International Journal of Computer Sciences and Engineering
Vol.**11**, Issue.**8**, pp.**01-08**, August **2023**
ISSN: 2347-2693 (Online)
Available online at: www.ijcseonline.org

ISSN: 2347-2693 (E)

Research Paper

An Approach to Build the Ergonomics of Interactive Software based on MDE

Thierry Noulamo[1*], Bernard Fotsing Talla[2], Jean Pierre Lienou[3], Alain Djimeli-Tsajio[4]

[1,2,4]Dept. of Computer Engineering/IUT Fotso Victor of Bandjoun, University of Dschang, Bandjoun, Cameroon
[3]Dept. of Computer Engineering, College of Technology/ University of Bamenda, Bamenda, Cameroon

**Corresponding Author: thierry.noulamo@gmail.com*

Received: 24/Jun/2023; **Accepted**: 26/Jul/2023; **Published**: 31/Aug/2023. **DOI**: https://doi.org/10.26438/ijcse/v11i8.18

Abstract: The design and implementation of interactive systems and Human-Machine Interfaces (HMI) use different techniques from both software engineering and ergonomics. To improve the productivity and quality of software, automating the development process is an important factor. User interfaces are nonfunctional but complex software components that play a vital role in the development of interactive applications. We propose in this paper an approach for the automatic production of Human-Machine Interfaces (HMI) for the development of interactive applications according to the Model-Driven Engineering (MDE) approach. A source Meta-Model called "DD_IHM" ("Description Diagram for Human-Machine Interfaces"), a target Meta-Model specific to the PHP language called "CGFP" (Context Grammar for PEAR) for the construction of HMIs and, a set of generic rules for transforming a model conforms to the source meta-Model into a model conforms to the target Meta-Model, written in the QVT language are develop. We apply this approach to the creation of a simple online registration platform.

Keywords: Interactive Systems, Ergonomie, Models Transformation, Context-free Grammar, QVT Language, Software productivity.

1. Introduction

According to the ISO 9126 standard, productivity is defined as “the ability of software products to enable users to expend an appropriate number of resources in relation to the efficiency achieved in a given specific context” [1], [2]. Nowadays, the implementation of quality software requires attention to both the process used for its development and its productivity [3][4][5].

An approach to improve productivity in software engineering is presented in this paper. We focus on the HMI layer of the multi-layered software architectures. The visual models and standardized notations are generally necessary for its development. The use of a multilayer architecture promotes the reusability and maintainability of the system [6]. The three-layer architecture is used: the presentation layer or external layer, corresponds to the implementation of HMI; the business layer or functional layer, corresponds to the construction of the needs of the system, and the data layer or internal layer.

We are interested in this paper by the presentation layer. As with all other phases of the system development process, the complexity of modern systems user interface requires good modeling technique and methodology. The classic approaches used in the literature are either task-based or model-based. According to Schlungbaum [7], the knowledge used in the development of interactive systems can be represented by models, hence our choice for the model-based approach. The realization of an operational model of HMI will retain our attention.

Many development tools integrate so-called interface management systems (UIMS), to facilitate the implementation of HMIs by non-IT specialists and also to improve productivity [8]. However, these tools are for the most part proprietary and cannot be used in a complete MDE process.

The MDE [9][10] which promotes the system model as the main elements of development, is today used as a basic approach for the implementation of complex systems [9], [11], [12], [13], [14]. However, the developer is constrained to use operational models so that an implementation can be systematically generated.

We propose in this work an approach to produce a Meta-Model for the design of operational HMI models. We enrich the design elements proposed in [15][16] with new concepts. The authors in [16] have proposed an approach based on two levels of transformation.

The UML interaction diagram obtained is translated to a model conforms to the meta-model specific to PHP. The translation rules implemented in the QVT language use a single level of transformation. The work is structured in seven sections.

We present in section 2 some works of the literature which focused on HMI modeling for the development of interactive systems. Our development approach is presented in section 3. The proposals made in terms of the source Meta-Model, the target Meta-Model and the QVT implementation of the transformation rules are given in section 4. Through an example, we show in section 5, the results from the automatic generation of a simple online registration system interface. In section 6, we present the deployment model of our system, highlighting the components necessary for the implementation of the GUI according to our approach. We end our study with section 7 which concludes and gives some future work.

2. Related Work

Nowadays, interacting with computer systems is part of our daily live. If such systems are well designed they will be very useful. Many tools in the literature facilitate the design of user needs concerning Human-Machine Interaction, we can cite: HTA (Hierarchical Task Analysis) [17], AMD (Analytical Method of Description) [18], UAN [19] extended in XUAN [20] and the "ConcurTaskTrees" notation (CTT1 [21]). The tools presented above are the starting point for the implementation of HMI. Although they are mostly user-centric, the computer models produced are not operational. The final system presents an architecture with several layers, the most prominent are: the interface layer, the business layer and the data layer. There are many models in the literature for HMI development: Seeheim [22], Arch [23] are used in layered models, MVC (Model, View, Controller), PAC (Presentation, Abstraction Control) [6] , [24], [25], [26] are used in agent models and the mixed models that try to exploit the advantages of the previous categories. We focus on "agent models" in this article. They all have in common the interface between the user and the system. This interface is called "view" in the MVC model and "Presentation" in the PAC model. However, no specification is made on the details of this layer.

In [27], the authors present a technique which makes it possible to generate adaptable user interfaces, starting from the specification of the task at the time of the execution by taking into account the context of use. The designer specifies a task template using the "ConcurTaskTrees" notation and its context-dependent parts and generates the UI.

In [28], the authors present a design technique and architecture of a Personal Universal Controller (PUC) which is a system for automatically generating high quality multimodal interfaces for remote control of complex devices. The system includes a communication protocol, adapters for translating proprietary device protocols to the PUC protocol, a specification language for describing functions, and a generator that automatically creates interfaces from specifications.

The approaches presented here cannot be used in an MDE process because the generation module produces a result that does not conform to any target meta-model.

In [16], the authors proposed an approach based on a two-level transformation, to automate the development of user interfaces usable in MDE.

The objective in this work is to automate the development of user interfaces usable in MDE. The resulting interaction diagram is translated into a PHP-specific Meta-Model, and using a single generic model transformation rule and is implemented in the QVT language. This overall objective is broken down into several specific objectives. The proposal of a "DD_IHM" source meta-model, whose elements are designed by extension of the UML meta-model. The proposal of a target meta-model called "CGFP" which is specific to the PHP language. The implementation of the transforming rules to a model conform to the source meta-model to a model conform to the target meta-model.

3. A One-level transformation approach

We present in Figure 1 our design approach, based on MDE. It uses a one-level transformation.

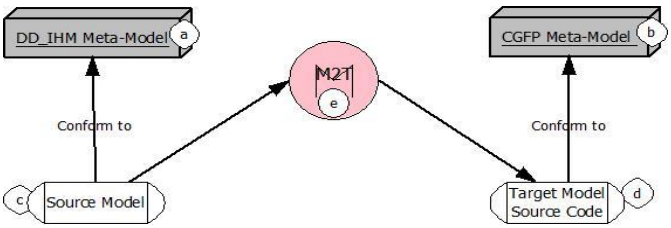


Figure 1. One-level transformation approach

We build the source model (c) using the elements of the "DD-IHM" meta-model (a). The engine (e) performs a model to text transformation. Its input is a model of the "DD-IHM" meta-model and its output is the model (d) conform to the target meta-model "CGFP" (b).

4. Materials and Method

In this section, we use the Unified Modeling Language (UML) [29], [30], [31], which has established itself as a standard for modeling systems according to the object-oriented approach. The elements of this modeling language allow us to represent the different meta-classes constituting the elements of our source Meta-Model. The target Meta-Model is also defined as a concrete syntax of the context-free grammar. A generic specification of transformation rules is given in to QVT language.

The "DD-IHM" Meta-Model

We propose in this section the elements of modeling of the meta-model "DD_IHM". It covers concepts for modeling Human-Machine Interfaces.

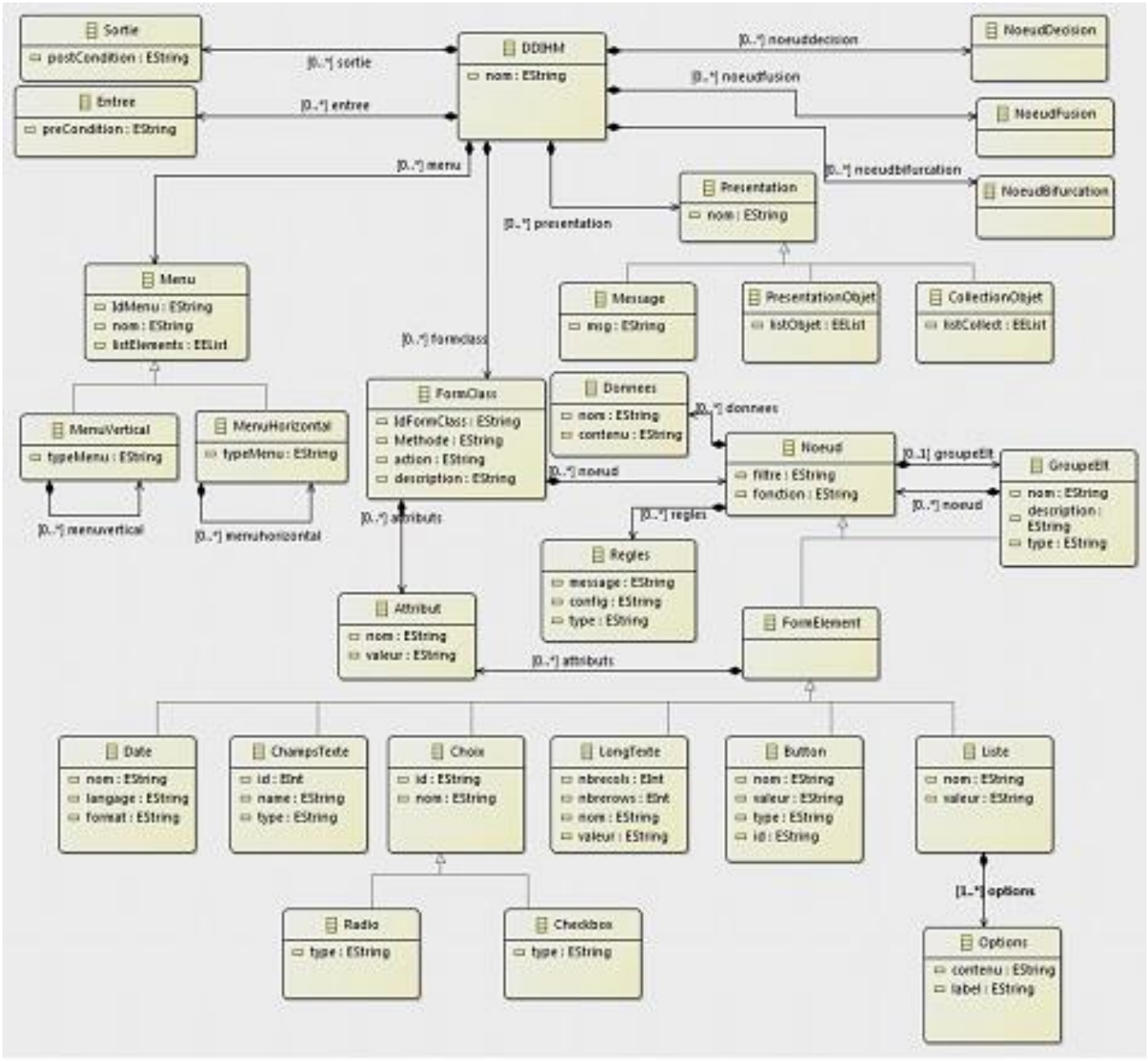
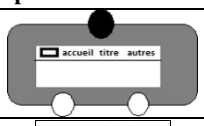
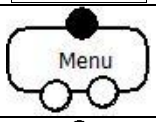
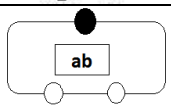
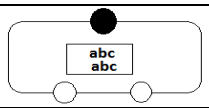
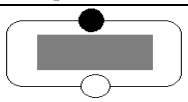
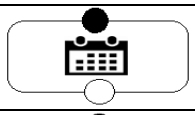
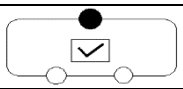
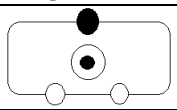
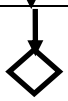
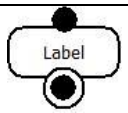


Figure 2. DD-IHM Meta-Model

Class, activity and "state machine" diagrams of UML[23] are used for the construction of new design elements. Table 1 presents the UML representation of the different modeling elements of our meta-model. The main elements that can be used for the construction of GUIs are given in figure 2. The form is modeled using the "FormClass" element which has two subclasses, "Node" and "Attribute". Three subclasses make up the “Node” element: the “Data”, “Rules” and “groupELT” subclasses. Form elements are labeled using the “Data” class. It is similar to the labels that exist in the HTML language. Constraint modeling is done using the “Rule” class, which are applied to form components. Each node is composed of at most one element group. Several subgroups make up a group. The "Menu" object is used to build the navigation components in the application. The menu has properties allowing to define its nature which can be vertical or horizontal. The class diagram is the basis of this proposal. With each node of the Meta-Model, we associate communication ports (input and output) to obtain a new elements call "Dynamic Object" which is an instance of a class of the basic diagram of figure 2. The composition of the "Dynamic Objects" by using the nodes of "activities diagram" included in the Meta-Model, gives us a new diagram called “Dynamic Objects Diagram” (DoD). The "DoD" makes it possible to model an HMI by connecting by carefully selecting dynamic objects’ inputs and outputs. An HMI is considered as a composite "DoD", using other "DoDs" as states of a UML statechart diagram. The DTD that validates the final model storage XML file is given in Listing 1 below. The elements of our source meta-model are presented in table 1 in the form of elementary "DoD"..

TABLE 1 HMI elements of the source meta-model

Element's names	Representation	Semantics
Form_Class		Create an encapsulation Elements
Group_Elt		Create an Element Group
Menu Node		Create a Navigation Menu
Field_Text		Create a Text Field
Long_Field_Text		Create a Long Text Field
Bouton		Create a Button Element
Date		Create a Date Element
CheckBox		Create a CheckBox Element
Radio		Create a Radio Field Element
Initial_Node		Begin Diagram Node , (UML Element)
End_Node		End Diagram Node , (UML Element)
Transition_Node		Transition Node, (UML Elemen)t
Test_Node		Bridge between two or more elements, (UML Element)
Synchronization_ Node		Synchronazation of form elements, (UML Element)
Label Node		Create a Label Element
Condition Node	**[Cond]**	Add a Condition to a Decision Nodes , (UML Element)

The CGFP Meta-Model

We present in this section the "CGFP" meta-model used as target meta-model. "CGFP" allows to produce an abstract representation of the QuickForm HTML code for the generation of an HMI in PHP. This description is given in the form of productions of a context-free grammar (see Listing 2). Note that a context-free grammar consists of a tuple G = (V, T, P, S) where:

- "V" is a finite set of variables also called non-terminals;
- "T" is a finite set of terminals;
- "P" is a set of production rules;
- "S" is the start symbol of the language being defined.

The transformation rules

We use in this work the QVT Language (Query/View/Transformation) more specifically the QVT-Operational model defined as a transformation standard by the Object Management Group (OMG) [32]. It act on a source models conforming to MOF meta-models to build target models. QVT-Operational is an imperative language designed for writing one-way transformations. The transformation process begins with the declaration of the source "DD_HMI" and target "CGFP" Meta-Models with the following instructions:

- modeltype DD_IHM uses "http://localhost/hmi2";
- modeltype CGFP uses "http://localhost/htmlQF.hs";

The structure of the transformation rules is given as follow:

Listing 1. Extract of the DTD

```
<?xml version="1.0"? encoding="ISO-8859-1">
<!DOCTYPE DoD_IHM_DIAGRAM[
<!ELEMENT FormNode(BeginNode, labelNode (Node)+, EndNode>
<!ATTLIST DoD_Form Name CDATA #REQUIRED Method CDATA #REQUIRED Action CDATA #REQUIRED>
<!ELEMENT BeginNode EMPTY>
<!ATTLIST BeginNode ID_Elements ID Next_node IDREF #REQUIRED>
<!ELEMENT LabelNode EMPTY>
<!ATTLIST LabelNode ID_Elements ID ID_Next IDREF #REQUIRED>
<!ELEMENT Node (StructNode| FormElem>
<!ELEMENT EndNode EMPTY>
<!ATTLIST EndNode ID_Elements ID  #REQUIRED>
<!ELEMENT StructNode(BeginNode | EndNode | ForkNode | JunctionNode | TestNode
                                                  | TransitionNode | CondNode)>
<!ELEMENT FormElemNode(LabelNode | Menu | Text | RadioNode | CheckBox | List
                                                  | Button | Group)>
<!ELEMENT ForkNode(FormElemNode)+>
<!ATTLIST ForkNode ID_Elements ID Next_node IDREFS #REQUIRED>
<!ELEMENT JunctionNode EMPTY>
<!ATTLIST JuntionNode ID_Elements ID Next_node IDREFS #REQUIRED>
<!ELEMENT TestNode(CondNode, CondNode+)>
<!ATTLIST TestNode ID_Elements ID Next_node IDREFS #REQUIRED>
<!ELEMENT CondNode(FormElem+)>
<!ATTLIST CondNode ID_Elements ID Next_node IDREF Condition DATA #REQUIRED>
<!ELEMENT TransitionNode EMPTY>
<!ATTLIST TransitionNode ID_Elements ID source_node IDREF target_node IDREF #REQUIRED>
<!ELEMENT MenusNode (BeginNode,(MenuNode+,(StructNode),MenuNode+),End_Node)*)>
<!ATTLIST MenuNode ID_Elements ID Name CDATA Value CDATA parent IDREF #REQUIRED>
<!ELEMENT InputElem (Rules)>
<!ATTLIST InputElem ID_Elements ID  #REQUIRED
        Name CDATA #REQUIRED
        Value CDATA
        Type CDATA #REQUIRED
        Multiligh CDATA
        NumberOfLigne CDATA
        NumberOfCol CDATA
        ID_Warning IDREF #REQUIRED
        ID_Sucess IDREF #REQUIRED>
<!ELEMENT ButtonElem (Rules)>
<!ATTLIST ButtonElem ID_Elements ID  #REQUIRED
        Name CDATA #REQUIRED
        Value CDATA #REQUIRED
        Type CDATA #REQUIRED
        Action CDATA #REQUIRED
        ID_Warning IDREF #REQUIRED
        ID_Sucess IDREF #REQUIRED>
<!ELEMENT RadioElem (Rules)>
<!ATTLIST RadioElem ID_Elements ID  #REQUIRED
        Name CDATA #REQUIRED
        Value CDATA #REQUIRED
        ID_Warning IDREF #REQUIRED
        ID_Sucess IDREF #REQUIRED>
<!ELEMENT CheckBoxElem (Rules)>
<!ATTLIST  CheckKBoxElem ID_Elements ID  #REQUIRED
        Name CDATA #REQUIRED
        Value CDATA #REQUIRED
        ID_Warning IDREF #REQUIRED
        ID_Sucess IDREF #REQUIRED>
<!ELEMENT SelectElem (SelectOption*, Rules*)>
<!ATTLIST  SelectElem ID_Elements ID  #REQUIRED
        Name CDATA #REQUIRED
        ID_Warning IDREF #REQUIRED
        ID_Sucess IDREF #REQUIRED>
<!ELEMENT  SelectOption EMPTY>
<!ATTLIST SelectOption Name CDATA #REQUIRED
        Value CDATA
        ID_Next IDREF #REQUIRED>
<!ELEMENT Rules EMPTY>
<!ATTLIST Rules Name CDATA #REQUIRED
        Value CDATA
        ID_Next IDREF #REQUIRED>
]>
```


Listing 2. The CGFP target meta-model

```
<FormHtmlQF> ::= <HEADER> <FormId> "=new_HTML_QuickForm('" <Name> "','" <Method> "','"
                                    <Action> "');" <FormElement>* <DISPLAY>
<FormElement> ::= <OptionsArray> <FormElt>  <AddElt>  <AddRule>*

<OptionsArray> ::= <OptionId> "=array(" <Option>* ");_"   | <OptionId> "='';"

<Option> ::= "'" <OptionName> "'=>'" <Value> "'"    |   ","

<FormElt> ::= <TextElt> | <RadioElt> | <CheckBoxElt> | <ListELt> | <GroupElt> | <DateElt>
                                                                | <ButtonElt>
<AddElt> ::= <FormId> "->addElement(" <ElementId> ");"

<AddRule> ::= "->addRule('" <Name> "','" <ReturnMsg> "','" <Type> "','"<Validation>" ');"

<TextElt> ::=<ElementId>"=new_HTML_QuickForm_Text('"<Name>"','"<Label>"',"<OptionId>");"
      | <ElementId> "=new_HTML_QuickForm_Textarea('"<Name>"','"<Label>"',"<OptionId>");"

<RadioElt> ::= <ElementId> "=new_HTML_QuickForm_Radio('"<Name>"','" <Label> "','"<Text>
                                                    "','" <Value> "'," <OptionId> ");"

<CheckBoxElt> ::= <ElementId> "=new_HTML_QuickForm_CheckBox('" <Name> "','"<Label>"','"
                                                    <Text> "',"<OptionId>");"

<ListELt> ::=<ElementId>"=new_HTML_QuickForm_Select('"<Name>"','"<Label>"',"<OptionId>");

<DateElt> ::= <ElementId>"=new_HTML_QuickForm_Date('"<Name>"','"<Label>"',"<OptionId>");"

<ButtonElt> ::= <ElementId> "=new"<FctName>"('"<Name>"','"<Label>"',"<OptionId>");"

<FctName> ::= "HTML_QuickForm_Button" | "HTML_QuickForm_Submit" | "HTML_QuickForm_Reset"

<Id> ::= "$" <Char>+

<OptionId> ::= <Id>

<ElementId> ::= <Id>

<FormId> ::= <Id>

<Name> ::= String

<Label> ::= String

<Validation> ::= "server" | "client"

<Type> ::= "required" | "maxlength" | "minlength" | "rangelength"| "email"| "lettersonly"
                    | "alphanumeric" | "numeric" | "nopunctuation"| "nonzero"
<OptionName> ::= String

<Value> ::= String

<HEADER> ::= "include('headerFile.php');"

<DISPLAY> ::= <FormId> "->display();"
```

Listing 3. The transformation rules

```
transformation hmi2htmlQF(in hmi :DD_IHM, out htmlQF :CGFP){
main(){
hmi.objects()[hmi::FormNode]->map frmNode2frmHtmlQF();
hmi.objects()[hmi::TextNode]->map txtNode2txtHtmlQF();
hmi.objects()[hmi::RadioNode]->map radioNode2radioHtmlQF();
hmi.objects()[hmi::CheckBoxNode]->map cbNode2cbHtmlQF();
hmi.objects()[hmi::ListNode] ->map lstNode2lstHtmlQF();
hmi.objects()[hmi::DateNode]->map dateNode2dateHtmlQF();
hmi.objects()[hmi::ButtonNode] ->map btnNode2btnHtmlQF();
}
...
```

Each input "hmi" model of type simpleHMI_MM is transformed into an htmlQF output model of type HtmlQF_MM. The transformation process is triggered by the main function named main().We give below the declaration of an example of transformation rule which allows to apply "frmNode2frmHtmlQF()" to the input model in order to produce the corresponding code in HTML_QuickForm.

hmi.objects()[hmi::FormNode] ->map frmNode2frmHtmlQF();

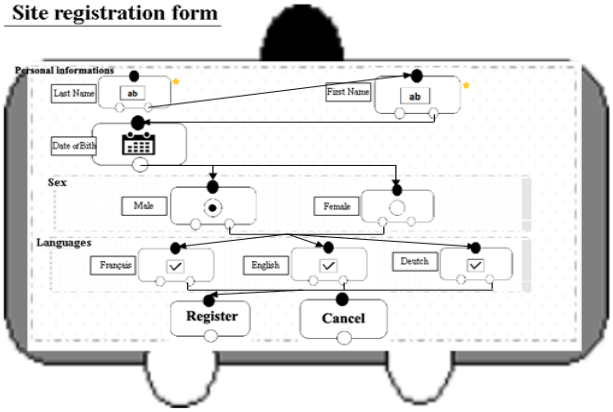


Figure 3. Registration form in our graphical editor

Every mapping is an operation associating an element from the input model with another element from the output model. For example, to transform a FormNode to FormHtmlQF, the corresponding mapping operation is:

Listing 4. The mapping code

```
mapping hmi::FormNode::frmNode2frmHtmlQF() : htmlQF::FormHtmlQF {
FormHtmlQF :="include('headerFile.php');"+self.Name+" id = new
HTML QuickForm ("'+ self.Name+"','"+ self.Method + "','" + self.Action+ "');
  " + hmi::FormNode::frmElements();
}
...
```

5. Results and Discussion

The code below given in Listing 5 is the XML file of an example GUI for an online registration platform.

Listing 5 XML description of the simple online registration platform.

```
1 <?xml version ="1.0 "encoding ="UTF-8"?>
2 <ddihm :DDIHM xmi : version =" 2.0 "
3 xmlns : xmi="http://www. omg.org /XMI"
4 xmlns : xsi ="http://www.w3.org /200 1 /
5 XMLSchema-instance "
6 xmlns : ddihm="http:// ddihm . com"
7 xsi : schemaLocation="http://ddihm . com
8 . . / MetaModel /DDIHM. e c or e ">
9 <form class
10 IdFormCl ass="1" Methode="post "
11 action ="Info Personnelles">
12 <noeud xsi:type =" ddihm : Group eElt "
13 nom="infoper "description="Informations
14 personnelles"type="fieldset">
15 <formelement xsi:type =" ddihm : ChampsTexte "
16 id ="1" name="nom" type ="inputtext">
17 <regles type="required"/>
18 <donnees nom="label" contenu ="Nom"/>
19 </form el ement>
20 <noeud xsi:type ="ddihm : Button "
21 nom="Envoyer" v al u e =" s ubmit "/>
22 <noeud xsi:type ="ddihm:Button "
23 nom="Annuler"
24 valu e="reset"/>
25 </formclass>
26</ddihm :DDIHM>
```

By applying the transformation rules defined above, the HTML_QuickForm code is generated as shown in Listing 4. This code displays as a result the online registration user interface of Figure 4.

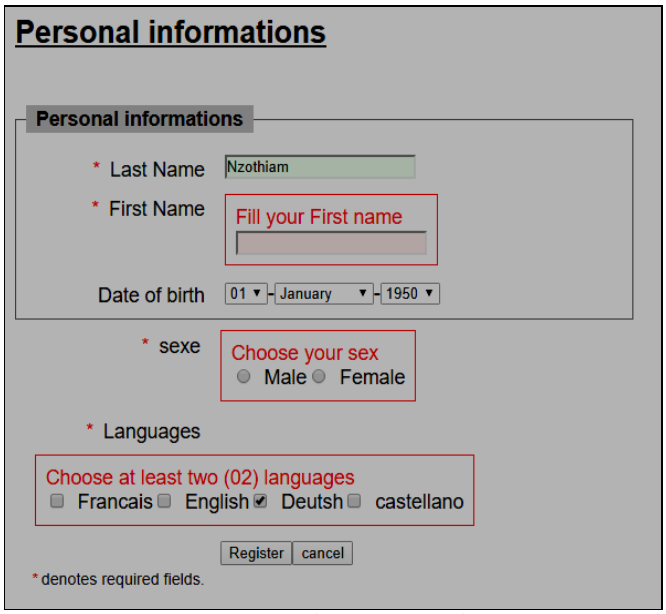


Figure 4. HMI of the registration form

Listing 6. Generated HTML QuickForm of the simple online registration platform

```
1 <?xml version="1.0"encoding="UTF-8"?>
2 <abstractform:Model
3 xmi : version ="2.0"
4 xmlns : xmi="http://www. omg.org /XMI"
5 xmlns : x s i ="http://www. w3.org / 2 0 0 1 / XMLSchema-
instance"
6 xmlns :abstractform ="http: / / www. mdt . com /
abstractform"
7 xsi:schemaLocation="http://www. mdt . com /
abstractformmetamodel / AbstractForm.ecore "
8 Appname="registration">
9 <forms id ="Etape2"description="Infopersonnelles">
10 <nodexsi:type="abstractform:Fieldset"label="
Informationpersonnelles"name="infopers">
11 <nodexsi:type="abstractform:InputText"id="
nom" name="nom">
12 <rulesxsi:type="abstractform:Required"
message="Veuillezentrervotrenom"/>
13 <dataname="label" value ="Nom"/>14 </node>
15 .
16 .
17 .
18 <nodexsi:type="abstractform:InputCheckBox "id="Deutsh " name="espagnol ">
19 <dataname="content " value ="castellano"/>
20 <attributname="value "value="es "/> 21 </node>
22 </node>
23 <nodexsi:type ="abstractform:InputS ubmit "name=" Envoyer "/>
24 <nodexsi:type="abstractform:InputReset" name
=" Annul er "/>
25 </forms>
26</abstractform:Model>
```


6. Deployment model

Figure 5 represents the result of the deployment of our system after installing the packages PEAR and HTML QuickForme2. The "PHP Server" directory is created and contains the Web server and the PHP script engine. In the PEARDIR sub-directory of “Serveur Web”, the PEAR package is installed. PEARDIR contains two sub-directories, the HTML sub-directory where the packages useful for QuickForm2 are deployed and DATA which contain the style sheet references and JavaScript code used by the QuickForm2 package. After transformation of the “DD_IHM” model, the PHP source file obtained is stored in the PEARDIR directory and is accessible via the URL “http://localhost/peardir/currency_registrationform.php”. A file named “head.php”, stored in the HTML sub-directory, contains the definition of the personal data of the result form.

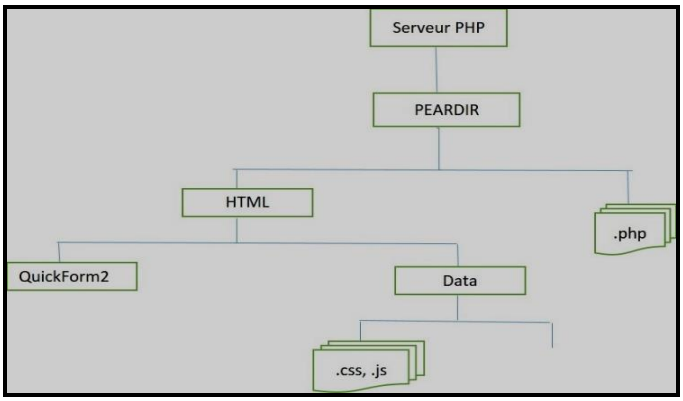


Figure 5. Deployment model

7. Conclusion and Future Scope

In this paper, an approach for the automatic production of the HMI dialogue layer is proposed. The work is part of a broader research perspective aiming to automate the complete process of software production. We presented a source meta-model "DD_IHM" which is an extension of the UML modeling language for the specification of HMI components, a target meta-model for the production of models conforming to the PHP language and finally a set of transformation rules implemented with QVT, allowing to pass from a source model to a target model. The quality of the results obtained in terms of ergonomics, the development time and the maintainability of the final result are factors in the assessment of our proposal. In our future work, we intend to extend this method to all layers of the architecture. For this, an extension of the source and target meta-models and the transformation rules will be necessary in order to take into account all the modeling elements.

Conflict of Interest

The authors declare it unrelated to this research

Funding Source

This work was supported in part by the Unit Research in Control and Applied Computing (URCAC) of the University of Dschang, Cameroon

Authors’ Contributions

Dr Noulamo contributes in the modelling and implementation of source meta-model, writing up of paragraph 2, 3, the structure of the paper and approved its final version.

Dr Fotsing contributes in writing up the transformation rules from source meta-model to target meta-model which was applied.

Dr Lienou contributes in the modeling and implementation of the target meta-model, write up of paragraph 4 and 5, translate and format the document.

Dr Djimeli contributes in implementing the transformation rules, final readings, translation and formatting.

References

[1]. Hernández-López, Adrián, Colomo-Palacios, Ricardo, Et García-Crespo, Ángel, “Productivity in software engineering: A study of its meanings for practitioners: Understanding the concept under their standpoint”, In: 7th Iberian Conference on Information Systems and Technologies (CISTI 2012). IEEE, pp.**1-6, 2012.**

[2]. Abran, Alain, Al-Qutaish, Rafa E., et Cuadrado-Gallego, Juan J. “*Analysis of the ISO 9126 on software product quality evaluation from the metrology and ISO 15939 perspectives*.” WSEAS Transactions on Computers, Vol.**5**, no.**11**, pp.**2778-2786, 2006.**

[3]. Anupriya et al., “*Survey on Various Productivity Measures of Software Development Teams*”, International Journal of Advanced Research in Computer Science and Software Engineering 4(6), pp.**462-464**, June **2014**

[4]. Bindia Tarika, “*Review on Software Analysis & Design Tools*”, International Journal of Computer Sciences and Engineering, Vol.**8**, Issue.**1**, pp.**115-119, 2020.**

[5]. Rakesh Kumar, Priti Maheshwary, Timothy Malche, “*Inside Agile Family: Software Development Methodologies*”, International Journal of Computer Sciences and Engineering, Vol.**7**, Issue.**6**, pp.**650-660, 2019.**

[6]. Pfaff, Günther E. (Ed.). “User Interface Management Systems: Proceedings of the Workshop on User Interface Management Systems held in Seeheim”, Springer Science & Business Media, **2012.**

[7]. Schlungbaum, Egbert Et Elwert, Thomas. “*Automatic User Interface Generation from Declarative Models.*” In: CADUI. pp.**3-17, 1996.**

[8]. Myers Brad A., “*User Interface Software Tools*”, ACM Transac-tions on Computer Human Interaction. Vol.**1**, no.2, pp.**64-103, 1995.**

[9]. R. B. Hailpern , P. Tarr, “*Model-driven development : The good, the bad and the ugly*”, IBM Systems Journal, Vol.**3**, no.**45**, pp.**1-25, 2006.**

[10]. Smita Agarwal, S. Dixit, Alok Aggarwal, “*Model to Model Transformation for Declarative Models*”, International Journal of Computer Sciences and Engineering, Vol.**6**, Issue.**11**, pp.**164-170, 2018.**

[11]. D. Schmidt, “*Guest Editor's Introduction:Model-Driven Engineering*”, Computer, Vol.2, no.9, pp.**25-31, 2006**, ISSN 0018-9162.

[12]. Warnars, H. L. H. S. “*Object-oriented modelling with unified modelling language 2.0 for simple software application based on agile methodology*” Behaviour & Information Technology, Vol.**30**, no.**3**, pp.**293-307, 2011.**

[13]. Noulamo, T. & Tanyi, E. & Nkenlifack, Marcellin & Lienou, Jean-Pierre & Alain Bernard, Djimeli Tsajio. “*Formalization method of the UML statechart by transformation toward Petri Nets*.”, IAENG International Journal of Computer Science. 45. Pp.**505-513, 2018.**

[14]. Tajouo, François & Noulamo, Thierry & Lienou, Jean-Pierre. “*Procedure for the Contextual, Textual and Ontological Construction of Specialized Knowledge Bases”,* European Journal of Electrical Engineering and Computer Science. 5. pp.**62-67, 2021**, doi:10.24018/ejece.2021.5.1.282

[15]. André, Étienne, Choppy, Christine, et Noulamo, Thierry. “*Modelling timed concurrent systems using activity diagram patterns*”, Book Chapter, Springer International Publishing, pp.**339-351, 2015.**

[16]. T. Noulamo, B. Fotsing Talla, M. Wane, L. H. Nzothiam Takou, "*A Model-Driven Approach for Developing WEB Users Interfaces of Interactive Systems*", International Journal of Computer Trends and Technology (IJCTT) – Volume 68 Issue 4 pp.**33-43, April 2020**, ISSN: 2231-2803.
[17]. A.J. Dix, J. Finlay, G. Abowd, R. Beale. "Human-Computer Interaction", Prentice Hall, **1993.**
[18]. Scapin, Dominique Et Pierret-Golbreich, Christine. "*Towards a method for task description: MAD Work with display units*", Elsevier Science Publishers, North-Holland, Vol.**89**, pp.**371-380, 1989.**
[19]. D. Rix, H.R. Hartson. "Developing User Interfaces: Ensuring Usability Through Product Process", Wiley Professional Computing John Wiley Sons, USA, **1993.**
[20]. Gray, Phil, England, David, et Mcgowan, Steve. Xuan: "*Enhancing the UAN to capture temporal relation among actions.*" People and Computers IX, pp.**301-312, 1994.**
[21]. Fabio Patern. "*Model-Based Design and Evaluation of Interactive Applications*", Springer, **2001.**
[22]. Hix, Deborah. "*Generations of user-interface management systems.*", IEEE software, Vol.7, no.**5**, pp.**77-87, 1990.**
[23]. Bass, Len, Faneuf, Ross, Little, Reed, et al. "*A metamodel for the runtime architecture of an interactive system*". SIGCHI Bulletin, Vol.**24**, no.**1**, pp.**32-37, 1992.**
[24]. Coutaz, J.. "Interfaces Homme-Ordinateur, Conception et Ralisation ", Dunod Informatique, Paris, **1990**
[25]. Coutaz, Joëlle Et Nigay, Laurence. "Architecture logicielle conceptuelle des systèmes interactifs." Analyse et conception de l'IHM, pp.**207-246, 2001.**
[26]. Vasilakis, Christos, Lecznarowicz, D., Et Lee, Chooi, "*Developing model requirements for patient flow simulation studies using the Unified Modelling Language (UML)*". Journal of Simulation, Vol.**3**, no.**3**, pp.**141-149, 2009.**
[27]. Clerckx, T., Luyten, K., Coninx, K. "*Generating Context-Sensitive Multiple Device Interfaces from Design.*" In: Jacob, R.J., Limbourg, Q., Vanderdonckt, J. (eds) Computer-Aided Design of User Interfaces IV. Springer, Dordrecht, **2005**, https://doi.org/10.1007/1-4020-3304-4_23.
[28]. Nichols, Jeffrey & Myers, Brad & Higgins, Michael & Hughes, Joseph & Harris, Thomas & Rosenfeld, Roni & Pignol, Mathilde. "*Generating remote control interfaces for complex appliances. UIST (User Interface Software and Technology)*", Proceedings of the ACM Symposium. pp.**161-170, 2002**, doi;10.1145/571985.572008.
[29]. Booch, G., Rumbaugh, J., Jacobson, I. "*Unified Modeling Language User Guide, The (2nd Edition)*" (Addison-Wesley Object Technology Series). Addison-Wesley Professionnal, **2005**
[30]. France, Robert B., Kim, D.-K., Ghosh, Sudipto, et al. "*A UML-based pattern specification technique*", IEEE transactions on Software Engineering, Vol.**30**, no.**3**, pp.**193-206, 2004.**
[31]. Noulamo, T., Djimeli-Tsajio, A., Lienou, J. P., Fotsing-Talla, B. "*Agent platform for the remote monitoring and diagnostic in precision agriculture.*" Engineering Letter, Vol.**30**, Issue.**3**, pp.**972–980, 2022.**
[32]. Bast, Wim; Murphree, Michael; Lawley, Michael; Duddy, Keith; Belaunde, Mariano; Gri_n, Catherine; Sendall, Shane; Vojtisek, Didier; Steel, Jim; Helsen, Simon; Tratt, Laurence; Reddy, Sreedhar; Ven-katesh, R.; Blanc, Xavier; Dvorak, Radek; Willink, "Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT) ", Object Management Group, May **2011**

AUTHORS PROFILE

Noulamo Thierry earned his Master's degree, Diploma of Advanced Studies (DEA), and Ph.D. in software engineering science from the University of Yaounde I in 2000, 2001, and 2010, respectively. He is currently working as **lecturer in Department of Computer Engineering** from Fotso Victor University Institute of Technology (IUT-FV) of Bandjoun, since 2004. He is a member of EANG since 2009, Life member of UR.A.I.A since 2008. He has published more than 11 research papers in reputed international journals and conferences. His main research work focuses on Process Automation using MDE and Muti-Agent approach. He has 18 years of teaching experience and 11 years of research experience.

Bernard Fotsing Talla obtained his Bachelor in Mathematics and Fundamental Computer Science in 1999 at the University of Dschang. He obtained his Master's degree, his DEA (Diploma of Advanced Studies) and his Ph.D in Computer Science at the University of Yaoundé I in 2001, 2003 and 2010, respectively. He is currently working as a Lecturer in the Department of Computer Engineering of the Fotso Victor University Institute of Technology (IUT-FV) of Bandjoun since 2008. He is a member of the LAIA (Laboratory of Automatic Control and Applied Computer Science) of the IUTFV since 2010. He has published about ten Research articles in international journals including IJCTT, IJSER, ARIMA and conferences including CARI which are also available online. His main research work focuses on the design of model-driven software architectures, the specification of formal languages for the description of models using formal tools such as Attributed Grammars, and very recently the use of algorithms of Machine Learning for the diagnosis of certain diseases. He has 13 years of teaching experience and a few years of research experience.

Jean-Pierre Lienou, PhD., Lecturer/Researcher, he obtained his MSc in System Engineering in Kiev Polytechnic Institute (National Technical University of Ukraine) and his PhD in University of Yaounde I (Cameroon). Former Maintenance Engineer at Labotech Medical, he was in charge of Medical imaging equipment. He joining the University of Dschang since 2002. He is actually the Head of Department of Computer Engineering, College of Technology, The University of Bamenda. He is a member of EANG since 2018. His Research fields are Method Engineering applied in control systems, Multi Agent Systems applied in power electric systems, cyber resilience and the use of various artificial intelligent techniques in diagnostic of complex systems. He manages a grant related to cyber resilience funded by US ARL.

Djimeli Tsajio Alain Bernard received B.S. (2001) and M.S. with thesis (2004) from the Faculty of Science of the University of Yaoundé 1 and the Ph.D. (2016) from the Faculty of Science of the University of Dschang, all in Cameroon in the field of Physics option Electronics. Since 2006, he have joined Fotso Victor University Institute of Technology of the University of Dschang as lecturer in the Department of Telecommunication and Network Engineering. He is member of UR.A.I.A of the present university where he is carrying out research in the field of Artificial Intelligence for biomedical process.

