Design Model Data Exchange Between Concurrent Engineering Facilities by Means of Model Transformation

Philipp M. Fischer
Volker Schaus
Andreas Gerndt

German Aerospace Center (DLR)
Lilienthalplatz 7
D-38108 Braunschweig, Germany

May 6, 2011

Abstract

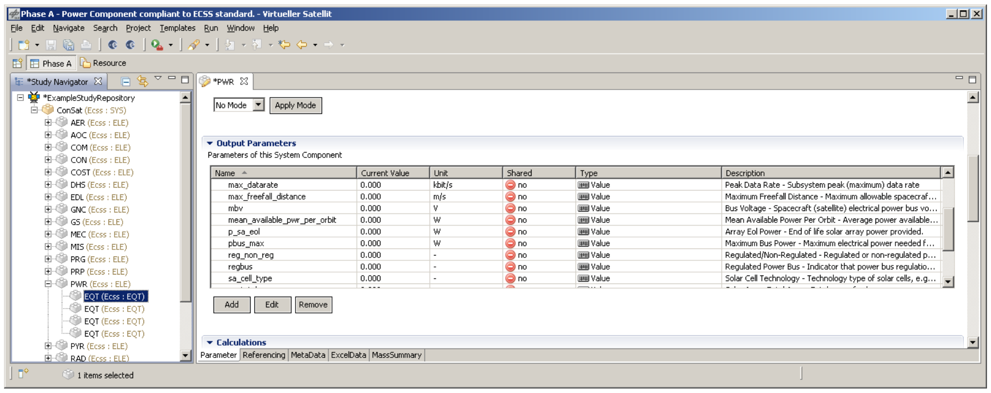
The German Aerospace Center (DLR) is developing the software Virtual Satellite for use in their Concurrent Engineering Facility. Throughout the design process of a spacecraft the software supports engineers to store and manage the data and results of their engineering sessions. The European Cooperation for Space Standardization (ECSS) is promoting a new technical memorandum for model based data exchange across the facilities. In order to enable data exchange from Virtual Satellite's own model representation to the ECSS specification, a transformation based on a triple-graph grammar has been applied. This approach allows engineers to import and export design information from the ECSS model. In addition, the grammar allows for synchronization of two instantiated models.

1 Introduction

In order to shorten time in early mission assessment studies, the European Space Agency (ESA) has established the Concurrent Design Facility (CDF). Other institutions and companies in the space environment have followed that example and built their own ones. These facilities provide room and equipment for domain experts and customers to perform studies following a defined process. In conjunction with tools and a design model the studies performed in this facility are based on quick iterations and exchange of design data to find a baseline concept for early assessments. With experts working collaboratively on the same design, changes to it are clearly visible and under attention of everyone, thus bringing consistency to the design. Besides, design time is reduced significantly. [1, 2]

With the establishment of the Concurrent Engineering Facility (CEF) [3] at the German Aerospace Center (DLR) in Bremen, the development of the software Virtual Satellite started. Target of the software is to replace the Excel based Integrated Design Model (IDM) by ESA [4]. The new software uses a data model that allows experts to define abstract elements to design a semantically correct model that can be filled with more detail during the study. The study stored in the data model is represented in a hierarchical top-down tree [1]. Figure 1 shows a screen shot of the software with a representation of the design tree in a classical navigator style on the left side. Furthermore, detailed information can be added in form of parameters using an editor shown to the right. Since the DLR is not just focusing on space missions, the design of the data model was targeting for flexibility to support different kind of studies.

Within the European space environment, concurrent design centers following the example of ESA's



Parameters of this System Component

Name	Current Value	Unit	Shared	Type	Description
max_datarate	0.000	kbit/s	no	Value	Peak Data Rate - Subsystem peak (maximum) data rate
max_freefall_distance	0.000	m/s	no	Value	Maximum Freefall Distance - Maximum allowable spacecraf...
mbv	0.000	V	no	Value	Bus Voltage - Spacecraft (satellite) electrical power bus vo...
mean_available_pwr_per_orbit	0.000	W	no	Value	Mean Available Power Per Orbit - Average power available...
p_sa_eol	0.000	W	no	Value	Array Eol Power - End of life solar array power provided.
pbus_max	0.000	W	no	Value	Maximum Bus Power - Maximum electrical power needed f...
reg_non_reg	0.000	-	no	Value	Regulated/Non-Regulated - Regulated or non-regulated p...
regbus	0.000	-	no	Value	Regulated Power Bus - Indicator that power bus regulatio...
sa_cell_type	0.000	-	no	Value	Solar Cell Technology - Technology type of solar cells, e.g...

Figure 1: Screen shot of the software Virtual Satellite

CDF are evolving. Some cases have already shown that data exchange among these facilities was necessary and will become essential in early design phases for the near future [5]. ESA is addressing this issue within the technical memorandum TM-10-25 and is promoting it towards a new standard. Part of it is the description of a data model for exchanging early phase spacecraft studies.

Due to the flexibility of the Virtual Satellite's data model, the proposed specification can easily be applied and represented. To be able to import and export data from the model described by the ECSS, various approaches from the area of model transformations have been investigated and a triple-graph grammar (TGG) is applied. This TGG allows for a selective transformation from and to both models, either in total or just on sub system level. Together with the EXPRESS based ECSS model, usage of the Standard Data Access Interface (SDAI) enables the data exchange among concurrent engineering and design facilities using STEP (ISO-10303) files.

2 Data Exchange Between ECSS and Virtual Satellite

With the proposed standard of ECSS [5], it is stated that collaborative work among the different design facilities in Europe becomes more important. As a result TM-10-25 is providing a data model to capture the data collected in a CDF-session. The data model itself is described in EXPRESS. In conjunction with the Standard Data Access Interface (SDAI) it offers an easy way to read and write STEP physical files conforming to ISO-10303-21. Even though ECSS suggests a web service for data exchange, the standardized STEP file is a good foundation for exchange of design data among the facilities.

Having the STEP file, data exchange among the various facilities becomes easier, compared to the so far used proprietary data formats like ESA IDM and DLR Virtual Satellite. Still, in order to feed the STEP file with data, the exchange from and to Virtual Satellite's data model and this STEP file is necessary. The following items will highlight requirements to achieve this goal:

1. It needs to be possible to export study data stored in Virtual Satellite into the STEP file to provide it to foreign facilities. In return data from other facilities which is already present in such a file has to remain untouched or synchronized with updates from Virtual Satellite.

2. It needs to be possible to import study data from a STEP file provided by different facilities. As well as data which is already present in the Virtual Satellite needs to remain untouched or synchronized by incoming data.

3. It needs to be possible to compare and selectively decide which parts of a study should be imported and exported.

To show how to tackle the mentioned requirements this paper will start by presenting both data models. It will be shown how the ECSS data model can be represented in the Virtual Satellite to allow for studies following this proposed standard. In preparation to exchange study data to the STEP file, model transformations are briefly discussed leading to model synchronization and triple-graph grammars. Finally, it will be presented how such a grammar has been applied and the way it offers a comfortable way for data exchange.

2.1 The ECSS Data Model

Figure 2 shows a simplified UML class diagram of the ECSS data model. The classes shown in orange color were described in TM-10-25 where as the red ones are taken from the EXPRESS representation of the model. The root element described in TM-10-25 is the *Option.* It represents a design option considered in a study [5]. The first level of decomposition to describe the study of a spacecraft is the *System.* With a link to itself it allows for system of systems modelling. Furthermore, the link to the *Element* is leading to the next level of detail. The *Element* itself can be split into *Instruments, Equipments or SubSystems.* To put further information into the study, *Equipments* can be enriched by *SubEquipments. Modes* are used to represent different states of the system or its components. They can be attached to either *System, Element or SubSystem. Parameters* collected in the study are bundled in *ParameterGroups* and can be attached to either *System, Element, Instrument, SubSystem, Equipment or SubEquipment.* All items are derived from *DataItem* (not shown in the diagram). This is an important aspect for the later implementation of the triple-graph grammar.

The ECSS data model with up to four levels of decomposition starting from *System* class is quite restrictive but gives a basic structure for design studies. It is considered complex enough for early design phases. Still it is flexible enough to manage the complexity and design work of the studied system. [5]

2.2 The Virtual Satellite Data Model

The data model of the Virtual Satellite is developed using the Eclipse Modelling Framework (EMF) [6]. Figure 3 shows a simplified UML diagram of it. The *Repository* represents the root of the data model. All data collected in a study is stored in the *Repository* The studied system and it's subsystems are represented in *SystemComponents* they are all contained in the *Repository.* To represent the hierarchy of the decomposition each *SystemComponent* manages its *Relations* to other *SystemComponents. Parameters* are stored in the *SystemComponents* and consist in general of a *Value* and *Unit. Modes* representing states of the studied system are attached to the *SystemComponents* as well. They can be referenced by *Parameters* through their optional *ModeValues.* Similar to the ECSS data model most components are derived from *ADataItem* which is important for the later implementation of the grammar. All these derived components carry *MetaData* which can be used to place additional information for requirements and constraints to the data model.

This model fulfills the requirement of not just storing spacecraft studies. With no restrictions on decomposition levels, it enables engineers to nest information with the amount of detail they need. The one to many *Relations* among the *SystemComponents* allow for building a study using a hierarchical approach. Additionally, it is also capable to store even more complex relations like cyclic dependencies or shared components. Together with the meta data attached to the *Relations* and *SystemComponents* it allows for example to represent optional sub systems or redundancies among equipments.

2.2.1 The ECSS Data Model represented by the Virtual Satellite

Looking at both models, the one from ECSS and the one from Virtual Satellite show similarities. Both have a containment for data collected in the studies. They are capable of storing the decomposition of the studied system as well as attached parameters that can be defined by the engineers. In both representations, modes can be attached to represent states of the system and subsystems.

Due to flexibility of the model used by DLR, data stored in the ECSS model can be represented by it in the following way. The information of an *Option* is stored in the *Repository.* The *System, Element, Instrument, Equipment, SubSystem* and *SubEquipment* are replaced by *SystemComponent*, whereas the corresponding ECSS level of decomposition is persisted in the *MetaData.* The *Parameters* are directly attached to the *SystemComponents* ignoring the *ParameterGroups.* The ECSS model hierarchy of the decomposition is persisted in the *Relations* where only one to many relations are used for copying the tree structure. Since the representation of modes is currently under discussion for the Virtual Satellite, no focus was placed on them during this work. Nevertheless it is possible to store the ECSS ones as well in the *SystemComponents.* Their derived subclass is persisted

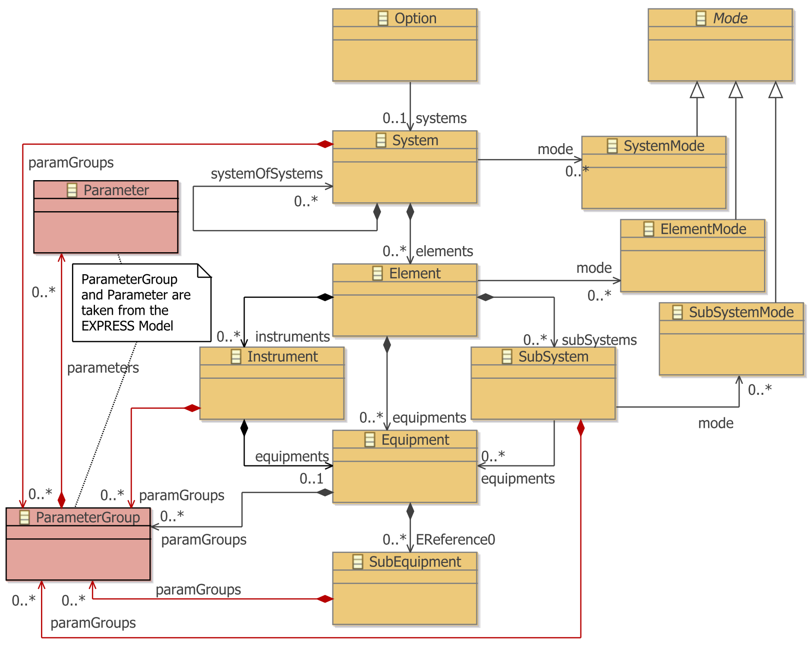


Figure 2: UML class diagram of the data model described in the TM-10-25 . It is extended by information taken of its EXPRESS description.

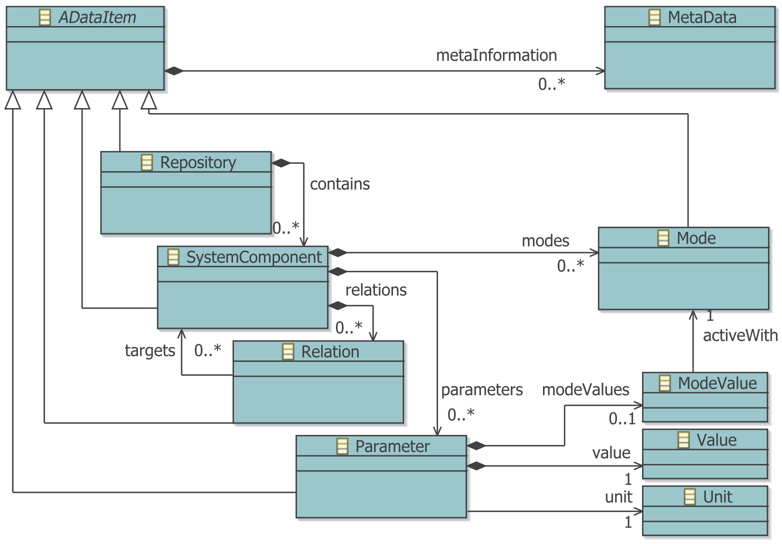


Figure 3: UML class diagram of the Virtual Satellite data model

by the *MetaData* that is attached to the *SystemComponents*.

2.3 Model Transformation by Synchronization

Looking into the area of bidirectional model transformation there are generally two different approaches. A batch oriented transformation, applying information from one model to another, or an incremental synchronization approach that is propagating modifications to the opposite side. The problem with the first batch oriented approach is that the transformation must be specified for both directions. Furthermore, it suffers in the case where the source model carries less information than the target model. Applying the transformation resets additional information in the target model when it is rebuild from scratch. [7, 8]

This problem also occurs in case of data import from the ECCS model. For example redundancy information for subsystems stored in the data model of the Virtual Satellite are not persisted in the ECSS one, hence it will get lost during such a transformation. The approach of model synchronization prevents this problem by not rebuilding the target model from scratch but instead updating it with information from the other data model [7]. Accordingly further refinements done in the Virtual Satellite can be merged with updates arriving in the ECSS format.

Investigating common approaches of model transformations [8, 9], often discussed tools are: the language *Queries, Views and Translation* (QVT), Extensible Stylesheet Language Transformation XSLT [10] as well as graph transformations. QVT is offered by the Object Modeling Group (OMG) and is capable of bidirectional transformations using the declarative language QVT Relations (QVT-R). Transformations using QVT-R are performed on Meta Object Facility (MOF) Models [11, 12]. In the case of the ECSS model this creates a problem, since it is not based on MOF, but the ISO-10303 standard part 25 offers an XMI serialisation of EXPRESS described models [13], which could be used to generate a MOF representation of the EXPRESS Model [14]. In combination with Virtual Satellite's EMF model being described in ECORE [6], which is compatible to EMOF being a subset of MOF, QVT-R seems to be a possible approach. Unfortunately two problems are arising with this approach. Firstly, the Java implementation of the SDAI (JSDAI), which is used in the Virtual Satellite, does not support the part 25 of the ISO standard [15]. Secondly, QVT-R is not explicitly discussing it, but these transformations are supposed to be bijective [16]. As mentioned earlier the ECSS model and the Virtual Satellite model may have content not being expressed in the other model, thus breaking this constraint by being non-bijective.

The second approach mentioned, using XSLT, is also not applicable. Although studies stored in both data models can be serialized into XML [6, 15], a transformation using XSLT is also not possible since it only allows for a transformation in one step [7]. This would lead to uni-directional batch transformations and not fulfil the requirement for synchronization.

Following the idea of synchronization, a triple-graph grammar (TGG) using graph rewriting rules based on the approaches described by Giese and Schürr will help [7, 17]. The rules consist of a left-hand side leading to a so called match and a right-hand side leading to a replacement. The rule is fired in case a match for the left-hand side (LHS) is found and the right-hand side (RHS) is altered accordingly [9]. The TGG based on these rules is represented in a meta model itself, a so called correspondence model. The rules of the TGG are referring with the LHS to one meta model and with the RHS to a second meta model. Being able to execute the transformation in both directions, left to right and vise versa, they are specifying a declarative definition of a bidirectional model transformation [7]. Implementing this TGG in Java supporting EMF and JSDAI, offers a simple way of connecting the two different model representations in ECORE and EXPRESS.

2.3.1 A triple-graph grammar connecting both data models

The core component of the TGG is *ACorr* which is shown in Figure 4. *ACorr* is referencing to *DataItem* and *ADataItem* of both models. Virtual Satellite is referenced on the left-hand-side while ECSS is referenced on the right-hand-side. The children containment and the reference to its parent rule are used by the transformation and parsing algorithms. From this general rule further rules for each possible match are derived. These rules store the knowledge to transform and synchronize a *Parameter* or a *SystemComponent* with an *Element* and so on. The rules displayed in the diagram are the ones for matching and transforming *Option* and *Repository, SystemComponent* and *System* as well as *Parameter* of both sides.

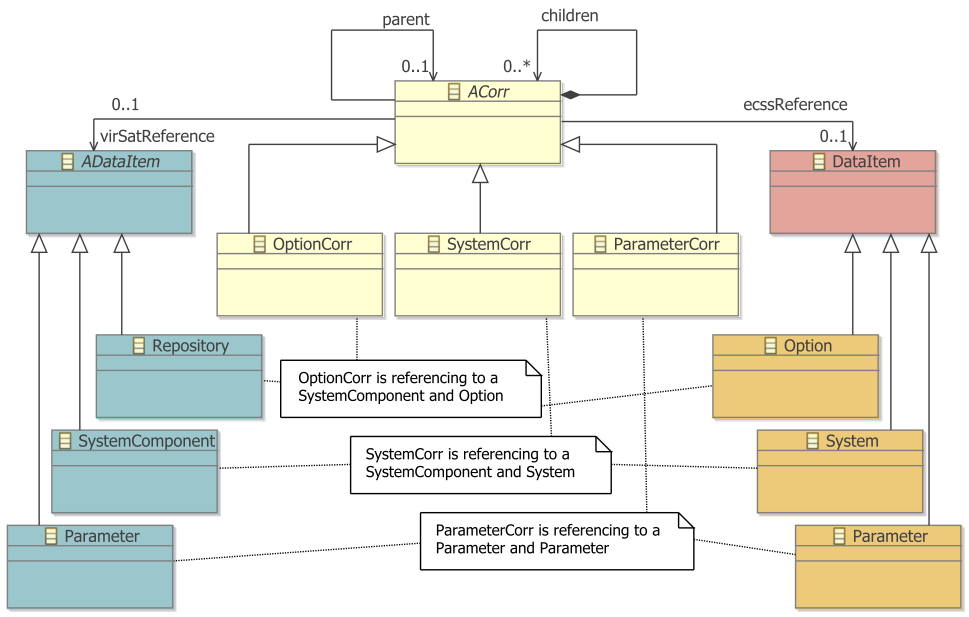

parent
children
0..1
0..*
ACorr
0..1
virSatReference
ecssReference
0..1
ADataItem
DataItem
OptionCorr
SystemCorr
ParameterCorr
Repository
OptionCorr is referencing to a SystemComponent and Option
Option
SystemComponent
SystemCorr is referencing to a SystemComponent and System
System
Parameter
ParameterCorr is referencing to a Parameter and Parameter
Parameter

Figure 4: UML class diagram of the triple-graph grammar with some of the rules needed to transform a Virtual Satellite study from and to an ECSS representation.

Further rules exist for *SystemComponent and Equipment, Instrument, Element* and *SubSystem*.

2.3.2 An Algorithm to build the correspondence model of the triple-graph grammar

In order to perform the transformation as well as the partial synchronization as stated in the requirements, two steps are needed. In the first step the correspondence model will be instantiated with the appropriate rules. This happens by parsing the source model and identifying the relevant objects in the target model. This instantiated correspondence model will allow to highlight similarities and differences across the target and source model. Within the second step, the actual transformation is executed, starting from any rule in the correspondence model. This allows for partial synchronization as stated in the requirements.

Figure 5 shows the algorithm that is parsing a model and tries to identify the corresponding object instance on the other side. Since the algorithm is the same for synchronising from and to both models, the terms LHS and RHS will be used instead of target and source. Starting from an *OptionCorr* rule, which refers to the root elements of both data models, the algorithm will look for all possible children on the LHS which will be *SystemComponents* tagged as *System* in their *MetaData*. Additionally the algorithm will look for RHS children which are all references to *System*. In the case where LHS children exist, the algorithm will try to find a match for every single child. As soon as a match is found it will be stored in a new *SystemCorr* rule and attached as child of *OptionCorr*. The match itself is based on probabilities looking to the name, the type, the description and identifier of the object. The match with the highest probability will be taken in case it exceeds a given threshold. For the case that no match is found, a new rule with no reference to the ECSS (RHS) model will be stored. The actual transformation algorithm will be able to use this information and add a new object instance to the RHS model according to the rule. This is where the parent reference comes into play, allowing the new instance to be attached to the correct parent object. After all LHS children are processed, the algorithm will proceed recursively on the newly generated child rules. On the level of the *ParameterCorr* rule, no children for the LHS and RHS side will be found since it represents our finest level of granularity for matching the models. As a result no additional rules will be added as children and the recursion returns.

Figure 6 shows an example result of the algorithm matching the Virtual Satellite data model to the ECSS data model. It highlights that the grammar displayed in the middle is following the hierarchy of the study on the left side. Besides the AOCS *SystemComponent* on *Element* level all objects of both models are matched. The *ParameterGroup* and the *Relations* are not matched directly since the information for synchronization is indirectly linked to the *SystemComponents* and *Parameters*. For example a *ParameterGroup* is created as soon as a *Parameter* is added to an empty *Equipment*.

2.3.3 The transformation using correspondence model of the triple-graph grammar

The actual transformation is performed by parsing the matched rules of the TGG, which are represented in the correspondence model. Same as the matching algorithm, the transformation can be applied LHS to RHS and vice versa. In addition the algorithm can be started from any rule within the correspondence model which allows for partial synchronization of the data models. For example it is possible to just synchronize one *Element* with its succeeding *Parameters*. To enable this functionality the algorithm starts looking for the RHS reference of the parent rule (see Figure 7). This is important since unmatched rules need to know the parent references by which a newly created object instance can be inserted correctly into the hierarchy. Accordingly the algorithm will ascend the tree of rules to the top until it reaches a parent with a RHS reference. In the worst case this is a *OptionCorr* since the *Option* and *Repository* objects always exist by convention since they are root elements of each data model. In the next step, the algorithm checks if the current rule is matched. In case it is, the information from the LHS model will be synchronized to the RHS. If the current rule is unmatched, a new RHS object instance will be created and attached. After synchronizing the data, the algorithm calls itself recursively on the child rules.

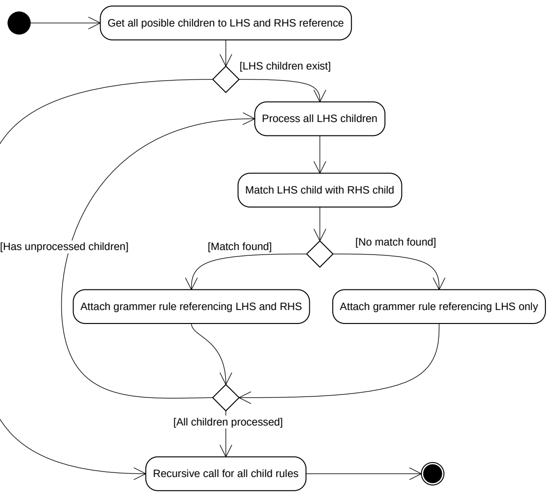

Get all posible children to LHS and RHS reference
[LHS children exist]
Process all LHS children
Match LHS child with RHS child
[Has unprocessed children]
[Match found]
[No match found]
Attach grammer rule referencing LHS and RHS
Attach grammer rule referencing LHS only
[All children processed]
Recursive call for all child rules

Figure 5: UML activity diagram of the algorithm to match the triple-graph grammar rules with the Virtual Satellite data model and the ECSS one

For example starting the algorithm from the unmatched *ElementCorr* rule in Figure 4 will perform the following steps: First it will identify the existence of the parent's RHS reference. Accordingly it will not call itself recursively to create a new parent object. In the following step the algorithm will realize that the current rule is unmatched and that a corresponding RHS object needs to be created and attached. In this case a new *Element* will be generated and attached to the parent being an instance of *System*. As a final step the content, for example the name, will be synchronized from the LHS to the RHS. The algorithm will exit since no child rules like *ParameterCorr* rules exist.

3 Summary

This paper explained the need for Virtual Satellite to be compatible with the upcoming ECSS standard. Looking to both data models, the one that is promoted by ECSS and the one from the Virtual Satellite identified a lot of similarities. Due to the flexibility of DLR's data model, the ECSS one can be represented inside, thus allowing for studies being compatible. Still the data exchange between concurrent engineering facilities required that stored study data can be transformed towards the ECSS specification. Furthermore it was necessary that data can be imported from the ECSS format as well. The DLR's data model is capable of storing more details and therefore data loss is imminent during common

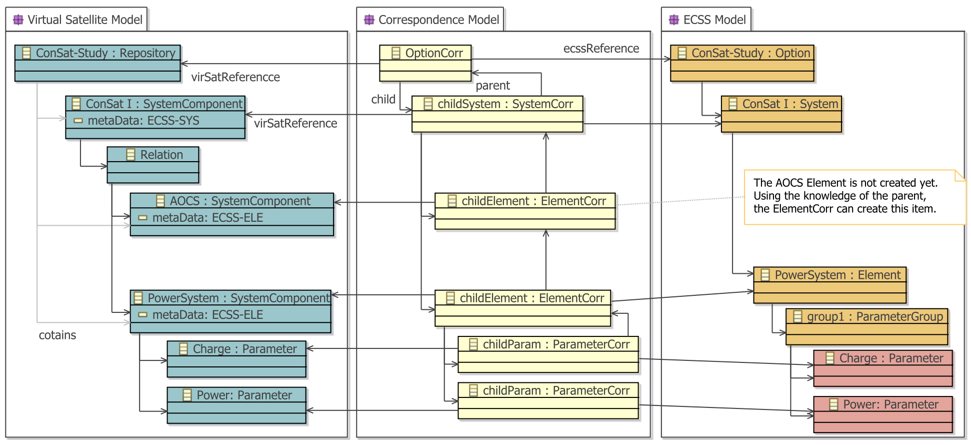


Figure 6: UML object diagram showing a matched Virtual Satellite and ECSS data model

batch-oriented transformation. To avoid this problem model transformation by synchronization was chosen and implemented using a triple-graph grammar. This grammar offered a simple way to connect Virtual Satellite's EMF model with the EXPRESS environment of ECSS. An algorithm was developed to match objects from both data models and to instantiate the correspondence model for the grammar. A second algorithm finished this work by successfully parsing the previously generated rules to synchronize both models. This synchronization of the models finally allows to export the study data into a STEP file for exchange with other CDFs.

For the future it is important to follow the development of TM-10-25 and to update this transformation accordingly. Even though the displayed transformation is a reasonable good starting point for the future. It is only focusing on subsets of the two data models. Increasing this scope in future developments might force to rethink some of the approaches shown in this paper. In particular the Java implementation of the TGG rules was time consuming and a model based approach like using a domain specific language comparable to QVT would be desirable.

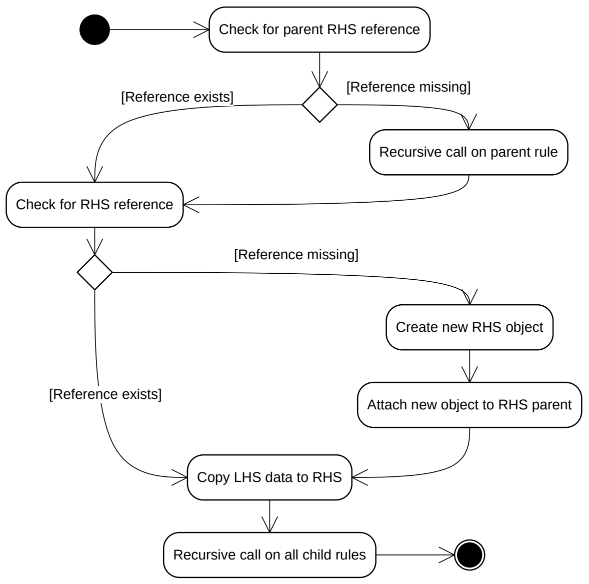


Figure 7: UML activity diagram of the algorithm to transform the models parsing the previously matched grammar rules

References

[1] V. Schaus, P. M. Fischer, D. Lüdtke, A. Braukhane, O. Romberg, and A. Gerndt, "Concurrent engineering software development at german aerospace center - status and outlook," in *Proceedings of the 4th International Workshop on System and Concurrent Engineering for Space Applications*, European Space Agency (ESA), 2010.

[2] M. Bandecchi, B. Melton, B. Gardini, and

F. Ongaro, "The ESA/ESTEC concurrent design facility," in *Proceedings of European Systems Engineering Conference EuSEC*, 2000.

[3] H. Schumann, A. Braukhane, J. T. Grundmann, R. Hempel, B. Kazeminejad, O. Romberg, and M. Sippel, "Overview of the new concurrent engineering facility at DLR," in *Proceedings of the 3rd International Workshop on System and Concurrent Engineering*, 2008.

[4] H. Schumann, H. Wendel, A. Braukhane, A. Berres, A. Gerndt, and A. Schreiber, "Concurrent systems engineering in aerospace: From excel-based to model driven design," in *Proceedings of the 8th Conference on Systems Engineering Research*, 2010.

[5] European Cooperation for Space Standadization, *Space engineering - Engineering design model data exchange (CDF)*, October 2010.

[6] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF Eclipse Modeling Framework*. Addison-Wesley, 2. ed., 2009.

[7] H. Giese and R. Wagner, "From model transformation to incremental bidirectional model synchronization," in *Software and Systems Modelling*, vol. 8, pp. 21–43, Springer-Verlag, 2009.

[8] P. Stevens, "A landscape of bidirectional model transformations," in *Post-Proceedings of Generative and Transformational Techniques in Software Engineering II*, 2007.

[9] S. Sendall and W. Kozaczynski, "Model transformation: The heart and soul of model-driven software development," in *IEEE Software*, vol. 20, pp. 42–45, IEEE Computer Society, September/October 2003.

[10] World Wide Web Consortium, http://www.w3.org/TR/xslt20/ (accessed 13-04-2011), *XSL Transformations (XSLT) Version 2.0*, January 2007.

[11] Object Management Group, http://www.omg.org/spec/MOF/2.0 (accessed 08-04-2011), *MOF Core Specification - Version 2.0*, January 2006.

[12] Object Management Group, http://www.omg.org/spec/QVT/1.1 (accessed 08-04-2011), *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification - Version 1.1*, January 2011.

[13] D. Price, "An introduction to ISO STEP part 25," (Presentation at http://www.exff.org/exff_legacy/docs/p25intro.pps (accessed 06-04-2011)), Eurostep Limited.

[14] Object Management Group, http://www.omg.org/spec/XMI/2.1.1/ (accessed 08-04-2011), *MOF 2.0/XMI Mapping - Version 2.1.1*, December 2007.

[15] LKSoftware GmbH, Webpage at http://www.jsdai.net/overview/features (accessed 02-04-2011), *JSDAI features*, April 2011.

[16] P. Stevens, "Bidirectional model transformations in QVT: semantic issues and open questions," in *Proceedings of the 10th International Conference on Model Driven Engineering Languages and Systems 2007*, 2007.

[17] A. Schürr, "Specification of graph translators with triple graph grammars," in *Graph-Theoretic Concepts in Computer Science*, vol. 903 of *Lecture Notes in Computer Science*, pp. 151–163, Springer Berlin / Heidelberg, 1995.