第 34 卷 第 5 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.5 Computer Engineering March 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)05—0265—03 文献标识码：A 中图分类号：TP311

航天飞控软件的二维容错体系结构设计

张卫民[1,2]
(1. 北京航空航天大学计算机学院，北京 100083；2. 北京航天飞行控制中心，北京 100094)

摘 要：介绍航天飞控软件系统的主要功能及其常规体系结构。设计开发航天飞控软件系统的一个二维容错体系结构。版本维 A 包括所有应用软件功能的完整功能进程，版本维 B 仅包括部分关键软件功能进程的二版本设计与实现，其中有原功能进程的全功能冗余设计，以及原功能进程的降级冗余设计。版本维主要实现进程级程序的容错功能。关键等级维实现的是不同关键等级进程之间的数据容错。如果数据从较高关键等级进程流向较低或相同关键等级进程，则数据交换可以直接进行，如果数据从较低关键等级进程流向较高关键等级进程，则必须由容错处理进程经过容错处理，才能流向目的进程。
关键词：容错；体系结构；航天飞控

Design of 2-dimensional Faults Tolerant Architecture for Spaceflight Control Software

ZHANG Wei-min[1,2]
(1. School of Computer, Beijing University of Aeronautics and Astronautics, Beijing 100083; 2. Beijing Aerospace Control Center, Beijing 100094)

【Abstract】 The main functions and the usual architecture of spaceflight control software systems are introduced. A 2-dimensional faults tolerant architecture of spaceflight control software systems is developed. Two dimensions are the version dimension and the critical level dimension. The version dimension A has all of the complete function processes. The version dimension B only has redundant designed processes of some of the most critical processes in version dimension A. Some of them have the same functions as the original processes, others have degraded functions. The version dimension realizes the program faults tolerance between processes. The critical level dimension realizes the data faults tolerance among different critical level processes. If the data from a process flows to lower or the same critical level processes, it can be exchanged immediately. If it flows to higher critical level processes, the data has to be processed by a special faults tolerant process program first. After being processed, the data is transmitted to the destinations.
【Key words】 faults tolerance; architecture; spaceflight control

在航天飞行任务过程中，飞控中心应用软件系统主要完成信息交换、数据计算以及对飞行器进行控制等任务。高质量的飞控软件系统对航天飞行任务的成功起着关键作用。为了圆满完成航天飞行控制任务，飞控软件系统必须具有很高的可靠性和安全性。软件容错技术是保证软件质量、提高软件可靠性的一种有效途径和措施[1]。在开发大型航天飞控软件系统时，可以在不同的软件层次上采用相应的容错技术来提高软件系统的容错能力。本文设计的二维容错体系结构可以实现进程级的冗余以及进程间的数据容错能力。

1 航天飞控软件的主要功能

在航天飞行任务过程中，飞控中心应用软件系统主要完成对外数据收发、各类遥测数据处理、外测数据处理、轨道计算、姿态计算、控制参数计算、测控计划生成、遥控处理、数据记录、实时监视、操作控制等任务[2]。为了便于论述，本文将航天飞控软件系统进行简化，假设其只包括如图 1 所示的 7 项软件功能，对这些软件之间的数据交换关系也进行简化，只考虑图中的数据交换。

(1)收发信处理软件通过通信处理机实现飞控软件与各测控站/船、航天发射场，以及其他任务中心之间的数据交换，负责通信数据的打包与分包处理、格式转换、发送频率控制等功能。

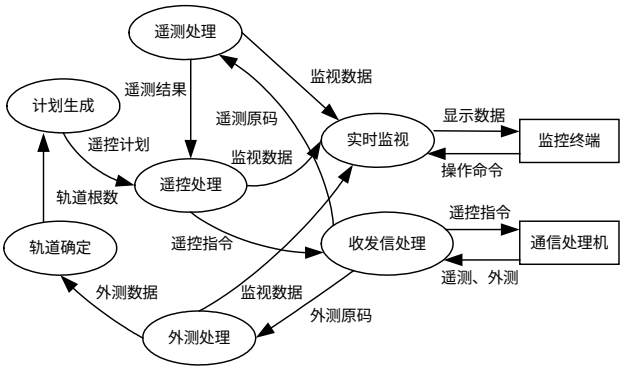

图 1 航天飞控软件数据流图

(2)遥测处理软件完成火箭遥测、卫星遥测/飞船遥测等各类遥测数据的同步、拼帧、计算等功能。

(3)遥控处理软件负责完成遥控计划执行、遥控指令生成、约束检验、指令发送，并根据遥测数据计算结果对指令执行结果进行验证。

作者简介：张卫民(1962－)，男，研究员、博士研究生，主研方向：软件工程，项目管理，软件可靠性，实时软件系统
收稿日期：2007-04-06 **E-mail：**zwm1962@sina.com

(4)外侧处理软件对各种测量数据进行处理，并完成弹道/轨道参数转换等功能。

(5)轨道确定软件根据各种测量数据计算得出飞行器运行轨道根数。

(6)计划生成软件根据任务总体计划、轨道根数以及各测控站/船的布局位置，计算生成测控站/船跟踪测量计划、飞控中心遥控发令计划。

(7)实时监视软件负责搜集各软件的运行状态信息，并对软件运行状态信息和各种关键数据进行显示。

2 航天飞控软件常规体系结构

航天飞控软件系统的全部功能由近百个进程实现。上面介绍的每个软件功能都包含多个程序进程。为了简化描述，假设每个软件功能仅有一个进程实现。飞控软件进程间主要通过数据传输中间件、系统全局段和数据文件 3 种方式进行数据交换。其体系结构如图 2 所示。

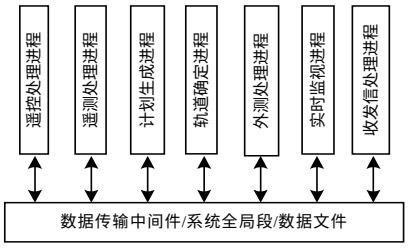

图 2 航天飞控软件体系结构

应用软件系统内的绝大多数数据通过数据传输中间件以电文的形式在进程间进行交换，可以说数据传输中间件是本软件系统中进程间传输数据的枢纽。进程间电文数据传输采用需求请求式，即由需要数据的进程通过调用打开邮箱服务，先打开一个邮箱，然后调用接收服务，申请所需要的数据类，当产生该类数据的进程通过调用数据发送服务发送该类数据电文时，数据发送服务将该数据类电文送入接收者邮箱。数据传输中间件只负责电文在进程间的交换，对电文内容不做任何处理。

有些关键的任务参数和状态，如火箭起飞时间、星箭分离/船箭分离时间、变轨发动机开关机时间，以及一些重要的飞行事件状态等，在整个飞控任务过程中要始终在系统中存在，并且要被多个软件进程使用。对于这类数据，采用系统全局段的方式实现数据的共享。该类数据由产生进程负责写入系统全局段，使用进程通过读取全局段来获得数据。系统全局段在系统启动时创建，凡是需要访问全局段的进程通过映射全局段服务建立对全局段的访问关系。

当进程间非实时交换数据时可以采用数据文件的形式。比如，外侧处理进程在测控弧段内将收到的所有测量数据以数据文件形式记盘，测控弧段跟踪结束后轨道确定进程读取该数据文件进行轨道计算，并将计算出的轨道根数记为数据文件，计划生成进程读取轨道根数文件并生成遥控发令计划文件，遥控处理进程根据遥控发令计划文件完成遥控处理和发令功能。

3 关键软件多版本容错

在航天测控系统中，测控计算机系统大多都采用双工热备份来提高系统的可靠性。但是，由于应用软件系统未能实现多版本开发，双工热备份仅起到了对计算机硬件的备份作用，如果应用软件出现问题往往导致双机软件同时失效。

航天飞控应用软件系统具有较高的安全性和可靠性要求，其中有些软件还具有很高的任务关键等级和实时性要求。但是，由于受到开发进度和资源的限制，目前还无法实现对整个飞控软件系统的多版本容错。不过，在进行体系结构设计时，可以考虑对其中核心的高关键等级进程，比如遥控发令处理进程、遥测处理进程等采用多版本容错技术，来提高其可靠性和安全性。

遥控发令处理进程是飞控软件中关键等级最高的软件，本文对它进行全功能二版本设计，也就是遥控进程 *A* 和遥控进程 *B* 都实现遥控发令处理的所有功能。但是，这 2 个进程由不同的软件开发人员进行设计、编码和测试。这 2 个进程功能一样，只是在最后通过收发信进程向外发出遥控指令时，要首先判断 2 个进程的错误计数状态，以决定是否实际发出遥控指令。遥控处理二版本容错示意图见图 3。

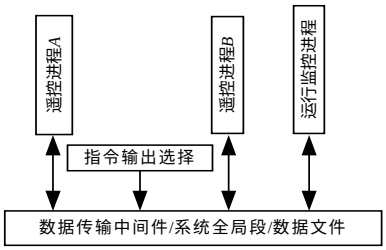

图 3 遥控处理二版本容错示意图

遥控进程 *A* 和遥控进程 *B* 分别设置有一个错误计数变量 *CmdErrCount_A* 和 *CmdErrCount_B*。2 个变量的初值均为 0。对错误计数变量的操作分为 2 部分。一部分是遥控进程本身对它进行操作，当遥控处理进程判出了影响发令的条件，如发现了错误的状态或参数时，将错误计数变量加 1，并结束本次遥控指令处理周期；在遥控进程的异常处理段将错误计数变量加 1；在遥控进程正常完成一个遥控指令处理周期后，不论该进程是否实际发出指令，将其错误计数清 0。为了及时发现遥控处理进程出现死锁、死循环等异常情况，本文增加一个进程运行监控进程，遥控处理进程要定时向监控进程发送“心跳”信息，当监控进程在一定时间间隔内收不到该“心跳”信息时，将其错误计数变量加 1。运行监控进程同时将错误计数情况通知软件操作人员，由操作人员决定是否需要重启或恢复某个遥控进程。

对于遥控进程 *A*，如果 *CmdErrCount_A* 小于或等于 *CmdErrCount_B*，则将遥控指令发出，否则不发。对于遥控进程 B 则相反，如果 *CmdErrCount_A* 大于 *CmdErrCount_B*，则将遥控指令发出。

由于遥测处理要消耗大量的 CPU 时间，在实时飞控软件系统中一般不允许像设计遥控处理进程那样，将遥测处理进程设计为全功能的多个版本。考虑到并不是所有的遥测参数都是同等关键和重要，可以从中挑选出一些关键的参数，比如作为遥控指令执行判据的参数、飞行器关键状态参数等，对它们进行冗余处理。也就是说，将遥测处理设计为一个全功能的处理进程 *A* 和一个降级的只处理关键遥测参数的遥测处理进程 *B*。

同样，对遥测进程 *A* 和遥测进程 *B* 分别设置一个错误计数变量 *TmErrCount_A* 和 *TmErrCount_B*。对这 2 个变量的操作和遥控处理错误计数相似，只是在遥测处理进程正确地完成一个处理周期后不是将错误计数清 0，而是将其减半，即 $TmErrCount\_A=\lfloor TmErrCount\_A/2 \rfloor$。

由于遥测处理进程 *B* 处理的遥测参数远远少于遥测进程

A，因此它出现错误的机会也会远远少于进程 *A*。而且遥测参数帧的下传频率一般比较高，少数的几次处理错误对飞控任务一般不会造成大的影响，同时由于进程 *B* 只是一个降级的处理程序，处理的参数很少，因此一般不会轻易将遥测结果数据切换到进程 *B*。这时，设置一个错误上限 *TmErrLimit*，只有当 *TmErrCount_A* 大于 *TmErrLimit*，并且大于 *TmErrCount_B* 时，才由进程 *B* 发出关键遥测参数处理结果电文，否则一直由进程 *A* 发出遥测参数处理结果电文。

4 不同关键等级软件间的容错

根据飞控软件系统各进程在任务中的重要程度、失效造成后果的严重程度，以及实时性要求等因素，本文将它们分为不同的关键等级。上述 7 个应用软件进程中，收发信处理和遥控处理进程具有最高的关键等级，即其关键等级为 1，遥测处理和外测处理进程的关键等级为 2，轨道确定、计划生成和实时监视进程关键等级为 3。

在软件开发过程中，对高关键等级软件的设计、编码和测试都要采取较为严格的质量保证措施。因此，一般情况下，较高关键等级的软件具有较高的软件质量，高质量的软件一般也会输出高质量的数据。由于受到开发资源的限制，对于低级别软件的开发，不可能像对高关键等级软件开发一样要求。也就是说，低级别的软件一般也会具有相对较低的软件质量和输出数据质量。

如前所述，飞控软件系统各进程通过数据传输服务、系统全局段和文件直接进行数据交换。这些交换数据中，既有从高关键等级进程流向低关键等级进程的数据，也有从低关键等级进程流向高关键等级进程的数据，同时也有相同关键等级进程间的数据交换。

对于某一个进程而言，如果它接收的某类数据来自较高或同等关键等级的进程，由于这类数据具有不低于该进程的质量等级，一般不会对该进程产生负面影响。相反，来自低关键等级进程的数据则有可能对较高级别的进程产生“污染”，特别是当这些数据对进程运行控制产生影响时更是如此。比如，最高关键等级的遥控处理进程，它要接收来自遥测处理进程的结果数据电文，还要读取计划生成进程产生的遥控发令计划，遥测结果电文和遥控发令计划都会影响遥控进程的运行控制过程，这些数据中的错误有可能使遥控处理进程失效。因此，对于从低关键等级进程流向高关键等级进程的数据，必须进行容错处理。

对某类数据的容错处理可以在使用该数据的进程内进行。但在飞控软件系统中有很多的数据电文和全局段数据要被多个进程使用，如果在各个进程中都要重复进行容错处理，会造成很大的开发资源和运行资源浪费。因此，本文在软件体系结构上就考虑对数据的容错，不同关键等级间的容错示意图见图 4。

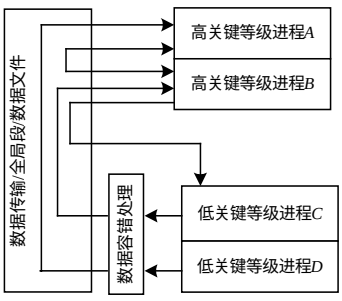

图 4 不同关键等级间的容错示意图

在该体系结构中，设计了专门进行容错处理的进程。当一个应用进程产生的数据要流向较低或相同关键级别的进程时，就直接发出电文、写全局段或写文件。但是，当一个应用进程产生的数据要流向较高关键级别的进程时，则首先将数据以电文方式传送给容错处理进程，由容错处理进程根据数据容错要求进行容错处理后再发出电文、写全局段或写文件。由于有的数据文件要记录大量的数据，考虑到数据传输的效率问题，以及数据文件产生后到被使用之间一般都相隔一段较长的时间，可以将数据文件的容错处理放在该文件生成之后进行，如图 5 所示。

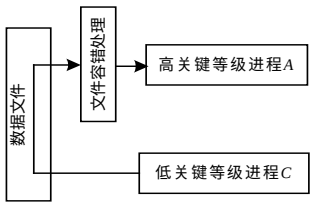

图 5 不同关键等级间文件的容错示意图

5 航天飞控软件二维容错体系结构

综上所述，可以得到航天飞控软件系统的一个二维容错体系结构，如图 6 所示。

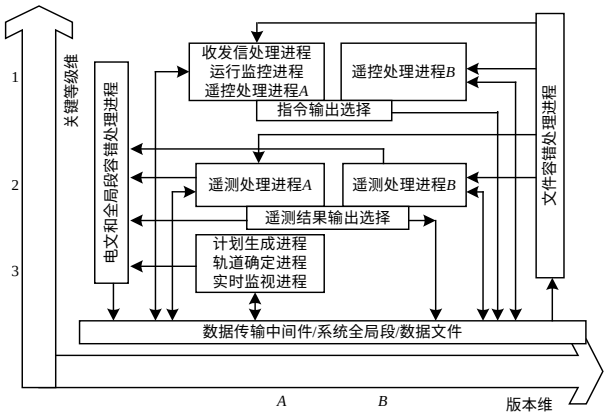

图 6 航天飞控软件二维容错体系结构

在二维容错体系结构中，版本维 A 包括所有应用软件功能的完整功能进程，而版本维 B 仅包括部分关键软件功能进程的二版本设计与实现，其中有的是原功能进程的全功能冗余设计，有的是原功能进程的降级冗余设计。版本维主要实现进程级程序的容错功能。

关键等级维 1 包括整个飞控应用软件系统中最为关键的几个进程，关键等级维 2 和关键等级维 3 包括的进程关键等级依次降低。2 个容错处理进程是比较特殊的进程，可以认为它们 2 个处于关键等级 1。不同关键等级的应用进程间进行数据交换时，如果数据是从较高关键等级进程流向较低或相同关键等级进程，则数据交换可以直接进行；如果数据是从较低关键等级进程流向较高关键等级进程，则数据必须首先经过容错处理进程进行容错处理，然后才能流向目的进程。也就是说，除了 2 个特殊的容错处理进程之外，进程间所有数据交换都只能流向较低或相同关键等级进程，不能流向较高关键等级进程。

6 结束语

软件容错设计是开发高可靠、高安全软件系统的一项主要内容。本文设计的容错体系结构较好地实现了 2 种进程级

（下转第 282 页）