Model-Driven Development based Transformation of Stereotyped Class Diagrams to XML Schemas in a Healthcare Context [*]

Eladio Domínguez[1], Jorge Lloret[1], Beatriz Pérez[1], Áurea Rodríguez[1], Ángel L. Rubio[2], and María A. Zapata[1]

[1] Dpto. de Informática e Ingeniería de Sistemas.
Facultad de Ciencias. Edificio de Matemáticas.
Universidad de Zaragoza. 50009 Zaragoza. Spain.
`noesis,jlloret,beaperez,arv852,mazapata@unizar.es`
[2] Dpto. de Matemáticas y Computación. Edificio Vives.
Universidad de La Rioja. 26004 Logroño. Spain.
`arubio@unirioja.es`

Abstract. The health sector uses clinical guidelines as instruments for helping decision making. We are interested in the development of a ubiquitous decision support system (UDSS) for clinical guidelines in order to help the medical staff in their decisions and in order to record a trace of the application of the guidelines. For the development of such a system in a Model–Driven Development (MDD) setting, we propose the use of class diagrams of the Unified Modeling Language (UML) with stereotypes and eXtensible Markup Language (XML) schemas. When both languages, UML and XML, have to share a common modelling space, the necessity of transforming UML models into XML schemas arises. However, to our knowledge, previous transformation proposals do not consider the case in which some profiles have been applied to the UML model. For this reason, in this paper we propose a set of rules for translating stereotyped UML class diagrams into XML schemas, storing a trace of the application of the guideline.

Key words: Clinical Guideline, UML Class Diagram, XML Schema

1. Introduction

The daily work of physicians involves the obligation of taking difficult decisions since they have implications for patients' lives. Specifically, guidelines and protocols are the most common way of assisting doctors in decision making. In general, guidelines and protocols incorporate procedures and analyses for different aspects of medical practice and management [14], but we focus our attention on guidelines used for diagnosis and prognosis. Within this context one

[*] This work has been partially supported by DGI (project TIN2005-05534 and FPU grant AP2003-2713), by the Government of La Rioja (project ANGI 2005/19), by the Government of Aragon and by the European Social Fund.

of the most important challenges is to provide the practitioner with computer assistance during the application of a guideline [16].

As a way of achieving this, we are studying, as a long term research goal, the automatic development of ubiquitous decision support systems (UDSS) for clinical guidelines [17]. We propose to tackle this goal by using a model-driven development approach (MDD) [18] so that, given a particular guideline, this is expressed by means of a model from which we aim at generating a UDSS for the guideline in question. There are several advantages that such a system provides to the medical staff (see [17] for a detailed study) but in this paper we want to focus our attention on the fact that a trace of the guideline application can be automatically recorded (physician's decisions, history of patient's states,...).

For the development of the UDSS in an MDD setting, we propose to use the Unified Modeling Language (UML) and the eXtensible Markup Language (XML) as modelling languages. In particular, UML class diagrams are used, as platform independent models (PIM), for modelling the trace of the execution of guidelines. But, whenever UML is applied to different domains [21], the necessity of adapting the UML proposal to a particular context arises. This adaptation could be achieved by using UML lightweight extension mechanisms, that is, by defining a UML profile [15]. We have employed this possibility so that stereotyped UML class diagrams are used to store the information that is generated while the guideline is being applied. Furthermore, the ubiquitous nature of this system makes data interchange a critical issue and therefore the use of XML Schema, as a platform specific model (PSM), appears to be a natural choice.

When both languages, UML and XML, are used within an MDD approach, a transformation between them must be provided. In our particular case, stereotyped UML class diagrams have to be translated into XML schemas. However, although there are many works in the literature proposing a transformation of UML class diagrams into XML schemas, after an exhaustive revision and comparison of such transformations, we conclude in [4] that none of the analyzed approaches specifies transformation rules for the general application of UML profiles. For this reason, as the main contribution of this paper, we propose a set of rules for translating stereotyped UML class diagrams into XML schemas. This set of rules is applied to our case of stereotyped UML class diagrams representing a trace of the application of some guideline.

The remainder of this paper is structured as follows. In the following section we discuss related work. In Section 3 we explain the case study and our proposal for automatic generation of a UDSS for a given guideline in an MDD setting. In section 4 we describe the transformation of stereotyped UML class diagrams into XML schemas. Finally, conclusions and future work are presented.

2. Related Work

Several proposals for providing computer assistance for the application of clinical guidelines can be found in the literature [7,16]. Many of them propose their own set of representation primitives to capture the structure of guidelines.

Among them, it is worth noting the Guideline Interchange Format (GLIF) and PROforma [7,14] which define specific ontologies for representing guidelines. Nevertheless, as is claimed in [8], the use of mainstream modelling languages such as UML and XML has potential benefits. In particular, both UML [2,11] and XML [3,19] are widely accepted as practical and beneficial formalisms within the healthcare realm. For this reason, we have chosen these modelling languages for the development of a UDSS [17].

A noteworthy characteristic of UML is the possibility of being adapted for solving certain types of problems in a particular domain by way of UML profiles. This feature is important within the medical context [21] since different specific necessities must be tackled in this domain such as data confidentiality [9], knowledge modelling [1] or the business aspects of hospitals [20], among others.

The problem is that although several authors propose algorithms for translating UML models into XML Schema, to our knowledge, none of them specifies transformation rules for the general application of UML profiles. In [4] we present a survey comparing different UML to XML schema transformation approaches (a total of 22 papers are analyzed) and, among them, only one [12] takes into account UML stereotypes, but without considering their properties.

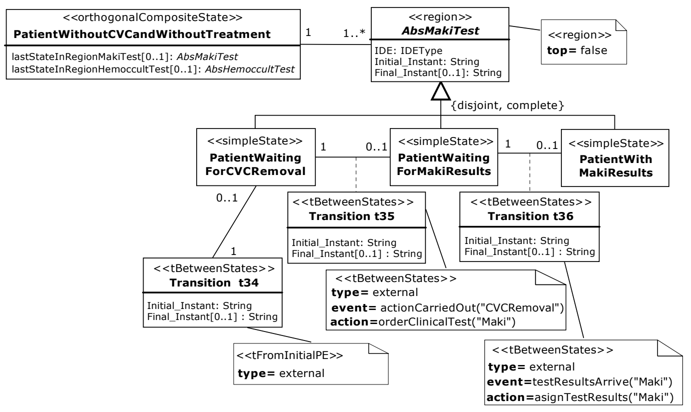
Given the necessity of translating UML stereotypes into XML Schemas for the development of medical systems, we consider it suitable to have a general mechanism available for such a transformation.

As for the development strategy of the decision support systems proposed in the literature [7,16], to our knowledge, none of these uses an MDD approach, so that this characteristic distinguishes our UDSS proposal [17] from others.

3. Case Study

Recently we have proposed an MDA–based approach to automatically develop ubiquitous decision support systems (UDSS) for clinical guidelines [17]. One of the goals of a UDSS is to guide the physician during the application of a guideline in a very specific way in order to help in her decision making. Another goal of such decision support systems is to record automatically the trace of the application of the guideline. That is, since the application of a guideline can vary depending on the specific characteristics of the patient, our aim is to record all the information resulting from its application, such as the patient's clinical conditions, the physician's decision, the antibiotic treatment given to the patient and the actions or tests that are carried out regarding the patient's clinical state. All this information will be stored as part of the patient's clinical history. This could be used for obtaining a summary of the application of the guideline, the purposes of physicians' future decisions, legal support, etc [6]. From now on, we will focus our attention on this second goal of our UDSS proposal.

In order to store the trace of the application of a guideline, first of all we propose to represent the dynamics of the given guideline by means of a statechart [15]. Next, a stereotyped UML class diagram is obtained starting from this statechart. The process we propose for generating the class diagram is based on



<<orthogonalCompositeState>>
PatientWithoutCVCandWithoutTreatment
lastStateInRegionMakiTest[0..1]: AbsMakiTest
lastStateInRegionHemoccultTest[0..1]: AbsHemoccultTest
1
1..*
<<region>>
AbsMakiTest
IDE: IDEType
Initial_Instant: String
Final_Instant[0..1]: String
<<region>>
top= false
{disjoint, complete}
<<simpleState>>
PatientWaiting ForCVCRemoval
1
0..1
<<simpleState>>
PatientWaiting ForMakiResults
1
0..1
<<simpleState>>
PatientWith MakiResults
0..1
1
<<tBetweenStates>>
Transition t35
Initial_Instant: String
Final_Instant[0..1] : String
<<tBetweenStates>>
Transition t36
Initial_Instant: String
Final_Instant[0..1] : String
<<tBetweenStates>>
Transition t34
Initial_Instant: String
Final_Instant[0..1] : String
<<tBetweenStates>>
type= external
event= actionCarriedOut("CVCRemoval")
action=orderClinicalTest("Maki")
<<tFromInitialPE>>
type= external
<<tBetweenStates>>
type= external
event=testResultsArrive("Maki")
action=asignTestResults("Maki")

Fig. 1. Part of the stereotyped UML class diagram of the IRC guideline.

some ideas proposed in [13], although our objective is different (in [13] the goal is to generate Java code). In addition to the ideas described in [13], we propose to annotate the class diagram with information about the UML statechart from which it comes, resulting in a stereotyped UML class diagram. The resulting stereotyped UML class diagram allows us to register information concerning the application of the guideline (patient's states, physician's decisions and clinical actions). Finally, in order to implement the physical storage, the stereotyped UML class diagram is transformed into an XML schema. So, in this context, a general mechanism that allows the transformation between these models is essential. The general mechanism we propose is described in the next section.

As an example of application, we will use a particular clinical guideline presented in [6]. This guideline is used by doctors in the Intensive Care Unit (ICU) when it is suspected that a patient has an infection related to an intravenous catheter (IRC). This guideline is necessary because the use of catheters is more and more frequent and they may be associated with complications, especially in Intensive Care Units (ICU) where patients in critical states suffer from a wide range of serious illness and injury. In particular, this guideline for the prevention of catheter-related infections (IRC guideline) establishes the criterion for the prevention, diagnosis and antibiotic treatment of infections caused by catheters.

For example, we can see a part of the stereotyped UML class diagram of the IRC guideline in Fig. 1. This UML diagram is stereotyped making use of a *profile for statechart execution persistence* (SEP profile) we have defined. This part of the diagram will allow the trace of the patient's states to be stored when the medical staff thinks the patient has an infection related to an intravenous catheter but the doctors do not yet know if the sick person has an infection. Moreover, the information relating to activities regarding Maki tests (such as clinical actions that are carried out, and timestamps in which the Maki test is ordered and its results are received) will be recorded. In particular, we can observe the class *TransitionT35* which corresponds to the change in the patient's state when the catheter is removed. In these circumstances, the patient's state while waiting for the catheter removal (*PatientWaitingForCVCRemoval*) changes to another state in which she is waiting for the results of the Maki Test (*PatientWaitingForMakiResults*). The *TransitionT35* class is described by means of two tag values, one indicating the event which has triggered the transition (*actionCarriedOut('CVCRemoval')*) and the other tag value representing the action which has been carried out when the previous event has taken place (*orderClinicalTest('Maki')*).

4. Translation of Stereotyped Models to XML Schemas

UML profiles have been used in different medical information systems with different aims, such as for data confidentiality [9] or for historical records as in our case. In the medical context they can be applied to different types of UML diagrams (such as use case, class or sequence diagrams). Taking this into account, our proposal aims to be generally applicable and therefore allows the translation of any stereotyped UML model into XML Schema. Nevertheless, since in our specific case we start from a stereotyped UML class diagram model, in this section, we show our proposal applied to this type of UML diagram.

Our approach consists of two translation steps. In the first step we propose to create an XML schema for each UML profile, which makes the XML schema reusable for any UML model to which the profile is being applied. In the second step, we translate the stereotyped UML model into an XML schema, which will make use of the XML schema resulting from the previous step.

One of the strengths of our approach is that we do not present 'yet another approach' to the mapping of UML models into XML Schemas, since any previously published mapping can be taken as a starting point. For the translation of UML class diagrams, we have used the approach of [12], since we consider that this approach is one of the most complete proposals for these transformations [4].

Next, we explain in detail our approach using as an example the transformation of the stereotyped UML class model for trace recording of Fig. 1.

4.1. Step 1: Translation of a UML Profile

As we have pointed out, we create an XML schema for each UML profile. In order to define it, we have taken into account not only several peculiarities of the clinical context as far as conceptual modelling is concerned, but also specific semantics of UML Profiles. Considering these factors, the first decision that we have taken is to define this XML schema without declaring a `targetNamespace`, that is, to define a Chameleon XML schema [10]. The main reason for defining such a type of schema is the following. Since several profiles can be applied to different UML models in the medical environment, it is of interest for the XML schemas created from those UML models to provide their own application–specific namespace no matter what the profile or profiles applied to it. So, we prefer not to declare a namespace for each profile XML schema; rather, we want containing schemas to provide their own namespace. In this sense, chameleon

```
1  <?xml version="1.0"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
3  <xs:complexType name="TransitionBetweenStatesDefinitionType">
4    <xs:sequence>
5      <xs:element name="type" type="TransitionKindEnume" minOccurs="1" maxOccurs="1"/>
6      <xs:element name="event" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
7      <xs:element name="guard" type="xs:string" minOccurs="0" maxOccurs="1"/>
8      <xs:element name="action" type="xs:string" minOccurs="0" maxOccurs="1"/>
9    </xs:sequence>
10   <xs:attribute name="profile" type="xs:string" fixed="SEPProfile"/>
11 </xs:complexType>
12 <xs:complexType name="StateDefinitionType">
13    <xs:sequence>
14       <xs:element name="entry" type="xs:string" minOccurs="0" maxOccurs="1"/>
15       <xs:element name="exit" type="xs:string" minOccurs="0" maxOccurs="1"/>
16       <xs:element name="do" type="xs:string" minOccurs="0" maxOccurs="1"/>
17    </xs:sequence>
18    <xs:attribute name="profile" type="xs:string" fixed="SEPProfile"/>
19 </xs:complexType>
20  <xs:complexType name="SimpleStateDefinitionType">
21   <xs:complexContent>
22     <xs:extension base="StateDefinitionType"/>
23   </xs:complexContent>
24 </xs:complexType>
25 ...
26 </xs:schema>
```

Fig. 2. Part of the SEP Profile XML schema.

schemas provide a lot of flexibility since they take on the `targetNamespace` attribute of the XML schema that includes them [10].

From now on we will refer to the obtained chameleon XML schema as the *profile XML schema*. It will be defined as follows. In each chameleon XML schema, every stereotype is represented as a global `complex type` element whose name is the name of the stereotype plus the word `DefinitionType`. This complex type will be defined in accordance with the following:

- when the stereotype has properties, this complex type will contain an XML `sequence` element including one XML `element` for each property of the stereotype. Each of these elements is named as the property. In addition, the multiplicity of the property is represented by using the `minOccurs` and `maxOccurs` attributes.
- whether or not the stereotype has properties, this complex type will have an XML `attribute` named `profile` of the type `string`. The value of this attribute will be the name of the profile to which the stereotype belongs.

For example, part of the profile XML schema obtained from the created SEP Profile is shown in Fig. 2. We show the translation of stereotypes «TransitionBetweenStates» (see lines from 3 to 11) and «SimpleState» (see lines from 20 to 24). For instance, in the translation of the definition of the «TransitionBetweenStates» stereotype, we have defined a complex type `TransitionBetweenStatesDefinitionType` with a sequence of elements, one for each property (see lines from 5 to 8) and an attribute named `profile` with the name of the profile (see line 10).

4.2. Step 2: Transformation of the Stereotyped Class Model

Once an XML schema is defined for each profile, the second step deals with the creation of the XML schema for the stereotyped class model, called *stereo-*

```
1  <?xml version="1.0"?>
2  <schema
3  xmlns="http://www.w3.org/2001/XMLSchema"
4  xmlns:cd="http://www.XMLproyect.com/UMLCD"
5  targetNamespace="http://www.XMLproyect.com/UMLCD"
6  elementFormDefault="qualified">
7   <include schemaLocation="SEPProfileDefinition.xsd"/>
8   <element name="TraceClassDiagram">
9     <complexType>
10      <sequence>
11        <element name="PatientWaitingForMakiResults"
12          type="cd:PatientWaitingForMakiResultsType"
13          minOccurs="0" maxOccurs="unbounded"/>
14        <element name="Transitiont35"
15          type="cd:Transitiont35Type"
16          minOccurs="0" maxOccurs="unbounded"/>
17        ...
18      </sequence>
19    </complexType>
20    <key ...>...</key>
21  </element>
22  <complexType name="PatientWaitingForMakiResultsType">
23    <complexContent>
24      <extension base="cd:AbsMakiTestType">
25        <sequence>
26         ...
27         <element name="simpleState"
28           type="cd:StateDefinitionType"/>
29        </sequence>
30      </extension>
31    </complexContent>
32  </complexType>
33 <complexType name="TransitionT35Type">
34   <sequence>
35     <element name="initial_instant" type="string"/>
36     <element name="final_instant" type="string"
37       minOccurs="0"/>
38      <element name="TransitionBetweenStates"
39             type="cd:TransitionBetweenStatesType"/>
40   </sequence>
41  </complexType>
42 <complexType name="TransitionBetweenStatesType">
43   <complexContent>
44     <restriction base="cd:TransitionBetween
45         StatesDefinitionType">
46       <sequence>
47         <element name="type" type="cd:TransitionKindType"
48              fixed="external"/>
49         <element name="event" type="string"
50              fixed="actionCarriedOut('CVCRemoval')"/>
51         <element name="action" type="string"
52              fixed="orderClinicalTest('Maki')"/>
53       </sequence>
54     </restriction>
55   </complexContent>
56  </complexType>
57   ....
58 </schema>
```

Fig. 3. The class diagram XML schema.

typed XML schema. The idea consists of extending the approach of [12] by adding new transformation rules concerning the stereotypes applied to the class model.

As we will see in this section, the complex types created in the profile XML schemas will be used for the translation of the stereotypes applied to the class model. Therefore, the final XML schema created for the stereotyped class model will include, for each profile applied to the class model, an XML `include` element, representing schema dependencies. For example, an extract of the XML schema obtained from the stereotyped class diagram for trace recording of Fig. 1 is shown in Fig. 3. In line 7 of this figure we include the SEP profile XML schema in order to reuse its defined complex types along the XML schema.

In order to explain our approach for step 2, we will show how to translate stereotyped UML classes, attributes and binary associations by using the skeletons presented in Table 1. We have distinguished two translation options, depending on whether the stereotype (or stereotypes) applied to the UML element has tagged values. In each option, we indicate in bold the XML code which corresponds to the translation of a stereotype of the UML class model. In addition, italics describe code which has to appear in the XML schema.

As we show in Table 1(see (a)(c)(e)), the translation of applied stereotypes without tagged values consists of defining an XML `element` whose `Type` is the complex type defined for that stereotype in the profile XML Schema. If the applied stereotype has tagged values (see (b)(d)(f)), an XML `element` and an additional global XML `complex type` are defined. This complex type is declared as a derivation by restriction [22] of the complex type defined for the stereotype in the profile XML schema. Therefore, each XML element which corresponds with a tag value of the applied stereotype is redeclared. Also, the XML `fixed` attribute with the corresponding tag value as value is added to the redeclared XML element.

Stereotyped UML class in which the stereotype has no tagged values	**Stereotyped UML class in which the stereotype has tagged values**

`<element name=className type=classNameType/>`
`<complexType name=classNameType>`
`    <sequence>`
`        XML elements which translate`
`          attributes of the UML class`
`        <element name=stereotypeName`
`          type=stereotypeNameDefinitionType/>`
`    </sequence>`
`    <attribute name="id" type="ID"`
`      use="required"/>`
`    XML attributes which translate`
`      attributes of the UML class`
`</complexType>`
	`<element name=className type=classNameType/>`
	`<complexType name=classNameType>`
	`   <sequence>`
	`      XML elements which translate`
	`         attributes of the UML class`
	`      <element name=stereotypeName`
	`         type=stereotypeNameType/>`
	`   </sequence>`
	`   <attribute name="id" type="ID"`
	`      use="required"/>`
	`   XML attributes which translate`
	`      attributes of the UML class`
	`</complexType>`
	
	`<complexType name=stereotypeNameType>`
	`   <complexContent>`
	`      <restriction`
	`         base=stereotypeNameDefinitionType>`
	`          <sequence>`
	`            Redeclared elements from the`
	`               elements of the base type`
	`          </sequence>`
	`      </restriction>`
	`   </complexContent>`
	`</complexType>`

(a)	(b)
Stereotyped UML attribute in which the stereotype has no tagged values	**Stereotyped UML attribute in which the stereotype has tagged values**

`<element name=attributeName`
`   type=attributeNameType/>`
`<complexType name=attributeNameType>`
`   <sequence>`
`      XML element (with name "value")`
`         which translates the UML attribute`
`      <element name=stereotypeName`
`         type=stereotypeNameDefinitionType/>`
`   </sequence>`
`</complexType>`
	`<element name=attributeName`
	`   type=attributeNameType/>`
	`<complexType name=attributeNameType>`
	`   <sequence>`
	`      XML element (with name "value") which`
	`         translates the UML attribute`
	`      <element name=stereotypeName`
	`         type=stereotypeNameType/>`
	`   </sequence>`
	`</complexType>`
	
	Include a complex type as the *stereotypeNameType* defined in (b) of this figure.

(c)	(d)
Stereotyped UML binary association in which the stereotype has no tagged values	**Stereotyped UML binary association in which the stereotype has tagged values**

`<element name=associationEndRolName`
`   type=associationEndRolNameType/>`
`<complexType name=associationEndRolNameType>`
`   <sequence>`
`      XML element (with name "id") which`
`         translates the UML association`
`      <element name=stereotypeName`
`         type=stereotypeNameDefinitionType/>`
`   </sequence>`
`</complexType>`
	`<element name=associationEndRolName`
	`   type=associationEndRolNameType/>`
	`<complexType name=associationEndRolNameType>`
	`   <sequence>`
	`      XML element (with name "id") which`
	`         translates the UML association`
	`      <element name=stereotypeName`
	`         type=stereotypeNameType/>`
	`   </sequence>`
	`</complexType>`
	
	Include a complex type as the *stereotypeNameType* defined in (b) of this figure.

(e)	(f)

Table 1. Our approach to translate several stereotyped UML elements.

Finally, the XML element generated for each applied stereotype is included as part of the translation of the stereotyped class element in the following way:

- if the transformation of the stereotyped UML element involves the creation of an XML `complex type` according to the chosen transformation (in our case the approach proposed in [12]), then the XML element generated for the stereotype is nested inside the last child of the sequence of the global complex type.

- otherwise, we propose to modify the translation approach of the stereotyped UML element so that it involves the creation of an XML `complex type`. In this way, the XML element created for the stereotype can be nested inside that complex type.

An example of our approach is shown in Fig. 3. In this XML schema, among other things, the translation of class *TransitionT35* with the stereotype «transitionBetweenStates» previously commented in Section 3 is depicted. For the translation of this stereotyped class, since its applied stereotype has tagged values, we have followed the approach of Table 1(b). We have created the XML element `transitionBetweenStates` of type `transitionBetweenStatesType` (see lines 38 and 39). Additionally, we have defined the complex type `transitionBetweenStatesType` as a derivation by restriction of the type `transitionBetweenStatesDefinitionType` (see lines from 42 to 56). In this case, we have added to that complex type three new elements corresponding to three of the tagged values of the applied stereotype. Each of those elements has the XML element `fixed`, indicating the specific value of the corresponding property. For instance, to translate the event that has triggered the transition (*actionCarriedOut('CVCRemoval')*), we use the XML element `event` defined with the fixed value `actionCarriedOut('CVCRemoval')` (see lines 49 and 50).

5. Conclusions and Future Work

In this paper we propose the use of stereotyped UML class diagrams to store a guideline application in an MDD setting. We also provide a general mechanism for the transformation of stereotyped UML class models into XML schemas.

There are several lines of future work. The main issue is the incorporation of our transformation proposal into our general framework of evolution of UML/XML models [5]. In particular, the development of an evolution tool is an ongoing project. As for the transformation of UML profiles, the definition of a procedure for the translation of UML constraints is a goal for further work.

References

1. M. S. Abdullah, R. Paige, I. Benest, and C. Kimble. Knowledge Modelling Using The UML Profile. In *Artificial Intelligence Applications and Innovations (AIAI)*, volume 204, pages 70–77. Springer-Verlag, 2006.
2. V. Aggarwal. The Application of the Unified Modeling Language in Object-Oriented Analysis of Healthcare Information Systems. *Systems Journal of Medical Systems*, 26(5):383–397, 2002.
3. T. Dart, Y. Xu, G. Chatellier, and P. Degoulet. Computerization of guidelines: towards a "guideline markup language". *Medinfo*, 10:186–90, 2001.
4. E. Domínguez, J. Lloret, B. Pérez, A. Rodríguez, A. L. Rubio, and M. A. Zapata. A Survey of UML Models to XML Schemas Transformations. 2007. Submitted for Publication.

5. E. Domínguez, J. Lloret, A. L. Rubio, and M. A. Zapata. Evolving XML Schemas and Documents Using UML Class Diagrams. In K. V. Andersen et al., editor, *Proceedings of DEXA 2005*, volume 3588 of *LNCS*, pages 343–352, 2005.
6. E. Domínguez, B. Pérez, A. Rodríguez, and M. A. Zapata. Medical protocols for taking decisions, in an ubiquitous computation context. *Novática*, 177:38–41, Sept-Oct 2005.
7. P. L. Elkin, M. Peleg, R. Lacson, E. Bernstam, S. Tu, A. Boxwala, R. A. Greenes, and E. H. Shortliffe. Toward Standardization of Electronic Guideline Representation. *MD Computing*, 17(6):39–44, 2000.
8. L. Hederman, D. Smutek, V. Wade, and T. Knape. Representing Clinical Guidelines in UML: A Comparative Study. In *Medical informatics in Europe 2002*, pages 471–477, Amsterdam, Netherlands, 2002. IOS Press.
9. R. Heldal, S. Schlager, and J. Bende. Supporting Confidentiality in UML: A Profile for the Decentralized Label. In *Third International Workshop on Critical Systems Development with UML*, 2004.
10. D. Hunter and et al. *Beginning XML (3rd Edition)*. Wrox, September 2004.
11. V. M. Jones, A. Rensink, and H. Brinksma. Modelling mobile health systems: an application of augmented MDA for the extended healthcare enterprise. In *Proceedings of the EDOC05*, pages 58–69. IEEE Computer Society, 2005.
12. T. Krumbein and T. Kudrass. Rule-Based Generation of XML Schemas from UML Class Diagrams. In *Proceedings of the XML Days at Berlin, Workshop on Web Databases (WebDB)*, pages 213–227, 2003.
13. I. Azim Niaz and J. Tanaka. Code Generation From UML Statecharts. In *SEA 2003*, 2003.
14. L. Ohno-Machado, J.H. Gennari, S.Ñ. Murphy, and et al. The GuideLine Interchange Format: A Model for Representing Guidelines. *Journal of the American Medical Informatics Association*, 5(4):357–372, 1998.
15. OMG. UML 2.0 Superstructure Specification, August 2005. Document formal/05-07-04. Available at `http://www.omg.org/`.
16. OpenClinical. Methods and tools for representing computerised clinical guidelines. Webpage, `http://www.openclinical.org/gmmsummaries.html`.
17. I. Porres, E. Dominguez, B. Perez, A. Rodriguez, and M. A. Zapata. Development of an Ubiquitous Decision Support System for Clinical Guidelines using MDA. 2007. Submitted for Publication.
18. B. Selic. The pragmatics of model-driven development. *IEEE Software*, 20(5):19–25, 2003.
19. A. Shabo, S. Rabinovici-Cohen, and P. Vortman. Revolutionary impact of XML on biomedical information interoperability. *IBM Systems Journal*, 45(2):361–372, January 2006.
20. A. Tanaka, Y.Ñagase, Y. Kiryu, and K.Ñakai. Applying ODP Enterprise Viewpoint Language to Hospital Information System. In *Proceedings of the Fifth IEEE International EDOC'01*, pages 188–192. IEEE Computer Society, 2001.
21. A. A. F. van der Maas, A. H. M. Ter Hofstede, and A. J. T. Hoopen. Requirements for Medical Modeling Languages. *Journal of the American Medical Informatics Association*, 8:146–162, 2001.
22. W3C. XML Schema Part 0: Primer Second Edition, October 2004. Available at `http://www.w3.org/TR/xmlschema-0/`.