Interoperability Plug-in between Petri Networks and SQL

Abderrazzak ZEDDARI and Ahmed ETTALBI

IMS Research Team, SIME Laboratory ENSIAS, Mohammed V University in Rabat, Morocco

Summary
Petri nets are a mathematical formalism for modeling a large set of dynamic systems. They are applied in practice by industry, academia, and in other domains. Its powerful positive is that there is different kind of Petri Nets, Ordinary Petri Nets, Timed Petri Nets, and Color
ed Petri Nets. So, it will be very beneficial to develop applications related to these Petri Nets. For this reason, the main objective of this work is to propose a translation process of a Petri Nets to Sql script following the MDA approach. The whole work is a framework intended to offer a code generator that will generate a whole user-defined application from a multiview system. This new framework will be used by companies that are working on agile software development to deliver to their customer a cloud environment dedicated to software production with a high level and scalable code and Application generator in order to minimize the time, cost and efforts.

Keywords

MDA, PNML, View and Viewpoint, Modeling, Colored Petri nets, Sql.

1. Introduction

In [1], we talk about the Petri Network Markup Language (PNML) which is a standard specification for PN that can be used to get all related data and meta-data of a graphical PN representation into an Xml format. In [2] the author introduces the basic theory of Petri nets and presents the most important of their properties relevant to manufacturing systems. He demonstrates the use of Petri nets in the preliminary design, that is, functional specification, modeling and evaluation of manufacturing systems. In [3], authors present a selection of the latest advances in the use of Petri nets for the modeling, analysis and management of communication networks and systems. It can be concluded that several publications talks about the large domain application of Petri Nets, so it will very beneficial to develop a whole system from a PN representation. For this reason we are working on a plug-in for communication between PN and Sql that was initiated in a recent work [4]. After having a valid Sql script, we will use the SpringRoo framework to generate a whole MVC application. The SpringRoo follows the Rapid Application Development (RAD) process and can reduce development cost and time. The rest of this paper is organized as follow: Section 2 presents an overview of the RAD process and some tools used with. Section 3 focuses on the View Point Petri Networks Markup Language which is an extension that we have proposed in [4]. Section 4 describes our proposed approach which is based on five rules and presents a case study to validate our proposed approach. We conclude this paper with conclusion and our further works.

2.RAD Process

2.1 Overview of the RAD Process

RAD (Rapid Application Development) is a model based on the concept that higher-quality products can be developed faster through more expedient processes, such as early prototyping, reusing software components and less formality in team communications. The next figure shows the RAD Process [5].

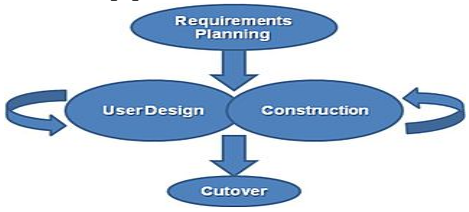

Fig.1 RAD Process

In this figure, we can see that the RAD Process starts with a requirements planning and finish with a delivery and cutover. During development, there is an iterative cycle that will be repeated for every customer request. This method gives a good quality with less cost.

2.2 Tools

RAD employs a variety of automated design and development tools, including CASE, 4GLs, visual programming and GUI builders. All of which help create prototypes and running applications faster than by coding program statements a line at a time. We present below a List of some RAD tools.

- Cross-platform RAD tools
    1. Code::Blocks [6].
    2. IBM Business Developer Extension [7].
    3. OpenObject (OpenERP) [8].

Manuscript received July 5, 2016
Manuscript revised July 20, 2016

- Desktop Rapid Application Development Tools
    1. MyEclipse [9].
    2. NetBeans [10].
- Database Rapid Application Development Tools
    1. Oracle Forms [11].
    2. Oracle Application Express (Oracle APEX) [12].

3. View Point PNML

3.1 Definition

The Petri Nets Markup Language (PNML) [1] is an XML-based interchange format for Petri nets defined by the standard ISO/IEC 15909. PNML supports any version of Petri net, new Petri net types can be defined by the so-called Petri Net Type Definitions (PNTD). The next figure shows an example of a Petri Nets Diagrams.

```
<pnml xmlns="http://www.pnml.org/version-2009/grammar/pnml">
<net id="n1" type="http://www.pnml.org/version-2009/grammar/ptnet">
<page id="top-level">
<name>
 <text>An example P/T-net</text>
</name>
<place id="p1" viewPoint="vp1">
    <graphics>
        <position x="20" y="20"/>
     </graphics>
    <name>
        <text>ready</text>
        <graphics>
            <offset x="0" y="-10"/>
         </graphics>
    </name>
    <initialMarking>
        <text>3</text>
        <toolspecific tool="org.pnml.tool" version="1.0">
            <tokengraphics>
               <tokenposition x="-2" y="-2" />
               <tokenposition x="2" y="0" />
               <tokenposition x="-2" y="2" />
            </tokengraphics>
         </toolspecific>
    </initialMarking>
</place>
....
```

Fig.2 PNML example

3.2 View Point PNML

In [4] we have proposed an extension of PNML which is the VP-PNML for View Point PNML. It represents the PNML related to a muliview system. This extension is based on new tags called: Views and Point Of Views. The next figure shows an example of View Point PNML

```
<views>
    <view>
        <id>v1_id</id>
        <name>v1</name>
        <attributes>
            <attribute type='champ'>Ref</attribute>
            <attribute type='champ'>brand</attribute>
            <attribute type='champ'>color</attribute>
            <attribute type='champ'>fuel</attribute>
            <attribute type='champ'>Ref</attribute>
            <attribute type='method'>ShowInfo()</attribute>
        </attributes>
    </view>
    <view>v2 ...</view>
    <view>v3 ...</view>
</views>
<pointOfViews>
    <pointOfView>
        <name>vp1</name>
        <placeName>p1</placeName>
        <placeId>p1</placeId>
        <views>
            <view>
                <id>v1_id</id>
                <state>read</state>
            </view>
            <view>
                <id>v2_id</id>
                <state>write</state>
            </view>
        </views>
    </pointOfView>
</pointOfViews>
```

Fig.3 View Point PNML example

4. VP-PNML to SQL Approach

4.1 Approach Description

The use of relational databases is very benefic on software development. Furthermore, the database design is the most important phase in a project because when designing a correct and coherent database schema, we can improve quality and cost during development. We present below the list of rules that should be respected during conversion.

Rule 1:
For each view that is to be converted the target generates CREATE TABLE statements.

Rule 2:
For each point of View that is to be converted the target generates CREATE TABLE statements with a foreign key linked to the recently created tables views associated to the point of view.

Rule 3:
Table name: The name of the view is used as the table name.
Primary key name: The name of the primary key column is defined via the view id parameter.
Primary key type: The data type of the primary key column depends upon the database system, for example *bigserial* in PostgreSQL, *integer* in Oracle...

Rule 4:
Each views property is converted into column definition statements.

Rule 5:
For Views Method:
If the method has no return type, the method is converted into column definition statements.
If the method has a return type, it will be added to the Source Code during development.

4.2 Case Studies

Multiview Class Car:
In [13], we have proposed and developed an example of modeling a multiview class Car using Colored Petri Network.

Below is the corresponding CPN:

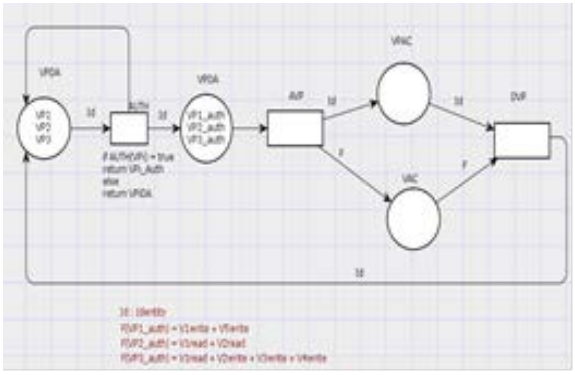

Fig.4 Colored Petri Net of the multiview class Car

In this example, we have 3 points of view: VP1, VP2 and VP3. In Table 1, we present for each viewpoint, the views composing it.

- Vw: View with Write state.
- Vr: View with Read state only.

Table 1: Composition of viewpoints in terms of views

VP1: Mechanic	*VP2: Client*	*VP3: Commercial*
V1w+V5w	V1r+V2r	V1w+V2w+V3w+V4w

In Table 2, we present the views of the class Car.

Table 2: Views related to the class Car

V1	*V2*	*V3*	*V4*	*V5*
Ref	discount	Recommended_Price	Modify_Info()	RegisterFailure()
brand	Selling_Price	Answer_Proposal()		RepairFailure()
color	ShowInfo()			
fuel				
consumption				
ShowInfo()				

Now, we will apply the conversion rules to this case:

Rule 1:
For each view that is to be converted the target generates CREATE TABLE statements.

```
CREATE TABLE V1 ( ... )
CREATE TABLE V2 ( ... )
...
```

Rule 2:
For each **point_of_view** that is to be converted, the target generates **CREATE TABLE** statements with a foreign key linked to all or some recently created tables.
For example, we have VP1 = V1+V5

```
create table VP1
(
   VP1_ID              int not null,
   V1_ID               int,
   V5_ID               int,
   primary key (VP1_ID)
);

alter table VP1 add constraint FK_REFERENCE_VP1_V1 foreign key (V1_ID)
      references V1 (V1_ID) on delete restrict on update restrict;

alter table VP1 add constraint FK_REFERENCE_VP1_V2 foreign key (V5_ID)
      references V5 (V5_ID) on delete restrict on update restrict;
```

Fig.5 Created Table

Rule 3 + Rule 4 + Rule 5:
The next figure shows the created table from the View.

```
CREATE TABLE V1
(
V1_Id int NOT NULL,
Ref varchar(255) NOT NULL,
brand varchar(255) NOT NULL,
color varchar(255) NOT NULL,
fuel varchar(255) NOT NULL,
consumption varchar(255) NOT NULL,
showInfo varchar(255) NOT NULL,

PRIMARY KEY (V1_Id)
)
```

Fig.6 Created Table

Below are the whole Schema tables.

VP1: Mechanic

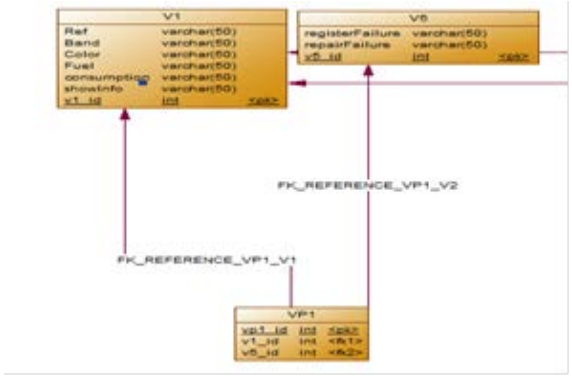

Fig.7 MCD Diagramm for Mechanic View Point

VP2: Client

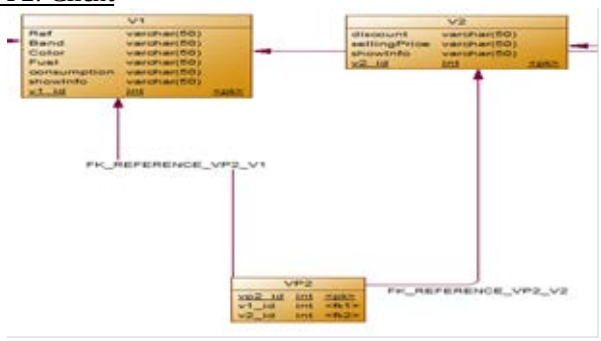

Fig.8 MCD Diagramm for Client View Point

VP3: Commercial

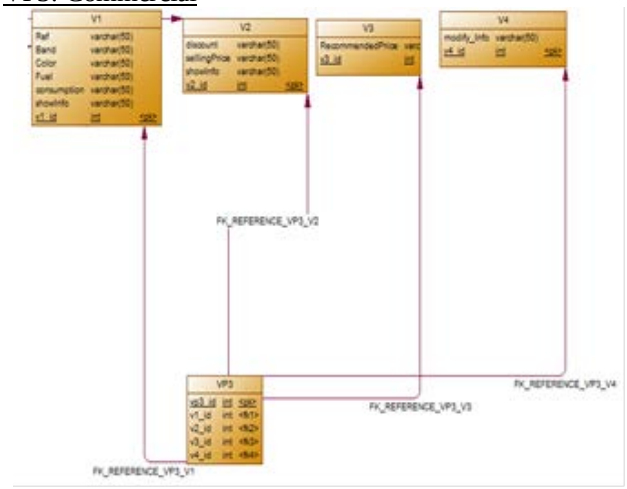

Fig.9 MCD Diagramm for Commercial View Point

The next figure presents the created database:

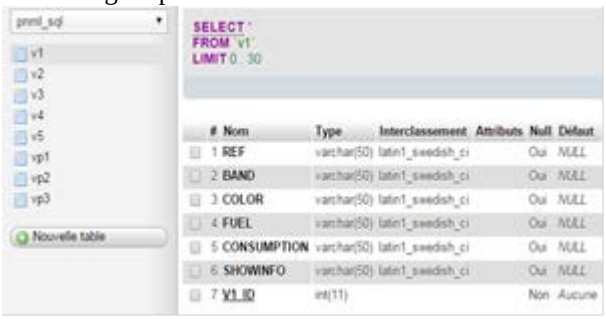

Fig.10 Generated DB for the multiview Class

In the next section we will show how to create a complete Web application from scratch using the RAD Tool: SpringRoo. The application we are going to develop will demonstrate many of the core features offered by this RAD tool. To demonstrate the development of an application using Spring Roo we will create a Web app for the multiview class car. The requirements for the car web application include the ability to create and manage car objects.

Figure 11 shows the list of the RooShell commands that can be executed in a few minutes to obtain a whole secured web application. The RooShell needs as input a relational database (Oracle, Mysql, ..). For each RDBMS a different Jdbc driver should be used for connectivity. After loading the driver we start the persistence step that generates the PoJo Model of our database. The last step is the generation of the user interfaces with the authentication and the authorization (AA) security setup

```
/* Load Database Java Connector in Roo Shell */
osgi start --url file:///D:/WDev/SpringRoo/mysql-connector-java-5.1.26.jar

/* Database Connection */
persistence setup --provider HIBERNATE --database MYSQL --databaseName roo --userName root --password

/* Database reverse engineering */
database introspect --schema dbo --enableViews
database reverse engineer --schema dbo --package ~.domain --testAutomatically --enableViews

/* View layer generation */
web mvc setup
web mvc all --package ~

/* Log layer configuration */
logging setup --level INFO

/* Login setup */
security setup

/* Packaging sources to load in remote repository */
perform package
```

Fig.11 Created Table

The figure 12 shows the Roo Shell terminal, where we can run the Roo commands. Each command is executed separately and sequentially.

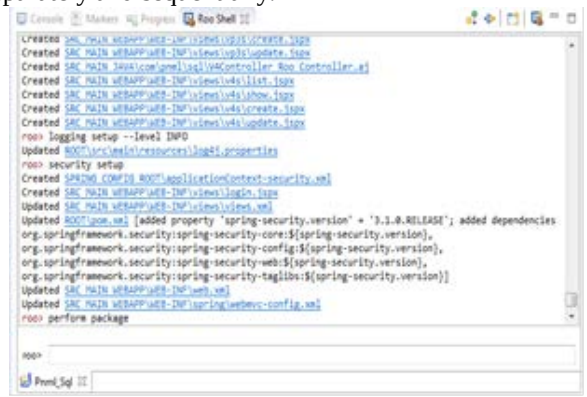

Fig.12 Spring Roo Shell Traces

The next figure shows the created Eclipse Project. In this project, we used the following technologies: Spring, Hibernate, Web Flow, Maven and Log4j as a logger.

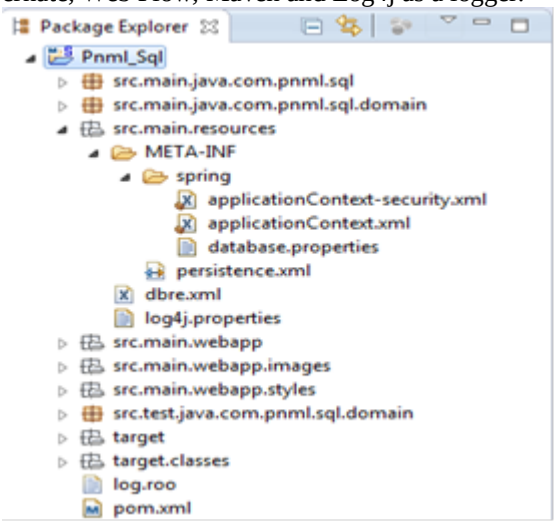

Fig.13 Generated Project in Eclipse

Below is the generated application:

Fig.14 Generated App: login Page

Fig.15 Generated App: Welcome Page

Multiview Class Software:

In [13], we have proposed and developed an example of modeling a multiview class Software using Colored Petri Network. The next figure presents the corresponding CPN.

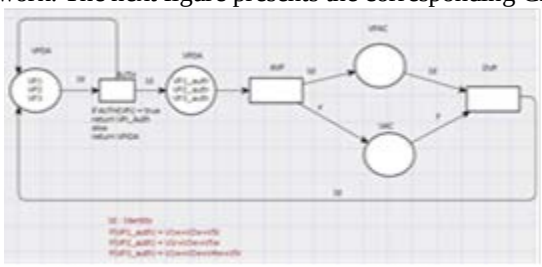

Fig.16 Colored Petri Net of the multiview class Software

In this example, we have 3 points of view: VP1, VP2 and VP3. In Table 3, we present for each viewpoint, the views composing it.

- Vw: View with Write state.
- Vr: View with Read state only.

Table 3: Composition of viewpoints in terms of views

VP1: Customer	***VP2: Developer***	***VP3: Manager***
V1w+V2w+V5r	V1r+V3w+V5w	V1w+V2w+V4w+V5r

In Table 4, we present the views of the class Software.

Table 4: Views related to the class Software

V1	***V2***	***V3***	***V4***	***V5***
name	price	language	cost	Document ation
deadl ine	changeDea dline()	chooseLanguage()	changeCost()	
		upgradeVersion()	showVersion()	
		updateDocumentati on()		

Now, we will apply the conversion rules to this example:

Below are the whole Schema tables of VP1, VP2 and VP3.

VP1: Customer

Fig.17 MCD Diagramm for Customer View Point

VP2: Developer

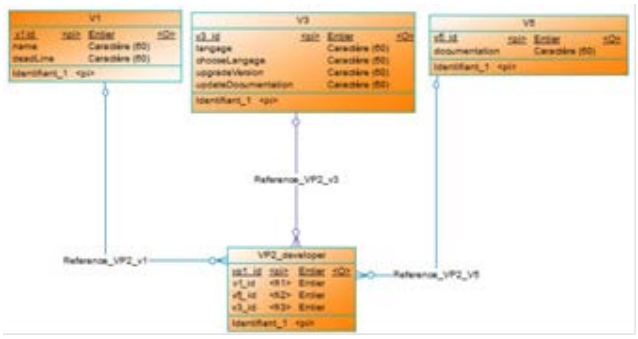

Fig.18 MCD Diagramm for Developer View Point

VP3: Manager

Fig.19 MCD Diagramm for Manager View Point

The next figure presents the created database from the below diagrams:

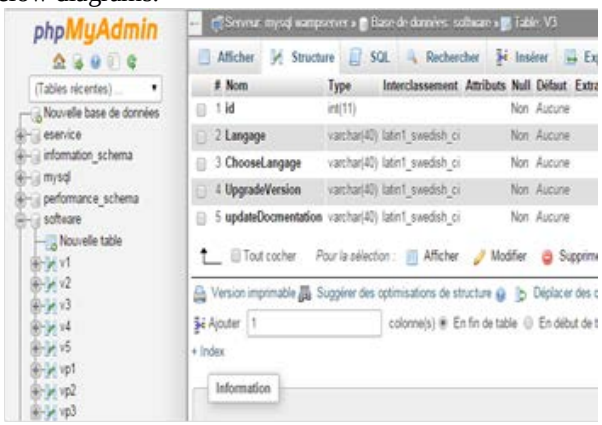

Fig.20 Generated DB for the Software Class

Below is the Eclipse Project.

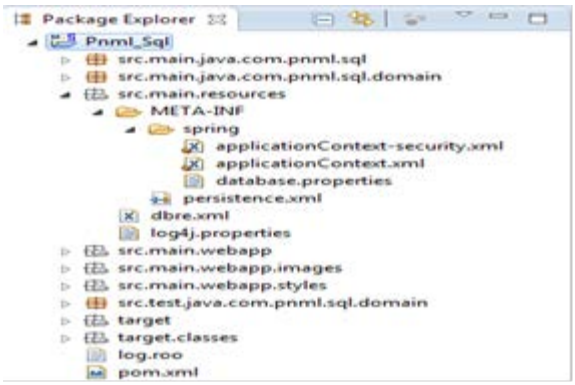

Fig.21 Generated Project in Eclipse

The next figure presents the generated application.

Fig.22 Generated App: login Page

5. CONCLUSION

In summary, we have proved that we can implement a plug-in to ensure the data conversion from a Petri Nets to a Sql schema. It consists on the use of the Petri Nets metadata such as the PNML files to gets all related informations and build our database. Our work would be more efficient if more researches people took part in it because the domain of Information and Communication Technology (ICT) evolved rapidly. We should continue to try to improve it. Our perspective is to develop a new framework intended for companies that are working on agile software development. With this framework, they can deliver to their customer a cloud environment dedicated to software production with a high level and scalable code and applications generator. Also this kind of generator will have a good contribution in the Cloud domain especially the SaaS level.

Acknowledgment

I express my warm thanks to all IMS team members and all the people who provided me with the facilities being required and conductive conditions for this article.

References

[1] M.Weber, E.Kindler, The petri net markup language, Lecture Notes in Computer Science, Humboldt-UniversitÂ¨at zu Berlin, Institut fur Informatik, 2003.

[2] S. K. Khator, A review of: Practice of petri nets in manufacturing. by f. dicesare, g. harhalakis, j. m. proth, m. silva and f. b. vernadat (chapman and hall, 1993). isbn 0-412-41230-6 [pp. 320].

[3] J. Billington, M. Diaz, G. Rozenberg (Eds.), Application of Petri Nets to Communication Networks. Lecture Notes in Computer Science, vol. 1605, Springer-Verlag, 1999, ISBN: 3-540-65870-X.

[4] A. ZEDDARI, A. ETTALBI, Cloud saas using mda approach on a multiview models, in: IEEE International Conference on Cloud Computing Technologies and Applications, June 2-6, 2015, Marrakesh, Morocco.

[5] J. Martin, Rapid application development, Macmillan, pp. 81-90, 1991

[6] http://codeblocks.org/.

[7] http://www-03.ibm.com/software/products/en/busdeveloper

[8] https://www.academia.edu/7578211/Open_Source_RAD_with_OpenERP_7.0

[9] New MyEclipse IDE for Spring Developers, John K. Waters, 1105 Media, 2010.

[10] https://netbeans.org/community/releases/roadmap.html

[11] http://www.oracle.com/technetwork/developetools/forms/forms11gr2newfeatures-497502-en-gb.pd

[12] "Oracle APEX 5.0 released today". Dimitri Gielis Blog. April 15, 2015. Retrieved December 10, 2015.

[13] A. ZEDDARI, A. ETTALBI, Towards a new framework for building a whole user-defined system from a colored petri networks, Journal of Communication and Computer (JCC), Vol. 12, No. 4, pp. 184-190, Avril, 2015.

[14] Martin, James (1991). Rapid Application Development. Macmillan. pp. 81–90. ISBN 0-02-376775-8

ZEDDARI Abderrazzak PhD student at the Higher National School of Computer Science and Systems Analysis (ENSIAS) Rabat**,** software development Engineer, Rabat.
Associate Degree in Science (Honors).
Java Senior Developer.
abderrazzak.zeddari@um5s.net.ma
azeddari@gmail.com

ETTALBI Ahmed Professor at Software Engineering Department of the Higher National School of Computer Science and Systems Analysis (ENSIAS) Rabat. His main research interests Object Modeling with Viewpoints, Software Architecture, Cloud Computing and Business Process Modeling architecture.
ettalbi1000@gmail.com