International Journal of Engineering& Scientific Research
Vol. 6Issue 3, Month 2018,
ISSN: 2347-6532 Impact Factor: 6.660
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com Double-
Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at:
Ulrich's Periodicals Directory ©, U.S.A., Open J-Gage as well as in Cabell's Directories of Publishing Opportunities, U.S.A

Graphical Domain Specific Language: Case study to explore the Model Driven Development (MDD)

Mais A. Khalil, Leicester University, Engineering Department, UK.
Bilal R. Al-Kaseem, Brunel University London, Engineering Department, UK.
Nazaraf Shah, Coventry University, School of Computing, Electronics, and Maths, UK.

Abstract

Modelling a new language that allows us to model new technical properties in an easier and simpler way, implement or describe solutions, or to explain the problem requirements in a more concise way is one of the fundamental challenges of computer science. The designing of a new language is a time consuming task, requires experience and is thus usually conducted by specialized language engineers and specialists. Nowadays, the need for new languages for various growing domains is significantly increasing. Recently, Domain-Specific Languages (DSLs) are becoming quite vital in software engineering. The significant question arising with this topic is why modelling important. In this paper an exploration example has been carried out to explain the main role of model driven aspects in facilitating the development process of a software product by implementing a navigational application model of student record domain as an example of graphical DSL.

Keywords:

MDD;
DSL;
UML;
EMF;

Author correspondence:

First Author,
Doctorate Program, Linguistics Program Studies
Udayana University,Jalan P.B.Sudirman, Denpasar, Bali-Indonesia

1. Introduction

The Aspect of Model Driven Development is not recent, and was studied extensively in diverse exploration papers [6], [3], [4], [5]. It is depended on the concept that, in order to develop a new software artifact, it is no longer required to write the real code. Instead, models can be used as a fundamental development tool. If we look in the classical software evolution process, it basically starts from writing down the requirements for the software product, which pursues with the analysis and design stage, and afterward moves to the implementation, testing and deployment stages. In this situation, it is not important which kind of the software development step we are looking at, whether it is classical waterfall, iterative or incremental approach. The significant elements are the artifacts that are applied during each development stage. These artifacts are text, some diagrams, and the ultimate code, which are mostly informal and loosely connected.In traditional software development, tools and notations are employed to show the details of the system that is being created to developers and the software architects.The present state of this process applies the Unifieded Modeling Language (UML),as an initial modeling notation. The UML allows to capture a range of significant properties of the system in corresponding models. Some of the UML modeling tools mostly support the traceability of requirements y with the help of supporting documents. But it is still difficult to keep the models synchronized with the code developed afterward. In Model Driven Development models become part of the developed software, this is because they are employed in each step of the development phase. The most famous implementation scheme for the Model Driven Development is Model

Driven Architecture (MDA) from Object Management Group (OMG). The development process with the MDA is shown in the Figure 1.

With the MDA the software development process begins with a computation independent model (CIM) explaining the business environment and business requirements of the system. The CIM is then refined to a platform independent model (PIM), which illustraites the system functionality, services and interfaces, but independent of any implementation concept or platform. The PIM is further refined to a platform specific model (PSM) which explores the realization of the system regarding the chosen software scheme.Later on the PSM model is used to generate the program code. The MDA process may look very much like traditional software development, but it is different.

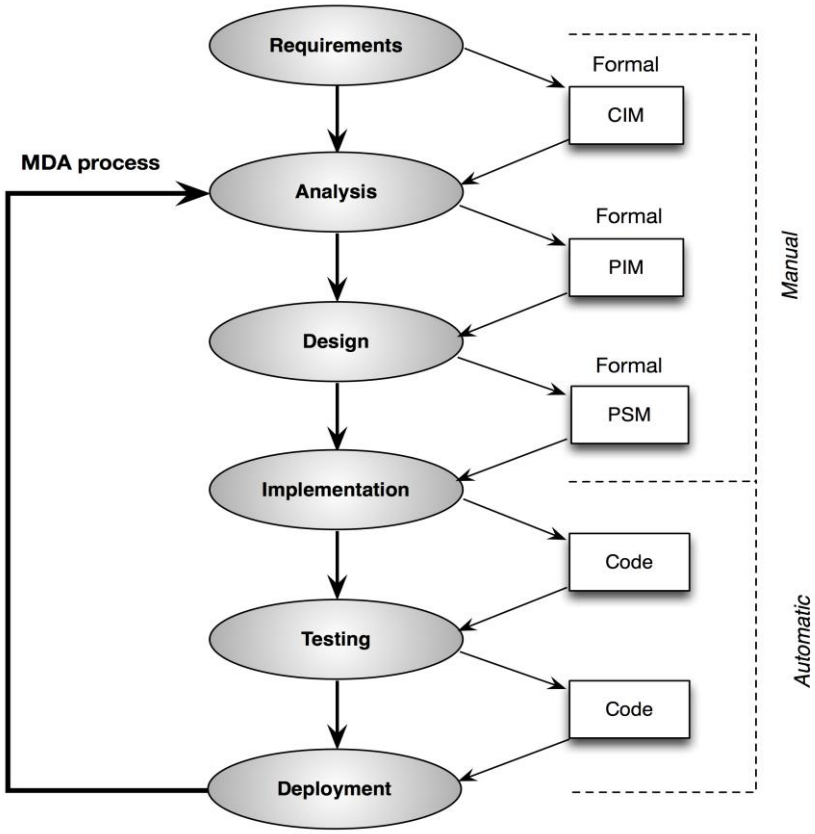

Figure 1: Software Development process with the MDA.

Plenty of tools are available for the code generation, but they do not go further thangeneration of the primary templates, and leave the code development on the developers. With the MDA models are used as a specification of the software on the requirements, analysis and design steps, and then for the automatic transformation to the code in theimplementation phase. The vital and important advantage of the MDA is that transformation between models is done automatically [7]. Four main steps in the Model Driven Architecture are presented in the Figure 2.

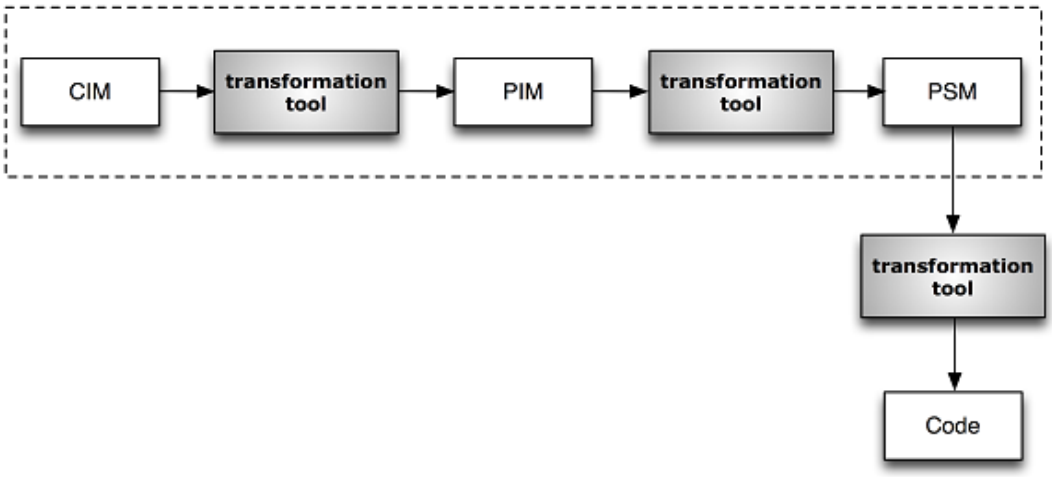

Figure 2: Four main steps in the MDA.

2. Genaral Purpose Language (GPL) VS. Domain Specific Language (DSL)

The definition of a domain-specific language is vague due to its dependability onterm "domain", which is itself unclear. Domain-specificity is not a clear-cut characteristic,but rather a gradual one[10]. For a better comprehension of DSLs, Voelter et al. [8] showprevailing variations between general-purpose languages and domain-specific languages,e.g. GPLs are always Turing-complete, while DSLs are often not; GPLs have a braod andcomplicated domain, while DSLs have limited and well-defined domains; GPLs have a lifespanof years to decades, while DSLs have a lifespan of months to years; and the progression and evolution of GPLs is often slow and standardized, while the evolution of DSLs is fast-tracked. it is clear that even more specialized languagesare effectivel. In short, a Domain-Specific Language is a languagethat is designed for a particular class of problems, called a domain.It is based on abstractions that are jointly aligned withthe field for which the language is developed. Specialized languages also come with a syntax appropriate for exploring these abstractions briefly.

3. Model Driven Development Principles

Model Driven Development is a paradigm for developing software quickly and efficiently. MDD is based on notion of construction of models and then transforming those models into system. Models represents higher level of abstraction and they are constructed used standard modelling languages. Models of the system represent the multiple views. *"MDD's defining characteristic is that software development's primary focus and products are models rather than computer programs"* [2]. MDD allows us to talk about concepts that are more close to domain and much less tied underlying programming language. The higher level of abstraction provides more choices to developer about how to realise models in executable code. Models enable the representation of software systems at a higher level of abstraction than GPLs and thus allow for greater reuse [9].

Unified Modelling Language (UML) has been used in his project to create class diagram representing our domain model. UML is industry standard visual modelling language used for modelling different view of a software system and it cater for specifying higher lever domain concepts for the system. UML is a mature and widely used modelling language for modelling object oriented systems. There has been number of automated tool available to transform UML model into implementation level artefacts of various object oriented programming languages. UML is widely used modelling language, however its primary focus never has been web application. UML 2 provides extension mechanism to deal with the modelling issues involved in web applications. The concepts involved in web applications are defined as stereo type provided in UML 2. Eclipse Modelling Framework(EMF) has been used to create class diagram and Ecore model of the system. EMF design tool has considerable limitation when it comes to modelling web applications. EMF design tool does not allow to represent stereo types and it also does not allow depict relationships such as build and submit that are core concepts in web application modelling. EMF has Ecore language for meta

modelling. By default ecore model are serialised in XMI format. XMI can be translated into various languages.
There are number of model to code transformation languages such as Xpand, Java Emitter Template (JET) and Acceleo. Xpand can be used to generate code for almost any known programming language. This project will employ Xpand for M2T. Xpand provides following features.

- It is a simple template language for producing generic text that helps to generate various type of code from the model
- Xpand has native support for Ecore.
- Xpand is available as Eclipse plugin and can be executed from within Eclipse.

Transformation technologies such as Atlas Transformations Language (ATL) or Query View Transformation (QVT) can also be used to generate output models (source code) from Ecore model.

4. Case Study

In This Paper we Introduce Navigational Model as a graphical DSL to demonstrate the primary concepts of the modelling approach. This work starts with a UML class diagram depicting static view of the chosen domain as a metamodel DSL. Navigational model diagram will be provided in order to illustrate the navigation view of the application. The navigation relationships between client pages and between client and server will be illustrated. Eclipse EMF has been used to generate meta model of the domain. The meta model is consist of two files know as ecore and genmodel. The ecore contains information about classes of the domain and genmodel contains information about coded generation.

4.1 DSL Matamodel (UML Class Diagram)

In order to express the domain concepts of the Component Diagram example, we need to create a metamodel. The metamodel is some kind of a schema for building models that names and describes the concepts of the domain. This metamodel is described using the UML class diagram, and includes all the concepts that we wanted to model and relationships between them.UML class diagram depicts static view of the model describing the behaviour and attributes of the model without providing implementation details of the methods. It also provided relationships between various classes involved in the model. These relationships are known as aggregation, generalisation and association, they represent composition, inheritance and connectivity respectively. Figure 3 shows class diagram of the student record domain. The diagram has been constructed using Eclipse Modelling Framework (EMF). Figure 4 depicts the associated Ecore model of the diagram 3. In next section navigation model of the application will be presented.

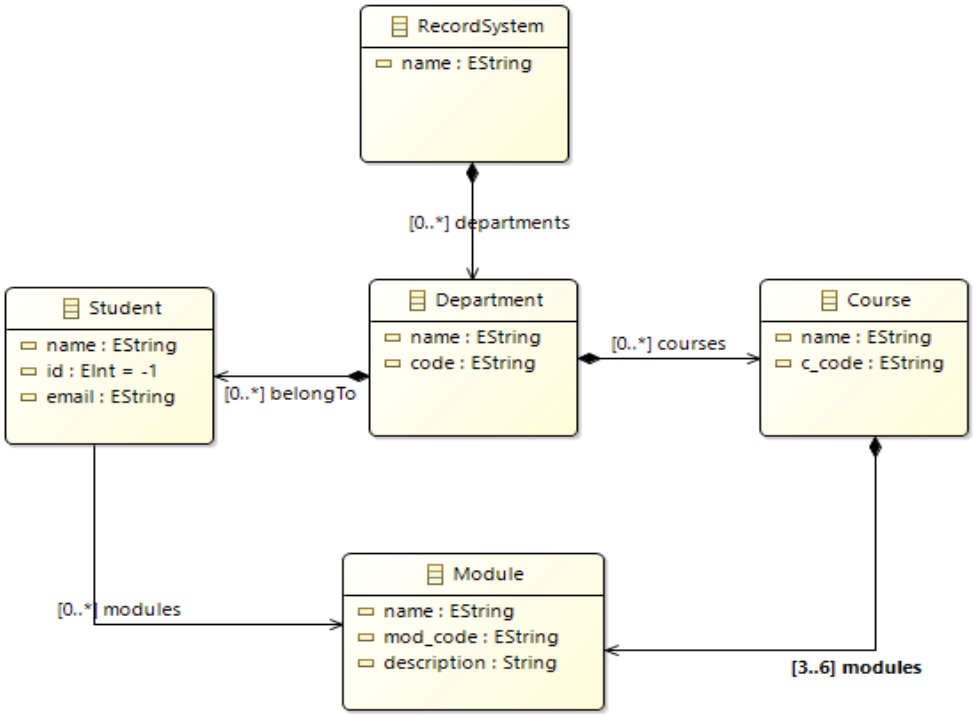

Figure 3: Student Record System Class Diagram (DSL MetaModel)

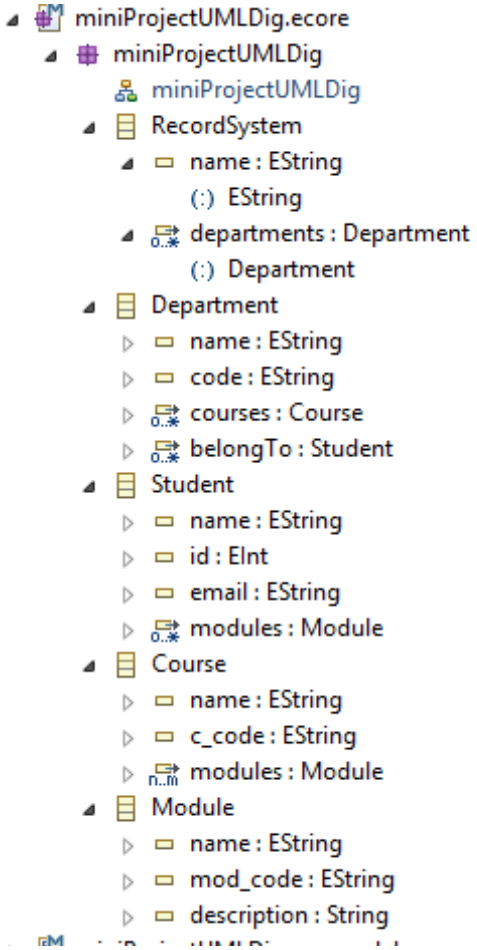

Figure 4: Associated Ecore Model

4.2 Application Navigation Model

Class diagrams are not enough for modelling web application as they do not explicitly described how application will be navigated by the users. Navigation model is an essential part of web application modelling for providing view of how application will be navigated. Navigation models are driven from class diagrams. In navigation model hyperlinks represent navigation path and they map to association relationship [1]. A client page has association either with a client page and a server page. The association between server page and client page is represented by <<build>> stereo type as server page interact with other server resources in order to generate client page. JavaScrips code is executed on client side and it represents the functions in client page whereas server side scripts such as JSP and PhP code represents function of server page. There is no direct association of client side and server side resources in web applications. <<Form>> tags are used to send data from a client page to a server page. Each form has an association relationship with a server page to which it is being submitted. The classes in Navigational Model are stereotypes, however EMF does not allow to use << and>> with classes names. Navigation involved in some important function shown in Figure 5.

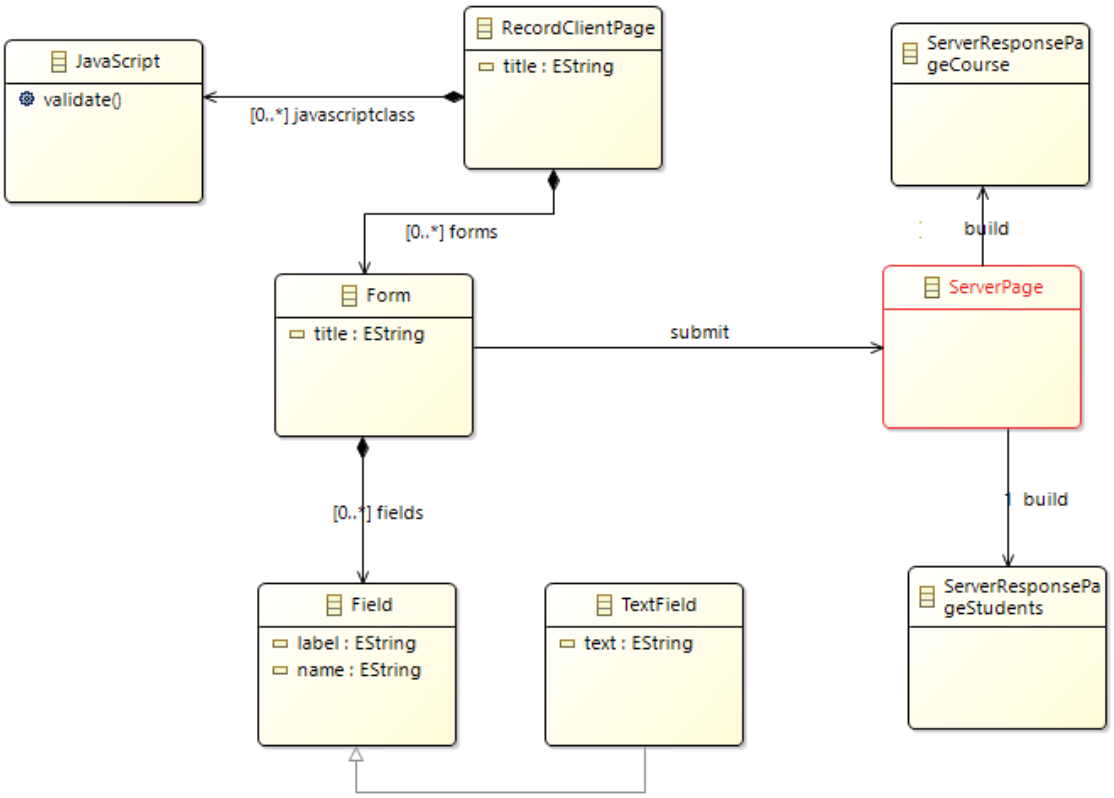

Figure 5: Navigational Model

4.3 Model to Text Transformation

It is necessary to define structure of the model before writing Xpand template to generate code. The following Ecore model is used for model to text transformation

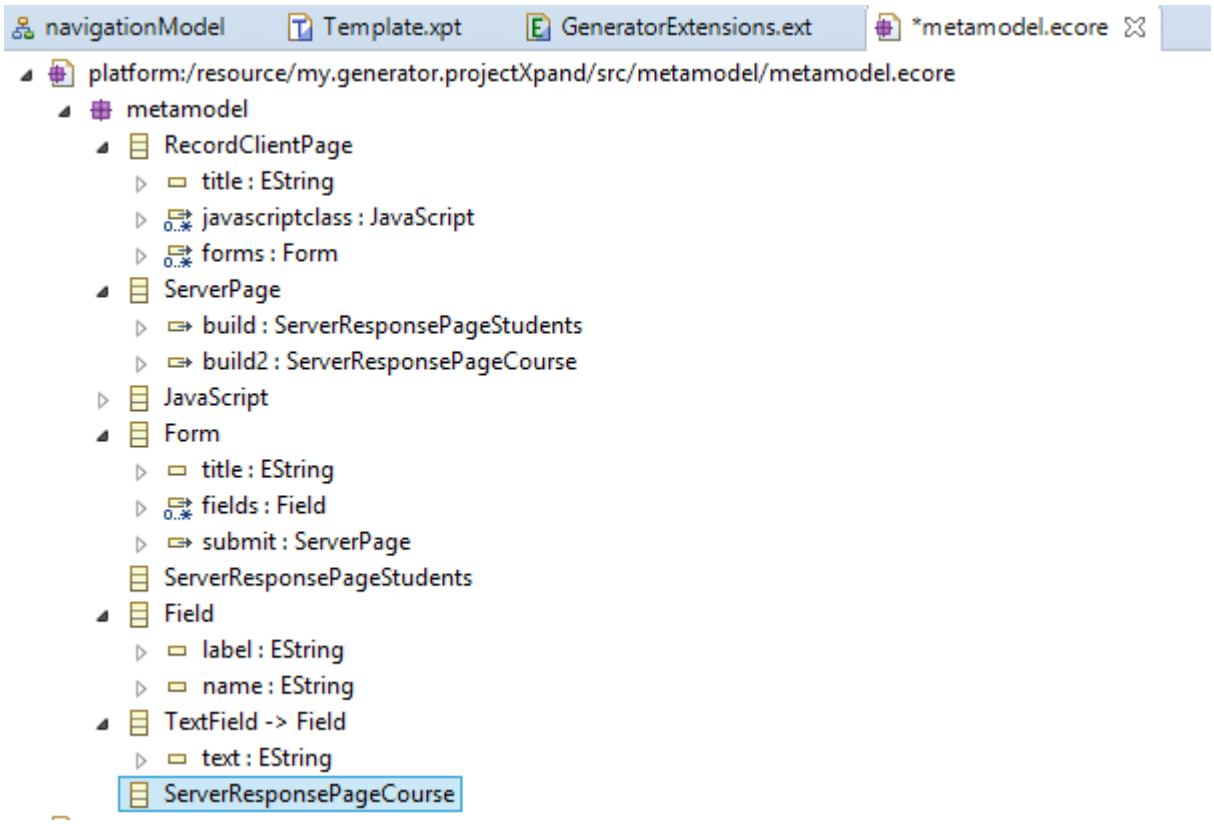

Figure 6: Ecore Model

Following steps are involved in model to text transformation using Xpand.

- Create Model: In Ecore model select classes and create their dynamic instances. Add values for the class attributes. The model for the ServerPage look like figure 7

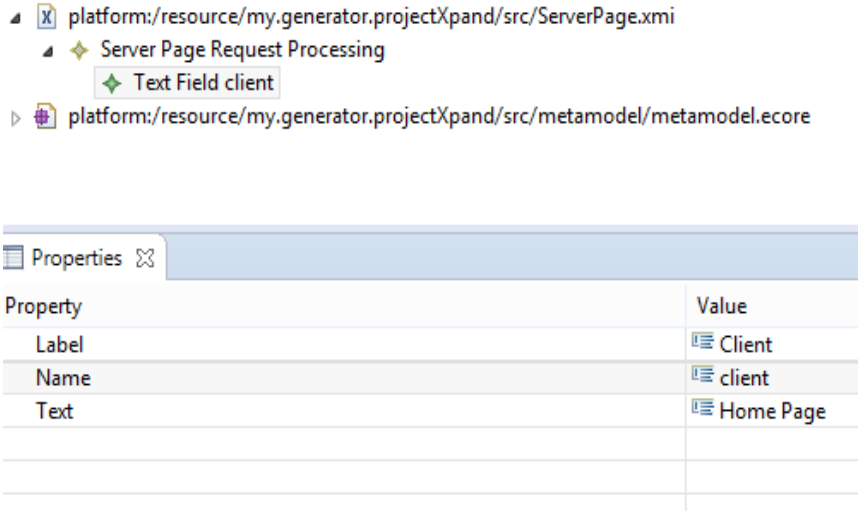

Figure 7: Sever Page Instance

Following is Xpand template file that is used to produce JSP page that will run on server side in a server which support JEE, such as Tomcat or GlassFish server.

```
«IMPORT metamodel»
«DEFINE main FOR ServerPage»
«EXPAND field FOREACHthis.resposes»
«ENDDEFINE»
«DEFINE field FOR TextField»
«FILE name + ".jsp"»
<%@pagecontentType="text/html"pageEncoding="UTF-8"%>
<!DOCTYPEhtmlPUBLIC"-
//W3C//DTDXHTML1.0Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1
-transitional.dtd">
<htmlxmlns="http://www.w3.org/1999/xhtml">
<head>
<title>«this.label»</title>

</head>

<body>

<p><b>«this.label» </P></p>

<ahref="<%=request.getContextPath()%>/«this.name».html">«this.text»</a>


</body>
</html>

 «ENDFILE»

  «ENDDEFINE»
```

```
<%@pagecontentType="text/html"pageEncoding="UTF-8"%>
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD XHTML 1.0
Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<htmlxmlns="http://www.w3.org/1999/xhtml">
<head>
<title> Client Request Processing ... by server</title>

</head>

<body>

<p><b> Client Request Processing ... by server </P></p>

<ahref="<%=request.getContextPath()%>/RecordClientpage.html"> Back to
Client</a>


</body>
</html>
```

Below is HTML translated out by GlassFish Server

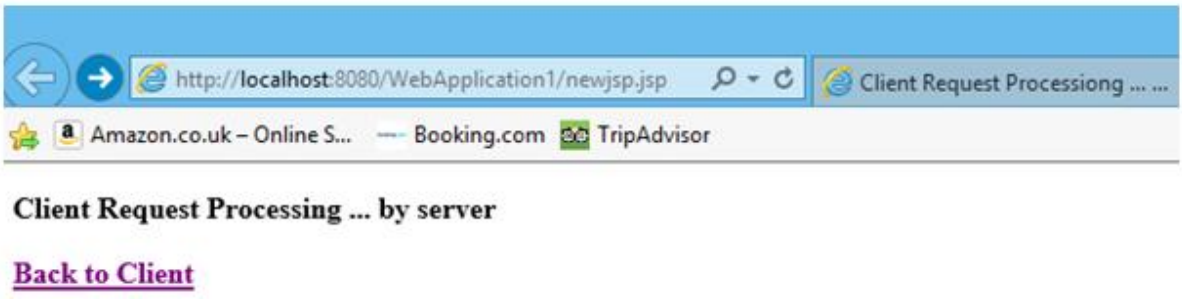

Figure 8: Sever Page

In following Xpand template for one client side page will be shown, remaining client side HTML pages can be generated similarly. I will start with Dynamic Instance creation diagram shown below

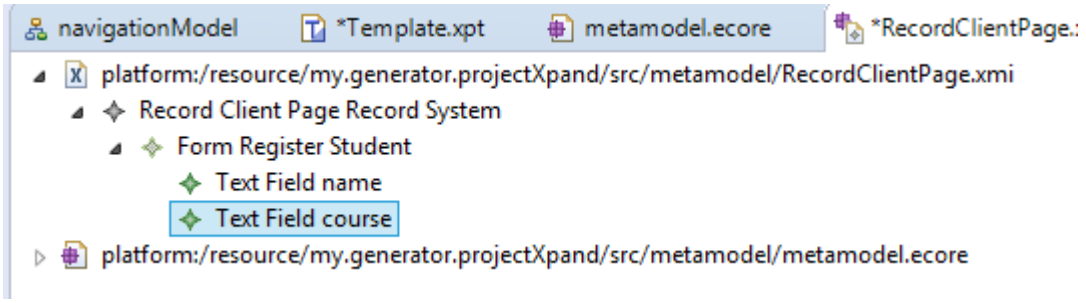

Figure 9: Client Page Instance

Below is client side registration form as shown in NetNeans IDE. The navigation between Client and Server Page is working as test on GlassFish application server.

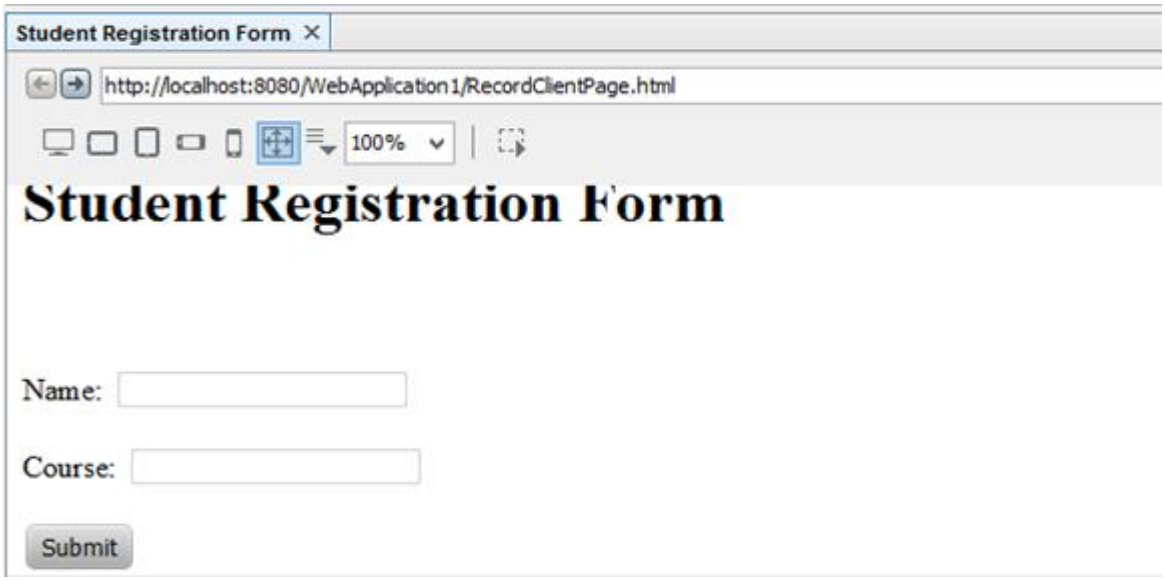

Figure 10: client side registration form

Below is Xpand template for generation HTML form for client page

```
«IMPORT metamodel»
«DEFINE main FOR RecordClientPage»
«EXPAND form FOREACH forms»
«ENDDEFINE»
«DEFINE form FOR Form»
«FILE"RecordClientPage.html"»
<!DOCTYPEhtmlPUBLIC"-
//W3C//DTDXHTML1.0Transitional//EN""http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<htmlxmlns="http://www.w3.org/1999/xhtml">
<head>
<title>«this.title»</title>

</head>
<body>
<divid="page-wrap">
<h1>«this.title»</h1><br/><br/>

<divid="form-area">
<formmethod="post"action="ServerPage.jsp">
<p> «EXPAND field FOREACHthis.field»
</P>
<inputtype="submit"name="submit"value="Submit"class="submit-button"/>
</form>
<divstyle="clear:both;"></div>
</div>
</div>
</body>
</html>
«ENDFILE»
«ENDDEFINE»


«DEFINE field FOR TextField»
<p>
<labelfor="«this.name»">«this.label»:</label>
<inputtype="text"name="«this.name»"id="«this.name»"/>
</p1>
«ENDDEFINE»
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Student Registration Form</title>
</head>
<body>
<div id="page-wrap">
<h1>Student Registration Form</h1><br /><br />
<div id="form-area">
<form method="post" action="ServerPage.jsp">
<p>
<label for="name">Name:</label>
<input type="text" name="name" id="name" />
</p>
<p>
<label for="course">Course:</label>
<input type="text" name="course" id="course" />
</P>
<input type="submit" name="submit" value="Submit" class="submit-button" />
</form>
```

Table 1: Translation Process

Script Code	**HTML output**	
<label for="«this.name»">«this.label»:</label>	**<label for="course">Course:</label>**	
<input type="text" name="«this.name»" id="«this.name»"/>	<input type="text" name="name" id="name" />	
«FILE"RecordClientPage.html"»	RecordClient.html	
«EXPAND field FOREACHthis.field»	<label for="name">Name:</label> <input type="text" name="name" id="name" /> <label for="course">Course:</label> <input type="text" name="course" id="course" /> </P>	
«this.title»	Student Registration Form	

5. Conclusion

It could be concluded from the above work that the graphical models provide a better overview and ease the understanding of models. Fortunately, also more sophisticated tools exist that allow software developers to design a new language with a reasonable effort. Consequently, an increasing number of DSLs (Domain Specific Languages) are designed to reinforce the productivity of software Engineers and developers within specific domains. DSL Tools are a sophisticated, but robust technology. Several tools are becoming better as well, so DSLs can be built with relatively little effort. It is convenient to use tools supporting the evolution of graphical DSLs such as the Eclipse Modelling Framework (EMF) as a point. Its key characteristic is the ease with which DSLs can be edited once their structure is defined. However, the definition of a DSL is not straightforwardtask. This article has only scratched the surface in terms of the possibilities of DSL Tools. However, I hope it can serve as a guidance for trying out more complex designs.

References

[1] Jim Conallen, "Modelling Web Application Architecture With UML", Communication of ACM, 1999.

[2]. Brian Selic, "The pragmatic of model driven development", IEEE Software, 2003.

[3] Anneke Kleppe; Jos Warmer; Wim Bast. MDA Explained: The Practice and Promise of the Model Driven Architecture. Addison-Wesley Professional, April 2003.

[4] Colin Atkinson; Thomas K□uhne. Model-Driven Development: A Metamodeling Foundation. IEEE Software, February 2003.

[5] Bran Selic. The Pragmatics of Model-Driven Development. IEEE Software, September 2003.

[6] Stephen J. Mellor; Kendall Scott; Axel Uhl; Dirk Weise. MDA Distilled: Principles of Model-Driven Architecture. Addison-Wesley Professional, March 2004.

[7] U. Tsyukh, "A Formal Modeling Notation for the Requirements of Work ow Systems", Technical University of Denmark, August 2010.

[8] M. Volter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. L. ¨ Kats, E. Visser, and G. Wachsmuth,
DSL Engineering - Designing, Implementing and Using Domain-Specific Languages. dslbook.org, 2013.

[9] J. O. Ringert, A. Roth, B. Rumpe, A. Wortmann, "Language and Code Generator Composition for Model-Driven Engineering of Robotics Component & Connector Systems", *Journal of Software Engineering for Robotics*, 2015.

[10] Sutii, AM Ana, "Modularity and reuse of domain-specific languages", Technische Universiteit Eindhoven,
2017.