# Model Driven Production of User Interactions For Web Applications

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

**Abdel Azziz Backkour**
5th year Student – Albaath University, Syria

*Abstract* - **A form-based service is one in which the flow of data between service (business logic) and user interface is described by a sequence of query/response interactions (forms in HTML), where the interaction (form in HTML) provides a user interface that presents service data to the user (such as a list of foods) then collects information from a user (the selected food) and passes it to the service.**

**Here we are going to provide a simple framework to develop form-based services in a device-independent manner. In this framework we focus on abstraction of the form (user interaction) in a way allowing us to separate the service logic from user interface description.**

**In order to realize our goal we present a "model-driven framework". We determine two types of information. Information describes the specification of the user interaction (form) and information describes the method of using these forms to develop the whole application.**

**To achieve that we produce two tools. The first tool allows the user interface designer to specify abstractly the form elements. The second one allows the application developer to load form specifications in order to build a service logic program specific to these forms. After binding the outputs of these tools we will have an application abstract model. Starting from this model, we can generate the whole application. This application could be a VoiceXml application, a traditional web application, a WAP application or others.**

**Keywords – Abstract User Interface, (MDE) Model Driven Engineering, Meta-modeling, Web Application Framework.**

## I. INTRODUCTION

In this paper, we develop a framework to standardize the web development domain independently of technologies or devices. For example, developers can write a single server-side program that can be used to display stock quotes on the web or read-back stock quotes over the phone. This program would use the same business logic code for getting the quotes, but would have code for two user interfaces one for speech and one for HTML. Also, the developed application can be oriented to traditional desktop browsers or to hand-held devices.

Indeed, we use MDE (Model Driven Engineering) approach to realize our framework. 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 interaction design specification domain and the service logic domain. The interaction specification domain includes the concepts permitting the definition of the form elements and their attributes. The service logic domain permits to apply control instructions (if, while, GetField, SetLabel, SetField etc.) on interactions in order to define the application logic.

Actually, we give our framework users two graphical tools. The first tool allows the user interaction designer to determine his/her user interfaces elements and the another one allows the application programmer to load the interaction specifications in order to build an application specific to these interactions. This control program binds the different interaction attributes with the control flow concepts (if, while, etc.) using binding concepts (condition, and, getField, setField, setLabel … etc).

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 framework. Section 5 gives some conclusions and future visions.

## II. OBJECTIVES AND APPROACH

We can easily remark that the application programmer needs to know about interaction specifications in order to build his/her application. Also, the interaction designer will not necessary being the end user and anyone else can reuse these interactions (forms). So, we suggest the separating between the interaction specification domain and the logical service programming domain.

Furthermore, we think to use models in order to organize the development of the web application. 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.

Figure 1 shows this framework's three meta-models and their relations. The first meta-model specifies the interaction domain (form elements …), the second specifies the control instructions which can be applied in order to build a user interaction sequence and the third binds between the interaction concepts and the control concepts.

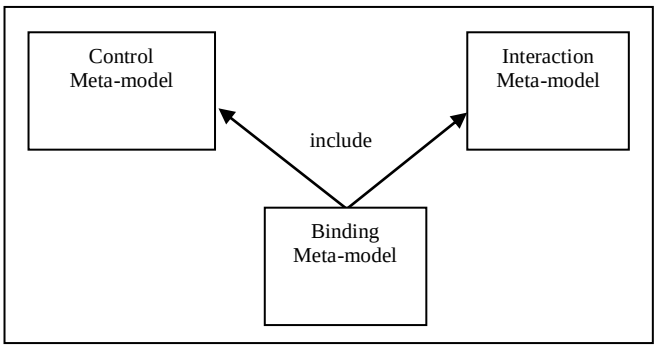


Figure 1: FRAMEWORK META-MODELS

## 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. Interaction meta-model*

Figure 2 shows the interaction meta-model. Up to now, we just need to know the interaction elements. These concepts enable the interaction designer to define the model that specifies an interaction with users.

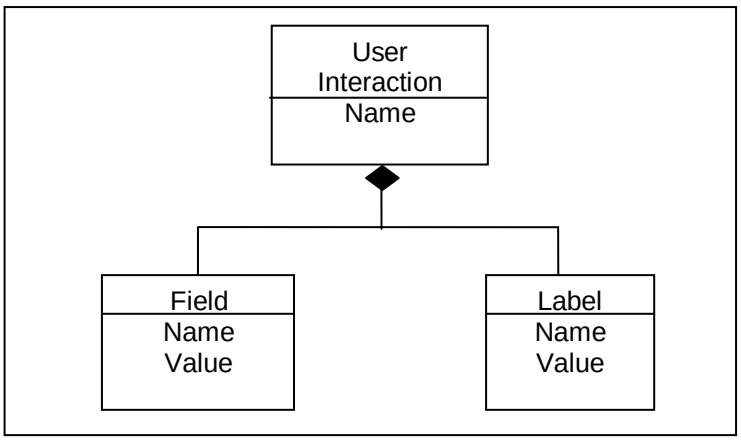


Figure 2: PAGE META-MODEL CONCEPTS

The interaction producer can use instances of the concept "Field" in his/her interaction model in order to take information from the user. The interaction producer can use instances of the concept "Label" in order to give the user some information.

We can specify several types of "Field" as "CheckButton", "RadioButton" etc. Also, we can specify several types of "Label" as "Image", "Sound", "Video" 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 interaction models as the element position, color, etc.

Starting from this interaction meta-model, we have automatically produced the graphical tool presented in Figure 3. This tool is useful for the "***interaction producer***", it allows him/her to specify user interaction(s) in a standard way. We have used the framework presented in [1] [2] to produce such tools. This framework starts from two models: the first describes the application domain and the second describes the desired graphic model. Furthermore, the produced tools are compatible with the MOF (Meta-Object-Facility) [7] and can interact with the repositories produced by the ModFact project [8].

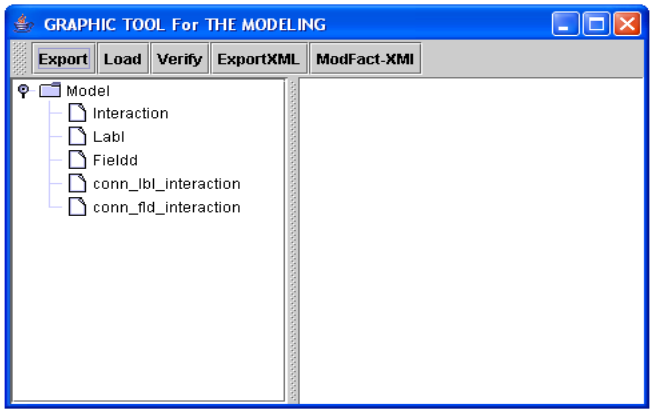


Figure 3: A GRAPHICAL TOOL FOR THE USER INTERACTION PRODUCER

A model defined using this tool describes the elements of an interaction. This model can be exported as XML file. This file will be used by the application programmers who use our web framework. These programmers have not to know XML language in order to develop the application because we have produced another graphic tool that help them as we will see in the sub-section C.

### B. *Control meta-model*

Figure 4 shows the control meta-model. In this meta-model, we specify the principal concepts of control instructions (*if, while, getFeild, setField, setLabel etc*.). These concepts are needed by the application programmer in order to define the user interaction sequence.

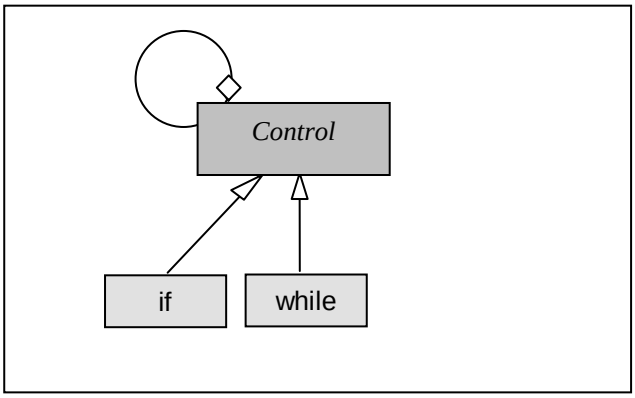


Figure 4: CONTROL META-MODEL CONCEPTS

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 two principal control concepts are: "if" and "while". These concepts inherit from "Control" concept. An instance of "if" can be used by the application programmer in order to choose between two user interactions according to the given response in a pervious user interaction. An instance of "while" can be used by the application programmer to repeat a user interaction until the user give a predefined value.

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

### C. *Binding meta-model*

A web application model binds between the user interaction 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 interaction fields. There are two type of the "Logic" concept: "Or" and "And" concepts. Also, we add a relation "Logic" between the interaction fields. This relation permits to create a complex field from several simple fields using instance of the "Logic" concept. Also, we have specified the "Condition" relation between the "Control" concept and the "Field" concept. This relation is inherited by the concept "if" because this latter inherits from the "Control" concept. We have defined an attribute of type "Then" for the concept "if". An instance of "Then" permits to choose an instance of "Interaction" and several instances of "InteractionCommand" in case the "condition" instance related to an "if" instance has been verified. We have defined in the same way the "Else" attribute for the "if" concept. This attribute can be used in case the "condition" instance related to the "if" has not been verified, but we did not show it in the figure 5 to still clear. The "Interaction Command" concept permits to specify actions that can be applied on the 'interaction' instance chosed in an instance of "Then" or "Else". In the same way, we can defined the relations between the control meta-model concept "while" and the interaction meta-model.

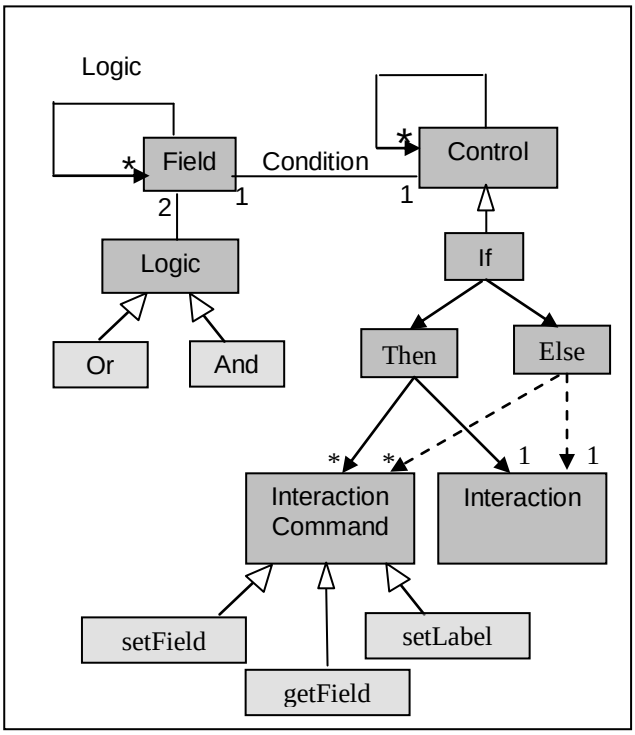


Figure 5: BINDING META-MODEL CONCEPTS

Starting from the binding meta-model we have automatically produced a modeling graphical tool. This graphical tool enables the "***application programmer***" to load several user interaction models and to create a user interaction sequence. Figure 6 shows the produced tool.

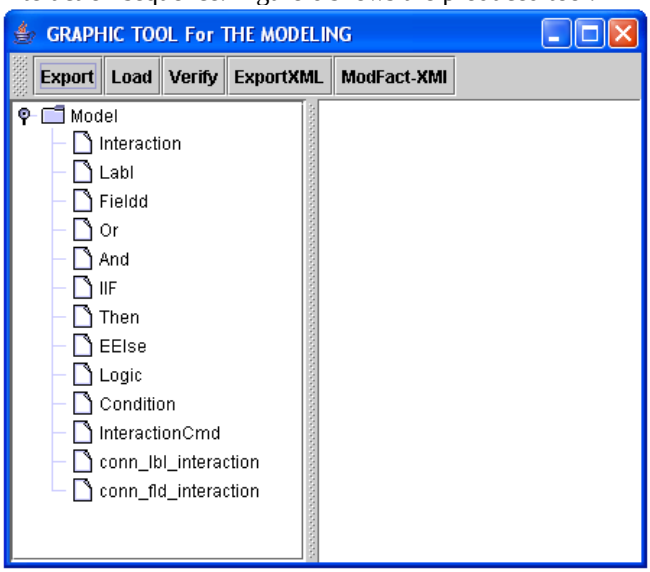


Figure 6: A GRAPHICAL TOOL FOR APPLICATION PROGRAMMER

A model defined using this tool represent an application program model specific to several user interactions. This model can be exported as XML file. The content of this file is abstract (i.e. independent of execution technologies or devices).

In fact, the graphical tool presented in figure 6 assists the model transformation process needed to bind the interactions models 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. Application Code Generator*

We have developed a template for each concept of the interaction meta-model. Also, we have developed several groups of templates. Each group of templates accords for each execution technology (traditional WEB application, WAP application, voiceXML application, etc.) supported by the code generator. The code generator understands the logical structure of the application model. It uses this model information to configure and connect the templates according to the application model and for a specific technology.

## IV. EXPERIMENT

In order to explain how to use our control framework, we present here a common case study. This case study implies two interactions : Login and Welcome.

Simply we have two interactions the first one called "Login". It consists of two labels: "username" & "password", and two fields: "nameFiled" & "passwordField".
The second interaction called "Welcome" it consists of two labels: the value of the first is "Hello" & the value of the second is the username entered by the user in the "Login" interaction.

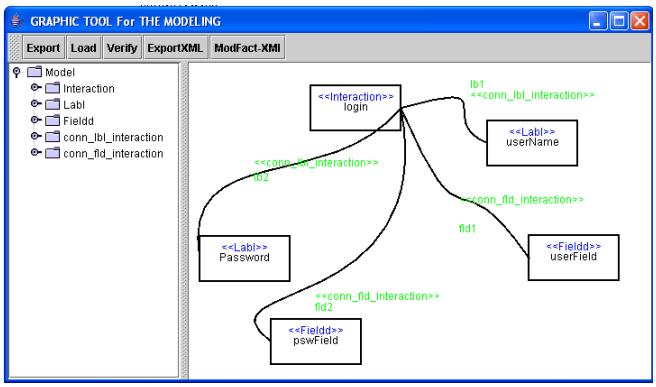


Figure 7: A LOGIN INTERACTIOON

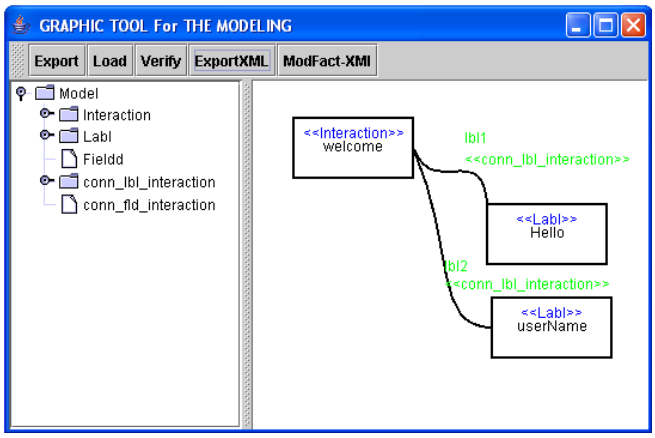


Figure 8: A WELCOME INTERACTIOON

The interaction producer can use the interaction editor tool in order to specify these interaction elements as it is shown in Figure 7 and Figure 8. After that, he/she can export this specification as an XML file and distribute this XML file to the persons who want to program an application using these interactions. Simplified formats of these specifications are shown in the figure 9 and Figure 10.

```
<?xml version="1.0" ?>
- <model metamodel="interaction" version="2.3.0">
    <Interaction id="Interaction:1203485386089" />
    <Labl id="Labl:1203485386189" />
    <Labl id="Labl:1203485386339" />
    <Fieldd id="Fieldd:1203485386440" />
    <Fieldd id="Fieldd:1203485386540" />
  - <conn_lbl_interaction id="conn_lbl_interaction:1203485387131"
      name="lb1">
      <Interaction id="Interaction:1203485386089" />
      <Labl id="Labl:1203485386189" />
    </conn_lbl_interaction>
  - <conn_lbl_interaction id="conn_lbl_interaction:1203485387231"
      name="lb2">
      <Interaction id="Interaction:1203485386089" />
      <Labl id="Labl:1203485386339" />
    </conn_lbl_interaction>
  - <conn_fld_interaction id="conn_fld_interaction:1203485387341"
      name="fld1">
      <Interaction id="Interaction:1203485386089" />
      <Fieldd id="Fieldd:1203485386440" />
    </conn_fld_interaction>
  - <conn_fld_interaction id="conn_fld_interaction:1203485387441"
      name="fld2">
      <Interaction id="Interaction:1203485386089" />
      <Fieldd id="Fieldd:1203485386540" />
    </conn_fld_interaction>
  </model>
```

Figure 9: LOGIN SPECIFICATION USING THE INTERACTION EDITOR

```
<?xml version="1.0" ?>
- <model metamodel="interaction" version="2.3.0">
    <Interaction id="Interaction:1203484657121" />
    <Labl id="Labl:1203484657221" />
    <Labl id="Labl:1203484657321" />
  - <conn_lbl_interaction id="conn_lbl_interaction:1203484657682" name="lbl1">
      <Interaction id="Interaction:1203484657121" />
      <Labl id="Labl:1203484657221" />
    </conn_lbl_interaction>
  - <conn_lbl_interaction id="conn_lbl_interaction:1203484657782" name="lbl2">
      <Interaction id="Interaction:1203484657121" />
      <Labl id="Labl:1203484657321" />
    </conn_lbl_interaction>
  </model>
```

Figure 9: LOGIN SPECIFICATION USING THE INTERACTION EDITOR

Let's suppose that one wants to use our web framework in order to program an authentication web application. So, he/she must load the figure 9 login specification and the figure 10 welcome specification in the application programming editor. After that, he/she uses the application programming editor to specify graphically his/her application program. Then, he/she exports his/her application program as XML file. After that this file will be used to generate the whole application and in the desired technology.

## V. CONCLUSION AND PERSPECTIVES

We use MDE (Model Driven Engineering) approach to organize the definition of form-based applications. We provide a framework that includes several models and 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 interaction models, this configuration give us a standard way in building abstract web application models. These models are independent of execution technologies and devices.

We used a code generator in order to produce the application files. The code generator understands the logical structure of the application model. It uses this model information to configure and connect the pre-built templates according to the desired technology.

## 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.

[7] OMG (Object Management Group), http ://www.omg.org/. ”Meta Object Facility (MOF) Specification”, March 2000. http://www.omg.org/cgi-bin/apps/doc?formal/01-11-02.pdf.

[8] “ModFact Project”. http://modfact.lip6.fr