ECSS-E-TM-10-23A
25 November 2011

Space engineering

Space system data repository

ECSS Secretariat
ESA-ESTEC
Requirements & Standards Division
Noordwijk, The Netherlands

Foreword

This document is one of the series of ECSS Technical Memoranda. Its Technical Memorandum status indicates that it is a non-normative document providing useful information to the space systems developers' community on a specific subject. **It is made available to record and present non-normative data, which are not relevant for a Standard or a Handbook.** Note that these data are non-normative even if expressed in the language normally used for requirements.

Therefore, a Technical Memorandum is not considered by ECSS as suitable for direct use in Invitation To Tender (ITT) or business agreements for space systems development.

Disclaimer

Published by:	ESA Requirements and Standards Division ESTEC, P.O. Box 299, 2200 AG Noordwijk The Netherlands
Copyright:	2011© by the European Space Agency for the members of ECSS

Change log

ECSS-E-TM-10-23A 25 November 2011	First issue

Table of contents

Figures

Introduction

The present Technical Memorandum forms part of the system engineering discipline (ECSS-E-10) of the space engineering branch of the ECSS system.

One of the responsibilities of the system engineering discipline is to “ensure the availability of product and process data which enables the complete system to be produced, tested, delivered, operated, maintained, and properly disposed of” (see ECSS-E-ST-10C, clause 5.6.1e).

In order to achieve this ECSS-E-ST-10C clause 5.6.3 requires:

> *“The system engineering organization shall define the engineering data to be stored in a data repository.”*
>
> *“The system engineering organization shall ensure that engineering data can be exchanged in electronic form between the different organizations in charge of the elements of the product decomposition levels via agreed and validated interfaces.”*

This Technical Memorandum establishes a formal specification of the semantics of the engineering data in order to enable the creation of engineering data repositories with standardised data exchange interfaces – a so-called *space system data repository* – thereby allowing interoperability between data repositories managed by different stakeholders. The intended normative parts of this Technical Memorandum are computer-interpretable and independent of any specific computer platform.

This Technical Memorandum is intended to evolve into an ECSS Standard in the near future. For the time being, it is not yet possible to establish a standard that has the maturity and industrial validation required for application in new or current space projects. In conjunction with related development and validation activities, this Technical Memorandum should be regarded as a mechanism for reaching consensus prior to building the standard itself.

What is meant by “space system data repository” in this context needs to be explained. Clearly it is impossible, and indeed undesirable, to create a single “data repository” to be imposed at some point in the future on all European space projects. Given the many different requirements on the kind of data to be exchanged for different (families of) space systems, the variety of ICT implementation technologies and the large number of different stakeholders in the space industry, such an approach would be entirely unfeasible and a guaranteed recipe for failure.

An implementation of the “space system data repository” is therefore expected to be a federation of different physical databases that are logically integrated in an interoperable architecture so that data can be exchanged effectively and reliably.

The future “space system data repository” standard needs to be stable over a long period of time. Computer technology and tools are evolving rapidly, so it is important to be able to upgrade the implementation technology without changing the data itself. This is possible in principle since the meaning (semantics) of space system data remains quite stable over time. In order to support long-term data archiving, any revisions, extensions or improvements of the standard should be “backwards-compatible” with existing datasets that comply with previous versions of the standard.

The expected benefits of such a standard include:

- a more efficient and effective space system lifecycle through better integration,
- risk reduction,
- reuse of previous developments,
- improved data quality, consistency and traceability.

This Technical Memorandum introduces the overall framework in which the following ECSS standards and technical memoranda can be used:

- ECSS-E-TM-10-20 "Space engineering – System engineering – Product data exchange" which addresses product data exchange.
- ECSS-E-ST-70-31 "Space engineering – Ground systems and operations – Monitoring and control data definition" which defines space system monitoring and control data (a subset of the engineering data addressed here).
- ECSS-E-TM-10-25 "Space engineering – System engineering – Engineering design model data exchange" which covers the exchange of engineering design model data in Phase 0 and Phase A.

1
Scope

This Technical Memorandum specifies the semantics of the data needed to support the engineering processes as specified in ECSS standards, thereby enabling the concept of a *space system data repository* to capture all engineering data produced during the complete space system lifecycle.

The objective of this Technical Memorandum is to define a mechanism whereby engineering data can be reliably and efficiently exchanged:

- Between all suppliers and customers involved;
- Between all engineering disciplines;
- At all levels of space system decomposition;
- Within and across all space system life cycle phases.

For this purpose, the Technical Memorandum provides the following:

- A conceptual data model which specifies the semantics of the relevant data;
- Requirements on the individual components of the space system data repository;
- Requirements on the integrated space system data repository.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this ECSS Technical Memorandum. For dated references, subsequent amendments to, or revision of any of these publications do not apply. However, parties to agreements based on this ECSS Technical Memorandum are encouraged to investigate the possibility of applying the more recent editions of the normative documents indicated below. For undated references, the latest edition of the publication referred to applies.

ECSS-S-ST-00-01	ECSS – Glossary of terms
ECSS-S-ST-00	ECSS System – Description, implementation and general requirements
ECSS-M-ST-10	Space project management – Project planning and implementation
ECSS-M-ST-40	Space project management – Configuration and information management
ECSS-E-ST-10	Space engineering – System engineering general requirements
ECSS-E-TM-10-20	Space engineering – System engineering – Product data exchange
ECSS-E-ST-40	Space engineering – Software
ECSS-E-ST-70	Space engineering – Ground systems and operations - Principles and requirements
ECSS-E-ST-70-31	Space engineering – Ground systems and operations - Monitoring and control data definition

3
Terms, definitions and abbreviated terms

3.1 Terms defined in other ECSS documents

For the purpose of this Technical Memorandum, the terms and definitions from ECSS-S-ST-00-01C apply, in particular for the following terms:

configuration baseline

space system

3.2 Terms specific to the present Technical Memorandum

3.1.1. conceptual data model

data model that captures the end-user needs in end-user terms

3.1.2. data

representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means

NOTE From [IEEE Std 610.12-1990] Standard Glossary of Software Engineering Terminology.

3.1.3. data model

representation in a formal data modelling language of the data objects of significance in a domain of application, the characteristics of significance of those data objects and the associations of significance between those data objects

3.1.4. data model module

grouping of related data model items in order to manage data model complexity

3.1.5. data repository

place where data is stored and maintained for later retrieval

NOTE The repository is typically an ICT system with data management capabilities and interfaces for data storage, retrieval, query, deletion, etc.

3.1.6. global conceptual data model

conceptual data model for the complete system life cycle and all disciplines

3.1.7. logical data model

data model organised in terms of a particular data management technology

EXAMPLE Relational model, object oriented model, hierarchical model.

3.1.8. physical data model

data model that defines how the data is physically stored on a computer

3.1.9. system element

an element of the (space) system with which data is associated

3.3 Abbreviated terms

The following abbreviations are defined and used within this technical memorandum:

Abbreviation	**Meaning**
API	application programming interface
CDF	concurrent design facility
CDR	critical design review
CIM	computation-independent model (MDA)
DDR	detailed design review
E/R	entity/relationship
ICT	information and communication technology
ITT	invitation to tender
MDA	model-driven architecture (OMG)
MMI	man-machine interface
ODBC	open database connectivity
OMG	object management group
ORM	object-role modelling
PDR	preliminary design review
PIM	platform-independent model (MDA)
PSM	platform-specific model (MDA)
RFP	request for proposal
SQL	structured query language
SSUM	space segment user manual
UML	unified modelling language (OMG)
W3C	world wide web consortium
XMI	XML metadata interchange (OMG)
XML	extensible mark-up language (W3C)
XSD	XML schema definition (W3C)

4 Background and concept

4.1 Introduction

Reliable electronic exchange of the data needed during the lifecycle of a space system is essential to further improve the efficiency and effectiveness of the engineering processes, and indeed all other lifecycle activities and processes.

Unfortunately, as many projects have experienced too often, getting the right data at the right moment to the right stakeholder is an enormous challenge. Finding the right version of the data and details of changes from previous versions can be very inefficient and frustrating. Making data available in a form that is suitable for exchange can be difficult, if not impossible.

Now that all individual disciplines, both engineering (e.g. mechanical, electrical, optical, radiofrequency communication, software) and non-engineering (e.g. cost, project management) in space projects have well established methods and tools, the next round of improvements needs to come from better integration of the lifecycle processes and the project teams. Figure 1 below shows the different disciplines involved during the space system lifecycle, as defined in ECSS.

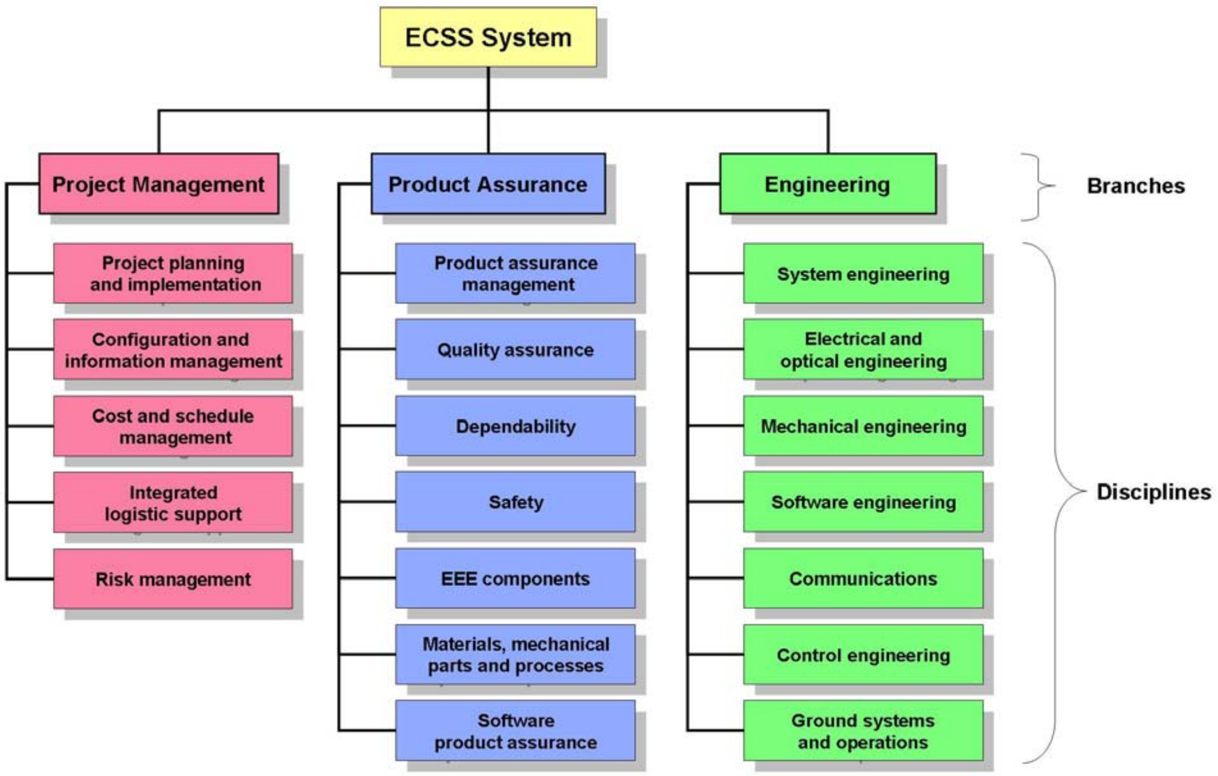

ECSS System
Project Management
Product Assurance
Engineering
Branches
Project planning and implementation
Configuration and information management
Cost and schedule management
Integrated logistic support
Risk management
Product assurance management
Quality assurance
Dependability
Safety
EEE components
Materials, mechanical parts and processes
Software product assurance
System engineering
Electrical and optical engineering
Mechanical engineering
Software engineering
Communications
Control engineering
Ground systems and operations
Disciplines

Figure 1 ECSS branches and disciplines derived from ECSS-S-ST-00C

Furthermore, in order to increase the efficiency of developing complex systems, concurrent and collaborative engineering processes and integrated multi-disciplinary project teams are increasingly applied. The customer-supplier relationship is more and more supported by, and dependent upon, reliable electronic data exchange e.g. through computer-assisted processes for procurement, design reviews, validation and verification.

There is also a trend away from a "document-centric" approach towards a "model-centric" or "model-based" approach in which all data is managed electronically.

This is taking place across the "extended enterprise", between the space system procurement agency and the prime-contractor, between prime- and sub-contractor, and so on down the complete customer-supplier[1] chain – see Figure 2[2] below.

[1] Depending on the procurement approach for a given space system element, customer and supplier could actually belong to the same organisation.

[2] Derived from ECSS-S-ST-00C, Figure 6-1.

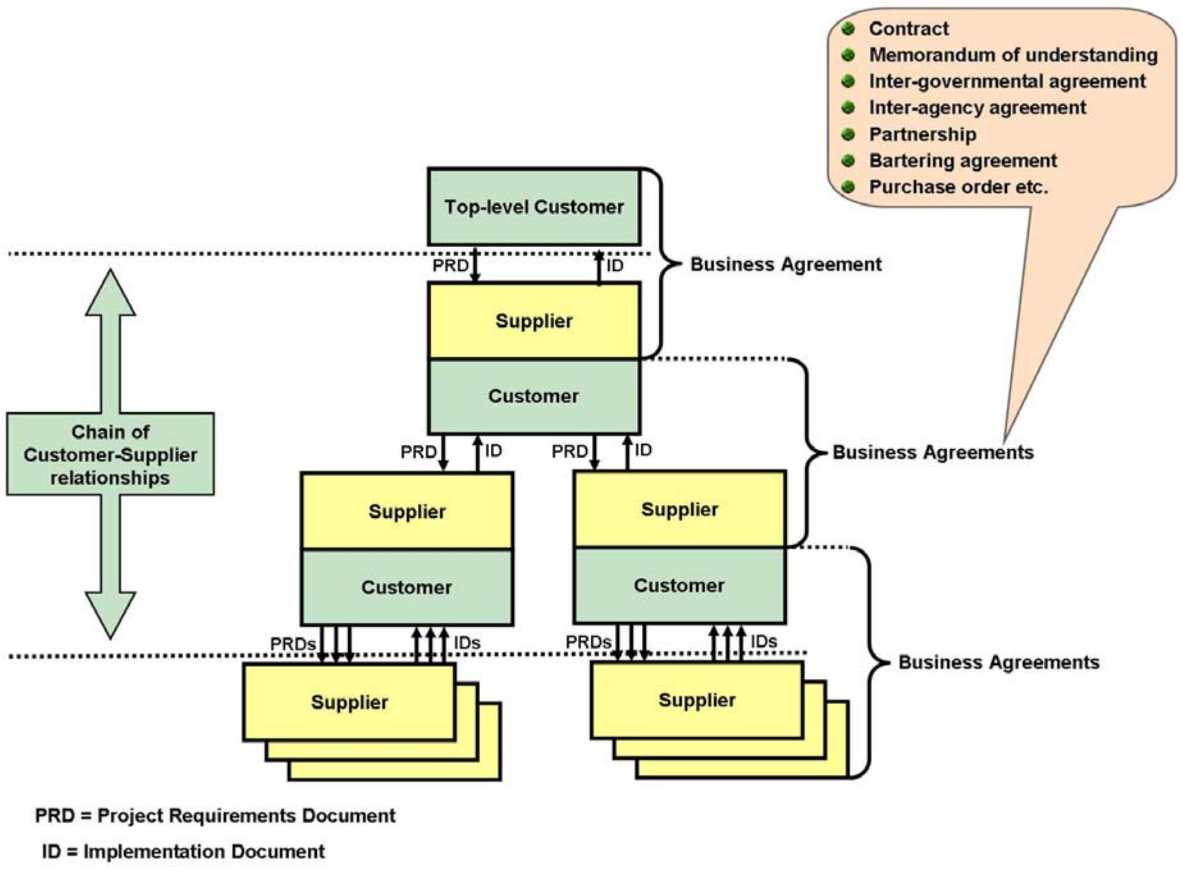


Figure 2 The customer-supplier chain of a space system project

The space industry is not unique in these aspects, the same is happening in other industrial sectors where complex systems are produced, such as aeronautics, defence, process-industry including exploration and mining, automotive, building-and-construction, and consumer electronics. Therefore, solutions and lessons learned from other industrial sectors need to be taken into account.

4.2 Software applications supporting the space system life cycle

4.2.1 Generic components

Any process within the space system life cycle uses and produces data. These processes are supported by many different software applications. All such applications include a data repository which provides the persistent data storage and other components such as:

- data import and export interfaces;
- external software application interfaces;
- man-machine interfaces;
- report generation;
- data consistency checking;
- data configuration/version control.

Figure 3 below shows a typical application and its data management components.

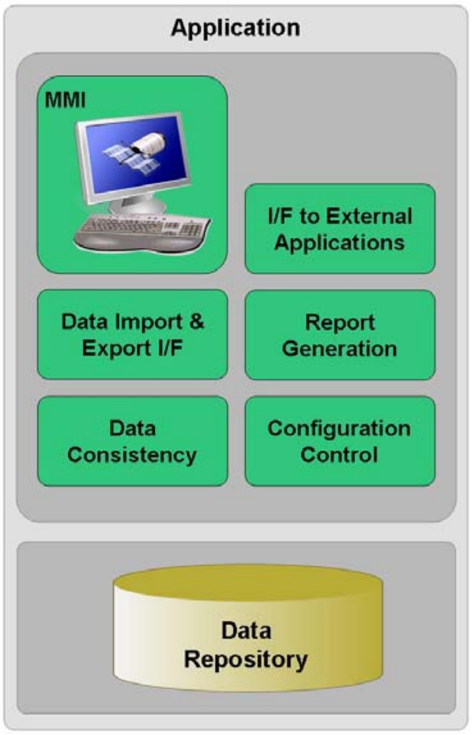


Figure 3 A view of a typical application and its data related components

4.2.2 Data exchange

During the lifetime of a given element of the space system, data is exchanged between different stakeholders for various purposes, for example:

- publishing of the ITT/RFP, including requirement specifications (customer → potential suppliers) at the start of the product procurement;
- submission of proposals (potential suppliers → customer);
- elaboration of design, cost and schedule data throughout the development lifecycle of the product e.g. PDR, DDR, CDR data packages (supplier ↔ customer);
- inter-discipline exchange e.g. between thermal and structural engineering, between software engineering and operations engineering (at supplier or customer level);
- delivery of final design and operational data along with the final product delivery (supplier → customer);
- reporting data (telemetry) and activity (telecommand, procedures) history data exchanged for evaluation and troubleshooting purposes during product operation (operator → space system customer and/or product supplier).

Data exchange (see Figure 4 below) can be:

- A one-time transfer of sets of data exported from one data repository and imported into another e.g. a file transfer.

  NOTE The sets of data exchanged in this manner are referred to as an electronic database in ECSS-M-ST-40C, clause 4.3.2.6 and ECSS-E-ST-70-31C, clause 6.1.

- Dynamic data sharing between two or more active software applications, e.g. repositories connected through web services.

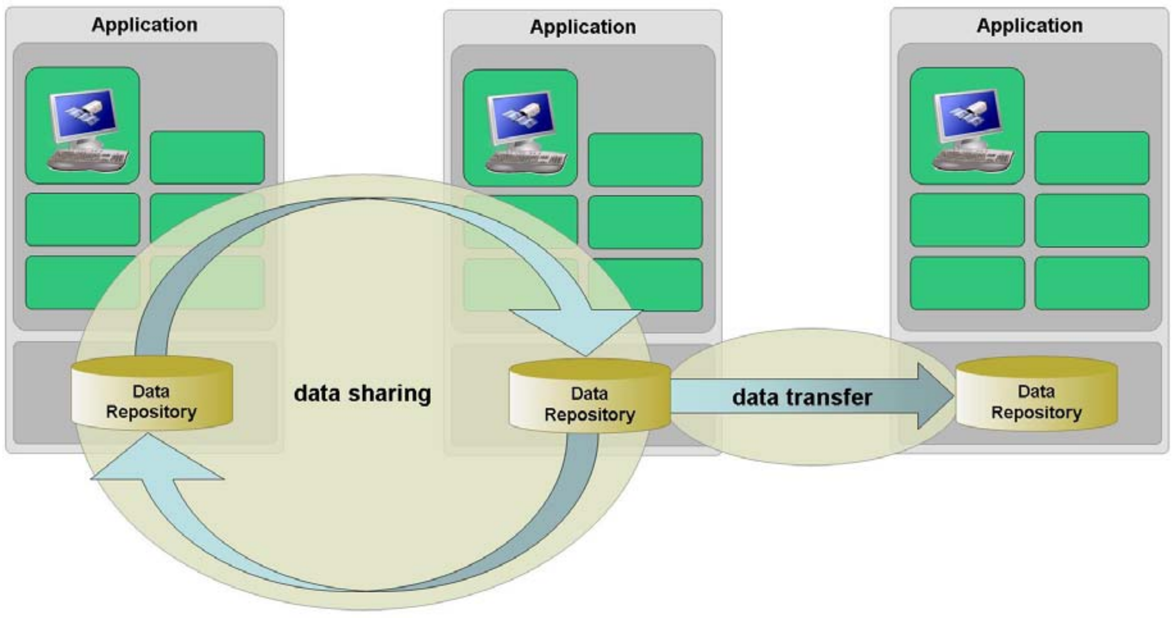


Figure 4 Data exchange scenarios between software applications

The reliable and efficient exchange of data between software applications requires, as a minimum, that the semantics of the data exchanged is the same for all parties involved. This implies that:

- It is impossible to standardise the database tools that are used across the whole space industry, due to the different business goals of the customers and suppliers, the existence of legacy systems and the cost and schedule impacts of developing software applications,.
- It is also ***not desirable*** to standardise the database tools because:
  - The use of different tools promotes innovation and healthy competition between the tool developers, and therefore ultimately benefits the end-users.
  - Subcontractors and lower-tier suppliers should not be required to master the complete set of tools that are used by all their customers and prime contractors. Being obliged to use many different tools, and to acquire and maintain skills in each of them, often implies prohibitive licence and training costs, in particular for small and medium size enterprises.

As a consequence of the foregoing, standardisation of the data to be exchanged needs to be achieved at the semantic level. To ensure that standardisation is long-lived in a rapidly changing technological world, it shall be independent of any specific software implementation technology i.e. it is inappropriate to standardise at the level of explicit exchange file formats, APIs or protocols such as XML/XSD, ODBC or Web Services.

If such semantic interoperability is achieved between software applications, then the concept of a unique space system data repository can be realised, consisting not of a single centralised and monolithic database, but rather as a federation of geographically- and temporally-dispersed physical databases that are logically integrated in such a way that data can be exchanged effectively and reliably between them (see Figure 5 below).

Furthermore, in order to comply with the principle of single source master data and for practical reasons, in many data exchange scenarios there is the need to copy only a subset of the data from one software application’s database within the space system data repository to another, while still retaining valid and resolvable links between all objects. This necessitates persistent “external references”, i.e. a consistent mechanism for referencing objects that exist outside the database that contains the specific link.

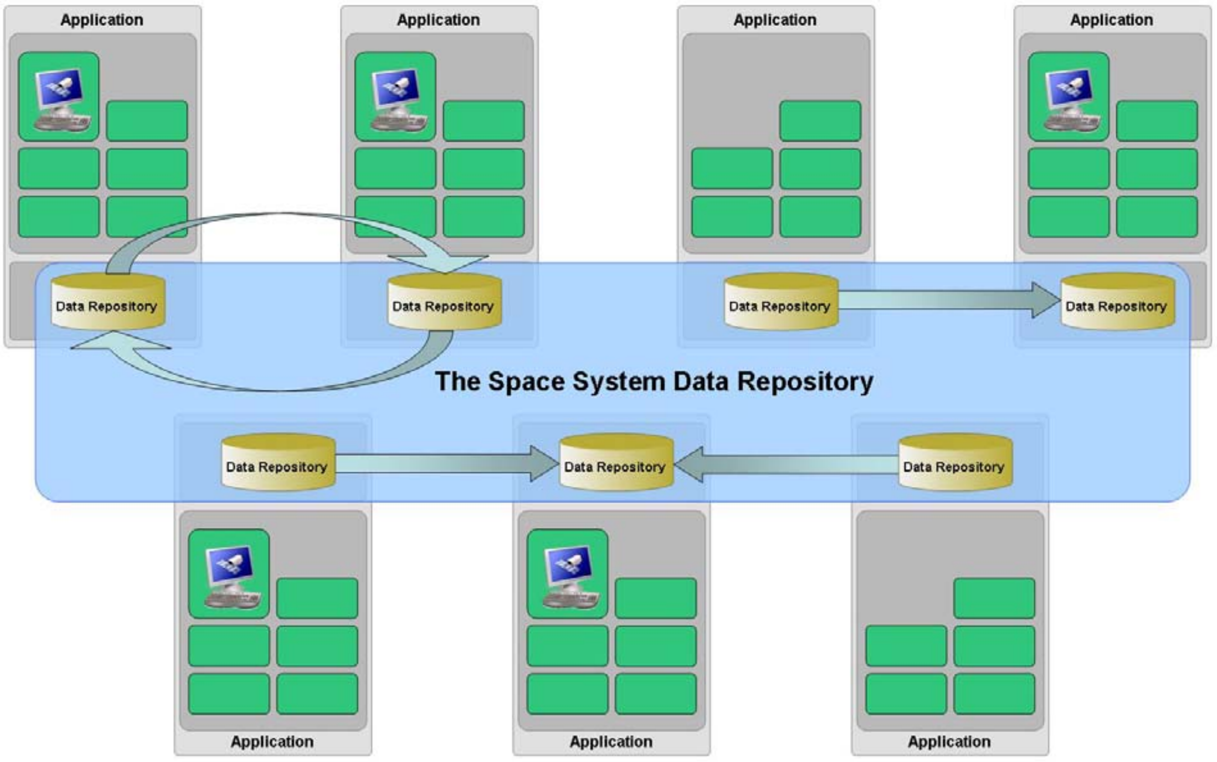


Figure 5 The "logical" space system data repository

4.2.3 Data configuration management

Data configuration management is a vital activity in a space system project comprising the establishment of baselines, configuration control and version control.

One of the objectives of the space system data repository is to enable the evolution of the configuration and version control processes defined in ECSS-M-ST-40 from a traditional document-centric approach to a data-centric approach. The different software applications that constitute a space system data repository should adhere to a common configuration management approach and contain a standard set of configuration and version control data.

4.2.4 Development lifecycle of space system software applications

In order to identify how semantic interoperability between multiple software applications can be achieved, it is necessary to consider first the overall development lifecycle of an individual software application.

The ECSS-E-ST-40 development processes are applicable to the development of a software application, namely: software requirements, architectural design, software design, implementation, validation, operation and maintenance.

In the particular case of database development, it is also necessary to model the data i.e. to create data models by the application of formal modelling rules. This data modelling is performed following a lifecycle that is based on a three-tier hierarchy of data models (also known as schemas), consisting of, in sequence:

- a conceptual data model
- logical data models
- physical data models

A conceptual data model specifies the semantics of the data relevant to a domain. It defines the data of significance to the end-users, its characteristics and the associations between data.

A logical data model is developed from a conceptual data model and is expressed in terms of a particular data modelling methodology, for example as relational tables and fields or as object-oriented classes and properties.

A physical data model is developed from a logical data model and is used to define the implementation details for storing the data objects on a computer.

Different logical and physical data models are required for the data repository and for the database interfaces (e.g. MMI, import/export etc.).

Figure 6 below shows how the data models map to the Object-Role Modelling (ORM) methodologies and to the Object Management Group's (OMG) Model-Driven Architecture (MDA). Annex C presents a wider overview of methodologies and a mapping between their main concepts.

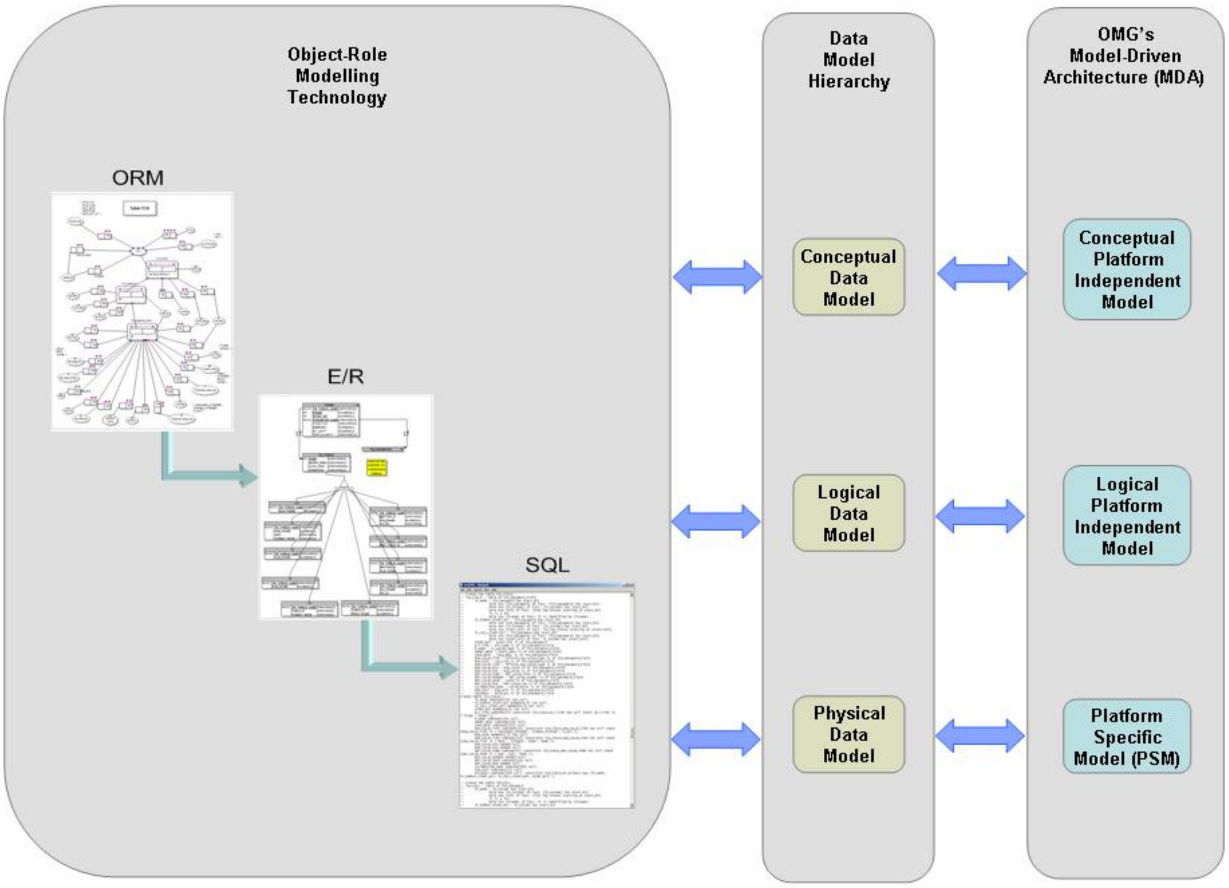

Object-Role Modelling Technology
ORM
E/R
SQL
Data Model Hierarchy
Conceptual Data Model
Logical Data Model
Physical Data Model
OMG's Model-Driven Architecture (MDA)
Conceptual Platform Independent Model
Logical Platform Independent Model
Platform Specific Model (PSM)

Figure 6 Mapping of data models to different technologies

Many of the historical problems associated with the development of databases stem from the fact that the data models development processes are not performed ***at the appropriate time*** in the overall database development lifecycle. Often, all three data models have been developed together during the software design phase and have only been available for review at the time of the Detailed Design Review (DDR).

The conceptual data model shall be developed during the software requirements phase and the logical data models during the architectural design phase, permitting their review at the software application SRR and PDR respectively. This recommended synchronisation between the data model lifecycle and the overall software application lifecycle is shown in Figure 7 below.

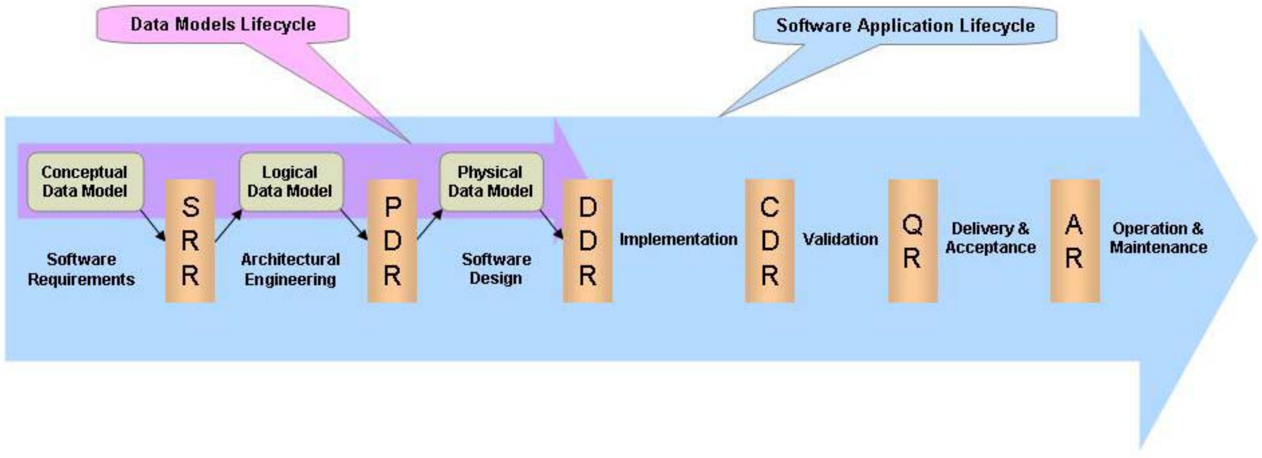


Figure 7 Data model lifecycle in relation to the software application lifecycle

4.2.5 Achieving semantic interoperability between software applications

In order to achieve semantic interoperability between (potentially) geographically dispersed software applications, the conceptual data model of all individual software applications shall be semantically compliant.

The approach promoted within this Technical Memorandum is that the conceptual data model of each individual software application shall be a sub-set of a global conceptual data model, whose scope embraces the complete space system lifecycle.

The global conceptual data model contains all the data items of significance across all space system disciplines including both “static” data items (e.g. design or definition data) and “dynamic” data items (e.g. test result data, telecommand history data for a test session or mission phase).

The lifecycle considerations of clause 4.2.4 assume the development of a new software application from scratch. In real-life however, projects often have to consider whether it is instead more effective to re-use existing software applications. Having assessed the compatibility of an existing software application with the global conceptual data model, the application could be extended by developing an interface adapter compliant with the global conceptual data model.

As a complement to the global conceptual data model, it is foreseen that there will also be reference data libraries containing predefined instances, for example:

- the names of space system decomposition levels;
- physical quantities and units;
- the requirements from ECSS standards.

4.2.6 Setting up the space system data repository for a space system project

Setting up the space system data repository for a given space system project involves the following steps:

1. Assign the role of design authority for the space system data repository.
2. Define the scope of the space system data repository for the project, for example:
    - which engineering disciplines are involved;

- which phases of the lifecycle are involved;
- which parts of the customer-supplier chain are involved.

3. Produce the project-specific global conceptual data model by tailoring the ECSS global data conceptual model and maintain it throughout the lifecycle.
4. Define the architecture of the space system data repository and maintain it throughout the lifecycle, for example:
   - which individual applications and data repositories are included;
   - what are the data exchange interfaces between these data repositories;
   - how are data repositories integrated and verified;
   - how is overall data consistency achieved;
   - how is overall data configuration and version management achieved;
   - how is data quality and availability throughout the complete space system lifecycle ensured.
5. Derive the “local” conceptual data model of each application from the project-specific global conceptual data model.
6. Implement, deploy and maintain the space system data repository infrastructure.

Steps 3, 4 and 5 of this process are illustrated in Figure 8 below.

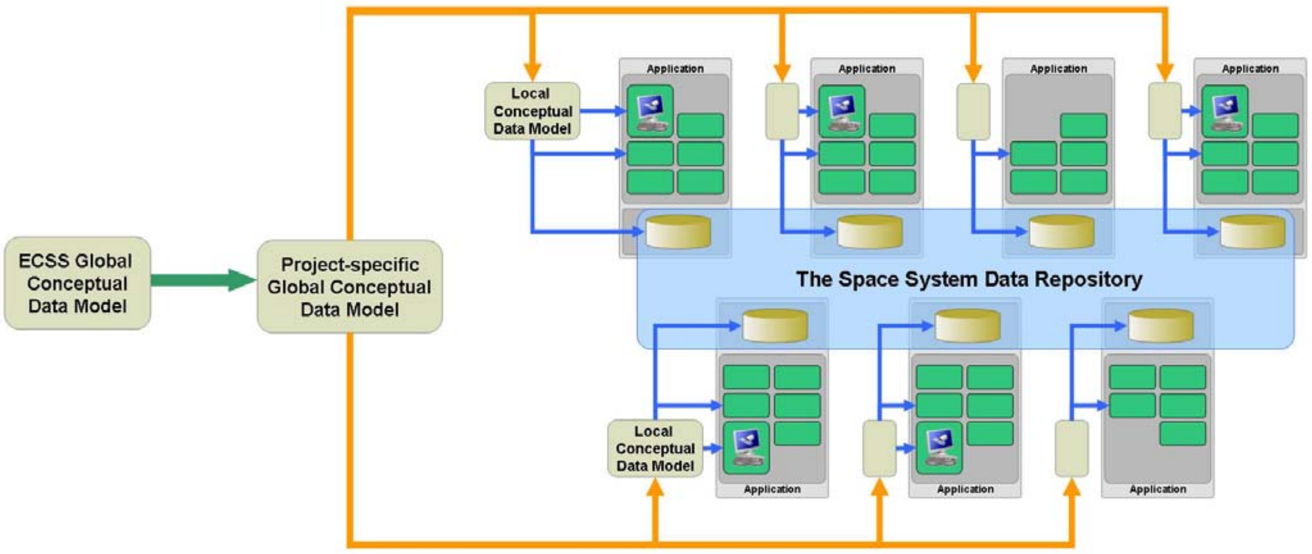

ECSS Global Conceptual Data Model
Project-specific Global Conceptual Data Model
Local Conceptual Data Model
Application
The Space System Data Repository
Local Conceptual Data Model
Application

Figure 8 Adoption of the ECSS global conceptual data model for a project

4.3 The space system global conceptual data model

4.3.1 Space system object types

In order to manage complexity, space systems are typically hierarchically decomposed into elements. Different kinds of decomposition exist such as functional breakdown and product tree.

Within the space system global conceptual model, the System Element is the primary object type. All data relating to a specific element of the space system is associated with a System Element.

The top-level System Element corresponds to the space system itself. Lower-level System Elements correspond to elements of any hierarchical decomposition.

Figure 9 below shows an example of a typical space system product tree.

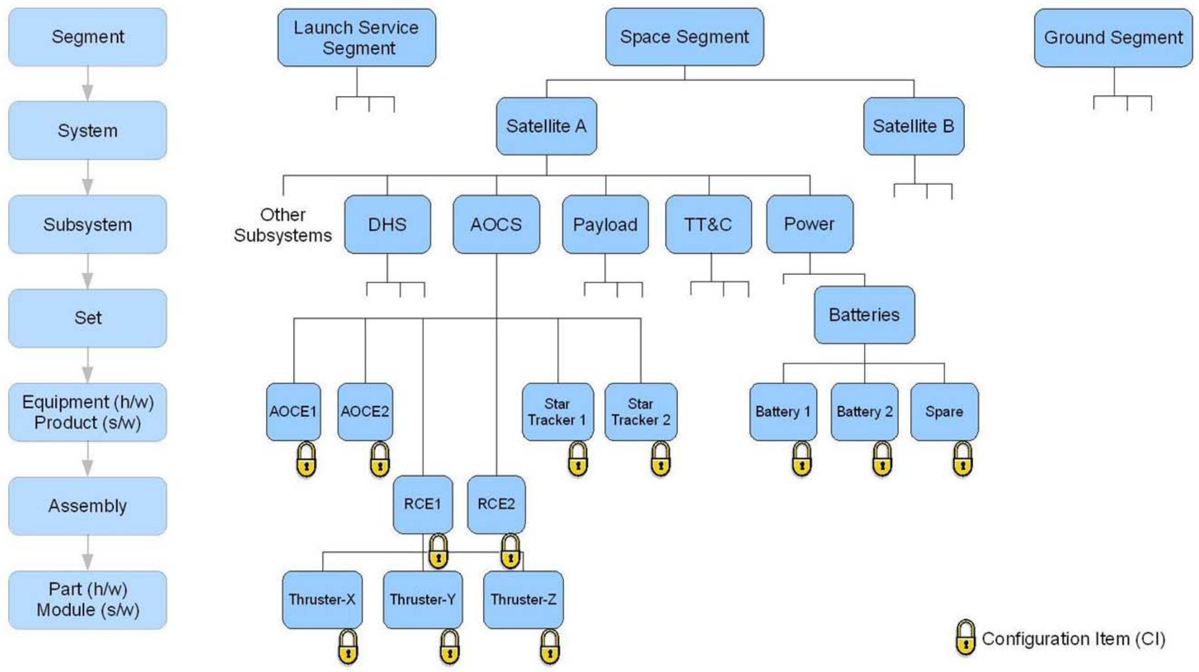


Figure 9 Typical space system product tree

System Elements may contain lower-level System Elements (see Figure 10) and other object types such as Requirements, Reporting Data (e.g. telemetry parameters), Activities (e.g. procedures, telecommands) etc. Each object type is uniquely identified by a name.

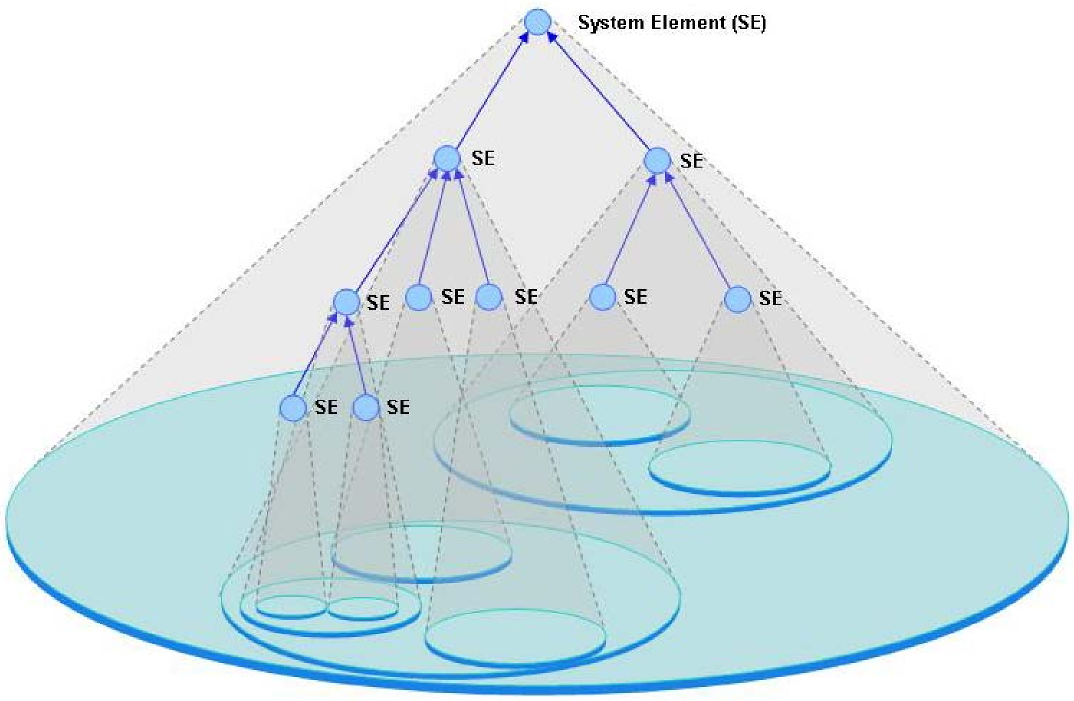


Figure 10 System Element composed of lower-level System Elements

4.3.2 Data modelling

4.3.2.1 Methodology

Conceptual data modelling is a view of the real world (i.e. human-oriented concepts) in terms of object types, their characteristics and relations between object types. Knowledge about the real world is expressed in terms of elementary facts, constraints and derivation rules. Different terminology is used for the same concepts within different modelling methodologies. Annex C contains a detailed review of current methodologies, the terms used in each methodology, and a mapping between these terms.

In the remainder of this document, the following terms are used:

- for conceptual modelling, object types consisting of a set of entity types, value types and relationships between them,
- for population, objects (object type instances) consisting of entities (entity type instances) and values (value type instances).

Some examples of the use of these concepts follow:

- The Requirement with number “5.6.3 of ECSS-E-ST-10C” has the name “Engineering Data”.
- The Document with reference “ECSS-E-ST-10C” has the name “Space engineering – System engineering general requirements”.

Within these examples there are two entity types (i.e. Requirement and Document), and four value types (i.e. number, reference, requirement name, document name).

The requirement number uniquely identifies a given requirement and the document reference uniquely identifies a given document. These value types are said to be the reference mode of the corresponding entity. A reference mode may be a combination of several value types. For example, the requirement number above could also be expressed as a couplet of value types: number “5.6.3” and document reference “ECSS-E-ST-10C”.

A conceptual model can be seen as a network of object types and relations, further refined by constraints and rules that shall or should be satisfied. Some of these relations are of “part of” nature and of special interest for determining the user views of the model, i.e. a conceptual model is not just a network of definitions but organised into hierarchical sets of definitions that represent the “user views” of the conceptual model.

To establish user views, the concepts of conceptual context, strong entity type and weak entity type are introduced. A conceptual context is associated with a given entity type within the overall network of definitions. Via the “part of” relations associated with this entity type, one can determine the overall set of definitions comprised of “lower level” entity types. Those entity types that are existence-dependent only on the entity type that defined the context are called “strong entity types” (relative to this context). All other entity types within the context are, by nature, existence-dependent on one of the strong entity types.

The following are examples of these concepts:

- System Elements have associated Activities, Reporting Data, Functions, Requirements etc. Activities have associated Arguments, each of which in turn consists of a set of values for: Argument Name, Argument Description, Argument Type, Arity, Default Value and Fixed Value (see ECSS-E-70-31C). The Activity, entity type is a strong entity type. The Argument entity type is a weak entity type consisting of six value types, see Figure 11 below. The part of the model consisting of the strong entity type Activity, the weak entity type Argument and the six associated value types corresponds to the object type called “Activity”, i.e. the object type has the same name as its corresponding strong entity type.

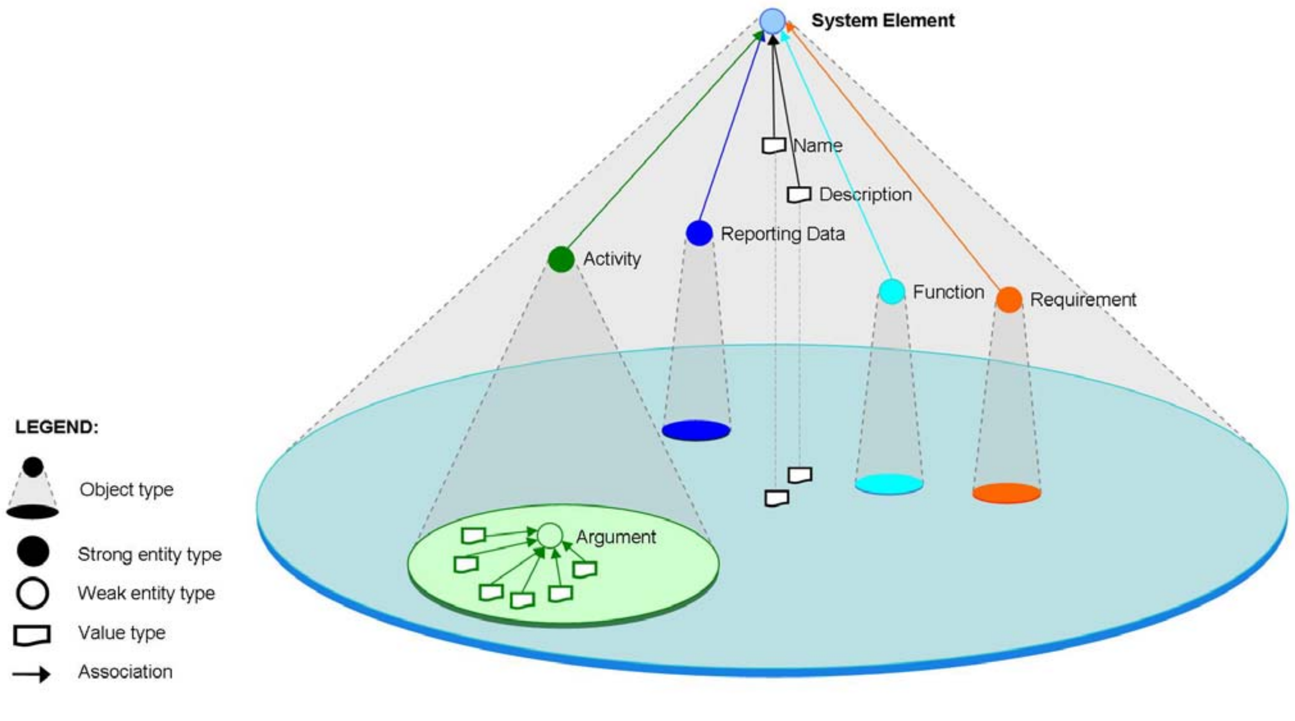


Figure 11 Example of strong entity types associated with a System Element

- The object type Requirement from Figure 11 above is shown in Figure 12 below and consists of the strong entity type Requirement with associated weak entity types (Constraints, Applicable Standards etc.) and value types (Identifier, Name, Criticality etc.).

  NOTE Refer to Annex A for a definition of the object type “Requirement”.

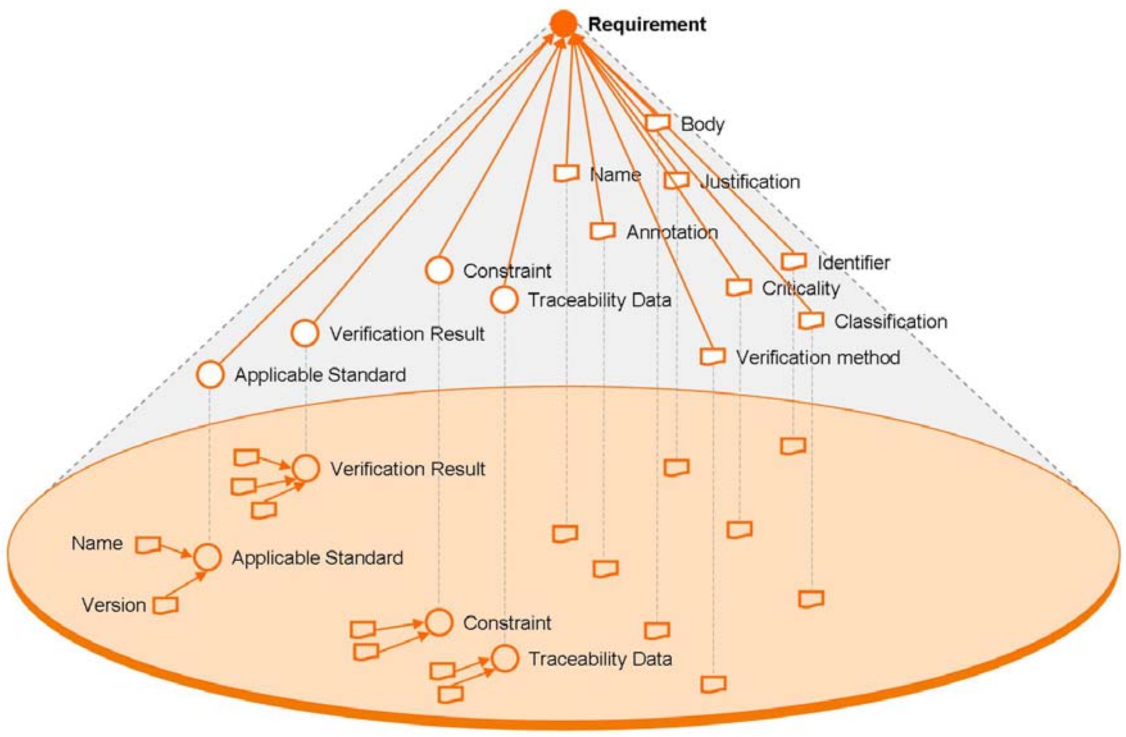


Figure 12 Example of the entity type Requirement

4.3.2.2 Naming

Within a given data repository, each object is identifiable by means of one or more names. The concept of name is introduced to refer to a unique identifier of a given object, assigned according to whether it is to be used by a human (where the objective is to easily identify the object) or by a software application (where a counter or other unique code would suffice).

Naming conventions are needed to avoid "naming collisions", i.e. ambiguous identifiers, that might occur when data is exchanged between suppliers and customers, or to ensure that names are meaningful.

To ensure the project-wide uniqueness of names, the following rules apply:

- where appropriate, a universally unique identifier (UUID) should be used, in compliance with ITU-T Rec. X.667 - ISO/IEC 9834-8.
- across the space system data repository, the uniqueness of a name must be ensured by concatenating the "relative name" (a name that is unique within a given local context, i.e. namespace) with the "reference path" i.e. when the context changes in a way that compromises naming uniqueness, the relative name is rendered unique in the new context by concatenating it with a reference path derived from the parent entity.

  NOTE An example of such a naming convention can be found in ECSS-E-ST-70-31C.

These rules imply that:

- At any level of the customer-supplier chain, each supplier is uniquely identified within the context of a customer.
- Within the context of a supplier, each data repository is uniquely identified.
- Within the context of a supplier, each electronic database delivery (i.e. data exchange) is uniquely identified.
- Within the context of a data repository or of an electronic database delivery, each System Element is uniquely identified.

- Within the context of a System Element and a given object type, each object is uniquely identified.

To ensure the availability of all data related to an object in the case where a new name is assigned, the stakeholder who assigns the new name must ensure that all information attached to the previous name of the object remains accessible. This implies either that the full information attached to the previous object name is replicated or that a link with the previous object name is retained.

4.3.2.3 Global Conceptual Data Model and its population

4.3.2.3.1 Population view

The population of the global conceptual data model evolves according with the space system life cycle. A typical view of the data model is shown in Figure 13 below. It mirrors the major lifecycle phases for a product, as identified in ECSS-M-ST-10C clause 4.4, namely:

- Analysis;
- Feasibility;
- Preliminary Definition;
- Detailed Definition;
- Qualification & Production;
- Utilisation;
- Disposal.

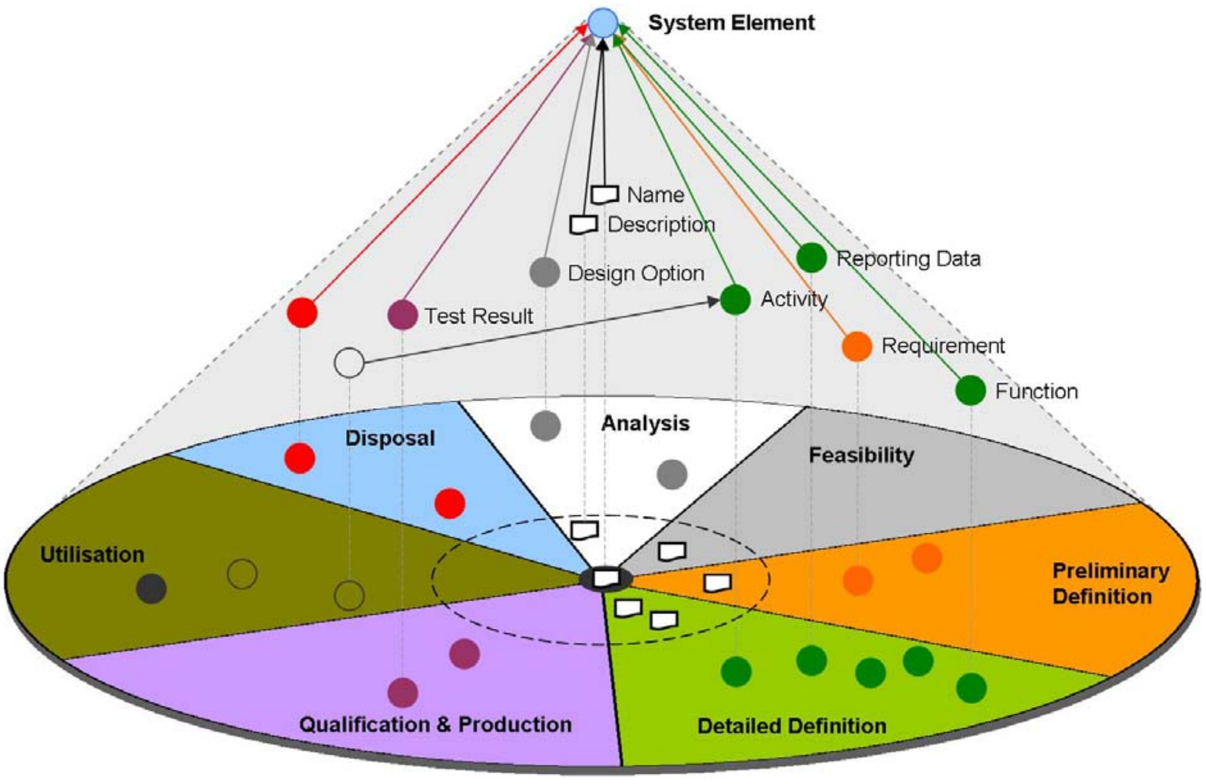


Figure 13 Data modules associated with a System Element

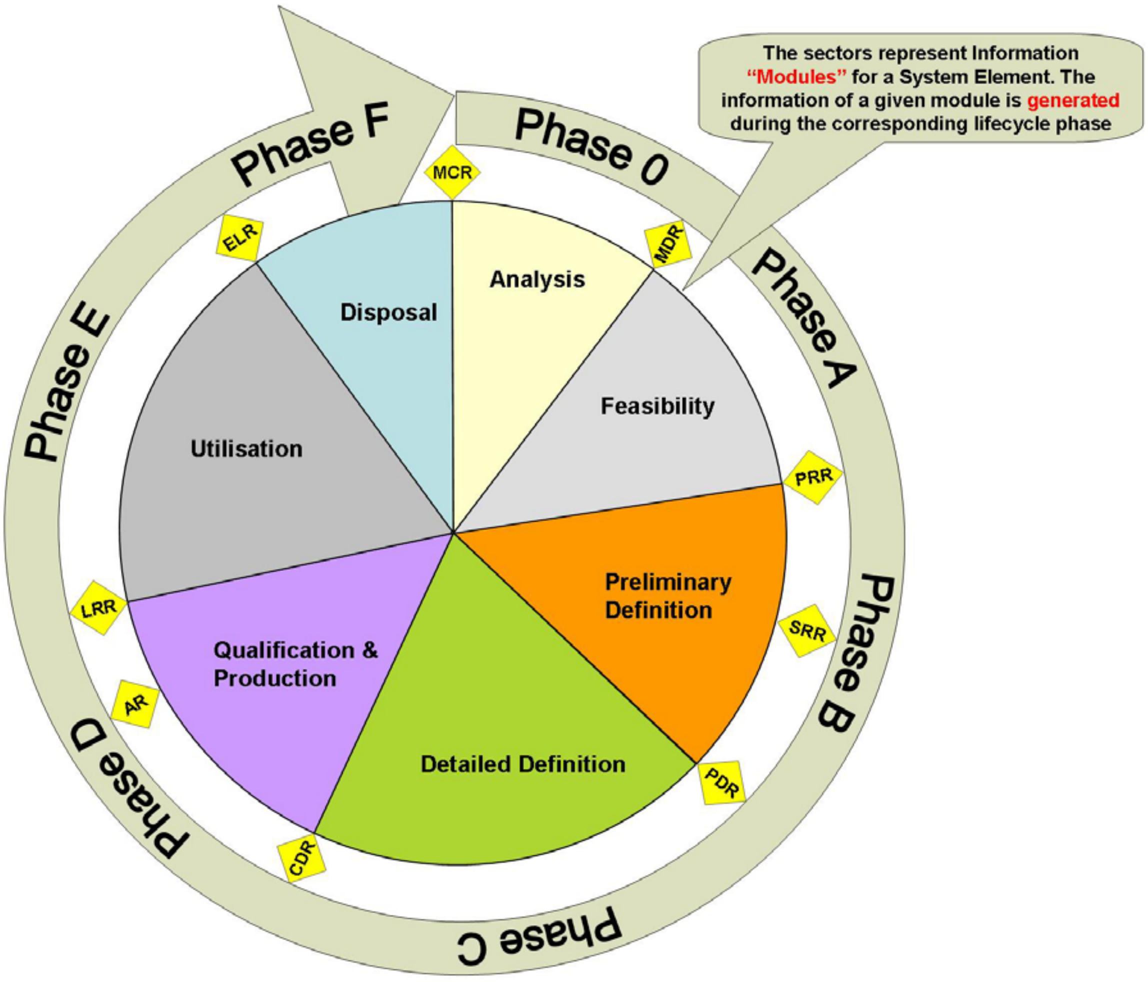
A principal advantage of this particular organisation is that data required for delivery at major reviews (MDR, PRR, PDR, CDR etc.) for a given System Element resides in the totality of the modules that precede the review in question – see Figure 14 below[3].

The data generated for a given System Element during a given phase will generally be used during that phase itself, however a subset of that data may also be used by different stakeholders during subsequent phases - and potentially different subsets during different phases. For example:

- the requirements for a System Element are cross-referenced throughout the design phase of the System Element;
- the design data for a System Element is used to both to prepare for the operation of that System Element and during its actual operation (e.g. for reference purposes in the case of an anomaly). This design/operation data is typically documented within a user manual, such as the space segment user manual (SSUM) defined in ECSS-E-ST-70C, Annex E.

Other object types may have a lifecycle that does not map exactly to their parent System Element lifecycle. For example, a reporting data object may:

- be created during the preliminary design phase of its parent System Element;
- have an associated interpretation function added during the qualification phase of its parent System Element;
- have an associated limit check added during in-flight operation.


The sectors represent Information "Modules" for a System Element. The information of a given module is generated during the corresponding lifecycle phase
Phase 0
Phase A
Phase B
Phase C
Phase D
Phase E
Phase F
MCR
MDR
PRR
SRR
PDR
CDR
AR
LRR
ELR
Analysis
Feasibility
Preliminary Definition
Detailed Definition
Qualification & Production
Utilisation
Disposal

[3] The lifecycle for some System Elements may include only a subset of the phases shown in Figure 14.

Figure 14 Relation of lifecycle phases and reviews to data modules

4.3.2.3.2 Engineering discipline views

A given stakeholder works within a context where the data of relevance to his activities is organised. From a modelling viewpoint, his "data model" consists of a view (i.e. a subset) of the global conceptual data model. This view consists of the subset of System Element data that is of relevance to the stakeholder. For example:

- a system engineer within the space system customer's project team has a system engineering view of the complete spacecraft;
- an AOCS engineer within the spacecraft prime contractor's team has a multi-disciplinary view of the AOCS subsystem.

All views include as a minimum the System Element name - see Figure 15 below.

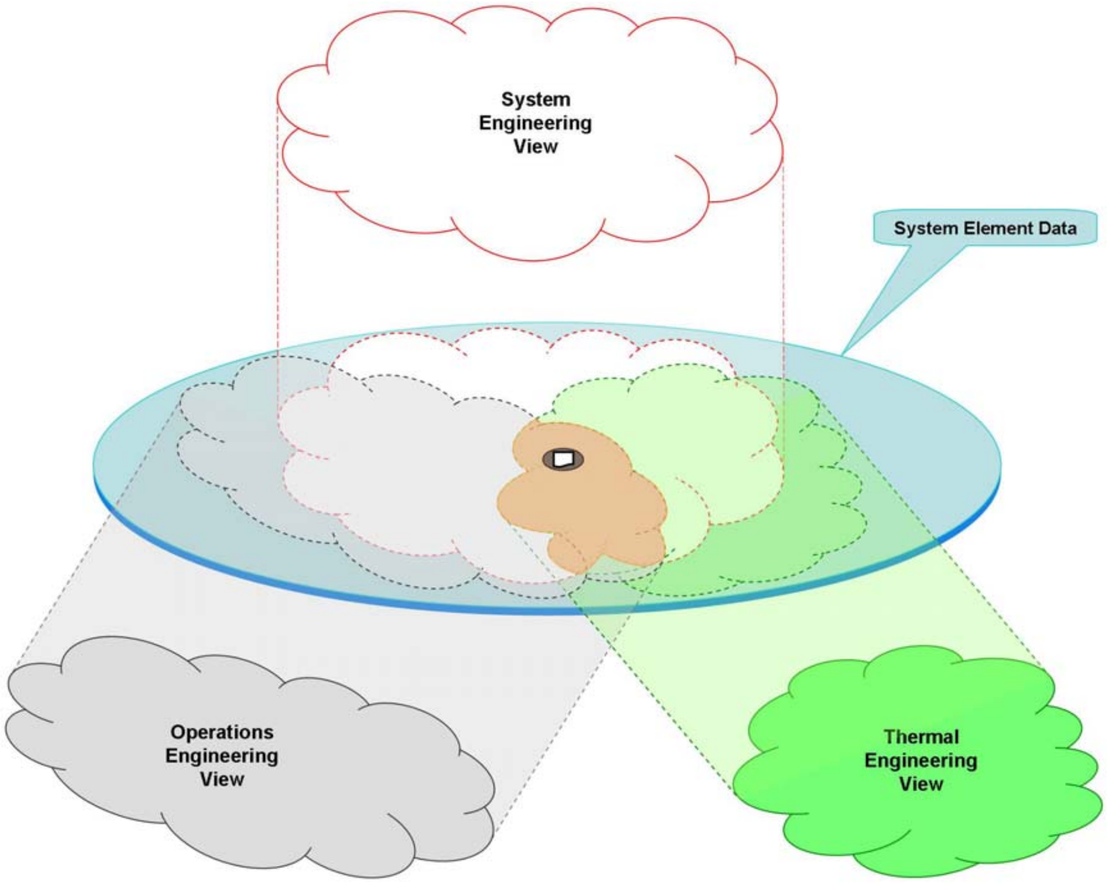


Figure 15 Illustration of different engineering discipline views of System Element data

5 Requirements

5.1 Introduction

Clause 5 contains a non-exhaustive list of requirements for the Space System Data Repository. This list will be elaborated in the future standard.

As identified in clause 4, two types of software application can coexist within the space system data repository:

- Those whose local conceptual data models comply with the global conceptual data model.
- Those whose local conceptual data models do not comply with the global conceptual data model but for which translation algorithms exist.

The following requirements should apply to each software application that participates in the space system data repository.

5.2 Requirements on individual software applications

a. The conceptual data model of the software application shall be semantically compatible with the global conceptual data model.

b. The conceptual, logical and physical data models shall be documented.

c. The mapping between the data models shall be traceable.

d. The mapping between the conceptual data model and the global conceptual data model shall be traceable.

e. Where semantic compatibility is achieved by translation, the translation algorithms shall be documented.

f. The need to import or export, or both import and export, data compliant with the global conceptual data model shall be stated explicitly.

g. If data import is required, then a software interface shall be provided for importing data compliant with the global conceptual data model.

h. If data export is required, then a software interface shall be provided for exporting data compliant with the global conceptual data model.

i. Where semantic compatibility is achieved by translation, the import or export software interface shall implement the translation algorithms.

j. The software application shall provide the capability to produce human readable documentation from its data.

EXAMPLE For the preparation of SRDs, ICDs, DDFs, DJFs, SSUMs (compliant with the corresponding ECSS DRD).

k. Automated verification and reporting of compliance of data import or export interfaces with the global conceptual data model shall be implemented.

EXAMPLE Compliance with a given constraint or rule shall be implemented.

l. Where both data import and export interfaces are required, it shall be possible to export the full content of the local database according to an ICD that complies with the global conceptual data model and to re-import the data without loss of information.

5.3 Requirements on the overall space system data repository

a. Data contained within the space system data repository shall be placed under configuration control.

b. The availability of data shall be ensured throughout the space system life cycle.

NOTE This can be achieved either by maintaining the local database system for the complete space system lifetime or by exporting the data in accordance with the requirements on the electronic database below.

c. The space system data repository shall support configuration management of all System Elements and associated objects in compliance with ECSS-E-ST-70-31C clause 6.3.

NOTE ECSS-E-ST-70-31C specifies configuration data that satisfies the ECSS-M-ST-40C configuration management rules.

d. It shall be possible to identify the differences between two versions of a given System Element and of a given object.

e. The space system data repository shall support the management of system configuration baselines.

NOTE “Configuration baseline” is defined in ECSS-M-ST-40C, clause 4.3.2.4.

f. Electronic databases exchanged between suppliers and customers shall be placed under configuration control at both ends of the exchange.

g. The quality of delivered data shall be reported, including the identification of known problems.

EXAMPLE Incompatibility with the conceptual data model, incompleteness of the delivered data or non-compliance with requirements.

h. Electronic databases exchanged between suppliers and customers shall comply with the syntax defined within an agreed interface control document (ICD).

i. Exchanged electronic databases shall include a digitally signed archive file that conforms to ECSS-M-ST-40C Annex M.

j. Data exchange ICDs shall comply with the global conceptual data model.

Annex A (informative) An example of an Object Type specification

A.1 Introduction

This annex contains an example of what is meant by specifying the data model (i.e. data requirements) associated with a given object type.

The selected object type is the “Requirement”.

Annex A.2 contains examples of the set of data requirements that constitute the “Requirement” object type definition.

These data requirements are specified using the formal object role modeling (ORM) language. The formal language notation can be found at http://www.ormfoundation.org

A.2 The Requirement data model

a. Within the context of a system element:

   1. a requirement is uniquely identified by a name.
   2. a requirement is uniquely identified by an identifier.

b. A requirement is defined by one and only statement.

c. A requirement is defined by zero or more alternative representations.

   NOTE An alternative representation could be a translation of the statement into another language such as Russian or Japanese, or a formal mathematical representation of the statement such as a MathML computer-interpretable mathematical expression of a quantitative requirement, e.g. $m_{equipment} \leq 12\text{kg}$

d. A requirement is associated with zero or more annotations.

e. A requirement is rationalised by zero or more justifications.

f. A requirement is classified by zero or more requirement categories.

g. The possible requirement categories include the following:

   1. Functional
   2. Mission

3. Interface
4. Environmental
5. Physical
6. Operational
7. Human factor
8. Logistics support
9. Product assurance
10. Configuration
11. Design
12. Verification
13. Performance
14. Implementation
15. RAMS

h. Additional requirement categories can be added at mission or at customer level.

i. A requirement is associated with at most one criticality category justified by zero or more criticality assessment reports.

j. The criticality categories associated with requirements are mission-specific.

k. A requirement is verified in accordance with one or more methods.

l. The possible verification methods for a requirement include at least the following values:

1. analysis;
2. inspection;
3. review of design;
4. test.

m. A requirement is verified according to zero or one logic.

n. The verification logic of a requirement is one of the following values:

1. open, i.e. the verification level and approach are not defined;
2. defined, i.e. the verification level and approach are defined;
3. agreed, i.e. the verification level and approach are agreed by the supplier and customer.

o. A requirement is verified by zero or more observations.

p. An observation is time tagged by a stamp.

q. An observation provides a status.

r. An observation is detailed by zero or one report (i.e. a free text).

s. The possible verification statuses are:

1. draft, i.e. the verification is made but its associated report is not verified;
2. agreed, i.e. the verification is made and the associated report has been verified by the engineering responsible;
3. approved, i.e. the verification state has been approved by the verification board;
4. verified, i.e. the verification states have been accepted by the customer.

t. A requirement is derived from at most one higher-level requirement.

u. The higher-level requirement may either be related to the same system element or to another system element.

v. A requirement is associated a level of compliance and at most one justification with the higher-level requirement.

w. The level of compliance with a higher-level requirement includes the following values:

1. compliant;
2. partially compliant;
3. not compliant.

x. A requirement is linked to zero or more requirements justified by at most one reason.

NOTE A link can be used, for example, for tracing dependence or interface.

y. A requirement has a limited applicability to zero or more specified models.

NOTE The requirements related to the models of a system element are specified in the system element data model (refer to Annex B). Such models may for example include the qualification model, the flight model.

NOTE When there is no limited applicability, the requirement is applicable to all models.

z. A requirement has a maturity represented by zero or one maturity category.

aa. A maturity category has one of the following values:

1. under drafting;
2. to be reviewed (by supplier);
3. to be validated (by customer);
4. in dispute (between customer and supplier);
5. applicable;
6. removed.

bb. A requirement has a flexibility represented by a flexibility category.

cc. A flexibility category associated with a requirement has one of the following values:

1. mandatory;
2. recommended;
3. nice to have.

Annex B (informative) UML model of the global conceptual data model

The global conceptual data model defined using UML class diagrams and additional explanations can be found from

http://www.purl.org/ecss/ecss-e-tm-10-23/annex-b

There you can find a stable version of the model, e.g. the one at the time of the publication of this document, and further development versions of the models.

Annex C (informative) Overview of data modelling methodologies

Many different methodologies exist with different concepts and notations for describing and specifying data models. Many of these methodologies are used to develop applications supporting the overall space system lifecycle.

There is currently no agreed worldwide standard for data modelling. This is a problem for the production of the global conceptual data model, since for it to become a success, one would like to express the data model using terms and notation that are readily understood by all intended readers.

Data modelling methodologies originate from two main disciplines:

- Software engineering, in particular database development, with two main streams: relational database management system development and object-oriented software engineering;
- Formal logic and ontology definition, including the recent semantic web developments, which has its roots in philosophy, artificial intelligence research and knowledge management.

These different backgrounds have resulted in many different concepts, approaches and notations. However, to a great extent these address the same or similar problems. Since it is always a considerable effort to become familiar with a particular methodology, very few people master more than a few methodologies. Naturally, most people have their personal or professional preferences and find it difficult to understand a data model expressed in a notation that differs from their preferred notation. This is a barrier to the adoption of an ECSS Technical Memorandum (and future standard) such as the present one.

In this annex, a brief description of several relevant candidate data modelling methodologies is presented, together with a correspondence matrix of concepts and terms. This information should assist the reader in understanding the global conceptual data model of the “Space System Data Repository”.

Further information about data modelling methodologies and mapping of the terminology amongst those methodologies can be found at:

http://www.purl.org/ecss/ecss-e-tm-10-23/annex-c

Bibliography

ISO 10303-1:1994	Industrial automation and integration – Product data representation and exchange – Part 1, Overview and fundamental principles (also known as "STEP" Standard for the Exchange of Product model data) http://www.tc184-sc4.org
MDA	OMG Model Driven Architecture http://www.omg.org/mda
ORM	Object-Role Modeling (ORM) methodology http://www.orm.net
UML Infrastructure	Unified Modeling Language: Infrastructure, Version 2.1.1, OMG Available Specification, formal/2007-02-04, http://www.omg.org/uml
UML Superstructure	Unified Modeling Language: Superstructure, Version 2.1.1,, OMG Available Specification, formal/2007-02-03, http://www.omg.org/uml
SysML	Systems Modeling Language, Version 1.1, OMG Available Specification, formal/2008-11-01 http://www.omgsysml.org