泽众软件测试网
电子期刊

SPASVO

第三十期

2014年11月

大型验证项目中的数据管理

云中应用性能管理（APM）的下一步

有了测试工具，傻瓜仍是傻瓜

移动app测试中的主要问题

带有机器人框架的.NET自动化测试

测试数据管理：创造性的解决方案

测试中“特殊数据”提出的挑战

智能手机发展中国家测试

上海泽众软件电子期刊

2014 年 11 月 第三十期

主办单位：上海泽众软件科技有限公司

联系电话：021-61079698 传真：021-61079698 转 8017

意见反馈：fangmh@spasvo.com 投稿：fangmh@spasvo.com

公司地址：上海市普陀区曹杨路 450 号绿地和创大厦 18 楼 1801 室

邮政编码：200063

公司主页：www.spasvo.com

目录

大型验证项目中的数据管理

代码验证包括检查一些规则，并且随着项目安全水平的提升，需要检查的规则的数量也要增加。可以用不同的工具检查，验证团队独立于开发团队之外的情况在安全关键系统中并不常见。集中式系统需要处理所有的缺陷信息并轻松生成报告。本文展示了一个开发用以管理大型项目的验证结果的数据管理系统。解决方案基于一个商务智能引擎。

1.引言

代码验证过程包括检查特定规则被遵守且特定代码如指定的执行。可以手工检查，但这是一件很累，易错的事，所以使用自动化工具才是最佳选择。要接受检查的规则数量是可变的且依赖项目安全水平。随着项目变大且安全水平增加，需要检查的规则的数量及要执行的代码变得很高，使用自动化验证工具势在必行。

验证任务由一个团队或一个不同于通常负责开发安全关键系统的公司执行时，所有检查结果都必须被管理以生成缺陷报告并计算可以被及时交付给开发团队的项目进展指标。一般情况下，不同的工具要进行所有的检查（通常是静态和动态分析），因此信息将来自于不同源头并且是不同的形式。管理来源不同的大量信息产生了问题，工具要汇总所有缺陷信息并管理它们以计算指标并以集中方式生成报告。

本文展现了按严格安全要求创建以管理大型项目验证任务的数据管理系统。解决方案是基于一个商务智能引擎的。

本文结构如下。第二部分简要概述了验证任务。第三部分介绍了推荐的验证流程，描述了不同活动和产品。第四部分讲了内部执行和商务智能引擎。第五部分展示了一些使用工具的真实案例结果。最后，第六部分做了一些总结。

2.验证任务

验证包括检查任务中创造的产品满足要求且正确。有各种验证技术，项目定义中的一项重要任务是为制定项目选择正确的验证技术。安全关键的开发是按标准管理的，比如：IEC 61508-3， EN-50128，DO-178B，且根据项目的安全水平，它们提供一系列可以考虑的技术。一般情况下，可以认为，所有标准中都需要有两个基本类型的代码分析验证技术：静态和动态分析。静态分析包括检查源代码而不运行，并寻找被认为是危险的或易错的结构。检查有很多规则，包括代码标准，但是一系列非常重要的规则是来自使用一个不安全语言的情况中的语言子集定义。大多数安全关键系统是用 C 或 C++编码的，且两种语言都被认为是不安全的，所以使用它们的方法是只使用一种被认为安全的语言子集。对于 C 和 C++，有两种被广泛使用的子集：MISRA C/C++和 JSF。两种子集都制定一系列必须遵守且必须检查的规则。动态分析包含执行部分代码以检查它如预期进行。过程包括定义一系列通过它的界面和（结果与规范中一致的）检查去执行的代码测试用例。一般来说，验证任务可以自动或手工执行。优先选自动化方法，但不可能总这样。静态分析可以是自动的，且有商用工具。动态分析不容易自动化，一般而言，它要求手工定义测试用例。另一个限制自动化使用的问题是：在安全关键系统中，工具必须被认证可以安全使用，因此可用工具的数量被减少了。另一个出现在一个真实项目中的问题是，通常，没有一个单独的工具可以检查项目要求的所有的规则。认证工具很贵，有时不会选择去买一些工具。还有一些项目或公司依赖的规则，比如检查主文件格式，对这些任务进行脚本特别检查。一般情况下，验证任务必须包括开发这些特殊脚本的计划。即使是挑选了工具，开发了脚本去执行特殊检查后，一些规则因为它们的复杂性而无法自动化，所以验证过程必须包括一些将被验证和验证工程师执行的手动检查。无论如何，手动检查的结果必须被输入管理系统。验证过程中的另一个重要因素是度量计算。一些度量是由工具计算的且如果值超出范围时它们可被当做一个缺陷（圈复杂，嵌套级别，函数长度等）。但也有其他度量是关于追踪基于随版本而变化的参数的过程演变的，所以有必要保存所有的信息并可随时用于计算这些度量。这些度量的一个实例是 Stability Index，它提供了一个版本间变化的指示，每个版本的既定类型的缺陷数量，等。

3.建议流程

验证流程必须考虑一些可以自动执行的任务及及其他手动执行的任务，但是所有情况下，都要把信息存到信息系统中。基本工作流程如图 1 所示。流程开始输入源代码，最后输出关于代码质量的报告，基本是一大堆没达到的规则。获取数据的流程包括使用各种工具执行所有要求进行的检查。数据输入模块是数据存储的进入点它获取不同来源不同形式的信息并存入数据库。数据来源有三个基本类型：商业分析工具，特别编写的脚本，和手工检查。手工检查中，有一个搜集数据并创建一个（给数据输入模块使用的文件的）特别脚本。数据输入功能使用一个固定格式的 XML 文件。开发的脚本以正确的文件格式创建输出，但是用转化层将商业工具的输出转化为正确的格式。存储和处理系统也能充分利用接收到的所有数据并生成报告。

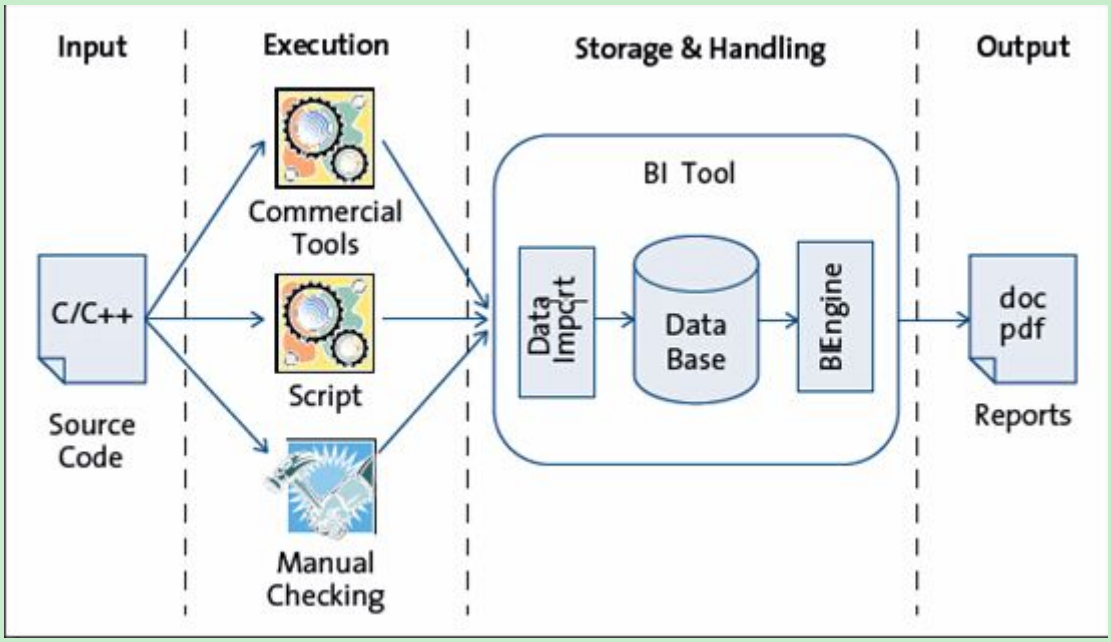


图 1.流程概貌

数据库不允许删除缺陷，但是当就算没有满足规定也有正当理由可以违背时，有一个机制接受缺陷。接受机制要求一个人的识别要负责接受并包含接受的具体理由。可被接受的缺陷可以被总结并列在报告中。图 2 展示了具体流程。如之前所见，静态分析包括自动化和手动任务。自动化任务直接提供一个如输出所示的 XML 文件，但是手动分析的结果也必须被处理以生成一个 XML 文件。动态分析并不是完全自动化的。测试用例必须手动定义，但其执行是自动的，结果被存入 XML 文件中。也要给数据库提供关于度量和检查规则的信息。

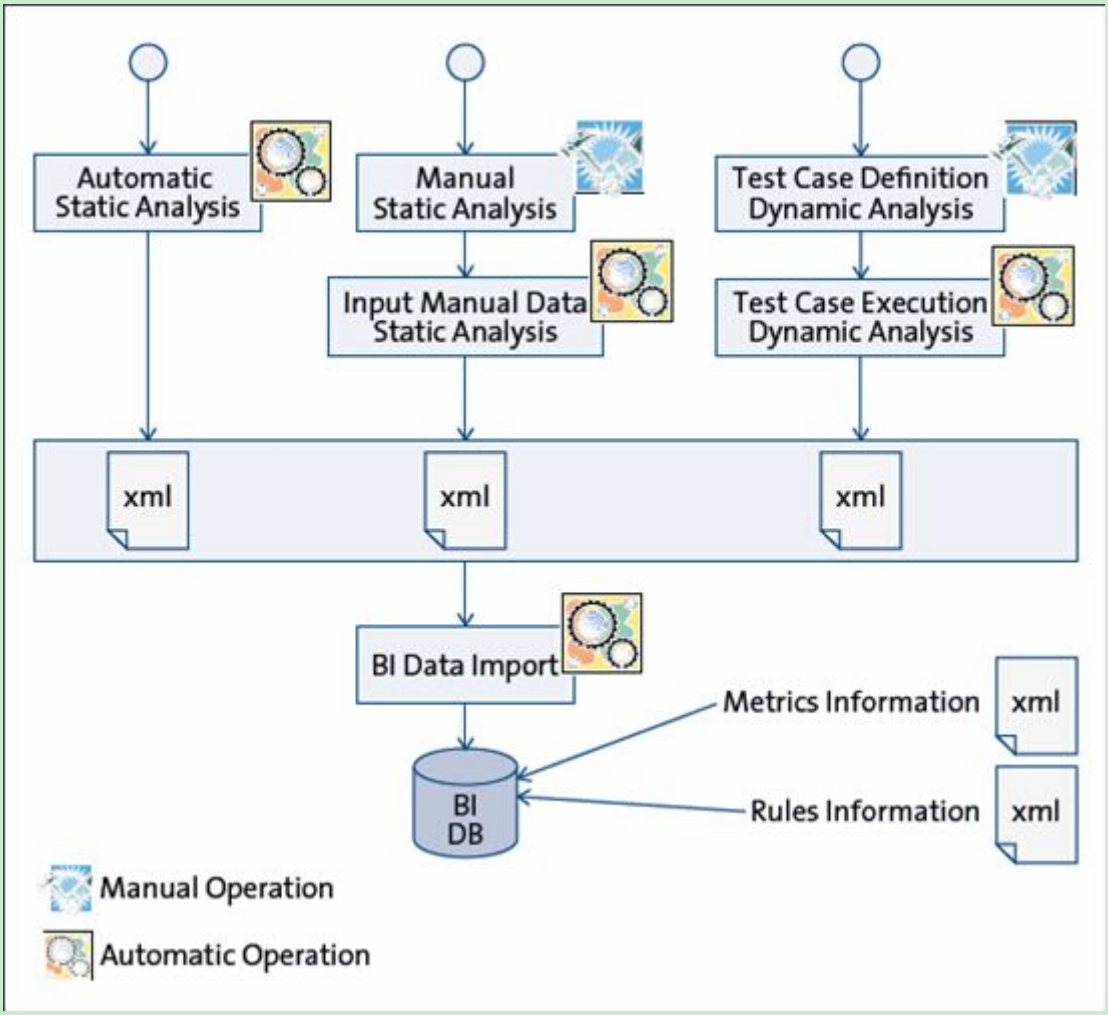


图 2. 流程详图

4.解决方案的内部结构

存储和处理系统构成缺陷管理的 BI 核心并为缺陷版本，列表及报告生成提供基本功能。数据库逻辑上是由三个不同类别的元素构成：缺陷，规则和度量。基本元素是缺陷，它包括明确识别缺陷所需的所有信息。作为检查流程的结果，缺陷记录是为每个规则或失败的动态测试而创建的。每个缺陷结构中包含的基本数据如图 3 所示，且由本地数据（项目，模块，文件，线路，等），尤其是关于缺陷，已发现缺陷的工具还有临界等的信息。系统必须熟悉规则列表。数据库规则元素包括识别规则所需的所有信息。规则结构包含关于项目中检查的每个规则的信息。创建包含除所发现的缺陷外的所有被执行的测试时，就需要该信息。图 4 展示了为规则存储的，包含了规则的基本定义及规则是否是主动的基本数据。度量结构包含关于每个计算度量的信息。创建报告也需要该信息。图 5 展示了为一个度量存储的基本数据，还包含了关于识别度量的信息，尤其是有效期的临界值及位置，以防数值超出范围。

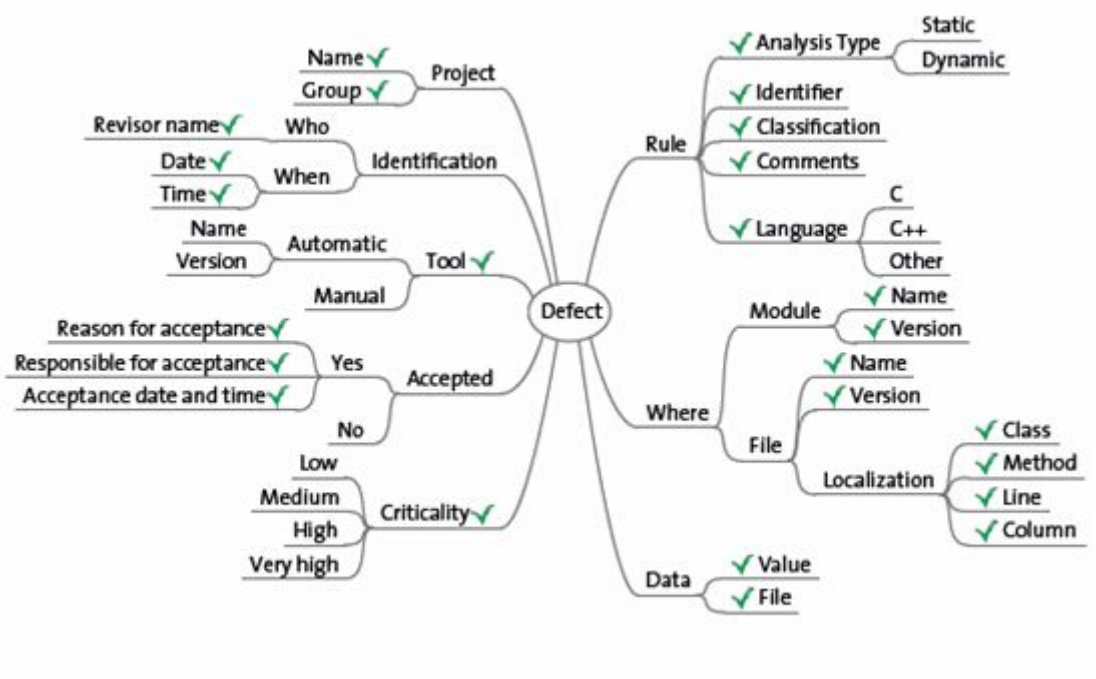


图 3. 缺陷数据结构

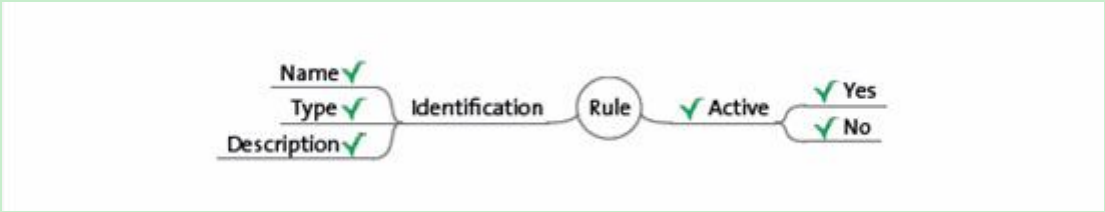


图 4. 规则数据结构

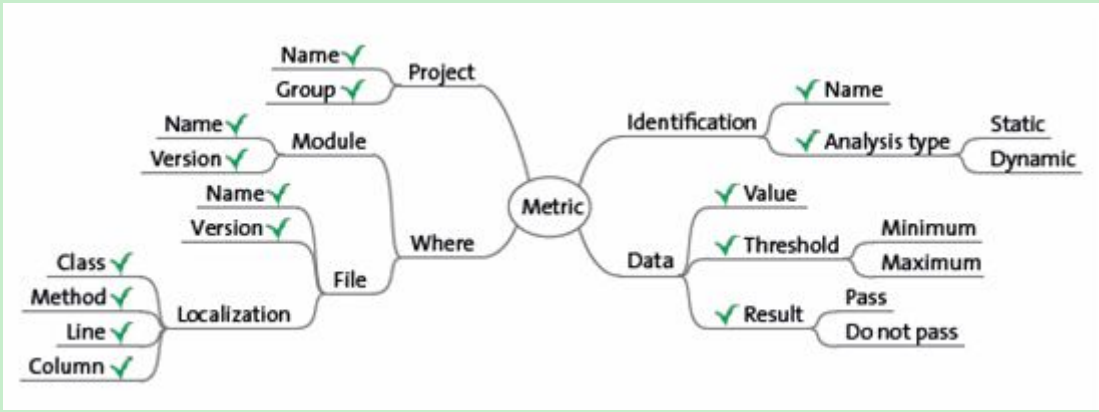


图 5. 度量数据结构

BI 系统的配置由一系列（定义需要检查的规则，包括它们的有效范围）文件构成。如果系统中有一种在配置中不具体的缺陷，一个错误就被解决了缺陷就被当做未知纳入系统。如果配置文件被升级增添新规则，那么之前未知规则的缺陷就被发现并充分纳入。内部 BI 结构由一个相关数据库组成。因此，前面图中展示的缺陷，规则和度量都被模化为相关表格。真正在使用的系统基于一个 SQLServer 数据库，该数据库有一个 web 应用程序，使用 SP Net 和 Web-IIS 用作 BI 引擎。这个结构可以移植到其他数据库和 web 服务器上。

图 6 展示了 BIweb 用户界面的主要前端。它已被简化，可以很方便地获取信息。缺陷和度量在不同的页面上。一条线路显示每个元素（缺陷或度量）。一条线路上显示的领域可以在任何地方添加，移除和安置，过滤器可以包含在内。这是生成报告的方法。该系统可以输出结果的电子数据表。

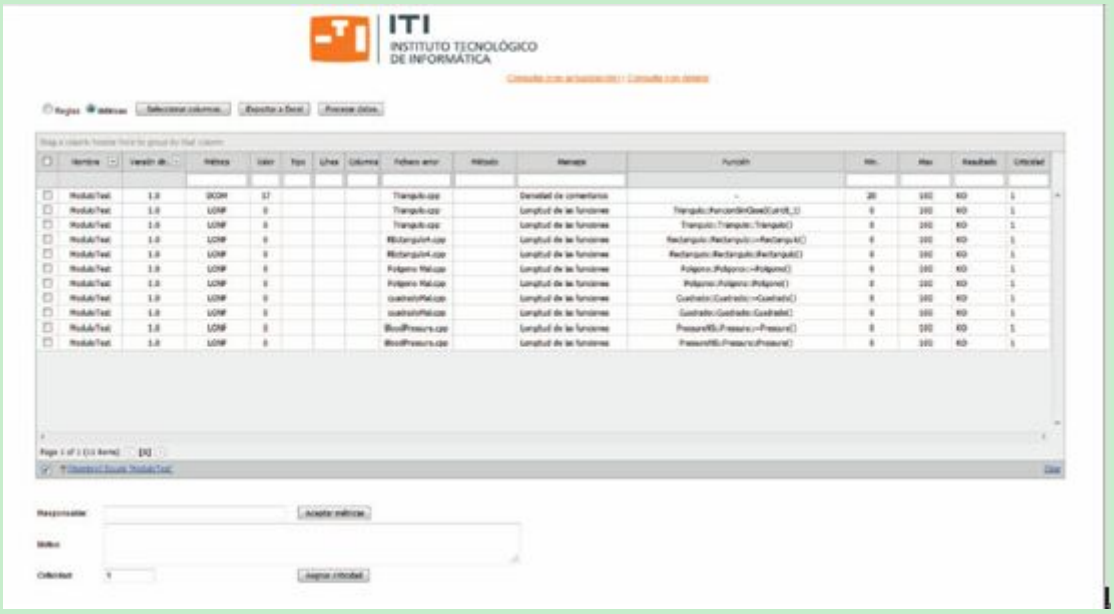


图 6. BI 用户界面

5. 使用实例

创建所描述的系统对在（软件质量部门作为其他部门的 V&V 设备的）公司里执行的多种内部项目有帮助，还对安全关键的项目（这些项目中，规则数量和工具种类，如前所述，真正充分挖掘了这样一个工具的作用）尤其有用。 例如，系统被用于 C 和 C++（包含 MISRA C/C++）开发的有超过 500.000 行代码和大约 450 个需要检查的规则的安全关键验证项目中。

作为一个外部验证团队，基于使用所示系统的工作流程包含了以下步骤：

1.源代码来自开发部门。源代码单元是一个模块（和一个明确的功能及界面一样），且它由一个版本号识别。

2.静态动态分析是由 V&V 工程师单独进行的。它可以由不同的工程师在不同时候执行。

3.每个分析的结果都被导入系统。

4.使用 BI，结果是由验证经理检查以找到假阳性或可以打破规则的例子。一些缺陷的状态可以改变。

5.报告由工具自动生成。

6.报告被发送给开发部门。

使用所描述的流程有蛮多显而易见的好处的。所有 V&V 活动以任何顺序进行，最终结果被存储到中央数据库，最开始，甚至可以在分析还未完成时就轻松创建报告。工具也提供一个机制来跟踪被接受的缺陷并生成特定报告。

6.总结

有很多可以使验证任务进行机动化并有所帮助的工具，但缺少管理来自不同工具且形式不同的数据的工具。开发的系统已被证明是一个很有用的工具，有助于减少创建报告的所需时间并可以轻易获取缺陷信息。

云中应用性能管理（APM）的下一步

新 APM 强调每笔交易的成本;帮助测试员提升其在将投资最大化中的战略性作用

企业进入云就会产生许多利益，包括业务更灵活，显著节省成本，当然还可以增加利润。连续应用程序性能管理（APM）支持整个应用程序生命周期，从多方面来讲，可以是实现这些利益的关键。近期，云中 APM 的重要性备受关注，尤其是当它涉及确保快速，可靠的终端用户应用程序体验时。云是不透明的，也就是说，使用云的应用程序的企业常常不深入洞察内部工作和他们所选择的云服务提供商的容量管理决策。只有一个方法让企业确信终端用户正在用基于云的应用程序得到可靠的体验。测试员需要从云“另一边”的真实终端用户的角度去估量性能。因为他们在“前线”，基于云的应用程序中发生性

能问题时测试员是第一个做决策的。然后深入诊断可以帮助确定性能问题的原因，无论是云，企自己的数据中心还是应用程序交付链中的另一个元素。如果问题原因是云，企业就可以利用这个有用信息警告云服务提供商并维护应用程序重绩效服务水平协议(SLAs)。

许多企业和他们的测试员已在使用 APM 以确保高水平的员应用程序性能方面做出显著进步。他们的应用程序或许已经够快够可信了，但却没有时间停止在云中做 APM 更好地理解基于云的应用程序的内部工作可以获得巨大的利益。云中 APM 的下一步就是优化基于云的应用程序的成本结构，就像应用程序性能对云投资回报有直接影响一样。再一次，测试员有战略性机会去帮助最大化云投资并提高底线。

换句话说，云所关注的是，APM 不仅仅是使应用程序更快，它还使得应用程序，尤其是最高产的应用程序成本效益更高。应用程序开发过程的每一步都必须将之考虑在内，因为即使是所谓的最好的应用程序也绝对不是完美的。持续改进，修正和优化可以极大地影响一个基于云的应用程序的性能和成本效益。 因此，回归测试和整个月影传说生命周期中都需要考虑这两方面。

再看搜索功能，除了快，测试员还需将之优化以便它可以提供更好的结果且不被每位测试员执行五遍甚至更多遍。这可以被视作功能优化，也可以降低执行成本，因为在云中，每笔交易都价值一美元。每笔交易减少数据库获取调用，同时一点也不加速搜索，可以节省成本。这是因为大多数云提供商按执行次数要价，比如 SOL。所以优化 SOL 的数目可能比节省 CPU 更具成本效益!用这种方法， 企业就完全绕开了资源优化的问题而直击成本优化问题，测试员可以真正地掌控要价。事实上，企业一点都没必要关心公共云中资源的使用，他们需要关心真正的终端用户应用程序 SLA 和成本效益——越来越多地，测试员是该知识的承办商。

另一个例子是购买功能。云中有没有终端用户花很多时间在上面，并消耗许多资源的特点——比如：产品简介或产片图示？通过了解终端用户怎样运用他们的时间，测试员深刻了解了在云中什么可以提升价格。此外，通过理解交易的成本结构及它会产生多少受益，测试员可以更好地设置优先级以便转变前的交易的功能，成本及性能可以被优化。

对于许多企业来说，迁移到云中的主要的明显的优势是可变的，这就避免了容量规划和大笔的预付费用的老办法：反之，企业可以，随其负载的增加，扩大他们的环境规模。但是可变性也有其弊端。很容易导致过度消费的计划容量，因为没有硬性限制，这就会超出成本估算。

因此，测试员直接理解终端用户怎么与应用程序交互很重要以及应用程序如何处理负载。这样，测试员就可以提供有价值的可以帮助引导更明智的云容量决策的信息。没有这个信息，执行就很盲目。

总之，终端用户体验管理使测试员理解了终端用户的行为及性能是如何影响转化率和业务的。但在一个公共云上，这只是 APM 的一部分。只有当企业 可以在保持成本不断降低的同时，满足终端用户对快且可信的应用程序的期待，他们才能取得云的成功并利用云提升他们的业务性能。通过专注于每笔交易的成本，测试员可以对企业盈利做出更大的战略贡献。

有了测试工具，傻瓜仍是傻瓜

测试工具：人们总是认为测试工具是每个测试难题的解决方案。有了工具实施，测试就会进行地很快，质量更高，自然也更便宜…… 可惜现实却是，测试工具实施要花上不少钱，而且投入还不一定有回报。到底为什么测试工具实施经常失败呢？

测试工具是什么？

为了了解测试工具实际是什么，首先考虑一下我们可以识别哪种测试工具很重要。人们通常认为测试工具只是用于自动执行测试的工具。但是，远不止如此，它们还是：（测试）管理工具；bug 追踪工具；版本控制工具；一般性（如：电子数据表）工具；自动代码测试工具；性能测试工具；当然还是自动执行测试的工具。

一般来说，测试工具帮助测试员做测试工作并使之更高效。这就是说，测试员可以利用工具更快，更好，更便宜地做任务。但是，测试自动化的这些好处没有一个可以保证。一切要看怎么使用工具及它们该如何被使用。

测试工具的实实施

大多数在 IT 业工作的人对许多 IT 实施失败一点也不惊讶。查一下失败率，你会发现基本超过 50%。但是，当一个组织决定实施一个测试工具时，突然人们就会期望实施不会遇到任何障碍。自然，将一个大型 IT 实施与一个相对较小的测试工具的实施相比较是不公平的。但这两样都在最终软件实施中确是事实。也意味着有风险和失败的可能性。既然我们已经肯定了这个事实，那么看一看可能会失败的事也不错。

在高水平上，这些可以被分为三个单独的元素：

▪人

▪流程

▪技术

我将反序谈谈这三个元素以及它们在测试工具实施中所扮角色，首先是最不重要的元素——技术，最后是最重要的——人。

技术

讲到测试工具，多数人立马会想到技术。一份测试活儿在选定平台上吗？它适合其他工具和/或被测软件吗？有足够多的硬件去运行工具吗？显然，这些都是实施测试工具时很重要的问题。如果工具不起作用，那它还有什么用呢？这就意味着一个测试工具的实施（或任何其他工具），非常有必要调查特定工具后的技术并将之放在组织内使用的技术一起。然而，正如之前所提到的，技术是实施测试工具时最不重要的元素，因此在试着实施测试工具时要最后考虑它。比它更重要的是就绪的或将被设计的流程和该工具将在这些流程中采用的部分.

流程

在我们开始谈论实施测试工具中流程的重要性前，让我们先看看“流程”到底是什么。牛津字典将它描述为“为了实现某特定最终目标而采取的一系列行为或步骤”。一个流程的行为或步骤是一系列的一部分，表明它们要按特定的顺序进行。这一系列动作的目的是获得特定结果。充分测试测试特定 SUT 以保证软件的特定质量水平。那么一个测试工具在这样的流程中起着什么样的作用呢？基本上，一个工具是用来使使用它的人生活的更轻松；它应该帮助任务更高效。因此，一个工具帮助流程就绪，例：通过让用户按特定顺序采取特定动作。也可以帮助使特定动作更简单且/或更好且/或更快。谈到一个项目中的测试工具时，就必须决定应该或可以用特定工具改进流程的那个部分。测试工具并不是一个小玩意。它不是要让你的同事对你用的新技术钦佩。它是改进流程以便更快、更好、更便宜地实现目标。在说任何工具或技术前，有必要问问你自己“为什么我们首先想要实施工具？我们真的需要工具吗？”最好再问问“我们能改进流程吗？”——丝毫不考虑使用工具！

现在，如果组织内部现存流程事实上可以改进且一个工具可以有效帮助改进，那么这时候就该决定哪个工具能够做到改进流程。将需求列出来或许是个不错的开端——考虑一下 must-haves，should-haves 以及 could-haves。不看技术。忽略需求或要求或许很诱人，因为你觉得他们无法满足现在的技术。当你列出可能会改进流程的工具需求时，就是时候看看特定工具和技术了。但是，看你的（现在或将来的）流程时，一定要考虑:任何流程都可能会在取得目标时失败。这就把我们引向了看测试工具时最重要的因素：人。

人

人可以创建或破坏任意一个项目。没有专注的人，最终任何项目都会失败。因此，人是迄今看测试工具时最重要的因素。实施测试工具时，首先要考虑的就是讲使用工具的人。当人们对现在的流程感觉满意且不懂为什么要改动时，那么最聪明的做法就是一点儿都不要改动。或者你可以让相关人员看到需要改进的原因以及改进后有啥好处。审视人这一要素尤其是测试员时，不少迹象表明需要对现在的流程做出改进，测试工具的实施要适当。例如：

▪测试员不再认为其工作有挑战性了。它成了一项例行公事。

▪测试员努力找出做手头任务的动力；他们更想接触新事物而不是一遍又一遍地执行同样的老测试。

▪测试员觉得他们的工作过时了。

▪测试员喜欢新技术的挑战，甚至可能为了最先进的测试工具是日常工作一部分的工作而离开当前工作。

▪▪测试员以前已经用过测试工具且信任它们。当这些迹象在一个项目中呈现出来时，明智的做法是深入调查工作上到底正发生什么。要做的事之一就是挑剔一下现在的流程，看看它们是否仍然可行，是否可以改进。测试工具的实施或许可以帮助改进流程并使测试员在他们的日常工作中更开心。但是，把事情安排地有条不紊很重要。一个测试工具绝不能成为任何问题的解决方案，它只能帮助解决问题。因此，审视测试工具的成功实施，总会按顺序用到人，流程，技术。人应该被包含在内，且大多数会选择改进流程。需建立流程并使之在引入新测试工具前要达到成熟的水平。做到这一点，那么就是时候考虑工具和技术。

为何测试工具实施会失败？

既然我们已经了解了人-流程-技术顺序的重要性，就有可能回答“为何测试工具实施会失败？”回答通常是技术被放到人和流程之前。当一个流程还不够成熟，那么工具绝对无法改进流程的。最大可能是强调流程仍没效率这一点。因此，在一个失败的流程里引入一个测试工具只会使问题更严重而无法帮助解决问题。当优先考虑技术而非人时，更有可能测试工具实施会失败。当人们对其在做的工作不满意时，单单一个工具也不能让他们突然满意起来。

生产效率低时，可能是流程出了什么问题。工具不会自动提高生产率。最后，当引入一个不被人们支持的测试工具时，实施必定失败。人们应该看到改进某特定流程的必要性，且应该意识到引入一个特定测试工具可能最后能帮助更高效地完成他们的工作。如果一个组织没有把这三点按序排好，那么“有了测试工具，傻瓜仍是傻瓜”这句话就成真了。或者换句话说，一家只为实施测试工具而实施测试工具的组织终会出丑的。

移动 app 测试中的主要问题

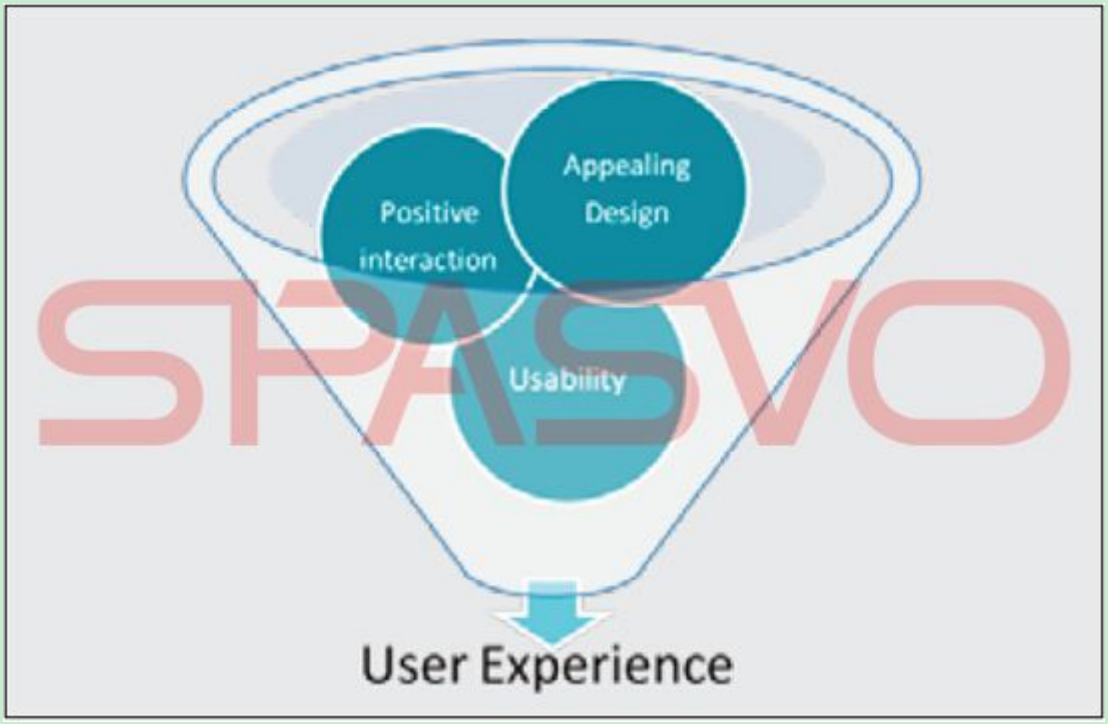
对于每个使用智能手机或平板的人来说，app 都是不可或缺的。一个 app 是针对一个特定环境开发的。在移动背景下，通常被称为移动 app。App 技术自 2007 年 iphone 的商业发布时就为人们所熟知了。苹果 App 商店的推出，打开了软件 app 的一个新的销售渠道。随之，用于其他操作系统的类似 App 商店，例如安卓市场，诺基亚商店，黑莓 App 世界，Mac App 商店，三星 Apps，以及 Windows Phone 市场，不久后也一一推出了。很长一段时间，大家只关注用于个人使用的 app，但现在情况改变了。 App 的商业用途越来越重要。企业正在使用金融，销售，市场或内部沟通的 app。此外，通过 web 服务或云平台可以与后端工具及移动设备进行交互的 B2B 或企业 app，正在不断占据市场份额。这一发展过程中，对有条不紊的质量管理的需求正在不断增加。

本文将展示移动 app 测试必须解决的四大问题以及所需的基本要求。而且还会描述一个敏捷开发模式中的测试方法集成：

- 用户体验
- 可用移动硬件的覆盖
- 连接
- 安全

用户体验

用户体验是一个 app 成功与否的关键。App 商店中 app 评分可以反映用户体验，差评说明销售亏损。所以用户体验在移动 app 测试中是一个重要问题。因为体验具有主观性，所以它无法直接被测出。但你要知道，你在测试过程中必须要将好的用户体验包含在成功的关键因素之中。



好的用户体验的成功因素包括吸引人的设计，积极的互动性和可用性。基于这三点，你可以得出以下三个测试标准：

User Experience	Test	Factors
Design	Layout	continuous, relative
	Navigation	logical
	Position of icons	logical, clearly laid out, functional
	Different display format	suitable
	Readability	look, display duration
	Language	grammar, spelling
	Text display	alignment, overlap, text wrapping
Interaction	Touch screen	gestures, input
	Motion sensor	pan, overturning
	Error messages, warnings	comprehensible
Usability	Screen set-up	logical, duration
	Action chains	logical, comprehensible
	Progress bars	timing
	Performance	load duration, multitasking

可用移动硬件的覆盖

移动设备的种类，尤其是安卓的正不断增加，而且不再可能概述移动设备市场。因为显示屏大小和现状、操作系统版本和基本设备特点等限制因素的不一致，质量管理很困难。还有一点，设备制造商的（安卓）操作系统的设备特殊定制，应明确地在设备上进行测试。测试时，有必要限制将会被 app 支持的设备。必须要标明系统组合（硬件/操作系统）和向下兼容性。为了最大化系统组合的覆盖，就必须要有灵活的策略来执行测试。

变量有：

▪仿真器，模拟器

▪beta 测试网“现场测试”

▪云移动测试，如 Mob4Hire， testCloud

▪众包移动测试，如 PerfectoMobile， Soasta

必须将这些变量组合起来并优先考虑项目背景。

连接

连接是移动 app 测试的另一大问题。理想状态下，app 的网络连接将在单独的线程中实现，这样就不会彼此干扰。此外，一个 app 必须适当地应对网络中断、延迟、变更和信号弱等情况。

如果这些应对由一个警告或一个缓存机制实现，那就应该在项目背景中标明。

安全

除了连接，安全是另一主要问题。

要点有：

▪认证和授权

▪数据安全

▪离线模式的数据分配

▪文件系统、选项、硬件、网络资源等

基本上，一个 app 的开发会在一个“沙箱”中进行，这样可以启用或禁用 app 外的资源。根据操作系统，访问权限是可编程或提前确定的。

明确确定适当的测试需要。

安全测试背景下的另一方面是防止通过“越狱”（iOS）或“刷机”（安卓）来非法获取。

除了这四大问题，移动 app 测试还要考虑一些基本要求。

移动 app 测试中的基本要求

App 测试的一个要求是一个 beta 测试环境。为此，操作系统制造商要么提供他们自己的测试环境，比如：Testflight (iOS)，要么可以使用任一商用工具，如：HockeyApp (Android)。

移动 app 测试市场上有许多测试工具。

不同之处在于用于建立测试集的增加模块如 Tosca Mobile，或有标准接口的独立工具。

在移动背景中，测试自动化是一个重要因素。一个众所周知的工具是 SeeTest (iOS/Android)，，Robotium(Android) 和 MonkeyTalk (iOS/Android)。所有要求都必须被有条不紊地测试。这里我描述了一个方法。

敏捷开发模式中的测试方法集成

App 是基于基础软件开发模式而开发和测试的。传统和敏捷模式都是比较常用的。

传统软件开发模式，像 V-Model，有高规划可靠性、标准以及简易的可扩展性和实用性等优点。缺点是：对文件和必要的调整有高要求，缺少灵活性。

相反，敏捷模式有高灵活性，快速系统部署，低文件需求，且基本不拘泥于形式。缺点是：时间和预算编制困难，敏捷团队内部有很多交流工作要做，尤其对团队成员的个人能力很依赖。

总之，app 是在紧张的“上市时间”周期内开发的。定制是通过不断升级执行的，基本是 app 用户的回馈结果。

这种种情况下，一个敏捷开发模式比传统模式的优点更多，因此我们在这里简单介绍一下。

敏捷模式中的测试工作可分为三大方面：

每个方面都由适当的符合 sprints (Scrum)时间表的测试方法支撑。使用敏捷模式时，有必要确保：

▪明确所完成的是一致的

▪ sprints 中详细计划了自动化，建立了模拟服务

▪ sprints 全程监控回归

▪明智地集成外部服务提供商（crowd， beta-tester network）

▪在过程中应用所获的测试经验

注意：并不是每个项目中敏捷测试都是最好的解决方案。

传统模式通常是更好的方法，尤其是在有明确而详细需求或紧缩预算和规定时间的特定公司问题的项目中。但是，如果使用一个敏捷模式就有必要持续监控产品积压以及适应其的基线。

总结

移动 app 测试中必须考虑很多方面重要的问题是用户体验，可用移动硬件的覆盖，连接和安全。一个移动 app 测试项目的质量标准是每个项目的个体，即测试标准的权重变化很大。将测试集成到基础软件开发模式中对一个成功的移动 app 测试项目至关重要。因为移动 app 项目的紧张时间表，有必要定制测试方法，调整测试范围并明确测试覆盖面。必须快速灵活地适应采取的 App 开发决定因素的方法。

带有机器人框架的.NET 自动化测试

测试自动化的好处我们都很清楚，更快地反馈问题，减少手工测试，持续集成就是其中随口可举的。测试团队成员越多，公司使用自动化越多，就越好。为此，我们必须脱离实施测试自动化的技术方面，而去考虑编写和运行自动化测试的非技术员工层面。无论开发团队是否在做敏捷验收测试驱动开发（ATDD），敏捷行为驱动开发（BDD）或使用传统的瀑布方法，团队可用来进行自动化的成员越多，自动化测试覆盖范围就越广。

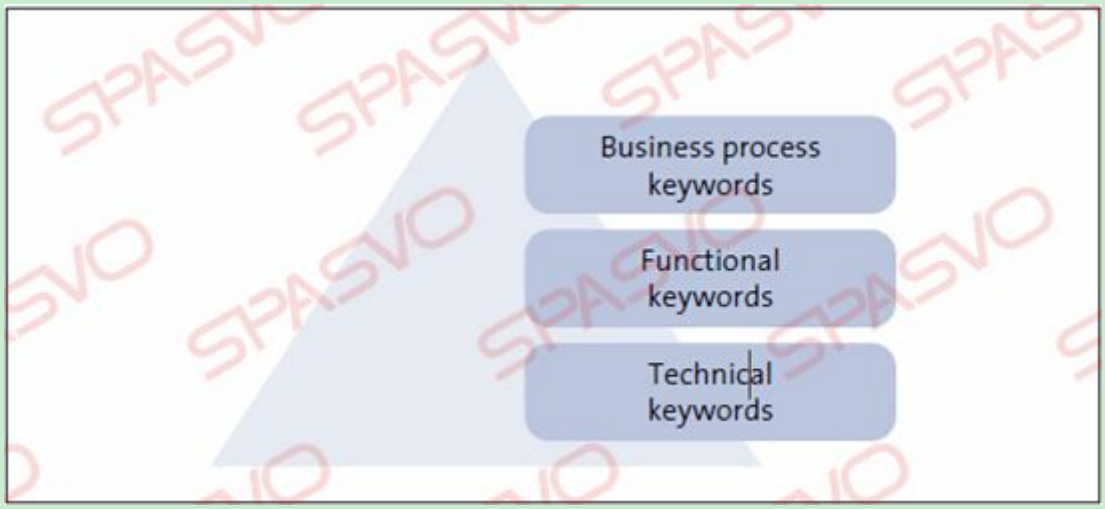
关键字驱动测试是一种由自动化工程师开发被测应用程序内可重复使用行为的方法。然后非技术用户就可以用任何输入参数将所得的可重复使用的关键字行为库进行排序，确定测试用例。比如，一个关键字可以是点击按钮（在一个按钮控件上点击）或输入文本（在一个文本框控件中输入文本），然后这些关键字就可以被用来填写一个登录表格并点击 OK 按钮。该方法的好处是：自动化工程师集中干他们擅长的，即开发关键字的测试自动化的脚本或编程；非技术测试员和企业用户使用这些关键字并基于他们的领域和产品知识来编写测试用例，最后使得自动化更加有效。

▪机器人框架是一个一般的基于关键字的测试自动化框架，它不依赖于任何一个特定的自动化工具（如：QTP， Ranorex， TestComplete， Selenium 等），却让自动化工程师插入用这类自动化工具的关键字。机器人已经安装了许多关键字并拥有一个非常成熟的功能集，包括：关键字的条件执行

- 测试用例和测试集的[setup]和[teardown]
- 数据驱动关键字文件，目录及流程管理的 FOR 循环
- HTML 测试报告

▪詹金斯 CI 集成将变量变为一个关键字并从测试用例关键字的关键字标注返回

▪最后列出的功能即能够在现有关键字上创建关键字，确保如图 1 所示的“关键字金字塔”的增长。



金字塔最底层是由自动化工程师开发的技术关键字构成：最顶端是应用程序内进行单个功能行为的功能关键字：最后，我们将基于功能关键字的业务流程关键字总结为在应用程序内构建业务流程。这种分层的一个例子如表 1 所示。

Technical Keywords	Functional Keywords	Business Process Keywords
Open Browser	Login	Purchase Product
Enter Text	Add Item to Cart	
Click Button	Checkout	
Close Browser	Enter Payment Details	

表 1. 关键字分层

这里，技术关键字是基于运行带有展示控件的自动化行为；功能关键字在应用程序中运行单个功能步骤；更高层的业务流程测试应用程序内端到端的流程。

即用机器人框架可以从 Python 和 Java 库中加载新的关键字；为了使用在.NET 中开发的关键字，就要利用远程服务界面。该界面是 XMLRPC 界面，机器人框架在上面给一个远程服务发送请求，执行一个关键字。当然这种远程服务可以用任何一种支持 XMLRPC 的语言来开发。尤其是，一个远程服务可以用.NET 开发让机器人框架执行关键字装配中的基于.NET 的关键字。这个方法如图 2 所示。

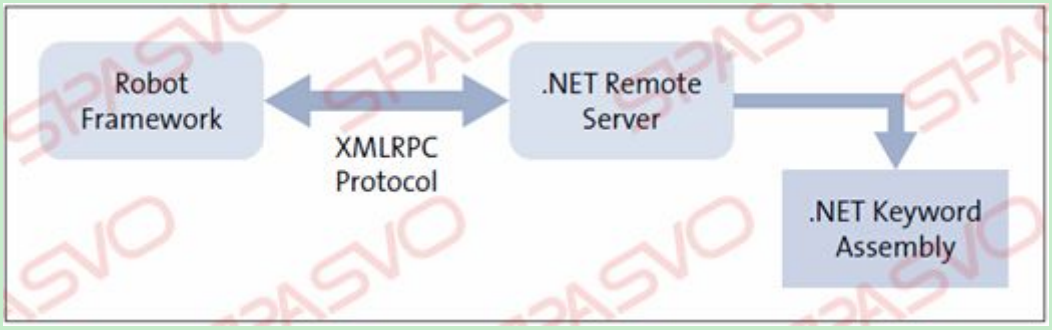


图 2. 远程服务结构图

N 机器人远程（见 http://code.google.com/p/nrobotremote/）是一个可以建立.NET 关键字库并通过 XMLRPC 协议将之公开给机器人框架的.NET 机器人框架远程服务器。结合了 N 机器人远程的机器人框架可以让最初的自动化工作用来计划哪些测试需要被自动化以及这些测试需要开发什么关键字，而不是在自动化测试计划可以开始前设计编写一个自动化框架。为 N 机器人远程开发.NET 关键字仅仅就是开发一个公开方法的类。每个方法都被视作一个关键字——例如：

```
1.  Public class KeywordClass {
2.      Public void ClickButton(String identifier) {
3.        //implemented in automation tool of choice
4.      }
5.      Public void EnterText(String identifier, String value) {
6.        //implemented in automation tool of choice
7.      }
8.  }
```

该关键字类公开了两个关键字：ClickButton 和 EnterText。通过在 N 机器人远程中创建关键字类并把机器人框架指向 XMLRPC 地址，这些都可以用选择的自动化测试工具实现且可以通过机器人框架来调用。如图 3 所示。

Settings	Value	Value	Value	Value
Library	Remote	http://localhost:8271	WITH NAME	NRobotRemote

Test Case	Keyword	Argument	Argument
Login As Admin	NRobotRemote.EnterText	Id=user_name	Admin
	NRobotRemote.EnterText	Id=user_password	Admin
	NRobotRemote.ClickButton	Id=OK	

在上面这个例子里，设置一列告诉机器人框架，http://localhost:8271 上有一个远程关键字服务（注意：除了本地主机，也可能是在另一机器上），且来自远程服务器的关键字将有前缀 N 机器人远程（任何前缀都可以）。测试用例一列则定义了一个叫做管理员登录的测试用例，调用关键字 EnterText 去输入用户名和密码，关键字 ClickButton 去点击 OK。这样一个测试有可能是在开发早期一个非技术测试员或企业用户所写的。关键字库本身可以被视作被测应用程序的域模型之上的薄薄一层。比如，如果使用 selenium 页面对象，那么关键字层就可以如图 4 所示，按顺序调用页面对象和方法。

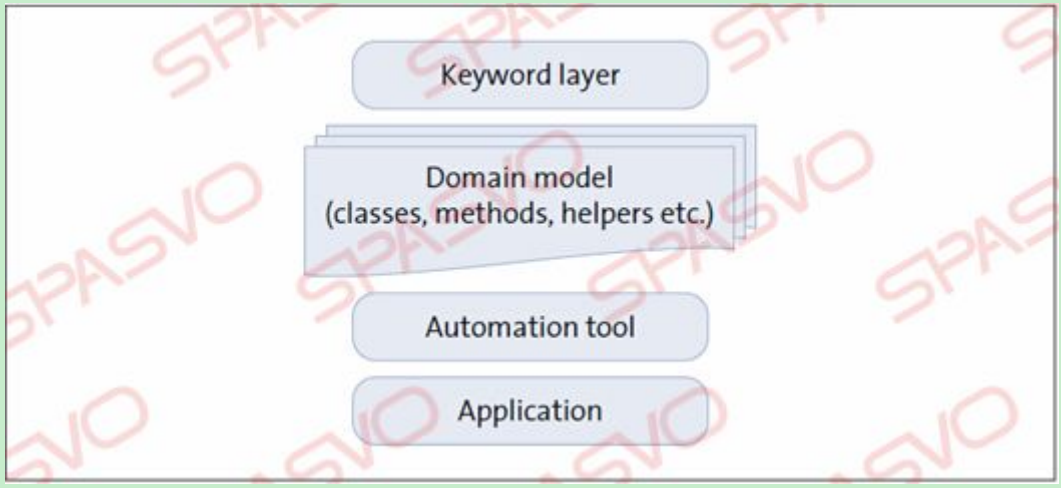


图 4. 关键字和域层

这种抽象概念在被测应用程序变化时提供了灵活性，经常，当应用程序变化时，只有域模型和关键字实施随之改变。测试用例实施并不需一直不变，因为它建在更高层。使用机器人框架和.NET 关键字也可以使不同的自动化工具在测试用例水平彼此整合，让自动化工程师可以灵活地为所需关键字行为选择最佳自动化工具。例如，如果在一家更大的公司，几支自动化团队已经被选去使用针对不同产品模块

的不同自动化工具，每队就可以用他们各自的开发语言为其模块开发一个域模型，如机器人框架可以加载 Python，Java 和（通过 N 机器人远程）.NET 关键字。测试用例编写者也同样可以在他们的测试用例里使用来自所有自动化团队的关键字。

总结

通过允许非技术用户在开发各个阶段编写测试用例，将自动化行为从抽象变为可以传给非技术测试员和业务用户以减少自动化工程师的瓶颈的可复用关键字。机器人框架，作为一个成熟的一般性的关键字框架，允许自动化项目一开始集中研究自动化测试和所需关键字，而不是设计并实施一个测试框架。使用机器人框架和 N 机器人远程将关键字测试自动化扩大到.NET，允许关键字利用.NET 框架和.NET 中自动化工具的优势，并允许把来自 Python 和 Java 的关键字与相同的测试用例相结合。

测试数据管理：创造性的解决方案

测试数据管理可能是测试专家职业生涯中要面临的最大挑战之一。还没有碰上测试数据紧缺或不完整测试数据的人算是相当幸运的。

我们并不孤单

缺少测试数据会影响开发员。几年前，我的一个任务里，开发员不得不猜测什么数据会进入数据库。在把他的代码给测试团队前，他绝对没有开始某种测试的环境。可以想象，测试阶段也有灾难。因此该项目开始后被中止近三年一点也不奇怪。或许这不是中止的主要原因，但绝对是一个成因。数据对任何系统都重要且绝对是测试一个系统的关键因素。数据为系统提供环境，，没有环境，开始测试阶段就值得商榷了。

我们真的需要它吗？

我们后退一步。我们为什么需要测试数据且我们该怎么计划去使用它？对于初学者，没有数据，你只测试应用程序的 GUI。以典型 GUI 为例，我们测试屏幕上的控件：按钮，下拉菜单，文本框等。即使这些测试会被限制，使得从前端 GUI 无法到达某些屏幕或功能， 因为没有输入正确数据。为了遵循系统中的某些流程，就需要具体数据。我们需要测试数据以确保企业规定被测且系统中不同流程被执行。想象一下没有数据的测试报告，我们开始测试了吗？

测试数据操作

为了让测试数据有效，我们需要在上面 CRUD（创建，读取，升级和删除）。测试专家面临的最大挑战之一是与第三方的集成。大多数情况下，测试数据只被读取，且数据数目被限。另一个潜在噩梦是第三方应用程序供应商不提前通知就改变测试数据。让第三方应用程序供应商保证你能获取他们的数据库闻所未闻。没什么阻止我们请求，但随时做好你的请求被拒绝的准备吧。测试数据及其管理对手工和自动化测试都很重要。两种情况中，测试专家旨在预测他们基于（他们在系统中输入的）数据的预期结果。多数情况中，测试专家无法创建或操作数据进入所要求状态。如果无法发现测试数据，就无法执行测试用例。

一个真实的例子

让我将我早期职业生涯中所经历的一次真实问题为你细细道来。我们不得不测试并将顾客管理系统自动化。一个单独的顾客账户上可以执行 100 多个不同的任务。比如锁定账户，解锁账户，查看余额，查看账户明细，激活邮箱，升级邮箱……

测试数据的问题是测试团队被赋予某些账号范围可以使用下游第三方相应测试数据。所以你可以创建你自己的测试数据并开始利用它，但你无法进行整个端到端的测试，因为新数据不会在下游系统上。

只有有限范围为了测试而被配置在下游系统上。所以你的测试会受限。

下个问题是测试员开始分享账号，或不请求或协调测试就使用测试数据。这就导致应该解锁的账户被锁，或拥有某些程序包的账户某天可以改变未来。测试同一个系统的不同功能时的不一致使一个有 16 名测试员的团队受到了挫折。

澄清一点，并不是所有手工测试都被影响了，但自动化确实是异常噩梦。自动化可以查询数据并找

到数据以供使用，但问题是，有时候数据就在那有时却不在，因为另一队成员不断在改变数据。自动化的一个优势是在测试执行前搜索数据。这种情况下，自动化运行就变得不可信了。我们绝对无法预测开始一次一整夜的自动化运行的测试数据是否充足。如果你无法在数据库中找到数据，最好的办法就是你自己创建数据。在这儿我不得不强调一下数据完整性的重要性：通过前端或通过执行，数据库上的某些失序的储存过程会破坏数据。

这会进一步阻碍测试工作并有可能造成由测试团队而不是开发团队引起的缺陷。让开发员判定缺陷原因很耗钱，最后却发现是测试团队自己破坏的。于是测试发布进程放缓了，自动化无法给投资满意的回报。

作为一个测试团队，我们逐步扩大问题，并请求设计师想出一个解决方案。几次会议后，制定出了一个计划。因为那时候想不出一个更好的词，我们称这个解决方案为“香草脚本”。那么它是干什么的呢？它是一个基本消除了系统外特定顾客数的所有数据的存储过程。我是说，所有数据，没错，就是所有的。主要是为了维护参照完整性且不破坏数据库。可想而知，这要尝试很多次才能成功，但三个月后我们想出了有效的解决方案。你们有些会觉得我们疯了——我们怎么可能会有这样一个脚本？如果将之投入生产呢？!这被视作发布流程和执行后测试的一部分。因为脚本是通过调用到一个存储流程执行的，存储流程要确保执行只在特定数据库名字和 IP 地址上完成。

这些问题按以下方法解决：

▪完整的终端到终端测试是可能的，因为香草脚本第一个运行，向下游系统发布命令删除支持他们的相关数据。重新创建该账号使得要重建一个下游，保证所有系统同步。

▪测试员没必要分享测试号。现在他们可以一遍又一遍地使用自动化去设置理想状态的用同一个账号的数据。

▪自动化也使用分配到的账号运行，所以我们总会有数据以供彻夜运行。

▪通过运行失序脚本破坏数据库的风险通过使用高级数据库开发员编写的香草脚本被消除了。

结果

结果绝对惊人。假设你要测试一个账户完整生命周期，从激活到删除，以及期间的所有任务。现在你可以做到！测试开始前，一名测试员运行香草脚本。现在，他们只需要让账户进入一个他们所需的特定状态以开始手动测试用例。这也使得我们能够用同一个账号为不同的软件包产品编写测试用例。自动化突然成功了。我们在 36 小时内运行 300，000 多个测试用例。反过来又产生了一个新需求：我们希望自动化运行地更快——但那在一般测试自动化和测试中却是一个问题。我们该如何解决第三方测试数的共享呢？自动化用一两个账号，手动测试员用剩下的。他们开始通过自动化使用香草脚本将账号设置为他们所希望的状态。关键字驱动的自动化是解决方案的关键，因为它可以让测试团队自己设计测试用例组合。

总结

测试数据管理可以创建或打破一个测试团队的精神。创造性的解决方案是需要的。不要停止寻找解决方案，最后总会有所收获。有时候解决方案就和在正确的时间向正确的人寻求帮助一样简单。

测试中“特殊数据”提出的挑战

一个精心设计的测试数据管理流程可以保证更高的测试覆盖率并减少终端产品中的缺陷。一个典型的测试数据管理流程包括测试数据需求阶段，期间测试和开发团队成员简单介绍并将所有要求的测试数据合并。还包括对重新测试的更新频率。然而执行测试项目时，我们却发现数据库不仅仅是一个值，一经发布它还有额外惊喜。以下三个数据尤其值得关注：

例 1：隐藏列

第一例便是一个医疗应用程序。病人数据是高度保密的；受数据保护行动的保护，病人对关于他们的隐藏数据保留某些权力。所有医疗保健单位都必须遵循这一点并确保别人绝对无法获取这些数据。我们正在测试的医疗应用程序有隐藏列。这些隐藏列在设计文件，要求规范或测试用例中无迹可寻。只有

在我们将之投入生产时，我们才意识到这些列的存在，且没有做过使用这些列的测试。这种情况下，我们怎么确保发布的产品没有缺陷呢？

例 2：多媒体数据

另一例：我们在网页用户可以在上面上传视频的网页上进行变更请求。因为用户有可能上传有煽动性的视频，敏感的东西都必须丢弃并从网页上删除。这只能靠人工干预和人为判断而不能靠自动化来完成。不过，我们还是需要用一些测试用例检测这些视频。我们该怎么创建测试数据？难道去上传敏感视频？这么做的话，敏感视频就会留在测试环境中并可能违反信息安全法规。

例 3：多个数据库

第三例出现在从两个不同源头中提取数据时。比如，遵循 HIPAA 的法规在个人数据上执行规则。例如个人姓名必须与他们社保卡上的一致。在一些自动服从的数据库中定义这些规则也是有可能的。一个典型的数据库结构是不允许有这些定义的。所以从两个这样的数据库中提取测试数据时，我们最后就会使用不兼容这些规定的测试数据，然后导致不完整的测试。

这些只是测试员会面临的多种情况中的三个。因此，除了常规测试数据管理流程，测试数据创建阶段还有必要包括以下内容：

▪一名能够预见数据库结构中这些数据的存在或不存在的领域专家。如果一些重要机密的信息在任一表格内丢失了，领域专家就可以查找去哪可以找到这些数据。测试团队会意识到这类信息的存在，至少在他们的测试用例中发现这类场景并未被覆盖；

▪一个让测试网页可以上传视频和其他多媒体文件（包括：定期审核一类的活动）的测试计划。有了一些语音识别工具的帮助，这类测试的自动化也是有可能实现的。这类数据的维护也应该被创建，存储和再次利用。一个有效的测试数据管理流程需要一个重要功能的准确环境和可操作数据。认真考虑测试数据管理策略和控制对高效彻底的测试很有必要。

智能手机发展中国家测试

在西欧和美国，移动设备业重点放在智能手机及其相关生态系统上。但是，对于世界上的大多数移动设备消费者来说，未来功能机可能仍独领风骚，估计 70%现今世上使用的移动设备都是功能手机。尽管比起智能机，功能机很简单，但它们的功能正变得越来越丰富，这对于测试它们的人来说成了个大大的难题。其中一些挑战在其他种类的测试中也有，而一些却是移动设备领域和功能机本身特有的。在我们研究这些特有挑战前，首先有必要解释一下为什么功能机和功能机消费者与购买使用智能机的人有的方面不同，有的方面又一样。

并没有官方定义来区分功能手机和智能机，这句话一点儿不错，两者最常用的区分根据是价格。但考虑到设备制造商在极其激烈的移动市场里的折扣尺度，这并不是最佳区分方法。为了本文的目的，我们可以假设功能手机是一个——运行有专门制造商且不允许真的多任务或第三方本机应用程序的操作系统的，并且屏幕很小的——移动设备。 根据用户所需成本以及制造和零件成本，功能机要比智能机便宜。它可能有摄像头也可能没有，它的处理器肯定比智能机慢，内存比智能机要小。功能机价位也可以拥有触摸屏，更多传统设备款式，如 ITU-T 键盘，现在也还有——事实上，功能机的款式和功能范围比直板触摸屏设备占主导的智能机的范围更广。

去了解消费者是如何使用功能机的也很重要，尤其是从测试的角度去考虑功能机时。虽然贵的（零售价在 60 到 120 欧元间）功能机，在西欧和美国可能打折，但打折通常只在预售期。功能机最大程度可低降到 20 欧元甚至更低。这对富裕的西方人来说没差，但对于发展中国家的消费者，这是一项重要的财务采购，经常需要好几个月的积蓄。这对设备的感知质量有很大影响，因此，要在发布前进行测试以便大家对质量有信心。功能机消费者对其设备更感自豪，因为他们或许已存了一段时间能够支付它们了，也别指望他们的骄傲会因为质量差而削弱。他们更可能把设备拿去修理并产生个更高的期待：相信那些设备不会坏。一个合理的比喻就是发达国家所拥有的电视或汽车。发展中国家的功能机消费者也有很高的期待。如果买得起的话，他们大多会买智能机，他们希望未来的功能机可以提供智能机那样的体

验。于是，我们就看见移动设备制造商不断尝试增加他们功能机的功能（尤其是诺基亚 S40 平台和三星的 SGH 平台），并一直在内存及处理性能低的平台上提供这些功能。这就意味着要最大程度地利用平台并让有效测试更显重要。

功能机消费者的典型使用情况与智能机的也不一样。在发展中国家尤其是非洲，一个村庄或许要共用一个手机；事实上，共用一个移动设备以确保与村外的交流的业务已建立了（如村庄手机项目：www.grameenfoundation.org/what-we-do/empowering-poor）。一些功能机提供如多个通讯录和通话记录的功能以支持多个用户。通过使用多个 SIM 卡省钱的做法在发展中国家很流行，在发展中国家，或许一个网络会提供廉价的本地语音通话，另一个网络提供廉价的外地语音通话。消费者通常会经常替换 SIM 卡，且设备可以支持一张以上的 SIM 卡。诺基亚提供可以像内存卡一样轻松替换 SIM 卡的设备，两张卡同时使用。其他制造商提供可以在同一个设备中支持四张不同 SIM 卡的设备。尽管功能机通常不为第三方开发提供给设备配置本机应用程序的性能，但通常一些第三方运行环境本身就有了。最常见的就是 JAVA 移动版本（JME， 正式名称为 J2ME）环境，通过第三方可以写出名为 MIDlets 并在设备上运行的小程序。JME 是世上最广泛使用的移动设备环境，它包含了标准 API。通过这些 API，第三方开发可以评估越来越多的本机功能，如通讯录，GPS，触摸屏和短消息。估计全球功能机占有 70%的市场份额。

一些设备也支持美国高通公司的无线二进制运行环境（BREW）平台。但它却不是一个像 JME 一样真正的虚拟环境，它不提供 API，且应用程序可以用 C， C++ 或 Java 编码。现在第三方应用程序的测试员面临的最大的挑战是平台碎片化。鉴于款式，屏幕尺寸和用于功能机的平台硬件各种各样，且设备制造商提供的设备一般少于标准 API 规定的全套设备且其中不少还是他们自己制造的，对大量设备进行测试有助于确保测试覆盖充分。可以用多种方法将应用程序放到设备里。最流行的是 app 商店，例如：GetJar 和 Nokia 商店。至于智能机，也可以用相同的方法下载并安装应用程序，且这些商店的用处很大——诺基亚商店里的设备有 100，000 多个应用程序在运行 S40，GetJar 一天有超过 3 百万的下载量，堪称世界第二大的 app 商店。确保 app 商店提交，下载，安装和卸载都被进行过测试很重要。所以，作为一名测试员，测试功能机和功能机应用程序时你尤其应该要注意哪些方面呢？以下内容至关重要：

智能手机应用程序需要测试哪些方面？

从某种角度来说，功能机与智能机并不是那么不同，两者间的差异，尤其是价格，正得越来越模糊。因此，一个成功的测试策略要关注应用程序功能，网络交互，压力和加载，合适的位置等方面。一个好的起点就是要让测试部看看现在的思维导图。尖端功能机现在支持地图，push email 和如愤怒的小鸟等游戏。但对于未来的功能机的发展空间，一些方面变得更加重要了：

1. 网络交互

功能机被用于——尤其是在发展中国家——移动网络不可靠，信号也不强的地方。因为用户和网络比乡下大，城市的基础设施通常更加超常负载，这意味着手机通讯发射塔到有移动手机的距离更大了。这说明任何移动应用程序都要能适应网络，或数据载体，损失或降级。考虑突然失去网络交互的情况以及被测应用程序上的影响，很关键。如在使用应用程序时进入及出了网络覆盖范围外的使用情况也是高度相关的。

2. 电池寿命

无论在什么平台上测试移动设备时，都要充分了解应用程序的电池寿命的影响。然而当人们考虑一个发展中国家的电池寿命的使用情况时，功能机的电池寿命会更重要。在电力支持不稳定和根本没有支持的国家，需要在“充电商店”里充电，那么良好的电池寿命就变得相当重要了。编写的不错的应用程序不应该耗费过多的电，且对电池影响进行测试（要么通过脱机监测仪，要么通过执行典型案例和凭借设备的电量指示灯来监测电池寿命）很重要。也需要慎重考虑电池用完时设备或应用程序的性能；电量低时会崩溃或退出吗？

3. 设备使用时间

功能机的电池寿命通常比智能机长，因此要很久才会电量用完或设备重启。使用得当也可以使得设备的使用时间更长。音乐播放器就是个典型的例子——在发展中国家，移动设备通常用作最基本的音乐播放器和收音机，因此音乐播放器可以运行 10 个小时甚至更久。确保应用程序本身继续运行且没有如

内存泄露一类的 bug，就需要用不同的方法去对其进行测试。对使用情况长时间测试在功能机中比在智能机中更重要，且可以发现短时间的功能测试无法发现的 bug。

4. SIM 卡

如果被测应用程序使用存储在 SIM 卡或内存卡上的数据，就要确保有同等数量的卡来测试。不是所有的 SIM 卡都做的一样，有些会比其他的慢上很多；这似乎是发展中国家（如印度）的一个特定问题。一些牌子的内存卡也相当慢，这会对设备或应用程序的功能产生影响。获取并测试与应用程序或设备的预期发布国家/地区相关的卡可以帮助减少不兼容导致的早期场地周转或卡运行慢的风险。

大量功能机都支持多张 SIM 卡，而且或许还支持在不关闭设备的情况下交替使用这些卡。下面包含了一些要考虑的相关使用情况：

▪不关闭应用程序就拿掉电池会怎样？这是替换 SIM 卡时常见的使用情况。

▪替换 SIM 卡时使用 SIM 卡里数据的应用程序会怎样？数据还在吗？还是被删掉了？

5. 数据共享

尽管数据的使用在西方相当普遍且其价格越来越便宜，但发展中国家的市场上却不一样，那里的消费者极具价格意识。因此应用程序（如基于代理浏览器的云）的使用，通过在交付到移动设备前压缩网页以减少数据使用，很流行。用户也极可能通过免费渠道（如蓝牙和红外线）分享内容。测试时将所有共享方法以及被测应用程序是否提供最低价方法都要考虑到位。

6. 可用性

相比智能机，功能机通常屏幕更小且有不同的输入法。功能机的屏幕可以小到 128 × 128 像素，测试时要确保应用程序充分利用有限屏幕。像文字大小，每页显示的信息细节，为了看到最重要的信息所要求的滚动次数等问题在屏幕如此小的情况下更显重要。正因应用程序要支持各种大小的屏幕，测试 UI 是否伸缩自如很重要。尽管大量功能机有机械键盘，通常是 ITU-T（只有数字）或标准的传统键盘配置，这使得在屏幕上选择更容易，但是越来越多的功能机开始用触摸屏了。 这些通常是更便宜的阻性板，需要按选地更精准，并确保最后应用程序或设备的可用性。考虑用户将怎么在屏幕上滚动，怎么用手指或笔尖选择区域。对多语言支持的考虑和测试也很重要。在说多种语言的国家，应用程序就需要支持多种语言，在多个国家发布的应用程序也是一样。有一点尤其要注意，就是如果合适的话就要确保应用程序支持从左到右和从右到左的语言且没有布局或可读问题。

7. 有限的设备内存和处理性能

功能机，因为价格便宜，所以比起智能机，它们设备内存更小，处理器规格更低。它们不太可能会有专门的图形处理器。因此，任何设备或应用程序的测试策略都要考虑内存满了和设备正被多设备使用按压的情况。用户使用功能机时通常其内存几乎都快满了，因为只有 10Mb，任何应用程序的行为都应该在这些情况下进行测试。尽管功能机操作系统几乎不支持多任务和背景的应用程序，但测试来自其他应用程序（如：来电和短信，音乐播放器）的干扰以及蓝牙数据同步， 蜂窝数据或 IR 也很重要。其他功能，如备份和恢复，也可以在一些功能机操作系统上运行。

8. 设备

如讨论第三方运行期间时所提到的，功能机里有不少的设备和操作系统。支持 JME 或 BREW 应用程序接口的水平以及可用硬件和软件配置的图表编号，会变得很混乱。建议花时间了解一下应用程序打算发布的地方的市场。看看哪些新设备更受欢迎，哪些设备市场上已经有了。这可以帮助一门心思地针对少数需要购来测试的设备。功能机的处理性能和设备内存低，因此设备上的测试比智能机上的更重要。模拟器不会，尽管流行的基于云的服务（如：Perfecto Mobile）支持一些功能机，但并不全面；且没什么比得上手上有设备来支持测试。

一个成功的功能机 APP 测试策略

为了成功地发布一个功能机的 app，就要考虑以上所有方面。功能机的市场还是很大的，因此任何 app 可能出现在数百万人手中，他们 app 如预期的那般好。测试要有效果，可以减少推出质量差的 app 的风险。至少将以下几点考虑在内才是明智之举：

▪在设备上测试，不要依赖模拟器。功能机碎片很大，设备本身的处理器速度慢，内存低，因此无法模拟。

▪电池使用寿命是功能机的关键。它的质量一定要好，用户或许无法轻易充电。被测 app 被使用时

确保电池使用寿命。

▪功能机用户往往受经济制约。他们会用最廉价的方法实施他们的特定使用情况。就是说要替换 SIM，通过蓝牙发送，复制到，再从内存卡中移除。被测 app 使用最廉价的方法吗？有用吗？

▪考虑可用性。功能机的屏幕小，app 一般支持更多语言，输入法与智能机也不一样。

▪功能机用户会使用信号差的不太可靠的网络来连接。被测 app 该如何应对？

▪功能机用户确确实实在用 app 商店。他们一天下载无数的 app 且那些商店基本都比你们想象的更大。确保 app 商店的提交，下载，安装及卸载都经过测试很重要。

▪功能机比智能机更快更频繁地耗尽处理器性能和内存。给被测 app 不断施压。如果你不这么做那么等着吧，用户会这么做的。

▪人们用不同于智能机的方法使用功能机。他们更常用像音乐播放器或收音机一类的 app。他们分享手机。他们替换 SIM 卡。花时间了解该类情况并对它们进行测试。花时间了解顾客并想想他们会怎么使用 app，或许不是你常用的方法。

将这些都考虑在内，功能机的 app 和设备本身要进行测试的方方面面很多，功能机市场份额仍很高。相比智能机用户，功能机用户更关注质量，都不愿忍受手机质量差。他们期待得更多，当一个 app 可以通过功能机在上百万甚至数十亿人手中时，，它应该在推出市场前接受最全面的测试。

有关培训、产品购买及试用授权方法等事宜

电话：021-61079698

Email： sales@spasvo.com

QQ：1404189128

SPASVO