Integrating the Development of Data Mining and Data Warehouses via Model-driven Engineering

Jose Zubcoff[1], Jesús Pardillo[2], Jose-Norberto Mazón[2], and Juan Trujillo[2]

[1] Department of Sea Sciences and Applied Biology,
University of Alicante, Spain
Jose.Zubcoff@ua.es

[2] Department of Software and Computing Systems,
University of Alicante, Spain
{jesuspv,jnmazon,jtrujillo}@dlsi.ua.es

Abstract. Data mining is one of the most important analysis techniques to automatically extract knowledge from large amount of data. Nowadays, data mining is based on low-level specifications of the employed techniques typically bounded to a specific analysis platform. Therefore, data mining lacks a modelling architecture that allows analysts to consider it as a truly software-engineering process. Bearing in mind this situation, we propose a model-driven approach which is based on (i) a conceptual modelling framework for data mining, and (ii) a set of model transformations to automatically generate both the data under analysis (that is deployed via data-warehousing technology) and the analysis models for data mining (tailored to a specific platform). Thus, analysts can concentrate on understanding the analysis problem via conceptual data-mining models instead of wasting efforts on low-level programming tasks related to the underlying-platform technical details. These time consuming tasks are now entrusted to the model-transformations scaffolding. The feasibility of our approach is shown by means of a hypothetical data-mining scenario where a time series analysis is required.

Keywords: data mining, data warehouse, model-driven engineering, model transformation, multidimensional modelling, conceptual modelling.

1 Introduction

Data-mining techniques allow analysts to discover knowledge (e.g. patterns and trends) in very large and heterogeneous data sets. Data mining is a highly complex task which requires a great effort in preprocessing data under analysis, e.g. data exploration, cleansing, and integration [1]. Therefore, some authors suggest the suitability of data-warehousing technologies [2] for improving the conventional knowledge discovery process by means of providing an integrated and cleansed collection of data over which data-mining techniques can be straight applied [3,4]. However, current data-mining literature has been focused on the presenting new techniques and improving the underlying algorithms [5], whilst the most known software platforms do not apply the data warehousing principles in data-mining design. To overcome this situation, several mechanisms have been proposed [6,7,8,9] to model data mining techniques in conjunction with data-warehousing technology from the early stages of design (i.e. conceptual). These data-mining models do not only support analysts in using and understanding the required data-mining techniques in real-life scenarios, but also allow designers to document the data-mining techniques in detail. Hence, these data-mining models are truly blueprints that can be used to manually obtain the required data-mining metadata as a basis of the implementation in a certain data-mining platform. However, this highly-complex task is only accessible to expert analysts and requires too much effort to be successfully completed [3,10].

In this work, we will go beyond the definition of new models, here we define a model-driven engineering [11] approach for data-mining. Moreover, we propose the use of a well-known visual modelling standard, the "unified modelling language" (UML) [12] for facilitating the design and implementation tasks. In order to spread the usage of data-mining models to a broader scope of analysts and reduce the required effort our approach automatically generate a vendor-specific data-mining implementation from a conceptual data-mining model, taking into consideration the deployment of underneath data warehouse (i.e. data under analysis).

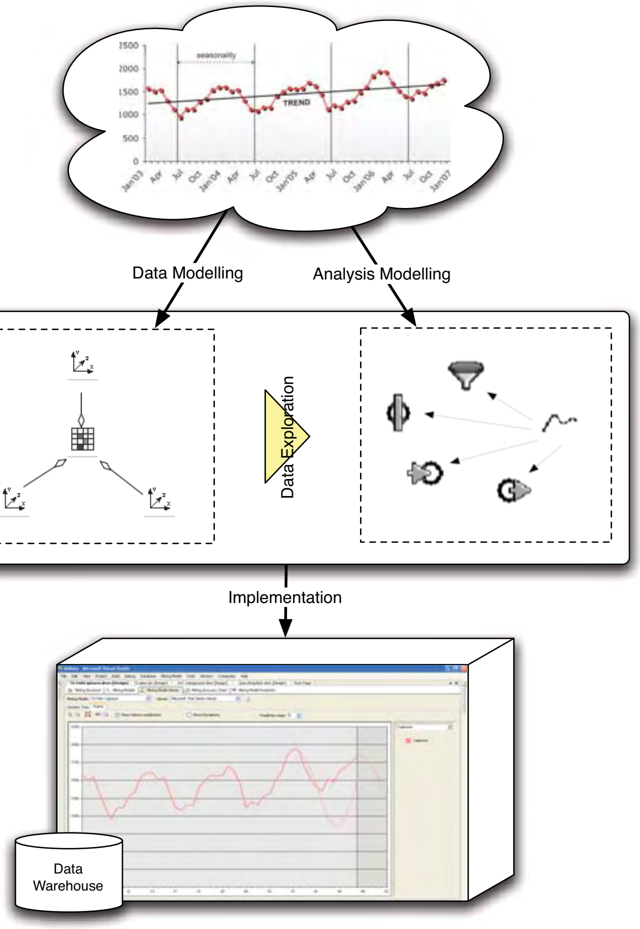


Fig. 1. Modelling example of a time-series analysis in data warehouses

Running example. In order to illustrate the discussion, in Fig. 1, we show an example of data-mining conceptual modelling. In this example, a small organisation is trying to highlight patterns and trends in the evolution of the fish-species population along time. Therefore, a time-series analysis has to be modelled [9]. Together with this data-mining technique, the data under analysis can be specified by exploring the underlying multidimensional data of the data warehouse. This crucial phase is done at a high level of abstraction, that is at conceptual level, focusing only on data mining concepts and avoiding platform specific details. This process is represented in the middle of Fig. 1. Finally, to obtain the results, our approach provides the required transformations for mapping the data-mining conceptual models to a vendor-specific data-mining implementation[3].

Outline. The rest of the paper is structured as follows: the next Section outlines the related work. Section 3.2 describes our model-engineering approach for data mining. A running example is used through the paper to clarify every theoretical detail Finally, Section 4 exposes conclusions, also sketching the future work.

2 Related Work

Current approaches for data-mining design can be classified on those that are a general description of data mining process, or those that are mathematical oriented and propose solutions at a very low-abstraction level. Therefore, these approaches overlook the definition of understandable artifacts which could be easily used by designers in a software engineering process. The main standard proposed for the data mining process is the "cross industry standard process for data mining" (CRISP-DM) [13]. This standard is a detailed description of each of the six phases of the data mining process. This standard neither proposes a concrete modeling tool nor present a conceptual model for data mining. CRISP-DM is focused on the description of how to perform a data mining task.

An overview of current data-mining modelling languages is provided in Table 1. The "common warehouse metamodel" (CWM) [12] and the "predictive model markup language" (PMML)[4][5] are really standards for the metadata interchange proposed by vendor-independent consortiums (OMG and DMG, respectively) between data-mining applications based on XML, but they cannot be used as analysis artefacts. The "data mining extensions" (DMX)[6] is a SQL-like language for (textually) coding data-mining models in the Microsoft Analysis Services platform, and therefore it is difficult to gain understanding of the data-mining domain. In addition, some data-mining libraries have been also proposed as a modelling mechanism. Two of the most known are the "extended library for Prudsys embedded solutions" (XELOPES)[7] (derived from CWM) and Weka[8]. They provide an entire framework to carry out data mining but, once again, they are situated at very low-abstraction level, since they are code-oriented and they do not contribute to facilitate understanding of the domain problem. On the other hand, there are software architectures related to data mining such as the "pattern-base management system" (PBMS)[9] designed to store and manage patterns obtained from the usage of data-mining techniques, but they cannot be considered a truly modelling proposal as we state herein.

All of these approaches have the same drawback, since they are focused on solving the technical scaffolding instead of providing analysts with intuitive artefacts to specify data mining. To the best of our knowledge, only the proposal described in Zubcoff *et al.* [6,7,8,9] provides a modelling framework (named as CDM in Table 1) to define data-mining techniques at a high-abstraction level by using the "unified modelling language" (UML) [12]. However, these UML-based models are mainly used as documentation. In this paper, we propose to extend this modelling framework as a first step in turning data mining into a

[3] The code is included in the appendix A
[4] URL: `www.dmg.org/pmml-v3-2.html` (March 2008)
[5] In Table 1, we exclude PMML due to the space constraints. PMML is similar to CWM but it is a language (Type field) coded in XML schema (Technology).
[6] URL: `msdn2.microsoft.com/en-us/library/ms132058(VS.90).aspx` (March 2008)
[7] URL: `www.prudsys.com/Produkte/Algorithmen/Xelopes` (March 2008)
[8] URL: `www.cs.waikato.ac.nz/ml/weka` (March 2008)
[9] URL: `www.pbms.org` (March 2008)

Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 2, No. 1, 2008

Table 1. Comparison of data-mining modelling languages

Language	CDM [6,7,8,9]	CWM	XELOPES	DMX	Weka
Type	metamodel	metamodel	library	query language	library
Technology	UML profiles	MOF Instance	CWM Extension	SQL-like	Java
Subject	interaction	interoperability	interoperability computation	querying	computation
Abstraction	high	middle	middle	low	low
Complexity	low	medium	high	medium	high
User type	analysts	data managers	data managers	data miners	data miners
Expertise	low	medium	high	medium	high

real software engineering process (see Fig. 1). Specifically, we use model-driven engineering concepts to (i) specify data-mining analysis in two technology-independent models (the multidimensional-data model of the underlying data warehouse and the data mining technique model), and (ii) provide transformations to automatically deploy data and analysis specifications into their physical implementations.

3 Model-driven Architecture for Data Mining in Data Warehouses

Our model-driven engineering approach for data mining advocate defining the underneath data warehouse (i.e. data under analysis) together with the data-mining technique. In this section, both tasks are explained, as well as the required transformations to obtain the data-mining implementation. Furthermore, our running example is used through this section to clarify the theoretical details.

3.1 Deployment of Data under Analysis

This section explains how to develop the underlying data warehouse required for data-mining to provide integrated and cleansed data. The data warehouse is based on a multidimensional model which defines the required data structures, namely facts and dimensions and their respective measures, hierarchies and attributes. Multidimensional modelling resembles the traditional database design [4]. First, a conceptual design phase is performed whose output is an implementation-independent and expressive conceptual multidimensional model for the data warehouse. A logical design phase then aims to obtain a technology-dependent model from the previously defined conceptual multidimensional model. This logical model is the basis for the implementation of the data warehouse. In previous work, we have aligned this process with a model-driven approach [14,15,16,17] in order to support designers to develop a conceptual multidimensional model and the automatic derivation of its corresponding implementation.

A conceptual multidimensional model for the running example is provided in the Fig. 2. This conceptual model has been defined by using our UML profile for multidimensional modelling [18]. A *capture* fact have been designed together with a set of four dimensions: *fish*, *ship*, *time*, and *location*. For each dimension, several levels are defined that form a hierarchy of aggregation. Locations have a <*site, marine area, region*> hierarchy, fish have a <*species, genus, family*> hierarchy, and the time dimension has the typical <*day, week, month, quarter, year*> hierarchy. Finally, ships have no hierarchies to be described, thus presenting only one aggregation level with data of the ship.

From this conceptual model, an implementation of the required data structures can be automatically obtained tailored to several specific platforms, e.g. relational [19] or multidimensional [20].

Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 2, No. 1, 2008

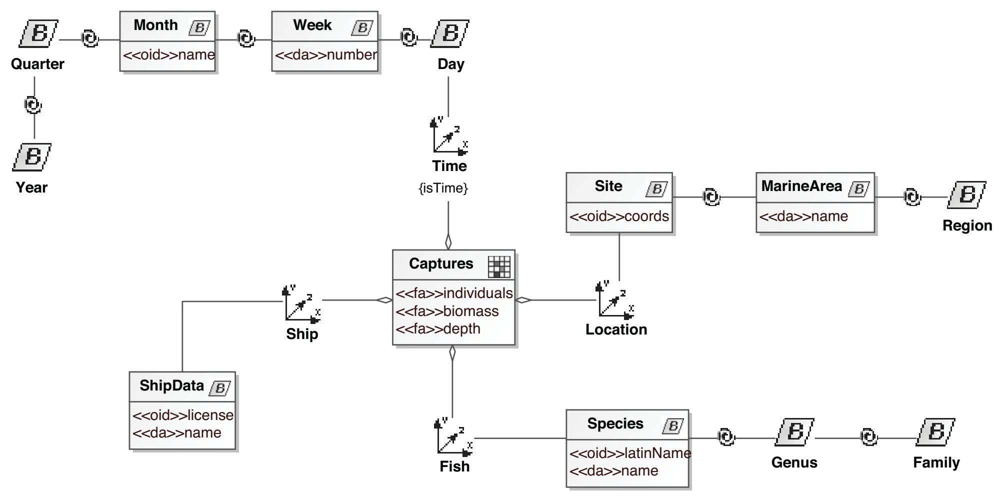


Fig. 2. The conceptual multidimensional model of the repository

3.2 Transformations for Model-driven Data Mining

Whilst the derivation of the data under analysis is traditionally performed through a three-step process, analysis techniques such as data mining present different requirements for their development. In this section, a model-driven engineering approach for the deployment of data-mining models together with the data under analysis is described.

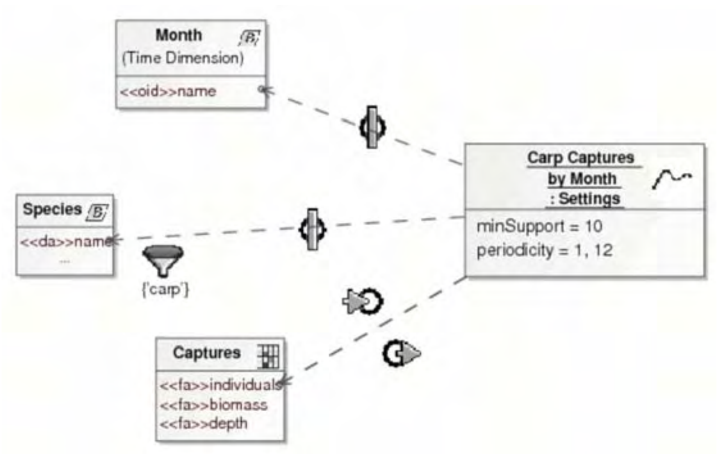


Fig. 3. The conceptual model of the time series analysis

The novelty of our approach is twofold: (i) it is based on defining vendor-neutral models of data-mining techniques together with the model of the underlying data warehouse, and (ii) the deployment of those data-mining techniques is done automatically. Therefore, on one hand, we use a modelling approach [6,7,8,9] for defining platform-independent models for several data-mining techniques. This approach is a high-level vendor-neutral modelling language to visually and easily specify analysis by means of applying data-mining techniques. Following the running example of Fig. 1, from the underlying multidimensional-data model of fish captures, an analyst can explore this model and specify its data-mining needs by the conceptual modelling of a time-series analysis in this case. Specifically, this conceptual model has been defined by using the UML extension presented in [9]. Therefore, a *carp captures by month* analysis for the *individuals of each captures* time series can be easily specified and carried out. The conceptual model defined for the required analysis is shown in Fig. 3. The analysis model also includes a time axis is specified in increments of *months* and a filter for the *carp specie*. Nevertheless, additional parameters of the data-mining technique can be also specified such as a *minimum data support* or a suggestion of the series *periodicity*.

On the other hand, this language is not directly implemented in any data-mining platform, and thus, it only acts as a blueprint of the executable analysis. Therefore, the model-transformation configuration has to be described in order to consider every kind of target platform from this platform-independent modelling language. In Fig. 4, we provide an overview of the required model-transformation architecture, stressing some of the current data-mining standards and platforms in the market.

The conceptual data-mining modelling framework in data warehouses (Zubcoff *et al.* [6,7,8,9]) is shown at the top of this model-driven architecture. Fig. 4 also shows the transformation paths to derive several implementations through mapping data-mining models to other languages that really have established an executable environment: CWM, XELOPES, DMX, or Weka acting as bridge. Depending to the characteristics of the analysis itself (*e.g.*, the required technique) or the data-mining solution available (*e.g.*, it can only be open-source platforms), we choose one of the transformation paths. Furthermore, Depending on the target-language representation, model-to-model or model-to-text transformations could be needed. Therefore, some of the data-mining solutions that are able to interpret the previous modelling languages are also represented in Fig. 4. On the lower side, some of the data-mining platforms are also represented. Whereas there are standards such as CWM that are vendor-neutral and many CWM-compliant tools can be considered (Oracle Miner, CWM4ALL, etc.), others such as DMX are commonly restricted to the platform for which they are originally were thought (the Microsoft's in this case).

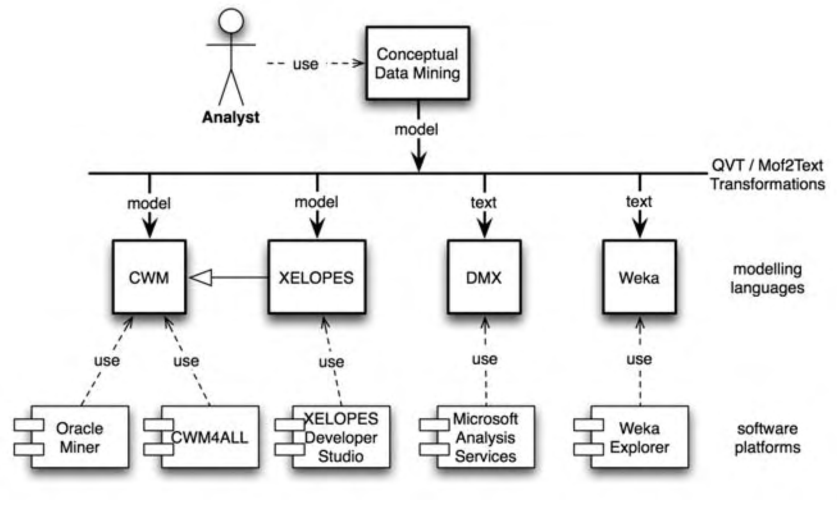
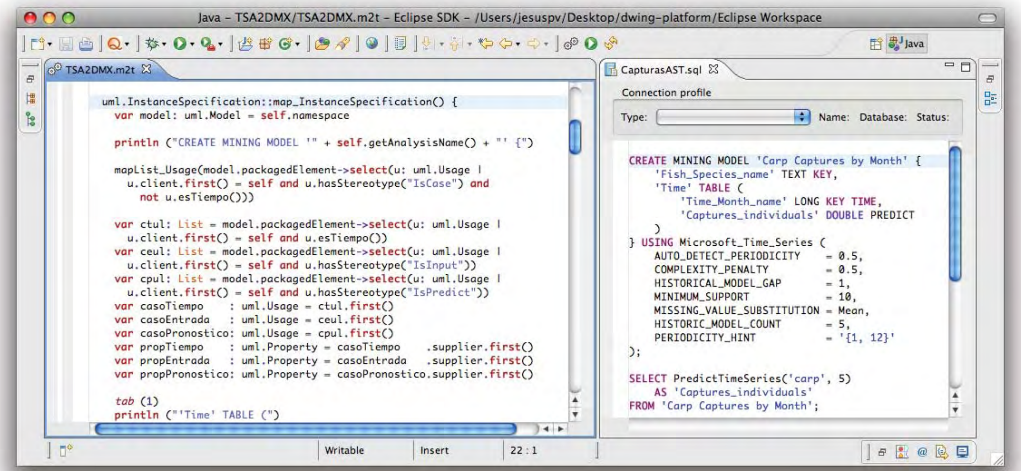


Fig. 4. Model-transformation architecture for data mining

From a technical point of view, we propose the usage of the "model-driven architecture" (MDA) [12] in order to implement these transformations between data-mining models. Within an MDA-based approach the "query/view/transformation" (QVT) language can be used as a standard mechanism for defining formal relations between MOF-compliant models that allows the automatic derivation of a implementation. Nevertheless, there are transformations that are applied from models (*i.e.*, MOF-based) to implement code (*i.e.*, textual modelling languages). In these cases, MDA offers the "MOF models to text transformation" (Mof2Text) language that allows us to specify transformations by means of textual templates in order to automatically derive the corresponding implementation.

3.3 Example Data-mining Transformation for a Specific Platform

Herein, we follow our running example for carrying out a time-series analysis of fish-species population (see Fig. 1), in order to automatically derive the implementation for a specific software platform. Specifically, in this paper, we employ the Microsoft Analysis Services[10] as the target platform. As we previously shown, this platform provides the DMX language to code data-mining models. Therefore, given the time-series analysis model of Fig. 1, we have designed the required mapping from this model into DMX code. Due to the space constraints, we exclude an abstract specification of the involved mapping, also omitting an example transformation of the data under analysis that can be found in [16]. Nevertheless, in Fig. 5, it is shown the implementation of this mapping (left-hand side) over the Eclipse development platform. In order to accomplish this task, we have used the MOFScript[11] plug-in for this platform. MOFScript is a transformation-language implementation of the Mof2Text standard language that enable us to specify model-to-text transformations in the "model-driven architecture" (MDA) [12] proposal. On the right-hand side, the resulting DMX code for the time-series analysis of Fig. 1 is shown.


Java – TSA2DMX/TSA2DMX.m2t – Eclipse SDK – /Users/jesuspv/Desktop/dwing-platform/Eclipse Workspace
Java
TSA2DMX.m2t
CapturasAST.sql
Connection profile
Type:
Name:
Database:
Status:
Writable
Insert
22 : 1

```
uml.InstanceSpecification::map_InstanceSpecification() {
  var model: uml.Model = self.namespace

  println ("CREATE MINING MODEL '" + self.getAnalysisName() + "' {")

  mapList_Usage(model.packagedElement->select(u: uml.Usage |
    u.client.first() = self and u.hasStereotype("IsCase") and
      not u.esTiempo()))

  var ctul: List = model.packagedElement->select(u: uml.Usage |
    u.client.first() = self and u.esTiempo())
  var ceul: List = model.packagedElement->select(u: uml.Usage |
    u.client.first() = self and u.hasStereotype("IsInput"))
  var cpul: List = model.packagedElement->select(u: uml.Usage |
    u.client.first() = self and u.hasStereotype("IsPredict"))
  var casoTiempo    : uml.Usage = ctul.first()
  var casoEntrada   : uml.Usage = ceul.first()
  var casoPronostico: uml.Usage = cpul.first()
  var propTiempo    : uml.Property = casoTiempo    .supplier.first()
  var propEntrada   : uml.Property = casoEntrada   .supplier.first()
  var propPronostico: uml.Property = casoPronostico.supplier.first()

  tab (1)
  println ("'Time' TABLE (")
```

```
CREATE MINING MODEL 'Carp Captures by Month' {
    'Fish_Species_name' TEXT KEY,
    'Time' TABLE (
        'Time_Month_name' LONG KEY TIME,
        'Captures_individuals' DOUBLE PREDICT
    )
} USING Microsoft_Time_Series (
    AUTO_DETECT_PERIODICITY    = 0.5,
    COMPLEXITY_PENALTY         = 0.5,
    HISTORICAL_MODEL_GAP       = 1,
    MINIMUM_SUPPORT            = 10,
    MISSING_VALUE_SUBSTITUTION = Mean,
    HISTORIC_MODEL_COUNT       = 5,
    PERIODICITY_HINT           = '{1, 12}'
);

SELECT PredictTimeSeries('carp', 5)
    AS 'Captures_individuals'
FROM 'Carp Captures by Month';
```

Fig. 5. Example of a Mof2Text transformation and the generated code

[10] URL: `www.microsoft.com/sql/solutions/bi` (March 2008)
[11] URL: `www.eclipse.org/gmt/mofscript` (March 2008)

Given Fig. 5, the mapping overview is as follows: each time-series analysis (represented by some kind of modelling element in the source metamodel)[12] is mapped into a data-mining model in DMX (`MINING MODEL` instruction). Every parameter of the analysis technique is also mapped into their DMX counterpart. In addition, the unspecified parameters in the conceptual model are later explicitly defined in the implementing code. On the other hand, each data under analysis (taken from the multidimensional model of the underlying data warehouse) is mapped into a data-mining attribute in DMX (by defining a table and then creating its corresponding column). Once the mapping is correctly established, the MOF-Script engine can interpret this one in order to translate a certain conceptual model of time-series analysis to the DMX code, and thus, implementing it in the Microsoft Analysis Services platform. Finally, within this solution, analysts can consult the data-mining results (see the bottom-side of Fig. 1) by visualising the obtained patterns and trends and extracting new knowledge from them.

4 Conclusion

Due to mathematical foundations of data-mining techniques, there are no formalised mechanisms to easily specify data-mining activities as a real software engineering process. In this paper, we propose a model-engineering approach for overcoming this limitation. On one hand, we provide a set of models to specify data-mining techniques in an vendor-neutral way that are close to the way of analysts thinking about data-mining (i.e. conceptual models). On the other hand, we provide transformations to automatically derive platform-specific models from the conceptual ones, altogether with the deployment of data under analysis [16,17]. Thus, analysts can only focus on data mining itself at an abstract level instead of distracting by details related to a certain vendor data-mining solution whilst the model transformations can automatically derive vendor-specific implementations in background for current data-mining platforms. Furthermore, our approach for data-mining modelling is also concerned about modelling the data under analysis, i.e. the data warehouse in order to provide analysts with a way of quickly understand for being close to their way of thinking about data. The data-mining techniques are smoothly integrated in this model.

Therefore, the great benefit of our approach is that, once we have established the model-driven architecture for both data under analysis and analysis techniques for data mining, analysts can model their data-mining related tasks easily in a vendor-neutral way whereas the model-transformations scaffolding is entrusted to automatically implement them in a certain platform.

Future Work. Our immediate future work covers other high-level mechanism to specify data-mining related tasks. For instance, we will study how current goal-oriented approaches for requirement analysis [21] can help us to guide the selection of data-mining solutions. In addition, we are investigating on the integration of the proposed data-mining framework together with the analysis technologies traditionally employed in the data-warehouse domain.

5 Acknowledgements

This work has been supported by the ESPIA (TIN2007-67078) project from the Spanish Ministry of Education and Science, and by the QUASIMODO (PAC08-0157-0668) project from the Castilla-La Mancha Ministry of Education and Science (Spain). Jesús Pardillo and Jose-Norberto Mazón are funded by the Spanish Ministry of Education and Science under FPU grants AP2006-00332 and AP2005-1360, respectively.

[12] Please, see [9] for an additional explanation of the modelling primitives involved in a time-series analysis.

References

1. Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann (1999)
2. Kimball, R., Ross, M.: The Data Warehouse Toolkit. Wiley (2002)
3. Inmon, W.H.: The Data Warehouse and Data Mining. Commun. ACM **49**(4) (1996) 83–88
4. Rizzi, S., Abelló, A., Lechtenbörger, J., Trujillo, J.: Research in data warehouse modeling and design: dead or alive? In: DOLAP. (2006) 3–10
5. Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press
6. Zubcoff, J.J., Trujillo, J.: A UML 2.0 profile to design Association Rule mining models in the multidimensional conceptual modeling of data warehouses. Data Knowl. Eng. **63**(1) (2007) 44–62
7. Zubcoff, J.J., Pardillo, J., Trujillo, J.: Integrating Clustering Data Mining into the Multidimensional Modeling of Data Warehouses with UML Profiles. In: DaWaK. (2007) 199–208
8. Zubcoff, J.J., Trujillo, J.: Conceptual Modeling for Classification Mining in Data Warehouses. In: DaWaK. (2006) 566–575
9. Pardillo, J., Zubcoff, J., Trujillo, J.: Un perfil UML para el análisis de series temporales con modelos conceptuales sobre almacenes de datos. In: IDEAS Workshop. (2007) 369–374
10. González-Aranda, P., M.E.M.S.S.J.: Towards a Methodology for Data mining Project Development: The Importance of abstraction. In: In: ICDM Workshops (FDM). (2004) 39–46
11. Bézivin, J.: Model Driven Engineering: An Emerging Technical Space. In: GTTSE. (2006) 36–64
12. Object Management Group: Common Warehouse Metamodel (CWM), Unified Modeling Language (UML), Model Driven Architecture (MDA), Query/View/Transformation Language (QVT), MOF Model to Text Transformation Language (Mof2Text). `http://www.omg.org` (March 2008)
13. CRISP-DM Consortium: CRISP-DM, version 1.0. `http://www.crisp-dm.org/` (may 2008)
14. Pardillo, J., Trujillo, J.: Integrated Model-driven Development of Goal-oriented Data Warehouses and Data Marts. (2008) In Press
15. Pardillo, J., Mazón, J.N., Trujillo, J.: Model-driven OLAP Metadata from the Conceptual Models of Data Warehouses. (2008) In Press
16. Mazón, J.N., Trujillo, J.: An MDA approach for the development of data warehouses. Dec. Support Syst. **In Press** (2007)
17. Mazón, J.N., Trujillo, J., Lechtenbörger, J.: Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. Data Knowl. Eng. **63**(3) (2007) 725–751
18. Luján-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. Data Knowl. Eng. **59**(3) (2006) 725–769
19. Mazón, J.N., Trujillo, J., Serrano, M., Piattini, M.: Applying MDA to the development of data warehouses. In: DOLAP. (2005) 57–66
20. Mazón, J.N., Pardillo, J., Trujillo, J.: Applying Transformations to Model Driven Data Warehouses. In: DaWaK. (2006) 13–22
21. Mazón, J.N., Pardillo, J., Trujillo, J.: A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses. In: ER Workshops. (2007) 255–264

6 Appendix A. Code of the model-transformation example

```
texttransformation AST2DMX (in uml:"http://www.eclipse.org/uml2/2.0.0/UML") {
  uml.Model::main() {
    file (self.name + ".sql")
    println ("-- Generated by MOFScript (" + date() + " " + time() + ")\n")
    mapList_InstanceSpecification(self.packagedElement
      ->select(is: uml.InstanceSpecification | is.hasStereotype("AnalisisST")))
  }
  module::mapList_InstanceSpecification(isl: List) {
    var fis: uml.InstanceSpecification = isl.first()
```

```
fis.map_InstanceSpecification()
    isl.remove(fis)
    isl->forEach(is: uml.InstanceSpecification) {
      newline (1)
      is.map_InstanceSpecification()
    }
  }
  uml.InstanceSpecification::map_InstanceSpecification() {
    var model: uml.Model = self.namespace
    println ("CREATE MINING MODEL '" + self.getAnalysisName() + "' {")
    mapList_Usage(model.packagedElement->select(u: uml.Usage |
      u.client.first() = self and u.hasStereotype("IsCase") and
        not u.esTiempo()))
    var ctul: List = model.packagedElement->select(u: uml.Usage |
      u.client.first() = self and u.esTiempo())
    var ceul: List = model.packagedElement->select(u: uml.Usage |
      u.client.first() = self and u.hasStereotype("IsInput"))
    var cpul: List = model.packagedElement->select(u: uml.Usage |
      u.client.first() = self and u.hasStereotype("IsPredict"))
    var casoTiempo    : uml.Usage = ctul.first()
    var casoEntrada   : uml.Usage = ceul.first()
    var casoPronostico: uml.Usage = cpul.first()
    var propTiempo    : uml.Property = casoTiempo    .supplier.first()
    var propEntrada   : uml.Property = casoEntrada   .supplier.first()
    var propPronostico: uml.Property = casoPronostico.supplier.first()
    tab (1)
    println ("'Time' TABLE (")
tab (1)
    propTiempo.map_Property()
    println (",")
    tab (1)
    propEntrada.map_Property()
    newline (1)
    tab (1)
    println (")")
    var sl: List
    sl.clear()
    self.slot->forEach(s: uml.Slot) {
      if (not s.definingFeature.name.equals("numPeriodos")) {sl.add(s)}
    }
mapList_Slot(sl)
newline (1)
var cl: List    = model.packagedElement->select(c: uml.Constraint)
var c : uml.Constraint = cl.first()
var b : Boolean = true // not c.constrainedElement->select(u:uml.Usage).isEmpty()
print ("SELECT PredictTimeSeries(")
if (b) {      print (c.specification.body.first() + ", ")}
var sl:List = self.slot->select(s:uml.Slot | "numPeriodos".equals(s.definingFeature.name))
var s : uml.Slot = sl.first()
println (s.value.first().value + ")")
tab (1)
```

```
println ("AS '" + propPronostico.class.name + "_" + propPronostico.name + "'")
println ("FROM '" + self.getAnalysisName() + "';")
  }
  module::mapList_Usage(ul: List) {
    var fu: uml.Usage = ul.first()
    fu.map_Usage()
    ul.remove(fu)
    ul->forEach(u: uml.Usage) {
      println (",")
      u.map_Usage()
    }
    println (",")
  }
  uml.Usage::map_Usage() {
    var p: uml.Property = self.supplier.first()
    p.map_Property()
  }
  uml.Property::map_Property() {
    var c: uml.Class = self.class
    var s: String = c.name + "_" + self.name
    tab (1)
    if (c.hasStereotype("Fact")) {
      print ("'" + s + "' DOUBLE PREDICT")
    } else if (c.getDimension().getValue("Dimension", "isTime")) { // assumes base
      print ("'" + c.getDimension().name + "_" + s + "' LONG KEY TIME")
    } else {
      print ("'" + c.getDimension().name + "_" + s + "' TEXT KEY")
    }
  }
  module::mapList_Slot(sl: List) {
    var fs: uml.Slot = sl.first()
    println ("} USING Microsoft_Time_Series (")
    fs.map_Slot()
    sl.remove(fs)
    sl->forEach(s: uml.Slot) {println (","); s.map_Slot()}
    newline (1)
    println (");")
  }
  uml.Slot::map_Slot() {
    var name: uml.LiteralString = self.definingFeature.name
    tab (1)
    if ("autoPerÃodo".equals(name)) {
    } else if ("complejidad".equals(name)) {
    } else if ("ventana".equals(name)) {
    } else if ("minSoporte".equals(name)) {
      'MINIMUM_SUPPORT = ' self.value.first().value
    } else if ("valoresAusentes".equals(name)) {
    //} else if ("numPeriodos".equals(name)) {
    } else if ("periodo".equals(name)) {
      'PERIODICITY_HINT = '
      mapList_LiteralUnlimitedNatural(self.value)
    }
  }
  module::mapList_LiteralUnlimitedNatural(lunl: List) {
    var flun: uml.LiteralUnlimitedNatural = lunl.first()
    "\'{"
    flun.value
    lunl.remove(flun)
    lunl->forEach(lun: uml.LiteralUnlimitedNatural) {
      ', ' lun.value
    }
    "}'"
  }
  uml.InstanceSpecification::getAnalysisName(): String {
    if (self.name <> null) {
      result = self.name
    } else if (self.ownedComment.first().body <> null) {
      result = self.ownedComment.first().body
    } else {
      result = "AST_" + date() + "_" + time()
    }
  }
  uml.Usage::esTiempo(): Boolean {
    if (self.hasStereotype("Caso")) {
      var p: uml.Property = self.supplier.first()
      result = p.class.getDimension().getValue("Dimension", "isTime") // assumes a base
    } else {
      result = false
    }
  }
  uml.Class::getDimension(): uml.Class { // assumes a base
    self.ownedAttribute->forEach(p: uml.Property) {
      if (p.name <> null) {
        if (p.name.equals("D")) {
          result = p.opposite.class.getDimension() // assumes opposite
          break
        }
      } else {
        if (p.opposite <> null) {
          var c: uml.Class = p.opposite.class
          result = p.opposite.class // assumes stereotyped by Dimension
          break
        }
      }
    }
  }
}
```