Multilanguage Design of a Robot Arm Controller: Case Study

G. Nicolescu, P. Coste, F. Hessel , P. LeMarrec, A. A. Jerraya
TIMA Laboratory 46, avenue Felix Viallet, 38000 Grenoble, France
gabriela.nicolescu@imag.fr

Abstract

This paper discusses a case study, the multilanguage design of a robot arm controller. The system is composed of two parts: the controller and the motors of the robot arm. The design starts with a multilanguage system-level specification. SDL was used for the description of the controller and Matlab was used for the description of the physical behavior of the motors. The multilanguage design flow includes the refinement of the modules and the refinement of the communication between these modules described in different languages. In order to validate the overall system through the design steps, cosimulation is performed at different levels of abstraction. This experiment shows that system level model may provide an early mock-up of the design before refining the intermodule communication.

1. Introduction

Nowadays, the use of more than one language correspond to a current need in embedded system design. The design of large systems may require the participation of several groups belonging to different companies and using different design methods, languages and tools [7].

In traditional design approaches, separate groups design the different parts of the system and the integration of the overall system is made at the final stage. This scheme may induce extra delays and costs because the synthesis of the interfaces between the different parts and the validation problems that arise at a very late stage of the design process. Therefore, new design approaches are needed to handle these cases where different languages and methods need to be used within the same design.

Multilanguage design constitutes an important step in this direction. It gives the designer the ability to validate the whole system behavior before the implementation of any of these parts. Multilanguage system design offers many advantages including efficient design flow and shorter time to market. The key issues are the validation of the overall system through cosimulation and the interlanguage communication synthesis. Cosimulation is currently very popular for electronic system design. Several existing works provide methodologies for hardware-software cosimulation at the RTL and behavioral level. Coware [14], Seamless [8] and the work described in [15] are typical environments supporting such a co-design scheme. Few systems in the literature tried to tackle the multilanguage co-design at the system-level. These are RAPID [13], Ptolemy [10] and SEA [9]. However very little work was done in the field of the refinement of communication between modules described using different languages. Recently, some codesign tools [12], [14] tried to tackle this problem for solving hardware-software interface synthesis, starting from a mixed hardware-software model where the specification of the communication interface is made using restricted or low level communication model.

The main contribution of this paper is the discussion of the multilanguage design application starting from a specification that include high-level communication model. We use a robot arm controller description realized in SDL and Matlab. The design flow includes the refinement of both communication and modules. This flow allows an early validation of the robot arm controller functionality and an early analysis and validation of the interaction between the controller and the mechanical part. In this experiment, we use MUSIC [3], a multilanguage design tool and MCI [5], a multilanguage cosimulation tool.

The rest of the paper is organized as follows. The next section introduces the application, a robot arm controller. Section 3 deals with the design process of the robot arm controller. The results obtained and the evaluations of the work are presented in sections 4 and 5. Finally, section 6 provides our conclusions.

2. The robot arm controller

The controller is a multi-motor controller of a robot arm. The control is intended to avoid discontinuous motor operation problems. The system receives from a host machine, the information about the movement that the robot arm must realize. Using this information, the controller adjusts the speed of each motor,

for a continuous movement of the arm. In order to allow a clear explanation, we will restrict the model to two motors only. Despite this simplification, the robot arm controller constitutes a quite complex system.

2.1. System requirements

The robot arm controller is a complex system composed of two communicating modules: an electronic part, the controller and a mechanical part, the two motors. Because of the nature of the behavior performed by these two modules, different languages are needed. The electronic part performs a control function and is easier to be described using an extended FSM based language. The mechanical part is easier to be described using a continuous model. This system is clearly a heterogeneous system and two languages are used for the high-level specification:

- SDL employs an extended finite states machine (EFSM) model of computation and is suitable for control oriented systems [2]. SDL processes communicate with each other asynchronously through infinite FIFOs. SDL will be used to describe the controller module.
- Matlab [11] can specify continuous calculation, data are transformed continuously from input stream to output stream. Streams are modeled as vectors or matrixes, and transformations are modeled as functions with input and output parameters. Matlab will be used to describe the motors module.

Since SDL and Matlab languages are based on different concept for data exchange, the interpretation of the link between the two subsystems will need a specific communication synthesis step [1]. Consequently, handling the communication between the electronic part and the mechanical part, becomes a difficult task [7].

2.2. System-level specification

The system-level specification is composed of the SDL controller specification and the Matlab motor specification. Figure 1 shows the abstract view of the controller and the motor algorithms.

Figure 1.a presents the controller algorithm. Using the information about robot arm trajectory, the controller computes the distance to go for each motors. This will be compared with the current position in order to generate the new speed value.

The motor algorithm is represented in Figure 1.b. Each motor reads the speed order sent by the controller and calculates its current position. This parameter is sent to the controller, in order to calculate the next speed order.

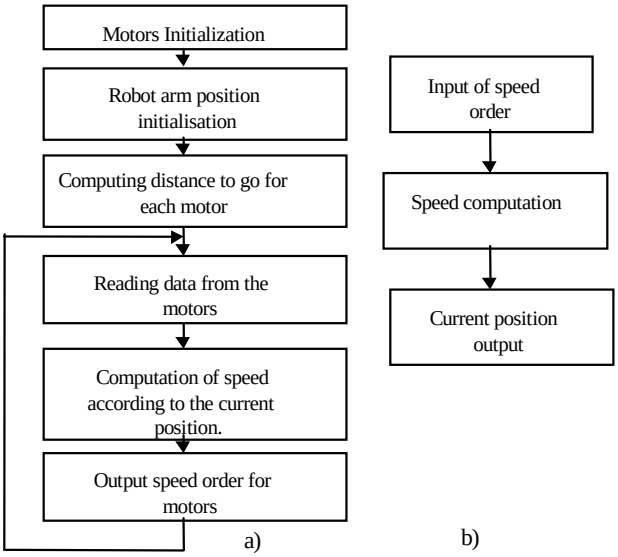

Figure 1. High level specification
a) Controller algorithm b) Motors algorithm

2.2.1. SDL controller specification. We used SDL for the specification of the controller. Figure 2 shows the SDL specification of the controller.

This specification is composed of three main blocs:

- The *Host Machine* represents the controller testbench. It initializes the controller by sending to it the information concerning the trajectory that the robot arm must realize.
- The *Controller* bloc calculates the speed required for each motor in order to make all the motors reaching the desired position in the same time. This block is composed of three processes: *Dist*, *Ctrl1* and *Ctrl2*. *Dist* receives the information concerning the trajectory that the robot arm must realize. It identifies for each motor the distance to go. *Ctrl1 and Ctrl 2* store up to date information about the current position of each motor. Comparing the current position of each motor with the required position, these processes generate the speed orders for the motors.
- The *Sampler* bloc is the interface between the controller and the two motors. It is responsible for the transformation of the continuous signals sent by the Matlab bloc (robot arm), in discrete signals for the SDL bloc (controller).

The interface of the SDL model is composed of logic ports, named accesses, which are used by high level primitives such as send, receive, put or get. Each access will be refined into several ports according to the selected protocol.

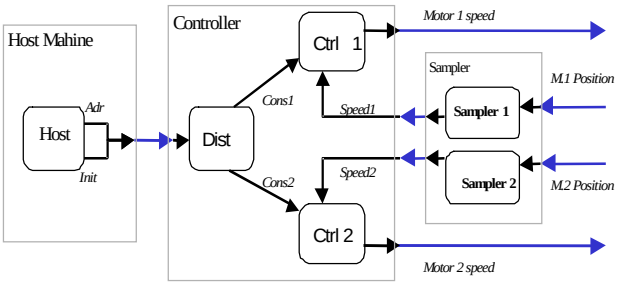

Figure 2. Structure of the robot arm controller described in SDL

2.2.2. Matlab specification of the motors. Matlab was used to model the physical behavior of the motors. Matlab specification is composed of two independent processes that describe the motor behavior. Each process executes the algorithm presented in figure 1.b. Figure 3 shows the Matlab schema for the mechanical environment.

On the Matlab side, communication is made through physical ports connected to wires. In this application, the interface is composed of five ports. Three are used to receive the speed data using a handshake protocol. The other two ports are used to send the position information to the SDL module.

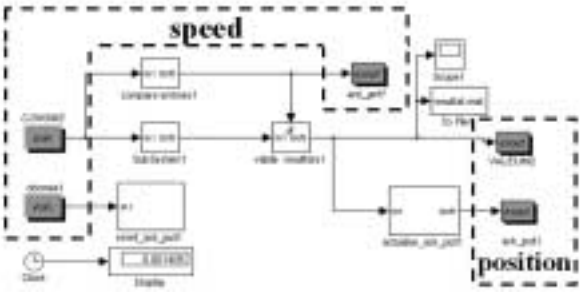

Figure 3. Motors Matlab Specification

2.2.3. The coordination between the Matlab and SDL modules. The overall system interconnections, including SDL and Matlab instances, are described using a coordination file that gives the interconnection between the blocs. This coordination file is a “system-level netlist” that specifies the interconnection between different modules. We used SOLAR intermediate format [6] for the description of the coordination between modules desribed in different languages.

SOLAR provides the concepts of heterogeneous channel and connector for the interconnection of modules with heterogeneous interfaces. The connector concept [5] hides detailed interfaces (set of physical ports). The heterogeneous channel allows to connect modules communicating via different protocols.

A module interface in SOLAR may include different entities: ports, accesses or connectors. A heterogeneous channel is an abstract net that links interface entity belonging to different modules.

Figure 4 shows the graphical representation of the coordination file, in our application case.

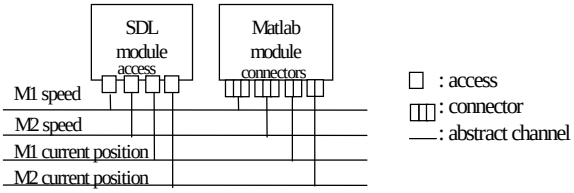

Figure 4. Graphical representation of the coordination file

The two modules are interconnected through four abstract channels (M1 speed, M2 speed, M1 current position, and M2 current position). Each abstract channel links an access belonging to the SDL module with a connector belonging to the Matlab module.

3. The design approach

The design flow combines three basic techniques (figure 5):

- The refinement of the high level specification into an implementation: in our case, this includes the refinement of the SDL module into a VHDL implementation.
- The communication refinement: this technique consists in converting the abstract channels described in section 2 into detailed protocols.
- The cosimulation: this technique allows the validation of the overall system at different abstraction level.

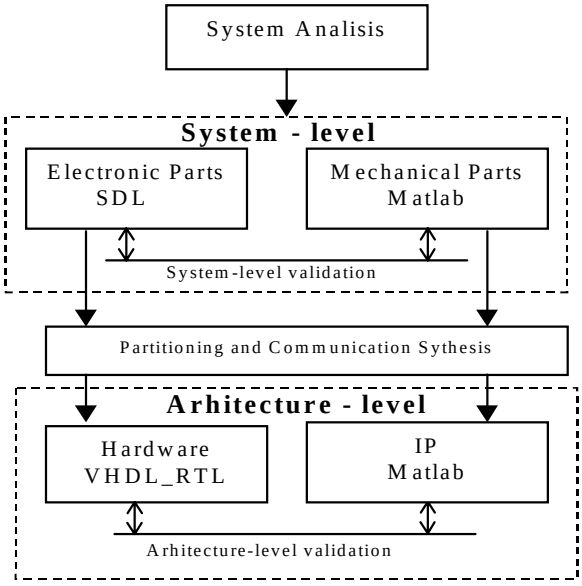

Figure 5. Design flow adopted for the robot arm controller design

The design flow includes four main stages:

- System-level specification and validation (SDL Matlab cosimulation).

The design starts with an analysis of the system requirements and a high level definition of the various function of the system. At this stage we obtain a multilanguage system-level model given in SDL-Matlab. The validation includes cosimulation and verification. This step ensures the functional validation of the system. During the cosimulation step the SDL and Matlab simulators are executed concurrently. This allows to debug and analyze the different modules. The cosimulation tool interprets the abstract communication between the SDL and the Matlab models.

- SDL refinement

This step generates a hardware model described in VHDL from the SDL module. The SDL model is composed of a set of concurrent high level processes communicating through message passing and synchronized through infinite queue. This model is refined into an implementation composed of VHDL blocs communicating through signals [4].

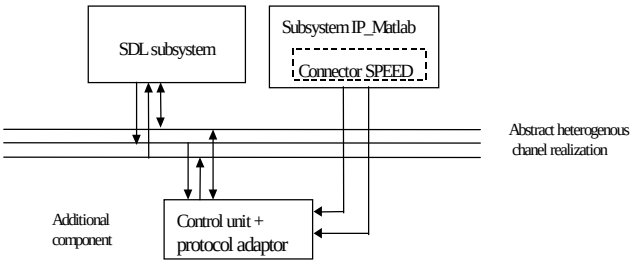

Figure 6. System modelisation after the communication synthesis

- Interlanguage communication synthesis

During this step the abstract communication channels are mapped on explicit communication protocol.

The communication synthesis is realized in two main stages [4]:

The first stage is the protocol choice and allocation. The communication protocols are specified in a communication library. The allocation stage determines if the protocol chosen to realize the abstract channel respects the requirements of this channel.

The second stage is the interface synthesis. This stage consists to realize the abstract channel in conformity with the chosen protocol. For each channel, a dedicated component is added. This component establishes the blocs' communication and adapts protocols when necessary.

Figure 6 shows an abstract channel mapped on explicit communication protocol after the communication synthesis process.

- Architectural validation (VHDL-Matlab cosimulation):

Cosimulation technique is used again in order to validate the produced VHDL-Matlab model and the communication protocol. VHDL blocs are executed on Synopsys VSS simulator and Matlab Simulink executes the models of the motors.

In our experiment case, we used the MUSIC tool for the modules and communication refinement and the MCI tool for the validation of the overall system at the different abstraction level.

4. Results

This section presents the main results regarding the cosimulation and the communication synthesis in the case of the robot arm controller.

The design of the robot arm controller has been realized starting from the initial specification in SDL and Matlab. The design flow has been achieved using MUSIC.

4.1. Cosimulation results

The simulation times results from the granularity of the simulation step. A step corresponds to a simulation cycle. It has different representations from an abstraction level to another:

	System Level (SDL-Matlab)	Architecture Level (VHDL-Matlab)
Simulation step	Transition	Computation Step
Full simulation cycles	3 280	57 000
Cycles/sec	31	11
Full Simulation Time	1min 45s	1h 30min

Table 1. Results of the system level and architecture level cosimulation

- At the system-level, a simulation step corresponds to a transaction between parallel processes. For our application, a full simulation scenario requires 3280 cycles.
- At the architecture level, a simulation step represents a typical behavioral VHDL computation step. In the case of our application a full simulation requires 57000 cycles.

Given the performance of the cosimulation tool, we are able to perform a full simulation of the application at the system-level in less than two minutes. At architecture-level, a full simulation can be done in about ninety minutes. In this case, the simulation time of the system level model is 45 times faster than the architectural model.

These results show the advantage of the system-level cosimulation because higher level cosimulation includes fewer cycles and is faster than lower level simulation regardless of the cosimulation speed.

4.2. Communication synthesis results

Table 2 shows the results of the interlanguage communication synthesis concerned to the number of communication lines in the corresponding description:

	Number of lines		
Model	Behavior	Communication	Increase
SDL	292	16 (5%)	
VHDL	2810	646 (23%)	962%

Table 2. The size of the SDL code and of the VHDL code produced.

In SDL, one instruction is sufficient for sending or receiving a signal: the communication protocol, the signal routing and the queues are implicit. During the refinement steps, the communication becomes explicit using specific protocols and communication controllers. The different modules are connected through physical bus. In the case of our application, at the system level, the communication represents 5% of the SDL specification and 23% of the VHDL implementation. This represents an increase of 962% of number of lines.

4.3. Evaluation

This section presents the evaluation and learned lessons from this application. These are related to the capabilities and the limitation of the complex systems design starting from a multilanguage system level specification.

The results presented in section 4 showed the benefit of a high level specification for heterogeneous systems. The system level specification allowed an earlier validation of the system. Errors were founded quickly, and was easier and cheaper to correct. Moreover, the SDL-Matlab cosimulation was 45 times faster than the VHDL-Matlab cosimulation time. In the case of our experiment, using the system level specification we realized the analysis and the optimization of the motors and controller algorithms. The interaction between the two modules was also validated. This stage ensured us that the system that will be implemented does not present functional errors.

The system level model was used as a mock-up of the system, in order to fix the final specification. Thus, at the architecture level one of the validation parameters was the conformity with the results obtained using the system level specification. Figure 7 shows the graphs of the output motors at the two abstraction level. We can see easily that the both curves are similar.

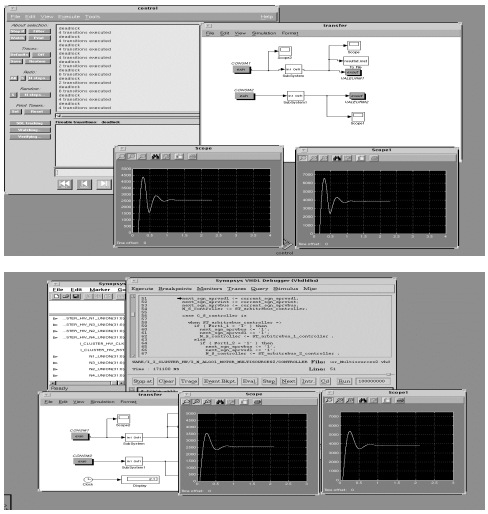

Figure 7. Similar results obtained from cosimulation at the two abstraction level

The design of the robot arm controller showed clearly the capabilities of the adopted approaches.

Their main strengths are:

- the communication synthesis approach adopted allowed the refinement of interfaces synthesis from SDL-Matlab into VHDL-Matlab specification. Moreover, it provided the adaptation of the different interfaces entity (accesses and connectors).
- each module was designed using the appropriate tool and design method.
- an eventual evolution of the system is possible. The modification of the one module functionality is independent of the other system modules. The addition of the other modules is also allowed.

However, this application pointed out several weaknesses in the approach. These are mainly related to the current version of the used system:

- the need of the communication library that must be provided by the user. In the case of our application the addition of a “multi rendez-vous” protocol was necessary.
- it was difficult to evaluate the results of the communication synthesis when selecting protocols. A high level estimator tool is also required to help the user during this step.
- the user must provide a coordination file written in a specific language (the SOLAR intermediate format). The different steps of the refinement process are transformations of this internal model. This intermediate format lacks the performances of a standardized language.

5. Conclusion

This paper presented the results of the multilanguage design of a robot arm controller. The initial description of the system was given in SDL and Matlab and an early analysis and assessment of the whole system was achieved. The design process was performed using, a multilanguage co-design environment. This included the refinement of the modules and the refinement of the communication between the different modules. The validation of the system at each abstraction level was achieved using cosimulation. This experiment has shown that the multilanguage system level model may be used as a mock-up of the design in order to validate the overall behavior. And this before the refinement of the different modules and the refinement of the inter-modules communication. This result was obtained thanks to the simulation speed and accuracy at the system level.

References

[1] P.Bjuréus, A. Jantsch, "Heterogenous System-level Cosimulation with SDL and Matlab", FDL'99, Lyon, France.

[2] F. Belina, D. Hogrefe and A. Sarma, "*SDL with Applications from Protocol Specification"*, Prentice Hall International, 1991.

[3] P. Coste, F. Hessel, Ph. Le Marrec, Z. Sugar, M. Romhdani, R. Suescuin, N. Zergainoh, A.A. Jerraya, "Multilanguage Design of Heterogenous Systems", in Proc. of CODES'99.

[4] J.M.Daveau, G. Marchioro, T. Ben-Ismail, and A.A.Jerraya "Protocol Selection and Interface Generation for Hw-Sw Codesign", *IEEE Trans. on VLSI Systems*, vol.5, no.1, 1997, pp. 136-144.

[5] F. Hessel, P. Coste, Ph. Le Marrec, N. Zergainoh, J.M. Daveau, A.A. Jerraya, "Communication Interface Synthesis for Multilanguage Specifications", in *Proceeding of RSP'99*, Clearwater, Florida, USA, pp. 15-20.

[6] A.A. Jerraya and K. O'Brein, "SOLAR: An intermediate format for system-level design and specification", in IFIP Inter. Workshop on Hardware/Software Codesign, Grassau, Germany, 1992.

[7] A. Jerraya, M. Romhdani, CA Valderama, Ph. Le Marrec, F. Hessel, G. Marchioro and J. M. Daveau, "Languages for system level specification and design", in *Hardware/Software Co-design: Principles and Practice*, J. Staunstrup and W. Wolf ed, Kluwer, 1997, pp. 307-357.

[8] R. Klein, "A Hardware Software Co-Simulation Environment", from RSP'96, IEEE CS Press, 1996, Miami, pp. 173-177.

[9] B. Kleinjohann, "Multilanguage Formalism MEDEA/ESPRIT conference HW/SW Codesign", 1998 (Sept), pp. X.1.1.-X.1.22.

[10] B. Lee and A. Lee, "Hierarchical Concurrent Finite State Machines in Ptolemy", Proc. of International Conference on Application of Concurrency to System Design, 1998, pp. 34-40.

[11] MathWorks1998. Matlab. http://www.mathworks.com

[12] M. O'Nils, "Specification, Synthesis and Validation of Hardware/Software Interfaces", Sthockholm, 1999.

[13] N.L. Rethlman and P.A.Wilsey, "RAPID: A tool for Hardware/Software Tradeoff Analysis", from *Proceedings CHDL'93*, Elsevier Science, Otawa, Canada, 1993 (Apr.).

[14] K. Van Rompaey, D. Verkest, I. Bolsens and H. de Man, "Coware – a Design Environement for Heterogenous Hardware/Software Systems", from the European Design Automation Conference, Geneve, 1996 (Sept.).

[15] D.E. Thomas and J.K. Adams and H. Schmit, "A model and methodology for Hardware-Software Codesign", *IEEEDTC*, 10(3), (Sept.), 1993, pp. 16-28.