An Integration of Rule Types and XML

Harindra Wijesekera
Hewlett Packard, harindra_wijesekera@hp.com

James A. Sykes
Swinburne University of Technology, jsykes@swin.edu.au

An Integration of Rule Types and XML

Harindra Wijesekera[a] and James A. Sykes[b]

[a]E-Solutions Division
Hewlett Packard, Sydney, Australia
harindra_wijesekera@hp.com

[b]School of Information Technology
Swinburne University of Technology, Melbourne, Australia
jsykes@swin.edu.au

Abstract

Meta data types for providing business rules are described. These business rules form part of the knowledge base of an organisation. The integration of knowledge with data and information is discussed. FORM, a natural language based, conceptual modelling technique is used to describe the meta data types.

XML has been recognised by many in the industry as the preferred data interchange language. XML is currently being used to interchange data and its schema. The use of XML to interchange rules is being investigated, in this paper. This will enable organisations to interchange business rules and their meta data, in addition to data and their schema. Such business rules can be interpreted and applied by the receiving systems, thus providing a basis for intelligent behavior when dealing with information being interchanged

Keywords

Modelling languages, Knowledge-based software, Requirements elicitation, Object-role modelling, Extensible Markup Language (XML), Unified Modelling Language (UML), Meta-modelling

INTRODUCTION

Data that is being generated at each business unit is now recognised as important for strategic decision-making. No longer can it be considered in isolation. Information that was treated previously as tactical or operational is now required for enterprise-wide re-use and decision-making. The globalization and internationalization of industries and ever-changing economic conditions are the main reasons for organisations to attempt leverage their information assets.

The use of meta data has been recognised by Halpin (1995), Campbell et al., Halpin (2000), Meta Data Coalition (1999), Open Information Model (1999) as critical for the management of information assets of a business. The Open Information Model (OIM) is a vendor neutral and technology independent specification. Unified Modeling Language (UML) is used as the formal specification language to provide core meta data types for enterprise level data. The OIM has recognised business rules as an important part of this specification. However, the UML based approach has deficiencies in the areas of natural language expression and representing associations based on unary relationships Arlow et al. Also various implementation level details are captured in the OIM, i.e., it is not a conceptual model. It is necessary that conceptual models are not constrained by implementation level details. They should only include conceptually relevant aspects, both static and dynamic, of the domain (ISO/TR 9007 1987). In this paper, rules are considered as part of the corporate knowledge. Also we have taken an alternative approach by employing the conceptual modelling language known as Formal Object Role Modelling (FORM) (Halpin 1995).

In a knowledge economy, it is important to combine traditional transaction-dependent systems development methods and techniques with knowledge-intensive techniques that are found in knowledge based systems development (Clancey 1983, Jih 1990) thus creating systems that could cope with both high transaction dependencies and knowledge. In this paper, knowledge is being incorporated in the form of *IF...THEN* rules.

XML (W3C Extensible Markup Language XML 1.0), has been recognised by the industry as the preferred data interchange language. Most computer systems today are capable of receiving and executing XML based formats. XML has been enhanced to provide a standard for the structure of data (known as XML Schema) being interchanged between computer systems. XML schema provides type level data of information being interchanged, including business data types (Customer, Product, Address etc.), data constraints and data concepts (hierarchical information for example).

The main objective of this paper is to provide FORM based meta data types to describe rules that occur in business. In addition, however, this work also describes extensions to XML schema by providing for the exchange of type level information about rules between computer systems. This will facilitate an environment where data, information and knowledge can be exchanged between business-to-business level applications.

The section on Business Rules provides an overview and an interpretation suitable for a fact-oriented approach. The section on Fact Oriented Representation introduces the required object types and describes the formation of rule types. The next section provides examples of rules that are described in a fact-oriented manner. A procedure for describing rules, during the requirements gathering activity, is then introduced that is followed by an XML representation of rules, illustrated with some brief examples. The final section lists conclusions and some propals for future work.

BUSINESS RULES

The use of rules to capture and document knowledge from domain experts is not a new concept. Rules as a technique for capturing knowledge became popular after the MYCIN project (Clancey 1983). The success of MYCIN has given rise to numerous research and product development activity in the area of knowledge based (en-coded) systems.

Rules that require arguments are more complex than other rules such as the definition of a term or a fact rule. The definition of terms and fact rules are stated as a part of the information base (Open Information Model 1999, Halpin 1995). Rules that are based on arguments are usually provided as text in modeling environments. The work based here deals with rules that contain arguments.

There are four parts to a rule.

- **IF part**
- **THEN part**
- **ELSE part**
- **NOTE part**

The IF part is known as the antecedent or the action part of the rule. This part is used to test the truth value(s) of available information. Checking for the “availability of a customer credit limit” is an example. An antecedent can test availability of instances and/or whether a given value of an instance conforms to specified criteria. The critical element of linking fact types and rules in the meta schema introduced in this paper is in the notion that fact instances, being propositions, have truth values. This maintains the convention of FORM, that the absence of an instance of a fact type from an information base means the proposition is false. Even though rules are specified at a type level the execution of rules are performed at an instance level. Fact types, (e.g. **Employee (Employee Id) ‘’ ...has Service Period (Number) ‘’)** will be used to provide input to the process of defining rule types. Fact types form part of an information base. Fact types are described with the use of Conceptual Schema Design Procedure (CSDP), as documented in Halpin (1995). An information base consists of various object types, relationships and instances.

The THEN and ELSE parts are known as the consequent or the result(s) of a rule. Setting the credit status of a customer to be “No more Credit Allowed” is an example. Consequents may be in the form of creating new fact instances, varying or deleting any available information of one or more fact instances.

The NOTE part of a rule can be used to provide some descriptive information or justifications to a rule. This can be useful at the time of rule interrogation and explanation.

The IF and THEN parts are mandatory. They provide the basic structure of any rule. The ELSE and NOTE parts are optional. The ELSE part helps knowledge analysts to provide alternative results (or courses of action) if the IF part is not true. It will assist the knowledge analyst in the requirements engineering activity by eliminating the need to provide additional rules which are redundant within the model.

A typical example of a rule is as follows:

IF (a == b AND c == d OR e == 1) THEN (e == 2 AND c == f)

In the above example a,b,c etc. are data items. They can be of any type – e.g., lexical or non-lexical objects, Numeric, Date, Values, List of values, Boolean and strings. Every item is viewed independently. Any relationships and associations that may exist between them are not defined as part of the rule, although they may be defined elsewhere. That is why traditionally rules are considered data oriented (EXSYS 1996, Ross 1994, The Business Rules Group 2001) as they are driven from items presented in rules.

FACT ORIENTED REPRESENTATION

The work described in this paper is based on representing rules in a fact-oriented manner while maintaining the basic structure of a rule described previously. The work of Debenham (1989, Vol.15) and Fullerton and James (1987) has also been carried out in the areas of capturing and modelling business rules. Debenham has used dependency diagrams to represent business rules where Fullerton and James have used facts to represent rules. Pierce, Creasy and Schouten have taken an approach where rules are defined in the form of logical expressions and combining them with NIAM fact types.

In this paper, in comparison with approaches suggested in the previous paragraph, FORM is being employed to deal with both information and knowledge.

FORM is a natural language based conceptual modeling method based on Object Role Modeling (ORM), Haplin (2000). In comparison with other popular methods such as Unified Modelling Language - UML (Quatrani 1998) FORM is known in the industry for its use of natural language expression (Arlow et al., Halpin 2000). The natural language and role based grammar coupled with FORM's ability to describe unary relationships has reduced the complexity of the suggested model and increased the comprehensibility of rules being described among domain users. The rule types described in this paper are tightly integrated with fact types, thus providing improvements over ORM based architectures such as Universal Architecture (UA) as described by Bollen, Nijssen.

The primary focus has been on requirements gathering at a conceptual level with no implementation level details stated in the model. A modeling language based on the power of natural language expression was selected, with a view to maintain simplicity and increase comprehensibility. Even though Fullerton and James have used fact types to represent rules, their approach does not conform to the previously described rule structure and they have not provided a formal approach. Debenham's approach is based on Binary-Relationship modelling and Horn clause logic. Relative to the approach described here, there is less comprehensibility because there is less use of natural language expression. The approach of Pierce, Creasy and Schouten also suffers from lack of comprehensibility and complexity with the use of logical expressions and fact types.

The form of representation described in this paper is important to provide declarative models that combine rules and facts into a single conceptual model. Both rules and facts are described using a common language. The natural language based modelling language has increased the comprehensibility of the model by providing a flexible framework to express requirements. The ability to describe rules based on fact types is important in the suggested approach, as it provides a tight integration between information and knowledge.

As illustrated in Figure 1, meta data types are introduced to assist the knowledge analyst in the process of gathering, documenting and refining rules that are found in organisations, relevant for business applications. The meta data model has been produced by applying the Conceptual Schema Design Procedure (CSDP) described by Halpin (1995).

Each antecedent section of a rule may consist of one or many arguments. These arguments are combined with 'AND' or 'OR' logical connectors. This structure applies to consequent, else-consequent, and input sections as well. The input section is used to supply object types that are required to interrogate fact instances at the time of executing rule types. The non-lexical meta-object type Position in Figure 1 allows the component parts of a rule to be described in a pre-defined sequence for clarity. It also maintains a unique number for each type. The non-lexical meta-object type Note is used to provide additional narrative information about the rule.

The object types described below are necessary to provide rule types. The arguments of a rule type will usually be defined with the use of fact types, as described previously. However, if fact types are not available they must first be defined by applying the CSDP. It is possible to discover new fact types and/or modify existing fact types during the process of expressing rule types in the model. The consequent and else-consequent sections work in a similar manner, i.e., either by linking existing fact types or defining new ones.

Constraints

The constraints C1, C2, C3, C15, and C16 are ORM uniqueness constraints, which express the requirement that the set of roles covered by the constraint arrow should be unique for each instance of the fact type. The remaining named constraints in Figure 1 (C4 - C14) are ORM mandatory role constraints, which express the requirement that each object type marked with such a constraint must have an instance of the associated fact type recorded for it.

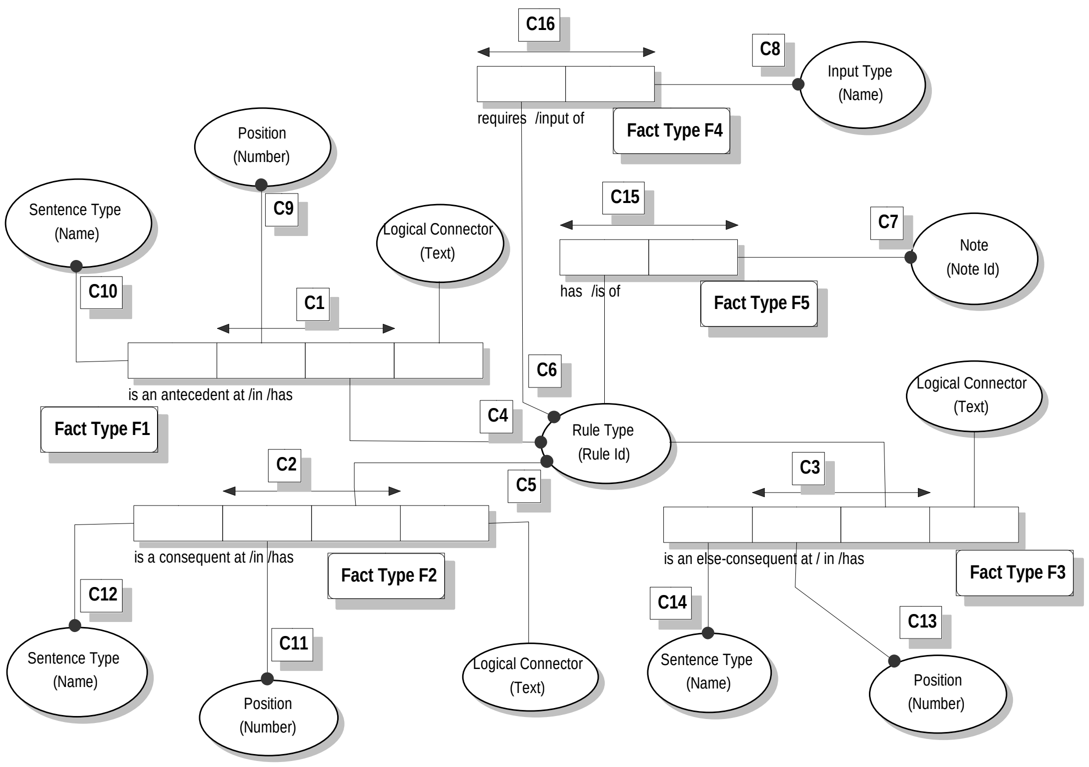

Figure 1: A Meta Schema diagram for Rule Types

Object Types

Rule Type – This object type is necessary to provide a unique identification for each rule type.

Sentence Type – Describes the manner in which arguments are formed. This applies to consequent and else-consequent as well.

A sentence type is a result of an association between one or more fact types object types, and/or values. In other words, sentence types are to be populated using fact types from the information base. It is assumed that all fact types are created as a result of applying the CSDP.

Following is an example of a Sentence Type:

Contribution Period (Number) ‘’ ...of Employee (Employee Id) ‘’ ...with a Condition Type of (Kind) ‘Greater than’ ...has Service Period (Number) ‘’ ...of Employee (Employee Id) ‘’

It is constructed from the following fact types:

Employee (Employee Id) ‘’ ...has/of... Contribution Period (Number) ‘’

Employee (Employee Id) ‘’ ...has/of... Service Period (Number) ‘’

As illustrated above, sentence types provide tight integrations by linking various artifacts from an information base.

Position – The position object type allows a sequence to be described for the parts of a rule.

Input – Used to describe the types of input that may be required, when a rule is being executed. Each rule should be supplied with one or many input details. As described earlier, rules are defined at a type level even though they are executed at an instance level and will require object types such as Customer Number and instance details (Customer Number ”123”, for example).

Note – This object should provide additional pieces of text or descriptive information.

Logical Connector – This object type provides a logical connector (AND/OR) that can be used to specify logical connections that may exist between arguments. Sections such as Antecedent, Consequent and Input may contain one or many arguments.

Refer to Figure-1 for a description of fact types and constraints placed on them.

AN EXAMPLE OF A FACT ORIENTED RULE

We will begin this rule gathering process by describing a requirement. Suppose that domain experts have specified a rule that limits customer borrowing over a specified credit line, as follows.

If Borrower's Cumulative Borrowing is greater than available Credit Line, then Borrower Credit Status should be set to 'No more credit allowed'.

The following fact types should be available in the information base. If not available they should be created by applying CSDP.

F1- Borrower (id) '' ... has/of… Cumulative Borrowing (Amount) ''

F2- Borrower (id) '' ... has/of... Credit Line (Amount) ''

F3- Borrower (id) '' ... has/of... Credit Status (Status)

The rule that is being examined consists of an antecedent and a consequent. They can be described using the meta schema as:

Sentence Type (Name) 'Cumulative Borrowing (Amount) '' ...of Borrower (id) '' ...with a Condition Type of (Kind) 'Greater than' Credit Line (Amount) '' ...of Borrower (id) ''' ...is an antecedent at **Position (Number)** '1' ...in **Rule Type (Rule Id)** 'R-1'

Sentence Type (Name) 'Borrower (id) '' ...has Credit Status (Status) {No more credit allowed} ...is of Condition Type (Kind) 'True'' ...is a consequent at **Position (Number)** '1' ...in **Rule Type (Rule Id)** 'R-1'

Input Type (Name) 'Borrower (id) ''' ...input of **Rule Type (Rule Id)** 'R-1'

Example 1:

For a given Borrower, it is necessary to check whether fact instances are available for fact types F1 and F2. If instances are available and the specified condition (Cumulative Borrowing Amount greater than available Credit Line) is true, the consequent actions will be performed. If false, the else-consequents actions will be performed. In this example, a new fact instance will be created or an existing one will be updated in the information base at the completion of the antecedent.

Various condition types such as >, <, =, >=, <=, IS, IS NOT etc. can be used to express complex operators in the antecedent and/or consequent of a rule. The process of working with such rules is similar to that described in the previous example.

Rules where other fact types are derived (Halpin 1995) can also be expressed with the use of the meta schema. Each derived fact type can be supported by a rule type that is closely associated with other available fact types. The following business rule is reproduced from Halpin (1995, page 394).

Agent can sell Car Type for Company, if "Agent sells Car Type and Agent represents Company and Company makes Car Type". Rule is expressed as follows.

Sentence Type (Name) 'Agent (id) '' ...sells Car Type (id) '' ...with a Condition Type of (Kind) 'True'' ...is an antecedent at **Position (Number)** '1' ...in **Rule Type (Rule Id)** 'R-2' ...has **Logical Connector (Text)** 'AND'

Sentence Type (Name) 'Agent (id) '' ...represents Company (id) '' ...with a Condition Type of (Kind) 'True'' ...is an antecedent at **Position (Number)** '2' ...in **Rule Type (Rule Id)** 'R-2' ...has **Logical Connector (Text)** 'AND'

Sentence Type (Name) 'Company (id) '' ...makes Car Type (id) '' ...with a Condition Type of (Kind) 'True'' ...is an antecedent at **Position (Number)** '3' ...in **Rule Type (Rule Id)** 'R-2'

Sentence Type (Name) 'Agent (id) '' ...sell Car type (id) '' for Company (id) '' ...is of Condition Type (Kind) 'True'' ...is a consequent at **Position (Number)** '1' ...in **Rule Type (Rule Id)** 'R-2'

Input Type (Name) 'Agent (id) ''' ...input of **Rule Type (Rule Id)** 'R-2'

Input Type (Name) 'Company (id) ''' ...input of **Rule Type (Rule Id)** 'R-2'

Input Type (Name) 'Car Type (id) ''' ...input of **Rule Type (Rule Id)** 'R-2'

Example 2:

There are three arguments for the antecedent section. Each argument is described using fact types that represent associations between Agent, Company and Car Type. They are also combined using "AND" logical connectors. Object types of 'Agent', 'Company' and 'Car Type' will be provided as input to the rule. They will be required to work with appropriate fact instances. The consequent is a new ternary fact type, which would represent an association between Agent, Company and Car Type. For this to eventuate, instances should be available for given fact types, as determined by the "True" Condition Type.

PROCEDURE FOR DESCRIBING RULES

Once a rule is discovered the following process will be adopted in the requirements gathering activity for describing rule types;

- Write the Rule as a piece of text
- Identify required Sections (antecedent, etc.)
- Identify Entity Type(s) that are required for the rule. Domain specific information plays a major role in identifying object types required for a rule. In a customer oriented domain 'Customer' and in an employee oriented domain 'Employee', for example. The reference modes of Customer (Customer id) and Employee (Employee id) will be used for identification of instances.

Examine the Antecedent Section

(Following has to be performed for each argument in the Antecedent section)

-1 Are Object Types and Fact Types available in the Fact Base to describe Sentence Type?
-2 If not available apply CSDP to establish object types and Fact types.
-3 Examine Sentence Type
-4 Ensure Fact Types are related to above described Entity Type(s)
-5 Populate Sentence Type by using sample instance(s) from the above-described Entity Type
-6 If not possible to populate the sentence type should review the following to ensure they are valid;
    Sentence Type,
    Entity Types,
    Fact types and their constraints
    Apply CSDP again, if necessary
-7 Are Object types valid for the required Condition Type?
    (Condition Types that can be used are: <,>=,<=,=,<>,IS,IS NOT,TRUE,FALSE,INTERSECTION,UNION)
-8 If not go back to -3

Examine the Consequent Section

(Following has to be performed for each result in the Consequent Section)

-1 Are Object Types and Fact Types available in the Fact Base to describe Fact Type?
-2 If not available apply CSDP to establish object types and Fact types
-3 Examine Sentence Type
-4 Ensure Fact Types are related to above described Entity Type(s)
-5 Populate Sentence Type by using instance(s) from the above-described Entity Type
-6 If not possible to populate the sentence type should review the following to ensure they are valid;
    Sentence Type,
    Entity Types,
    Fact types and their constraints
    Apply CSDP again, if necessary
-7 Are Object types valid for the required Condition Type?
    (Condition Types that can be used are: <,>=,<=,=,<>,IS,IS NOT,TRUE,FALSE,INTERSECTION,UNION)
-8 If not go back to -3

Examine the Else-Consequent Section

Steps required for Else-Consequent is the same as Consequent

Examine the Input Section

The Input Section should reflect Entity Type(s) that are required for the execution of the rule. They should provide sufficient details to interrogate fact instances and generate new fact instances or modify its values such as Status codes.

The validity of input types can be checked by simulating the execution of a rule. If the knowledge engineer is unable to find valid instances from the information base it may be necessary to re-examine the antecedent section, the consequent section, the fact types and their constraints.

Examine the Note Section

Note is just a simple piece of text that can be used to provide additional descriptions of a rule. The knowledge analyst can provide further information and/or clarification.

XML REPRESENTATION OF RULES

With the growth of the Internet (Berners-Lee 1999), XML based interchange formats have emerged as the preferred method of exchanging data between entities such as individuals, corporations and governments. XML is researched, developed and advocated by the World Wide Web Consortium (W3C) as a non-vendor centric data interchange format.

It would be useful for corporations to interchange not just items of data but also business rules in XML format. This includes schema definitions of rules and rule instances. Such rules can be applied by various parties involved in a business process at the point of business interactions to ensure conformance to prevailing business conditions. This will also provide a required level of intelligence for XML based transactions. It is envisaged that XML processors will be developed to parse and process such rule based interactions.

Bird et al., have described an algorithm to map ORM based diagrams to XML schemas. The XML schema can then be used to generate XML instances. The mapping algorithm described by Bird et al., is being used to generate XML schemas for rule types and rule instances. There are three major steps to this algorithm. However, the existing algorithm is inadequate for describing rule types and instances. Hence extensions are suggested for the incorporation of rule based meta schema.

Step 1: Generate type definitions for each ORM object type

As a pre-requisite for this step, it is necessary to identify all major object types. Major object types are defined as ‘key concepts’ that are most important in a conceptual model. Such object types are selected by identifying the most important participant in some fact types (Campbell et al.,). As described in Figure-1, the major object type for meta schema diagram is Rule Type. This has been determined with the application of rules specified by Bird et al.

As described (W3C XML Schema Part 0), two types of data are required to specify XML schemas. ORM reference types and reference modes are represented in XML schema as ’simple types’ (may include value or range constraints) and ORM entity types are represented as ‘complex types’. Following describes simple and complex types, the required constraints for rule based meta schema, as illustrated in Figure- 1.

```
<?xml version=”1.0”?>
<schema targetNamespace=http://www.swin.edu.au/rule  xmlns=”http://www.swin.edu.au/rule”
xmlns:rule=http://www.swin.edu.au/rule

<!—simple types -->
<simpleType name=”PositionNumber” base=”integer”>
            <minInclusive value=”1”/>
            <maxInclusive value=”*”/>
</simpleType>
<simpleType name=”LogicalConnector” base=”string”/>
<simpleType name=”SentenceTypeName” base=”facttype”/>
<simpleType name=”NoteId” base=”string”/>
<simpleType name=”InputTypeName” base=”entity”/>
<simpleType name=”RuleTypeId” base=”string”/>
```

```
<!—complex types -->
<complexType name=”SentenceType”>
            <attribute name=”name” type=”rule:SentenceTypeName” minOccurs=”1”/>
</complexType>
<complexType name=”Note”>
            <attribute name=”name” type=”rule:NoteId” minOccurs=”0”/>
</complexType>
<complexType name=”InputType”>
            <attribute name=”name” type=”rule:InputTypeName” minOccurs=”1”/>
</complexType>
<complexType name=”RuleType”>
            <attribute name=”name” type=”rule:RuleType” minOccurs=”1”/>
</complexType>
```

‘SentenceTypeName’ is the reference mode of an entity type which may be a result of an association between one or more fact types, and/or values. The ‘base’ value of ‘SentenceTypeName’ should reflect this criterion. In order to provide N-ary relationships and nesting of fact types a ‘base’ value of ’ fact type’ is being introduced as an extension to the original algorithm. This will facilitate the formation of arguments and results of a rule type. Also a ‘base’ value of ‘entity’ is being introduced to represent various entity types that may exist in the model. It is required to describe input that is necessary for execution of rule types.

Step 2: Build complex type definitions for each major fact type grouping

Once major object type(s) are identified, other fact types will be mapped as sub-elements of the identified object type. Rule type has already been identified as the major object type of the meta schema diagram. The following XML fragment represents fact type F1 from Figure 1. The other fact types can be similarly represented.

```
<complexType name=”RuleFacts” base=”rule:RuleType” derivedBy=”extension”>
            <element name=”antecedent” minOccurs=”1” maxOccurs=”*”>
            <complexType>
            <element name=”antecedentSection” type=”rule:SentenceType”/>
            <element name=”antecedentPosition” type=”rule:PositionNumber”/>
            </complexType>
            </element>
            .
            .
            .
</complexType>
```

Step 3: Create a root element for the whole schema and add keys and key references

XML documents are traditionally based on root nodes. Therefore, it is necessary to declare a root node for proper XML representation. The root node for rule types is being declared as “RuleMetaSchema”. As per Bird et al., a sub element is being created for each major fact type group, underneath the declared root. Following describes uniqueness constraint for fact type F1. Uniqueness constraints required for other fact types can also be represented in a similar manner

```
<element name=”RuleMetaSchema”>
            <element name=”Rule” type=”RuleFacts” minOccurs=”0” maxOccurs=”*” />
</element>

<!—keys and uniqueness constraints -->
<!—keys are not required in Figure-1, therefore not described here -->
<!—uniqueness constraints are described as follows -->

<unique name=”antecedentKey”>
            <selector>RuleMetaSchema/Rule</selector>
            <field>@RuleId</field>
            <field>antecedentSentence/@name</field>
            <field>antecedentPosition/@PositionNumber</field>
</unique>
```

At the completion of steps 1 to 3 an XML based schema is available for Rule Types. Some extensions have also being described to cater for unique representation requirements of rule types. The final step in this process would be to populate schema using previously described examples.

A rule can be populated at two levels, namely, type level and instance level. The type level describes the antecedents, consequents, else-consequents and input types at a fact type level, whereas the instance level describes rules at fact instance level involving specific entity instances, e.g., Borrower, Company, Agent etc. An example XML based type and instance level population described below.

```
<Rule id="R-1">
<antecedent>
  <SentenceType>
    <Borrower>
            <id></id>
            <CumulativeBorrowing></CumulativeBorrowing>
            <ConditionType>Greaterthan</ConditionType>
            <CreditLine></CreditLine>
    </Borrower>
  </SentenceType>
  <Position Number="1"/>
</antecedent>
<consequent>
  <SentenceType>
    <Borrower>
            <id></id>
            <CreditStatus> No more credit allowed </CreditStatus>
            <ConditionType> True</ConditionType>
     </Borrower>
  </SentenceType>
  <Position Number="1"/>
</consequent>
<input>
  <Borrower>
            <id></id>
  </Borrower>
</input>
</Rule>
```

The following XML fragment provides instance level details for the sentence type in the antecedent part of the rule.

```
     <Borrower>
            <id>001</id>
            <CumulativeBorrowing>25000</CumulativeBorrowing>
            <ConditionType>Greaterthan</ConditionType>
            <CreditLine>24000</CreditLine>
    </Borrower>
```

The instance models described above, as a pre-requisite, require domain details to be described as fact types and fact instances. This activity should be performed with the use of the CSDP. In relation to the above examples, fact and instance models should be available for associations that exist in Borrower, Agent, Company and Car Type domains. The above examples describe the ways in which fact types should be used in forming Sentence Types.

CONCLUSIONS AND FUTURE WORK

This paper has introduced an ORM based method for representing the kinds of rules that are found in businesses. The rules that are found in businesses are of type derivations and/or constraints (Martin & Odell 1997, Meta Data Coalition 1999). This also falls into line with the notion of describing systems at a declarative level [the "what" aspect] as against the idea of writing numerous procedures [the "how" aspect] (Date 2000).

The ORM based approach to defining rules is advantageous due to the level of integration that it permits with the fact base and instance base. This was illustrated by Figure 1 and the use of examples 1 and 2. Not only do rules depend on fact types, entity types and values, they also rely on accurate definitions (constraints, for example) of each fact type. The natural language capabilities of ORM also provide a good foundation for defining rules for effective comprehension by domain users (Halpin 2000, 1995 and Arlow et al.,).

The fact-oriented meta schema facilitates an iterative and incremental approach to gathering rules. It allows the business model to be developed in a natural process based on discovery, by allowing the knowledge analyst to initiate the business information elicitation process with rules, entities or fact types.

The dawn of the "knowledge economy" has raised the importance for enterprises to deduce, and create knowledge from information. Enterprise-wide systems development methods and techniques should have the capability to capture domain knowledge. The meta schema introduced in this paper allow users to capture knowledge and closely relate them with items of information schema.

The meta schema can be further enhanced in the future by incorporating aspects of business process modeling and event modeling. This should provide a mechanism to document all interactions that exist among business processes, business sub-processes, rules, information and various events that exist in the UoD. The goal of developing a business model with the use of a 'single modelling language' would then have been achieved. Such a model would also be closely representative of the 100% percent principle and conceptualization principle (ISO/TR 9007, 1987).

It is expected that tools could be developed in the future to support capture and representation of business rules based on the meta schema introduced in this paper. Such tools should facilitate describing and documentation processes of associations and interactions that take place between business rule types, fact types and fact instances. In the future, it should also be possible to execute a model of this nature from its description. This should eliminate the need to map business rules into another programming language prior to use by business users.

The ability to generate XML Schema and XML instances should facilitate the process of interchanging data, information and knowledge among various sources. The integrated approach has also contributed towards the ability to move smoothly back and forth between information capture and knowledge capture in a requirements gathering activity, with the use of natural language to facilitate validation.

REFERENCES

Arlow, Jim., Emmerich, Wolfgang., Quinn, John., (1998) 'Literate Modelling-Capturing Business Knowledge with the UML', UML 98, http://www.literatemodelling.com/papers.htm

Berners-Lee, Tim., (1999) 'Weaving the Web', Orion Business

Bird, L., Goodchild, A., Halpin, T., (2000) 'Object Role Modelling and XML Schema', ER 2000, 19th International Conference on Conceptual Modeling. October 9 - 12. Salt Lake City, Utah, USA

Bollen, Peter., 'A process description language and method for organizational processes in Universal Informatics', Department of Management Sciences, Faculty of Economics and Business Administration, University of Limburg, PO Box 616, Maastricht, 6200 MD, The Netherlands

Campbell, L.J., Halpin, T.A., Proper, H.A., (1996) 'Conceptual Schemas with Abstractions-Making flat conceptual schemas more comprehensible', Data & Knowledge Engineering, 20, pp 39-85

Clancey, William. J., (1983) 'The Epistemology of a Rule-Based Expert System – a Framework for Explanation', Artificial Intelligence

Creasy, Peter., (1989) 'ENIAM: A more Complete Conceptual Schema Language', Proceedings of the Fifteenth International Conference on Very Large Data Bases Aug

Date, C. J., (2000) 'What Not How- The Business Rules Approach to Application Development', Addison Wesley

Debenham, J.K., 'Knowledge Systems Design', Prentice Hall, 1989

Debenham, John., 'A Foundation for Knowledge Modelling', Australian Computer Science, Vol. 15, No 1, pp 413-424

EXSYS Inc , 'EXSYS Rule Book: Expert System Development Tool – User Manual', 1996

Fullerton, Paul. D., James, Julie. A., 'The potential of Knowledge Based Systems in Business Planning', Proceedings of the Australian Computer Conference, pp 649-670, September1987

Halpin, Terry., 'Conceptual Schema & Relational Database Design', Second Edition, Prentice Hall Australia, 1995

Halpin, Terry., 'An ORM Meta Model', The Journal of Conceptual Modeling, October 2000, Issue Number 16, www.inconcept.com/jcm

ISO/TR 9007, 'Information processing systems – Concepts and terminology for the conceptual schema and the information base', International organization for standardization, 1987

Jih, W.J.Kenny., 'Comparing Knowledge-Based and Transaction Processing Systems Development', Journal of Systems Management, May 1990

Martin, James., Odell, James. J., 'Object Oriented Methods – A Foundation (2nd Edition)', Englewood

Cliffs, N.J. Prentice Hall 1997

Meta Data Coalition, 'Business Engineering Model Business Rules', Open Information Model, Review

Draft, July 15 1999. http://www.mdcinfo.com/OIM/models/BEM.html

Nijssen, Shir., 'A General Analysis Procedure', Recent Advances in Universal Informatics, Nijssen Adviesbureau voor Informatica b.v. 1994, PNA Training, Beutenaken 36, 6278 NB Beutenaken, Netherlands

Open Information Model, Version 1.1 Proposal, August 1999

http://www.mdcinfo.com/OIM/MDCOIM11.html

Quatrani, Terry., 'Visual Modeling with Rational Rose and UML', Addison Wesley, 1998

Ross, Ronald. G., 'The Business Rule Book: Classifying, Defining and Modeling Rules' Boston, Massachusetts, Database Research Group Inc., 1994

Schouten, Han., 'A repository for Logical Expressions', The Journal of Conceptual Modeling, April 1999, Issue Number 8, www.inconcept.com/jcm

The Business Rules Group (formerly known as the Guide Business Rules Project), 'Defining Business Rules – What are they really?', Final Report, revision 1.3, July 2001

W3C Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/1998/REC-xml-19980210

W3C XML Schema Part 0- Primer, http://www.w3.org/TR/xmlschema-0/

COPYRIGHT