MEDINFO 2001
V. Patel et al. (Eds)
Amsterdam: IOS Press

XML and the VITAL Standard: The Document-oriented Approach for Open Telemedicine Applications

Angie Anagnostaki[a], Sotiris Pavlopoulos[a], Dimitris Koutsouris[a]

[a] *Biomedical Engineering Laboratory, Department of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece*

Abstract

This paper describes an effort to create a common, document-oriented architecture for the interchange of medical data in healthcare telemedicine applications. Key components are: The VITAL standard specifying a common (medical device independent) representation of Vital Signs Information and the Extensible Markup Language (XML) specifying the document specifications form, an architecture that, in aggregate, define the semantics and structural constraints necessary for the exchange of vital signs and related medical data. The modelling and design technique for the described application has been the Unified Modelling Language (UML). The XMI (XML Metadata Interchange Format) of the Object Management Group (OMG) provided the meta-model for this application, for sharing objects using XML, via the transfer of the application's UML model to XML documents and DTDs.

Keywords:

Standards; Object-Oriented; Healthcare; XML; XMI; UML

Introduction

Bio-signal monitoring devices have developed in an unstructured manner in terms of communication interfaces. The need for technical standardization in healthcare environments enabling communication in a structured and open way, with clinical, administrative and research benefits, is obvious. Towards this direction, a number of industrial health informatics and telematics standards have developed and are being evaluated in the past few years.

Into the light of the above, the "VITAL" ENV 13734 standard (Vital Signs Information Representation) [1] of the European Standardization Committee CEN provides the definition of a common (device independent) representation of Vital Signs Information as well as the definition of a common model for accessing this information. This standard addresses the definition and structuring of information that is communicated or referred to in communication between application entities. It also specifies an inventory of a schematic presentation and identification scheme for all types of Object-Oriented modelling elements used in its Information Model, according to the Coad & Yourdon "Object Oriented Analysis" (OOA) technique.

The implementation of a telemedicine application following the definitions of such standard must take into account the O-O character of VITAL and should take advantage of the relevant state-of-the-art industrial software development techniques and development environments that exist.

Given the variability in the clinical structures, underlying information models, degree of semantic encoding, use of standard healthcare terminologies, platform- and vendor-specific features, it is currently difficult to store and/or exchange clinical data and documents with retention of computer-processable semantics over both time and distance. The idea of a common data architecture, vendor-neutral and platform-independent that can accommodate a diverse set of structures and clinical data points directly to the World Wide Web Consortium (W3C) standard, Extensible Markup Language (XML). CEN TC251 Working Groups have expressed strong interest in the potential of XML in the implementation of health informatics standards, especially for messaging applications that require standardization in the representation of data. XML provides a means to share and communicate clinical information in an effective manner.

Materials and Methods

Modelling and design of the application

The VITAL-compliant model

The Domain Information Model (DIM) of the VITAL standard is an object-oriented model that consists of objects, their attributes and their methods which are abstractions of real world entities in the domain of (vital signs information communicating) medical devices. The DIM comprises of a static model, which organizes the problem domain into packages, namely subjects, who group together objects, related to one another. The VITAL DIM

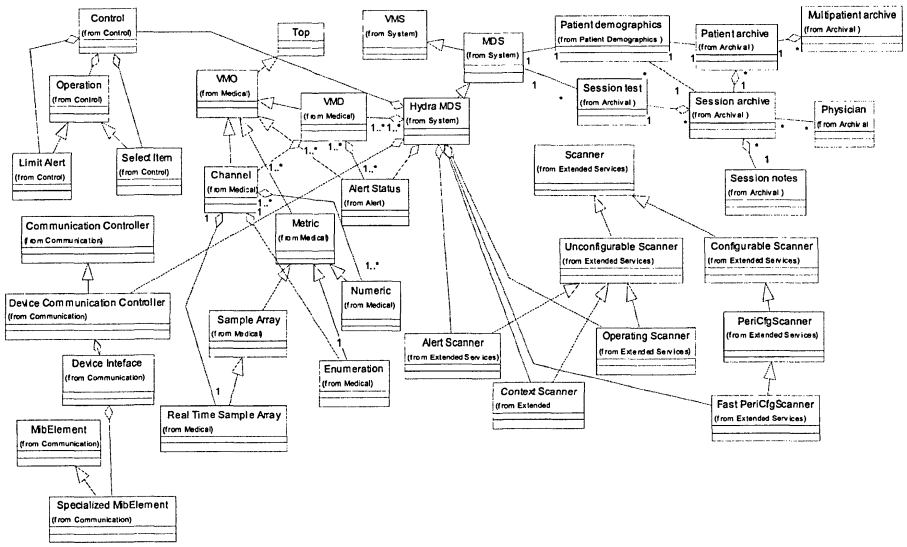

Figure 1 Object class diagram for the implementation of the overall model

defines eight subjects: Medical, System, Patient, Service and Control, Archival, Alert, Extended Services and Communication.

The model of our system utilized objects of all Subjects defined in the VITAL standard. The Medical Device System (MDS) object comprised of three Virtual Medical Devices (VMDs), namely the ECG VMD, the BP VMD and the SpO_2 VMD, which are abstractions for the related device subsystems (hardware or even pure software) of a monitoring device and represent the medical information that the application wishes to acquire and monitor. For the selection of the appropriate model to represent the above pure medical information VMDs, we have consulted the (complementary to VITAL) IEEE 1073 [2] series of standards, which define a protocol for the exchange of measurement data from clinical instruments (patient monitors, infusion pumps, etc.).

From the VITAL Subjects that we used for modelling our system, the Medical Subject and related objects are responsible for marshalling all static attributes of the medical data -waveform or non-waveform-, the System Subject for system specific information, the Alert and Extended Services Subjects for alert management and reporting of the medical data dynamic attributes, the Patient and Archival Subjects for data management at the HealthCare Centre, the Control Subject for remote control services and the Communication Subject for communication related information.

The Unified Modelling Language (UML) technique

Although the VITAL standard is described by the Coad & Yourdon "Object Oriented Analysis" (OOA) model, our telemedicine application was designed and modelled with the Unified Modelling Language (UML) notation [3]. A UML model provided the class hierarchy for modelling ECG, Pulse Rate, Blood Pressure and SpO_2 data, as well as technical and physiological alert management.

The next Figure 1 illustrates the Object Class Diagram, which shows the VITAL Domain Information Model for the implementation of the overall model.

The XML implementation

XML (Extensible Markup Language) [4] is a proper subset of SGML [5,6] (Standard Generalized Markup Language, ISO 8879:1986). XML reduces a document to a word in a known context-free grammar through a process of markup. The formal markup specification for a collection of documents is the Document Type Definition (DTD). XML Documents are then written to conform to a particular DTD, enabling them to be automatically parsed and validated against that DTD. Our application has been structured in XML DTDs, harmonized with the evolving VITAL Domain Information Model (DIM). This information model served as the central schema defining the semantics for all messages [7] and documents relevant to our application.

Our experience with applying XML in healthcare has shown that no clear and distinct rules or strategies are available for mapping medical data or messages to XML. Data could be presented as well as elements or attributes.

The XMI meta-model of the application

While XML is a great way to share data, but you need something more to share objects. XMI (XML Metadata Interchange Format) [8] is the new standard from Object

Management Group (OMG) [9] for sharing objects using XML. Therefore, XMI has been used to provide the meta-model of our application. XMI specifies open information interchange for object-oriented models and data using XML.

The XMI specification integrates XML, the W3C specification, with the Unified Modeling Language (UML) and the Meta Object Facility (MOF) specifications adopted by OMG, by providing a standard way to convert objects into XML. The standard covers the transfer of UML models and MOF meta models. It identifies standard XML DTD's to allow the exchange of UML and MOF information.

XMI generation rules

Here's the mapping scenario of our application. The application's UML model is expressed in XMI. XMI makes XML even easier by leveraging UML's graphical ability to generate XML and defines two sets of generation rules for creating XML documents and XML DTDs. XMI document generation specifies how to serialize objects into an XML stream. The XML DTD generation specifies how to create a DTD that matches your objects from their class definitions.

XMI DTD architecture

XML consists of two parts: documents and DTDs (Document Type Declaration). Documents contain the information as a set of tags, while DTDs specify the rules for how tags may be used in a document. In XML, tags, also called Elements, form a tree-structured hierarchy. When tag is nested inside another, it is referred to as the "content" of the containing tag. XMI defines two sets of rules that provide open interchange and leverage the capabilities of XML. The two sets of rules in XMI are DTD generation and document generation. The DTD generation is used to specify an interchange format, and the document generation creates documents that use a given XMI DTD.

Suppose one Subject of our VITAL-compliant application model: the Medical Subject. The next Figure 2 shows the Medical Subject as a model in UML, namely the UML Object Class Diagram, which shows the subset of the VITAL DIM for the implementation of the Medical Subject for our system model. This diagram has been adapted from a UML (Rational ROSE'98) model. Some mandatory Attributes and Operations are shown.

One class in this model is the Metric class, which is super-class for the Sample Array, Enumeration and Numeric classes. Also, the Metric class contains several class-attributes: MetricSpecification, UnitCode, etc.

An XMI DTD can be generated from the UML model of the Medical Subject. There is one XML element for each class and class-attribute in the model. For example, for the Metric class, the elements are Metric.MetricSpecification, Metric.UnitCode, etc. Each class-attribute has its own XML element declaration, which may contain text values directly ("#PCDATA") or may contain references (XMI.reference) to another location for defining the value externally, in the cases where the values aren't ideal for XML, e.g. blocks of binary data or bitmaps. An example of an XMI DTD for one class (the Metric class) of our application's model can be seen in Figure 3.

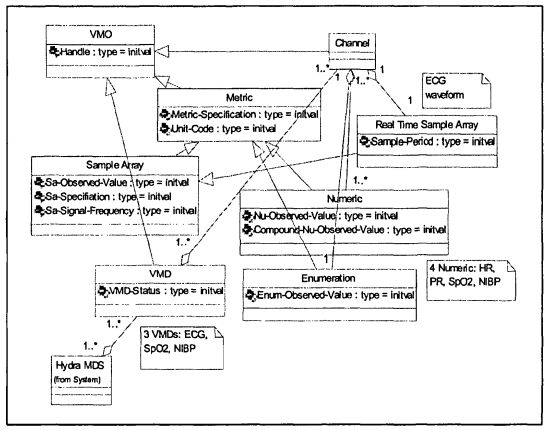

Figure 2 Object Class diagram for the Medical Subject

```
<!ELEMENT Metric (Metric.MetricSpecification, Metric.UnitCode,
XMI.extension*)? >
<!ATTLIST Metric %XMI.element.att; %XMI.link.att;>
<!ELEMENT Metric.MetricSpecification (#PCDATA | XMI.reference)* >
<!ELEMENT Metric.UnitCode (#PCDATA | XMI.reference)* >
```

Figure 3 XMI DTD for the Metric class of the Medical Subject

The application exchange using this DTD is exchanging an XMI document containing a specific Subject definition. The document uses the elements from the generated XMI DTD generated from the UML model. Thus, an XMI-generated DTD for UML allows interchange of object-oriented UML models and class definitions. This UML can thus be interchanged between design tools and IDEs using the UML DTD. An XMI-generated DTD for Java provides interchange of Java classes, a DTD for IDL enables interchange of IDL interfaces, etc. Standardizing a set of DTDs alone is not in itself enough for interchange in the general case, since DTDs do not have the ability to express the semantic meaning appropriate for the model. Instead of looking to the DTD alone to standardize interchange, UML or similar models must be the source for standards.

Every XMI DTD contains the elements generated from an information model, plus a fixed set of element declarations that may be used by all XMI documents. These fixed elements provide a default set of data types and the document structure, starting with the top-level XMI element. Each XMI document contains one or more elements called XMI that serves as a top level container for the information to be transferred. XMI is a standard XML element and may stand alone in its own document or may be embedded in XML or HTML documents. The XMI element contains the following structural elements:

- Header, which contains version declarations and optional documentation regarding the transfer.
- Content, which contains the core information that is to be transferred.
- Differences, which specify the differences between two XMI documents to be transferred. This is useful in cases such as small changes to a large set of information
- Extensions, which allows the transfer of private tool information beyond that already present in a DTD.

XMI document architecture

Following is the Metric class example of our application model as an XMI document using elements from the generated XMI DTD. The document begins with XML processing instructions declaring the character set and the location of the DTD to use. Since this is a reference, the DTD does not need to be transmitted along with the document. The top level element is XMI which declares that the version of XMI standard being used is 1.0 and sets the structure for the rest of the transfer. The header contains optional documentation and the content is a Metric element with the appropriate values for each of its tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XMI SYSTEM "Metric.dtd">
<XMI xmi.version="1.0" >
<XMI.header>
<XMI.documentation>
An example of the VITAL Metric class.
</XMI.documentation>
</XMI.header>
<XMI.content>
<Metric>
<Metric.MetricSpecification>VMD ECG</ Metric.MetricSpecification >
<Metric.UnitCode>NOM_DIM Degrees Celsius</ Metric.UnitCode >
</ Metric >
</XMI.content>
</XMI>
```

Figure 4 XMI Document using the Metric XMI DTD

XMI architecture for the application's database

XMI works with a wide range of information beyond UML. Examples include database information, Java etc. For our application, database information is especially important. The OMG standard Common Warehouse Metadata Interchange (CWMI), which enables the expression of database information and schemas in a common form that can be interchanged using XMI, has been used in our application for the interchange of a relational database. The representation for the database followed the schema of that a Relational Database contains Tables that in turn contain Columns. Tags in the DTD define the database, its tables and columns and their names. Relational databases and tables also contain tables and columns respectively, which are also represented as tags. The structure of a database is commonly called a "schema."

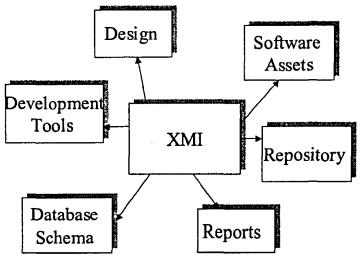

Figure 5

Results and Discussion

Object-orientation in software development is not a new concept. However, the practical adoption of object technologies, particularly distributed object technologies, has been a slow process. In healthcare, fueled by the advances in Internet, web and Java-based technologies, there has been a growing awareness of the need and usefulness of distributed object technologies [10].

There are two key considerations for using XML. One is that the human comprehensibility and verbosity of XML documents leads to very large documents, often on the order of megabytes. XML is a natural candidate for compression, often reducing the size of documents by over 90% with standard algorithms such as those used in zip. Sending compressed XML across networks, often slower than local disks, is likely to result in net savings of time. The advantage by the above statement for telemedicine applications that exchange bulky data, acquired, for example, by image processing applications is more than obvious.

The other key consideration for XML is identifying how to structure the DTD so that everyone can share information. XMI provides a way to keep these definitions synchronized using DTD generation. This consistency lets tools know how to traverse any XMI document or DTD in a regular manner to find the information needed.

Moreover, the XML design in the telemedicine and healthcare domain is already wide and supports various applications: dictionary techniques and terminology for diseases documentation, Electronic Patient Record, messaging techniques for exchanging healthcare information into a clinical context, standardization of clinical practice guidelines and procedures, and generally all applications that handle the migration from legacy to structured data.

As far as the XMI standard is concerned, Figure 5 shows the open world of XMI, where one can see the types of application development tools that interchange information using XMI as the standard. These applications include: Design tools, including o-o UML tools, Development tools, Databases, Data Warehouses and Business Intelligence tools, Software assets, Repositories and report generation, documentation tools and web browsers.

Conclusion

Pervasive support of XML, UML and XMI standards throughout the industry and widely available supporting technology, including repositories and databases, significantly reduces the time and cost to provide product interoperability in distributed heterogeneous software environments. The above-described application is but a small example of the efforts that exist in the Healthcare domain and points to a major trend in the utilization of object technologies in Healthcare. From the perspective of users of these technologies such as hospitals and clinics, adoption of component technology is driven by the promise of simplification, primarily in the management of the applications and the reduction in duplication of functionality.

Acknowledgements

This paper presents a piece of work complementary to the "VITAL-HOME" project [11,12], funded by the European Commission-Directorate General Enterprise, under the Information Society Initiative for Standardization (ISIS) Programme. The authors would like to thank all the project participants for their significant contribution.

References

[1] CEN/TC251/PT5-021, "Health Informatics - Vital Signs Information Representation - VITAL", CEN ENV13734, Final Draft, July 1999.

[2] D.F. Franklin, "Communication Layers of the MIB (Medical Information Bus)," IEEE Ninth Annual conference of the Engineering in Medicine and Biology Society, pp. 1213-1214, 1987.

[3] J. Rumbaugh, I. Jacobson, G. Booch, *The Unified Modeling Language Reference Manual*, Addison-Wesley series, 1998.

[4] Alschuler L, Brennan S, Rossi Mori A, Sokolowski R, Dudeck J. SGML/XML in healthcare information exchange standards. Proc SGML Europe '98. Graphic Communications Association, 1998: 425-33.

[5] Dolin E, et. al.: SGML as a Message Interchange Format in Healthcare, AMIA fall Symposium 1997.

[6] Dolin R, Alschuler A, Bray T, Mattison J. SGML as a message interchange format in healthcare.JAMIA Fall Symposium Supplement 1997: 635-9.

[7] Message Oriented Middleware Association: www.moma-inc.org

[8] Jagannathan V, Friedman S, Wreder K, Alsafadi Y, et al. XMI and transcription process automation. Proc HIMSS '99. 1999 February.

[9] Object Management Group (OMG): www.omg.org.

[10] Alschuler L. First do no harm. A standard for electronic communication in healthcare. Dudeck J et al (ed.). New Technologies in Hospital Information Systems. Vol. 45 Studies in Health Technology and Informatics. Amsterdam: IOS Press; 1997.

[11] A. Anagnostaki, E. Kyriacou, M. Reynolds, S. Pavlopoulos, A. Lymberis, D. Koutsouris, "Integration of CEN/TC251/PT5-021 "VITAL" preENV Standard and of "DICOM Supplement 30.0" into a Telemedicine System for Vital Signs Monitoring from Home", Proceedings of the Chicago 2000 World Congress on Medical Physics and Biomedical Engineering, Chicago, USA, 2000

[12] S. Pavlopoulos, A. Anagnostaki, D. Koutsouris, A. Lymberis, P. Levene, M. Reynolds, N. Georgiadis, C. Lambrinoudakis, D. Gritzalis, "Vital Signs Monitoring from Home with Open Systems", Proceedings of the MIE2000 - GMDS2000 Medical Infobahn for Europe, Hanover, Germany, 2000

Address of correspondence

Angie Anagnostaki
Biomedical Engineering Laboratory, Department of Electrical and Computer Engineering, National Technical University of Athens, 9 Iroon Polytechniou, 15773, Athens, Greece
Tel: +30 1 7722430, Fax: +30 1 7722431

E-mail: angiea@biomed.ntua.gr