ICS 43.040

CCS T 35

中华人民共和国国家标准

GB/T 34590.6-XXXX

代替 GB/T 34590.6-2017

道路车辆 功能安全

第6部分：产品开发：软件层面

Road vehicles — Functional safety —

Part 6: Product development at the software level

（ISO 26262-6:2018，MOD）

（征求意见稿）

（本草案完成时间：2021年4月1日）

在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上。

XXXX-XX-XX 发布　　XXXX-XX-XX 实施

国家市场监督管理总局
国家标准化管理委员会　发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

GB/T 34590-XXXX《道路车辆 功能安全》分为以下部分：

——第1部分：术语；

——第2部分：功能安全管理；

——第3部分：概念阶段；

——第4部分：产品开发：系统层面；

——第5部分：产品开发：硬件层面；

——第6部分：产品开发：软件层面；

——第7部分：生产、运行、服务和报废；

——第8部分：支持过程；

——第9部分：以汽车安全完整性等级为导向和以安全为导向的分析；

——第10部分：指南；

——第11部分：半导体应用指南；

——第12部分：摩托车的适用性。

本文件为GB/T 34590-XXXX的第6部分。

本文件代替GB/T 34590.6-2017《道路车辆 功能安全 第6部分：产品开发：软件层面》，与GB/T 34590.6-2017相比，除结构调整和编辑性改动外，主要技术变化如下：

——修改了标准适用范围，由“量产乘用车”扩大到“除轻便摩托车外的量产道路车辆”；

——新增了对商用车辆的相关要求和示例、对摩托车的适应性要求等；

——新增了基于模型的开发方法下开展软件安全要求定义，软件架构设计，软件单元设计和实现，软件架构设计的开发，软件单元的设计和实现，软件组件的设计、实现和集成，软件验证安全活动的指导；

——新增了软件架构层级安全分析及相关失效分析的应用指导（见附录E）；

——新增了软件开发环境文档的要求（见5.5.1）；

——新增了嵌入式软件测试用例的得出方法的要求（见11.4.3）；

——删除了启动软件层面产品开发的要求（见2017版第5章）；

——删除了软件架构层面的错误探测机制（见2017版的7.4.14表4）；

——删除了软件架构层面的错误处理机制（见2017版的7.4.15表5）；

本文件使用重新起草法修改采用了ISO 26262-6：2018 《道路车辆 功能安全 第6部分：产品开发：系统层面》。

本文件与ISO 26262-6：2018的技术性差异及其原因如下：

——关于规范性引用文件，本文件做了具有技术性差异的调整，以适应我国的技术条件，调整的情况集中反映在第2章“规范性引用文件”中，具体调整如下：

用修改采用国际标准的GB/T 34590.1-XXXX代替ISO 26262-1：2018；

引　言

ISO 26262是以IEC 61508为基础，为满足道路车辆上电气/电子系统的特定需求而编写。

GB/T 34590修改采用ISO 26262，适用于道路车辆上由电子、电气和软件组件组成的安全相关系统在安全生命周期内的所有活动。

安全是道路车辆开发的关键问题之一。汽车功能的开发和集成强化了对功能安全的需求，以及对提供证据证明满足功能安全目标的需求。

随着技术日益复杂、软件和机电一体化应用不断增加，来自系统性失效和随机硬件失效的风险逐渐增加，这些都在功能安全的考虑范畴之内。GB/T 34590通过提供适当的要求和流程来降低风险。

为了实现功能安全，GB/T 34590-XXXX（所有部分）：

a) 提供了一个汽车安全生命周期（开发、生产、运行、服务、报废）的参考，并支持在这些生命周期阶段内对执行的活动进行剪裁；

b) 提供了一种汽车特定的基于风险的分析方法，以确定汽车安全完整性等级（ASIL）；

c) 使用 ASIL 等级来定义 GB/T 34590 中适用的要求，以避免不合理的残余风险；

d) 提出了对于功能安全管理、设计、实现、验证、确认和认可措施的要求；及

e) 提出了客户与供应商之间关系的要求。

GB/T 34590针对的是电气/电子系统的功能安全,通过安全措施(包括安全机制)来实现。它也提供了一个框架，在该框架内可考虑基于其它技术（例如，机械、液压、气压）的安全相关系统。

功能安全的实现受开发过程（例如，包括需求规范、设计、实现、集成、验证、确认和配置）、生产过程、服务过程和管理过程的影响。

安全问题与常规的以功能为导向和以质量为导向的活动及工作成果相互关联。GB/T 34590涉及与安全相关的开发活动和工作成果。

图1为GB/T 34590的整体架构。GB/T 34590基于V模型为产品开发的不同阶段提供参考过程模型：

———阴影”V”表示 GB/T 34590.3-XXXX、GB/T 34590.4-XXXX、GB/T 34590.5-XXXX、GB/T 34590.6-XXXX、GB/T 34590.7-XXXX 之间的相互关系；

———对于摩托车：

GB/T 34590.12-XXXX的第8章支持GB/T 34590.3-XXXX；
GB/T 34590.12-XXXX的第9章和第10章支持GB/T 34590.4-XXXX。

———以“m-n”方式表示的具体章条中，“m”代表特定部分的编号，“n”代表该部分章的编号。

示例：“2-6”代表GB/T 34590.2-XXXX的第6章。

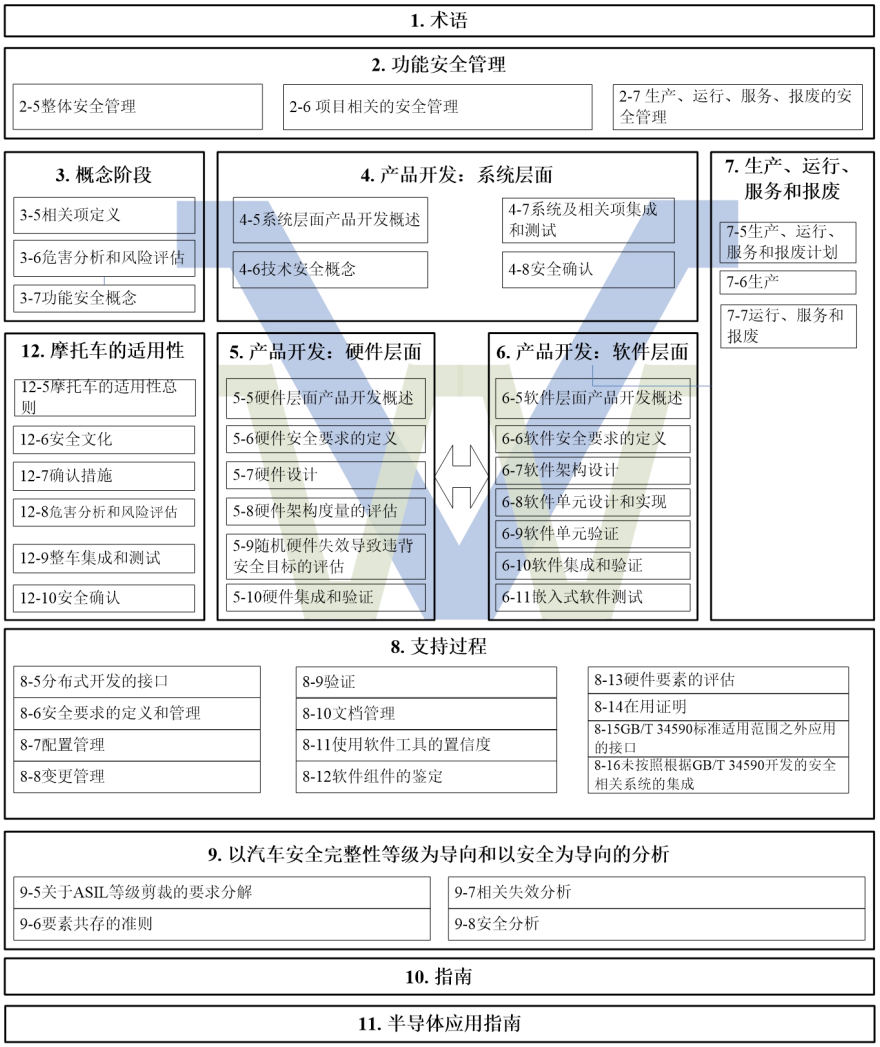


图 1　GB/T 34590-XXXX 概览

道路车辆 功能安全
第6部分：产品开发：软件层面

1 范围

GB/T 34590的本部分规定了车辆在软件层面产品开发的要求，包括：

——软件层面产品开发的概述；

——软件安全要求的定义；

——软件架构设计；

——软件单元设计和实现；

——软件单元验证；

——软件集成和验证；及

——嵌入式软件测试。

本文件规定了使用可配置软件的相关要求。

本文件适用于安装在除轻便摩托车外的量产道路车辆上的包含一个或多个电气/电子系统的与安全相关的系统。

本文件不适用于特殊用途车辆上特定的电气/电子系统，例如，为残疾驾驶者设计的车辆。

注：其他专用的安全标准可作为本文件的补充，反之亦然。

已经完成生产发布的系统及其组件或在本文件发布日期前正在开发的系统及其组件不适用于本文件。对于在本文件发布前完成生产发布的系统及其组件进行变更时，本文件基于这些变更对安全生命周期的活动进行裁剪。未按照本文件开发的系统与按照本文件开发的系统进行集成时，需要按照本文件进行安全生命周期的裁剪。

本文件针对由安全相关的电气/电子系统的功能异常表现而引起的可能的危害，包括这些系统相互作用而引起的可能的危害。本文件不针对与触电、火灾、烟雾、热、辐射、毒性、易燃性、反应性、腐蚀性、能量释放等相关的危害和类似的危害，除非危害是直接由安全相关的电气/电子系统的功能异常表现表现而引起的。

本文件提出了安全相关的电气/电子系统进行功能安全开发的框架，该框架旨在将功能安全活动整合到企业特定的开发框架中。本文件规定了为实现产品功能安全的技术开发要求，也规定了组织应具备相应功能安全能力的开发流程要求。

本文件不针对电气/电子系统的标称性能。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 34590.1-XXXX，道路车辆 功能安全 第1部分：术语(ISO 26262-1:2018，MOD)

GB/T 34590.2-XXXX，道路车辆 功能安全 第2部分：功能安全管理(ISO 26262-2:2018，MOD)

GB/T 34590.3-XXXX，道路车辆 功能安全 第3部分：概念阶段(ISO 26262-3:2018,MOD)

GB/T 34590.4-XXXX，道路车辆 功能安全 第4部分：产品开发：系统层面(ISO 26262-4:2018，MOD)

GB/T 34590.5-XXXX，道路车辆 功能安全 第5部分：产品开发：硬件层面(ISO 26262-5:2018，MOD)

GB/T 34590.7-XXXX，道路车辆 功能安全 第7部分：生产、运行、服务和报废(ISO 26262-7:2018,MOD)

GB/T 34590.8-XXXX，道路车辆 功能安全 第8部分：支持过程(ISO 26262-8:2018，MOD)

GB/T 34590.9-XXXX，道路车辆 功能安全 第9部分：以汽车安全完整性等级为导向和以安全为导向的分析(ISO 26262-9:2018，MOD)

3 术语、定义和缩略语

GB/T 34590.1-XXXX界定的术语、定义和缩略语适用于本文件。

4 要求

4.1 目的

本章规定了：

a) 如何符合 GB/T 34590-XXXX；

b) 如何解释 GB/T 34590-XXXX 中所使用的表格；及

c) 如何解释各章条基于不同的 ASIL 等级的适用性。

4.2 一般要求

如声明满足GB/T 34590-XXXX的要求时，应满足每一个要求，除非有下列情况之一：

a) 按照 GB/T XXXXX.2-XXXX 的要求，安全活动的剪裁已经实施并表明这些要求不适用；或

b) 不满足要求的理由存在且是可接受的，并且按照 GB/T XXXXX.2-XXXX 的要求对该理由进行了评估。

标有“注”或“示例”的信息仅用于辅助理解或阐明相关要求，不应作为要求本身且不具备完备性。

将安全活动的结果作为工作成果。应具备上一阶段工作成果作为“前提条件”的信息。如果章条的某些要求是依照ASIL定义的或可剪裁的，某些工作成果可不作为前提条件。

“支持信息”是可供参考的信息，但在某些情况下，GB/T 34590-XXXX不要求其作为上一阶段的工作成果，并且可以是由不同于负责功能安全活动的人员或组织等外部资源提供的信息。

4.3 表的诠释

本文件中的表是规范性或资料性取决于上下文。在满足相关要求时，表中列出的不同方法有助于置信度水平。表中的每个方法是：

a) 一个连续的条目（在最左侧列以顺序号标明，如 1、2、3）；或

b) 一个选择的条目（在最左侧列以数字后加字母标明，如 2a、2b、2c）。

对于连续的条目，高度推荐和推荐的方法按照ASIL等级推荐予以使用。高度推荐或推荐的方法允许用未列入表中的其它方法替代,此种情况下，应给出满足相关要求的理由。如果可以给出不选择所有条目也能符合相应要求的理由，则不需要对缺省方法做进一步解释。

对于选择性的条目，应按照指定的ASIL等级对这些方法进行适当的组合，而与这些方法在表中是否列出无关。如果所列出的方法对于一个ASIL等级来说具有不同的推荐等级，宜采用具有较高推荐等级的方法。应给出选择组合方法或选择单一方法满足相应要求的理由。

注：在表中所列出方法的理由是充分的。但是，这并不意味着有倾向性或对未列到表中的方法表示反对。

对于每种方法，应用相关方法的推荐等级取决于ASIL等级，分类如下：

——“++每表示对于指定的ASIL等级，高度推荐该方法；

——“+—对表示对于指定的ASIL等级，推荐该方法；

——“o—对表示对于指定的ASIL等级，不推荐也不反对该方法。

4.4 基于 ASIL 等级的要求和建议

若无其它说明，对于ASIL A、 B、 C和D等级，应满足每一章条的要求或建议。这些要求和建议参照安全目标的ASIL等级。如果在项目开发的早期对ASIL等级完成了分解，按照GB/T XXXXX-9第5章的要求，应遵循分解后的ASIL等级。

如果GB/T 34590-XXXX中ASIL等级在括号中给出，则对于该ASIL等级，相应的章条应被认为是推荐而非要求。这里的括号与ASIL等级分解无关。

4.5 摩托车的适用性

对于适用于GB/T XXXXX.12要求的摩托车的相关项或要素，GB/T XXXXX.12的要求替代本文件和GB/T XXXXX.2的相应要求。

4.6 卡车、客车、挂车和半挂车的适用性

对卡车、客车、挂车和半挂车的特殊规定以（T&B）来表示。

5 软件层面产品开发概述

5.1 目的

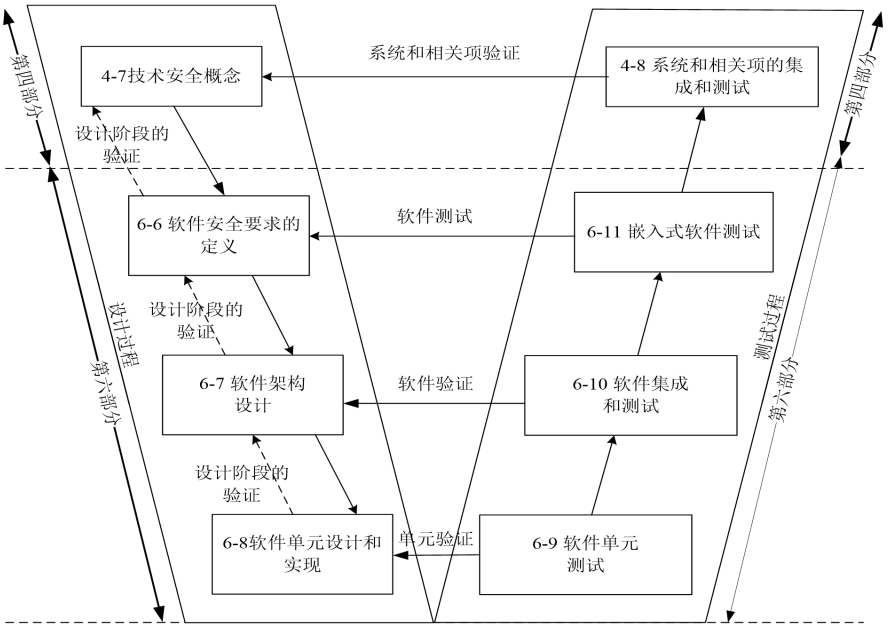
本章的目的是：

a) 确保合适且一致的软件开发流程；及

b) 确保合适的软件开发环境。

5.2 总则

图2给出了软件开发阶段的参考模型。附录C中提供了可配置软件处理的详细信息。



注：图中 GB/T 34590 每部分的特定章用以下方式标示：“m-n”，“m”代表部分号，“n”代表章号，例如，“4-7”代表 GB/T 34590.4-XXXX 的第 7 章。

图 2 软件层面产品开发阶段参考模型

注1：敏捷软件开发的方式和方法也可以适用于安全相关软件的开发，但是如果安全活动按照该方式剪裁，应该考虑GB/T 34590.2-XXXX，6.4.5。然而，敏捷方式和方法的使用不能省略安全措施或忽略实现功能安全所必需的基本文档、流程或严格的产品安全完整性要求。

示例1：由测试驱动的开发可用于提高要求的质量和可测试性。

示例2：基于自动化构建系统的持续集成可以支持各个子阶段的一致性并促进回归测试。这种构建系统通常执行代码生成、编译和链接、静态代码分析、文档生成、测试和打包。根据工具链和工具配置，该系统允许重复生成以及在变更后生成可比较的软件、文档和测试结果。

注2：开发特定相关项的嵌入式软件时，也可以考虑信息安全，见GB/T 34590.2-XXXX，5.4.2.3。为了能够开发软件，本章中讨论了关于要使用的建模、设计和/或编程语言以及指南和工具应用的特定主题。

注3：用于软件开发的工具，包括软件工具以外的工具。

示例3：测试阶段使用的工具。

5.3 本章的输入

5.3.1 前提条件

无。

5.3.2 支持信息

可考虑下列信息：

——经鉴定合格的软件工具（见 GB/T 34590.8-XXXX，第 11 章）；

——用于建模、设计和编程语言的设计和编码指南（来自外部）；

——方法应用的指南（来自外部）；及

——工具应用的指南（来自外部）。

5.4 要求和建议

5.4.1 在开发相关项的软件时，使用的软件开发过程和软件开发环境应：

a) 适用于开发安全相关的嵌入式软件，包括方法、指南、语言和工具；

b) 支持软件开发生命周期中的各跨子阶段和各自工作成果的一致性；及

c) 与系统和硬件开发阶段在所需的交互和信息交换的一致性方面兼容。

注1：相关项软件开发的阶段、任务和活动的排序，包括迭代步骤，目的是为了确保相应的工作成果与硬件层面的产品开发（本文件GB/T 34590.5）和系统层面的产品开发（本文件GB/T 34590.4）保持一致。

注2：软件工具准则评估报告（本文件GB/T 34590.8-XXXX，11.5.1）或软件工具的鉴定报告（本文件GB/T 34590.8-XXXX，11.5.2）可以为工具的使用提供输入。

5.4.2 当选择一种设计语言、建模语言或编程语言时，应考虑准则：

a) 明确易理解的定义；

示例：语法和语义的明确定义或对开发环境配置的限制。

b) 如果建模用于需求工程和管理，定义和管理安全要求（按照 GB/T 34590.2-XXXX 第 8 章）的适用性；

c) 支持模块化、抽象化和封装化的实现；及

d) 支持结构化构造的使用。

注：汇编语言能用于那些不适合使用高级编程语言的软件部分，如与硬件接口的底层软件、中断处理程序、或对时间敏感的算法。然而，使用汇编语言可能需要对所有软件开发阶段进行适当的应用或剪裁（例如，第 8 章的要求）。

5.4.3 考虑到表 1 中列出的主题，相应的指南或者开发环境应涵盖适合于建模、设计或者编程语言（见 5.4.2）的准则，并且这些准则并不完全由语言自身决定。

示例 1：MISRA C（见参考文献[3]）是编程 C 语言的编码指南，并包括自动生成代码的指南。

示例 2：在具有自动代码生成功能的基于模型的开发中，可以在模型层面以及代码层面中应用该准则。可以考虑适当的建模风格指南，包括 MISRA AC 系列。商业工具的风格指南也可作为可能的指南。

注：可以为特定的相关项开发修改现有的编码指南和建模指南。

表 1 建模和编码指南涵盖的主题

通则		ASIL 等级			
		A	B	C	D
1a	强制低复杂度[a]	++	++	++	++
1b	语言子集的使用[b]	++	++	++	++
1c	强制强类型[c]	++	++	++	++
1d	防御性实施技术的使用[d]	+	+	++	++
1e	使用值得信赖的设计原则[e]	+	+	++	++
1f	使用无歧义的图形表示	+	++	++	++

1g	风格指南的使用	+	++	++	++
1h	命名惯例的使用	++	++	++	++
1i	并发方面[f]	+	+	+	+

[a] 可能需要将该通则与本文档的其他要求进行适当的折中。

[b] 主题 1b 的目标包括：

——排除模棱两可的语言构造，这些构造可能由不同的建模者，程序员，代码生成器或编译器以不同的方式解释；

——从经验中排除容易导致错误的语言构造，例如，条件分配或局部和全局变量的相同命名；

——排除可能导致未处理的运行时错误的语言构造。

[c] 1c 的目的是在语言中没有固有强类型的原则情况下强加这些原则。

[d] 防御性实施技术示例：

——除法运算之前验证除数（非零或在特定范围内）；

——检查由参数传递的标识符，以验证调用函数是否为预期的调用者；

——在 switch-case 语句中使用 default 分支。

[e] 可能需要验证基本假设，适用范围和条件的有效性。

[f] 进程或任务的并发性不仅限于在多核或多处理器运行环境中执行软件。

5.5 工作成果

5.5.1 软件开发环境文档，由 5.4.1～5.4.3 和 C.4.1～C.4.11 的要求得出。

6 软件安全要求的定义

6.1 目的

该子阶段的目的是：

a) 定义或细化由技术安全概念和系统架构设计规范导出的软件安全要求；

b) 定义软件实现所需的安全相关功能和特性；

c) 细化在 GB/T 34590.4-XXXX 第 6 章最初定义的软硬件接口要求；及

d) 验证软件安全要求和软硬件接口要求是否适用于软件开发，及验证它们与技术安全概念和系统架构设计规范的一致性。

6.2 总则

在GB/T 34590.4-XXXX，第6章定义的系统架构设计阶段中，技术安全要求被细化并分配给硬件和软件。软件安全要求的定义特别考虑硬件约束及其对软件的影响。该子阶段包括软件安全要求的定义，以支持后续设计阶段。

6.3 本章的输入

6.3.1 前提条件

应具备下列信息：

——技术安全要求规范，按照 GB/T 34590.4-XXXX，6.5.1；

——技术安全概念，按照 GB/T 34590.4-XXXX，6.5.2；

——系统架构设计规范，按照 GB/T 34590.4-XXXX，6.5.3；

——软硬件接口规范，按照 GB/T 34590.4-XXXX，6.5.4；及

——软件开发环境文档，按照 5.5.1。

6.3.2 支持信息

可考虑下列信息：

——硬件设计规范(见 GB/T 34590.5-XXXX，7.5.1)；及

——软件非安全相关功能和特性的定义（来自外部）。

6.4 要求和建议

6.4.1 软件安全要求得出时，应考虑所需的安全相关的软件功能和特性，其失效可能违背分配给软件的技术安全要求。

注 1：软件安全要求或者直接来源于分配给软件的技术安全要求，或者是对软件功能和特性的要求（这些要求如果不满足可能违背分配给软件的技术安全要求）。

示例 1：软件安全相关功能可以是：

——使标称功能可以安全执行的功能；

——使系统达到或维持安全状态或降级状态的功能；

——与安全相关硬件要素故障探测、指示和减轻相关的功能；

——与操作系统、基础软件或应用软件本身失效探测、指示和减轻有关的自检或监控功能；

——在生产、运行、服务和报废过程中与车载测试和非车载测试相关的功能；

——允许在生产和服务过程中对软件进行修改的功能；或

——与性能或对时间敏感的操作相关的功能。

示例 2：安全相关的特殊特性包括对错误输入的鲁棒性、不同功能之间的独立性或免于干扰、或软件的容错能力。

注 2：安全导向分析（本文件 7.4.10 或 7.4.11）可用于识别额外的软件安全要求或为其实现提供证据。

6.4.2 软件安全要求的定义应按照 GB/T 34590.4-XXXX，6.4.1 和 6.4.3 的技术安全要求、技术安全概念和系统架构设计得出，并应考虑如下内容：

a) 安全要求的定义和管理，按照 GB/T 34590.8-XXXX，第 6 章；

b) 已定义的系统和硬件的配置；

示例 1：配置参数可包括增益控制、带通频率和时钟分频。

c) 软硬件接口规范；

d) 硬件设计规范的相关要求；

e) 时间约束；

f) 示例 2：由系统层面要求的响应时间得出执行或反应时间。

g) 外部接口；及

示例 3：通讯和用户接口。

h) 对软件有影响的车辆、系统或者硬件的每个运行模式以及运行模式之间的转换。

示例 4：运行模式包括下电或休眠、初始化、正常运行、降级和用于测试或刷新的其他高级模式。

6.4.3 如果对软件安全要求进行了 ASIL 等级分解，应满足 GB/T 34590.9-XXXX，第 5 章的要求。

6.4.4 在 GB/T 34590.4-XXXX 第 6 章初步定义的软硬件接口规范，应细化到可以通过软件正确控制和使用硬件的程度，并应描述硬件和软件间每个与安全相关的依赖性。

6.4.5 如果嵌入式软件除了执行 6.4.1 定义的安全要求的功能外，还执行了其他功能，则应按照所应用的质量管理体系的要求提供这些功能及其特性的规范。

6.4.6 应由负责系统开发、硬件开发和软件开发的人员共同验证细化后的软硬件接口规范。

6.4.7 应按照 GB/T 34590.8-XXXX 第 6 章和第 9 章的要求，对软件安全要求和细化后的软硬件接口规范进行验证，以提供证据证明：

a) 软件开发的适用性；

b) 与技术安全要求的符合性和一致性；

c) 与系统设计的符合性；及

d) 与软硬件接口的一致性。

6.5 工作成果

6.5.1 软件安全需求规范，由 6.4.1～6.4.3 和 6.4.5 的要求得出。

6.5.2 软硬件接口规范（细化的），由 6.4.4 的要求得出。

注：该工作成果参照 GB/T 34590.5-XXXX，6.5.2 相同的工作成果。

6.5.3 软件验证报告，由 6.4.6 和 6.4.7 的要求得出。

7 软件架构设计

7.1 目的

该子阶段的目的是：

a) 开发满足软件安全要求和其他软件要求的软件架构设计；

b) 验证软件架构设计适合满足所要求 ASIL 等级的软件安全要求；及

c) 支持软件的实现与验证。

7.2 总则

软件架构设计以层次结构的形式表示软件架构要素以及他们的交互方式。描述了静态方面，如软件组件之间的接口、和动态方面，如进程序列和时序行为。

注：软件架构设计并不一定局限于某个微控制器或 ECU。每个微控制器的软件架构也在此子条中论述。

软件架构设计既能满足软件安全要求，又能满足其他软件要求。因此，在该子阶段中，与安全相关和非安全相关的软件要求在同一个开发过程中处理。

软件架构设计提供了实现软件要求和 ASIL 等级所需的软件安全要求的方法，也提供了管理软件详细设计和实现复杂度的方法。

7.3 本章输入

7.3.1 前提条件

应具备下列信息：

——软件开发环境文档，按照 5.5.1；

——软硬件接口规范(细化的)，按照 6.5.2；及

——软件安全需求规范，按照 6.5.1。

7.3.2 支持信息

可考虑下列信息：

——技术安全概念（见 GB/T 34590.4-XXXX，6.5.2）；

——系统架构设计规范（见 GB/T 34590.4-XXXX，6.5.3）；

——可用的经鉴定合格的软件组件（见 GB/T 34590.8-XXXX，第 12 章）；及

——软件非安全相关功能和特性的定义以及其他软件要求的定义，按照 6.4.5（来自外部）。

7.4 要求和建议

7.4.1 为避免软件架构设计和后续开发活动中的系统性故障，软件架构设计的描述应满足表 2 中列出的软件架构设计标记法所支持的特征：

a) 可理解性；

b) 一致性；

c) 简单性；

d) 可验证性；

e) 模块化；

f) 抽象性；

注：抽象性可以通过使用层次化结构、分组方案或视图来支持，以涵盖架构设计的静态、动态或部署方面。

g) 封装性；及

h) 可维护性。

表 2 软件架构设计标记法

方法		ASIL 等级			
		A	B	C	D
1a	自然语言[a]	++	++	++	++
1b	非形式记法	++	++	+	+
1c	半形式记法[b]	+	+	++	++
1d	形式记法	+	+	+	+

注：UML®、SysML®、Simulink®和 Stateflow®都是商业可用的合适产品的例子。这些信息是为了方便本文档的用户而提供的，并不构成国际标准化组织对这些产品的认可。

[a] 自然语言可以补充标记法的使用，例如，某些主题更容易用自然语言表达，或为使用此标记法提供解释和理由。

[b] 半形式记法可以包括伪代码或使用 UML®、SysML®、Simulink®或 Stateflow®的建模。

7.4.2 在软件架构设计开发中，应该考虑下述方面：

a) 软件架构设计的可验证性；

注：这意味着软件架构设计和软件安全要求之间的双向可追溯性。

b) 可配置软件的适用性；

c） 软件单元设计与实现的可行性；

d） 软件集成测试中软件架构的可测试性；及

e） 软件架构设计的可维护性。

7.4.3 为避免系统性故障，应使用表 3 列出的原则，使软件架构设计具有以下特征：

a） 可理解性；

b） 一致性；

c） 简单性；

d） 可验证性；

e） 模块化；

f） 封装性；及

g） 可维护性。

表 3 软件架构设计原则

方法		ASIL 等级			
		A	B	C	D
1a	软件组件的适当分层结构	++	++	++	++
1b	限制软件组件的规模和复杂度 [a]	++	++	++	++
1c	限制接口规模 [a]	+	+	+	++
1d	每个组件内强内聚 [b]	+	++	++	++
1e	软件组件间松耦合 [b,c]	+	++	++	++
1f	恰当调度的特性	++	++	++	++
1g	限制中断的使用 [a,d]	+	+	+	++
1h	软件组件的适当空间隔离	+	+	+	++
1i	共享资源的适当管理 [e]	++	++	++	++

[a] 在原则 1b、1c 和 1g 中，“限制”表示与其他设计考虑进行平衡后的最低程度。

[b] 例如，原则 1d 和 1e 可通过分隔关注点的方法实现，这些关注点代表了对特定概念、目标、任务或目的相关的软件部分进行识别、封装和操作的能力。

[c] 原则 1e 针对软件组件间相关性的管理。

[d] 原则 1g 可以包括最小化数量，或使用具有明确优先级的中断来实现确定性。

[e] 在共存的情况下，原则 1i 适用于共享硬件资源以及共享软件资源。这种资源管理可以用软件或硬件实现，包括安全机制和/或防止共享资源访问冲突的过程措施，以及探测和处理共享资源访问冲突的机制。

注 1：基于表 3 所列原则的适当折中是必要的，因为这些原则不互相排斥。

注 2：高复杂度的指标可以是：

——高度分支化的控制流或数据流；

——分配给单个设计要素的要求过多；

——某个设计要素的接口过多或设计要素之间的交互性过多；

——参数类型复杂或过多；

——全局变量过多；

——难以为错误探测和处理的合适性和完整性提供证据；

——难以达到要求的测试覆盖率；或

——只有少数专家或项目参与者能够理解。

注3：这些特性和原则也适用于软件例行程序（例如，中断处理的服务例行程序）。

7.4.4 软件架构设计应被开发到能够识别出软件单元的程度。

7.4.5 软件架构设计应描述：

a) 软件架构要素的静态设计方面；及

注 1：静态设计方面涉及：

——包括分级层次的软件结构；

——数据类型和它们的特征参数；

——软件组件的外部接口；

——嵌入式软件的外部接口；

——全局变量；及

——包括架构的范围和外部依赖的约束。

注 2：在基于模型的开发的情况下，结构建模可能是整个建模活动的固有部分。建模的结构可能取决于所选的建模语言。

b) 软件架构要素的动态设计方面。

注 3：动态设计方面涉及：——事件和行为的功能链；——数据处理的逻辑顺序；——控制流和并发进程；——通过接口和全局变量传递的数据流；及——时间的限制。

注 4：为确定动态行为（如任务、时间片和中断），需要考虑不同的运行状态（如开机、关机、正常运行、标定和诊断）。

注 5：为描述动态行为（如任务、时间片和中断），需要定义通讯关系及其在系统硬件（如 CPU 和通讯通道）上的分配。

7.4.6 软件安全要求应按层次分配给软件组件，直至软件单元。因此，每个软件组件应按照分配给它的任何需求的最高的 ASIL 等级来进行开发。

注：在设计开发和要求分配过程中，可能需要按照第 6 章对软件安全要求进行拆分或进一步细化。

7.4.7 如果复用一个没有按照 GB/T 34590 进行开发的未修改的现有软件架构要素，为满足分配的安全要求，则应按照 GB/T 34590.8-XXXX，第 12 章的要求进行鉴定。

注 1：使用经鉴定合格的软件组件不影响第 10 章和第 11 章的适用性。然而，第 8 章和第 9 章描述的某些活动可以被省略。

注 2：按照 GB/T 34590 开发的复用软件要素的适用性可在软件架构设计验证阶段来实现。

7.4.8 如果嵌入式软件不得不实现不同 ASIL 等级的软件组件，或实现安全相关及非安全相关的软件组件，除非软件组件符合 GB/T 34590.9-XXXX 第 6 章定义的共存准则，否则全部嵌入式软件必须按照最高 ASIL 等级来处理。

7.4.9 如果用软件分区（本文件附录 D）实现软件组件间免于干扰，那么应该确保：

a) 共享资源的使用方式应确保软件分区免于干扰；

注 1：一个软件分区内的任务彼此之间不能免于干扰。

注 2：一个软件分区不能改变其他软件分区的代码或数据，也不能控制其他软件分区的非共享资源。

注 3：一个软件分区从共享资源获取的服务不能被另一个软件分区影响。这包括相关资源的性能，以及对资源调度访问的使用率、延迟、抖动和持续时间。

b) 由专用的硬件特性或等效方法来支持软件分区（该要求适用于 ASIL D 等级，按照 4.4）；

示例：存储器保护单元等硬件特性。

c) 实现软件分区的软件要素是根据分配给分区软件任何要求的最高 ASIL 等级开发的；及

注： 一般来说操作系统提供或支持软件分区。

d) 软件分区有效性的证据会在软件集成和验证期间生成（按照第 10 章）。

7.4.10 应按照 GB/T 34590.9-XXXX，第 8 章在软件架构层级执行安全导向分析，目的是：

——提供软件的适用性证据证明具备了相应的 ASIL 等级要求所需的特定的安全相关功能和特性；

注 1：安全相关的特殊特性包括独立性和免于干扰。

——识别或确认软件的安全相关部分；及

——支持安全措施的定义并验证其有效性。

注 2：安全措施包括从安全导向分析中得出的安全机制，并可涵盖与随机硬件失效和软件故障有关的问题。

注 3：关于在软件架构层级运用安全导向分析和选择适当安全措施的更多信息，本文件附录 E。

7.4.11 如果软件安全要求的实现依赖于软件组件间免于干扰或足够的独立性，则应按照 GB/T 34590.9-XXXX ，第 7 章进行相关失效及其影响分析。

注：在软件架构层级应用相关失效分析的更多信息，本文件附录 E 和 GB/T 34590.9-XXXX ，附录 C。

7.4.12 应根据软件架构层级安全导向的分析（按照 7.4.10 和 7.4.11）结果，采用错误探测和错误处理的安全机制。

注 1：附录 E 提供了判断失效模式是否需要安全机制的指导。

注 2：用于错误探测的安全机制包括：

——输入输出数据的范围检查；

——合理性检查（例如，使用期望行为的参考模型、断言检查、或不同来源信号的比较）；

——数据错误探测（例如，检错码和多重数据存储）；

——外部要素监控程序执行，例如，通过专用集成电路（ASIC）或者其他软件要素来执行看门狗功能。监控可以是逻辑监控或时间监控，或者两者的结合；

——程序执行的时间监控；

——设计中的异构冗余；或——在软件或硬件中实施的访问冲突控制机制，与授权访问或拒绝访问安全相关共享资源有关。

注3：用于错误处理的安全机制可能包括：

——为了达到和维持安全状态的功能关闭；

——静态恢复机制（例如，恢复块、后向恢复、前向恢复以及通过重试来恢复）；

——通过划分功能的优先级进行平稳降级，从而最小化潜在失效对功能安全的不利影响；

——设计中的同构冗余，主要侧重于控制运行相似软件的硬件中瞬态故障或随机故障的影响（例如，软件在时间上的冗余执行）；

——设计中的异构冗余，意味着在每个并行路径中使用不同的软件，主要侧重于预防或控制软件中的系统性故障；

——数据纠错码；或

——在软件或硬件中实施的访问许可管理，与授权访问或拒绝访问安全相关共享资源有关。

注4：可在系统层面对软件安全机制（包括一般的鲁棒性机制）进行评审，以分析其对系统行为的潜在影响和与技术安全要求的一致性。

7.4.13 应该对嵌入式软件所需资源进行上限预估，包括：

a) 执行时间；

b) 存储空间；及

示例：用于存储堆和栈的 RAM，用于存储程序和非易失性数据的 ROM。

c) 通讯资源。

7.4.14 软件架构设计应按照 GB/T 34590.8-XXXX，第 9 章来进行验证，并使用表 4 中所列出的软件架构设计验证方法，为实现下列目标提供证据：

a) 软件架构设计满足对应 ASIL 等级的软件安全要求；

b) 软件架构设计的评审或审核能够为设计满足对应 ASIL 等级的软件安全要求提供证据；

c) 与目标环境的兼容性；及

注：目标环境是指软件运行的环境，包含了操作系统、基础软件以及在 7.4.13 中详细描述的目标硬件及其资源。

d) 与设计指南保持一致。

表 4 软件架构设计验证方法

方法		ASIL 等级			
		A	B	C	D
1a	设计走查[a]	++	+	o	o
1b	设计检查[a]	+	++	++	++
1c	对设计中的动态行为进行仿真	+	+	+	++
1d	生成原型	o	o	+	++
1e	形式验证	o	o	+	+
1f	控制流分析[b]	+	+	++	++
1g	数据流分析[b]	+	+	++	++
1h	调度分析	+	+	++	++

[a] 在基于模型的开发的情况下，这些方法可以在模型中应用。

[b] 控制流和数据流分析可以限制在安全相关组件和它们的接口上。

7.5 工作成果

7.5.1 **软件架构设计规范**，由 7.4.1～7.4.13 的要求得出。

7.5.2 **安全分析报告**，由 7.4.10 的要求得出。

7.5.3 **相关失效分析报告**，由 7.4.11 的要求得出。

7.5.4 **软件验证报告**，由 7.4.14 的要求得出。

8 软件单元设计和实现

8.1 目的

该子阶段的目的是：

a) 按照软件架构设计、设计准则和所分配的支持软件单元实施和验证的软件要求，开发软件单元设计；及

b) 实现所定义的软件单元。

8.2 总则

应基于软件架构设计，开发软件单元的详细设计。详细设计可以是模型化的形式。

源代码层面的实施可以按照由软件开发环境中的设计手动或自动生成。

为了开发每一个软件单元设计，软件安全要求和非安全相关的要求都要实现。因此，该子阶段，安全相关的和非安全相关的要求都按照同一个开发流程来处理。

8.3 本章的输入

8.3.1 前提条件

应具备下列信息：

——软件开发环境文档，按照 5.5.1；

——软硬件接口规范（细化的），按照 6.5.2；

——软件架构设计规范，按照 7.5.1；

——软件安全需求规范，按照 6.5.1；

——配置数据，按照 C.5.3，如果适用；及

——标定数据，按照 C.5.4， 如果适用。

8.3.2 支持信息

可考虑下列信息：

——技术安全概念（本文件 GB/T 34590.4-XXXX，6.5.2）；

——系统架构设计规范（本文件 GB/T 34590.4-XXXX，6.5.3）；

——非安全相关功能和软件特性的定义（来自外部）；

——安全分析报告（本文件 7.5.2）；及

——相关失效分析报告（本文件 7.5.3）。

8.4 要求和建议

8.4.1 如果软件单元是安全相关要素，应符合本子阶段中的要求。

注：“安全相关”是指该单元要实现安全要求，或该单元不满足与其他单元的共存准则（本文件 GB/T 34590.9-XXXX，第 6 章）。

8.4.2 软件单元设计和实现应：

a) 适用于满足分配给具有 ASIL 等级的软件单元的软件要求；

b） 符合软件架构设计规范；及

c） 符合软硬件接口规范，如果适用。

示例：与其他软件单元接口的一致性和完整性；输入或输出数据的正确性、准确性和及时性。

8.4.3 为避免系统性故障并确保软件单元设计具有以下特性，软件单元设计应使用表 5 中列出的标记法进行描述。

a） 一致性；

b） 可理解性；

c） 可维护性；及

d） 可验证性。

表 5 软件单元设计的标记法

方法		ASIL 等级			
		A	B	C	D
1a	自然语言[a]	++	++	++	++
1b	非形式记法	++	++	+	+
1c	半形式记法[b]	+	+	++	++
1d	形式记法	+	+	+	+

[a] 自然语言可以补充标记法的使用，例如，某些主题更容易用自然语言表达，或为使用此标记法提供解释和理由。

示例：设计复杂要素时为避免自然语言可能产生歧义，可以将活动图与自然语言结合使用。

[b] 半形式记法包括伪代码或者使用 UML®、SysML®、Simulink® 或 Stateflow®建模。

注：UML®、SysML®、Simulink® 和 Stateflow®都是商业可用的合适产品的例子。这些信息是为了方便本文档的用户而提供的，并不构成国际标准化组织对这些产品的认可。

注：在应用自动代码生成的基于模型的开发的情况下，将软件单元设计的表示方法，用于作为代码生成基础的模型中。

8.4.4 软件单元的定义应将功能表现和内部设计描述到必要的详细层级以支持其实现。

示例：内部设计可包含对寄存器使用和数据存储的限制。

8.4.5 应运用表 6 列出的源代码层面软件单元设计和实现的设计原则，以具有如下特性：

a） 基于软件架构设计，软件单元内的子程序和函数执行的正确次序；

b） 软件单元间接口的一致性；

c） 软件单元内和软件单元间的数据流及控制流的正确性；

d） 简单性；

e） 可读性和可理解性；

f） 鲁棒性；

示例：避免不合理值、执行错误、以零做除数、数据流及控制流错误的方法。

g） 软件修改的适宜性；及

h） 可验证性。

表 6 软件单元设计和实现的设计原则

方法		ASIL 等级			
		A	B	C	D
1a	子程序和函数采用一个入口和一个出口[a]	++	++	++	++
1b	无动态对象或动态变量，否则需要在其产生过程中对其进行在线测试[a]	+	++	++	++
1c	变量初始化	++	++	++	++
1d	不能重复使用变量名称[a]	++	++	++	++
1e	避免全局变量，否则需证明对全局变量的使用是合理的[a]	+	+	++	++
1f	限制使用指针[a]	+	++	++	++
1g	无隐式类型转换[a]	+	++	++	++
1h	无隐藏数据流或控制流	+	++	++	++
1i	没有无条件跳转[a]	++	++	++	++
1j	无递归	+	+	++	++
[a] 方法 1a、1b、1d、1e、1f、1g 和 1i 可能不适用于在基于模型开发中用到的图形模型标记法。					

注：对于 C 语言来说，MISRA C（本文件参考文档[3]）涵盖了表 6 列出的很多原则。

8.5 工作成果

8.5.1 软件单元设计规范，由 8.4.2～8.4.5 的要求得出。

注：在基于模型的开发的情况下，运用表 5 和表 6 所列方法实现的模型和支持说明文档，定义软件单元。

8.5.2 软件单元实现，由 8.4.5 的要求得出。

9 软件单元验证

9.1 目的

本子阶段的目的是：

a) 提供证据证明软件单元设计满足分配的软件要求且适合于实施；

b) 验证按照 7.4.10 和 7.4.11 实施的安全分析得出的安全措施得到适当实施；

c) 提供证据证明所实现的软件单元符合单元设计，并满足根据所需的 ASIL 等级分配的软件要求；及

d) 提供充分证据，证明软件单元不包含与功能安全相关的非预期功能和特性。

9.2 总则

软件单元设计和已实现的软件单元通过使用（如：评审、分析和测试）组合进行了验证。

为了验证单个软件单元设计，同时考虑了软件安全要求和所有非安全相关要求。因此，

在此子阶段中，安全相关和非安全相关要求在同一个开发流程中处理。

9.3 本章的输入

9.3.1 前提条件

应具备下列信息：

——软硬件接口规范（细化的），按照 6.5.2；

——软件架构设计规范 ，按照7.5.1；

——软件单元设计规范，按照8.5.1；

——软件单元实现，按照8.5.2；

——配置数据，按照C.5.3，如果适用；

——标定数据，按照C.5.4，如果适用；

——安全分析报告，按照7.5.2；及

——软件开发环境文档，按照5.5.1。

9.3.2 支持信息

可考虑下列信息：

——无。

9.4 要求和建议

9.4.1 如果软件单元是安全相关要素，则应符合本条的要求。

注 1：按照 GB/T 34590.1 的定义，“安全相关要素”是指该单元实现安全要求，或此单元不满足与其他单元的共存原则（本文件 GB/T 34590.9-XXXX，第 6 章）。

注 2：本章的要求针对安全相关的软件单元；其他软件标准（本文件 GB/T 34590.2-XXXX，5.4.5.1）可用于其他软件单元的验证。

注 3：对于基于模型的软件开发，模型实现的相应部分也是验证计划的对象。根据所选的软件开发流程，验证对象可以是从该模型生成的代码、模型本身或两者都包含。

9.4.2 应按照 GB/T 34590.8-XXXX，第 9 章的规定，通过采用表 7 所示方法的适当组合，对软件单元设计和已实现的软件单元进行验证，以提供证据证明：

a) 符合第 8 章中有关单元设计和实现的要求；

注 1：软件安全要求包括软件的功能和特性。

注 2：在模型层面执行验证可以替代在源代码层面执行验证，如果代码生成保留模型的特性（例如，所使用的代码生成器有足够的置信度）。

示例：有效实施错误检测和错误处理机制的证据，以实现软件单元对错误输入的鲁棒性。

b) 源代码与其设计规范的符合性；

注 3：在基于模型的开发的情况下，要求 b）仍然适用。

c) 符合软硬件接口规范（按照 6.4.4），如果适用；

d) 确信没有非预期功能和特性；

e) 足够的资源支持其功能和特性；及

f) 实施由安全导向分析（按照 7.4.10 和 7.4.11）得出的安全措施。

表 7 软件单元验证方法

方法	ASIL 等级

		A	B	C	D
1a	走查[a]	++	++	o	o
1b	结对编程[a]	+	+	+	+
1c	检查[a]	+	++	++	++
1d	半形式验证	+	+	++	++
1e	形式验证	o	o	+	+
1f	控制流分析[b, c]	+	+	++	++
1g	数据流分析[b, c]	+	+	++	++
1h	静态代码分析[d]	++	++	++	++
1i	基于抽象解释的静态分析[e]	+	+	+	+
1j	基于需求的测试[f]	++	++	++	++
1k	接口测试[g]	++	++	++	++
1l	故障注入测试[h]	+	+	+	++
1m	资源使用评估[i]	+	+	+	++
1n	如果适用，在模型和代码之间背靠背对比测试[j]	+	+	++	++

[a] 对于基于模型的开发，如果有证据证明所使用的代码生成器可信，则这些方法将应用在模型层面上。

[b] 方法1f和1g可用于源代码层面。这些方法同时适用于手动代码开发和基于模型的开发。

[c] 方法1f和1g可作为方法1e、1h或1i的一部分。

[d] 静态分析是一个集合术语，它包括诸如搜索源代码文本或模型以查找与已知故障匹配的模式或符合建模或编码准则的分析。

[e] 基于抽象解释的静态分析是扩展静态分析的集合术语，包括诸如通过添加语义信息来扩展编译器分析树之类的分析，这些语义信息可以检查是否违反了定义的规则（例如，数据类型问题、未初始化的变量）；包括控制流图生成和数据流分析（例如，捕获与竞争条件和死锁、指针误用相关的故障）、或甚至元编译及抽象代码/模型解释。

[f] 单元层面的软件要求是基于需求测试的基础。包括软件单元设计规范和分配给软件单元的软件安全要求。

[g] 此方法能够为所使用和交换的数据的完整性提供证据。

[h] 在软件单元测试时，故障注入测试是指为了9.4.2所述的目的修改被测试的软件单元（例如，将故障引入软件）。这种修改包括注入任意错误（例如，通过损坏变量的值、引入代码突变或通过损坏CPU寄存器的值）。

[i] 只有在目标环境上执行软件单元测试或目标处理器的仿真器充分支持资源使用测试时，才能正确执行资源使用评估的某些方面。

[j] 此方法需要一个能够模拟软件单元功能的模型。在这里，以相同的方式对模型和代码进行了模拟，并将模拟结果进行了比较。

示例：在基于模型设计的情况下，非浮点运算的结果可以进行比较。

9.4.3 应使用表 8 列出的方法得到测试用例，以恰当定义符合 9.4.2 的软件单元测试的测试用例。

表 8 软件单元测试用例的得出方法

方法		ASIL 等级			
		A	B	C	D
1a	需求分析	++	++	++	++
1b	等价类的生成和分析[a]	+	++	++	++
1c	边界值分析[b]	+	++	++	++
1d	基于知识或经验的错误推测[c]	+	+	+	+

[a] 可基于划分输入输出来识别等价类，为每个等价类选择一个有代表性的测试值。

[b] 该方法用于接口、接近边界的值、与边界交叉的值及超出范围的值。

[c] 错误猜测测试可基于经验学习流程中收集的数据和专家判断。

9.4.4 为了评估验证的完整性并提供证据证明已充分实现单元测试目标，应确定在软件单元层面的要求覆盖率，同时应按照表 9 列出的度量对结构覆盖率进行测定。如果认为已实现的结构覆盖率不充分，应定义额外的测试用例或提供基于其他方法的理由。

注 1：在没有理由情况下，结构覆盖率无目标值或低目标值被认为是不充分的。

示例 1：结构覆盖率分析可以显示基于要求的测试用例的不足、要求的缺陷、无作用码、无效代码或非预期的功能。

示例 2：对于可接受的死代码（例如，用于调试的代码）或不同软件配置的代码区段，可以给出接受所达到的覆盖率水平的理由，或可以使用补充方法（例如，检查）验证未被覆盖的代码。

示例 3：理由可基于最新技术水平。

表 9 软件单元层面的结构覆盖率度量

方法		ASIL 等级			
		A	B	C	D
1a	语句覆盖率	++	++	+	+
1b	分支覆盖率	+	++	++	++
1c	MC/DC（修改条件/判定覆盖率）	+	+	+	++

注 2：通过使用适当的软件工具可以确定结构覆盖率。

注 3：在基于模型的开发的情况下，结构覆盖率分析可以利用相似的模型结构覆盖率度量在模型层面进行。

示例4：如果有证据证明结构覆盖率在模型层面和代码层面等效，则在模型层面执行的结构覆盖率分析可以替代源代码覆盖率度量。

注 4：如果“检测代码”用于确定结构覆盖率水平，则有必要提供证据，证明检测对测试结果没有影响，这可以通过使用“非检测代码”重复测试具有代表性的测试用例实现。

9.4.5 对于软件单元测试的测试环境，应考虑目标环境从而使其适合于达到单元测试的目的。如果软件单元测试不是在目标环境下执行，应分析源代码和目标代码的差异及测试环境和目标环境之间的差异，以便在后续测试阶段的目标环境中，定义额外的测试。

注 1：测试环境和目标环境之间的差异，可出现在源代码或目标代码中，例如，由于不同处理器的数据字和地址字的不同位宽引起的差异。

注 2：根据测试范围，使用适当的测试环境（例如，目标处理器、处理器仿真器或开发系统）执行软件单元测试。

注 3：软件单元测试可以在不同的环境中执行，例如：

——模型在环测试；

——软件在环测试；

——处理器在环测试；或

——硬件在环测试。

注 4：对基于模型的开发，在模型层面执行软件单元测试，随后，在模型和目标代码之间进行背靠背的比较测试。对于测试目标而言，背靠背比较测试用于确保模型与自动生成的代码的行为是一致的。

9.5 工作成果

9.5.1 **软件验证规范**，由 9.4.2～9.4.5 的要求得出。

9.5.2 **软件验证报告（细化的）**，由 9.4.2 的要求得出。

10 软件集成和验证

10.1 目的

此子阶段的目的是：

a) 定义集成步骤并集成软件要素，直至嵌入式软件完全集成；

b) 验证由软件架构层面的安全分析得出的已定义的安全措施得到适当实施；

c) 提供证据证明集成的软件单元和集成的软件组件符合软件架构设计的要求；及

d) 提供充分证据，证明集成软件不包含与功能安全相关的非预期功能和特性。

10.2 总则

在此子阶段，按照软件架构设计，对软件要素之间特有的集成层次和接口进行验证。软件要素的集成和验证的步骤直接对应着软件的分层架构。

嵌入式软件可能由安全相关和安全无关的软件要素组成。

10.3 本章的输入

10.3.1 前提条件

应具备以下信息：

——软硬件接口规范（细化的），按照 6.5.2；

——软件架构设计规范，按照 7.5.1；

——安全分析报告，按照 7.5.2；

——相关失效分析报告，按照 7.5.3，如果适用；

——软件单元实现，按照 8.5.2；

——配置数据，按照 C.5.3，如果适用；

——标定数据，按照 C.5.4，如果适用；

——软件开发环境文档，按照 5.5.1；

——软件验证规范，按照 9.5.1。

10.3.2 支持信息

可考虑如下信息：

——经鉴定合格的软件组件，本文件 GB/T 34590.8-XXXX，第 12 章；

10.4 要求和建议

10.4.1 软件集成的方法应定义和描述将各个软件单元分层集成到软件组件中的步骤，直到整个嵌入式软件全部被集成，并应考虑：

a) 与 10.4.2 中规定的验证目标实现的相关性；

b) 与软件集成相关的功能相关性；及

c) 软件集成和软硬件集成之间的相关性。

注：对基于模型的开发，软件集成可被模型层面的集成和后续由集成的模型生成自动代码来代替（本文件附录 B）。

10.4.2 应按照 GB/T 34590.8-XXXX，第 9 章的要求，通过表 10 提供方法的适当组合，验证软件集成，以证明分层集成的软件单元、软件组件和集成的嵌入式软件实现：

a) 与软件架构设计的符合性，按照第 7 章；

b) 与软硬件接口规范的符合性，按照 6.5.2；

c) 已定义的功能；

d) 已定义的特性；

示例：由于不存在无法访问的软件而产生的可靠性、对错误输入的鲁棒性、由于有效的错误检测和处理而产生的可靠性。

e) 支持功能的足够资源；及

f) 按照 7.4.10 和 7.4.11 的安全导向分析得出的安全措施的有效性，如果适用。

注 1：安全措施可包括软件分区。

表 10 软件集成验证方法

方法		ASIL 等级			
		A	B	C	D
1a	基于需求的测试[a]	++	++	++	++
1b	接口测试	++	++	++	++
1c	故障注入测试[b]	+	+	++	++
1d	资源使用评估[c, d]	++	++	++	++
1e	模型和代码之间的背靠背比较测试，如果适用[e]	+	+	++	++
1f	控制流和数据流的验证	+	+	++	++
1g	静态代码分析[f]	++	++	++	++

1h	基于抽象解释的静态分析[g]	+	+	+	+

[a] 分配给架构要素的软件要求是基于需求测试的基础。

[b] 在软件集成测试时，故障注入测试是指将故障引入软件以实现 10.4.3 中所述的目的，特别是测试与安全机制相关的软硬件接口的正确性。这包括注入任意故障以测试安全机制（例如，通过损坏软件接口）。故障注入也可用于验证免于干扰。

[c] 为了确保以足够的裕量满足受硬件架构设计影响的要求，应确定诸如平均和最大的处理器性能、最小和最大执行时间、存储使用情况（例如，堆栈使用的 RAM、程序和数据使用 ROM）以及通信链路的带宽（例如，数据总线）等特性。

[d] 只有在目标环境上执行软件集成测试或目标处理器的仿真器充分支持资源测试时，才能正确执行资源使用评估的某些方面。

[e] 这个方法需要一个可以模拟软件组件功能的模型。在这里，通过相同的方式激励模型和代码，并比较彼此输出的结果。

[f] 静态分析是一个集合术语，包括诸如架构分析，资源消耗分析以及在源代码文本或模型中搜索与已知故障相匹配的模式或是否符合建模或编码准则（如果尚未在单元层面进行验证）的分析。

[g] 基于抽象解释的静态分析是扩展静态分析的集合术语，其中还包括诸如通过添加可以检查是否违反定义规则的语义信息来扩展编译器解析树的分析（例如，数据类型问题，未初始化的变量），控制流图的生成和数据流分析（例如，捕获与竞态条件和死锁，指针滥用有关的故障），甚至元编译和抽象代码或模型解释（如果尚未在单元层面进行验证的话）。

注 2：对基于模型的开发，验证对象可以是与软件组件相关的模型。

10.4.3 为了能够给按照 10.4.2 选择的软件集成测试方法定义恰当的测试用例，应使用表 11 所列的方法。

表 11 软件集成测试用例的得出方法

方法		ASIL 等级			
		A	B	C	D
1a	需求分析	++	++	++	++
1b	等价类生成与分析[a]	+	++	++	++
1c	边界值分析[b]	+	++	++	++
1d	基于知识或经验的错误推测[c]	+	+	+	+

[a] 可基于划分输入输出来识别等价类，为每个等价类选择一个有代表性的测试值。

[b] 该方法适用于接近边界的、与边界交叉的以及超出范围的参数值或变量值。

[c] 错误推测测试可基于经验学习流程中收集的数据和专家判断。

10.4.4 为了评估验证的完整性并提供证据证明已充分实现集成测试的测试目标，应确定测试用例在软件架构层级对要求的覆盖率。如有必要，应规定额外的测试用例，或提供基于其他方法的基本原理。

10.4.5 按照 4.4，该条适用于 ASIL (A)、(B)、C 和 D 等级：为了评估测试用例的完整性，并提供证据证明已充分实现集成测试的测试目标，应按照表 12 列出的方法来评估结构覆盖

率。如果实现的结构覆盖率被认为是不充分的，应定义额外的测试用例或提供基于其他方法的理由。

示例：结构覆盖率分析可以显示基于需求的测试用例的不足、要求的缺陷、无作用码、无效代码或非预期功能。

表 12 软件架构层级的结构覆盖率

方法		ASIL 等级			
		A	B	C	D
1a	函数覆盖率 [a]	+	+	++	++
1b	调用覆盖率 [b]	+	+	++	++

[a] 方法 1a 是指软件中执行的软件子程序或函数的百分比（定义本文件 IEC 61508-7:2010， C.5.8）。

[b] 方法 1b 是指已执行的软件子程序或函数相对于软件中这些子程序或函数的每个已实现调用的百分比。

注：证据可以通过实施适当的软件集成和测试策略来提供。

注 1：通过适当的软件工具可确定结构覆盖率。

注 2：在基于模型的开发中，可以使用模型的类似结构覆盖率度量在模型层面执行软件集成测试。

10.4.6 应验证作为生产发布（按照 GB/T 34590.2-XXXX，6.4.13）一部分的嵌入式软件包含了全部已定义的功能和特性，且仅包含不损害软件安全要求符合性的未定义功能。

示例：在此上下文中，未定义的功能包括用于调试或检测的代码。

注：如果可确保这些未定义的功能不被执行，这是一种符合本要求的可接受的方法，否则就要做变更（本文件 GB/T 34590.8-XXXX 第 8 章）将这些代码移除。

10.4.7 软件集成测试的测试环境，应考虑目标环境从而使其适合于达到集成测试的目的。如果集成测试没有在目标环境中执行，应分析源代码和目标代码之间的差异以及测试环境和目标环境之间的差异，来定义后续测试阶段中在目标环境中的附加测试。

注 1：测试环境与目标环境之间的差异可出现在源代码或目标代码中，例如，由于不同处理器的数据字和地址字的不同位宽引起的差异。

注 2：根据测试范围和集成的层级，使用适当的测试环境进行软件要素测试。这些测试环境可以是用于最终集成的目标处理器，或者是用于之前集成步骤的处理器模拟器或开发系统。

注 3：软件集成测试可在不同环境中执行，例如：

——模型在环测试；

——软件在环测试；

——处理器在环测试；或

——硬件在环测试。

10.5 工作成果

10.5.1 **软件验证规范（细化的）**，由 10.4.2～10.4.7 要求得出。

10.5.2 **嵌入式软件**，由 10.4.1 的要求得出。

10.5.3 **软件验证报告（细化的）**，由 10.4.2 的要求得出。

11 嵌入式软件测试

11.1 目的

该子阶段的目的是提供证据，证明该嵌入式软件：

a) 在目标环境执行时满足安全相关要求；及
b) 不包含与功能安全相关的非预期功能和特性。

11.2 总则

此活动的目的是提供证据，证明嵌入式软件可以在目标环境中满足其要求。这些证据能通过使用其他验证活动的适当的成果来提供。

注：嵌入式软件测试由与软件实现之外的人员执行是一个好的实践。

11.3 本章的输入

11.3.1 前提条件

应具备下列信息：

——软件架构设计规范，按照 7.5.1；

——软件安全需求规范，按照 6.5.1；

——嵌入式软件，按照 10.5.2；

——标定数据，按照 C.5.4，如果适用；

——软件开发环境文档，按照 5.5.1；及

——软件验证规范（细化的），按照 10.5.1。

11.3.2 支持信息

可考虑下列信息：

——技术安全概念（本文件 GB/T 34590.4-XXXX，6.5.2）；

——系统架构设计规范（本文件 GB/T 34590.4-XXXX，6.5.3）；

——集成和测试策略（本文件 GB/T 34590.4-XXXX，7.5.1）；及

——集成和测试报告（本文件GB/T 34590.4-XXXX，7.5.2）。

11.4 要求和建议

11.4.1 为验证嵌入式软件在目标环境中是否满足软件安全要求，应在表 13 所列的适当测试环境中进行测试，并按照 GB/T 34590.8-XXXX 第 9 章的要求。

注：可以复用已有的测试用例，如来自软件集成测试的测试用例。

表13 用于执行软件测试的测试环境

方法		ASIL 等级			
		A	B	C	D
1a	硬件在环	++	++	++	++
1b	电子控制单元网络环境[a]	++	++	++	++
1c	整车环境	+	+	++	++

[a] 示例包括集成了车辆部分或全部电气系统的测试台架、“lab-car”或“mule-car”（杂合车或骡子车），以及残余总线仿真。

11.4.2 嵌入式软件的测试应采用表 14 所列的方法进行，以提供证据证明嵌入式软件满足其各自 ASIL 等级要求的软件要求。

表14 嵌入式软件的测试方法

方法		ASIL 等级			
		A	B	C	D
1a	基于需求的测试	++	++	++	++
1b	故障注入测试[a]	+	+	+	++
[a] 在软件测试时，故障注入测试是指通过破坏标定参数等方式将故障引入软件中。					

11.4.3 为了能够为按照 11.4.2 的要求执行软件测试而定义适当的测试用例，应使用表 15 中列出的方法得出测试用例。

表15 嵌入式软件测试用例的得出方法

方法		ASIL 等级			
		A	B	C	D
1a	需求分析	++	++	++	++
1b	等价类的生成与分析	+	++	++	++
1c	边界值分析	+	+	++	++
1d	基于知识或经验的错误猜测	+	+	++	++
1e	功能相关性分析	+	+	++	++
1f	操作用例分析[a]	+	++	++	++
[a] 软件操作用例的示例可以包括现场软件更新、仅在引导加载程序确保软件完整性的情况下启动标称应用程序、嵌入式软件在不同操作模式下的安全相关行为，例如，启动、诊断、降级、断电（进入睡眠状态）、通电（唤醒）、校准、不同 ECU 之间的模式同步功能或用于保护生产人员的下线专用测试模式。					

11.4.4 嵌入式软件的测试结果应根据以下方面进行评估：

a) 与期望结果的一致性；及

b) 软件安全要求的覆盖率。

注：这包括配置和标定范围的覆盖率。本文件附录 C。

11.5 工作成果

11.5.1 软件验证规范（细化的），由 11.4.1～11.4.3 的要求得出。

11.5.2 软件验证报告（细化的），由 11.4.1～11.4.4 的要求得出。

附录 A

（资料性）

产品开发软件层面管理的概览和工作流程

表 A.1 提供了产品开发软件层面特定阶段的目的、前提条件和工作成果的概览。

表 A.1 产品开发软件层面概览

章	目的	前提条件	工作成果
5 软件层面产品开发概述	本章的目的是： a) 确保合适且一致的软件开发流程；及 b) 确保合适的软件开发环境。	（无）	5.5.1 软件开发环境文档。
6 软件安全要求的定义	该子阶段的目的是： a) 定义或细化由技术安全概念和系统架构设计规范导出的软件安全要求； b) 定义软件实现所需的安全相关功能和特性； c) 细化在 GB/T 34590.4-XXXX 第 6 章最初定义的软硬件接口要求；及 d) 验证软件安全要求和软硬件接口要求是否适用于软件开发，及验证它们与技术安全概念和系统架构设计规范的一致性。	——技术安全要求规范，本文件 GB/T 34590.4-XXXX，6.5.1； ——技术安全概念，本文件GB/T 34590.4-XXXX，6.5.2； ——系统架构设计规范，本文件 GB/T 34590.4-XXXX，6.5.3； ——软硬件接口规范，本文件 GB/T 34590.4-XXXX，6.5.4；及 ——软件开发环境文档，本文件 5.5.1。	6.5.1 软件安全需求规范； 6.5.2 软硬件接口规范（细化的）； 6.5.3 软件验证报告。
7 软件架构设计	本子阶段的目的是： a) 开发满足软件安全要求和其他软件要求的软件架构设计； b) 验证软件架构设计适合满足所要求 ASIL 等级的软件安全要求；及 c) 支持软件的实现与验证。	——软件开发环境文档，本文件 5.5.1； ——软硬件接口规范(细化的)，本文件6.5.2；及 ——软件安全需求规范，本文件 6.5.1。	7.5.1 软件架构设计规范； 7.5.2 安全分析报告； 7.5.3 相关失效分析报告； 7.5.4 软件验证报告。
8 软件单元设计和实现	本子阶段的目的是： a) 按照软件架构设计、设计准则和所分配的支持软件单元实施和验证的软件要求，开发软件单元设计；及 b) 实现所定义的软件单元。	——软件开发环境文档，本文件 5.5.1； ——软硬件接口规范(细化的)，本文件6.5.2； ——软件架构设计规范，本文件 7.5.1； ——软件安全需求规范，本文件 6.5.1； ——配置数据，本文件C.5.3， 如	8.5.1 软件单元设计规范； 8.5.2 软件单元实现。

		果适用； ——标定数据，本文件C.5.4 ，如果适用。	
9 软件单元验证	该子阶段的目的是： a）提供证据证明软件单元设计满足分配的软件要求且适合于实施； b）验证按照 7.4.10 和 7.4.11 实施的安全分析得出的安全措施得到适当实施； c）提供证据证明所实现的软件单元符合单元设计，并满足根据所需的 ASIL 等级分配的软件要求；及 d）提供充分证据，证明软件单元不包含与功能安全相关的非预期功能和特性。	——软硬件接口规范（细化的），本文件 6.5.2； ——软件架构设计规范，本文件 7.5.1； ——软件单元设计规范，本文件 8.5.1； ——软件单元实现，本文件 8.5.2； ——配置数据，本文件 C.5.3，如果适用； ——标定数据，本文件 C.5.4，如果适用； ——安全分析报告，本文件 7.5.2； ——软件开发环境文档，本文件 5.5.1。	9.5.1 软件验证规范； 9.5.2 软件验证报告（细化的）。
10 软件集成和验证	此子阶段的目的是： a） 定义集成步骤并集成软件要素，直至嵌入式软件完全集成； b） 验证由软件架构层面的安全分析得出的已定义的安全措施得到适当实施； c） 提供证据证明集成的软件单元和集成的软件组件符合软件架构设计的要求；及 d） 提供充分证据，证明集成软件不包含与功能安全相关的非预期功能和特性。	——软硬件接口规范（细化的），本文件 6.5.2； ——软件架构设计规范（本文件 7.5.1）； ——安全分析报告，本文件 7.5.2； ——相关失效分析报告，本文件 7.5.3，如果适用； ——软件单元实现，本文件 8.5.2； ——配置数据，本文件 C.5.3，如果适用； ——标定数据，本文件 C.5.4，如果适用； ——软件开发环境文档，本文件 5.5.1； ——软件验证规范，本文件 9.5.1。	10.5.1 软件验证规范（细化的）； 10.5.2 嵌入式软件； 10.5.3 软件验证报告（细化的）。
11 嵌入式软件测试	此子阶段的目的是提供证据，证明该嵌入式软件： a)在目标环境执行时满足软件安全要求；及 b)不包含与功能安全相关的非预期功能和特性。	——软件架构设计规范，本文件 7.5.1； ——软件安全需求规范，本文件 6.5.1； ——嵌入式软件，本文件 10.5.2； ——标定数据，本文件 C.5.4，如果适用； ——软件开发环境文档，本文件	11.5.1 软件验证规范（细化的）； 11.5.2 软件验证报告（细化的）。

		5.5.1；及 ——软件验证规范（细化的），本文件 10.5.1。	
附录 C 软件配置	软件配置的目的是： a)使不同应用的软件行为变化可控； b)提供证据证明配置数据和标定数据满足所需 ASIL 等级要求；及 c)提供证据证明专用嵌入式软件及其标定数据适合生产发布。	本文件安全生命周期中与软件配置相关阶段的适用的应用前提条件。	C.5.1 配置数据规范； C.5.2 标定数据规范； C.5.3 配置数据； C.5.4 标定数据； C.5.6 验证规范（细化的）； C.5.7 验证报告（细化的）； C.5.8 软件架构设计规范（细化的）； C.5.9 软件开发环境文档（细化的）。

附录 B

（资料性）

基于模型的开发方法

B.1 目的

本附录阐述了软件层面开发过程中基于模型的开发方法的可能使用收益和潜在问题。

注：本附录并不意味着所提及的基于模型的开发方法仅局限于软件开发。

B.2 总则

本条提供了 MBD 各种用例（如要求、设计、验证/测试）共有的基本信息。第 B.3 条提供了基于示例性用例的特殊考虑。

B.2.1 引言

模型可被用来表示开发阶段所需的信息或信息视图，以及用于表示一个或多个软件要素或软件环境的抽象设计及实现。模型本身可能包含多种细化的层级或在较低层级引用细化的模型（例如，每个模型要素具有黑盒和白盒视图的层次结构）。模型是使用建模标记法来开发的。

建模标记法可以是图形和/或文本。它们可以是形式化（例如，具有基本数学定义的标记法）或半形式化（例如，非完备定义语义的结构化标记法）。建模标记法可以是国际标准（如 UML®）或公司特定的。它们一般基于如类、框图、控制图或状态表（表示状态和状态间的转换）等概念。在本附录中，只考虑了模型和建模标记法，它们包含了足够用于其预期用途的已定义的语法和语义（即，没有此类定义的示意图不被视为模型）。除了使用 MBD 的特定原因（例如，仿真或代码生成）外，特别是当不同成员协作时，充分定义的语法和语义是达成以下准则的基础，如可理解性、明确性、正确性、一致性和使用模型描述的信息或工作成果的可验证性。

除了建模标记法外，建模指南和/或合适的工具是达成充分定义的语法、语义和与符合建模指南的手段。

示例：作为指南的一部分，命名规则或风格指南支持了诸如“可理解”准则的实现，而排除模糊结构的语言子集选择同时支持了“可理解”和模型的正确转化和执行。当仿真或转化模型时，合适的仿真器或代码生成器能够补充诸如确定的执行顺序或确定的转化要求的语义，如对所提供的仿真器或代码生成器的详细行为有充分了解。

在本附录中，将进一步考虑以下基于模型的开发方法的用例：

——软件安全要求的定义（第 6 章和 GB/T 34590.8-XXXX，第 6 章）；

——软件架构设计开发（第 7 章）；

——软件单元设计与实现，有无自动代码生成（第 8 章）；

——软件组件的设计、实现和集成，包括从软件组件模型自动代码生成（第 8 章，第 9 章和第 10 章）；及

——验证（静态和/或动态）（第 9 章，第 10 章和第 11 章）。

B.2.2 通用适用性方面

可以基于以下方面评估所选的基于模型的开发方法是否适用于预期的应用程序，包括建模技术、标记法、语言或工具：

——在相应的开发阶段实现由 GB/T 34590 规定的准则（例如，可理解性、完整性、可验证性、明确性、准确性）；

——所选方法的知识和经验；

——语法和语义定义的充分性；

——应用领域的充分性（例如，实时行为、数据结构、定点与浮点微控制器软件的生成）；

——模型在软件生命周期中的作用（例如，安全要求的开发、表达架构设计的静态或动态方面）；

——通过实施抽象、封装、层次和模块化原则，对管理复杂度的支持；

——内部/外部开发方之间的协作模型（例如，数据交换期间的数据一致性）；

——对软件工具所需和可用的置信度（本文件 GB/T 34590.8-XXXX，第 11 章）；或

——作为选定的 MBD 方法的一部分或包含在 MBD 环境中的软件组件(例如，软件库)的鉴定(本文件 GB/T 34590.8-XXXX，第 12 章)。

B.2.3 MBD 对软件生命周期的潜在影响

一个模型可以代表本文件的多个工作成果（例如，要求、架构设计、详细设计以及基于模型的包含代码生成的软件集成）。与将生命周期数据分开的传统开发过程相比，可出现“软件安全要求”、“软件架构设计”、“软件单元设计和实现”等阶段更紧密的结合。这种方法的潜在益处（例如，连续性、跨软件生命周期的信息共享、一致性）极具吸引力，但这种方法也可引入导致系统性故障的问题（本文件 B.3）。

B.3 示例软件开发用例的特殊注意事项

本节基于示例用例提供与益处和潜在问题相关的注意事项。

B.3.1 软件安全要求的定义（第 6 章和 GB/T 34590.8-XXXX，第 6 章）

模型可用于捕获在安全要求级别实现的软件的功能、行为、设计或特性（例如，开/闭环控制、受控系统、状态和转换、监控功能或独立特性），并通过仿真或形式化的方法来进行要求的验证和确认。

模型可以是实现安全要求特性的合适方法，也是满足如 GB/T 34590.8-XXXX，第 6 章所述的使用半形式或形式记法来要求的方法。

示例：当使用基于模型的方法定义状态机时，相应的模型以直观、明确和可理解的方式捕获所需的状态和（状态）转换。事实上，与相同的状态机的文本规范相比，这样的模型表示了一个以上的原子要求，这是可以接受的。一个适当的理由是：该模型能在不重复信息的情况下更好地实现和保持规范的完整性、可验证性和内部一致性。

但是，请注意，通过使用要求模型，有可能会遗漏所选的基于模型的开发方法不能表示的安全要求。因此，GB/T 34590.8-XXXX 第 6 章强调了文本要求和其他记法的适当组合。

示例：与针对丢失或错误输入的软件功能的鲁棒性相关的要求。

B.3.2 软件架构设计的开发（第 7 章）

模型可用于按照多个视角来捕获软件架构，特别是：

a) 静态方面（例如，组件、其接口和数据流、组件属性）；及

b) 动态方面（例如，事件的功能链、调度、消息传递）。

模型和建模指南可以是实现安全相关架构设计的特性和原则的合适方法，也是满足第 7 章所述的半形式或形式记法要求的方式。

但是，请注意，一种建模方法可能无法表示所需的所有方面（例如，针对实现软件功能的模型通常无法描述包括基础软件或操作系统要素在内的总体架构设计），并且如果没有进一步的文本解释（例如，由不同的合作方提供），模型可能无法被完全理解。

B.3.3 软件单元的设计和实现（第 8 章和第 9 章）

模型，连同建模和编码指南，可用于在更高抽象层次上的详细设计和软件单元本身的开发。通过使用 MBD，可以从模型中通过代码生成功能直接生成可执行代码。

MBD 通过形式化方法、模拟仿真、静态分析或模型在环/软件在环测试来实现自动化一致性检查（例如，数据类型、使用前数据的正确定义）和验证。与手动编码相比，模型自动生成代码可能降低编码错误的风险，这取决于代码生成器的置信度（本文件 GB/T 34590.8-XXXX，第 11 章）。

但是，请注意，例如：

——如果没有以其他适当的形式记录，则可能会遗漏不能用建模标记法表示的设计方面。如果所有的测试都是从这样一个“不完整”的模型中派生出来的，即使是单元测试（例如，使用背靠背测试）也可能无法揭示出这一点；

——如果只对预期的功能建模，则可能无法实现安全相关的代码特性（如鲁棒性）。代码特性通常是通过模型特性和代码生成特性的适当组合来实现的（例如，如果除法是由代码生成器“按原样”生成的，那么将在模型级别检查避免除数为零。如果代码生成器使用鲁棒设计编码模式处理除数为零，则可以减轻模型中除数为零的分析）；

——错误的寻址或数字精度方面的处理可能会导致故障（例如，如果模型只是从控制工程中继承的）。例如，由控制工程师的计算机环境和目标环境的不同精度特性引起的问题；

——当从连续模型创建软件行为下的离散模型时（例如，值和/或时间离散化的影响、计算的执行时间、处理并发任务），可能会引入故障；

——仿真或验证结果（例如，一致性检查的结果）可能依赖于工具，因为不同的仿真引擎或模型检查器可能会偏离已实现的语义（例如，对于建模语言的语义未涉及的算法方面）；

——自动编码（执行时间和内存消耗）情况下的资源使用可能不同于手动编码情况下的资源使用。此外，资源优化的手段可能更加有限；

——在自动编码的情况下，可能更难验证所生成的源代码等效于手写代码。

B.3.4 软件组件的设计、实现和集成，包括从软件组件模型自动生成代码(第 8、9 和 10 章)

模型，连同建模和编码指南，可用于在更高抽象层面上的软件设计、实现和集成。因此，软件组件甚至是整个嵌入式软件都可能通过集成的模型生成自动代码的方式实现。

这种方法意味着不同的软件开发阶段的紧密结合，因此，需要仔细评估这种方法的益处和潜在问题(本文件上面提供的提示)，包括实施适当的安全措施(例如，使用单独的测试模型来派生额外的测试用例)。

B.3.5 验证(静态和/或动态)(第 9、10 和 11 章)

模型可以用来验证工作成果(例如，设计模型、代码)。这样的模型叫“验证模型”。例如，验证模型能用作背靠背测试、生成测试用例或表示与安全性相关的特性的基础，也能作为形式验证的参考(这需要验证的模型也是形式化的)。

模型还可以用作启用或支持验证活动的手段(例如，通过模拟被测设备的环境来建立闭环控制的硬件在环测试所需的对象模型)。

使用模型进行验证可以实现工作成果(包括软件)中的故障/失效的早期和有效的检测、更有效的测试用例生成、高度自动化测试甚至是形式验证技术。

但是，请注意，例如：

——如果缺乏证据，证明模型适合于验证目的，相关结果的有效性可能会受到质疑(例如，不适当的对象模型可能导致不正确的测试结果)；

——对于有效的结果，验证模型的成熟度与验证对象的目标成熟度相匹配；

——从同样的模型(也用于代码生成)产生的测试用例不能作为验证模型本身和从模型生成的代码的唯一来源。

附录 C
（规范性）
软件配置

C.1 目的

软件配置的目的是：

a）使不同应用的软件行为变化可控；

b）提供证据证明配置数据和标定数据满足所需的 ASIL 等级要求；及

c）提供证据证明专用嵌入式软件及其标定数据适合生产发布。

C.2 总则

软件配置允许使用配置数据和标定数据开发特定应用的嵌入式软件的变体（本文件图 C.1）。

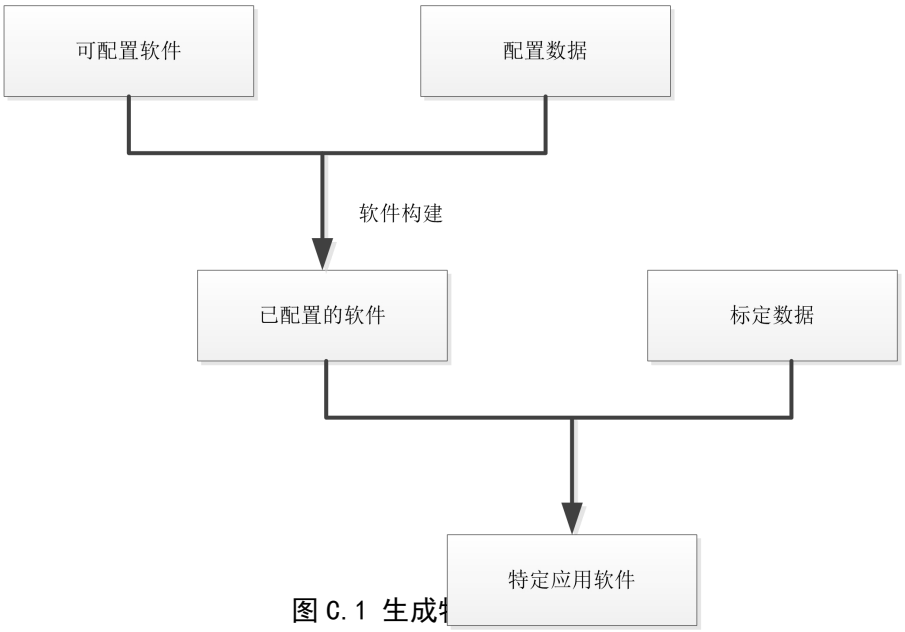


图 C.1 生成特

C.3 本章输入

C.3.1 前提条件

前提条件按照应用软件配置的相关阶段。

C.3.2 支持信息

本文件应用软件配置的相关阶段中的适用的支持信息。

C.4 要求和建议

C.4.1 为确保在安全生命周期中可配置软件的正确使用，应对配置数据进行定义，应包含以下内容：

a）配置数据的有效值；

b）配置数据的目的和用法；

c）范围、比例、单位；

d）配置数据不同要素之间的相互依赖。

C.4.2 配置数据及其规范应按照 GB/T 34590.8-XXXX 第 9 章提供证据证明：

a）配置数据符合按照 7.5.1 的软件架构设计规范和按照 8.5.1 的软件单元设计规范；

b）使用的值在其规定的范围内；及

c）与其他配置数据的兼容性。

注：对配置软件的测试在软件生命周期的测试阶段内执行（本文件第 9 章、第 10 章、第 11 章和 GB/T 34590.4-XXXX 第 7 章）。

C.4.3 配置数据的 ASIL 等级应等于应用于该数据的可配置软件的最高 ASIL 等级。

C.4.4 应按照 GB/T 34590.8-XXXX 第 9 章，针对所考虑的相关项开发中将要使用的配置数据集对可配置软件进行验证。

注：只有其行为依赖于配置数据的那部分嵌入式软件要针对配置数据集进行验证。

C.4.5 对于可配置软件，可按照图 C.2 或 C.3 所示的简化的软件安全生命周期应用。

注：下列验证活动的组合能实现对已配置软件的完整验证：

a） 可配置软件的验证；

b） 配置数据的验证；及

c） 已配置软件的验证。

可通过如下任意一种方式实现：

——在 a）中验证允许的配置数据的范围，并说明它符合在 b）中定义的范围；或

——说明与 b）中允许的配置数据范围的符合性，并执行 c)。

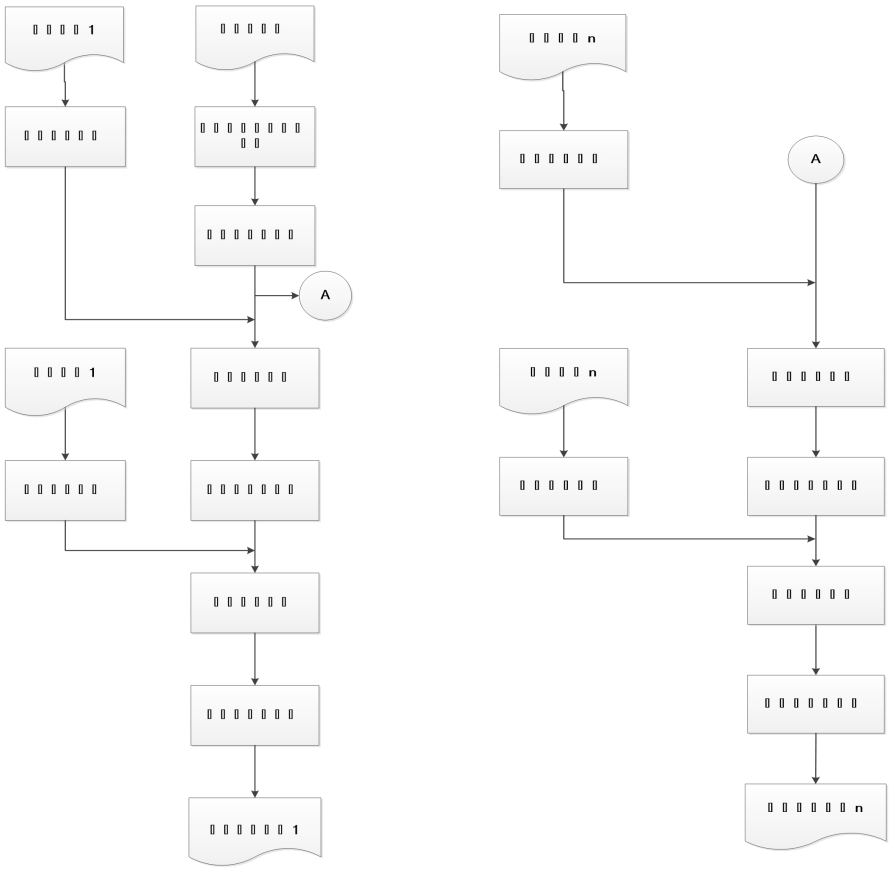


图 C.2 可配置软件和不同配置数据及不同标定数据的软件开发的参考阶段模型的变型

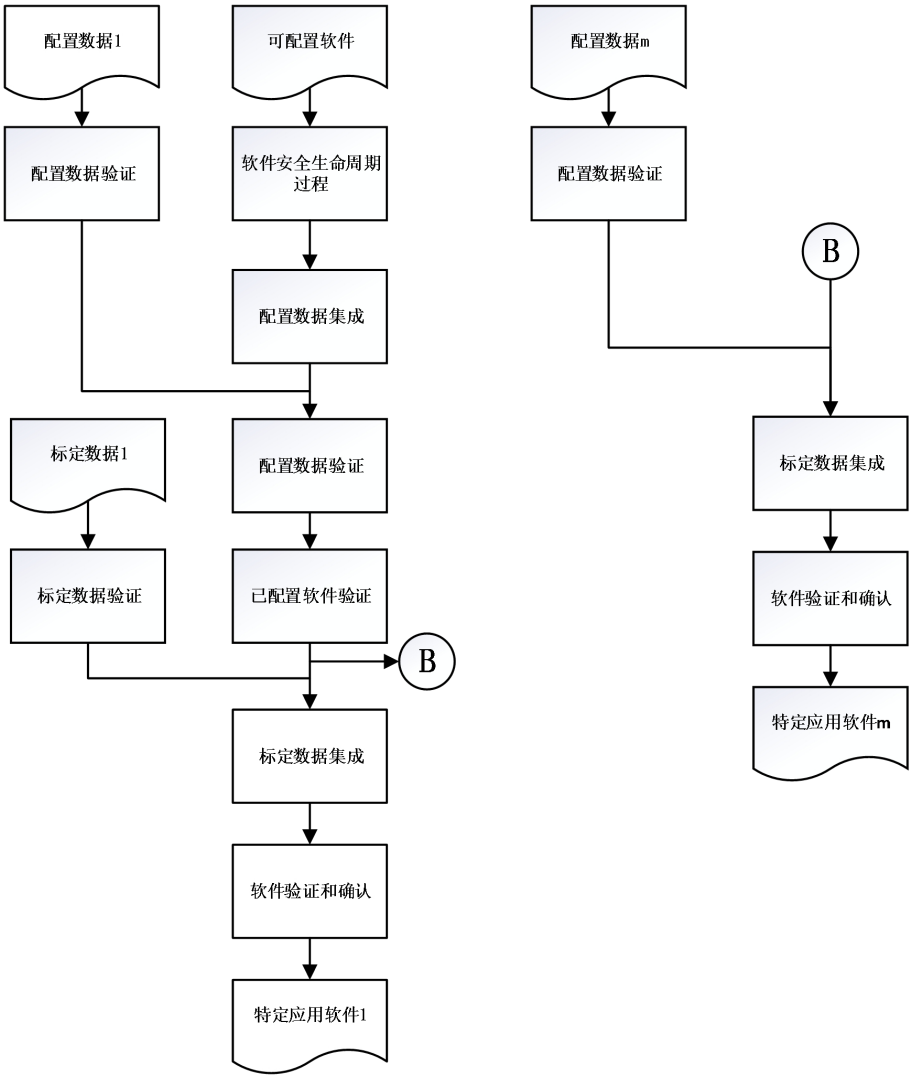


图 C.3 可配置软件和相同配置数据及不同标定数据的软件开发的参考阶段模型的变型

C.4.6 应定义与软件组件关联的标定数据以确保配置后软件的正确运行和预期性能，应包含：

a) 标定数据的有效值；

b) 标定数据的目的和用法；

c) 范围、比例和单位，以及它们对运行状态的依赖（如果适用）；

d) 不同标定数据之间已知的相互依赖；及

注 1：相互依赖可存在于同一标定数据集内的标定数据之间，或者不同标定数据集的标定数据之间，如：不同电子控制单元的软件中执行的应用于相关功能的标定数据。

e) 配置数据和标定数据之间的已知的相互依赖。

注 2：配置数据可对使用标定数据的已配置软件有影响。

C.4.7 标定数据的 ASIL 等级应等于其可能违反的软件安全要求的最高 ASIL 等级。

C.4.8 应按照 GB/T 34590.8—XXXX，第 9 章验证标定数据的定义，以提供证据证明：

a) 定义的标定数据合适并符合软件安全要求 6.5.1；

b）定义的标定数据符合软件架构设计规范 7.5.1 和软件单元设计规范 8.5.1；及

c）定义的标定数据与其他定义的标定数据是一致且兼容的，以防止非预期影响。

C.4.9 用于生产发布的标定数据应按照 GB/T 34590.8—XXXX，第 9 章进行验证，以提供证据证明：

a）发布的标定数据符合其规范（本文件 C.4.6）；及

b）嵌入式软件的已标定的、应用特定的变体提供了定义的安全相关功能和特性。

注：标定数据的验证也可以在系统层面执行。

C.4.10 为探测安全相关的标定数据的非预期变化，应使用表 C.1 中所列的数据非预期变化的探测机制。

表 C.1 数据非预期变化的探测机制

方法		ASIL 等级			
		A	B	C	D
1a	标定数据合理性检查	++	++	++	++
1b	标定数据的冗余存储和比较	+	+	+	++
1c	使用错误检测码检查标定数据[a]	+	+	+	++
[a] 错误检测码也可按照 GB/T 34590.5-XXXX 在硬件中实现。					

C.4.11 标定数据的生成和应用应定义如下：

a）应遵循的流程；

b）生成标定数据的工具；及

c）验证标定数据的流程。

注：标定数据的验证可包含检查标定数据值的范围或者不同标定数据之间的相互依赖关系。

C.5 工作成果

C.5.1 配置数据规范，由 C.4.1～C.4.4 的要求得出。

C.5.2 标定数据规范，由 C.4.6～C.4.8 的要求得出。

C.5.3 配置数据，由 C.4.2～C.4.4 的要求得出。

C.5.4 标定数据，由 C.4.7～C.4.10 的要求得出。

C.5.5 验证规范（细化的），由 C.4.2、C.4.4、C.4.5、C.4.7、C.4.8、C.4.10 和 C.4.11 的要求得出。

C.5.6 验证报告（细化的），由 C.4.2、C.4.4、C.4.5、C.4.7、C.4.8、C.4.10 和 C.4.11 的要求得出。

C.5.7 软件架构设计规范（细化的），由 C.4.10 的要求得出。

C.5.8 软件开发环境文档（细化的），由 C.4.1～C.4.11 的要求得出。

附录 D

（资料性）

避免软件要素间的干扰

D.1 目的

本附录的目的是提供能引起软件要素（例如：不同软件分区中的软件要素）间干扰的故障示例，此外，本附录也提供一些可用于防止或探测并减轻所列出的故障的处理机制的示例。

注：在开发过程中评估用于防止或探测并减轻相关故障的处理机制的能力和有效性。

D.2 总则

D.2.1 免于干扰的实现

为了开发或评估软件要素间免于干扰的实现，可考虑典型故障的影响及其可能导致的失效的传播。

D.2.2 时序和执行

关于时序限制，对于每个软件分区中执行的软件要素可考虑下列故障的影响，例如：

——执行受阻；

——死锁；

——活锁；

——执行时间的不正确分配；或

——软件要素间的不正确同步。

示例：可考虑的处理机制例如：循环执行调度、固定优先级调度、时间触发调度、处理器执行时间监控、程序执行次序监控和到达率监控。

D.2.3 存储

关于存储，对于每个软件分区中执行的软件要素可考虑下列故障的影响，例如：

——内容损坏；

——数据不一致（例如，由于数据获取期间发生更新）；

——堆栈上溢或下溢；或

——对已分配给其他软件要素的内存进行读或写访问。

示例 1：可使用的安全措施如：存储保护、奇偶校验位、纠错码（ECC）、循环冗余校验（CRC）、冗余存储、内存访问限制、内存访问软件的静态分析、内存静态分配。

示例 2：适当的验证方法可以被视为详细的安全分析，以识别需要采取保护机制的关键存储。静态和语义代码分析、控制流和数据流的分析结果可以提供证据证明免于干扰。

D.2.4 信息交换

关于信息交换，针对每个发送方或接收方，可考虑如下所列各故障的原因或故障的影响：

——信息重复；

——信息丢失；

——信息延迟；

——信息插入；

——信息伪装或信息的不正确寻址；

——信息次序不正确；

——信息损坏；

——从发送方传送到多个接收方的信息不对称；

——发送方发送的信息只能被部分接收方接收；或

——通信信道阻塞。

注：在不同软件分区或不同 ECU 中执行的要素间的信息交换包括信号、数据、消息等。

示例 1：可使用 I/O 设备、数据总线等方式进行信息交换。

示例 2：可使用的处理机制如：通信协议、信息重复、信息回送、信息确认、I/O 引脚的适当配置、分离的点到点的单向通信对象、明确的双向通信对象、异步数据通信、同步数据通信、事件触发数据总线、带有时间触发访问的事件触发数据总线、时间触发的数据总线、最小时间片和基于优先级的总线仲裁。

示例 3：通信协议可包含的诸如通信对象的标识符、保持活动的消息、活动的计数器、序列号、错误检测码和纠错码的信息。

示例 4：适当的验证方法可以被视为详细的安全分析，以识别需要采取保护机制的关键数据交换。静态和语义代码分析以及控制流和数据流分析的结果能提供证据证明免于干扰。

附录 E

（资料性）

软件架构层级安全分析及相关失效分析的应用

E.1 目的

本附录阐述了安全分析及相关失效分析在软件架构层级可能的应用。本附录中提供的示例旨在支持对这些分析的理解，并为应用提供指导。

E.2 总则

E.2.1 分析的范围和目的

通过在软件架构层级应用安全分析或相关失效分析，可以检查或确认嵌入式软件按照所分配 ASIL 等级的完整性要求提供指定功能、行为和特性的能力。

嵌入式软件按照所分配 ASIL 等级的完整性要求提供指定的功能和特性的适用性，通过以下方式进行检查：

——识别可引发因果链而导致违反安全要求的设计缺陷、条件、故障或失效（例如，使用归纳或演绎的方法）；及

——分析可能出现的故障、失效或因果链对软件架构要素所需的功能和特性的影响。

相关的软件架构要素之间所达到的独立性或免于干扰的程度，可通过分析如下软件架构设计的相关失效来检查，即：

——可能导致多个且相互独立的软件要素功能异常表现的单一事件、故障或失效（例如，级联失效和/或共因失效，包括共模失效）；及

——可能从一个软件要素传播到另一个软件要素，引发因果链而导致违反安全要求的单一事件、故障或失效（例如，级联失效）。

由于以下原因，可能需要实现软件架构要素之间的独立性或免于干扰：

——ASIL 等级分解在软件层面的应用（本文件 6.4.3）；

——软件安全要求的实现（本文件 7.4.11），例如，提供软件安全机制有效性的证据，例如被监控要素与监控器之间的独立性；或

——所需软件架构要素的共存（本文件 7.4.6、7.4.8 和 7.4.9 和 GB/T 34590.9-XXXX，第 6 章）。

软件架构层级的安全分析及相关失效分析的作用是支持设计规范和设计验证活动。

此分析也可以揭示给定安全要求的不完整性或不一致性。

软件架构层级安全分析及相关失效分析的结果也是以下活动的基础：

——产品中有效安全机制的定义与实施；或

——开发期间适当的安全措施的确定，以适用于预防或探测并控制在这些分析过程中识别出的相关故障或失效。

图 E.1 说明了软件架构层级的安全导向分析在软件开发中的作用，以及软件安全要求、软件架构设计和安全计划间基本的交互关系。

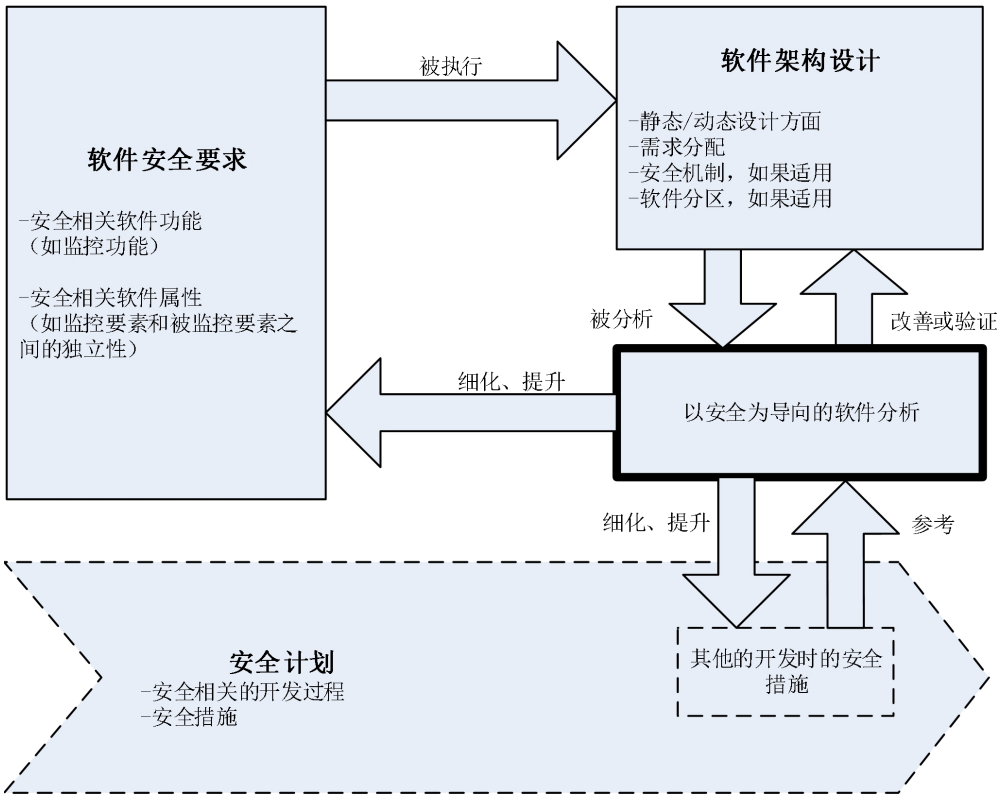


图 E.1 安全导向软件分析在开发过程的作用图示

示例：图 E.2 说明了共享资源（例如，共享的处理要素）使用时的冲突造成的干扰。如图 E.2 所示，一个 QM 软件要素干扰并阻止了 ASIL 等级软件要素的及时执行。这样的干扰也可能发生在不同 ASIL 等级的软件组件之间。此图说明了有无免于干扰机制对软件执行的影响。通过在软件中引入“检测点”并对检测点进行超时监控，可探测到时序干扰并触发合适的响应。

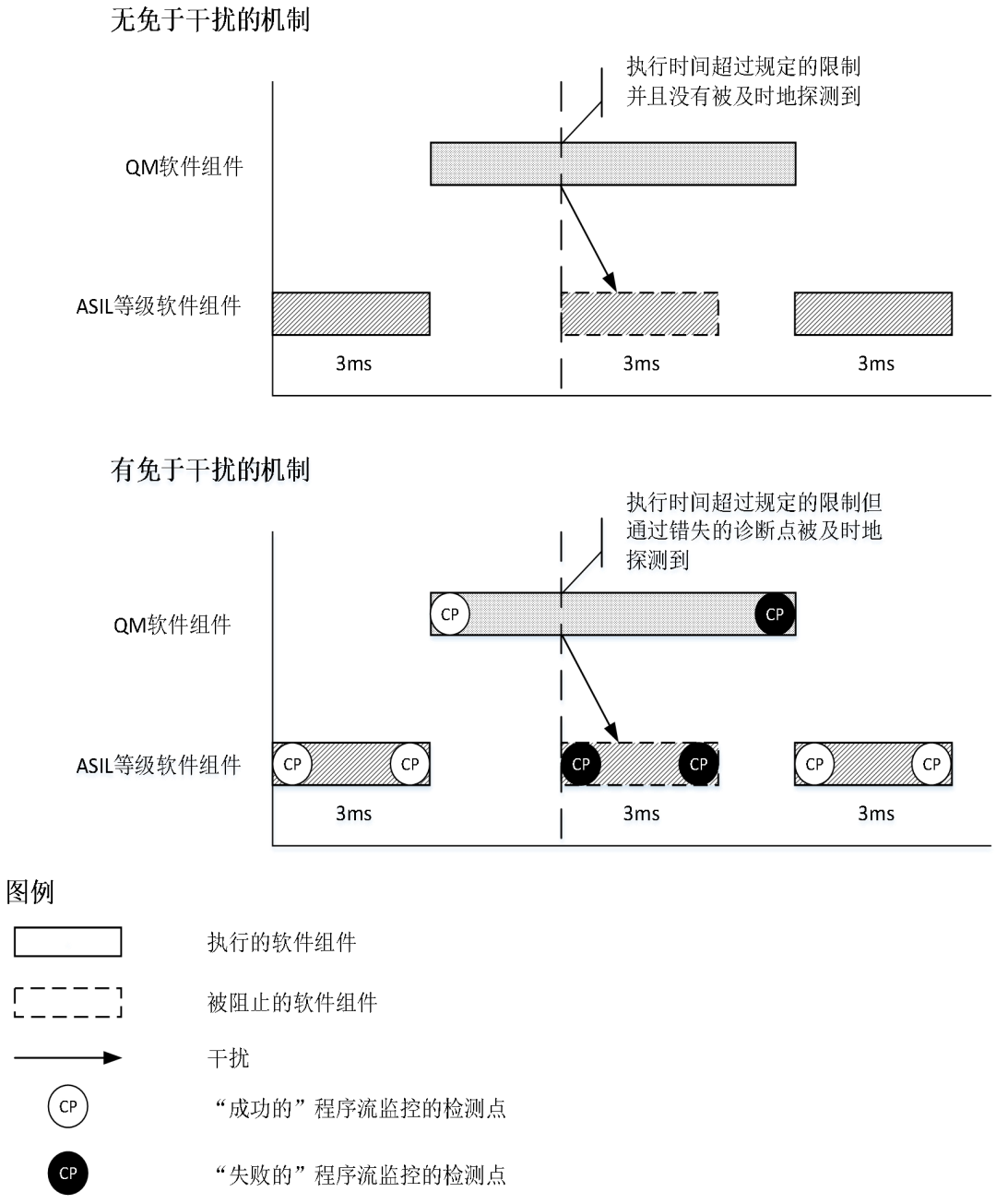


图 E.2 时序干扰导致级联失效的示例

安全分析及相关失效分析旨在应用于软件架构层级。而代码层面的安全分析（例如，针对运行时的错误，如被零除，超出数组索引边界的访问）既不需要也不认为必要的原因如下：

——当软件开发过程严格采用了本文件的要求，这类故障的残余风险可以认为足够低。包括选择和使用适当的方法、方式和开发原则，如建模和编程指南、设计和实施规则、模型和/或代码审核、语义或静态分析、单元和集成测试；及

——在实现相应架构要素时，这类故障通常在要素接口上引起和架构层面分析时同样的功能异常表现。因此更高层面的分析已经提供了证据，证明所选择的技术安全机制是有

效的，或者开发期间的安全措施是充分的。

E.2.2 安全分析及相关失效分析在系统、硬件和软件层面的关系

在开发生命周期的多个阶段，使用了安全分析及相关失效分析。这些分析可以是互相关联的。

在系统层面启动的关注点分离方法中，通常由硬件开发负责确保处理单元具有足够低的硬件故障残余风险。因此，在执行软件架构层级的安全分析及软件相关失效分析时可不考虑硬件故障。

然而，在某些硬件条件下（例如，处理单元的包括瞬态故障随机硬件故障的诊断覆盖率不足的情况），考虑硬件故障的负面影响可能更为合适。在这种情况下，针对特定硬件/软件交互的特定分析应参考以下示例：

——映射到多核系统的不同处理单元的特定软件；

——和静态/动态软件架构设计相关的处理元件的详细设计（软件在处理元件上执行）；或

——针对软件相关的硬件故障所选择的软件安全机制可实现的诊断覆盖率。

注：可通过故障注入技术来支持特定硬件/软件交互分析（本文件 GB/T 34590.11-XXXX, 4.8）。

E.2.3 分布式软件开发（包含 SEooC 软件要素的开发）的安全分析

嵌入式软件和其软件要素通常由多个组织（包括 OEM、Tier 1、Tier 2、半导体供应商和软件供应商）通过分布式开发完成，甚至作为独立于环境方式开发（例如，作为 SEooC 开发的基础软件、硬件相关的软件、商业现成软件或操作系统）。

分布式软件开发中有意义的以安全为导向的分析应包括以下要点：

——分析范围、分别或联合进行分析的流程或方法的定义和协议（包括信息/文档交换或假设确认）；

——关于软件安全分析过程的总体职责和参与组织的通用规则的定义和协议（例如，高层级组件及其之间的访问规则）；

——使用模块化方法进行分析时接口的定义和协议（例如，不同范围之间接口达成一致的软件故障模型、使用的方法、信息/文档的交换）；及

——分析验证的定义和协议。

E.3 执行分析的通则

软件架构设计提供了支持实现功能安全和论证达成功能安全的高层次规则和决定，例如：组件的功能安全分类，组件间的访问规则。

达成有意义分析的通则需要确定：

——分析的目标，见 E.3.1；

——范围和边界，见 E.3.2；

——用于执行分析的方法，见 E.3.3；及

——支持分析的意义性、客观性和严谨性的方法或准则，见 E.3.4。

E.3.1 确定分析目标

目标由第 7 章和 GB/T 34590.9-XXXX 的第 7 章和第 8 章提供。

示例：鲁棒性或确定性执行的分析中，违背分配给软件的安全相关功能或特性。

E.3.2 确定待分析的特定架构设计的范围和边界

根据分析的目标，确定范围并着重于架构设计的相关部分。

分析范围可受以下因素影响：

——在 DIA 中定义的与交付相关的范围和相应职责；

——分析的特定目标；

——基于“良好”设计原则的架构策略（本文件 7.4.3）；或

——架构设计所需特性，源于更高层面安全概念中免于干扰或充分独立性的要求。

示例1：当在安全分析中考虑与外部发送方或接收方交换安全相关数据时，合适的端到端数据保护机制可能被用作论据，证明基础软件可以被视为“黑盒”。

示例2：带运行模式软件功能间的依赖性，例如，某个安全相关功能仅在其他“关键”功能被安全模式切换功能关闭时才可以被“激活”。

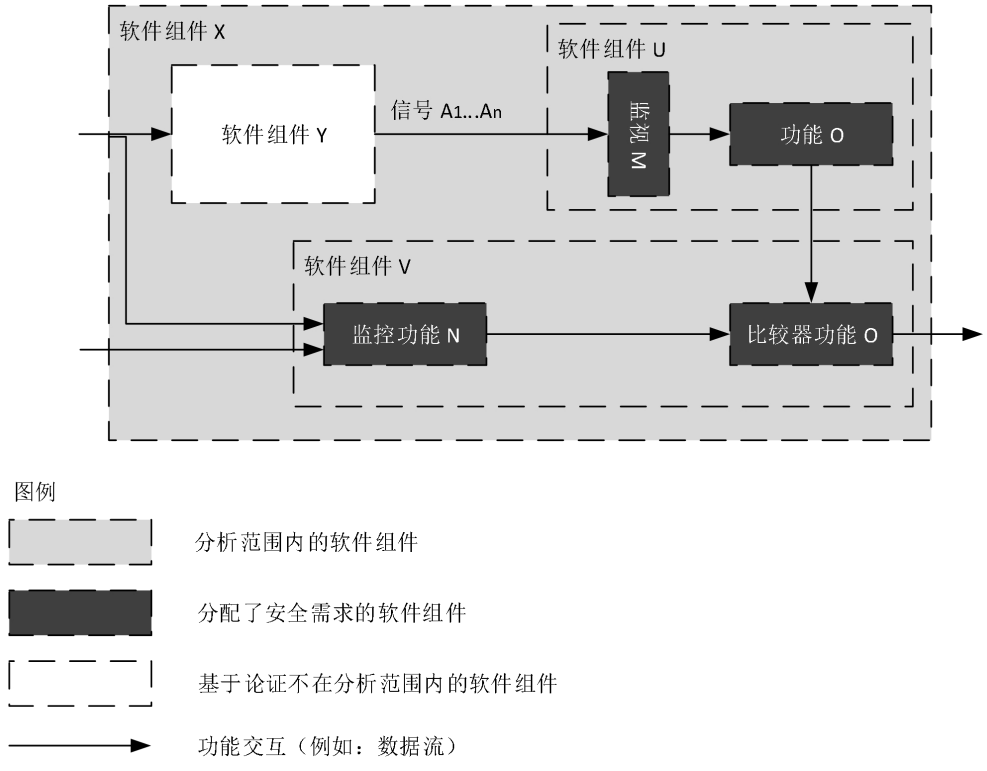


图 E.3 关于安全论证的分析范围

示例3：图E.3展示了要分析的软件，按照安全概念，安全相关功能仅在软件组件U中实现，并由软件组件V中实现的监控功能所控制。架构分析范围的确定阐明了所需的安全论证，并支持下述内容中的后续步骤。在这个示例中，仅当对软件组件X、软件组件V和软件组件U的分析能够确定免于干扰时（例如，在监视功能M中没有因错误的信号A1至An所导致的级联失效），软件组件Y才可以免除进一步的详细分析）。

E.3.3 确定分析所采用的方法

由于软件的特殊性质（例如，没有因磨损或老化而导致的随机故障，也缺少成熟的概率方法），在系统或硬件层面建立的分析方法通常不能未经修改就应用于软件，否则无法得出有意义的结论。

安全分析方法的共同特征是，它们通常包括一些手段，用于实施结构化方法，储备所获知识和得出结论。

示例：在故障树、功能网、要素列表或FMEA的风险优先级编号中用于描述故障及其相关性的语法和语义。

软件架构设计是安全分析和相关失效分析的基础，其描述相关软件的静态方面（例如，显示功能要素及其接口/关系的框图表示）以及动态方面（例如，顺序图、时序图、调度图或状态图表示）。

按照软件架构设计的安全分析和相关失效分析可以遵循功能和/或处理链，同时考虑静态和动态方面。在此类分析过程中，可以使用软件故障模型或问题，例如，“源自或影响此软件组件的何种特定故障或失效可能导致违背指定的安全要求”，来识别设计中与安全相关的薄弱点。

软件架构设计的细化可以支持充分详细的安全性分析和相关失效分析，以获取足够详细信息，从而识别较低级别的故障或失效。

这种分析的详细程度与以下方面有关：

——细化架构设计时范围的适当选择；

——支持实现分析的特定目标的特性；

——基于“良好”设计原则支持的架构策略的论据（本文件 7.4.3）； 或

——由包含达到的免于干扰的充分独立性支撑的论据。

E.3.4 支持有意义、客观、严谨的分析需要考虑的方面

E.3.4.1 按照GB/T 34590.9-XXXX 相关失效分析的耦合因素类别

在 GB/T 34590.9-XXXX 附录 C 中，定义了相应的软件层面的耦合因素类型示例，可用于改进要求独立性的软件在功能或处理链上执行相关失效分析的结果。

E.3.4.2 使用附录D中的内容

附录 D 提供了在分析软件要素间的干扰时需考虑的软件故障模型，例如关于:

——时序和执行；

——内存；或

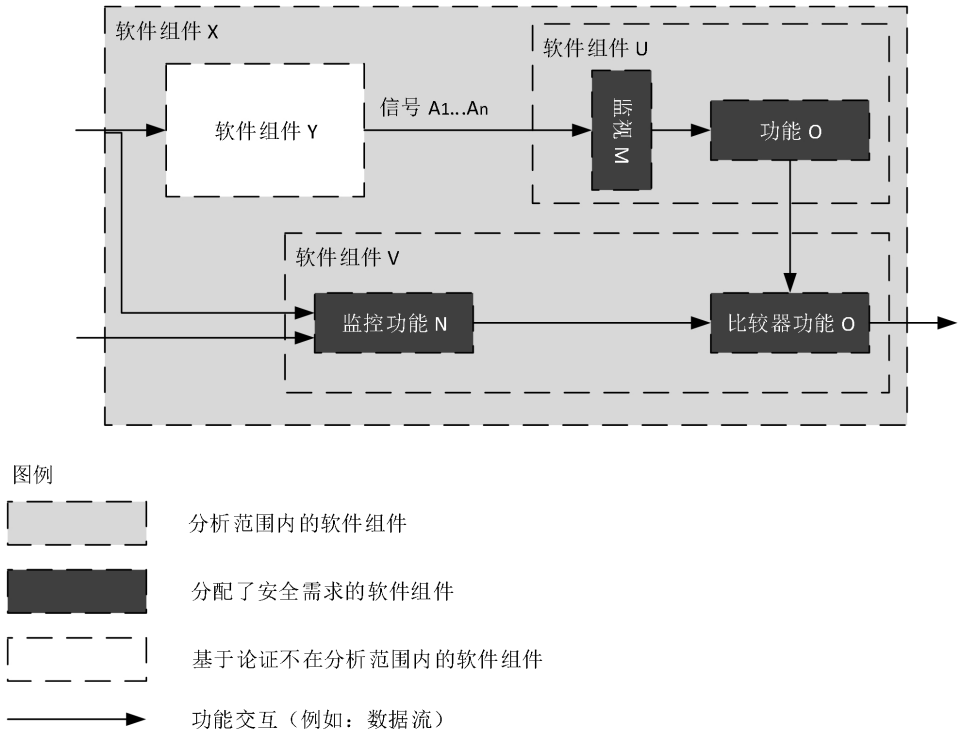
——信息交换。

E.3.4.3 使用引导词的分析

引导词可用于系统地检查与特定设计意图的可能偏差，并确定其可能的结果。在安全导向的分析过程中，使用引导词生成问题去检查架构要素的特定功能或特性。当使用引导词时，使用引导词重复执行每个设计要素的特定功能或特性的安全导向分析，直到预定的引导词都被检查过。因此，引导词是系统地进行这种分析并支持完整性论据的一种手段。

数据流或类似的图表通常用于描述软件架构设计的功能方面（例如，功能或处理链）。图 E.4 显示了软件组件 Y、U 和 V 以及相关功能或处理链的交互（例如，基于数据流）。

引导词可以用来识别薄弱点、故障和失效。选择合适的引导词取决于接受检查的功能、行为、特性、接口或交换数据的特征。



图E.4 软件架构设计框图

以设计意图为基础，通过综合设计属性、相关引导词及其解释，获得引导词，以达到对分析的共识。表 E.1 展示了应用于图 E.4 设计中引导词选择的示例。

表E.1 与软件执行、调度或通讯相关的引导词选择的示例

被检查的功能或特性	引导词示例	解释	附加参考
信号流A_1到A_n	在..之后/晚于	信号过晚或顺序混乱	附录D描述了相关事项： D.2.4 信息延迟； D.2.4 错误的信息顺序； D.2.4 通讯通道访问阻塞；

			D.2.2 运行阻塞； D.2.2 死锁； D.2.2 活锁； D.2.2 运行时间的错误分配； D.2.2 软件组件间的错误同步。
	在..之前/早于	信号过早或顺序混乱	附录D描述了相关事项： D.2.4 错误的信息顺序； D.2.2 软件组件间的错误同步。
	无	没有信号	附录D描述了相关事项： D.2.2 执行阻塞； D.2.2 死锁； D.2.2 活锁； D.2.2 软件组件间的错误同步。
信号值A_1到A_n	多于	信号值高于允许范围	——
	少于	信号值低于于允许范围	——

在分析过程中，可以在使用演绎法或归纳法时应用引导词。在分析过程中，如果需要的话，可以进一步细化引导词（例如，为了更好地匹配特定的设计方面）。

在分析过程中，按照所选择的方法（例如，后续功能或处理链）使用引导词来生成问题，使用这些问题检查设计并揭示可能的薄弱点及其后果。表 E.2 所示为信号 A1 到 An 的由引导词支持的分析示例。

表E.2 引导词支持的分析示例

引导词示例	解释	后果	安全措施	所需活动
多于	信号值超出允许范围	功能O将产生错误输出	监视M限制信号A_1到A_n至允许的最大范围	实现监视M
少于	信号值低于允许范围	功能O将产生错误输出	监视M限制信号A_1到A_n至允许的最小范围	实现监视M
不同于	信号A_1到A_n的值不一致	功能O将产生错误输出	监视M使用物理依赖检查信号A_1到A_n的一致性	实现监视M

E.4 安全及相关失效分析中识别的薄弱环节的减轻危害策略

软件架构层级的安全分析和相关失效分析结果允许选择适当的安全措施，以防止或探测和控制可能违背安全要求或安全目标的故障和失效。

安全措施确定策略可以基于以下考虑：

——每个已识别故障的关键性，基于其是否违背分配给架构要素的安全目标或安全要求以及其所分配的 ASIL 等级；

——改进架构设计是否能够消除已识别的薄弱环节或降低已识别故障的关键性；

——已确定的开发期间安全措施的有效性，以及这些措施是否能够基于其关键性和相

应的理由充分减轻已识别故障的危害（例如，在分析中考虑使用软件故障模型的已定义验证活动的有效性理由）；

——已确定的安全机制的有效性，用于减轻开发期间措施不够充分的已识别关键故障的影响；及

——软件架构设计的复杂性（例如，接口和软件组件的数量），或软件组件的复杂性（例如，分配要求的数量或组件规模）。

示例：对于比较复杂的软件，仅基于开发期间措施的论据是不太合适的。

按照上述策略，在运行期间并不总是需要执行能够预防或探测和控制此类故障或失效的安全机制。

参考文献

[1] ISO/IEC 12207:2008, Systems and software engineering — Software life cycle processes.

[2] GB/T 20438-2017（所有部分）,电气/电子/可编程电子安全相关系统的功能安全.

[3] MISRA C:2012, Guidelines for the use of the C language in critical systems, ISBN 978-1-906400-10-1, MIRA, March 2013.

[4] MISRA AC GMG.Generic modelling design and style guidelines,ISBN 978-906400-06-4, MIRA, May 2009.

[5] ISO/IEC/IEEE 29119:2013,(all parts),Software and systems engineering — Software testing.

[6] BIEMAN J.M.,DREILINGER D.,LIN L.“Using fault injection to increase software test coverage,”in Software Reliability Engineering,1996.Proceedings.,Seventh International Symposium on,vol.,no.,pp.166-174,30 Oct-2 Nov 1996 doi: 10.1109/ISSRE.1996.558776.

[7] JIA Y.,MERAYO M.,HARMAN M.2015) Introduction to the special issue on Mutation Testing.Softw.Test.Verif.Reliab.,25:461-463.

[8] ISO 26262-12:2018,Road Vehicles — Functional safety — Part 12: Adaptation of ISO 26262 for Motorcycles.