汽车电子应用基于风险的测试

—— 规避软件崩溃的 6 大要素

汽车嵌入式软件开发是一项特别的业务。在这复杂的环境下，制定一个基于风险的测试策略有十分重大的意义。凭借在软件和系统工程项目质量管理方面，积累了十多年的经验，尤其是汽车行业方面的经验，Squoring Technologies 公司就如何建立并领导一个高效合理的风险测试方案，提出了 6 大关键要素。

如今，如果量产车没有安全气囊系统、巡航控制系统、显示距离下次技术检修还有多长时间的嵌入式汽车诊断系统、停车辅助系统，或者是仪表盘上没有时钟，会是什么样的情况？如果高档车没有集成 GPS 系统、蓝牙连接系统、智能前照灯，以及智能雨刷，靠什么在市场中获得成功呢？

软件，汽车的新引擎

在过去的 30 年里，为了保障乘客的安全，增强功能的可靠性，提高汽车的性能和舒适度，嵌入式电子产品在汽车市场逐渐发展起来。在未来的几年里，这一发展趋势还会更加迅速，更加电动、更加自动化的互联汽车将会大规模地发展。

随着汽车行业的大力发展，软件已经是一个主要的横向组件，成了防止意外和故障风险不可或缺的元素。以致现在汽车系统中的代码行的数量已经超过了一些应用在航空工业中的软件的代码行数。实际上，现在的汽车模型平均需要嵌入 1000 万到 2000 万代码行（通常用 C 语言，C++或汇编语言编写），而 F22 战斗机仅使用了 170 万行代码，波音 787 也只用了 650 万行代码而已。据 Frost 和 Sullivan 估计，在不久的将来，代码行数还可能增加到现在的 10 倍。

Oliver Wyman 在一次研究中预测：一年之内，一辆汽车 40%的价值将依赖于软件组件，到 2025 年上路的互联汽车将不少于一亿辆。（Ernst 和 Young）换句话说，这就是未来。所以现在软件出现故障和硬件出现故障同样危险。

专用于汽车行业特定的软件

由于受到复杂的限制，所以汽车行业的软件开发是一项特别的，甚至是一项独一无二的业务。这些限制有：

- **高分布容量：**汽车是现在使用最为广泛的交通工具。嵌入式软件在逻辑上存在很多潜在的异常和故障。
- **要确保高可靠性：**因为车辆的核心功能（刹车系统、安全气囊等）关系到用户的安全，所以制造商一定要确保软件的可靠性和稳定性。任何的缺陷都可能严重损害品牌声誉，并且大大增加修复的成本，还会影响该型号的汽车的销售业绩。Toyota 就遭遇了这些问题。（参见下文的实例）
- **用户无法掌控系统：**航空公司的飞行员可以切换到手动模式，而且他们还能够掌控飞机上的其它器械。但是驾驶员就只会开车而已，对于其它方面可能就不太了解。所以，实

- 际上，开发人员是唯一能操控软件的人。所以，一定要在源头上规避风险。
- **市场竞争激烈：**虽然行业趋于集中，但是随着汽车市场的开放，越来越多的公司和品牌进入市场，这又有力地促进了竞争。所以，在汽车生产过程中，由于软件方面的成本不断增加，我们一定要优化开发步骤，以取得竞争优势。
- **上市时间：**制造商必须能够尽快将新发展的技术，尤其是互联方面的技术，应用到他们的系统中。苹果公司在去年的日内瓦车展上公布的“Carplay”，让大家看到了未来将可以怎样提高驾驶的舒适度。
- **交付不够灵活：**在软件交付速度方面，汽车行业的软件交付速度与移动设备行业的产品交付更为相似，而不是航空航天业。虽然可能汽车还没有完全整合好，但是要修改已经交付的软件部件还是很困难，而且代价很大。到目前为止，比起召回成百上千的车辆所花的费用，测试相关的成本还是比较低的。

基于风险的测试：影响竞争力的重要因素

我们都知道，汽车软件的开发和测试非常复杂。实际上，为了保持竞争力，同时还不影响系统的可靠性，我们需要平衡好产品的验证工作，以达到“足够好”的目标。也就是说，要用合理的成本，在合理的时间范围内，使产品的可靠性、稳定性、性能和安全性达到可接受的水平。

如果没有合理的风险评估，是无法达到上面所讲的平衡的。Automotive SPICE 标准也认为风险评估是核心实践方法。此外，ISO 26262 也要求进行风险评估，并且定义了针对汽车行业的风险评估方法，不过分析风险的时候，可适当灵活处理。

因此，必须制定一个基于风险的测试策略。这种测试方法，可以定义对各种不同风险进行测试的优先级,执行并控制软件组件的测试。不同种类的风险有:被测功能的功能性危险程度，特定环境和项目的局限性，代码的复杂性和稳定性，与开发团队的经验和专业技能相关的风险，等等。

凭借在软件和系统工程项目质量管理方面积累了十多年的经验，尤其是在汽车行业的经验，Squoring Technologies 公司就如何建立并领导一个高效合理的基于风险的测试方案，提出了 6 大关键要素。

一. 定义待测组件的优先级

确定待测软件组件的优先级是一项核心工作，包括：以使风险最小化为目标，安排测试和验证组件的顺序。如果功能性危险程度（缺陷的影响，以对人身和材料损害风险的等级作为衡量指标，见 ASIL – 汽车功能完整度等级）是定义第一优先级的显著因素，那么就只需要考虑到一小部分定义好的组件即可。

对于其它组件（通常情况下，80%-90%的组件不是高安全的），可根据以下信息定义优先级：

- **组件的历史情况：**如果某个组件已经经过验证，并且在过去的几个版本里并没有进行更新（修正或演进），那么测试该组件就不需要重点测试。因为从某种意义上讲，该组件已经经受了时间的检验。相反地，如果某个组件在之前的发布版本中，反复出现问题，那么就需要对该组件进行重点测试。
- **代码复杂度：** CNES 研究发现，如果源代码的复杂度是原来的两倍，那么代码出错的

概率就会是原来的 4 倍。在危险程度相同的情况下，代码比较复杂的组件要优先测试，以便确保所有的缺陷都能被发现。在此基础上，再使用专门为嵌入式汽车软件设计的度量和 HIS 复杂度阈值进行评估。

- **每个测试阶段所分配的时间：**可以根据距离产品交付所剩的时间来分配，危险程度相同的组件所分配的时间应该差不多。

二. 打破信息孤岛

从逻辑上讲，要 “定义待测组件的优先级”，首先必须了解那些能够帮助定义这些优先级的元素（组件的历史情况，复杂度，延期情况，等等）。因此，除了已有技术资料的数据之外，还需要收集其它数据。

如果在验证过程中，能够进行自动化的多源数据收集，就可以估量并具体化待测组件存在的风险，从而有助于减少风险。

下面列举的几个数据提供程序，可帮助监控测试活动：

- 需求管理：各个功能的 ASIL 关键度，各种规则，等
- 配置管理：组件和源代码的历史情况，开发人员的资质，等
- 源代码分析：复杂度度量，是否符合 MISRA 规范，等
- 问题追踪和标签管理：某个指定的函数存在多少未解决问题，反复出现的代码问题，等
- 项目、资源和计划管理：距离交付所剩的时间和尚未完成的任务，已有的开发资源，等

当然，上面所列的信息并不全面：您可以加入任何其它有用的信息来丰富您的指标，以便更清楚地了解组件及其生态系统的质量。

三. 为测试设限

一个有效的测试策略需要考虑到花费和截止日期的问题，所以测试中一定要加入停止准则。可依据以下两点定义测试需求：

- **根据以下规则，通过管理“代码覆盖率大门”，限定所要达到的代码覆盖率：**
  - 优先级较高的功能：代码覆盖率要达到 100%
  - 优先级中等的功能：代码覆盖率要在 60%到 80%之间
  - 优先级较低的功能：代码覆盖率要在 30%到 60%之间
- **限定所要进行的测试的性质：**
  - MC/DC 覆盖（修正的条件/判定覆盖）：优先级非常高的高安全功能必须进行 MC/DC 覆盖测试。这种测试方法不仅是最耗精力的，也是最昂贵的。
  - 分支覆盖：适用于优先级较高的功能。总的来说，分支覆盖的花费是 MCDC 的一半。
  - 语句覆盖：大多数情况下用于优先级较低的功能。这种测试方法的花费只是 MCDC 的 1/4。

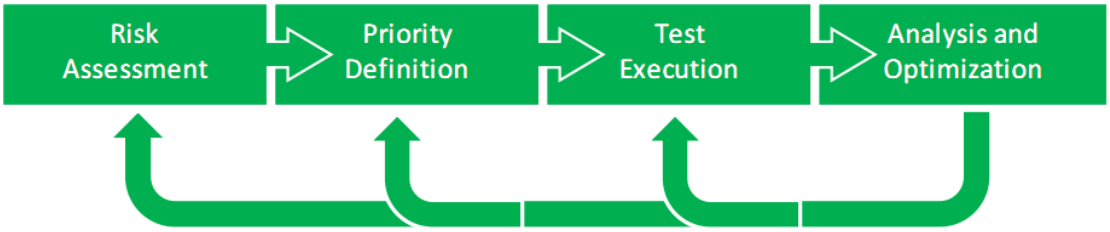
综上所述，我们可以通过设定限制条件，避免掉进“过度”测试的陷阱。根据事先定义好的优先级来进行测试，就能够用非常合理的方法减少可能存在的风险。

四. 在重复中不断优化测试策略

为了实现“足够好”的目标，我们常常需要优化测试策略。在测试过程中，我们可以通过调整以下元素，来优化测试策略。现在就是通过实际操作检验测试策略的时候了：

- 修改风险评估：修改数据收集的范围。通过减少信息量来降低干扰，或者通过增加信息量来提升风险分析功能，并增强测试监控的可靠性。
- 修改优先级：待测组件的优先级，可能需要根据新收集到的数据进行修改。
- 调整测试的性质：如果测试组件的优先级被重新定义了，那么可能也需要根据新的优先级，来调整各个组件的测试性质（MCDC，分支覆盖，或语句覆盖）。

重复测试，以及测试过程中根据实际情况对测试策略所作的修改，将会不断优化组件验证流程，从而避免“爆炸性”后果以及潜在的危险。



五. 报告信息

测试相关指标的报告的质量对测试策略能否取得成功有很大的影响。性能指标交付能够更好地对正在进行的测试活动进行解释。因为通过性能指标的交付情况，我们可以客观地了解产品的概况，所以我们就能够根据实际情况对工作进行合理调整，也可以在进入下一阶段之前，确定到底要不要继续推进。

为了使这类指标能够简化验收工作，并保障验收工作的顺利进行，一定要满足以下要求：

- 所取得的成绩，达到了可以继续“推进”的标准
- 能够清楚解释“不推进”的原因
- 团队要能够了解并完成相关任务，以达到可以继续“推进”的标准。

最后，为了保证测试策略能够被理解和接受，一定要执行流程中的每一步，并且根据实际情况对每一步进行适当调整（模型和指标的原型，部署，投资回报率的调整……）。根据方法的成熟度，以及所得到的结果等实际情况，将信息传达给相关人员：

- 开发和验证团队：内部或外包供应商（第三方应用程序的维护/验收）
- 质检部门：品质经理，方法和工具的管理者
- IT 主管和中层管理人员
  董事会和公司内部的相关部门

六. 切勿重蹈覆辙

虽然制造商之间已经形成标准化的软件架构模块（尤其是汽车开放系统架构[AutoSAR]方面），而且在开发最佳实践方法和流程方面也有规律可遵循，但是汽车行业的软件开发还是件很复杂的事情。

因为汽车行业对软件的可靠性要求比较高，所以这个领域的标准和最佳实践框架，在软件行业中是最成熟，文献最齐全的。

评估开发流程：

- Automotive SPICE：改编自 ISO/CEI 15504，适用于汽车行业。该标准提供了一个对开发和验证流程进行评估的框架。由 VDA 发行的《Automotive SPICE 流程评估模型》（“Automotive SPICE Process Assessment Model”）提供了详细的评估步骤，丰富了评估方法。
- ISO-26262：发行于 2011，该标准为确保道路车辆的电气电子系统的性能安全提供了框架和应用模型。其中有一个完整章节专门讲述了软件开发问题。

评估代码质量：

- MISRA 标准：首次发布于 1998 年，之后由 MISRA（汽车工业软件可靠性协会）定期更新。这两个标准规定了 C/C++编程规范，以便在开发阶段，防止和减少与软件执行相关的风险。
- HIS 度量：由奥迪（Audi）、宝马（BMW）、戴姆勒（Daimler）、保时捷（Porsche）、大众（Volkswagen）发起，HIS 框架实际上定义了代码复杂度的门槛（圈复杂度，GOTO 的数量，稳定性指标……），以适用于主要的汽车系统。这是软件质量方面现成的、最准确、最实用的文档之一。

代码质量：日本丰田汽车的案例

美国（俄克拉荷马法院）2013 年 10 月对日本丰田汽车的判决，倡导汽车行业提高软件质量标准。由于车辆质量问题引发车祸，造成了一人死亡，该日本汽车公司被判处三百万美元的赔偿金。
该车祸源于一次意外加速，据推测，这是由于 ECM（引擎控制模块，控制引擎的组件）存在软件故障而引起的。

Barr Group 进行了专业技术调查，全面检查了故障系统 —— 记录显示该制造商生产的车辆发生了 700 多起类似情况 —— 评估结果清楚表明，软件未遵循 MISRA，HIS 和 ISO 26262 标准：

- 故障安全系统的设计不良，尤其是主要安全功能
- 源代码特别复杂，比较容易引发问题：有 67 个功能的代码超过了圈复杂度的阈值（50）
- 油门系统（ETCS，丰田电子油门系统）的代码质量被评定为“不合格”
- 发现了 80000 多条违反 MISRA-C 规则的行为
- 未进行同行评审，或同行评审工作做得不充分
- 没有工具追踪软件缺陷

使用合适的工具来完成测试工作

在测试初期，使用数据表格来确认测试验证模型的可行性是完全可以接受的。但是，如果没有合适的工具，在实际情况下（如：模块或应用程序的完整信息），是无法工业化地部署并实现指标的。

下面所列的方法可以指导您如何选择合适的工具：

- 从源代码分析工具中收集数据（软件构件的详细目录，HIS 度量计算法，MISRA-C 和 MISRA-C++编程规则管理，等）
- 将代码构件整合为软件组件
- 整合软件配置管理工具的数据，以便发现代码的变化，从而根据实际情况调整测试工作
- 通常情况下，需要收集、合并、整理与组件相关的各种数据（需求管理工具，配置，问题追踪软件，等）
- 设计和显示基于角色的仪表盘（专属于软件测试人员，项目经理，开发人员，IT 经理，QA 经理，等）

在所有情况下，所选的工具都必须支持正在使用的验证方法，并能够管理测试活动。

SQUORE Automotive 是汽车行业用于管理主要嵌入式软件项目的解决方案

因为汽车行业的所有参与者都需要解决质量和功能安全质量方面的战略性问题，因此 Squoring Technologies 公司特意创建了一个特殊版本的 Squore 仪表盘：Squore Automotive 采用 Automotive SPICE 的基本惯例，以优化测试策略，并证明代码符合汽车工业的质量要求（ISO 26262， HIS，MISRA……）。

该方案完全符合汽车工业的需求。在这一领域，Squoring Technologies 已经服务了很多被引荐的客户。

为了使基于风险的测试策略取得成功，SQUORE Automotive 将所有主要的因素都集中了到一个工具中。SQUORE Automotive 是一套完整、可靠、高效的解决方案，可以使组件质量达到要求，而且还能优化成本、缩短交付时间。

优化基于风险的测试策略的创新功能

为了使基于风险的测试策略取得成功，有很多功能是必须满足的，而 SQURE 依据前面详细介绍过的 6 大要素，将这些功能都集中到了一个特殊的工具中。这 6 大要素为：

1. **定义优先级**
   根据决策标准定义的基于风险的测试策略，适用于各个阶段：单元测试，集成测试，回归测试
2. **打破信息孤岛**
   - 综合性的高效源代码分析器，适用于 C,C++, Ada, C#, Java
   - 从已在使用的第三方工具中导入数据的插件：Klocwork, QA-C, Coverity, Test RealTime, Polyspace, Logiscope, Tessy, PC-lint…
3. **为测试设限**
   从能够立刻发现危险组件的深入分析，到测试最基本的函数或类函数
4. **在重复中不断优化测试策略**
   - 根据性能和趋势指标，不断改善验证过程
   - 根据测试策略的演进，不断缩放并重新配置分析模型
5. **报告信息**
   - 通过性能指标和趋势分析，全面了解开发流程：立刻发现违背和偏离计划的行为

— 通过集中管理不符合要求的数据，自动发出警报通知，以及共享“待办事项”列表，来增强团队合作。

6. **切勿重蹈**

使用标准中“现成的”标准化控制点：HIS 复杂度度量，MISRA 编码规则，代码重复，稳定性指标。

通过以下方式快速获得投资回报：

- 尽早发现缺陷，从而提高可靠性
- 自动执行 ISO 26262 标准中规定的验证方法
- 证明交付物符合质量要求
- 提高汽车制造商和供应商之间的信任度
- 通过监控技术债（Technical Debt）降低维护成本
- 采用 Automotive SPICE 介绍的基本实践方法