安全关键系统的防危性技术研究*

杨仕平** 熊光泽 桑 楠

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】为设计高防危性的安全关键系统，阐述了安全关键系统防危性的本质含义。并从设计方面研究防危性的实现机制，其中重点研究了基于防危核的高防危保障技术。同时提出了基于反射式技术的编程语言——Open C++的实现机制，为防危核的实现探索了新的途径，也为安全关键系统探索了有效的防危新机制。

关 键 词 安全关键; 防危性; 防危核; 反射; 可信性

中图分类号 TP302.8 文献标识码 A

Research on Safety Technology of Safety Critical Systems

Yang Shiping Xiong Guangze Sang Nan

(Shool of Computer Science and Engineering, UEST of China Chengdu 610054)

Abstract In order to design the high safety of safety critical systems, in the paper, the nature of safety of safety critical systems was narrated in detail, established good theory foundation for designing high dependability safety critical systems. At the same time, realization mechanisms of safety were deeply discussed, and high safety safeguard technology based on safety kernel was emphatically researched. In order to realize safety kernel, a kind of programming language based on reflective theory ——Open C++ was brought forward, exploring a new approach used to realize safety kernel, at the same time, a novel safeguard mechanism based on safety kernel is adaptable for other safety critical systems.

Key words safety critical；safety；safety kernel；reflection；dependability

安全关键系统(Safety Critical Systems，SCS)是指系统功能一旦失效将引起生命、财产的重大损失以及环境可能遭到严重破坏的系统。这类系统广泛存在于航空航天、国防、交通运输、核电能源和医疗卫生等诸多安全关键领域中。随着高新技术在军事领域的日益应用，许多武器装备系统越来越复杂，错误也越难检测和避免，安全隐患自然也越严重。错误的主要来源也逐渐从硬件转移到软件上，如许多防御系统，尤其是武器系统，包括传感器、执行器、通信、决策控制器等。这些系统通常具有多台相连的计算机，用于这些系统的软件常常必须保证既能满足时间限制，同时又要保证系统中的某物理环节出现故障或失灵时转换到理想状态(继续运行或降级使用)。当今控制软件的出错概率大于系统零部件的出错概率。因此，软件的可信性已经成为安全关键系统可信性的决定因素，安全关键系统中用于安全关键控制的软件为安全关键软件[1]。软件的可信性包括可用性、可靠性、防危性(Safety，防止危险发生)、机密性、完整性及可维护性。本文讨论软件可信性中的防危性，更能反映安全关键软件在安全关键系统中的本质作用——防止系统发生危险性(或灾难性)事故。同时阐述了防危性的本质含义，其目的是在充分理解防危性的同时能设计出高防危性的系统，还讨论了可提高系统高防危性的最新设计与实现技术。

2002年9月10日收稿

* 总装部预研基金项目，编号：2000J6.7.1.DZ0206

** 男 28岁 博士生 主要从事实时操作系统的防危核机制与实现、安全关键系统方面的研究

1 安全关键系统防危性的本质含义

随着用户需求日趋复杂及安全关键系统高防危性的需要，硬件作为实现传统安全关键系统已面临着巨大的挑战。由于软件具有灵活、无物理损耗等固有特性，用软件来实现安全关键系统防危性相关的部分无疑更具吸引力。安全关键系统其防危性具有以下七种特殊含义。

1.1 防危性是一个系统性问题

Leveson指出防危性不单纯是一个软件问题，而是一个系统性问题。本质上，软件并不直接作任何危险的事，如果软件作为一个子系统而导致整个系统失效，则该软件在特殊环境下可被认为是不安全的。因而在系统级，软件可被看成一个或多个组件，这些组件的失效将诱导系统发生灾难性的事故。

1.2 防危性可用风险来度量

防危性本质上是指系统可以避免对生命、财产及环境造成严重损害。但该定义过于简单、模糊，不能直接用于系统防危性的度量。完全安全的系统在特殊的环境下可能没有任何用处，如没有装入核材料的核电系统(该系统完全安全，但不能发电)。因而系统防危性必与风险性相关，风险可被定义为

$$Risk = \sum_{hazard} e(hazard) \times P(hazard)$$

式中 $e(hazard)$ 表示对重大事故所造成后果的度量(如死亡的人数、经济损失等)，$P(hazard)$表示特殊事故发生的概率。现实生活中不存在绝对安全的系统，只能通过限制事故发生的范围或减少事故发生的概率来降低系统正常运行的风险。

1.3 可靠不一定防危

可靠与防危是两个不同的系统概念，可靠性是对系统持续正常工作能力的度量；防危性则是对系统远离危险条件的度量。即可靠性是指系统如何持续稳定地实现其应有的正常功能；而防危性则是指实现系统功能的同时能避免重大事故的发生。一个系统可靠并不意味着防危，如航空控制系统在某组件失效的情况下仍可在飞行员的控制下安全向前飞行，我们称该系统是可靠的。但如果迎面同时飞来一架飞机，失效情况下的继续飞行将极可能导致危险事故(碰撞)的发生，此时航空系统如果能检测到碰撞即将发生并同时采取避免措施防止事故发生，则我们称该系统具有防危性。反之，具有防危性的系统并不一定具有可靠性，如火车控制系统无论在什么情况下总是采取停车措施，显然该系统具有极高的防危性，但同时失去了系统应有的可靠性(一般故障下应继续前进)。

1.4 防危性必具有机密性

安全关键系统中的安全关键组件必须具有较高的机密性、保证安全关键软件本身及其要使用的数据不会被非授权代理(软件或人员)篡改。如果其软件或数据被篡改，则正在执行的组件将不满足其规定的安全需求，甚至导致灾难性事故。

1.5 防危是过程性技术而非事后性技术

安全关键系统防危性的实施应在系统的整个开发生命周期中进行，即防危措施与系统设计融为一体，而不应在系统“成功”开发后增加额外的保护系统。如通过烟火检测器来实现的消防保护系统会受限于检测系统、保护系统本身的可靠性，与之相反的技术是防火材料的使用及限制或减少易燃物质的使用会取得更好的防危效果。

1.6 软件不应轻易取代硬件

软件的优点是实现灵活、易于修改且接近零重量，软件一旦被成功开发，其复制的代价较低；与软件相反，硬件的复制代价却很高。用具有相同功能的软件去取代相应的硬件可以降低系统的开发成本，但这样做却是非常危险的。硬件由于遵循一定的物理特性，其失效的方式相对于软件而言更易预测，因而硬件也更易取得较高的可靠性。对于软硬件集成的安全关键系统，一般用硬件进行总体控制(如出现紧急情况时，火车司机可使用的机械停车系统)，而用软件则进行更具体的控制(发生轻微故障时应继续运行)。

1.7 安全关键软件的开发工具本身也应该是安全的

应尽量使用已被验证为可靠的开发工具，如自动代码生成软件ObejectGEODE等，这样可减小人为地引

入错误到安全关键软件中。

2 安全关键系统防危性的设计技术

如何建立高防危性的安全关键系统，应从开发的管理和设计两方面着手。本文就设计问题论述了当前最新的、切实可行的防危性设计技术。

安全关键系统的防危性设计从大的方向可分为：优化结构设计和容错设计，优化结构设计其目的是减少软件缺陷，而软件容错设计的目的则是防止一定数量的未知软件失效[2]。优化结构设计与容错设计可进一步细分为：

1) 半形式化方法(Semi-Formal Methods，Semi-FM)、形式化方法 (Formal Methods，FM)与语言[3]；
2) 实时设计技术(如DARTS等)；
3) CASE工具、图形化设计工具及表格化设计工具的使用；
4) 看门狗定时器的使用；
5) *N*版本程序，
6) 恢复块；
7) 防危核。

前三项技术属于优化结构设计，其中FM是基于离散数学和形式逻辑的一种方法，它对设计型故障有一定的避错作用，FM能避免二义性，系统需求分析和各种文档说明的二义性是设计型故障的主要来源；形式逻辑在设计中的应用，使设计和证明按一条逻辑线路进行，较传统方法更完整、周密。然而FM并未得到广泛的应用，其原因是：使用FM的方式是手工进行的，并没有丰富的工具支持，除非使用FM的人具有较高的素质。而半形式化方法接近自然语言，较纯粹的FM易于使用，目前业界已证明可行的方法是同时使用Semi-FM和FM。基于FM的CASE工具，如ObjectGEODE集形式化方法设计与代码生成于一体，其生成的安全关键软件具有较高的防危性。而另一工具软件是Rational公司的RT-ROSE，它在形式化设计方面也能取得很好的效果。

后四项属于容错设计，对安全关键系统而言，有时很难分清故障是由软件引起的还是由硬件引起的。故障发生时，系统对安全关键软件容错的预期目标主要有：

1) 系统仍能完全实现规定的功能；
2) 降级使用；
3) 使系统转入预先规定的安全状态，防止造成更大的损失。

要完成这些功能，需要综合运用检错、*N*版本程序、恢复块、分离与保护、防危核等容错技术。本文将对*N*版本程序与防危核技术作详细论述。

2.1 *N*版本程序

*N*版本程序基本思想是不同的设计人员对同一需求说明的理解未必犯相同的错误，不同的设计技术编写的程序，其错误是相互独立的。软件运行时，将*N*个版本的软件至于相同的条件下，同时投入运行，并将它们的输出结果进行表决处理以决定最终输出，从而屏蔽错误，达到容错的目的。*N*版本程序最大的缺陷是对相同失效模式显得无能为力，为克服此缺陷，可将*N*版本程序设计成*N*个功能逐渐升级(或降级)的软件。如宇宙飞船中的控制软件可设计为4个功能逐渐增强的版本：版本1主要避免与附近物体及空间站发生碰撞；版本2主要避免碰撞的同时接近空间站；版本3主要避免碰撞的同时接近空间站，然后绕空间站运行；版本4完成版本3功能的同时减少燃油量。实际应用时，4个版本应同时运行，其中版本4应作为主程序，检测到有险情时，应降级到版本3上运行，如再检测到险情时，则应继续降级到版本2甚至版本1上运行。

2.2 防危核技术

为提高安全关键系统的防危性，J. Rushby等人提出了防危核(Safety Kernel)的设计思想[4]，它是一个应用级的核，防危核在安全关键系统中的地位与作用如图1所示。导致安全关键系统发生灾难性事故的系统错误可分为消极错误与积极错误两种，其中消极错误是指系统做了不该或不允许发生的事，如自动飞行控制软件在飞机处于飞行状态时，启动关闭发动机装置；而积极错误则是指系统没有完成规定要做的事，如飞机

飞行过程中，检测到迎面有飞行物时，本应绕道飞行，但却继续向前飞行。防危核的主要优势在于防止消极错误的发生，其原因是防止系统做不该做的事比保障系统完成所期望的任务更易于实现和验证。所以在设计高防危性的安全关键系统时，为防止系统发生灾难性事故，直接防止系统做不期望的事显得切实可行有效，这也是防危核的设计思想本质所在。实际使用时防危核利用一组专门定义的防危策略，监控应用软件对系统设备的操作请求，通过应用软件与系统设备的隔离，拒绝其中可能导致生命、财产损失的操作请求，避免由于错误的设备操作请求而引起的设备错误动作，从而保证系统达到所需的防危性要求。

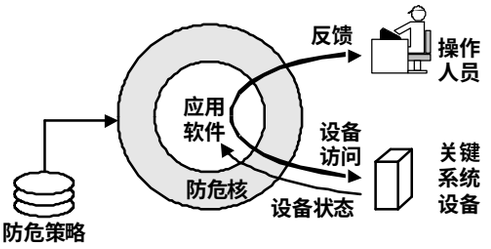

图1　防危核在安全关键系统中的地位与作用

为了便于防危核正确性的检验，防危核应尽可能的小，让系统所有的防危策略都由防危核来实施是不现实的，因此，防危策略的选择是实现防危核的关键之一。在实际应用中，通过故障树分析，将造成设备错误操作的故障源分为若干类，对不同的故障源制定相应的防危策略。再根据一定的原则选择由防危核强化的防危策略，剩余防危策略由应用软件配合完成。使用防危核技术，将过去为保证安全关键系统正常运行而对整个安全关键软件进行验证转变为对一个规模较小的防危核进行验证，因而防危核本身的可信性更易得到保证。防危核的优点可总结为：1) 防危策略由防危核实现，无须过多考虑应用软件的实现、更改和验证；2) 防危核自身代码量较小，结构简单，易于验证；3) 简化了应用软件的实现、验证；4) 系统设备由防危核统一控制。

防危核在系统中是作为一个服务器实现的，通过底层的软件平台捕获应用软件的设备操作请求，并将其传递给防危核加以验证。显然，这样的监控机制开销较大，对某些对时间特性有特殊要求的安全关键系统可能并不适用。除此，防危核对所有的设备操作请求都进行验证也是不必要的，加之在实际应用中功能性与非功能性(防危性)难以分离，同时它也不能很好地防止积极错误的发生，所以防危核在实际应用中也遇到了较大的困难，目前Anderson等人提出的防危壳(Safety Shell)在一定程度上克服了防危核的缺陷[5]。

3　安全关键系统防危性的实现技术

对于安全关键系统的防危性，先确定防危目标，再用具有很强说服力的证据证明系统能取得极高的防危性，而不是描述将如何实现该防危目标。为实现高防危性安全关键系统中的安全关键软件，其可用实现技术有：1) 语言的选择；2) 编译器的选择；3) 代码的检查；4) 形式化代码评审。对于安全关键系统中软件的开发，本文重点讨论语言选择问题。美国国防部(DOD)在很早就规定了三军必须使用Ada语言，由此可见语言的选择对安全关键软件的开发的重要，本文提出一种适用于安全关键软件开发的面向对象程序设计语言 ——Open C++[6]。由于面向对象程序设计采用了封装与继承等技术，它降低了系统的复杂度，同时也提高了可靠性，安全关键系统的设计必须与其很好结合。Open C++是一种“反射”式的面向对象程序设计语言，其中“反射”指系统不但对外部数据完成计算，而且能通过表示其自身结构与计算方法进行容错处理。就使得各种容错机制和例程处理可以方便地集成到面向对象程序设计中去。Open C++集成容错处理的原理如图2所示。在实现安全关键软件时，可把安全关键应用程序分为元级(Meta_Level)和基级(Base_Level)两部分，其中在元级中实现非功能性部分，如容错处理等，而在基级中实现功能性部分。当其它应用程序调用基级中的方法时，通过Open C++特有的反射机制捕获(Intercept 或Trap)该调用请求，即图2中步骤①；然后在元级程序中进行容错处理、访问控制权限认证等，如通过容错处理或认证，则可调用基级中的方法，即步骤②，否则拒绝该不合理调用请求；当调用完基级中的方法后还要对调用结果进行容错处理或正确性确认等，即步骤③，这样可防止不合理的调用结果返回给调用程序；当调用结果通过元级程序的正确性认

证后，便把调用结果返回给调用程序，即步骤④。使用反射式的Open C++编程语言可在输入与输出两方面都实施防危技术，因而非常适合于安全关键软件的开发。

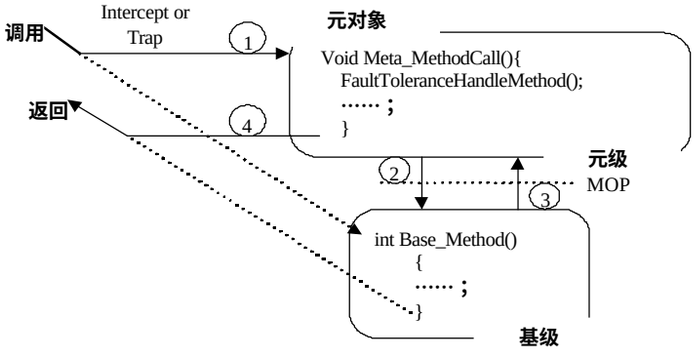

图2 Open C++集成容错技术的原理

基于防危核的安全关键系统实际上是把整个系统分为两大部分，关键部分在防危核中实现，非关键部分则在其余应用软件中实现，这种系统分解为使用反射式编程语言Open C++实现基于防危核的安全保障机制奠定了基础。具体实现时，应在元级程序中实现防危核，而在基级中实现与防危性无关的其余应用部分(即数据处理等功能性部分)。这样当其他系统或操作人员调用基级中的方法时，由于Open C++特有的反射机制的存在，将对其调用请求及调用返回结果进行防危性认证，并拒绝可能导致系统发生灾难性事故的请求与结果，从而提高了整个安全关键系统的防危性。

4 结 束 语

创建高防危性的安全关键系统存在诸多困难，对安全关键软件的设计与实现更是重点、难点所在，相对于普通商用软件的开发它需要投入更多的人力、物力及财力。防危性不应在系统开发成功后再添加进去，必须从开始就融入系统的开发过程中，不断地检验在开发过程中增加不需要、不安全的功能，同时也必须确保不会移去需要的功能。为提高安全关键系统的防危性，本文所提出了能有效防止消极错误发生的防危核设计思想，同时也分析了Open C++用于安全关键软件开发的有效性。两者的有机结合对提高安全关键系统的防危性将是全新的开创。

参 考 文 献

[1] Nancy G L. Software safety in embedded computer systems[J]. Communications of the ACM, 1991, 34(2): 34-46

[2] Anderson E, Katwijk J V, Zalewski J. New method of improving software safety in MissionCritical real-time systems[C]. Proc. 1999 International System Safety Conference, Orlando, Florida, 1999

[3] Diller A Z. An introduction to formal methods[M]. Ed.2, John Wiley&Sons, New York, NY, 1994

[4] Wika K J, Knight J C. On the enforcement of software safety policies[C]. Proceedings of the Tenth Annual Conference on Computer Assurance(COMPASS), Gaithersburg, MD, 1995: 83-93

[5] Wika K J. Safety kernel enforcement of software safety policies[D]. Ph.D. dissertation, Department of Computer Science, University of Virginia, Charlottesville, VA, 1995

[6] Chiba S A. Metaobject protocol for C++[C]. in ACM conference on Object-Oriented Programming, Systems, Languages and Applications, Austin, TX, 1995: 285-299

编 辑 孙晓丹