Requirements and Dependability Obstacles That Must Be Overcome

Carlos Henrique Netto Lahoz
Aeronautics and Space Institute - IAE
lahoz@iae.cta.br

This article presents a study on the identification of the main obstacles that must be overcome in the requirements engineering phase of critical computer systems in order to ensure that these activities will help improve the system's dependability. The idea is to present how technology, process and people can interact with software and system engineering as well as the whole organization in order to fulfill the system´s mission with safety and success. Besides that, the intention is to contribute with the discussion about requirements practices in critical computer systems, particularly in the space domain, as well as helping us understand important system and software requirements engineering problems. This approach comes from the experience acquired during the development of the on-board software for the Brazilian Satellite Launch Vehicle and the study of emerging technologies in software engineering.

Introduction

Software plays a strategic role in system engineering nowadays, mainly in computer systems. More and more functionality that was implemented by hardware is now being implemented by software. This increasing dependency of our society on computer systems, regarding the understanding, creation and maintenance of physical, biological and social systems, creates a necessity to improve the man abilities to understand and project engineering systems, especially those classified as complex systems (Magee and de Weck, 2004). Bar-Yam (2003) defines a complex system as being the one that, besides considering a reasonable number of distinct variables, integrates many systems, presents high performance constraints as well as functional demands, high response rates, and specific protocol. Johnson (2006) added that the complex systems are those that can present multiple interactions that occur among different components and can also identify many system states based on temporal flow interactions, as a response to the quick changes on the system and its environment.

Leveson (2009) explains on her new online book that we are designing systems (complex ones) with potential interactions among the components that cannot be thoroughly planned, understood, anticipated, or guarded against. The operation of some systems is so complex that it's a challenge to understand and be fully aware of their potential hazardous behavior. The software has tried to make these systems easier to be implemented, in a more integrated way. However, because of such a high number of components interacting dynamically, the strong coupling allows dysfunctions between parts of the system. The problem, according Leveson, is that some systems are being developed in such a way that the management of this increasing interactive complexity makes it too difficult for us to manage. This complexity and the coupling make it difficult for the designers to consider all the potential states of the system as well as for

the operators to deal with all the situations (handle all normal and abnormal situations) and disturbances safely and effectively. Besides that, the author stated that in the space project domain, the vast majority of software accidents were related to flawed requirements and misunderstandings about what the software should do (Leveson, 2004, 2009). The generation of software requirements for critical systems is one of the major sources of errors in system development.

A manner of minimizing potential safety problems in critical computer systems, as well as in space projects, is to introduce safety considerations early, during the software requirement activities, and to promote a deeper interaction between system and software engineers, using their methods of capturing, specifying and managing requirements.

This work intends to summarize some matters related to safety requirements involving the system and the software engineering in computer critical systems focused in space applications. A table with the main obstacles that the requirements activities must overcome is presented, and some considerations about technical, managerial and cultural aspects are made.

Requirements and Dependability in Space Systems

Specifically in the space area, the search for continuous systems automation has increased considerably the complexity of the mission and the steps required to achieve it. Ingham (2004) states that, in spite of the complexity that comes from automation systems (the use of robots in spacecrafts for example), it has become an economic necessity when it comes to continuing the progress into space. The authors also point out that on the space systems projects there is a gap between the requirements on software specified by systems engineers and the implementation of these requirements by software engineers. The software engineers must perform the translation of the requirements into software code, hoping to accurately capture the systems engineer's understanding of the system behavior, which is not always explicitly specified. This gap opens up the possibility for the misinterpretation of the systems engineer's intent, potentially leading to software errors.

Lutz and Mikulski (2004) add that in many embedded systems that have some kind of human interaction, especially in the critical systems domain such as in a space vehicles, the information regarding the requirements is known during the whole system development process, and could be extended more than it was predicted.

Another study conducted by Hecht and Buettner (2005), from 1998 to 2000, reported that approximately half of the anomalies observed in space vehicles were related to software. The authors affirmed that, in a system under development, the definition of software requirements is the most common source of mistakes. Regarding the flaws that resulted from the requirements, half of them have as their cause a poor description of the requirements, ambiguity, mistakes and lack of clarity. The other half is a result of the omission of the complete requirements.

It is also possible to affirm that inadequate practices of requirement specification and lack of knowledge of the system, leading to a poor understanding of the operational system's context and its quality requirements are among the most common reasons for the failure of critical projects. In the conclusion of the investigation of the accident of rocket Ariane 501 (Lyons, 1996), from the European Space Agency (ESA) poor

practices of requirement specification were identified. The investigation report of the aircrafts accidents of the Mars Climate Orbiter – MCO (NASA-MCO, 1999) and Mars Polar Lander – MPL (NASA-JPL, 2000), recommended a more adequate training of the development teams due to problems in the specification of the system's requirements.

Hjortnaes (2003) observes, in the analysis of more than eighteen ESA projects, lack of maturity and stability of the baseline of software requirements when the development process begins. He also mentions that in many of the analyzed reports, the development of the requirements was not conducted in an adequate way or there was not a correct understanding of them. Regarding the issue of systems dependability, to deny the fact that the software is an item of critical safety that only provides information and does not directly release energy into the system is becoming an argument less and less acceptable. Software plays an increasingly important role in accidents (Leveson, 2004). The growing dependency of the software on information systems is introducing new paths for the risk caused by the loss of information or by incorrect information, possibly leading to an unacceptable physical, scientifically or financial loss. The software systems have reached such a high level of complexity and coupling among their components that the ability to control their interactions is beyond human. The behavior of a computer system may not be totally planned, understood, anticipated or totally trustful. Mechanisms of trustfulness, such as redundancy in this situation, may make the problem worse, adding such complexity that can result on an accident so far not predicted (Leveson, 2009).

During the integration and tests of the aircrafts Voyager e Galileo, 387 mistakes were detected (Lutz, 1993). Among those, the ones potentially related to safety have appeared more commonly from discrepancies between the specifications of documented requirements and the necessary requirements for the correct functioning of the system and the poor understanding or the software interface with the rest of the system. On the MPL accident report, the use of FMECA (Failure Modes and Effects and Criticality Analysis) e FTA (Fault Tree Analysis) is recommended, with the appropriate redundancy mechanisms for failures to be eliminated, in spite of the discussion about the effectiveness of the use of those conventional techniques to identify software failures.

Specifically regarding the project of the Brazilian satellite launcher, the official investigation report of the third prototype (VLS-1 V03) accident (Serviço Publico Federal, 2004), and the independent study proposed by Almeida and Johnson (2005) emphasize that not enough attention was paid to the development of a safety culture appropriate to the project. There was a lack of review, external audit and validation of risk evaluation experts during the project´s development. Almeida and Johnson also state that in some areas of the Brazilian space program the importance of the safety engineering management has not been acknowledged, as it is recommended by the space industries of North America and Europe.

To invest on the improvement of the safety activities associated with the requirement engineering process, in terms of crossing barriers on the identification of dependability requirements, even the requirements regarding system in the space organizations have to be motivated in order to guarantee the accomplishment of the main goals of the mission with safety and success.

Obstacles That Must Be Overcome

Taking into account the studies mentioned in the last section, it is easy to conclude that in several space projects, even when the organizations involved are highly mature space agencies, there are many problems related to requirements and safety culture. Andreou (2003) emphasizes that human, social and organizational factors play a decisive role in software development in terms of determining functional and non-functional characteristics of software products. The process introduced by Andreou defines some steps for recording these factors based on certain software quality characteristics that are treated as principal components for conducting requirements identification.

In addition, Johnson's studies (Johnson, 2003) about the Mars Climate Orbiter incident investigation, took into consideration people, process and technology points of view to identify and discuss some ways to solve problems related to system hazards. The idea is to use these points of view to map and prioritize the obstacles that must be overcome, relying also on the studies presented by Leveson (2004) and the experience of the author in space software development, so as to mitigate the lack or the poverty of software systems requirements specifications (LAHOZ et al., 2006).

Table 1 summarizes some most frequent requirement and dependability problems or obstacles, from technology, process and people points of view.

Table 1: Obstacles associated with requirements and dependability

VIEW	OBSTACLES
1. Technology	1.1 Inadequate requirements engineering techniques. 1.2 Inadequate model representation of the system components and their interactions. 1.3 Lack of safety approach.
2. Process	2.1 Lack or poor process model. 2.2 Partial process monitoring. 2.3 Inadequate communication.
3. People	3.1 Different system and software cultural approach. 3.2 Limited understanding of the problem domain. 3.3 Inefficient management.

Technology 1.1 - Inadequate RE techniques: From a technological point of view it is noticeable when there is an inadequate or non-existent use of requirements elicitation and analysis techniques. Stakeholder analysis, interviewing, document studies, prototyping or even design workshops are often misused, not being able to offer completely reliable results.

Overcoming obstacle 1.1: The development team must have the necessary capability and training in software engineering tools and requirements engineering methods. The inadequate use of resources, such as the one available in CASE tools, in model checking and other support resources for requirements representation and validation, could be the cause of requirements misunderstandings and project mistakes which, on their turn, can lead to ignorance, suspicion, and even the team and project´s discredit. Another important point is that the project subcontractors should use appropriate and satisfactory requirements elicitation engineering techniques, such as design workshops, prototyping , functional and hazard analysis, in order to understand the space system domain, especially regarding the identification the safety properties and constraints. During the requirements analysis phase, the subcontractor must adopt specific techniques for the elicitation of the non-functional requirements such as reliability, availability, maintainability and safety (RAMS), among others directly related to dependability. Various techniques may be used to support hazard analysis such as HAZOPS and fault-tree analysis and these can be the basis for deriving dependability requirements (Sommerville, 2003). Nevertheless, other techniques that can help the identification of functional and non-functional requirements must be used, as well as methods for gathering information, such as interviews, questionnaires, observation, and the study of documents or existing software.

Technology 1.2 - Inadequate model representation of the system: An inappropriate model representation of the system components and their interactions may not show the behavioral interdependence between specifications of the system components and their consequences. This poor representation may not appear until very advanced phases of a system development, causing financial loss and problems with project schedules. The lack of safety approach, mainly in immature environments in terms of concern with safety, may go unnoticed in the initial phases of the elicitation activities. However, it will surely impact more on the advanced phases of the development or appear only after the system comes into operation, which is even worse.

Overcoming obstacle 1.2: A good way to evaluate if the risks associated to each component are acceptable would be to design the system in terms of components, goals, soft-goals, tasks and resources (Yu, 1995) and after that perform a hazard analysis, based on HAZOP guidewords, applied to those system components (Lahoz, 2009), . If that is the case, new system goals can be set right away. The creation of a technique that is able to elicit dependability goals can promote new ways to minimize safety problems from the requirements specification phase until the development code phase. Another great benefit is the use of the same language between heterogeneous teams, such as system engineers and software engineers, in order to create a common and understandable knowledge about the subject under analysis, developing more trustfulness hardware and software solutions.

Technology 1.3 - Lack of safety approach: Overconfidence in the system engineering decisions might lead to short-cuts on the conduction of the safety activities, causing an

increase on the risk factor of the whole project. This behavior is not new and it is extremely difficult to identify when the engineering culture of an organization is outdated and compromised with the success or failure of a project. Besides that, digital technology has created a radical change on space engineering, but systems and safety engineering techniques have not kept the same pace. Many ways of approaching to prevent accidents that dealt with electromechanical components – such as the redundancies of components in order to protect the system against the failure of individual components is not useful when it comes to controlling accidents which cause is the use of digital and software systems. (Leveson, 2004).

Overcoming obstacle 1.3: The problems that are potentially related to safety have originated (more frequently) from discrepancies among the requirement specifications and the user needs, and also the misunderstandings regarding the software interface with the rest of the system. Many times, software conversion variables are defined based on input/output communication with hardware command and control equipment that is not rigorously analyzed. It is necessary to discuss a better way of perform the specification review and requirement analysis – system and software specifications must be able to be easily reviewed and to be easily understood by a wide range of specialists. The specification must describe what the system and the software should not do. Leveson (2009) stated that standards and industry practices often forbid such negative requirements statements. The result is that software (and system) specifications often describe nominal behavior well but present a very incomplete description with respect to required software behavior under off-nominal conditions, rarely describing what the software is not supposed to do.

Process 2.1 - Lack or poor process model: From a process point of view the lack or poor process model, as well as a precarious safety approach are strongly related to an immature team and organization. Many managers and developers aren´t convinced of the value of developing and using processes during the requirements engineering process. Also, for reasons external to the project, such as the non-availability of qualified personnel, inefficient organizational structure, time and resource constraints, requirements activities are neglected in favor of others, such as the ones directly related to code development. And even when an elicitation process is established, the partial monitoring of the process makes the follow-up of the changes occurred during the life cycle of a requirement during the system´s development practically impossible.

Overcoming obstacle 2.1: There is a lack of investment and organization commitment regarding to creating and maintaining process quality models, especially related to the dependability requirements for critical computer systems in a long-term basis. It is necessary to conduct the requirement engineering activities in an easy way through the formalization of simple steps, including the creation of system models followed by the application of a safety engineering technique. Those activities, depending on the level of detail intended, cannot last a period of time that enables the project schedule to be followed. Even if process improvement standards such as CMMI or the ECSS (2009), proposed by the European Space Community, have already been adopted, the disconnection between the system and software engineering processes is clearly notable. Formal reviews and verification and validation activities during requirement and analysis process must be defined in a way to evaluate if the adopted process and product solution meet the dependability requirements such as safety and availability.

Process 2.2 - Partial process monitoring: Complacency and discounting the risks associated with software are the main causes of most accident factors (Leveson, 2004). An incomplete set of tests and inadequate reviews of the requirement changes are also the cause of process monitoring problems.

Overcoming obstacle 2.2: It is important to consider verification and validation as high priority activities during the system development and devote special attention to the resources allocated to testing. In addition, the software complexity should be overestimated and the tests effectiveness should be underestimated. The project management needs to monitor exhaustively the products presented in each review and follow-up the product process improvement.

Process 2.3 - Inadequate communication: Another frequent obstacle in projects is the inadequate communication among stakeholders. The inexistence of an efficient communication process helped by an automated tool can cause many problems. Without effective communication, it can be impossible for the stakeholders to evaluate in a consistent way all the requirements and to remain stable during project life cycle.

Overcome the obstacle 2.3: In spite of the use of email as a quick and practical way of communication among the stakeholders, many of the important decisions taken by email are not appropriately formalized. Any type of communication that might affect the requirements and that have some kind of impact on the project should not be restricted to an email or telephone call. Even though it sounds obvious, it happens rather frequently. A formal requirement configuration management must be established and continuously updated. Regarding the dependability issue, it can be very serious. When there is a need to investigate an accident that occurred due to a change in a dependability requirement requested only by email (without a change impact analysis) can have serious consequences.

People 3.1 - Different system and software cultural approach: From a people point of view, different cultural approaches among systems and software designers may become an obstacle impossible to overcome. Software is a relatively new discipline, according to systems engineers, and many times the participation of the software team in the system requirements activities is not viewed as important as it should be (Gonzales, 2005). Also, the author has stated that it is necessary to examine some fundamental beliefs and assumptions, shared values, behavioral norms, and artifacts to gain insight into how the software and systems engineering communities approach the requirements discipline. Both communities have matured in their understanding of the requirements engineering discipline; both are beginning to understand that capturing requirements requires understanding the whole problem before developing a solution.

Overcoming obstacle 3.1: Even with the increasing dependency of the space systems in relation to the software, many system engineers believe that the software architecture strongly depends on the hardware architecture. It is impossible for them to think that software architecture could have a strong influence in the project solutions, in such a way that could cause hardware changes. Many times, when there is a need of solving hardware problems, software changes are suggested in order to reduce the impact on the project costs. Some of these problems could be solved if the software team could take part in the decision making process of the project alternatives, when the solutions to be adopted are still being defined. There are must be create a mechanism that motivate the

discussion of how each engineering area sees and deals with the requirements and dependability. Talks about these topics over different cultural approaches should be stimulated through discussion forums and workshops when both engineering areas can present their approach and views of the system. In the end of the discussion, a consensus about the hardware and the software roles in the project should be met.

People 3.2 - Limited understanding of the problem domain: The limited understanding of the problem domain, which together with the partial comprehension of the project dependability issues (from the stakeholders views), negatively impacts any requirements elicitation techniques. Besides that, the practice of requirements' elicitation is still performed without sharing results between both engineering areas in a way to perform the verification and validation still on the initial phases of the project. Without a clear and precise understanding of the problem domain it is difficult to interact with the stakeholders and go beyond the individual ways in which each one views the system. Kauppinen et al (2002) mentions that many times customers typically do not know what users really need and what is essential in their tasks. The author emphasizes that it is necessary to link business goals to technical requirements via user needs and user requirements.

Overcoming obstacle 3.2: The experience of the teams has to be shared as early as possible, avoiding mistakes on the identification and understanding of the non-functional requirements related to dependability. In addition, the change of a team during the development phase of a space project (that could take a few years) can cause the loss of precious system and project knowledge, which in many cases is not easily recovered. Many times the replacements are not as qualified and trained for that position. It is mandatory for those who will perform the role of identifying and validating the requirements to have the necessary skills to conduct those activities in a way that they are able to identify the explicit and implicit requirements of project dependability. Also, as for inappropriate requirements management Kauppinen et al (2002) suggest that cross-functional teams are a useful approach in eliciting user needs. The teams could use the knowledge and views of members with varied backgrounds. It is easier to convey the user needs identified to projects when a project manager participated in the team-work effort.

People 3.3 - Inefficient management: Many times the project management does not handle well the safety engineering related activities. Even performing risk analysis, hazard analysis and FMECA in the beginning of the project, not all the detected problems are identified until the end of the qualification and acceptance reviews. The project management also presents, in many cases, an attribution of responsibility without the total understanding about what each group was doing. The roles of the people involved are not clearly set and the project leaders do not maintain the necessary authority and responsibility regarding some participants, in such a way that important problems could be proprietarily solved.

Overcome the obstacle 3.3: The dependability, especially regarding system safety, should have an important role in the organization, working effectively without the influence of other authorities, under deadline pressure and goals to be reached. In spite of the practical difficulties of reconciling technical project issues with management, budget, and political questions, a cultural change should take place. As a matter of fact, an accident in a project causes a commotion in society and in the high administration of

the space sector that can make positive changes on the management of future projects. However, as time goes by, this initiative can get weaker, due to the influence from the pressure of the same sectors. Even more serious is an inefficient management of the elicitation activities: the project manager must balance interests, determine priorities and foster consensus among stakeholders. Without such balance, opposing currents of seemingly common interests may slowly and gradually undermine a project. Finally, the dependability must be clearly recognized as a value inside the organization and the project and integrated into all engineering activities (mainly requirements activities). System and software project leaders should be place high priority on dependability and communicate this objective clearly and consistently to the rest of the group and the organization.

Final Considerations

This paper presents the main obstacles that must be overcome in the requirement engineering for the identification of the system´s dependability, especially due to a technological, process and cultural change of the people as well as the organization involved in the computer critical systems development. Whatever the phase of the project or the discipline involved, this change must be based on quality, and more specifically, on the definition of goals that help increase the degree of confidence on the functioning of the system.

The discipline of RAMS (Reliability, Availability, Maintainability and Safety) must be dealt with by the requirement engineer in an integrated way with the system and software teams as well as the management of the project. The intention is to be able to obtain a wider and more complete view of the dependability issues through a well defined process. This integration should allow the sharing of skills and values of each engineering profile and also create a single and strong safety culture.

We must reach a balance between the system engineering aversion to risk and the software engineering tendency to take fashionable software approaches. The implementation of the safety culture on the early stages of the mission analysis will enable a clear recognition of its value by the participants, being integrated in all the activities related to the engineering, being able to be account and to have an (relatively) independent leadership status.

What is expected, on the organizational level as well as on the cultural and engineering levels is that the process brings a new mindset, strongly focused on dependability, forcing the teams to position themselves in a pro-active way in terms of fulfilling the requirements regarding reliability and safety.

It is necessary that all computer-related resources, on board systems, ground systems and any other support space systems, are submitted to some kind of dependability analysis or evaluation of their contribution or not to the success of the mission. All the development phases of a product should be observed under the dependability point of view, and in case any type of unsafe behavior is detected, non-functional goals should be defined in the sense of guaranteeing that the system keeps a safe behavior.

References

Almeida, I. M., Johnson, C. W. 2005. Extending the Borders of Accident Investigation: Applying Novel Analysis Techniques to the Loss of the Brazilian Space Programme's Launch Vehicle VLS-1 V03. http://www.dcs.gla.ac.uk/~johnson/papers/papers.html

Andreou A. S. 2003. Promoting software quality through a human, social and organisational requirements elicitation process, *Requirements Engineering* 8: 85–101, Spinger-Verlag, London.

Bar-Yam, Y. 2003. *Unifying Principles in Complex Systems, In: Converging Technology for Improving Human Performance - Nanotechnology, Biotechnology, Information Technology and .Cognitive Science* (NBIC). ROCO, M. C.; BAINBRIDGE, W. S. Cambridge: Kluwer Academic Publisher, 2003. 467 p.

Gonzales, R. 2005. Developing the Requirements Discipline: Software vs. Systems, *IEEE Software*, March/April 2005, pp 59-61.

Hecht, M., Buettner, D. 2005. Software testing in space programs. *Crosslink*, v. 6, n. 3 (Fall2005). http://www.aero.org/publications/crosslink/fall2005/06.html

Hjortnaes, K. 2003. ESA software initiative. ESA Report, May, http://klabs.org/richcontent/software_content/presentations/esa_software_initiative_final_may_7.pdf

Ingham M. D., Rasmussen R. D., Bennett M. B., Moncada A. C. 2004. Generating requirements for complex embedded systems using state analysis. 55th International Astronautical Congress, Vancouver.

Johnson, C. W. 2003. *A Handbook of Accident and Incident Reporting*, Glasgow University Press, Glasgow.

Johnson, C. W. 2006. What are emergent properties and how do they affect the engineering of complex systems? *Reliability Engineering and System Safety*, v. 91, n. 12, Dec. 2006. p. 1475-1481. http://www.dcs.gla.ac.uk/~johnson/papers/ emergence.pdf.

Kauppinen, M., Kujala, S., Altio, T., Lehtola, L. 2002. Introducing Requirements Engineering: How to Make a Cultural Change Happen in Practice, IEEE Joint International Conference on Requirements Engineering, Essen. pp 43-51.

Lahoz, C. H. N., Camargo Júnior, J. B., Abdala, M. A. D., Burgareli, L. A. 2006. A software safety requirements elicitationn study on critical computer systems. 1st International Conference on System Safety, IET, London.

Lahoz, C. H. N. 2009. Elicere: O processo de elicitação de metas de dependabilidade para sistemas computacionais críticos: estudo de caso aplicado a área espacial. Doctorate thesis, Universidade de São Paulo, São Paulo.

Leveson, N. G.. 2004. The Role of Software in Spacecraft Accidents, *AIAA Journal of Spacecraft and Rockets*, Vol. 41, No. 4, July 2004.

———. 2009. *Engineering a Safer World*, Aeronautics and Astronautics, Massachusetts, Institute of Technology, Draft of New Book, 2009. http://sunnyday.mit.edu/book2.pdf

Lutz, R. R. 1993. Analyzing software requirements errors in safety-critical, embedded systems. IEEE International Symposium on Requirements Engineering, San Diego: IEEE Computer Society Press. p. 53-65

Lutz, R. R., Mikulski I. 2004. C. Ongoing requirements discovery in high-integrity systems. *IEEE Software*, March / April 2004, v. 21, n. 2. p. 19-25.

Lyons, J. L. 1996. Report of the inquiry board into the failure of flight 501 of the ariane 5 rocket. European Space Agency, Paris.

Magee, C. L., de Weck, O. L. 2004. Complex system classification. 14th Anual International Symposium of the INCOSE, International Council on System Engineering, Toulouse.

NASA-JPL, 2000. Report on the loss of the mars polar lander and deep space 2 missions. Technical Report JPL D-18709, Jet Propulsion Laboratory, California Institute of Technology.

NASA-MCO, 1999. Mars climate orbiter: mishap investigation board, phase I report. Technical report, NASA Headquarters, Washington.

Serviço Publico Federal, 2004. Ministério da Defesa, Comando da Aeronáutica, Departamento de Pesquisas e Desenvolvimento, Relatório da Investigação do acidente ocorrido com o VLS1-V03, em 22 de agosto de 2003, em Alcântara, Maranhão, São José dos Campos. `http://www.aeb.gov.br`

Sommerville, I., An Integrated Approach to Dependability Requirements Engineering, 11th Safety-Critical Systems Symposium, 2003, `http://www.cs.st-andrews.ac.uk/~ifs/Teaching/MScCritSysEng2010/Coursework%20(PDF)/DependabilityReqEngineering.pdf`

Yu, E. 1995. Modelling strategic relationship for process reengineering. Toronto. Phd. Thesis, Computer Science Department, University of Toronto, Toronto.

BIOGRAPHY

Carlos H. N. Lahoz works in Aeronautics and Space Institute (IAE) in Brazil, where coordinates two projects: the on-board software and the control bench software for the Brazilian Satellite Launch Vehicle (VLS). His PhD studies were in safety and dependability area at Escola Politecnica da Universidade de Sao Paulo (POLI/USP). He is a Assistant Professor since 2003 at Universidade Paulista (UNIP), where teaching in Computer Science graduate course. During 2005-2006 was Invited Professor in the Aerospace Master Engineering course at Instituto Tecnologico da Aeronautica (ITA). He received the Best Student Paper/Brazil award in INCOSE 2009.

Contact Information

Instituto de Aeronautica e Espaco – IAE. Praca Marechal Eduardo Gomes, 50. Vila das Acacias. Sao Jose dos Campos (SP) cep 12228-904. (Brasil); Tel: +55 (12) 39474901; Fax: +55 (12) 39475019; Email: lahoz@iae.cta.br