第35卷 第21期 计 算 机 工 程 2009年11月
Vol.35 No.21 Computer Engineering November 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)21—0280—03 文献标识码：A 中图分类号：TP393

基于FSM和事件驱动的卫星管理软件设计

张合生，金玉红，李 杰，盖建宁
(中国航天科技集团第八研究院航天电子技术研究所，上海 201109)

摘 要：随着航天飞行器结构的复杂和软件规模的膨胀，软件的结构成为软件设计的关键技术，针对该问题，提出以有限状态机为模型的软件模块化设计方法。该设计按事件产生、消息(事件)队列维护、事件响应的处理流程，建立事件驱动机制，构建软件系统结构。实验结果表明，该设计提高了软件的可靠性和开发效率，使其更易于维护和扩展，并降低了失效风险。

关键词：有限状态机；事件驱动；管理软件；卫星

Design of Satellite Management Software Based on FSM and Event-driven

ZHANG He-sheng, JIN Yu-hong, LI Jie, GAI Jian-ning
(Institute of Aerospace Electronic Technology, Eighth Institute, China Aerospace Science and Technology Corporation, Shanghai 201109)

【Abstract】 With the structure of aerospace craft increasingly becoming complex and its software scale being more and more inflated, the design of software framework becomes the key technology of the software design. Aiming at this problem, this paper proposes a software modularized design method based on Finite State Machine(FSM) model. It establishes event-driven mechanism and constructs system structure of the software according to the process flow which is event generating, event queue maintaining and event answering. Experimental result indicates that the design improves the reliability and developing efficiency of software, makes it easier vindicating and expanding, and reduces the disabled risk.

【Key words】 Finite State Machine(FSM); event-driven; management software; satellite

1 概述

面对航天新技术的发展和用户的迫切需求，如何有效降低航天型号的高风险，是我国航天事业面临的诸多问题之一[1]。同时软件规模、复杂度及其在整个系统中的功能比重急剧上升，使软件的可靠性与安全性问题日益突出[2]，因此，软件的结构设计成为软件设计甚至整个卫星研发的核心内容。一个合理的软件系统结构和成功的软件设计思路，能降低模块间的耦合度，并从很大程度上提高软件模块的可重用度，增加软件的维护性。可见，优良的软件设计方法有利于系统功能的扩展，可以降低软件设计的难度。

目前，世界上大多数卫星采用了分布式体系结构，卫星的各个分系统利用高速数据现场总线、低速数据现场总线交换数据和控制信息。例如，ESA分布式星务管理系统体系结构包括：有效载荷信息处理分系统，大容量数据存储分系统，数据控制管理分系统，姿轨控分系统，I/O处理分系统等。美国海军研究实验室E. O. HULBURT中心提出的高低速混合网络结构是以CPCI内总线为主干构建的分级分布式体系结构。国内的大中型卫星也大都采用分布式体系结构，配备串行总线(1553B, CAN)组成星务数据管理系统。星务计算机是星务数据管理系统的核心设备，运行于其上的管理软件是管理逻辑的直接实现者，与其分布式体系结构相对应，该软件采用模块化、层次化的设计方法。

有限状态机(Finite State Machine, FSM)模型，为软件的模块化设计提供了一个可行的方案。该软件设计方法大多应用于较大规模的软件设计，例如，文献[3-4]研究了FSM模型在数控机床软件中的应用。随着卫星软件规模的膨胀，FSM模型在航天领域的应用价值逐步得到体现。由于卫星各分系统功能的相对独立，因此可以将卫星功能的调整看成若干个状态之间的变换[5]，鉴于此，软件模块的设计便转化为FSM的设计。在基于FSM模型软件模块设计的基础上，采用硬件服务、应用模块、事件驱动的软件层次化体系结构，实现整个软件的控制流程。

本文在分析FSM模型和基于事件驱动的软件体系结构的基础上，以某小型探测器为例，设计并实现了其管理软件。

2 软件模块的FSM模型

2.1 FSM的基本概念

FSM是由有限的状态和相互之间的转移构成的，在任何时候只能处于给定数目的状态中的一个。当接收到一个输入事件时，状态机产生一个输出，同时可能伴随状态的转移。有限状态机包括以下一些构成要素：

(1)状态：行为模型的基本组成部分，反映了系统中某个对象所处的阶段和活动情况；

(2)转移：对象从一个状态转移到另一个状态的过程；

(3)事件：引起对象状态转化的事件及条件；

(4)动作：在状态转移时，对象所采取的行动。

用集合的方式表述如下：

有穷输入集A：$A=\{a_0, a_1, \cdots, a_n\}$；

作者简介：张合生(1981－)，男，硕士，主研方向：卫星系统与软件平台；金玉红，高级工程师、硕士；李 杰、盖建宁，工程师、学士

收稿日期：2009-05-09 **E-mail：**8103250412@sina.com

有穷输出集 Z：$Z=\{z_0, z_1, \cdots, z_n\}$；

内部状态集 S：$S=\{s_0, s_1, \cdots, s_n\}$；

状态转换映射υ：$S\times A\rightarrow S$；

输出映射ξ：$S\times A\rightarrow Z$。

FSM 的工作原理：在给定一组状态集 S 和输入(事件)集 A 的前提下，若输入 a_n 和状态 s_n 确定，在映射υ和ζ作用下，则确定 FSM 的下一个状态 s_{n+1} 和输出 z_{n+1}。该工作原理通过状态转移表、状态转换图等方式描述。

2.2 模块设计的 FSM 实现

依据上述 FSM 的基本原理，设计卫星某个分系统的软件模块，从而实现系统功能的模块化设计。根据系统(模块)的功能以及与外部的接口关系，确定模块的内部状态集、输入集、输出集，假设模块的内部状态集为 $S=\{s_0, s_1, s_2, s_3, s_4\}$，输入集为 $A=\{a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$，输出集 $Z=\{z_0, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8\}$。模块内部状态转换关系如表 1 所示。

表 1　模块内部状态转换关系

状态	s_0	s_1	s_2	s_3	s_4
s_0	-	-	6	-	9
s_1	1	-	-	8	-
s_2	2	4	-	-	-
s_3	3	5	-	-	-
s_4	-	-	7	-	-

在表 1 中，“ - ”表示不会发生的状态转换，数字表示状态转换的序号，如，数字“ 2 ”表示状态 s_0 到 s_2 的转换，数字“ 6 ”表示状态 s_2 到 s_0 的转换。上述序号所表述的转换都视为不可逆的转换，一个可逆的转换要用 2 个不同的序号表示(如转换 6 和转换 2 共同表示一个可逆的转换)。在确定系统可能出现的状态转换后，可以确定状态转换与输入输出集的映射关系(υ和ζ)，如表 2 所示。

表 2　状态转换与模块输入输出的关系

状态转换序号	状态转换事件	状态转换输出
1	a_0	z_0
2	a_1	z_1
3	a_2	z_2
4	a_3	z_3
5	a_4	z_4
6	a_5	z_5
7	a_6	z_6
8	a_7	z_7
9	a_8	z_8

在表 2 中，状态的转换序号即表 1 所确定的可能发生的状态转换序号，引起状态转换的事件是 FSM 输入集的某个成员，而转换后的输出是输出集的成员。在实际的模块设计中，可能会存在若干个输入引起一种状态转换的情形，也可能出现若干种状态转换相同输出的情况，因此，应根据具体的模块进行相应处理，其原则是使模块的输入输出尽量简化，利于程序的设计和实现。

3　基于事件驱动的软件体系结构

卫星的分布式体系结构，为其核心控制软件-管理软件的层次化设计提供了参考。卫星管理软件运行于综合电子系统之上，而其他分系统通过系统总线或者专有数据接口与综合电子系统进行信息交互，因此，管理软件从层次上分为 3 个层次：驱动层，应用层，事件管理层。在对系统应用层进行以 FSM 为模型的模块化设计的基础上，采用事件(消息)驱动机制对各个模块(系统应用程序)进行管理。

(1)驱动层：驱动层是管理软件最底层的代码，对系统运行安全起着重要作用，因此，驱动服务代码的设计应注重运行效率、可靠性设计。它直接控制硬件或直接与硬件通信，为其上层的应用层模块提供系统硬件驱动服务。如，对于一个串口通信服务，它可能包括串口的打开、串口的关闭、串口数据的读取、串口数据的写入、硬件错误的获取等操作。

(2)应用层：应用层的设计是管理软件功能实现的关键。应用层的模块包括 3 个部分：事件处理器，事件发生器，功能代码，其中，事件处理器可以与 FSM 的输入集对应，把事件作为模块的输入，根据输入调整模块的状态(亦即改变功能代码的分支)；事件发生器负责让模块在新的状态下产生输出，以事件的方式发送给消息管理层。

(3)事件管理层：事件管理层包括事件接收器，消息队列管理器，事件分发器。消息包含消息源信息、消息目标模块信息、消息优先级、消息类型等信息。事件管理层的工作过程为：事件(消息)产生后，由事件接收器接收，插入消息队列；消息队列管理器按一定的算法对事件排队；事件分发器按照消息目标模块信息发送给相应的模块。

基于事件驱动机制的管理软件结构如图 1 所示。

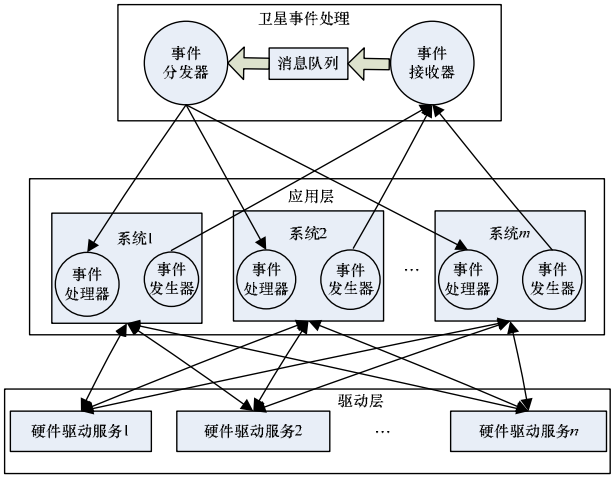

图 1　基于事件驱动机制的管理软件结构

上述的事件处理机制称为分散式事件管理，它把消息产生、消息处理分散到各个模块，同时把软件失效的风险分散到各分系统，有利于软件的可靠性。对于较小的分布式系统，完全按照分散式事件管理可能会导致系统的冗余度增加，因此，对分散式事件管理机制进行改进，称之为集中式事件管理，其结构如图 2 所示。

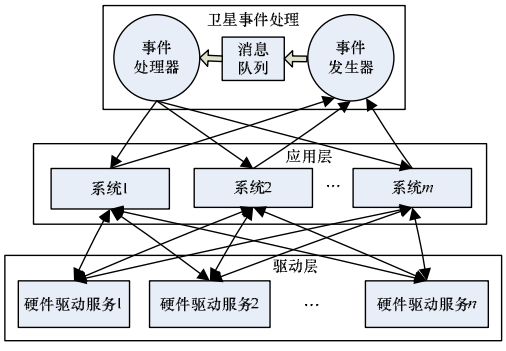

图 2　集中式事件管理机制

集中式事件管理的过程为：将各个模块的输出进行集中，按一定的规则产生系统事件，交由消息队列管理，消息队列按一定的算法对事件进行排队，最后事件处理器取出事件并调用相应的事件处理函数，实现对各个模块的输入控制。由于简化了结构，因此集中式事件管理机制更易于实现，但当

消息或者事件较多时，消息生成规则将膨胀，逻辑会变得较复杂，同时其在系统扩展性方面不如分散式事件处理机制。

4 探测器工程实验

以某小型深空探测器为例，进行具体的实例说明。由于受到体积和重量的限制，卫星在遵从分布式体系的前提下，提高了卫星平台的集成度，摒弃了传统总线，因此直接用同步串行数据通道、异步串行数据通道作为分系统与综合电子系统的数据通信手段。其系统分布如图 3 所示。

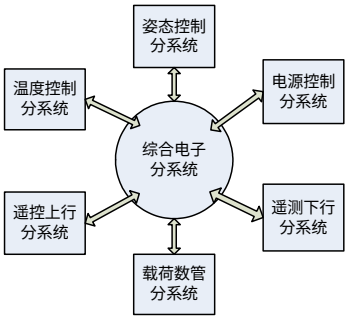

图 3 探测器系统分布

管理软件遵从集中式事件管理的机制，由于系统事件数量较少，因此软件去除了事件排队机制，采用事件即时处理方式，其结构如图 4 所示。

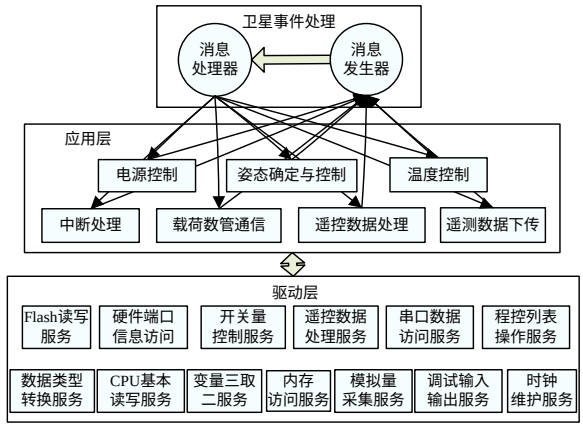

图 4 管理软件系统层次结构

按图 4 所示的体系结构，并依据 FSM 的模块设计方法，设计并实现了探测器管理软件。该软件集成了轨道控制软件、程控测控软件、底层驱动、任务管理等功能。与以往卫星分散控制、中断驱动的开发方式相比，该方法使软件结构更清晰，性能更易于控制，可靠性、可维护性也在系统级得到保障。

由此可见，随着卫星型号研制周期的缩短，以及性能更高的处理器的应用和高级语言的普及，在今后的卫星软件研制中，采用并强化本文的软件设计思路，可以在卫星的可靠性、可维护性、运行性能、开发效率等方面起到重要作用。

5 结束语

本文从分析目前卫星管理软件的可靠性入手，阐述了一种以 FSM 为模型的系统模块化设计方法，并在模块化设计的基础上，分析了基于事件驱动的管理软件系统层次结构设计。与 FSM 模块化设计方法相结合，探讨了事件驱动的基本原理和实现过程，分析了分散式事件处理机制和集中式事件处理机制以及不同处理机制之间的差异。结合某小型探测器的系统特征，采用集中式事件处理机制实现了其管理软件设计。实验结果表明，以 FSM 为系统模型，以事件驱动为系统结构的软件设计方法具有巨大的航天应用价值，值得进一步研究和推广。

参考文献

[1] 孙来燕. 中国航天的发展战略和重点领域[J]. 中国工程科学, 2006, 8(10): 1-13.

[2] 周新蕾, 刘正高. 航天软件可靠性安全性技术应用发展趋势[J]. 质量与可靠性, 2006, (3): 41-49.

[3] 孙维堂, 刘永贤, 张 禹. 有限状态机在开放式数控系统中的应用[J]. 东北大学学报, 2007, 28(8): 1174-1177.

[4] 李为建, 王 文, 秦 兴. 有限状态机在数控系统软件中的应用研究[J]. 组合机床与自动化加工技术, 2003, (4): 50-53.

[5] 姜春英, 房立金, 赵明扬. 基于有限状态机与 Petri 网的系统分析与设计[J]. 计算机工程, 2007, 33(18): 245-248.

编辑 陆燕菲

（上接第 279 页）

4 结束语

本文设计方法将广义表的数据结构作为系统配置时的数据动态存储结构，用来完全描述一个 OIL 配置情况，能够更加便捷地描述和校验系统配置的各对象以及对象各项属性间的关联关系，从而增强配置过程中对各对象配置的可校验性，降低对系统配置人员的条件限制。

本文方法使用户界面更加人性化，实现语言利用 C++的类封装特性以及无类型指针的实现方式，使校验和代码的生成效率更高。

参考文献

[1] 李秀梅, 杨国青. OSEK/VDX 标准与车控电子产品开发[J]. 单片机与嵌入式系统应用, 2005, (4): 27-30.

[2] 章亮飞, 李银国. 嵌入式实时操作系统 AutoOSEK 的设计[J]. 计算机工程, 2007, 33(16): 53-55.

[3] The OSEK/VDX Group. OSEK/VDX System Generation OIL[EB/ OL]. (2004-07-01). http://portal.osek-vdx.org.

[4] The OSEK/VDX Group. OSEK/VDX Operating System[EB/OL]. (2005-02-17). http://portal.osek-vdx.org.

编辑 陆燕菲