A UML Profile to Couple the Production Code Generator TargetLink with UML Design Tools

Malte Falk[1], Stefan Walter[2], and Achim Rettberg[1,3(✉)]

[1] Carl von Ossietzky University Oldenburg, Oldenburg, Germany
malte-falk@gmx.de
[2] dSPACE GmbH, Paderborn, Germany
swalter@dspace.de
[3] Hella Electronics, Lippstadt , Germany
achim.rettberg@iess.org

Abstract. When modelling architecture of complex embedded systems proper architecture languages and tools are necessary. UML [1] has become a proven and well accepted design language to express system as well as software architecture. For definition of internal behavior of components composed and specified during software design, Simulink is commonly used, especially for automotive and aerospace applications. As common practise, code is generated directly from such behavior models. Therefore, code generators such as TargetLink [2] are used. In this paper we propose a UML profile to describe specific properties necessary to adapt UML models to the code generator TargetLink.

1 Motivation

To cope with today's growing complexity of requirements in embedded systems development adequate development methods are crucial to develop high quality systems. Today's embedded systems often consists of distributed functionalities where software components could run on the same or on a different hardware platform within the distributed system. To define such complex system structures an architectural design phase in the development process is essential. As a de-facto standard language to be used in software and system architecture modelling the Unified Modelling Language (UML) [1] and the System Modelling Language (SysML) [3] is widely used in different fields. At a certain step during development of embedded systems, one have to model and implement the behavior of the defined software components. A common tool for behaviour modelling is Simulink from The Mathworks [4]. Especially in the automotive industry but also for development of airborne software both modeling environments UML and Simulink are used extensively. After modelling the behavior of a software component, in best case the embedded code to be integrated on the electronic control unit is directly generated from the behavior model, in this paper from Simulink.

© IFIP International Federation for Information Processing 2017
Published by Springer International Publishing AG 2017. All Rights Reserved
M. Götz et al. (Eds.): IESS 2015, IFIP AICT 523, pp. 185–196, 2017.
https://doi.org/10.1007/978-3-319-90023-0_15

Generating code from Simulink software specifications which is then directly used on electronic control units is de-facto standard in state of the art development processes especially in the automotive industry. The technique of autocode generation is well accepted and widely used especially in safety critical projects. One of the major tools used by the industry is TargetLink from dSPACE GmbH [2]. In this paper we propose a mapping between UML component architecture models and TargetLink models.

2 Related Work

Various research work deal with mapping of MATLAB/Simulink models to UML like [5]. In contrast to this paper, [5] discusses a possibility to illustrate the MATLAB/Simulink models and the behaviour of it in UML and replaces Simulink hereby.

Other publications discuss a possibility to describe systems with UML and redescribe it in MATLAB/Simulink. In [6] it is discussed how software architectures which are described in UML could be transformed to simulation models which are described with MATLAB/Simulink. These publications describe a similar approach compared to this paper. In contrast to other publications this paper describes a possibility to map a TargetLink model, which is based on MATLAB/Simulink, to UML. The benefit is to reuse properties such as interface description, etc. from an architecture model and annotate it to a TargetLink model, which is later used for automatic code generation.

3 TargetLink

TargetLink is a Toolbox for MATLAB/Simulink in order to generate series production code by a push of a button from a Simulink model. Beside generation of regular ANSI-C code for fixed point or floating point processors it is also possible to generate code for certain processor/compiler combinations. To enhance a Simulink model to a TargetLink model used for code generation, certain parameters need to be set, to describe how the code generated by TargetLink looks like. To maintain all parameters for all blocks within the TargetLink model the TargetLink Data Dictionary is used. The TargetLink Data Dictionary represents a data container in order to describe all elements. These elements represent information for the model design, the code generation and the implementation of a model on an electronic control unit. TargetLink elements are described by various properties and can be referenced by TargetLink models. In the following section we will introduce and discuss the relevant TargetLink elements used for UML mapping.

3.1 TargetLink Data Dictionary Elements

The TargetLink Data Dictionary is used to parameterize and describe all elements used in TargetLink models. Below all elements which are relevant to be mapped to UML are specified. A more in-depth and formal description of the various element properties can be found in [7,8].

Pool. The TargetLink Data Dictionary is divided into three areas *Config*, *Pool*, and *Subsystem*. In this paper only the *Pool*-area is considered. This area contains all data elements which are required for code generation. Examples of elements in the Pool area are scaling values of fixpoint variables the or type definition of variables.

Blocks. Blocks are the main elements in TargetLink Data Dictionary and therefore essential for the code generation. Based on the Block description, Simulink Blocks could be generated directly from the TargetLink Data Dictionary. Even though various types of Blocks are available inside the TargetLink Data Dictionary, this paper focusses only on Blocks of type *TL_Function*. Those Blocks represent the base architecture of the software component to be developed.

Signature and SignaturePort. A Signature contains Signature-Ports, whereas Signature-Ports are divided in *in*-, *out*-, and *user*-ports. Signatures are used to describe which in- and out-ports are available at the TL_Function Block. In contrast to in- and out-ports, user-ports are freely configurable for different use cases, e.g. calibration-ports.With ports it is possible to connect TL_Function Blocks with Variables, either incoming or outgoing.

Variable. Variables are used to exchange information between different TL_Function Blocks. The properties of variables define the appearance within the generated code. The actual type of Variable is defined by a *Typedef* element as described in the following paragraphs. A min, max range can be set by a reference to a *Scaling* element as described in the following paragraphs.

Typedef. The Typedef element specifies the datatype of a Variable element. In the TargetLink Data Dictionary Typedefs are referenced by Variables.

Scaling. A Scaling is used to define the value range of a Variable. This is necessary in case of fixed-point code generation. Furthermore, it is possible to define values for Least Significant Bit (LSB), Offset, and, physical unit within the Scaling properties.

Module. The characteristics of the code modules in the generated code are specified by Module elements, e.g. the memory location. Properties of these elements are TargetLink specific and do not have any implication on architecture level. Due to this fact it is not considered in scope of this paper.

FunctionClass. FunctionClass properties define how the generated code for a referenced TL_Function Block would look like. Because FunctionClasses are usually predefined, they are not considered in the further course of this paper.

VariableClass. VariableClasses are similar elements compared to FunctionClasses. The difference is that VariableClasses specify Variables in the generated code. Because VariableClasses are usually predefined, they are not considered in the further course of this paper.

TargetLink Data Dictionary Model. In the context of this paper a TargetLink Data Dictionary model is a finite set of all above described elements. In Sect. 4 of this paper a mapping of TargetLink Data Dictionary model to UML is described.

3.2 Relationship Between TargetLink Elements

This section describes the relationship between the TargetLink Data Dictionary elements as specified in Sect. 3.1. This description could be later used to derive the connection between the different UML elements in an UML profile. An analysis of the dependencies could be realised by a tree structure diagram with breadth-first search.

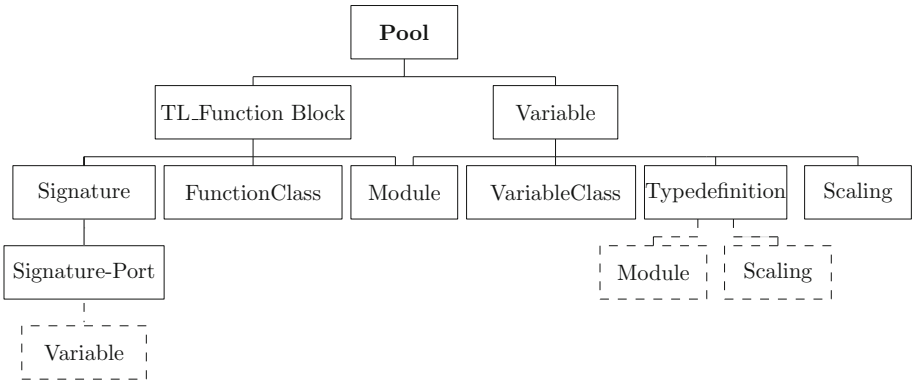

Fig. 1. Tree structure diagram of the Pool area

Figure 1 shows the Pool area with all defined elements of the TargetLink Data Dictionary. The figure clearly shows the dependencies between all elements. It is obviously depicted in Fig. 1 that Variables are independent from other TargetLink Data Dictionary elements. This means a Variable could exist without a TL_Function Block, e.g. global Variables. Figure 1 also indicates that Variables and TL_Function Blocks are split into different branches. This implies that those two elements need to be considered differently when defining the UML profile.

Another fact of the tree structure shows that a TL_Function Block is linked to Variables over a Signature and Signature-Port, whereas one Signature-Port of a Signature is linked to exactly one Variable. On the other hand a Variable could be linked to different Signature-Ports.

4 Prototype TargetLink Model in UML

Based on the previous definitions, in this section a prototype for describing TargetLink models in UML is suggested. The section is subdivided into two subsections. The first subsection explains the UML diagram type selected to be used for expressing the TargetLink model elements. In the second subsection it is described which element of the selected UML diagram type is mapped to a specific element of the TargetLink Data Dictionary.

4.1 Composite Structure Diagram

With composite structure diagrams it is possible to describe the internal structure of a class or the collaboration between different classes. It is similar to the component diagram and therefore as described in [9] on page 93 it could be combined with such diagram types. In addition, a composite structure diagram provides two different views on a system. The first view is the structural-dynamic view which focusses on the function of the system and which parts are needed for the functions. The second view is shown in Fig. 2 as a class diagram.

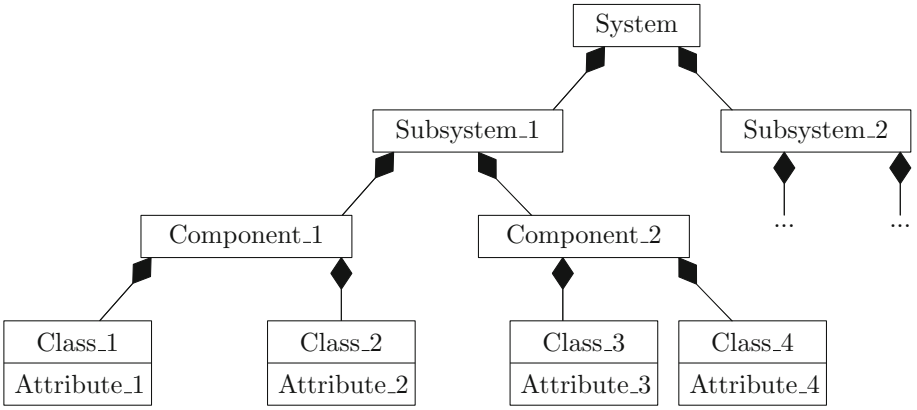

Fig. 2. Structural static view [8, p. 194]

In a first step the system is partitioned into various subsystems. In the following these different subsystems are divided into various components. Each component comprises several classes with attributes. This way to describe systems can be found on system level (Hardware) or on software level (GUI split into application, windows, etc.) [8, p. 194].

The metaclasses of the composite structure diagram are classes, parts, ports, interfaces and connectors. In the following all relevant elements are shortly described. A class is a kind of construction plan for an object. Parts are used to split a class into subsets. Parts could be extended by ports in order to implement

interaction points for communication between elements. To connect different ports of parts connectors are used. In the following section a possible prototype which describes how TargetLink Data Dictionary elements are mapped to UML elements is presented [9, p. 125ff].

4.2 Prototype

The prototype differentiates between a system and variable level. The distinction was made based on results of the previously described tree structure diagram analysis. On system level it is possible to describe TL_Function Blocks, Signatures and Signature-Ports. Whereas on variable level Variables, Typedefs and Scalings are described. In Figs. 3 and 4 both, the system level and variable level is depicted.

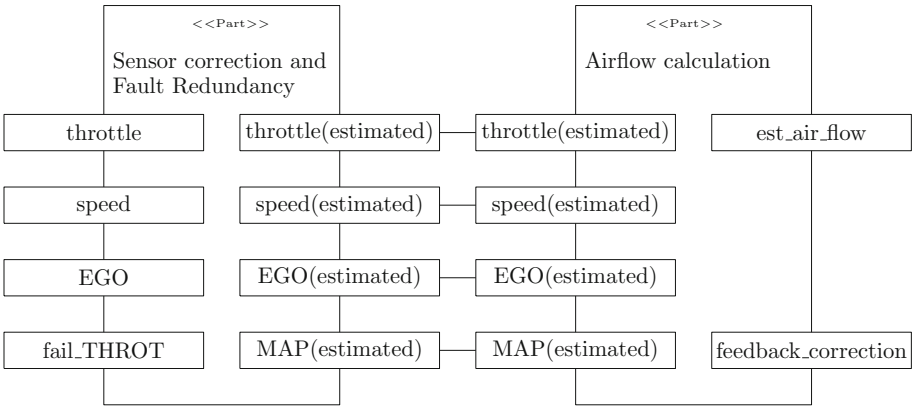

Fig. 3. System level in UML

UML Parts are used to separate a system into different subsystems whereas one Part contains the specification of a subsystem as a piece of the entire system. A TL_Function Block is an element that encloses all other elements except the global Variables with its Typedefs, Scalings, etc. A TargetLink model can consist of several TL_Function Blocks. Therefore, a TL_Function Block contains a certain aspect of the specification of a model. Thus, in Fig. 3 TL_Function Blocks are depicted as Parts. The formal description as defined in [8, p. 23f] summarizes TL_Function Blocks and Signatures. For this reason Parts include the description of Signatures as well.

Signature-Ports are mapped to UML Ports in the concept described in this paper. Parts could use UML Ports for communication with other UML Parts. Signature-Ports enable communication between TL_Function Blocks using Variables, as shown in Fig. 1. On Variable level all variables a TL_Function Block could contain are specified.

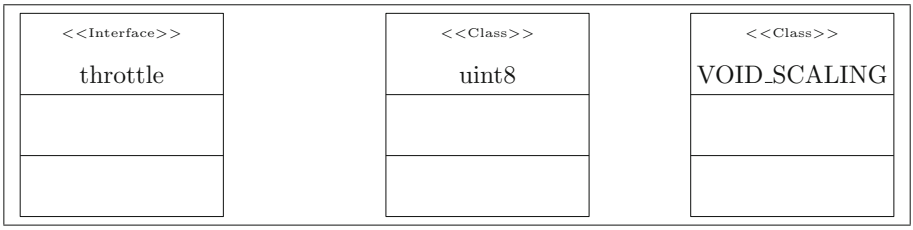

Fig. 4. Variable level in UML

As shown in Fig. 1, on Variable level it is possible to describe Variables and its dependencies. Furthermore, specification of variable types (Typedefs) and scalings are done on Variable level. Figure 4 shows an excerpt from a specification of Variable, Typedef and Scaling elements on Variable level used for communication between different TL_Function Blocks via Ports. In the TargetLink context Variables specify the interface of TL_Function Blocks. Therefore, in this concept Variable elements are mapped to UML interfaces.

Typedefs and Scalings represent a construction plan for Variables in TargetLink Data Dictionary. These elements describe the appearance of Variables in the production code. In UML classes have a similar functionality. They influence objects and provide so some kind of construction plan by attributes and operations [9, S.32]. For this reason Scaling and Typedef elements are represented by classes.

With mapping UML Parts to Signatures and TL_Function Blocks, UML Interfaces to Scalings and Typedefs, UML Class to Variables and UML Ports to Signature-Ports all profile relevant TargetLink elements are described in the UML profile. These elements are a subset of all TargetLink elements.

5 Definition of the TargetLink UML Profile

In the previous section a possibility on how to specify TargetLink Data Dictionary elements in UML was described. To enable specifying TargetLink Data Dictionary relevant elements in UML composite structure diagrams, the semantics of those diagrams need to be extended. In UML such kind of extensions are defined in UML profiles, mainly consisting of *stereotypes*, *tagged values* and *constraints*. Stereotypes expand the UML metaclasses and represent the extended elements. Tagged values expand stereotypes by properties as name-value pairs. Constraints define additional constraints for the stereotypes and enable users to check a model by the specified constraints. In the further course of this paper first the structure of different packages of the TargetLink Data Dictionary UML profile is described. In a second step the structure of the profile is illustrated. A more in-depth description of the UML profile can be found in [8].

5.1 Package Structure

This subsection describes the structure of the various packages of the TargetLink Data Dictionary UML profile. Different UML packages partition the various required elements of the UML profile. Figure 5 shows the structure of the TargetLink Data Dictionary profile with different packages.

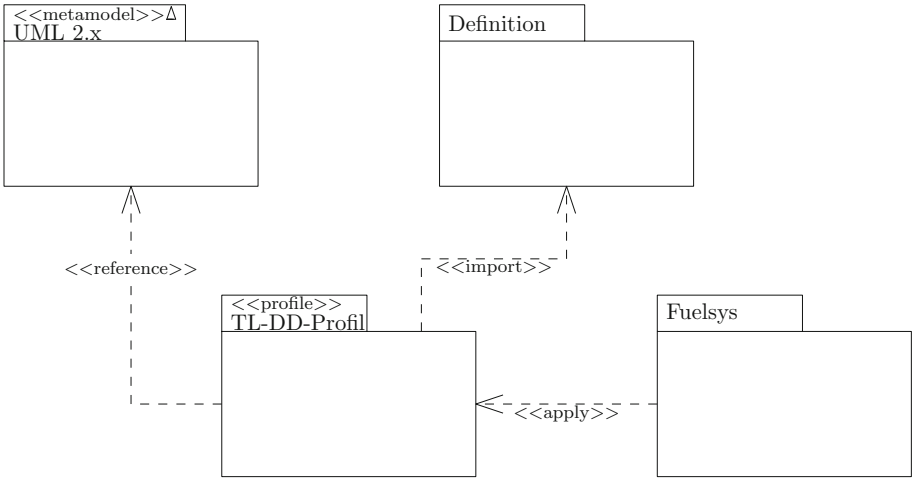

Fig. 5. Package structure

The figure shows a subdivision of the structure in four different UML packages. The package *UML 2.x* provides all metaclasses of the UML version 2.x. Because the TargetLink Data Dictionary profile expands the semantics of a composite structure diagram it is essential to use UML 2 or higher [10].

The UML 2 metaclasses used by the package *TL-DD-Profil* are provided via *<<reference>>*-connection to the package *UML 2.x*. This connection enables to expand metaclasses, such as ports or classes, with stereotypes. This kind of connection is called metamodel-reference because all metaclasses of UML 2.x which are provided with the *<<metamodel>>*-package are referenced by the profile package.

The definition of the profile itself can be found in the package *TL-DD-Profil*. The stereotype *<<profile>>* is used to declare that the package contains UML profiles. The *TL-DD-Profil* represents the key package within the package diagram. It contains all required metaclass extensions such as stereotypes, tagged values representing the TargetLink Data Dictionary properties and additional constraints. Using the *<<import>>*-connection the *TL-DD-Profil* package imports different predefined elements from the *Definition*-package.

As mentioned above, the Definition-package describes different elements to be used in the *TL-DD-Profil*. Among others, basic data types of the TargetLink

Data Dictionary are determined in this package. For example, predefined elements such as VariableClasses are defined in this package. The *Fuelsys*-package represents the actual application model which is described by applying the TargetLink UML profile.

5.2 Profile Structure

The following section explains the internal structure of the profile package TL-DD-Profil based on the previous prototype. The different metaclasses in the profile package are extended by stereotypes. The different metaclasses are provided by the <<reference>>-connection between profile package and UML 2.X package which is illustrated in Fig. 5. The different extensions of metaclasses by stereotypes are depicted in Fig. 6. This paper concentrates on the structure and the dependencies between the different stereotypes. A precise description of the UML profile is given in [8].

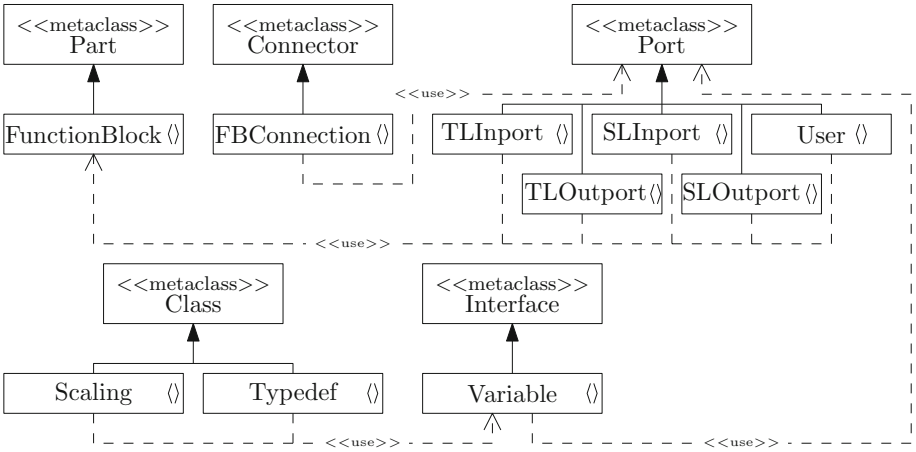

Fig. 6. Profile structure

The different <<*use*>>-connections, as depicted in Fig. 5, represent the associations between the different stereotypes. Furthermore, the connection shows which elements are in communication. For example the figure obviously shows the connection between TL_Function Blocks and different ports (e.g. TLInport, SLInport, ...).

Moreover, Fig. 5 shows which metaclass is extended by which stereotype. The prototype in Sect. 4.2 defines Scaling elements as a UML class. Therefore, the metaclass Class is extended by the stereotype Scaling. Another example is the stereotype Typedef which extends the metaclass Class as well. The Scaling element and the Typedef element will be provided by both extensions in the UML profile. Another element that is provided by the UML profile is the stereotype

Variable. In Sect. 4.2 Variable elements are defined as interchangeable elements that carries information. These elements are mapped to UML interfaces. Therefore, the UML metaclass interface is extended by the stereotype Variable. The so called variable level that is described in Sect. 4 only contains those three element types.

TL_Function Blocks, Signatures, and all kinds of Signature-Ports are applied on system level. Each TL_Function Block specifies a part of the entire system in the prototype and are therefore mapped to UML Parts. Thus, in Fig. 6 it is depicted that the metaclass Part is extended by stereotyp TL_Function Block.

Various stereotypes, representing different kind of communication ports, extend the metaclass Port. In UML, ports specify interaction points of parts and its environment. In the profile ports are used to represent the different Signature-Ports. Signature-Ports are interaction points of TL_Function Blocks used to consume and provide Variables. Signature-Ports can represent in-, out-, or user-ports. This is the reason why the metaclass Port is extended by different stereotypes. In detail the stereotypes differentiate only by its names. With the various Signature-Port stereotypes a user can directly recognize the kind of the Signature-Port.

The stereotype FBConnection extends the metaclass Connector. Signature-Ports can be connected to each other via FBConnection. These connections can not be mapped to the TargetLink Data Dictionary but in UML it shows the connection between all elements via ports. A user could easily recognize the interconnection between different TL_Function Blocks. Those connections represent the information flow within a system. A more detailed description of the UML profile can be found in [8]. The following section gives an overview on how to couple UML tools and the TargetLink Data Dictionary to exchange information.

6 Information Exchange UML - TargetLink Data Dictionary

With the UML profile defined in this paper it is possible to already add TargetLink Data Dictionary properties which will be used in a later step for production code generation to UML elements during system design. However an automatic exchange of those properties between the UML model specified during design phase and the TargetLink Data Dictionary is not yet possible. A common format to exchange information between programs and software tools is XML. Also the TargetLink Data Dictionary is able to import XML files containing a specific data structure. By importing XML files into the TargetLink Data Dictionary it is possible to either expand or overwrite available elements.

A standard for exchanging models between UML tools is the XML Metadata Interchange (XMI) which is published by the Object Management Group (OMG). The import and export of this XML based data format is supported by all major UML tools. XMI files describe complete UML diagrams in textual form. Even if XMI is a standard data format, each UML tool implements its own

interpretation of this standard, which slightly differs in structure of properties and used XMI vocabulary. In other words although XMI is a standard, each XMI-file exported from a different UML tool looks different. For this reason it is difficult to design a generic compiler for the transformation between XMI and TargetLink Data Dictionary XML.

In the following section we briefly describe the steps undertaken to design a transformation between UML XMI and TargetLink Data Dictionary XML. Due to the variations in the XMI standard as mentioned in the previous section, we focussed our analysis on the two most commonly used tools in the industry, Enterprise Architect [11] and Rational Rhapsody [12]. As a first step we have implemented the previously described UML profile in both tools. Based on this profile we design a demo system that includes all in Sect. 5 defined elements. In the second step the demo model was exported to XMI, using the standard export mechanism of the UML tools. Based on the exported XMI files we have to analyse the structure and the different vocabularies of both files. On the basis of analysis results a grammar is developed which describes the relevant items of the XMI export of both UML tools.

After the gramatic was derived from the XMI export, now the TargetLink Data Dictionary XML file is analysed. For each export, the structure and the vocabulary of this file is always the same. Therefore, it is sufficient to consider all in Sect. 3.1 defined TargetLink Data Dictionary elements. Also for the TargetLink Data Dictionary export a grammar is defined, which is similar to the one developed for the XMI export. By specifying transformation rules the XMI grammar is mapped to the XML grammar of the TargetLink Data Dictionary. The transformation rules describe an unambiguous mapping between the information of the TargetLink Data Dictionary and the XMI file.

As a last step an algorithm was implemented that is able to compile the XMI grammar in the XML grammar. If the grammar is expressed in an adequate language, e.g. antlr [13], a parser generator could be used to generate this algorithm automatically. The definition of the grammar and the transformation rules is not focus of this paper. A detailed declaration of these grammars can be found in [8].

7 Conclusion and Future Work

This paper summarizes a possibility to describe TargetLink Data Dictionary elements in UML to provide a smoother transition between software architecture definition and functional component design. Therefore, a UML profile was defined which provides various TargetLink Data Dictionary elements to be used in UML diagrams. The specified UML profile extends semantics of the composite structure diagram. Using this UML profile it is possible to connect UML tools to the production code generator TargetLink. This coupling is an important step in development of complex software architectures where software design tools such as UML as well as automatic code generators such as TargetLink became a defacto standard. The smooth transition between UML tools and

TargetLink supports the consistent exchange of software components and its interfaces between different engineering teams, responsible for software architecture and software component design. Following this approach, each development team could import their subsystems from the entire software architecture description automatically into the TargetLink Data Dictionary to describe the functional behaviour with MATLAB/Simulink and generate production code with TargetLink.

With the concept described in this paper we could not solve the problem that every UML tool needs its own compiler due to the ambiguous definition of XMI structure and vocabulary. A generic methodology of transforming UML XMI files to TargetLink Data Dictionary readable XML files is subject to future work.

This paper is a rough overview about the topic. More in-depth information on different aspects e.g. details on grammar specification and complete definition of the UML profile can be found in [8].

References

1. Unified Modeling Language (UML) Resource Page. http://www.uml.org
2. dSPACE TargetLink - Automatische Seriencode-Generierung. https://www.dspace.com/de/gmb/home/products/sw/pcgs/targetli.cfm
3. OMG Systems Modeling Language. http://www.omgsysml.org
4. MathWorks. http://de.mathworks.com/
5. Sjöstedt, C.-J., Shi, J., Törngren, M., Servat, D., Chen, D., Ahlsten, V., Lönn, H.: Mapping simulink to UML in the design of embedded systems: investigating scenarios and transformations (2007)
6. Walter, S: Übersetzung von UML-Software-Spezifikation in Simulationsmodelle. Fern Universität in Hagen (2014)
7. TargetLink Help Desk for Releases 2014-A
8. Falk, M.: Definition eines Profils zur Kopplung des Seriencode-Generators TargetLink an UML-Werkzeuge. Carl von Ossietzky University Oldenburg (2015)
9. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley, Reading (2005). Shanklin, J.C. ISBN 0-321-26797-4
10. Rupp, C., Queins, S.: UML 2 glasklar - Praxiswissen fr die UML-Modellierung, 4th edn. Hanser Verlag, Munich (2012). ISBN 978-3-446-43057-0
11. Enterprise Architect - Model Driven UML Tools. http://www.sparxsystems.de/start/startseite/
12. Rational Rhapsody family. http://www-03.ibm.com/software/products/de/ratirhapfami
13. ANTLR. http://www.antlr.org
14. OMG Systems Modeling Language - The official OMG SysML site. http://www.omgsysml.org