测试系统构建基础知识

# 测试执行软件

目录

## 引言

大多数测试系统基本上围绕两个概念进行设计：效率和成本。无论是在消费电子行业还是在半导体生产领域，测试工程师都关心测试系统的独立测试时间和总吞吐量，以及这些参数如何影响资源。当应用不断扩展到包含多个测试、多种仪器和多个待测设备(UUT)时，便不可避免地需要监测测试执行软件，以解决成本和效率问题。

测试执行软件通常作为定制化的解决方案进行部署，或作为商用现成(COTS)产品直接购买。在典型的构建V.S.购买论证中，测试架构师必须确定自行编写测试执行程序更合理，还是投资和整合现有解决方案更具成本效益。在决定定制还是购买测试执行软件之前，有必要了解此类软件的目的和核心功能。本指南总结了测试执行软件的主要功能，并探讨其实际应用场景。

## 背景

测试执行软件可以自动化和简化大型测试系统。 测试执行软件处于软件堆栈的最上层，整合了测试的基本功能，如测试执行、结果采集和报告生成。 该解决方案的特点并不是针对特定的UUT，因此各种应用可以使用该测试执行软件作为框架。 这意味着使用LabVIEW图形化语言、C、.NET或其他语言编写测试代码的开发人员可以专注于测试特定设备，而所有UUT的常用功能都由最上层测试执行软件维护。 总而言之，从开发、成本和维护的角度来看，测试执行软件以有效的方式定义了此类常用功能。

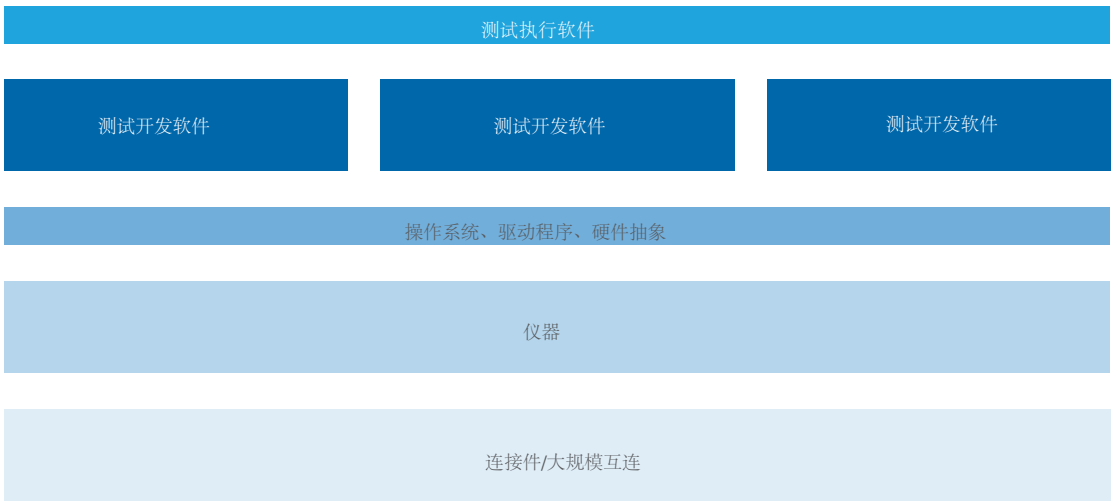


图1.测试管理人员通过完成在更高层次抽象的所有测试中通用的任务，允许单个测试开发与整个测试系统的架构需求分离。

## 测试执行软件的特性

取决于公司的规模、特定测试仪的规模以及待测设备的种类，测试执行软件可以从简单到复杂，不一而足。本指南概述测试执行软件可能包含的常见特性。一些特性对于测试执行软件的所有实现至关重要，而其他特性则属于附加功能，并不一定必要。每个特性列出了完成开发的估算时间。这些估算是基于数百个自动化测试客户的经验，详情请参与《测试执行软件—构建还是购买？ 基于NI TestStand的财务分析比较》。

### 测试序列开发环境

测试执行软件提供了构建测试序列的开发环境。这是最基本的特性，也是一个复杂的特性，为整个执行了提供开发接口。序列架构包括实现分支或循环逻辑的能力、导入测试限制值的方法以及独立测试代码的规范和整理。与测试代码的交互需要具有使用各种编译格式的灵活性，例如DLL、VI和脚本，以及与不同开发环境集成的能力。测试管理软件还可以使用源自源代码控制提供程序的测试代码。

在定制的测试执行软件中实现测试序列开发环境可能需要大约100人天才能完成，而商业解决方案提供了现成的环境。由于开发环境提供的功能范围，此特性如果作为内部解决方案，需要最长的开发时间。而且，该特性是序列架构体验的基础，不能被忽略。

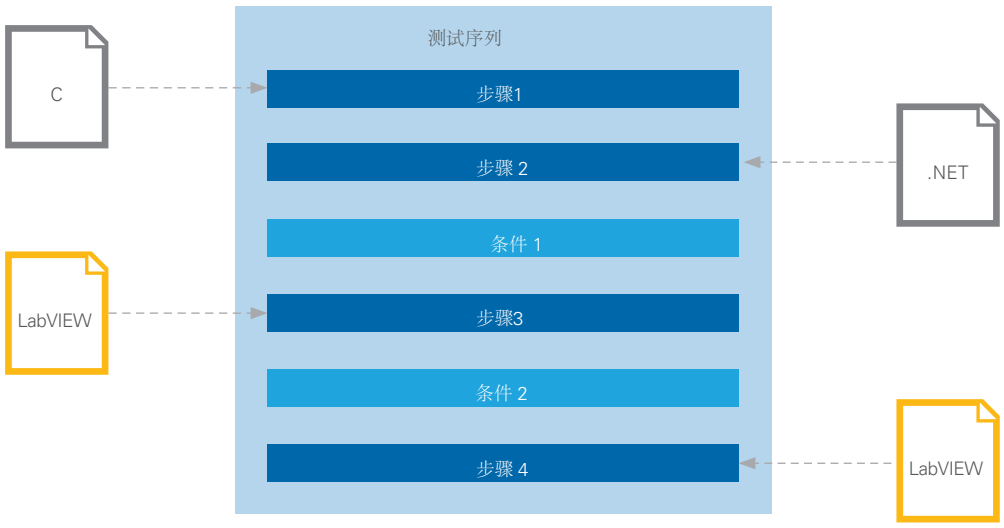


图2. 高效的序列开发环境可帮助测试工程师开发和调试复杂序列，并将序列调用道现有测试代码中。

## 自定义操作界面

操作员界面是操作员通过其与测试系统交互的显示器。它通常允许选择关键输入参数，例如UUT标识号、要执行的测试序列或报告路径。它还包含一个“运行”或“开始”按钮来控制执行。许多大型测试系统如今需要因应用或公司而异的专业化图形化用户界面，并且可让开发者灵活选择编程语言。除了定制之外，这种高度功能性接口包括加载、显示和运行测试序列的能力，以及交互式用户提示、执行进度指示器、测试数据的可视化和本地化。

设计自定义操作界面需要花费8到32个人日的开发时间。COTS解决方案提供了现成的库和UI控件，可以减少该时间。开发自定义操作界面可能是一个重要的时间投资，且不论测试执行解决方案是自行开发还是购买现成的。认为操作界面对其系统不那么重要的测试工程师会指示操作员在开发环境中工作。

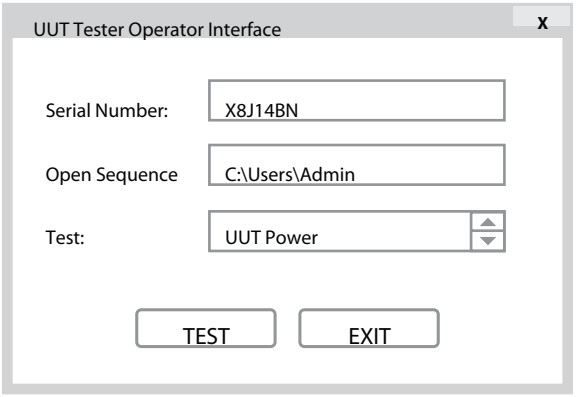


图3.用户操作员界面能够识别给定测试序列的的唯一UUT、公司、应用程序、测试和操作员角色。

## 序列执行引擎

测试执行软件的一个核心组成是序列生成引擎。序列执行引擎负责评估UUT所需的所有操作。这包括调用独立的测试代码、创建测试之间执行的流程，以及在测试之间管理数据。序列生成引擎是负责执行给定测试序列的程序，无论是在开发环境中、通过自定义操作界面，还是在部署的测试程序上。

内部实现序列执行引擎需要至少15个人天。但是，它是所有测试执行软件的必备功能。

## 结果报告

鉴于测试执行软件的抽象角色，这部分软件功能负责整合独立的测试数据、临时存储到内存中以及发布综合测试结果。报告可以有各种格式，包括XML、text、HTML和ATML。数据也可以在执行之后存储到数据库。测试执行软件通过可扩展的报告选项实现多样化的报告格式。结果报告是许多测试系统的必要组成部分。

从零开始开发结果采集和报告生成器可能需要大约15人天，取决于所需的具体报告。由于COTS解决方案中内置了报告生成器，因此可以定制结果报告，只需一人天甚至更少的时间就可以提供应用所需的报告。

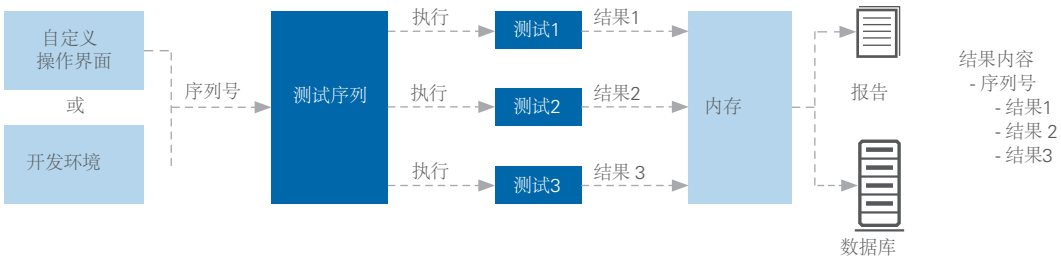


图4. 测试执行软件在测试系统的一部分作用是整合执行过程中生成的结果，并生成报告发送给数据库。

## 用户管理

根据使用人的角色和级别将测试执行级别区分开来是有必要的，用户管理工具可有效地划分总测试架构师、编写和调试测试代码的每个测试开发人员以及运行测试的操作员或生产经理之间的责任。 赋予特定用户的功能甚至会有密码保护，以防止误用测试序列。

在定制测试执行软件中实现用户管理系统需要大约5个人日的开发时间。 虽然对于使用测试执行软件不是必需的，但是用户管理工具不需要大量的人力投入即可实现，而且可以简化测试执行软件职责的执行。

| 权限 | 架构师 | 开发人员 | 操作人员 |
|---|---|---|---|
| 编辑 | √ | — | — |
| 保存 | √ | — | — |
| 部署 | √ | √ | — |
| 循环 | √ | √ | — |
| 运行 | √ | √ | √ |
| 退出 | √ | √ | √ |

| 用户 | 级别I |
|---|---|
| Mark | 操作员 |
| Larry | 操作员 |
| Julie | 开发人员 |
| Scott | 开发人员 |
| Lauren | 架构师 |

表1. 类似于Windows文件权限，用户管理器可将测试执行软件的角色和责任区分开来。

## 并行测试功能

并行测试涉及同时测试多个待测件，并能够维持正确的代码模块性能、结果采集和UUT跟踪。并行测试方法的范围从流水线执行（按测试顺序执行，但测试执行软件可以同时测试多个socket）一直涵盖到动态优化、批处理或其他复杂的执行操作。

对于测试执行开发人员来说，实施并行测试通常是非常耗时，从头开始开发需要花费100人日。尽管并行测试可能需要大量时间来开发，但是对于大型测试系统，通过扩展执行来降低对吞吐量的要求通常至关重要。许多公司在首次部署测试执行软件时并没有考虑并行测试，后来才意识到他们最终需要这个功能，不过已经为时已晚。

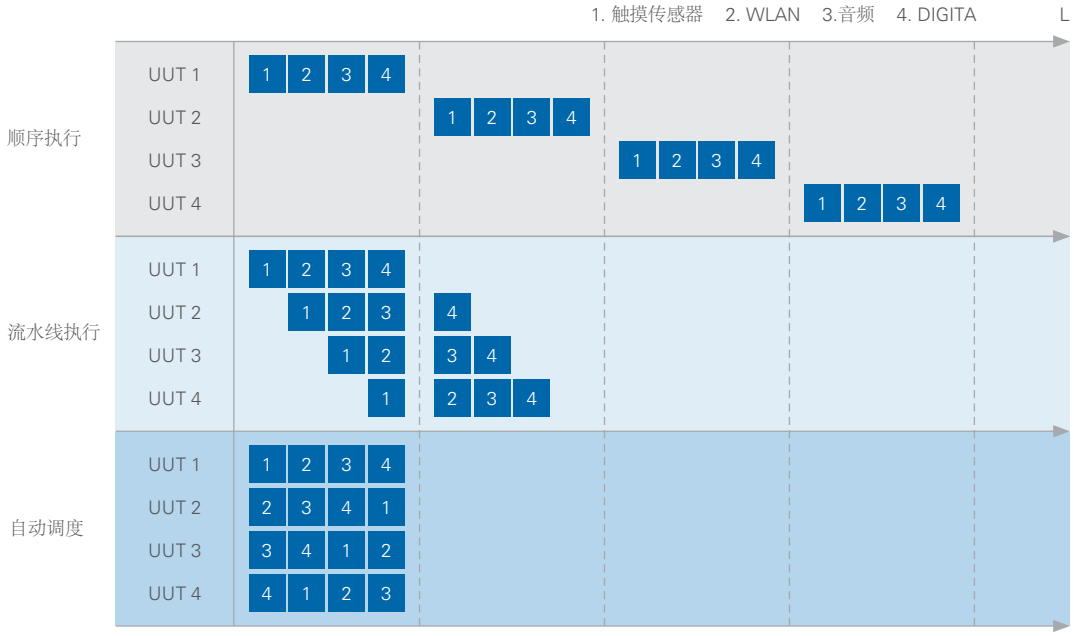


图5. 并行测试功能可显著增加系统吞吐量，而且无需重新构建测试执行软件的架构。

## 单元/设备跟踪和序列号扫描

同时测试多个UUT时，可能需要对每个被测设备进行唯一的标识和跟踪。 该信息可以与单元或批量级的特定分析的测试结果存储在一起，或者在出现问题时精确定位错误的来源。 设备跟踪可以由操作员通过键盘手动输入，也可以在读取条形码之后由全自动扫描仪加载UUT信息。

从头开始开发此类功能可能需要五个人日，如果采用COTS解决方案，大约需要一个人日进行自定义。 并不一定每个测试系统都需要UUT跟踪。但该功能适用于需要高容量、高吞吐量测试的应用，例如半导体或消费电子行业。

## 测试部署工具

大多数大型测试系统不是孤立构建的；通常是多个测试站点或整个生产车间的整体解决方案。测试执行软件提供了一种机制或实用程序来将整个软件堆栈打包到一个内置可发布单元中，从而在系统部署中发挥关键作用。测试系统可以以各种方式分布 - 架构师可能希望部署测试系统的映像或包含所有必需的依赖性和运行时间的完全功能的安装程序。有关此主题的更多信息，请参阅《构建测试系统基础知识》系列之《软件部署》白皮书。

部署是一项非常重要的任务，从头开始可能需要一个开发团队花费20人日的时间来完成。借助商用测试执行软件的现成部署实用程序，可能仍然需要三个人日的时间来成功部署。考虑到该特性对于多个测试站点的适用性，通常需要具有测试执行解决方案。

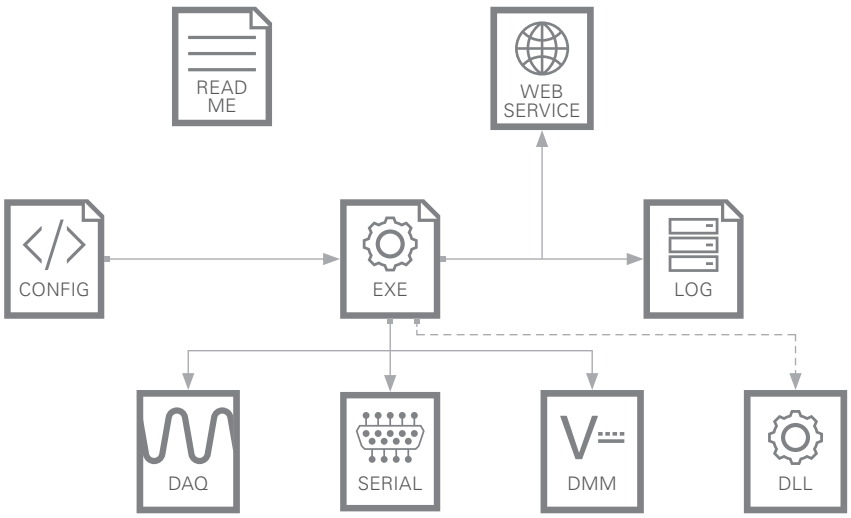


图6. 部署涉及使用部署工具或编译服务器将测试系统的所有必要组件封装起来，然后发布到所需的测试站。

## 维护

与大型测试系统中的任何其他组件一样，测试执行软件必须得到适当的维护，以确保其性能能够经受时间的考验。 这包括扩展以纳入新测试、升级以维持软件或操作系统升级的兼容性，并修复检测到的任何错误。 维护测试执行解决方案甚至延伸到文档领域。 文档是操作人员、开发人员和架构师在使用测试执行软件时所依赖的重要资源。

虽然很难预测特定测试人员的需求，但是每年开发自定义测试执行程序时需要在初期将15%的时间用于维护。 总开发时间包括生成足够文件所需的时间（估计20天）。测试执行软件支持的时间间隔可能会有所变化，这会大大改变成本的估算值。但是不建议在实施任何测试执行解决方案时减少维护工作。

## 实际场景 1

Jonathan是一家小公司设计实验室的测试工程师，该公司致力于提供低成本的消费电子产品。每个设备通过一个远程控制器进行控制。 Jonathan负责编写测试代码，用于在设备进入生产之前验证远程控制器和原型之间的发射器 - 接收器通信。 随着公司生产规模不断扩大，Jonathan可用于每台设备执行必要测试的时间减少了。 他需要自动执行现有的验证代码，以便可以将更多的时间花在为新设备编写代码上。 因此，他决定使用测试执行软件来执行测试代码序列。

下表列出了Jonathan对测试执行软件的要求。

| 功能 | 实现 |
|---|---|
| 测试序列开发环境 | Jonathan需要一个开发环境来构建序列。他编写的测试代码已经相当模块化，所以他只需要在这个环境中调用和循环测试代码。 |
| 自定义操作界面 | Jonathan想要能够以最小的交互执行一组测试代码。 他希望仅指定一些相关参数来识别设备、需要的测试和报告路径。 但是，由于他是终端操作者，因此将接口与开发环境分离并不重要。 |
| 序列执行引擎 | 这是Jonathan测试执行解决方案的核心需求。 每个测试由几个独立的LabVIEW VI组成，这些VI必须按顺序执行。 |
| 结果报告 | 现有的测试代码目前会提示用户为给定的原型生成新的技术数据管理流(TDMS)文件。 随后的每个VI将最少的测试结果存储到这个文件中。 测试执行软件只需要自动创建此TDMS文件，然后执行测试代码的其余部分，并以根据约定生成结果。 |
| 用户管理 | 用户管理不是一个主要事项，因为Jonathan将负责设计、开发和操作这个测试执行软件。 |
| 并行测试功能 | Jonathan每次仅对一个原型进行测试，因此不需要担心UUT的数量。 |
| 单元/设备跟踪和序列号扫描 | 由于测试是在设计原型上进行的，因此没有分配的序列号需要跟踪。 相反，Jonathan会通过操作员在运行时输入的唯一名称来跟踪每个UUT。 |
| 测试部署工具 | Jonathan不打算将这个代码部署到其他测试仪。 他的测试台是设计实验室独有的，与制造设施独立开来。 |
| 维护 | 这个项目完全属于Jonathan。 他将实施和维护任何被选中的测试执行软件。 他不打算文档记录他的工作，因为他将是唯一一个操作和使用这个测试执行软件的人。 |

表2. Jonathan对测试执行软件解决方案的需求主要是现有验证代码的简单自动化。

Jonathan决定自行开发一个测试执行软件。 他没有复杂的序列执行或报告需求，也没有计划将此系统部署到其他用户或测试站。 如果购买商业解决方案，是不会看到投资回报的，因为大多数功能都派不上用场。 相反，他只需依靠所掌握的软件知识和以前购买的应用软件，就可以开发一个序列执行器来满足其需求，而且只需短短10天即可完成。

Jonathan使用LabVIEW软件开发测试执行程序。 他通过一个简单的界面构建一个解决方案，使操作员能够调用一组已确定的测试步骤并选择TDMS日志的路径。 当为新原型引入了额外的测试时，Jonathan偶尔会对序列执行器进行小的改动。 总的来说，由于部署了这个序列执行器，设计实验室的工作效率提高了。

在本例中，开发测试执行软件Jonathan满足其需求的最佳选择。 通常，内部开发解决方案是需要序列执行或完全自动化时采取的第一步，并且与生产环境的需求相比，可能更适合于测试台应用。

**如果...**

- 几个月后，设计实验室聘用了一位新的测试工程到，并开始协助测试过程。 这个工程师将如何学习序列执行工具的操作方法，或有效地解决显示的任何错误或漏洞？
- Jonathan调到另一个部门或离开公司。 那么更新或维护序列执行软件所需的知识又该如何维护？
- 当序列执行器有必要执行整个原型的功能评估。 它如何将不同工程师用其他语言编写的额外测试代码、不同的编程方法和报告技术整合在一起？
- 测试执行软件需要移植到生产环境中，以确保测试的一致性。 这些解决方案能否满足这些需求？

## 实际场景 2

Dave的公司正在设计一个新的功能测试仪，部署到生产线的最后。目前，UUT测试是通过手动执行一系列现有的零散的代码片段来实现。这个过程显着限制了生产线的产量，而且Dave希望在该过程的自动化中使用测试执行软件。该公司没有对测试执行软件进行标准化，每个小组通常都是从有限的商业解决方案和无数的定制解决方案中选择自己所需的。

下表列出了Dave对测试设备提出的要求。

| 功能 | 实现 |
|---|---|
| 测试序列开发环境 | 必须具有能够支持测试执行软件主要功能的高效开发环境。 环境必须能够支持LabVIEW、.NET和Python代码的排序。 |
| 自定义操作界面 | Dave最终需要一个为公司定制的操作界面。 除了"运行“按钮，他还想删除大多数功能。 |
| 序列执行引擎 | 这对于系统解决产量问题是 不可避免的。 |
| 结果报告 | 目前，每次测试单独将数据记录到SQL数据库。 因此，需要测试执行软件能够整合所有结果，将总结果传送到数据库并用序列号进行标识。 |
| 用户管理 | 大多数与测试仪的交互由生产环节的操作者进行。 Dave希望通过用户管理工具或可自定义的界面，删除操作员的开发权限。 |
| 并行测试功能 | 只要测试仪的吞吐量与生产吞吐量相匹配，Dave就不需要同时测试多个UUT。 |
| 单元/设备跟踪和序列号扫描 | 序列号用于标识每个组件和组装的UUT。 条形码扫描器用于跟踪此类信息。 测试执行程序必须能够在执行的多个测试之间传输此类信息。 |
| 测试部署工具 | Dave需要将最终产品部署到另外10个测试仪中。 |
| 维护 | 测试工程部门将维护测试执行软件，无论是内部解决方案的全部功能还是COTS选项的相关功能。 |

表3. Dave对测试执行软件的评估主要基于对功能测试仪的吞吐量需求。

为了做出决定，Dave还从财务角度对测试装置进行权衡。他估计，新的测试仪将包括一个大型高性能PXI机箱和嵌入式控制器的组合。基于评估UUT所需的测试，机箱将包含从数据采集卡和模块化仪器（如数字化仪和任意波形发生器）到射频测试设备等多个模块。不论测试执行解决方案如何，每个测试台的成本将在10万美元左右。

在评估软件堆栈时，Dave指出，购买COTS解决方案会增加项目成本。测试执行软件的开发许可证需要几千美元，每增加一个测试仪的部署许可证将增加了500美元的成本。

Dave相信他可以通过在Python中构建自定义解决方案来节省测试执行软件的成本。该语言是开源的，而且开发环境是免费的 - 他们相信这两种优势将抵消自行开发测试执行程序所需的额外时间。

测试工程团队精通Python，在要求的时间截点内提供核心功能 - 顺序序列引擎、数据库连接和复用现有测试代码。测试执行软件已成功部署到生产线。测试工程师偶尔会被调用来修复一个或多个测试仪的漏洞。

**如果……**

- 生产线上的生产需求增加，导致现有测试执行软件无法满足产量需求。 这时需要增加并行测试功能。
  - 尝试实现此功能需要多少额外的开发时间？ 这会如何影响自定义与COTS解决方案的成本比较？
  - 假设由于Python语言的多处理限制，测试仪的吞吐量无法满足需求。Dave的团队需要购买额外的硬件来复用当前的解决方案，或者从头重新开发另一个测试执行软件。 这将如何进一步影响自定义与COTS解决方案的成本比较？
- 测试工程团队不能总是为其他重要事项维护或升级测试执行软件。
  - 当这种需求出现并且团队无法提供帮助时，生产将如何受到影响？ 因此导致的停机如何影响系统的维护成本？
  - 测试工程团队花在维护测试仪的时间应如何量化？ 这个因素如何影响系统维护成本？

## 实际场景 3

Karen在一家设计和生产小型医疗设备的公司工作。 每个产品都有自己的全自动化生产线。虽然每个小组都采用测试执行软件进行最上层的系统管理，但该公司还没有对解决方案进行标准化。 最近，一个新的测试经理就职，对测试执行软件标准化很感兴趣。Karen的任务是选择商业解决方案、现有内部产品还是进行新的开发来实现测试执行软件的标准化。

Karen为负责每个产品的各个小组制定了以下要求列表。

| 功能 | 实现 |
| --- | --- |
| 测试序列开发环境 | 测试开发人员需要一个灵活的开发环境，特别是可以与其LabVIEW和VB.NET代码连接。Tortoise SVN用于源代码控制，因此开发环境需要与此工具集成。 |
| 自定义操作界面 | 测试经理希望根据正在开发或测试的产品自定义操作界面。 操作员表示他们希望在监督测试仪时有一个进度指示器来指示测试状态。 |
| 序列执行引擎 | 这是所有测试仪必需的功能。 |
| 结果报告 | 所有生产系统必需符合公司的HTML报告标准。 |
| 用户管理 | 测试工程团队由几个系统架构师和较多的测试开发人员组成。 测试经理想要分离这两个角色之间的职责。 |
| 并行测试功能 | 在对组装单元进行功能测试时，生产线会一次评估一个UUT。 但是，板卡级测试应该优化提高执行速度。 为了满足所有测试仪的需求，需要进行并行测试。 |
| 单元/设备跟踪和序列号扫描 | 公司每个产品和板卡的UUT信息通过操作员输入进行跟踪。 |
| 测试部署工具 | 公司有一个专门的测试工程师团队，负责编写测试代码。 这个团队必须能够将实验室的开发环境部署到所需的生产环境。 目前，这是手动完成的。 |
| 维护 | 测试经理需要一个正式的维护计划作为标准化工作的一部分。这个计划的一部分需要适应公司今年晚些时候当前系统生命周期结束时进行的操作系统迁移。 |

表4. Karen对测试执行软件的兴趣源于对各种测试仪进行标准化的需求。

考虑到这一标准，Karen排除了所有现有的内部解决方案。因为这些解决方案大多数是针对某个功能进行开发的。架构的一致性很低，无法扩展到其他生产线，特别是在序列生成需求、操作界面自定义和高效部署实践方面。此外，当软件出现问题或者对设备进行修改时，难以跟踪哪些测试工程师负责特定测试执行程序。

因此，Karen向她的经理提议采用商业解决方案。测试执行软件由知名的供应商提供，该供应商的其他硬件和软件工具应该已经应用在测试仪中。此测试管理软件的现成即用功能可以满足测试仪所需的序列执行要求，并采用要求的特定报告格式。测试执行软件包括一组可满足其他测试仪需求的工具，包括用户管理工具和部署实用程序。鉴于由商业供应商对其进行维护，Karen的经理不必担心软件无法兼容之后的操作系统迁移。

Karen的公司最终成功地做出了决定。总的来说，测试执行软件提供了一个灵活的框架，适用于不同的生产线。购买的测试执行软件进行标准化给公司带来了额外的好处。此外，由供应商负责提供培训，以便于测试工程师能够适应新软件。测试执行软件的购买还包括维护合同，由供应商提供常规补丁和升级。公司还可以获得技术支持资源，可以帮助解决其测试序列问题。

商业解决方案仍然是Karen公司的标准。当测试工程师由于晋升、退休或离职需要换人时，测试经理可以聘请具有测试执行程序开发经验的人员。该公司成功地从过时的操作系统迁移了两个完整版本，同时仍然保留所选择的测试执行软件。随着新产品的开发，可扩展架构可以持续满足生产需求。

## 结论

无论公司规模、行业或个人测试标准如何，都必须部署测试执行软件来实现最上层的系统管理。 这意味着需要引入一定程度的抽象，将系统的常见功能与测试代码的特定功能分开。 在构建最终解决方案之前，必须对测试执行软件的需求进行完整的评估。 许多测试工程师都在构建还是购买测试执行软件之间犹豫不定。 做出决定需要从成本、功能和维护角度仔细考虑每个解决方案的优势。

## 下一步

TestStand是业界标准的测试管理软件，可帮助测试和验证工程师更快速地构建和部署自动化测试系统。 TestStand包括一个可立即运行的测试序列引擎，支持多种测试代码语言、灵活的结果报告和并行/多线程测试。

虽然TestStand包含许多现成功能，但仍设计为高度可扩展。 因此，全球成千上万的用户选择了TestStand来构建和部署自定义自动化测试系统。NI提供培训和认证计划，每年培养和认证了超过1000名TestStand用户。

了解更多关于TestStand