SpaceOps 2010 Conference
 Delivering on the Dream
 Hosted by NASA Mars
25 - 30 April 2010, Huntsville, Alabama

AIAA 2010-2288

Dependability Benchmark for PUS-based Satellite On-Board Software

Paulo C. Véras[1] and Emilia Villani[2]
Aeronautical Institute of Technology – ITA, São José dos Campos, SP, 12228-900, Brazil – {pcv, evillani}@ita.br

Henrique Madeira[3]
University of Coimbra, Portugal, 3030, Portugal – henrique@dei.uc.pt

and

Ana M. Ambrosio[4]
Ground System Division, National Institute for Space Research, São José dos Campos, SP, Brazil, 12227-010 – ana@dss.inpe.br

This paper presents a benchmark aimed to provide an unbiased basis for characterizing OBDH software of different satellites with regards to dependability attributes. Instead of considering only the delivered product, the proposed benchmark covers a set of criteria for accepting or not OBDH artifacts created along the lifecycle development. The benchmark is based on the ECSS European standards that guide both the development lifecycle and the services to be provided by OBDH software.

I. Introduction

IN the last few decades, the software role in space embedded computing systems has increased. However, the attention and effort dedicated to its design and validation have not increased in the same way. As a result, software has also played a key role in most of the main recent space accidents[1].

In order to change this scenario, software dependability must be a major concern of space projects. In this context, this paper proposes a dependability benchmark for satellite embedded software. The dependability benchmark is the specification of a procedure to assess a set of measures usually related to software behavior in the presence of faults.

The embedded computing systems of a satellite are in the Attitude and Orbit Control System (AOCS), in the On Board Data Handling (OBDH) system, and in payload equipment control. This proposal is restricted to the OBDH software. (OBDH dependability benchmark – OBDH D-Bench). OBDH software manages the communication between the satellite and the ground station, as well as monitors and controls the status of almost all satellite subsystems in order to maintain proper health of the satellite.

The European Cooperation for Space Standardization - ECSS[15] provides a set of standards to support the development of space products. These standards cover a broad range of application area, such as mechanical, software engineering, control engineering and ground system.

Although ECSS is a European initiative, these standards have also been adopted in Brazil and other countries around the world.

Among the ECSS standards, ECSS-70-41A[2] proposes the Packet Utilization Standard, also known by its acronym PUS. This standard addresses the utilization of telecommand and telemetry for the remote monitoring and control of spacecraft. It defines a set of services that covers all the fundamental requirements for spacecraft operation, as well as the structure of the packets used in the communication between the spacecraft and the ground segment (i.e. the control centre in Earth) and the content of these packets.

[1] PhD Student, Department of Mechanical Engineering, Praça Marechal Eduardo Gomes, 50, Vila das Acacias
[2] PhD Professor, Department of Mechanical Engineering, Praça Marechal Eduardo Gomes, 50, Vila das Acacias
[3] PhD Professor, Department of Informatics Engineering, Polo II, Pinhal de Marrocos
[4] PhD Professor, Ground System Division, Av. dos Astronautas, 1758, Jardim da Granja.

The PUS is considered the starting point for the definition of the dependability benchmark in our work. The benchmark being proposed by this paper is named "OBDH D-Bench".

The OBDH D-Bench aims to provide an unbiased basis for characterizing the OBDH software of different satellites with regards to dependability attributes. Its main motivation is to be a set of criteria for comparing OBDH solutions and approve or not OBDH artifacts for a new space mission.

Although the OBDH services are standardized, OBDH software components cannot yet be bought as COTS (commercial off-the-shelf) components. Each new mission generally develops (or contracts the development of) a new OBDH. The software development cycle is organized in phases. At the end of each phase, the software development team must submit a set of documents to revision. The development cycle, the revision milestones, and the documents delivered at each revision are also regulated by the ECSS standards, such as Ref. 3.

In this context, a dependability benchmark that assesses only the OBDH software delivered at the end of the full development cycle is not sufficient, as the detection of problems at this stage may seriously compromise the mission schedule. Because of that potential impact, the target of the proposed OBDH dependability benchmark is not only the OBDH software but also the other artifacts submitted for approval at the space mission milestones. The proposed dependability benchmark is therefore composed of a set of dependability benchmarks, each one associated to a specific revision milestone of the space mission. The purpose is to evaluate the quality of the OBDH and detect problems as early as possible in the stages of the mission lifecycle. This proposal is only possible because the ECSS standards define not only the services to be implemented by the OBDH but also the artifacts submitted to each space mission revision.

The next section describes the main related work recently developed by academic and industry projects. Section 3 introduces the services provided by an OBDH and presents the development cycle and review milestones defined by the ECSS standards. It also briefly describes the artifacts delivered at each revision. Section 4 proposes the dependability benchmark. Section 5 presents some conclusions and details suggested future work.

II. Related Work

One of the first projects on dependability benchmarks was the DBENCH project, funded by the EU Fifth Framework Programme, from 1998 to 2002. This project focused on benchmarks to evaluate and compare dependability of COTS and COTS-based systems in embedded, real-time, and transactional systems. This project attempted to standardize an approach to evaluate and compare different computer systems and components. Examples of benchmarks for embedded systems are Ref. 4, which focused on Real-time Kernels in On-board Space Systems and Ref. 5 that addressed Automotive Systems.

Since then, different benchmarks have been proposed both in academia and industry. A benchmark for database and transactional server systems was proposed in Ref. 6. Sun Microsystems has proposed a high-level framework to availability benchmarks[7]. This has led to the definition of two benchmark systems: robustness on handling maintenance events such as replacement of a failed hardware component or the installation of a software patch and system recovery[8]. IBM has also invested in a benchmark to quantify the automatic capability at computing system level[9].

In Ref. 10 a benchmark to assess a human-assisted recovery process is defined while in Ref. 11 and Ref. 12 benchmarks for WEB is addressed.

Recently, the Transaction Processing Performance Council (TPC)[13], an organization composed of major vendors of database and transaction processing software, is focused on defining benchmarks for such applications. In 2008, the AMBER Project[14], funded by the EU Seventh Framework Programme (FP7), aimed to prepare a research agenda for resilience assessment for the Information and Communication Technologies associating benchmarking, assessment, and measurements aspects.

To date, we don't have knowledge of a proposal that suggests a dependability benchmark taking into account the full development lifecycle and the artifacts produced by the software development process.

III. Standards Applied to Develop OBDH Software

According to ECSS standards, the development of OBDH software follows a pre-defined lifecycle, produces a set of documents[3], and implements the software as services specified in Ref. 2. Details of the lifecycle, documents, and services are given as follows.

A. The Lifecycle

The production of a space system is organized into customer-supplier relationships[15]. The space software development is defined in the ECSS-E-40 standard[3]. In the general case, the customer is the procurer of the software and the hardware for a system. Reviews are the main interaction points between the customer and supplier. Project reviews are examinations of technical status of a project and associated issues at a particular point in time. The reviews relevant to the software engineering processes are the System Requirement Review (SRR), Preliminary Design Review (PDR), Detailed Design Review (DDR), Critical Design Review (CDR), Qualification Review (QR) and Acceptance Review (AR). The detailed description of each review can be found in Ref. 16. The reviews occur at different levels in the customer-supplier hierarchy and are sequenced according to the overall system level planning. The occurrence of these reviews is subject to the achievement of milestones during the planned sequence. The software engineering processes, together with their reviews and milestones, are planned in relation to the immediate higher-level development processes.

The first review to be performed is the SRR, whose products to be analyzed are the IRD (Interface Requirement Document), the RB (Requirement Baseline) and the DJF (Design Justification File). In the second review (PDR) the documents to be generated are the TS (Technical specification), the Interface Control Document (ICD), the DJF, which is an evolution of the same one generated in the previous review, and the DDF (Design Definition File). In the following reviews, the products submitted are evolutions of the artifacts in the previous reviews.

The ECSS Standard does not imply the adoption of some specific software life cycle. Instead, it provides some constraints related to the execution of the processes. Figure 1 shows the processes and reviews defined by the ECSS Standard. The rectangles represent the processes throughout the software life cycle, the circles represent the reviews, and the arrows represent the product flow. A brief description of the processes is given in the following.

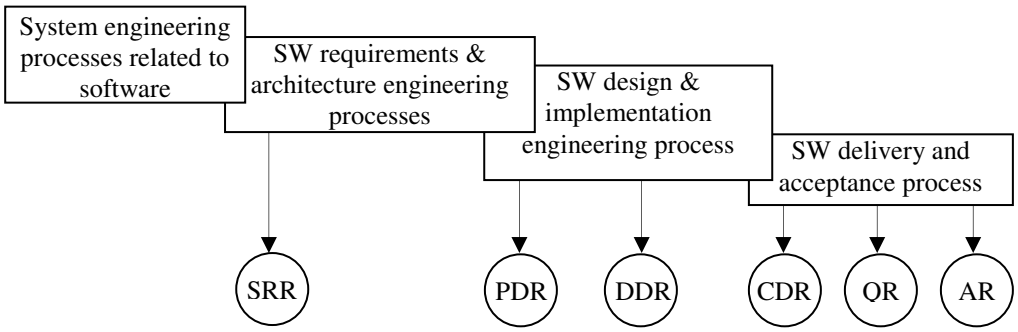

Figure 1. Simplified overview of the software life cycle processes (adapted from Ref. 3)

System engineering processes related to software

These processes produce the information for input to the SRR. It establishes the functional and performance Requirements Baseline (RB) needed for the software development.

Software requirements and architecture engineering process

This process consists of the elaboration of the Technical Specification (TS), which is the supplier's response to the requirements baseline. This process can start in parallel with or after the elaboration of the requirements baseline. The technical specification contains a precise and coherent definition of functions and performance for all levels of the software to be developed. The preliminary Interface Control Document (ICD) is generated at this time. The Design Justification File (DJF), whose content is the result of all significant trade-offs, feasibility analyses, make-or-buy decisions and supporting technical assessments, is generated at this time too. This process is completed by the PDR.

Software design and implementation engineering process

Although this process can start before the PDR, it is after the PDR when the results of the requirements and architecture engineering process are reviewed and baselined. These are used as inputs to the design and implementation engineering process. All elements of the software design are documented in the Design Definition File (DDF), which contains all the levels of design engineering results (including software code listings). The

rationale for important design choices and analysis and test data that show that the design meets all requirements is added to the DJF during this process. The results of this process are the input to the CDR, which signals the end of the design activities.

Software delivery and acceptance process

This process can start after the CDR and when the software validation activity is complete with respect to the technical specification. This process includes an AR, with the DJF as input. The AR is a formal event in which the software product is evaluated in its operational environment. It is carried out after the software product is transferred to the customer and installed in an operational setting. Software validation activities terminate with the AR.

B. The OBDH software and the Package Utilization Service (PUS) Standard

The most important functions provided by the OBDH software of a satellite are: (i) supervision of the satellite communication busses; (ii) reception, checking and decoding of telecommands from the ground station; (iii) storage of telecommands for a pre-defined period; (vi) collection and buffering of measurements made by other units and subsystems; (v) organization, encoding and storage of telemetry frames; (vi) downlinking of telemetry to the ground station; (vii) management and distribution of time signals.

In order to standardize this set of functions, the European space agency created the *Package Utilization Services (PUS)* Standard[2]. The PUS identifies a number of operation concepts that imply distinct sets of capabilities to be implemented on-board the satellite along with corresponding monitoring and control facilities on the ground. These sets of capabilities constitute "services", the monitoring and control of which shall be achieved via a well-defined set of interactions between the provider of the service (i.e. an on-board application process) and the user(s) of the service (e.g. a ground system). The services provided by the standard are defined according to their commonality, coherence and implementation independency (i.e. a service corresponds to a group of functionalities applicable to many missions; the capabilities to be provided by a service are closely related and their scope should be unambiguously specified; and a service should neither assume nor exclude a particular satellite on-board architecture (hardware or software).

Each service (currently a set of services) has a minimum capability set and can have one or more additional capability set(s). The minimum capability set corresponds to the minimum coherent implementation of that service.

The PUS defines the application-level interface between ground and space in order to satisfy the requirements of electrical integration, testing, and flight operations. The flight operation is defined for the purpose of remote monitoring and control of subsystems and payloads. PUS is applicable to any mission, independent of its domain of application and independent of its orbit or ground station coverage characteristics. However, it may only be partially applicable to a given mission. The operational concepts and corresponding services contained in the PUS cover a wide spectrum of operational scenarios. Subsets of these operational concepts and services may be applied to a given mission. The choice of what subset shall be used in a new mission shall be made early in its design phase, resulting in the need to tailor the PUS to comply with the requirements of the particular mission.

PUS provides the idea of an application process, which is an entity, uniquely identified by an *application process identification* (APID), capable of generating telemetry source packets and receiving telecommand packets. Different APIDs shall be assigned to different on-board data sources or sinks and also within the same source or sink where there are packet streams with different bandwidth characteristics and operational significance.

C. The Telecommand Verification Service: an example

In order to illustrate the OBDH Dbench, proposal, the "telecommand verification" service (whose service type is 1) will be used as an example. According to Ref. 2, the telecommand verification service provides the capability for explicit verification of each distinct stage of execution of a telecommand packet, from on-board acceptance through to completion of execution. Figure 2 shows this service in the context of a space mission.

The telecommand processing is made of the following stages:

a) *Acceptance of the telecommand by the destination application process* - All checks to be applied to the telecommand packet prior to the start of execution shall be performed. This shall include verification that the telecommand has not been corrupted and that the application process supports the service (or sub-service) requested.

b) *Start execution of the telecommand* - This stage shall be performed immediately after reception, but for a complex command it may also be delayed pending some level of pre-execution validation. The checks performed shall include feasibility and validity checks, such as confirmation that the telecommand parameters are correctly encoded, are within their range of values, and are valid for the current state of the service.
c) *Progress of execution* - The various steps which reflect the progress of execution of the telecommand shall be telecommand-specific in nature.

The telecommand verification service shall generate a report if a telecommand fails at any of its identified stages of execution. It shall also generate a report of successful completion of the same stages, if this has been requested in the acknowledgment flags in the telecommand packet header. These reports shall provide auxiliary data for the ground system to fully understand the report (e.g. to identify the nature and cause of a telecommand failure).

The minimum capability set of this service shall consist of the telecommand acceptance report of success and telecommand acceptance report of failure. PUS provides a set of additional capabilities that are optional to the telecommand verification service.

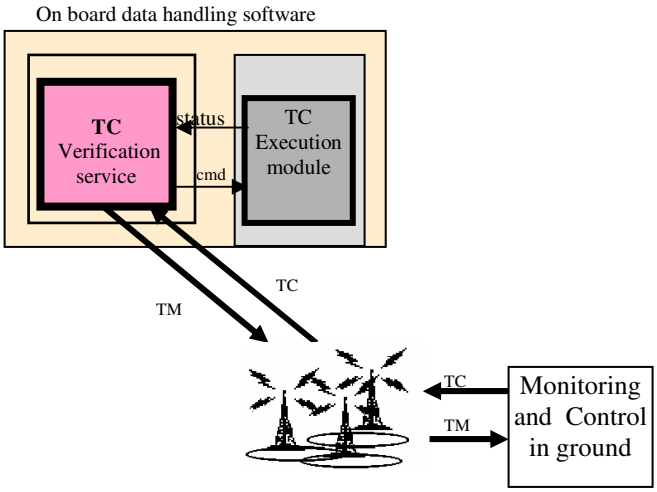

Figure 2. Telecommand verification service.

IV. The Proposed OBDH Dependability Benchmark

A. The Structure of the OBDH D-Bench

The OBDH dependability benchmark (OBDH D-Bench) is composed of a set of benchmarks, each one associated with a specific review of the software engineering processes of a space mission (see Fig. 1). At each review, the corresponding benchmark must assert how dependable the final product would be by analyzing the set of artifacts provided to review. It must check if the artifacts contain evidence that the dependability attributes have been taken into consideration in the previous phase of the development cycle. The techniques used in each benchmark correspond to the kind of artifact submitted to the review. If the artifact is a list of requirements, the verification technique is to check if the requirements contain the appropriate answers to a set of questions. If the artifact is a software product, the verification technique is a set of test cases. The metrics proposed for each benchmark quantify the results of the review. They will allow to precisely deciding for a pass/not pass at the end of each review. If approved, the project can go to the next phase. If not, the artifacts must be redone. Software quality metrics are not included in the OBDH D-Bench.

The benchmarks that compose the OBDH D-Bench are:

a) SRR (System Requirements Review) D-Bench
b) PDR (Preliminary Design Review) D-Bench
c) DDR (Detailed Design Review) D-Bench
d) CDR (Critical Design Review) D-Bench
e) QR (Qualification Review) D-Bench
f) AR (Acceptance Review) D-Bench

B. The OBDH D-Bench Dimensions

The specification of the OBDH dependability benchmark follows the proposal of the DBENCH project[17] and is based on three benchmark dimensions: *categorization*, *measure,* and *experimentation*. Categorization includes the specification of benchmark context and the benchmark target. Measure includes the definition of the type and nature of the measures. Experimentation includes the specification of the system under benchmark (SUB), the workload, and the fault load.

The OBDH D-Bench context is the embedded software normally implemented for an On Board Data Handling (OBDH) system of a satellite. It is important to highlight that the OBDH D-Benchs focus on the application software that implements the PUS services. It does not include the operating system or the hardware platform.

In order to illustrate the OBDH D-Benchs target and system, we detail the SRR D-Bench. The SRR D-Bench Target (SRR-BT) is the set of OBDH software documents related to the implementation of the PUS services and submitted to the System Requirements Review (SRR). According to Ref. 3, these documents are the IRD, the RB, and the DJF (see section III). Particularly, the RB contains the mission requirements related to the OBDH software and the list of PUS services that must be implemented in the OBDH software, with the appropriate refinement.

The system under benchmark (SRR-SUB) is composed of the SRR-BT and other documents that may be necessary to check the consistency of the target but are not under review. The documents that are part of the target in the SRR D-Bench will be part of the SUB in the next benchmark, the PDR_SUB, as they are needed to compare if the new documents are consistent with the previous ones. The relationship between BT and SUB of two benchmarks is illustrated in Fig. 3.

The workload and the faultload at this D-Bench are checklists of questions whose answers must be found in the SRR BT. In order to better illustrate the workload and faultload of the SRR D-Bench we use as an example the "telecommand verification service" of the PUS (service type 1), which has already been described in Section 3.

According to Ref. 17, the workload represents a typical operational profile of the system. In the case of the OBDH D-Benchs, it means to check if the system under design is consistent with the PUS specification of ECSS.

For the telecommand verification service, examples of questions that compose the workload for the SRR DBench (RB, IRD and DJF documents) include:

- Does the specification of the service sub-types satisfy the minimum set defined by the ECSS-E-70-41A standard (i.e. does it include the telecommand acceptance reports for the cases of success and failure)?
- Is the specification of the service sub-types consistent with the mission needs? Example: if the sub-types are restricted to the minimum set, does any command of the telecommand list require a report of execution completed?
- Does the specification define the points that are left open by the ECSS-E-70-41A standard? Example: does it define the possible values and corresponding meaning for the parameters field of the telemetry source packet of the failure telecommand acceptance report, according to the nature of the telecommand?

The faultload consists of a set of faults and exceptional conditions that are intended to emulate the real threats to the system. In the case of the *telecommand verification service, e*xamples of questions comprising the faultload include:

- Does the RB specify the possible failures for the telecommand acceptance routine? Does it use the ECSS-E-70-41A standard for the code in the telemetry source packet? The ECSS-E-70-41A standard is:
  - 0 = illegal APID (PAC error);
  - 1 = incomplete or invalid length packet;
  - 2 = incorrect checksum;
  - 3 = illegal packet type;
  - 4 = illegal packet subtype;
  - 5 = illegal or inconsistent application data.
- Does the RB specify the length of the telecommand packet for each telecommand?

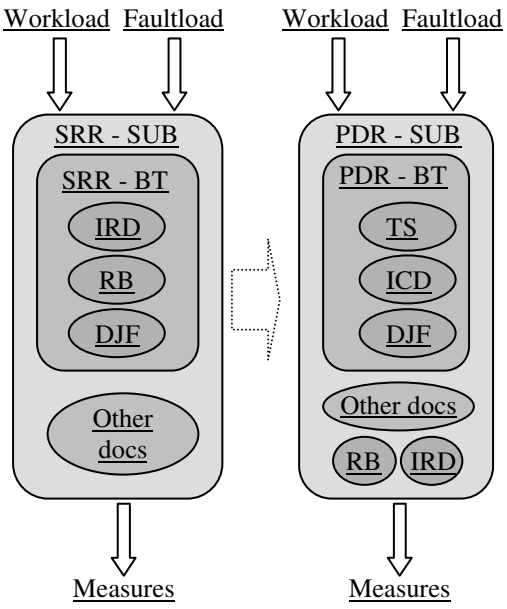

Figure 3. Benchmark schema overview.

Given a certain service, the definition of the faultload is based on knowledge of experts and on the analysis of review reports of previous missions. Examples of classes of fault considered in the faultload are *hardware faults to be tolerated*, *missing treatment of field values exceptions, etc.*

It worth observing that in the experimentation the faults are not injected, they are checked against an artifact.

The SRR D-Bench and the other OBDH D-Benchs must be tailored to the mission needs. Obviously, if a mission does not implement one specific service or sub-services, the regarding questions are not applied. It is important to observe that although the OBDH D-Benchs requires a tailoring to different missions, it is still suitable for comparing

different OBDH software because each service is always evaluated in the same way for all the missions, not matter how complex the OBDH is.

There are two sources for the definition of the workload and faultload. The first one is the service definition in the ECSS-E-70-41A. The questions of this type derive from the service functionality. If we consider a V cycle of development for the OBDH, the questions of this type that are applied to a text artifact on a D-Bench on the left leg of the V (e.g. SSR D-Bench) are the test cases applied to the software product on the same level on the right side of the V (e.g. AR D-Bench).

The second type of question is based on the information the artifacts must contain and describes the minimum content of each document. These questions are specific to each OBDH D-Bench.

The measures obtained by the application of the workload and faultload are YES/NO answers, which are equivalent to binary results of a set of test cases. The questions are also divided into 'excluding' and 'not excluding' questions. A positive answer for the excluding questions is mandatory for a 'pass' result. Example: if the RB does not include the minimum set of sub-services for the telecommand verification service, the result of the SRR D-Bench is 'not pass', even if all the other answers are positive. Once the results of the excluding questions have been analyzed, the not excluding questions are applied to the artifacts. The answers of the not excluding questions are weighted in an algebraic expression of the SRR D-Bench coefficient that reflects the dependability of the OBDH at this stage of design. A minimum value for the SSR D-Bench must be achieved for approval in this design review, otherwise the artifacts must be corrected (re-done) before the project is allowed to go on.

C. The OBDH D-Bench Validation

This section describes how the key benchmarking properties of repeatability, portability, representativeness, scalability, and simplicity of use are addressed by the OBDH D-Bench.

Repeatability guarantees statistically equivalent results when the benchmark is run more than once in the same environment (i.e., using the same SUB, the same workload and faultload). The repeatability of the OBDH D-Bench depends on the objectiveness of the questions that compose the benchmark. The questions must be formulated such that the possible answers can be enumerated. Furthermore, the difference between the possible answers must not be subject to the interpretation of the person that applies the benchmark. When two or more people are evaluating the same set of artifacts they must give the same answers to the same questions.

The portability of the OBDH D-Bench is achieved if the set of the benchmark questions can be applied to development processes of different enterprises. This portability is at the same time assured and restricted by the ECSS standards. It is important to observe that the issues that are left open in the standards will not be evaluated by the benchmark.

The representativeness of the OBDH D-Bench is obtained by an appropriate specification of the workload and faultload. The definition of the workload is strongly related to the ECSS standards and can be considered a relatively simple task. The most difficult point is the definition of the faultload. It must consider all important fault modes of the OBDH. For this purpose the database used for the faultload definition is based on historic data of different missions. A field study has being performed in order to verify the most frequent errors on space software requirements. This will allow the use of realistic errors in the definition of the faultload. This work has been done as a research task, but in the future it will be maintained by the engineering coordination at INPE.

The scalability property is assured by the possibility of making a tailoring of the OBDH D-Bench. This tailoring has a one-to-one relationship with the tailoring of the ECSS standard that is done in the beginning of the OBDH design. While the design tailoring chooses the services that shall be implemented for a specific mission, the benchmark tailoring chooses the services that shall be verified, which are obviously the same.

The simplicity of use of the OBDH D-Bench is not assured. It depends on the format of the artifacts submitted to revision, which is not defined in the ECSS standards.

Finally, the complete validation of the OBDH D-Bench will be achieved through its extensive use.

V. Conclusion and Future Work

This paper proposes a dependability benchmark for the OBDH application software of satellites. The new feature of the benchmark is that it encompasses not only the OBDH software product but also a set of artifacts generated during the software lifecycle. The OBDH D-Bench is composed of a set of dependability benchmarks (D-Benchs) associated with the project review points, structured according to the relevant ECSS standards. Each D-Bench estimates the dependability of the final software product by analyzing the artifacts submitted to review. The result is a 'pass'/'not pass' recommendation. If not approved the software development must be revised before going to the next development phase.

The OBDH D-Benchs provide generic ways of characterizing the typical faults that can be found in the contents of each artifact produced in the previous phase. The questions injected in the artifact check its completeness, redundancy, and consistency in order to measure the dependability of the final product.

Currently, the OBDH D-Benchs are under refinement. The next steps are to test them in a number of space software projects and establish the appropriate values that result in the product approval in each phase.

Acknowledgments

We thank Dr. Brian Duncan, from the Applied Physics Laboratory in Laurel, Maryland, US, co-chair for the SpaceOps conference for the excellent review and comments done. We also thank CAPES and Brazilian Space Agency (AEB)/ITASAT Project for the funds to support this research. Thanks the team of University of Coimbra, Portugal, by the suggestions and motivation.

References

[1]Leveson, N., "Role of software in spacecraft accidents", *Journal of Spacecrafts and Rockets*, Vol. 41, No. 4, 2004, pp. 564-575.

[2]ECSS Space engineering: Ground systems and operations – Telemetry and telecommand packet utilization, ECSS-E-70-41A Standard, 2003.

[3]ECSS Space Engineering: Software – Part1: Principles and requirements, ECSS-E-40 Part 1B Standard, 2003.

[4]Madeira, H., Some, R., Moreira, F., Costa, D., and Rennels, D., "Experimental evaluation of a COTS system for space applications", *The International Conference on Dependable Systems and Networks*, DSN, Bethesda, Maryland, USA, June 2002.

[5]Ruiz, J. C., Yuste, P., Gil, P., and Lemus, L., "On Benchmarking the Dependability of Automotive Engine Control Applications", *IEEE/IFIP International Conference on Dependable Systems and Networks*, Florence, Italy, June 2004.

[6]Wilson, D., Murphy, B., and Spainhower, L., "Progress on defining Standardized Classes for Comparing the Dependability of Computer Systems", *Workshop on Dependability Benchmarking*, Washington, D.C., USA, 2002, pp. F1-5.

[7]Zhu, J., Mauro, J., and Pramanick, I., "R3 - A Framwork for Availability Benchmarking", *International Conference on Dependable Systems and Networks*, San Francisco, CA, USA, 2003, pp. B-86-87.

[8]Mauro, J., Zhu, J., and Pramanick, I., "The System Recovery Benchmark", *Pacific Rim International Symposium on Dependable Computing,* Papeete, Polynesia, 2004.

[9]Lightstone, S. et. al., "Towards Benchmarking Autonomic Computing Maturity", *Proceedings of the. First IEEE Conference on Industrial Automatics*, Banff, Canada, Aug. 2003.

[10]Brown, B., Chung, L., Kakes, W., Ling, C., and Patterson, D. A., "Experiences with evaluation human-assisted recovery processes", *Proceedings of the International Conference on Dependable Systems and Networks*, June 2004

[11]Durães, J., Vieira, M., and Madeira, H., "Dependability Benchmarking of Web-Servers", *Proceedings of the 23rd International Conference on Computer Safety, Reliability and Security*, Potsdam, Germany, Sept. 2004.

[12]Vieira, M., Laranjeiro, N., and Madeira, H., "Benchmarking the Robustness of Web Services", *Proceedings of the 13th IEEE Pacific Rim Dependable Computing Conference*, Melbourne, Victoria, Australia, Dec. 2007.

[13]Transaction Processing Performance Council. URL: http://www.tpc.org [cited 25 February 2010].

[14]AMBER – Assessing, Measuring and Benchmarking Resilience. URL: http://www.amber-project.eu [cited 25 February 2010].

[15]ECSS system – Description, implementation and general requirements, ECSS-S-ST-00C Standard, 2008.

[16]ECSS Space Project management: Organization and conduct of reviews, ECSS-M-30-01A Standard, 1999.

[17]Kanoun, K., and Spainhower, L., *Dependability Benchmarking for Computer Systems*, Wiley-IEEE Computer Society Press, Hoboken, New Jersey, 2008.