A Seed for a STEP Application Protocol for Systems Engineering

Erik Herzog & Anders Törne
Real Time Systems Laboratory, Department of Computer and Information Science,
Linköpings Universitet, Linköping, Sweden
e-mail: {erica, andto} @ida.liu.se

Abstract

An information model supporting core systems engineering design data is presented. The model provides, in conjunction with STEP (ISO 10303) framework services, an infrastructure that facilitates data exchange between design tools used in the systems engineering process. The model has been developed within the SEDRES project (ESPRIT 20496) and will serve as a seed for the standardisation of a furture systems engineering application protocol within the STEP framework.

1. Introduction

In the aerospace industry, as well as in other industry sectors, embedded computer systems are increasingly used to enhance and extend the functionality of previously purely mechanical or electrical systems. This development has led to more versatile, but also more complex systems, and system complexity increase with each new developed product generation. To cope with complexity and stringent requirements on availability, reliability and real-time response more emphasis is placed on the early stages of system development — i.e., systems engineering. Computer based specification, design and analysis tools have been introduced to assist systems engineers in this process.

This has introduced new problems. Computer based tools typically cover specific aspects of the systems engineering process. Moving information between tools, in cases where there is an overlap in information coverage, is difficult due to the absence of accepted tool independent data formats. The resulting situation has been characterised as one of “islands of automation” [2]. Furthermore, in co-operation projects it is often the case that partner organisations uses different tools for accomplishing the same task. Also in this case automated information exchange is obstructed by the lack of tool support for data exchange. As a result, specifications sent for further refinement or analysis in foreign tools often have to be re-entered manually into the receiving organisation’s tool-set, a tedious and error prone process. The lack of exchange mechanisms also complicate upgrades from legacy tools to modern ones. Information entered or maintained in the legacy tool will not be accessible from the modern tool unless tool interfaces are developed.

The work presented here has been supported by ESPRIT 20496 and by NUTEK under grant P10473-1

Tool integration and especially data integration methods [17], [13] have been proposed for enabling data exchanges between tools and a lot of effort have been placed into the development of tool neutral repositories such as PCTE [10] and infrastructure that facilitates for tool integration. But these effort have neither been widely accepted nor led to any significant increase in data exchange capabilities for systems engineering tools [16]. In a critical article, Brown and McDermid [3] attributes the lack of success for these methods to the misconception that open tools automatically leads to integrated tools. The view expressed is that open frameworks at best helps integration. Unless significant effort is placed on the development of common information models agreed by tool users and vendors the vision of an open environment will stay a vision.

The approach towards data integration for systems engineering tools presented in this paper focuses on identification and modelling of information relevant to systems engineering for computer intensive systems in the aerospace sector. However, we believe the model to be general enough to cover applications beyond the primary focus. A formal language, EXPRESS , has been used to formalise the information model. The model is intended to be used, in combination with the extensive implementation support offered by the STEP (ISO 10303) framework, to facilitate information exchange between systems engineering tools.

The rest of this paper is outlined as follows. Section 2 presents an overview of and the benefits of the STEP framework. Section 3 outlines the scope on the information model, as well as the process used for developing the model. Section 4 presents the model structure and the modelling principles used when developing the model. A few examples in section 5 illustrates how the model supports key concepts in the selected systems engineering process. Related work is presented in section 6 and the paper is concluded with a summary and an outline of future work in section 7.

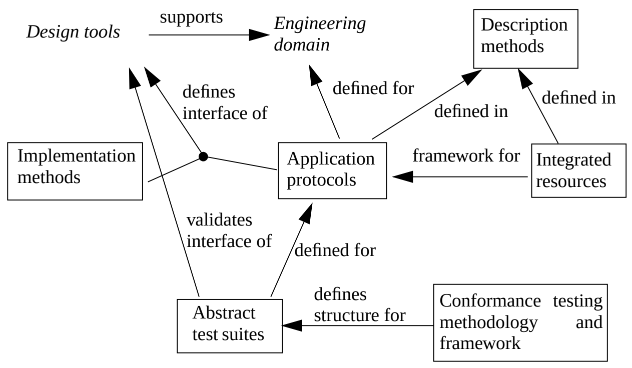


Figure 1. STEP classes and their relationships

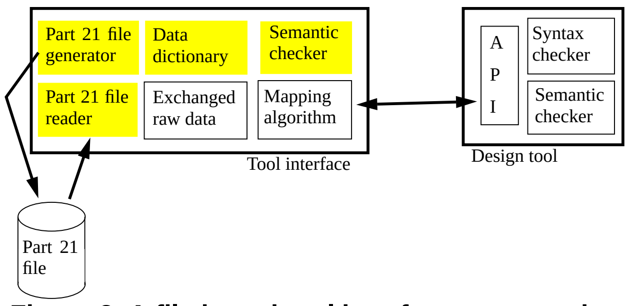


Figure 2. A file based tool interface generated from an EXPRESS data model. Shaded elements can be generated automatically from the EXPRESS model.

2. STEP

This section outlines the STEP standard framework and the benefits in using STEP components for information modelling and for achieving data exchange capabilities between tools. The overview is necessarily brief — in-depth information on STEP and its components are available in [12] and [14]. The objective of STEP is to provide the framework for the unambiguous representation and exchange of computer-interpretable product data throughout the life of a product. The framework is independent of any particular computer system. STEP is a modular standard, there is large number of parts in different levels of standardisation. The parts can be divided into six classes (see Figure 1). A brief description of each class is presented below.

An *Application Protocol* (AP) is a definition of an information and process model for an engineering domain, i.e., Configuration controlled design (AP203). The structure of an AP information model is defined in terms of a *Description method.* The standard description method used within STEP is the language EXPRESS [14]. It is based on an extended entity relationship formalism supporting object orientation concepts and specification of constraints on data and relationships. *Integrated resources* provide a set of generic definition that are shared among application protocols to facilitate interoperability. Data exchange is facilitated via *Implementation methods*. There are standard implementation methods defined for file based transfers and for specification of repository interfaces. *Conformance testing methodology and framework* define the validation process for STEP implementations and *Abstract test suites* provides a comprehensive set of test cases for an AP

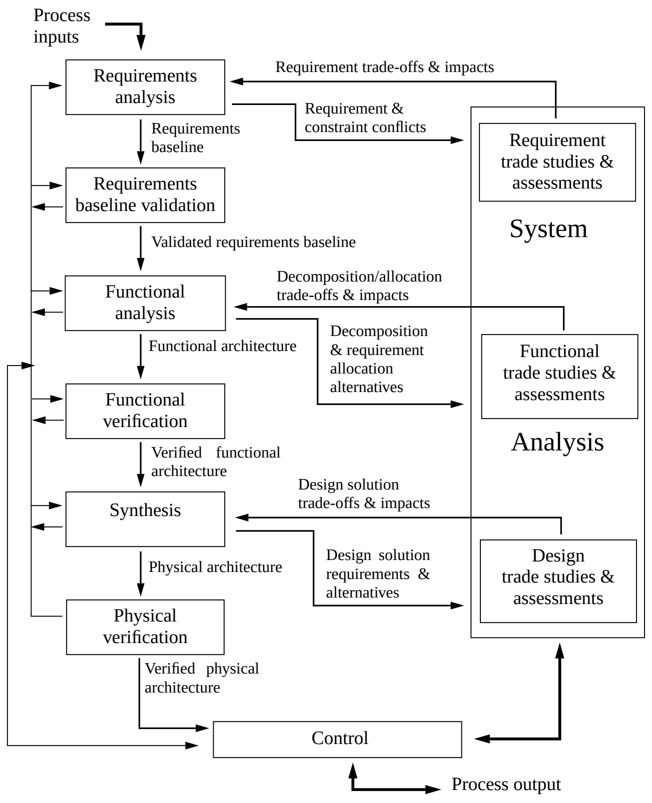
The service offered by STEP allow for realisation of domain specific data exchange structures as well as domain specific repositories with uniform interfaces. As EXPRESS is a formal language, it is possible to transform EXPRESS models to representations in languages such as C or SQL, which in turn may be used to define the structure of repositories for model data.

Compared with the traditional method for data exchange, i.e., manually written interface software for reading/writing tool specific data formats, an information model driven approach have the following advantages:

- *2n* interfaces are required to integrate *n* tools if a common information model is used compared with *n(n-1)* interfaces if no standard model is employed.
- The existence of formally defined exchange formats allows for automation of large parts of the interface development process (see Figure 2). Those parts not easily automated, i.e., the mapping from entities in the information model to the corresponding entities in a tool’s meta-model are supported by a formal description — in this case the EXPRESS model. This shall be compared with the manual process of mapping between two proprietary formats that has to be used if no common model is defined.
- An information model based approach allow users to express their view on important domain information. In this sense the approach not only facilitates information exchange across tool boundaries — it also allow users to express requirements on future tools.

3. Scope

The systems engineering information model presented in this paper is the Capability 2 model that has been developed within the SEDRES project (ESPRIT 20496). The model has been developed in co-operation with systems engineers from major European aerospace companies (Aerospatiale, Alenia, BAe, DASA and SAAB). The aim of the project has been to create an information model that can serve as a first draft in the development of a standardised application protocol for systems engineering within the STEP (ISO 10303) framework. The first steps towards this goal have been taken with the acceptance of a systems engineering as working group within ISO. The model presented here also serve as the initial draft for the standardisation work.



Process inputs
Requirements analysis
Requirement trade-offs & impacts
Requirement & constraint conflicts
Requirements baseline
Requirement trade studies & assessments
Requirements baseline validation
Validated requirements baseline
System
Functional analysis
Decomposition/allocation trade-offs & impacts
Decomposition & requirement allocation alternatives
Functional architecture
Functional trade studies & assessments
Functional verification
Analysis
Verified functional architecture
Design solution trade-offs & impacts
Synthesis
Design solution requirements & alternatives
Physical architecture
Design trade studies & assessments
Physical verification
Verified physical architecture
Control
Process output

Figure 3. The systems engineering process according to IEEE 1220

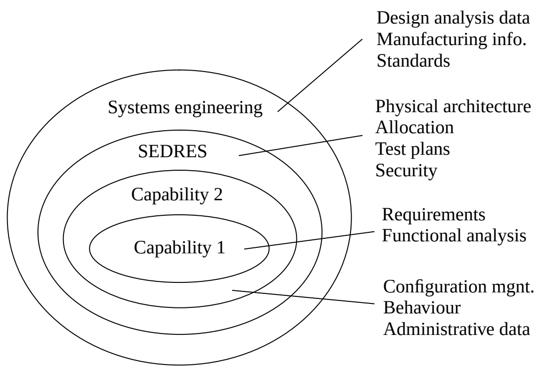


Figure 5. Information model development

The main purpose of the information model is to facilitate exchange of systems engineering information between tools, not to define 'another' development method. The focus is on supporting a harmonised set of features of tools and methods used in systems engineering today.

Systems engineering and the information relevant to the engineering process can take several shapes depending on application domain and the position of a practitioner. In this work we focus on information related to the process of design and validation of systems as described in IEEE 1220 [7](see Figure 3). The primary model scope is further restricted to design related information.

3.1 Model development

Three information models will be developed within the project. Each model is an extension of the previous. The partitioning is illustrated in Figure 5. Currently (July 1998) tool interfaces, based on the Capability 2 model, are being implemented for a number of COTS tools. The interfaces will be used in information exchanges to validate the model. These will be executed during the fall of 1998, and the results will influence the final information model that will be delivered to the project partners and the ISO working group by March 1999.

In the SEDRES project the task of Linköping University has been to communicate with systems engineering experts at the industrial partners to elicit detailed requirements on the information model, to analyse the requirements and, based on the analysis, create an information model fulfilling the requirements. Since each industrial partner has a specific view on systems engineering, influenced by development process and specification tools used, the task has been iterative in nature and has included a substantial degree of harmonisation. The objective has also been to identify and include concepts envisaged for future specification tools in the information model.

4. The SEDRES Information model

This section introduces the SEDRES Capability 2 information model [6] and presents how key requirements are met by constructs in the model. Although it is only supporting a subset of the domain it is quite a large model with over 130 entities. Due to the size of the model only excerpts are presented in this paper.

4.1 Model overview

A systems engineering information model calls for support for a wide set of information. To make the model more manageable it has been divided into a number of units of functionality (UoF). The intention is that an interface developer can find information related to a particular tool capability in one UoF. For the Capability 2 model the following six UoFs were identified.

- Configuration management — contains entities that define the model framework as well as entities supporting version management, object grouping and administrative information.
- Requirements, entities for representing, e.g., requirements, requirement analysis and systems related information.

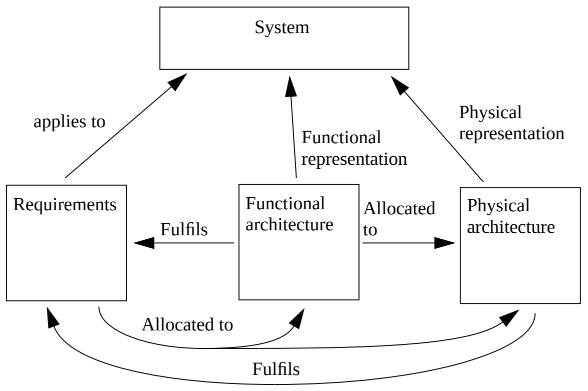


Figure 6. Model conceptual view.

- Functional design, entities for representing, e.g., functions, function hierarchies and data types.
- Behaviour, entities for representing, e.g., finite state machines and their interaction with functions.
- Physical architecture, entities for abstract specification of physical objects.
- Graphics, basic entities for representing basic shapes and object positions on design diagrams, such as data flow and state transition diagrams.

Coupling between, and within, the UoFs is intentionally kept weak to improve flexibility. This allows for transparent representations of partial as well as complete system specifications. Instantiation combinations that are not meaningful are prohibited by EXPRESS rules.

Reuse. The structure of the information model is tailored for reuse of design elements. Two entity types are used for representing elements such as requirements and functions. A ‘definition’ entity provides the static definition of an object, while an ‘instance’ entity provide information on how a definition is used in a particular system and how it relates to other objects in the system. Version management is supported for all ‘definition’ type objects.

The structure is motivated by the desire to be able to reuse validated elements of existing specifications and on management of multiple objects with identical definitions — only the shared definition needs to be modified to update all instances. In contemporary file based tools the benefit of this construct is limited to elements in the same design. Future design repositories and tools are expected to increase the benefits of the construct.

Graphics. The definition of object semantics is separated from object presentation information (graphics). Layout information is considered optional in the information model, but is supported so that tools receiving data can recreate an approximate layout of an original design diagram. No implementation should rely on the existence of graphical information in exchanged data.

Behaviour. Capturing behaviour related information proved to be the most challenging part of the modelling effort. The problem is due to the multitude of formalisms for representing the dynamic aspects of a specification. Four main formalisms, apart from pure textual specifications, where identified (each with a wide variety of extensions).

- Finite state machines, flat and hierarchical,
- Petri nets,
- Synchronous specifications,
- Causal chains

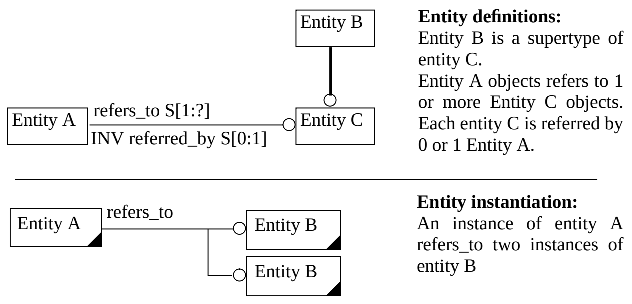
Provided that behaviour specifications are detailed and expressed in a formal language it is possible to perform automatic transformations between specifications in different formalisms. However, information may be lost in the transformation. For the Capability 2 model the inclusion of a detailed, unified model was considered. However, the size of the task, in combination with the limited amount of resources available for implementing tool interfaces for Capability 2, were factors which excluded the option.

Instead a pragmatic approach was selected. Three of the four behaviour formalisms listed above are supported in Capability 2 — finite state machines, synchronous specifications and causal chains. Semantic checks on the model are only supported to a limited degree. This approach should only be viewed as a temporary solution. Subsequent model versions are expected to support a uniform formal specifications of system behaviour.

The system and traceability. The main building blocks in the model are those for representing the system under specification, requirements, functional and physical aspects of the system. A complete system specification consists of the set of requirements posed on the system, the functional architecture, complete with behavioural specifications, that satisfies the functional requirements and the physical architecture describing the components selected for realising the system. This concept is illustrated in Figure 6. The relationships for representing allocation of objects (for instance, allocation of functional requirements to a function) and traceability aspects are important components for realising this structure.

5. Examples

A set of small examples are used to illustrate some information model constructs. The examples are all related to a fictitious system – an ‘academic’ aircraft. The structure of the example and the information provided is purely for demonstrating model constructs. The graphical notations presented in Figure 7 are used in the examples. The intention with the examples is to show the structure of the model. To preserve clarity only excerpts are shown of the model. For the same reason the full list of entity attributes have in some cases been suppressed.



Entity definitions:
Entity B is a supertype of entity C.
Entity A objects refers to 1 or more Entity C objects. Each entity C is referred by 0 or 1 Entity A.

Entity instantiation:
An instance of entity A refers_to two instances of entity B

Figure 7. The used graphical notation — EXPRESS-G data modelling (top) and model instantiation (bottom).

5.1 System views

In the information model the abstract entity **system_view** defines a view on a system under development. Two sub-types of **system_view, application** and **operational_scenario**, are provided for analysis of the system under specification. The entity **application** is used for representing the complete system. It is a collector of all information that relates to a particular system under design. As such the **application** entity can be used for representing system baselines.

For analysis purposes it may be of benefit to partition a system into the modes the system can operate in. In the information model the entity **operational_scenario** defines a mode of a system. Any number of **operational_scenarios** may be defined for an **application** object.

The model also allows for representation of dynamic relationships between working modes, e.g., under which conditions a system may change operational mode. This information is captured by the entity **scenario_relationship**. The relation between two **operational_scenario** objects defined by a **scenario_relationship** object is uni-directional, from the relating to the related operational_scenario object. If there exists a bi-directional relation between two **operational_scenario** objects, it shall be modelled using two objects of type **scenario_relationship**.

It shall be noted that the instantiation of none of these objects are forced by the model. Tools without a systems view may use the model without penalty.

Example 1: Operational scenarios

The example is shown in Figure 8. Two operational modes related to the landing phase of the academic aircraft have been identified – ‘landing’ and ‘on ground’. The transition condition from mode ‘landing’ to mode ‘on ground’ have been identified to be ‘weight on wheels = TRUE’. To preserve clarity in the example some descriptive attributes of the entities are not shown.

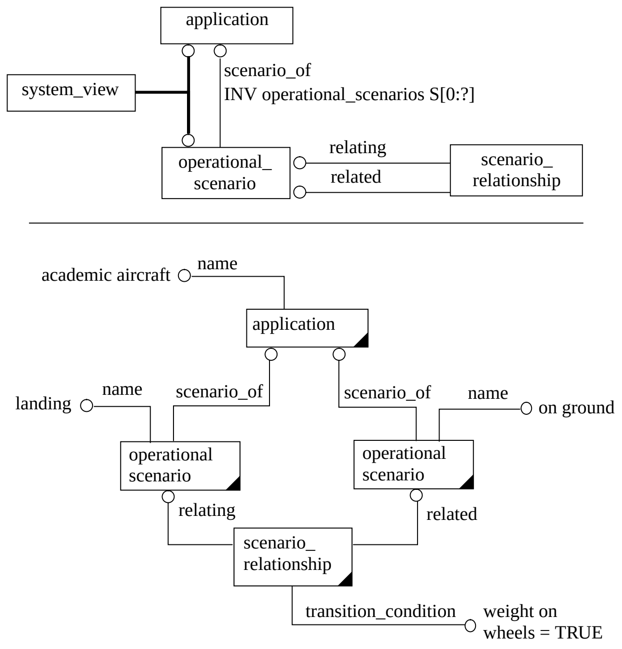


Figure 8. System structure EXPRESS-G (top) and instantiated example (bottom).

5.2 Requirements on a system

The information model supports the four classes of requirements identified in IEEE 1220 – functional, physical, operational requirements and constraints. For the Capability 2 model all requirements are assumed to be textual.

Requirements may have complex definitions and a requirement may be shared by many systems/sub-systems. For this reason the definition and the use of requirements is separated in the model. Each **requirement_instance** object has a **requirement_definition** object that provides the definition of the requirement. The existence of separate objects for representing the requirement use and definition also allows for context dependent operations on **requirement_instance** objects without having to change the reference to the definition of the requirement. For instance, if a **requirement_instance** relevant for a particular system is used to derive new requirements for the system it does not have any effect on the potentially shared definition of the requirement.

Requirement_instance objects may be assigned to the **operational_scenario** and/or **application** objects they apply to. They may also be allocated to individual **function_instance** or **physical_instance** objects to represent how a specific requirement influences the system architecture. A requirement with a complex definition may be broken down into basic requirements. The requirement analysis process is non trivial so the system specification is likely to go through several revisions before a final baseline can be formed.

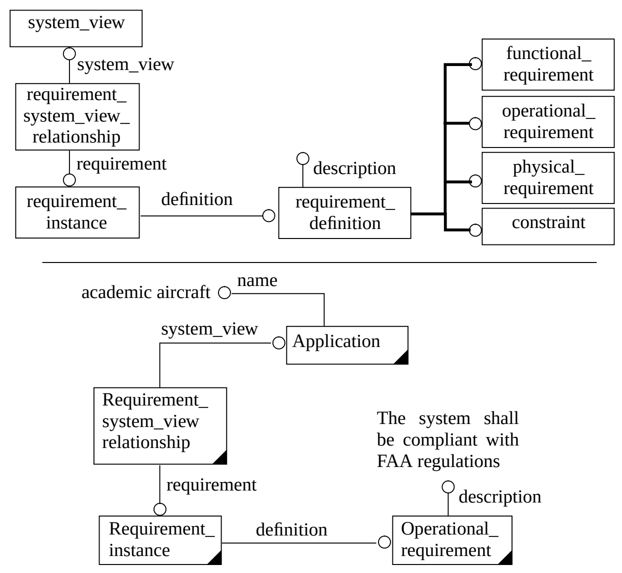


Figure 9. Assignment of requirements to a system, EXPRESS_G (top) and instantiated example (bottom).

Example 2: Requirement assignment

In this example (Figure 9) a single requirement is assigned to a system. Any number of **requirement_instance** objects can be assigned to an application, and a **requirement_definition** object can serve as a definition for several **requirement_instance** objects

5.3 System functional architecture

The information model supports the formulation of the system functional architecture. The main elements of the functional architecture are the **function_context_definition** providing a functional description of the environment the system shall operate in, and the entities for representing functions, data stores, data item, external entities etc.

Typically, a functional description of a system is realised as a hierarchy of increasingly specialised functions under the function context description. Leaf functions may have a formal mathematical definition or be described using a natural language.

A functional hierarchy may be associated with **application** and **operational_scenario** objects to define the functional architecture for a system via a **context_system_view_relationship** object. The physical architecture may be associated with the system under specification in a similar fashion. Furthermore, elements of the function hierarchy may be assigned to the physical elements that shall realise the specified functionality.

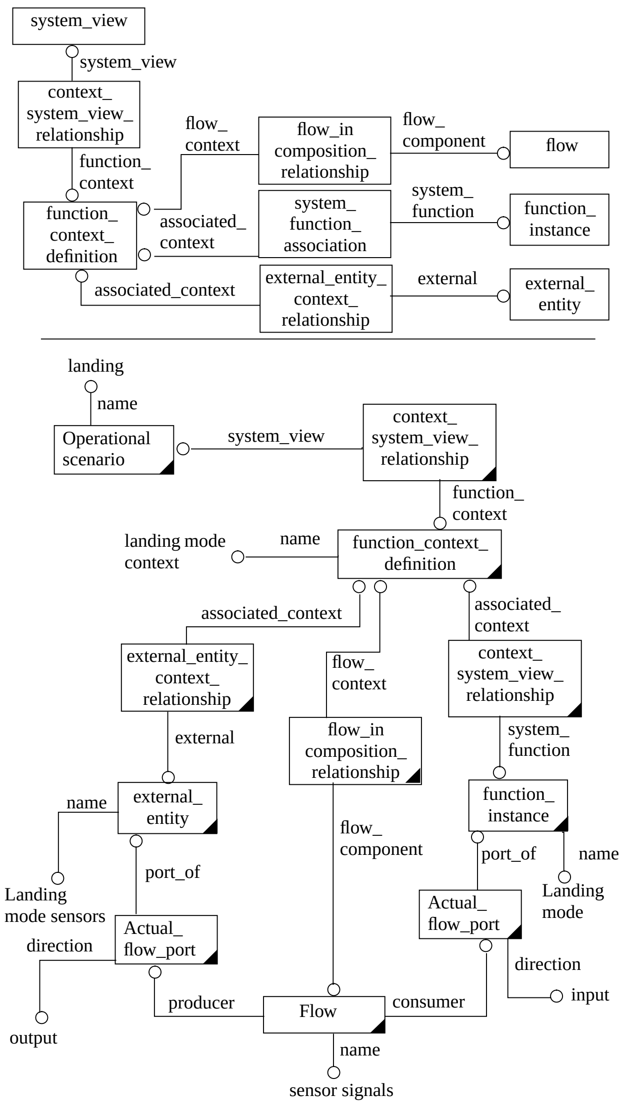


Figure 10. Functional context description for landing mode for the academic aircraft.

Example 3: Function context

In Figure 10 the assignment of a simple function context definition to a system is illustrated. In the example the functional context is defined for the operational scenario object ‘landing’ as defined in Example 1.

In the example there is a single external entity supplying input to the system, and there is a flow of information from the external entity to the system. The model is incomplete in the sense that the type of information flow is not specified and there is no **function_definition** object providing a definition for the system function. Yet the example illustrates how a functional architecture is associated with the system and how information flows between functions is realised in the model.

6. Related work

In this paper we have presented key aspects of the systems engineering information model developed within the SEDRES project. The intention with the model is that it shall serve as a medium for information exchange between design tools used in systems engineering without specifying the actual mechanisms for the exchange. There are many similar projects within the STEP framework, but mainly for different aspects of mechanical engineering, e.g., AP203 [1]. Information models related to the one presented herein have been published by Jackson [8] and Oliver [11]. The structure of the SEDRES model is in many areas similar to these models, but the SEDRES model is more detailed since it is explicitly intended for information exchange. In its current shape the SEDRES model is also less extensive than [11], but the intended final scope is similar.

Data models for software CASE data interchange have been published by CDIF [4]. CDIF have released a model for functional design [5] that is similar in scope to the Functional analysis UoF of the SEDRES model. It is our impression that the focus of CDIF is to facilitate for data exchange between similar tools rather than to support a specific engineering process.

Preliminary studies on opportunities for integrating requirement management tools has been performed within INCOSE's tool integration interest group [9], along with a study where information exchange scenarios and requirements are identified [15]. We are not aware of any work beyond this step in INCOSE.

7. Summary and Future Work

An information model supporting core systems engineering information has been presented. The information model has been provided as input for the standardisation process of STEP AP 233, a systems engineering application protocol. The information model is approaching some degree of maturity but there are several areas that need to be improved, in particular the models describing system behaviour and configuration management. The process towards a standard is long and the finalised standard is at least a few years ahead in time. It is our intention to continue extending and improving the model in the standardisation process. To increase the prospects for a successful standard we would like to encourage comments, proposals and other contributions to the work present herein.

Acknowledgements

The authors would like to thank all member of the SEDRES team for their hard and inspiring work. They have made invaluable contributions during the development of the model.

References

[1] 10303-203 *Industrial automation systems and integration - Product data representation and exchange - Part 203: Application protocol: Configuration controlled design*. ISO 10303, 1 edition, Dec. 1994.

[2] Brown, A. et al. *Principles of CASE Tool Integration*. OXFORD Press, 1994.

[3] Brown, A. and McDermid, J. Learning from ipse's mistakes. *IEEE SOFTWARE*, pages 23–28, Mar. 1992.

[4] CDIF. *CDIF CASE Data Interchange Format - Overview*. EIA/IS-106. EIA, 1994.

[5] CDIF. *CDIF Integrated Meta-model Data Flow Model Subject Area*. EIA/IS-115. EIA, 1995.

[6] Herzog, E. and Scerri, P. *Sedres draft standard /2 data model*. Technical report. May 1998.

[7] IEEE. *IEEE Trial-Use Standard for Application and Management of the System Engineering Process, IEEE STD 1220-1994*. IEEE Press, 1994.

[8] Jackson, K. An information model for computer based systems. In *1994 Tutorial and Workshop on Systems Engineering of Computer-Based Systems*, pages 32–43. IEEE Computer Society Press, 1994.

[9] Jones, D. et al. *Interfacing requirements management tools in the requirements management process - a first look*. Technical report, INCOSE, 1997.

[10] Long, F. and Morris, E. *An overview of pcte: A basis for a portable common tool environment*. Technical report, CMU/SEI, Mar. 1993.

[11] Oliver, D. A draft integration of information models: Complement model and oliver model. In *Tutorial and Workshop on Systems Engineering of Computer-Based Systems*, pages 44–69. IEEE Computer Society Press, 1994.

[12] Owen, J. *STEP - An Introduction*. Information Geometers, 1 edition, 1993.

[13] Schefström, D. and van den Broek, G., editors. *Tool Integration - Environments and Frameworks*. John Wiley and Sons, 1993.

[14] Schenk, D. and Wilson, P. *Information Modeling: The EXPRESS Way*. 1994.

[15] Schier, J. et al. *Tool integration interest group report: Scenarios leading towards a concept of operations for an integrated systems engineering environment*. Technical report, INCOSE, 1996.

[16] Sutherland, J. et al. *Functionality integration within and between software development tools*. In Baise, editor, *Proceeding of the 5th Software Quality Conference*, volume 1, pages 135–145, July 1996.

[17] Wasserman, A. I. Tool integration in software engineering environments. In Long, F., editor, *Software Engineering Environments*, LINCS 467, pages 137 – 149, Sept. 1989.