鼠标时，我看的东西会不一样。我可以随时停止“测试设备”的运行，接手进行手工的测试，从而找到重要的 BUG。我发现这种方式是发现测试点的一个很好的视点。

二、 观察、研究、自动化

自动化的“测试设备”是我进行探索性测试的一个很好的起点，但是我很快就学会结合其它形式的自动化测试。例如，当陆续发现一些有时候出现，有时候不出现的 BUG 之后，我们的时间变得紧迫，我们要帮助开发人员“追捕”这些 BUG，从而修改掉。我会使用自动化的“测试设备”来为我导航，然后我再进行相关区域的手工测试。我意识到我要花费很多的时间来探索当前的测试，我修改了一下“测试设备”，让它接收我的测试数据并重复循环地执行。我怀疑那个 BUG 之所以有时候出现有时候不出现，是因为两个用户同时在应用程序的同一个区域执行同一个动作。为了模拟这种情况，我运行一个自动化测试脚本来创建测试数据，另外一个则重复执行测试步骤，而我则手工地在另外一台机器上测试。最后我重现了那个 BUG，因为它需要两个用户同时访问一个普通的管理功能，输入合适的测试数据。

最近，我手工地对一个 Web 应用程序进行测试，这个 WEB 程序允许输入很长的文本。与其人工地输入字符串并计算字符个数，我使用工具来产生测试数据。PerlClip 是完成这种任务的一个很好的工具。其中一种输入方式是“指定个数的字符串”。如果我让 PerlClip 创建一个长度为 255 的字符串，它会很快地产生并添加到 Windows 的粘贴板中。我只需要把它粘贴到应用程序中即可。

PerlClip 帮我节省了一些时间，但是我的测试不仅限于输入数据，还有在每次输入文本并提交后退出、关闭浏览器，这会有点冗余、重复；重复地登录、导航到我需要测试的

页面，这让我变得很心烦意乱。有时候这是有用的，因为它可以让我暂时离开当前的测试区域，观察其它区域的有趣得行为，但是我还是需要最小化“分心” - 我需要快速和精准的导航。由于是在 Java 环境中，所以我使用测试工具 Web Application Testing in Java（WatiJ）来帮助我测试。WatiJ 能像用户一样地操纵 IE，回放测试脚本。

不需要多长时间，我就创建了一个简单的 WatiJ 测试脚本，它可以登录并导航到我需要测试的页面，然后停止。然后我接管 Web 浏览器并输入 PerlClip 产生的数据。这样我执行测试的速度就会快很多，并且我可以在不使用键盘和鼠标的情况下观察应用程序。

这种新的观察视点给我带来了好处，当我的 WatiJ 测试启动一个新的 Web 浏览器进程并试图登录时，我发现处理器时间在每次脚本运行时会略有不同。这引起我的好奇 - 这可能是一个 BUG 的源头。这在我亲自用键盘输入时是不会注意到的，因为它仅仅是一些很小的不同。我记录下这个行为并提醒我自己在完成当前的测试任务后，回来探索一下这个问题。

当我完成我的测试任务后，我决定调查一下登录页面的行为。因为每次登录页面的响应时间仅仅是一些很小的不同，我知道我需要重复一些类似的测试很多遍。我还需要从探索性测试的角度来设计这个测试，因此我需要足够灵活、可随时更改不同输入。我创建了一个新的 WatiJ 测试脚本，这个脚本会被 JUnit 的测试框架执行。这样我可以控制自动化测试脚本，并且从一个不同的、有利的位置观察应用程序。

当我创建了一个可以输入用户名和密码、单击登录按钮，并检查登录错误信息的 WatiJ 脚本后，我开始添加我的测试数据。我把 PerlClip 产生的不同的测试数据存储在 Java 的“.properties”文件中，包含不同长度的字符串，有些很短，有些很长 - 比典型的用户名和密码要长得多。我给我的 JUnit 测试类添加一个 Setup 方法，用于创建一个新的 IE 浏览器进程，导航到登录页面，然后加载测试数据。Listing1 中显示了这样一个测试用例：

```
public void testInputLength() throws Exception {
    ie.textField(name, "user").set(System.getProperty("inputLength.10"));
    ie.textField(name, "password").set(System.getProperty("inputLength.10"));
    ie.button("Login").click();
    assertTrue(ie.containsText("Login Failure"));
}
```

Listing 1

这个 JUnit 测试使用了 WatiJ 类库来运行 IE，在用户名和密码输入区域输入测试数据，然后单击登录按钮。在测试用例中，使用了名为“inputLength.10”的测试数据，意味着测

试输入字符串长度为 10 的测试数据，然后做一个 JUnit 断言（Assertion） - 检查测试的结果，断言一个匹配的错误信息“Login Failure”会出现。这个断言可以精确地捕获任何与指定的错误信息不匹配的消息，而这些可能是我在快速地重复测试时遗漏的。

测试脚本如预期般测试通过了，因此我决定改变测试脚本，使用更长点的输入字符串。我准备了一个 255 个字符长的测试数据，修改测试脚本，保存，然后再运行一遍。我发现这次的登录处理时间要长很多。为了进一步地探索，我在 JUnit 类中创建了一组类似的测试，而且添加了一个“teardown”方法用于在每次测试后关闭 IE。

当我使用很短的字符串和使用很长的字符串时，登录速度的差异是很明显的。我添加一个包含 5000 个字符的测试数据的测试脚本，这是有“预谋”的“攻击”型测试。这次的测试结果是应用程序出错了，抛出一个 Java 异常的堆栈信息。修改测试脚本，经过几分钟的测试之后，我可以断言输入超过 4000 个字符的登录名和密码就会导致程序出错、出现溢出错误。我不能确定为什么应用程序会慢下来，但是我注意到这个现象，设计一些测试，使用自动化测试工具可以帮助我快速地找到一个重要的 BUG。

一位程序员在我浏览我的测试结果时刚好经过，看到那个错误信息出现在我的屏幕上。他拉了个椅子坐下来，“Cool！你用的是 JUnit 吗?”当我给它演示错误重新步骤时，他问道。他找我要了那个测试用例，这样他可以在自己的机器上运行。他修改了代码后，JUnit 的“红条”（测试错误）变成了“绿条”（测试通过），他叫我过去并演示给我看。我们一起改进测试，并签入到源代码控制器，重新构建了一个编译版本。

使用自动化测试工具，结合我的观察和调查分析技巧，我们很快就发现了一个之前没有注意到的 BUG。一个好的测试工具与测试员的分析思维的结合是找到这个 BUG 的“秘诀”。更进一步地，当我使用与开发人员一样的语言和工具时，我们的合作变得更加容易。

三、 手工测试和自动化测试混合进行

有时候，自动化测试的鼓吹者会嘲笑手工测试，而手工测试者则怀疑执行那些无人值守的自动化测试是否精确反映了他们所要做的测试。通常，这种思维上的划分就已经阻止了综合使用两者的技巧的机会。然而，自动化测试和手工测试并不一定非得是对立的。“交互式的自动化测试”在两者之间搭建了一个桥梁，并开辟了一种测试应用程序的新方法。

不同的测试工具可以帮助激发不同类型的测试点。Marshall McLuhan 说：“我们磨利了我们的工具之后，工具就会帮助我们更加强大”。在我使用工具增强我的手工测试之前，我不习惯于简单快速地改变条件、观察。有时候我甚至不会看到一些改变、或者一些在背

后出现的错误，因为我仅仅通过 GUI 界面来测试，没有使用工具来进行更深入的观测。当我确实看到问题之后，我需要更多的时间和思考如何创建测试来执行。现在，我习惯于使用工具来观察服务器、查询那些被测试程序使用的数据，这有助于执行不同类型的测试。我的测试类似于半机械人 - 一部分是人、一部分是机器。

这是最近的一个例子。一个测试人员在 Web 程序的后台数据库执行一些 SQL 脚本时发现有些数据是冗余的，她找我来看。她使用了一个自动化的工具来观测，她估计这个 BUG 可能会在运行类似类型的测试时出现。问题是这个测试需要跑一个很长的流程，涉及若干个界面，输入很多数据，所以手工地执行会花费很多的时间，而且还跟输入的变量有很大的关系，测试人员需要精确地记住输入的所有数据。我在我的机器上安装了 Ruy 和 Web Application Testing in Ruby(Watir)，所以我可以马上开始考虑如何设计这个测试，让这些工具帮助我完成测试。我开始设计测试，这更像试验多一点，我可以重复一个测试用例 10 次，看其中有多少次出现了 BUG。我需要最小化可变的因素，控制我的输入，而且我需要很快地重复类似的测试。我创建了一个 Watir 脚本来模拟用户的操作流程，把所有用户输入作为变量存储，这样我就可以更加简单地控制。这花费了我大概 15 分钟的时间来创建。然后在我的监督之下运行了几次这个测试脚本，观察脚本回放的过程。我们观测到 BUG 出现了几次，并且发现后台数据的冗余在这个过程中出现了。迅速地，我们得到了我们需要的信息，可以告诉开发人员开始检查和修改这个 BUG，并且我们有了一个脚本可以用于 BUG 修改过程时重现 BUG，而且这个脚本也可以用于其它测试任务。

这位测试人员开始时会很奇怪我为什么打开了编程环境，但是当我解释我在做什么时，她很感兴趣地说“能不能给我这个脚本？我也要学学如何做类似的测试!”很快地，我帮她安装了 Ruby 和 Watir 并运行起来，创建她自己的脚本。我突然吃惊地发现 Watir 大大增强了我的 Web 应用程序测试能力。当 Watir 项目 3 年前开始的时候，我是其中一个参与者，并受到它的设计思想的影响，是我进行测试自动化的首选。现在，Watir 和其它的一些工具能很好地整合到我的 Web 测试工作中。我使用使用其中的一些工具来模拟用户操作，而其中的一些用于监测程序，我依赖这些工具的反馈信息来指导我的测试。现在我依赖这些工具，它们已经整合到我的手工测试活动中。很难把我的测试严格地划分成单纯的手工测试或自动化测试。我发现我在这种方式的测试过程中能观察和研究出很多潜在的错误。

把人的思维和自动化测试工具整合在一起能帮助激发更多的观察和发现。当你整合这

些测试活动到一起时，你会发现你的测试旅程变得更加有趣。

黑盒软件测试

译者：韩云玲

域测试-可复用测试矩阵

用测试矩阵解决日常问题

• 在进行了几次简单的数值输入方面的测试后，你已经初窥测试的技巧。边界值分析优点是简易且填充合理，但大多数时候这只是浪费时间。

• 用一个测试矩阵来显示/追踪一系列的测试用例，这与边界值在本质上是相同的。

——举例来说，对于大部分输入栏，你将会做一系列同样的测试： 检查字段如何处理边界，异常字符、功能键，等等。

——作为另外的一个例子，对于大部分文件来说，你将会在文件处理上运行本质上一样的测试。

• 矩阵是一个显示重复测试的简洁方式。

——行中列出你正在测试的对象。

——在列中显示测试点。

——校对下那些你实际完成在单元格中的测试点。

• 矩阵是一个简单测试的简洁组织原，对功能测试和域测试尤其有用。

• 矩阵组测试用例本质上相同。

——举例来说，对于大部分输入栏，你将会做一系列一样的测试： 检查字段如何处理边界，异常字符、功能键，等等。

——作为另外的一个例子，对于大部分文件来说，你将会在文件处理上运行本质上一样的测试。

• 矩阵式结构：

——行中列出你正在测试的对象。

——在列中显示测试点。

——校对下那些你实际完成在单元格中的测试点。

可复用测试矩阵

数字（整型）输入框														
		空值	下边界值	上边界值	下边界值-1	上边界值+1	0	负数	下边若干数字或字母	上边界若干数字或字母	空值（清除默认值）	上边界外若干数字或字母	非数字

这仅仅包含了我通常所用的测试矩阵的前面几列，不过，已经可以表达我的测试理念。

整数-输入测试举例

- 空值
- 有效值
- 下边界值
- 上边界值
- 下边界值-1
- 上边界值+1
- 下边界之外
- 上边界之外
- 0
- 负数
- 下边界若干数字或字母
- 上边界若干数字或字母
- 空值（清空默认值）
- 上边界之外的若干数字或字母
- 非数字
- 错误数据类型（例如小数到整数）
- 表达式
- 空格
- 非印刷字符（例如 Ctrl+字母）
- DOS 文档名保留字（例如*.:）
- ASCII 码上界（128-254）
- 大写字母
- 小写字母
- 修改键（如 Ctrl, Alt, Shift-Ctrl,等等）
- 功能键（F2,F3,F4 等等）

测试矩阵的典型应用

• 你可以为几乎任何类型的变量创建测试矩阵。举例来讲，可以想象列出所有的硬件（包含连接，电力等）错误情况可能会导致一个文件保存操作的失败。

• 你经常可以跨产品和项目的重用一个测试矩阵

• 你可以创造一个与此类似的矩阵用于更广泛范围的问题。无论何时你可以指定多个测试作用于同一类的对象，你期望测试几个这样的对象，那你可以把多种测试都放在一个矩阵中。

• 如果你执行了一个测试，并且程序通过，那么把此单元格标注绿色。

•如果程序失败，则标注此单元格红色，并为此 BUG 填写缺陷报告且记录此 BUG 数。

• 如果测试已经实现自动化，那么在单元格中写下自动化编号或标识符或档案名称。

写文件时的错误处理

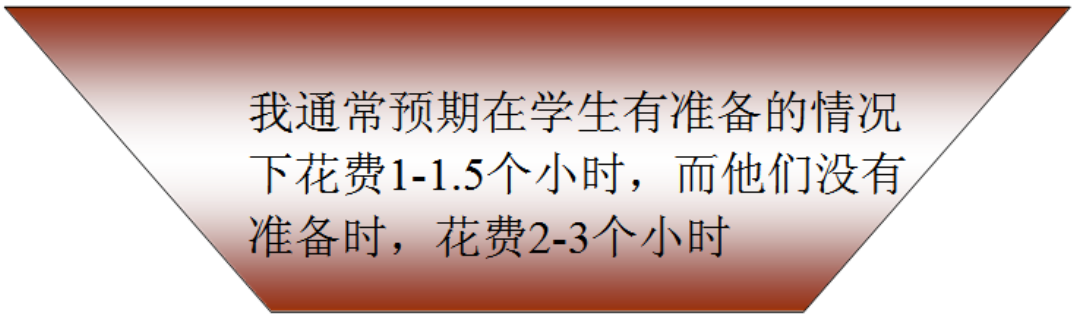
• 本地磁盘空间已满	• 网络磁盘空间已满
• 本地磁盘空间不足	• 网络磁盘空间不足
• 对受保护的本地磁盘写操作	• 对受保护的网络磁盘写操作
• 受损坏（读/写错误）的本地磁盘	• 受损坏（读/写错误）的网络磁盘
• 未经格式化的本地磁盘	• 打开文件后移动网络磁盘
• 打开文件后从驱动上移开本地磁盘	• 网络磁盘等待超时
• 等待本地磁盘回来连线超时	• 保存到网络磁盘键盘/鼠标输入/输出
• 保存至本地磁盘键盘及鼠标输入/输出	• 保存至网络磁盘时其他中断
• 保存至本地磁盘时其他中断	• 保存至网络时本地电力中断
• 保存至班底磁盘时电力中断	• 保存至网络时网络中断

课后作业：文件命名矩阵

• 明天，我们将藉由特定主题的头脑风暴来说明创造一个测试矩阵的过程

• 请在家花 15 分钟时间写下一个文件命名测试的列表。带着你的笔记，写上名字，下次上课前交上一份拷贝。

• 我们将就作业举行头脑风暴会



矩阵构建头脑风暴

头脑风暴原则：

- 不要批评其他人的贡献。
- 笑话亦可，且通常很有价值。
- 目的是获取大量想法，可稍后过滤。
- 记录者和主持人应保留自己的意见。

主持人及记录者原则

- 练习耐性：目标是获取大量想法。
- 鼓励沉默寡言者开口。
- 记录时用多种色彩。
- 附和讲话者所说。
- 记录下讲话者所说。
- 三个 10 原则，沉默是金。
- 转层面分析。

部分参考文献：

- S. Kaner, Lind, Toldi, Fisk & Berger, 扩大决策参与中主持人的引导
- Freedman & Weinberg,检查与技术复审初排手册
- Doyle & Straus,如何使会议起作用

矩阵

• 问题是什么？

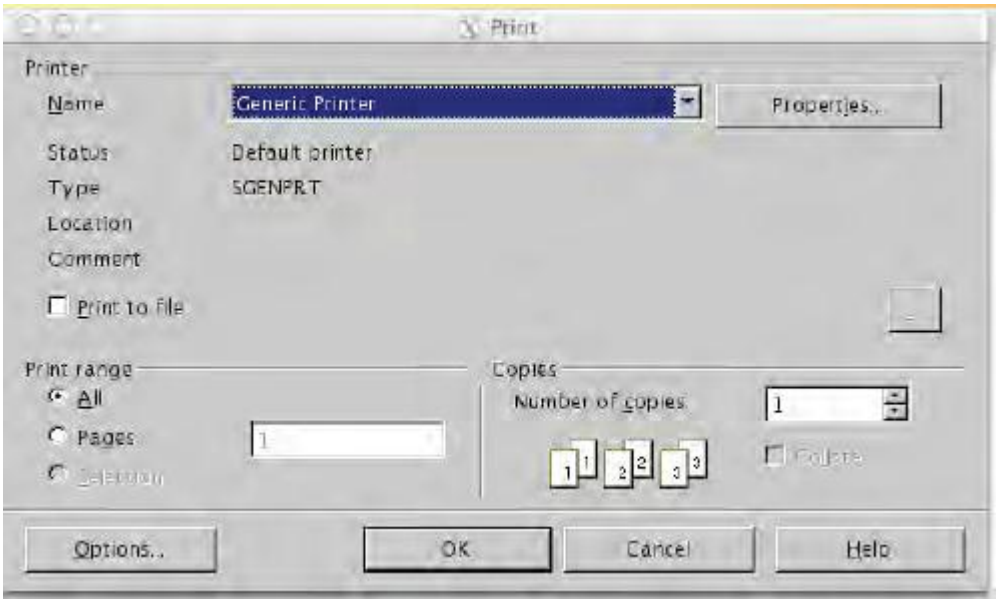
——如果你认为获取的结果过时，那将怎样？（如果这一项目提出了新的计划，为把标准测试覆盖，将做何操作？）

——你是否需要每次都执行所有测试？（或者永远？）

——如果自动化测试编号改变了那将如何？我们仍存在一个并不难理解的维护问题。

域测试-理解域测试

简单练习



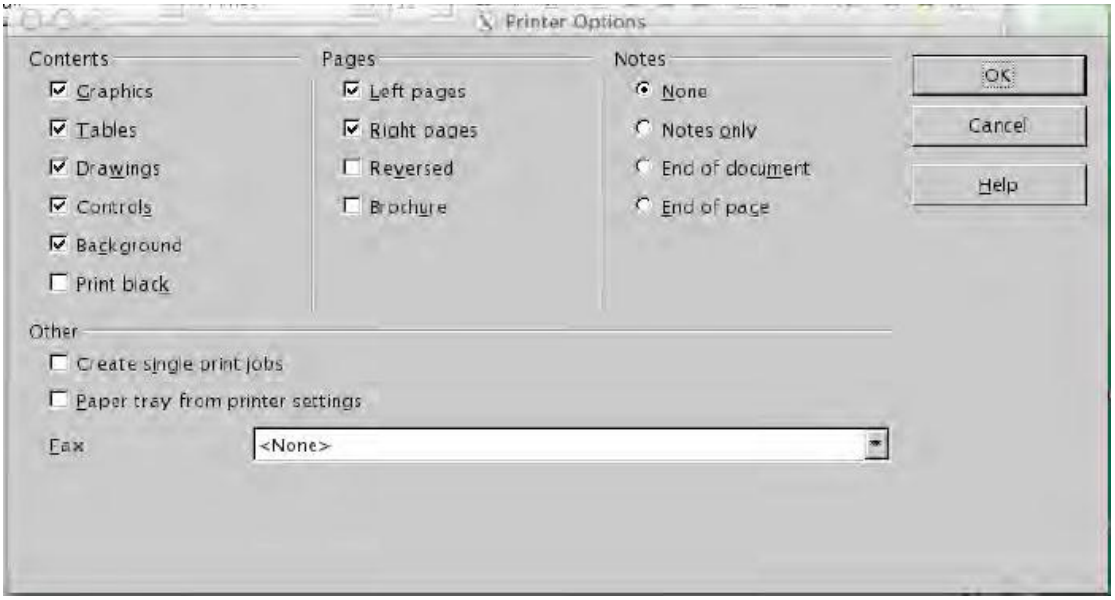
这是一个 Open Office 的打开的对话框。想象

1、 你能输入到拷贝份数区域最大的拷贝数是 999，或

2、 你的打印机管理着多份拷贝数，最大为 99。

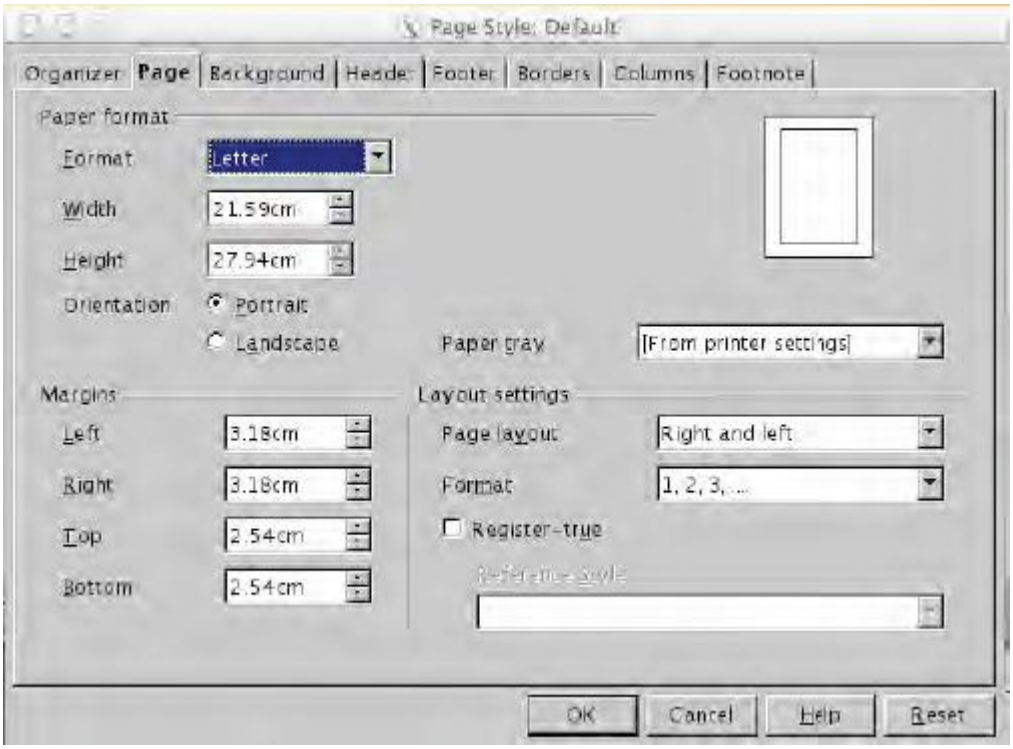
对于每种情况，先做一个传统的域分析。

这些变量的域分析？



- 你会对这些变量做域分析吗？
- 这样做，你将从中收获什么？

浮点类型域分析



- 对页宽做个域分析。
- 想想：这个与整数类型的分析有什么不同？

练习

对于下列各项

-列出重要的变量。

-列出有效与无效的等价类。

-列出测试用例的边界值。

-在边界表中展示结果。

1、FoodVan 为通过网络订货的客户提供食品杂货。为了决定是否购买更多的客货车，FV 跟踪那些要求加车的客户。办事员每天都把电话数输入到数据库中。基于之前的经验，数据库对于数量超过 400 的电话设定为一个挑战（询问：“你确定吗？”）

2、foodvan 提前一天安排司机。为了符合转让资格，一名司机必须有特别许可或者她必须在 30 天内到她将被分配的换班开车。

理解域测试

• 早在有人提出关于域测试的理论描述之前，人们就把数值按等价类来划分。

• 域测试最重要的理念是，它为从域中取样提供了一个理性的基础。

• 定义：域

- 数学上

• 域的功能是定义功能的所有输入值的集合。

• 范围（或输出域）的功能是所有功能可以实现的结果集。

- 前期的域测试描述侧重于输入，不过我们习惯性地把分析应用于输出（测试计算机软件，第一版，1988，反映了这一实践）

在域测试中，我们把一个域细分为子域（等价类），然后使用子域中的数值进行测试

1. 什么是等价类？

什么使得等价类有价值，有四个观点。每一个都具有其实际意义。

- *直观的相似*：如果两个测试彼此如此相似，测试值是相等的，那么看起来两个都测试是没有意义的。

• 这是最早的观点并且也是最容易教导的

- 对于微妙的用例或多变量少有指导

- 明确说明为等价：如果详细说明书中指明程序以同种方式处理，那两个测试值是相同的。

- 测试者抱怨，遗漏了详细说明书可能会花大量的时间在些详细说明书上。

- 注意力集中于被指明的事情上，但这可能导致被指明特征下的更多缺陷。

- *等效路径*：如果两个测试驱使程序执行相同的路径，那值是相等的。（例如，执行 IF 中的同一个分支）

- 测试人员应是一个程序员

- 测试人员应根据代码设计测试

- 有些作者声称，一个完整的域测试将造就一个完备的分支覆盖

- 从类中撷取一个而非其他是没有什么依据的

- 两个值可能使程序执行相同的路径，但却有着十分不同的影响。（例如超时或不超时对后续程序的影响；又如文字处理器以及输出可能相同，但不同的打印机会导致不同的解释或结果）

- *基于风险*：鉴于你的理论可能存在误差，你期待从中获得相同的结果-两个测试值相同。

- 主观分析，因人而异。这取决于你的期望（以及，你可以预期什么）

- 两个值相对于一个潜在的缺陷是等价的，但相对于其他而言是非等价的。

2. 测试等价类中的哪些值呢？

大多数针对域测试的讨论是从几个假设开始的：

(a) 域是连续的 【这点容易被忽视 --CK】

(b) 域是线性的（域中的值可以在数轴上画出）或者，至少，域是一个有序集（给定两个组成部分，其中一个大于另外一个或二者相等）

(c) 导致程序不同分支的比较是简单的，线性的不等式

“远离这些假设是可能的，但成本可能会很高。”--Clarke, Has sell , & Richardson, 388 页

- 如果我们以路径的方式来思考，我们能用值驱使程序执行正确的路径吗？这种方法在以覆盖率为重点的测试中比较常见，但不幸的是，它并不能发现大量的缺陷。参见 Hamlet & Taylor

• 如果你能在数轴上标注输入空间，那么标出从一个等价类到另一个等价类转换的点或区间的相边界标志。据说，这些都是等价类中好的成员，因为，程序更可能在边界处失败。

程序更可能在边界处失败吗？

- 假设程序设计：

• 输入<10	结果：错误信息
• 10<=输入<=25	结果：输出“hello”
• 25<=输入	结果：错误信息

- 一些错误的类型
  - 程序中不适合用数字
    - 所有的数值都将导致错误
  - 不等式说明错误（例如，输入<=25，而非<25）
    - 仅检测边界
  - 边界值误输入（例如，输入<52，位置错误）
    - 检测边界及可能会导致错误处理的数值
- 边界值（此例中，测试 25）可以捕获全部三个错误
- 非边界值（如 53）仅可捕获三个缺陷中的一个
- 侧重于边界本就是基于风险的
- 不过明确提出基于风险的方法更进一步
  - 考虑多种不同风险
  - 按风险划分执行
  - 按风险选择执行的值
    - 如果等价类中没有其他的份子比最佳代表更容易暴露缺陷，那么等价类中的这个份子就是最佳代表（相对于潜在错误而言）
      - » 边界值往往是最佳代表
      - » 我们可找到不属于边界值的最佳代表
      - » 我们可在无序域中找到最佳代表

基于风险的等价类

- 考虑这些用例，它们配对成测试等价类吗？

- 如果你测试	你是否会测
51+52	52+53
53+54	54+55
55+56	56+57
57+58	58+59
59+60	60+61
61+62	62+63
63+64	64+65
65+66	66+67
67+68	68+69

- 给出下列潜在的缺陷

这些用例即使存在也不会触发缺陷	这些用例会触发缺陷

另一个例子：无明确的边界

字符	ASCII 码	
/	47	
下边界 0	48	
1	49	
2	50	
3	51	参考
4	52	计算机软件测试
5	53	9-11 页
6	54	
7	55	
8	56	
9	57	
上边界 :	58	
A	65	
a	97	

- 仍是那个 99+99 的程序
- 输入第一个值
- 等待几秒
- 输入第二个值
- 假设我们的客户端在输入延迟 600 秒以上超时。这对你如何测试是否产生影响？
- 假设客户端从服务上接收数据，客户端没有超时，同时，服务器端延迟超过 300 秒就表示超时
  - 你应用的路径分析中是否覆盖到这种超时呢？
  - 你要测什么样的边界值呢？在哪个域中？

总结：等价类和代表值

如果你期待两个测试有相同的结果（通过/失败），那这两个测试属于同一个等价类。根据定义来看，测试统一等价类的多个成员，是重复测试。

在一个有序集中，边界标志着一个等价类向另一个转换的点或者区间。程序更可能在边界点失败，所以，这些都是（简洁，数字化的）等价类应用的最好部分。

更普遍的是，你期待将一个可能测试空间细分为若干相关的类然后运行其中部分用例。你想从每个类中选择最有力的测试。我们乘这些最有利的测试为每个类的最佳代表。

外部参考：分层抽样：http://www.wikipedia.org/wiki/Stratified_sampling

一个新的边界和等价类表

变量	风险（潜在失败）	应该不会触发失败的类	会触发失败的类	测试用例（最佳代表）	备注
第一个输入	在范围之外的数值上失败	-99 到 99	最小-100，最大 100	-100，100	
	未能将范围内外划分清楚			-100，-99，100，99	
	数字误分类	非数字	0 到 9	0(ASCII 48) 9(ASCII 57)	
	非数字误分类	数字 0-9	48-57 之外的其他 ASCII	/(ASCII 47) ;(ASCII 58)	

注意，我们已经摒弃了“有效”和“无效”的问题。这使得我们无需考虑“有效”概念，对划分策略一概而论。

- 举例来说，打印机等价类（讨论设备兼容性测试，见 Kaner et al，第八章）

有序集举例

这么多域分析的例子涉及数据库或简单数据输入域，部分测试人员不会一概而论。这是一个只适合传统类/边界值分析模板中其他变量的示例。

- 数字范围
- 字符码
- 某些事情做了多少次
（如共享一个产品使用的限额）
（如内存耗尽前可运行多少次）
- 邮件列表中有多少名字记录在数据库中，是电子表格，书签以及所写的变量表达式
- 文件大小（注意特殊值，如正好 64K，正好 512K 等）
- 每页文件大小（比对页边距）（跨页边距，页大小）
- 一页中以内存要求一页的形式存在的文档的大小。这可能仅是以分辨率形式划分的 X 页制解调器等而不同的大小。不过如果我们考虑压缩，那将会更
（有事发生时）加复杂。
- 相同输出事件（如打印文档）
- 数据输入速度（按键、菜单之间的时间间隔）
- 输入速度—处理并发事件
- 连接/活跃的设备数量
- 消耗/可以系统资源（处理，堆栈等）
- 日期和时间
- 变量总和的规模，或其他一些计算值
（想下二进制和数字）
- 你输入（数字位数）的大小或字符串的大小
- 连接字符串的大小
- 路径规格的大小
- 路径名的大小
- 一份文档的大小（字符级）
- 可用的内存数量（>128meg,>640k，等）
- 视觉分辨率，屏幕大小，色帧
- 操作系统版本
- 随一组“兼容”打印机，声卡，调
- 等价事件多次
- 计时：事件 A 和 B 间距多长（以何种次序）
- 超时的时间长短（从刚刚之前到之后）
--什么事件是重要的呢？
- 转换算法（优化）（不同的计算功能）
- 近期的事件，第一个事件
- 输入或输出强度（电压）
- 电压转换的速度/强度（如从极小到极大的声音）

无序集合

• 抽样问题：

- 大概有 2000 多个 Windows 兼容的打印机，且每个有多个驱动。我们不可能全部测试。

• 他们是无序的，但无论如何我们将可能划分等价类，并从中找到最佳代表。

• 这里有两个在 1991－1992 年间发展程序（桌面发行与地址簿）的例子。

那个时间最初的打印机分组：

-原惠普

-惠普- LJ II

-PostScript Level I

-PostScript Level II

-爱普生 9 针等

镭射二代兼容打印机，拥有庞大的群组（或许是 300 台打印机，这取决于我们如何定义）

1. 这些群组中应该包括 LJII，LJII+,以及 LIIP，LJIID 兼容字群组？
2. 什么才是这些类的最佳代表呢？

举例：图形复杂化错误处理

--惠普原二代是一个弱的用例。

举例：特殊形式

--惠普原二代擅长于纸张处理方面。我们工作用的打印机在纸张处理方面却比较差。

我们从同一等价类中抽取不同的最佳代表，这取决于我们试图查找何种缺陷。

几乎等同的打印机的额外疑问举例：

- 同边距，作为惠普原二代对新打印机的抵消？
- 相同的可打印区域？
- 细微差别的相同处理（Postscript 打印机不同）

无序集的更多举例

• 这里有更多的例子是关于那些变量不适用于传统模式的等价类，但是其中我们可以抽取足够多的价值。那，这些的边界用例是什么呢？

• 一个普通组的成员关系

- 如雇员及非雇员
- 如全职，兼职或是合同工

• 等价类硬件

- 如兼容调制解调器，视频卡，路由器

• 等价输出事件

- 或许任何报告都可以回答这个简单的问题：程序将打印报告吗？

• 等价操作环境

- 如 Windows 3.1 的英文及法文版本

变量间的交互

不同于考虑一个单一范围数值的单个变量，一个变量可能有不同的范围，如一个月中的某天，下列日期中：

1-28

1-29

1-30

1-31

我们分析时间范围，并细分每月天数不同为不同的集合

（2 月）

（ 4 月， 6 月， 9 月， 11 月）

（ 1 月， 3 月， 5 月， 7 月， 8 月， 10 月， 12 月）

为了测试，你希望从每个集合中取出一个。有可能会是或不是月份中的“边界”。边界值是这些日期，1-28,1-29，等等

»这是 Jorgensen 提出的不错分析

»软件测试－艺术家的方法

另一个关于交互的例子：

• 当我们思考某些值基于某些输入变量的输入变量时，交互的思考是很重要的。这里是一些在测试中让学生感到头痛的例子：

• I,J 和 K 是整数。有个程序计算公式：K=I*J。对于这个问题，仅考虑你在 I 和 J 中输入整数数值的用例。从 I 和 J（联合）对变量 K 的影响这个点上考虑等价类分析。区分你将执行的边界测试（你将输入的 I 和 J 的数值），如果：

- I,J,K 是无符号整数

- I,J,K 是有符号整数

域测试

• 长项：

- 可以通过相似的小测试集来发现最高概率的缺陷

- 方法直观，简洁，易于教导和理解

- 可很好延伸至向多变量状况

• 盲点或弱点

- 缺陷不是在边界或其他明显的特殊用例中。

•“程序员主导假说”可能有误导之嫌。

- 此外，实际的域通常是不可知的。

- 回归测试依赖的最佳代表常引导我们多测试这些用例或少测试那些值，这些是，或者几乎是，一样好的。

基于关键字的测试自动化

译者：李琴

一、关键字概念

建立基于关键字的测试设计和测试自动化的前提是：构成任何应用程序的离散功能性业务事件可以使用短文本描述（关键字）和相关联的参数值对（变量）进行描述。例如，大多数应用程序要求用户登录；此业务事件的关键字可以是“登录用户”，参数可以是“用户 ID”和“密码”。通过设计关键字来描述离散功能性业务事件，测试员开始建立一个可用于创建关键字测试案例的通用关键字库。这便是创建语言（关键字)以描述应用程序内一系列事件（测试案例)的实际过程。

如果正确地实现和维护，关键字将呈现有关投资的良好回报，因为每一个业务事件都是作为离散的实体设计、自动化和维护的。然后这些关键字可用于设计关键字测试案例，但对关键字本身的设计和自动化开销已支付。当任何给定的关键字内发生更改时，将很容易找出受影响的测试案例并进行适当的更新，另外，关键字本身的任何设计和更新仅执行一次。将其与录制/回放相比（每次运行测试案例时，捕捉特定业务事件或业务事件的一部分) - 如果登录时启动 100 个测试案例，则此事件将会自动运行 100 次，且有 100 种事例需要维护。

1. 关键字开发

关键字的开发应采用与任何正式的开发工作相同的方式来完成。关键字需要设计、编码、实现和维护。

2. 设计

测试设计员负责关键字的设计-关键字的设计至少应包括：关键字名称、关键字描述和关键字参数。

3. 关键字名称

标准的关键字命名规范是先起草，接下来允许设计员进行有效地共享关键字。关键字名称应该以执行的操作 <action> 开头，接下来是功能实体 <entity>，然后是描述性文本 <text> （如果需要），以下是几个常见实例：

- 登录用户 – 登录用户
- 输入用户姓名 - 输入用户姓名

- 输入用户地址 - 输入用户地址
- 验证用户姓名 - 验证用户姓名
- 挑选用户记录 – 挑选用户记录

关键字名称应该是关键字所执行的操作的速记描述。

4. 关键字描述

关键字描述应该描述关键字的行为，并包含足够的信息供测试自动化工程师构建关键字。对于设计员而言，描述是关键字定义，而对于自动化工程师而言，则是功能规格。该描述必须简短而准确-以下是关键字是“登录用户”的示例：

登录用户描述：在登录页面输入特定的用户 ID 和密码，然后按“确定”按钮。

5. 关键字参数

关键字参数应捕捉可影响由关键字定义的实时业务事件的所有业务输入。获取适当的参数列表的最简单可靠的方法是采取“捕捉显示事物”的方法。对于关键字“登录用户”，应用程序显示三个元素：“用户 ID”、“密码”和“确定”按钮 – 用于支持此关键字的两个必需的参数是“用户 ID”和“密码”。“确定”按钮不是必需的参数，因为关键字描述陈述“确定”按钮需要经常使用。如果有多个按钮（如“确定”、“取消”和“退出”），则需要第三个参数“按按钮”，并需要修改关键字描述。

6. 编码

自动化测试工程师选取测试下的关键字名称、描述、参数、测试应用程序和关键字开发标准并构建密码。如果关键字方面有任何问题，自动化工程师将与测试设计员联系以修改设计来阐明关键字的目的。如果存在任何自动化/工程问题，自动化工程师将与开发组和工具生产商联系来找出符合自动化框架的相应自动化解决方案。

7. 实现

关键字实现使用与任何可共享项目资源相同的路径。完成的关键字至少应满足以下条件：通过测试设计员的审核，自动化工程师的单元测试、功能测试，并集成到项目“Testware”中。该过程不需要很复杂或扩展性，但必须确保任何实现的关键字都公布于测试组并能实现预期的功能。

8. 维护

出现以下情况时需要进行关键字维护：检测到关键字故障，更改业务事件或修改关键字标准。关键字维护遵循和关键字开发相同的部署路径：设计、编码和实现。

二、关键字测试案例

关键字测试案例是设计用于对正在进行测试的一个或多个应用程序的一个或多个方面进行测试或运用的一系列关键字。关键字测试案例必须经过设计、执行和维护的。写关键字测试案例是测试设计员/测试员的职责，仅当关键字测试案例执行过程中出现故障时需要自动化工程师介入。请注意：关键字设计范例常在缺少关键字自动化时使用 – 这是一个有效的独立测试设计范例。

1. 设计

关键字测试案例设计包括计划测试案例的目的，使用关键字建立测试案例，以及针对正在进行测试的应用程序来测试设计。乍一看，这似乎与测试案例设计的任何其他方法没什么两样，但是关键字测试案例设计与任何徒手/文本形式的测试案例设计之间存在显著的差别。关键字测试案例设计的特征有：一致性 – 每次都使用相同的关键字来描述业务事件，数据驱动 – 关键字包含执行测试步骤所需的数据，自动生成文档 - 关键字描述包含设计员的目的详细信息，可维护性 – 有了一致性，接下来便是可维护性，最终能够支持自动化，而不需要从测试设计变换到脚本自动化。测试设计员不需要成为测试自动化工程师就能获取测试自动化的权限。

2. 执行

通过按顺序执行关键字步骤，测试员可以手动执行关键字测试案例执行 – 这应作为关键字验证过程的一部分执行。测试案例是使用自动化关键字构建的，可以使用测试自动化工具或集成的测试管理工具来执行。不管是否使用自动化，测试案例执行都是一种机械练习。测试案例应该包含执行测试案例，以及确定该操作成功或失败的所有必需信息。

3. 维护

当应用程序行为或在一个或多个测试案例中使用的关键字设计中发生更改时，必须进行测试案例维护。正确实现的关键字框架将允许测试员通过一些查询机制来查找关键字的所有实例 – 将通常令人痛苦的查找受影响的测试案例的过程减少到一个简单步骤。而且，良好实现的关键字框架应该支持对关键字实例的全局更改。

三、关键字实现

1. GUI (图形用户界面)

基于 GUI 的应用程序的关键字方案是最容易理解和实现的。大多数共享软件、免费软件和关键字测试的商业应用程序都涉及该领域。

2. API (应用程序编程接口)

基于 API 的应用程序关键字解决方案表面上看来很复杂，但这些应用程序一旦细分成离散功能业务事件，其行为便与同等 GUI 应用程序一样了。如果业务事件为”登录用户”，则无论使用哪个应用程序机制来实现该事件都无关紧要，如果业务驱动相同，关键字查找和行为将相同。有几个与 API 领域相关的关键字解决方案提供商，且同一提供商通常具有 GUI 应用程序的解决方案。

3. 电信 (通信协议)

电信领域的关键字解决方案 (例如 SS7)要求对电信协议非常了解. 有几个提供该领域关键字解决方案的提供商。

四、关键字和测试阶段

1. 单元测试

关键字可用于单元测试，但不建议这么做。应该由开发组使用开发套件中可用的工具和技术来进行单元测试。

2. 功能 (综合测试)

关键字测试解决方案专注于作为离散功能业务事件设计和实现关键字，为功能测试提供低本高效并可维护的测试框架。事实上，如果需要或期望基于 GUI 或 API 的应用程序的测试自动化，有一些框架可匹配其短期或长期 ROI (投资回报)。

3. 系统测试

将关键字从功能测试提升到系统测试阶段的基于关键字的测试解决方案可帮助加速测试过程。有效的关键字框架将允许测试设计员把功能级关键字组合成系统级关键字。系统级关键字处理完整的业务事件，而不是组成业务线程的离散功能业务事件。例如，一个系统级的关键字可以是 “完整客户应用程序”，该关键字可以是由这一系列功能级关键字组成：“输入客户姓名”，“输入客户联系信息”，“输入客户个人信息”和“保存客户记录”。

4. 用户验收测试

关键字可用于用户验收测试，但不建议使用，除非这是测试的扩展阶段。终端用户团体使用产品最佳执行用户验收测试中可用的工具、技术和过程。

大卫•W.约翰逊，资深计算机系统分析家，具有 20 多年的 IT 经验，在跨几个产业的业务需求分析、软件设计、软件开发、测试、培训、实现、组织评估和业务解决方案支持方面有着举足轻重的地位。在过去的 10 年内，曾开发了有关实现“Testware”方面的特定专门技术，包括：测试策略、测试计划、测试自动化和测试管理解决方案。在部署全球实时解决方案方面具有丰富的经验，有利于改进软件质量、测试效率以及测试能效性。这使得技术技能、业务知识和应用“正确的解决方案”的能力之间实现唯一结合，从而满足了客户的需要。要联系大卫，请发邮件至 DavidWJohnson@Eastlink.ca

浅谈自动化测试的脚本框架

作者：朱伟

一、 引子

随着这几年测试的重要性被越来越多的公司认识到,测试行业也如火如荼的发展起来,测试不再是可有可无,谁都可以做,也不再是比开发矮一个台阶的职务了,而象征测试的一些特色的技能也被更多的人关注和重视.其中自动化测试就是一个特殊的领域.随着技术的发展,自动化测试已经不是少数几个大公司专用的测试技术,很多公司都开展了这方领域的学习和尝试,整理这几年在论坛上发贴的关注度不难发现,从前的问题很多是询问工具下载地址,破解方法和入门学习手册,而今不少是在谈论自动化测试的构建,这是因为大家在实践中会遇到了一些瓶颈,经常会发现脚本维护的工作量远远高于手工测试.这样的效率不是大领导们希望看见的,也是自动化测试必须解决的问题,于是很多人开始对脚本的管理和可重用性,可扩展性进行很多坚持不懈的研究.

二、 介绍

社会上有一些主流的自动化设计框架的理论和介绍,本文笔者将结合工作使用的 silktest 和 QTP,介绍对自动化框架的认识和在项目的实践的体会,很多观点只代表个人意见.

在此笔者将自动化脚本分 5 层,分别如下:

测试用例层

提取层

应用层

普通层

工具层

三、 解释

以下笔者将介绍这几个层的概念.

测试用例层

该层就是我们实现测试用例的脚本.

在此我把脚本分为两类,一类叫做主脚本,另一类叫做子脚本,从字面上理解,子脚本是

实现各个功能点的脚本,当然也会提取出一些共用的方法,我们会放在提取层中,下面会提起.而主脚本就是根据业务流程的需要来控制各个子脚本,来实现脚本间的数据传递.

题外话,我们会在主脚本中使用数据驱动,来控制脚本的实现各种场景的流程.在此个人感觉 silktest 的 test suites 和 test plan 的概念是一个很现成的管理脚本的方法,而在 QTP 中,我们可以建立一个 main 的 action 来实现这样的功能.

1、提取层

本层是一个项目脚本可维护性的关键,我们会把项目共用的常量,通用的方法,函数放在该层.而这些方法和函数往往是这个自动化项目中最难设计的一块,不同的项目可能使用的方法和思路也不一样.项目中很大的一部分工作量会消耗在这里.比如我会把脚本进入的初始化的方法放在这里,该方法将帮我配置我的运行环境,从服务器上拷贝数据,当然也会针对一些已知的或者或未知的异常情况进行处理,来恢复正常的下个脚本的运行.在 silktest 中,我们会放在 INC 文件下,而 QTP 我们可以自己写一些 VBS 的文件等外部文件来存放和维护.

2、应用层

我们的对象库就是放在本层,做自动化的人都清楚对象库的好坏直接决定了该项目自动化的成败,相对于 QTP 有强大的对象库管理的功能, silktest 相对就是一个比较原始和底层的对象管理模式,更像使用描述性编程的 QTP,当然两者之间还是有很多差异.在此笔者建议假如是大规模的项目,不妨专人负责对象库的维护和管理,该人需要很高的自动化经验,他的责任不仅仅在项目初期把底层的对象库建立起来,在中期和后期对对象库的维护,更多的是针对项目的特殊情况对工具层提供的方法进行继承和扩展.比如我们可以对 close 的方法进行扩展,添加自己的 LOG 日志,甚至可以结合普通层中对于性能的方法进行一些性能数据的采集等.在项目的实践过程中,会有事半功倍的效果的.

3、普通层

这层是衡量一个公司自动化水平的标志,这里集合了很多项目的经验,是一个通用的方法的集合,通常的我们会把自己的 LOG 日志的方法,对数据库的操作等等,当然也有一些通用的函数方法放在这里,比如说上面说到的采集性能数据的方法,捕获异常的方法等等.

4、工具层

该层就是工具提供的一些方法,比如 click,dbclick 等

此外无规矩不成方圆,文档,备注的维护也是不可忽视的重点,一般的自动化测试都会持续半年以上,所以制订相应的流程,文档规范化是一个好的项目必备的要求.

四、 结束语:

本文只是笔者自己的一些经验的分享,希望可以抛砖引玉的引起大家的一起学习进步,毕竟在国内自动化的普及和深度都在初级阶段,更需要我们的一起努力来提高.有时间更希望能拿一些具体的例子来讨论这个话题.

此外本文的题目也让笔者煞费苦心,因为自动化框架的范围很大,而我只是对其中的一部分进行了一些自己的总结,只是实在想不出什么更好的题目,暂且就这样定吧!谢谢。