# CloudSatCD: An e-Engineering Solution to Space Systems Concept Design

Alex Barbosa Bastos
*Space Engineering and Technology PG*
*National Space Research Institute – INPE*
*São José dos Campos, Brazil*

Walter Abrahão dos Santos
*Computing and Applied Mathematics Lab*
*National Space Research Institute - INPE*
*São José dos Campos, Brazil*

## Abstract

*This paper proposes a cloud infrastructure for Space Systems e-Engineering utilizing the Software as a Service (SaaS) model. The system exposes services tailored to the concept design phase of satellite manufacturing. The solution relies on two previous works named SatBudgets and SpaceESB for budgeting automation and service wrapping, respectively. It is expected that the environment will promote collaboration among project stakeholders and empower horizontal communications in the systems engineering team. Basically, a cloud infrastructure, a groupware and all the ancillary software components are open source and configurable so that flexibility and scalability may be attained.*

## 1. Introduction

Systems engineering is an interdisciplinary field focused on how to best engineer complex projects and manage them throughout its life cycle with proper exploitation of processes, tools and, concepts from correlated areas [1].

The application of this approach to space systems justifies as it is necessary to deal with the intrinsic complex details of satellites projects which presents high criticality levels depending on the final mission. This is particularly true for one of the project phases, namely the concept design [2].

As a modern trend, development of many complex products, like satellites, are becoming distributed with project teams geographically and temporally dispersed. These topological barriers only add complexity to the project development efforts. Problems due to lack of coordination, inefficient communications infrastructure, time-zone and cultural differences, lack of interaction and collaboration are mentioned as the key factors for project delays and even its failure [3]. Needless to mention, that these same facts are present in ordinary program offices where teams are collocated or nearby the main organization.

In this context, e-Engineering as a web collaborative environment for engineering efforts may contribute with the scenario previously mentioned.

A collaborative environment is basically a virtual environment where all project stakeholders, even distributed ones, are able to negotiate, rapidly interact and discuss, share knowledge and work together in a particular task [4].

In this work, we extend two previous endeavors towards providing an e-engineering environment, an application named SatBudgets [5] and its enterprise service bus extension named SpaceESB [6] both targeting also the concept design phase of satellite projects. The extension is undertaken by cloud-enabling its core functionalities and providing e-engineering services in order to help mainly systems engineers, managers and major decision makers.

This paper is organized as follows. Section 2 deals with general resources and technologies that will be employed. Section 3 presents briefly two previous works which will be mainly added to the proposed solution. A prototype of the collaborative environment, in high level abstraction, and a brief description of the implementation are presented in Section 4. Finally, Section 5 closes the work with comments and conclusions.

## 2. Background

The service-oriented computing (SOC) is an emerging paradigm for the development and integration of distributed applications, which aims to support interoperability among components running on different platforms [7].

This paradigm promotes the idea of assembling application components into a network of services that can be loosely coupled to create dynamic business processes, flexible and agile applications that harness organizations and computing platforms. With this any piece of code and any application component implemented in a system can be reused and transformed into available network services [8]. We now discuss some of the current solutions available to deal with these systems.

### 2.1. Groupware

Groupware systems are based on computing and telecommunications technologies that help groups of users to perform an activity [8]. Basically it is a kind of software to support cooperative work.

This type of system allows many users to work simultaneously on a same project. While a single user system focuses on the individual, a groupware focuses on the group itself. When working on a project where communication is essential between collaborators, groupware contributes to a faster and clearer information exchange. Its goal is to enable multiple perspectives, knowledge and assistance in solving problems together, in order to save time and money in coordinating group work [9].

Generally, groupware tools help people in these aspects [9]: Communication - helps people share information; Coordination - helps people to coordinate their personal papers with others; and Collaboration - helps people work together.

Collaborative systems allow people to communicate with each other, regardless of location and time. Likewise, they facilitate informal communication, automation and faster task completion, allowing the work team to accomplish tasks more effectively, efficiently and creatively [10].

Some common characteristics of Groupware solutions are [11]: E-mail and voice mail; calendars and schedules - allow automatic checking of the electronic calendar of team members in search of vacant hours; agendas and online programming - program and notify members about scheduled meetings; projects and activities management; knowledge management - by organizing and sharing administrative information in forms, training data libraries, articles, intranets, extranets, etc.; mapping of staff expertise areas; discussion forums; videoconferencing.

## 2.2. Cloud Computing

Usually deployment models for computing services characterize the management and availability of the computational resources, the provision of services to end-users as well as categorizing these users into different classes [12].

A private cloud is usually operated by a single organization and it could be managed by the same organization or by a third company. It could be hosted on its own local data center or in a remote one. This model tends to empower organizations with more control over consumers than those using public clouds [12]. The techniques used to provide these features can be at network management level, service provider configurations and, authentication and authorization techniques [13]. This option is crucial for organizations that need data security and confidentiality.

Nevertheless, the cloud computing model does not mean that clients always have to cross the Internet to get content. A local cloud, also known as presentation virtualization, ignores the service provider component and allows them to manage content in their own data center. With a local cloud it is possible to keep a server in-house and clients may connect directly to it. This allows the offering of computing resources to users just as utilities [14].

Therefore, it is a cloud infrastructure inside the organization´s doors [15] and creates an environment capable of executing and implementing cloud computing features such as virtualization and networked layered services. At the same time, more rigid policies and requirements in security, latencies, SLA (Service Level Agreement) and, existing data center resources. This provides efficiency and flexibility while reducing costs.

A private cloud offers many of the benefits from public cloud environment. The difference is that in the private cloud, data and process are managed inside the organization with no constraints in network bandwidth, security and legal requirements that a public service may entail. Besides this, private cloud services offer the provider and the users major infrastructure control, improved security and resilience since user and network access is restricted [16].

## 2.3. Software-as-a Service - SaaS

There are many service deployment models [12] but the one adopted here is suitable to the organization type, the control over the computing environment, level of abstraction for its utilization.

SaaS is a type of cloud service where applications of interest to many users are now hosted in the cloud as an alternative to local processing and storage. The applications are offered as services by providers and accessed by customers using applications like a web browser. All control and network management, operating systems, servers and storage is done by the service provider [17]. SaaS is simply a cloud provider that conveys a particular piece of software you want to use on their servers.

Unlike the model where the software is locally installed, access to the SaaS application is made over the Internet via browser. The provider centrally and transparently hosts the application and data deploying patches and updates as required [17].

## 2.4. Service-Oriented Computing - SOC

The SOC paradigm promotes the idea of assembling components application from a network of services where any piece of code and any component application, deployed in a system, can be reused and transformed into an available network service [8].

The service-oriented concept is based on the composition of applications by discovering and invoking network services ready to perform a task which is not dependent on a particular programming language and operating system. This empowers

 

organizations to expose their core competencies in a network, e.g., the Internet, using standard languages and protocols, as well as self-descriptive interfaces. SOA is the key vision to achieving this and is a logical way to design a software system to provide services for both end-user applications and for other services in a distributed network via detectable published interfaces [8].

Basically, SOA allows building systems from loosely coupled components (services, generally web) which may be combined dynamically [18]. Furthermore, SOA defines 3 main concepts:

- Roles: a Service Provider, a Service Requestor and, a Service Registry;
- Operations: Publish, Find and, Bind
- Standards: SOAP (Communications), WSDL (Service interface description) and, UDDI (Service Discovery).

## 3. Previous Works – SatBudgets and SpaceESB

The activities related to satellite concept design involves various domain disciplines which need to be highly woven so that a proposed solution meets mission requirements. At project inception, satellite architects shall be capable to allocate requirements for each component part and they shall be coherent with the entire satellite for correct emergent systems behavior [19].

Usually, many tasks during concept design are developed using computer-aided applications some distributed throughout an organization. Integration of these legacy systems and the steady market growth for collaborative environments turns service wrapping of these applications somewhat attractive allowing project stakeholders to interact and optimize project management.

Next, we refer to two previous computer applications, SatBudgets and SpaceESB, that together and cloud-enabled will constitute the proposed solution.

### 3.1. SatBudgets

SatBudgets is an evolving tool to help on automated budgeting for the concept design of satellite projects [5]. Currently it deals with some mechanical, electrical, on-board computing budgets and relies on the satellite model written in the System Modeling Language, SysML [1] as a design integration entity. The SysML block diagram is, so far, the mostly exploited diagram where the satellite architecture is described, block relationships defined and, subsystems packaged.

This software, whose workflow is depicted at Figure 1, extracts design parametrical information from the SysML model which expresses the satellite architecture being coalesced from contributions of the specialists in the various systems engineering disciplines. Furthermore, it has a rule-based mechanism which enforces businesses rules from this domain to be followed and allows them to be triggered as necessary for budgeting processing. Finally, it generates outputs as reports and graphs. This allows for some design space explorations (DSE) to be undertaken in order to arrive in a most-viable design.

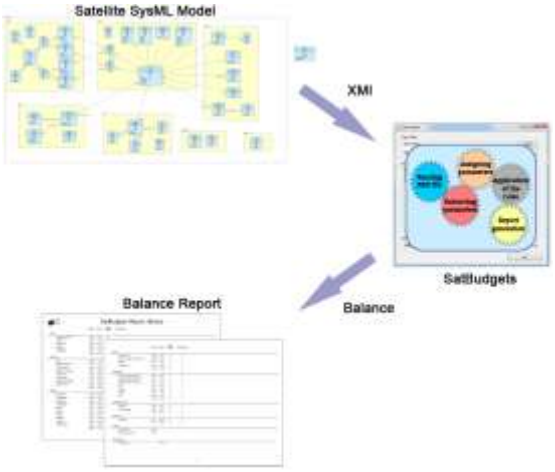


**Figure 1. A typical SatBudgets workflow [5]**

The tool workflow starts with the satellite SysML model, processing through a chain of stages and final output generation. Each budget functionality is offered via the tool GUI interface so that systems engineer can select which design concerns are being explored. Since budgeting has been encapsulated it is now suitable for being service wrapped and services being offered in a bus which leads us to the next topic.

### 3.2. SpaceESB

SpaceESB is an Enterprise Service Bus (ESB) extension for reaching budgeting which may be remotely invoked using a service-oriented architecture (SOA) paradigm [6]. As shown in Figure 4, the ESB enables distributed access to some budget services and allows some engineering workflow to be supported. This is particularly interesting since we presently have some external project partners and this solution can deal transparently with design coupling issues. Hence, the approach taken facilitates towards information sharing and integration, platform independence and computer systems interoperability. Additionally, it allows that the impact of any major design decisions taken by the design team be rapidly evaluated and incorporated tending to reduce misconceptions risks [6].

Data exchange is performed through SOAP (Simple Object Access Protocol) messages, which

are encapsulated in XML (eXtensible Markup Language). This allows client systems to consume services by communicating with the SpaceESB, which in turn generates a service request and returns to the consumer.

The SpaceESB has an interface to the SatBudgets; see Figure 2 for details, so that it can access remote budgets via web services through SpaceESB. This creates a collaborative environment independent of platform and implementing some of the activities pertaining to the processes of concept design phase, typically, system level trades and concept evaluations.

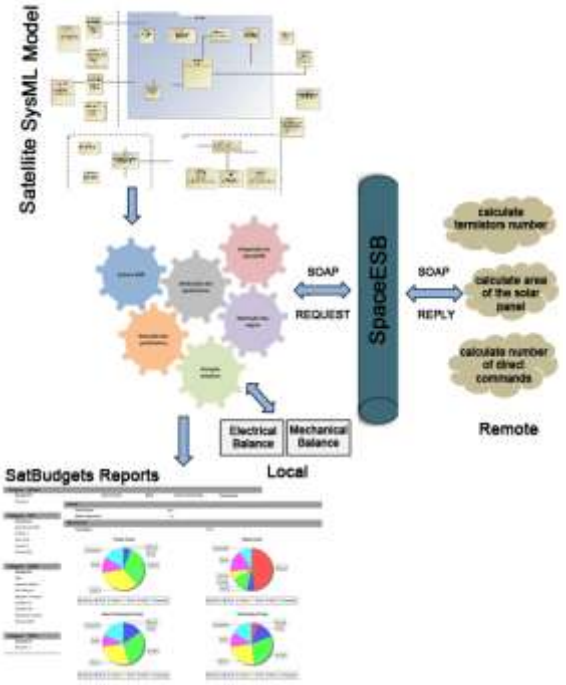


**Figure 2. SpaceESB integration and execution into SatBudgets [6].**

## 4. The CloudSatCD solution

The SOC paradigm can be applied to an e-Engineering infrastructure for supporting the concept design phase of space systems and may contribute to expedite internal processes execution normally performed during satellite projects.

The proposed solution, depicted in Figure 3, is based on cloud computing working in a SaaS scheme and it uses a services-based approach to offer a combination of infrastructure computing, management, storage and software services transparent to the user [8].

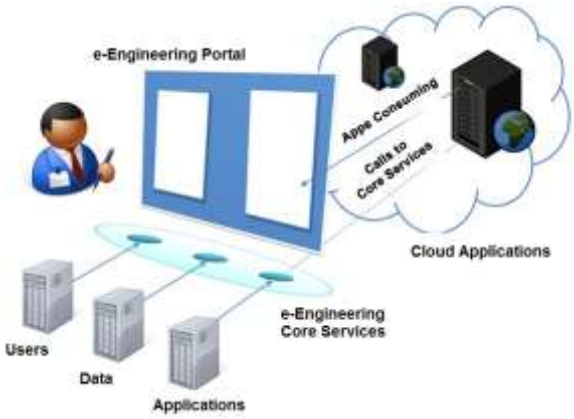


**Figure 3. An e-Engineering extension to a standard e-Science infrastructure**

### 4.1. CloudSatCD Layered Structure

The envisaged collaborative environment has a layered e-infrastructure in a high level of abstraction. Each one of the layers is described hereafter.

The Client Layer is the layer where the users are, i.e. consumers of infrastructure services provided by e-Engineering.

The Access Layer is the layer that implements an interaction interface between users and the underlying architecture, with resources provided by it. The access layer shall perform the following tasks: user´s authentication; work monitoring; data submission; data visualizations; user´s collaboration; access to groupware services; and access to SatBudgets as a service via SpaceESB. This layer basically embodies the web portal which provides access to service options.

The Mediation Layer is a layer responsible for abstracting the presence of SpaceESB services provided by the application and it shall perform the following assignments: interaction between applications and SatBudgets; interaction between applications and the actual SpaceESB; storage, search, discovery, registration and execution of web services; parameter configurations and web service executions. This layer acts like a middleware containing a set of tools that can be used for the creation of infrastructures and distributed applications. The set of tools allows safe sharing of services, databases and other tools. Some of these tasks are being investigated to be mapped directly into the ESB infrastructure itself as it can transparently perform mediation and service discovery.

The Service Layer is composed of services provided by the e-infrastructure, basically SatBudgets and a Groupware.

The Data Layer is a repository that contains the user data and services as well as any other information and documents related to the concept

design of satellites. This forms the SaaS cloud computing infrastructure.

The proposed architecture, sketched at Figure 4, is designed initially to work either in an Intranet and/or Extranet open to collaboration with partners as security and confidentiality demands are present in eventual project information. Nevertheless, the structure shall be able to communicate with other networks for discovery of services provided by them.

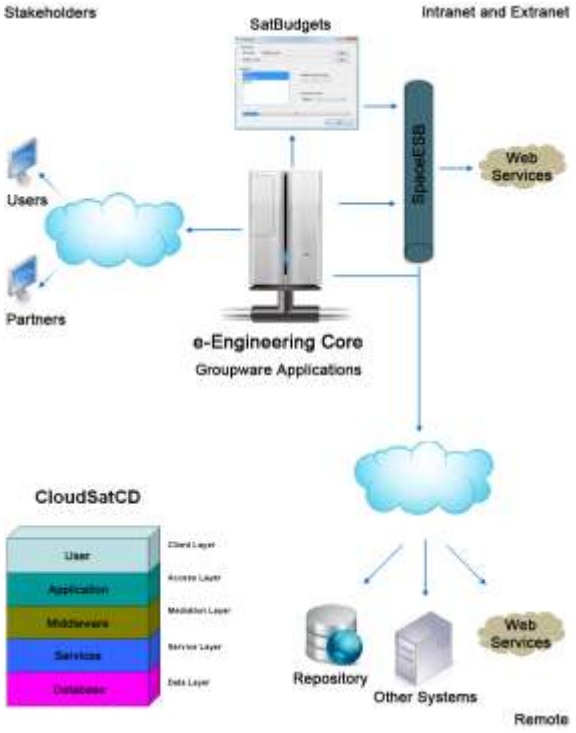


**Figure 4. SatBudgets/SpaceESB integration to the e-Engineering cloud.**

## 4.2. The CloudSatCD Implementation

The e-Engineering structure mashes up the applications SatBudgets, SpaceESB and a groupware, as indicated in Figure 5, making every cloud resource available to users via a web browser. This solution allows a variety of clients ranging from desktop to mobile terminals like smart phones. By having a groupware capability, project teams members can also have access to a variety of applications residing in the cloud.

For the CloudSatCD architectural realization, open-source tools and solutions are being prioritized so that they can be modified and adapted if required. For the cloud implementation it is being used, OpenNebula, a software for building and managing data centers and cloud-based infrastructures. In the OpenNebula environment, a front-end main node is responsible for performing infrastructure managing tasks, creation, deliver and management of virtual machines which are executed in the cluster nodes. The communication between them employs SSH (Secure Shell) inside a private network while using a shared file system, normally NFS (Network File System) [20].


**Figure 5. Mashup ecosystem for the e-Engineering infrastructure**

The cluster nodes run hypervisors which are responsible for the virtualization and the life cycle control of the virtualized systems. This extends the existing computing capabilities so it can run the various guests´ operating systems in many virtual machines simultaneously. By using OneVBox, one can have access to VirtualBox but a variety of hypervisors comes with OpenNebula supporting KVM, Xen, VMware.

All the infrastructure management can be performed either from a prompt line or from the OpenNebula Sunstone [20], a web-enabled operations Center residing at the front-end node.

The tools responsible for providing the e-Engineering applications are located in the created virtual machines since CloudSatCD only provides software to final users and OpenNebula is an IaaS (Infrastructure as a Service) facility. Thence, as depicted in Figure 6, the systems responsible for the application software are hosted in the VMs instantiated by the cluster nodes managed by the front-end node.

The front-end and cluster nodes run a Linux Ubuntu Server, an open source operating system which has a high flexibility and integration level to OpenNebula, besides easy install and management in a complete server.

In the created VMs there is a central core with the groupware application, EGroupware, which is an online multi-purpose collaboration tool in a single platform. EGroupware provides information sharing, address bookmarks, calendar, agenda, event and project tasks management, online file and document management server, data exchange and synchronization, e-mail, content management to name a few. Among its many features, some are highlighted: (1) setting usage constraints according to specific groups (admin, project manager, user, etc), it is possible to allow application executions and manage access permissions at user, group and

application levels by using ACL (Access Control List); (2) backup and retrieval, synchronization with mobile devices via SyncML and protocols CalDAV/CardDAV/GroupDAV; (3) synchronization of data for desktop clients; (4) client-to-desktop connection to file manager via WebDAV; (5) file attachment and configuration management; (6) document organization in different categories and status; (7) navigation in the project tree and project Gantt chart visualization at a desired level; (8) project status in different views (time, costs); (9) creation and addition of hierarchical project and definition of goals and constraints; (10) information publication in calendar and agenda; (11) tracking system, file and wiki management; (12) private or group chatting; among others.

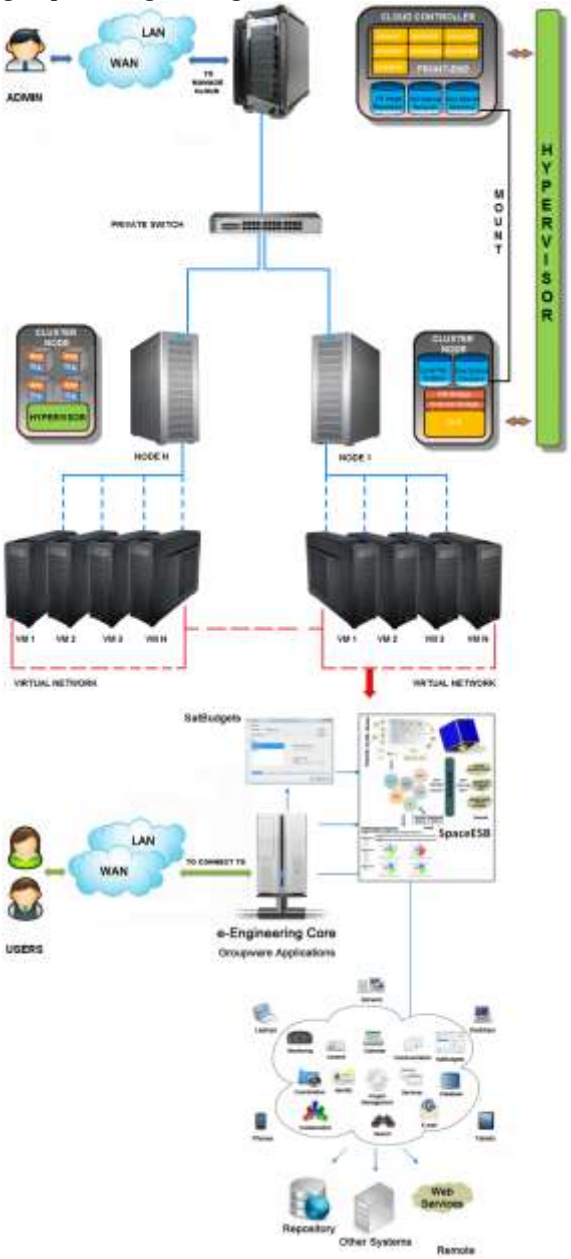


**Figure 6. Overview of CloudSatCD infrastructure and application**

The main operating system that runs in the VMs is FreeBSD, an open source Unix-based operating system, from the BSD family (Berkeley Software Distribution). Since FreeBSD is robust, stable and flexible, it is commonly used in main servers where it provides portability, efficiency, simplified software system administration, optimized update processes. FreeBSD is supporting the architecture underneath the E-groupware,

The infrastructure back-end and the repository responsible for data layer is MySQL, which is accessed by applications written in Java and PHP.

The Glassfish application Server is responsible for the Java applications while the Apache web Server is responsible for those written in PHP including EGroupware install and repository management via PHPMyAdmin, a good web interface for managing MySQL.

Even though the Apache web Server is the direct responsible for the PHP applications, the Glassfish application server is capable of running Java code mimicking PHP by using Quercus which is a bridge component between Java and PHP. This allows Java services integration to PHP scripts automatically taking into account its resources and enabling a mixed approach to web and service applications.

Both servers employ security at encrypted transport and data exchange levels via SSL (Secure Sockets Layer). These are two developing platforms for services offered to the e-Engineering infrastructure gathering best of both worlds.

The integration of these tools to the legacy applications, SatBudgets and SpaceESB is performed via SOA using the WSDL interfaces. The user has access to legacy applications via the Egroupware web page which works as a mashup container for all applications.

In order to deal with web services, one can use the Apache ServiceMix suite, which is a powerful runtime platform that can be used for building integrated applications acting like an ESB and middleware. Additionally, CloudSatCD has a SOA client capable of accessing web services via browser.

External users and project partners can access CloudSatCD as long as they have granted authorization. This employs an encrypted IP tunnel over a VPN (Virtual Private Network) which is managed by the OpenVPN software. This is done when a tunnel is set between servers and the PPTP protocol (Point-to-Point Tunneling Protocol) and eventual clients are managed since nowadays almost all IP-enabled devices natively support PPTP being only configuration needed.

In this way, e-engineering users have access to a web interface which provides them a variety of resources and applications for satellite conceptual design in a cloud computing scheme with no need to install or keep an infrastructure. Additionally, it is expected that it will:

- Facilitate the generation of web interfaces tailored for the satellite projects as well as the interaction of systems engineers and project partners via the web;
- Encourage the development of systems engineering through an appropriate infrastructure that meets the initial satellite projects and
- Promote project reuse and data sharing and facilitate the management of research projects.

Currently, the CloudSatCD is being prepared for a concept proof and field trials in order to collect usage and workload statistics. As new services are planned to be added, architectural flexibility, scalability and customization for the currently solution will also be evaluated so that it will evolve little by little as needed.

## 5. Conclusions

Computer systems have become a vital part of modern research and engineering practices. They are also contributing to the design of complex products such as satellites where project design tends to become spatial and temporally distributed.

The paradigm of distributed computing makes information ubiquitous to a great number of devices and software platforms through network services. By using cloud computing with open source tools and open standards, it is possible to increase flexibility and reduce costs at the same time.

In this paper, we proposed a SaaS cloud-based environment which basically integrates a groupware to two legacy applications, SatBudgets and SpaceESB. This sets up an e-Engineering environment focused initially on helping in the concept design, a key initial phase of a satellite project.

The complete solution is based in a 5-layered system which hides the cloud complexity creating the e-engineering abstraction. This allows users to transparently interact with a mash-up web portal at clients' machine. Afterwards, it communicates over the network with the other layers that may be resident in other federated machines. The user can command executions that require processing on remote machines, retrieval of objects in data repositories, collections, and tools needed for statistical processing and / or visualization.

The described e-Engineering environment is currently under final assembly and tests where main results will be reported shortly validating hopefully its initial purposes.

## 6. References

[1] S. Friedenthal, A. Moore, R. Steiner, "OMG systems modeling language (OMG SysML) tutorial", Systems Engineering for the Planet, INCOSE, 2008.

[2] W.J. Larson, J.R. Wertz, "Space Mission Analysis and Design", 3rd edition, 1999.

[3] J. Grudin, "Why CSCW applications fail: problems in the design and evaluation of organization of organizational interfaces". Proceedings of the 1988 ACM conference on Computer-supported cooperative work. ACM Press New York, NY, USA. pp. 85–93, 1988.

[4] G. Booch and A. W. Brown, "Collaborative development environments", 2002.

[5] W. A. Dos-Santos, B.B.F. Leonor, S. Stephany, "A Knowledge-Based Approach to Deal with Conceptual Satellite Design". Lecture Notes in Computer Science - Conceptual Modeling - ER 2009, v. 1, p. 487–500, 2009.

[6] A.C.C. Souza, W.A. Dos-Santos, "Automating Services for Spacecraft Concept design via an Enterprise Service Bus", Proceedings of the ISPE Concurrent Engineering - CE2011, Massachusetts Institute of Technology – MIT, USA, Jul. 2011.

[7] N.M. Josuttis, "SOA in Practice: The Art of Distributed System Design" USA, O'Reilly, 352 p., 2007.

[8] M. P. Papazoglou, P. Traverso, S. Dustdar and F Leymman, "Service-oriented computing: state of the art and research challenges", 2007.

[9] J. Rama and J. Bishop, "Survey and comparison of CSCW groupware applications". University of Pretoria, South Africa. 2006. in press.

[10] N. Leavitt, "Is Cloud Computing Really Ready for Prime Time?", IEEE Computer Society Magazine, pp. 15-20, Jan. 2009.

[11] R. Baecker, "Readings in Groupware and Computer-Supported Cooperative Work", San Mateo CA: Morgan Kauffmann, 1993.

[12] W. Jansen and T. Grance, "Guidelines on Security and Privacy in Public Cloud Comnputing". NIST - National Institute of Standards and Technology, Special Publication 800-144, 2011.

[13] F. R. C. Sousa, L.O. Moreira, J. C. Machado, "Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios". Universidade Federal do Ceará – UFC, In ERCEMAPI, 2009.

[14] A. T. Velte, T. J. Velte, R. Elsenpeter, "Cloud Computing: Computação em Nuvem – Uma abordagem prática". Ed. Alta Books, 352 p., Rio de Janeiro, 2011.

[15] M. Koopmans, "Refining the cloud stack". A whitepaper on first lessons learned in the NEON project, 2010.

 

[16] L. Badger, T. Grance, R. Patt_Conner, J. Voas, “Cloud Computing Synopsis and Recommendations”. Recommendations of the National Institute of Standards and Technology. NIST - National Institute of Standards and Technology, Special Publication 800-144, 2011.

[17] M. Veras, “Cloud Computing: Nova arquitetura de TI”. Ed. Brasport, 240p., Rio de Janeiro, 2012.

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

[19] P. N. de Souza, "Curso introdutório em tecnologia de satélites. Engenharia e tecnologia espaciais: ETE/CSE – Instituto Nacional de Pesquisas Espaciais", in portuguese, unpublished, 2011.

[20] C12G Labs S.L., "Designing and Installing your Cloud Infrastructure OpenNebula 3.6". Rev20120710. 27p., 2012.

 