SpaceOps COMMITTEE QUARTERLY

SOCQ No. 2: 1st Quarter 2010 Back to Products

CONTENT OF ITEM 2

TITLE

*"**SICSDA:** An Adaptative Configurable Distributed Software Architecture Applied to Satellite Control Missions"*

AUTHOR(S)

Adriana C. Thomé, INPE; Maurício G. V. Ferreira, INPE; João Bosco S. Cunha, UNIFEI - Brazil

CORRESPONDENT

Dr. Maurício G. V. Ferreira, INPE, Mauricio@ccs.inpe.br

TECHNICAL REVIEWER

Mr. Terry D. Linick, JPL, U.S.A.

EDITORIAL SUPERVISOR

Dr. Eduardo W. Bergamini, INPE

EDITORIAL SUPPORT

Ms. Síntique R. dos Santos, INPE

VERSION

9 March, 2010

MORE INFORMATION

www.inpe.br

SICSDA: AN ADAPTIVE CONFIGURABLE DISTRIBUTED SOFTWARE ARCHITECTURE APPLIED TO SATELLITE CONTROL MISSIONS

[1]ADRIANA CURSINO THOMÉ
adriana.thome@dir.inpe.br
[1]MAURICIO G. V. FERREIRA
mauricio@ccs.inpe.br
[2]JOÃO BOSCO S. CUNHA
School of Engineering of Itajubá / UNIFEI – Brazil
bosco@unifei.edu.br

[1] National Institute for Space Research (INPE)
Av. dos Astronautas, 1758 - São José dos Campos - SP - CEP 12227-010 – Brazil
[2] School of Engineering of Itajubá (UNIFEI) – Brazil
Caixa Postal 50 – CEP 37500-903 - Itajubá – MG - Brazil

ABSTRACT

This paper proposes an adaptive configurable distributed software architecture applied to satellite control missions called SICSDA (Adaptive Distributed Satellite Control System). The main purpose of this architecture is to control more than one satellite through one set of computers, enabling the choice of each satellite to be monitored in any given period of time. This architecture allows a new mission to be settled without the need for the creation and addition of a specific software component for the satellite being launched, thus minimizing the effort needed to adapt the complete system to the new requirement. It also provides domain specialists and software developers with the capability to configure, if necessary, attributes and business rules to the satellites already launched, adding new elements to business domain without the need of extra codification. The functionalities offered by the application, for example, telemetry visualization and the sending of telecommands, can be distributed into a network pre-defined domain. The system charge distribution service will define the objects location, what means that each machine in the network will be able to have a different view of the metadata stored in the database. A "view", in this context, is the piece of the adaptive object model that will be instantiated in that machine.

Key-words: adaptive object models, metamodels, distributed systems.

1. Introduction

This work presents an adaptive configurable distributed software architecture applied to satellite control missions called SICSDA (Adaptive Distributed Satellite Control System). This architecture was proposed as a Doctoral Thesis of Applied Computing Course at Brazilian National Institute for Space Research (INPE).

INPE, as one of the main organizations involved in the accomplishment of Brazilian´s space program, has taken the responsibility for producing and controlling the Brazilian satellites. In its initial stage, the Brazilian space program comprises the launching of four satellites, all of which have already been launched: SCD1 (Data Collector System 1) and SCD2 (Data Collector System 2), both data collector satellites; and CBERS1 (China Brazil Earth Research Satellite 1) and CBERS2 (China Brazil Earth Research Satellite 2), both remote sensing satellites.

INPE has created a robust infrastructure, aiming at the development of its capability for fully producing and controlling satellites, composed by the Satellite Control Center (CCS), by two remote tracking stations (Cuiabá and Alcântara Stations), by a communication network (RECDAS), as shown in Figure 1; and finally by a satellite control software application (SICS). The remote stations are located at Cuiabá and Alcântara and they build together with CCS the ground support for the satellites in orbit. CCS is located in São José dos Campos and it is responsible for assuring the perfect operation of the satellite. RECDAS links CCS with Cuiabá and Alcântara stations.
A specific machine or a specific set of machines is used to run an application for collecting the data and for controlling the internal state of each satellite. The software application is uniquely developed to control each satellite from Earth. When a new satellite is launched, a new software application must be developed or adapted to that specific satellite, and a set of machines must be allocated to run this software, causing an extra development cost of hardware and software. This is necessary because each satellite has its own features that normally vary a little bit from one satellite to another. The access to these applications is restricted to satellite controllers physically located at CCS.

The problem stated above caused one to think about the creation of an architecture for the development of a more adaptive software system to control satellites of different kinds making use of the same machine or of the same set of machines. The requirements on the new system should include the capability for expanding it without many difficulties, for example in the case of adding features of a new satellite to be controlled, and the capability for making these additions without jeopardizing the quality of the overall system being modified.

These requirements led further to the proposal for the creation of an architecture to be used for the development of applications that need to evolve as the problem domain evolves, based on the idea of building more configurable software components, which are flexible and adaptive, thus allowing the system to adapt itself to domain necessities more easily, that is, allowing the system to match requirements evolution while maintaining its quality.

According to the literature on the subject, one way for doing this is to move some aspects of the system, like business rules for example, to databases, making them easier to be modified. The resulting model allows systems to adapt themselves to new domain necessities through the modification of the values stored in the databases, instead of modifications of the code [14].

This encourages the development of software features that allow decision makers and managers to introduce new elements to existing software without the need of additional coding, and that these changes are introduced in the domain models at runtime, reducing significantly the time to incorporate new ideas to the software.

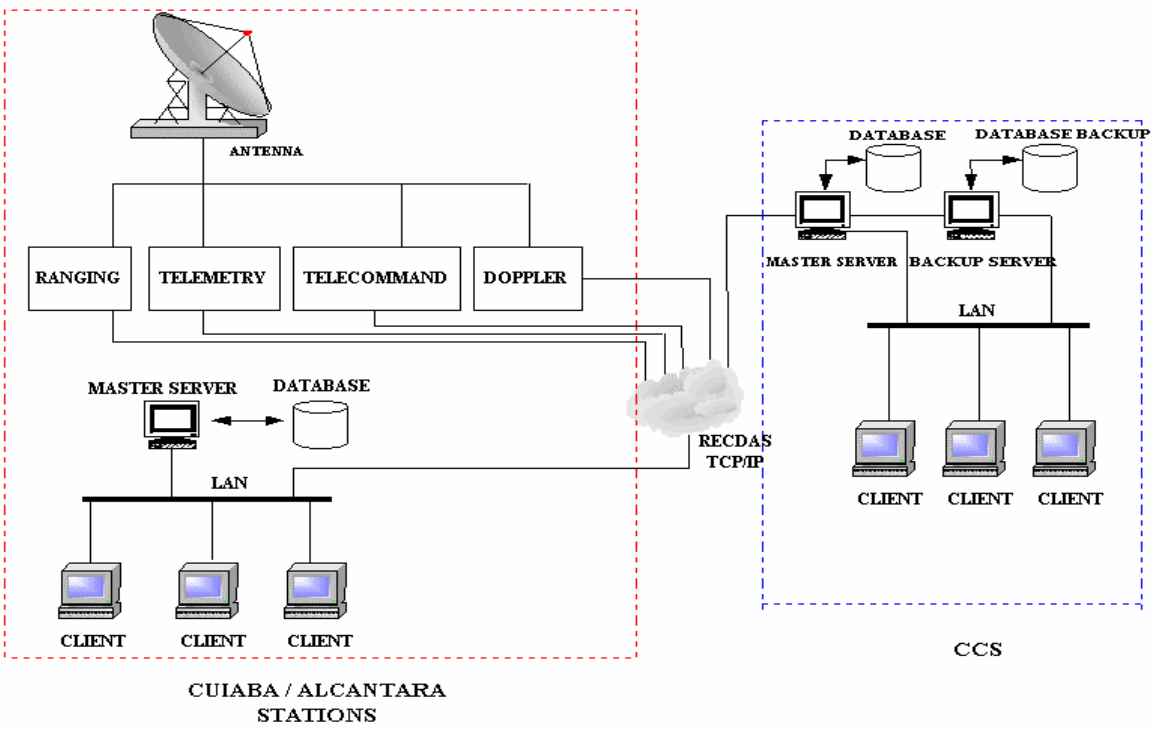

Fig.1- Satellites control system simplified architecture.
SOURCE: adapted from [7].

Architectures that can dynamically adapt themselves to new user's requirements at runtime are called "reflexive architectures" or "meta-architectures. An "Adaptive Object Model Architecture" is a particular kind of reflexive architecture that holds object oriented systems that can be extended to incorporate new elements [14].

Thus, an adaptive object model is a system that represents classes, attributes, relationships, and behavior as metadata. The system is a model based on instances rather than classes. Users change the metadata (object model) to reflect changes in the domain. These changes can modify the system behavior. In other word, the system stores its object model in a database and interprets it. Consequently, the object model is adaptable; when the descriptive information is modified, the system immediately reflects those changes.

The design of adaptive object models differs from most object-oriented designs. Normally, object-oriented design would have classes for describing the different types of business entities and associates attributes and methods with them. The classes model the business, so a change in the business causes a change to the code, which leads to a new version of the application.

An adaptive object model does not model these business entities as classes. Rather they are modeled by descriptions (metadata) that are interpreted at runtime. Thus, whenever a business change is needed, these descriptions are changed which are then immediately reflected in the running application [5].

The use of an adaptive object model approach (AOMs) in the development of systems can ease some kinds of problems that can be found by software developers, mainly the problems related to system flexibility, evolution and maintenance; reducing drastically the total cost of software development [6].

2. SICSDA Architecture

The main purpose of this architecture is to control more than one satellite through one set of computers, enabling the choice of each satellite to be monitored in any given period of time.

Another important point is to have an architecture that allows a new mission to be settled without the need for the creation and addition of a specific software component for the satellite being launched, thus minimizing the effort needed to adapt the complete system to the new requirement

SICSDA architecture models the satellite control application based on adaptive object models. Therefore, in SICSDA architecture, problem domain objects, for example, telemetry, telecommand, ranging; instead of being located in the code that implements the application, will be implemented in an object oriented database to be interpreted and instantiated at runtime. It means that the system will have a generic code that will be able to handle the different satellite object models too.

SICSDA needs to be fault tolerant, so this architecture will be distributed, and the functionalities offered by the application; for example, telemetry visualization and the sending of telecommands, can be distributed into a network pre-defined domain. It means that application objects can be instantiated in different machines of the network, causing a distribution of the system code, and making each machine in the network being able to have a different view of the metadata stored in the database. A “view”, in this context, is the piece of the adaptive object model that will be instantiated in that machine.
In addition, SICSDA architecture suggests the use of distributed databases through the replication of the database for each node of the network. It is important to remark that it will be necessary to have mechanisms that help the control and the storage of metadata recording, and mechanisms that help the versions control.

SICSDA architecture will be configurable because it will be able to: **(1)** allow domain specialists and developers to re-configure the database to handle new classes, by creating these classes, their attributes and methods at runtime; and **(2)** allow, at runtime, the choice of which satellite metadata to use, causing a new object model instantiation in the generic code each time that this kind of context change is asked by the user, that is, each time the user wants to control a different satellite.

SICSDA architecture will be dynamic or adaptive because it will be able to handle possible changes in the business domain, allowing it to match the evolution of the business requirements and to adapt itself to the user’s necessities. Figure 2 shows the structure of SICSDA.

This way, domain specialists (satellite controllers and satellite engineers) and software developers can adapt the system to arrange new classes, creating, at runtime, these classes and their attributes. Thus, we can say that SICSDA architecture is configurable considering business rules, since it is possible for new business rules (or new methods) to be associated to a domain class at runtime.

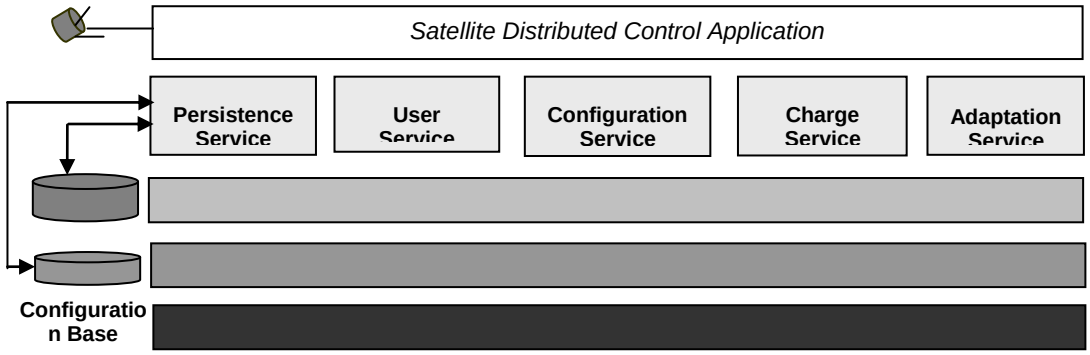

Fig.2 – SICSDA Architecture.

Following, the elements that appear in Figure 2 are detailed:

- **Satellite Control Application:** contains the objects of the software that controls the satellites (telemetry, telecommand, ranging, etc).

- **Persistence Service:** is responsible for storing and retrieving the metadata from the database. Additionally, this service is responsible for storing and recovering users' authentication data and objects configuration data (nodes where the architecture is installed) from a database called "Configuration Base.

- **Configuration Service:** belongs to the system presentation layer and is responsible for maintaining the metadata and the objects configuration in the nodes where the architecture is installed. Offers to domain specialists and software developers an appropriate interface to perform their activities.

- **Adaptation Service:** is responsible for providing the metamodel that allows the objects recovered from the database to be adequately initialized and modified at runtime.

- **Satellite Simulator:** software that simulates the interaction with satellites.

- **Charge Service:** is responsible to give the initial charge in the system, in other words, to run the system charge routine in the nodes where the architecture is installed. Therefore, after one, through users interface, informs the satellite needed to be monitored in a piece of time, the charge system loads the respective satellite metamodel from the database, interacting, at this moment, with persistence service. So, the charge service is going to instantiate the objects recovered from the database, creating the objects of the chosen satellite application and constructing the real object model of the application at runtime.

- **User Service:** belongs to the system presentation layer and it is responsible for offering to domain specialists an appropriate interface to visualize telemetry, to emit telecommands and to obtain distance and calibration measures for the desired satellite. Besides, this service is responsible for providing for the users the appropriate interface to login in the system.

Thus, as shown in Figure 2, the Charge Service charges the satellite metadata corresponding to the satellite that was chosen to be monitored in a given period of time. To do that the Charge Service activates the Persistence Service that is responsible for recovering the metadata from the database. The Charge Service instantiates the objects recovered from the database, creates the objects of the chosen satellite application, and constructs, in this way, the real object model of the application.

The user can interact with the objects through the User Service. If the user wants to initiate the monitoring of a satellite, the User Service activates the Charge Service to charge the appropriate objects from the database. The data from satellites can be obtained through the Satellite Simulator.

The application objects will be distributed and can be accessed through middleware. The connection service will be responsible to locate an object in the system.

If the satellites metadata need to be changed to reflect changes in the domain, the user responsible for the configuration of the system can do these changes using an appropriated interface. When this kind of change occurs the Adaptation Service is activated. The Adaptation Service, as a consequence, activates the Persistence Service, that will be responsible for reflecting in the database these changes.

3. Final Results

Relying in the class diagram that represents the business domain classes for SCD1, SCD2 e CBERS2 satellites, it was possible to obtain a generic class diagram that represents the Satellite Control System business domain metamodel. This metamodel was achieved after applying the following design patterns: pattern TypeObject, pattern Property, pattern Accountability e pattern Strategy. More details about these patterns can be obtained in [5] e [14].

After the generic class diagram achievement, the metamodel was oriented to fit the chosen implementation platform. As was established J2EE as the implementation platform, each class that appears in the generic class diagram was represented as an Entity Enterprise Java Bean.

The implementation of SICSDA architecture was composed of: Java as the programming language, Cache as the database management system, JBOSS as the application server and Jbuilder as the development environment.

The metamodel was represented in database and the metadata to each satellite could be stored through a graphic interface offered by the Configuration Service. Through this interface, it is possible to change at runtime the classes and their attributes, and also to associate new methods to the classes.

The database was populated with satellites metadata through the Configuration Service interface. The metadata stored were based, as much as possible, in values approximated to values used in real control systems.

Users interact with the system to perform an operation, for example, “View Telemetry”, “Send Telecommand” or “Obtain Measures”, trhough the interface provided by the User Service. Thus, it is possible, depending on the satellite that is desired to control, ask for an operation to the system and recover the values associated to this operation. This operation is, actually, dynamically invoked, since its name and the information about its parameters and returned values are only known at runtime. In this manner, it is possible to associate methods already created to new kind of messages, what means that it is possible, in a point of view, to change the system behavior at runtime.

4. Conclusion

This work proposes an adaptive configurable distributed architecture for the development of satellite control systems that are based on adaptive object models called SISDA.

One of the main advantages of using an architecture based on adaptive object models is, on the one hand, the ease to make changes in the system, since it allows developers and domain specialists to extend the system dynamically, that is, they can “code without coding”.

On the other hand, the possibility of creating new business rules without changing system code brings up some difficulties. One reason for this is that, to make it possible, it is necessary to build a specific domain language, what makes the developers inherit all the problems associated to the development of any language, like the construction of debuggers, and the provision of features to make system documentation and the control of versions.

Besides, it is necessary to build an interactive interface so that developers and domain specialists can create new rules at runtime, what requires tools and support for the use of graphical interfaces as well.

Furthermore, architectures based on adaptive object models require the construction of generic code. This generic code is usually difficult to be built since it is necessary to have a high level of abstraction.
Another important issue is the performance of the system. Since the adaptive object model needs to be interpreted to build the real object model and in order for it to reflect the changes made to the system, the system execution is affected accordingly, requiring adequate software and hardware support.

In addition, developers that are not familiarized with this kind of architecture find them more difficult to understand and to maintain than the traditional architectures, since two systems co-exist: the generic code written in an object oriented codification language and the AOM that is interpreted. On the other hand, developers familiarized with this kind of architecture believe that it makes the system easier to maintain in comparison with traditional architectures, since the systems have less code lines and a small change in it can typically cause a big change in the system being run.

Despite of the fact that many aspects point out that, at least in the beginning, architectures based on AOMs require more effort to be built than the traditional architectures, this work intends to take a significant step towards systems reusability, since adaptive systems have the property of matching business requirements evolution, and at the same time, they have a general-purpose architecture, what make them highly configurable and adaptable.

Another important step can be taken in the direction of systems maintainability. This is possible because the effort to make changes in the system can be minimized, since changes in the code can be minimized substantially. In addition, with this kind of architecture it is possible to allow domain specialists to make some changes in the system by themselves, increasing the maintainability and minimizing developer intervention in the system evolution.

Besides, one also hopes to take an important step towards economy, since future missions will be able to use all the hardware and software investment that has been made for previous missions.

The work proposed here has a multidisciplinary nature, since it is a binding of some features that have been explored independently in different research areas until now. Besides, this work is based upon relevant efforts made in the past for the creation of an architecture to model satellites control application based individually on the SOFTBOARD and on the SICSD architectures presented in [3] e [4], respectively.

The studies developed originated the thesis proposed in [12] and had been presented and/or published in [8], [9], [10], [11], [12], [13]; and originated the studies developed in [1] and [2].

This way, one hopes to be collaborating with to the advance of computational research in Brazil, and to be contributing to the success of the Brazilian space mission, by proposing a new alternative architecture for the development of satellites control software, and mainly, by proposing a new approach for the development of adaptive systems.

5. References

[1] ALMEIDA, W. R. **Uma abordagem para a persistência dos modelos de objetos de sistemas distribuídos, configuráveis e adaptáveis.** São José dos Campos, 2004. Dissertação (Mestrado) em Computação Aplicada. Instituto Nacional de Pesquisas Espaciais (INPE).

[2] CARDOSO, P. E. **Modelo de objetos dinâmico aplicado ao processamento de telemetrias de satélites.** São José dos Campos, 2004. Dissertação (Mestrado) em Computação Aplicada. Instituto Nacional de Pesquisas Espaciais (INPE).

[3] CUNHA, J. B. S. **Uma abordagem de qualidade e produtividade para desenvolvimento de sistemas de software complexos utilizando a arquitetura de placa de software – SOFTBOARD.** São José dos Campos, 1997. Tese (Doutorado) em Computação Aplicada. Instituto Nacional de Pesquisas Espaciais (INPE).

[4] FERREIRA, M. G. V. **Uma arquitetura flexível e dinâmica para objetos distribuídos aplicada ao software de controle de satélites.** São José dos Campos, 2001. Tese (Doutorado) em Computação Aplicada. Instituto Nacional de Pesquisas Espaciais (INPE).

[5] JOHNSON, R.; YODER, J. W. The adaptive object-model architectural style. **IEEE/IFIP Conference on Software Architecture 2002 (WICSA3´02).** Canada. August of 2002.

[6] LEDECZI, A.; ET AL. Synthesis of self-adaptive software. **IEEE Aerospace Conference Proceedings.** USA. V 4, pg. 501-507. 2000.

[7] ROZENFELD, P.; ORLANDO, V.; FERREIRA, M.G.V. Applying the 21th century technology to the 20th century mission control. **SPACE OPS 2002.** USA. October of 2002.

[8] THOME, A. C.; ET AL. SICSDA – An adaptive configurable distributed software architecture applied to satellite control missions. **XVII SBES - Simpósio Brasileiro de Engenharia de Software – VIII Workshop de Teses em Engenharia de Software.** Manaus, AM**.** October of 2003.

[9] THOME, A. C.; ET AL. Uma Arquitetura de Software Distribuída, Configurável e Adaptável Aplicada às Várias Missões de Controle de Satélites (SICSDA).**III Worshop dos Cursos de Computação Aplicada do INPE**. São José dos Campos, SP. Novembro de 2003.

[10] THOME, A. C.; ET AL. Establishing an adaptive configurable distributed software architecture applied to satellite control missions. **Eighth International Conference on Space Operations – SpaceOps 2004**. Montreal, Canada. May of 2004.

[11] THOME, A. C.; ET AL. SICSDA: an Adaptive Configurable Distributed Software Architecture Applied to Satellite Control Missions. **18th European Conference on Object-Oriented Programming - ECOOP 2004.** Oslo, Norway. June of 2004.

[12] THOME, A. C.; ET AL. **SICSDA – uma arquitetura de software distribuída, configurável e adaptável aplicada às várias missões de controle de satélites**. São José dos Campos, 2004. Tese (Doutorado) em Computação Aplicada. Instituto Nacional de Pesquisas Espaciais (INPE).

[13] **THOME, A. C.; ET AL (f). Uma arquitetura de software distribuída, configurável e adaptável aplicada às várias missões de controle de satélites.** Revista DaVinci. **Volume II, número 1, pg. 117-132**. **ISSN 1807-6432. 2005.**

[14] YODER, J. W.; ET AL. Architecture and design of adaptive object-models. **ACM Sigplan Notices.** Vol. 36, Fasc. 12, pg. 50-60, December of 2001.