Model Based Standardization of a procedural language: Conceptualization of the ECSS-E-ST-70-32

Nieves Salor Moral[(1)], Massimiliano Mazza[(1)], Simone Dionisi[(2)]

Vitrociset Belgium, permanent establishment in the Netherlands[(1)]
'sGravendijkseweg, 53 - 2201CZ Noordwijk (NL), +31(0)71 3649770
E-mail: n.salor_moral@vitrocisetbelgium.com
m.mazza@vitrocisetbelgium.com

Vitrociset Belgium, permanent establishment in Germany[(2)]
Lise-Meitner-Strasse 10, 64293 Darmstadt (Germany), +49(0)6151 9573415
Email: s.dionisi@vitrocisetbelgium.com

ABSTRACT

One of the standards in the ECSS family (initiative from ESA since several years) is the 70-32 one, commonly known as PLUTO. It aims to specify a language for the automation of procedures, using activities as the main driver. In an effort to abstract the scope of the standard and make it as general as possible, it is full of ambiguities problems. The consequences are increased due to the use of the informative annex as the core grammar of new implementations instead of using it as a mere guide as it was its original purpose. Thus, provoking that full compliance has not ever been reached since the standard was released.

As the need to automate the creation, validation and operation of sequences of activities that must be performed (i.e. procedures) in missions still exists, the standard is still applicable. Besides this, the possibility to reuse and interoperate procedures between segments/missions/business has become a priority in the community. However, this task is still performed manually in most cases.

This paper presents a possible conceptualization of the standard into a meta-model which will answer the needs exposed and allow expanding the potential scope of business beyond the space sector and situations in which it can be applied without an important investment. However, and due to the need to be compatible with previous implementations, the new metamodel will be fully compliant with the informative textual grammar implemented in current systems, and also with different ones (e.g. graphical grammar).

INTRODUCTION

The existence of a published standard does not mean that it is useful, correct or fit for any particular use. The people who apply or specify it have the responsibility to consider the available standards, specify the correct one, enforce compliance, and use it correctly. The European Space Agency (ESA) begun a standardisation process several years ago under the name ECSS (European Cooperation for Space Standardization) trying to unify the needs of space projects in order to develop systems able to

interoperate and cross-support other while reducing risks, development and costs by establishing best practises.
One of the standards in the family is [1]., commonly known as PLUTO. It aims to define a procedural language which can be used during the development, testing and operation stages of a space mission. [1]. is notable for its emphasis on the abstraction through the use of natural language as the chosen method to define procedures and thus, not constraining the standard scope and possibilities.
Surprisingly, instead of using the offered abstraction and complying with the semantics, community is tightly bound to the specified informative (i.e. not mandatory) grammar. As any natural language grammar, it possesses intrinsic ambiguity which hinders the complete automation of procedures without imposing extra constraints to the language.
Besides discrepancies between procedure languages, users demand user friendlier, simpler and more straight-forward ways to create procedures without having to know the structure limitations and grammar constructions. As the future and its needs cannot be foreseen, but have to be taken in consideration, the need for a global conceptual model instead of a grammar takes prevalence.
This paper presents the option being performed in the ASE-5 project to unify the procedural languages specifications into a common conceptual model (i.e. conceptualization) which will support extensions while maintaining the general semantics consistency and that later can be specified by custom grammars.

CONCEPTUALIZATION APPROACH

In order to achieve full compliance with the current standard, the first required step has to be the formalisation of only the semantic and not the grammar (i.e. avoiding all the implementation details). This means understanding full compliance with the standard as compliance not with the grammar but with its meaning (i.e. semantics).
The formalisation is achieved creating a conceptual model of the language (called meta-model from now onwards) that later, can be specified with different logical models or grammars, in which the current grammar of the standard will be included. The third step would consist on their implementation through different parsers/compilers to create the executable versions.

Data Formalisation

The first formalisation to be done will take the latest version of [2] and extract the structures to be satisfied by any language used for the development of automated test and operation procedures.
From those, data concepts and relationships will be extracted and specified according to ORM methodology as facts represented with the NORMA tool. Constraints and derivations will be added at the end.
Once every requirement has been assessed, the model will be checked and validated against inconsistencies or missing information. In order to complete the model, external sources with experience in the standard will be enquired to provide feedback.
The process is equivalent to the three-level architecture in database modelling. This parallelism allows making connections between the database and procedure conceptual models.

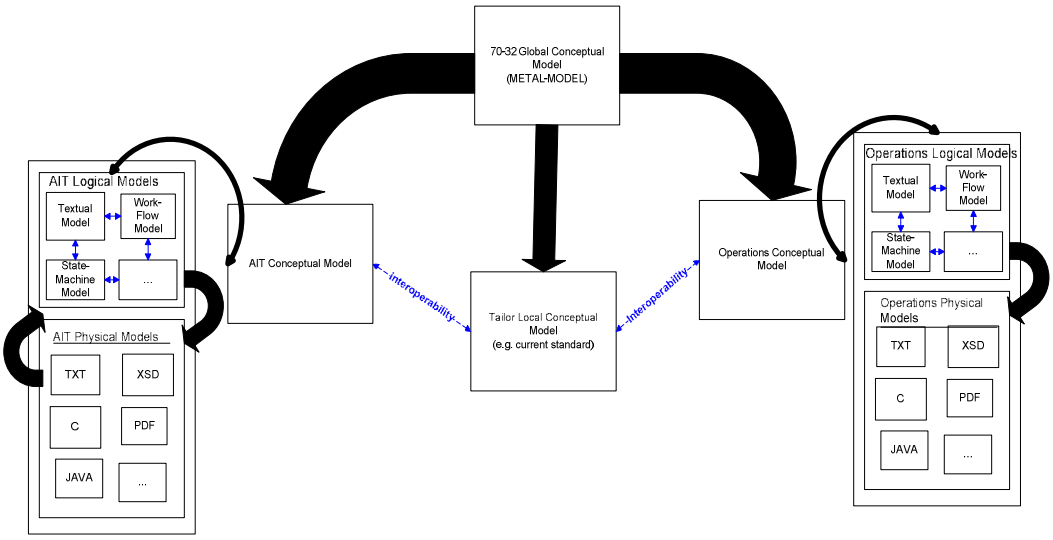

Fig.1. Three level data conceptual formalisation

The three-layer architecture of data models branching from a common conceptual head achieves two goals:

- The correct exchange of information and reusability between different tailoring (e.g. AIT, Operations, Simulations, Backwards Systems…) and/or different sessions.
- The creation of a unique extension point so only a model will be maintained instead of n partial ones.

Sharing information will be possible and correct due to the inherited semantic compatibility between local models. Sharing semantics ensures equality in the concepts and their relationships, as they must comply with the same rules and constraints.
Improvements and extensions usually have an exponential inconsistency risk when merging tailoring. Through the use of the common conceptual global model, any extension must be performed first in the meta-model, hence validating the extension. Afterwards the modification is propagated to the concrete tailoring leading to a unique model validation and a complete view of the system and its possibilities.
Once the conceptual models are tailored, logical models can be defined. Logical models are understood as the different grammars compliant with the semantics of the specific conceptual model, and thus the meta-model.

Grammar Formalisation

Based on the need to 'write', or better said 'design', activities of type procedure compliant to the standard, the physical models have to be validated and verified.
While data validation allows checking the correctness of the designed concept and business validation checks the accuracy of the tasks required for initiating the execution of the activity; the correct structures of the language (i.e. syntax of the design) must be also validated.
Syntax, usually, is validated through a grammar. By the standard definition, the grammar must allow natural language context free definitions. Grammars will be specified using Extended Backus–Naur Form *(*EBNF), railroads or logic predicates compatible with [6]. to automate the parsing process.

Business Requirement Formalisation

At high-level, [1] describes an algorithm to define procedures. This algorithm is based on a series of tasks, called steps, which must be performed at run-time.
Every algorithm is a process handling data (input, output and intermediate). Consequently, the standard describes conceptual processes where each step is another embedded process and which data is the one conceptually formalised in its totality in the meta-model.
All the required information for modelling a process (steps, data, actors, order) can be represented in a formal notation using [2]. This task will provide a common behavioural conceptual model.
Additionally, most of the current BPM tools offer not only the syntax analyser and verification of the model, but also the code automation for testing and/or operational purposes.

DATA CONCEPTUALISATION CASE

The conceptualization process is based on the data model of the procedural language. Although UML is more widely applied for this type of work, Fact-Base Modelling will be the method used for performing this task because of its clearer distinction between conceptual and logical levels.

Data Integration During Conceptualization

[3] defines the Space System Model (SSM), which contains the activities of type procedure described by [1]. Due to this intrinsic relation between the two standards shared concepts exist, and thus, they must be part of the conceptualization process. However, the definitions of these concepts must be the same, or at least compatible.
The integration may be done through addition (i.e. the term in one standard is completely described in the other) or merging (i.e. each standard complements the definition of the other). In those cases definitions cannot be merged, an inconsistency is raised and it has to be reviewed by customer in order to decide the solution.
As [3] defines the SSM, based on System Elements, Activities, Reporting Data and Events and [1] describes only the activities of type procedures; these two terms definitions have to be integrated.
[1] generally defines activities as it is shown in Fig. 2 bellow, while in [3] the definition is much more complex containing extra information like criticality, identifying redundant activities or expressing information for each types of activity. However, the two formalisations could be merged together because the hierarchy of activity types and its identification is the same, hence the integration is possible.

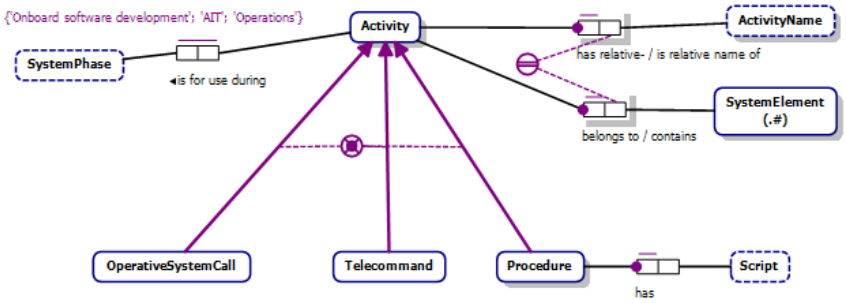

Fig.2. Activity term top-level conceptualization according to [1]

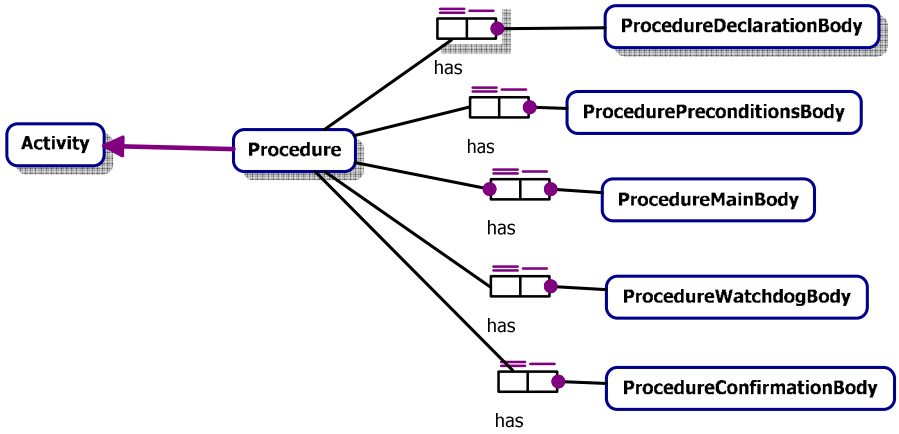

Fig.3. Procedure term top-level conceptualization according to [1]

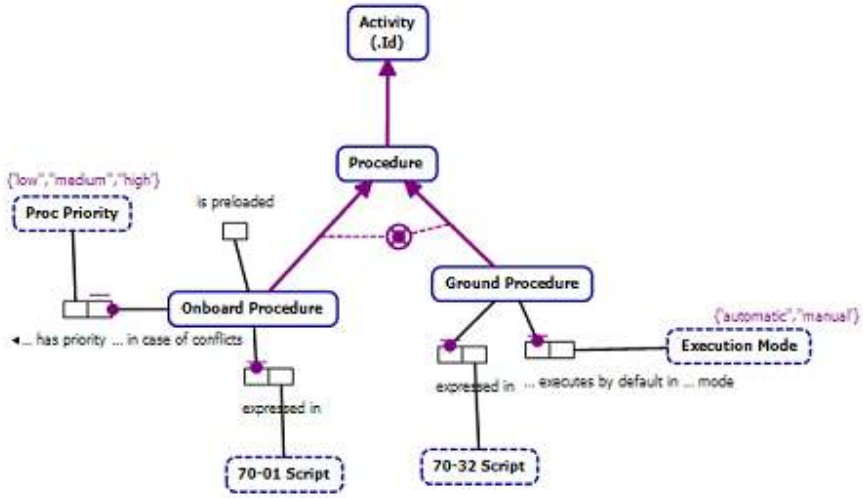

Fig.4. Procedure term top-level conceptualization according to [3]

Regarding the *Procedure* term Although Fig.3. and Fig 4. model the same term, the scopes are different because, [1] only describes a subset of the procedure hierarchy covered in [3] and hence, the definitions are different. However, the entry point for the merging appears in Fig.4. substituting the value named *70-32 script* with the model of Fig.3. and renaming the entity type as 70-32 Procedure.

Transformations

In order to deploy an efficient system, the mobility between the different levels of the formalisation must be provided. The more automation provided for the inter-level communication, the more scalable and efficient the system will be. However, not only inter-level communications must be assessed but also intra-level one. Transformations will allow exchanging instantiations between systems using different conceptual/logical/physical models compliant with the meta-model.
Transformations have to ensure data is not lost and is always possible to transform back into the original form.

From Meta-Model to Conceptual

In this exercise, meta-model is understood as the conceptual model of the whole standard together with the introduced additions. The specific conceptual model is a subset of the meta-model only with the required elements of the specific tailoring. Hereby, going from meta-model to conceptual is a straight-forward step consisting on constraining the scope of elements used in the specific conceptual model.
As the meta-model will be generated in ORM and will be used as central point of validation for every add-on to the standard, the conceptual one does not need to be validated.

From Conceptual to Logical

Logical models contain the logical data model of the procedure in a hierarchical or relational model directly derived from the conceptual one using the NORMA tool.
Besides the data model, the grammar definition will be designed and presented as logical model. Grammars will follow these rules:

- They will refer to the elements contained in the SSM uniquely.
- The grammar will be context-free.
- The grammar will not have left-recursion and will be under the LL(*) grammar.
- The grammar will be defined in EBNF.
- The grammar must be readable to the user.

Thanks to these rules, processes performed or called from the grammars will also be modelled in BPM. Traceability between data, processes and grammars will be maintained through the models; hence executable version of the grammar (i.e. the physical compiler) can invoke those processes defined from the formalised data.

From Logical to Physical

Logical data model will be instantiated into a concrete physical representation which will depend on the needs (such as performance, costs, and requirements). However, by default a relational database and the XML files physical levels will be implemented. This level will also be generated by the tool (e.g. NORMA) without having to work around the output.
Additionally as the logical model also contained the grammars compliant with the conceptual model, this part of the transformation to physical will be ensured by compilers and parsers. The syntax of the

grammar will be validated against the compilers and the data referred within that syntax will be verified using the logical data model of the SSM
Lastly, external interactions with the grammar shall also be modelled as business processes that interact with the data and with the structures that handle them. These processes will be instantiated from the logical models of BPM into portable executing processes (i.e. BPEL) called within the compiler.

Interactions between models

Current architectural trends encourage transparency and open developments (e.g. different grammars, compilers). However, the interoperability goal shall not be forgotten. Hereby, transformations between same level models (e.g. between two grammars) must also be ensured.
There are two possible approaches to the same-level interoperability issue:

- Create a 'translator' between models at the same level
- Convert the compared models to the common previous level (level either conceptual or logical) and translate them into the same level model by going down the hierarchy.

Each approach has its advantages and disadvantages, yet they are compatible with each other. These approaches have been assessed in previous works [4] referring to data models, but not with business and grammar formalisations.
The first method offers a direct approach between models that provides a translator/compiler between the two grammars. On one hand it requires only one process by avoiding travelling the architecture tree and the different conversions. In consequence the possibility of losing information decreases if the models are compatible.
On the other hand, modifying the input/output models imposes having to modify each compiler/translator. The second drawback against this approach is the combinatorial number of required compilers between different same-level models. And lastly, in case models come from different higher-level models, translations cannot be fully done because they contain different elements.
Even though the second method is more tedious, is also more secure. Firstly, it ensures compared models share the same elements before translating. With the verification results, the user shall choose the path to follow; either choosing which parts to translate or forcing the translation and accepting the possibility of having corrupted data. Besides this advantage, modifications impact is reduced because only global model compilers will have to be changed and local ones will inherit those changes.
In conclusion, these approaches can be combined depending on the use the customer will require. From an efficiency point of view, it is recommended to follow the second approach while using the first approach only for those local most used same-level compilers.

APPLICABILITY OF THE META-MODEL

[1], up to the moment, is applied in a very constrained number of situations according to VTCB's feelings because its potential scope of applicability is by far greater. Having a formal language able to create procedures in natural language that control instruments, send/receive messages, call to other activities, etc. can be reused in several different business.
These convey not only the space-related one; but also robotics (i.e. controlling robotics devices, planning), augmented reality (e.g. creating the environment depending on the situation), management (e.g. creating the instructions for taking-up a position, filling forms, making deliveries to clients, organising missions…), software engineering (e.g. defining requirements), creating templates, etc.

As the main logic behind [1] is to define everything as an activity, the examples above are only some of the possible cases where the standard can be applicable; but it offers an idea of the potential it may have, if focused correctly.
Defining and using properly the previously described meta-model, all these applications may be developed without having to invest a lot of work. They will only require the definition of the concrete activities that can be called upon by a procedure in the conceptual model and link them either with already existing commands or with newly developed ones.

CONCLUSIONS AND FUTURE WORK

The output of the formalisation process being carried out in the ASE-5 Project will be the conceptual data model and it conceptual business process model. These two models will constitute the inputs for the change requests of the current standard.
Besides the models, the logical XML Schema compliant with the data model will also be provided as a mean to validate and share procedure data. If successful in its validation and verification, the model may be integrated in the ESTEC Reference Facility.
Thanks to the conceptualization through ORM formalisation inconsistencies within and between standards are raised and clarifications of the concepts are made. Forward engineering the different knowledge sources will help during customer validation as the complete view of the Universe of Discourse (UoD) is presented in ways both software engineering and non-engineering users can understand and trace.
Extensions to the current standard are already being discussed based on the raised needs of the standard stakeholders (e.g. definition of global variables, increase of standard functions) and also due to the foreseen improvement in the preparation and execution automation they can have associated (e.g. use of aliases for elements and activities).

REFERENCES

[1] ECSS-E-70-32C - Ground systems and operations-procedure definition language; issue 2.0, July 2008

[2] BPMN 2.0- http://www.omg.org/spec/BPMN/2.0/PDF (January 2011)

[3] ECSS-E-70-31C - Ground systems and operations-Monitoring and control data Definition; issue 3.0, July 2008

[4] Robert J. L. Schmaal, Herman Balsters, Serge Valera, "Formal Specification of a Meta Hierarchical Logical Data Model Using Object Role Modeling". *OTM Workshops 2011,* pp. 370-379

[5] Aditi Barthwal and Michael Norrish, *A Formalisation of the Normal Forms of Context-Free Grammars in HOL4*, ISBN 3-642-15204-X 978-3-642-15204-7; (2010)

[6] Terence Parr, *The Definitive ANTLR Reference: Building Domain-Specific Languages*, ISBN: 978-0-9787-3925-6 (207)