使用 Parasoft C++test 以满足 SIL 需求

提升电气/电子/可编程电子（E/E/PE）相关系统的功能性安全

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。

安全性将是今后电气/电子/可编程电子安全性相关系统中的一个主要课题。新功能越来越多地触及到了安全性工程的范畴。每个用以限制风险在可接受范围内的功能都可被称作*安全功能*。为了实现安全功能，这些功能需要满足安全功能的需求（安全功能的作用）以及安全完整性需求（该功能以令人满意的方式运行的可能性）。未来对包含安全功能的开发以及集成将更进一步地提升对安全系统开发过程以及提供满足所有合理性安全对象依据的需求。

随着复杂性、软件的功能以及硬件层面实施日益增加的趋势，系统化的失效以及偶发硬件失效的风险将越来越多。通过提供可行的需求以及流程，IEC 61508 中包含了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 Parasoft C++test 的使用将如何帮助软件开发团队满足特定 SIL 级别的需求。本文首先介绍了 IEC 61508 标准中所定义的 SIL 的概念。其次将介绍 Parasoft C++test 是为汽车领域软件开发以及测试提供最佳实践的集成解决方案。最后将阐述 Parasoft C++test 如何满足或部分满足特定 SIL 需求的软件开发。

软件完整性级别

IEC 61508 标准中所定义的安全完整性级别（SIL）是用以度量给定安全功能目标发生危险失效的可能性的指标（SIL1-SIL4）。每个安全性相关系统中的安全功能都需要指定相应的软件完整性级别。一般而言，某个特定的 E/E/PE 安全性相关系统中都会实现多个安全功能。如果这些安全功能的安全完整性需求存在着不同，除非它们之间的实现存在着充分的独立性，否则，这些安全功能中最高的安全完整性级别将适用到整个 E/E/PE 安全性相关系统中。

根据 IEC 61508 标准，安全完整性级别是根据该设计功能对于某种需求（对于低级别需求模式操作而言）执行失败或每小时发生危险失效（对于高级别需求或持续模式操作而言）的平均概率而进行评估的。

IEC 61508 标准中列出了为实现各个安全完整性级别的需求。随着安全完整性级别的提升，这些需求将变得越来越严苛，从而进一步降低危险的失效发生的可能性。

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：

[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。

- **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。

通过一组实用的 GUI 小工具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配置、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。

使用 C++test 满足 SIL 需求

下述表格描述了 Parasoft C++test 是如何支持实现特定 SIL 安全功能的软件开发生命周期的。请注意，此处所提供的信息只是用来对 C++test 在 SIL 相关的确认与认证过程中的一个简要介绍。如需澄清 IEC 61508 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专家进行咨询。如果关于如何在 IEC 61508 确认与认证过程中使用 C++test 你还有其它的问题，请联系你的 Parasoft 代表。

下述符号在以下表格中用以标记：

- R – 与 IEC 61508 标准中所推荐的功能相匹配的方法
- HR – 与 IEC 61508 标准中所强烈推荐的功能相匹配的方法

C++test 功能性描述中参考了 IEC 61508-3，附录 A 中的相应的技术或指标，例如（表 A.3：1）参考了 IEC 61508-3，表 A.3 中的技术 1。

编码标准一致性——静态代码分析

C++test 功能	SIL 1	SIL 2	SIL 3	SIL 4
编码标准一致性模块——通用				
将 C++test 用作 C 编程语言的静态分析工具（表 A.3:1）	HR	HR	HR	HR
增强特定编码标准（表 A.4:5）	R	HR	HR	HR
使用静态分析（表 A.9:3）	R	HR	HR	HR
分析类型				
使用代码度量指标（例如函数大小、函数参数计数等）来增强结构化编程（表 A.4:6）	HR	HR	HR	HR
增强业界认知的编码标准规则集，诸如 MISRA C/C++、JSF、HIS 源码度量指标等（表 B.1:1）	HR	HR	HR	HR
增强特定编码习惯（表 B.1:1）	HR	HR	HR	HR
增强特定格式习惯（表 B.1:1）	HR	HR	HR	HR
使用代码度量指标（例如圈复杂度、要点复杂度等）以增强低复杂度代码的策略（表 A.9,5，表 A.10:3）	R	R	R	R
使用编码标准以避免常见失效（表 A.10:5）		R	HR	HR
使用编码标准以增强仅使用语言某一子集的策略，例如避免不安全的结构（表 A.3:3）			HR	HR
特定编码标准指南				
查找函数的多个退出点（表 B.9:4）	HR	HR	HR	HR
查找隐式类型转换以增强强类型化策略（表 A.3:2）	HR	HR	HR	HR

C++test 功能	SIL			
	1	2	3	4
使用源码度量指标来降低软件模块大小（表 B.9:1）	HR	HR	HR	HR
查找无条件跳转（表 B.1:7）	R	HR	HR	HR
报告动态对象的不安全使用（表 B.1:2）	R	HR	HR	HR
增强信息隐藏/封装的策略（表 B.9:2）	R	HR	HR	HR
增强保守性实现技术的策略——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（表 A.4:3）		R	HR	HR
报告动态变量的不安全使用（表 B.1:3a）		R	HR	HR
报告递归函数（表 B.1:6）		R	HR	HR
报告指针的不安全使用（表 B1:5）		R	HR	HR
增强使用失效断言的编程策略（表 A.2:3a）	R	R	R	HR
使用源码度量指标来限制函数的参数数量（表 B.9:3）	R	R	R	R

值得指出的是，SIL3 以及 SIL4 不推荐在 C 编程中不使用特定的编码标准以及静态分析工具。

Bug Detective – 静态数据以及执行流分析

C++test 功能	SIL			
	1	2	3	4
Bug Detective——通用				
使用数据流分析（表 B.8:4）	R	HR	HR	HR
使用控制流分析（表 B.8:3）	R	HR	HR	HR
为数据流图标中的变量使用抽象的可能值对源代码进行分析（表 B.8:8）	R	R	HR	HR
Bug Detective——特定规则示例				

C++test 功能	SIL			
	1	2	3	4
报告错误的指针使用（表 B.1:5）		R	HR	HR

自动同行代码审查

C++test 功能	SIL			
	1	2	3	4
自动同行代码审查模块——通用				
使用同行代码审查模块对源码进行推演（表 B.8:9）	HR	HR	HR	HR

单元测试

C++test 功能	SIL			
	1	2	3	4
单元测试模块——通用				
单元测试执行（表 A.5:4，表 A.7:3）	HR	HR	HR	HR
自动单元测试生成模块				
使用边界值进行自动单元测试生成（表 B.2:1，表 B.3:3）	R	HR	HR	HR
使用工厂函数为自动单元测试生成准备输入参数值（表 B.2:5）	R	R	R	HR
使用随机输入组合进行自动单元测试生成（表 A.5:1）		R	R	HR
测试管理模块				
使用用户定义的测试用例——无论手动编写的还是使用测试用例向导创建的——来测试某个给定需求的特定原子用例（表 A.5:4，表 A.7:3）	HR	HR	HR	HR
使用数据源为特定需求的功能等效原子用例有效地提供多重输入（表 A.5:4，表 A.7:3）	HR	HR	HR	HR
使用测试用例浏览器对测试用例以及审查测试用例状态进行管理（表 A.5:2）	R	HR	HR	HR

C++test 功能	SIL			
	1	2	3	4
功能桩函数				
使用桩函数按照给定需求中的设置对已执行的测试套件流程进行控制（表 A.5:4）	HR	HR	HR	HR
使用功能桩函数为自动单元测试执行替换用户接口（表 A.5:6）	R	R	HR	HR
使用桩函数在测试中提供故障条件（表 B.2:2）	R	R	R	R
覆盖率模块				
为结构测试进行语句、分支以及 MC/DC 代码覆盖率分析（表 B.2:6）	R	R	HR	HR

注意，C++test 可以：

- 在插装或非插装模式下运行单元测试——例如显示覆盖率插装不会影响测试结果。
- 在生产环境中与目标设备或仿真器上执行单元测试。

应用程序监测

C++test 功能	SIL			
	1	2	3	4
应用程序监测模块——通用				
对正在运行的应用程序进行监测，并报告检测到的运行时问题（表 A.9:4）	R	HR	HR	HR
覆盖率模块				
为结构测试进行语句、分支以及 MC/DC 代码覆盖率分析（表 B.2:6）	R	R	HR	HR

值得指出的是：

- C++test 可以在生产环境中的目标设备或仿真器上监测应用程序的运行。

总结

Parasoft C++test 能帮助软件开发团队完全或部分满足软件开发过程中 IEC 61508 文档中软件完整性级别的需求。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。无论你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——你都可以使用我们专业且屡获殊荣的产品来提高你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