第35卷 第12期 计 算 机 工 程 2009年6月
Vol.35 No.12 Computer Engineering June 2009
·软件技术与数据库· 文章编号：1000—3428(2009)12—0073—03 文献标识码：A 中图分类号：TP302

一种提高星载软件可靠性的开发方法

段星辉，华建文，代作晓，金小强
(中国科学院上海技术物理研究所，上海 200083)

摘 要：星载软件工作环境的特殊性要求其具有高可靠性和容错性。现有软件可靠性低于硬件可靠性，星载软件失效在系统失效中占有很大比例。分析星载软件特点，根据其开发流程，阐述在软件开发各阶段提高其可靠性和容错性的措施。实践结果证明，此类容错技术是必要且有效的。

关键词：星载软件；可靠性；容错性；冗余

Developing Method for Improving Reliability of Onboard Software

DUAN Xing-hui, HUA Jian-wen, DAI Zuo-xiao, JIN Xiao-qiang
(Shanghai Institute of Technical Physics, Chinese Academy of Sciences, Shanghai 200083)

【Abstract】Because of its special working circumstance, onboard software should be of high reliability and fault-tolerance. A system failure is mostly due to its software failure. One of the reasons is that software reliability is behind the one of hardware. This paper analyzes the characteristic of onboard software. According to the software developing flow, it expounds some measures to improve the software reliability and fault-tolerance during the software developing process. Practical results show that these fault-tolerance technologies are necessary and effective.

【Key words】on-board software; reliability; fault-tolerance; redundancy

星载软件的某个小缺陷可能导致整个系统失效。软件失效将造成灾难性后果，如1996年美国阿丽亚娜5号运载火箭爆炸。开发高可靠性的星载软件是保障整个系统可靠运行的关键之一。在一个大的系统中，软件错误一般占整个系统错误的65%左右。根据美国国防部和NASA的统计，在现有武器系统和航天航空领域中，软件可靠性约比硬件系统的可靠性低一个量级[1]。

1 星载软件的特点

星载软件是一种实时嵌入式软件，要求在规定时间内实现对复杂系统的控制，其实时性要求通常较高。它在星上完成的主要任务包括：遥控指令的解析、执行(或下传给下层模块)，数据采集与处理，遥测数据下传，检测敏感器件和执行机构的故障，进行故障隔离和重构等。星载软件最重要的特点是其运行载体和工作环境具有特殊性。

星载计算机要忍耐火箭起飞时的冲击、振动等苛刻的力学环境，并要承受宇宙空间的高温、低温、高真空、高辐射等极端条件。在太空环境里，星载计算机每时每刻都会受到空间高能粒子的辐射，其后果之一是发生单粒子翻转事件，即高能粒子使计算机存储器或寄存器的0, 1状态发生翻转。单粒子翻转可能使系统出现逻辑功能混乱，它对存储器、寄存器、计数器等逻辑器件的影响很严重。若出现单粒子翻转，则可能出现标志错误、程序走飞或死锁等后果，影响系统功能。由于在太空中运行，星载软件的可维护性差，无法像地面软件一样方便地进行修改和升级。

因此，在开发星载软件时，开发者应充分考虑星载软件的上述特点，从而有针对性的采取措施，以保障星载软件可靠性。

2 提高星载软件可靠性的方法

根据星载软件的特点，可以采取减少软件设计本身的缺陷和提高软件抗干扰能力2种措施来提高其可靠性。下文将按软件开发流程，即软件设计、软件实现、软件测试、软件维护的纵向开发过程具体探讨如何提高星载软件可靠性。

2.1 软件设计阶段

软件工程化技术是保证软件可靠性的基础，它提出了一些软件开发的基本原则和要求。例如，必须像硬件一样，将大型软件的设计、生产和检验分开，由不同人员完成软件设计、程序编码和软件测试；保留设计文档、框图和测试记录，使软件的开发和使用具有可追溯性。对于复杂软件系统的设计，开发者可以使用UML进行基于模型的实时系统和嵌入式系统的开发。UML允许开发者从概念的定义快速转入到概念的测试中，减少了早期风险且促进了对问题空间的检索。概念性缺陷可以在许多与之相关的瑕疵产生前被尽早识别并修正，从而在更短的进度时间内构建出质量更高的系统。

软件设计采用自顶而下和模块化设计的方法，建立一套结构良好的程序系统，将软件设计成由相对独立、功能单一的模块组成的系统，使软件的耦合度最弱、内聚度最高。软件规模与复杂度是影响软件可靠性的一个主要因素。把一个大的系统划分成相对独立的模块，降低了系统实现难度，从而降低了软件在实现过程中出错的可能性，有利于提高软件可靠性。模块化设计方法使软件便于测试与维护。

成熟程序语言的选择是开发者在软件设计初期需考虑的因素之一。开发者一般采用汇编语言与高级语言混合编码。

基金项目：国家“863”计划基金资助项目“超高光谱分辨率红外光谱仪”(2007AA12Z179863)

作者简介：段星辉(1981－)，男，博士研究生，主研方向：嵌入式系统，星载仪器软件；华建文，研究员、博士生导师；代作晓，副研究员；金小强，博士

收稿日期：2009-01-28 **E-mail：**starlight_ustc@yahoo.com.cn

虽然我国于 1992 年将 ADA 语言定为国军标语言，但目前仍然以 C 语言为主进行开发。如果系统需要操作系统的支持，则应选择已被验证可行的操作系统，这是保证软件系统可靠性的关键之一。NASA 采用 VxWorks 作为火星探测器计算机系统的操作系统。在国内，pSOS 已在 SJ-5 卫星上被成功运用。清华大学的“纳星 1 号”采用 μC/OS-II 操作系统。上述操作系统已被证明可以用作星载操作系统，开发者能从中选择合适的操作系统。

在星载软件开发的设计阶段，采用软件工程化管理和模块化设计方法，选择成熟的开发语言与可靠的操作系统，是确保软件可靠性的基础，对于大规模复杂星载软件设计尤为重要。

2.2 软件实现阶段

在软件实现过程中，应充分考虑星载计算机会受辐射干扰、随机干扰、元件失灵、算法失灵等因素影响。采用数字滤波、冗余等软件容错技术可以低成本地解决数据采集不可靠、控制失灵、程序运行失常等故障。其中，冗余技术是一个关键技术，其基本思想是用空间和成本换取系统可靠性。冗余分为同构冗余和异构冗余[2]。同构冗余使用通道副本或精确复制品，异构冗余使用不同方式执行相同功能。软件可以单独在数据或控制过程方面实现冗余，或对 2 个方面同时实现冗余。下文结合星载软件的特点，具体讨论提高系统软件可靠性的措施。

2.2.1 特殊编码方式

为消除单粒子翻转造成的影响，对于重要的标志和故障判别字等关键变量，不使用单一的 0, 1 来判别模式，而以某种特定的 16 位编码形式给出特征。例如，本文采用 interrupt_flag 变量标志 CAN 是否发生接收中断，不用其值为 1 标识发生中断，为 0 标识未发生中断，而是当其值为 0XAAAA 时标识发生中断，为 0X5555 标识未发生中断。即使某个内存位发生跳变，也不会让系统误以为发生中断，可以防止由单粒子效应等原因造成的内存位跳变引起的误操作。

2.2.2 数据存储冗余设计

数据的冗余存储可以采用同构冗余和异构冗余。同构冗余存储可能存储 3 次数据，并在使用时从中取 2 次存储的数据。异构冗余存储可能以二进制反码格式存储数据的拷贝或进行循环冗余校验。编码时，对一些关键变量冗余存储。关键变量主要包括大循环控制变量、有限状态机控制变量、重要的全局指针、重要的全局标志等[3]。上述变量关系到程序运行的总体进程，发生单粒子翻转的概率比一些临时局部变量大很多，造成的后果更严重。以循环体的设计为例具体说明如下：

未采用冗余时：

```
for(i=0;i<10000;i++)
{//执行信号处理 }
```

采用冗余后：

```
for(i1=0,i2=0,i3=0;i<10000;i1++,i2++,i3++)
{i=voter(i1,i2,i3);   //表决
//执行信号处理}
```

采用同构冗余方法，通过复制变量 *i*，在使用 *i* 时用复制品 *i*1, *i*2, *i*3 的多数表决(3 取 2)，增加了关键变量 *i* 的可信度，极大减小了 *i* 的翻转概率。其他关键变量的设计与此类似。

2.2.3 程序存储区的定期检测

在程序固化前，对整个程序代码进行奇偶、求和或循环冗余校验，把校验结果作为常量存储在 3 个位置。当程序从 ROM 或 Flash 中引导到 RAM 中运行时，它在一定时间内对自身程序区进行校验，并将结果与 3 个校验值的表决结果对比。如果两者不一致，则认为程序区发生了单粒子翻转，此时可以采取相应措施进行处理，例如通过程序重新引导的方法恢复正常。

2.2.4 关键寄存器的定期刷新

对一些关键寄存器，如中断屏蔽寄存器 IMR，如果其发生单粒子翻转，后果会很严重。例如，通过设置 IMR 中某一位为 1 使能了相应的中断，如果恰好该位在程序运行过程中发生翻转，即由 1 变为 0，屏蔽了该中断，就会使该中断服务请求得不到响应，这对系统来说通常是致命的。对此类寄存器的处理方法是定时刷新，即在合适的时候对其进行重新写入操作，以避免寄存器值发生跳变造成的后果。

2.2.5 控制过程的冗余设计

软件可以在数据上或控制过程中实现冗余。一个控制过程可以通过在同一台计算机上复制控制环来实现同构冗余，也可以通过在一台计算机上使用 PID 控制环，在另一台计算机上使用模糊逻辑或神经网络算法，以解决同样的问题或由不同团队编写的相同算法。前者可以通过连续运行 3 次控制算法，对运算结果进行表决。后者同时运行 3 个不同算法，最后进行表决。异构冗余较健壮，因为相异冗余防止了系统故障和随机故障。如果由于星载计算机上的一个软件控制算法缺陷造成控制不当，装载相同控制算法的星载计算机将无法弥补该缺陷，而采用不同算法的计算机则可能改变该情况。

2.2.6 星载计算机的冗余设计

采用双机热备份方式，即一主一从，对星载计算机进行冗余设计。当计算机设备发生故障(硬件故障和软件故障)时，可以启动从设备。从设备在软件上可以和主设备一致，或在某个控制算法程序方面不一致。相比前者，在主机没有瘫痪、只是存在软件缺陷时，后者可能弥补上述缺陷。

2.2.7 数据的处理设计

涉及的数据包括遥控指令数据、遥测数据和仪器状态数据。为防止操作人员发送指令时出现失误或指令在传输过程发生错误，输入模块要检查数据的合法性，检查遥控注数注入的数据值是否在系统允许范围内，或者其数据校验正确与否，从而及时进行相应处理以消除影响。对于遥测数据，软件在下传遥测数据时，加入相应校验码(如 CRC)，遥测系统接收到数据时，先对校验码进行判断，正确时才保留本次遥测数据，否则放弃。仪器状态数据一般有电压、温度、电流等。星载计算机如要检测一个子模块供电电压，为消除随机干扰，可以采用软件数字滤波技术保证数据可靠性。常用的软件数字滤波技术有程序判断滤波、中值滤波、均值滤波等，根据不同情况采用不同算法。

2.2.8 软件陷阱法和软件看门狗

软件陷阱法和软件看门狗可以使“脱轨”程序返回“正轨”。软件陷阱法的指导思想是用某种“引导指令”填满未使用的程序内存空间，作为“陷阱”来捕捉“跑飞”的程序。该“引导指令”将捕获的程序强行引向一个特定地址，在该处设置了一段专门进行错误处理的程序来恢复系统的正常运行。

若程序受干扰并弹飞到一个临时构成的死循环中，软件陷阱将失去作用，此时系统会完全瘫痪。软件看门狗可以成功解决此类问题，其具体实现方法如下：在初始化程序中设置看门狗的工作方式，计算各指令执行时消耗的时间，以该时间为看门狗的时间间隔，并设置看门狗中断子程序，此子

程序复位整个系统。

2.3 软件测试阶段

软件的可靠性取决于软件开发的方法与过程，以及软件产品的测试与验证。上述可靠性措施都是针对软件设计前期的工作，为进一步提高可靠性，必须进行测试。软件测试是为了发现程序错误而执行程序的过程，它是软件开发过程保证可靠性的一个重要环节。先对每个程序模块进行单元测试，验证其功能性。然后在此基础上进行组合测试，验证整个系统功能。测试时采用自底向上的渐增式组装方式，逐步组装所有模块。单元测试与组合测试只能发现逻辑和功能错误，星载软件作为实时嵌入式软件，在进行系统联试时，还要进行实时软件测试。在测试过程中，故障注入是一种用于评测星载计算机系统可靠性的有效办法[4]。

2.4 软件维护阶段

在地面上运行时，本文设计很合理，且在软件上采用上述方法确保了系统可靠性。但当星载软件在空中运行时，笔者发现某个软件模块存在缺陷(星载备份模块无法弥补该算法缺陷)，导致系统不能正常工作。此时，需要修改星载软件以改正算法。星载软件的可维护性差，不能像运行在地面上的软件那样方便地进行修改和升级。星载软件不易维护，但不是无法维护。可以通过在轨编程技术，将一个完整的软件功能模块(模块化设计是实现在轨编程功能的基础)的可执行代码，用连续地址的遥控注数方式注入计算机内存，以替换原来的功能模块。通过该方式消除软件功能模块自身的故障(或扩充控制功能，更改控制规律)[5]。

在轨编程功能是通过遥控注数方式，将星载软件的可执行代码注入星载计算机 RAM 区，替代原先固化在 ROM 中的相应指令来实现。在轨编程以子程序(软件模块)为基本替代单位。将可能进行在轨编程的子程序入口地址放在一个子程序的地址表中，软件调用任何子程序时，均通过子程序地址表获得子程序的入口地址。在轨编程时，先将修改的代码(即一个或多个子程序)通过遥控注数注入 RAM 区，然后注数修改子程序地址表中该子程序的入口地址，使之指向 RAM 区的代码。原先 ROM 区的该子程序不再被调用，程序将调用 RAM 区代码。

为了保证在轨编程功能的可靠性，可以采取以下可靠性措施：

(1)对 RAM 区的子程序地址表进行 3 取 2 处理。

(2)校验和检验。为了确保修改后的代码能通过注数真实正确地注入 RAM 编码区，可以对代码进行校验和检验，即在地面和星上分别对同一段代码进行校验和检验并对比结果，从而确定是否在星地注数传输过程中有错误发生。

一个高可靠性和鲁棒性的星载软件应具有在轨编程能力，以保证即使运行于天上的星载软件出现问题，地面系统仍然可以通过在轨编程实现对某个模块的修改与替换，达到恢复系统正常工作的目的，从而延长软件的生命周期。

3 结束语

提高星载计算机系统的可靠性需要从硬件和软件 2 个方面入手。本文从软件角度出发，探讨了如何开发高可靠性的星载软件。

参考文献

[1] 李芳华. 星载软件可靠性设计方法[J]. 上海航天, 2003, 20(3): 24-27.

[2] Douglass B P. 嵌入式与实时系统开发——使用 UML、对象技术、框架与模式[M]. 柳 翔, 译. 北京: 机械工业出版社, 2005.

[3] 邢克飞. 星载高性能 DSP 加固设计方法研究[J]. 电子器件, 2007, 30(1): 206-209.

[4] 彭俊杰. 一种用于星载系统可靠性评测的软件故障注入工具[J]. 宇航学报, 2005, 26(6): 823-827.

[5] 朱 虹, 王海燕. 一种星载软件在轨编程功能的设计和实现技术[J]. 上海航天, 2004, 21(1): 26-31.

编辑 陈 晖

(上接第 72 页)

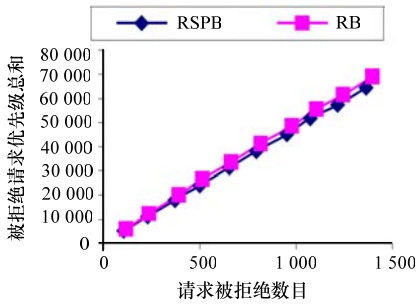

图 5 被拒绝请求优先级总和随请求被拒绝数目变化趋势及对比

5 结束语

本文介绍的网格环境下“基于优先级与效益函数的资源预约调度”通过将优先级与效益函数结合，解决了在网格资源较为缺乏的情况下，提高资源预约成功率的问题。通过模拟与文献[5]中的 Resource Broker 进行比较，在考察指标上均优于后者。尽管算法本身在本文中仅应用于 CPU 资源的预约，但是可以更为一般性地应用于其他网格资源，如存储、网络资源的预约中。此外，对效益函数还可加入其他 QoS 指标，如作业截止时间、最大花费等。

协同预约也是网格预约技术一个重要的方面，在本文基础上，下一步要对此问题进行研究。

参考文献

[1] Garimella G. Advance CPU Reservations with the Dynamic Soft Real-time Scheduler[D]. Illinois, USA: University of Illinois at Urbana-Champaign, 1999.

[2] Bagrodia R, Meyer R, Takai M, et al. PARSEC: A Parallel Simulation Environment for Complex System[J]. IEEE Computer, 1998, 31(10): 77-85.

[3] Kim K, Nahrstedt K. A Resource Broker Model with Integrated Reservation Schemep[C]//Proc. of IEEE International Conference on Multimedia and Expo2000. New York, USA: [s. n.], 2000.

[4] Chu Haohua, Nahrstedt K. A Soft Real Time Scheduling Server in UNIX Operating System[C]//Proc. of European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services. Darmstadt, Germany: [s. n.], 1997.

[5] Smith W, Foster I. Scheduling with Advanced Reservations[C]//Proc. of International Parallel and Distributed Processing Symposium. Cancun, Mexico: [s. n.], 2000.

编辑 任吉慧