AUTOMATING SERVICES FOR SPACECRAFT CONCEPTUAL DESIGN VIA AN ENTERPRISE SERVICE BUS

Ariana C. Caetano de Souza[1] e Walter Abrahão dos Santos[2]

National Institute for Space Research - INPE
São José dos Campos, SP, Brazil
[1]acaetano@img.com.br e [2]walter@dss.inpe.br

Abstract: *As space system designs are growing more complex and market demands bigger, technical issues related to concept development become more important and difficult. Moreover, due to pressure on cost and time, many space projects nowadays demand distribution as they undergo their development divided among several project partners, sometimes temporally and geographically separated hampering the information exchange and adding risk. The conceptual design is a key phase that maps client needs to product use functions and is where functional architecture (and sometimes the physical architecture) is decided upon. Typically, the design specifications and constraints impose a heavy burden on systems-of-systems engineering. This paper shows how some processes of the space mission conceptual phase can be standardized and automated using a Service-Oriented Architecture (SOA) paradigm. By employing an enterprise bus service (ESB), named SpaceESB, applications become distributed and its reuse promoted. The final configuration is being integrated to an application which consumes these services.*

Keywords: *Conceptual Design, Systems Engineering, Service-Oriented Architecture, Enterprise Service Bus*

1 Introduction

Recent interest in collaborative environments for promoting cloud-enabled systems engineering has been risen in the literature (Ross *et al.*, 2006). Space systems engineering demands this type of environment as it requires essentially multidisciplinary expertise ranging from onboard computing to launcher vehicle interfacing. Nevertheless sometimes experts may not be temporally and / or geographically co-located.

Difference factors such as time zone, language barriers, numbering and units of measurement conventions and different IT platforms may all promote an adverse effect on the project timeline and costs.

In parallel, there is a growing demand generated by clients of services provided by space systems which increases pressure for good space system engineering on delivery time reduction, higher quality and performance as well as cost reduction (Bandecchi *et al.*, 1999).

As space system designs are growing more complex, technical issues related to conceptual development, also referred as Phase A, become more important and difficult. Conceptual design maps client needs to a functional architecture, and sometimes, even the physical architecture.

The design specifications and constraints typically demand good systems-of-systems engineering and particularly on requirements engineering which drives the whole system's life cycle. Henceforth, taking suitable decisions at this project phase ends up paying dividends on schedule, performance, cost and risk. Therefore agility and flexibility in the execution of intrinsic processes in the Conceptual Design are necessary.

This highly-coupled and distributed scenario can be tackled thanks to the availability of open standards to reduce barriers between different platforms as well as infrastructure to support the offer of services. This abstraction is possible via SOA and web services (Erl, 2004) which are being adopted to make business processes more efficient and effective. These technologies contribute to shape business processes, create solutions, design, develop and deliver services.

This work uses the concept of ESB - Enterprise Service Bus (Shin, 2007), here in a customized version called SpaceESB that exposes a set of budgeting services to support the conceptual design phase of a satellite project. This allows systems engineering partners to consume services regardless of platforms. For illustration, the set of budgeting services here considered comprises of three simple services expressed by their WSDL (Web Services

Definition Language) interfaces (Shin, 2007) to the SpaceESB. Ultimately, this realizes a prototype environment for collaborative and distributed space systems engineering.

This paper is organized into the following. Section 2 presents a brief introduction to SOA, web services and, the concept of SpaceESB. The creation of budgeting services is briefly described in section 3. The implementation of services is shown in section 4. Finally, section 5 closes it with conclusions.

2 Background

As organizations grow they acquire an ever-increasing number of applications distributed across a number of departments and sites as well as sharing information between these applications. SOA arose out of this need to allow intercommunication between applications (Ross *et al.*, 2006) as it entails developing loosely coupled interfaces to applications (services). By combining these services, it is possible to develop adhoc applications (mash-ups) as required.

2.1 Service-Oriented Architecture

SOA is a software architecture style whose basic principle advocates that the functionalities implemented by the applications should be delivered as services enabling greater reuse, redundancy reduction and, greater efficiency in maintenance (Erl, 2004). Frequently these services are organized through a "service bus" (Shin, 2007) that provides interfaces, or contracts, accessible via web services or another form of communication between applications. The SOA, shown generically in Fig. 1, is based on principles of distributed computing paradigm and uses the request / reply to establish the communication between client systems and the systems that implement those services (Wu *et al.*, 2008).

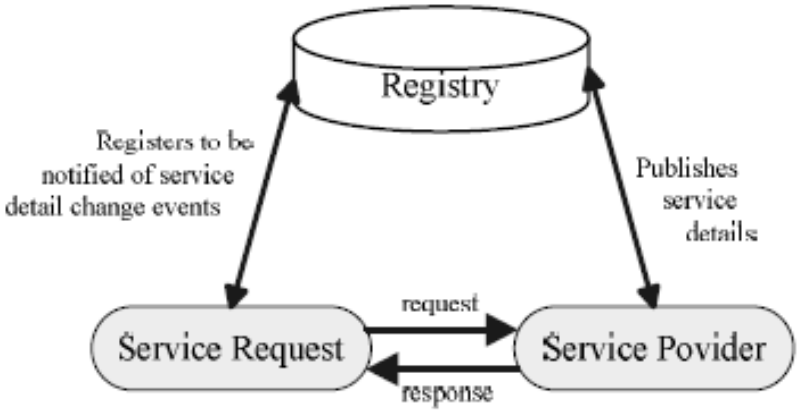

Figure 1. Basic Schematic of a SOA Environment.

SOA is based on three fundamental elements, namely:

- Services - represent the business functions and are technology-independent;
- Interoperability - obtained through an infrastructure called Enterprise Service Bus - ESB, which combines the distributed services on different technologies and platforms;
- Loose coupling - is related to reduction of dependencies between systems, hence it is possible to achieve greater fault-tolerance and reduction on change impacts.

Therefore, SOA is not a COTS tool but rather an approach that when applied to business provides greater scalability, flexibility and growth, also serving as a bridge between business and ICT – Information and Communication Technology.

2.2 Web Services

The web services are one of the possible ways of realizing the SOA abstraction as they can integrate applications through messages based on XML (eXtensible Markup Language) usually employing HTTP (Hypertext Transfer Protocol).

There are many types of web services, but the most known and well-used are: RPC, WS-* (WSDL / SOAP) and REST (Moro *et al.*, 2009). The WS-* architecture is the mostly used nowadays. The main specifications for this architecture are SOAP (Simple Object Access Protocol), WSDL e UDDI (Universal Description, Discovery and Integration) (Erl, 2004).

The service description, and how to access it, is defined by a document that uses the WSDL language. The registration of a service and its location is defined by the UDDI. Thence for service publication and consumption, as Fig. 2 illustrates, client and service can exchange messages enveloped on the SOAP protocol and transmit data represented in XML.

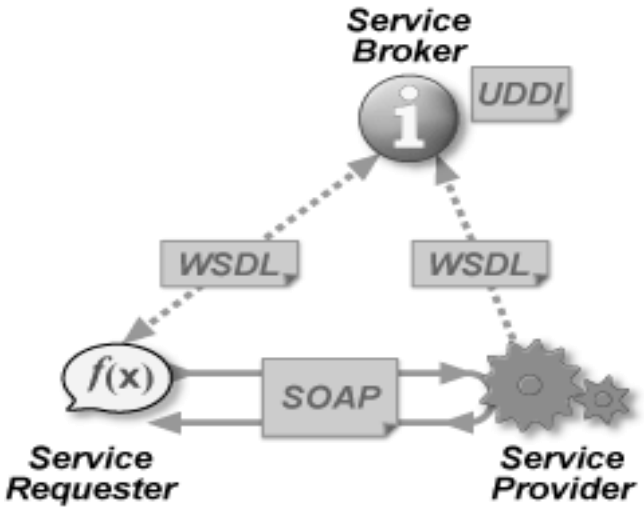

Figure 2. Basic scheme of web services and its messaging system.

2.3 Enterprise Service Bus

An ESB is an infrastructure that enables high interoperability between services, allowing exposed services to be consumed by clients. Its layout is sketched in Fig. 3. The main responsibilities of an ESB are: (1) Data transformation; (2) (Intelligent) routing; (3) Dealing with reliability; (4) Service management; (5) Monitoring and logging; (6) Providing connectivity and; (7) Dealing with security, among others.

As some systems engineering activities for space systems are becoming more complex and distributed, due to issues previously mentioned, one possible solution for this problem is the creation of a dedicated ESB, the SpaceESB, for this part of the project life cycle. Thereby, information related to the conceptual design would be available as services which could be invoked from anywhere regardless of the underlying platform. This increases team communication as impacts on decisions taken by one team are rapidly evaluated by the other team members involved thus precluding misconceptions.

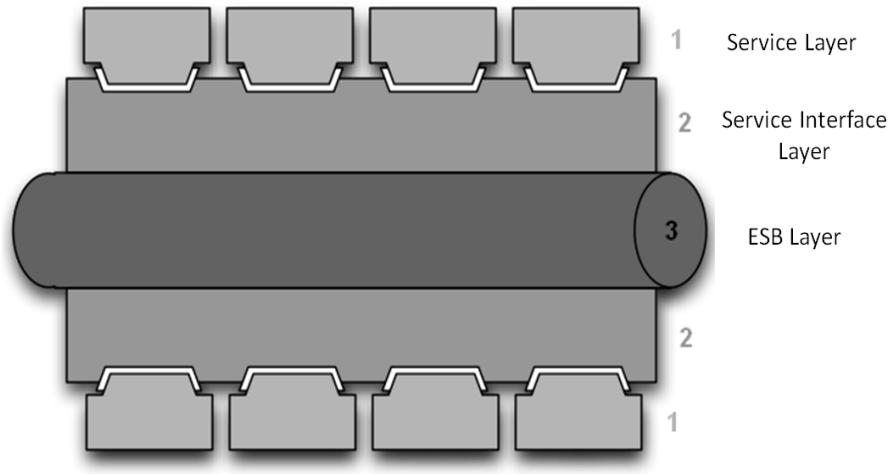

Figure 3. Generic structure of an Enterprise Service Bus (Longo, 2009).

3 Budgeting Service Automation for Satellite Conceptual Design

One of the first steps to a SOA deployment is to identify which activities will be provided as services independently executed and generating well defined results. The business functionalities are mapped into these services and they are composed by parts, named operations, which encapsulates the complexity of existing business rules.

Typically in a satellite conceptual design, suitable architectures are sought that successfully matches mission objectives (Larson and Wertz, 1991) just like any space design exploration. As a simple illustration of activities at this early phase, this paper presents the budgets required to evaluate the amount of thermistors, number of direct commands for critical onboard functionalities and, solar panel area. Briefly discussed, each one of these estimates is implemented as a service based upon systems engineering business rules. A simplified set of business rules from (Hardwick, 2009) were used to program the web service logic.

3.1 Budgeting the Number of Thermistors

The satellite thermal subsystem is generally responsible for keeping all on-board equipment in a suitable operating temperature range at all times.

This subsystem may have two alternatives for control strategies, either a passive or active control, see (Larson and Wertz, 1991) for details. As shown in Fig. 4, the passive strategy employs materials, coatings, surface finishing, thermal properties whereas the active approach employs thermostatic heaters, heat pipes, and pumping systems with radiators and heat exchange.

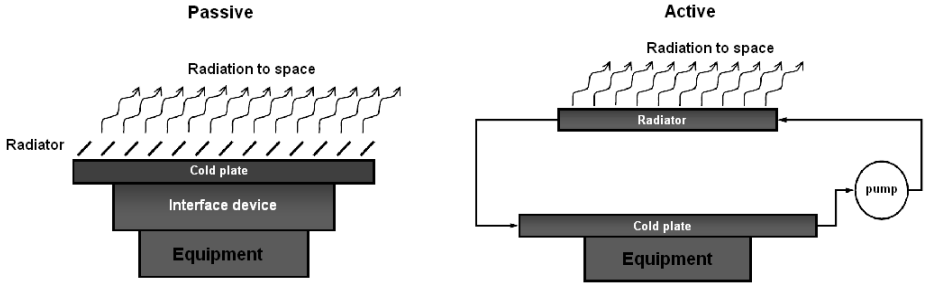

Figure 4. Some active and passive thermal control schemes (Souza, 2008).

One key component in the Thermal Control Subsystem is the sensing element, usually a thermistor. During the conceptual phase it is necessary to estimate the required number of thermistors assigned for each satellite

component that needs thermal monitoring. This budget affects other coupled subsystems design for example, on-board processing, power and harnessing.

3.2 Budgeting the Number of Direct Command

Direct Commands are not processed and executed by the on-board computer, but are directly hard-coded and executed. These are mainly dedicated for mission-critical command execution like the following equipment items: On-board computers, batteries and telecommunications transceivers. This particular budget affects the satellite reliability and other coupled subsystems design like on-board processing, commuunications and harnessing, for example.

3.3 Budgeting the Solar Panel Area

The power subsystem, see (Larson and Wertz, 1991) for details, is sketched in Fig. 5 and is mainly responsible for (1) the provision of power mainly acquired from solar panels; (2) energy storage generally via batteries; (3) power conditioning, conversion, regulation and distribution to all subsystems and equipments.

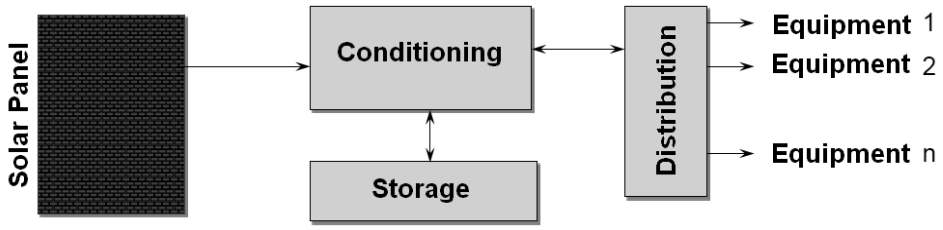

Figure 5. A typical block diagram for the power subsystem (Souza, 2008).

In order to meet the power demands, the required solar panel area needs to be properly evaluated and an extra margin for power generation should be taken into account allowing for battery charge and discharge cycles. This particular budget is highly vital and it affects almost all subsystems design.

4 Implementation of the Budgeting Web Services

The realization of a SOA abstraction needs a set of tools and a development environment in order to create its business models, its services and, its overall derived elements.

After the service definition and operations models, one has to perform the transition of the models to computer systems, in this case to a web service. For this work, the programming language chosen is Java and the development environment chosen is Netbeans, a free integrated development environment (IDE) which has SOA plug-ins resources for service creation, orchestration and development of composite applications.

The first implementation step is the creation of the web services just mentioned previously. At the end of the web services creation, a WSDL file is also generated. The out coming WSDL file contains details on the service description, its operations and the access conditions.

The second implementation step is realizing the underlying service orchestration. The web service creation only implements the service operations, but it is essential to implement also the operation execution flow.

The orchestration is responsible for defining the services invocation sequence and conditions. BPEL (Business Process Execution Language) is the chosen strategy for service orchestration which application is depicted in Fig. 6. BPEL is an XML dialect that defines services interactions.

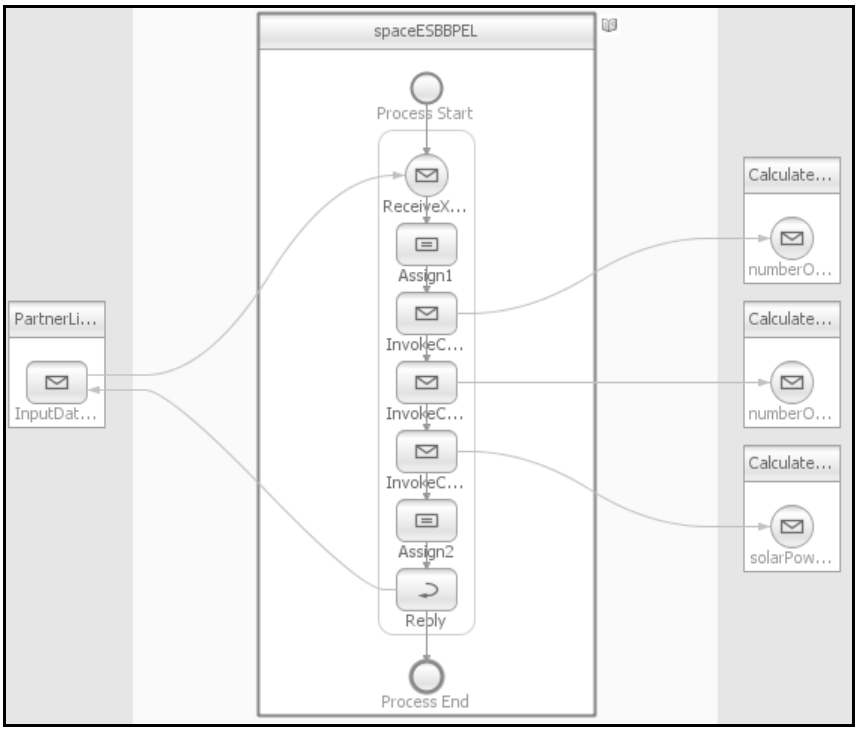

Figure 6. BPEL diagram for three budgeting web services.

At the right-hand side of Fig. 6 lies the service responsible for performing budgeting calculations while at the left-hand side, the requesting WSDL file. The center part displays the workflow taken for all 3 service consumptions. In this case, the execution flow is rather simple: after the data input from the requesting WSDL file, all services are called and after their completion, results are sent back to the requestor.

The next implementation step is the creation of the composite application which is an application built by combining multiple services. It consists of functionalities drawn from several different sources within a Service-Oriented Architecture (SOA). The components may be individual web services, selected functions from within other applications, or entire systems whose outputs have been packaged as web services (often legacy systems). Figure 7 shows the composite application created, containing the BPEL module, the services and how services will be exposed.

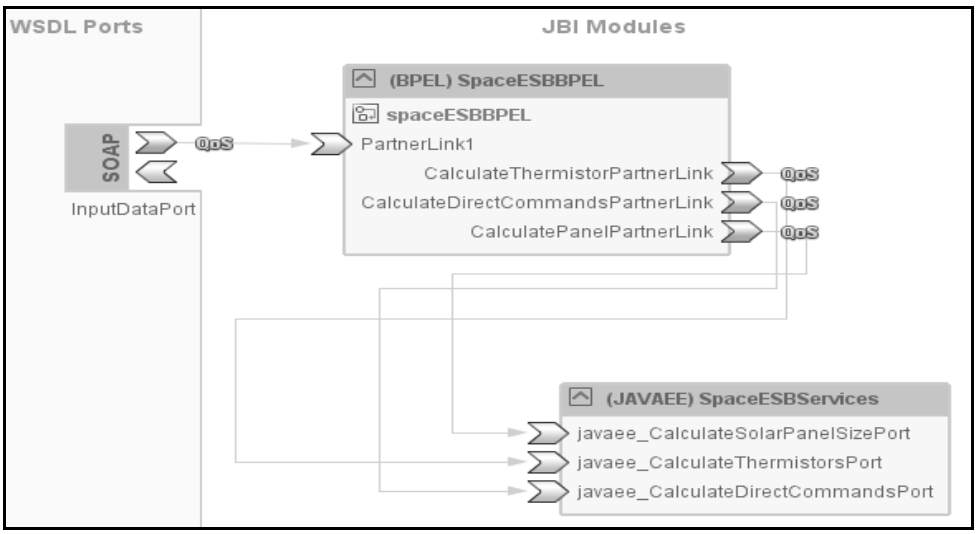

Figure 7. Composite Application for web service exposition.

The last implementation stage is the realization of the composite application deployment to the ESB Server which makes it available for all project partners who may need those functionalities. This work employs the GlassFish ESB (Somekh, 2008) which is a binary distribution of the Open ESB components, a project which implements an ESB runtime using Java Business Integration as its foundation. The GlassFish ESB, as his name suggests, comes with a GlassFish server, an open source application server which provides support for hosts' services such as security and transaction management and it is also distributed with Netbeans IDE.

In order to test the operation of SpaceESB, a prototype of a Java application has been created to simulate stubs and drivers from the SatBudgets application (Leonor, 2010) as shown in Figure 8. SatBudgets uses XML Metadata Interchange (XMI) information exchange between a satellite SysML model and its initial requirements budgetings via a rule-based knowledge database captured from satellite subsystems experts. Furthermore, in order to enable collaboration among project partners, web services are directly invoked by SatBudgets via the SpaceESB.

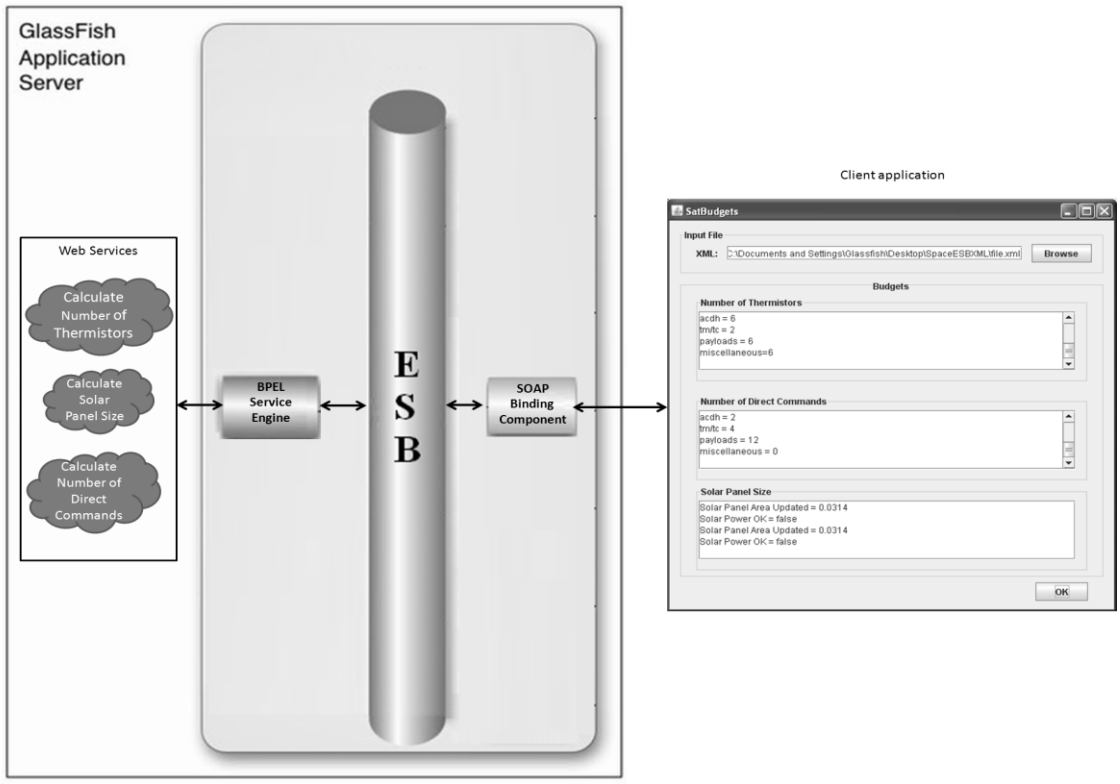

Figure 8. Scheme for service execution using the SpaceESB integrated to a dummy SatBudgets application.

The execution scheme shown in Figure 8 allows the SatBudgets application to communicate with the SpaceESB which in turn accesses the services and returns the budget to the requester. All data exchange is performed via XML files wrapped inside SOAP messages. The creation methodology used here can be extended to any other services. A trial is being currently planned that will populate the SpaceESB adding more functionalities and running in a fully distributed environment.

5 Conclusions

Due to its complex and multi disciplinary nature, the flawless design, construction and launching of space systems is quite challenging. Experts from many fields are required to cooperate to ensure project success. Sometimes these tasks are geographically and/or temporally distributed demanding a cloud enabled environment.

In order to support the conceptual design phase of a satellite project, this paper shows how a set of three simple budgeting services can be coupled to a customized Enterprise Service Bus, named SpaceESB. The selected

services where all related to early budgeting activities concerned with thermal control, commandability and power generation which affects also other subsystems.

The mapping of a project activity to a Java service which implements these functionalities have been briefly presented using BPEL and a WSDL file was generated which describes the interface to the SpaceESB. This approach is general enough for the inclusion of new services populating the SpaceESB. The implementation has been done using only open source tools on all its steps.

This approach allows process automation which at the space conceptual design is desirable as it allows project partners to actively participate in the decision-making process providing greater agility and flexibility and helping to cope with pressures on costs and time in this systems domain.

References

Bandecchi M., Melton B.; Ongaro F. Concurrent Engineering Applied to Space Mission Assessment and Design. In ESA Bulletin, 1999.

BPEL tutorial. SearcSOA.com Available at http://searchsoa.techtarget.com/generic/0,295582,sid26_gci1330911_mem1,00.html?ShortReg=1&mboxConv=searchSOA_RegActivate_Submit& Access on: Jun., 14th 2010.

Erl, T. Service Oriented Architecture: A Field Guide to Integrating XML and Web Services, The Prentice Hall, 2004.

Hardwick, A. A Webservice-based collaborative systems engineering for space applications. Submitted MSc Report to the School of Computer Science and Information Systems, Birkbeck College, University of London, 2009.

Larson W. J. Wertz J. R. Space Mission Analysis and Design, third edition, Microcosm Press, 2004.

Leonor, B.B.F A Knowledge-Based and Model-Driven Requirements Engineering Approach to Conceptual Satellite Design (in Portuguese). INPE MSc Report, 2010.

Longo J. S. C. SOA with OpenESB, Sun Microsystems. 2009. Available at <http://api.ning.com/files/Sn6i83r-auY8oBXNNYPn61ZsEMqGd3sU-421*dPFgL0a9aX*MnF0rHU0uZzcxDGOxOq8WCIDX5pk6XchShvtTKV*Y-L3lfEc/SOAcomOpenESB.pdf>. Access on: Oct., 10th 2010.

Moro, T. D., Dorneles, C. F., Rebonatto, M. T. Web services WS-* versus Web Services REST. 2009. Available at <http://www.upf.br/computacao/images/stories/TCs/arquivos_20092/Tharcis_Dal_Moro.pdf>. Access on: Nov., 11th 2010.

Ross, J. and et al. Enterprise Architecture as Strategy: Creating a Foundation for Business Execution. Cambridge, Harvard Business School Press, 2006.

Shin S. SOA using Open ESB, BPEL, and NetBeans. Sun Microsystems. 2007. Available at <http://docs.huihoo.com/openesb/soabpelopenesb.pdf>. Access on: Nov., 13th 2010.

Somekh M. Introduction to Sun GlassFish Enterprise Service Bus. Software Infrastructure Group Sun Microsystems. Screencast. 2008. Available at <http://developers.sun.com/docs/javacaps/tutorials/screencasts/Composite/start.html>. Access on: Jan., 15th 2011.

Souza, P.N. Introductory Course on Satellite Technology. Lecture Notes. Slides – INPE, 2008.

Wu Q.; Zhou C., Jing T. Research on SOA based framework of collaborative design system for complicated product. In International Conference on Computer Science and Software Engineering, 2008.