使用 **Parasoft C++test** 以符合 **ISO 26262** 软件标准

在汽车软件中实现功能性安全

引言

安全功能越来越多地在电气、电子或可编程电子系统中得到实现。这些系统一般都是非常复杂的，这就使得在实际中完整地判断每个失效模式(failure mode)或测试所有可能的行为成为了不可能完成的任务。虽然预测其安全方面的性能非常困难，但测试仍然是非常有必要的。关键的挑战在于设计一种能够预防危险性失效或能在这些失效发生的时候对其进行控制的系统。

安全性将成为未来汽车开发中的一个关键因素。新功能——不仅仅在驾驶员辅助系统领域，同样在车辆动态控制以及主动和被动安全系统中——越来越多地触及到了安全性工程的范畴。未来的开发以及对这些功能的集成将会进一步提升对安全系统开发流程的需求，同时还需要提供所有合理安全性目标已满足的证据。

随着复杂性增强、软件内容以及机电层面实现的趋势，系统化的失效以及随机硬件失效的风险有显著增加的趋势。通过提供可行的需求以及流程，ISO/DIS 26262 包含了能够将这些风险降低到可接受程度的指南。

本文档的宗旨在于详细阐述如何使用 Parasoft C++test 帮助汽车软件开发组织遵守 ISO/DIS 26262 标准。本文将首先介绍 ISO/DIS 26262 的标准及其目标。然后将讨论 Parasoft C++test 是为软件开发以及测试提供自动化最佳实践的集成解决方案。最后，将阐述使用 Parasoft C++test 所能全部或部分满足的 ISO/DIS 26262 标准细节。

关于 ISO/DIS 26262

ISO/DIS 26262 是为了满足汽车领域中对电气/电子（E/E）系统中应用层面的需要，对 **IEC 61508** 进行修订后的版本。国际标准草案（DIS）是自 2009 年 6 月起对公众开放的最新版本。相应的国际标准（IS）将于 2011 年 6 月公布。

ISO/DIS 26262:

- 提供汽车安全生命周期（管理、开发、生产、操作、服务、停运），并且支持在这些生命周期的各个阶段中对这些事件进行剪裁；
- 提供汽车专用的、基于风险的方法来判定风险的种类（汽车安全完整性级别——ASIL）；
- 使用 ASIL 来指定相应器件必需符合的可接受的驻留风险级别的安全性需求；并且
- 提供验证以及确认措施的需求以确保满足了充分和可接受的安全性级别。

ISO/DIS 26262 覆盖了整个开发流程中功能安全性的所有方面（包括诸如需求说明、设计、实现、集成、验证、确认以及配置等）。

标准文档中的*第 6 部分*明确指出产品开发在所处的软件级别。这包括产品开发初始阶段所处的软件级别的需求、软件安全性需求的说明、软件架构设计、软件单元设计以及实现、软件单元测试、软件集成和测试以及软件安全性需求的验证。

ISO/DIS 26262 标准中所定义的方法应该按照相应的 ASIL 级别进行选择，例如 ASIL 的级别越高，相应需要采用的措施就越严苛。

软件工具的使用能够简化并自动进行 ISO/DIS 26262 标准要求的安全性相关项目或元素开发所必需的事件或任务。该标准文档中的*第 8 部分的第 11 章节*描述了软件工具资格认证的流程。软

件工具的资格认证的目的是在开发安全性相关项目或元素时所使用的软件工具的合适性提供依据，使用户能够可靠地执行 ISO/DIS 26262 标准所要求的事件以及任务。如果关于如何合格地使用 C++test 确认以及验证 ISO/DIS 26262 有任何疑问，请联系您的 Parasoft 代表。

关于 **Parasoft C++test**

Parasoft C++test 是经广泛证明的最佳实践集成解决方案，它能有效提高开发团队工作效率和软件质量。C++test 能促进：

- **静态分析** —— 静态代码分析，数据流静态分析以及度量指标分析
- **同行代码审查**流程自动化——准备、提示以及追踪
- **单元测试**——单元测试创建、执行、优化以及维护
- **运行时错误检测**——内存访问错误、泄漏、崩溃以及其它

这为开发团队提供了一种实际的方式来预防、发现以及纠正相关的错误，从而确保其 C 以及 C++代码如预期般工作。为了促进快速修复，每个检测到的问题都基于可配置的严重级别分配进行排序，并自动分发到相应代码的开发者处，同时分配到他或她的 IDE 环境中，并配以该问题代码处的直接链接以及如何修正该问题的描述。

对于嵌入式以及交叉平台开发而言，C++test 可以同时用在基于宿主机以及基于目标平台的代码分析以及测试流程中。

自动进行代码分析以监测是否遵守标准

采用合适的编码策略可以建立预防性的编码习惯，从而消除整类编码错误的发生。C++test 对这些代码进行静态分析，以检测其是否遵守相应策略。如需配置 C++test 以增强某个团队或组织专用的编码标准策略，用户可以使用内建以及自定义规则来建立自己的规则集。代码分析报告可以按不同的形式生成，包括 HTML 以及 PDF 格式。

数百条的内建规则——包括 MISRA、MISRA 2004、全新的 MISRA C++标准的实现，HIS 源码度量指标、Meyer 在《Effective C++》以及《Effective STL》等书中推荐的以及来自其它主流源码的准则——有助于从使用不当的 C/C++ 语言识别潜在缺陷，实施最佳编码实践，以及提升代码的可维护性和可重用性。通过图形化的 RuleWizard 编辑器创建的自定义规则，可以增强标准 API 的使用并防止单个缺陷被发现后类似的应用程序缺陷重复出现。

在不运行软件的情况下识别运行时缺陷

作为 C++test 先进的数据流静态分析模块，BugDetective 能模拟可行的应用程序执行路径——这些路径可能会跨越多个函数和文件——并判断这些路径是否会触发特定类别的运行时缺陷。能够发现的缺陷包括使用未初始化或无效的内存、空指针解引用、数组及缓存溢出、除零、内存和资源泄漏以及各种无效代码（dead code）。这种不运行软件即可发现运行时错误的能力对于嵌入式代码是尤其有价值的，因为在嵌入式开发中要进行详细的运行时分析以检测这些错误往往是无效或不可能的。

由于在开发者的 IDE 环境中提供了对潜在缺陷的完整路径追踪，C++test 极大地简化了缺陷分析过程。自动化到代码的交叉链接能帮助用户迅捷地跳转到某段高亮显示的分析路径中的任意一点上。

合理化的代码审查

代码审查被认为是发现代码缺陷最有效的方法。不幸的是，由于认为代码审查会耗费大量额外的精力，很多组织都未充分使用代码审查。C++test 的代码审查 (Code Review)模块能自动地准备、提示以及追踪同行代码审查，从而建立起非常高效的面向团队的审查流程。所有代码审查的状态，包括审查者所添加的所有注释都会被 C++test 的基础设施[1]加以维护并进行自动分配。C++test 支持两种典型的代码审查流程：

- **提交后代码审查**。该模式是通过自定义的源码控制器接口来自动识别代码库中的代码更改而实现的，并基于预设的更改代码与审查者的映射关系创建代码审查任务。
- **提交前代码审查**。用户可以在桌面上选择某一文件集并分配到审查者进行审查，或者自动地检测所有本地更改源代码的方式来初始化代码审查。

团队代码审查的效果可以通过 C++test 的静态分析能力进一步地增强。由于自动监视团队的编码策略，从而消除了对逐行审查代码的必要性。一旦代码被提交进行审查，相应的违例就已经被发现并清除了。这样，审查者就可以更加注重对算法的检查，对设计本身的审查，以及查找那些自动化工具无法检测到的隐蔽的错误。

监测应用程序以发现内存问题

应用程序内存监测是用于消除严重的内存相关缺陷并具有零误报率的最著名方法。对正在运行的应用程序进行监测以查找某类特定问题——诸如内存泄漏、空指针、未初始化内存以及缓存溢出等——测试过程一旦结束后，结果将会立即可见。

无需进行高级并且耗时的测试活动，被插装的应用程序——为进行监测添加了额外的代码——将通过标准的功能测试并标注出所存在的所有问题。应用程序可以在目标设备、模拟的目标平台或宿主机上执行。收集到的问题将直接显示在开发者的 IDE 中，并提供了解以及修复该问题的详细信息（包括内存块大小、数组索引、分配/释放堆栈的追踪等）。

覆盖率度量指标将在应用程序执行时进行收集。这些指标可以用来说明哪部分应用程序经过了测试，以及调整回归单元测试用例集（作为功能测试的补充）。

运行时错误检测允许用户：

- 通过简单的功能测试查找复杂的内存相关问题——例如内存泄漏、空指针、未初始化内存以及缓冲溢出等
- 通过应用程序运行来收集代码覆盖率
- 通过在真实目标环境中执行被监测的应用程序来提升测试结果的精确性

具有覆盖率分析的单元及集成测试

C++test 的自动化能力极大地提升了测试新开发或既有代码的正确性和可靠性的效率。C++test 自动地生成完整的测试套件，包括测试驱动以及用于各个独立函数的测试用例，这些测试套件都是以类似 CppUnit 格式的纯 C 或 C++代码的形式提供的。这些测试套件，无论是否进行修改，都可以用于对代码的功能行为进行初始化验证。通过使用边界用例条件，这些自动生成的测试用例同样能检测函数对未预期输入的相应状况，从而发现潜在的可靠性问题。

完全通过一组具体的 GUI 实用程序来简化测试的创建和管理。图形化的测试用例向导（Test Case Wizard）能帮助开发者为选中的功能快速创建黑盒功能测试套件，而不必要担心它们之间的

[1] 译注：测试基础设施(Infrastructure)指机构中与测试相关的各种硬件、软件、服务及数据通信设施等的总和。

内部运作或嵌入数据的关联性。数据源向导（Data Source Wizard）能协助参数化测试用例以及桩函数——使开发者以最少的努力提升测试范围和覆盖率。桩函数视图（Stub View）能促进桩函数的分析以及生成，在该视图中，将显示出代码中所用到的所有函数并允许用户为任何在测试范围内不可用的函数创建桩函数——或为某个特定测试目的更改既有函数功能。测试执行以及分析集中在测试用例浏览器（Test Case Explorer）中，该浏览器统一管理所有既有项目测试套件并提供明确的通过/失败状态。这些功能对于支持自动化的持续性集成和测试“随工程进度而测试”的开发是非常有帮助的。

通过自动记录运行时测试结果的测试断言来捕捉既有的软件行为，自动生成和用户编写的测试用例都可以用来生成回归测试用例的基础。随着代码库的演进，C++test 重新运行这些测试并将当前测试结果与最初捕捉到的“黄金集”进行比较。C++test 可以轻松地配置为使用不同的执行设置、测试用例以及桩函数以支持在不同环境中的测试（例如不同的持续性集成阶段、测试未完成的系统或测试已完成系统中的某个特定部分）。

多重度量指标测试覆盖率分析器，包括语句、分支、路径以及 MC/DC 覆盖率，能帮助用户评估测试套件的效率以及完成度，同时还能显示测试以及验证需求的遵从性，例如 DO-178B。测试覆盖率是通过所支持的所有覆盖率度量指标中高亮显示的代码来表示的——在 GUI 中或以不同颜色标注列举出的报告。覆盖率摘要报告包括文件、类以及函数数据可以不同的格式生成。

可配置的详细报告

C++test 的 HTML、PDF 以及自定义格式的报告可以通过 GUI 控件或选项文件来进行配置。标准的报告中包含代码分析以及测试结果通过/失败概要信息、已分析文件列表以及代码覆盖率概要信息。这些报告可以通过自定义设置以包含当前可用的静态分析检测列表、对于测试中扩展测试输出的通过/失败状态、关键度量指标的趋势图参数以及带颜色标注的所有代码覆盖率结果的完整代码列表。通过基于角色的邮件发送过滤器，生成的报告可以自动通过电子邮件进行发送。除向相关标注缺陷代码的开发者直接发送数据之外，C++test 还会向管理人员和团队负责人发送概要报告。

高效的团队部署

C++test 能建立一种高效的流程来确保软件验证任务能够集成在团队的既有工作流程中并自动化进行——使开发团队的精力能够更加注重于真正需要人类智慧的任务。缺陷的审查以及纠正是通过自动化任务指定及分配实现的。每个发现的缺陷都将进行优先级排序，指定到该段代码的相应开发者处，并分发到他或她的 IDE 中，同时提供到该段代码的交叉链接以及完整数据。为帮助管理人员评估及记录趋势，集中报告机制能确保提供质量状态以及过程的实时可视化报告。这些数据同时还能帮助确定是否需要额外措施来满足内部目标或证明符合质量策略。

团队工作流程模块

将 Parasoft C++test 与 Parasoft Concerto 相集成，能帮助团队针对既定策略以及所遵守的流程标准来配置工作流程，包括 CMMI 以及 SPICE。Concerto 提供完善的追踪能力，这些追踪能力包括源代码、自动化测试、对所有工程部件（需求、缺陷/增强以及任务）的手动测试。通过追踪以及分析软件开发度量指标以及过程，Concerto 能够判断哪些事务是耗费人力和时间的，同时使得开发团队能够快捷地发现特定的问题以及问题区域。

C++test 符合 ISO/DIS 26262 软件标准

下面将描述使用 Parasoft C++test 来实现或部分实现 ISO/DIS 26262 中第六章：*产品开发：软件级别*的特定部分。请注意，此处所提供的信息只是对 C++test 在 ISO/DIS 26262 软件确认与认证过程的一个简要介绍。如需澄清 ISO/DIS 26262 标准所定义的任何需求，请具体参考该标准或向功能安全性方面的专家进行咨询。如果关于如何在 ISO/DIS 26262 确认与认证过程中使用 C++test 还有其它的问题，请联系您的 Parasoft 代表。

5 相应软件级别中产品开发的初始化

ISO/DIS 26262 标准的第 6 部分定义了关于软件开发以及验证的一般性信息。

5.4.5

该条标准定义了软件开发流程中的方法以及工具的合适应用。

需要的方法	C++test 验证方法
选中的方法以及相应的软件工具应该得以使用	使用 C++test 作为软件确认工具能够满足该需求（需要使用工具资质认证）

5.4.6

该条标准描述了达成软件设计及实现正确性的需求。此处所描述的方法同时适用于建模以及编程语言。

需要的方法	C++test 验证方法
增强低复杂度的策略	报告圈复杂度、基本复杂度以及额外的软件代码度量指标
使用语言的子集	增强编码标准，例如对不安全语言结构的检测
增强强类型化的策略	查找隐式类型转换
使用保守性实现技术	增强保守性编程是通过使用合适的编码标准规则达成的，例如，检查 *malloc* 函数的返回值，检查调用函数返回值的错误代码等
使用既定的设计原则	增强业界广泛认知的编码标准规则集，例如 MISRA C/C++、JSF、HIS 源码度量指标等等
使用非二义性的图形化表述	增强特定格式转换
使用风格指导	增强特定的编码习惯

需要的方法	C++test 验证方法
使用命名习惯	增强特定的命名习惯

8 软件单元设计以及实现

本节定义了如下一些流程：

- 指定软件单元。
- 实现软件单元。
- 设计以及实现的确认。

8.4.4

该条标准指定了软件单元设计与实现的设计原则。

需要的方法	C++test 验证方法
软件单元实现的设计原则，例如*变量初始化*、*禁止隐式类型转换*等。	编码标准一致性分析，包括： • MISRA C 规则 • MISRA C++规则 • 附加标准

关于 C++test 对软件单元实现的特定设计原则支持的更多详细信息，请参考《使用 Parasoft C++test 以满足 ASIL 需求》这一文档。

8.4.5

该条标准指定了检测软件单元设计以及实现的确认方法。

需要的方法	C++test 验证方法
控制流分析	控制流分析（Bug Detective）
数据流分析	数据流分析（Bug Detective）
静态代码分析	编码标准一致性分析
源代码审查	自动化同行代码审查模块
源代码推演	自动化同行代码审查模块

9 软件单元测试

本部分定义了软件单元测试的计划、定义以及执行的流程。

9.4.1

该条标准描述了关于单元测试执行的一般性信息。

需要的方法	C++test 验证方法
单元测试执行	单元测试执行模块 显示结果的报告模块
单元测试说明	测试用例向导模块允许进行图形化的测试用例说明 可配置的单元测试生成模块根据定义的说明创建测试套件 测试用例浏览器模块显示带有通过/失败状态，已定义的测试用例的列表

9.4.2

该条标准描述了用于指定以及执行单元测试的方法。

需要的方法	C++test 验证方法
基于需求的测试	将测试用例与需求和/或缺陷进行映射——与团队工作流程（Team Workflow）模块一起使用 用户定义的测试用例——无论是手动编写的测试还是使用测试用例向导创建的测试用例
接口测试	为自动化单元测试执行，使用功能桩函数替换用户接口
故障注入测试	使用功能桩函数来增强故障条件 使用不同的前置条件集（例如 min、max 以及试探值）以自动化进行单元测试用例生成

值得指出的是 C++test 允许将测试用例进行打包以便更加方便地对测试套件进行管理（例如仅执行某一类测试用例）。

9.4.3

该条标准定义了创建测试用例时应该使用的方法。

需要的方法	C++test 验证方法
需求分析	为某一给定条件的特定原子用例（atomic cases）定义测试用例——例如通过： • 手动编写的测试用例 • 使用测试用例向导创建的测试用例 • 提供功能桩函数以按照给定的用例的需求来驱动控制流 • 使用数据源为功能性等效测试提供多个输入/输出的集合
等效类的生成以及分析	使用工厂函数为自动化单元测试生成准备输入参数值的集合 使用数据源在测试中高效地使用广泛的输入值
边界值分析	自动生成的测试用例——可能使用： • 试探值 • 边界值 使用数据源在测试中高效地使用广泛的输入值
错误猜测	使用功能桩函数机制向经测试过的代码中注入故障条件 使用 Bug Detective 查找的结果作为编写会发现错误条件的测试用例的启发

9.4.4

该条标准定义了评估测试用例的完整性的方法。

需要的方法	C++test 验证方法
语句覆盖率	覆盖率模块
分支覆盖率	覆盖率模块
MC/DC（修改条件/决定性条件）	覆盖率模块

值得注意的是 ISO/DIS 26262 第六部分中的第 9.4.4 条指出如果使用插装代码来决定覆盖率的程度，可能需要证明这样的插装对于测试结果不会造成影响。这可以通过在 C++test 中同时运行插装以及未插装的测试来确定。

9.4.5

该条标准定义了测试环境的需求。

需要的方法	C++test 验证方法
单元测试的测测试环境应该尽量与目标环境相符合。	目标设备与仿真器中所进行的单元测试执行用以测试不同的测试环境，例如软件在环（software-in-the-loop）、处理器在环（processor-in-the-loop）或者硬件在环（hardware-in-the-loop）

10 软件集成以及测试

本章节描述了软件组件集成以及验证相应生成的嵌入式软件的流程。

10.4.2

该条标准描述了关于执行软件集成测试的一般性信息。

需要的方法	C++test 验证方法
集成测试	带运行时错误检测的应用程序监测 显示应用程监测结果的报告模块

10.4.5

该条标准定义了评估集成测试完整程度的方法。

需要的方法	C++test 确认方法
功能覆盖率	覆盖率模块

10.4.7

该条标准定义了集成测试环境的需求。

需要的方法	C++test 验证方法
软件集成测试的测试环境应该尽量与目标环境相符合。	目标设备与仿真器中所进行的单元测试执行用以测试不同的测试环境，例如软件在环（software-in-the-loop）、处理器在环（processor-in-the-loop）或者硬件在环（hardware-in-the-loop）

总结

Parasoft 的 C++test 能帮助汽车软件开发团队实现遵守 ISO/DIS 26262 标准并满足嵌入式程序的标准作业程序（SOP）。包括编码标准一致性分析、数据以及控制流分析、单元测试、应用程序监测、工作流程组件以及自动化同行代码审查过程在内的广泛的分析手段——与包含高级别细节的可配置的测试报告能显著地促进软件验证过程中所需进行的工作。

关于 **Parasoft**

在过去 20 年中，Parasoft 一直致力于研究应用程序中软件错误的原因及表现。我们的方案将我们的研究成果嵌入到提升质量的可持续性软件开发过程中。这能带来更好的编码基础，更加可靠的功能单元以及鲁棒性更好的业务流程。无论您是在设计面向服务架构（SOA），升级既有系统或是改进质量流程——您都可以使用我们专业且屡获殊荣的产品来提高您的开发效率和应用程序的质量。如需更多信息，请访问：http://www.parasoft.com。

联系 **Parasoft**

全球总部
101 E. Huntington Drive,
2nd Floor Monrovia, CA 91016
USA
Toll Free: (888) 305-0041
Tel: (626) 305-0041
Fax: (626) 305-3036
Email: info@parasoft-embedded.com
URL: http://www.parasoft-embedded.com
URL: http://www.parasoft.com

欧洲总部
Parasoft SA
Kielkowskiego 9, Krakow 30-703
Poland
Phone +48 12 290 91 01
Fax +48 12 290 91 02
Email: info-pl@parasoft.com

中国总部
R m. 902, Golden Magnolia Plaza, W. 1 Dapu Road
Huangpu District
Shanghai
Phone (021) 6093-2819
Email: info-china@parasoft.com

Parasoft SA (法国)
Chateau de Sainte Assise, 77240 Seine Port
France
Phone (33 1) 64 89 26 00

Fax (33 1) 64 89 26 10
Email: sales@parasoft-fr.com

Parasoft 德国公司
Mähringer Weg 45
89075 Ulm
Germany
Phone +49 89 4613323-0
Fax +49 89 4613323-23
Email: info-de@parasoft.com

Parasoft 英国公司
The Lansdowne Building
2 Lansdowne Road
Croydon.
CR9 2ER
United Kingdom
Phone +44 (0)208 263 6005
Fax +44 (0)208 263 6100
Email: sales@parasoft-uk.com

其它地区
请浏览 http://www.parasoft.com/contacts