International Conference on Small Satellites: Missions and Technology
9-13 de setembro de 1996
Madri, Espanha

THE OBJECT-ORIENTED ANALYSIS OF THE ON BOARD COMPUTER SOFTWARE FOR THE BRAZILIAN SCIENTIFIC MICROSATELLITE - SACI-1

Mário Eugênio Saturno Walter Abrahão dos Santos

saturno@dea.inpe.br wasky@lac.inpe.br

INPE - Instituto Nacional de Pesquisas Espaciais - São José dos Campos - Brazil

Phone: +55-12-325.6186, Fax: +55-12-325.6225

ABSTRATCT:

This paper presents the object-oriented analysis (OOA) of the on board software of the SACI-1, the firts Brazilian Microsatellite for Scientific Application. Emphasis is put on the object analysis using the OMT (Object Modeling Technique) as the underlying OOA methodology and employing the best features of a set of others OOA methods. The on board computer (OBC) hardware architecture uses a fault-tolerant transputer system with dedicated I/O interfaces implementing a RS-422 protocol for bus and payload communications. The OBC software has a specific nucleous where a set of parallel and concurrent applications are run. The OBC software design is being a challenge due to its high complexity and diversity of function. Hence the adoption of a software developmemnt methodology based on the object-oriented paradigm. The application software has the following components: the gorund commmunication, housekeeping and payload data, payload communication and attitude determination and control. A responsability-driven approach is used for the classes decomposition of the OBC software and a set of guidelines presented for the functional and dynamics analysis.

1. INTRODUCTION

This paper addresses the object-oriented analysis (OOA) of the on board computer (OBC) software of the first Brazilian microsatellite, SACI-1. The analysis process guides itself through many analysis methods to cope with its purpose.

Traditionally, software systems analysis has been tackled by structured methods as presented in [Yourdon 89] nevertheless the OOA displays some advantages over the structured approach [Rumbaugh 91].

The OBC hardware architecture is set around a transputer-based system comprising three central processing units (CPU) in a fully redundant scheme. Occam, the natural programming language for transputer systems, is emploied as the developing language for the resident software. Basically, the OBC software architecture deals with the following aspects: bus and payload data acquisition, downlink message enconding, uplink command decoding and distribuition and attitude control.

2. THE SACI-1 MICROSATELLITE

SACI-1 is the first of a series of scientific microsatellites under development at INPE (The Instituto Nacional de Pesquisas Espaciais) [Dos Santos 96]. The project is based upon novel technologinal solutions as well as on previous background in developments for the Brazilian space program.

SACI-1 will weight 60 Kg and will have a almost circular low earth polar orbit inclined 98.5 degrees to the earth equator at a 750Km altitude. During visible passes, scientific data from any of its four on board experiments will be transmitted to ground. A Long March IV launcher is expected to put on orbit the microsatellite piggybacked to the CBERS (China-Brazil Earth Resources Satellite) whose launch is due on October, 1997.

The payload comprises four scientific experiments, namely:

- PLASMEX, a study of the plasma bubbles in the B ionosphere;
-
- PHOTOMETER, a measurement of the terrestrial airglow emissions and
- MAGNEX, a triaxial high precision magnetometer to investigate the geomagnetic field.

3. THE SACI-1 ON BOARD COMPUTER

The SACI-1 On Board Computer (OBC) is responsible for the overall on board data acquisition, data message encoding and transmission to ground; command reception, decoding and distribution; experiment control, rotation and attitude computing and controlling.

The computer comprises three fault-tolerant central processing units (CPU) based on the INMOS T805 transputers and three I/O interfaces. They are connected through 10-Mbps serial links. Each CPU contain a Watch-Dog Timer used to detect CPU malfunctioning and to generate a flag to the other CPU.

Each transputer has a main memory divided into a ROM of 128 kBytes (32 kBytes x 32 bits) and a RAM of 128 kBytes (32 k x 44, 32 bits for data plus 12 bits for parity checks) protected by an error detection and correction circuit (EDAC) that detects single and double errors and correct single errors and an extended memory to store data, a RAM of 4 Mbytes (1 M x 32 bits). The extended memory has a circuit for latch-up detection as it is prone to it.

Each interface is internally redundant and connected to two processing units. The Telemetry and Telecommand interface receives telecommands frames from the ground communication system at 19200 bps and sends telemetry frames back at 250000 bps. The Serial interface implements a communication protocol based on RS-422 between the OBC and experiments at 19200 bps. The Data Acquisition and Command interface acquires thermistor, analog and digital readings from the satellite subsystems and generate on/off commands to internal configuration control.

The T805 is a 32 bit transputer with an integral high speed floating point processor. The transputer is an INMOS microprocessor with four full duplex communications links which facilitates CPU networking enabling parallel and concurrent applications as well as high performance computing. It has an excellent interprocessor communication support, process scheduling and management, microsecond counter, very small cross section, good power consumption.

Transputers can be programmed in most high level languages, but to gain most benefit from the transputer architecture, the software must be programmed on its home language, Occam. Occam is a high level language and provides the maximum program efficiency to explore all the special features of the transputer architecture. Additionally, Occam also provides a framework for designing concurrent systems.

As known, electronic components have sensitivity to space radiation. A single event upset (SEU) occurs when a radiation particle change the content of a memory cell. Also, the total ionizing dose shifts the threshold, increasing power consumption and eventual failure. Finally, a single event latch-up occurs when a particle promotes a short circuit in component.

4. WHY AN OBJECT-ORIENTED DEVELOPMENT APPROACH?

The object-oriented development dues specifically with the three initial phases of a software life cicle which includes: analysis, design and implementation. It embodies a new way of seeing real world problems, focusing in the manipulation of software development essentials rather than in the accidents that may emerge from transposing it to a written code.

Thus the underlying concern of the object-oriented development on the identification and organization of application-domain concepts, rather than their implementation in a programming language, be it object-oriented or not. This applies for our case where the OOA is used for a system whose final implementation language is not object-oriented and suitable extentions are to be undertaken.

As emphasis is put on initial conceptial issues leaving implementation details on a second plane, chances are the final system may suffer less from design flaws which demand more expensive corrections. Moreover, this approach is less restrictive concerning design choices and may lead to a richier final product.

Finally, object-oriented development helps specifiers, developers and customers to talk to each other thanks to a higher abstraction level. Therefore it can serve as a tool for specification, analysis, documentation, interfacing and programming which almost cover the whole span of softwar development.

For an overview of object technology and its applications refer to [El-Rewini 95].

5. ON BOARD SOFTWARE CONSTRAINTS AND REQUIREMENTS

The SACI-1 systems engineering and users in the scientific community have come up with the following list of constraints and requirements for the on board software:

5.1 Software Architecture Requirements

The OBC software shall consist of two hierarchical layers:

- FT Router: the basic software which shall control the application program execution, provide the interface between application programs and hardware and the communication and synchronization between application programs;
- Application Programs: the application software which shall perform the OBC tasks mecessary to meet mission requirements as depicted in Fig. 1.

Fig. 1 - SACI-1 On Board Computer Software Architecture.

5.2 General Constraints

The OBC application programs must be written in Occam programming language which optimizes CPU performance and code generation. These application programs shall be modelled by a set of parallel processes

running simultaneously. The mechanism for the parallel modeling will be supported by Occam that is based upon the concept of parallel in addtition to sequential execution and provides automatic communication and synchronization between concurrent processes. The CPU sharing among processes will controlled by the CPU itself since transputers incorporate this built-in facility.

5.3 Functional Requirements

The OBC functional requirements are listed below:

- Receive telecommand frames from the TC input driver, analyze them and distribute them to the other process;
- Upon reception of time tagged telecommands, store them and send them for proper on/off actuation. The same applies for the case of command messages received from other processes;
- Implement the spin and attitude control;
- Acquire on board status data and format it for telemetry generation;
- Receive payload experiment data and store them in the on board memory for further processing like data compression;
- Receive event and error reports from the various on board processes and store them in the housekeeping data area;
- Perform diagnosis of the OBC memory boards and send the diagnosis results to client processes requesting them;
- Perform the multiple event upset error protection for the mass memory;

6. OBJECT-ORIENTED ANALYSIS OF THE ON BOARD SOFTWARE

The OOA analysis of the on board software is based on a cocktail of OOA techniques using theirs best features:

- OMT - Object Modeling Technique of [Rumbaugh 91] deals with three models: the object model which describes the objects in the system and their relantionships; the dynamic model, describing the interactions among objects in the system and the functional model, descrinbing the data transformations of the system.
- CRC cards - based on the Responsibility Driven Design of [Wirfs-Brock 90] which have three components: class, responsability and collaboration all displayed in one side of a CRC card while class atribuites are representated in the back of the CRC cards.
- Coad-Yourdon - based in [Coad 92] which emphasizes the identification of subjetcs and its objetcs (services, attribuites and relationships).

6.1 The Object Model

The Object model was strongly influenced by the Coad-Youdon and Wirfs-Brock OOA methodology.

6.1.1 Subject Identification

Based on the constraints and requirements presented earlier we identy the major subjects of the OBC software as depicted in Fig. 2 and their respective responsibilities:

- Ground Communications: shall send all on-board data (telemetries) to ground and receive telecommands (data/commands) from the ground station.
- Experiment Communications: shall send commands to experiment processors and receive experiment data from them.
- Data-Handling: shall collect environment and experiment data and store them into memory blocks.
- Attitude Control: shall verify spin and attitude control and correct them as necessary.

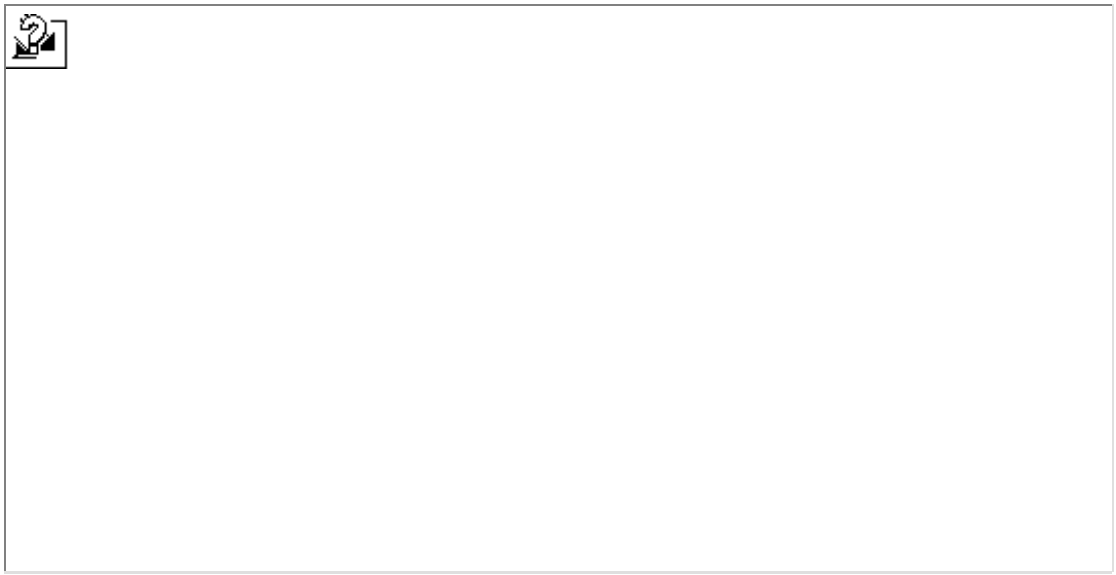

Fig. 2 - OBC Software Subject Identification based on Coad-Youdon.

6.1.2 - Objects Identification

The next step, object identification we enhance the Coad-Yourdon guidelines to incorporate directions of [Gomma 93] tailored for object determination in the domain of real-time systems. Therefore, real time objetcs can be abstracted in the following classification: external device I/O objetcs, control objects, data abstraction objects, algorithm objects and user role objects.

For the purpose of clarity in nomenclature, we will substitute the term ‘object’ of the Coad-Yourdon and Gomma methodologies to the term ‘class’ since this is more general in the sense that an object is an instance for a specific class. Similarlly, we will replace the term ‘subject’ for ‘subsystem’, worth mentioning that subsystems here does not have the same semantics of (spacecraft) subsystems used in the systems engineering jargon. This new nomenclature will help us incorporating some of the ideas of [Wirfs-Brock 90] later on.

Henceforth, in Table 1, we have a list of classes identified per their respective susbsystems together with their acronyms.

In the [Wirfs-Brock 90] methodology, subsystems and their classes have a similarity with the ideas of modular cohesion and coupling in a hierarchical software architecture according to the structured analysis [Pressman 92]. Thus a need to represent collaborations among classes to cope with their responsabilities and the way these collaborations are undertaken is by a network of interobject messages as roughly shown on Fig. 3.

Tab.1 - OBC Software Classes Identification per Coad-Yourdon & Gomma.

UNDERLYING SUBSYSTEMS	PERTINENT CLASSES
Ground Communications	• ground (**Ground**) • telemetry (**Tm**) • telecomand (**Tc**)
Experiment Communications	• experiment communications (**ExpComm**) • experiment (**Exp**)
Data-Handling	• experiment data acquisition (**ExpDatAcq**) • environment data acquisition (**EnvDatAcq**) • housekeeping (**HousKeep**) • diagnosis (**Diag**) • Fourier analysis (**Fourier**) • memory manager (**MemMgr**)
Attitude Control	• attitude control and determination (**AttCtrlDet**) • spin control and determination (**SpinCtrlDet**)

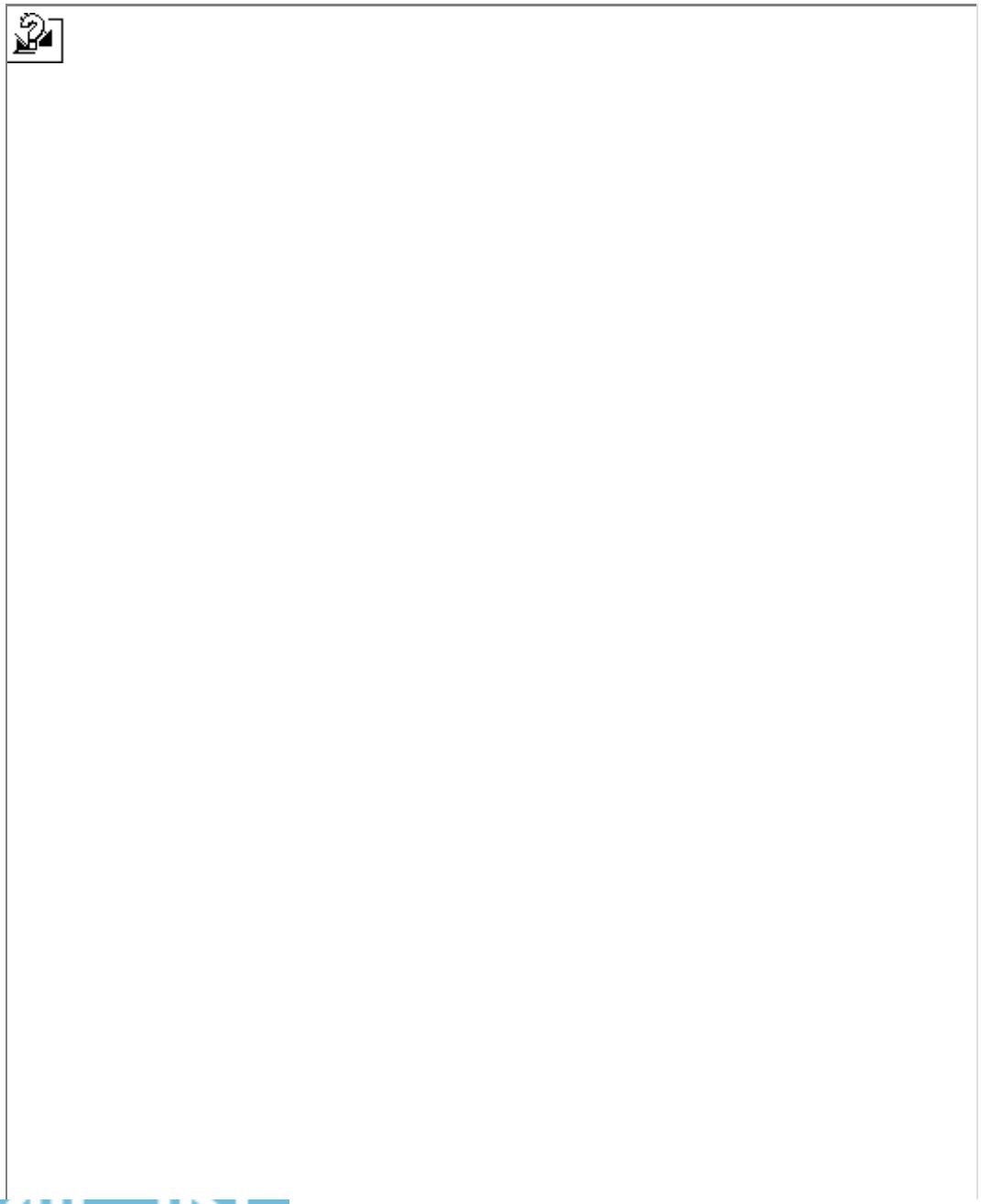

Fig. 3 - Collaboration graph among OBC software classes.

Next, we explicit the CRC cards fo each classes containing its responsabilities, atributes and collaborations:

6.2 Futher Developments

Many issues are to be tackled for the complete OOA and these are our present interests for the completion of the work:

- Organize a class hierarchy so that data and methods can be reused.
- Derive the functional analysis per the OMT methodology and IEEE stantards since both seems to converge on theirs tools
- Derive the dynamics model per [Gomma, 93] and the Use CASE Maps [Buhr, 96] methodologys both properly tailored for OO Real Time Systems
- Finally, deal with the mapping of an OOA into an Occam-based implementation for that [Rumbaugh, 91] and [Coad, 92] will serve as guidelines. Futhermore, since transputers communicate with one another by message exchanging, code mapping can mimic the way objects communicate in an OO paradigm.

7. CONCLUSIONS

This paper presented the initial developments of using an object-oriented approach for the analysis phase of the software which will be on board of SACI-1.

SACI-1 is a first of a series of microsatellites whose payloads were specially built for the Brazilian and abroad science communities.

The computer hardware architecture includes: a fault-tolerant system composed of 3 T805 transputers (32 bits) and 3 specific purpose I/O interfaces (telemetry and command, RS-422 serial communications, A/D and Thermistor inputs and On/Off outputs for bus/payload configuration).

The on board software architecture is divided in to two main modules: (1) a nucleous which provides interface between the on board applications and the hardware and provides fault-tolerance facilities and (2) a set of application programs peforming the tasks for mission completion.

The OMT (Object Modeling Technique) was used as the prime analysis tool where a cocktail of OOA techniques has given their contribuition, some with a novel view and others as an object extension of a structured approach.

The OOA developed so far, contemplates only the object model while the functional and dynamics analysis are still to be investigated. This work points out to a list of things to be done in order to accomplish the whole OOA process. Futhermore, a special mapping will be elaborated so that a non-object oriented implementation language, Occcam on our case, could be efficiently used.

BIBLIOGRAPHY

[Budd 91] T. Budd - *An Introduction to Object-Oriented Programming* - Addison-Wesley, 1991.

[Buhr 96] R.J.A. Buhr & R.S. Casselman - *Use CASE Maps for Object-Oriented Systems* - Prentice Hall, 1996.

[Coad 92] P. Coad & E. Yourdon - *Object-Oriented Analysis* - 2nd. Edition, Prentice Hall, 1992.

[Dos Santos 96] W. A. Dos Santos *et al*. - *SACI-1 Configuration Status* - INPE Internal Report

[El-Rewini 95] H. El-Rewini & S. Hamilton - Object Technology - A Virtual Roundtable - *IEEE Computer*, v.28(10), October, 1995, pp. 58-72.

[Gomma 93] H. Gomma - A Behavioral Analysis Method for Real-Time Control Systems - *Control Engineering Practice*, v.1(1), February, 1993, pp. 115-120.

[Pressman 92] R. S. Pressman - *Software Enginneeirng - A Practicioner's Approach* - 3rd Edition, MacGraw-Hill, 1992.

[Rumbaugh 91] J. Rumbaugh - *Object-Oriented Modeling and Design* - Prentice Hal, 1991.

Designing Object-Oriented Software - Prentice Hall, 1990.

[Yourdon 89] E. Yourdon - *Modern Structured Analysis* - Prentice Hal, 1989.