doi: 10.5028/jatm.v8i2.553

Design of a Nanosatellite Ground Monitoring and Control Software – a Case Study

Freddy Alexander Díaz González[1], Pablo Roberto Pinzón Cabrera[1], Claudio Marcel Hernández Calderón[1]

ABSTRACT: The growing countries that have carried out the development of CubeSat missions for academic purposes do not offer aerospace engineering programs at their universities. This causes difficulties for traditional engineers upon the formal use of different standards and frameworks for aerospace development, such as the European Cooperation for Space Standardization and Space Mission Analysis and Design . One way in which traditional software engineers can easily understand the structure of an aerospace framework, in order to apply it on the development of CubeSat mission software parts, is comparing its most important elements in relation to the elements suggested by a more familiar method. In this paper, we present a hybrid framework between the ECSS-E-ST-40C standard and the Rational Unified Process, which can be used by traditional software engineers as a guide model for the development of software elements in academic nanosatellite missions. The model integrates the processes and documentation suggested by the ECSS-E-ST-40C with the disciplines, workflows and artifacts suggested in Rational Unified Process. This simplifies the structure of ECSS-E-ST-40C and allows traditional software engineers to easily understand its work elements. The paper describes as study case the implementation of the hybrid model in the analysis and design of ground monitoring and control software for the Libertad-2 satellite mission, which is currently being developed by the Universidad Sergio Arboleda in Colombia.

KEYWORDS: CubeSat, European Cooperation for Space Standardization, Rational Unified Process, Monitoring, Graphical User Interface, 3D.

INTRODUCTION

The development of academic satellites projects, mostly based on the CubeSat standard, has become the best alternative for those countries who want to start research and development on aerospace themes (Woellert *et al.* 2011). The feasibility of using Commercial off-the-shelf (COTS) components for the development of satellite modules and the possibility to be secondary cargo on the rocket launchers, through modules as the Poli-Picosatellite Orbital Deployer (P-POD), allow the launch of small satellites (1 – 10 kg) (Buchen 2014) with shoestring budgets (Woellert *et al.* 2011); however, due to the lack of experience that growing countries have in research and development of satellite technology, their universities do not have yet specific programs to train aerospace engineers; for this reason, academic satellite missions such as CubeSat should be developed with local engineers, trained in conventional development of hardware (HW) and software (SW), with assistance of aerospace experts trained in the United States (US) and Europe (Nader *et al.* 2014). The only country in Latin America that provides aerospace engineering programs is Brazil, at the Universidade Federal de Minas Gerais, Universidade de Brasília and the Universidade do Vale do Paraíba (Armellini *et al.* 2012). The nearest approaches to aerospace programs in Latin America are aeronautical engineering degrees offered in Ecuador, Argentina, Peru, among others. Currently in Colombia, satellite missions aim to generate experience in the development of aerospace technology in medium term, through the training of professionals with the concept of learning by doing (Villamil and Mayorga 2013), and the creation of university programs focused on the aerospace theme; however, as this process

1.Universidad Sergio Arboleda – Escuela de Ciencias Exactas e Ingeniería – Bogotá – Colombia.

Author for correspondence: Freddy Alexander Díaz González | Universidad Sergio Arboleda – Escuela de Ciencias Exactas e Ingeniería | Calle 74 14-14 | Zip Code: 110221201 – Bogotá – Colombia | Email: freddy.diaz@correo.usa.edu.co

Received: 9/15/2015 | **Accepted:** 02/15/2016

matures, local engineers are responsible for the development of the first satellite missions.

Due to the peculiarity that exists in the operational context of a satellite system, it is necessary the use of development methodologies that address the characteristics of aerospace type, to guide the different stages of analysis, design, implementation, validation and operation of each subsystem of the mission. The most commonly used are those proposed by the National Aeronautics and Space Administration (NASA), called Space Mission Analysis and Design (SMAD) (Webster and Corcoran 2007; Puschell 2011), and standards of the European Cooperation for Space Standardization (ECSS), proposed by the European Space Agency (ESA) (Raphael *et al.* 2014). Nevertheless, these models are oversized for small academic missions as CubeSat ones. One way to provide local engineers with the appropriation of aerospace concepts suggested by these models is by identifying relationships between common elements of conventional development methodologies known by local engineers and aerospace development models, such as milestones, steps, artifacts, suggested activities and life cycles.

Given that SMAD and ECSS have a similar structure, a comparative analysis of the life cycle phases, milestones, artifacts and activities defined, among ECSS and the Rational Unified Process (RUP), is presented in this study (Ramos *et al.* 2010). RUP is one of the best known SW development methodologies and it is commonly used by local engineers. The proposed analysis results in a hybrid framework, whereby local engineers can appropriate and guide the development process of the different academic satellite SW elements. To validate the proposed hybrid model, the design of the ground monitoring and control SW for Libertad-2 nanosatellite – as the main component of the Mission Control Center (MCC) – is taken as a study case.

The paper is presented in the following order: first, a description of Libertad-2 satellite mission; the importance of developing SW as a fundamental element of a nanosatellital system is explained; the presentation of SMAD and ECSS standards is done; ECSS-E-ST-40C standard is specifically described; and a presentation of commercial SW development methodologies is made. Then, the life cycles of RUP and ECSS-E-ST-40C are compared, the milestones, artifacts and activities are shown and the hybrid model is structured. Finally, the design process of the ground monitoring and control SW for Libertad-2 MCC is explained, applying the proposed model; results and conclusions are presented, and future research arises.

METHODOLOGY

The Libertad-2 nanosatellite mission of the Universidad Sergio Arboleda, in Colombia, aims to continue with the development of academic satellites, started in 2007 with the launch of the Libertad-1 picosatellite (Llorente and Leguizamón 2014), by developing a 3U CubeSat-type nanosatellite with an Optical Payload (OPL) to carry out remote sensing (RS) on agricultural areas of Colombia. In addition, the Libertad-2 will use an S-Band frequency in order to send the OPL data to the Earth (Díaz *et al.* 2015).

Figure 1 represents an external visual of Libertad-2. The OPL will be located in the left unit, consisting of an embedded system, a multispectral complementary metal-oxide-semiconductor (CMOS)-type sensor, and a lens with approximately 80 ground sample distance (GSD) to obtain images of the Earth's surface from a low Earth orbit (LEO), accompanied by the microstrip antenna, used for S-Band transmission. The on-board computer (OBC) and electrical power system (EPS) embedded systems will be placed on the central unit, as well as the HW of the UHF/VHF and S-Band radios. At the moment, Libertad-2 satellite mission of the Universidad Sergio Arboleda is in Preliminary Design Review (PDR) phase of the ECSS model, and functional prototypes and some flying prototypes of the OBC, OPL, EPS subsystems and microstrip antenna have been developed (Díaz *et al.* 2015).

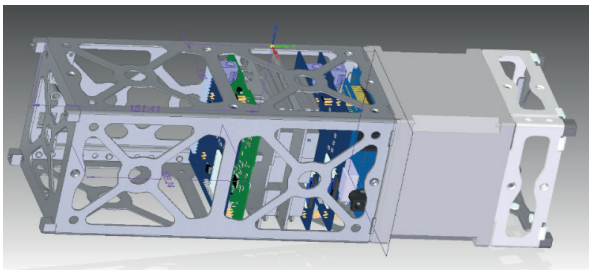

Figure 1. Partial view of the Libertad-2 satellite structural design.

SW development is an activity of great importance in the life cycle of nanosatellite mission, because both the nanosatellite spacecraft (SC) and the MCC should integrate parts of SW applications, fitted to the operational requirements of each mission. This makes it difficult to use standard or commercial SW in the nanosatellite subsystems (Sand *et al.* 2013). These software pieces in conjunction with HW devices should fulfill the main flight operations. In the case of a CubeSat-type SC,

embedded software components with real time subsystem control are needed (Díaz *et al.* 2014); in the case of MCC, desktop software applications are required, whereby it is possible to control the devices of the ground station (GS) communications system (Díaz *et al.* 2015).

The MCC is the ground system that allows mission operators to manage the SC life cycle operations (Funase *et al.* 2007). It consists of a series of SW applications that allow the execution of standard operations, such as the SC trajectory prediction or the SC tracking, by rotating antennas in azimuth and elevation values; and also the management of telecommands and telemetry data through communication with UHF/VHF and S-Band radios. In addition, the MCC can integrate SW applications for processing images captured by the payload, using tools such as MATLAB. Front-end applications are also used to display the operation data (Fischer and Scholtz 2010). Figure 2 shows the example of an MCC for a CubeSat mission.

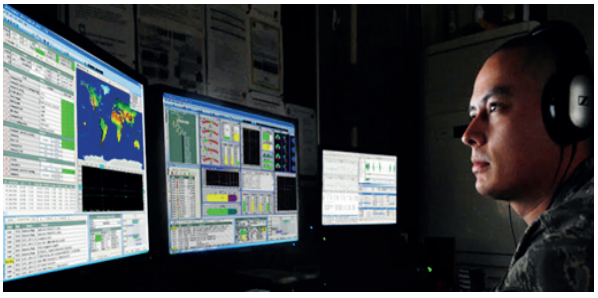

Figure 2. Interaction of operators with the CubeSat through MCC (Braxton Tech 2015).

In the same way that happens with most engineering projects, development of SW applications requires the use of a methodology to structure, plan and control the various tasks and activities of the development process, through a specific framework. The most widely used frameworks for developing SW are the processes of design and implementation, as detailed in traditional or robust methodologies; in this paper, we call these methodologies as commercial methodologies (CM), which use the unified modeling language (UML) as the main tool for modeling (Thüm *et al.* 2014); however, for the development of computer parts as elements of an aerospace system, there are specific methodologies for these aerospace modules. In this paper, we call these methods as aerospace methodologies (AM), which describe the detailed processes for SW development that is required in the SC and MCC (Webster and Corcoran 2007; Puschell 2011; Raphael *et al.* 2014; European Cooperation for Space Standardization 2009a).

SW elements of the SC and MCC should consider non-functional quality requirements such as reliability, robustness and fault tolerance in their operations (Laizans *et al.* 2014). Thus, the processes of analysis, design, implementation and validation of the elements of the satellite system are a fundamental requirement to the mission and should consider the characteristics of the critical systems. To guide the overall life cycle of the mission and the development of all subsystems, there are two main AM or work proposals that can be used: the SMAD, proposed by NASA, and standards of the ECSS, proposed by the ESA.

SMAD proposes a methodology to develop aerospace missions, beginning with the definition of the objectives of the mission, followed by the design, construction, launch and operation of the SC, and ending with the de-orbit. Its main objective is to guide the development of an aerospace system in a quickly, efficiently and cheaply way. Its life cycle is based on the definition given by NASA in NPR7120.5E (Webster and Corcoran 2007) and consists of seven phases, each defined by key decision points (KDP) and a series of major revisions (Puschell 2011). The purpose of each review varies according to the stage and the section of the mission in which one is working. Four groups of processes are identified: (a) Exploration of concepts; (b) Detailed development; (c) Production and deployment; and (d) Operation and support. The Exploration of concepts results in the definition of the mission, its components, cost and schedule. The Detailed development provides a detailed definition of the system, its components and technologies. In the Production and deployment, building of flight HW and SW and the launch of the SC are performed. Finally, Operation and support is related to the day-to-day space mission, maintenance and closing operations.

As SMAD, ECSS proposes a life cycle of seven phases for developing the entire aerospace project. The ECSS standard covers the main aspects related to development of aerospace systems and seeks to offer a user-friendly guide to the procedures in each of the subsystems.

The ECSS-E-ST-40C standard from the engineering branch provides guidance for the development of aerospace SW, in which the mission ground segment SW is included. ECSS suggests six major revisions to develop SW: (a) System Requirements Review (SRR); (b) PDR; (c) Critical Design Review (CDR); (d) Quality Review (QR); (e) Acceptance Review (AR); and (f) Operational Readiness Review (ORR), which must be executed during the proposed life cycle, divided into nine groups of processes, which consider requirements definition activities, design, implementation and validation of SW: (5.2) software related system requirements process;

(5.4) software requirements and architecture design process; (5.5) software design and implementation engineering process; (5.6) software validation process; (5.7) software delivery and acceptance process; (5.8) software verification process; (5.3) software management process; (5.9) software operation process; and (5.10) software maintenance process. Each of these processes is responsible for a group of specific activities of the aerospace software development life cycle and produces a series of specific artifacts that support the documentation (European Cooperation for Space Standardization 2009b).

For the development of commercial SW applications, either embedded type, desktop or web, there are several methods of work, which can be divided into two main categories: agile or robust methodologies.

Agile methodologies are flexible to change, based on close interaction with the customer looking for their feedback; it is not rigid in terms of roles, working groups and offers only the necessary documentation. Methodologies such as Extreme Programming (XP), Scrum, Lean Software Development (LSD), Kanban, Open Unified Process (OpenUP), Rapid Application Development (RAD), among others, are being widely used in recent times for SW application development where a quick-to-market is required (Dingsoyr *et al.* 2012).

These methodologies have characteristics like robustness in terms of activities, iterations, tasks, detailed documentation and constant revisions; they are rigid and inflexible and, in the recent decades, have ceased to be used in the commercial field. However, the RUP, which uses a heavyweight and traditional methodology, is the best known method in SW development (Lopez and Blobel 2009).

The RUP is a full-guided SW engineering methodology with a disciplined approach to assign tasks and responsibilities for the SW development. Its goal is to ensure the production of SW with high -quality attributes that meet the needs of the final user within an established budget and schedule. This is an iterative and incremental process that guides the development of a standard SW product focused on architecture and led by the UML (Jacobson *et al.* 2000).

RUP defines four phases for the development of SW: (a) Inception; (b) Elaboration; (c) Construction; and (d) Transition. Every phase is defined by a milestone; these are: (a) Life cycle Objective Vision; (b) Life cycle Architecture; (c) Initial Operational Capability; and (d) Product Release, which aims to ensure that core workflows (business modeling, requirements, analysis, design, implementation, testing and deployment) evolve evenly over the entire SW life cycle (Jacobson *et al.* 2000).

ACADEMIC SATELLITE MISSIONS

In the last decade, after the start of the CubeSat era, several authors have made approaches to frameworks which help to guide the development process of SW pieces as elements of the SC, identifying activities, defining life cycles, general process structures etc.

These researches (Spangelo *et al.* 2012, 2013; Kaslow *et al.* 2015) explain a fully-structured framework based on Model-Based Systems Engineering (MBSE) and Systems Modeling Language (SysML) to guide the modeling of CubeSat missions, in which both the space segment and the ground segment are considered. Moreover, Kaslow *et al.* 2014 explains how the MBSE model can be used for a simulation of the operation of a CubeSat using MATLAB. Anderson *et al.* (2014) incorporates to it a standard property for the development of business satellites; however, the MBSE model is focused on the system modeling only and does not include the entire development life cycle. Additionally, the framework does not address the ECSS or SMAD methodologies, which guides the development from an aerospace perspective and incorporates neither activities distribution elements nor defined artifacts, such as those presented in RUP, in which management, organization and control processes for the whole software project are included.

Huang *et al.* (2012) explain the use of agile methodologies like SCRUM and XP for development of HW and SW elements of academic satellites and consider as study case the Multi-Mission Bus Demonstrator Project (MBD), conducted by The Johns Hopkins University – Applied Physics Laboratory (JHU – APL). However, any of the characteristics and properties of aerospace development are taken into account.

Asundi and Fitz-Coy (2013) propose a framework for CubeSat mission design based on a systems engineering approach. The model intends to use a flow-down in order to model the requirements, subsystems operations, components interfaces and tasks flows, but also ignores aerospace development processes.

Ziemke *et al.* (2011) present a framework of SW development for embedded systems in small SC, integrating examples of development tools, programming language, design patterns and concepts of Service Oriented Architecture (SOA), aligned with the ECSS processes, but its purpose is to suggest a technical model in terms of design and programming operations and does not propose activities, deliverables or development phases, related to development management.

Pradels *et al.* (2012) explain the process for developing a reusable ground segment to monitor and control the payload of a

small satellite, using a framework that includes the ESCC-E-ST-70 standard and where a life cycle phase with activities and specific documentation based on experience gained from earlier satellite missions is proposed, something similar to the objective of this study; however, its purpose is not to make an approach to commercial SW development methodology through the proposed model in order to facilitate the assimilation by traditional development engineers.

Bürger *et al.* (2014) present the systems engineering process used to develop the first Brazilian CubeSat launch platform called AESP14 and a CubeSat with the same name, in which the activities of the project and a series of documents related to engineering systems, technical specifications and procedures are described. Although the described process takes into account the ECSS standard, an approach is not made to any specific development methodology.

Chaieb *et al.* (2015) explain how is possible to use the System-of-Systems (SoS) and SOS Engineering (SOSE) methodology to design and operate a CubeSat-type SC; however, it is not focused on the SW for SC subsystems and does not propose a life cycle, activities or artifacts to the development.

Mohammad *et al.* (2013) propose a systems engineering framework for the design of CubeSat missions called Open Space Box Modeling (OPEN-SBM), based on the System Requirements Design (SRD) methodology, in which modeling graphic components are used. Nevertheless, no standard for aerospace development is contemplated.

Finally, Brandstätter and Eckl (2009) present a model for compatibility management in the development of SC components, aligned with ECSS-M-30A standard, based on a multidisciplinary approach to systems engineering. The authors propose activities and processes, as well as the usage of UML for the design; however, the model does not focus on the development of SW elements for an academic SC.

On the other hand, some studies support the academic development of satellites as the axis of the formation of undergraduate, master's and doctorate students from different programs and also as the improvement of educational processes within universities. Schilling (2006) explains the manner in which the development of CubeSat picosatellites has been integrated into the curricula of Computer Science and Spacemaster – Master's Program in Space Science and Technology at the University of Würzburg and gives as an example the design of the ground control center of one of the CanSat missions. Bürger *et al.* (2014) describe how the main activities and disciplines of AESP14 mission are coupled with the semester courses of college, using the program to teach undergraduate students systems engineering concepts applicable to an aerospace mission.

The review of the state-of-the-art allows identifying the existence of several models or frameworks applicable to the development of the various subsystems of an academic SC (HW/SW) and their integration, using as a methodological basis recent systems engineering concepts or theories as MBSE. Nonetheless, there are few focused specifically to provide traditional software engineers and ownership of the development process of the SC SW parts. The papers that have this approach contemplate agile methodologies, propose their own frameworks and some of them align the proposed models to aerospace development standards as ECSS. The studies propose from simple organizational structures of mission work teams, flow activities proposals and engineering processes to complex project structures that include life cycles, artifacts, activities, and even introduce concepts of design patterns and SW programming. However, after this literature review, we could not find a study that integrates the features of a commercial SW development methodology as used in RUP with a methodical proposal of a SW development model for aerospace applications as ECSS, allowing the local engineers in developing countries to use a work guide that eases the development of the different SW elements of CubeSat mission, performing specific activities, consolidating specific artifacts, and running a series of revisions during the phases of a certain life cycle.

HYBRID-ACADEMIC-AEROSPACE MODEL FOR SD

The approach of a Hybrid-Academic-Aerospace Model for Software Development (H4ASD) aims to provide traditional engineers with the understanding of the different processes of aerospace engineering that must be taken into account in the development of SW parts, fundamental in an academic SC. A working model is clear if a life cycle consisting of certain phases is defined, in which a set of activities associated with artifacts or documents by each executed activity is proposed.

Aerospace engineering frameworks proposed in SMAD and ECSS include specific processes for the development of the different mission SW applications. Nevertheless, their main feature, as standards, is that they are structured to operate in aerospace missions such as industrial satellites and launches of ferries with crew; so these models are robust and require experienced aerospace engineers who can lead the different development processes. In

this sense, it is inefficient to use them strictly as a methodological guide in the development of academic SC. Although both models, SMAD and ECSS, include the development of the SC along the life cycle of the mission, that is, taking into account the stages of pre-launch, launch and operation, the ECSS_standard is a bit more open to describe activities for the development of SW elements in its ECSS-E-ST-40C document.

At present, agile type methodologies are the most used for the development of commercial SW that must be quick-to-market, since their features are fast teamwork, consolidating only the necessary documentation and obtaining functional SW parts. However, these are not robust enough to be applied in the development of aerospace SW. Due to the characteristics of critical systems, the development of SW parts for aerospace missions needs to be supported in sufficient documentation, on which it would be possible to make traceability, modifications and new requirements that arise during the development of the SC. If the mission does not have the formality and documentation, like as that provides a heavyweight methodology, it would not be possible to develop future missions with the aim of obtaining knowledge and engineering development. That work would be lost. Conversely, RUP model, presenting a detailed description of the entire SW life cycle, specifying each of the iterations, artifacts and activities, both in the structural model of the system and the dynamic one, typical characteristics of a robust model, makes it the best alternative to be applied in the development of aerospace SW. The robustness of RUP is the product of formalism that exists in the documentation process described through its artifacts tree, compared with lightweight (agile) methodologies, where there is much less documentation.

Considering the above, the structuring of H4ASD is based on a "match" between ECSS-E-ST-40C model, as aerospace SW approach, and RUP, as commercial SW approach, as well as on identifying their common, comparable and complementary elements, obtaining a ECSS + RUP model.

The base of the H4ASD is associated with the similarity between both life cycles and workflows or disciplines. This means that both RUP and ECSS-E-ST-40C suggest similar processes and groups of activities that can be linked to form a single life cycle, to guide the development process. Figure 3 shows the overlap between life cycles, intensity or duration of workflows and respective revisions or milestones in the development process in both models. The top part of Fig. 3 shows the seven stages (0, A, B, C, D, E, F) of the ECSS life cycle project to develop a complete aerospace mission in order to show the way in which RUP and SW development project fits within the aerospace project (gray columns).

The characteristics of the RUP's Inception phase are accurately aligned with phase A (Feasibility); Elaboration phase corresponds to the phase B (Preliminary definition), in which

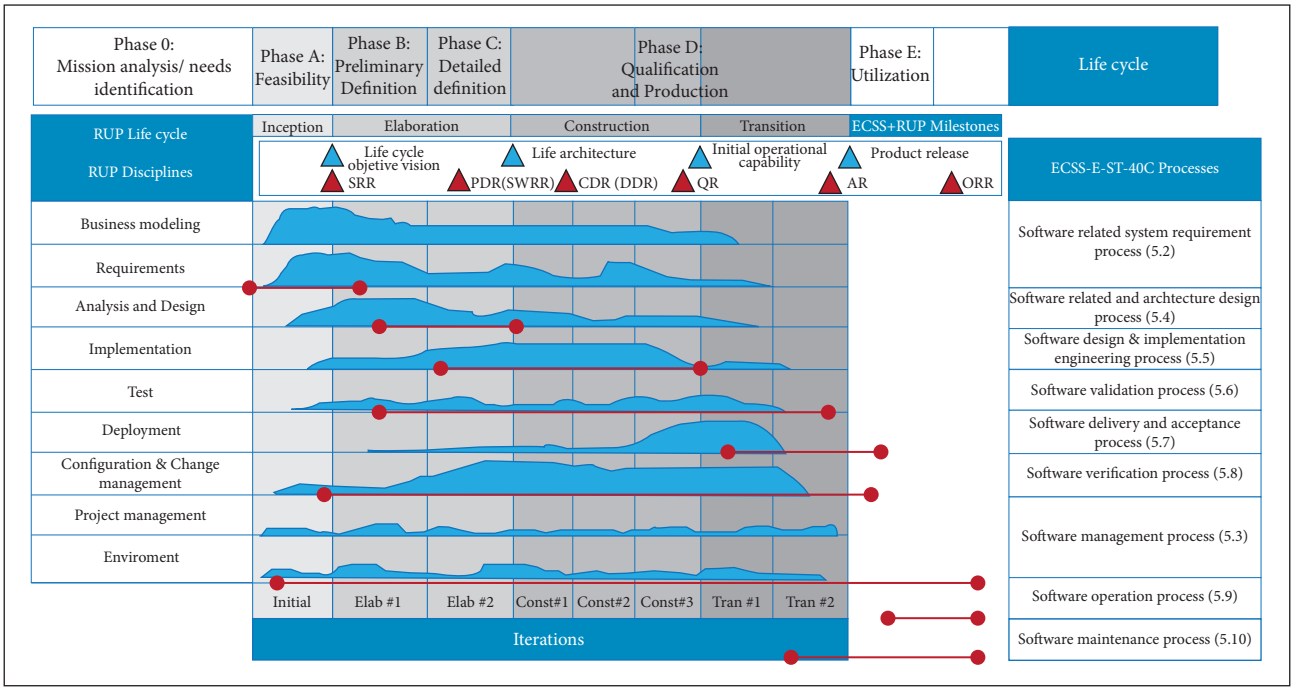

Figure 3. ECSS-E-ST-40C and RUP life cycles match.

the preliminary design of the subsystems is performed, and half of phase C (Detailed definition), in which the final designs are consolidated; the Construction phase runs between the second half of phase C, in which the flying prototypes of subsystems are manufactured and other mission elements are implemented, and the first half of phase D (Qualification and Production), in which the full integrated SC is obtained. Finally, there is the SW Transition, which corresponds to the second half of phase D, in which the SC is released and deployed in orbit. The RUP life cycle does not include phase 0 (mission analysis/needs identification), where the definition of the mission is carried out, and neither phases E (Utilization) and F (Disposal).

The second part of the match is the association between the RUP disciplines (workflows) and ECSS-E-ST-40C processes. The RUP workflows are represented in duration and intensity of activities for each phase (blue curves), considering its iterations, and ECSS represents the processes in extent relative to the total development of SW life cycle (red bars). The RUP business modeling and requirements workflows are similar to the workflow proposed in ECSS-E-ST-40C: 5.2 (software related systems requirement process); the RUP analysis and design workflow is the same process described in 5.4 (software requirements and architecture design process); the RUP implementation workflow is analogous to 5.5 (software design and implementation engineering process); the RUP test workflow corresponds to 5.6 (software validation process); and the RUP deployment workflow aligns to 5.7 (software delivery and acceptance process). The additional RUP disciplines or work-flows also coincide with ECSS-E-ST-40C processes. The configuration and change management flow refers to the 5.8 (software verification process), and the project management and environment flow can be associated to 5.3 (software management process). In software, verification processes are intended to confirm if the application is being made based on the design that was approved. For this, it is necessary that effective management of the artifacts that were generated during the analysis design and implementation phases takes place. Thus, the configuration and change management discipline may be associated with 5.8 (software verification process), because it is responsible for managing the process of release of such artifacts, including documentation. On the other hand, the discipline proposed integration tasks are strongly related to the process of SW verification.

The comparative analysis of the life cycles of both models ends with revisions or milestones that each one suggests. Although the times when they are running are similar — at the end of each phase —, and some reviews collected similar information, ECSS suggests a greater number of revisions. At the end of the Inception phase, RUP suggests the execution of life objective vision, which is aligned with the ECSS System Requirement Review (SRR). At the end of the Elaboration phase, RUP proposes the life cycle architecture, with a similar purpose to the ECSS PDR. After the first iteration of the Construction phase, ECSS intends to conduct the CDR milestone that is not covered in RUP. Then, at the end of this phase, RUP proposes the initial operational capability, which is aligned with the ECSS QR. Finally, at the end of the Transition phase, an AR is executed by ECSS-E-ST-40C and the product release, by RUP. The last revision proposed by ECSS is the ORR, but its implementation is given for phase E of SMAD in which RUP and the development of SW element must be already completed.

RUP defines in general terms a number of activities for each phase of the SW life cycle. The activities are related to one of the different disciplines of the methodology, organized as follows: in the Inception phase, the SW scope must be formulated, the main constraints to technological, operational and administrative levels must be defined, a requirements baseline must be consolidated, business cases (use cases) are planned, a candidate SW architecture is synthesized according to the use cases, and finally the preparation of the application development environment is performed. In the Elaboration phase, RUP suggests planning the Construction phase iterations, refining the most critical use cases for the application, refining the proposed architecture, choosing the components that should be developed, selecting those to be reused and picking those which definitely should be bought; finally, the installation of the development platform must be performed. Then, in the Construction phase, the first thing that should be done is the technological, human and time resources management; after this, in an iterative way, the different components must be coded according to a detailed design, and unit testing must be performed to get the first version of the SW, where a product evaluation must be carried out. Finally, for the Transition phase, RUP suggests executing testing on the entire piece of SW; based on these results, a refinement of the application performance characteristics must be done, correcting bugs and improving usability features. A final version of the user's manuals and general documentation must be obtained, the SW deployment process is done, the user's training is conducted related to the system management, and, finally, an evaluation of the final product in relation to the vision, scope, limitations and requirements baseline is performed.

Meanwhile, ECSS-E-ST-40C defines a set of grouped activities, according to the different processes. For the software related

systems requirement process (5.2), the standard suggests an analysis of the requirements baseline and makes the definition of the verification and SW integration requirements. In the software management process (5.3), the SW life cycle, development phases, the checkpoints and the technology budget must be defined, and the development and coding tools must be selected. In the software requirements and architecture design process (5.4), the following should be done: to define the modeling, integration and verification process, analyze the requirements baseline, and, finally, perform the preliminary design of the SW components. In the software design and implementation engineering process (5.5), the following must be performed: to execute the detailed design of the components, design the internal and external communication SW interfaces according to the system, create algorithms (business flow) of each of the components, code the flows, build applets, integrate the different SW layers, perform functional testing, and, finally, develop the first version of the application manual. For software validation process (5.6), the following must be done: to define components test cases, validate the requirements baseline, run the components test cases, verify SW project documentation, verify the technology budget, and verify the validation regarding the requirements baseline.

In the software delivery and acceptance process (5.7), the following must be accomplished: to create the application installation package, define the installation process, install the SW in the system, verify proper installation, document the installation process in an installation manual, support operators and users, define the acceptance process, perform the acceptance activity, and, finally, verify the results of acceptance activity. The last of the ECSS-E-ST-40C processes aligned within the RUP life cycle is the software verification process (5.8), in which the design of test cases, the source code, the integration of the SW layers, and navigation must be verified — in the case that it is a GS SW element — to, finally, generate the final version of the application user's manual. Figure 4 shows all the activities proposed by ECSS-E-ST-40C (red), grouped by process, and the activities proposed by RUP (blue), grouped by phase.

In addition to the proposed revisions and milestones for each working model, both ECSS-E-ST-40C and RUP suggest a basic structure of artifacts that must be generated during SW development project. Although they differ in technical approach, both documentation structures have similar information artifacts that can be linked to create a hybrid artifacts structure in which the approaches of aerospace engineering and SW engineering are taken into account.

Figure 5 shows an alignment of the various artifacts proposed by each working model, separated according to each of the four phases of RUP. Most RUP artifacts can be associated to ECSS-E-ST-40C documentation, since the information contained is similar. For instance, the View artifact of RUP must contain the description of the needs and expectations of the SW to develop, much like the ECSS Software System Specification (SSS), where

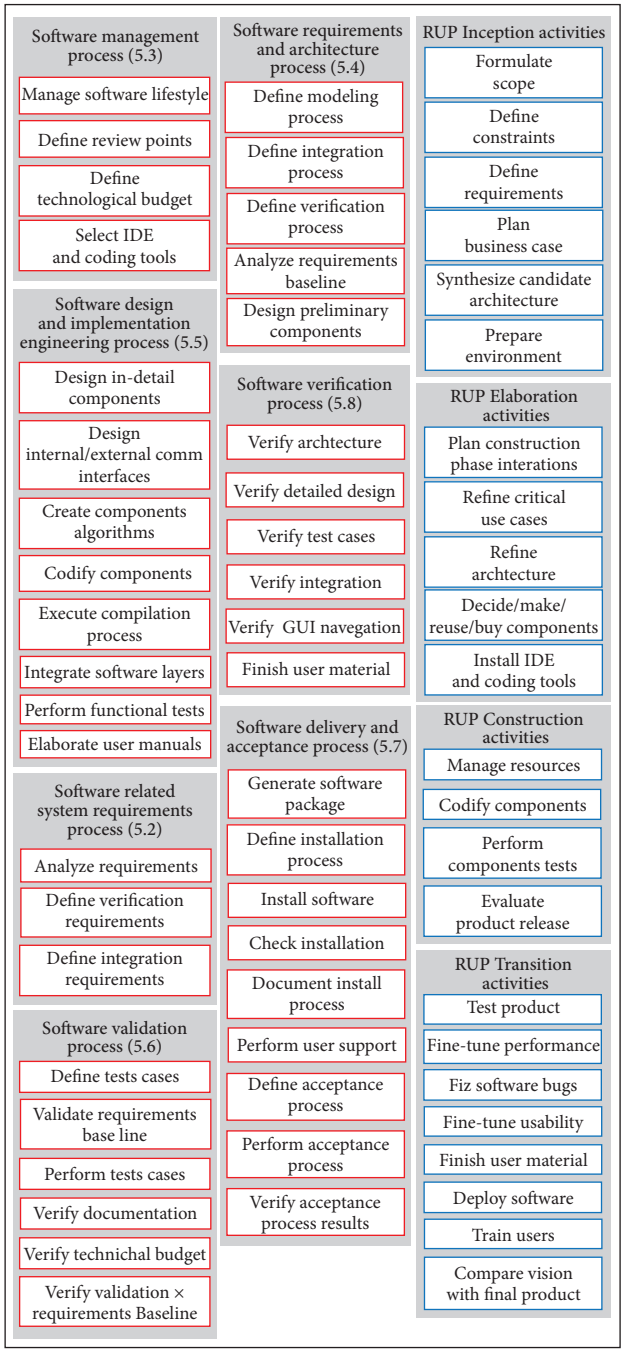

Figure 4. Activities proposed by ECSS-E-ST-40C and RUP.

the definition of the SW is proposed as part of the mission. In the same way, artifacts as RUP Software Architecture Document (SAD) and ECSS Design Definition File (DDF) can be associated.

In other artifacts, the information is the same in both models, and even their names are the same, for instance, the ECSS Software Requirements Specification (SRS), Software Development Plan (SDP) and Software Design Document (SDD), which corresponds to the RUP analysis, design and implementation models. Finally, although most documentation can be linked with each other, some RUP artifacts are so typical of conventional SW development methodology that does not have similar artifacts in ECSS structure, such as SW use case model.

Using the comparison results between the two models, a joint work structure for the development of SW parts as elements of an academic-type aerospace mission is defined, in which considerations as critical system properties and SW engineering conventional concepts are taken into account. The H4ASD proposes to execute sequentially and in an interrelated manner the activities proposed in ECSS-E-ST-40C and RUP, on the basis of the RUP life cycle and flows, to which one can associate the execution of different ECSS-E-ST-40C processes. It also defines a hybrid artifacts structure in which it is possible to consolidate all SW development project information. Next, the H4ASD is presented, describing the coupling of ECSS-E-ST-40C processes with the RUP life cycle and the hybrid activities flows for each phase. Finally, the structure of artifacts and documentation is provided.

The H4ASD life cycle (Fig. 6) seeks to ease the understanding of different aerospace SW development processes suggested by the ECSS-E-ST-40C and the order in which these can be run during the four phases of RUP. In order to see the details of the meaning

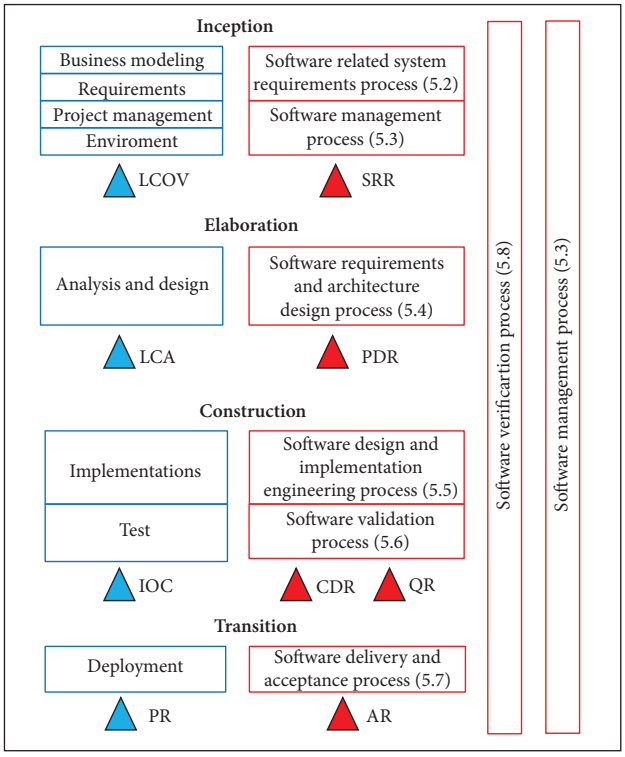

Figure 6. H4ASD Life Cycle.

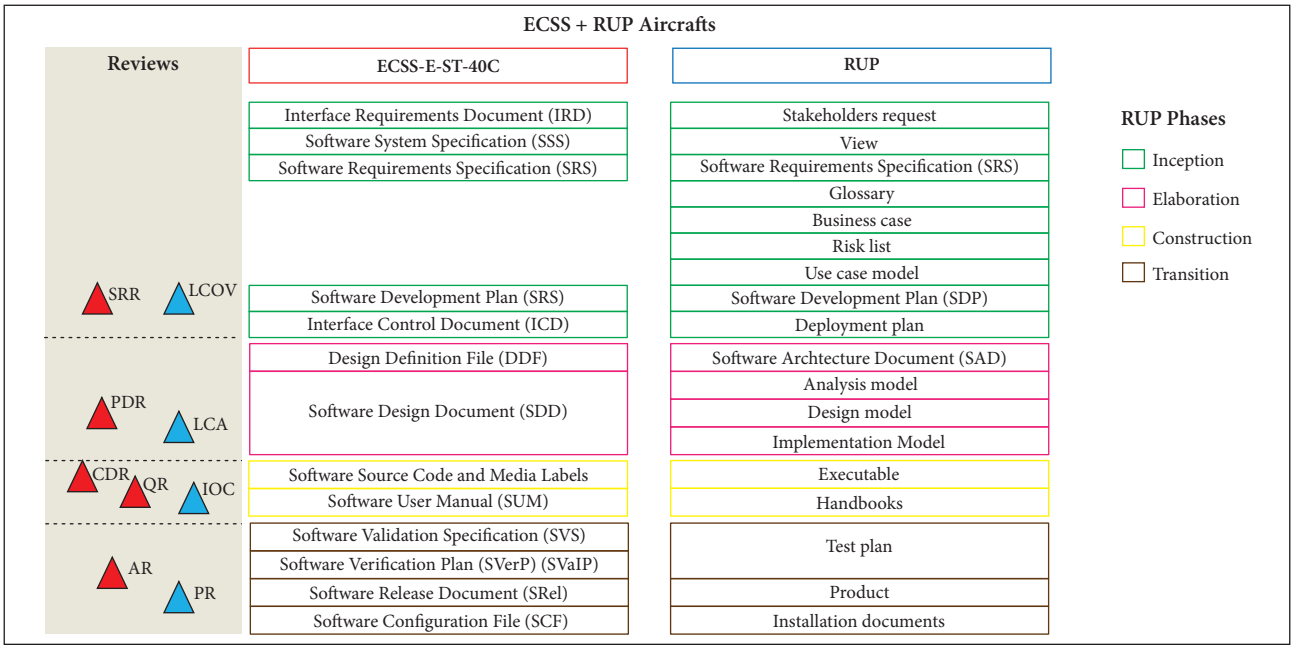

Figure 5. Comparison between ECSS-E-ST-40C and RUP artifacts.

of all abbreviations in the triangles (Fig. 6), consult the ECSS-E-ST-40C document. Considering the results of the comparison between the life cycles of both work methodologies, and not forgetting that both are based on a concept of iterative and incremental execution, the H4ASD life cycle simplifies the working model, combining the disciplines or workflows that are carried out with greater intensity during each phase of RUP with its ECSS-E-ST-40C analogues processes. Although a sequential life cycle is proposed, seeking to facilitate the understanding of the activities order, RUP as a methodology and framework can be supported on models such as waterfall or V model to structure the iterative workflows. Selecting a model (waterfall or V) to map the processes of an aeroespacial framework as ECSS would not be correct because ECSS is also structured as a methodology and not only as a model. The model is only the skeleton of methodology life cycle. Workflows and processes associated with H4ASD RUP phases are described next.

- Inception: In this phase, two ECSS-E-ST-40C processes must be carried out; the (5.2) process, referred to the business modeling and requirements RUP flows, and the (5.3) process, referred to the project management and environment RUP flows.
- Elaboration: At this stage, one should run the ECSS-E-ST-40C (5.4) process; it is aligned with the RUP analysis and design flow.
- Construction: In this phase, two ECSS-E-ST-40C processes must be performed; the (5.5) process, referred to the implementation flow, and the (5.6) process, referred to the RUP test flow.
- Transition: In the last phase, the ECSS-E-ST-40C process to be executed is the (5.7), which is aligned with the RUP deployment flow.

The ECSS-E-ST-40C (5.8) and (5.3) processes are more related to the management, control, and organization of SW development project as a whole; so that the implementation of ECSS-E-ST-40C should be done throughout the RUP life cycle.

H4ASD activities correspond to a link the activities proposed by RUP for each of the four phases, and between activities of the ECSS-E-ST-40C processes which are aligned with these phases, according to the life cycle described in Fig. 5, and the results of the previous comparison. The H4ASD proposed in this paper suggests the following integration activities for each RUP phase:

- In the Inception phase, the activities proposed by RUP and the activities suggested in the ECSS-E-ST-40C (5.3) and (5.2) processes are integrated. The flow begins with the definition of the SW management, its life cycle and the review points (5.3); then, it carries out RUP SW engineering activities, as the definition of the scope and limitations, the first version of the business cases (use cases) and the candidate architecture. The most important activity of this phase is the requirements definition, which must be supported in the activities of (5.2) process, as this not only suggests the definition of the design requirements as RUP does, but also proposes verification requirements and integration requirements definition. The (5.2) process is also very specific in defining the requirements related to Human Machine Interface (HMI), database (DB), real-time processing, security, data formats, among others. Figure 7 shows the flow of activities for H4ASD Inception phase.
- For the Elaboration phase, there are the activities suggested in the ECSS-E-ST-40C (5.3), (5.4), (5.5), (5.6), and (5.8) processes, which are integrated with the RUP activities flow. The main result of the execution of this phase is the detailed design of the SW element, using the results of the first phase. The (5.4) process is the one with a major role in the Elaboration, since it begins with the definition of the modeling, integration, testing and test cases processes. After this first part, it starts the design, the refinement of use cases and the components preliminary design. The (5.5) process runs with the components detailed design using UML, and the first version of program algorithms is created. The final part of this stage is the preparation and installation of the Integrated Development Environment (IDE); (5.6) and (5.8) processes enable a continuous validation of the major design activities in the phase, such as the refinement of the requirements, architecture and components detailed design. Figure 8 shows the flow of activities for the H4ASD Elaboration phase.
- The Construction phase aims to make the implementation and coding of the detailed design, resulting from the Elaboration phase. For this phase, a combination of the activities suggested in the ECSS-E-ST-40C (5.5), (5.6) and (5.8) processes is carried out; however, the (5.5) process is the one with the greatest impact on the workflow. The first activity of this phase is suggested by RUP, related to the initial management of deployment resources (technology/work hours/component/developer). After that, the iterative and incremental coding process formally begins: first with the coding — an activity suggested by both RUP and the (5.5) process —; then, subprograms are compiled, and unit tests suggested by RUP and the (5.6) process

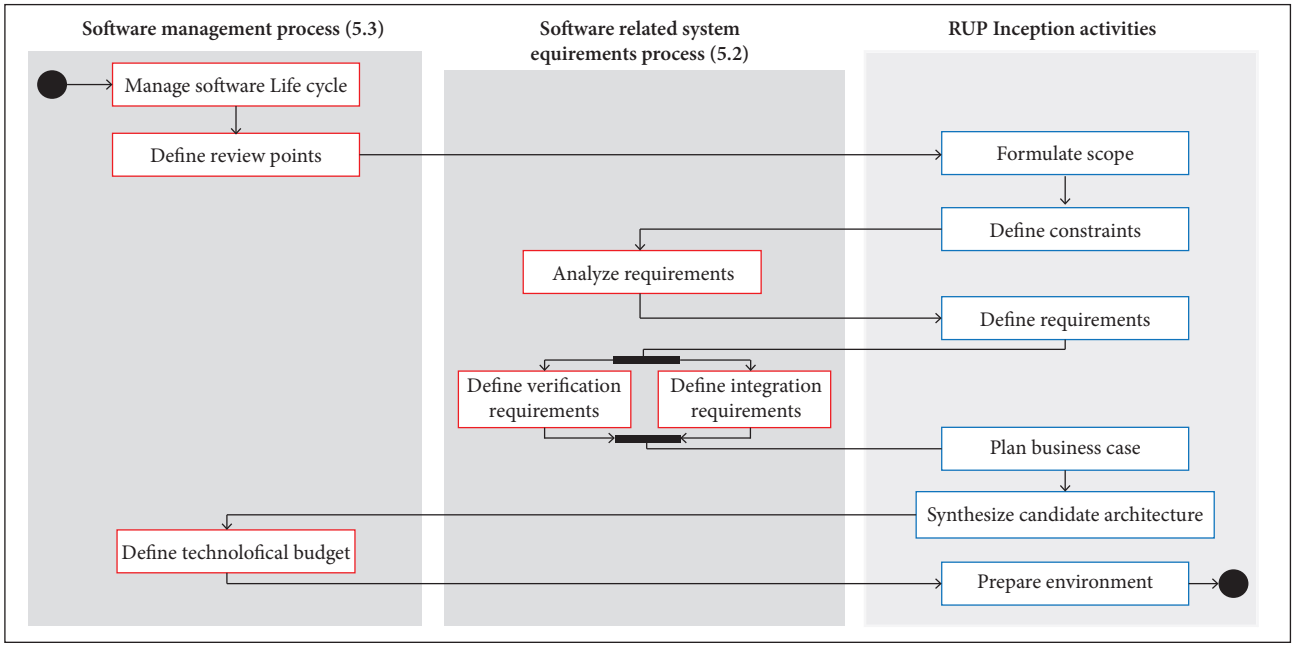

Figure 7. Inception phase activities for the H4ASD.

are executed, SW layers (business/logic, user interfaces and data) are integrated to obtain the first version of the application, general functional tests are performed on the program, and, finally, the first version of the SW manual is consolidated. The (5.8) process is executed in parallel for each iteration of coding, verifying the source code, software versions and the final product. Figure 9 shows the flow of activities for the H4ASD construction phase.

- The Transition phase of the model aims at refining the full version of the application, prior to delivery to the customer, that is, prior to the launch of the SC and the start of the operation phase, and it should run in parallel with the integration phase of the mission. In this last workflow, the activities in ECSS-E-ST-40C (5.6) and (5.8) processes are performed; however, the (5.7) process and the activities of RUP are the most relevant, because they run at first on the activities flow. In the first part, one must perform a refinement of the SW element in different levels, as performance, usability and also fixing of the bugs found in the results of functional tests. Then, the activities suggested by the (5.7) process, related to the installation, training for users — if it is the case of MCC SW —, and acceptance by the mission leaders, must be completed. The (5.8) process is important as the final part of the Transition flow, as the final stage of SW development project, and as complement for the acceptance activity, proposing to verify the documentation, the compliance of technological budget and requirements. The activities of the SW life cycle end with a comparison of the vision outlined at the beginning, in connection with the final product, and the verification of the acceptance process. Figure 10 shows the activities flow for H4ASD Transition phase.

The artifacts structure of the H4ASD shows a connection between the documentation suggested by RUP and ECSS-E-ST-40C throughout the life cycle of the SW element, as seen in Fig. 11.

Most artifacts have the same goal in the presentation of information and can be worked as a single document, such as SRS and the SDP, or the RUP SAD and ECSS DDF.

The structure is complemented by RUP own documentation that is not specified in ECSS, as the glossary and the business case. The use case model is a key-element in the structure of the proposed documentation, because, within a SW engineering process, the use cases are the ones that determine the scope of application within the system and its initial design. For an academic aerospace mission, the use case model should include all operations of the system, whether the embedded SW of a SC subsystem or a MCC SW element. The UML modeling for SW, such as analysis, design, and implementation models, can be included in the SDD. Finally, the specification of executable SW element may be included as part of the Software Release (SRel) and Software Configuration File (SCF) documents. Following the philosophy of H4ASD, it is

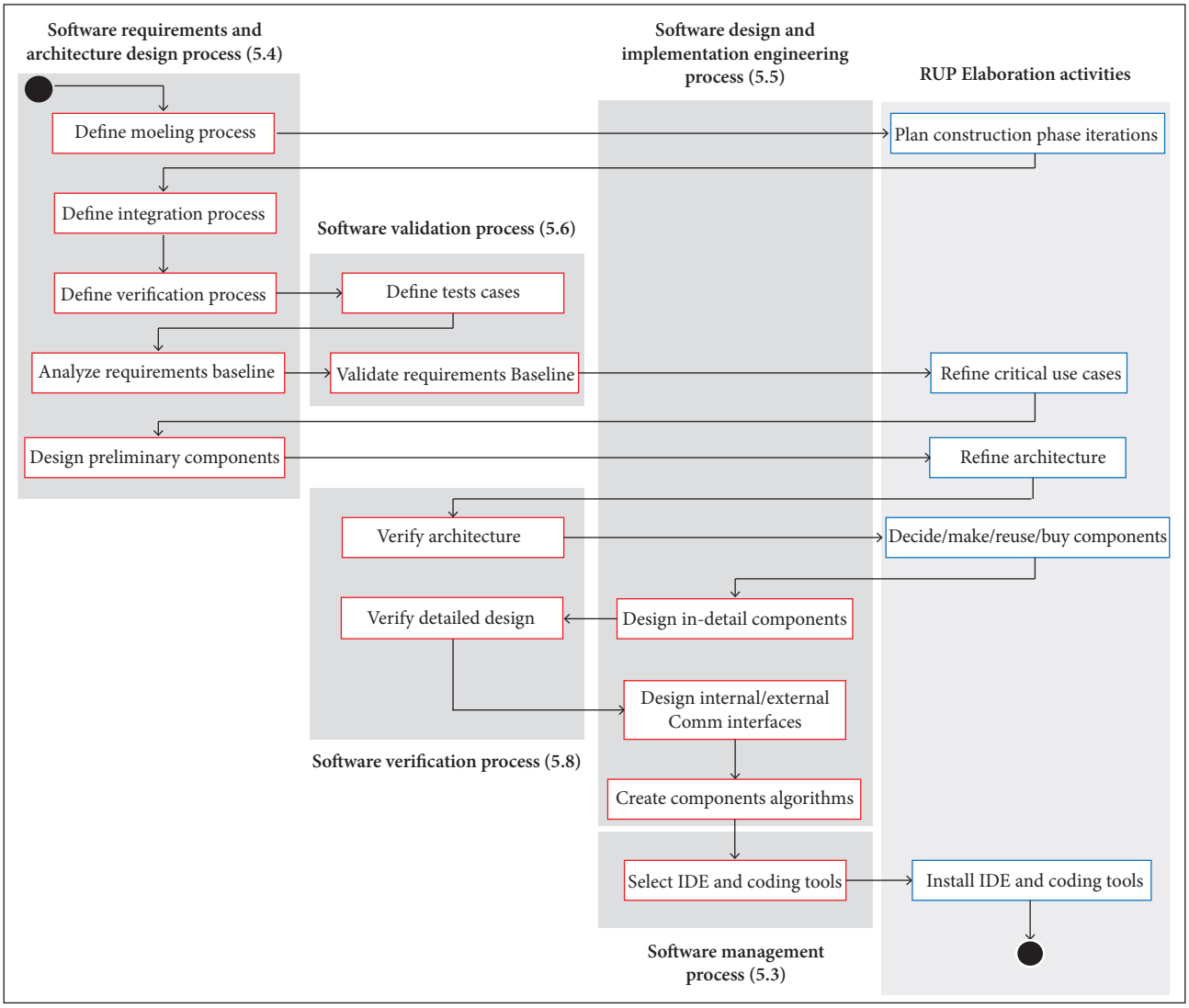

Figure 8. Elaboration phase activities for the H4ASD.

required that, for each pair of artifacts (Fig. 11), at least one version or each of the documents is built up. However, it is possible to choose which UML diagrams are used to model the software. The only compulsory diagram is the use case diagram. In order to see the details of the meaning of all abbreviations in the red boxes (Fig. 11), consult the ECSS-E-ST-40C document.

DESIGN OF THE GSCM&C

The sequence of tasks of the GSCM&C as the main element of the MCC arises due to the movement that the satellite does on a LEO and its relative position with GS after it has been deployed, resulting in three pass phases: (a) pre-pass; (b) real-time; and (c) post-pass. In phase (a), the trajectory of the SC should be predicted, and the operator must be allowed to program the telecommands to be sent. In phase (b), the antennas' auto tracking is activated, pre-programmed telecommands are sent, and their receptions are confirmed by the SC. The Beacon and telemetry-on-demand data are received, and, finally, each of the received parameters is evaluated in relation to their operating ranges in order to determine the state of the SC. In phase (c), the GSCM&C should display on screen the results of the satellite assessment, generate bug reports or alerts, and save the operation data (telecommands, telemetry and OPL) in the DB. Finally, operators must perform the analysis of the SC operation in the pass, using simulations, statistical graphs and data records through the GSCM&C GUI.

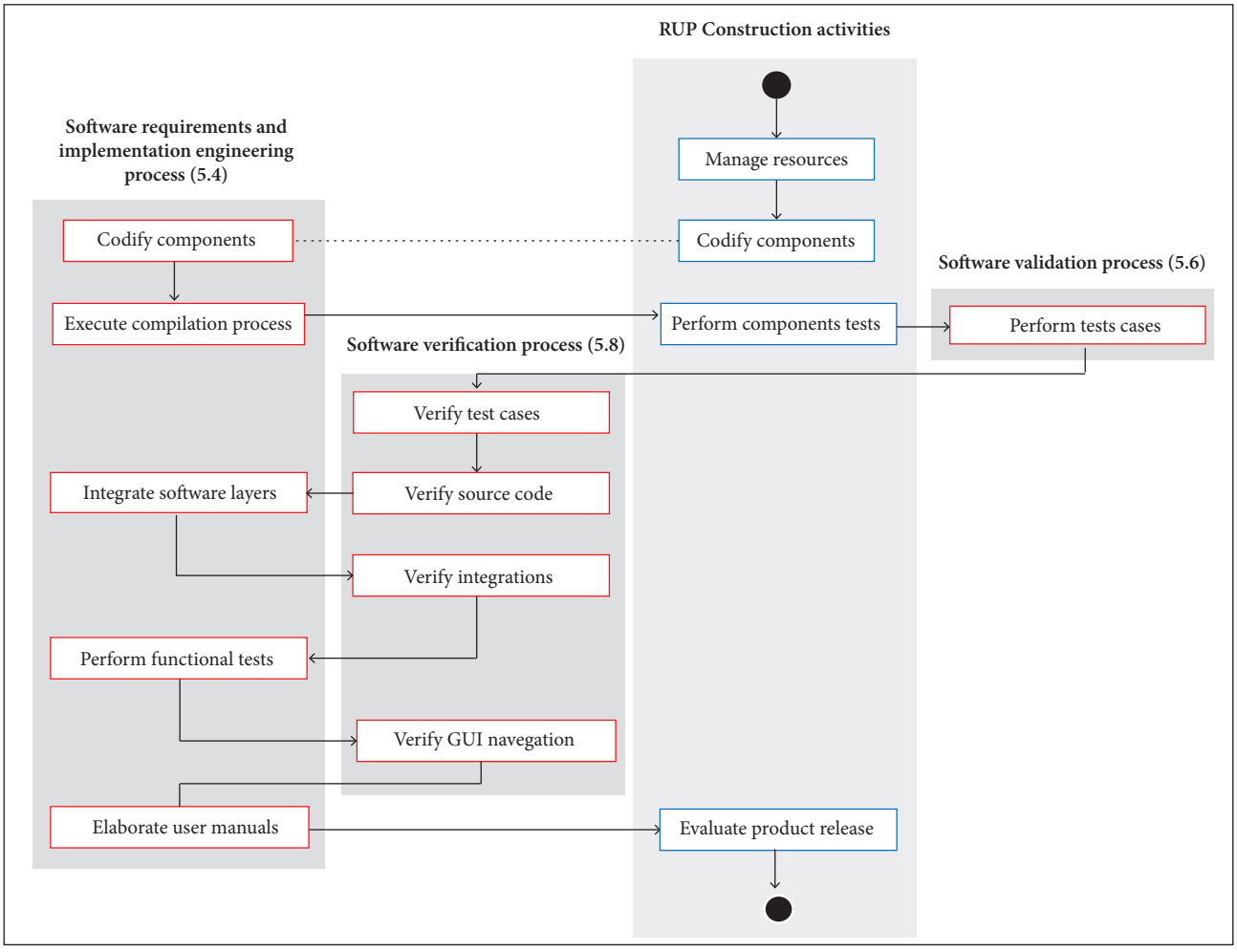

Figure 9. Construction phase activities for the H4ASD.

SD INCEPTION PHASE

The purpose of the Libertad-2 mission is to capture images of the Earth's surface by using an OPL with a multispectral sensor and a set of custom made lenses. The data that the sensor will deliver will be raw and compressed using the Huffman algorithm and wavelet transforms, so that the processing will take place on the ground through a MATLAB application. This application should be part of GS MCC. The data will be downloaded from the SC, by a 2.2-GHz S-Band downlink with a custom protocol, which must be integrated to the UHF/VHF communication. The GSCM&C should then receive the OPL RAW data and deliver them to MATLAB for reconstruction. To perform specific tasks as the satellite "tracking", in which the prediction path of the nanosatellite on LEO and the antennas' rotation are integrated, it was decided to use the Systems ToolKit (STK), a software application developed by Analytical Graphics Incorporated (AGI) Company. Because the GSCM&C GUI represents the "front-end" of the entire mission, it was decided to perform a system design based on interaction with 3-D elements, through which the operator can dynamically select the different elements and components of the nanosatellite to display their operating data and thus perform monitoring. From the definitions of this activity, the Stakeholders Request (IRD) and Vision (SSS) artifacts were generated.

A list of requirements for each mission subsystem (attitude determination and control subsystem, electrical power subsystem, command and data handling, OPL, Communication subsystem S-Band and UHF/VHF) and an additional list of general requirements were defined. The consolidated requirements structure for the development of the GSCM&C is composed of the following groups of requirements: <<Passes data>>, <<Telecommand>>, <<Telemetry>>, <<OPL S-Band>>, <<Beacon UHF/VHF>>, <<GUI Control>> and <<GUI Monitoring>>. From the results of this activity, the SRS artifacts were generated.

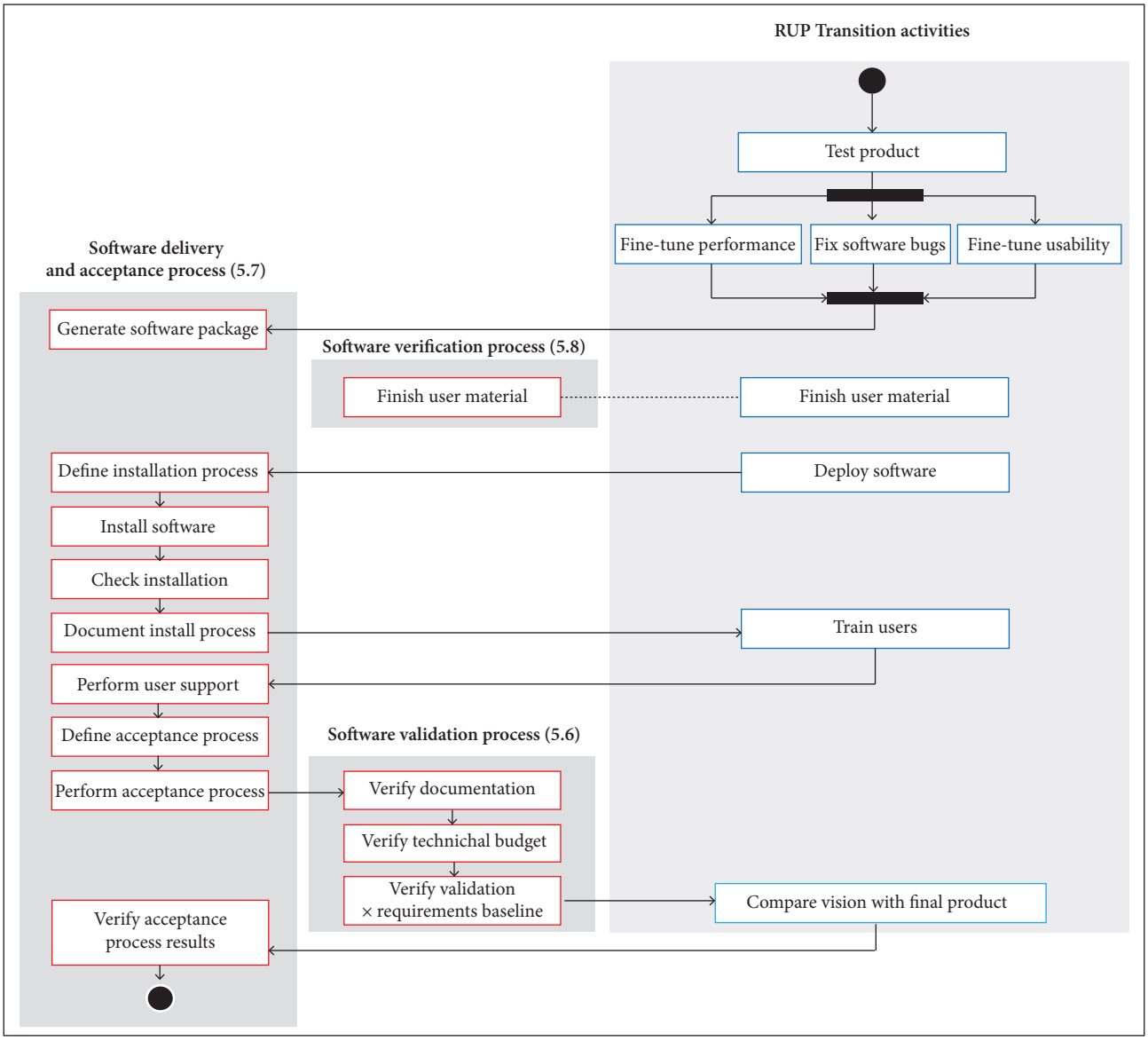

Figure 10. Transition phase activities for the H4ASD.

In Fig. 12, six primary use cases are shown, associated with two actors: the SW operator and the GS communications radios (UHF/VHF and S-Band). The operator can perform three main tasks: "Send Data", "Mission Monitoring", and "View Payload Images". For these operations to be executed by the operator, the UHF/VHF radio must be able to send and receive information to and from the SC using the AX.25 protocol, and S-Band radio must be able to receive data from the SC using a custom made link protocol, as shown in the box "COMM". In addition, the operator must configure the system to perform the antennas' rotation. This is possible through the use of a commercial "Tracking Application" or by adding records of trajectory calculations made by the STK. From each one of the primary use cases, "extends" or "include" cases are linked, involving an additional operation for their execution or depending of a prior execution of other operations. From the result of this activity, the glossary, business case, and use case model artifacts were generated.

The GSCM&C must be run on a server, which must be directly connected to the communications devices, such as transceivers and antenna's rotors. Furthermore, it should implement a DB, in order to save all information obtained from the mission operation. With these elements defined, the Libertad-2 GSCM&C architecture is set as shown in Fig. 13. The system consists of three logical layers, the basic layered model for software development: "User

Layer", "Business Layer", and "Data Layer". Each layer operates individually, but interacts with specific components of the others.

The user layer of the architecture presents data on the screen through a GUI, composed by Extensible Markup Language (XML) and XML-based (FXML) files built with a specific IDE for GUI design, 3-D Java libraries and OpenGL. The business layer contains all of the robust functional code — in this case, written in Java using the Java Development Kit (JDK8u5). The main components of the software are the "CORE" commissioned to the temporary control of the entire system, calculated with respect to the satellite passes over the GS, the control of the antennas in horizontal coordinate system, the link protocol that manages the data transmission and reception using the radios, and finally the "Java Database Connectivity" component, which manages the data control in the repository.

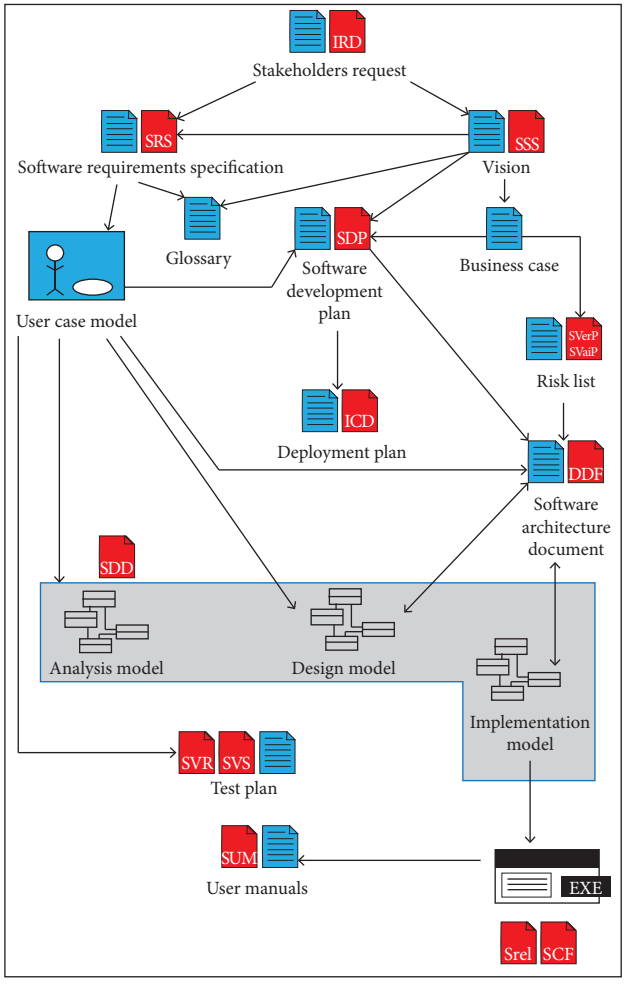

Figure 11. H4ASD artifacts tree.

The business and user layers act on the client and the data layer, on the server. This forms a client-server architecture. Although both client and server can run on the same computer, it is also possible to have a separate data engine in a different physical server, so that it is possible to connect several client applications to the data structure. From the result of this activity, the SDP and SAD artifacts were generated.

SD ELABORATION PHASE

Figure 14 corresponds to the composite structure diagram for the Libertad-2 MCC. The system can be divided into three main parts: the mission team, the MCC server, and the communication peripheral devices. The mission team refers to the elements that have to interact with the GSCM&C. These elements have direct influence on the GS operation, as in the case of the group of flight dynamics analysis (GFDA), the system operators (SO), and the group on satellite images research (GSIR), or even an indirectly influence, in the case of mission directors (MD). Each one has its specific functions and responsibilities.

The GFDA is the actor in charge of operating the STK software; in the SO actor, the users interact directly with the GSCM&C; and the GSIR is in charge of operating MATLAB. The second part of the diagram shows the main components of the GS server, where the STK, the GSCM&C, and MATLAB must be executed. The third part of the diagram represents the

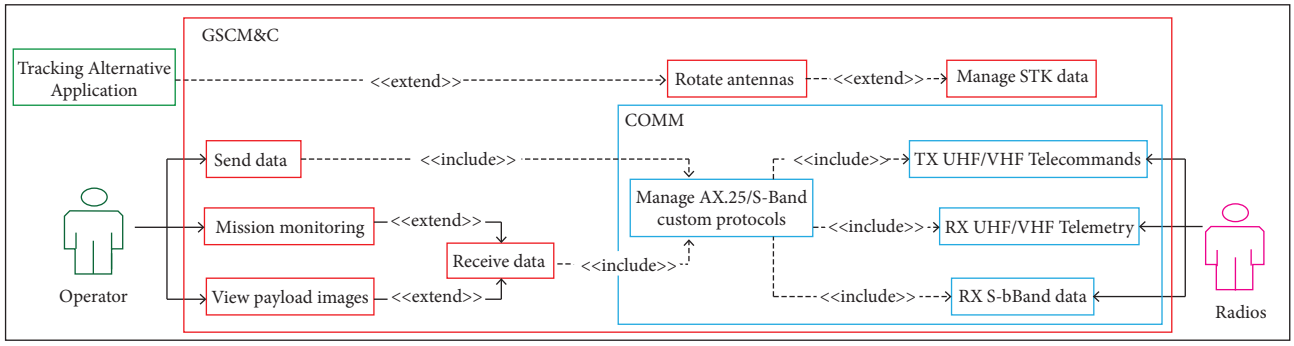

Figure 12. GSCM&C general use cases.

physical devices that compose the GS, which also becomes in the operational context of the GSCM&C. The GSCM&C should interact with the S-Band, UHF/VHF rotors, the UHF/VHF transceiver, and S-Band receiver.

To establish communications between the server and the communication devices, "RS232C"-type serial communications are needed in most cases; the interfaces can be USB or Ethernet, as in the case of some S-Band radios. From the result of this activity, a more complete version of the case model and the first version of the SDD were generated as part of the analysis model.

In order for station operators to analyze, monitor and process the information downloaded from the satellite, it is required a DB that allows the storage and retrieval of information in a practical and efficient way. The data of the satellite mission reflect the results of the satellite in operation, so its value is immense.

The CORE module of the GSCM&C centralizes its operation based on the passes predictions that are loaded from the STK, since all software operations during the mission life cycle, especially those having to do with communication, depend on the start, extent, and end data of each pass. The Entity Relationship Model (ERM) for the Libertad-2 mission GSCM&C consists of five functional business chains associated to a principal one called "Passes". Figure 15 shows one of the five functional chains of the ERM: the S-Band reception business chain (1), in which both the frames received by the receiver radio and the RAW image data, associated with a given number of frames, are stored.

The other chains of functional dependencies in the model are related to: (2) the overall system data and the SC general status; (3) the antennas' rotation; (4) the telemetry on demand and Beacon data as well as the evaluation of state in each telemetry parameter; (5) the last programmed telecommands and a relationship between sent telecommands and received data. From the result of this activity, the version of SDD was improved as part of the design model.

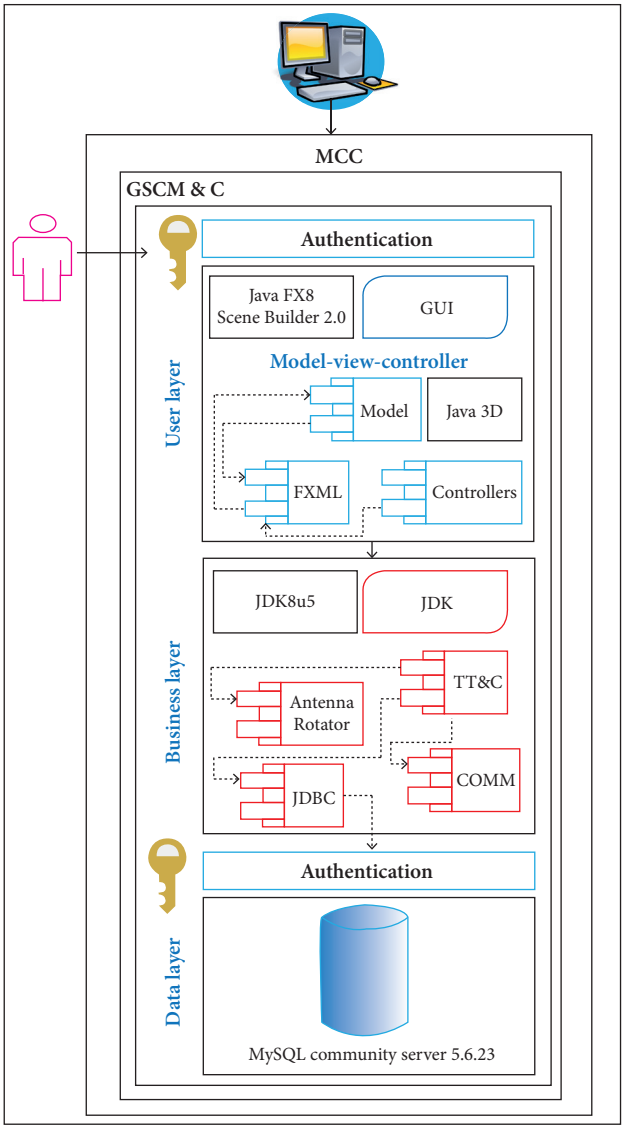

Figure 13. GSCM&C architecture with respect to MCC.

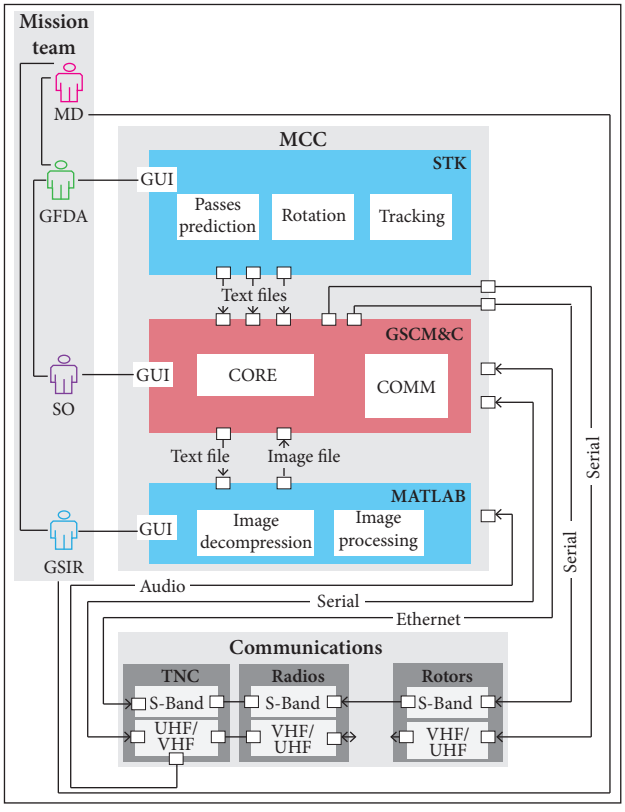

Figure 14. Composed structure of the GSCM&C as part of the MCC.

Figure 16 shows the GSCM&C main screen layout, in which the positioning of the various controls, panels, and data labels

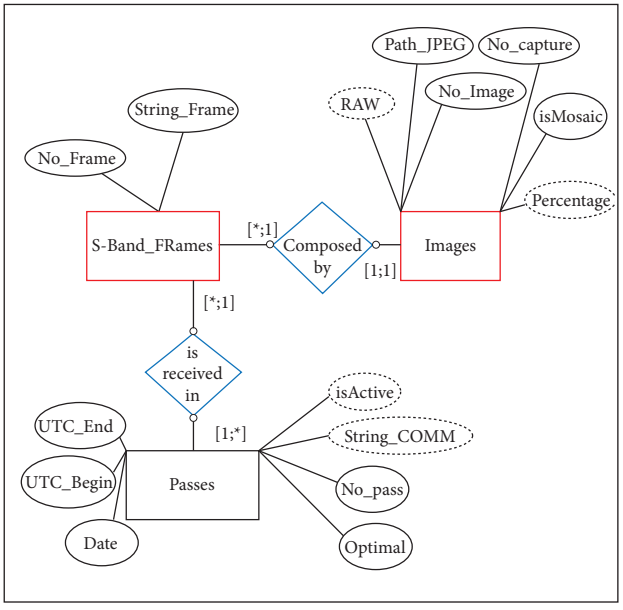

Figure 15. Some entities of the GSCM&C data model.

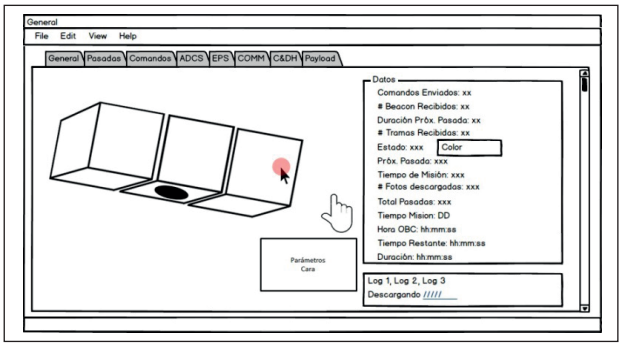

Figure 16. GUI design for the GSCM&C main window.

was defined, so that the different operations and use cases were included. From this window, the access to the telecommands programming, the STK data management, and the monitoring of each of the SC subsystems with a 3-D model interaction were contemplated. The implementation of a dynamic GUI, with visually pleasing elements of user interaction, such as the use of 3-D objects, ensures the usability of the application to perform the tasks of satellite mission monitoring and ground control.

The design of the GUI also provides: the possibility to observe the trajectory prediction data; control of the operating parameters of the GS radios; a window to program the telecommands to send and consult the telemetry data records; observation of the monitoring charts and the tracing of special operations. The version of SDD is improved with the results of this activity.

As show in Fig. 17, the MCC operations flow begins with the execution of satellite trajectory simulations in the STK (1). The data provided by the STK must be saved in text files in a specific location on the server file system, in order to be uploaded to the GSCM&C then (2). The first file delivery the records of each one of the satellite passes over the GS, specifying the "Start time", "Stop time", and the duration of the last pass in seconds. After the STK files are generated, the Operation System (OS) opens the GSCM&C and loads the passes prediction and rotation (AZ/EL) data (3) (4). The "DataManager" entity manages the loaded data using the DB and provides the prediction data to the "TimeControl" entity (5).

This entity is responsible for controlling the execution time of each operation during the mission life cycle through the MCC. The OS can program the telecommands to send through the GUI (6), and the "DataManager" feature saves them in the DB (7). At the right time, the "TimeControl" and "AntennaControl" entities are activated, so that these entities begin to track the SC in real time (8).

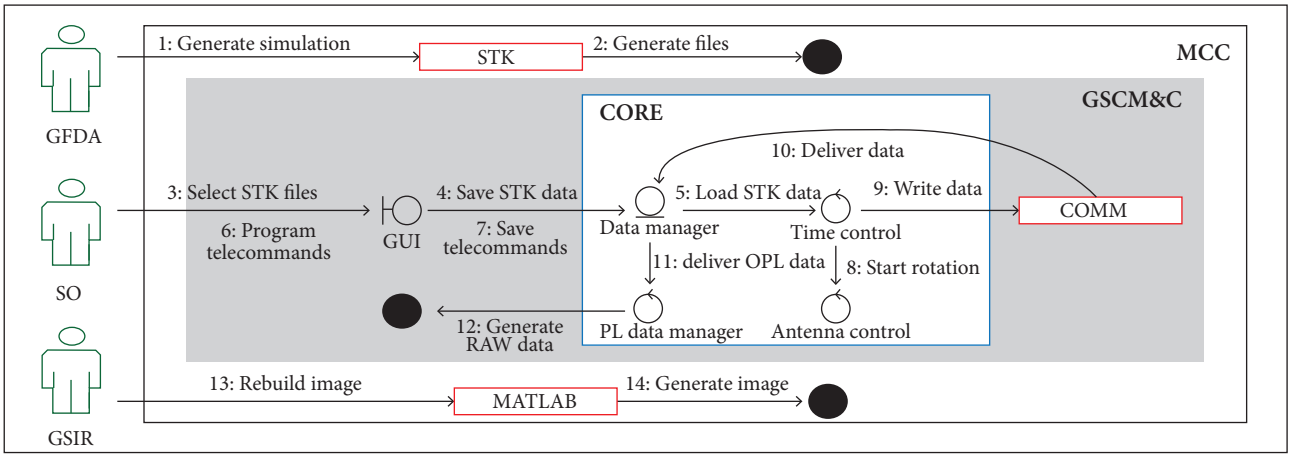

Figure 17. GSCM&C MCC operations sequence.

Immediately afterwards, the entity actives the "COMM" module and delivers the Uplink data, so this module begins the transmission (9). In parallel, the "COMM" receives the telemetry, Beacon and OPL data and delivers them to the "DataManager" entity (10), which organizes the RAW data frames for each individual image (11) and then generates the corresponding files in a specified location of the server file system (12). With this operation, the GSCM&C completes the tasks of the SC real-time phase.

With the RAW files on the server, the GUI runs the MATLAB program and rebuilds the image of the selected file (13). MATLAB generates a file in image format and saves it into a new directory location. Thus, the data captured by the nanosatellite OPL can be displayed on the screen of the GSCM&C. Figure 17 shows the flow of the GSCM&C main tasks regarding the MCC, through which it is possible to display the OPL reconstructed images based on the received RAW data. From this activity, the version of SDD was improved as part of the design model.

SD CONSTRUCTION PHASE

The construction phase is directly related to the implementation or coding of the SW element. In the case of the GSCM&C, a complete version of the application must be obtained, in which all the operations defined in the requirements, related to the SC monitoring and control, are included.

For the Libertad-2 GSCM&C implementation, several development applications were used. NetBeans 8.0.1 was used as IDE, MySQL Workbench 6.1 CE was used as Computer Aided Software Engineering (CASE) tool, in order to build the DB Structured Query Language (SQL) script, and the Java Development Kit (JDK) Standard Edition (SE) 8u51 was used as primary language. The whole environment was installed on a Windows operating system. This activity results in a deployment diagram corresponding to the GSCM&C implementation model.

The implementation of the Liberatd-2 GSCM&C results in a JavaFX application with the following navigation features: the operator can access only with a username and password provided by the (MySQL) DB administrator; when the operator enters into the application, a main window is presented (Fig. 18).

In the left side, the Libertad-2 3-D (Java3D) model is displayed, with which the operator can interact using the computer's mouse or touchscreen in order to visualize general information such as the structure faces temperature, the state of OPL lens, the power from the solar panels, and the output power of the Microstrip antenna. On the right side, operating statistics, such as the number of sent telecommands, the number of received Beacons, the duration of the last and next pass, the number of received S-Band frames, and other data, is displayed. By using the 3-D model, it is possible to select the different hardware components in order to observe the microcontrollers, memories, and sensors operation data in detail, using the telemetry data that have been received by UHF and have been stored in the DB. The subsystems shown in the 3-D model are: the OBC, the ADCS, EPS, OPL, magnetometer, UHF/VHF radio, and the S-Band radio.

If the operator wants to view the data in a more dynamic way, in order to make comparative analysis between the operation of two or more components, with a double-click selection on one of the (PCB) cards in the 3-D model, the application shows the subsystem monitoring panel, in which the operator is able to visualize N customizable charts and the records list of N components simultaneously, as shown in Fig. 19.

At the top of the application, there is a series of tabs controls whereby the operator can access the different operations, as the management of STK passes data, the scheduling of telecommands according to six operation modes, the visualization of behavior data in the UHF/VHF radios kit, the visualization of behavior data in the S-Band radios kit, and, finally, the records of historical

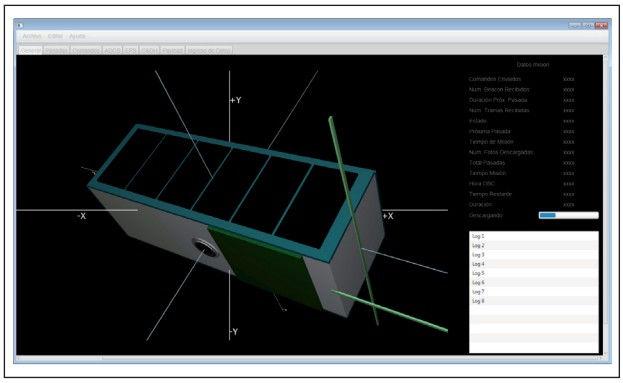

Figure 18. Libertad-2 3-D view on the GSCM&C main window.

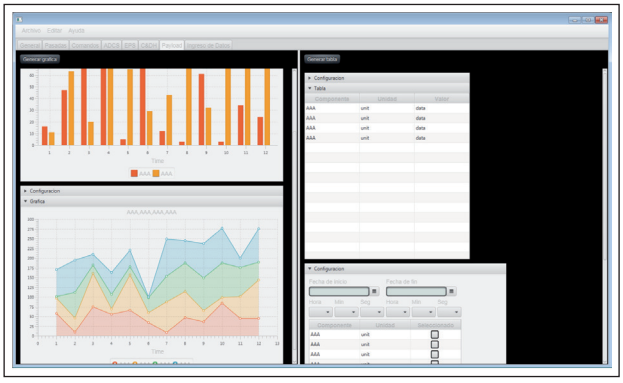

Figure 19. Charts view on the GSCM&C monitoring window.

image captures made by the OPL. As a result of this phase, the program executable files, the versions of the SRel, SCF, and *Software User Manual* (SUM) documents were obtained.

RESULTS AND CONCLUSION

In order to assess the functionality of H4ASD as a methodological tool for developing SW elements for academic satellite missions, some points are discussed ahead, which reflects the scope and limitations on the applicability of the model.

The use of RUP within the approach was aimed in order that traditional engineers, especially computer science engineers, could have clear concepts of software engineering that allowed them to easily assimilate and understand the proposed model, when using it for the development of a SW element in an academic-type aerospace mission. Being RUP the best known software development model, above the agile methodologies, it is easier for traditional engineers to understand the operation of at least a part of the ECSS standard. In addition, RUP suggests the use of UML, the most used modeling language, which also represents an advantage for the traditional engineer.

The structuring of the RUP elements is very similar to the ECSS-E-ST-40C; however, each one has different approaches (aerospace/commercial) that are complementary. This results in a work model in which the conventional software engineering concepts are applied using UML, and also the operating constraints of space context are incorporated, which allows the model to be used to guide the development process of both the SC embedded SW and MCC SW pieces.

The H4ASD integrates elements of both models and should not be seen as something that increases the difficulty, because the aim was to help it by taking a sequential life line to organize the activities, i.e. without losing the focus of the iterative and incremental method, an essential feature in order to refine and improve the analysis, design, and implementation models. The flow of activities for each phase of H4ASD is sequential, considering that the team for the development of academic satellite missions consists of a small group of students and traditional engineers, where there is not a complex team hierarchy as the commercial aerospace missions; for this reason, a roles model was not contemplated in the H4ASD approach.

The description of the GSCM&C development process for the Libertad-2 satellite mission as a study case of the H4ASD was mostly focused on the analysis and design phases, as they correspond to the most critical phases in the development process. If a design is correct and includes all system operations as an assembly, the implementation will be successful too. The use of 3-D objects in the GSCM&C Look&Field was performed in order that MCC operators could dynamically display and visualize information about Libertad-2 in-orbit operation. This was made following the proposals of design concepts as User Experience (UX) and Getting Real (GR).

The best way to validate the contribution made by the H4ASD is describing the results it has generated in the development of various software elements of the Libertad-2 satellite system, including the GSCM&C, which was described as a case study.

The H4ASD has allowed students of industrial, electronics, systems, and telecommunications engineering at the University Sergio Arboleda to be able to work together with local project engineers in the development of various software elements of the mission, specifically the following:, in the OPL — Huffman algorithm, wavelet transform compression, and OPL embedded control module; in the OBC — the on-board communication module and the C&DH; finally, the image reconstruction software for the MCC.

The model was presented to the developing team of the Libertad-2, was successfully received by two experts in aerospace engineering (mission director and engineering dean), and was clearly understood by research professors. It was also presented in the following undergraduate courses: software engineering, real-time operating systems, and embedded software for satellite applications, with a total of 18 students, of which 12 are currently working with specific development tasks.

The development process of SW parts from an academic aerospace system runs between A and D ECSS phases, i.e. Feasibility and Qualification and Production, regardless of the SW development model that is decided to use. In this sense, the SW part must be developed in parallel with the other mission subsystems, with iterative and incremental tasks.

The H4ASD approach allows traditional software engineers to easily understand how ECSS-E-ST-40C work model is structured for the development of SW in aerospace missions, as well as all processes, activities, and tasks suggested, using RUP as a comparator.

Leaving aside the level of effort that RUP describes for workflows, the periods of execution of each discipline during the life cycle are aligned similarly to the ECSS-E-ST-40C processes. This is what allows the coupling of both models into a single structure. The match between the ECSS-E-ST-40C processes

and RUP workflows, the complementary activities diagrams, and the artifacts tree breakdowns the ECSS-E-ST-40C structure and reduces its complexity, so that traditional software engineers can apply it as a complement of the suggested work in model in a conventional and known SW development methodology, in this case, RUP. Although the engineer try to be strict when using the ECSS-E-ST-40C, the most critical tasks for the development of the SW part, such as analysis and design, depend on the use of UML and RUP model. Therefore, artifacts, such as business cases, are complementary to ECSS-E-ST-40C and a vital part of the process.

FUTURE WORK

The aim of the research was to propose a working structure that would allow traditional SW engineers to guide the development of any SW part in the aerospace system, which is why the ECSS-E-ST-40C standard was selected. For the mission MCC SW, the ECSS-E-ST-70C standard can be contemplated, which contains specific processes descriptions for the development of ground stations. The study should be aimed at breaking down the rest of the work models of the ECSS "E" branch (Engineering Policy and Principles), as ECSS-E-ST-20C (Electrical and Electronics) or ECSS-E-ST-50C (Communications), using conventional models for the development of electronic or telecommunications systems.

ACKNOWLEDGEMENT

The H4ASD approach presented in this paper is framed within the "Satellite Libertad-2" program, which is currently carried out at the Escuela de Ciencias Exactas e Ingeniería of the Universidad Sergio Arboleda, in Bogotá, Colombia, in which it is expected to deploy in LEO a CubeSat-type nanosatellite of 4 kg with OPL that allows to capture images of Colombian surface in order to perform precision agriculture studies.

REFERENCES

Anderson L, Cole B, Yntema R, Bajaj M, Spangelo S, Kaslow D, Friedenthal S (2014) Enterprise modeling for CubeSats. Proceedings of the 2014 IEEE Aerospace Conference; Big Sky, USA.

Armellini F, Kaminski PC, Beaudry C (2012) Integrating open innovation to new product development — the case of the Brazilian aerospace industry. Int J Technol Learn Innovat Dev 5(4):367-384. doi: 10.1504/IJTLID.2012.050738

Asundi SA, Fitz-Coy NG (2013) CubeSat mission design based on a systems engineering approach. Proceedings of the 2013 IEEE Aerospace Conference; Big Sky, USA.

Brandstätter M, Eckl C (2009) Multi-disciplinary system engineering and the Compatibility Modeling Language (U)CML. Journal of Systemics, Cybernetics & Informatics 7(2):11-16.

Braxton Tech (2015) Telemetry, Tracking and Commanding (TT&C) ControlPoint™ (AceCP) [accessed 2015 Jun 10]. http://www.braxtontech.com/products/satellite-ttc/

Buchen E (2014) SpaceWorks' 2014 Nano/Microsatellite Market Assessment; [accessed 2016 Mar 16]. http://www.sei.aero/eng/papers/uploads/archive/SpaceWorks_Nano_Microsatellite_Market_Assessment_January_2014.pdf

Bürger EE, Loureiro G, Lacava PT, Carrera DHZ, Hoffmann CT (2014) The CubeSat AESP14 and its systems engineering development process. Proceedings of the 1st Latin American IAA CubeSat WorkShop; Brasília, Brazil.

Chaieb S, Wegerson M, Straub J, Marsh R, Kading B, Whalen D (2015) The OpenOrbiter CubeSat as a System-of-Systems (SoS) and how SoS Engineering (SoSE) aids CubeSat design. Proceedings of the 10th Annual System of Systems Engineering Conference; San Antonio, USA.

Díaz F, Camargo C, Pinzón P (2015) Software de monitoreo y control para la estación terrena de un nanosatélite. Bogotá: Fondo de Publicaciones de la Universidad Sergio Arboleda.

Díaz F, Quintero S, Triana J, Morón D (2014) Aproximación a los sistemas de percepción remota en satélites pequeño. Bogotá: Fondo de Publicaciones de la Universidad Sergio Arboleda.

Dingsoyr T, Nerur S, Balijepally V, Moe NB (2012) A decade of agile methodologies: towards explaining agile software development. J Syst Software 85(6):1213-1221. doi: 10.1016/j.jss.2012.02.033

European Cooperation for Space Standardization (2009a); [accessed 2015 Jun 10]. http://www.ecss.nl/

European Cooperation for Space Standardization. Secretariat ESA-ESTEC (2009b) Space project management project planning and implementation ECSS-M-30B Draft 14. Noordwijk, The Netherlands: Requirements & Standards Division.

Fischer M, Scholtz AL (2010) Design of a multi-mission satellite ground station for education and research. Proceedings of the Second International Conference on Advances in Satellite and Space Communications; Athens, Greece.

Funase R, Takei E, Nakamura Y, Nagai M, Enokuchi A, Yuliang C, Nakada K, Nojiri Y, Sasaki F, Funane T, Eishima T, Nakasuka S (2007) Technology demonstration on University of Tokyo's pico-satellite "XI-V" and its effective operation result using ground station network. Acta Astronaut 61(7-8):707-711.

Huang PM, Darrin AG, Knuth A (2012) Agile hardware and software system engineering for innovation. Proceedings of the 2012 IEEE Aerospace Conference; Big Sky, USA.

Jacobson I, Booch G, Rumbaugh J (2000) El proceso unificado de desarrollo de software. Madrid: Addison Wesley.

Kaslow D, Anderson L, Asundi D, Ayres B, Iwata C, Shiotani B, Thompson R (2015) Developing a CubeSat Model-Based System Engineering (MBSE) Reference Model-interim status. Proceedings of the 2015 IEEE Aerospace Conference; Big Sky, USA.

Kaslow D, Soremekun G, Kim H, Spangelo S (2014) Integrated model-based systems engineering (MBSE) applied to the simulation of a CubeSat mission. Proceedings of the 2014 IEEE Aerospace Conference; Big Sky, USA.

Laizans K, Sünter I, Zalite K, Kuuste H, Valgur M, Tarbe K, Noorma M (2014) Design of the fault tolerant command and data handling subsystem for ESTCube-1. Proc Est Acad Sci 63(2S):222-231. doi: 10.3176/proc.2014.2S.03

Llorente JDG, Leguizamón GAP (2014) Estimación de la cantidad de potencia suministrada por las celdas fotovoltaicas de un CubeSat. Tecnura 18(41):53-63. doi: 10.14483/udistrital.jour.tecnura.2014.3.a04

Lopez DM, Blobel BGME (2009) A development framework for semantically interoperable health information systems. Int J Med Inf 78(2):83-103. doi: 10.1016/j.ijmedinf.2008.05.009

Mohammad A, Straub J, Korvald C, Grant E (2013) Model-based software engineering for an imaging CubeSat and its extrapolation to other missions. Proceedings of the 2013 IEEE Aerospace Conference; Big Sky, USA.

Nader MR, Carrion MH, Uriguen MM (2014) The Ecuadorian experience in space: the nee satellite constellation. Proceedings of the 21st IAA Symposium on Small Satellite Missions (B4); Toronto, CA.

Pradels G, Baroukh J, Queyrut O, Sellé A, Malapert JC (2012) CNES solution for a reusable payload ground segment. Acta Astronaut 81(2):610-622. doi: 10.1016/j.actaastro.2012.08.036

Puschell JJ (2011) Formal requirements definition. In: Wertz JR, Everett DF, Puschell JJ, editors. Space mission engineering: the new SMAD. Hawthorne: Space Technology Library. p. 105-123.

Ramos DB, Loubach DS, da Cunha AM (2010) Developing a distributed real-time monitoring system to track UAVs. IEEE Aero Electron Syst Mag 25(9):18-25. doi: 10.1109/MAES.2010.5592987

Raphael D, Stone R, Guevara D, Fraction J (2014) Command & Data Handling for the magnetospheric multiscale mission. Proceedings of the 2014 IEEE Aerospace Conference; Big Sky, USA.

Sand J, Goehner K, Korvald C, Berk J, Straub J (2013) Payload processing aboard an open source software CubeSat. Proccedings of the Spring 2013 CubeSat Workshop; San Diego, USA.

Schilling K (2006) Design of pico-satellites for education in systems engineering. IEEE Aero Electron Syst Mag 21(7):S9-S14. doi: 10.1109/MAES.2006.1684269

Spangelo S, Cutler J, Anderson L, Fosse E, Cheng L, Yntema R, Kaslow D (2013) Model based systems engineering (MBSE) applied to Radio Aurora Explorer (RAX) CubeSat mission operational scenarios. Proceedings of the 2013 IEEE Aerospace Conference; Big Sky, USA.

Spangelo S, Kaslow D, Delp C, Cole C, Anderson L, Fosse E, Gilbert B, Hartman L, Kahn T, Cutler J (2012) Applying Model Based Systems Engineering (MBSE) to a standard CubeSat. Proceedings of the 2012 IEEE Aerospace Conference; Big Sky, USA.

Thüm T, Kästner C, Benduhn F, Meinicke J, Saake G, Leich T (2014) FeatureIDE: an extensible framework for feature-oriented software development. Sci Comput Program 79:70-85. doi: 10.1016/j.scico.2012.06.002

Villamil DCC, Mayorga JAA (2013) Análisis de rendimiento de dos sistemas operativos en tiempo real implementados en dos arquitecturas de hardware para la selección del sistema embebido que soportará el Software Command And Data Handling de la Misión Satelital Libertad 2. Proceedings of the 11th Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2013); Cancun, Mexico.

Webster J, Corcoran P (2007) NPR 7120.5 and NASA's Program/Project On-line Library and Resource Information System (POLARIS). Proceedings of the 2007 IEEE Aerospace Conference; Big Sky, USA.

Woellert K, Ehrenfreund P, Ricco AJ, Hertzfeld H (2011) Cubesats: cost-effective science and technology platforms for emerging and developing nations. Adv Space Res 47(4):663-684. doi: 10.1016/j.asr.2010.10.009

Ziemke C, Kuwahara T, Kossev I (2011) An integrated development framework for rapid development of platform-independent and reusable satellite on-board software. Acta Astronaut 69(7-8):583-594. doi: 10.1016/j.actaastro.2011.04.011