MODEL-BASED SOFTWARE DEVELOPMENT AND CODE GENERATION FOR AVIONIC SYSTEMS

Ines Fey[*], Matthias Hoffmann[*], Dieter Reiners[#],

[*]DaimlerChrysler AG, Research and Technology
Software Technology Research Lab
Alt-Moabit 96A, D-10559 Berlin, Germany

[#]DaimlerChrysler AG, Research and Technology
Power Electronics and Actuators Research Lab
Goldsteinstr. 235, D-60528 Frankfurt, Germany

{Ines.Fey | Matthias.Hoffmann | Dieter.Reiners@DaimlerChrysler.com }

Keywords: *model-based development, code generation, certification, SCADE, Matlab, RTCA/DO-178B*

Abstract

The use of software-based systems for safety-critical applications within civil aviation steadily increases, e.g. flight control functions today are to a large extent realized as software. Because of the criticality of these avionic systems, best quality and high safety is demanded for such software components. In order to guarantee the observance of these requirements, the development process is subject to consequential regulations and standards.

This paper introduces a new approach for a model-based development process for safety critical applications in civil aviation which is conform to the requirements of the RTCA/DO-178B document. This process called Qualified System Development Process (QSDP) was defined by DaimlerChrysler Research. The focus of the QSDP is the software development by modeling and code generation using the commercial available tools Matlab and SCADE.

1. Introduction

The use of software-based systems for safety-critical applications within civil aviation steadily increases, e.g. flight control functions today are to a large extent realized as software. Growing function scope, increasing complexity of systems and interconnections between various system components introduce new requirements for development processes and tool support. Furthermore, tight limitations of resources (time, market, people) require more efficient and cost-effective ways for software development.

Because of the criticality of avionic systems, best quality and high safety is demanded for the embedded software components. In order to guarantee the observance of these requirements, the development process is subject to consequential regulations and standards. Especially the standard RTCA/DO-178B has to be observed in the field of software development within civil aviation.

The current development of embedded software applies various techniques with low interaction. In order to describe the systems various different notations and tools are used. This leads to time consuming and error-prone transfers which usually occur manually. Errors in the realized software are often detected very late in the development process, usually during the HW/SW integration phase. Removing errors will then become very time and cost consuming.

Approaches for model-based development with the code generation based upon it offer

good opportunities for cutting down costs and ensuring software quality. Tools like Matlab[1] or MATRIX$_X$[2] which are widely used in the field of embedded controller development, however, do not meet the entire requirements of RTCA/DO-178B for code generators. That is why it was necessary to search for a new solution.

This paper contains the concept of the Qualified System Development Process (QSDP) which has been specified to satisfy the development constraints demanded by the RTCA/DO-178B. QSDP has been developed within a national research program by Daimler-Chrysler Research in collaboration with Dasa Airbus.

The QSDP concept is based on the use of model-based specifications and on the use of a qualified automatic coding mechanism. It thus supports automated generation of certifiable software on the basis of executable specifications with clearly reduced expenses for the implementation and testing and for the certification of this software according to aviation regulations.

The main focus of the QSDP is the software development. Nevertheless, considering the integration of the software development into the overall system development process the interfaces between software and system development were defined as well as the organization of processes accompanying tasks like requirements and configuration management.

The QSDP complies with the specific requirements of DASA Airbus and is focused on the design of systems of high criticality. It can therefore be concluded that the process – tailored to the respective requirements – can be applied for software development for different prospective projects for the development of avionics systems, independent of the respective software criticality.

To assure an efficient software development process and simultaneously sufficiently take into consideration the aspects of safety, all steps within this process have to be optimized and supported by adequate tools. During the QSDP project different scenarios for an automation of this process were discussed. In this context a Qualified System Development Environment(QSDE) has been specified and prototypically implemented.

The core of this environment is the task of modeling, simulation, and code generation using the commercial tools Matlab and SCADE. The transfer of the models from one tool to another is an essential part of the approach. Adequate translation mechanisms have helped to reach a consistent modeling support for this application. Furthermore, several special QSDP tools support the wide automation of the development steps.

Chapter 2 of this paper offers an overview of the advantages and constraints of model-based development and code generation. Chapter 3 refers to the constraints regarding Software development which have to be complied with according to RTCA/DO-178B. Chapter 4 describes the QSDP process. The paper concludes with a summary and outlook on further work in this context.

2. Model-based Code Generation

Controllers for software-based systems are increasingly defined in a graphical environment and are validated with simulation capabilities which are provided by development tools. Such a model-based software development for embedded systems is more and more recognized.

The precise description of the behavior and of the functional contexts with models offers with the help of data flow diagrams and state automatons extensive support for the developer, which is needed especially for very complex contexts like primary or secondary flight control.

Beside to the more precise, comprehensible and structured description of the function scope the approach of a model-based development offers a further two substantial advantages

- model-based function validation by applying the simulation capability and
- model-based code generation.

The simulation ability facilitates an early test of the functionality. By applying simulations for model testing it is possible to obtain faster feedback on the modeled functionality and in so doing changes can be introduced with low expenditure of costs and time. Because those errors in the functionality of the software which are detected when it is integrated into the hardware cause higher costs than insufficiencies found early in the development process with the help of simulations. In this way the overall development cycle can be broken up into several, shorter cycles (see Fig. 1), making them more efficient [8].

The utility of simulations at the early development stage is evident for systems which require special hardware. This hardware is often developed parallel to the software and is thus only available late in the development process. In that case the controller software can be executed in a simulated environment which represents the hardware behavior.

affect the system in the actual run-time environment. For example, whether non-functional requirements have been fulfilled (e.g. response times) can only be tested in the target hardware.

Model-based code generation means to automate the manual step from graphical function description to implementation. Based on the functional description the coding can be done by code generators which are supplied by the simulation tools. Erroneous and time consuming manual coding thus becomes redundant. Source code is generated within a couple of minutes. Hence the cycle times for changes are considerably shortened because the elimination of errors can directly be graphically performed in the model and code can be generated from the changed model by simply pressing a button. Prerequisite for the appropriate application of the advantages of code generation is a model whose functionality has already been sufficiently and systematically tested with the help of

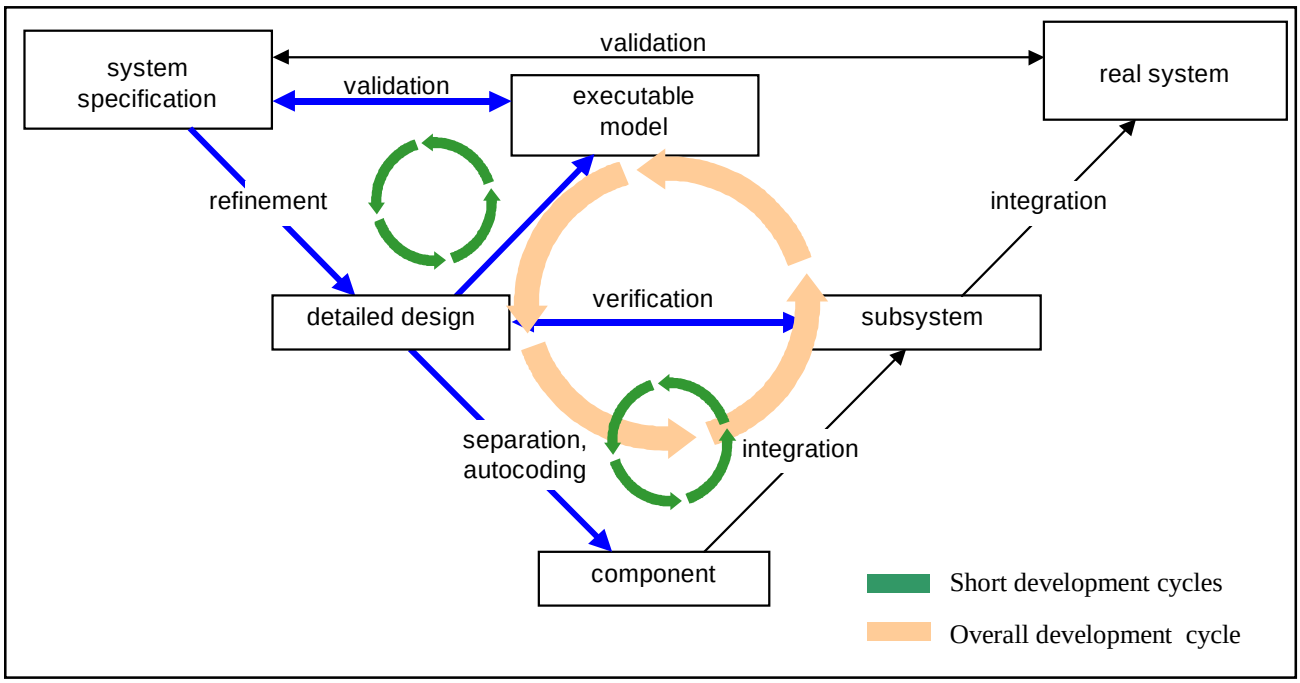

Fig. 1: V-shaped development process

However, simulations do not completely replace the later testing activities because several errors are hardware-dependent and only simulations and which is valid for the code generator.

A further aspect of using code generation is the reduction of the testing effort. If it is assumed that the generated code represents the identical function as the model, those tests that have already been considered in the form of model simulations become redundant as software tests. Considering that the test takes up up to 50% of the overall development effort this offers considerable options for cost reduction.

Model-based development and code generation are supported by various tools. Norms and standards in the field of civil aviation, however, introduce additional requirements which have to be fulfilled by the development process as well as by the applied tools. The tools are rarely specialized for development of civil aircraft applications and do not completely comply with aviation specific regulations. The aim during the design of the QSDP was to utilize the herein before mentioned advantages of model-based development and code generation in the civil aviation context especially for the use within Dasa Airbus.

3. Certification Aspects

The development of systems for civil aviation application is subject to various regulations, e.g. JAR25[7] and RTCA/DO-178B[10]. RTCA/DO-178B is of particular importance for the software development because it represents aspects of authorization for flying systems and equipment with regard to the applied software.

The RTCA/DO-178B describes all software development activities. These are assigned to the various processes - Planning Process, Integral Processes, SW Development Processes - into which the overall development process is subdivided. The standard specifies all preconditions, aims, and results for the processes. It also recommends which activities serve the achievement of these objectives and thus the certification of the systems.

The main focus of that document is predetermining aims and results that are to be achieved in the development process. It is therefore of major importance for the definition of a Qualified System Development Process.

The specific methods and tools needed for achieving the ends are not specified. The advantages of the tool application according to RTCA/DO-178B are the possibility to test automatically, the guaranteed adherence to standards, the avoidance of error sources through automatic generation, and the implementation of error tolerance strategies [5]. It has to be ensured, however, that the aim of the processes is not compromised by the tool application. The usage of tools in certain areas therefore needs to be certified in the form of tool qualification.

The qualification of a tool is necessary if the error behavior of the tool can lead to an unidentified error in the software[10]. This can happen if the output of the tool won't be tested and the tool either generates or tests parts of the software product. An example instanced by the RTCA/DO-178B is a tool which automatically generates the source code from low-level requirements. If the source code in that case is not tested or only partly tested, the generation tool has to be qualified.

Therefrom independent the source code in general is subject to certain safety-relevant requirements regarding consistency, the accordance to low-level requirements and software architecture, verifiableness, conformity to coding standards, traceability, and correctness. Some of these requirements (e.g. accordance to low-level requirements and conformity to coding standards) can be automatically secured from the generation process and need not be verified by later tests if the code generator is qualified.

Hence, in order to properly benefit from the options of reducing the costs of implementation and test activities, a qualified code generator has to be used.

All these requirements regarding modell-based code generation show exemplarily the necessity of considering the RTCA/DO-178B standards when planning and structuring the Qualified System Development Process. Only then it is possible to ensure that the developed

software will be approved together with the avionic system. For a complete overview about QSDP relevant regulations and requirements regarding the use of qualified tools refer to [9] and [10].

Within the research project QSDP DaimlerChrysler Research together with Dasa Airbus evaluated the constraints for this process, defined it and prototypically provided a possible tool environment. Further constraints like compatibility to Airbus partners and the need of tool support by approved and commercial tools resulted from the potential application field within Dasa Airbus.

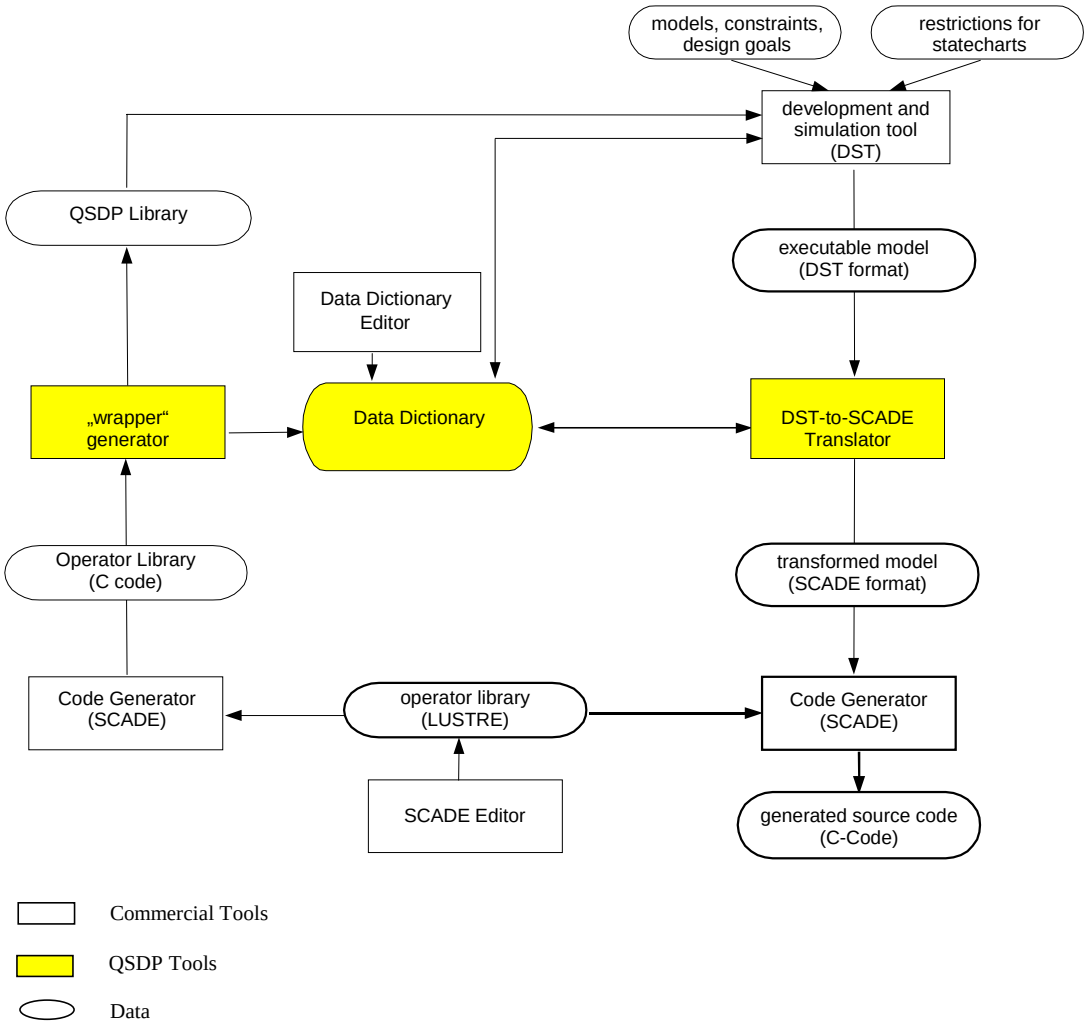

Fig. 2: Idea of the QSDP

4. Qualified System Development Process

The aim of developing the Qualified System Development Process (QSDP) was to define a process which is based on the approaches of model-based development and code generation and which also complies with the RTCA/DO-178B as well as with specific requirements of the application field within Dasa Airbus.

Though the definition of QSDP includes the concrete steps of software development as well as the embedding of the procedure into the system development, the following will concentrate mainly on the software development activities.

4.1. Definition of the QSDP

The most important prerequisite for software development with code generation and the utilization of automating this function (also see chapter 3) is the application of a qualified code generator. Since the qualification of tools according to RTCA/DO-178B is very complex, it is only the SCADE code generator which fulfills this prerequisite. That is why the SCADE code generator has to be considered a central element in the QSDP.

The SCADE tool [4] does offer the possibility of a dataflow-oriented function description, however, due to a different approach it does not provide the function scope which developers are familiar with thinking of tools like Matlab and MATRIX$_X$. Especially the used time model [3] and the restricted simulation features do not support the function devised for the QSDP [6]. Otherwise, the tight restrictions regarding implementation relevant data like value domains and data types of signals offer a suitable basis for the generation of reliable code. Summarizing the features of SCADE it is a capable tool for the detailed software specification and code generation. But analysis and design of continuous control components are not supported. Since latter is an integral element of the development of avionic control systems a second development tool has been introduced into the QSDP workflow which closes this gap.

Therefore two tools are applied in a combined manner during the QSDP. On the one hand a so-called development and simulation tool (DST) is applied in order to provide the necessary analysis and simulation capabilities. Tools like Matlab or MATRIX$_X$ can be used as DST. On the other hand the above mentioned specification facility of SCADE is used for the definition of implementation relevant data and finally for the code generation. The functional overlapping when applying these tools should, however, be kept as low as possible.

In order to realize this concept a tool environment Qualified System Development Environment (QSDE) was defined. Core of this environment are modeling, simulation, and code generation using the commercial tools Matlab and SCADE. So as to ensure a far reaching automation of these development phases some separate process steps have to be supported by QSDP specific tools. For those functions that are not covered by available commercial systems special tools have been specified within the scope of the project. Fig. 2 shows the idea of the Qualified System Development Process and the interconnection of the commercial tool families. The most important components and the workflow of the QSDP are presented in the following sections.

4.2. Tool Environment

Two components have been identified as indispensable for this environment: a development and simulation tool (DST) and a code generator which enables the generation of certifiable software. Various tools are available for the DST. These, however, differ in their functionality [6]. For the prototypical realization of the QSDE Matlab/Simulink/Stateflow (Matlab) was used as development and simulation tool and SCADE was applied as the code generation tool. Furthermore, an editor for ASCII-based files is needed in the QSDE.

Next to these commercially available tools additional tool support is needed for an extensive automation of the process. Especially the translation of the models from the Matlab environment to SCADE is performed by special QSDE tools. Fig. 3 gives an overview of the Qualified System Development Environment. The gray components represent the QSDP specific parts of this tool set.

Essentially three components are needed for automating the transformation of a Matlab model according to SCADE format (compare fig. 2): the "Wrapper" Generator, the DST-to-SCADE-Translator and a Data Dictionary. The latter is used for exchanging information between the named tools as well as for storing additional user information.

Most important for the acceptance of the process is the integration of that newly developed tools with the current commercial ones in such a manner that an automated environment for the development of safety-critical software becomes available. The separate tools and their application in the QSDP will be described in the following.

4.2.1 Development and Simulation Tool (DST)

The development and simulation tool, in this case Matlab, is an efficient tool for modeling and simulation activities which supports block oriented as well as state-chart oriented notations. Extensive functions for the simulation of different system components like continuous control and discrete state control are also supplied.

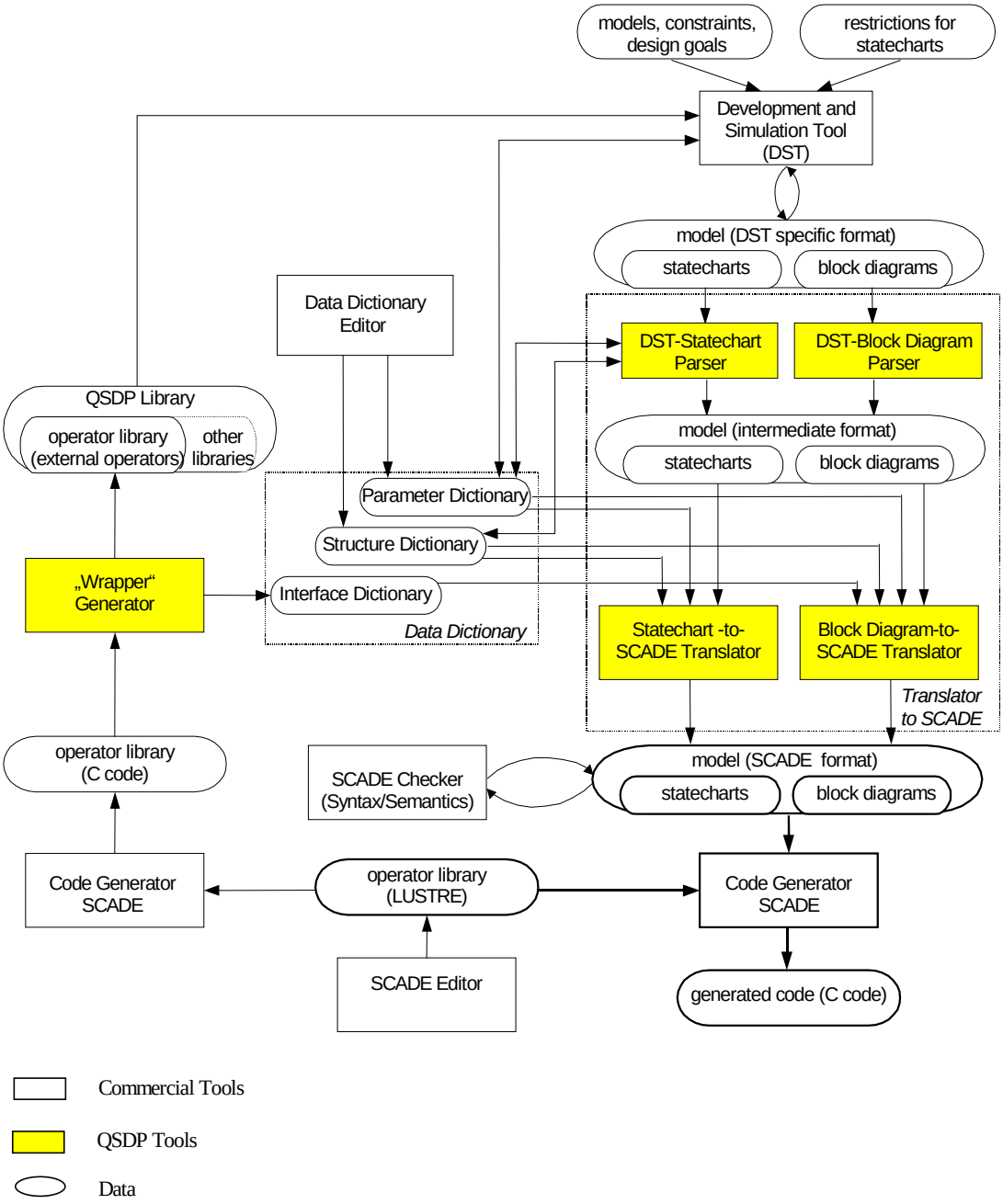

Fig. 3: Details of the Qualified System Development Environment (QSDE)

During the QSDP the complete modeling and simulation is performed in the Matlab environment. The following constraints have to be considered when modeling the component which is later on used for code generation

- only elements from the special QSDP library can be used for modeling the controller
- parameters have to be defined explicitly using the Data Dictionary
- a QSDP naming convention for signals has to be observed.

This ensures that the code generation can follow.

All other model parts which are needed for the simulation like the environment model can be generated by utilizing the complete Matlab functionality.

The function description can then be validated with the help of the simulation possibilities. From the graphical description a separate ASCII-file (.mdl-file) is generated for the controller component which in turn can be further processed by the QSDE tools.

4.2.1 The DST-to-SCADE-Translator

The DST-to-SCADE-Translator converts the ASCII-file created by Matlab into the SCADE format in two steps.

First the Matlab model is converted into an intermediate format and then based on this description the SCADE format of the model is created. During the translation process the data dictionary information is used to complete the SCADE model.

4.2.2 SCADE

Mainly the code generator and the syntax checker from SCADE[4] are used during the QSDP. The code generator is applied for generating source code from those models that have been transformed into SCADE format. These models can be checked by the syntax checker prior to this activity. In exceptional cases the SCADE editor can also be used, e.g. if the operator library (see 4.2.4) is to be extended.

4.2.3 Text Editor

An additional editor is also needed which edits ASCII files within the normal development environment. This editor will at first be used as Data Dictionary Editor. Based on the experience gained from the first evaluation examples a special QSDE Data Dictionary Editor can be developed and realized which facilitates the specification of the needed data via a suitable GUI and also executes syntax checks.

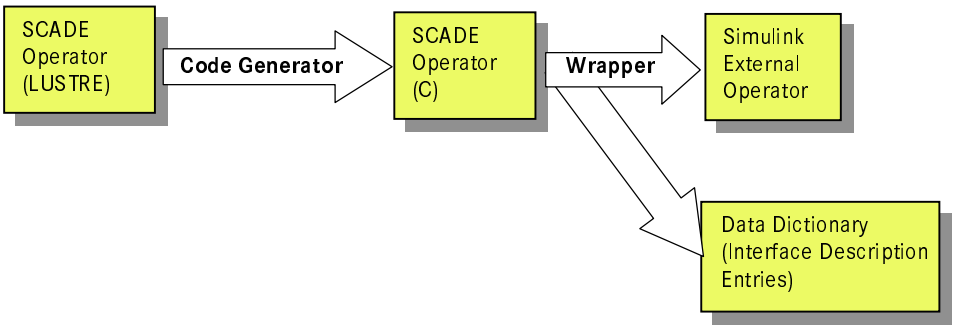

Fig. 4: Generating external operators for Simulink using the QSDP-Wrapper

4.2.4 The "Wrapper"-Generator

The "Wrapper"-Generator is an automatic translation program which converts LUSTRE operators and user libraries in SCADE format in such a way that they can be used within the DST environment as external operators. The interface dictionary is created during this transformation. It contains all information about the interface of the original SCADE operators which would otherwise get lost during the translation process (e.g. input and output data types). The idea of the wrapping mechanism is

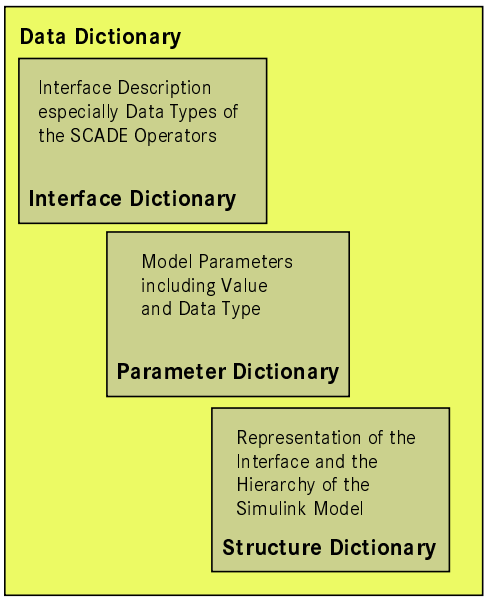

Fig. 5: Parts of the data dictionary

shown in Fig. 4. The external operators are provided by a specialSimulink library so-called QSDP-Library.

4.2.3 The Data Dictionary

The Data Dictionary is used for saving additional information which is not directly included in the DST model but needed for the code generation with SCADE. These data are assigned to the following groups with respect to their content and source:

- Interface information
- Information on the modeling structure
- Information on model parameters.

The Data Dictionary (Fig. 5) thus consists of three parts: Interface Dictionary, Data Dictionary and Structure Dictionary. Fig. 3 shows which tools access these dictionaries, either by reading or writing. It has to be noted that the Parameter Dictionary and the Structure Dictionary can be directly altered by the user with a Data Dictionary Editor. The Interface Dictionary, however, contains the information that has to be automatically determined.

The QSDP tool environment comprises commercial systems used in practice as well as tools especially designed for QSDP. In this way the integration gap between the "modeling and simulation" phase and the "detailed specification and code generation" phase can be closed with the help of the QSDP-specific tools. The developer thus obtains a tool environment adapted to the QSDP concept for the efficient generation of safety critical software.

5. Summary and Outlook

A number of standards apply for the development of avionic software whose requirements bear upon the criticality of the systems.

Aim of the Qualified System Development Process and the QSDP tool environment is to provide far reaching conceptual tool support for the development of certifiable avionic software. The focus of this environment is the constructive work in software development from modeling the software system to code generation.

With the help of this approach software can be developed by using state-of-the art techniques (e.g. functional descriptions in the form of data-flow and state chart models). The quality of the overall process increases.

The tool environment designed during the project offers the necessary devices for automating the operations, in this way relieving the developer of time consuming manual translations between tools. No typos or more severe inconsistencies between the different model notations will weaken the quality assurance. The code generated with the SCADE Code Generator is certifiable according to RTCA/DO-178B. This approach can therefore also be used for applications of the highest criticality.

Within the scope of the QSDP project the application of the concept has been demonstrated using the example of a water ballast system. Therefore a prototype of the Qualified System Development Environment has been implemented. On the basis of experiences from this application the tool chain implementation can now be adjusted.

6. References

[1] *Using Simulink*. The MathWorks Inc., Natick, MA, (1997)

[2] *'Introducing MATRIX$_X$ Version 6.0'*. Integrated Systems Inc., 1997

[3] *SAO+/DF Language Reference Manual, Version 1.3.2*. Verilog SA, Paris, 1996

[4] *SCADE Release Note, Version 2.1*. Verilog SA, Paris, 1998

[5] Fett,A,, Rumprecht, B. *Stand der Technik bei der Qualifizierung von Werkzeugen zur Software-Entwicklung*, Technical Report No. F3S-97-005, Daimler-Benz AG, Berlin, 1997

[6] Fey, I., Hoffmann, M., Janning, J., John, G. *Werkzeugbetrachtung: Spezifikations- und Simulationswerkzeuge*, internal Project Report, Daimler-Benz AG, Berlin, 1997

[7] JAR 25 - *Joint Airworthiness Requirements*. Part 25: Large Aeroplanes, Change 13. Joint Airworthiness Authorities, 1990

[8] Reiners, D., Fey, I., Thomas, C., *Autocoding sicherheitsrelevanter Software in der Luftfahrt*. Proceedings DGLR conference, Berlin, 1999

[9] Thomas, C., Munk, J.-P. *Anforderungsanalyse: Zulassungsaspekte*. internal Project Report, 1997

[10] RTCA/DO-178B: *Software Considerations in Airborne Systems and Equipment Certification.* Requirements and Technical Concepts for Aviation (RTCA) Inc., 1992