Towards Standard Control Protocol through Internet Using MDE Approach

Bassem KOSAYBA
Department of Software Engineering & Information Systems
Damascus University, Syria
script.java@gmail.com

Raneem SALEH, Rain ALSALEH
4th year Students – Albaath University, Syria

Abstract - **The computer sciences have witnessed a real tendency for using Internet applications and as a result a big number of techniques have existed and performed a lot of development in a standard time, but the big development to the internet applications has achieved by coming an XML technique.**

We propose in this paper a new protocol "ControlXML" to standardize the remote circuits control domain through Internet with benefiting of an XML technique. By achieving it we compass some of the users' wishes. These users own the feasible ideas, but they suffer from the programming experiment dipping. In order to realize our protocol we present a "model-driven framework". We determine two types of information. Information describes the abilities of the circuit and information describes the method of using the circuit to solve some problems.

To achieve that we produce two tools. The first tool allows the circuit designer to specify the circuit abilities. The second one allows the control circuit programmer to load a control circuit specification (commands and states) in order to build a control program specific to this circuit. After binding the outputs of these tools we will have an XML file(s) which will be sent to a remote virtual machine that can understand the logical structure of this XML file(s) and generates control signals to the circuit(s) around it.

Keywords - ControlXML protocol, (MDE) Model Driven Engineering, Meta-modeling, Control framework, Circuit Control standardization.

I. INTRODUCTION

We propose a new protocol "ControlXML". This protocol permits to standardize the control domain of remote circuit(s) through Internet .We can say that our new protocol is similar to the following protocols: VoiceXML which represents the ability of transferring the sound as text through Internet, WebServices which is a protocol for calling far methods through Internet and SVG which is concerning to transfer of images as texts through Internet. The common base between all theses protocols and mine is the ability to send the information as XML files through Internet. And in the receiver side, there is a program that can understand the file logical structure.

Indeed, we use MDE (Model Driven Engineering) approach to realize our protocol. MDE is a new application development approach aims to automate the use of models in order to build systems. In our work, we have used meta-models to separate between two basic domains: the circuit specification domain and the circuit programming one. The circuit specification domain includes the concepts permitting the definition of the circuit's commands and states. The circuit programming domain permits to apply control instructions (if, while, for and delay) on circuit's state and commands in order to define the circuit control program.

Actually, we give our protocol users two graphical tools. The first tool allows the circuit designer to determine his/her circuit's capabilities and the another one allows the control circuit programmer to load a control circuit specification (commands and states) in order to build a control program specific to this circuit. This control program binds the circuit's commands and states with the control flow concepts (if, for, while and delay) using binding concepts (condition, and, or…etc). The circuit control program can be sent to a far computer deals with it.

Our paper will present in section 2 our objectives and approach. Section 3 explains the framework implementing our approach. Section 4 gives a case study to show how to use our control framework. Section 5 gives some conclusions and future visions.

II. OBJECTIVES AND APPROACH

We can easily remark that the circuit programmer needs to know about circuit's capability to use it in order to solve his/her problems. Also, the circuit designer is the responsible for the circuit capabilities specification and he/she will not necessary being the end user. So, we suggest the separating between the circuit specification domain and the circuit programming domain. From a control viewpoint, the circuit specification contains the definition of circuit

commands and states. In addition, the circuit program is a sequence of circuit commands.

Furthermore, we think to use models in order to organize the development of a circuit program. Models are abstractions of systems. High levels of abstraction allow easier understanding and handling of systems. Moreover, new approaches like the MDE (Model Driven Engineering) [3] aim to automate the use of models. MDE encourages the identification and the separation between the different system concerns. MDE structure the application development in several models and model transformations. In MDE framework, we specify a meta-model for each system aspect. So, we can design separately the system different concerns through the definition of system models [4]. After that, the MDE framework builds the system through the model transformations [6]. These transformations permit the integration of system different concerns. The main idea of MDE approach is to use the models at different levels of abstraction. After that, the MDE process specifies a sequence of models and defines how to go starting from a model to another model [5]. Briefly, MDE allows us to specify a methodology for defining problems and how to go towards solutions. Moreover, MDE allows us to capitalize the problem specifications. The problem specifications are the models used in MDE process. Also, MDE allows us to capitalize the know-how to go from the specification to the solution. The know-how specifications are the model transformations used in MDE process. The model transformation defines clearly the rules that permit to go from model to another.

We provide a model-driven framework to realize our protocol. Figure 1 shows this framework's three meta-models and their relations. The first meta-model specifies the circuit domain (states, commands …), the second specifies the control instructions which can be applied (if ,for….) in order to build a circuit command sequence and the third binds between the circuit concepts and the control concepts.

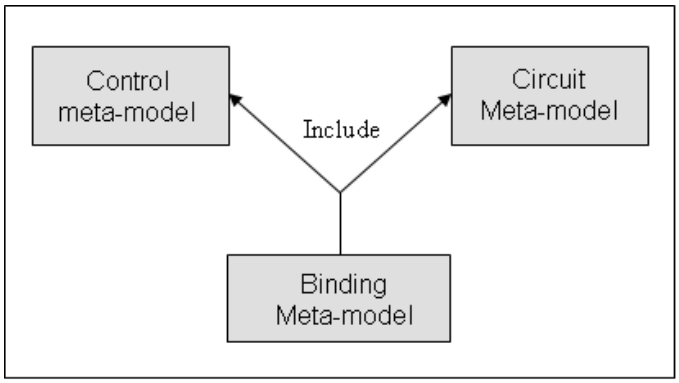

Figure 1: FRAMEWORK META-MODELS

III. FRAMEWORK IMPLEMENTATION

In order to implement the already presented framework: we have to specify the three meta-models presented in the figure 1 (i.e. their concepts and relations). Also, we have supported these meta-models by graphical tools. We have used the framework presented in [1] [2] in order to produce these tools. This framework [1] [2] produces graphical modeling tools starting from meta-models. These graphical tools allow the users to define models conform to the meta-models used to produce them. In the following sub-sections, we will present in details the three meta-models of our control framework and the produced graphical tools.

A. Circuit meta-model

Figure 2 shows the circuit meta-model. Up to now, we just need to know all possible circuit states and acceptable commands. These concepts enable the circuit producer to define the model that specifies his/her circuit capabilities.

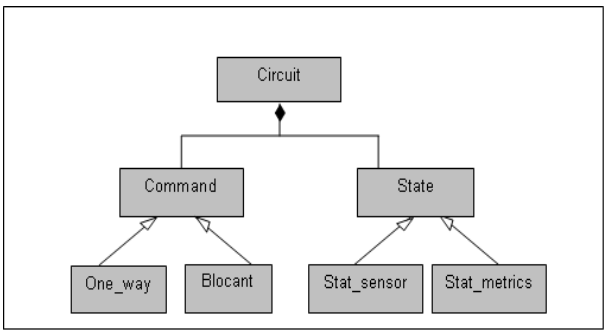

Figure 2: CIRCUIT META-MODEL CONCEPTS

The circuit producer can use instances of the concept "Command" in his/her circuit model in order to define his/her circuit acceptable commands. We think that we can determine for each circuit command: the binary output, a brief description and constraints. There are two special types of commands: "One_way" command and "Blocant" command. We suppose that we can send a command to the circuit after "Blocant" command if the circuit enters in a defined state. The "Blocant" command defines this state. The "One_way" command can impose a delay that we will wait its end in order to send another command toward the circuit. Such, the role of circuit programmer is simplified because he/she is liberated from dealing with all these details. In fact, he/she is concerned by the command sequence and the virtual machine is concerned by these details and when to send a command after another.

The circuit producer can use instances of the concept "State" in his/her circuit model in order to define his/her circuit states. We think that we can determine for each state: the special values and their meaning. There are two special types of states: "Stat_sensor" state and

"Stat_metrics" state. We can use instances of the "Stat_sensor" concept to describe the circuit environment states as the temperature, the pressure, etc. Instances of the "Stat_metrics" concept can be used in order to describe the devices states (e.g. voltage, etc.).

Until now, we focus on the general organization of our framework. After several real experiences using this framework, we can strictly specify all the concepts necessary to define the circuit models.

Starting from this circuit meta-model, we have automatically produced the graphical tool presented in Figure 3. This tool is useful for the "***circuits producer***", it allows him to specify his circuit(s) in a standard way.

A model defined using this tool describes the states and commands of a circuit. This model can be exported as XML file. This file will be used by the circuit programmers who use our control framework. These programmers have not to know XML language in order to program the circuit because we have produced another graphic tool that help them as we will see in the sub-section C.

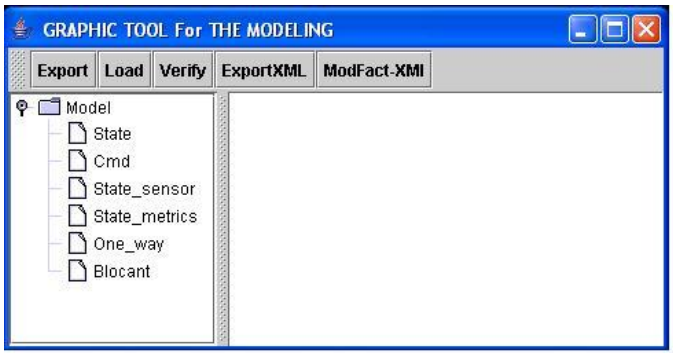

Figure 3: A GRAPHICAL TOOL FOR THE CIRCUIT PRODUCER

B. *Control meta-model*

Figure 4 shows the control meta-model. In this meta-model, we specify the principal concepts of control instructions (*if, for and while*). These concepts are needed by the circuit programmer in order to define the circuit command sequence.

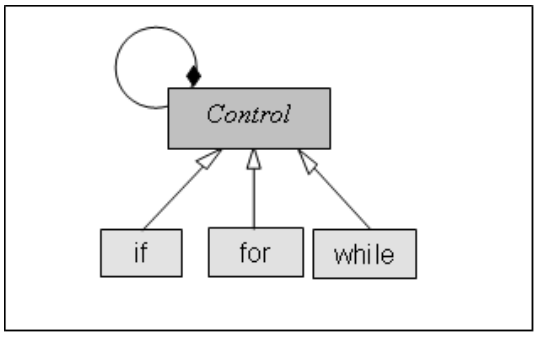

Figure 4: CONTROL META-MODEL CONCEPTS

The Concept "Control" is abstract. We have specified it in the meta-model in order to be able to define nested "control" instances. The meta-model's three principal control concepts are: "if", "for" and "while". These concepts inherit from "Control" concept. An instance of "if" can be used by the circuit programmer in order to choose between two groups of circuit commands according to a circuit state. An instance of "for" is used to repeat a group of circuit commands for "n" times. An instance of "while" can be used by the circuit programmer to repeat a group of circuit commands while a circuit state has a defined value.

We have not produced a graphical tool starting from this meta-model because there is no need to create a control model independently on a specific circuit.

C. *Binding meta-model*

A circuit program model binds between the circuit model elements and the control elements. The relations that enable this binding are specified in the Binding meta-model. Figure 5 shows a part of the Binding meta-model. In this part, we have defined the concept "Logic" that defines a logical relation between two circuit states. There are two type of the "Logic" concept: "Or" and "And" concepts. Also, we add a relation "Logic" between the circuit states. This relation permits to create a complex state from several simple states using instance of the "Logic" concept. Also, we have specified the "Condition" relation between the "Control" concept and the "State" concept. This relation is inherited by the concept "if" because this latter inherits from the "Control" concept. We have defined a relation "Then" between the "if" concept and the "Command" concept. An instance of this relation permits to choose some circuit commands in case the "condition" instance related to an "if" instance has been verified. The "Else" relation between the "if" concept and "Command" concept permits to choose some circuit commands in case the "condition" instance related to the "if" instance has not been verified.

In the same way, we have defined the relations between the control meta-model concepts "while" and "for" and the circuit concepts. Moreover, we have added the "Delay" command that permits to the circuit programmer to separate between two circuit commands by a period of time. This command is added here because it is specific to the circuit programmer and it is not a real circuit command but it is a directive for the virtual machine.

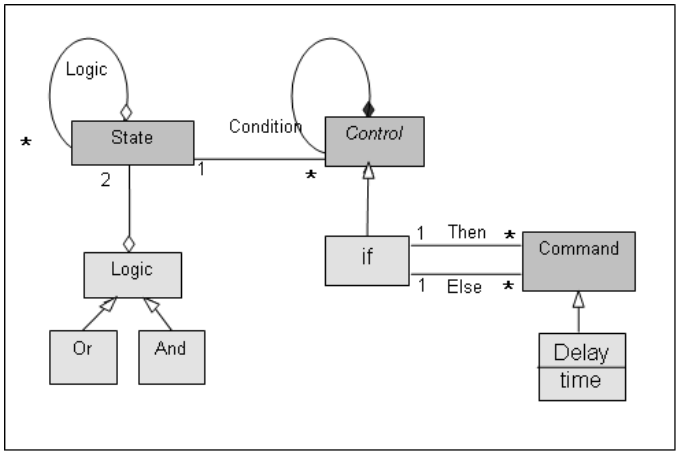

Figure 5: BINDING META-MODEL CONCEPTS

Starting from the binding meta-model we have automatically produced a modeling graphical tool. This graphical tool enables the "***circuit programmer***" to load a circuit model and to create a command sequence specific to the loaded circuit. Figure 6 shows the produced tool.
A model defined using this tool represent a control program model specific to a certain circuit. This model can be exported as XML file. This file will be sent to a far computer related to the circuit. On the far computer, the virtual machine presented in the sub-section D reads this file and sends the appropriate signals to the related circuit.

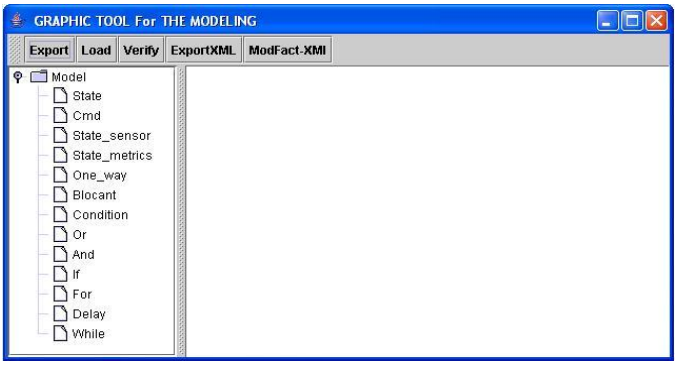

Figure 6: A GRAPHICAL TOOL FOR CIRCUIT PROGRAMMER

In fact, the graphical tool presented in figure 6 assists the model transformation process needed to bind the circuit model and the control model. We would refer here that we were able to produce this binding tool because we have defined the binding concepts as a meta-model.

D. Virtual Machine

The virtual machine is an XML parser that can understand the logical structure of the received XML file. The XML parser wait for a logical structure conforms to the Binding meta-model. So, This file represents a "circuit control program". This representation is independent from all technologies, plate-forms and programming languages. Our virtual machine transforms this program to control signals and sends them to the control circuits. Finally, we must refer that the virtual machine programming is not limited with a specific programming language and it can be supported by mobiles, personal computers and so on.

IV. EXPERIMENT

In order to explain how to use our control framework, we present here a virtual case study. We want developing a program to control a car glass wiper circuit.

As it shown in figure 7 our circuit simply consist of step Motor attached to a bar ends with a simple light emitter and two sensors S, S2.
When sensor faces the light it is triggered and outs (1) while the second sensor outs (0).

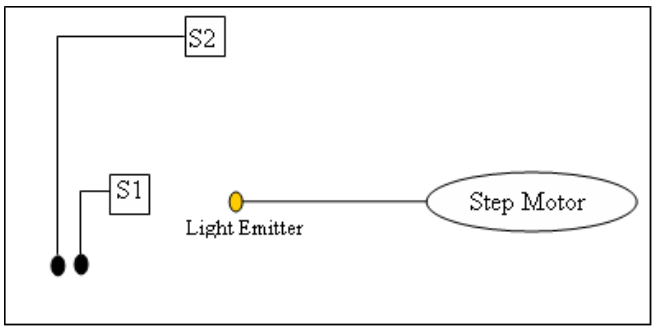

Figure 7: A CAR GLASS WIPER CIRCUIT

We assume that the circuit provides the next four commands:

- Start.
- Move_Right : let bar move toward right.
- Move_Left : let bar move toward left.
- Stop.

The circuit producer can use the circuit editor tool in order to specify this circuit capability as it is shown in Figure 8. After that, he/she can export this specification as an XML file and distribute this XML file to the persons who want programmer this circuit. A simplified format of this circuit specification is shown in the figure 9.

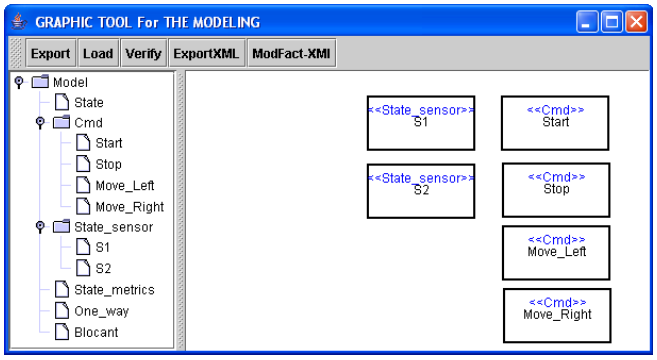

Figure 8: CIRCUIT SPECIFICATION USING THE CIRCUIT EDITOR

```
<?xml version="1.0" ?>
<model metamodel="Circuit" version="2.3.0">
  <Cmd id="Cmd:1202589410177" />
  <Cmd id="Cmd:1202589410277" />
  <Cmd id="Cmd:1202589410377" />
  <Cmd id="Cmd:1202589410477" />
  <State_sensor id="State_sensor:1202589410578" />
  <State_sensor id="State_sensor:1202589410678" />
</model>
```

Figure 9: CIRCUIT SPECIFICATION AS XML FILE

Let's suppose that one wants to use our control framework in order to control this circuit. So, he/she must load the figure 9 circuit specification in the circuit programmer editor. After that, he/she uses the circuit programmer editor to specify graphically his/her circuit control program. Then, he/she exports his/her circuit control program as XML file. This file will be loaded by a virtual machine connected to the glass wiper circuit. The virtual machine transforms this XML file into control signals towards the glass wiper circuit.

V. CONCLUSION AND PERSPECTIVES

This paper presents a proposal for a new protocol "**ControlXML**" to control the different devices through Internet. Moreover, this protocol will be used as MDD (Model Driven Development). So, we have provided a framework that includes several models and model transformation process. Furthermore, we support this framework by graphical tools necessary to define the needed models and a graphical tool to bind the control models and the circuit models, this configuration give us a standard way in building circuits and control of them.

We used virtual machine which is a program written by Java language (or any other languages). This program understands the logical structure of XML file and transforms it to control signals whose send toward the circuits.

From this framework, we attend to build programs to control complex systems as robots. The robot contains several control circuits that must work in a synchronized way. So, we must be able to describe the synchronization information between the combined circuits. Then, we have to specify synchronization concepts in the binding meta-model. Such, one can change rapidly the robot program and send the new program to the robot through Internet.

VI. ACKNOWLEDGEMENTS

Finally, we would like to thank Dr. Malek ALI for his help that facilitates our work and that allows us to begin this research in the Albaath University.

REFERENCES

[1] Bassem Kosayba, "A framework for Model Driven Production of Graphic Modeling Tools", IEEE ICCTA, Damascus, Syria, April 2006.

[2] Bassem Kosayba, Raphael Marvie, Jean-Mark Geib, "Model Driven Production of Domain-Specific Modeling Tools", In 4th OOPSLA Workshop on Domain-Specific Modeling (DSM'04), Vancouver, Canada, October 2004.

[3] S. Kent, "Model Driven Engineering", Third International Conference on Integrated Formal Methods – 2002.

[4] Jean BEZIVIN. "On the Unification Power ofModels. Software and System Modeling", 4(2) :171–188, 2005. http://www.sciences.univ-nantes.fr/lina/atl/www/papers/OnTheUnificationPowerOfModels.pdf.

[5] Marten J. VAN SINDEREN Giancarlo GUIZZARDI, Luis Ferreira PIRES. "On the role of Domain Ontologies in the design of Domain-Specific Visual Modeling Languages". In The Second Workshop on Domain-Specific Visual Languages at OOPSLA, Seattle, WA, USA, November 2002. http://www.cis.uab.edu/info/OOPSLA-DSVL2/Papers/Guizzardi.pdf.

[6] Jean BEZIVIN. "From Object Composition to Model Transformation with theMDA". In TOOLS'USA, Volume IEEE TOOLS-39, Santa Barbara, August 2001. http://www.sciences.univ-nantes.fr/info/lrsg/Recherche/mda/TOOLS.USA.pdf.