ICS 25.040
N 10

中华人民共和国国家标准

GB/T 20438.7—2017/IEC 61508-7:2010
代替 GB/T 20438.7—2006

电气/电子/可编程电子安全相关系统的功能安全　第7部分:技术和措施概述

Functional safety of electrical/electronic/programmable electronic safety-related systems—Part 7:Overview of techniques and measures

(IEC 61508-7:2010,IDT)

2017-12-29 发布　　2018-07-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　　次

前　言

GB/T 20438《电气/电子/可编程电子安全相关系统的功能安全》分为七个部分：

——第1部分：一般要求；

——第2部分：电气/电子/可编程电子安全相关系统的要求；

——第3部分：软件要求；

——第4部分：定义和缩略语；

——第5部分：确定安全完整性等级的方法示例；

——第6部分：GB/T 20438.2和GB/T 20438.3的应用指南；

——第7部分：技术和措施概述。

本部分为GB/T 20438的第7部分。

本部分按照GB/T 1.1—2009给出的规则起草。

本部分代替GB/T 20438.7—2006《电气/电子/可编程电子安全相关系统的功能安全　第7部分：技术和措施概述》，与GB/T 20438.7—2006相比，主要技术变化如下：

——增加了专用集成电路(ASIC)设计技术和措施概述(见附录E)；

——增加了软件安全生命周期各阶段属性的定义(见附录F)；

——增加了安全相关的面向对象软件的开发指南(见附录G)。

本部分使用翻译法等同采用IEC 61508-7:2010《电气/电子/可编程电子安全相关系统的功能安全　第7部分：技术和措施概述》。

本部分由中国机械工业联合会提出。

本部分由全国工业过程测量控制和自动化标准化技术委员会(SAC/TC 124)归口。

本部分起草单位：机械工业仪器仪表综合技术经济研究所、北京国电智深控制技术有限公司、北京和利时系统工程有限公司、西门子(中国)有限公司、上海中沪电子有限公司。

本部分主要起草人：史学玲、熊文泽、田雨聪、杨柳、周有铮、冯晓升、罗安、李佳、刘晓东、梅豪、方来华、徐皑冬、叶均、左信、钱大涛。

本部分所代替标准的历次版本发布情况为：

——GB/T 20438.7—2006。

引　言

由电气和电子器件构成的系统，多年来在许多应用领域中执行其安全功能。以计算机为基础的系统（一般指可编程电子系统）在其应用领域中用于执行非安全功能，并且也越来越多地用于执行安全功能。如果要安全并有效地使用计算机技术，有关决策者在安全方面有充足的指导并据此做出决定是十分必要的。

GB/T 20438 针对由电气和/或电子和/或可编程电子（E/E/PE）组件构成的、用来执行安全功能的系统安全生命周期的所有活动，提出了一个通用的方法。采用统一的方法的目的是为了针对所有以电为基础的安全相关系统提出一种一致的、合理的技术方针。主要目标是促进基于 GB/T 20438 系列标准的产品和应用领域国家标准的制定。

注 1：在参考文献中给出了基于 GB/T 20438 系列标准的产品和应用领域标准的例子（见参考文献[1]，[2]，[3]）。

在许多情况下，可用多种基于不同技术（如机械的、液压的、气动的、电气的、电子的、可编程电子的等）的系统来保证安全。因而不得不考虑各类安全策略，不仅要考虑单个系统中的所有组件的问题（如传感器、控制器、执行器等），还要考虑不同安全相关系统组合后的问题。因此当 GB/T 20438 在关注电气/电子/可编程电子（E/E/PE）安全相关系统的同时，也提供了一个框架，在这个框架内，基于其他技术的安全相关系统也可被考虑进去。

在各种应用领域里，存在着许多潜在的危险和风险，包含的复杂性也各不相同，从而需应用不同的 E/E/PE 安全相关系统。对每个特定的应用，将根据特定应用的许多因素来确定所需的安全措施。GB/T 20438 作为基本原则可在未来的产品和应用领域国家标准制定和已有标准的修订中规范这些措施。GB/T 20438

——考虑了当使用 E/E/PE 系统执行安全功能时，所涉及的整体安全生命周期、E/E/PE 系统安全生命周期以及软件安全生命周期的各阶段（如初始概念、整体设计、实现、运行和维护到退役）；

——针对飞速发展的技术，建立一个足够健全且广泛满足未来发展需求的框架；

——使涉及 E/E/PE 安全相关系统的产品和应用领域的国家标准得以制定；在 GB/T 20438 的框架下，产品和应用领域的国家标准的制定在应用领域和交叉应用领域宜具有高度一致性（如基本原理，术语等）；这将既具有安全性又具有经济效益；

——为实现 E/E/PE 安全相关系统所需的功能安全，提供了编制安全要求规范的方法；

——采用了一种可确定安全完整性要求的基于风险的方法；

——引入安全完整性等级，用于规定 E/E/PE 安全相关系统所要执行的安全功能的目标安全完整性等级；

注 2：GB/T 20438 没有规定每个安全功能的安全完整性等级的要求，也没有规定如何确定安全完整性等级。而是提供了一种基于风险概念的框架和技术范例。

——建立了 E/E/PE 安全相关系统执行安全功能的目标失效量，这些量都同安全完整性等级相联系；

——建立了单一 E/E/PE 安全相关系统执行安全功能时，目标失效量的一个下限值。这些 E/E/PE 安全相关系统运行在：

——低要求运行模式下，下限设定成要求时危险失效平均概率为 10^{-5}；

——高要求或连续运行模式下，下限设定成危险失效平均频率为 10^{-9}/h。

注 3：单一 E/E/PE 安全相关系统不一定是单通道架构。

注 4：对于非复杂系统，通过安全相关系统的设计实现更优目标安全完整性是可能的。但对于相对复杂的系统（例如可编程电子安全相关系统），这些限值代表了目前能够达到的水平。

——基于工业实践中获取的经验和判断，设定了避免和控制系统性故障的要求。即使发生系统性故障的可能性一般不能量化，但 GB/T 20438 允许为一个特定的安全功能做出声明，即如果标准中的所有要求都满足，认为与安全功能相关的目标失效量已达到；

——引入了系统性能力，该能力表明一个组件为满足规定的安全完整性等级要求时，系统性安全完整性的置信度；

——采用多种原理、技术和措施以实现 E/E/PE 安全相关系统的功能安全，但没有明确地使用失效-安全的概念。然而，如果能够满足标准中相关条款的要求，则“失效-安全”的概念和“本质安全”原则可能被应用，并且采用这些概念是可接受的。

电气/电子/可编程电子安全相关系统的功能安全　第7部分:技术和措施概述

1　范围

1.1　GB/T 20438的本部分包含了GB/T 20438.2和GB/T 20438.3有关的各种安全技术和措施的概述。

参考文献仅作为各种方法和工具或示例的基本参考,不一定代表当前技术水平。

1.2　GB/T 20438.1、GB/T 20438.2、GB/T 20438.3和GB/T 20438.4是基础的安全标准,虽然它不适用于低复杂的E/E/PE安全相关系统(见GB/T 20438.4—2017的3.4.3),但作为基础安全标准,各技术委员会可以在IEC指南104和ISO/IEC指南51的指导下制定相关标准时使用。GB/T 20438.1、GB/T 20438.2、GB/T 20438.3和GB/T 20438.4也可作为独立标准来使用。GB/T 20438的横向安全功能不适用于在IEC 60601系列指导下的医疗设备。

1.3　各技术委员会的责任之一,是在其标准的起草工作中尽可能使用基础的安全标准。在本部分中,本基础安全标准中的要求、测试方法或测试条件只有在这些技术委员会起草的标准中已明确引用或包含时适用。

1.4　图1表示了GB/T 20438的整体框架,同时明确了本部分在实现E/E/PE安全相关系统功能安全过程中的作用。

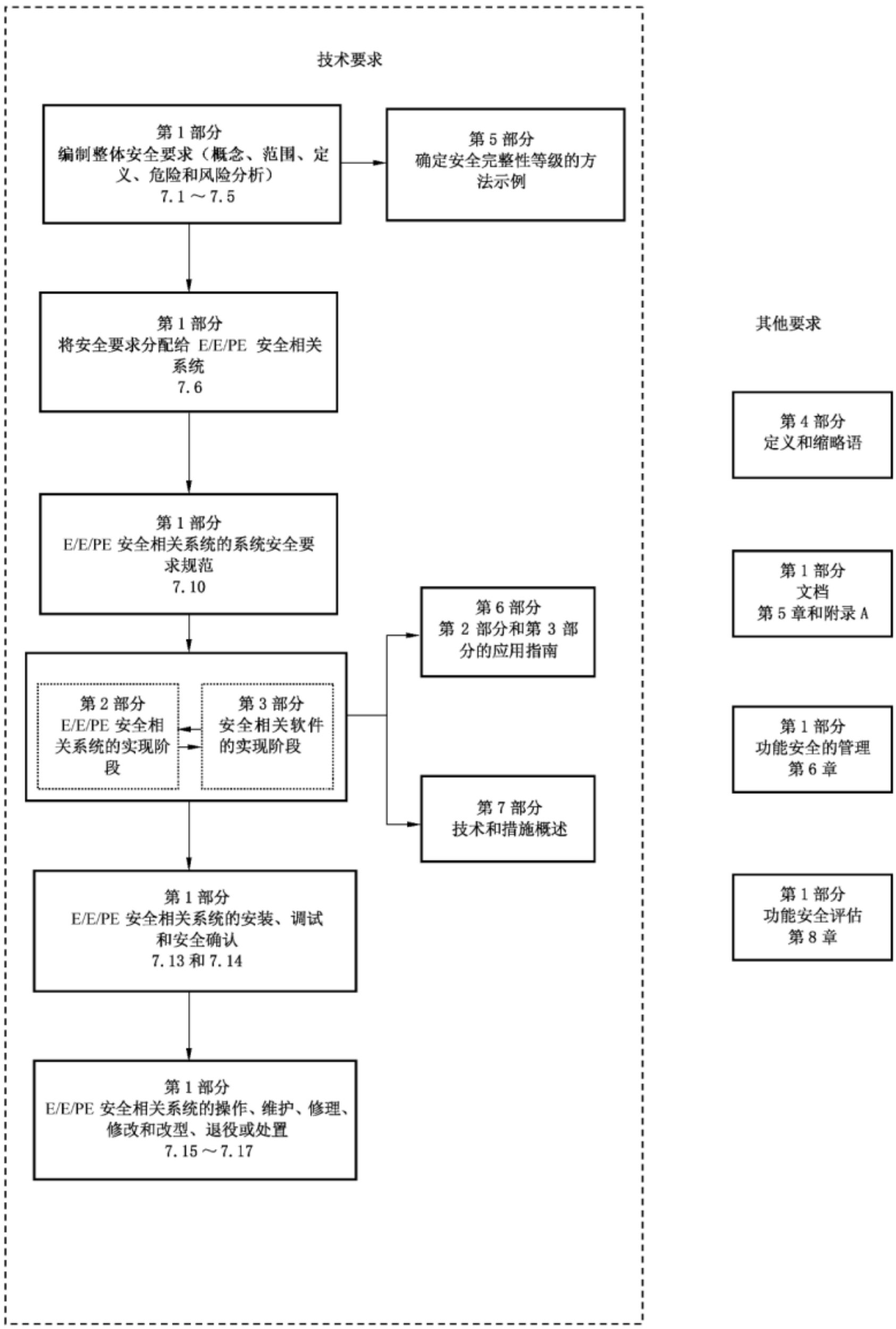


图 1　GB/T 20438 的整体框架

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 20438.4—2017 电气/电子/可编程电子安全相关系统的功能安全 第4部分：定义和缩略语(IEC 61508-4:2010，IDT)

3 定义和缩略语

GB/T 20438.4—2017 界定的定义和缩略语适用于本文件。

附 录 A
（资料性附录）
E/E/PE 安全相关系统的技术和措施概述：随机硬件失效控制（参看 GB/T 20438.2）

A.1 电气

整体目标：控制机电元件中的失效。

A.1.1 利用在线监视检测失效

注：在 GB/T 20438.2—2017 的表 A.2、表 A.3、表 A.7 和表 A.13～表 A.18 中引用了本技术/措施。

目的：通过监视 E/E/PE 安全相关系统在响应受控设备（EUC）正常（在线）运行时的行为来检测失效。

描述：在某些条件下，可用（例如）EUC 的时间行为信息来检测失效，例如，作为 E/E/PE 安全相关系统组成部分的一只开关，由 EUC 正常驱动，如果在预定的时间开关并未改变状态，则将检测到一次失效，通常要定位失效部位是不可能的。

A.1.2 继电器触点监视

注：在 GB/T 20438.2—2017 的表 A.2 和表 A.14 中引用了本技术/措施。

目的：检测继电器触点的失效（例如被熔接）。

描述：强制接触（或者正导向接触）继电器使它们的触点刚性的接在一起，假定有两组反向触点，分别为 a 和 b，如果常开触点 a 被熔接，则当继电器线圈断电时，常闭触点 b 就不能闭合，因此，当继电器线圈断电时，监视常闭触点 b 的吸合可用来证明常开触点 a 已打开。常闭触点 b 闭合的失效表明触点 a 的一次失效，所以对任何受触点 a 控制的机器而言，监视电路应保证安全关机或保持关机状态。

参考文献：

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Verriegelung-seinri-chtungen. F. Kreutzkampf, W. Hertel, Sicherheitstechnisches Informations-und Arbeitsblatt 330212, BIA-Handbuch.17, Lfg.X/91, Erich Schmidt Verlag, Bielefeld

www.BGIA-HANDBUCHdigital.de/330212

A.1.3 比较器

注：在 GB/T 20438.2—2017 的表 A.2、表 A.3、表 A.4 中引用了本技术/措施。

目的：为了尽早检测一个独立处理单元或者比较器中的（非同时的）失效。

描述：利用一个硬件比较器周期性地或者连续地比较独立处理单元的信号。可以在外部测试比较器，或者它本身可使用自监视技术。监测到的处理器行为的差异将产生一条失效报文。

A.1.4 多数表决器

注：在 GB/T 20438.2—2017 的表 A.2、表 A.3 和表 A.4 中引用了本技术/措施。

目的：为了检测和防护三个或三个以上硬件通道之一的失效。

描述：表决单元使用多数原理（3 个中有 2 个、3 个中有 3 个、或者 n 个中有 m 个）来检测和防护失效。表决器自身可由外部测试，也可使用自监视技术。

参考文献：

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

A.1.5 空闲电流的原则(断电跳闸)

注:在 GB/T 20438.2—2017 的表 A.16 中引用了本技术/措施。

目的:为了在切断电源或掉电时执行安全功能。

描述:当触点断开并且没有电流流过时就执行安全功能。例如,当使用制动器来制止一台电机的危险运动时,制动器将随安全相关系统中触点的闭合而开放,随安全相关系统中触点的打开而制动。

参考文献:

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

A.2 电子

整体目标:控制固态部件中的失效。

A.2.1 利用冗余硬件进行测试

注:在 GB/T 20438.2—2017 的表 A.3、表 A.15、表 A.16、表 A.18 中引用了本技术/措施。

目的:为了检测失效而使用硬件冗余(即使用不必执行过程功能的附加硬件)。

描述:冗余硬件可用来以一个适当的频度检测指定的安全功能。要实现 A.1.1 或 A.2.2,通常必须使用这种方法。

A.2.2 动态原理

注:在 GB/T 20438.2—2017 的表 A.3 中引用了本技术/措施。

目的:通过动态信号处理来检测静态失效。

描述:强制改变其他的静态信号(内部或外部产生的)可帮助检测部件中的静态失效。通常这种技术与机电部件相联系。

参考文献:

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitsechnisches Informations-und Arbeitsblatt 330220,BIA-Handbuch,Erich-Schmidt Verlag,Bielefeld,1993

http://www.bgia-handbuchdigital.de/330220

A.2.3 标准测试访问端口和边界扫描架构

注:在 GB/T 20438.2—2017 的表 A.3、表 A.15 和表 A.18 中引用了本技术/措施。

目的:为了控制和观测一片 IC(集成电路)的每个引脚处发生的情况。

描述:边界扫描测试是一种 IC 设计技术,此技术通过解决怎样访问 IC 内的电路测试点的问题提高 IC 的可测试性。在由内核逻辑、输入/输出缓冲器组成的一个典型边界扫描 IC 中,内核逻辑和与每个 IC 引脚相邻的输入/输出缓冲器之间插入了一个移位寄存器级。各个移位寄存器级都包含在一个边界扫描组元中。通过标准测试存取端口,边界扫描组元可控制和观测一个 IC 的每个输入/输出引脚处发生的情况。把芯片上的内核逻辑同接收到的外围部件的激励隔离开然后执行一次内部自测试,可完成 IC 内核逻辑的内部测试。这些测试可用来检测 IC 中的失效。

参考文献:

IEEE 1149-1: 2001 IEEE standard test access port and boundary-scan architecture, IEEE

Computer Society,2001,ISBN:0-7381-2944-5

A.2.4 （不再使用）

A.2.5 监视冗余

注：在 GB/T 20438.2—2017 的表 A.3 中引用了本技术/措施。

目的：通过配备几个功能单元，监视每个单元的行为以检测失效，在检测到这些单元的行为有任何差异时就开启到一种安全工况的转换。

描述：至少由两个硬件通道来执行安全功能。监视这些通道的输出，当检测到一个故障时（即当各通道的输出信号不相同时）就启动一个安全工况。

参考文献：

Elektronik in der Sicherheitsechnik. H. Jürs, D. Reinert, Sicherheitechnisches Informations-und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993

http://www.bgia-handbuchdigital.de/330220

Dependability of Critical Computer Systems 1. F. J. Redmll, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

A.2.6 带自动检验的电气/电子部件

注：在 GB/T 20438.2—2017 的表 A.3 中引用了本技术/措施。

目的：为了通过定期检验安全功能来检测失效。

描述：在起动过程之前测试硬件，并且按适当的间隔时间反复测试该硬件。只有在每次测试都无问题时，受控设备（EUC）才继续运行。

参考文献：

Elektronik in der Sicherheitstechnik. H. Jürs, D. Reinert, Sicherheitstechnisches Informations-und Arbeitsblatt 330220, BIA-Handbuch, Erich-Schmidt Verlag, Bielefeld, 1993

http://www.bgia-handbuchdigital.de/330220

Dependability of Critical Computer Systems 1. F. J. Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0.

A.2.7 模拟信号监视

注：在 GB/T 20438.2—2017 的表 A.3 和表 A.13 中引用了本技术/措施。

目的：提高所测信号的可信度。

描述：凡在选择时，应优先使用模拟信号而不使用数字开/关状态。例如，在使用常见的信号电平容差监视时，都是用模拟信号电平来表示跳闸或者安全状态。该技术提供了连续监视以及变送器可信度的较高级别，减少了变送器传感功能必要的验证试验的频度。外部接口，例如脉冲线路也需要测试。

A.2.8 降额

注：在 GB/T 20438.2—2017 的 7.4.2.13 中引用了本技术/措施。

目的：提高硬件部件的可靠性。

描述：通过把系统设计成在低于最大规范额定值下运行来保证硬件部件在某一应力级下运行。降额是保证在所有额定工作环境下，部件可在低于它们的最大应力级下工作的实际作法。

A.3 处理单元

整体目标：辨别导致处理单元结果错误的失效。

A.3.1 利用软件进行自测试:有限数量的模板(单通道)

注:在 GB/T 20438.2—2017 的表 A.4 中引用了本技术/措施。

目的:尽早检测出处理单元中的失效。

描述:使用不需考虑任何特殊安全要求的标准技术构建硬件。借助附加的软件功能完全可实现失效检测,这种软件功能使用至少两个互补的数据模板(例如十六进制的 55 和十六进制的 AA)来执行自测试。

A.3.2 利用软件进行自测试:漫步位(WALKING BIT)(单通道)

注:在 GB/T 20438.2—2017 的表 A.4 中引用了本技术/措施。

目的:尽早检测出处理单元的物理存储器(例如寄存器)和指令译码器中的失效。

描述:利用附加软件功能完全可实现失效检测,该软件功能使用可测试物理存储器(数据和地址寄存器)和指令译码的一个数据模板(例如漫步位模板)执行自测试。但诊断覆盖率只有 90%。

A.3.3 由硬件支持的自测试(单通道)

注:在 GB/T 20438.2—2017 的表 A.4 中引用了本技术/措施。

描述:附加的特殊硬件装置支持自测试功能以便检测失效。例如,该装置可能是一个硬件单元,它根据看门狗原理周期性地监视某个位模板的输出。

A.3.4 编码处理(单通道)

注:在 GB/T 20438.2—2017 的表 A.4 中引用了本技术/措施。

目的:尽早检测出处理单元中的失效。

描述:设计处理单元时使用了特殊的失效辩别或者失效校正电路技术。迄今,这些技术还只适用于比较简单的电路而未普及;但不排除将来的发展。

参考文献:

Le processeur codé:un nouveau concept appliqué à la sécurité des systèmes de transports.Gabriel, Martin,Wartski,Revue Générale des chemins de fer,No.6,June 1990

Vital Coded Microprocessor Principles and Application for Various Transit Systems. P. Forin, IFAC Control Computers Communications in Transportation,79-84,1989

A.3.5 利用软件进行相互比较

注:在 GB/T 20438.2—2017 的表 A.4 中引用了本技术/措施。

目的:尽早检测出处理单元中的失效,本方法使用了动态软件比较。

描述:两个处理单元互相交换数据(包括结果、中间结果和测试数据)。在每个单元中使用软件对数据进行一次对比,检测到的差异将产生一条失效报文。

A.4 不可变内存范围

整体目标:检测不可变内存中信息的修改。

A.4.1 字保存多位冗余(例如使用改进的汉明码进行 ROM 监视)

注 1:在 GB/T 20438.2—2017 的表 A.5 中引用了本技术/措施。

注 2:参见 A.5.6 使用改进的汉明码进行 RAM 监视,或者通过错误检测和纠正码(EDC)检测数据失效和 C.3.2 错误检测和纠正码。

目的:检测一个16位字中所有的一位失效,所有的双位失效,某些3位失效和某些全部位的失效。

描述:为了产生汉明距离至少为4的一个改进的汉明码,可用几个冗余位来扩充存储器的每一个字。每当读一个字时,检验冗余位就能确定是否发生了错误。当发现有一个差异时,就会产生一条失效信息。此程序也可通过计算数据字及其地址的连接冗余位来检测寻址失效。

参考文献:

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967

Error detecting and error correcting codes. R. W. Hamming, The Bell System Technical Journal 29(2), 147-160, 1950

A.4.2 修改的校验和

注:在GB/T 20438.2—2017的表A.5中引用了本技术/措施。

目的:检测所有奇数位失效,即全部可能位的大约50%失效。

描述:借助一种合适的算法来创立一个校验和,此算法使用了一个内存块中的全部字。校验和可作为一个附加字存储在ROM中,或者把一个附加字附加给内存块以保证校验和算法产生一个预定的值。在后面的内存测试中,再使用同一算法建立一个校验和,并把结果同存储的或者定义的值进行对比。如果发现有差异,则产生一条失效报文。

A.4.3 单字(8位)的签名

注:在GB/T 20438.2—2017的表A.5中引用了本技术/措施。

目的:检测一个字的所有一位失效和所有多位失效,可检测所有可能位失效的约99.6%。

描述:(既可使用硬件也可使用软件)通过一种循环冗余校验(CRC)算法把一个内存块的内容压缩成一个内存字。一种典型的CRC算法是把块的整个内容当作字节串行或者位串行数据流来处理,根据这种算法,使用一个多项式发生器来执行一个连续的多项式除法。除得的余项就代表压缩的存储内容——即是存储器的"签名"——并被存储起来。在后面的测试中又计算一次签名,并把这个签名同早先存储的签名进行比较。当它们有差异时,就产生一条失效报文。

A.4.4 双字(16位)的签名

注:在GB/T 20438.2—2017的表A.5中引用了此技术/措施。

目的:检测一个字中所有的一位失效和所有的多位失效,可检测所有可能位失效的99.998%。

描述:本程序使用一种循环冗余校验(CRC)算法来计算一个签名,而结果值至少有两个字长,像单字情况中那样,扩展的签名被存储、重新计算和比较。当存储的和重算的签名之间有差异时就产生一条失效报文。

A.4.5 块复制(例如利用硬件或者软件进行比较的双重ROM)

注:在GB/T 20438.2—2017的表A.5中引用了本技术/措施。

目的:检测全部位失效。

描述:在两个存储器中复制地址空间。第一个存储器以正常方式工作。第二个存储器包含同样的信息并且同第一个并行存取。比较它们的输出,当检测到有差异时就产生一条失效报文。为了检测某类位错误,必须在两个存储器中的一个中逆向存储数据,当读数时,再次反向。

A.5 可变的存储区

整体目标:检测寻址、写、存储和读过程中的失效。

注：软错误列在 GB/T 20438.2—2017 的表 A.1 中，作为运行期间可以被检测出的故障，或者进行分析以导出安全失效分数。软错误的起因包括：(1)封装损坏导致的 Alpha 粒子，(2)中子，(3)外部 EMI 噪声，(4)内部串扰。外部 EMI 噪声由其他国际标准要求覆盖。

Alpha 粒子和中子的影响宜由运行时安全完整性措施来控制。对硬差错有效的安全完整性措施对软错误不一定有效，例如，漫步路径法(walk-path)，乒乓测试法(galpat)等 RAM 测试法都无效。但是象奇偶校验和 ECC 等能复读存储器单元的监控技术是有效的。

当辐射事件引起足够的电荷干扰扭转或翻转低能量的半导体存储单元、寄存器、锁存器或触发器的数据状态时，软错误就会发生。之所以被称为软错误，是因为电路本身并未受到辐射的永久损坏。软错误分为单位翻转(SBU)或单事件翻转(SEU)和多位翻转(MBU)。

如果受干扰电路是存储单元或触发器的存储元件，状态被保存直到下一次(预期的)写操作，新数据就会被正确存储。在组合电路中，这个影响是一个小故障，因为从部件来的连续的能量流会驱动此节点。对连接线和通信线的影响也可能是一个小故障。不过，由于较大的电容，α 粒子和中子的影响被认为是微不足道。

软错误可能和各类可变内存相关，比如动态内存(DRAM)、静态内存(SRAM)、微处理器中的寄存器组、快速缓冲贮存区(cache)、流水线(pipelines)、设备(如 ADC、DMA、MMU)配置寄存器、中断控制器、复杂定时器等。对 alpha 粒子和中子的灵敏度由核心电压和几何结构共同作用的结果。在 2.5 V 核心电压，特别是低于 1.8 V 的核心电压更小的 geometries 需要更多评估和更加有效的保护措施。

根据下面从 a 到 i 的报告记录，内存(嵌入式)的软错误率在 700 Fit/Mbit 到 1 200 Fit/Mbit 范围之间。这是通过比较采用硅工艺实现的器件得到的一个参考值。直到最近 SBU 差错被认为是主要的差错，但是最近的报道(参见下列 a)称：在 65nm 以下技术下，所有软错误率(SER)里面 MBU 占据的比例越来越高。

下列文献和资料有软错误的详细描述：

a) Altitude SEE Test European Platform(ASTEP) and First Results in CMOS 130 nm SRAM, J-L. Autran, P. Roche, C. Sudre et al. Nuclear Science, IEEE Transactions on Volume 54, Issue 4, Aug. 2007 Page(s): 1002-1009

b) Radiation-Induced Soft Errors in Advanced Semiconductor Technologies, Robert C. Baumann, Fellow, IEEE, IEEE TRANSACTIONS ON DEVICE AND MATERIALS RELIABILITY, VOL 5, NO. 3, SEPTEMBER 2005

c) Soft errors' impact on system reliability, Ritesh Mastipuram and Edwin C Wee, Cypress Semiconductor, 2004

d) Trends And Challenges In VLSI Circuit Reliability, C. Costantinescu, Intel, 2003, IEEE Computer Society

e) Basic mechanisms and modeling of single-event upset in digital microelectronics, P. E. Dodd and L. W. Massengill, IEEE Trans. Nucl. Sci., vol. 50, no. 3, pp. 583-602, Jun. 2003

f) Destructive single-event effects in semiconductor devices and ICs, F. W. Sexton, IEEE Trans. Nuci. Sci., vol 50, no. 3, pp. 603-621, Jun. 2003

g) Coming Challenges in Microarchitecture and Architecture, Ronen, Mendelson, Proceedings of the IEEE, Volume 89, Issue 3, Mar 2001 Page(s): 325-340

h) Scaling and Technology Issues for Soft Error Rates, A Johnston, 4th Annual Research Conference on Reliability Stanford University, Ocotober 2000

i) International Technology Roadmap for Semiconductors(ITRS), several papers.

A.5.1 “检测板(checkerboard)”法或“跨步(march)”法 RAM 测试

注：在 GB/T 20438.2—2017 的表 A.6 中引用了本技术/措施。

目的：检测主要的静态位失效。

描述：把含有数个 0 和数个 1 的一个检测板写入面向位的存储器的存储单元中。成对地检查存储单元以保证它们的内容相同和正确无误。这个成对的第 1 单元的地址是可变的，而此对的第 2 单元的地址则由第 1 单元的地址逐位取反形成。第 1 次运行时，存储器地址区从可变地址朝高地址方向扩展，而第 2 次运行时则朝低地址方向扩展，在预先指定一个反向值的情况下，两次运行应是一样的。如出现

有差异则产生一条失效报文。

在一次“跨步”法 RAM 测试中,一个面向位的存储器的单元将由一个均匀的位流初始化。在第一次运行时,按升序检查这些单元,检查每个单元的内容是否正确并将它的内容反向。在第二次运行时按降序和同样的方式处理第一次运行中建立的后台。在第 3 次或第 4 次运行中反向预赋值的情况下,将重复头两次运行。如果出现有差异就会产生一条失效报文。

A.5.2 “漫步路径(walkpath)”法 RAM 测试

注:在 GB/T 20438.2—2017 的表 A.6 中引用了本技术/措施。

目的:为了检测静态和动态位失效,以及存贮单元之间的串扰。

描述:用一个均匀的位流初始化要测试的存储范围,第 1 个单元被反向并检查其余的存储区以确保后台是正确无误的。此后第 1 单元再次反向从而使它回复到它的初始值,对下一个单元也重复整个操作过程。在反向的后台预赋值情况下执行“漫游位模型”的第 2 次运行。当出现有差异时就会产生一条失效报文。

A.5.3 “跳步模式(galpat)”法或者“透明跳步模式(transparent galpat)”法 RAM 测试

注:在 GB/T 20438.2—2017 的表 A.6 中引用本技术/措施。

目的:为了检测静态位失效和大比例动态耦合。

描述:在“galpat”RAM 测试法中,先一致地(即全为 0 或者为 1)初始化选择的存储范围。测试最初存储单元,然后倒置第 1 个存储单元,并检查所有剩余的单元以确保它们的内容正确无误。每读取一个剩余单元之后,也检验反向的单元。对选定的存储范围中的每个单元都重复此操作过程。在相反的初始化情况下执行第二次运行。任何差异就会产生一条失效报文。

“透明的 galpat”测试法是上面的操作过程的一种变种:更换初始化所选存储范围内的所有单元,现存内容保持不变,使用签名来比较单元集合的内容。选择在所选范围内要测试的第 1 个单元,计算范围中其余所有单元的签名 S1,并把它存储起来。然后把要测试的单元反向,重新计算所有其余单元的签名 S2(在每读取一个剩余单元之后,也检验反向的单元。)比较 S2 和 S1,任何差异就产生一条失效报文。重新反向被测单元使之重建原先的内容,重新计算其余所有单元的签名 S3,把 S3 同 S1 进行比较,任何差异就产生一条失效报文。按同样的方法测试所选存储范围中的所有存储单元。

A.5.4 “Abraham”RAM 测试法

注:在 GB/T 20438.2—2017 的表 A.6 中引用了本技术/措施。

目的:为了检测存储单元之间所有的固定型失效和耦合失效。

描述:检测故障的比例超过“galpat”RAM 测试法。整个存储器测试需要执行的操作次数约为 $30n$,n 是存储器中的单元数。为了在操作循环过程中使用透明法进行测试,需要通过分割存储器并在不同时间段测试每个分区。

参考文献:

Efficient Algorithms for Testing Semiconductor Random-Access Memories.R.Nair,S.M.Thatte,J.A.Abraham,IEEE Trans.Comput.C-27(6),572-576,1978

A.5.5 一位冗余(例如,使用一个奇偶校验位进行 RAM 监视)

注:GB/T 20438.2—2017 的表 A.6 中引用了本技术/措施。

目的:为了在被测试的存储范围内检测所有可能的位失效的 50%。

描述:把存储器的每个字都扩展 1 位(奇偶校验位),此位给每个字补齐偶数个或奇数个逻辑 1。每次读数据字时将检验它的奇偶性。如发现 1 的个数有错时,就产生一条失效报文。应这样选择偶或

奇偶性,使得在一次失效事件中,无论是 0 字(全 0)还是 1 字(全 1)都不是有效的,此时字也不是有效代码。当计算数据字和它的地址连接的奇偶性时,奇偶校验也可用来检测寻址失效。

A.5.6 利用一个修改的汉明码进行 RAM 监视,或者利用差错检测和纠错码(EDC)检测数据失效

注 1:在 GB/T 20438.2—2017 的表 A.6 中引用了本技术/措施。

注 2:可参看 A.4.1"字存储多位冗余(例如,利用修改的汉明码进行 ROM 监视)"和 C.3.2"差错检测和纠错码"。

目的:为了检测所有的奇位失效,所有的 2 位失效,某些 3 位和多位失效。

描述:把存储器的每次操作扩展几个冗余位从而产生具有一个汉明距离至少为 4 的一个修改过的汉明码。每次数据被读取时,通过检验冗余位可以确定是否发生了一次讹错。当发现有差异时,就产生一条失效报文。当计算数据字和它的地址连接的冗余位时,此程序也可用来检测寻址失效。

参考文献:

Prüfbare und korrigierbare Codes. W. W. Peterson, München, Oldenburg, 1967

Error detecting and error correcting codes. R. W. Hamming, The Bell System Technical Journal 29(2), 147—160, 1950

A.5.7 具有硬件或者软件比较和读/写测试的双重 RAM

注:在 GB/T 20438.2—2017 的表 A.6 中引用本技术/措施。

目的:为了检测全位失效。

描述:在两个存储器中复制地址空间。第一个存储器以常规方式工作。第 2 个存储器包含同样的信息并且同第一个存储器并行存取。比较两个输出,当检测到差异时就产生一条失效报文。为了检测某些类型的位错误,两个存储器中的一个存储的数据必须反向,当读出时再次反向。

A.6 I/O 单元和接口(外部通信)

整体目标:为了检测输入和输出单元(数字、模拟、串行或者并行)中的失效以及防止不允许的输出传送给过程。

A.6.1 测试模式

注:在 GB/T 20438.2—2017 的表 A.7、表 A.13 和表 A.14 中引用了本技术/措施。

目的:为了检测静态失效(固定型失效)和串扰。

描述:它是一种与数据流无关的输入和输出单元的循环测试。它用一种定义的测试模式来比较观测值和对应的预计值。测试模式信息、测试模式接受及测试模式评价都必须相互独立。受控设备不应受到不允许的测试模式的影响。

A.6.2 代码保护

注:在 GB/T 20438.2—2017 表 A.7、表 A.15、表 A.16 和表 A.18 中引用了本技术/措施。

目的:为了检测输入/输出数据流中随机硬件和系统性失效。

描述:本程序可保护输入和输出信息免受系统和随机硬件引起的失效。代码保护提供了以信息冗余和时间冗余为基础的、与数据流有关的输入、输出单元的失效检测。典型地是把冗余信息叠加在输入或输出数据上。它提供了监视输入或输出电路正确运行的一种方法。许多技术都能用,例如,在一个传感器输出信号上可以叠加一个载频信号。为了监视逻辑单元和最后的执行器之间流经的一个信号的有效性,逻辑单元可以检验附加到一个输出通道上的载频或者冗余码位的存在。

A.6.3 多通道并行输出

注：在 GB/T 20438.2—2017 的表 A.7 中引用了本技术/措施。

目的：为了检测随机硬件失效(固定型失效)、外部影响引起的失效、定时失效、寻址失效、漂移失效和瞬态失效。

描述：它是用于检测随机硬件失效的一个具有独立输出的、与数据流有关的多道并行输出。通过外部比较器进行失效检测。当发生一次失效时，立即关掉受控设备。这种措施只有在诊断测试间隔期间数据流改变时才有效。

A.6.4 受监视的输出

注：在 GB/T 20438.2—2017 的表 A.7 中引用了本技术/措施。

目的：为了检测个体失效、由外部影响引起的失效、定时失效、寻址失效、漂移失效(模拟信号)和瞬态失效。

描述：输出同无关的输入之间有关数据流的比较从而保证同一个定义的容差范围(时间、值)相符。检测到的一次失效并不总是同输出故障有关。只有在诊断测试间隔期间数据流改变时，此措施才有效。

A.6.5 输入比较/表决

注：在 GB/T 20438.2—2017 的表 A.7 和表 A.13 中引用本技术/措施。

目的：为了测量个体失效、由外部影响引起的失效、定时失效、寻址失效、漂移失效(对于模拟信号)和瞬态失效。

描述：独立输入的一种与数据流有关的比较，从而确保同一个定义的容差范围(时间、值)相符。有 2 选 1、3 选 2 或者更多的冗余方法。此措施只有在诊断测试间隔期间数据改变时才有效。

A.7 数据通路(内部通信)

整体目标：为了检测由信息传送中的一个故障引起的失效。

A.7.1 一位硬件冗余

注：在 GB/T 20438.2—2017 的表 A.8 中引用了一技术/措施。

目的：为了检测所有的奇位失效，即数据流中所有可能失效位的 50%。

描述：用一行(位)扩充总线并且借助奇偶校验可把这个附加行(位)用于检测失效。

A.7.2 多位硬件冗余

注：在 GB/T 20438.2—2017 的表 A.8 中引用了本技术/措施。

目的：为了检测通信过程中总线上和串行传输链路中的失效。

描述：用 2 行(位)或多行(位)扩充总线，为了借助汉明码技术检测失效，将使用这些附加行(位)。

A.7.3 完全硬件冗余

注：在 GB/T 20438.2—2017 的表 A.8 中引用了本技术/措施。

目的：利用比较两条总线上的信号检测通信过程中的失效。

描述：为了检测失效，总线数量被加倍并使用了附加行(位)。

A.7.4 使用测试模式进行检查

注：在 GB/T 20438.2—2017 的表 A.8 中引用了本技术/措施。

目的:为了检测静态失效(固定型失效)和串扰。

描述:这是一种与数据流无关的数据通路循环测试。它使用一个定义的测试模式来比较观察值和相应的预期值。

测试模式信息、测试模式接受、测试模式评价必须彼此完全无关。受控设备不应受到测试模式不允许的影响。

A.7.5 传输冗余

注:在 GB/T 20438.2—2017 的表 A.8 中引用了本技术/措施。

目的:为了检测总线通信中的瞬态失效。

描述:依次把信息传送几次。这种重复法只对于瞬态失效才有用。

A.7.6 信息冗余

注:在 GB/T 20438.2—2017 的表 A.8 中引用了本技术/措施。

目的:为了测量总线通信中的失效。

描述:成块地并连同每个块的一个计算出的检验和传输数据。接收机重新计算接收数据的检验和,并把结果同接收到的检验和作比较。

A.8 电源

整体目标:为了检测或者允许一个电源故障引起的失效。

A.8.1 使用安全断电的过压保护

注:在 GB/T 20438.2—2017 的表 A.9 中引用了本技术/措施。

目的:为了保护安全相关系统免受过压。

描述:及早检测过压使之能通过掉电例行程序或者转换到第二电源把所有输出转换到一个安全工况。

参考文献:

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

A.8.2 电压控制(次级)

注:在 GB/T 20438.2—2017 的表 A.9 中引用了本技术/措施。

目的:为了监视次级电压并当电压超过规定范围时就启动一个安全工况。

描述:监视次级电压,当它超出规定范围时,就启动一次掉电或转换到第 2 电源。

A.8.3 具有安全断电的掉电

注:在 GB/T 20438.2—2017 的表 A.9 中引用了本技术/措施。

目的:在所有安全关键信息被保存的情况下切断电源。

描述:及早检测过压或欠压使之内部状态能保存在非易失性存储器中(如必要的话),并使所有的输出能通过掉电例程设置或转换到一个安全工况,或者转换到第 2 电源。

A.9 时序的和逻辑的程序序列监视

注:在 GB/T 20438.2—2017 的表 A.15、表 A.16 和表 A.18 中引用了本技术/措施。

整体目标:为了检测一个有缺陷的程序序列。当在错误的序列或时段中处理一个程序的各个单元(例如软件模块、子程序或者命令)时或者当处理机的时钟有毛病时,就会存在一个有缺陷的程序序列。

A.9.1 具有分离时基而无时间窗的看门狗

注:在 GB/T 20438.2—2017 的表 A.10 和表 A.11 中引用了本技术/措施。

目的:为了监视程序序列的行为和合理性。

描述:为了监视计算机的行为和程序序列的合理性,定期触发具有分离时基的外部定时单元(例如看门狗)。在程序中正确地设置触发点是很重要的。不按一个固定的周期触发看门狗,但规定了最大时间间隔。

A.9.2 具有分离时基和时间窗的看门狗

注:在 GB/T 20438.2—2017 的表 A.10 和表 A.11 中引用了本技术/措施。

目的:为了监视程序序列的行为和似真性。

描述:为了监视计算机的行为和程序序列的似真性,定期触发具有分离时基的外部定时单元(例如看门狗)。在程序中正确设置触发点是很重要的。给出了看门狗的一个上、下限。如果程序序列占用的时间比预定的时间长或者短,就会采取应急行动。

A.9.3 程序序列的逻辑监视

注:在 GB/T 20438.2—2017 的表 A.10 和表 A.11 中引用了本技术/措施。

目的:为了监视各个程序段的正确顺序。

描述:使用软件(计数程序、临界程序)或者使用外部监视设备监视各个程序段的正确顺序。在程序中正确设置校验点是很重要的。

A.9.4 程序序列的时序和逻辑监视的组合

注:GB/T 20438.2—2017 的表 A.10 和表 A.11 中引用了本技术/措施。

目的:为了监视各个程序段的行为和正确的顺序。

描述:只有在也正确执行程序段的顺序时才会触发监视程序序列的一台时序设备(例如看门狗计时器)。

A.9.5 具有在线检验的时序监视

注:在 GB/T 20438.2—2017 的表 A.10 和表 A.11 中引用本技术/措施。

目的:为了检测时序监视中的失效。

描述:在起动时检验时序监视,并且只有当时序监视正常工作时,才可能启动。例如,在起动时,可以用一个加热的电阻来检验一个热传感器。

A.10 通风和加热

注:在 GB/T 20438.2—2017 的表 A.16 和表 A.18 中引用了本技术/措施。

整体目标:为了控制通风和加热中的失效,和/或 监视它们是否与安全有关。

A.10.1 温度传感器

目的:当系统开始在规定的温度范围之外工作之前,检测温度过高还是温度过低。

描述:温度传感器监视 E/E/PES 的大多数临界点处的温度。在温度偏离规定范围之前就采取应

急行动。

A.10.2 风扇控制

目的:为了检测风扇运转的不正常。

描述:监视风扇的运转是否正常。当一台风扇工作不正常时,就要采取维护(或者最终得采取应急)行动。

A.10.3 通过热熔断器启动安全断电

目的:在系统的工作温度超过技术条件的规定之前,就断掉安全相关系统的电源。

描述:一个热熔断器被用来切断安全相关系统的供电,对一个可编程电子系统(PES)来说,可通过一个存储有应急行动所需的全部信息的掉电例行程序来引起断电。

A.10.4 来自温度传感器和条件报警的交错报文

目的:为了指示安全相关系统正工作在技术条件规定的温度范围之外。

描述:监视温度,当温度超出规定范围时就产生一个报警。

A.10.5 强制通风制冷的连接和状态指示

目的:利用强制风冷却防止过热。

描述:监视温度,当温度高于规定的上限时就引起强制风冷却。并告之用户系统此时的状态。

A.11 通信和大容量存储器

整体目标:为了控制在与外部源和海量存储器通信过程中的失效。

A.11.1 分隔开电力线和信息线

注:在 GB/T 20438.2—2017 的表 A.16 中引用了本技术/措施。

目的:为了最小化信息线中大电流感生的串扰。

描述:分隔开电力供电线同传送信息的线路。可能在信息线上感生电压脉冲的电场随距离增大而减小。

A.11.2 多线路的空间分隔

注:在 GB/T 20438.2—2017 的表 A.16 中引用了本技术/措施。

目的:为了最小化多线路中大电流感生的串扰。

描述:载有重复信号的线路彼此分隔开。可能在多线路上感生电压脉冲的电场随距离增大而减小。这种措施也减少了共同原因失效。

A.11.3 提高抗干扰性

注:在 GB/T 20438.2—2017 的表 A.16 和表 A.18 中引用了本技术/措施。

目的:为了减小对安全相关系统的电磁干扰。

描述:可以使用比如屏蔽和滤波这样的设计技术来提高安全相关系统对电力线或信号线或者静电放电产生的辐射或者传导电磁干扰的抗扰性。

注:参见表 16 和表 17 安全相关系统的抗干扰要求和工业应用中实施安全相关功能(功能安全)的设备抗干扰要求。

参考文献：

IEC/TR 61000-5-2：1997，Electromagnetic compatibility（EMC）—Part 5：Installation and mitigation guidelines—Section 2：Earthing and cabling

Principles and Techniques of Electromagnetic Compatibility，Second Edition，C. Christopoulos，CRC Press，2007，ISBN-10：0849370353，ISBN-13：978-0849370359

Noise Reduction Techniques in Electronic Systems. H. W. 0tt，John Wiley Interscience，2nd Edition，1988

EMC for Product Designers. T. Williams，Newnes，2007，ISBN：0750681705

Grounding and Shielding Techniques in Instrumentation，3rd edition，R. Morrison. WileyInterscience，New York，1986，ISBN-10：0471838055，ISBN-13：978-0471838050

A.11.4 反价的(Antivalent)信号传输

注：在 GB/T 20438.2—2017 的表 A.7 和表 A.16 中引用了本技术/措施。

目的：为了检测在多信号传输线中相同的感应电压。

描述：使用反价的(Antivalent)信号(例如逻辑 1 和 0)来传输所有的重复信息。可以通过一个反价比较器来检测共同原因失效(例如由电磁发射引起的)。

参考文献：

Elektronk in der Sicherheitstechnik. H. Jürs，D. Reinert，Sicherheitstechnisches Informations-und Arbeit-sblatt 330220，BIA-Handdbuch.20.Lfg.V/93，Erich Schmidt Verlag，Bielefeld.

http://www.bgia-handbuchdigital.de/330220

A.12 传感器

整体目标：为了控制安全相关系统的传感器中的失效。

A.12.1 参考传感器

注：在 GB/T 20438.2—2017 的表 A.13 中引用了本技术/措施。

目的：为了检测传感器的工作不正常。

描述：使用了一个独立的参考传感器来监测过程传感器的工作。参考传感器以适当的时间间隔检验所有的输入信号从而检测过程传感器的失效。

参考文献：

Guidelines for Safe Automation of Chemical Processes. CCPS，AIChE，New York，1993，ISBN-10：0-8169-0554-1，ISBN-13：978-0-8169-0554-6

A.12.2 启动可靠的开关

注：在 GB/T 20438.2—2017 的表 A.13 中引用了本技术/措施。

目的：通过开关凸轮和触点之间的直接机械连接断开一个触点。

描述：通过开关凸轮和触点之间的直接机械连接，启动可靠的开关将断开它的常闭触点。这就保证了任何时候，只要开关凸轮处于吸合位置，开关触点一定是断开的。

参考文献：

Verriegelung beweglicher Schutzeinrichtungen. F. Kreutzkampf，K. Becker，Sicherheitstechniches Informations-und Arbeitsblatt 330210，BIA-Handbuch.1.Lfg.IX/85 Erich Schmidt Verlag，Bielefeld

A.13 最终元件(执行器)

整体目标:为了控制安全相关系统的最终元件中的失效。

A.13.1 监视

注:在 GB/T 20438.2—2017 的表 A.14 中引用了本技术/措施。

目的:为了检测一个执行器的工作不正常。

描述:监视执行器的工作(例如通过一个继电器的启动可靠的触点,参见 A.1.2 中继电器触点监视)。此监视采用的冗余可用来触发应急行动。

参考文献:

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

Zusammenstellung und Bewertung elektromechanischer Sicherheitsschaltungen für Ver-riegelungsein-richtungen.F.Kreutzkampf,W.Hertel,Sicherheitstechnisches Informations-und Arbeitsblatt 330212,BIA-Han-dbuch.17.Lfg.X/91,Erich Schmidt Verlag,Bielefeld

A.13.2 多个执行器的交叉监视

注:在 GB/T 20438.2—2017 的表 A.14 中引用了本技术/措施。

目的:为了通过比较结果来检测多个执行器中的故障。

描述:使用一个不同的硬件通道来监视多个执行器中的每一个执行器。当出现差异时,就采取应急行动。

A.14 抗物理环境的措施

注:在 GB/T 20438.2—2017 的表 A.16 和表 A.18 中引用了本技术/措施。

目的:为了防止物理环境(水、灰尘、腐蚀物)的影响引起失效。

描述:设计的设备的机壳要能耐受预计的环境。

参考文献:

GB 4208—1993 外壳防护等级(IP 代码)

附 录 B
（资料性附录）
E/E/PE 安全相关系统的技术和措施概述：系统性失效的避免（参看 GB/T 20438.2 和 GB/T 20438.3）

注：本附录中的许多技术适用于软件，但在附录 C 中不再重复。

B.1 一般测量和技术

B.1.1 项目管理

注：在 GB/T 20438.2—2017 的表 B.1～表 B.6 中引用了本技术/措施。

目的：为了避免失效，在开发和测试安全相关系统时采用了一种组织模型、一些规则和措施。

描述：最重要和最好的措施是：

——建立一个组织模型，特别是用于在质量保证手册中制定的质量保证；以及

——在交叉项目和项目专用指南中制定建立和确认安全相关系统的规则和措施。

下面确立了许多重要的基本原则：

——设计组织的定义：

- 各组织单位的任务和责任；
- 质保部门的职权；
- 质量保证（内部检验）独立于开发。

——序列计划（活动模型）的定义：

- 确定在执行项目的过程中所有有关的活动，包括内部检验和它们的日程表；
- 项目更新。

——内部检验的标准化程序的定义：

- 检验（检验理论）的计划、执行和检查；
- 次品的放行机制；
- 重复检验的妥善保管。

——配置管理：

- 版本的管理和检查；
- 修改效果的检测；
- 修改后的一致性检验。

——采用质保措施定量评估：

- 采集要求；
- 失效统计。

——采用计算机辅助通用方法，工具和人员培训。

参考文献：

GB/T 19001—2008，质量管理体系 要求

ISO/IEC15504(all parts)，Information technology—Process assessment

CMMI：Guidelines for Process Integration and Product Improvement，2nd Edition. M. B. Chrissis，M. Konrad，S. Shrum，Addison-Wesley Professional，2006，ISBN-10：0-3213-7967-0，ISBN-13：978-3212-7967-5

Guidelines for Safe Automation of Chemical Processes. CCPS，AIChE，New York，1993，ISBN-10：0-8169-0554-1，ISBN-13：978-0-8169-0544-6

Dependability of Critical Computer Systems 1.F.J.Redmill,Elsevier Applied Science,1988,ISBN 1-85166-203-0

B.1.2 编制文档

注 1：在 GB/T 20438.2—2017 的表 B.1～表 B.6 中引用了本技术/措施。

注 2：另见 GB/T 20438.1—2017 的第 5 章和附录 A。

目的：通过把开发过程中的每一步编写成文件从而避免失效和便于评估系统安全性。

描述：在评估过程中必须论证操作能力和安全性，以及在开发中牵涉的各参与方的关注。为了能显示对开发的关注，以及为了保证任何时候都能验证安全证据，建立文档是特别重要的。

重要的通用措施是引入指南和计算机辅助，即

——指南，其

- 规定分组计划；
- 要求内容的检查表；
- 确定文档的形式。

——借助计算机辅助的和组织化的项目库来管理建立文档。

各种措施有：

——分开建立文档：

- 要求方面的文档；
- 系统方面的文档(用户文档)；
- 开发文档(包括内部检验)。

——根据安全生命周期对开发文档分组；

——定义标准化的文档模块，根据这种文档模块可以汇编文档；

——清楚的标记文档的各组成部分；

——正式的版本更新；

——选择清楚的和可理解的描述方式：

- 确定用的形式符号；
- 介绍、验证和表达意图用的自然语言；
- 举例用的图形表示；
- 图形元素的语义定义；和
- 专用字的目录。

参考文献：

GB/T 19898—2005 工业过程测量和控制 应用软件文档集

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0544-6

B.1.3 分离开 E/E/PE 系统的安全功能与非安全功能

注：在 GB/T 20438.2—2017 的表 B.1 和表 B.6 中引用了本技术/措施。

目的：为了防止系统的非安全部分以不期望的方式影响安全部分。

描述：在规范中，应判定是否能够把安全相关系统和非安全相关系统分离开。应清楚的说明两个部分衔接的规范。一份清楚的规范可减少测试安全相关系统的工作量。

参考文献：

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0544-6

B.1.4 多样化硬件

注：在 GB/T 20438.2—2017 的表 A.15、表 A.16 和表 A.18 中引用了本技术/措施。

目的：使用具有不同失效率和失效类型的多样化元件，检测受控设备运行过程中的系统性失效。

描述：对一个安全相关系统的各个通道，使用不同类型的元件。这将减少共因失效（例如过压、电磁干扰）的概率并提高检测这种失效的概率。

由于存在执行一个所需功能的不同方法，例如不同的物理原理，从而提供了解决同一问题的其他渠道。存在几种多样化形式。功能多样化使用了各种办法来达到同样的结果。

参考文献：

Guidelines for Safe Automation of Chemical Processes.CCPS，AIChE，New York，1993，ISBN-10：0-8169-0554-1，ISBN-13：978-0-8169-0544-6

B.2 E/E/PE 系统设计要求规范

整体目标：为了要编制一份尽可能完备、无歧义、无矛盾、易于验证的规范。

B.2.1 结构化规范

注：在 GB/T 20438.2—2017 的表 B.1 和表 B.6 中引用了本技术/措施。

目的：通过建立部分要求的一种层次化结构来减少复杂性。为了避免各要求之间的接口失效。

描述：本技术把功能规范构建成许多部分要求，从而使这些部分要求之间存在的关系变得最可能简单、可视。这种分析一步一步更加精确直到能够区别那些小而清晰的部分要求为止。最后的精确结果是一种部分要求的层次化结构，它提供了总体要求规范的一个框架。这种方法着重于各部分要求之间的接口，它对避免接口失效特别有效。

参考文献：

ESA PSS 05-02，Guide to the user requirements definition phase，Issue 1，Revision 1，ESA Board for Software Standardisation and Control(BSSC)，ESA，Paris，March 1995，

ftp://ftp.estec.esa.nl/pub/wrn/wme/bssc/PSS0502.pdf

Structured Analysis and System Specification. T. De Marco，Youdon Press，Englewood Cliffs，1979，ISBN-10：0138543801，ISBN-13：978-0138-543808

B.2.2 形式化方法

注 1：关于专用形式化方法的细节可参看 C.2.4。

注 2：在 GB/T 20438.2—2017 的表 B.1、表 B.2 和表 B.6 中引用了本技术/措施。

目的：形式化方法把数学推理方法的原则转换到技术体系的规范和实施，因而增加规范和实施的完整性、一致性或正确性。

描述：形式化方法提供了一种在系统开发规范和/或实施阶段描述该系统的一种方法。这些形式化的描述是系统功能和/或结构的数学模型。

因此明确的系统描述可以达到对潜在系统理解力的提高。（例如，任何一个自动机的状态被描述为初始状态，输入和自动机的转换方程）

选择一个合适的形式化方法是一项艰巨的任务，需要对系统充分了解，包括其发展过程和可能用到的一系列数学模型（见以下备注）。

注 3：模型的关注点理论（属性）与仿真模拟相比为系统提供了更多的信心和担保，例如研究选定系统的行为活动。

注 4：形式化方法的弊端可能为：

——比较抽象；

——特定阶段捕获相关功能的局限性；

——在实施时工程师理解模型有难度；

——系统生命周期内开发、分析和维护模型的高投入；

——能创建、分析模型的有效工具的可用性；

——开发、分析模型的人员能力的可用性。

注 5：形式化方法共有的重点显然是系统的目标函数建模，往往不强调系统对于故障的鲁棒性。因此必须采用相应的包括系统鲁棒性的形式化方法。

参考文献：

Formal Specification: Techniques and Applications. N. Nissanke, Springer-Verlag Telos, 1999, ISBN-10:1852330023

B.2.3 半形式化方法

注 1：在 GB/T 20438.3—2017 的表 B.7 中采用其他半形式化的软件相关技术扩展了这个附录 B 的列表。列表如下：

——逻辑/功能块图：在 GB/T 15969.3 中描述；

——顺序图：在 GB/T 15969.3 中描述；

——数据流图：见 C.2.2；

——有限状态机/状态转换图：见 B.2.3.2；

——时间佩特里(Petri)网：见 B.2.3.3；

——实体-关系-属性数据模型：见 B.2.4.4；

——消息序列图：见 C.2.14；

——判定/真值表：见 C.6.1。

目的：为了清楚地和一致地表达一份规范的各部分，以便能检查出某些错误、遗漏和不正确的行为。

注 2：在 GB/T 20438.2—2017 的表 B.1、表 B.2 和表 B.6 以及在 GB/T 20438.3—2017 的表 A.1、表 A.2、表 A.4、表 B.7、表 C.1、表 C.2、表 C.4 和表 C.17 中引用了本技术/措施。

B.2.3.1 概述

目的：为了证明设计满足它的规范。

描述：半形式化方法为在开发一个系统的某个阶段编制该系统的描述(即规范、设计或者编码)提供了一种方法。在某些情况下，可用机器分析该描述，或者为了显示系统各方面的行为，可把该描述制作成动画。此动画可对系统满足实际要求以及规定的要求提供了额外的置信度。

在下面的条款中将描述两种半形式化方法。

B.2.3.2 有限状态机/状态转换图

注：在 GB/T 20438.3—2017 表 B.5、表 B.7、表 C.15 和表 C.17 中引用了本技术/措施。

目的：建模、验证、规定或实施一个系统的控制结构。

描述：能够用它们的状态、它们的输入和它们的行动来描述许多系统。当处于状态 S1 时，在接受到输入 I 时，一个系统将会执行动作 A 并转换到状态 S2。通过描述一个系统处于每个状态中时，每个输入导致该系统的动作，就能完整地描述一个系统。产生的系统模型叫做一个有限状态机(或有限状态自动机)。常常是把它画成一个所谓的状态转换图，此图显示系统怎样从一个状态转移到另一个状态；或者把它画成一个矩阵，在矩阵中，维数是状态和输入，矩阵元包含当系统处于给定状态中时，由接受输入导致的动作和新状态。

一个复杂系统或者具有一个自然结构的系统的情况可以在一个分层的有限状态机中反映出来。状态图是一种可以嵌套的状态转换图(目标的状态分为可以并行演化的两个或更多子状态，并且可以在某

点重新组合为一个状态),这增加了状态转换标记法的表达能力,但在安全相关系统中也增加了不受欢迎的额外的复杂性。状态图有一个形式化的(数学上)规范,状态转换图可以适用于整个系统或某些对象或者它的元素。

可以检查表示成一个有限状态机的规范或设计的:

——完备性(在每个状态、对每个输入,系统必定有一个动作和新状态);

——一致性(每个状态/输入对只描述一次状态改变);

——可达性(是否能通过任何输入序列从一个状态到达另一状态);以及

——没有无限循环或无出路状态,等等。

它们是关键系统的一些重要属性。很容易开发支持这些检查的工具,并且可以使用基于有限状态机的不同模型(形式化语言、佩特里(Petri)网、马尔可夫图形等等)。而且也存在能够自动生成测试用例的算法,这些用例用于验证一个有限状态机实现或者制作一个有限状态机模型的动画。状态转换图和状态图被各种工具广泛支持,包括图表的绘制和检查,生成描述状态机执行的代码。

他们也可以用于失效概率的计算,参见 B.6 和 C.6 部分。

参考文献:

Introduction to Automata Theory, Languages, and Computation(3rd Edition). J. Hopcroft, R. Motwani, J. Ullman, Addison-Wesley Longman Publishing Co, 2006, ISBN:0321462254

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès-Lavoisier, 2009, ISBN:978-2-7462-1991-5

B.2.3.3 时间佩特里(Petri)网

注:在 GB/T 20438.3—2017 的表 B.5、表 B.7、表 C.15 和表 C.17 中引用了本技术/措施。

目的:通过分析和再设计模型化系统行为的有关方面、评估及尽可能地提高安全性和操作要求。

描述:佩特里(Petri)网是一个有限状态自动机的特殊情况,属于图形理论模型一类,适用于在呈现并发性和有异步行为的系统中表示信息和控制流。

佩特里(Petri)网是一个位置和变迁网。位置可被"标记"或"不标记"。当输入该网的所有输入位置被标记时,就"使能"一次变迁。当使能时,就允许(而不是强迫)"触发"。如果它触发,引起变迁的输入位置就变成未标记的了,并且代之以变迁产生的每个输出位置被标记。

在模型中可把潜在的危险表示成特定的状态(标记)。可把佩特里(Petri)网模型扩展成描述系统的时间特性。虽然"传统的"佩特里(Petri)网集中在控制流方面,已经提出了把数据流包括到模型中去的几种扩展方案。

这些也对执行蒙特卡罗模拟提供有效的支持,以实现失效概率的计算,参见 B.6.6.8。

参考文献:

Timed Petri Nets: Theory and Application. Jiacun Wang, Springer, 1998, ISBN 0792382706

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès-Lavoisier, 2009, ISBN:978-2-7462-1991-5

B.2.4 计算机辅助的规范工具

注:在 GB/T 20438.2—2017 的表 B.1 和表 B.6 中以及在 GB/T 20438.3—2017 的表 A.1、表 A.2、表 C.1 和表 C.2 中引用了本技术/措施。

B.2.4.1 概述

目的:使用形式化规范技术以便于自动检测歧义性和完备性。

描述:本技术可产生一个数据库形式的规范,这种形式可被自动检查,从而评估一致性和完备性。

规范工具能把用户指定的系统的各种特征制作成动画。通常,技术支持不仅建立规范,也能建立设计和项目生命周期的其他阶段。可根据以下条款对规范工具进行分类。

B.2.4.2 面向无特定方法的工具为了不用特殊方法所面向的工具

目的:通过提供提示和各相关部分间的连接,帮助用户编写一份好的规范。

描述:规范工具可接替用户的一些日常工作并支持项目管理。它不强求任何特殊的规范方法。在方法方面的相对独立性允许用户在建立规范时有很大的自由度,但为用户提供所必需的专门支持很少。这会使得精通系统更为困难。

B.2.4.3 面向模型的层次分析程序

目的:在规范中避免不完整、歧义和矛盾,例如通过保证动作和数据的描述在不同的抽象层次下的一致性,帮助用户编写一份好的规范。

描述:本方法给出了各抽象层次(精确等级)下,要求的系统的一个功能表达式(结构化分析)。这些模型组成了一个巨大的方法库,有限自动机就是一类广泛用于描述离散/数字系统演变的模型。微分方程在描述连续/模拟系统上具有类似的思路和目的。各个层次下的分析对动作和数据两者都有作用。在层次之间和同一层次的两个功能单元(模块)之间评估歧义性和完备性是可能的。(例如,系统模型的任何一个状态都可以用它的初始状态、输入和自动状态机的转换方程来描述。

注:基于模型描述的问题,可能是抽象级别的,在捕获特定阶段所有相关功能上的限制,从业者应该理解模型的难度(从阅读句法到理解),要在系统的整个生命周期中尽力开发、分析并保持一个模型,支持构建和分析模型的有效工具的可行性(开发这样的工具是一项需付出巨大努力的工作)和工作人员能够开发和分析模型的可行性。

参考文献:

System requirements analysis. Jeffrey O. Grady, Academic Press, 2006, ISBN 012088514X, 9780120885145

B.2.4.4 实体-关系-属性的数据模型

目的:关注系统中的各实体以及它们之间的关系,帮助用户编写一份好的规范。

描述:把要求的系统描述成一些对象和它们之间的关系的一个集合。工具使我们能确定系统能解释哪些关系。通常,这些关系允许描述对象、数据流的层次结构、数据之间的关系,以及哪些数据是取决于特定生产过程的。为了用于过程控制,已对传统的程序进行了扩充。检查能力和对用户的支持与所说明的各种关系有关。另一方面,大量可能的表达式使本技术的应用变得很复杂。

参考文献:

Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle. Karl Eugene Wiegers, Microsoft Press, 2003, ISBN 0735618798, 9780735618794

B.2.4.5 诱因和回答

目的:通过识别激励—响应关系,帮助用户编写一份好的规范。

描述:系统的各对象之间的关系用“诱因”和“回答”的一种表示法来说明。使用了一种简单和容易的护充语言,这种语言包含有代表对象、关系、特性和结构的语素。

B.2.5 检查表

注:在 GB/T 20438.2—2017 的表 B.1、表 B.2、表 B.6 和 GB/T 20438.3—2017 的表 A.10、表 B.8、表 C.10 和表 C.18 中引用本技术/措施。

目的:对安全生命周期阶段系统的所有重要方面,引起足够的注意和管理关键性评估,以保证没有遗漏确切要求的全面覆盖。

描述:由执行检查表的人来回答一系列问题。许多问题具有普遍性,评估员对这些问题应视为最适合于正在评估的特定系统那样来解释它们。检查表可用于整体安全生命周期、E/E/PE 系统和软件安全生命周期的各阶段并特别适合作为帮助功能安全评估的一种工具。

为了适应正在确认的类型繁多的系统,大多数检查表包含的问题适用于许多类型的系统。结果,在使用的检查表中的问题可能和正在讨论的系统无关,这时就应不管这些问题。相应的,对于一个特定的系统需要对一个标准的检查表进行补充,其中包含的问题是专门针对正在讨论的系统。

在任何一种情况下,使用检查表的关键取决于工程师选择和应用检查表的专门知识和判断能力。工程师选择检查表采取的决定和任何附加的或者多余的问题都应全部编入文档并证明是合理的。目的是要保证能评审检查表的应用并且保证使用相同的判据都应能达到同样的结果。

在完成一份检查表时,对象要尽可能简洁。当需要扩充证明时,应通过引用附加文档来进行这种扩充。编写每个问题的结果,应使用合格、不合格和不确定或者一些类似的受限的应答集合。这种简洁大大简化了获得有关检查表评估结果的全部结论的程序。

参考文献:

IEC 60880:2006,Nuclear power plants—Instrumentation and control systems important to safety—Software aspects for computer-based systems performing category A functions

The Art of Software Testing,Second Edition.G.Myers et al.,Wiley & Sons,New York,2004,ISBN 0471469122,9780471469124

Software Quality Assurance:From Theory to Implementation.Daniel Galin,Pearson Education,2004,ISBN 0201709457,9780201709452

GB/T 5094(所有部分) 工业系统、装置与设备以及工业产品 结构原则与参照代号

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

Risk Assessment and Risk Management for the Chemical Process Industry.H.R.Greenberg,J.J.Cramer,John Wiley and Sons,1991,ISBN 0471288829,9780471288824

B.2.6 规范的检查

注:在 GB/T 20438.2—2017 的表 B.1 和表 B.6 中引用了本技术/措施。

目的:为了避免规范中的不完备性和矛盾。

描述:检查是一种普通的技术,在这种技术中,由一个独立的小组来评估一份规范文档的各种质量。检查小组向拟制者提问题,拟制者必须圆满地回答提问人。检查小组的成员不包括规范的拟制人员。要求的独立程度由系统要求的安全完整性等级确定。独立的检查人员应能在不用涉及任何其他规范的情况下,以一种无可非议的形式,重建系统的运行功能。他们还必须检验在操作和组织措施中包括的所有相关的安全和技术问题,这个程序已被证明在实际应用中是很有效的。

参考文献:

IEC 61160:2005,Design review

The Art of Software Testing,Second Edition.G.Myers et al.,Wiley & Sons,New York,2004,ISBN 0471469122,9780471469124

Software Quality Assurance:From Theory to Implementation.D.Galin,Pearson Education,2004,ISBN 0201709457,9780201709452

B.3 E/E/PE 系统的设计和开发

整体目标:为了产生一个符合规范的安全相关系统的稳定的设计。

B.3.1 遵循指南和标准

注: 在 GB/T 20438.2—2017 的表 B.2 中引用了本技术/措施。

目的:为了遵循应用领域标准(GB/T 20438 中未规定)。

描述:在设计安全相关系统的过程中应遵循一些指南。这些指南首先可指导实现无失效的安全相关系统,其次可简化其后的安全确认。它们可以是通用的、某个项目专用的或者只专用于某单个阶段。

参考文献:

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

B.3.2 结构化设计

注: 在 GB/T 20438.2—2017 的表 B.2 和表 B.6 中引用了本技术/措施。

目的:通过建立部分要求的层次结构来减少复杂性,避免各要求之间的接口失效。简化验证。

描述:当设计硬件时,应使用专门的判据或者方法。例如需要:

——一种分层的结构化电路设计;

——使用已生产的和经测试的电路部件。

类似地,当设计软件时,使用结构图表使之能够建立软件模块的一种无歧义的结构。这种结构可显示模块的相互关系、模块之间传递的精确数据,和模块之间存在的精确控制。

参考文献:

GB/T 5094 工业系统、装置与设备以及工业产品 结构原则与参照代号

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

Software Design.D.Budgen,Pearson Education,2003,ISBN 0201722194,9780201722192

An Overview of JSD,J.R.Cameron,IEEE Trans SE-12 No.2,February 1986

Structured Development for Real-Time Systems (3 Volumes). P. T. Yourdon, P. T. Yourdon Press,1985

Structured Development for Real-Time Systems (3 Volumes). P. T. Ward, S. J. Mellor, Yourdon Press,1985

Applications and Extensions of SADT.D.T.Ross,Computer,25-34,April 1985

Essential Systems Analysis.St.M.McMenamin,F.Palmer,Yourdon Inc,1984

Structured Analysis(SA):A language for communicating ideas.D.T.Ross,IEEE Trans.Software Eng,Vol.SE-3(1),16-34

B.3.3 使用经试用并证明效果良好的元件

注: 在 GB/T 20438.2—2017 的表 B.2 和表 B.6 中引用了本技术/措施。

目的:通过使用具有专门特性的元件减小大量首次和未检测到的故障的风险。

描述:在安全性方面根据组件的可靠性,制造商选择经试用并证明效果良好的元件(例如使用经运行测试过的物理单元来满足高安全要求,或者仅在安全存储器中存储安全的程序)。存储器的安全性与未授权的访问和环境影响(电磁兼容性、辐射等)以及在发生一次失效的事件中组件的响应有关。

参考文献：

IEC 61163-1：2006，Reliability stress screening—Part 1：Repairable assemblies manufactured in lots

B.3.4 模块化

注：在 GB/T 20438.2—2017 的表 B.2 和表 B.6 中引用了本技术/措施。

目的：降低复杂性和避免失效，这些与子系统间接口有关。

描述：在设计的各层次上的每个子系统都被清楚地定义，并且限制它们的大小(仅几个功能)。子系统之间的接口尽可能保持简单，并把交叉部分(即共享数据、信息交换)减到最小。也限制了各个子系统的复杂性。

参考文献：

The Art of Software Testing，Second Edition. G. Myers et al.，Wiley & Sons，New York，2004，ISBN 0471469122，9780471469124

Software Engineering for Real-time Systems. J. E. Cooling，Pearson Education，2003，ISBN 0201596202，9780201596205

Software Reliability—Principles and Practices. G. J. Myers，Wiley-Interscience，New York，1976，ISBN-10：0471627658，ISBN-13：978-0471627654

B.3.5 计算机辅助设计工具

注：在 GB/T 20438.2—2017 的表 B.2 和表 B.6 及 GB/T 20438.3—2017 的表 A.4 和表 C.4 中引用了本技术/措施。

目的：为了更系统化地执行设计规程。包括已经可用的和经测试过的合适的自动结构组件。

描述：在硬件和软件设计过程中，应当使用可行的、并经复杂系统证明合理的计算机辅助设计工具(CAD)。这些工具的正确性应如此展示：通过专门测试、有大量成功应用历史，或者通过其输出的独立验证，这些输出来自正设计的特殊安全相关系统。

工具的选择应考虑它们的集成度。即，如果工具能协同工作，其中一个工具的输出以恰当的内容和格式自动地输入到随后的工具，则他们是集成的，这样就降低了中间结果二次导入时人为错误的可能性。

参考文献：

Overview of Technology Computer-Aided Design Tools and Applications in Technology Development，Manufacturing and Design. W. Fichtner，Journal of Computational and Theoretical Nanoscience，Volume 5，Number 6，June 2008，pp.1089-1105(17)

The Electromagnetic Data Exchange：Much more than a Common Data Format. P. E. Frandsen et al. In Proceeding of the 2nd European Conference on Antennas and Propagation. The Institution of Engineering and Technology(IET)，2007，ISBN 978-0-86341-842-6

Software engineering：Update. Ian Sommerville，Addison-Wesley Longman，Amsterdam；8th ed.，2006，ISBN 0321313798，9780321313799

Software Engineering. Ian Sommerville，Pearson Studium，8. Auflage，2007，ISBN 3827372577，9783827372574

B.3.6 模拟

注：在 GB/T 20438.2—2017 的表 B.2、表 B.5 和表 B.6 中引用了本技术/措施。

目的：为了对电气/电子电路元件的功能性能和正确度量两方面进行一次系统、全面的检查。

描述：借助一个软件行为模型，在一台计算机上仿真安全相关系统电路的功能。电路的各个元件每

个都有它们自己的模拟行为,并且通过观察每个元件的边缘数据来检验这些元件连成的电路的响应。

B.3.7 检查(复审和分析)

注:在 GB/T 20438.2—2017 的表 B.2 和表 B.6 中引用了本技术/措施。

目的:为了揭示规范和实现之间的不一致。

描述:检验和评价安全相关系统的规定功能以保证安全相关系统符合规范中给出的要求。有关实现的和产品使用的任何疑点都编入文档,因此这些疑点可得到解决。在检查过程中,同走查相比,作者是被动的,而检查员是主动的。

参考文献:

IEC 61160:2005,Design Review

Software engineering:Update.Ian Sommerville,Addison-Wesley Longman,Amsterdam;8th ed.,2006,ISBN 0321313798,9780321313799

Software Engineering.Ian Sommerville,Pearson Studium,8.Auflage,2007,ISBN 3827372577,9783827372574

The Art of Software Testing,Second Edition.G.Myers et al.,Wiley & Sons,New York,2004,ISBN 0471469122,9780471469124

ANSI/IEE 1028:1997,IEEE Standard for software reviews

Dependability of Critical Computer Systems 3.P.G.Bishop et al.,Elsevier Applied Science,1990,ISBN 1-85166-544-7

B.3.8 走查

注:在 GB/T 20438.2—2017 的表 B.2 和表 B.6 中引用了本技术/措施。

目的:为了揭示规范和实现之间的不一致。

描述:检验和评价安全相关系统设计的规定功能以保证安全相关系统符合规范中给出的要求。有关产品实现和使用的疑点和潜在的弱点都编入文档以便能得以解决。和检查相反,在走查过程中,作者是主动的而检查员是被动的。

参考文献:

Software engineering:Update.Ian Sommerville,Addison-Wesley Longman,Amsterdam;8th ed.,2006,ISBN 0321313798,9780321313799

Software Engineering.Ian Sommerville,Pearson Studium,8.Auflage,2007,ISBN 3827372577,9783827372574

ANSI/IEEE 1028:1997,IEEE Standard for software reviews

Dependability of Critical Computer Systems 3.P.G.Bishop et al.,Elsevier Applied Science,1990,ISBN 1-85166-544-7

Methodisches Testen von Programmen.G.J.Myers,Oldenbourg Verlag,München,Wien,1987

B.4 E/E/PE 系统操作和维护规程

整体目标:拟制有助于避免安全相关系统在操作和维护过程中失效的规程。

B.4.1 操作和维护说明书

注:在 GB/T 20438.2—2017 的表 B.4 中引用了本技术/措施。

目的:为了避免安全相关系统在操作和维护过程中出错。

描述:用户说明书包含了使用和维护安全相关系统的最基本的信息。在特殊情况下,这些说明通常还包括安装安全相关系统的例子。所有说明必须要容易读懂。复杂的操作步骤和相互关系可用图表描述。

参考文献:

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

B.4.2 用户友善性

注:在 GB/T 20438.2—2017 的表 B.4 中引用了本技术/措施。

目的:为了减小在操作安全相关系统过程中的复杂性。

描述:安全相关系统的正确操作与人的操作水平有关。通过考虑有关的系统设计和工作场地的设计,安全相关系统的开发者必须保证:

——尽量减少人为干预的需要;

——必要的干预应尽可能简单;

——把因操作员错误产生的潜在伤害降到最小;

——根据人机工程学的要求来设计干预设备和指示装置;

——操作员的设施应简单、标注清晰、使用直观;

——即使是在恶劣情况下,操作员也不用过度操劳;

——干预操作规程和设备的培训应适合受训用户的知识水平和技能。

B.4.3 维护友善性

注:GB/T 20438.2—2017 的表 B.4 中引用了本技术/措施。

目的:为了简化安全相关系统的维护规程并设计有效诊断和修理的必要方法。

描述:预防性维护和修理通常都是在最后限期内在困难的环境条件和压力下进行。因此,安全相关系统的开发人员应保证:

——尽量减少安全相关维护措施的必要性,在理想情况下,最好没有;

——对于不可避免的修理应具有足够的、切合实际并易于掌握的诊断工具——这些工具应包括所有必要的接口;

——如果必须开发或者购买单独的诊断工具,则应及时得到这些工具。

B.4.4 限制操作可能性

注:在 GB/T 20438.2—2017 的表 B.4 和表 B.6 中引用了本技术/措施。

目的:为了减少常规用户的操作可能性。

描述:本方法通过以下几点减少了操作可能性:

——限制特殊运行模式中的操作,例如通过按键开关;

——限制操作组件的数量;

——限制通常可能的运行模式数量。

参考文献:

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

B.4.5 只能由熟练的操作员操作

注:在 GB/T 20438.2—2017 的表 B.4 和表 B.6 中引用了本技术/措施。

目的:为了避免因使用错误引起的操作失效。

描述:要把安全相关系统的操作员培训到能适应安全相关系统的复杂性和安全完整性等级的程度。培训包括学习生产过程的基础知识和了解安全相关系统和受控设备之间的关系。

参考文献:

Guidelines for Safe Automation of Chemical Processes.CCPS,AIChE,New York,1993,ISBN-10:0-8169-0554-1,ISBN-13:978-0-8169-0554-6

B.4.6 防止操作员出错

注:在 GB/T 20438.2—2017 的表 B.4 和表 B.6 中引用了本技术/措施。

目的:防止操作员操作系统时的各种错误。

描述:通过真实性检查或者监视受控设备,可检测输入错误(值、时间等)。为了把这些设备汇集到设计中,在相当早的一个阶段就有必要说明哪些输入是可能的和哪些输入是允许的。

B.4.7 (未用)

B.4.8 修改保护

注:在 GB/T 20438.2—2017 的表 A.17 和表 A.18 中引用了本技术/措施。

目的:利用一些技术手段来防止修改安全相关系统的硬件。

描述:例如利用传感器信号的真实性检查、技术过程检测以及自动起动测试,自动地检测修改或变换。当检测到一个修改时,就采取应急动作。

B.4.9 输入确认

注:在 GB/T 20438.2—2017 的表 A.17 和表 A.18 中引用了本技术/措施。

目的:在受控设备动作之前,操作员可自己检查操作过程中的错误。

描述:在一个输入被传送给受控设备之前,经安全相关系统传送给受控设备的一个输入将反馈回给操作员,使得操作员有可能检测和校正一个错误。同时系统设计应考虑到异常的、无缘无故的人员动作的速度上下限以及人的反应倾向。这样可以避免例如操作员按键比预计的快而引起系统把一次双击读成一次单击,或者因为系统(显示)对快击反应太慢而把两次按键误读成一次。在输入关键数据时,连续地多次同样的击键并不有效。无限多次按"Enter(回车)"键或者"yes(肯定)"键也绝不会导致系统不安全的动作。

当操作员还未作决定并让系统等待时,为了提供必要的条件,应包含具有多个选择问题(是/否等)的暂停程序。

除非在设计软件及硬件时考虑到了这种需要,否则重新启动一个安全可编程电子系统的能力将使系统十分脆弱。

B.5 E/E/PE 系统集成

整体目标:为了避免在集成阶段产生失效以及揭示在本阶段和前面阶段产生的失效。

B.5.1 功能测试

注:在 GB/T 20438.2—2017 的表 B.3 和表 B.5 中以及 GB/T 20438.3—2017 的表 A.5、表 A.6、表 A.7、表 C.5、表 C.6 和表 C.7 中引用了本技术/措施。

目的:揭示在规范和设计阶段产生的失效。避免在软件和硬件的实现和集成期间产生失效。

描述:在功能测试过程中,应审查和观察是否已达到规定的系统特性。提供给系统的输入数据足以

说明一般预期的工作特性。观察输出并把它们的响应同规范给出的响应作对比。与规范的不符合性和表现出的规范的不完整性都被写进文档。

为多通道架构设计的电子元件的功能测试通常把制造的元件与已确认的伙伴元件一起进行检测，除此之外，建议结合同一批的其他伙伴元件一起测试成品元件，以便揭示共同模式故障，而用其他办法则该类故障会被掩藏而不能揭示出来。

另外，系统的工作能力必须足够，参看 C.5.20 中的指导。

参考文献：

Software Testing and Quality Assurance. K. Naik, P. Tripathy, Wiley Interscience, 2008, Print ISBN:9780471789116 Online ISBN:9780470382844

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122,9780471469124

Practical Software Testing: A Process-oriented Approach. I. Burnstein, Springer, 2003, ISBN 0387951318,9780387951317

Dependability of Critical Computer Systems 3. P. G. Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.5.2 黑盒测试

注：在 GB/T 20438.2—2017 的表 B.3、表 B.5 和表 B.6 中和 GB/T 20438.3—2017 的表 A.5、表 A.6、表 A.7、表 C.5、表 C.6 和表 C.7 中都引用了本技术/措施。

目的：为了检验实际功能条件下的动态行为。为了揭示失效从而满足功能规范和评估实用性和鲁棒性。

描述：在一个规定的环境中，利用规定的测试数据(这些数据是根据制定的判据从规范中系统导出的)执行一个系统或者程序的功能。这将揭露出系统的行为并允许同规范进行比较。不需使用系统内部结构的知识来指导测试。测试的目的是要确定功能单元是否能正确执行规范要求的所有功能。形成等价类的技术是黑盒测试数据判定的一个例子。借助规范可把输入数据空间细分成专用的输入值范围(等价类)。于是由下列情况构成各种测试用例：

——来自允许范围的数据；

——来自不允许范围的数据；

——来自范围极限的数据；

——极值；

——以上各类的组合。

为了选择各种测试活动(模块测试、集成测试和系统测试)中的测试用例，其他判据也可能是有效的。例如，在一个确认的框架内系统测试的判据“极值工作条件”是可信的。

参考文献：

Software Testing and Quality Assurance. K. Naik, P. Tripathy, Wiley Interscience, 2008, Print ISBN:9780471789116 Online ISBN:9780470382844

Essentials of Software Engineering. Frank F. Tsui, Orlando Karam. Jones & Bartlett, 2006. ISBN 076373537X,9780763735371

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122,9780471469124

Systematic Software Testing. Rick D. Craig, Stefan P. Jaskiel. Artech House, 2002. ISBN 1580535089,9781580535083

B.5.3 统计测试

注：在 GB/T 20438.2—2017 的表 B.3、表 B.5 和表 B.6 中引和了本技术/措施。

目的：为了检查安全相关系统的动态行为和评估它的实用性和鲁棒性。

描述：本方法使用根据实际操作输入预期的统计分布—操作分布图选择的输入数据来测试一个系统或程序。

参考文献：

A discussion of statistical testing on a safety-related application. S Kuball, J H R May, Proc. IMechE Vol.221 Part O: J.Risk and Reliability, Institution of Mechanical Engineers, 2007

Practical Reliability Engineering. P.O'Connor, D.Newton, R.Bromley, John Wiley and Sons, 2002, ISBN 0470844639, 9780470844632

Dependability of Critical Computer Systems 3. P.G.Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

Dependability of Critical Computer Systems 1. F.J.Redmill, Elsevier Applied Science, 1988, ISBN 1-85166-203-0

B.5.4 现场经验

注 1：在 GB/T 20438.2—2017 的表 B.3、表 B.5 和表 B.6 中引用了本技术/措施。

注 2：一种类似的办法另见 C.2.10，一种统计方法另见附录 D，二者都在软件的范围中。

目的：无论在 E/E/PE 系统集成过程中和/或 E/E/PE 系统安全性确认过程中，各种应用得到的现场经验可用作避免故障的一种办法。

描述：经验表明不会出现或仅可能出现不重要故障的元件或子系统，在许多不同应用中和足够长时期内使用时都不会有实质性改变。特别是对具有许多可能功能的复杂元件（例如操作系统、集成电路）而言，开发者应注意哪些功能已经历过现场测试。例如考虑故障检测的自测试例程：如在操作周期内，没有发生过硬件故障，就不能说例程已被测试过，因为它们还从未经受过故障检测。

为了使用现场经验，必须满足以下要求：

——规范不变；

——在各种应用中要有 10 个系统；

——运行时间要有 10^5 h 和至少 1 年的服役期。

注 3：不同的行业标准可以指定不同的数字。

通过卖方和（或）操作公司的文件证明现场经验，该文件必须至少要包括：

——系统和它的元件的准确名称，包括硬件的版本控制；

——用户和应用时间；

——运行时数；

——选择为完成证明的系统和应用的规程；

——故障检测、故障登记以及故障排除规程。

参考文献：

IEC 60300-3-2:2004, Dependability management—Part 3-2: Application guide—Collection of dependability data from the field

Guidelines for Safe Automation of Chemical Processes. CCPS, AIChE, New York, 1993, ISBN-10: 0-8169-0554-1, ISBN-13: 978-0-8169-0554-6

B.6 E/E/PE 系统安全性确认

整体目标:为了证明 E/E/PE 安全相关系统符合 E/E/PE 系统安全要求规范和 E/E/PE 系统设计要求规范。

B.6.1 在环境条件下测试功能

注:在 GB/T 20438.2—2017 的表 B.5 中引用了本技术/措施。

目的:为了评估安全相关系统是否能耐受典型的环境影响。

描述:把系统置于各种环境条件下(例如根据 IEC 60068 系列或 IEC 61000 系列标准),评价安全功能的可靠性(以及和上述标准的相容性)。

参考文献:

GB/T 2421.1—2008 电工电子产品环境试验 概述和指南。第 1 次修改(1992)

IEC 61000-4-1:2006, Electromagnetic compatibility(EMC)—Part 4-1: Testing and measurement techniques—Overview of IEC 61000-4 series

Dependability of Critical Computer Systems 3. P.G. Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.6.2 浪涌抗扰性测试

注:在 GB/T 20438.2—2017 的表 B.5 和表 B.6 中引用了本技术/措施。

目的:为了检验安全相关系统应付峰值浪涌的能力。

描述:系统装入一个典型的应用程序,使所有外部线路(所有数字、模拟和串行接口以及总线连接和电源等)都必须得到标准的噪声信号。为了获得一个定量的说明,明智的作法是仔细地逼近浪涌极限。如果功能失效,则不符合所选择噪声级别的要求。

参考文献:

GB/T 17626.5—2008 电磁兼容 试验和测量技术 浪涌(冲击)抗扰度试验

C37.90.1—2002, IEEE Standard for Surge Withstand Capability(SWC) Tests for Relays and Relay Systems Associated with Electric Power Apparatus

B.6.3 (未用)

B.6.4 静态分析

注:在 GB/T 20438.2—2017 的表 B.5 和表 B.6 中及 GB/T 20438.3—2017 的表 A.9、表 B.8、表 C.9 和表 C.18 中引用了本技术/措施。

目的:为了避免系统性故障,此故障可导致受试系统在工作多年后,迟早都会发生事故。

描述:这种系统性的和可能用计算机辅助的方法可检查原型系统的特有静态特性从而可保证所讨论的要求的完备性、一致性和无歧义性(例如构造指南、系统规范和仪表数据表)。静态分析是可再现的。它适用于已达到完成一个很好定义阶段的原型机。对于硬件和软件的静态分析的一些例子有:

——数据流的一致性分析(比如测试一个数据对象是否在任何地方都被解释成同一值);

——控制流分析(比如路径确认,不可存取代码确认);

——接口分析(比如调查各种软件模块之间变量传递);

——数据流分析以检测可疑的变量建立、引用、删除顺序;

——遵守专用指南的测试(例如爬电距离和间隙、装配距离、物理单元的布置、机械敏感物理单元、曾经引入的专用物理单元)。

参考文献：

Static Analysis and Software Assurance.D.Wagner,Lecture Notes in Computer Science,Volume 2126/2001,Springer,2001,ISBN 978-3-540-42314-0

An Industrial Perspective on Static Analysis.B A Wichmann,A A Canning,D L Clutterbuck,L A Winsborrow,N J Ward and D W R Marsh.Software Engineering Journal.,69—75,March 1995

Dependability of Critical Computer Systems 3.P.G.Bishop et al,.Elsevier Applied Science,1990,ISBN 1-85166-544-7

B.6.5 动态分析和测试

注：在 GB/T 20438.2—2017 表 B.5 和表 B.6 中以及 GB/T 20438.3—2017 的表 A.5、表 A.9、表 B.2、表 C.5、表 C.9 和表 C.12 中引用了本技术/措施。

目的：通过检查样机在即将完成阶段的动态特性来检测规范失效。

描述：通过把计划的工作环境下的一些典型的输入数据加给安全相关系统的一个接近运行的样机进行安全相关系统的动态分析。当观察到的安全相关系统的行为同要求的行为一致时，分析是满意的。必须校正安全相关系统的任何失效，然后必须重新分析新的运行版本。

参考文献：

The Concept of Dynamic Analysis.T.Ball,ESEC/FSE '99,Lecture Notes in Computer Science,Springer,1999,ISBN 978-3-540-66538-0

Dependability of Critical Computer Systems 3.P.G.Bishop et al,.Elsevier Applied Science,1990,ISBN 1-85166-544-7

B.6.6 失效分析

注：在 GB/T 20438.2—2017 的表 B.5 和表 B.6 中引用了本技术/措施。

B.6.6.1 失效模式和影响分析(FMEA)

目的：通过系统性的检查一个系统的元件的所有可能失效源以及确定这些失效对系统的行为和安全性的影响来分析一个系统的设计。

描述：通常是通过一次工程师会议来进行分析。依次分析一个系统的每个部件从而给出部件的一组失效模式、失效的起因和影响(局部和整体系统级)、检测程序和建议。如果建议被接受，它们将作为采取的补救行动编入文档。

参考文献：

GB/T 7826—2012 系统可靠性分析技术 失效模式和影响分析(FMEA)程序

Risk Assessment and Risk Management for the Chemical Process Industry.H.R.Greenberg,J.J.Cramer,John Wiley and Sons,1991,ISBN 0471288829,9780471288824

Reliability Technology.A.E.Green,A.J.Bourne,Wiley-Interscience,1972,ISBN 0471324809

B.6.6.2 因果图

注：在 GB/T 20438.3—2017 的表 B.3、表 B.4、表 C.13 和表 C.14 中引用了本技术/措施。

目的：为了以一种紧凑的图表形式分析和建立一个系统中可能形成的事件序列的模型，此事件序列是基本事件组合的结果。

描述：本技术可看成是故障树和事件树的一种组合分析。从一个致命事件开始，一幅因果图通过描述运行成功或失败的是/否门被向前跟踪。这样就创建了事件或受控情形的事件结果。结果图(如事件树)也就为每一次失效所创建。把致命事件作为顶端事件，从致命时间开始和向后跟踪能画出整体的事

件树。在正向跟踪中,确定了由一个事件产生的可能后果。图形包含顶点符号,这些符号描述了从顶点开始沿各分支传播的条件,也能包括时间延迟。还可用故障树来描述这些条件。为了使图形更简洁,传播路线可与逻辑符号组合。定义了一个标准的符号集合以供因果图使用。图表可用于生成故障树,并可用这些图来计算发生某些致命后果的概率。它也可以被用来产生事件树。

参考文献:

IEC 62502,Analysis techniques for dependability—Event tree analysis(ETA)[1]

The Cause Consequence Diagram Method as a Basis for Quantitative Accident Analysis.B.S.Nielsen,Danish Atomic Energy Commission,Riso-M-1374,1971

B.6.6.3 事件树分析(ETA)

注:在 GB/T 20438.3—2017 的表 B.4 和表 C.14 中引用了此技术/措施。

目的:为了用图表形式建立事件序列的模型,这种事件序列是在一个起始事件之后,在一个系统中可能产生的;并因此也为了指示可能产生的严重后果。事件树是难以从头建立且使用后果图是很有帮助的。

描述:在图顶部记录了紧随起始事件之后的相继事件有关的序列条件。在序列中从起始事件下面开始(它们是分析的目标)到第 1 个条件,画了一条线。在那里,图分成“Yes(是)”和“No(否)”两个分支,它们描述了以后的事件对条件的依从情况。对这两个分支来说,每个都以类似的方式继续到下一条件。但不是所有的条件都和分支有关。一个分支继续到序列的末端,并且按这种方式构造的树的每个分支代表一个可能的后果。在序列中提供的条件是独立的,根据序列中的条件的概率和数目,可用事件树来计算各种后果的概率。作为条件很少是完全独立的,这种计算应由熟练的分析人员来进行并慎重考虑。

参考文献:

IEC 62502,Analysis techniques for dependability—Event tree analysis(ETA)[2]

Risk Assessment and Risk Management for the Chemical Process Industry.H.R.Greenberg,J.J.Cramer,John Wiley and Sons,1991,ISBN 0471288829,9780471288824

B.6.6.4 失效模式,影响和危害性分析(FMECA)

注:在 GB/T 20438.3—2017 的表 A.10、表 B.4、表 C.10 和表 C.14 中引用了本技术/措施。

目的:为了确定在设计或者操作过程中需要特别关注哪些部件和采取哪些必要的控制措施,需要评定部件的危害性等级,这种等级通过单点失效将毁坏、损害系统或者使系统功能下降。

描述:这种方法与 FMEA 相当,但它有一个或多个列来说明其危害性,并有许多方法进行评定。最耗费人力的方法是美国汽车工程师协会(SAE)在 ARP926 中描述的方法。在该规程中,由发生在一个危害性模式中每百万次运行过程中预定的某种特殊类型的失效数来表示任一部件的危害性数值。危害性数值是 9 个参数的一个函数,它们中的大多数都是必须测量的。确定危害性的一种很简单的方法是部件失效的概率乘以可能发生的损坏;这种方法类似于简单的风险因数评估法。

参考文献:

GB/T 7826—2012 系统可靠性分析技术 失效模式和影响分析(FMEA)程序

Software criticality analysis of COTS/SOUP.P.Bishop,T.Clement,S.Guerra.In Reliability Engineering & System Safety,Volume 81,Issue 3,September 2003,Elsevier Ltd.,2003

Software FMEA techniques.P.L.Goddard.In Proc Annual 2000 Reliability and Maintainability Symposium,IEEE,2000,ISBN:0-7803-5848-1

B.6.6.5 故障树分析(FTA)

注:在 GB/T 20438.3—2017 的表 B.4 和表 C.14 中引用了本技术/措施。

目的:帮助分析将会导致危险的严重结果的事件或事件组合,并计算顶端事件的发生概率。

描述:从一个可能直接引起危险或者严重后果的事件("顶端事件")开始分析下去,找到导致该事件的原因,这需要通过使用逻辑运算(与,或等)的几个步骤来完成。并按同样的方法分析中间原因,直到基本事件,分析停止。

本方法是图形法并使用了一组标准化的符号来绘制事件树。在分析的最后,故障树表示出了从基本事件(一般部件失效)到顶端事件(整个系统的失效)的功能连接关系。本技术原来主要用来分析硬件系统,但也可适用于软件失效分析。这种技术可以用于定性故障分析(识别失败的情景:基本项或最小割集),半定量(根据其概率进行场景的排序)和定量计算顶端事件的概率(见 C.6)。

参考文献:

IEC 61025:2006,Fault tree analysis(FTA)

From safety analysis to software requirements. K.M. Hansen, A.P. Ravn, A.P, V Stavridou. IEEE Trans Software Engineering, Volume 24, Issue 7, Jul 1998

B.6.6.6 马尔可夫(Markov)模型

注:在分析硬件安全完整性的时候,可靠性方框图要参考 GB/T 20438.6—2017 的 B.1 使用这项技术。

目标:通过状态转换图对系统行为进行建模并且对概率方面的系统参数(如不可靠度、不可用度、MTTF、MUT、MDT 等)进行评估。

描述:在有向图中它是一个有限态自动机(参见 B.2.3.2)。节点(圆圈)代表状态,在节点间的边(箭头)表示发生在不同状态之间的转换(如失效、维修等)。这些边都被附上不同的权值来表示对应的失效率和维修率。齐次马尔可夫过程的基本属性是表明系统的接下来的状态只和现在的状态有关:即由状态 N 到状态 $N+1$ 的变化和前一个状态 $N-1$ 无关。这表明所有模型的概率规则都是指数类型的。

所有的失效事件、失效状态、失效率都可以用这种方法来描述,从而能够获得对系统的精确描述,例如检测或没检测到的失效事件或者是一种更严重的失效等。检验测试间隔也可以用多相马尔可夫过程来建模,即某相末端状态(例如在检测测试之前)的概率可以用来作为计算下一相的初始条件(例如在检验测试执行之后各种状态的概率)。

马尔可夫技术对于那些由于组件失效或维护而使冗余等级随时间变化的多系统来说是一种很好的建模工具。其他如故障树分析(FTA)、失效模式与影响分析(FMEA)等经典方法由于没有计算对应可能性的组合公式,所以不能用于对在系统整个运行周期产生的失败影响进行合适的建模。

在最简单的情况下,描述系统概率的公式可以在手册中查到或者手工推导而得到,并且一些简化方法(如减少状态的数量)也可以被应用到更加复杂的情况中。

然而,从数学角度上说,齐次马尔可夫图仅仅是对常系数线性微分方程的简化和常规设定。这种方法已经被研究了相当长的时间并且有强大的理论基础来操作它们。因此,当模型的规模增长时使用上述方法是十分有效的,并且它们可以在各种各样的软件包中方便地使用。

我们知道图表的大小会随着器件数量以指数增长:这就是所谓的组合爆炸。如果要实现精确的应用,这种方法仅仅适用于小系统。

当不得不使用非指数规则时(半马尔可夫模型),就可以使用蒙特卡洛模拟(参见 B.6.6.8)。

参考文献:

IEC 61165:2006,Application of Markov techniques

The Theory of Stochastic Processes. R. E. Cox and H. D. Miller, Methuen and Co. Ltd., London, UK, 1963

Finite MARKOV Chains. J. G. Kemeny and J. L. Snell. D. Van Nostrand Company Inc, Princeton, 1959

The Theory and Practice of Reliable System Design. D. P. Siewiorek and R. S. Swarz, Digital

Press,1982

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès-Lavoisier, 2009, ISBN:978-2-7462-1991-5

B.6.6.7 可靠性框图(RBD)

注1:在GB/T 20438.6—2017的附录B中引用了本技术/措施。

注2:参考C.6.4中的"可靠性框图"。

目的:为了以图形化格式对一组必然发生的事件和系统或任务成功运行的条件进行建模。与其说是一种分析方法,不如说它是一种表示方法。

描述:分析的目标一般被表述为包含块、线以及逻辑连接的成功路径。这种路径由图表的一端开始并通过块和节点到达图表的另外一端。其中块表示一种条件或事件,如果它们为真或者发生了,则路径导通。如果路径遇到了一个节点,如果满足节点的逻辑它就继续导通。当它到达一个端点时,可以沿着流出线继续导通。如果图表中存在着至少一条成功的路径,那么分析的目标正在正确地执行。

可靠性框图是对已建模的系统的结构化表达。它是这样一种电路:当电流找到了一条从输出到输入的通路,就意味着已建模的系统工作正常;当电路被切断的时候就表示已建模的系统失败。这就产生了最小割集的理论,它表示了各种错误(例如可靠性框图被"割断"的地方)导致了已建模系统的失败。

数学上讲,可靠性框图和故障树类似。它表示了把各个独立的元件(工作或失效)和整个系统状态(工作或失效)连接在一起的逻辑功能。因此,它的计算就如同故障树中的一样。

参考文献:

IEC 61078:2006, Analysis techniques for dependability—Reliability block diagram and boolean methods

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès-Lavoisier, 2009, ISBN:978-2-7462-1991-5

B.6.6.8 蒙特卡洛(Monte-Carlo)模拟

注:在GB/T 20438.3—2017的附录B.4和GB/T 20438.6—2017的附录B中引用了本技术/措施。

目的:当一般的分析方法不奏效时通过产生随机数字来对真实世界现象进行模拟。

描述:蒙特卡洛仿真通常用来解决下列两种问题:

——概率问题,即用随机数字来产生随机现象;

——确定性问题,它被数学性的转化为一个等价的概率问题(比如积分计算)。

蒙特卡洛模拟理论是用随机数字来表达出正常和非正常的系统模型。这些行为模型是通过状态转移模型(马尔可夫图、佩特里网、形式化语言等)提供的。蒙特卡洛仿真法就是通过从已被观测的统计结果中生成大量的统计样本来完成的。

在使用蒙特卡洛模拟法的时候要注意偏差、允差、噪声都要取合理的数值。这要通过置信区间来实现,而置信区间可以很容易地从仿真中观察到。和一般的分析方法相反,蒙特卡洛仿真法可以完成自我逼近。小概率事件基本不会发生因此就没有必要定义,从而简化了模型。

蒙特卡洛模拟方法的一般原则是重新声明和重新表示问题,这样相比于追踪初始化问题能获得更加精确的结论。

在上文中所示,标准的蒙特卡洛仿真法不仅可以用来计算安全完整性等级(SIL),还能够处理可靠性数据的不确定度。就如今的计算机来说,SIL 4级的计算很容易达到。

参考文献:

Monte Carlo Methods. J.M. Hammersley, D.C. Handscomb, Chapman & Hall, 1979

Sécurisation des architectures informatiques. Jean-Louis Boulanger, Hermès-Lavoisier, 2009,

ISBN:978-2-7462-1991-5

B.6.6.9 故障树模型

注 1：参照 GB/T 20438.6 在硬件完整性分析中使用这种技术。

注 2：故障树的方法在上面已经被描述为安全性确认的方法(参见 B.6.6.5)。同时他们也在失效分析和概率计算中得到广泛地应用。

目的：为了通过系统性的、自上而下的图形化(因-果)方法建立连接基本事件(失效模式)和顶端事件(意外事件)的逻辑功能。

描述：它不仅仅是一种为了一步一步地帮助分析提高模型的方法，也是为了概率计算而使用的一种数学模型。表述如下：

——通过识别和分类失效情况(基本项或最小割集)来定性分析；

——根据事件发生的可能性来排序，完成半定量分析；

——通过计算顶端事件的可能性来定量分析。

像可靠性框图(RBD)一样，故障树是连接各个元件状态(失效或工作)和整个系统状态(失效或工作)的逻辑(布尔型)功能。因此，当元件为独立情况的时候，可以通过把基本概率性质应用于逻辑功能来完成概率计算。由于静态模型只能在真实(即常量)概率情况下工作，所以这种方法并不是非常容易。依时间变化的概率应该小心处理。例如，那些包含周期性检验测试元件的安全系统的 PFD_{avg} 不能被直接计算出，即使是对于那些工作在连续模型的安全系统的 PFH 来说也是很困难的。因此，只有那些具备良好数学基础的可靠性工程师才能用这种方法进行不可用度/PFD 和不可靠度/PFH 的计算。

对于简单的故障树来说计算可以用手工完成，但是在近 50 年来又出现了许多求解复杂逻辑公式的方法。现今我们一般使用二元决策图(BDD)的方法来解决，它是一种被写入电脑内存的逻辑公式的精简代码，也是现阶段在工业系统中进行精确的概率计算的唯一方法。同时，这种方法也足够快速使我们能够求解蒙特卡洛仿真的不确定度。

参考文献：

IEC 61025:2006，Fault tree analysis(FTA)

B.6.6.10 广义随机佩特里网模型(GSPN)

注 1：参照 GB/T 20438.6 在硬件完整性分析中使用这种技术。

注 2：佩特里网已经被作为半形式化方法提出(见 B.2.3.3)。它们也可以在硬件安全完整性中较好地使用。

目的：为了图形化的建立接近真实模型系统的正常功能和非正常功能的模型，以便向蒙特卡洛仿真提供有效的支持。

描述：如 B.2.3.3 中所示，这是一个异步有限状态自动机，除了当建模一个安全系统的非正常安全功能时，执行半形式化确认没有好的属性追踪，这是由于安全系统的非、半正规模型导致的。这些所谓的地点(图中画圈表示)表示了潜在状态，所谓的过渡(图中画正方形表示)表示了有可能发生的事件。此外，被标记的地点(见 B.2.3.3)、信息或判断可以用来使能这些过渡的，从过渡完成到过渡开始的时间可以是确定的或者随机的。这就是为什么这些佩特里网络被叫做“广义随机”的佩特里网络。

佩特里网络组成了灵活的行为模型，因而被证明能很好地支持蒙特卡洛仿真(见 B.6.6.8)。除了蒙特卡洛仿真自身的精度总是已知的，所有其他方法(从属关系、组合爆炸、非指数法则等等)的限制就可以被克服了。按照现今电脑发展水平来说解决 SIL 4 的方程问题也不是什么难事儿了。

参考文献：

IEC 62551，Analysis techniques for dependability—Petri net modelling(CD1)3

Sécurisation des architectures informatiques. Jean-Louis Boulanger，Hermès-Lavoisier，2009，ISBN:978-2-7462-1991-5

B.6.7 最坏情况分析

注：在 GB/T 20438.2—2017 的表 B.5 和表 B.6 中引用了本技术/措施。

目的：为了避免由环境条件和部件公差不相宜的组合引起的系统失效。

描述：根据一种理论检验和计算系统的工作能力和确定的部件尺寸。把环境条件改变到它们允许的最高边缘值。检查系统最实质性的反响并把这些响应同规范进行比较。

B.6.8 扩展的功能测试

注：在 GB/T 20438.2—2017 的表 B.5 和表 B.6 中引用了本技术/措施。

目的：为了揭示规范、设计和开发阶段的失效。检验安全相关系统在稀有的，或者非规定的输入事件中的行为。

描述：扩展的功能测试将评审安全相关系统响应输入条件的功能行为，这些输入条件预计是很少见的(例如主要失效)或者它们是超出安全相关系统规范之外的(例如操作错误)。对稀有条件的情况，观察到的安全相关系统的行为将同规范进行比较。在未规定安全相关系统的响应的情况下，应检查由观察到的响应维持的设备安全性。

参考文献：

Software Testing and Quality Assurance. K. Naik, P. Tripathy, Wiley Interscience, 2008, Print ISBN:9780471789116 Online ISBN:9780470382844

The Art of Software Testing, Second Edition. G. Myers et al., Wiley & Sons, New York, 2004, ISBN 0471469122, 9780471469124

Dependability of Critical Computer Systems 3. P.G. Bishop et al., Elsevier Applied Science, 1990, ISBN 1-85166-544-7

B.6.9 最坏情况测试

注：在 GB/T 20438.2—2017 的表 B.5 和表 B.6 中引用了本技术/措施。

目的：为了测试在最坏情况分析过程中规定的情况。

描述：在最坏情况下测试系统工作能力和确定的部件尺寸。环境条件改变到它们的最高容限值，检查系统最重要的响应并把它们同规范进行比较。

B.6.10 故障插入测试

注：在 GB/T 20438.2—2017 的表 B.5 和表 B.6 中引用了本技术/措施。

目的：为了在系统硬件中引入或者模拟故障并把响应编入文档。

描述：这是评估可靠性的一种定性方法。为了描述故障位置和类型以及怎样引入它，最好使用详细的功能块、电路和接线图；例如：可切断各模块的电源；可开路或短路电源线、总线、地址线；可开路或短路各部件或者它们的端口；可使继电器闭合或断开失效、或者在错误的时间开合等。例如可按 IEC 60812 的表Ⅰ和表Ⅱ中所示对产生的系统失效进行分类。原则上讲，只引入单稳态故障。然而，在内部诊断测试并未揭示出一个故障或者一个故障并未变得一目了然的情况下，它可能留在系统中并且要考虑到第二个故障的影响。故障数轻易就可增加到几百个。

应参考系统的多学科工作组和卖方目前所作的工作。应计算或估算对有严重后果的故障两次失效之间的平均工作时间。如果算出的时间很短，则应进行修改。

参考文献：

GB/T 7826—2012 系统可靠性分析技术 失效模式和影响分析(FMEA)程序

GB/T 18272.5—2000 工业过程测量和控制 系统评估中系统特性的评定 第 5 部分：系统可信性评估。

附 录 C
（资料性附录）
实现软件安全完整性的技术和措施的综述（参看 GB/T 20438.3）

C.1 概述

本附录中包含的技术的综述既不能被认为是完整的也不能认为是详尽的。

C.2 要求和详细的设计

注：在 B.2 中可找到有关的技术和措施。

C.2.1 结构化图解方法

注：在 GB/T 20438.3—2017 的表 A.2 和表 A.4 中引用了本技术/措施。

C.2.1.1 概述

目的：结构化方法的主要目的是通过把注意力集中到生命周期的初期阶段来提高软件开发的质量。本方法旨在通过精确的和直观的规程和符号（计算机辅助的）来达到该目的，从而可用一条逻辑顺序和一种结构化形式确定和记录要求并实现特征。

描述：存在许多结构化方法。一些方法打算用于传统的数据处理和事务处理功能，而另一些方法更适于过程控制和实时应用（倾向于更严格的安全性）。UML（见 C.3.12）包含许多结构化符号的例子。

结构化方法实质上是“思维工具”，旨在系统地发觉和划分一个问题或系统。它们的主要特征如下：

——具有逻辑顺序的思维方式，即把一个大问题分解为可管理的一些阶段；

——总系统的分析和建立文档，包括环境及要求的系统。

——要求系统中的数据和功能的分解；

——检查表，即需要分析的项目的分类表；

——低智能内务操作——简单、直观、实用；

——通常重点强调开发一个预定系统的图解模型，并且计算机辅助软件工程（CASE）工具支持全部的方法。

分析和记录问题和系统实体（例如过程和数据流）的支持符号往往是准确的，但是表达这些实体执行的处理功能的符号往往是非形式的。但某些方法确实部分地使用了（数学上的）形式化符号（例如，规范的表达或有限状态机）。精确性的提高不仅缩小了错误理解的范围，还提供了自动处理的范围。

结构化符号的另一个好处是它们的直观性，它能使用户直观地检查一个规范或设计，而不需要用户强大但未声明的知识。

本附录较详细地描述了一些结构化方法。

参考文献：

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

Software Design.D.Budgen,Pearson Education,2003,ISBN 0201722194,9780201722192

C.2.1.2 CORE——受控的要求表达（Controlled Requirements Expression）

目的：为了保证能确定和表示所有的要求。

描述:本方法想用来在买方/最终用户和分析员之间连接一座桥梁。它在数学上并不严密但有助于交流——CORE是为要求表达而不是为规范而设计的。本方法是结构化的,其表达经过各个精细化阶段。CORE法鼓励对问题的各种看法,引入系统使用的环境知识以及各种类型用户的不同观点。CORE包括识别背离"总设计"的导则和手段。可以校正或者明显地标识这些背离并把它们编入文档。因而规范可能不完整,但可检测未辨别出的问题和高风险区域,它们是在其后的设计中必须要考虑。

参考文献:

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

Requirements Engineering. E. Hull, K. Jackson, J. Dick. Springer, 2005, ISBN 1852338792, 9781852338794

C.2.1.3 JSD——杰克逊系统开发(Jackson System Development)

目的:包括从要求一直到代码的软件系统(特别着重于实时系统)的一种开发方法。

描述:JSD是一种分阶段的开发规程,在这个过程中开发者依据真实世界行为进行建模,系统功能则以这些行为为基础,确定所需的功能并把它们插入模型,同时把产生的规范变换成目标环境中可实现的规范。因此它包括规范和设计及开发的传统阶段,但也采用了与传统的自顶向下方法有些不同的观点。

此外,它特别着重于发现真实世界中实体的初始阶段,该真实世界与正在建立的系统有关并关系到建造它们的模型以及它们可能发生什么情况。一旦已完成"真实世界"的这种分析和建立起一个模型,就能分析系统所需功能,从而确定怎样把它们归纳入这个真实世界模型中。产生的系统模型随模型中所有过程的结构化描述而扩充,并且整个模型被变换成将在目标软件和硬件环境中运行的一些程序。

参考文献:

Systems Analysis and Design. D. Yeates, A. Wakefield. Pearson Education, 2003, ISBN 0273655361,9780273655367

An Overview of JSD.J.R.Cameron.IEEE Transactions on Software Engineering,SE-12,No.2,February 1986

C.2.1.4 实时 Yourdon(Real-time Yourdon)

目的:实时系统的规范和设计。

描述:作为本技术基础的开发方案假定正在开发的一个系统有三阶段进化。第一阶段包含建立一个"基本模型",此模型描述了系统所需的行为。第二阶段包含建立一个描述结构和机制(即当实现时,具体化要求的行为)的实现模型。第三阶段涉及实际实现系统的硬件和软件。这三个阶段大体相当于传统的规范和设计及开发阶段,不过更为着重于每一阶段开发者从事于建立一个模型的活动。

基本模型有两部分:

——环境模型,方案包含描述系统和它的环境之间的边界以及描述系统必须作出响应的外部事件;

——行为模型,方案包含描述系统响应事件进行的变换以及描述系统为了响应必须保持的数据。

实现模型还分成一些子模型,它们包含分配给处理机的各个过程以及这些过程分解成的软件模块。

为了获得这些模型,本技术结合了许多其他众所周知的技术:数据流图、变换图形、结构化英语、状态转换图和佩特里(Petri)网。另外,本方法包含根据已画出的模型,在纸上或者用机器模拟一个建议的系统设计的技术。

参考文献:

Real-time Systems Development. R. Williams. Butterworth-Heinemann, 2006, ISBN 0750664711, 9780750664714

Structured Development for Real-Time Systems(3 Volumes). P. T. Ward and S. J. Mellor. Yourdon Press, 1985

C.2.2 数据流图

注：在 GB/T 20438.3—2017 的表 B.5 和表 B.7 中引用了本技术/措施。

目的：为了用一个图表形式描述流过一个程序的数据流。

描述：数据流图记录了数据输入是怎样变换成输出的，图中的每一步表示一个不同的转换。

数据流图由三种组元构成：

——带注释的箭头——代表进出转换中心的数据流，注释记录了是什么数据；

——带注释的圈——代表转换中心，注释记录了转换；

——运算符[and(与)，xor(异或)]——这些算符被用来连接带注释的箭头。

数据流图中的每个圈可看作是一个可独立应用的黑盒，只要它的输入可用，它就可把这些输入变换成它的输出。主要的优点之一就是他们显示转换，但不用对怎样实现这些转换作任何假定。一幅纯数据流图并不包括控制信息或者排序信息，但它便于实时的扩充，比如实时 Yourdon(参看 C.2.1.4)。

编制数据流图的最佳方法是考虑系统的输入及其转化为输出的过程。每个圈必须代表一个不同的变换——它的输出应与它的输入在某些方面有所不同。构建一幅数据流图是一个系统设计创造性部分，并不存在确定图的总体结构的规则。同所有设计一样，它是通过逐步精细化初始设计从而产生最后图形的一个迭代过程。

参考文献：

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

GB/T 1526—1989 信息处理 数据流程图、程序流程图、系统流程图、程序网络图和系统资源图的文件编制符号及约定

ISO/IEC 8631:1989, Information technology—Program constructs and conventions for their representation

C.2.3 结构图

注：在 GB/T 20438.3—2017 的表 B.5 中引用了本技术/措施。

目的：为了用图形显示一个程序的结构。

描述：结构图是一种符号表示法，此法是数据流图的补充。结构图像树一样，描述了编程系统和各部分的一个分层结构，用图形展示了这种结构。它们记录了怎样按程序单元的一种层次结构来实现数据流图的元素。

一幅结构图显示了程序模块之间的关系而不包含这些单元启动顺序的任何信息，画这些图时使用了以下 4 种符号：

——注释有模块名的矩形；

——连接这些矩形从而创建结构的线；

——带环的箭头(空心的)，注释有传给和来自结构图中各元素的数据名(通常平行于连接图中各矩形的线画带环的箭头)。

——带环的箭头(实心环)，注释有从结构图中的一个模块到另一模块的控制信号名，也是平行于连接图中两个模块的线画箭头。

从任一非平凡数据流图有可能派生出许多不同的结构图。

数据流图描绘了系统中的信息和功能之间的关系,结构图描绘了实现系统元素的途径。两种技术虽然分别表示了系统的不同侧面,但都是有效的。

参考文献:

Software Design & Development.G.Lancaster.Pascal Press,2001,ISBN 1741251753,9781741251753

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

C.2.4 形式化方法

注:在 GB/T 20438.3—2017 的表 A.1、表 A.2、表 A.4 和表 B.5 中引用了本技术/措施。

C.2.4.1 一般要求

目的:采用基于数学的一种方法来开发软件。它包括形式设计和形式编码技术。

描述:形式化方法提供了一种在系统规范、设计或实现的某个阶段描述一个系统的方法。产生的描述的形式是一种严密的符号表示法,它可通过数学分析检测各种类型的不一致性或者不正确性,此外,在某些情况下可用机器分析该描述,其精确性类似于使用编译器对源程序进行语法检查的精度,或者可把该描述制作成动画从而显示所描述的系统各方面的行为,动画可给出对系统满足真实要求及形式上规定的要求的额外置信度,这是因为它提高了人们对规定行为的识别能力。

形式化方法通常将提供一种表示法(通常使用离散数学的某种形式)和一种用来产生该表示法的描述技术,以及用来检查各种正确性属性描述的各种分析形式。

在本条的以下小节中描述了几种形式化方法——CCS、CSP、HOL、LOTOS、OBJ、时序逻辑、VDM 和 Z。注意,其他的技术,比如有限状态机和佩特里(Petri)网(见附录 B),也可根据使用该技术时,这些技术符合某个精确的数学基础的严密程度,把它们看成是形式化方法。

参考文献:

Formal Specification: Techniques and Applications. N. Nissanke, Springer-Verlag Telos, 1999, ISBN 1852330023

The Practice of Formal Methods in Safety-Critical Systems.S.Liu, V.Stavridou, B.Dutertre, J.Systems Software 28,77—87,Elsevier,1995

Formal Methods: Use and Relevance for the Development of Safety-Critical Systems. L. M. Barroca, J.A.McDermid, The Computer Journal 35(6),579—599,1992

How to Produce Correct Software—An Introduction to Formal Specification and Program Development by Transformations.E.A.Boiten et al, The Computer Journal 35(6),547—554,1992

C.2.4.2 通信系统的演算(CCS)

目的:CCS 是描述和推断并行通信过程系统行为的一种方法。

描述:CCS 是有关系统行为一种数学演算。系统设计建模为按顺序运行或者并行运行的独立过程的一个网络。这些过程可通过端口(类似于 CSP 的通道)进行通信,只有在两个过程准备就绪时才能进行通信。不确定性也能建立模型。从整个系统的一个高层抽象描述(称为一条轨迹)开始,可能进行系统的逐步细化,使之成为通信过程的一个组合体,它的总行为就是整个系统所需的行为。同样,也可以按自下而上的方式组合各过程,并使用与合成规则有关的推理规则来演绎最后得到的系统的属性。

参考文献:

Communication and Concurrency.R.Milner.Pearson Education,1989,ISBN 9780131150072

C.2.4.3 通信顺序进程(CSP)

目的:CSP 是用于并行软件系统(即:并行运行的通信过程的系统)的规范的一种技术。

描述:CSP 为过程系统规范提供了一种语言,并为验证过程的实现满足它们的规范(描述成一条轨迹,即一个允许的事件序列)提供了证明。

按独立过程的一个网络建造一个系统的模型,这些过程既可顺序组合也可并行。用每个过程可能的所有行为来描述该过程,这些过程可通过通道进行通信(同步或交换数据),只有在两个过程准备就绪时才能进行通信。能建立事件相关时序的模型。

CSP 的理论基础早已直接纳入 INMOS 传送机的体系结构中,并且 OCCAM 语言允许在一个传送机网络上直接实现一个 CSP 规定的系统。

参考文献:

Communicating Sequential Processes: The First 25 Years. A. Abdallah, C. Jones, J. Sanders (Eds.). Springer, 2004, ISBN 3540258132, 9783540258131

C.2.4.4 高阶逻辑(HOL)

目的:它是一种形式化语言,这种语言打算作为硬件规范和验证的一种依据。

描述:HOL 指的是一种特殊的逻辑符号表示法和它的机器支持系统,这两者是由英国剑桥大学计算机实验室开发的,逻辑符号表示法基本上取自丘奇(Church)的简单类型理论,机器支持系统则以可计算的功能逻辑(LCF)系统为基础。

参考文献:

Higher-Order Computational Logic. J. Lloyd. In Computational Logic: Logic Programming and Beyond, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2002, ISBN 978-3-540-43959-2

C.2.4.5 时间排序规范语言(LOTOS)

目的:LOTOS 是用于描述和推理并行通信过程系统行为的一种方法。

描述:时间排序规范语言(LOTOS),是以带有来自相关代数学 CSP 和电路计算(CIRCAL)附加特征的 CCS 为基础的。它通过结合基于抽象数据型语言 ACT ONE 从而克服了 CCS 在处理数据结构和值表示式方面的弱点。然而在描述抽象数据类型时可使用 LOTOS 的过程描述成分并结合其他形式化方法。

参考文献:

Model Checking for Software Architectures. R. Mateescu. In Software Architecture, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2004, ISBN 978-3-540-22000-8

ISO 8807: 1989, Information processing systems—Open Systems Interconnection—LOTOS—A formal description technique based on the temporal ordering of observational behaviour

C.2.4.6 OBJ

目的:为了在实现之前利用用户反馈信息和系统确认信息提供一个精确的系统规范。

描述:OBJ 是一种代数规范语言。用户将要求规定为代数方程的形式。通过影响抽象数据类型(ADT)的运算,规定系统行为或结构特征,一个 ADT 就像一个 ADA 包一样,只不过运算符的工作情况可见而实现细节是“隐藏”的。

一种 OBJ 规范和其后的逐步实现适合于同其他形式化方法一样的形式证明技术。此外,因为 OBJ 规范的结构特征可用机器执行,因此从规范本身可直接实现系统确认。执行过程实质上是通过持续的方程置换(重写)直到获得规定的输出值为止来评价一个功能。这种可执行性允许设想系统的最终用户

在系统规范阶段就能够直观认识最终系统,而不需精通基础的形式规范技术。

正如其他所有的 ADT 技术的情况一样,OBJ 只适用于顺序系统或者并行系统的顺序部分。OBJ 已被用于各种规模的工业应用的规范。

参考文献:

Software Engineering with OBJ: Algebraic Specification in Action. J. Goguen, G. Malcolm. Springer,2000,ISBN 0792377575,9780792377573

C.2.4.7 时序逻辑

目的:安全性和运行要求的直接表达式,及这些属性在其后的开发步骤中得以保持的形式化证明。

描述:标准的一阶谓词逻辑不包含时间概念,通过增加情态算符(例如“此后”(henceforth)和“最终”(eventally)),时序逻辑可扩展一阶逻辑。这些算符可用来限定有关系统的一些声明。例如,安全属性可能需要包含“此后”,而要求的系统的其他状态可能需要借助某些另外的起始状态达到“最终”。使用状态(行为)序列来解释时序公式。“状态”由何组成取决于选择的描述等级。它可以指整个系统,一个系统部件或者计算机程序。

在时序逻辑中并不直接处理量化的时间间隔和约束。必须通过建立附加的时间状态把绝对定时处理成状态描述的一部分。

参考文献:

Mathematical Logic for Computer Science. M. Ben-Ari. Springer, 2001, ISBN 1852333197, 9781852333195

C.2.4.8 VDM,VDM++——维也纳(Vienna)开发方法

目的:顺序(VDM)和并行实时(VDM++)程序的系统化规范和实现。

描述:VDM 是基于数学的一种规范技术,通过允许依据规范证明其正确性来细化实现。

规范技术是以模型为基础的,该技术将系统状态建模为集合理论结构(基于结构描述的不变量(谓词)),以及通过将其先决和后置条件规定为系统状态来进行建模的状态的运算。为了保留系统不变量,可验证这些运算。

通过将系统状态具体化为目标语言的数据结构,并通过将运算细化为目标语言的程序,可完成规范的实现。具体化和细化步骤将引出证明它们的正确性的责任。设计者应决定是否要履行这些职责。

原则上讲,VDM 用在规范阶段,但也可在产生源码的设计和实现阶段中使用。它仅适用于并行系统中顺序结构程序或者顺序过程。

面向对象的和并行实时扩充的 VDM,即 VDM++,是基于 ISO 语言 VDM-SL 和面向对象的语言 Smalltalk 的一种形式规范语言。

VDM++提供了一个很宽的结构范围,因而用户可在形式上用一种面向对象的格式来规定实时系统。在 VDM++中,一个完整的形式规范由分类规范的一个汇集和任选的一个工作区组成。

VDM++的实时规定是:

——为了表示一个方法主体中的当前时刻和方法引用时刻,应提供时序表达式;

——给方法增加一个定时的后续表达式以便为正确实现规定执行时间的上(或下)限;

——已经引入时间连续变量。利用假设子句和效果子句,可以规定这些时间函数之间的关系(例如微分方程)。这个特征已证明对工作在一个时间连续环境中的系统要求的规范是很有用的。细化步骤可产生这些类型的系统的离散软件解。

参考文献：

ISO/IEC 13817-1：1996，Information technology—Programming languages，their environments and system software interfaces—Vienna Development Method—Specification Language—Part 1：Base language

Systematic Software Development using VDM.C.B.Jones.Prentice-Hall.2nd Edition，1990

Conformity Clause for VDM-SL，G.I.Parkin and B.A.Wichmann，Lecture Notes in Computer Science 670，FME'93 Industrial-Strength Formal Methods，First International Symposium of Formal Methods in Europe.Editors：J.C.P.Woodcock and P.G.Larsen.Springer Verlag，501—520

C.2.4.9 Z

目的：Z 是用于顺序系统的一种规范语言表示方法，也是一种允许开发者从一个 Z 规范开始进行可执行的算法的设计技术，该算法允许对照规范验证它们的正确性。

Z 主要用于规范阶段，但也已延伸至设计和实现。它最适合于开发面向数据的顺序系统。

描述：像 VDM 一样，规范技术是以模型为基础的，该技术将系统状态建模为集合理论结构[基于结构描述的不变量(谓词)]，以及通过将其先决和后置条件规定为系统状态来进行建模的状态的运算。为了保留系统不变量从而证实其一致性可证明这些运算，将把一个规范的形式部份分成一些模式，这些模式允许通过细化构建规范。

典型地，一个 Z 规范是形式 Z 和用自然语言的非形式化说明文本的混合物(形式化文本本身过于简要因而不易读，并且常常需要解释它的目的，而非形式化自然语言容易变得模糊和不准确)。

与 VDM 不同，Z 是一种表达式而不是一个完整的方法，然而已开发出一种辅助方法(称为 B)，它可同 Z 一起使用。B 方法依据的是逐步细化的原理。

参考文献：

Formal Specification using Z，2nd Edition. D. Lightfoot. Palgrave Macmillan，2000，ISBN 9780333763278

The B-Method.S.Schneider.Palgrave Macmillan，2001，ISBN 9780333792841

C.2.5 防御性编程

注：在 GB/T 20438.3—2017 的表 A.4 中引用了本技术/措施。

目的：为了产生一些程序，在执行它们的过程中，这些程序可检测异常控制流、数据或数据值，并且以一种预定的和可接受的方式对这些异常作出反应。

描述：为了对控制或数据异常进行检查，在程序设计过程中可以使用许多种技术。为了减少错误的数据处理的可能性，在一个系统的整个程序设计过程中，可系统地使用这些技术。

有两类有重叠的防御性技术。本质错误—安全软件被设计为可适应其本身设计上的缺陷，这些缺陷可能是设计或编码中的错误或者不正确的要求造成的，下面列出了一些防御技术：

——检查变量的范围；

——在可能的情况下，检查值的可信性；

——在子程序入口处检查输入子程序的参数的类型、大小和范围。

从程序功能和变量的物理意义这两方面来看，上述三条建议有助于保证由程序处理的数值是合理的。

只读和读写参数应分开并应检查它们的存取。这些功能应把所有参数处理成只读参数。字面常量不可写。这有助于检测意外的重写或者误用变量。

容错软件被设计为在其自身环境中或使用外部标准或期望的条件来“预计”失效，并按预先规定的方式工作。包含以下技术：

——检查具有物理意义的输入变量和中间变量的可信性；

——检查输出变量的影响，最好直接观察相关系统状态的改变；

——检查软件的配置，包括预期硬件的存在和可存取性，还要检查软件本身是否完整——这一点对于在维护过程之后保持完整性是特别重要的。

有些防御程序设计技术，比如控制流序列检查也能应对外部失效。

参考文献：

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

Dependability of Critical Computer Systems: Guidelines Produced by the European Workshop on Industrial Computer Systems, Technical Committee 7 (EWICS TC7, Systems Reliability, Safety, and Security). Elsevier Applied Science, 1989, ISBN 1851663819,9781851663811

C.2.6 设计和编码标准

注：在 GB/T 20438.3—2017 的表 A.4 中引用了本技术/措施。

C.2.6.1 一般要求

目的：为了提高可验证性，为了促进团队化的客观方法以及为了实施一种标准的设计方法。

描述：项目一开始，参加者就协商将遵守的规则。这些规则包含要遵守的设计和开发方法（例如：JSP、MASCOT、佩特里（Petri）网等）以及有关的编码标准（参看 C.2.6.2）。

制定这些规则是为了使开发、验证、评估和维护更容易。因此，他们应把适用的工具、特别是分析程序和逆向工程工具考虑进去。

参考文献：

IEC 60880:2006, Nuclear power plants—Instrumentation and control systems important to safety—Software aspects for computer-based systems performing category A functions

Verein Deutscher Ingenieure. Software-Zuverlassigkeit-Grundlagen, Konstruktive Massnahmen, Nachweisverfahren. VDI-Verlag, 1993, ISBN 3-18-401185-2

C.2.6.2 编码标准

注：在 GB/T 20438.3—2017 的表 B.1 中引用了本技术/措施。

目的：为了减少在安全相关代码中出错的可能性，并且易于验证。

描述：以下原则表明了安全相关的编码规则（对于任何编程语言）怎样协助符合 GB/T 20438.3 的规范要求，并且实现“期望的属性”（见附录 F）。应该考虑可使用的支持工具。

GB/T 20438.3—2017 要求与推荐	编码标准建议
模块化方法（表 A.2-7，表 A.4-4）	软件模块大小限制（表 B.9-1）和软件复杂度控制（表 B.9-2），例如： ● “局部”规模、复杂性度量和限制（用于模块）的规范 ● “全局”规模、复杂性度量和限制（用于总体模块组织）的规范 ● 参数数目限制/固定的子程序参数数目（表 B.9-4） 信息隐藏/封装（表 B.9-3）：例如，鼓励使用专门的语言特性。 充分定义的接口（表 B.9-6），例如： ● 明确规范的函数原型 ● 失效断言编程（表 A.2-3a）和数据验证（7.9.2.7），使用函数的前置条件和后置条件的、断言的、数据类型不变量的明确规范

表（续）

GB/T 20438.3—2017 要求与推荐	编码标准建议
代码的可理解性 ● 提升代码的可理解性(7.4.4.13) ● 易读的、可理解的和可测试的(7.4.6)	促进有意义的、明确的名称的命名规范 例如：避免可能被混淆的名称(例如 IO 和 I0)。 数值型值的符号名称。 源代码文件的程序和准则(7.4.4.13)。例如： ● 解释为什么和含义(不仅解释什么) ● 警告 ● 其他影响 在可行的情况下，源代码中应该包含以下信息(7.4.4.13)： ● 合法实体(例如：公司，作者等) ● 描述 ● 输入和输出 ● 配置管理的历史 (参见模块化方法)
可验证性和可测试性 ● 便于验证和测试(7.4.4.13) ● 便于设计和编程差错的检测(7.4.4.10) ● 形式化验证(表 A.5-9) ● 形式化证明(表 A.9-1)	● “关键”的库函数的封装，检查前置和后置条件 ● 鼓励使用能表达特定数据元素或函数的使用限制的语言(例如：const) ● 对于支持验证的工具：符合所选工具限制的规则(假如这不妨碍更重要的目标) ● 限制使用递归(表 B.1-6)和其他形式的循环依赖 (参见模块化方法)
符合指定设计的静态验证(7.9.2.12)	特定设计概念或约束实施的编码指南。例如： ● 循环行为的编码指南，保证最大周期时间(表 A.2-13a) ● 时间触发架构的编码指南(表 A.2-13b) ● 事件驱动架构的编码指南(表 A.2-13c) ● 静态确定最大迭代次数的循环(除了循环设计的无限循环) ● 静态资源分配(表 A.2-14)和避免动态对象的编码指南(表 B.1-2) ● 访问共享资源的静态同步的编码指南(表 A.2-15) ● 符合有限制的使用中断的编码指南(表 B.1-4) ● 避免动态变量的编码指南(表 B.1-3a) ● 动态变量安装的在线检查(表 B.1-3b) ● 确保使用其他编程语言的兼容性的编码指南(7.4.4.10) 促进设计可追溯性的导则
-语言子集(表 A.3-3) ● 取缔不安全的语言特性(7.4.4.13) ● 只使用已定义的语言特性(7.4.4.10) ● 结构化程序设计(表 A.4-6) ● 强类型编程语言(表 A.3-2) ● 不允许自动类型转换(表 B.1-8)	排除非结构化设计的语言特性。例如： ● 有限制的使用指针(表 B.1-5) ● 有限制的使用递归(表 B.1-6) ● 有限制的使用类似 C 中的联合体(union) ● 有限制的使用类似 Ada 或 C++中的异常 ● 不允许在高级语言程序中有非结构化的控制流(表 B.1-7) ● 在子程序和函数中只有一个入口/一个出口点(表 B.9-5) ● 不允许自动类型转换 ● 有限制使用函数原型中未明确的副作用(例如，静态变量) 在条件的评估和所有形式断言中无副作用。 编译器专有特性的有限制使用，或备有文件证明。 潜在的误导语言结构有限制的使用。 使用了以上语言特性后，仍需应用的规则

表（续）

GB/T 20438.3—2017 要求与推荐	编码标准建议
良好的编程实践(7.4.4.13)	在适用时： • 必要的时候，编码指南用来确保以正确的顺序计算浮点表达式(例如，“a－b＋c”不总是等于“a＋c－b”) • 在浮点数比较中：使用不等式(小于，小于等于，大于，大于等于)代替严格的等式 • 条件式编译和“前处理”的指南 • 系统性的检查返回条件(成功/失败) 记录可执行代码的生成，并在可行时自动生成可执行代码(makefile)。 避免函数原型中未明确的副作用。当存在这种副作用时，记录它们的指南。 当运算符的优先级不是绝对明显时可加括号。 捕获假定不可能的情形(例如，在C“switch”中的一个默认情况)。 关键模块“包装”的使用，特别是，检查前置条件和后置条件以及返回条件。 符合已知的编译器差错和编译器评估设定的限制的编码指南

C.2.6.3 无动态变量或者动态对象

注：在 GB/T 20438.3—2017 的表 A.2 和表 B.1 中引用了本技术/措施。

目的：为了排除：

——不需要的或未发现的存储器重复占位；

——在(安全相关的)运行时间内资源的分配瓶颈。

描述：本措施中提到的动态变量和动态对象是指：在运行时才分配给它们的存储器和确定的绝对地址，分配的存储器的值和它的地址与分配瞬时系统的状态有关，这意味着不能用编译器或者任何别的离线工具来检查它。

因为动态变量和对象的数目以及分配给新动态变量或对象的现存空闲内存空间与分配瞬时的系统状态有关，当分配或使用变量或对象时有可能发生故障。例如，当系统分配存储单元时，闲置内存量不足，另一变量的存储内容就可能无意中被重写。当不使用动态变量或对象时就可避免这些故障。

在动态行为不能通过某些静态分析准确预测时(例如，在程序执行前)，就需要在动态对象的使用上加以限制，因此可以保证可预测的程序执行。

C.2.6.4 在建立动态变量或动态对象过程中的在线检查

注 1：在 GB/T 20438.3—2017 的表 B.1 中引用了本技术/措施。

目的：为了检查在发生分配前要分配给动态变量和对象的存储器是空闲的，从而保证在运行时动态变量和对象的内存分配不会影响现存的变量、数据或代码。

描述：本措施中提到的动态变量和动态对象是指：在运行时才分配给它们的存储器和确定的绝对地址(从这个意义上讲变量也是对象实例的属性)。

在给动态变量或对象分配内存之前，借助硬件或软件来检查存储器以便保证它是空闲的(例如用以避免堆栈溢出)。如不允许分配(例如当存储器不足以应付所确定的地址时)必须采取适当的行动。在已经使用一个动态变量或者目标之后(例如退出一个子例程之后)，曾经分配给它的整个存储器必须要释放。

注 2：一种替代办法是统计证明对一切情况来说，存储器都是够用的。

C.2.6.5 有限制的使用中断

注：在 GB/T 20438.3—2017 的表 B.1 中引用了本技术/措施。

目的：为了保持软件可验证和可测试。

描述：应限制使用中断。当它们能简化系统时才使用中断。在被执行的功能的关键部份（例如时间上的转折点、数据改变的转折点）禁用中断处理软件。当使用中断时，不可中断的部份应有一个规定的最大计算时间，使之能计算禁止一次中断的最大时间。中断使用和屏蔽应全部编成文档。

C.2.6.6 有限制的使用指针

注：在 GB/T 20438.3—2017 的表 B.1 中引用了本技术/措施。

目的：为了避免在没有首先检查指针的范围和类型的情况下由存取数据引起的问题。为了支持模块化测试和软件验证。为了限制失效的后果。

描述：在应用软件中，只有在存取之前检验指针数据类型和值的范围（为了保证参考指针是在正确的地址空间内）时，方可在源码级使用指针计算。不能采用任务之间的直接引用来实现应用软件的任务间通信，应通过操作系统来进行数据交换。

C.2.6.7 有限制的使用递归

注：在 GB/T 20438.3—2017 的表 B.1 中引用了本技术/措施。

目的：为了避免不可验证的和不可测试的子例程调用。

描述：当使用递归时，必须要有一个明显的判据，此判据可预测递归深度。

C.2.7 结构化编程

注：GB/T 20438.3—2017 的表 A.4 中引用了本技术/措施。

目的：结构化编程是一种设计和实现程序的方法，这种方法使得程序不需要执行就能进行分析，并使其不可测行为的可能性尽可能降到最低。

描述：下面的原则可用来降低结构的复杂性：

——把程序分成适当小的软件模块，从而保证尽可能地使它们去耦并且所有的相互作用都是显式的。

——使用结构化构造来组合软件模块控制流，包括序列、迭代和选择；

——保持到一个软件模块的可能通路要少，输入和输出参数之间的关系要尽量简单；

——避免复杂的分支，特别要避免高级语言中的无条件转移（goto）；

——只要可能，使回路约束和分支同输入参数联系起来；

——作为决定分支和回路判定的计算要简单。

除了效率优先级最高的情况（例如某些安全关键系统）之外，应使用有助于上述方法的程序设计语言的特征而不用（声称）是更有效的其他特征。

参考文献：

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985, 9780521780988

A Discipline of Programming. E. W. Dijkstra. Englewood Cliffs NJ, Prentice-Hall, 1976

C.2.8 信息隐藏/封装

注：在 GB/T 20438.3—2017 的表 B.9 中引用了本技术/措施。

目的:为了防止意外的访问数据或子程序并因此为了支持一个好的程序结构。

描述:所有软件模块全局可访问的的数据可能被这些模块的任何一个意外地和不正确地修改。这些数据结构的任何改变可能都需要仔细的代码检查和扩充性修改。

信息隐藏是减少这些困难的一种常用方法。关键数据结构被“隐藏”,只有通过一个定义的存取子程序集才能操纵它。这种办法允许修改内部结构或者增加另外的子程序而不会影响其余软件的功能行为。例如,一个名字目录可以有存取子程序“插入”、“删除”和“查找”。可以重写存取子程序和内部数据结构(例如使用不同的查找方法或者把名字存放在一张硬盘上)而不会影响使用这些子程序的其余软件的逻辑行为。

在这种联系中,应使用抽象数据的概念,如果不能提供直接支持,则有必要检查抽象未被无意破坏。

参考文献:

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985,9780521780988

On the Design and Development of Program Families.D.L.Parnas.IEEE Trans SE-2,March 1976

C.2.9 模块化方法

注:在 GB/T 20438.3—2017 的表 A.4 和表 B.9 中引用了本技术/措施。

目的:为了限制系统的复杂程度,把一个软件系统分解成一些小的易理解的部分。

描述:模块化方法包含软件项目的设计、编码和维护阶段的若干规则。根据设计过程中使用的设计方法不同规则有所不同。大多数方法都包含以下规则:

——一个软件模块(或等价的,子程序)应有能完成定义良好的单项任务或功能;

——应限制和严格定义软件模块之间的衔接,软件模块的内聚性应是很强的;

——如果有多级软件模块,应建立子程序集合;

——应把子程序的大小限定为某个规定值,典型地为 2～4 个屏幕尺寸;

——子程序只有一个入口和一个出口;

——软件模块应通过接口与其他软件模块通信——在使用全局或者共用变量的情况下,它们应构成良好,存取可控并且它们的使用在每个实例中应是合理的;

——应把所有的软件模块接口全部编入文档;

——任何软件模块的接口应只包含它的功能所必需的那些参数。然而,此建议被这种可能性复杂化,即编程语言可能允许缺省参数或使用了面向对象的方法。

参考文献:

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122,9780471469124

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985,9780521780988

C.2.10 使用可信的/经验证的软件组件

注:在 GB/T 20438.3—2017 的表 A.2、表 C.2、表 A.4 和表 C.4 中引用了本技术/措施。

目的:为了避免对每一新应用都需对软件设计和组件重新确认或者重新设计。为了利用未经正式或严格验证、但已长期使用的设计的优点。为了利用已在其他应用中验证且存在大量验证证据的已有软件组件的优点。

描述:这项措施证明软件组件足以摆脱系统设计故障和/或运行失效。

从最基本的部分建立一个复杂系统通常是不切实际的。利用主要组件通常是必要的(“组件”,见GB/T 20438.4—2017的3.4.5和3.2.8)这些组件之前被开发用来提供一些有用的功能,并且这些功能能被重复使用于实现新系统的一部分。

良好设计的及结构化的PE系统由许多软件组件构成,这些软件组件明显不同并且以一种明确规定的方式进行交互。建立一个能被重复用在多种用途的通用软件组件库来容纳大部分资源,这些资源对确认多个应用可共享的设计是必需的。

然而,对于安全相关的应用,足够的置信度是必要的,即包含这些已有组件的新系统具有所需的安全完整性,且安全不被已有组件的某些不正确行为影响。

为了获得一个已有组件能被准确知道的行为的置信度,有两种观点:

——分析组件的全面运行的历史,证明这个组件已经“经使用证明”;

——评估已经收集在组件行为中的验证证据的主体,以确定组件是否符合GB/T 20438的要求。

C.2.10.1 经使用证明的

只有在极少数情况下,“经使用证明的”(见GB/T 20438.4—2017的3.8.18)才是一个充分的论据,证明一个受信任的软件组件达到必要的安全完整性。对于具有许多可能功能的复杂的组件(例如,操作系统),确定组件的哪些功能实际上充分经使用证明是很重要的。例如,自检例程用于检测故障,如果在运行期间没有发生失效,我们就不能把用于故障检测的自检例程作为经使用证明的。

如果软件模块符合以下标准,那么它被视为经使用证明的:

——未改变的规范;

——在不同应用中的系统;

——服务历史至少一年;

——按照安全完整性等级或适当的要求次数的运行时间;证明非安全相关的失效率低于

- 在置信度95%时,每次需求(或每年)为10^{-2},发生300次要求(或运行300年);
- 在置信度99.9%时,每次需求(或每年)为10^{-5},发生690 000次要求(或运行690 000年)。

注1:见附录D对上面数值估算的数学支持。见B.5.4相似的措施和统计学的方法。

——所有的运行经验必须与一个已知软件组件的功能要求概要相关,以确保增强的运行经验真正增强与要求概要相关的软件组件行为的知识;

——没有安全相关的失效。

注2:一个失效在一种情形下可能不是安全关键的,而在另一种情形下是安全关键的,反之亦然。

为了能验证软件组件符合这些标准,必须记录以下几项:

——准确标识每一系统和它的组件,包括版本号(对软件和硬件而言)

——用户的标识和应用的时间;

——运行时间;

——用户应用系统的选择规程和应用实例;

——检测和记录失效及消除故障的规程。

C.2.10.2 评估验证证据

已有软件组件(见GB/T 20438.4—2017的3.2.8)是已经存在但并非专门为目前的项目或SRS开发的软件。已有软件可以是商业上可使用的产品,或者它已被一些组织开发用于一个先前的产品或系统。已有软件可能依据或未依据GB/T 20438要求开发。

为了评估包含已有软件的系统的安全完整性,应有验证证据来确定已有组件的行为。它可能来自(1)组件供应商自己的组件开发过程的文档和记录,或(2)它可能通过额外的资质认证创建或补充,此活动由新的安全相关系统的开发者或第三方承担。这就是定义了潜在可重复使用的软件组件的功能和限

制的“符合项安全手册”。

在任何情况下,符合项安全手册必须存在(或必须被创建),足以用于对一个具体的安全功能进行完整性评估,该安全功能全部或部分地取决于重复使用的组件。否则,必然得出保守的结论:组件不适于安全相关的重复使用。(这不是说,组件在任何情况下都不能证明适用,而仅仅是在这个特定的情形下没有找到足够的证据。)

GB/T 20438 对符合项安全手册的内容有具体的要求,见 GB/T 20438.2—2017 附录 D 和 GB/T 20438.3—2017 附录 D,以及 GB/T 20438.3—2017 的 7.4.2.12 和 7.4.2.13。

作为一个内容非常简短的说明,符合项安全手册将包含以下内容:

——组件的设计是已知的和有记录的;

——组件经过系统化的方法验证和确认,这种系统化方法有所有组件设计及代码的审查和有记录的测试;

——组件未使用的和不必要的功能将不会妨碍新系统满足其安全要求;

——在新系统中的组件的所有可信的失效机制已经确定,并已实施适当的缓解。

新系统的功能安全评估必须确定重复使用的组件严格地应用在能力的限制范围内,这种能力已经由符合项安全手册中给出的证据和假设所证明。

参考文献:

Component-Based Software Development: Case Studies. Kung-Kiu Lau. World Scientific, 2004, ISBN 9812388281,9789812388285

Software Reuse and Reverse Engineering in Practice. P. A. V. Hall (ed.), Chapman & Hall, 1992, ISBN 0-412-39980-6

Software criticality analysis of COTS/SOUP. P. Bishop, T. Clement, S. Guerra. In Reliability Engineering & System Safety, Volume 81, Issue 3, September 2003, Elsevier Ltd., 2003

C.2.11 可追溯性

目的:为了保持生命周期各阶段的连贯性。

描述:为了确保由生命周期活动获得的软件符合安全相关系统正确运行的要求,确保生命周期阶段间的连贯性是必要的。这里的一个关键概念是,在活动之间的可追溯性。这本质上是一个影响分析,用于检查(1)在早期阶段做出的决策在稍后阶段得到充分实施(向前追溯),(2)在稍后阶段的决定确实由早期决定要求和决定的。

向前追溯关注于检查在稍后的生命周期阶段需求被充分实现。向前追溯在安全生命周期的一些点是有价值的。

——从系统安全要求到软件安全要求;

——从软件安全要求规范到软件架构;

——从软件安全要求规范到软件设计;

——从软件设计规范到模块和集成测试规范;

——从硬件/软件集成系统和软件设计要求到硬件/软件集成测试规范;

——从软件安全要求规范到软件安全确认计划;

——从软件安全要求规范到软件变更计划(包括重验证和重确认);

——从软件设计规范到软件验证计划(包括数据验证);

——从 GB/T 20438.3—2017 的第 8 章的要求到软件功能安全评估计划。

向后追溯关注于检查每一实现的决定都被某些要求清晰论证(广义的实现,不限于代码)。如果无法论证,则在实现中包含一些不必要的、增加复杂性且不一定解决任何安全相关系统的真正要求。向后追溯在安全生命周期的一些点是有价值的。

——从安全要求到获知的安全需求；
——从软件架构到软件安全要求规范；
——从软件详细设计到软件架构；
——从软件编码到软件详细设计；
——从软件安全确认计划到软件安全要求规范；
——从软件变更计划到软件安全要求规范；
——从软件验证(包括数据验证)计划到软件设计规范。

参考文献：

Requirements Engineering. E. Hull, K. Jackson, J. Dick. Springer, 2005, ISBN 1852338792, 9781852338794

C.2.12 无状态软件设计(或有限状态设计)

注：GB/T 20438.3—2017 的表 A.2 中引用该技术/措施。

目的：限制软件行为的复杂性。

描述：考虑一个软件程序处理一系列的事件：它接受一系列的输入并响应每一个输入产生一个输出。程序也可能在一个“状态”记忆它的一些或全部的历史，并在计算怎样响应将来的输入时考虑这种状态。

响应一个特定输入的程序的输出是完全取决于输入，那么程序是无记忆的或无状态的。没有事件被任何早期的事件影响，在这个意义上，每一输入/输出的处理是完整的，并且一个特定的输入总是产生相同的输出。

相比之下，程序在计算它的输出中既考虑到它的特定输入也考虑到它的记忆状态，那么程序能够实现更复杂的行为，因为它能在不同的时间响应相同的输入并传送不同的输出。对特定输入的响应可能取决于输入接受的环境(即前面的输入和输出)。和一些应用(通常是通信)相关的进一步考虑是，程序的行为可能对存储状态改变特别敏感，无论是不经意还是恶意的引入。

无状态设计是一种通用方法，旨在通过避免或减少软件设计中状态信息的使用以最小化软件行为的潜在复杂性。

参考文献：

Introduction to Automata Theory, Languages, and Computation(3rd Edition). J. Hopcroft, R. Motwani, J. Ullman, Addison-Wesley Longman Publishing Co, 2006, ISBN: 0321462254

Stateless connections. T. Aura, P Nikander. In Proc International Conference on Information and Communications Security (ICICS' 97), ed Yongfei Han. Springer, 1997, ISBN 354063696X, 9783540636960

C.2.13 离线数值分析

注：GB/T 20438.3—2017 的表 A.9 引用该技术/措施。

目的：为了确保数值计算的准确性。

描述：在一个数值函数由于使用有限的理想函数或数字表示的计算中，数值不准确度可能提高。当一个函数被一个无穷级数(如傅里叶级数)的有限数目项逼近时，将引进截断误差。物理计算机里实数的有限精度表示引入了舍入误差。对于除了最简单计算外的任何以浮点数表示的计算，必须检查计算的有效性以确保真正实现了应用要求的精度。

参考文献：

Guide to Scientific Computing. P.R. Turner. CRC Press, 2001, ISBN 0849312426, 9780849312427

C.2.14 报文序列图

注：GB/T 20438.3—2017 的表 B.7 和表 C.17 引用了该技术/措施。

目的：为了在软件开发(包括需求和软件架构设计)的早期阶段协助获取系统要求。在 UML 中，此表示法被命名为“报文序列图”。

描述：报文序列图是一种依据系统参与者(参与者可能是一个人，一个计算机系统，或软件组件或对象，取决于设计阶段)之间通信来描述系统行为的图解机制。对每一个参与者，在图上画一条垂直的“生命线”，生命线之间的箭头用来表示报文。接受报文的动作可有选择的以方框的形式在图表中显示。建立收集的情形(描述想要的和不想要的行为)作为要求的系统行为规范。这些情形有多种用途。它们可以给终端用户动画演示系统行为。它们也可以转化为一个系统可执行文件执行。它们可以形成测试数据的基础。

UML 包含对报文序列图的原始概念的扩展，它提供了一个更紧凑的形式，报文序列图是选择和迭代结构形式，这种结构允许情形分支及循环。子图也可以被定义，它可以被许多更高级别的序列图引用。定时器和外部事件也可被表示。

参考文献：

“Message Sequence charts”，D.Harel，P.Thiagarajan.In UML for Real：Design of Embedded Real-Time Systems.ed.L.Lavagno.Springer，2003，ISBN 1402075014，9781402075018

ISO/IEC 19501：2005，Information technology—Open Distributed Processing—Unified Modeling Language(UML) Version 1.4.2

C.3 架构设计

C.3.1 故障检测和诊断

注：在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的：为了检测一个系统中的故障(这些故障可能导致一次失效)，从而为减少失效影响的对应措施提供依据。

描述：故障检测是检查一个系统的错误状态的活动(这些错误状态是被检查的子系统内的一个故障引起的)。故障检测的主要目的是阻止错误结果的影响。当一个系统同并行组件共同起作用时，检测到其自身的错误结果就放弃控制，称为自检。

故障检测基于冗余(主要检测硬件故障——见 GB/TB20438.2—2017 的附录 A)和多样性(软件故障)原理。需要用某种表决来确定结果的正确性。专门方法包括：失效断言编程，N 版本编程和多样化监视器技术，对硬件而言包括：引入附加传感器，控制回路，差错检验码等。

可以通过检查各级的值域或时间域、特别是物理的(温度、电压等)、逻辑的(差错检测码)、功能的(断言)或者外部的(可信性检验)来实现故障检测。为了能进行故障跟踪，可把这些检验的结果储存起来并把它们同数据联系起来。

复杂系统是由子系统构成的，故障检测、诊断和故障补偿的效率与各子系统中相互作用的复杂程度有关，这种相互作用的复杂性可影响故障的传播。

应在最小的子系统级使用故障诊断，因为较小的子系统可以更详细地诊断故障(差错状态检测)。

全企业综合信息系统通常能把安全相关系统的状态包括诊断测试信息传递给另一监控系统。当检测到一个异常时，就会突出显示它并在它发展成一种危险情况之前触发纠正功能。最后，如果确实发生事故，这种异常的记录有助于随后的调查。

参考文献：

Dependability of Critical Computer Systems 1.F.J.Redmill，Elsevier Applied Science，1988，ISBN

1-85166-203-0

C.3.2 差错检测和纠正码

注：在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的：为了检测和纠正敏感信息中的差错。

描述：对一条 n 位信息而言，生成一个 K 位编码块，此块使之能检测和纠正 r 个差错。两类编码的例子是汉明(Hamming)码和多项式代码。

应注意，在安全相关系统中，通常需要废弃故障数据而不是纠正它，因为只有一部分预定的差错才能得以正确纠正。

参考文献：

Fundamentals of Error-correcting Codes, W. Huffman, V. Pless. Cambridge University Press, 2003, ISBN 0521782805, 9780521782807

C.3.3 失效断言编程

注：在 GB/T 20438.2—2017 的表 A.17 中和 GB/T 20438.3—2017 的表 A.2 和表 C.2 中引用了本技术/措施。

目的：为了在执行一个程序的过程中检测软件设计的残余故障，从而防止系统安全关键失效并继续高可靠地运行。

描述：失效断言程序设计方法遵循的理念是检查一个先决条件(在执行一个语句序列之前，对初始条件进行有效性检查)和一个后续条件(在执行一个语句序列之后检查结果)。如不满足先决条件或者后续条件，处理将报告出了差错。

例如：

断言〈先决条件〉；

动作 1；

：

：

动作 X；

：

：

断言〈后续条件〉；

参考文献：

Exploiting Traces in Program Analysis. A. Groce, R. Joshi. Lecture Notes in Computer Science vol 3920, Springer Berlin/Heidelberg, 2006, ISBN 978-3-540-33056-1

Software Development—A Rigorous Approach. C.B. Jones, Prentice-Hall, 1980

C.3.4 多样化监视器

注：在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的：为了防止软件规范和实现中的残余故障，这些故障对安全有不利的影响。

描述：有两种不同的监控方法：(1)监视器和被监视的功能在相同的计算机，它们之间的独立性有一定保证；(2)监视器和被监视功能在不同的计算机。

多样化监视器是按照一个不同规范实现的外部监视器，在一台独立的计算机上实现。这种多样化监视器只涉及确保计算机执行安全动作，但不一定纠正错误动作。多样化监视器连续监视主计算机。多样化监视器防止系统进入一个不安全状态。此外，如果它检测到主计算机正进入一个潜在的危险状态，系统必须通过多样化监视器或主计算机恢复到一个安全状态。

多样化监视器的硬件和软件应该按适当的 SIL(安全完整性等级)分类和考核。

参考文献:

Requirements based Monitors for Real-Time Systems, D.Peters, D.Parnas. IEEE Transactions on Software Engineering, vol.28, no.2, 2002

C.3.5 软件多样化(多种程序设计)

注: 在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的:为了防止系统的安全关键失效并继续高可靠地运行,在执行一个程序过程中检测和屏蔽残余的软件设计和实现的故障。

描述:在多样化程序设计中,以不同的方式 N 次设计和实现一个给定的程序规范。给 N 个版本都提供同样的输入值,并比较 N 个版本产生的结果,如认为结果有效,将把结果传送到计算机输出。

一个本质要求是 N 个版本在某种程度上相互独立,从而他们不会因为相同的原因同时失效。实际上,要实现和证明版本间的独立性是非常困难的,这个独立性是 N 个版本方法的基础。

N 个版本可在多个独立的计算机上并行运行,或者所有版本都能在同一台计算机上运行并且结果须经一次内部表决。对 N 个版本可使用不同的表决策略,这要根据下列应用要求而定。

——如果系统有一个安全状态,则要求完全一致(所有的 N 一致)是可行的,否则将使用一个使系统达到安全状态的输出值。对简单的跳闸系统来说,表决偏向安全。在这种情况下,如果任一版本要求一次跳闸,安全动作就将是跳闸,典型地,这种方法只使用两种版本($N=2$)。

——对没有安全状态的系统而言,可使用多数表决策略。对于不存在集体一致的情况,为了使选择正确值的机会最大化(例如取中间值,在恢复一致性之前,暂时冻结输出等)可使用概率方法。

本技术不能消除残余的软件设计故障,也不能避免规范转化中的差错,但它提供了在这些故障或差错影响安全性之前检测和屏蔽的一种方法。

参考文献:

Modelling software design diversity—a review, B.Littlewood, P.Popov, L.Strigini. ACM Computing Surveys, vol 33, no 2, 2001

The N-Version Approach to Fault-Tolerant Software, A. Avizienis, IEEE Transactions on Software Engineering, vol.SE-11, no.12 pp.1491-1501, 1985

An experimental evaluation of the assumption of independence in multi-version programming, J. C.Knight, N.G.Leveson. IEEE Transactions on Software Engineering, vol.SE-12, no 1, 1986

In Search of Effective Diversity: a Six Language Study of Fault-Tolerant Flight Control Software. A.Avizienis, M.R.Lyu and W.Schutz. 18th Symposium on Fault-Tolerant Computing, Tokyo, Japan, 27-30 June 1988, IEEE Computer Society Press, 1988, ISBN 0-8186-0867-6

C.3.6 反向恢复

注: 在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的:为了在存在一个或多个故障时保障正确的功能运行。

描述:当检测到一个故障时,系统就复位到一个较早的内部状态,该状态的一致性已在过去被证明过。这种方法意味着内部状态通常保存在所谓明确定义的检验点上。可以全局性地(对整个数据库)或者增量地(只在检验点之间的改变)进行。然后系统必须同时通过使用日志(动作的审核跟踪)和补偿(消除这些改变的所有影响)或者外部(手动)相互作用等方式补偿这些改变。

参考文献:

Looking into Compensable Transactions. Jing Li, Huibiao Zhu, Geguang Pu, Jifeng He. In Software Engineering Workshop, 2007. SEW 2007. IEEE, 2007, ISBN 978-0-7695-2862-5

Software Fault Tolerance(Trends in Software,No.3),M.R.Lyu(ed.),John Wiley & Sons,April 1995,ISBN 0471950688

C.3.7 故障恢复重试机制

注：在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的：利用重试机制从一个已发现的故障状况尝试恢复功能。

描述：在发现一个故障或差错状况的事件中，通过重新执行同样的代码尝试恢复。恢复重试可以是一次软件超时或者一次任务监视动作之后完整的重新引导和重新起动过程，或者是一次小的重新调度和重新起动任务。重试技术常用于一个通信故障或者差错恢复中，并且利用一个通信协议差错(检验和等)或者一个通信确认响应时间超时来标记重试状况。

参考文献：

Reliable Computer Systems: Design and Evaluation, D. P. Siewiorek, R. S. Schwartz. A. K. Peters Ltd.,1998,ISBN 156881092X,9781568810928

C.3.8 适度降级

注：在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的：在失效时放弃比较不关键的一些系统功能以保持更关键的功能可用。

描述：本技术给出将被系统执行的各种功能的优先级，设计保证了当执行所有系统功能的资源不足时，将优先执行高优先级的功能。例如，差错和事件记录功能的优先级比系统控制功能的优先级低，在此情况下，当与差错记录有关的硬件发生故障时，系统控制将继续进行。另外，当系统控制硬件发生故障，而差错记录硬件并未出故障时，差错记录硬件将接管控制功能。

该技术主要适用于硬件但也适用于包含软件的整个系统。从最上层设计阶段开始就必须考虑本技术。

参考文献：

Towards the Integration of Fault, Resource, and Power Management, T. Siridakis. In Computer Safety, Reliability, and Security: 23rd International Conference, SAFECOMP 2004. Eds. Maritta Heisel et.al. Springer, 2004, ISBN 3540231765, 9783540231769

Achieving Critical System Survivability Through Software Architectures, J.C. Knight, E.A. Strunk. Springer Berlin/Heidelberg, 2004, 978-ISBN 3-540-23168-4

The Evolution of Fault-Tolerant Computing. Vol. 1 of Dependable Computing and Fault-Tolerant Systems, Edited by A. Avizienis, H. Kopetz and J.C. Laprie, Springer Verlag, 1987, ISBN 3-211-81941-X

Fault Tolerance, Principle and Practices. T. Anderson and P. A. Lee, Vol. 3 of Dependable Computing and Fault-Tolerant Systems, Springer Verlag, 1987, ISBN 3-211-82077-9

C.3.9 人工智能故障纠正

注 1：在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的：借助引入方法和过程模型的组合以及某种在线安全性和可靠性分析，以便能用一种很灵活的方式应对可能的危险。

描述：基于人工智能(AI)的系统将以一种很有效的方式，在一个系统的不同通道中支持故障预测(计算发展趋势)，故障纠正、维护和监控动作，因为规则可以从规范直接导出并可对照这些规范来检验规则。本方法，特别是当使用功能或者描述形式的模型和方法的一种组合时，可有效地避免某些共同原因故障，这些共因故障是即使在考虑了某些设计和实现规则的前提下也会隐性引入的。

为了满足要求的安全完整性，应这样选择这些方法：即故障可得到纠正，失效的影响将减到最小。

注 2：有关纠错数据的警告可参看 C.3.2，关于本技术的负面意见可参看 GB/T 20438.3—2017 的表 A.2 中第 5 项。

参考文献：

Fault Diagnosis:Models,Artificial Intelligence,Applications.J.Korbicz,J.Koscielny,Z.Kowalczuk,W.Cholewa.Springer,2004,ISBN 3540407677,9783540407676

C.3.10 动态再配置

注：在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的：在有一个内部故障时保持系统功能。

描述：系统的逻辑架构必须要能把它映射成系统可用资源的一个子集，结构要求能检测一个物理资源中的一次失效，然后重新把逻辑结构映射回剩余的起作用的有限资源。虽然本概念传统上更多地局限于用来恢复有故障的硬件单元，但如果有足够的“运行时冗余”允许一个软件重试或者如有足够的冗余数据使得单独的和隔离开的失效不重要的话，它也适用于有故障的软件单元。

在系统设计的第一阶段就必须考虑本技术。

参考文献：

Dynamic Reconfiguration of Software Architectures Through Aspects.C.Costa et al.Lecture Notes in Computer Science,Volume 4758/2007,Springer Berlin/Heidelberg,2007,ISBN 978-3-540-75131-1

C.3.11 实时安全和性能：时间触发架构

注 1：在 GB/T 20438.3—2017 的表 A.2 中引用了本技术/措施。

目的：带可预测行为的关键实时系统的可组合性与容错的透明实现。

描述：在一个时间触发架构(TTA)系统中，所有系统活动被发起并基于一个全局同步的时基进程。每个应用在时间触发的总线被分配一个固定时间段，它包含了每个应用工作之间的交换信息，因此其只能根据确定的时间表来交换。在事件驱动的系统中，系统活动在不可预测的时间点被任意事件触发。TTA 的主要优点有(参见参考 Scheidler,Heiner 等。)：

——可组合性，这大大减少了进行测试和证明系统所需的努力；

——容错性的透明实现，使该架构被高度推荐用于安全关键应用；

——提供一个全局同步的时基，有利于分布式实时系统的设计。

节点之间的通信通过一个依据静态时间表的时间触发协议 TTP/C(见参考 Kopetz,Hexel)实现，决定何时发送报文和接收到的报文是否与特定的电子模块相关。对总线的访问通过一个来自全局时间概念的循环时分多址(TDMA)模式控制。

TTP/C 协议保证(见参考 Rushby) TTA 的网络节点(参见参考 Kopetz,Bauer)的四个基本服务(核心服务)：

——确定性和及时报文传输：信息的传送在一个提前已知的时限内，从发送组件的输出端口至接收组件的输入端口。通过时域防火墙接口的时间触发的通信服务提供了容错传输服务，它通过设计和最小化组件之间的耦合消除了控制误差传播。在实时应用中，报文以最小延迟和抖动的及时传输对实现控制稳定至关重要。

——容错时钟同步：通信控制器生成一个容错全局同步的时基(精度在几个时钟抖动内)提供给主机子系统。

——失败节点的一致性诊断(会员服务)：在不到一个 TDMA 循环延迟的时间内，通信控制器告知每个 SRU(“最小的可替换的单元”)一个组中的其他每一个 SRU 的状态。

——强大的故障隔离：一个恶意的故障主机子系统(包括它的软件)可以产生错误数据输出，但不可以以任何其他方式干涉一个 TTP/C 组的其余部分的正确运行。时域中的失败沉默通过通信控制器的时间触发行为保证。

注 2：其他时间触发协议包括 FlexRay 和 TT-Ethernet(时间触发 Ethernet)。

参考文献：

Time-Triggered Architecture (TTA). C. Scheidler, G. Heiner, R. Sasse, E. Fuchs, H. Kopetz, C. Temple.In Advances in Information Technologies: The Business Challenge, ed.JY.Roger.IOS Press, 1998,ISBN 9051993854,9789051993851

A Synchronisation Strategy for a TTP/C Controller.H.Kopetz,R.Hexel,A.Krueger,D.Millinger, A.Schedl.SAE paper 960120, Application of Multiplexing Technology SP 1137, Detroit, SAE Press, Warrendale,1996

The Time-Triggered Architecture. H. Kopetz, G. Bauer. Proceedings of the IEEE Special Issue on Modeling and Design of Embedded Software, October 2002

An Overview of Formal Verification for the Time-Triggered Architecture. J. Rushby: Invited paper, Oldenburg, Germany, September 9-12, 2002. Proceedings FTRTFT 2002, Springer LNCS 2469, 2002, ISBN 978-3-540-44165-6

C.3.12 统一建模语言(UML)

注：在 GB/T 20438.3—2017 的表 B.7 中引用了本技术/措施。

目的：提供一组全面的符号用于建模复杂系统的所需行为。

描述：UML，顾名思义，是一个需求和设计符号的集合，为软件开发提供全面支持。UML 的一部分是基于其他的方法中首先引入的符号(比如系统序列图和状态转换图)，其他部分是独一无二的 UML 符号。虽然一些可以使用的符号没有必要进行面向对象编程，然而 UML 是十分偏向于面向对象的概念。UML 由大量的商用 CASE 工具支持，其中很多都是能够自动从 UML 模型生成代码。

最普遍适用于安全相关系统的规范和设计的 UML 符号法如下：

——类图；

——用例；

——活动图；

——状态转换图(状态图)；

——系统序列图。

其他 UML 符号法与软件架构设计(软件结构)表达相关，这里没有具体列出。

B.2.3.2 中描述了状态转换图，C.2.14 中描述了系统的序列图。下面三个部分描述了其他的符号法。

C.3.12.1 类图

类图定义软件必须处理的对象的类。他们是基于早期的实体关系属性图，但适合于面向对象设计。每个类(其中将会有一个或多个实例在运行时作为已知对象)被表示为一个矩形，并且各种不同的类之间的关系用线或箭头表示。每一个类提供的操作或方法，每个类的数据属性，都可以添加到图表中。可表示的关系由它们的基数引用关系(A 类的一个实例可以引用一个或许多 B 类的实例)和带有可能的额外方法和属性的专业化的关系(类 X 是一个类 Y 的细化)组成。可以描述多重继承。

C.3.12.2 用例

用例提供了响应特殊情形的系统的期望行为的文本描述，其通常来自外部参与者的观点，包括系统的人类用户和外部系统。给定用例的交替子场景可以是用于表示可选的行为，尤其是在差错响应用例中。用例的集合被编制用来提供一个足够完整的系统需求规范。使用条款可以是开发更严格模型的起点，例如系统的序列图和活动图。

用例图提供了包含在用例中系统和参与者的形象化的表示，但是并不严格，只有用例的文本部分对

规范是重要的。

C.3.12.3 活动图

活动图显示了预期的由软件组件执行的动作序列(通常是在面向对象设计中的一个对象),包括顺序和迭代行为(某些方面看起来非常像一个流程图)。但是活动图允许并行描述对一系列组件的操作,图中箭头表示组件间的交互。一个活动在开始之前必须等待来自其他活动的一个或多个输入的同步点通过一个类似于佩特里(Petri)网络节点的符号表示。

参考文献:

ISO/IEC 19501:2005, Information technology—Open Distributed Processing—Unified Modeling Language(UML) Version 1.4.2

C.4 开发工具和编程语言

C.4.1 强类型编程语言

注: 在 GB/T 20438.3—2017 的表 A.3 中引用本技术/措施。

目的:通过使用一种语言来降低故障概率,这种语言允许使用编译器进行高级检查。

描述:当编译一个强类型程序设计语言时,对变量类型的使用进行许多检查,例如,在过程调用和外部数据存取中,对任何不符合预定规则的应用,编译都将会失败并产生一条出错报文。

通常这样的一些语言允许根据基本的语言数据类型(比如整型、实型)定义用户定义的数据类型。然后按同基本类型完全相同的办法使用这些类型。为了保证使用的类型正确应加强严格检查,即使程序是由分离的编译单元建造的,整个程序都要实施这些检查,甚至当引用来自独立编译的软件模块时,这些检查也能保证函数参数的数目和类型相匹配。

强类型语言一般支持良好的软件工程实践的其他特征,比如容易分析的控制结构(例如 if..then..else(如……则……否则……),do..while(当……时,做……)等),这些结构可产生良好的结构程序。

强类型语言的典型例子是 Pascal、Ada 和 Modula2。

参考文献:

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985,9780521780988

C.4.2 语言子集

注: 在 GB/T 20438.3—2017 的表 A.3 中引用了本技术/措施。

目的:为了减少引入程序设计故障的概率和增大检测任何残余故障的概率。

描述:使用例如静态分析法检查语言以便区分出易出错或难于分析的结构。然后定义一种排除这些结构的语言子集。

参考文献:

Practical Experiences of Safety-and Security-Critical Technologies, P. Amey, A. J. Hilton. Ada User Journal, June, 2004

Safer C: Developing Software for High-integrity and Safety-critical Systems. L. Hatton, McGraw-Hill, 1994, ISBN 0077076400,9780077076405

Requirements for programming languages in safety and security software standard. B. A. Wichmann. Computer Standards and Interfaces. Vol.14, pp 433-441, 1992

C.4.3 经认证的工具和经认证的翻译器

注：GB/T 20438.3—2017 的表 A.3 中引用了本技术/措施。

目的：在开发软件的各个阶段，工具对帮助开发者是有必要的。只要可能，就应认证这些工具使之能具有有关输出正确性的某种置信度等级。

描述：一般是由一个独立团体（通常是国家级），对照单独设立的判据（典型地是国家或者国际标准）来认证一个工具，理想地，应认证使用于所有开发阶段（规范、设计、编码、测试和确认）以及使用于配置管理的工具。

目前，通常只有编译（翻译）器正式经过认证，它们由国家认证机构实施。并且它们根据国际标准（比如 Ada、Pascal 的那些标准）对编译器（翻译器）进行测试。

注意，经认证的工具和经认证的翻译器通常只根据它们各自的语言或过程标准进行过认证，一般并未对安全性进行过任何认证。

参考文献：

The certification of software tools with respect to software standards, P.Bunyakiati et al.In IEEE International Conference on Information Reuse and Integration, IRI 2007, IEEE, 2007, ISBN 1-4244-1500-4

Certified Testing of C Compilers for Embedded Systems. O. Morgan. In: 3rd Institution of Engineering and Technology Conference on Automotive Electronics. IEEE, 2007, ISBN 978-0-86341-815-0

The Ada Conformity Assessment Test Suite(ACATS), version 2.5, Ada Conformity Assessment Authority, http://www.ic.org/compilerstesting.html, Apr.2002

C.4.4 工具和翻译器：通过使用提高置信度

注：在 GB/T 20438.3—2017 的表 A.3 中引用了本技术/措施。

目的：为了避免在开发、验证和维护一个软件包的过程中出现的翻译器失效引起的任何困难。

描述：在先前的许多项目中并未发现性能异常的证据的情况下，才使用一个翻译器。除非存在正确运行的一些其他保证（例如：参看 C.4.4.1），应避免使用没有运行经验或者带有任何已知的严重故障的翻译器。

在一个与安全相关的项目过程中，当翻译器存在微小缺陷时，记录下有关的语言结构并应小心避免使用这些结构。

使用本措施的另一种方案是把语言的使用只限在常用的特征。

这些建议是根据许多项目的经验提出来的。不成熟的翻译器已显示出它对于任何软件开发都是一个严重的障碍。它们使得开发一个安全软件成为不可能。

事实上，目前不存在证明所有工具或者翻译器都正确的方法。

C.4.4.1 源程序和执行代码的比较

目的：为了检验用来产生一幅 PROM（可编程只读存储器）映像的工具未向产生的映像中引入任何错误。

描述：为了得到成分“对象”模块逆向设计 PROM 映像。把这些“对象”模块逆向设计成汇编语言文件。使用适当的技术，把这些逆向生成的汇编语言文件同最初用来产生 PROM 的实际源文件进行比较。

本技术的主要优点是，用于产生 PROM 映像的工具（编译器、连接器等）不必针对所有程序进行确认。本技术可验证用于特定安全相关系统的源文件的转换的正确性。

参考文献：

Demonstrating Equivalence of Source Code and PROM Contents.D.J.Pavey and L.A.Winsborrow. The Computer Journal Vol.36,No.7,1993

Formal demonstration of equivalence of source code and PROM contents:an industrial example.D. J.Pavey and L.A.Winsborrow.Mathematics of Dependable Systems,Ed.C.Mitchell and V.Stavridou, Clarendon Press,1995,ISBN 0-198534-91-4

Assuring Correctness in a Safety Critical Software Application.L.A.Winsborrow and D.J.Pavey. High Integrity Systems,Vol.1,No.5,pp 453-459,1996

C.4.5 合适的编程语言

注：在 GB/T 20438.3—2017 的表 A.3 中引用了本技术/措施。

目的：为了尽可能多地支持 GB/T 20438 的要求，特别是在防御性编程、强类型编程、结构化编程，或许还有断言编程中，所选编程语言易于产生一个便于验证的代码并有助于程序的开发、验证和维护。

描述：应充分地、清楚地定义该语言。该语言应是面向用户或问题的而不是面向处理机/平台的机器语言。广泛使用的语言或它们的子集比专用语言更好。

除了已经提到的特征之外，该语言应提供：

——程序块结构；

——翻译时检查；和

——运行时类型和数组越界检验。

本语言应有助于：

——使用小型的和可管理的软件模块；

——专用软件模块中数据访问的限制；

——变量子范围的定义；和

——任何其他类型的防错构造。

如果系统的安全运行与实时约束有关，那么，语言还应提供异常/中断处理。

需要用一个适当的翻译器、合适的预存软件模块库、一个调试程序和一些工具(用于版本控制和开发两方面)来支持语言。

当前，在开发 GB/T 20438 时，还不清楚面向对象的语言是否比其他传统的语言更好。

使验证困难并因此应避免的特征包括：

——除子程序调用外的无条件转移；

——递归；

——指针，堆或任何类型的动态变量或对象；

——在源码级的中断处理；

——循环、程序块和子程序的多入口和多出口；

——隐式变量初始化或声明；

——变体记录及其等效性；以及

——过程参数。

低级语言，特别是汇编语言，存在因它们面向处理机/平台机的固有特性产生的问题。

一种要求的语言属性是它的设计和使用应产生执行结果可预知的程序。给定一种适当定义的编程语言时，就存在一个能保证可预测程序执行结果的子集。(一般)不能统计地确定这个子集，然而许多静态约束有助于保证执行的可预测性。典型地，这需要证明数组索引是在限界之内以及不会发生数字溢出等。

表 C.1 给出了建议的具体的编程语言。

参考文献：

Concepts in Programming Languages. J. C. Mitchell. Cambridge University Press, 2003, ISBN 0521780985,9780521780988

IEC 60880:2006, Nuclear power plants—Instrumentation and control systems important to safety—Software aspects for computer-based systems performing category A functions

GB/T 15969.3—2005 可编程序控制器 第3部分：编程语言

ISO/IEC 1539-1:2004, Information technology—Programming languages—Fortran—Part 1: Base language

ISO/IEC 7185:1990, Information technology—Programming languages—Pascal

ISO/IEC 8652:1995, Information technology—Programming languages—Ada

GB/T 15272—1994 程序设计语言 C

ISO/IEC 10206:1991, Information technology—Programming languages—Extended Pascal

ISO/IEC 10514-1: 1996, Information technology—Programming languages—Part 1: Modula-2, Base Language

ISO/IEC 10514-3:1998, Information technology—Programming languages—Part 3: Object Oriented Modula-2

ISO/IEC 14882:2003, Programming languages—C++

ISO/IEC/TR 15942:2000, Information technology—Programming languages—Guide for the use of the Ada programming language in high integrity systems

表 C.1 具体的编程语言的建议

编 程 语 言	SIL1	SIL2	SIL3	SIL4
1 ADA	HR	HR	R	R
2 具有子集的 ADA	HR	HR	HR	HR
3 Java	NR	NR	NR	NR
4 具有子集的 Java(不包含无用单元收集，或包含不引发应用代码明显停止的无用单元收集)。见附录 G 面向对象的安全相关软件开发指南	R	R	NR	NR
5 PASCAL(见注释 1)	HR	HR	R	R
6 具有子集的 PASCAL	HR	HR	HR	HR
7 FORTRAN77	R	R	R	R
8 具有子集的 FORTRAN77	HR	HR	HR	HR
9 C	R	—	NR	NR
10 具有子集和编码标准并使用静态分析工具的 C	HR	HR	HR	HR
11 C++(见附录 G 面向对象的安全相关软件开发指南)	R	—	NR	NR
12 具有子集和编程标准并使用静态分析工具的 C++(见附录 G 面向对象设施使用指南)	HR	HR	HR	HR
13 汇编程序	R	R	—	—
14 具有子集和编码标准的汇编程序	R	R	R	R
15 梯形图	R	R	R	R
16 具有定义的语言子集的梯形图	HR	HR	HR	HR
17 功能框图	R	R	R	R

表 C.1（续）

编　程　语　言	SIL1	SIL2	SIL3	SIL4
18　具有定义的语言子集的功能框图	HR	HR	HR	HR
19　结构化文本	R	R	R	R
20　具有定义的语言子集的结构化文本	HR	HR	HR	HR
21　顺序功能图	R	R	R	R
22　具有定义的语言子集的顺序功能图	HR	HR	HR	HR
23　指令表	R	—	NR	NR
24　具有定义的语言子集的指令表	HR	R	R	R

注 1：HR、R、—和 NR 在 GB/T 20438.3—2017 的附录 A 中作了说明。

注 2：系统软件包括作为系统组成部分而配置的操作系统、驱动程序、嵌入功能和软件模块，典型地，软件由安全相关系统卖方提供。宜小心选择语言子集以避免可能引起实现故障的复杂结构。宜检查语言子集使用是否得当。

注 3：应用软件是为专门的安全应用而开发的软件。在许多情况下，由最终用户或者一个面向应用的承包商开发软件。当同一建议有许多编程语言时，开发者宜选择在工业或工厂中人们常用的那一种。宜仔细选择语言子集以避免可能会引起实现故障的复杂结构。宜检查语言子集使用是否得当。

注 4：如果在表中未列出一种具体的语言，绝不能认为它被排除在外了。它宜符合 GB/T 20438。

注 5：有许多 Pascal 语言的扩展，包括 Free Pascal.参考包括这些扩展的 Pascal。

注 6：Java 设计有一个运行时无用单元收集器。Java 的子集可以被定义不需要无用单元收集。一些 Java 实现提供了先进的无用单元收集，它们在程序执行时释放空闲内存，从而阻止当没有可用内存时执行停止一段时间。硬实时的应用不宜使用任何形式的无用单元收集。

注 7：如果 Java 实施要求实时 Java 中间代码的翻译，那么翻译器必须视为安全相关软件的一部分，并且被视为与 GB/T 20438.3—2017 的要求一致。

注 8：表项 15-24，见 IEC 61131-3。

C.4.6　自动软件生成

注：GB/T 20438.3—2017 的表 A.2 引用了该技术/措施。

目的：使更易于出错的软件实现的任务自动化。

描述：系统用一种比传统执行码更高层次的抽象模型（一个可执行规范）描述。模型被一个代码生成器自动转换成可执行形式。目的是通过消除易出错的手动编码任务来提高软件的质量。更进一步的潜在好处是，更复杂的设计可以在更高抽象层次进行。

参考文献：

Embedded Software Generation from System Level Design Languages，H Yu，R.Domer，D.Gajski. In "ASP-DAC 2004：Proceedings of the ASP-Dac 2004 Asia and South Pacific Design Automation Conference，2004"，IEEE Circuits and Systems Society.IEEE，2004，ISBN 0780381750，9780780381759

Transforming Process Algebra Models into UML State Machines：Bridging a Semantic Gap?.M.F. van Amstel et.al.In Theory and Practice of Model Transformations：First International Conference，ICMT".ed.A.Vallecillo.Springer，2008，ISBN 3540699260，9783540699262

C.4.7　测试管理和自动化工具

注：GB/T 20438.3—2017 的表 A.5 引用了该技术/措施。

目的:鼓励系统的和全面的方法进行软件和系统测试。

描述:使用合适的支持工具在系统开发中机械化劳动密集且易出错的任务,并且带来测试管理的系统化方法的能力。支持的可用性有助于更彻底的正规测试和回归测试的方法。

参考文献:

Managing the Testing Process:Practical Tools and Techniques for Managing Hardware and Software Testing.R.Black,John Wiley and Sons,2002,ISBN 0471223980,9780471223986

C.5 验证和修改

C.5.1 概率测试

注:在 GB/T 20438.3—2017 的表 A.5、表 C.15、表 A.7 和表 C.17 中引用了本技术/措施。

目的:为了得到被研究软件的可靠性属性的定量数值。

描述:该方法产生软件可靠性的统计估计,该定量数值考虑到了有关的置信度等级和显著性并能给出:

——每一要求的一个失效概率;

——在某一时段的一个失效概率;和

——错误控制的一个概率。

从这些数值可导出其他一些参数,比如:

——无失效执行的概率;

——存活率;

——可用性;

——MTBF(平均失效间隔时间)或者失效率;以及

——安全执行的概率。

概率考虑是以概率测试或操作经验为基础的。通常,测试用例或者观察的操作用例数目都很大。典型地,要求运行模式的测试占用的时间比连续运行模式测试占用的时间少得多。

一般使用自动测试工具来提供测试数据和监控测试输出。大量的测试要在具有适合过程模拟外围设备的大型主计算机上运行。根据系统和随机硬件两种观点来选择试验数据。例如,整体测试控制保证了一个测试数据分布,而随机选择能管理各个测试用例的细节。

如上所述,各个测试的工具、测试执行和测试监控由详细的测试目的确定。其他重要条件由数学前提给出,此数学前提是在测试评价满足预定测试目的时必须满足的。

有关任何测试对象行为的概率数值还可从操作经历导出,倘若满足同样的条件,评价测试结果就可使用同样的理论。

实际上,很高的可靠性很难通过这些技术证明。

参考文献:

A discussion of statistical testing on a safety-related application. S Kuball, J H R May. Proc IMechE Vol.221 Part O:J.Risk and Reliability,Institution of Mechanical Engineers,2007

Estimating the Probability of Failure when Testing Reveals No Failures,W.K.Miller,L.J.Morell, et al..IEEE Transactions on Software Engineering,VOl.18,NO.1,pp33-43,January 1992

Reliability estimation from appropriate testing of plant protection software,J.May,G.Hughes,A. D.Lunn.IEE Software Engineering Journal,v10 n6 pp 206-218,Nov 1995(ISSN:0268-6961)

Validation of ultra high dependability for software based systems,B.Littlewood and L.Strigini. Comm.ACM 36(11),69-80,1993

C.5.2 数据记录和分析

注：在 GB/T 20438.3—2017 的表 A.5 和表 A.8 中引用了本技术/措施。

目的：为使验证、确认、评估和维护较容易，软件项目中的所有数据，判定和基本原理都编成文档。

描述：在一个项目期间，保持详细的记录文档，该文档包括：

——对每个软件模块执行的测试；

——判定和它们的基本原理；

——问题和解决办法。

在项目进行过程中和结束时，可分析这个文档以便建立各种各样的信息。特别是当开发项目期间作出某些判定的基本原理还不为维护工程师所知时，数据记录对计算机系统的维护是很重要的。

参考文献：

Dependability of Critical Computer Systems 2.F.J.Redmill，Elsevier Applied Science，1989，ISBN ISBN 1851663819，9781851663811

C.5.3 接口测试

注：在 GB/T 20438.3—2017 的表 A.5 中引用了本技术/措施。

目的：为了检测子程序接口中的错误。

描述：测试的详细程度或者完整性分成几级是可行的。最重要的一些级别是对下列量的测试：

——处于它们的极限值的所有接口变量；

——分别处于它们的极限值的所有接口变量，以及处于额定值的另一些接口变量；

——遍历其范围内所有值的单个接口变量，以及处于额定值的其他接口变量；

——组合(只有简单接口这种组合才是可行的)中的所有变量的所有值；

——与每个子程序的每次调用有关的规定的测试条件。

当接口不包括检测错误参数值的断言时，这些测试是特别重要的。在对已有子程序生成新配置后，它们也是重要的。

C.5.4 边界值分析

注：在 GB/T 20438.3—2017 的表 B.2、表 B.3 和表 B.8 中引用了本技术/措施。

目的：为了检测发生在参数极限值或边界值处的软件错误。

描述：根据等价关系(参看 C.5.7)把程序的输入范围分成若干输入类别。测试应包括这些类的边界值和极限值。测试将检查和程序中规范一致的规范的输入域的边界。在直接和间接翻译中使用值零通常易出错并应特别注意：

——零除数；

——空白 ASCⅡ字符；

——空栈或者表元素；

——满矩阵；

——零表项。

通常，输入的边界直接对应于输出范围的边界。应把测试用例写为将输出强制成它的限定值。还应考虑是否有可能规定一个能使输出超过规范边界值的测试用例。

如果输出是一个数据序列，例如一个打印的表，应特别注意第一个和最后一个元素以及不包含元素、只包含一个和两个元素的表。

参考文献：

The Art of Software Testing，second edition.G.J.Myers，T.Badgett，T.M.Codd，C.Sandler，John

Wiley and Sons,2004,ISBN 0471469122,9780471469124

C.5.5 错误推测

注：在 GB/T 20438.3—2017 的表 B.2 和表 B.8 中引用了本技术/措施。

目的：为了消除常见的编程错误。

描述：测试经验和直觉，结合对被测系统的认知和兴趣，可把一些未分类的测试用例附加到计划的测试用例集上。

特殊的值或者值的组合易出错。可以从审查检查表得出某些感兴趣的测试用例。也要考虑系统是否足够健壮。例如，在面板上按按钮太快或者太频繁？同时按两个按钮会发生什么情况？

参考文献：

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

C.5.6 错误播种

注：在 GB/T 20438.3—2017 的表 B.2 中引用了本技术/措施。

目的：为了弄清一组测试用例是否充分。

描述：在程序中插入(播种)一些已知类型的错误并在测试条件下执行测试用例程序。如只发现播种的部分错误，测试用例集是不合适的。发现的播种错误与播种的错误总数之比可用来估算真实的发现的错误与错误总数之比。它给出估算剩余错误数的可能性并因此也给出了剩余的测试工作。

$$\frac{\text{发现的播种错误}}{\text{播种的错误总数}} = \frac{\text{真实的发现错误}}{\text{真实错误的总数}}$$

检测到所有播种的错误既可能反映测试用例集是合适的，也可能反映播种的错误太容易发现了。本方法的局限性在于要想得到可用的结果，错误的类型和播种的位置必须反映真实错误的统计分布情况。

参考文献：

Software Fault Injection: Inoculating Programs Against Errors. J. Voas, G. McGraw. Wiley Computer Pub., 1998, ISBN 0471183814, 9780471183815

Faults, Injection Methods, and Fault Attacks. Chong Hee Kim, Jean-Jacques Quisquater, IEEE Design and Test of Computers, vol.24, no.6, pp.544-545, Nov., 2007

Fault seeding for software reliability model validation. A. Pasquini, E. De Agostino. Control Engineering Practice, Volume 3, Issue 7, July 1995. Elsevier Science Ltd

C.5.7 等价类和输入划分测试

注：在 GB/T 20438.3—2017 的表 B.2 和表 B.3 中引用了本技术/措施。

目的：为了使用最少的测试数据充分地测试软件。通过选择考验软件所需的输入范围的分区来得到测试数据。

描述：本测试策略是以输入的等价关系为基础的，该关系确定了输入范围的一个划分。

为了覆盖早先规定的所有划分，应对测试用例进行选择。从每个等价类至少要选一个测试用例。

对于输入划分存在两种基本的可能性，它们是：

——从规范得出的等价类——规范的解释既可是面向输入的，例如选择值按相同的方法进行，也可以是面向输出的，例如值的集合产生相同的功能结果。

——从程序的内部结构得出的等价类——从程序的静态分析确定等价类结果，例如值的集合产生同样的执行路径。

参考文献：

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

Static Analysis and Software Assurance. D. Wagner, Lecture Notes in Computer Science, Volume 2126/2001, Springer, 2001, ISBN 978-3-540-42314-0

C.5.8 基于结构的测试

注：在 GB/T 20438.3—2017 的表 B.2 中引用了本技术/措施。

目的：为了使用考验某些程序结构子集的测试。

描述：在程序分析的基础上，选择一组输入数据，从而可考验大部分（并且常常是预定的目标）程序代码。代码覆盖率的措施取决于要求的严格度，如下所示。在所有情况下，应以 100% 的覆盖为目标；如果不可能实现 100% 的覆盖率，不能实现 100% 覆盖率的原因应记录在测试报告中（例如，如果硬件问题出现，才能进入防御性代码）。下列列表中的前四种技术在 GB/T 20438.3—2017 的表 B.3 中被特别提到并且被测试工具广泛支持；剩余的技术也应该考虑。

——入口点（调用图表）覆盖率：确保每个子程序或函数至少被调用一次（这是最不严格的结构覆盖度量）。

注：在面向对象的语言中，可以有多个同名的子程序，子程序适用于一个多态类型的不同变种（重新定义的子程序），这种类型可以通过动态调度来调用。在这些情况下每一个这样的重新定义的子程序都宜进行测试。

——语句：确保所有代码中的语句都至少被执行一次。

——分支：应检验每个分支的两边。对某些类型的防御性代码这可能是不实际的。

——复合条件：考验一个复合条件分支（即用 AND（与）/OR（或）链接的）的每个条件。参看 MCDC（修改的条件判定覆盖，参考 DO-178B）。

——LCSAJ：一个线性代码序列和转移（LCSAJ）是包括条件语句和用一个转移结束的任何代码语句序列。由于执行早期代码强加给输入数据的约束，许多潜在的子路径是不可行的。

——数据流：根据数据的使用情况选择执行路径。例如，一条路径中同一变量既被读又被写。

——基本路径：从开始到结束的一个有限路径的最小集合中的一条路径，因此包括所有弧线。（在这个基本集合中路线的重叠式组合可形成通过程序那一部分的任何路径），已经显示出，所有基本路径的测试对查找错误是有效的。

参考文献：

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

RTCA, Inc. document DO-178B and EUROCAE document ED-12B, Software Considerations in Airborne Systems and Equipment Certification, dated December 1, 1992

C.5.9 控制流分析

注：在 GB/T 20438.3—2017 的表 B.8 中引用了本技术/措施。

目的:为了检测差的和有潜在错误的程序结构。

描述:控制流分析是查找代码可疑区的一种静态测试技术,该代码的可疑区并不遵循好的编程作法。对程序进行分析可产生一个有向图,此图能进一步分析:

——不可访问的代码,例如无条件转移,它留下执行不到的代码块。

——打结的代码:良结构代码有一个可简化的控制图,该代码能简化成一个单节点。相反,差结构代码只能简化成几个节点构成的一个结。

参考文献:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

C.5.10 数据流分析

注:在 GB/T 20438.3—2017 的表 B.8 中引用了本技术/措施。

目的:为了检测差的和有潜在错误的程序结构。

描述:数据流分析是一种静态测试技术,此技术把从控制流分析得到的信息同在各代码分区中读或写的变量有关的信息组合起来,可以检查分析:

——在给变量赋值之前读出的那些变量——可以通过声明一个新变量时总是分配给它一个值来避免这一点。

——在读之前多次写的那些变量——这可指示省略的代码。

——只写而决不会读的那些变量——这能指示多余代码。

一个数据流的异常结构不一定直接相应于一个程序错误,但如果避免了这种异常,代码就很少包含错误。

参考文献:

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

C.5.11 符号执行

注:在 GB/T 20438.3—2017 的表 B.8 中引用了本技术/措施。

目的:为显示源码和规范之间的一致性。

描述:在所有赋值中,在用右值替换左值之后再评价程序变量。条件分支和循环翻译成布尔(Boolean)表达式。最后结果是每个程序变量的一个符号表达式。如果给定真实的数据,该表达式是程序将要计算的值组成的公式。可对照预期的表达式来检验它。

符号执行的相关使用是作为一种系统化的方法:通过程序逻辑路径来生成测试数据。符号执行设备可能被合并入一个集成的工具中,来为软件组件测试和代码分析提供一种工具。

参考文献:

Using symbolic execution for verifying safety-critical systems. A. Coen-Porisini, G. Denaro, C. Ghezzi, M. Pezzé. Proceedings of the 8th European software engineering conference, and 9th ACM SIGSOFT international symposium on Foundations of software engineering. ACM, 2001, ISBN: 1-58113-390-1

Using symbolic execution to guide test generation. G. Lee, J. Morris, K. Parker, G. Bundell, P. Lam. In Software Testing, Verification and Reliability, vol 15, no 1, 2005. John Wiley & Sons, Ltd

C.5.12 形式化证明(验证)

注：在 GB/T 20438.3—2017 的表 A.5 与表 A.9 中引用了本技术/措施。

目的：使用了理论和数学模型及规则，证明程序针对其某种抽象模型的正确性。

描述：测试是一种检验程序正确性的常见方式。然而，实际值的程序的复杂性通常无法实现详尽的测试。因此只有一小部分可能的程序可以用这个方法检查。相反，形式化验证应用数学运算于一个程序的数学表示，为所有可能的输入来建立确定的程序的运行状况。

系统的形式化验证要求使用精确数学含义的语言描述的一个程序及其所需行为(即，规范)的抽象模型。这个规范可能是完整的，或者它可能被限定到特定的程序属性：

——功能正确性属性，即程序应当显示一个特定的功能；

——安全(即一些不好的运行状况永远不会发生)和活性(即一些好的运行状况最终会发生)属性；

——时间属性，即一些运行状况将在特定时间发生。

形式化验证的结果是一项严密的论证：针对所有可能的输入，与该规范有关的程序的抽象模型是正确的。即模型满足特定的属性。

但是，该模型的正确性并不直接证明实际程序的正确性，需进一步说明：针对关心的属性，该模型是一个实际程序的精确抽象。一些实际关心的程序属性不能以数学方法形式化(例如大多数计时和调度，或主观的属性。例如“清晰而简单的”用户接口，或者确实不易用形式化语言表达的任何属性或设计目标)。因此形式化验证不会完全取代模拟和测试，而是针对所有的输入，通过提供进一步的证据来支持程序的正确运行来补充这些技术。当形式化验证可以确保程序的抽象模型的正确性时，通过测试来保证实际期望的程序运行状况。

在设计阶段，形式化验证的使用可能通过在设计阶段的早期发现重大错误和疏忽来大大减少开发时间，从而减少了设计与测试间迭代的时间。

一些实际应用中的形式化方法在 C.2.4 中有描述：例如，CCS，CSP，HOL，LOTOS，OBJ，时序逻辑，VDM 和 Z。

C.5.12.1 模型检查

模型检查是一种方法，这种方法是针对反应的和并发的系统的形式化验证。给定一个有限状态结构，该结构描述了系统的运行状况。如果一个写成时序逻辑公式形式的属性保持或不违反该结构，则该属性得到了检查。高效算法(如，SPIN、SMV 和 UPPAAL) 被用于自动地和全面彻底地遍历结构的整个状态。当属性不能维持时，就生成一个反例。它显示在结构中属性是如何被违反的，以及用于调查系统的非常有用的信息。模型检查可以检测传统检查和测试无法发现的“深层缺陷”。

要注意，在分析精细复杂性方面，模型检查是有效的。其在一些低 SIL 应用方面可能有效，但是如果精细复杂性存在于高 SIL 应用程序中，则需要小心。

参考文献：

Is Proof More Cost-Effective Than Testing?, S. King, R. Chapman, J. Hammond, A. Pryor. IEEE Transactions on Software Engineering, vol.26 no.8, August 2000

Model Checking. E. M. Clarke, O. Grumberg, and D. A. Peled. MIT Press, 1999, ISBN 0262032708,9780262032704

Systems and Software Verification: Model-Checking Techniques and Tools. B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, and P. Mckenzie, Springer, 2001, ISBN 3-540-41523-8

Logic in Computer Science: Modelling and Reasoning about Systems. M. Huth and M. Ryan. Cambridge University Press, 2000, ISBN 0521652006,0521656028

The Spin Model Checker: Primer and Reference Manual. G. J. Holzmann. Addison-Wesley, 2003, ISBN 0321228626, 9780321228628

C.5.12.2 （无效）

C.5.13 复杂性度量

注：在 GB/T 20438.3—2017 的表 B.9 与表 C.19 中引用了本技术/措施。

目的：为了从软件本身的属性或者从它的开发史或者测试史预测程序属性。

描述：这些模型评价软件的某些结构属性，并把它同一个希望的属性（比如可靠性或者复杂性）联系起来。这些措施中的大部分需要使用软件工具评价。下面给出了可以使用的一些度量：

——图形理论复杂性——本度量方法可用于生命周期的初期进行比较评估，它以程序控制图的复杂性为基础，复杂性用它的圈复杂度（cyclomatic number）来表示。

——启动某个软件模块的方式数（可访问性）——一个软件模块越多被访问，就越可能得到调试。

——Halstead（赫尔斯梯德）型度量学——这种措施通过统计操作符和操作数来计算程序长度。它提供了复杂性和长度的一种量度，它为估算今后开发资源时作比较提供了一个基准。

——每个软件模块的入口/出口数——最小化入口/出口点数是结构化设计和编程技术的一个关键特征。

参考文献：

Metrics and Models in Software Quality Engineering. S. H. Kan. Addison-Wesley, 2003, ISBN 0201729156, 9780201729153

C.5.14 形式化审查

注：在 GB/T 20438.3—2017 的表 B.8 中引用了本技术/措施。

目的：为了揭示软件组件的缺陷。

描述：形式化审查是一种审查软件材料的结构化过程：它由制造材料之外的人执行，来找出缺陷和确保制造者改进材料。除了在熟悉阶段为审查者提供简要说明，被审查者不应参加审查过程。形式审查可能被执行在软件开发生命周期的任何阶段的特定软件组件上。

审查发生前，审查人员应当熟悉要审查的材料。在审查过程中审查人员的角色应明确定义。审查日程应该准备好。基于所需软件组件的属性，进入和退出的准则应该明确。进入的准则是在审查发生前必须满足的准则或要求。退出的准则是完成特定的审查过程所必须满足的准则或要求。

在审查中，审查的调查结果应该被仲裁人正式记录，仲裁人的任务是为了促进审查。所有的审查人员应该达成调查结果的一致。缺陷应该被归类于 a）在验收前需要改正；或 b）通过给定的时间/里程碑实现需要的改正。在完成审查之后，被识别的缺陷应该明确指派改正缺陷的被审查者。依赖于被识别的缺陷的数目和范围，仲裁人可以确定是否有必要对软件材料作进一步审查。

参考文献：

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

Fagan, M. Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal 15, 3(1976): 182-211

C.5.15 走查(软件)

注：在 GB/T 20438.3—2017 的表 B.8 中引用了本技术/措施。

目的：揭示规范和实现之间的差异。

描述：走查是一种非形式化的技术，是由软件组件的制造者在有他人在场的情况下实行，目标是发现软件组件中的缺陷。走查可在软件开发生命周期任何阶段产生的特定软件组件上执行。

安全相关系统规定的功能被检查和评估，来确保安全相关系统遵守在规范中的给定的要求。任何与产品的实现和使用相关的疑点要记录下来，这样这些疑点就可能得到解决。与形式化审查相比，在走查程序中作者是积极的。

参考文献：

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

C.5.16 设计复审

注：在 GB/T 20438.3—2017 的表 B.8 中引用了本技术/措施。

目的：揭示软件设计中的缺陷。

描述：设计复审是一个形式化的，文档化的，综合的和系统性的软件设计检查，来评估设计要求和设计能力以满足这些要求并确定问题及建议解决方案。

设计复审提供了一种方法来评价针对输入要求的设计状态，也提供了一种鉴定进一步改进的机会的方法。随着生命周期活动的进展，以及主要详细设计到达一个里程碑时，应实施设计复审来复审所有的接口方面，确保设计能被验证以确保设计满足要求。这样一个回顾主要是为了验证设计者的工作并且应该被视为是一种证实和精细化的活动。

参考文献：

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

The Art of Software Testing, second edition. G. J. Myers, T. Badgett, T. M. Codd, C. Sandler, John Wiley and Sons, 2004, ISBN 0471469122, 9780471469124

IEC 61160:2005, Design review

Space Product Assurance, Sneak analysis—Part 2: Clue list. ECSS-Q-40-04A Part 2. ESA Publications Division, Noordwijk, 1997, ISSN 1028-396X, http://www.everyspec.com/ESA/ECSS-Q-40-04A_Part-2_14981/

C.5.17 原型设计/动画

注：在 GB/T 20438.3—2017 的表 B.3 和表 B.5 中引用了本技术/措施。

目的：为了根据给定的约束检查实现系统的可行性。为了把系统的规范制定者的理解传送给用户以便发现误解。

描述：选择系统功能、约束和性能要求的一个子集。使用高级工具建立一个原型。在这一阶段，不

需要考虑比如目标计算机、实现语言、程序规模、可维护性、可靠性和可用性这些约束。对照用户判据来评价原型并且通过这种评价来修改系统要求。

参考文献：

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

C.5.18 过程仿真

注：在 GB/T 20438.3—2017 的表 A.7，表 C.7，表 B.3 和表 C.13 中引用了本技术/措施。

目的：为了在无实际使用环境的情况下，测试一个软件系统的功能及系统与外界的接口。

描述：建立一个系统，该系统只用于测试目的，它模拟受控设备(EUC)的行为。

仿真可以只是软件或者是软件和硬件的组合。它必须：

——提供输入，这些输入等同于实际安装受控设备时存在的输入。

——忠实代表受控设备，对被测软件的输出做出响应。

——提供操作员输入的条件，以提供被测系统所必须应对的干扰。

当正在测试软件时，仿真可以是带有它输入和输出的目标硬件的仿真。

参考文献：

EmStar: An Environment for Developing Wireless Embedded Systems Software. J Elson et al. http://cens.ucla.edu/TechReports/9_emstar.pdf

A hardware-software co-simulator for embedded system design and debugging. A. Ghosh et al. In Proceedings of the IFIP International Conference on Computer Hardware Description Languages and Their Applications, IFIP International Conference on Very Large Scale Integration, 1995. IEEE, 1995, ISBN 4930813670,9784930813671

C.5.19 性能要求

注：在 GB/T 20438.3—2017 的表 B.6 中引用了本技术/措施。

目的：为了制定一个软件系统的可证明的性能要求。

描述：对系统和软件需求规范两者执行一次分析以便确定所有通用和专用、明显的和隐含的性能要求。

检验每个性能要求以便逐个：

——确定获得成功的判据；

——对照成功判据确定一次测量是否能获得成功；

——确定这种测量的可能精度；

——确定项目分段，在这些阶段估算这些测量；

——确定项目分段，在这些阶段进行测量。

为了得到性能要求、成功判据和可能的测量的一张表，则要分析每个性能要求的可行性。主要的目标是：

——每个性能要求至少与一次测量相关；

——在可能的情况下选择可能用于开发初期的精确、有效的测量；

——规定必选的和可选的性能要求和成功判据；

——在可能的情况下，对于多个性能要求利用使用单次测量的可能性。

参考文献：

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

C.5.20 性能建模

注：在 GB/T 20438.3—2017 的表 B.2、表 B.5 中引用了本技术/措施。

目的：为了保证系统工作能力足以满足规定的要求。

描述：要求规范包括具体功能的允许量和响应要求，也许还要和使用系统总资源的约束相结合。提出的系统设计将借助以下办法同说明的要求进行对比：

——生成一个系统过程及过程间相互作用的模型；

——通过每个过程确定资源的使用，例如，处理机时间、通信带宽、存储器件等；

——确定平均的和最差情况条件下对系统提出的要求的分布；

——计算在平均的和最差情况下各个系统功能的允许量和响应时间。

对简单的系统而言，使用分析方法可能就足够了，而对较复杂的系统而言，某种形式的模拟可能更适于获得精确的结果。

在详细的建模之前，可以使用一种较简单的“资源预算”检验，它可把所有过程的资源需求总和起来。当需求超过设计的系统能力时，设计是不可行的。即使设计通过了这种检验，性能建模将显示由于资源不足产生的延迟和响应时间过长。为了避免这种情况，工程师常常使用总资源的一部分(例如50%)来设计系统以便减小资源不足的可能性。

参考文献：

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

C.5.21 雪崩/过载测试

注：在 GB/T 20438.3—2017 的表 B.6 中引用了本技术/措施。

目的：为了给测试对象加上一个超高工作负荷以便显示测试目标可以毫不费力地承受额定的工作负荷。

描述：存在各种可使用于雪崩/过载测试的测试条件。这些测试条件中的一些包括：

——如果是以轮询方式工作，则当处在正常条件下时，每个时间单位测试对象的输入的变化要增大很多。

——当工作在要求模式时，则每个时间单位向测试对象的要求数目增加超过正常条件。

——如果一个数据库的规模起某种重要作用，则把它增大超过正常条件。

——影响装置分别调到它们的最高或者最低速度。

——在极端情况时，所有影响因素尽可能同时提供给边界条件。

在这些测试条件下，可评价测试对象的时间行为，还可观察负荷变化的影响，并能检查内部缓冲器或者动态变量、堆栈等的正确规模。

参考文献：

Software Engineering for Real-time Systems. J. E. Cooling, Pearson Education, 2003, ISBN 0201596202,9780201596205

C.5.22 响应时间特性和存储约束

注：在 GB/T 20438.3—2017 的表 B.6 中引用了本技术/措施。

目的：为了保证系统满足它的时序和存储要求。

描述：系统和软件的要求规范包含具体功能的存储和响应要求，也许还要结合对使用系统总资源的约束。

为了确定在平均的和最差情况条件下的要求的分布而进行了一种分析。该分析需要估算每个系统

功能使用的资源和占用的时间，有几种办法可得到这些估算，例如，与一个现有系统或原型设计进行比较，以及对时间关键系统进行时间基准测试。

C.5.23 影响分析

注：在 GB/T 20438.3—2017 的表 A.8 中引用了本技术/措施。

目的：为了确定一个软件系统的一个改变或者增强将对该软件系统中的其他软件模块以及其他系统的影响。

描述：在对软件进行一次修改或增强之前，为了确定这种修改或增强对该软件的影响，还为了确定哪些软件系统和软件模块将受影响，应进行一次分析。

在完成分析之后，需要对重新验证软件系统的问题作出决定。这与受影响的软件模块数、受影响的软件模块的关键程度和特性的改变有关。可能的决定有：

——只重新验证被改变的软件模块。

——重新验证所有受影响的软件模块，或者

——重新验证整个系统。

参考文献：

Requirements Engineering. E. Hull, K. Jackson, J. Dick. Springer, 2005, ISBN 1852338792, 9781852338794

C.5.24 软件配置管理

注：在 GB/T 20438.3—2017 的表 A.8 中引用了本技术/措施。

目的：软件配置管理的目的是为了保证当那些可交付项有改变时，几种开发的可交付项的一致性。一般配置管理可用于硬件和软件开发两方面。

描述：软件配置管理是在整个开发过程中使用的一种技术(见 GB/T 20438.3—2017 的 6.2.3)。实质上，它要求记录每个重要的可交付项的每个版本的产生及各个可交付项的不同版本之间的每种关系的产生。产生的文档允许开发者确定一个可交付项(特别是它的一个组件)的改变对其他可交付项的影响。特别是可以从一致的组件版本集可靠地重建各系统或子系统。

参考文献：

Software engineering: Update. Ian Sommerville, Addison-Wesley Longman, Amsterdam; 8th ed., 2006, ISBN 0321313798, 9780321313799

Software Engineering. Ian Sommerville, Pearson Studium, 8. Auflage, 2007, ISBN 3827372577, 9783827372574

Software Configuration Management: Coordination for Team Productivity. W. A. Babich. Addison-Wesley, 1986, ISBN 0201101610, 9780201101614

CMMI: guidelines for process integration and product improvement, Mary Beth Chrissis, Mike Konrad, Sandy Shrum, Addison-Wesley, 2003, ISBN 0321154967, 9780321154965

C.5.25 回归确认

注：在 GB/T 20438.3—2017 的表 A.8 中引用了有关的技术和措施。

目的：为了确保从回归测试中得到有效的结论。

描述：大型或复杂系统的完整的回归测试通常会需要很多精力和资源。在可能的情况下，限制回归测试到只覆盖在系统开发的那个阶段关心的系统方面，这种做法是可取的。在这部分回归测试中，有一个对部分测试的范围的清晰理解是必要的，并且只得到与系统经过测试的状态相关的有效结论也是必要的。

参考文献：

Managing the Testing Process:Practical Tools and Techniques for Managing Hardware and Software Testing.R.Black,John Wiley and Sons,2002,ISBN 0471223980,9780471223986

C.5.26 规范和设计的动画演示

注：在 GB/T 20438.3—2017 的表 A.9 中引用了有关的技术和措施。

目的：通过对规范的系统性检查指导软件验证。

描述：软件的表现被检查来确定最终可执行软件的运行状况，这种表现比可执行代码(例如规范或者高层次设计)更抽象。这种检查在一些方面是自动的(取决于由更高层表达的抽象的性质和层次所提供的可能性)以便模拟该运行状况和可执行软件的输出。这种方法的一种应用是生成测试(或者“指示物”)：能随后被应用于可执行软件，这样在某种程度上是测试过程自动化。这种方法的另一种应用是使用户接口活泼起来，以便非技术的终端用户能够得到对软件开发者工作所依据规范的详细意义的一些了解。这样就提供了一种在二者之间交流的有价值的方法。

参考文献：

Supporting the Software Testing Process through Specification Animation.T.Miller,P.Strooper.In Proceedings of the First International Conference on Software Engineering and Formal Methods (SEFM’03), ed. P. Lindsay. IEEE Computer Society, IEEE Computer Society, 2003, ISBN 0769519490,9780769519494

B model animation for external verification. H. Waeselynck, S. Behnia, In Proceedings. Of the Second International Conference on Formal Engineering Methods,1998.IEEE Computer Society,1998, ISBN 0-8186-9198-0

C.5.27 基于模型的测试(测试用例的生成)

注：在 GB/T 20438.3—2017 的表 A.5 中引用了有关的技术和措施。

目的：为了从系统模型中促进高效的自动测试用例的生成和为了生成高度可重复测试程序组。

描述：基于模型的测试(MBT)是一个黑盒方法，在该方法中常见的测试，比如测试用例生成(TCG)和测试结果评估是基于被测系统(SUT)的模型(应用程序)。通常，但不是只有，系统数据和用户行为用有限状态机、马尔科夫过程、定值表或类似的(EL-Far,2001,generalized)来建模。此外，基于模型的测试可以与源代码级测试覆盖率措施组合起来，并且功能模型能依据存在的源代码。

基于模型的测试是有效测试用例/程序的自动化生成，它使用系统要求和规定功能的模型(软件技术,2009)。

由于测试是非常昂贵的，对于自动化测试用例生成工具有巨大的需求。因此，基于模型的测试是当前的一个非常活跃的研究领域，导致大量的测试用例生成(TCG)工具可选择。这些工具通常从模型的运行部分中提取测试程序组，保证满足一定的覆盖率要求。

该模型是被测系统(SUT)期望的运行状况的抽象的、部分的表示。从这个模型中，衍生出测试模型，并建立一个抽象测试程序组。测试用例来源于这种抽象测试程序组，并且针对系统来执行，且测试也能针对系统模型运行。用 TCG 的 MBT 基于形式化方法并与形式化方法的使用强烈相关，所以与安全完整性等级(SIL)相关的推荐是相似的：对较高 SIL 高度推荐，对较低 SIL 不要求。

具体活动通常包括：

——建立模型(从系统需求中)；

——产生预期的输入；

——产生预期的输出；

——运行测试；

——比较实际输出与预期的输出；

——决定进一步行动(修改模型、产生更多的测试、估计软件的可靠性/质量)。

针对用户/系统运行状况的表达模型，测试可以来自于不同的方法和技术，例如：

——通过使用判定表；

——通过使用有限状态机；

——通过使用语法；

——通过使用马尔可夫链模型；

——通过使用状态图；

——通过定理证明；

——通过约束逻辑编程；

——通过模型检查；

——通过符号执行；

——通过使用一个事件流模型；

——反应系统测试：并行分层有限自动化；

——等等。

基于模型的测试最近专门把安全关键领域作为目标。它使规范和设计中的歧义在早期暴露，提供了自动生成许多不重复的有效测试的能力，来评估回归测试程序组和评价软件可靠性及质量，以及减少了测试程序组的更新。

由 El-Far(2001)和软件技术 2009(见参考文献)提供了全面的概述，其他细节和特定问题在其他参考文献中讨论。

参考文献：

T. Bauer, F. Böhr, D. Landmann, T. Beletski, R. Eschbach, Robert and J. H. Poore, From Requirements to Statistical Testing of Embedded Systems Software Engineering for Automotive Systems-SEAS 2007, ICSE Workshops, Minneapolis, USA

Eckard Bringmann, Andreas Krämer; Model-based Testing of Automotive Systems In: ICST, pp. 485—493, 2008 International Conference on Software Testing, Verification, and Validation, 2008

Broy M., Challenges in automotive software engineering, International conference on Software engineering(ICSE '06), Shanghai, China, 2006

I.K.El-Far and J.A.Whittaker, Model-Based Software Testing. Encyclopedia of Software Engineering(edited by J.J.Marciniak). Wiley, 2001

Heimdahl, M.P.E.: Model-based testing: challenges ahead, Computer Software and Applications Conference(COMPSAC 2005), 25—28 July 2005, Edinburgh, Scotland, UK, 2005

Jonathan Jacky, Margus Veanes, Colin Campbell, and Wolfram Schulte, Model-Based Software Testing and Analysis with C#, ISBN 978-0-521-68761-4, Cambridge University Press 2008

A.Paradkar, Case Studies on Fault Detection Effectiveness of Model-based Test Generation Techniques, in ACM SIGSOFT SW Engineering Notes, Proc.of the first int.workshop on Advances in model-based testing A-MOST '05, Vol.30 Issue 4.ACM Press 2005

S.J.Prowell, Using Markov Chain Usage Models to Test Complex Systems, HICSS '05: 38th Annual Hawaii, International Conference on System Sciences, 2005

Mark Utting and Bruno Legeard, Practical Model-Based Testing: A Tools Approach, ISBN 978-0-12-372501-1, Morgan-Kaufmann 2007

Hong Zhu et al.(2008). AST '08: Proceedings of the 3rd International Workshop on Automation of Software Test. ACM Press. ISBN 978-1-60558-030-2

Model-Based Testing of Reactive Systems Advanced Lecture Series, LNCS 3472, Springer-Verlag, 2005, ISBN 978-3-540-26278-7

Model-based Testing, SoftwareTech July 2009, Vol.12, No.2, Software Testing: A Life Cycle Perspective, http://www.goldpractices.com/practices/mbt/

C.6 功能安全评估

注：在 B.6 中也可找到有关的技术和措施。

C.6.1 判定表(真值表)

注：在 GB/T 20438.3—2017 的表 A.10 和表 B.7 中引用了本技术/措施。

目的：为了提供复杂逻辑组合和关系的一个清楚的和有条理的规范和分析。

描述：本方法使用了二维表来简洁地描述布尔程序变量之间的逻辑关系。

本方法的简明性和表格特性使它适于作为一种分析用代码表示的复杂逻辑组合的方法。

当把本方法用作一个规范时，它有可能是可执行的。

C.6.2 软件危险和可操作性分析(CHAZOP, FMEA)

目的：为了确定在一个建议的或者现存的系统中的危险、它们可能的起因和后果以及为减少它们发生的概率而建议的行动。

描述：一个由所研究的整个系统方面的专家组成的工程师小组通过有计划的系列会议对设计进行结构化检查。他们考虑设计的功能方面和实际中如何操作系统(包括人的活动和维护)两个问题。组长鼓励小组成员创造性地揭示潜在危险，通过将系统各部分关联到若干引导词："none(没有)"、"more of(更多的)"、"Less of(更少的)"、"part of(……的一部份)"、"more than(大于……)"(或者"as well as(以及)")和"other than(而不是……)"推动会议进程。考虑各种使用的条件和失效模式的可行性、为何产生、可能的后果(有无危险?)、如何避免以及为避免而使用的技术的代价。

在项目开发的许多分阶段都要进行危险分析，但最有效的执行时间是早到足以影响主要设计和可操作性判定的阶段。

HAZOP 技术在流程工业中逐渐成熟，但应用于软件时需做修改。对于 PES HAZOP(或者 Computer HAZOP—"CHAZOP")已提出了各种派生的方法。通常这些方法引进了一些新引导词，并提出了系统性地包含系统和软件架构的方案。

参考文献：

OF-FMEA: an approach to safety analysis of object-oriented software intensive systems, T. Cichocki, J.Gorski. In Artificial Intelligence and Security in Computing Systems: 9th International Conference, ACS '2002. Ed. J. Soldek. Springer, 2003, ISBN 1402073968, 9781402073960

Software FMEA techniques. P. L. Goddard. In Proc Annual 2000 Reliability and Maintainability Symposium, IEEE, 2000, ISBN: 0-7803-5848-1

Software criticality analysis of COTS/SOUP. P. Bishop, T. Clement, S. Guerra. In Reliability Engineering & System Safety, Volume 81, Issue 3, September 2003, Elsevier Ltd., 2003

C.6.3 共因失效分析

注 1：在 GB/T 20438.3—2017 的表 A.10 中引用了本技术/措施。

注 2：另见 GB/T 20438.6—2017 的附录 D。

目的：为了确定多系统或者多子系统中潜在的失效。这种失效在多个部分中同时出现，会削弱冗余

的好处。

描述:安全系统的硬件中常常使用冗余和多数表决。这可避免部件和子系统中妨碍数据正确处理的随机硬件失效。

但有些失效对多个部件或子系统是共同的。例如,当在单间房内安装一个系统时,空调的缺点可能会削弱冗余的好处。系统的其他外部影响,比如火灾、水灾、电磁干扰、坠机和地震也同样如此。系统也可能受与运行和维护有关的意外事故的影响。因此,为运行和维护编制充分和详尽的规程、以及对运行和维护人员进行全面培训是十分重要的。

内部影响也是共因失效的一个主要原因。它们能起源于共同的或同样的部件和它们的接口的设计缺陷以及部件的老化。共因失效分析必须搜查系统的这些潜在的共同失效。共因失效分析的方法是:通常的质量控制;设计复审;由一个独立小组进行验证和测试;根据类似系统反馈的经验分析实际的意外事故。然而分析范围超出了硬件范围。即使在一个冗余系统的各个通道中使用软件多样化,还是有可能在软件方法中存在一些共性,它们将引起共因失效,例如共用的规范中的错误。

当不是严格地在同一时间发生共因失效时,可以借助多通道之间的比较方法采取预防措施,这种比较方法可以在这种失效成为所有通道共有之前检测该失效。共因失效分析应把这种技术包括在内。

参考文献:

Reliability analysis of hierarchical computer-based systems subject to common-cause failures. L. Xing, L. Meshkat, S. Donohue. Reliability Engineering & System Safety Volume 92, Issue 3, March 2007

C.6.4 可靠性框图

注 1:在 GB/T 20438.3—2017 的表 10 中引用了本技术/措施,在 GB/T 20438.6—2017 的附录 B 中使用了本技术/措施。

注 2:也可见 B.6.6.7“可靠性方框图”。

目的:用图解形式建立系统或任务成功运行所必需的事件集和必备的条件。

描述:分析的目标被表示为一个由方框、线和逻辑接点组成的成功的路径。成功的路径从图的一边开始,经过方框和接点,到达图的另一边。一个方框表示了一个条件或者一个事件,并且如果条件正确或这个事件已经发生,则路径能通过该方框。如果该路径到达接点,逻辑接点是满足的,则路径继续。如果它到达分叉点,可以沿着所有分开的路径继续。如果存在至少一条成功的路径穿过图,则分析的目标就运行正常。

参考文献:

IEC 61025:2006, Fault tree analysis(FTA)

From safety analysis to software requirements. K. M. Hansen, A. P. Ravn, A. P, V Stavridou. IEEE Trans Software Engineering, Volume 24, Issue 7, Jul 1998

IEC 61078:2006, Analysis techniques for dependability—Reliability block diagram and boolean methods

附 录 D
（资料性附录）
确定预开发软件的软件安全完整性的一种概率法

D.1 一般要求

本附录根据操作经验提供了关于使用一种概率法来确定预开发软件的软件安全完整性的原始指南。本方法被认为是特别适合作为操作系统、程序库模块、编译器和其他系统软件的认证的组成部分。本附录提供了一个指示：哪些技术是可能的，而只有能胜任统计分析的那些人才能使用这些技术。

注：本附录使用了术语 Confidence level(置信级)，在 IEEE352 中对其作了描述。在 IEC 61164 中使用了一个等效的术语：Significance level(显著级)。

本技术还用来示范软件安全完整性等级的超时增加。例如，按 GB/T 20438.3 的 SIL1 要求建立的软件在大量应用中成功地运行一段适当的时间之后，会显示出达到 SIL2。

表 D.1 显示了经验的无失效要求数和为一个特定安全完整性等级进行认证所需的无失效运行小时数。

按 D.2 中概述的那样，运行经验可以进行数学处理以便补充或者代替统计测试，可以把几个现场得到的运行经验结合起来(即把处理的请求次数或者运行的小时数加起来)，但只是在下列情况下才有可能：

——在电气/电子/可编程电子(E/E/PE)安全相关系统中使用的软件版本是和正在申请的运行经验的那个版本一样的；

——输入空间的操作简表是相似的；

——有一个报告和文档化失效的有效的系统；以及

——满足有关的先决条件(参看 D.2)。

表 D.1 安全整性等级的置信度的必要历史

SIL	低要求运行模式	处理的要求数		高要求或者连续运行模式	总的运行小时数	
	(按要求执行其设计功能的失效频率)	$1-\alpha=0.99$	$1-\alpha=0.95$	(每小时一次危险失效频率)	$1-\alpha=0.99$	$1-\alpha=0.95$
4	$\geqslant 10^{-5}\sim<10^{-4}$	4.6×10^{5}	3×10^{5}	$\geqslant 10^{-9}\sim<10^{-8}$	4.6×10^{9}	3×10^{9}
3	$\geqslant 10^{-4}\sim<10^{-3}$	4.6×10^{4}	3×10^{4}	$\geqslant 10^{-8}\sim<10^{-7}$	4.6×10^{8}	3×10^{8}
2	$\geqslant 10^{-3}\sim<10^{-2}$	4.6×10^{3}	3×10^{3}	$\geqslant 10^{-7}\sim<10^{-6}$	4.6×10^{7}	3×10^{7}
1	$\geqslant 10^{-2}\sim<10^{-1}$	4.6×10^{2}	3×10^{2}	$\geqslant 10^{-6}\sim<10^{-5}$	4.6×10^{6}	3×10^{6}

注 1：$1-\alpha$ 表示置信级。

注 2：关于先决条件和得出本表的详情可参看 D.2.1 和 D.2.3。

D.2 统计测试公式和它们的应用举例

D.2.1 低要求运行模式的简单统计测试

D.2.1.1 先决条件

a) 测试数据分布等于在线运行过程中要求的分布。
b) 相对于一次失效的原因而言,各测试运行在统计学上是互不相关的。
c) 存在一个检测可能发生的任何失效的适当的机制。
d) 测试实例数 $n>100$。
e) 在 n 个测试实例过程中没有发生失效。

D.2.1.2 结果

在置信级 $1-\alpha$ 时,失效概率 P 由下式给出:

$$p \leqslant 1-\sqrt[n]{\alpha} \text{ 或者 } n \geqslant -\frac{\ln\alpha}{p}$$

D.2.1.3 例子

表 D.2 低要求运行模式的失效概率

$1-\alpha$	P
0.95	$3/n$
0.99	$4.6/n$

按 SIL3 在置信度为 95%时的要求的失效概率,在先决条件下,使用公式得出的测试实例为 30 000 个。表 D.1 汇总了每个安全完整水平的结果。

D.2.2 测试一个低要求运行模式的输入空间(域)

D.2.2.1 先决条件

唯一的必要条件是选择测试数据从而在输入空间(域)内给出一个随机的均匀分布。

D.2.2.2 结果

目的是找到一个测试次数 n,这个数是根据正在测试的低要求功能(比如一次安全停机)的输入精度阈值 δ 计算出的必需值。

表 D.3 两个测试点的平均距离

1	$\delta=1/n$
2	$\delta=\sqrt[2]{1/n}$
3	$\delta=\sqrt[3]{1/n}$
k	$\delta=\sqrt[k]{1/n}$
注:k 为任何正整数,值 1、2 和 3 仅仅是个例子。	

D.2.2.3 例子

考虑一次仅有两个变量 A 和 B 有关的安全停机，如果已对划分输入变量对 A 和 B 分区的阈值被正确地处理成 A 和 B 测量范围的 1%这样的一个精度作过验证，在空间 A 和 B 中所需的均匀分布的测试实例数就是：

$$n=1/\delta^2=10^4$$

D.2.3 高要求或者连续运行模式的简单统计测试

D.2.3.1 先决条件

a) 测试数据分布等于在线运行过程中的分布。
b) 不失效的概率的相对减少正比于考虑的时间间隔的长度，不然就是一个常数。
c) 存在检测可能发生的任何失效的一个适当的机制。
d) 测试延续一段测试时间 t。
e) 在 t 内不发生失效。

D.2.3.2 结果

失效概率 λ、置信级 $1-\alpha$ 和测试时间 t 之间的关系是：

$$\lambda=-\frac{\ln\alpha}{t}$$

失效概率反比于两次失效之间的平均工作时间：

$$\lambda=\frac{1}{MTBF}$$

注：GB/T 20438 对每小时的失效概率和 1 小时内的失效率不加区别。严格地说，失效概率 F 和失效率 f 之间的关系为 $F=1-e^{-ft}$。但在 GB/T 20438 的范围内，失效率小于 10^{-5}，在这样小的值时 $F\approx ft$。

D.2.3.3 例子

表 D.4 高要求或者连续运行模式时的失效概率

$1-\alpha$	λ
0.95	$3/t$
0.99	$4.6/t$

为了检定在一个置信级为 95%的情况下，两次失效之间的平均时间至少为 10^8 h，至少需要 3×10^8 h 的测试时间，并且还必须满足先决条件，表 D.1 汇总了每个安全完整性等级所需的测试数。

D.2.4 完全测试

把程序当作是装有已知的球数 N 的一个缸。每个球代表程序的一个重要特性。随机地抽取这些球，并在检查后更换另一个。当抽取完所有的球时，也就达到了完全测试。

D.2.4.1 先决条件

a) 测试数据这样分布：在等概率的情况下测试程序 N 个属性的每一个；
b) 测试的各次执行彼此无关；

c) 发生的每个失效都能发现；

d) 测试事例数 $n \gg N$；

e) 在 n 个测试事例期间不发生失效；

f) 每执行一次测试将测试一个程序属性(一个程序属性是在一次执行中可测试的一个属性)。

D.2.4.2 结果

测试所有程序属性的概率 P 由下式给出：

$$p = \sum_{j=0}^{n=1} (-1)^j \binom{N}{j} \left(\frac{N-j}{N}\right)^n \quad \text{或者} \quad p = 1 + \sum_{j=1}^{N} (-1)^j C_{j,N} \left(\frac{N-j}{N}\right)^n$$

其中：$C_{j,N} = \dfrac{N(N-1)\cdots(N-j+1)}{j!}$

在评价本公式时，通常只有第一项才重要，因为实际中的实例数 $n \gg N$。这使所有 j 大的那些项都小。在表 D.5 中也可看到这个结果。

D.2.4.3 例子

考虑一个已在几台设备中使用多年的程序。总计至少已执行了 7.5×10^6 次运行。估计有 1/100 的运行满足上述先决条件。所以已完成的 7.5×10^4 次运行能进行统计评价。估计一次全数测试将执行 4 000 次测试运行。这种估计是保守的。根据表 D.5，所有程序属性不被测试的概率等于 2.87×10^{-5}。

当 $N=4\ 000$ 时，与 n 有关的第一项的值是：

表 D.5 测试所有程序属性的概率

n	P
5×10^4	$1-1.49\times10^{-2}+1.10\times10^{-4}-\cdots$
7.5×10^4	$1-2.87\times10^{-5}+4\times10^{-10}-\cdots$
1×10^5	$1-5.54\times10^{-8}+1.52\times10^{-15}-\cdots$
2×10^5	$1-7.67\times10^{-19}+2.9\times10^{-37}-\cdots$

实际上，所作的这种估计是保守的。

D.3 参考文献

在下列文献中可以看到有关上述技术的更多信息：

IEC 61164:2004, Reliability growth—Statistical test and estimation methods

Verification and Validation of Real-Time Software, Chapter 5. W. J. Quirk (ed.). Springer Verlag, 1985, ISBN 3-540-15102-8

Combining Probabilistic and Deterministic Verification Efforts. W. D. Ehrenberger, SAFECOMP 92, Pergamon Press, ISBN 0-08-041893-7

Ingenieurstatistik. Heinhold/Gaede, Oldenburg, 1972, ISBN 3-486-31743-1

IEEE 352:1987, IEEE Guide for general principles of reliability analysis of nuclear power generating station safety systems

附 录 E
（资料性附录）
专用集成电路（ASIC）设计技术和措施概述

注：GB/T 20438.2—2017 引用了这个附录所包含的技术和措施概述。本附录既不能被认为是完整的也不能认为是详尽的。

E.1 用（V）HDL 对设计描述

目的：硬件描述语言的概要性的功能描述，例如：VHDL 或 Verilog。

描述：硬件描述语言的高度抽象级别的功能描述，如 VHDL 或 Verilog。所用的硬件描述语言应该允许面向功能和/或面向应用的描述，并且应该是从后期实施细节中抽象出来的。应通过硬件描述语言的运算符和赋值来实现数据流，分支，算术和/或逻辑运算，而不需要手动将其转换为所使用库的逻辑门。

注：为便于简化，“硬件描述语言的高度抽象级别的功能描述”在文档的其余部分用（V）HDL 来表示。

参考文献：

IEEE VHDL，Verilog + Standard VHDL Design guide

E.2 原理图输入

目的：使用供应商库提供的门和/或宏来绘图的电路图的功能描述。

描述：通过原理图输入的电路功能的描述。要实现的功能应该通过实例化（导入）基本的逻辑电路元素，如与、或、非，结合由复杂算术和逻辑功能组合成的宏，然后互连。复杂的电路应依据功能进行分区，可以分布在不同的图纸上，并分层互连。互连的信号应该是唯一定义的，在整个层次有明确的信号名称。只要适用，应尽量避免使用全局信号（标签 Lables）。

E.3 结构化描述

注：另见 C.2.7“结构化编程”和 E.12“模块化”。

目的：电路功能的描述应结构化使之易读，即电路功能的描述不需仿真即可直观理解。

描述：使用（V）HDL 或原理图输入实现的电路功能的描述。推荐使用容易辨认和模块化结构。每个模块应该以同样的方式来实施，并应以通过明确定义子功能使之易读的方式来描述。推荐严格区分实现的功能和他们之间相互的连接，即模块。模块是通过实例化其他子模块来实现的，包含明确的实例模块的互连而不应该包含任何逻辑电路。

E.4 经使用证明的工具

目的：应用经使用证明工具来避免系统性失效，这些工具已通过在不同的项目中经过充分长期实践的验证。

描述：大部分用于 ASIC 和 FPGA 设计的工具包括复杂的软件，这些软件不能被视为完全遵循其正确功能而没有任何故障，而且很有可能由于错误的操作而导致故障发生。因此，在 ASIC 和 FPGA 的设计中应首选具有“经使用证明”特征的工具。这意味着：

——应用经长期或经大量用户在具有相同复杂度的不同项目中使用(在一个类似的软件版本)的工具。

——每个 ASIC/FPGA 设计者长期操作工具形成足够多的经验。

——使用常用工具(足够数量的用户),从而有可用的关于已知失效和变通方法(带"缺陷列表"的版本控制)。这些信息应该很容易地整合到设计流程和帮助文件中,以避免系统性失效。

——针对内部工具数据库的一致性检查和有效性检查避免出现错误输出数据。标准工具检查内部数据库的一致性,例如综合工具和布置与走线工具之间的数据库一致性,以运行唯一的数据。

注:一致性检查是所用工具的固有属性,并且设计者对其影响有限。因此,如果提供了手动一致性检查的可能性,设计者宜充分地使用它。

E.5 (V)HDL 仿真

注:见 E.6"模块级功能测试"。

目的:通过仿真的方式对(V)HDL 描述的电路进行的功能验证。

描述:通过仿真整个电路或每个子模块从而验证功能。(V)HDL 仿真器检测由内部电路状态的变化所造成的,作为输入激励的结果的,一个输出序列。通过可向前追溯的输出信号序列("波形图")或一个在设计中设置的称为试验台(test bench)的特殊环境,实施检测到的输出序列的验证。所选的仿真器应该有"经使用证明"的特性,以提供正确的结果,并屏蔽可能由仿真器本身或建模错误引起的信号故障时序(尖峰,三态追踪)的行为。

E.6 模块级功能测试

注:见 E.5"(V)HDL 仿真"和 E.13"验证场景的覆盖率"。

目的:"自下而上"的功能验证。

描述:在模块级验证已实现的功能,例如,通过仿真。被测模块将在一个称为"试验台"的典型的虚拟测试环境中实例化,并由在代码中包含的测试模板来激励。对规定的功能,至少有一个足够高的覆盖率,且测试用例应包含可能存在的所有特殊情形。相对于对输出信号进行手动检查,应优先考虑使用"试验台"的代码来自动验证输出序列。

E.7 顶层功能测试

注:见 E.8"嵌入在系统环境的功能测试"。

目的:ASIC 的验证(整个电路)。

描述:测试的目的是验证整个电路(ASIC)。

E.8 嵌入在系统环境的功能测试

注:见 E.7"顶层功能测试"。

目的:嵌入在系统环境中指定功能的验证。

描述:此测试将验证电路(ASIC)在其系统环境中的整体功能,例如在包含位于电路板或其他地方的所有其他组件的系统环境中测试。建议通过对电路板上所有相关组件建模并与所创建的模型一起对 ASIC 进行仿真,以验证正确的功能,包括时序行为。一个完整的功能测试还包括那些只有在失效存在时才激活的模块的测试。

E.9 异步结构的使用限制

目的:避免综合过程中的典型的时序问题;避免在仿真和综合过程中由不足的建模导致的歧义;可测性设计。

描述:源于组合逻辑的诸如 SET 和 RESET 信号的异步结构在合成和生成电路的过程中容易受尖峰或时序反转的影响,因此应尽量避免。而且不足的建模可能无法被综合工具适当地解释,这会导致仿真过程中不确定的结果。此外,异步结构是低可测的,或者是完全不可测的,所以大大地降低生产测试和在线测试的测试覆盖率。因此建议数量有限的时钟信号的完全同步设计的实施。在多相时钟的系统中,所有的时钟应该来源于同一个主时钟。时序逻辑的时钟输入应始终使用不包含任何组合逻辑的时钟信号。时序单元的异步 SET 和 RESET 输入应始终使用不包含任何组合逻辑的同步信号。主 SET 和 RESET 都应使用两个触发器做同步。

E.10 主输入的同步和亚稳态的控制

目的:避免建立和保持时序违例导致的不明确的电路行为。

描述:从外围设备输入的信号通常是异步的,并可以随意改变其状态。使用 ASIC/FPGA 的同步时序电路的单元对这些信号的直接处理,例如触发器,会产生建立和保持时间违例,从而导致 ASIC/FPGA 不可预测的时序和功能行为。最终可能会发生记忆元件的亚稳态性。因此每个异步输入信号应与同步 ASIC 电路进行同步,以避免功能歧义。建议采取以下措施:

——应该用两个连续的记忆元件(触发器)或一些等效的电路来同步输入信号,以实现一个可预见的功能行为。

——每个异步输入信号都应经过上述定义的方式进行基础的同步,即每个异步信号都通过这样一个同步电路接入。如果有必要,同步电路的输出可用于多路访问。

——同步电路应用于并行总线信号的稳定性测试,并控制采样点附近的数据一致性。

E.11 可测性设计

注 1:见 E.31“测试结构的实现”。

目的:避免不可测或低可测的结构,以实现生产测试或在线测试的高覆盖率。

描述:可测性设计主要避免包括:

——异步结构;

——锁存器和片上的三态信号;

——线与/线或逻辑和冗余逻辑。

子电路组合的深度,在测试过程中起着重要的作用。一个完整的测试所要求的测试模板会随着电路组合深度的增加而指数增加。因此,即使用足够的手段,高组合深度的电路也只能是低可测的。

一个面向可测性设计的方法可确保达到所需的测试覆盖率。由于实际的测试覆盖率在设计过程的非常晚的阶段才确定下来,因此对于“可测性设计”考虑不足的问题可能大幅降低可实现的测试覆盖率,从而导致更多的工作量。

注 2:测试覆盖率通常是由检测的固定型(stuck-at)故障的百分比决定的。

E.12 模块化

注:见 C.2.8“信息的隐藏/封装”,C.2.9“模块化方法”和 E.3“结构化描述”。

目的:电路功能的模块化描述。

描述:把整体功能清晰的划分到具有有限功能的不同的模块中。从而建立了带有精确定义接口的模块的透明性。在所有设计层次的每个子系统,被明确定义并被限制大小(只有少数的功能)。子系统之间的接口要尽可能的简单,使得界面(即共享的数据,信息的交换)最少。各个子系统的复杂性也受到限制。

E.13 验证场景的覆盖率(测试台)

目的:对功能性测试期间所应用的验证场景进行定量及定性评估。

描述:在功能性测试期间定义的验证场景质量,即所使用的测试模板(激励(stimuli))以验证特定功能,包括所有特殊情况,如果存在的话,应该对其建立定性和/或定量文档。用定量方法达成的测试覆盖和所用的功能性测试深度均应建立文档。结果覆盖应符合等级所要求的覆盖率。任何例外均应建立文档。在定性方法方面,应评估被验证电路代码的被验证代码行、指令、路径("代码覆盖")数量。

注:由于硬件描述的高度并行性,专用的"代码覆盖"分析的实际意义不大,并且需要通过彻底的检查证明。"代码覆盖"一般用于证明未覆盖的功能代码。

E.14 遵循编码指南

目的:严格遵循编码类型可导致语法和语义正确的电路代码。

描述:语法编码规则有助于建立易读代码并允许更好的文档,包括版本控制。典型地,组织和注释指令块或模块的规则可在此处提及。

语义编码规则有助于避免构建含有歧义的电路功能实现所导致的缺陷,以防止典型的实施问题。例如,典型的规则是避免异步结构或产生不可预知时序的结构。通常,使用锁存器或将数据与时钟信号耦合会导致这种含混。

设计导则被推荐以避免在 ASIC 开发过程中的系统性设计的失误失效。特定形式的编码类型会限制设计效率,但其优越性是在 ASIC 设计过程中可避免错误。以下是特别应注意的:

——避免典型的代码编码脆弱或失效;

——限制使用可能产生问题的结构,这些结构可产生不确定的结果;

——可测试设计;

——透明化易用代码。

编码类型实例:

1. 代码应包含尽可能多必要的注释以便理解功能和实现细节。在设计之前必须定义惯用的约定。在设计阶段应该检查所定义的需遵循的约定。
   1.1. 标准首部,包括历史、对规格书的交叉引用、职责和随设计产生的数据(如版本号、更改要求等)。
   1.2. 易读的模板:等价的过程应使用同样的程序描述,即使用对反复出现的过程(if-then-else,for 等)的预定义模板。
   1.3. 明确可读的命名约定,即大写/小写字母、前缀和后缀、端口名称之间的明确区别、内部信号、常量、变量、低电平有效(×××_n)等。
   1.4. 模块大小约束和每个模块的端口数量限制,以加强代码可读性。
   1.5. 结构化的和防御性代码开发,即状态信息应封装在 FSM 中(信息隐藏)以提供容易的代码改变。
   1.6. 应实施真实性检查,如范围检查等。

1.7. 避免下述结构/指令：

——对总线信号使用升序范围(x 到 y)；

——Verilog 中的“Disable”指令(相当于 goto 指令)；

——多维数组(>2)，记录；

——有符号和无符号数据类型的组合。

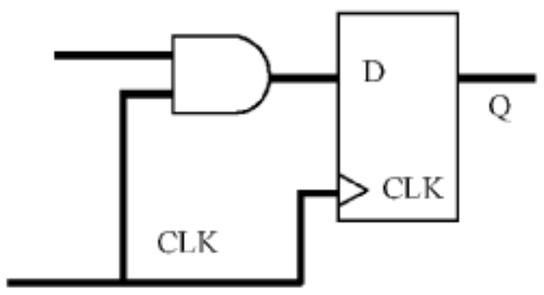
2. 全同步设计(时钟允许取自主时钟)

2.1. 模块输出应同步，还应支持可测试和静态时序分析。

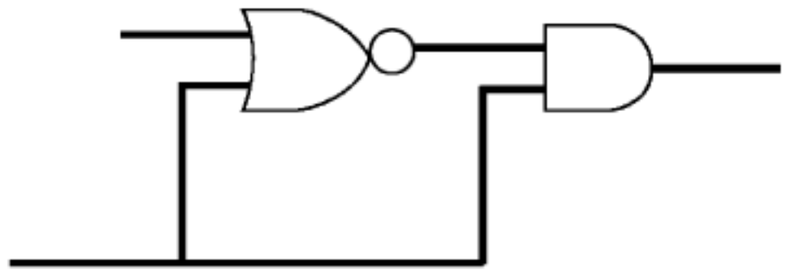
2.2. 门控时钟的处理应带有特别预防措施。

3. 避免数据与时钟的耦合以增加可测性、布局前后数据的复现性和与 RTL(寄存器传输级)行为的符合性。

4.冗余逻辑是不可测的，应该避免：



数据与时钟信号的耦合



冗余逻辑

5. 组合逻辑中的反馈回路应避免，由于这会产生不稳定的设计且不可测。

6. 推荐全扫描(Full-scan)设计。

7. 避免锁存器以增加可测性，并降低合成时的时序约束。

8. 主复位和所有异步输入应使用两个连续的记忆元件(触发器)或一个等效电路(亚稳态)进行同步。

9. 推荐避免异步置位/复位，除非是主复位。

10. 模块端口上的信号电平应是类型“标准逻辑位(std_logic)”或“标准逻辑矢量(std_logic_vector)”。

E.15 代码检查器的应用

目的：由代码检查工具进行编码规则(“编码类型”)的自动验证。

描述：代码检查工具的使用有助于大范围地进行编码类型的自动遵循并生成在线文档。然而，自动代码检查器一般只验证代码的语法和语义，因此这类工具的使用应结合通用编码规则(“工具特定的”)以及由设计者分别进行实现和评估的项目特定的编码规则。

E.16 防御性编程

见 C.2.5。

E.17 仿真结果文档

目的：为成功仿真所需全部数据建立文档，以验证指定电路的功能。

描述：在模块、芯片或系统层次的功能仿真所需的全部数据应妥善建立文档，以便：

——在“交钥匙”方式中，在以后任何时期重复进行仿真。

——证明全部功能规范的正确性和完整性。

为此目的，下列数据库应存档：

——仿真环境,包括所用工具的全部软件,如仿真器、合成器及其相应版本以及所需的仿真库。
——仿真日志文件,包括完整详细的与仿真有关的时间、所使用的工具及其版本和完整的仿真期间必要的工作报告。
——全部有关的仿真结果,包括信号流、尤其是手动检查和已获得结果的文档。

E.18 代码审查

注 1:亦见 C.5.14“形式化审查”。

目的:复审电路描述。

描述:复审电路描述应通过以下方法完成:

——检查编码类型。
——比对规范书验证所描述的功能性。
——检查防御性编码、错误和异常处理。

注 2:如果未实施(V)HDL 仿真,则代码审查的完整性及达成的结果宜达到采用(V)HDL 仿真所达成的质量。

E.19 走查

注 1:亦见 C.5.15“走查”。

目的:通过走查方法复审电路描述。

描述:代码走查包含一个走查团队选择的一组小型测试用例,该程序的一组典型的输入及其相应的期望输出。测试数据按照程序逻辑进行人工跟踪。

注 2:作为独立的措施,该方法宜仅应用于非常低复杂度的电路。如果未实施(V)HDL 仿真,走查的完整性及达成的结果宜与用(V)HDL 仿真所达成的结果具有同样的质量。

参考文献:

IEC 61160:2005,Design review

E.20 应用经确认的软内核

目的:通过应用经确认的软内核避免软内核在运行期间的失效。

描述:如果厂家确认软内核,则应满足以下要求:

——应完成用于安全相关系统操作的软内核确认,要求具有至少等同于或高于系统所计划的安全完整性等级。
——应遵循对软内核确认所必要的所有假定和约束。
——用于软内核确认的所有必要的文档应方便可用,亦见 E.17“仿真结果文档”。
——每个厂家规范均应得到严格遵守并具有文档化的证据。

E.21 软内核的确认

注:亦见 E.6“模块级功能测试”。

目的:通过在设计生命周期中的软内核的确认避免软内核运行失效。

描述:如果软内核不是明确地为运行于安全相关系统而开发的,则生成代码应在用于任何源代码确认的同等前提下得到确认。这意味着所有可能的测试用例均应定义并实施。此后应通过仿真进行功能验证。

E.22 门级网表仿真以检查时序约束

目的:在合成期间对时序约束的独立验证。

描述:由合成产生的门级网表的仿真,包括线路延迟和门延迟的回编(backannotation)。应产生激励以按这样的方式仿真该电路,从而可覆盖大部分时序约束并包括所有最不利时间路径情况。一般地,如果在功能测试期间可得到充分的测试覆盖,用于实施 E.6 或 E.7 的激励可作为选择激励的合适的依据。电路应在最高的规定时钟频率条件下,按最好情况和最差情况测试。

时序验证可通过自动检查目标库记忆元件(触发器)的建立及保持时间,以及通过电路的功能验证完成。功能验证应主要通过观察芯片输出执行,这可通过比较电路的输出信号和一个适当的参考模型或电路的(V)HDL 源代码实现自动操作。此测试被认为是"回归测试",与手工测试相比,应作为输出信号测试的首选。

注:通过使用此措施,仅那些在仿真期间得到实际激励的路径的时间行为得到了验证,因此一般来说,该措施不能够提供完整的电路时序分析。

E.23 传播延迟的静态分析(STA)

目的:在合成期间实现的时序约束独立验证。

描述:静态时间分析(STA)分析网表(电路)的,由合成工具考虑回编(back-annotation)(即由合成工具预估的尚未执行的实际仿真的线路延迟,以及门延迟)产生的全部路径。因此一般来说,这可以得到全部电路的时序约束的完整分析。被测电路应在最高的规定时钟频率条件下,按最好情况和最差情况,并将所用时钟的抖动和占空比偏移考虑在内进行分析。非相关时间路径的数量可通过采用适当的设计技术限制在某一最低限度。推荐在开始设计之前调查、分析和规定可产生易读结果的惯用技术。

注:可假定 STA 覆盖了全部已有的时间路径,如果:

a) 时序约束已正确定义。

b) 被测电路仅包含可被 STA 工具分析的时间路径,即全同步电路的情况。

E.24 门级网表(gate netlist)对比仿真参考模型的验证

目的:合成门级网表的功能对等检查。

描述:对由合成工具产生的门级网表进行仿真。该仿真所使用的电路验证激励与在 E.6"模块级功能测试"和 E.7"顶层功能测试"期间分别对模块级和最高级进行功能验证所使用的激励完全相同。功能验证应主要通过观察芯片输出实现。这可通过比较电路输出信号与一个合适的参考模型或电路的(V)HDL 源代码自动完成。此测试被认为是"回归测试",与手工测试相比,应作为输出信号测试的首选。

注:通过使用此措施,仅在仿真期间被实际激励过的那些路径得到了功能行为的验证,因此,其测试覆盖仅等同于在原来的功能测试期间分别对模块级或最高级进行功能测试的覆盖范围。该措施有可能通过形式化等价测试补充。在所有情况下,(V)HDL 源代码的功能测试宜用合成工具生成的最终网表执行。

E.25 门级网表与参考模型的比较(形式化等价测试)

目的:独立于仿真的功能等价检查。

描述:将(V)HDL 源代码描述的电路功能与合成的门级网表电路功能进行比较。基于形式等价原理的工具能够验证不同表示形式(如(V)HDL 或网表描述)的电路之间的功能一致性。如果使用这种

策略,就不需要进行功能仿真而只需进行独立功能检查。要成功应用这种措施,必需满足下述条件:仅当所使用的工具能被证明是完全等价的,且不论是手工或自动审查的描述是相同的,才能保证成功的应用了这些措施。

注:可以将该措施与 E.24"使用仿真对比验证门级网表与参考模型"组合使用。在所有情况下,宜使用由合成工具生成的最终网表来进行(V)HDL 源代码的功能验证。

E.26 检查供货商的要求和约束

目的:依靠检查供货商的要求来避免生产过程中发生失效。

描述:仔细检查供货商的要求和约束能够提升产品可靠性。例如,检查最小及最大扇入、扇出,最大线长(线延迟),信号的最大转换速率等。如果违反规定,除了对生产过程的需求有影响外,还会对用于仿真的模型的合理性造成很大影响。违反供货商的要求与限制会导致错误的仿真结果从而生成不想要的功能。

E.27 约束、结果与工具的合成文档

目的:所有规定约束的文档,这些约束对于用以生成最终门级网表的最优合成来说是必需的。

描述:合成约束与结果的文档是必需的,原因如下:

——在任何后续阶段再现合成;

——生成独立的合成结果用于验证。

必需的文档包括:

——合成环境,包括所用的工具和带有实际版本的合成软件,合成库和已定义的约束与脚本。

——带有时间戳的合成日志文件,带有版本的所用的工具以及完整的合成文档。

——带有估计时间延迟的生成的网表(标准延迟格式(SDF)文件)。

E.28 应用经使用证明的合成工具

目的:基于工具将电路的(V)HDL 描述转化为门级网表。

描述:依靠连接适当的门和目标 ASIC 库电路原语,实现基于工具的电路功能(V)HDL 源代码映射。依靠合成限制(如时间(时钟频率)和芯片面积)中推导出的最优结果来从不同的可能实现中选取一个满足功能的实现。

E.29 应用经使用证明的目标库

注:参见 E.4"经使用证明工具"。

目的:避免由故障目标库所引起的系统失效。

描述:用于开发 ASIC 的合成与仿真目标库是由公共数据库导出的,因此不具有独立性。系统性失效的典型例子包括:

——电路元件的真实行为与建模行为之间的混淆;

——不充分的建模,如建立与保持时间。

因此,只有经使用证明技术以及目标库才能被用来设计执行安全功能的 ASIC。这意味着:

——在项目中已被使用了很长时间且具有类似复杂度和时钟频率的目标库;

——在足够长时间周期内技术与相应目标库具有可用性,以便可对库进行足够准确的建模。

E.30 基于脚本的方法

目的:结果可重复性与合成周期的自动化。

描述:合成周期的自动化与基于脚本的控制,包括应用约束的定义。除了完整的合成约束的准确文档之外,还能够有助于在相同条件下变更(V)HDL 源代码之后再现网表。

E.31 测试结构的实施

目的:设计可测试的 ASIC,以保证最终产品测试。

描述:可测试性设计能够设计出易于测试的电路,这可通过实现不同的测试结构来实现。例如:

——扫描路径:在扫描技术中,或是全部(全扫描设计)或是触发器(部分扫描设计)被连接在一个单链或是多链以构建一个移位寄存器链。扫描路径允许自动生成电路的整个逻辑的测试模板。生成测试模板的工具被称为 ATPG(自动测试模板生成器)。实现扫描路径能够极大提升电路的可测试能力,在适当努力下能够实现大于 98%的测试覆盖率。因而建议在可能的情况下实现全扫描路径。

——NAND 树:在 NAND 树中,电路的所有主要输入以级联方式连接以构建链。应用适当的测试模板("漫步位(walking bit)")就可以测试输入的切换行为(时序与触发电平)。NAND 树提供直接方式以描述主要输入特性。如果不能以其他方式测试电路的切换行为,建议实现它。

——内置式自测试(BIST):实现片上测试模式生成器就可有效执行电路自测试(特别是嵌入式存储器的自测试)。依靠应用伪随机测试模板并评估实现电路结构的签名,BIST 能够自动验证电路结构。BIST 被建议用作存储器测试的附加方法。扫描路径测试能够被 BIST 代替。

——静电流测试(IDDQ 测试):静态的 CMOS 电路主要在切换事件期间消耗电流。因此,只要测试模板保持不变,一个无缺陷电路消耗小到可以忽略不计量的电流(<1 μA,漏电流)。IDDQ 测试非常有效,在使用少量测试模板之后,它即可提供大于 50%测试覆盖率。IDDQ 测试可应用于功能测试模板以及由 ATPG 生成的合成测试模板。这种测试方法在实践中已被证明非常有效,它能够检测到其他测试很难甚至不能检测到的失效。因此该方法被应用于除常规产品测试之外的附加测试。

——边界扫描:该测试用于验证下述内容:根据 JTAG 标准进行印刷电路板上组件互连。相同的理念可用于验证芯片级模块的互连。边界扫描主要推荐用于提升印刷电路板的可测试能力。

E.32 基于仿真的测试覆盖率估计

目的:在产品测试期间确定执行测试架构已实现的测试覆盖率。

描述:依靠故障仿真可以确定扫描路径测试、BIST、功能测试模板或其他方法所能实现的测试覆盖率。在故障仿真期间,测试模式可被应用于一个被插入故障的电路中。电路对于所应用激励的故障响应对应于所插入的故障,因此与测试覆盖率关联。故障仿真允许检测"固定型(stuck-at)故障"("固定 1"和"固定 0"),所实现的测试覆盖率代表所应用的测试模板的质量。故障仿真可有效检测与逻辑有关的故障,例如对于部分扫描路径来说,对于不属于扫描路径部分的故障这种方法很有效。

E.33 使用 ATPG 工具估计测试覆盖率

目的:在产品测试期间确定合成测试模板(扫描路径,BIST)所能实现的测试覆盖率。

描述:当前有不同的方法用以实现带有扫描路径的电路的伪随机或算法测试模板。在合成期间合成工具(如 ATPG)创建一个未检测到的故障目录。因此可以估计测试覆盖率并定义所应用测试模板的测试覆盖率下限。注意测试覆盖率仅限于被扫描路径覆盖的电路逻辑。存储器模块、BIST 模块或未被集成于扫描路径的电路部分不在测试覆盖率估计考虑的范畴。

E.34 应用硬核经使用证明的评价

目的:在应用硬核期间避免系统失效。

描述:硬核一般被视为代表想实现功能的黑盒,由提供想实现的电路器件的目标技术的版图数据基础组成。处理可能的功能失效的方法与分立器件(如标准微处理器,存储器等)相似。如果对于目标技术来说所使用的核能被视为经使用证明的组件,不验证其功能的正确性也可以使用该硬核。电路的其余部分应被彻底验证。

E.35 应用已确认的硬核

注:参见 E.6“模块级功能测试”。

目的:在应用硬核期间避免系统性失效。

描述:由于核以及假设约束的复杂特性,核的确认应该由供货商在设计阶段基于(V)HDL 源代码执行。该确认仅对已应用的器件的配置和目标技术有效。

E.36 硬核在线测试

注:参见 E.13“验证场景的覆盖率(测试台)”。

目的:在应用硬核期间避免系统性失效。

描述:使用在线测试来验证所用硬核的正确功能及实现。使用这种方法需要有效的测试概念,并且所用概念的评估过程需要建立文档。

E.37 设计规则检查(DRC)

目的:供货商设计规则验证。

描述:基于供货商设计规则验证生成的版图,例如最小线长、最大线长以及若干有关版图结构布置的规则。完整且正确的 DRC 运行需要详细建立文档。

E.38 布局与原理图对比的验证(LVS)

目的:布局的独立验证。

描述:LVS 从布局数据基础提取电路功能,并将所提取的包含互连的电路组件与输入的网表进行比较。这可确保电路布局与规定电路功能的网表之间的等价性。完整且正确的 LVS 运行需要详细建立文档。

E.39 为使用时间小于 3 年的工艺技术附加裕量(>20%)

目的:在生产工艺和参数显著波动的情况下,确保所实现电路功能的鲁棒性。

描述:对于小结构(例如小于 0.5 μm)来说,实际的电路行为是由许多交叠的物理作用定义的。实

际上,由于缺少详细知识及必要的简化,并不能推出电路组件的准确模型。随着几何结构的减小,线延迟越来越起到主导作用。线路上的信号延迟以及线路间交叉耦合能力超过比例关系增长。与门延迟相比信号延迟不再可以忽略。估计的线延迟随着几何结构的减小风险增加。

因此,如果用于设计电路的工艺使用时间小于3年,建议为其最小和最大时间约束留出足够的裕量(>20%),以确保在生产过程中参数显著波动或缺少精确模型情况下电路功能仍能保证正确。

E.40 老化测试

目的:确保制造芯片的鲁棒性。清除早期失效。裸片芯片产品不必借助烧入测试证明其鲁棒性,而是通过其他方式(如晶圆级压力方法)证明。

描述:老化测试应该在最高可容忍温度(125 ℃)下进行。测试时间是由目标安全完整性等级或特定的(例如来自ASIC制造商的)老化建议决定的。老化可用于:

——清除早期失效(浴盆曲线的开始段,失效率逐渐降低的部分);

——证明在制造和测试期间早期失效已被清除(即,经生产线生产出的器件已经处于浴盆曲线的常数失效率区域)。

E.41 应用经使用证明的设备系列

目的:确保所制造芯片的可靠性。

描述:安全设计的制造商应该有充分的所用可编程器件技术和开发工具的应用经验。

E.42 经使用证明的生产工艺

目的:确保所制造芯片的可靠性。

描述:经使用证明生产工艺应具有充分的生产经验。

E.43 生产工艺的质量控制

在器件生产工艺中的质量措施与控制机制确保连续的工艺控制。例如测试结构的光学或电学控制、温湿度偏压测试或温度周期测试(见IEC 60068-2-1,IEC 60068-2-2等)。

E.44 器件制造质量许可

应通过所选择的应力测试,如温湿度测试或温度变化测试(见GB/T 2423.1,GB/T 2423.2等)来证明器件质量。器件制造商应给出证明。

E.45 器件的功能质量许可

所有器件将进行功能测试。器件制造商应给出证明。

E.46 质量标准

ASIC制造商应提供充分的质量管理措施,例如下列文档中规定的内容:质量&可靠性手册:GB/T 19000认证或标准供应商质量评估(SSQA)。

附 录 F
（资料性附录）
软件安全生命周期各阶段属性的定义

表 F.1 软件安全要求规范
（见 GB/T 20438.3—2017 的 7.2 和表 C.1）

	属性	定义
1.1	针对由软件来确保的安全要求的完整性	软件安全要求规范规定了来自于安全生命周期早期阶段并分配给软件的所有安全要求和约束。 安全要求和约束往往作为软件安全要求规范活动的输入进行陈述，包括规定软件必须做什么和必须避免什么
1.2	针对由软件来确保的安全要求的正确性	软件安全要求规范对于分配给软件的安全要求和约束提供了合适的解决方案，目的是确保规定的内容在所有可能的条件下真正保证安全
1.3	没有规范本身的错误，包括：语义含糊	软件安全要求规范内在的完整性和一致性。即，对于从其声明的内容中能够导出的功能和情况提供所有必要的信息；没有相互矛盾的表达，没有不相容的陈述。 与安全要求的完整性和一致性不同的是，内在的完整性和一致性仅可基于软件安全要求规范自身进行评估
1.4	安全要求的可理解性	在假设阅读者具有要求知识的前提下，所有需要阅读软件安全要求规范的人，即使在早期没有介入项目的情况下，能够不太费力的就能理解。一个重要的目的是便于验证和可能的修改
1.5	针对由软件来确保的安全要求，没有非安全功能的不利干扰	软件安全要求规范中没有对于 EUC 安全无关的要求，目的是避免软件的设计和实施中不必要的复杂性，进而减少故障的风险和安全无关的功能干扰或损害安全功能的风险
1.6	为验证和确认提供基础的能力	软件安全要求规范能够导出测试和检查。该测试能产生直接证据，以证明软件满足软件安全要求规范

表 F.2 软件设计和开发：软件架构设计
（见 GB/T 20438.3—2017 的 7.4.3 和表 C.2）

	属性	定义
2.1	针对软件安全要求规范的完整性	软件架构设计涉及到了由软件安全要求规范提出的所有安全要求和约束
2.2	针对软件安全要求规范的正确性	软件架构设计为特定的软件安全要求提供了适合的解决方法
2.3	没有本身的设计错误	软件架构设计和设计文档无故障。这里所说的故障是指那些与任何特定的软件安全要求无关的故障。 例如：死锁，访问非授权的资源，资源泄露，内部的不完整性（即，无法处理由结构设计本身导出的所有情况）

表 F.2（续）

	属性	定义
2.4	简单性和易懂性 行为的可预测性	软件架构设计允许在所有特定情况下正确和准确的预测软件的运行状态。 尤其是在包含错误和失效的情况下。 可预测性尤其意味着软件运行不依赖于不能被设计者和使用者控制的项目
2.5	可验证和可测试的设计	软件架构设计和设计文档应允许和促进产生可靠的证据，证明设计已正确考虑到所有特定的软件安全要求并且设计是没有内部故障的。 可验证性可能隐含的引入一些特性，比如简单性、模块化、清晰度、可测试性、可证明性等，取决于使用的验证技术
2.6	故障裕度	软件架构设计要保障软件在存在错误（内部错误、操作错误和系统外部错误）时，行为是安全的。 防御性设计可以是主动的或者被动的。主动的防御式设计包括相同特征检测、报告和包容错误、适度降级，以及恢复正常操作前清除一切不良影响。被动式防御包括的特点是，对于那些软件不应执行任何特定操作的特定类型错误或特定条件（雪崩输入、特殊的数据和时间），保证软件不受其影响
2.7	抵御外部事件造成的共因失效	软件架构设计应易于识别共因失效模式，并有效防范失效

表 F.3 软件设计和开发：支持工具和编程语言

（见 GB/T 20438.3—2017 的 7.4.4 和表 C.3）

	属性	定义
3.1	工具对具有所需软件属性的软件生成的支持程度	意味着提供错误检测，或者消除易发生错误的结构
3.2	工具的操作和功能的清晰度	全面覆盖的条款和工具操作所有方面的反馈
3.3	输出的正确性和可重复性	对任何给定的输入，工具输出的一致性和正确性

表 F.4 软件设计和开发：详细设计

（见 GB/T 20438.3—2017 的 7.4.5、7.4.6 和表 C.4）

	属性	定义
4.1	针对软件安全要求规范的完整性	采用详细的软件设计和开发方法，以确保产生的软件达到分配给软件的所有安全要求和约束
4.2	针对软件安全要求规范的正确性	有特定证据可声明，所开发的软件达到了分配给软件的安全要求
4.3	避免内部设计故障	所开发的软件没有内部故障 如：死锁、访问非期望的资源、资源泄露
4.4	简单、易懂 行为可预测	通过客观的、有说服力的测试和分析，说明所开发软件的行为是可预测的
4.5	可验证和可测试的设计	开发的软件是可验证和可测试的
4.6	故障裕度和故障检测	保障开发的软件在存在错误情况下行为安全的技术和设计
4.7	避免共因失效	识别共因失效模式，并有效防范失效的技术和设计

表 F.5　软件设计和开发:软件模块测试和集成
(见 GB/T 20438.3—2017 中 7.4.7、7.4.8 和表 C.5)

	属性	定义
5.1	对照设计规范,测试和集成的完整性	软件测试充分彻底地检查软件行为,确保软件符合软件设计规范的要求
5.2	对照设计规范,测试和集成的正确性(成功完成)	完成模块测试任务,有具体的证据可声明安全要求得到满足
5.3	可重复性	重复每一次评估得到一致的结果。执行评估是模块测试和集成的一部分
5.4	清晰定义的测试配置	已经使用正确版本的组件和软件完成了模块测试和集成,具有已声明的结果,并允许该结果与已完成软件的具体配置连接

表 F.6　可编程电子集成(硬件和软件)
(见 GB/T 20438.3—2017 的 7.5 和表 C.6)

	属性	定义
6.1	对照设计规范,集成的完整性	集成提供了系统组件适当的深度和覆盖面,确保系统在可预见的操作条件和系统失效条件下,执行预定功能,而不执行非预定功能。 这包括用于验证的原理、设计以及集成方面的目标等级(例如模块间相互作用的完整性验证)
6.2	对照设计规范,集成的正确性(成功完成)	集成基于正确的假设,例如,预期结果的正确性、所考虑的使用条件的正确性以及测试环境的代表性等。 集成任务完成后,有具体的证据可声明安全要求得到满足
6.3	可重复性	重复每一次评估得到一致的结果,执行评估是软件集成的一部分
6.4	清晰定义的集成配置	保证使用正确版本的组件和软件,按照文档规定有效实施了集成。具有已声明的结果,并允许该结果与已完成软件的具体配置连接

表 F.7　系统安全确认的软件方面
(见 GB/T 20438.3—2017 的 7.7 和表 C.7)

	属性	定义
7.1	对照软件设计规范,确认的完整性	软件确认涉及软件设计规范的所有要求
7.2	对照软件设计规范,确认的正确性(成功完成)	软件确认任务完成后,有具体的证据可声明安全要求得到满足
7.3	可重复性	重复每一次评估得到一致的结果,执行评估是软件确认的一部分
7.4	清晰定义的确认配置	清晰而简明的定义系统、要求和环境

表 F.8　软件修改

（见 GB/T 20438.3—2017 的 7.8 和表 C.8）

	属性	定义
8.1	对照修改要求，修改的完整性	对于修改的功能、安全、技术和操作后果有正确理解，修改按照规定的程序得到授权人员的许可
8.2	对照修改要求，修改的正确性	修改达到其规定的目标
8.3	避免引入固有的设计错误	修改不会引入新的系统性失效，如被 0 除，索引或指针越界，使用未初始化变量等
8.4	避免不必要行为	依据在软件安全要求规范中声明的约束，修改不能引入规范中必须避免的行为
8.5	可验证和可测试的设计	软件设计能够做到，使修改的作用能够得到全面检查
8.6	回归测试和验证覆盖	软件设计能够做到可以进行有效的和全面的回归测试以证明经过修改后的软件仍然满足软件安全要求规范的要求

表 F.9　软件验证

（见 GB/T 20438.3—2017 的 7.9 和表 C.9）

	属性	定义
9.1	相对前一阶段，验证的完整性	验证有能力证明软件满足软件安全要求规范的所有相关要求
9.2	相对前一阶段，验证的正确性（成功完成）	验证任务完成后，有具体的证据可声明安全要求得到满足
9.3	可重复性	重复每一次评估得到一致的结果，执行评估是验证的一部分
9.4	清晰定义的验证配置	已对正确版本的组件和软件实施验证，具有已声明的结果，并允许该结果与已完成软件的具体配置连接

表 F.10　功能安全评估

（见 GB/T 20438.3—2017 的第 8 章和表 C.10）

	属性	定义
10.1	对照标准，功能安全评估的完整性	软件功能安全评估对以下方面作出明确声明：符合规范的程度，作出的判断，补救措施和时间进度的建议，得到的结论和接受、有条件的接受、不接受的建议以及对于这些建议的任何时间限制
10.2	对照设计规范，功能安全评估的正确性（成功完成）	完成软件功能安全评估任务，有具体的证据可声明安全要求得到满足
10.3	可追踪地结束所有确定的问题	明确声明在软件功能安全评估过程中发现的问题已被解决
10.4	无需大量重新评估的改变后修改软件功能安全评估的能力	软件功能安全评估具有软件修改并获得修订结论后，无需重新进行完整的功能安全评估，而只需对修改的那部分进行重新评估的能力

表 F.10（续）

	属性	定义
10.5	可重复性	对于确定的项目和文档，根据一致的、计划好的和开放的程序执行功能安全评估，其可以让包括系统供应商、用户、维护人员和监管人员在内的所有受这个判断影响的各方能详细审查评估和所得判断的基础。 功能安全评估允许独立的有能力的人员重复的执行整个评估的一部分
10.6	时效性	功能安全评估在适当的频率下执行，频率与软件安全生命周期阶段相关，评估至少早于已确定的危险出现之前，并能够及时报告不足。 当要求作为评估决策的输入时，测试、安全监测和分析等的结果是可用的
10.7	清晰定义的配置	软件功能安全评估允许其结果与已得到功能安全评估证实的具体系统配置连接

附 录 G
（资料性附录）
安全相关的面向对象软件的开发指南

GB/T 20438 的所有建议适用于面向对象软件的软件设计。由于面向对象的方法提供了不同于过程和功能方法的信息，下面所列包含了那些需要具体考虑的建议：

——理解类的层次结构，并识别以给定方法调用执行的软件功能（包括何时使用现有的类库）；

——基于结构的测试（GB/T 20438.3—2017 的表 B.2 和 GB/T 20438.7—2017 的 C.5.8）。

表 G.1 和表 G.2 针对面向对象软件的使用提供了资料性的指导，以补充 GB/T 20438.3—2017 表 A.2 和表 A.4 提供的更通用的规范性指导。

表 G.1 面向对象的软件架构

	建议	详情	SIL1	SIL2	SIL3	SIL4
G1.1	从应用领域概念到架构的类的可追溯性	注 1	R	HR	HR	HR
G1.2	使用合适的框架，常用的组合类和设计模版 **注**：当使用现有的框架和设计模版时，对于已开发软件的要求适用于这些框架和模版。	注 2	R	HR	HR	HR

注 1：从应用领域到类的架构的可追溯性是次要的。

注 2：例一：对于一个想要作为安全相关项目的部分，可能已经存在一个非安全相关项目的框架，该框架已成功地解决了类似的任务且为项目参与人员所熟知。则建议使用该框架。

例二：解决安全相关项目密切相关的子任务时可能需要不同的算法。对于访问算法，可以选择策略模式。

例三：安全相关项目的部分内容，可能包括对内部和外部的相关人员发出的相应的警告。可以选择观察者模式来组织这些警告。这个要求不适用于库。

注 3：它通常是一个抽象的基本类，其对派生的具体类提供访问。

表 G.2 面向对象的详细设计

	建议	SIL1	SIL2	SIL3	SIL4
G2.1	类应该只有一个对象	R	R	HR	HR
G2.2	仅当派生类是其基本类的细化时，才能使用继承	HR	HR	HR	HR
G2.3	继承的深度由编码标准限制	R	R	HR	HR
G2.4	严格控制下的操作（方法）重写	R	HR	HR	HR
G2.5	多重继承仅用于接口类	HR	HR	HR	HR
G2.6	从未知类的继承			NR	NR
G2.7	要验证面向对象的库的再使用是否满足表中的建议	HR	HR	HR	HR

注 1：也就是说，一个类由其所承担的责任而具有特征，即，注意数据和数据的操作有密切关系。

注 2：需要注意避免对象之间的循环依赖。

在表 G.3 中非正式定义了上面使用的术语。

表 G.3　一些相关术语

术语	非正式定义
基本类	具有派生类的类。一个基本类有时也被称为上层类或父类
派生类	从另一个类(基本类)继承属性和/或操作的类(属性和操作的集合)。派生类有时称为亚类或子类
框架	程序的结构,在许多情况下预先制定,以便于填入特定的应用
重写	在运行时,在同一署名和继承结构上,用另外一个操作(方法,子程序)更换一个算法(方法,子程序);面向对象的语言和程序的属性;实现多态性
操作签名	操作(子程序,方法)的名称,连同其参数(命令)和类型,有时也包括返回类型。如果具有相同的名称,数量和参数的类型,两个签名是相同的;在某些语言中,返回类型也必须是相同的

参 考 文 献

[1] GB/T 2421.1—2008 电工电子产品环境试验 概述和指南

[2] GB 4208—1993 外壳防护等级(IP 代码)

[3] GB/T 7826—2012 系统可靠性分析技术 失效模式和影响分析(FMEA)程序

[4] IEC 60880:2006 Nuclear power plants—Instrumentation and control systems important to safety—Software aspects for computer-based systems performing category A functions

[5] IEC 61000-4-1:2006 Electromagnetic compatibility(EMC)—Part 4-1:Testing and measurement techniques-Overview of IEC 61000-4 series

[6] GB/T 17626.5—2008 电磁兼容 试验和测量技术 浪涌(冲击)抗扰度试验

[7] IEC/TR 61000-5-2:1997 Electromagnetic compatibility(EMC)—Part 5:Installation and mitigation guidelines—Section 2:Earthing and cabling

[8] IEC 61025:2006 Fault tree analysis(FTA)

[9] GB/T 18272.5—2000 工业过程测量和控制 系统评估中系统特性的评定 第5部分:系统可信性评估

[10] IEC 61078:2006 Analysis techniques for dependability—Reliability block diagram and boolean methods

[11] GB/T 15969.3—2005 可编程序控制器 第3部分:编程语言

[12] IEC 61160:2005 Design review

[13] IEC 61163-1:2006 Reliability stress screening—Part 1:Repairable assemblies manufactured in lots

[14] IEC 61164:2004 Reliability growth—Statistical test and estimation methods

[15] IEC 61165:2006 Application of Markov techniques

[16] IEC 61326-3-1:2008 Electrical equipment for measurement,control and laboratory use—EMC requirements—Part 3-1:Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions(functional safety)—General industrial applications

[17] IEC 61326-3-2:2008 Electrical equipment for measurement,control and laboratory use—EMC requirements—Part 3-2:Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions(functional safety)—Industrial applications with specified electromagnetic environment

[18] IEC 81346-1:2009 Industrial systems,installations and equipment and industrial products—Structuring principles and reference designations—Part 1:Basic rules

[19] GB/T 19898—2005 工业过程测量和控制 应用软件文档集

[20] GB/Z 29638—2013 电气/电子/可编程电子安全相关系统的功能安全 功能安全概念及GB/T 20438系列概况

[21] GB/T 21109(所有部分) 过程工业领域安全仪表系统的功能安全

[22] GB 28526—2012 机械电气安全 安全相关电气、电子和可编程电子控制系统的功能安全

[23] IEC 62308:2006 Equipment reliability—Reliability assessment methods

[24] ISO/IEC 1539-1:2004 Information technology—Programming languages—Fortran—Part 1:Base language

[25] GB/T 1526—1989 信息处理 数据流程图、程序流程图、系统流程图、程序网络图和系统资源图的文件编制符号及约定

[26] ISO/IEC 7185:1990 Information technology—Programming languages—Pascal

[27] ISO/IEC 8631:1989 Information technology—Program constructs and conventions for their representation

[28] ISO/IEC 8652:1995 Information technology—Programming languages—Ada

[29] ISO 8807:1989 Information processing systems—Open Systems Interconnection—LOTOS—A formal description technique based on the temporal ordering of observational behaviour

[30] ISO/IEC 9899:1999 Programming languages—C

[31] ISO/IEC 10206:1991 Information technology—Programming languages—Extended Pascal

[32] ISO/IEC 10514-1:1996 Information technology—Programming languages—Part 1:Modula-2,Base Language

[33] ISO/IEC 10514-3:1998 Information technology—Programming languages—Part 3:Object Oriented Modula-2

[34] ISO/IEC 13817-1:1996 Information technology—Programming languages,their environments and system software interfaces—Vienna Development Method—Specification Language—Part 1:Base language

[35] ISO/IEC 14882:2003 Programming languages—C++

[36] ISO/IEC/TR 15942:2000 Information technology—Programming languages—Guide for the use of the Ada programming language in high integrity systems

[37] GB/T 12668.502 调速电气传动系统 第5-2部分:安全要求 功能

[38] IEC 60601(all parts) Medical electrical equipment

[39] GB/T 2423.1 电工电子产品环境试验 第2部分:试验方法 试验A:低温

[40] GB/T 2423.2 电工电子产品环境试验 第2部分:试验方法 试验B:高温

[41] GB/T 19000 质量管理体系 基础和术语

[42] GB/T 20438.1—2017 电气/电子/可编程电子安全相关系统的功能安全 第1部分:一般要求

[43] GB/T 20438.2—2017 电气/电子/可编程电子安全相关系统的功能安全 第2部分:电气/电子/可编程电子安全相关系统的要求

[44] GB/T 20438.3—2017 电气/电子/可编程电子安全相关系统的功能安全 第3部分:软件要求

[45] GB/T 20438.6—2017 电气/电子/可编程电子安全相关系统的功能安全 第6部分:GB/T 20438.2和GB/T 20438.3的应用指南

索　引

汉语拼音索引

中华人民共和国
国家标准
电气/电子/可编程电子安全相关系统的
功能安全 第7部分:技术和措施概述
GB/T 20438.7—2017/IEC 61508-7:2010
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100029)
北京市西城区三里河北街16号(100045)
网址:www.spc.org.cn
服务热线:400-168-0010
2017年11月第一版
*
书号:155066·1-57855

GB/T 20438.7-2017