Vector Software
白皮书

如何评估嵌入式软件测试工具

你的测试工具有些什么功能？

在过去的几年中，测试自动化工具市场已经变得混乱。所有的工具都声称能做同样一件事情：自动化测试。维基百科仅列出了 38 条 C/C++测试框架。然而当潜在的用户在浏览产品资料或观看简单的演示时，许多的这些测试工具看上去都是极其相似的。

本文主要为工程师在评估软件测试自动化工具，尤其是动态的测试自动化工具时应考虑的问题提供参考。

你不能通过阅读资料表来评估一个测试工具

所有的资料表都是大同小异的。商业用语都是相同的，比如：“行业领导者”、“独一无二的技术”、“自动化的测试”和“领先技术”等。屏幕截图也都是相似的：“条形图”、“流程图”、“HTML 报告”和“状态的百分比”。让人看得都 麻木了。

什么是软件测试？

所有做过软件测试的人都知道测试有很多类型。为简单起见，本文中我们将讨论以下三个方面：

- **系统测试：** *测试完全集成的应用系统*
- **集成测试：** *测试集成的子系统*
- **单元测试：** *测试一些单个的文件或类*

在大家做过的系统测试中，有一些跟最终用户将要做的操作是一样的。请注意，我们说的是“一些”，而不是“全部”。导致应用程序运行失败出现 bug 的最常见的情况是在程序的输入域中输入了预料之外的未经测试过的输入组合。

做集成测试的人不多，做单元测试的就更少了。如果你做过集成或单元测试，你很可能会痛苦地意识到从应用程序的其余部分隔离单个文件或一组文件必须生成的测试代码的数量是巨大的。在最严格的测试等级中，测试代码的编写量大于被测程序代码量的情况并不少见。因此，这些级别的测试通常应用于高安全性和高可靠性系统的市场中，比如航空、医疗设备、汽车电子和轨道交通等。

什么是“自动化测试”？

众所周知，手工的单元和集成测试过程是非常费力和费时的；因此所有进入这个市场出售的工具都将大力宣传“自动化测试”作为他们的获利点。但是，什么是“自动化测试”？自动化对不同的人理解会不一样。对于许多工程师来说，“自动化测试”意味着他们可以按下一个按钮，或者在“勾选框”中打上绿色的勾，表示他们的代码是正确的，或用“红色 X”表示执行失败。

遗憾的是这样的工具是不存在的。更重要的是，如果这种工具真的存在，你会想要使用它吗？好好想想吧。工具告诉你，你的代码是“OK”的，这意味着什么呢？它意味着代码格式化很好吗？也许吧。它意味着符合你的编码标准吗？也许吧。它意味着你的代码是正确的吗？显然没有！

完全自动化的测试是无法达到的，也不是可取的。自动化应处理测试过程中那些基本的逻辑算法的部分和劳力密集的部分。这将使软件工程师有更多的时间去做价值更高的测试工作，比如设计更好和更完整的测试。

评估工具时要问到的一个合理的问题是：“这个工具提供了多大程度的自动化？”这其实也是一个公司试图计算工具的投资回报率时所面对的最模糊的而又最主要的不确定性因素。

测试工具剖析

测试工具通常提供了各种各样的功能。对于不同的工具，厂商使用的名称是不同的，一些工具可能会缺失某些功能。作为一个公共的参考框架,我们为你评估的测试工具中可能存在的“模块” 选择了下面的名称：

解析器（Parser）	解析器模块让工具能够理解你的代码。它读取代码，并为代码创建一个中间表示（通常以树结构的形式）。基本上与编译器所做的相同。输出或”解析数据”通常被保存在一个中间语言（IL）文件中。
代码生成器	代码生成器模块采用“解析数据”来构建测试框架的源代码。
测试框架（Test harness）	虽然测试框架不是工具中的专门部分;但在测试框架的体系结构中所作的测试会影响工具所有的其它功能。因此，评估工具时框架结构是非常重要的。
编译器	编译器模块允许测试工具调用编译器来编译和链接测试框架。
目标环境（Target）	目标模块使测试可以在不同的运行时环境中轻松地运行，包括支持模拟器、仿真器、嵌入式调试器和商业 RTOS。
测试编辑器	测试编辑器允许用户使用脚本语言或复杂的图形用户界面（GUI）为测试用例设置先决条件和预期值（通过/失败的标准）
覆盖率	覆盖率模块使用户能获取执行每个测试所覆盖的代码部分的测试报告
报告	报告模块允许获取的各种数据被生成项目文档。
命令行(CLI)	命令行（CLI）使工具的使用进一步自动化，可以从脚本、makefile 文件等来调用工具。
回归测试	回归测试模块允许针对一个应用程序的某个版本创建的测试用例在新版本中重新运行。
集成	与第三方工具的集成是一个充分利用你对测试工具的投资的有趣的方式。常见的集成有与配置管理工具、需求管理工具和静态分析工具的集成。

后面的部分将详细说明你应该如何评估这些候选工具中的每一个模块。

测试工具分类/自动化水平

由于所有工具不可能都包含上述所有的功能或模块，同时也因为不同的工具所提供的自动化水平有着很大的差异，因此我们对测试工具大致分为下面几类。你的候选测试工具将属于其中的一个类别。

手工	“手工”工具，通常创建一个空测试框架，并要求你手工编写执行测试用例所需的测试数据和逻辑。通常情况下，它们会提供一种脚本语言和/或一套库函数/方法，可以用来做一些常规的事情，比如测试断言，或创建格式化的测试报告文档。
半自动化	“半自动化”工具可以把“手工”工具提供的一些自动功能加上图形界面，但仍然需要手工编写代码和/或脚本以测试更复杂的结构。此外,“半自动化”工具可能会缺少一些“自动化”工具具有的模块。仅仅内建支持在少数示例的嵌入式目标环境上部署。
自动化	“自动化”工具，具有上一节中列出的每个功能区或模块的功能。这类工具将不需要手动编码，并支持所有语言结构，以及在各种不同的嵌入式目标环境上部署.

工具的微妙差异

除了比较工具的功能和自动化水平以外，评估和比较所使用的测试方法也是很重要的。例如，当你用一些工具创建一个测试项目时，除非你自己动手尝试去完成一些操作，否则这些工具仅仅将文件加载到它的 IDE，而不会做任何创建测试框架或测试用例的工作。

这可能隐藏了工具的潜在缺陷，因此重要的是不仅要将你的代码加载到工具中，而且还要尝试为你正在测试的类中的每个方法建立一些简单的测试用例。工具是否会建立一个完整的测试框架？所有的桩函数（Stubs）都是自动创建的吗？你可以使用图形用户界面（GUI）来定义测试用例的参数和全局数据吗？或者如果你正在进行手工测试，你是否需要编写测试代码呢？

类似地，工具之间对嵌入式目标环境的支持也有很大的差异。如果一个厂商说：“我们支持所有的编译器和目标环境”，那你需要警惕了。这话的含义其实是说：“你需要自己去完成让我们的工具能在你环境中运行的所有工作”。

如何评估测试工具

以下几节将详细描述在软件测试工具的评估过程当中你应该研究的问题。理想情况下，你应该通过亲手测试正在评估的每一个工具，来确认这些信息。

由于本文下面的部分技术性很强，我们想向你解释一下下文的表述格式。对于每一个部分，我们都有一个标题来描述将要讨论的问题，一个关于该问题重要性的描述，还有一个“要点”部分来概括需要考虑的问题。

另外，除了表述格式以外，我们也应对专业术语作出注解。 “函数/方法（function）”一词是指一个 C 函数或 C + +类的方法，“单元”是指一个 C 文件或 C + +类。最后，请记住，几乎所有的工具都能在一定程度上支持“要点”部分中提到的问题，你的工作是评估自动化程度、易用性，以及支持的完整性。

解析器和代码生成器

要建立一个 C 解析器是相对容易的，但要建立一个完整的 C + +解析器是非常困难的。工具评估过程中需要回答的问题之一是:“解析器技术的健壮性和成熟性如何”？一些工具厂商使用商业解析器，他们从解析器技术公司获得许可，还有一些厂商使用自主开发的解析器。使用你项目中具有代表性的复杂的代码结构来评估工具，可以验证解析器和代码生成器的健壮性。

要点:

- 解析器技术是商业外购的还是自主研发的？
- 支持哪些语言？
- C 版本的工具和 C + +版本的工具是否相同？
- 全部是 C ++语言实现的，还是局限于其中一部分？
- 工具能测试我们最复杂的代码吗？

测试驱动程序

测试驱动程序是控制测试的“主程序”。下面是一个驱动程序的简单例子，它将用来测试标准 C 库中的正弦函数：

```
#include <math.h>
#include <stdio.h>
int main () {

   float local;
   local = sin (90.0);
   if (local == 1.0) printf ("My Test Passed!\n");
   else printf ("My Test Failed!\n");
   return 0;
}
```

虽然这是一个很简单的例子，“手工”工具可能会要求你手工输入（和调试）这一小段代码片段，“半自动化”工具可能会给你某种脚本语言或简单的图形用户界面（GUI）来输入正弦函数的输入参数值。“自动化”工具将提供一个用于构建测试用例、集成代码覆盖率分析的全功能图形用户界面（GUI），一个集成调试器，以及与嵌入式目标环境的集成部署。

不知你是否注意到，该驱动程序有一个 bug。那就是正弦函数实际上应该使用弧度值而不是度数来作为角度的输入值。

要点

- 驱动程序是自动生成的，还是自己编写的？
- 我可以不写任何代码来测试下面的内容吗：
  - 测试覆盖一定范围的值
  - 组合测试
  - 数据分区测试（等价类）
  - 输入值列表
  - 预期值列表
  - 预期异常值
  - 信号处理
- 我可以在相同的测试中建立一个调用不同方法的序列吗？

为关联的函数/方法打桩

当你想控制在测试过程中关联函数/方法的返回值时，你就需要为关联函数/方法建立替代值，打桩是集成和单元测试中非常重要的部分，因为它允许你从应用程序的其它部分中隔离出被测代码，更容易触发你所感兴趣的单元或子系统的执行。

许多工具都需要手动生成测试代码，才能使得桩（stub）不仅仅只是返回一个静态的标量值（return 0;）。

要点

- 桩是自动生成的，还是你为它们写的代码？
- 是否自动支持复杂的输出（结构体，类）？
- 每次调用桩能返回不同的值吗？
- 桩会记录它被调用的次数吗？
- 桩会记录多个调用的输入参数吗？
- 你可以对标准 C 的库函数比如 malloc 打桩吗？

测试数据

“半自动化”和“自动化”工具生成测试用例的基本方法有两种。一种是“数据驱动”架构，另一种是“单一测试”架构。

对于数据驱动架构，所有被测单元都创建了测试框架，并支持在这些单元中定义的所有函数/方法。当要运行测试时，工具只是简单地通过一个数据流，如文件句柄或像一个通用异步收发器(UART)一样的物理接口，提供执行所需的测试数据。

对于“单一测试”架构，在每次运行测试时，工具都将建立该测试的测试驱动程序，编译并链接成一个可执行文件。这里有两点需要注意：首先，单一测试方法所需要的额外的代码的生成、编译和链接在测试执行的时候都将花费更多的时间；其次，你最后要为每个测试用例建立一个单独的测试框架。

这意味着候选工具可能看上去在运行一些有名无实的测试用例，但可能在更复杂的测试中却无法正确地工作。

要点

- 测试框架是数据驱动的吗？
- 执行测试用例（包括所有代码生成和编译时间）要多久？
- 可以在测试工具 IDE 之外编辑测试用例吗？
- 如果不能，我是否已经用复杂的示例代码充分地玩转该工具以了解工具的所有限制了？

测试数据的自动化生成

一些“自动化”工具提供了测试用例在一定程度上的自动化创建。这可以用不同的方法来完成。以下段落描述了其中的一些方法：

MMM 极小-中间-极大值测试用例	MMM 的测试将重点函数/方法的输入类型的边界值。 C 和 C ++代码通常不会控制超出边界范围的输入值。工程师已在他们头脑中已经清楚函数/方法的值域范围，他们却往往没有控制超出边界范围的输入。
EC 等价类	等价类测试为每个数据类型创建了“分区”，并从每个分区选取了一个样本值。设定相同分区中的值将以同样的方式触发应用程序。
RV 随机值	随机值测试会为每个函数/方法的参数设置随机值组合。
BP 基本路径测试	基本路径测试使用基础路径分析来测试贯穿被测程序的各条可能的执行路径。基本路径测试可以自动达到一个高水平的分支覆盖率。

在考虑构建自动化测试用例时，要记住的关键一点是它们使用的目的。自动化测试可以很好地用于测试应用程序代码的健壮性，而不是正确性（即使它们提供了很高的代码覆盖率）。对于正确性，你创建的测试用例必须是基于应用程序应该要做什么（测试需求），而不是基于它做了什么（代码）。

与编译器的集成

与编译器的集成需要注意两点。一点是允许测试框架（Test harness）进行自动编译和链接，不需要用户去设置编译器所需的选项。另一点是允许测试工具兼容编译器所使用的任何特定的扩展语言。尤其是交叉编译器，它们提供非 C / C + +语言标准的扩展是很常见的。一些工具使用宏定义方法（＃）将这些扩展定义为空字符串。这种拙劣的方法是非常不好的，因为它改变了编译器产生的目标代码。例如，考虑下面的带有 GCC 属性的全局外部变量：

```
extern int MyGlobal __attribute__ ((aligned (16)));
```

如果你的候选工具在定义全局变量 MyGlobal 时不能维持其属性不变，那么因为内存边界对齐不一样，代码在测试时和实际部署时将会表现得不一样。

要点

- 工具是否能自动编译和链接测试框架？
- 该工具是否能兼容并实施编译器特定的语言扩展？
- 编译器有什么类型的接口（IDE，CLI 等）？
- 工具是否提供接口用来从你的开发环境中导入项目设置，还是必须手工导入？
- 如果该工具导入了项目设置，该导入功能是通用的还是局限于特定的编译器或编译器系列的？
- 测试工具是否和调试器集成以允许你调试测试用例？

支持嵌入式目标测试

在这一节中，我们将使用术语“工具链”来表示整个交叉开发环境，包括交叉编译器、调试接口（模拟器）、目标板和实时操作系统（RTOS）。重要的是要考虑候选工具是否能很完善地与你的工具链集成，并且要清楚如果你迁移到一个不同的工具链上，该工具中需要改变些什么。

此外，了解目标环境集成的自动化水平和健壮性也很重要。正如前面提到的：如果某个厂商说：“我们支持所有的编译器和目标板。”他们的意思是说：“你需要自己去完成让我们的工具能在你环境中运行的所有工作。”

理想情况下，你选择的工具将允许你只需点击一个“按钮”即可完成“测试执行”所涉及的所有复杂工作，包括将测试用例下载到目标环境，并将测试结果捕获到主机平台，，从而不需要用户再做任何特殊的操作。

嵌入式目标测试的另一个难题是硬件的可用性。通常情况下，硬件与软件是并行开发的，或者硬件的可用性是有限的。一个关键的特征是能否先在一个本地的环境开始测试，后期再过渡到实际的硬件环境。理想情况下，工具的使用是不受硬件条件限制的。

要点

- 能支持我的工具链吗？如果不能，那它能有办法被支持吗？“支持”是什么意思呢？
- 我能在主机系统上建立测试，然后将它们用于目标环境的测试？
- 测试框架是怎样下载到目标上的？
- 测试结果是如何被获取并返回到主机的？
- 能现成地支持哪些目标、交叉编译器和 RTOS？
- 为一个新的工具链建立支持的工作由谁来完成？
- 工具链集成的任何部分都是用户可配置的吗？

测试用例编辑器

显然，使用测试工具时，你大部分的时间都会花在测试用例编辑器上。如果本文中之前提到的各方面都实现了真正的自动化，那么搭建测试环境和目标连接所占的时间量将是最小的。记得我们在开始时说的，作为工程师，你会想将更多的时间用在设计更好和更完整的测试上。

当你在评估工具时，需要回答的关键的问题是为复杂的结构设置输入和预期值有多困难？市场上的所有工具都提供了一些简单的方法来设置标量值（scalar values）。例如，你的候选工具是否提供了一个简单而直观的方式来构建一个类？用一个抽象的方法（比如一个向量或一个映射）来建立一个 STL 容器如何？这些都是在测试用例编辑器中要评估的问题。

本文的下面部分有“支持”，也有“自动化支持”。在评估你可能感兴趣的结构时要考虑到这一点。

要点

- 能否显示标量值范围？
- 是否显示数组的大小？
- 是否可以简单地用标签设置最小值和最大值而不用直接设值呢？这对当类型改变时保持测试的完整性是很重要的。
- 是否支持特殊的浮点数（例如，为 NaN，+/ - 无穷大）？
- 你可以做组合测试（在一定范围内对 5 个参数取一系列不同的值，让工具来完成那些值的所有组合）吗？
- 编辑器是否支持不同“数的进制”，让你可以很容易地交替输入不同进制的数，比如十六进制、八进制和二进制值？
- 对于预期的结果，你可以很容易地为浮点值输入绝对容错范围（如+/ - 0.05）和相对容错范围（例如：+/ - 1%）吗？
- 测试数据可以很容易地从其它数据源（如 Excel）导入吗？

代码覆盖率

大部分“半自动化”工具和所有的“自动化”的工具都内置了一些代码覆盖率分析功能，使你能看到测试用例所覆盖的那部分应用程序的指标。一些工具以表格形式展示这些信

息，一些展示为流程图，还有一些以在源代码中添加标注的形式展示。表格用于汇总是很方便的，但如果你想实现100%的代码覆盖率，在源代码中标注才是最好的。这种形式通过对已覆盖、部分覆盖和未覆盖的结构标注不同的颜色，来展示最初的源代码文件。它使你可以轻松地看出要达到100%覆盖率所需要补充的测试用例。

了解插装（instrumentation）的影响也很重要。额外的源代码添加到你的应用程序中。有两方面需要考虑：一是目标代码的增加，还有就是运行时系统开销。重要的是要明白，你的应用程序是内存受限的还是实时受限的（或是两者）。这将有助于你了解哪个问题对你的应用程序是最重要的。

要点

- 每种类型的插装，代码增加的大小是多少？
- 每种类型的插装，运行时系统开销的增加是多少？
- 插装过程可以集成到你的编译系统(“make” system 或 “build”system)吗？
- 呈现给用户的覆盖率结果如何？是否有带标注的图形化的覆盖率浏览列表，或只是度量表？
- 从目标环境获得的覆盖率信息如何？过程灵活吗？数据能缓存到 RAM 中吗？
- 是否支持语句、分支（或判定）和修正条件/判定（MC / DC）覆盖？
- 能在一次执行中获取多种覆盖类型信息吗？
- 覆盖数据能跨多个测试环境进行共享吗（例如一些系统测试过程中获取的覆盖率信息，和从单元和集成测试中获取的覆盖率信息相结合）？
- 你可以使用覆盖率数据单步遍历测试执行过程，在不使用调试器的情况下观察通过你的应用程序的控制流程吗？
- 你可以在单个报告中获得执行所有测试用例的总的覆盖率吗？
- 工具符合 DO-178B 标准和 FDA 对医疗器械的“Intended use”标准吗？

回归测试

使用测试工具有两个基本的目的。首先是为了节省测试时间。如果你读到这里，我想你会同意这一点！其次是使创建的测试可作用于应用程序的整个生命周期。这意味着，建立测试所花费的时间和金钱产出的应该是可重用的和易于配置管理（版本管理）的测试用例，即使应用程序随时间有所改变也不例外。你的候选工具要评估的主要问题是需要“保存”哪些具体的内容，以便将来能运行相同的测试，以及如何控制测试的重运行。

要点

- 为了回归测试，需要对哪些文件做配置管理？
- 工具是否有一个完整的，文档化的命令行接口（CLI）？
- 这些文件是纯文本的还是二进制的？这会影响你使用 diff 工具来评估这些文件随时间的变化情况的效率。
- 必须要对工具生成的架构文件（harness）做配置管理吗？
- 能与配置管理工具的集成吗？
- 为一个单元创建一个测试用例，现在改变一个参数的名称，并重建你的测试环境。需要多长时间？复杂吗？
- 工具是否支持数据库技术和统计图表来分析测试执行和代码覆盖率随时间变化的趋势。
- 你是否可以用同一组测试用例自动测试多个不同版本的代码？
- 是否支持分布式测试，允许各部分测试用例在不同的物理机上运行,以加快测试速度？

报告

大多数工具都会提供相似的报告。它们至少应该创建一个易于理解的报告来展示输入、预期输出、实际输出以及预期值和实际值的比较。

要点

- 支持哪些输出格式？HTML?Text?CSV?XML?
- 是否可以方便地同时获取一个针对整个项目的概要报告和一个针对单个函数/方法的详细报告？
- 报告内容是用户可配置的吗？
- 报告格式是用户可配置的吗？

与其它工具的集成

暂且不论特定工具的质量或使用效果，所有的工具需要在多厂商的环境中运行。大公司已经花费了很多的时间和金钱来收购很多小公司，意欲为所有人提供能做任何事情的“工具”。有趣的是，这些大型的工具套件的整体使用效果往往远不及各部分的总和。看上去，这些公司往往就像只是在将4-5个很酷的小工具打包成一个笨重的但又不可用的工具。

除了与我们已经提到的与开发工具链集成以外，对测试工具最有用的集成还有与静态分析、配置管理和需求管理工具的集成。所有人都希望把他们的测试资源放到配置管理下，使测试用例能够被他们复用，并且大多数人还想跟踪他们的需求和测试用例。

要点

- 你的候选工具集成了哪些工具，最终用户能另外添加集成吗？

测试工具其它不错的功能

好了，我们已经完成了关于“测试工具解剖”的探讨。之前描述的所有功能，应该在任何一个被认为是自动化的测试工具中都能找到。在接下来的几节中，我们将列出一些不错的（虽然不太常见）的功能，以及说明这些功能重要性的理由。这些功能可能不同程度地适用于你具体的项目。

真正的集成测试/多单元测试

集成测试是单元测试的扩展。它用来检查各单元之间的接口，需要你将一些能构建某种功能过程的单元组合在一起。许多工具声称可以通过使用测试框架链接多个单元的目标代码来支持集成测试。这种方法将多个被测文件构建到测试框架的可执行程序中，但它却没法触发这些其它单元内部的函数/方法。理想情况下，你将可以在单个测试用例中，以任何顺序调用任何单元中的任何函数/方法。测试单位之间的接口，通常会发现很多应用程序中隐藏的异常和缺陷。事实上，对那些没有做过单元测试的项目来说，集成测试也许是一个不错的开始。

要点

- 我能在测试环境中包含多个被测单元吗？
- 我能为这些类创建复杂的测试场景，在一个测试用例中，执行一个跨越多单元的函数序列吗？
- 我能获取多单元的代码覆盖率指标吗？

动态桩

动态桩意味着你可以任意地地打开和关闭个别函数/方法的桩。这使你可以在测试单个函数/方法时将所有其它函数/方法打桩（即使它们就存在于被测函数所在的同一个文件中）。对于非常复杂的代码，这是一个非常好的功能，它使得测试工作更加容易。

要点

- 能在函数/方法级别任意挑选打桩的对象吗？还是仅限于单元级别（同一个文件/类中）？
- 能在每个测试用例中打开和关闭桩函数/方法吗？
- 桩函数/方法能被自动生成吗？（参考前面的章节）

库和应用程序级线程测试（系统测试）

系统测试的挑战之一是，对完全集成的应用程序的测试，可能需要用户按下按钮，翻转开关，或在控制台进行输入。如果应用程序是嵌入式的，那么输入的控制将更加复杂。假设你可以在函数级别触发你完全集成的应用程序，类似于集成测试是怎样完成的一样。这将允许你仅仅依赖于应用程序的API即可构建复杂的测试场景。

一些更现代的工具允许你用这种方式来测试。这种测试方式的另外一个好处是你不需要源代码来测试应用程序。你只需要API的定义（通常是头文件）。这种方法使测试人员能够以自动化和可编写脚本的方式来进行系统测试。

敏捷测试和测试驱动开发（TDD）

测试驱动开发意味着将测试带入开发过程比以往任何时候都早。在你的应用程序代码完成之前就构建你的测试用例，而不是编写应用程序代码完成后再进行单元测试。这是一种流行的新的开发方法，秉承的是“测试优先”和不间断测试的思想。如果你打算使用敏捷开发法，你的自动化工具必须要支持这种测试方法。

与需求管理工具的双向集成

如果你关心需求与测试用例的关联，那么测试工具与需求管理工具的结合将会是很有用的。如果你对这个功能感兴趣，接口双向是很重要的，从而使得当需求被标记到测试用例时，测试用例的信息，比如测试名称以及通过/失败状态，也可以导出到你的需求数据库。这将使你确认你的需求测试的完整性。

工具认证

如果你处于一个对研发和测试有特定规范的行业，如航空电子、医疗器械、轨道交通、汽车电子或高安全工业控制等领域，那么你将有责任对构建和测试你应用程序的工具进行“认证”。

这样的认证要求提供相应的文档来说明工具应该做什么（工具操作需求）和证明工具的操作合乎那些需求的测试。理想情况下，厂商拥有这些现成的材料以及曾经使用过你行业认证数据的客户档案。

要点

- 工具厂商能提供专门针对你的目标环境和工具链制作的认证材料吗？
- 哪些项目成功地使用了这些材料？
- 这些材料是如何获得许可的？
- 材料是如何定制和批准用于特定项目的？
- 如果是一个 FAA 的项目，这些用这些材料已经成功地用于通过 Do-178B 的 A 级认证了？
- 如果是一个 FDA 项目，工具是否被验证过可以用于“Intended use”？。

结论

希望本文提供的有用信息，能对你选择测试工具厂商的产品有所帮助。对不同的项目，提出的每个点的重要性将相对地有所不同。我们最后的建议是：

- 用能代表你应用程序复杂性的代码来评估候选工具。
- 用你项目将要使用的相同的工具链（嵌入式环境）来评估候选工具。
- 与厂商的老客户们沟通，问他们一些本文中列出的问题
- 咨询工具技术支持团队。试试直接提交一些问题给技术支持团队——而不是销售代表。

最后，请记住，大多数嵌入式软件测试工具，多少都会支持“要点”部分中提到的项目。你的工作就是评估：自动化程度，易用性，和支持的完整性。

创提信息科技（上海）有限公司 – Trinity Technologies

专注于嵌入式软件研发质量和自动化测试的方案和咨询服务，提供覆盖软件测试整个流程的完整的解决方案，包括从研发前期的代码级测试到后期的系统级测试，从静态分析到动态测试，从编码检查，单元测试、集成测试到性能测试和测试覆盖率分析等。

公司通过专业的自动化工具（如 DT10, VectorCAST, PRQA, SQUORE 等）和服务满足不同客户对软件质量和测试的需求，持续协助客户改进软件研发质量和效率。客户主要集中在高安全和高可靠性领域，如国防和航空航天、轨道交通、汽车电子、医疗器械、工业控制、通讯和电力电子等行业。公司提供的领先的解决方案不仅为数以百计的客户提高产品质量，还协助客户遵循高安全和高可靠性行业的合规性要求，如 DO-178B/C，IEC61508, EN50128, ISO26262, IEC62304 和 MISRA 等行业标准，并获得相关机构认可和认证。

TRINITY TECHNOLOGIES
创提信息科技（上海）有限公司
www.trinitytec.com.cn

地址：上海市浦东新区张江高科张衡路 200 号 3 栋 3207E 室
成都市高新区盛和一路 88 号康普雷斯 B 座 1914 室
电话：021-51328530（上海办公室）
028-83397325 （成都办公室）
邮箱：info@trinitytec.net