Data Model and Tool Support for a consistent Functional Verification Chain in Space Projects

Bartha, Sven[(1)]; Osborne, Steve[(2)]; Dr. Richters, Mark[(3)]; Plaßmeier, Frank[(4)]; Sondermann, Heiner[(5)]; Nicklaußen, Dirk[(6)] Hummel,Günter[(7)]

Astrium GmbH[(1)]
88039 Friedrichshafen, Germany
E-mail: Sven:Bartha@astrium.eads.net

European Space Agency, ESTEC[(2)]
Keplerlaan1 -PO Box 299 - 220 AG Noordwijk ZH, The Netherlands
E-mail:Steve.Osborne@esa.int

Astrium GmbH[(3)]
Airbus Allee-1, 28199 Bremen, Germany
E-mail: Mark.Richters@astrium.eads.net

Astrium GmbH[(4)]

Airbus Allee-1, 28199 Bremen, Germany
E-mail: Frank.Plassmeier@astrium.eads.net

Astrium GmbH[(5)]
88039 Friedrichshafen, Germany
E-mail: Heiner.Sondermann@astrium.eads.net

Astrium GmbH[(6)]
Airbus Allee-1, 28199 Bremen, Germany
E-mail: Dirk.Nicklaussen@astrium.eads.net

Astrium GmbH[(7)]
88039 Friedrichshafen, Germany
E-mail: Guenter.Hummel@astrium.eads.net

ABSTRACT

In the frame of the ESA study "Automation of Space System Test Data Collection, Processing and Reporting" Astrium has:

- analyzed and documented the processes and roles related to functional verification
- developed a conceptual datamodel
- investigated the various possible S/W approaches to implement a tool to supporting the functional verification chain
- investigated implications on existing tools
- started to develop a software demonstrator

Besides the direct implementation into dedicated tools, the results of the study could also be of interest to define interfaces for toolboxes around the future European Ground System Common Core (EGS-CC). This paper focuses on the first two tasks of the study. Together with the conceptual datamodel a configuration control approach called "island of information" was defined, in order to keep trace of changes of data and groups of data in the model. In order to allow domain experts rather than S/W experts to participate in the discussion a system analysis approach was followed using simple and easy to read notations.

INTRODUCTION

"The overall objective of verification is to demonstrate, through a dedicated process, that the deliverable product meets the specified requirements"[1]. Consequently verification starts with a set of applicable requirements and ends with a Verification Control Document (VCD), which provides the necessary close-out information for each requirement.

Although looking simple and straightforward, verification is often a significant percentage of the overall effort invested in a space project. In particular functional verification of requirements to be verified by tests is often underestimated and hence a frequent cause for schedule slippage and cost overrun.

A significant effort in functional verification goes into mapping of data to support questions like:

- Is each requirement verified?
- A requirement has changed, which tests need be re-specified?
- Which part of the Check-Out S/W implements which part of the test specification?
- Which test report corresponds to which procedure, which test specification and which requirements?
- Which NCRs and which engineering documentation/data are related to which test result?

Very often project specific approaches, processes and tools are applied, leading to unnecessary re-discussions, tool re-development or adaptations.

The goal of the ESA Study [2] is to form the basis for a tool set optimizing the way how all of the interconnected data related to functional verification can be generated and maintained. It should automate all generation of derived data, all mappings and all consistency checks as far as possible. This includes generation of all functional verification ECSS documents, such as Test Plans, Test Specifications, Test Procedures, Test Reports and Verification Control Documents.

A very important opportunity for the development of European tools are future tool boxes around the EGS-CC. Defined interfaces could greatly boost cooperation and efficiency within the European space industry.

Task 1: Identify and Document Processes and Roles of Functional Verification

The task was carried out following a two-step approach:

Step 1: Define the inputs and outputs of all processes of functional verification. This was implemented based on DeMarco Dataflow Diagrams, as used in classical structured analysis.

Step 2: Define who conducts which task under which conditions and at which point in time, i.e. define the roles and the time sequence of processes. Step two was implemented using Business Process Model Notation (BPMN).

Fig. 1 shows the top level data flow diagram of the Functional Verification Process. It consists of "Verification Engineering", which is understood as the definition of the Test Plan and the Test Specification, "Test Preparation", "Test Execution", "Test Data Post Processing", "Mapping of Requirements to Close-Out", which is the VCD contribution of functional verification. It includes

furthermore "Test Bench Engineering", which is understood as definition and integration of the test benches. A test bench is understood as the complete assembly of items under test (e.g. flight units, Onboard S/W …) and the environment necessary to conduct the test (e.g. SCOEs, Simulators …).

The processes "Verification Engineering", "Test Bench Engineering" and "Test Preparation" have been further refined into lower level of dataflow diagrams (not shown in this paper). Separate dataflow diagrams have been setup for "As Built Process", "Configuration Control", "Configuration control S/W" and "Configuration Control H/W". The external interfaces of functional verification have been documented in a level 0 context diagram.

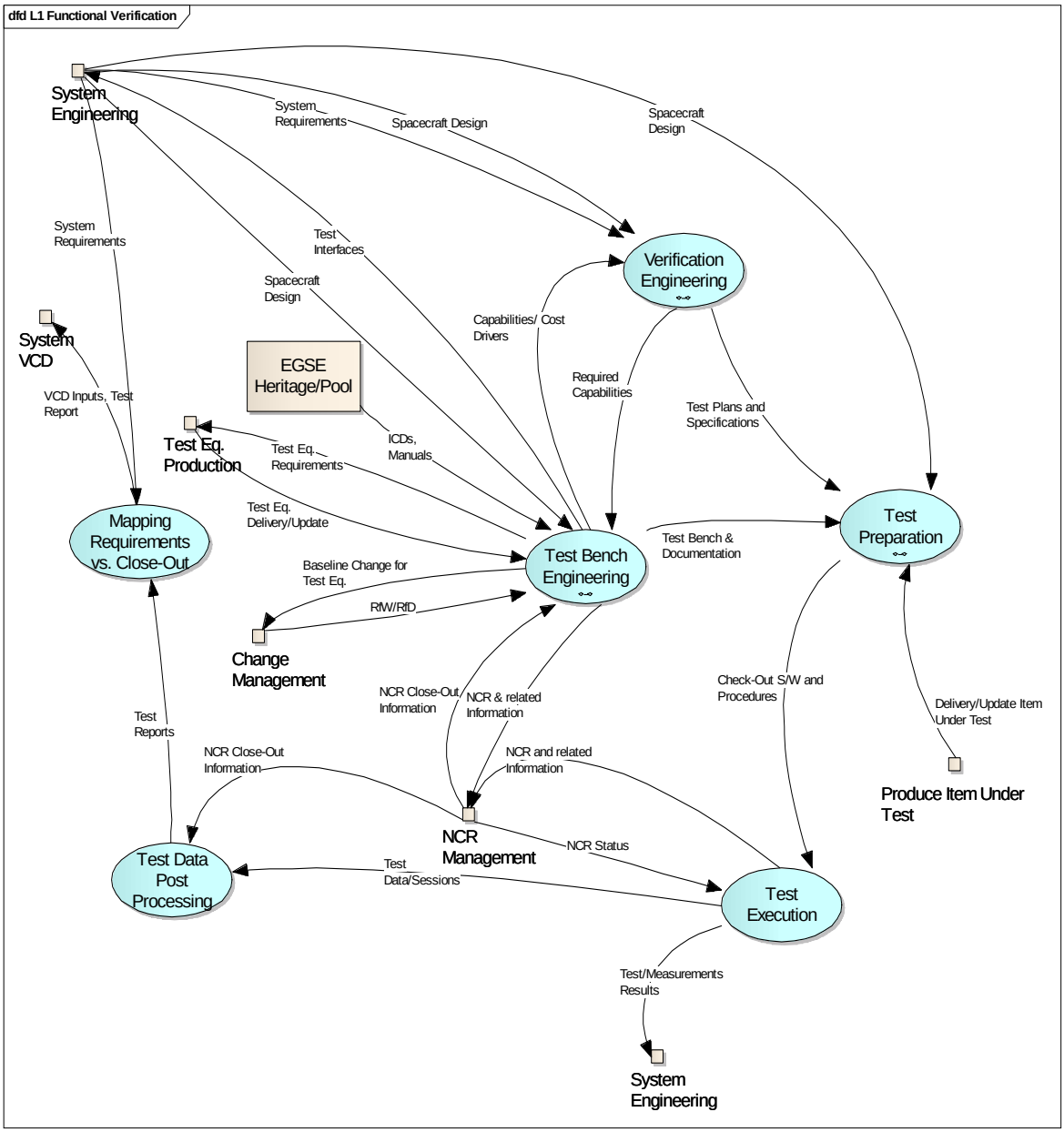

Fig. 1 Level 1: Functional Verification

The diagrams might look over-crowded, but in fact a lot more implicit dataflows exist, which are not shown for simplicity:

- Feedback: Whenever data is fed as input to a process, it might be rejected with the request for modification/correction.
- Review/Approval: Typically data exchanged between processes is reviewed and/or approved by third party (quality, customer, independent experts, etc)
- Configuration Control: Whatever data is formally exchanged is usually also subject to formal configuration control.
- Schedule Monitoring: All processes are monitored and scheduled by project management and hence have an interface into this process.

Fig. 2 shows the top level business model process diagram of functional verification starting from requirements and ending with acceptance. The time axis in this figure is top down, each column is related to one role. Small diamonds indicate processes branching and converging. An additional plus sign is added to the diamonds when parallel execution takes place. In the absence of the plus sign only a single branch is executed, depending on the indicated condition.

The following roles have been identified: "Customer", "System Engineer" (in charge of S/C design), "Verification Engineer" (in charge of test planning and definition), "Test Engineer" (in charge of test preparation and execution), "AIT Manager" (in charge of managing integration and test schedule and logic), "Item under Test Supplier" (in charge of developing a part of the S/C, which is subject to functional verification) and "Test Bench Engineer" (in charge of development an integration of test benches). Fig. 2 shows that establishment of the verification plan & spec, the production of the items under test and the definition/development of the test benches are executed in parallel. This makes it apparent, why this early phase of functional verification is often difficult in case of low heritage projects. Similar to the dataflow diagrams also the BPNM diagram have been simplified for readability. Implicit flows not shown are: feedback, iterations due to versions and staggered delivery of information, modifications of schedule and logic due to NCRs, baseline changes, and other circumstances. Most of this has been treated by generic diagrams, i.e.: "Baseline Change Process", "Concurrent Engineering", "Hierarchical Coordination", "Feedback (Formal Review)", "Feedback (Informal & Continuous)", "NCR process", "Request for Waiver".

Specific refined BPMN diagrams have been setup for: "User Requirements to Acceptance", "Establish Verification Plan", "Establish Verification Plan (Req. Driven)", "Test Equipment Engineering, Production, Development Follow-Up & Acceptance", "Test Equipment. Setup & Integration" and "Test Preparation".

Besides having a defined and clear way on how functional verification processes can be assigned on roles and ordered in time, a further conclusion from the BPMN analysis can be drawn: A workflow oriented tool to support functional verification is very likely not flexible enough for application in real world projects. A data driven tool, where the user can edit any information at any point in time allowing the user to trigger consistency checks as required seems to be more appropriate.

It is further noticeable, that two slightly different ways of establishing a Verification Plan resulted from discussions. The first option starts from the requirements ("the text book approach") the alternative option starts from the functional decomposition. The later approach is more likely to support re-use of verification plans and specifications from one project to another.

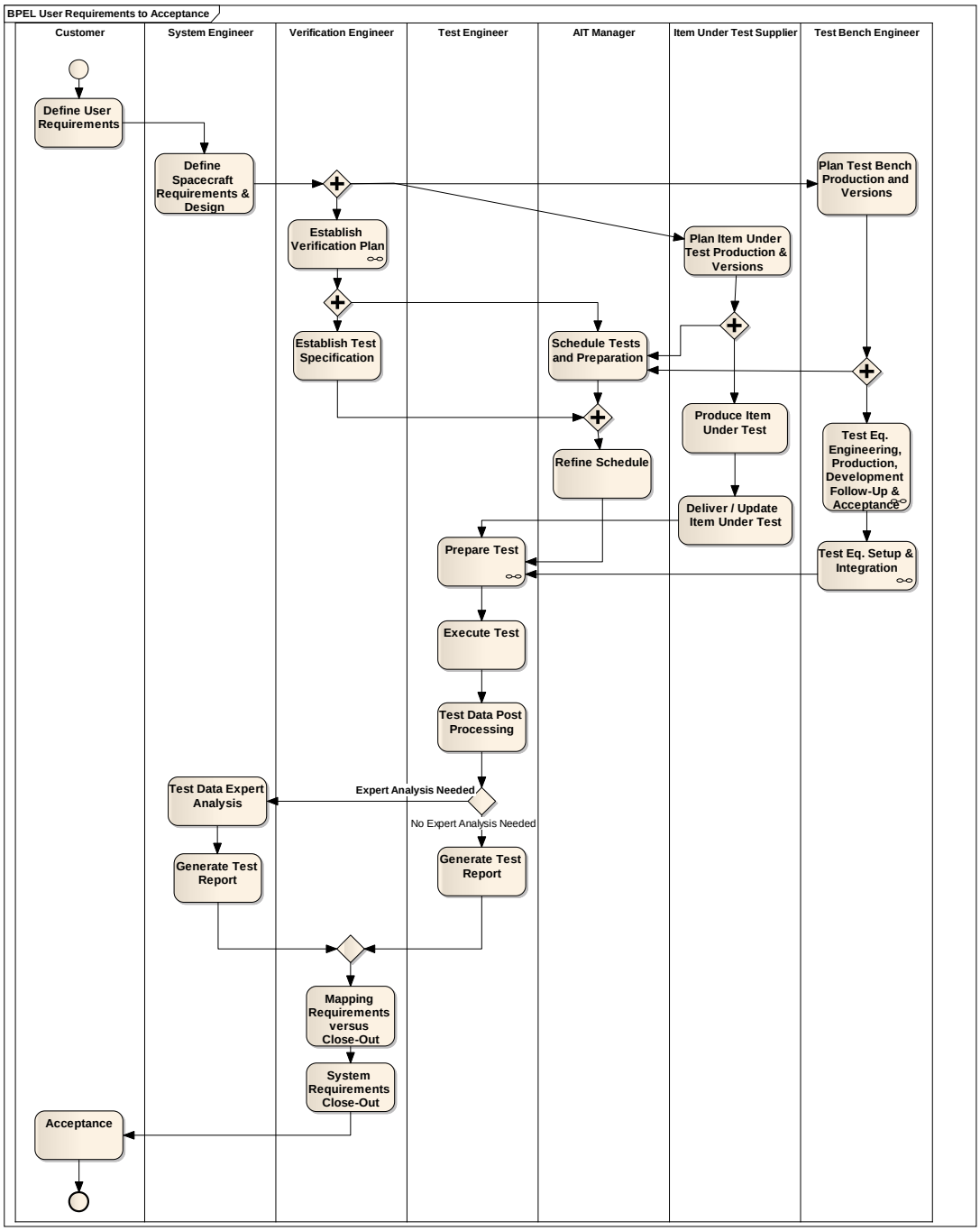

Fig. 2 Business Process Model of Functional Verification

Task 2: Develop a Conceptual Data Model of Functional Verification

Task 2 was conducted in three steps:

- Conceptual data model on entity level, get the multiplicities right

- Discussion of versioning / config control of the database itself
- Detailed data content of the entities

For the conceptual data model the UML (Unified Modelling Language) class diagram notation was used. However it was very much restricted for simplicity reasons. Each class can be seen as a table, the only connection between classes allowed was associations with the multiplicity fully defined. Object oriented features like inheritance have not been used. This allows direct implementation of the datamodel into either relational databases or object-oriented environments. It furthermore keeps the diagrams simple in the sense that each line has the same meaning (an association) and each box has the same meaning (a class / a table).

The resulting overall data model (see Fig. 3) contains 46 entities (= classes or tables) and about 80 associations, most of them representing many-to many relationships. From an S/W point of view this could be considered a small to mid-size data model, i.e. not overly complex in implementation and maintenance. On the other hand a user of the data would not be able to have the model fully in mind when working with the data. Hence a good graphical user interfaces and support for consistency checks are mandatory.

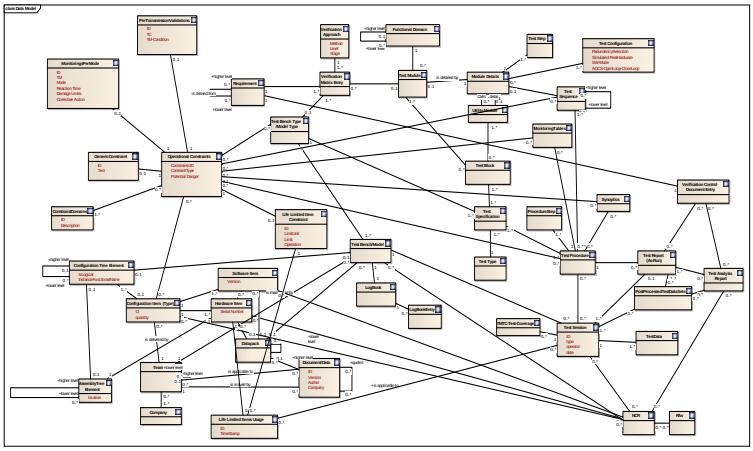

Fig. 3 Complexity of the Overall Data Model

Task 2 explains the overall datamodel collecting closely subsets of the entities into separate diagrams, such that in the end all entities and all association have been explained. For example the entities, which go into the verification plan, are shown in Fig. 4

The following Views on the Conceptual Data Model have been generated: "Verification Plan", "Test Specification", "Test Preparation & Execution", "Verification Control", "Operational Constraints", "Configuration Control".

Versioning / Config Control of the Database itself

As can be seen in Fig. 3 all data in the conceptual data model is directly or indirectly connected. However not all data is generated at the same time and even worse some of the data forms the baseline for other later data to be filled. (e.g. the data which goes into the verification plan would be approved and "frozen", before the test specification is issued). It must be possible to check consistency of early phase approved baselines with late phase data. It must also be possible to check impacts of changes in

"approved" parts of the data on the data under work. Last, but not least different teams could be working on different baselines.

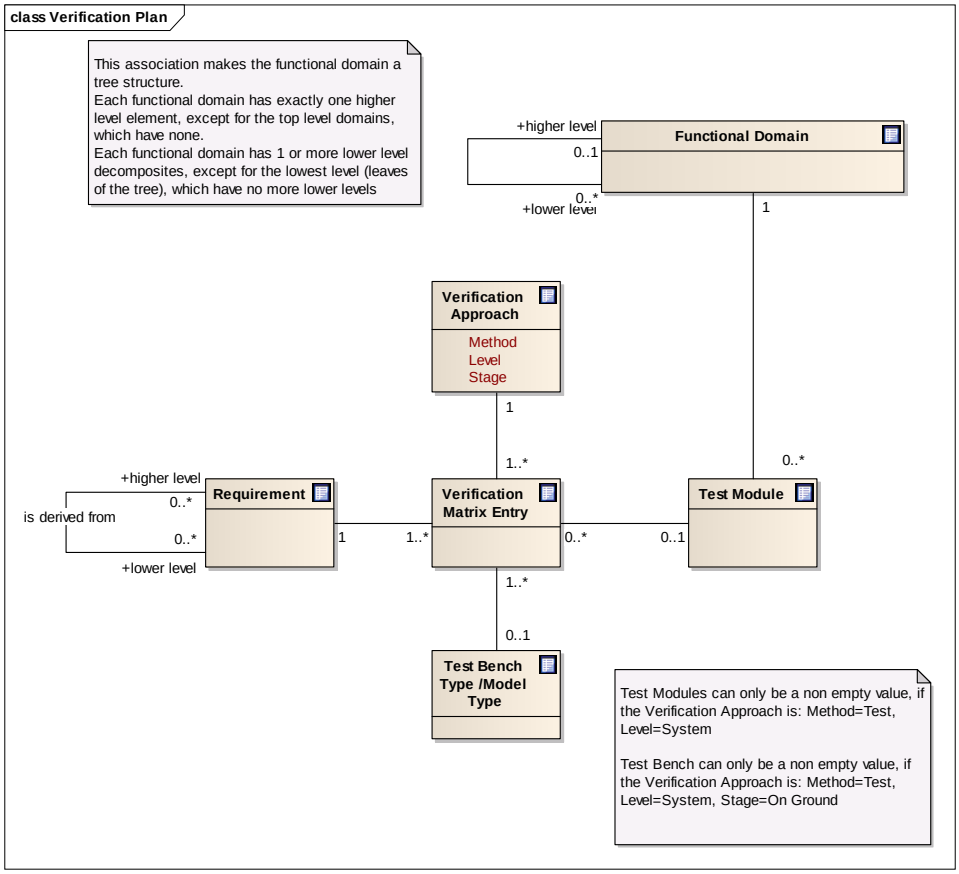

Fig. 4 Verification Plan Data Model

As a consequence the possibility to freeze the whole database at milestones and to give this one version name will not be sufficient.

For a typical space project the overall model will contain thousands of small pieces of information, hence a different independent version for every small piece of information will not work either.

Therefore a feature is needed allowing to assign versions to data subsets of the database. These subsets are called “islands of information” in the study.

Definition:

An “island of information” is a set of objects (rows in tables) serving a common purpose and hence requiring a separate version control.

Example: All Data which goes into the Test Specification of AOCS Normal Mode Testing.

Two different types of Islands exist. Type 1 (Simple Type): All objects of one or more classes form an island. Example: Baselines in DOORS. Type 2 (Deep Search Type): The island starts with one object (table entry) and collects all other directly or indirectly associated objects out of set of classes. Example: Objects that form Specification Island AOCS_TSPE_V2_0.

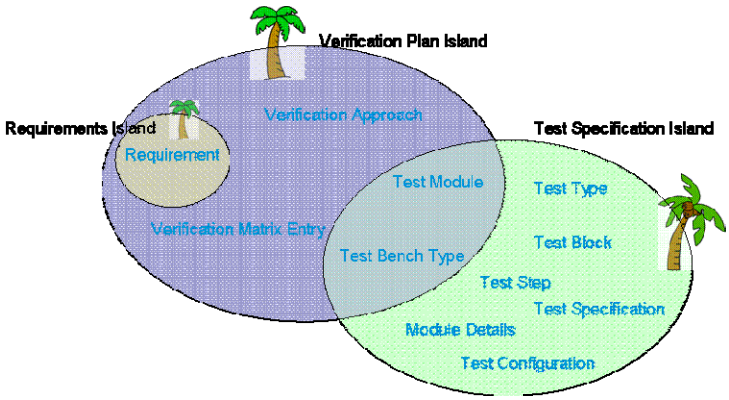

Fig. 5 Examples of some Islands of Information.

The following rules for islands of information can be defined:

- The intersection between “Islands of Information" is not empty.
- Two islands are consistent, if the objects in the intersection have the same version
- It is not permitted to have “Offshore-association”, i.e. association between entities of different islands, if not in the intersection

A detailed set user of requirements for the implementation of "islands of information" has been established. The "island of information" principle would need to be extended in case concepts like inheritance or aggregation are used in the conceptual data model.

REFERENCES

[1] ECSS E-ST-10-02C Verification standard

[2] "Automation of Space System Test Data Collecting, Processing and Reporting", ESTEC Contract 4000102822, Steve Osborne

[3] "Test Automation Process Model Description", AUS.TN.ASD.ENG.00001, issue 1a, 2.12.2011, S.Bartha

[4] "Test Data Specification Across The Life Cycle", AUS.TN.ASD.ENG.00002, issue 2b, 8.12.2011, S.Bartha,

[5] "Specification of Informatics Tools for the Support of Space System Verification Processes", ASD-TN-03, issue 1, 22.06.2012, F. Plaßmeier

[6] "Implications For Current Processes And Tools", AUS.TN.ASD.ENG.00004, issue 1, 24.07.2012, G.Hummel