Model Formulation ■

XML Schema Representation of DICOM Structured Reporting

K. P. LEE, PhD, JINGKUN HU, MS

Abstract **Objective:** The Digital Imaging and Communications in Medicine (DICOM) Structured Reporting (SR) standard improves the expressiveness, precision, and comparability of documentation about diagnostic images and waveforms. It supports the interchange of clinical reports in which critical features shown by images and waveforms can be denoted unambiguously by the observer, indexed, and retrieved selectively by subsequent reviewers. It is essential to provide access to clinical reports across the health care enterprise by using technologies that facilitate information exchange and processing by computers as well as provide support for robust and semantically rich standards, such as DICOM. This is supported by the current trend in the health-care industry towards the use of Extensible Markup Language (XML) technologies for storage and exchange of medical information. The objective of the work reported here is to develop XML Schema for representing DICOM SR as XML documents.

Design: We briefly describe the document type definition (DTD) for XML and its limitations, followed by XML Schema (the intended replacement for DTD) and its features. A framework for generating XML Schema for representing DICOM SR in XML is presented next.

Measurements: None applicable.

Results: A schema instance based on an SR example in the DICOM specification was created and validated against the schema. The schema is being used extensively in producing reports on Philips Medical Systems ultrasound equipment.

Conclusion: With the framework described it is feasible to generate XML Schema using the existing DICOM SR specification. It can also be applied to generate XML Schemas for other DICOM information objects.

■ *J Am Med Inform Assoc.* 2003;10:213–223. DOI 10.1197/jamia.M1042.

The Digital Imaging and Communications in Medicine (DICOM) Structured Reporting (SR) standard² improves the expressiveness, precision, and comparability of documentation about diagnostic images and waveforms. It supports the interchange of clinical reports in which critical features shown by images and waveforms can be denoted unambiguously by the observer, indexed, and retrieved selec-

tively by subsequent reviewers. Findings may be expressed by the observer as text, codes and numeric measurements, or via location coordinates of specific regions of interest within images or waveforms, or references to comparison images, sound, waveforms, curves, and previous report information. The observational and historical findings recorded by the observer may include any evidence referenced as part of an interpretation procedure. Thus, DICOM SR not only supports the reporting of diagnostic observations but also can document fully the evidence that evoked the observations. This move toward structured, coded reports presents new care-improvement opportunities beyond mere storage and distribution. Coding medical information for selective retrieval allows computers to assist in processing the increasingly large amounts of clinical data collected during

Affiliations of the authors: Philips Research, Briarcliff Manor, New York.

Correspondence and reprints: K. P. Lee, PhD, Philips Research, 345 Scarborough Road, Briarcliff Manor, NY 10510; e-mail: <kp.lee@philips.com>.

Received for publication: 10/30/01; accepted for publication: 09/16/02.

normal operation of a healthcare enterprise. When judiciously combined with knowledge-based information processing, we can significantly enhance clinical decision-making. As with any structured data in healthcare, benefits exist in outcome analysis and point-of-care applications.

Integrating the Healthcare Enterprise (IHE)⁸ is an ongoing initiative aiming at promoting and supporting the integration of healthcare systems to improve the efficiency and effectiveness of clinical practice. IHE defines a number of Integration Profiles, which are real-world situations that can be effectively managed by careful and consistent implementation and assembly of functionality provided by standards such as DICOM and others. Most profiles involve the exchange of information from various sources, and interoperability is key to successful achievement.

The DICOM specification, including SR, is maintained in Microsoft Word format and published as a PDF file. This format is unfortunately not amenable to machine processing and has the potential for misinterpretation. We believe that it is essential to provide access to clinical reports across the health care enterprise by using technologies that facilitate information exchange and processing by computers as well as provide support for robust and semantically rich standards, such as DICOM. For example, the HL7⁷ standard is developing its Clinical Document Architecture (CDA) for exchanging and processing electronic healthcare documents and uses Extensible Markup Language (XML)⁶ technologies. XML is a set of technologies that define a universal data format for tree-based, hierarchically formed information. A number of specifications extending its range and power, such as Extensible Stylesheet Language (XSL), Document Object Model (DOM), and XSL Transformations (XSLT), have already been developed. XML offers the advantages of platform independence and web awareness, and many XML tools are readily available. Thus XML technologies can provide a solution for enterprise-wide access to clinical information including medical reports. Having SR documents encoded in XML allows them to be validated for conformance to the standard, exchanged easily across the healthcare enterprise, and processed by computers.

To facilitate a uniform understanding of an XML encoding of medical reports, it is necessary to define a document type definition (DTD). Such a DTD has been derived from a Unified Modeling Language (UML) model of the DICOM SR information model.¹³ However, a DTD cannot fully represent DICOM data types and value representations because both UML

and DTD have inherent limitations. XML Schema¹⁴ from the World Wide Web Consortium (W3C) provides more expressive power and allows rich structure and data type definition (among others) in XML documents. We propose a framework to generate XML schemas for SR automatically from the DICOM SR specification to ease valid encoding of SR documents in XML.

The reader is assumed to have a working knowledge of the DICOM standard and XML technologies. D. Clunie's book¹ is an excellent source of information on SR.

Limitations of XML DTDs

A DTD is used to describe the permissible elements and attributes in an XML document, primarily in terms of structures and restrictions of "document-like" objects such as articles and books. DTDs work well for this purpose. However, XML is being applied more and more to nontraditional areas such as databases, interprocess communications, e-business, and medical information management.¹¹ These applications have much richer data type requirements that cannot be met by DTDs. XML Schema is intended to address some of the shortcomings of DTDs. For a short comparison of schemas and DTDs, see Mertz.¹⁰

DICOM data types are very rich in constraints. For instance, DICOM defines approximately 20 types of strings in terms of the maximum lengths they can have, but in contrast a DTD cannot even constrain the length of a string. DICOM supports numerous data types such as string, short, and float, whereas DTD supports only the string type. In addition, DICOM defines attribute* types with specific optionalities: Type 1, Type 1C, Type 2, Type 2C, and Type 3. Type 1 is mandatory and must have a value. Type 1C is treated as Type 1 when certain conditions are met, but is optional otherwise. Type 2 is mandatory but can be empty. Type 2C is treated as Type 2 under certain conditions. And Type 3 is optional and can be empty. DTD cannot specify such granularities of conditionality; only two possibilities exist: required or optional. Moreover, it cannot constrain an element or an attribute to always have a non-empty value.

*It is unfortunate that the same term "attribute" is used to mean different things in XML and in DICOM. However, both usage patterns are well established. The context will indicate which one is meant.

To illustrate such limitations, consider the DICOM attribute Patient's Name (0010,0010).[†] It is a Type 2 attribute with a value that is a string with maximum length of 64 characters. This attribute can be represented in an XML DTD as follows:

```
<!ELEMENT patients_name EMPTY>
<!--ATTLIST patients_name
CodingScheme CDATA #FIXED "DCM"
CodeId CDATA #FIXED "(0010,0010)"
ValueType CDATA #FIXED "string"
Value CDATA #REQUIRED
CodeMeaning CDATA #FIXED "Patient's Name" -->
```

However, it cannot specify that the length of the value of the Value attribute of patients_name cannot be more than 64 characters.

Another example is Patient's Age (0010,1010). In DICOM definition, the value of this attribute must consist of three digits followed by one of the letters 'Y', 'M', 'W' or 'D.' We can specify an XML element patients_age as follows in a DTD:

```
<!ELEMENT patients_age EMPTY>
<!--ATTLIST patients_age
CodingScheme CDATA #FIXED "DCM"
CodeId CDATA #FIXED "(0010,1010)"
ValueType CDATA #FIXED "string"
Value CDATA #REQUIRED
CodeMeaning CDATA #FIXED "Patient's Age" -->
```

But there is no standard way to define a pattern of three digits followed by 'Y', 'M', 'W' or 'D' in a DTD.

DTDs have other shortcomings that make them inconvenient to process programmatically. One is that they do not support namespaces,¹² which are becoming more and more important as the use of XML increases. Another is that a DTD is written with syntax different from that of an XML document, which means that DTDs cannot be processed easily by the variety of XML tools available.

XML Schemas have more capabilities than DTDs for specifying the validity of XML documents with rich structure and data-type definitions.

Features of XML Schema

XML Schema is an XML definition language for describing and constraining the content of XML doc-

uments. After two years of intensive work, XML Schema is now a W3C Recommendation.¹⁴ It extends the capabilities of DTD by providing more detailed descriptions of data types and constraints on valid entry values such as ranges. In addition, XML Schema is more object-oriented with support for the notion of inheritance and allows abstract types, super types, and subtypes.

Other features of XML Schema enhance its usability. It supports namespaces, which enable definitions and declarations from different vocabularies to be freely intermixed in an XML document. A schema can be divided into several schema documents when it becomes larger. This increases its readability and reusability and enables better access control and maintenance. XML Schemas are themselves valid XML documents, which makes it possible to use the same tools to validate both schemas and their instances.

The following sample XML Schema type definitions address the issues raised above. The first is a type definition that restricts the value of a Patient's Name DICOM attribute to be a string of no more than 64 characters (Figure 1).

The next example specifies that the value of Patient's Age must conform to a particular pattern (Figure 2).

More details on how these types are obtained can be found below.

A Framework for Generating XML Schemas for DICOM SR

DICOM SR defines three information object definitions (IODs): Basic, Enhanced, and Comprehensive. Each IOD contains several normalized or composite object classes called information entities (IEs). An IE consists of a number of Modules, which contain a set of atomic attributes, sequence attributes, or Macro attributes; and sequence attributes or Macro attributes themselves may also contain atomic attributes, sequence attributes, or Macro attributes, and so on recursively. Each IE, Module, or attribute is defined by tables in the DICOM specification and may have different usage, type, or value representation. Our approach to generating XML Schema for SR is to represent each table in XML using a straightforward encoding process and then transform the resulting XML files to schemas using XSLT technology. DICOM attribute names are converted to schema element names using a fixed set of conventions. For

[†]DICOM attributes are uniquely identified by a tag, which is a pair of numbers.

```

<xsd:element name="patients_name">
  <xsd:complexType content="empty">
    <xsd:attribute name="CodingScheme" use="fixed" value="DCM"/>
    <xsd:attribute name="CodeId" use="fixed" value="(0010,0010)"/>
    <xsd:attribute name="codeMeaning" use="fixed" value="Patient's Name"/>
    <xsd:attribute name="Value" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="64"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

```

Figure 1. Schema for patient's name.

example, the DICOM name Patient's Name is converted to the schema name patients_name. The schema generation process has two passes:

1. The XML files are processed once to produce one file containing data type definitions for all the attributes needed. This file is then sorted and pruned for duplicates, resulting in a single definition for each DICOM attribute. An advantage is that these definitions are now sorted in alphabetical order, making it easier to locate any specific data type definition.
2. A second pass is performed over the XML files to generate the final schemas. These schemas reuse the data type definitions obtained in the first pass.

The schema generation process is illustrated with an example below, and the generated top-level schema for the SR IOD is given in Appendix A. The set of generated schemas collectively represent a schema for SR. There are obviously interdependencies among these schemas. Schemas that are used by others will have to be included in the target schemas (as illustrated in Appendix A). In addition, SR has constraints that cannot be easily expressed in schema notation. These will have to be enforced by other means, *e.g.*, other constraining language, XSLT stylesheets or external functions written in a conventional programming language. Some preliminary results are reported by Lee.⁹

```

<xsd:element name="patients_age">
  <xsd:complexType content="empty">
    <xsd:attribute name="CodingScheme" use="fixed" value="DCM"/>
    <xsd:attribute name="CodeId" use="fixed" value="(0010,1010)"/>
    <xsd:attribute name="codeMeaning" use="fixed" value="Patient's Age"/>
    <xsd:attribute name="Value" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="[0-9]{3}[DWMY]"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
</xsd:element>

```

Figure 2. Schema for patient's age.

```

<xsd:schema targetNamespace = "http://www.philips.com/pms"
            xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
            xmlns:pms = "http://www.philips.com/pms">
<xsd:include schemaLocation = "patient_module.xsd"/>
    . . .
</xsd:schema>

```

Figure 3. Root element and namespaces of a schema.

Root Element and Namespaces

The root element of an XML schema is an element called schema in the XML Schema namespace (Figure 3). Its attributes define the target namespace and other namespaces used in the schema (and additional qualifications if necessary).

In this example, the target namespace is <http://www.philips.com/pms> as given by the value of the targetNamespace attribute and denoted as pms. <http://www.w3.org/2001/XMLSchema> is the namespace for the XML Schema Recommendation and abbreviated to xsd. Other namespaces used are <http://www.w3.org/1999/XSL/Transform>, the namespace for XSLT and abbreviated to xsi, and <http://www.w3.org/2001/XMLSchema-Instance>, the namespace for schema instances and abbreviated to xsi.

A schema may include one or more external schemas. The above example shows the inclusion of a schema called patient_module.xsd.

Generation Example

We illustrate the schema generation process by showing how a DICOM macro is transformed. Table 1 is a

sample macro—Numeric Measurement Macro (C.18.1–1 from reference 3). This is a typical macro that contains an atomic attribute, two sequence attributes, and a nested macro.

This macro is represented in XML in a straightforward way as shown in Appendix B. The resulting schema for a complexType called numeric_measurement_macro is listed in Appendix C and a portion is reproduced in Figure 4.

The following points should be noted:

1. A DICOM Sequence (such as the Measured Value Sequence) is represented as a sequence of XML elements (measured_value_sequence_item) in order to accommodate DICOM attributes with multiple values. In this case the measured_value_sequence_item type is defined in Appendix D.
2. The Measured Value Sequence is of Type 2. This is reflected in the attribute nillable="true" in the schema element measured_value_sequence. Such an element is required to be present in any valid XML instance document, but can have an empty value. As is noted in the specification: "The Measured Value Sequence (0040,A300) may be

Table 1 ■

Numeric Measurement Macro Attributes

Attribute Name	Tag	Type	Attribute Description
Measured value sequence	(0040,A300)	2	This is the value of the Content Item. Shall consist of a Sequence of Items conveying the measured value(s), which represent integers or real numbers and units of measurement. Zero or one Items shall be permitted in this sequence.
>Numeric value	(0040,A30A)	1	Numeric measurement value. Only a single value shall be present.
>Measurement units code sequence	(0040,08EA)	1	Units of measurement. Only a single Item shall be permitted in this sequence.
>>Include 'Code Sequence Macro' Table 8.8-1			Defined Context ID is 82.

```

<!--Definition of numeric_measurement_macro-->
<xsd:complexType name="numeric_measurement_macro">
  <xsd:sequence>
    <xsd:element name="measured_value_sequence" nillable="true">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="measured_value_sequence_item"
            type="pms:measured_value_sequence_item" minOccurs="0"
            maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="VR" fixed="SQ"/>
        <xsd:attribute name="codingScheme" fixed="DCM"/>
        <xsd:attribute name="codeId" fixed="(0040,A300)"/>
        <xsd:attribute name="codeMeaning" fixed="Measured Value Sequence"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

```

Figure 4. numeric_measurement_macro complexType.

empty to convey the concept of a measurement whose value is unknown or missing." Such a case will appear in an instance XML document as `<measured_value_sequence xsi:nil="true"/>`.

3. The table specifies that "zero or one Items shall be permitted" in a Measured Value Sequence. In the generated schema this is represented with the

minOccurs attribute set to 0 and the maxOccurs attribute set to 1 for the element `measured_value_sequence_item`.

4. We have added four schema attributes to all generated elements that identify uniquely the original DICOM attributes. They are VR, codingScheme, codeId and codeMeaning, corresponding to the

```

<xsd:complexType name="measurement_units_code_sequence">
  <xsd:sequence>
    <xsd:element name="measurement_units_code_sequence_item"
      type="pms:measurement_units_code_sequence_item"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="VR" use="fixed" value="SQ"/>
  <xsd:attribute name="codingScheme" use="fixed" value="DCM"/>
  <xsd:attribute name="codeId" use="fixed" value="(0040,08EA)"/>
  <xsd:attribute name="codeMeaning" use="fixed"
    value="Measurement Units Code Sequence"/>
</xsd:complexType>
<xsd:complexType name="measurement_units_code_sequence_item">
  <xsd:sequence>
    <xsd:element name="code_sequence_macro"
      type="pms:code_sequence_macro"/>
  </xsd:sequence>
</xsd:complexType>

```

Figure 5. measurement_units_code_sequence complexType.

```

<xsl:template name="do_include">
  <xsl:comment>Start of external schemas</xsl:comment>
  <xsd:include schemaLocation="dataTypes.xsd"/>
  <!-- include each schema only once and do not include if there is a recur-
sive reference -->
  <xsl:for-each select="//include[not(.=preceding::include) and not(name =
/module_attributes/name) and not(name = /macro_attributes/name)]">
    <xsl:sort select="."/>
    <xsl:element name="xsd:include">
      <xsl:attribute name="schemaLocation"><xsl:value-of select="concat(name,
'.xsd')"/></xsl:attribute>
    </xsl:element>
  </xsl:for-each>
  <xsl:comment>End of external schemas</xsl:comment>
</xsl:template>

```

Figure 6. XSLT template for including required schemas.

Value Representation of a DICOM attribute, the coding scheme used, and the unique code ID as well as a human readable code meaning. All are obtained from Part 6—Data Dictionary.⁴ These attributes are strictly speaking not necessary but we have decided to include them to provide an extra level of validity checking in an XML instance document.

This example also shows how a macro included in a table is processed. The Numeric Measurement Macro has a Measurement Units Code Sequence which includes a Code Sequence Macro (Table 8.8-1 of reference 3). The type definition for measurement_units_code_sequence is given in Figure 5. The included macro becomes an element code_sequence_macro of type code_sequence_macro, which is defined separately and simply included (see the next subsection).

Including External Schemas

The definition of an IOD, IE, Module, or Macro frequently refers to other Modules or Macros. These included Modules and Macros are themselves defined by their own tables and therefore need not be defined further in the including table. All that needs to be done is to include the corresponding schemas. The schema for the SR IOD given in Appendix A is such an example. Types for all the included modules are defined elsewhere and simply included here.

To generate the appropriate include instructions in the schema, the transforming stylesheet must look through the XML file for an IOD, IE, Module or Macro and determine which schemas need to be

included. A schema should be included only once even if there are multiple references in a table. If there is a recursive reference, then it has to be detected to avoid a schema trying to include itself. The template to process inclusion is called do_include and reproduced in Figure 6.

This is extracted from the stylesheet to transform modules and macros. There is a corresponding template for the stylesheet for transforming IODs.

SR XML Schemas and Validation

We have developed XSLT stylesheets to implement the framework described. The XSLT processor together with stylesheets generates an XML schema for SR automatically. The only manual intervention is choosing a VR when a DICOM attribute has a choice of VR (e.g. Pixel Padding Value (0028,0120) has a VR of US or SS).

The informative SR Content Tree example in Annex K of reference 3 has been encoded in XML and validated against the generated schemas. The schemas are currently used extensively in creating reports produced by Philips Medical System ultrasound equipment.

Conclusions and Future Work

We have proposed a representation of the DICOM SR IOD in XML and a framework for automatically generating XML schemas from the representation. It addresses processing rules for IODs, IEs, Modules, Macros, atomic attributes, sequence attributes, and

their usages/types. These schemas will benefit XML implementation of DICOM SR for any medical equipment supplier. The framework can also be used for other DICOM IODs such as Waveforms. However, there are aspects of this framework that need further refinement to enhance the usefulness of the generated schemas. Among these:

1. Representation of DICOM Value Representation currently takes into account only the length attribute and some patterns. Other constraints need to be incorporated into the schema. For example, certain control characters must be excluded from some string types.

2. Many DICOM constraints are difficult or impossible to specify with a schema. There are additional mechanisms to increase the expressive powers of constraint specification, but because of the nature of some DICOM constraints, a programming approach seems to be the only way to address all the constraints. Some preliminary results have been reported by Lee.⁹

We are also extending this framework to allow automatic generation of XML schema for SR templates as described in reference 5.

References ■

1. Clunie DA. DICOM Structured Reporting. Bangor, PA, PixelMed Publishing, 2000.
2. Digital Imaging and Communications in Medicine: <http://medical.nema.org>.
3. DICOM Part 3—Information object definitions. Available at: http://medical.nema.org/dicom/2001/01_03PU.PDF.
4. DICOM Part 6—Data dictionary. Available at: http://medical.nema.org/dicom/2001/01_06PU.PDF.
5. DICOM Part 16—Content Mapping Resource, http://medical.nema.org/dicom/2001/01_16PU.PDF.
6. Extensible Markup Language (XML) 1.0 W3C Recommendation 10 February 1998. Available at <http://www.w3c.org>.
7. Health Level Seven: <http://www.hl7.org>.
8. Integrating the Healthcare Enterprise: <http://www.rsna.org/IHE>.
9. Lee KP. Expressing DICOM SR constraints in XML. In Lemke HU, et al (eds): Proceedings of CARS 2002. New York, Springer-Verlag, 2002, pp 491-496.
10. Mertz D. Comparing W3C XML Schemas and Document Type Definitions (DTDs). Available at <http://www-106.ibm.com/developerworks/library/x-matters7.html>.
11. Mikula N, Levy K. Schemas take DTDs to the Next Level—Use schemas to prepare for the demands of e-business and e-commerce. Available at: <http://www.xmlmag.com/upload/free/features/xml/1999/01win99/nmwin99/nmwin99.asp>.
12. Namespaces in XML, World Wide Web Consortium 14 January 1999. Available at: <http://www.w3.org/TR/1999/REC-xml-names-19990114/>.
13. Tirado-Ramos A, Hu J, Lee KP. IOD-based UML representation of DICOM structured reporting: A case study on transcoding DICOM to XML. J Am Med Inform Assoc 2002;9: 63-71.
14. XML Schema: <http://www.w3c.org/XML/Schema>.

Appendix A. Generated Top-Level Schema for SR

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace=http://www.philips.com/pms
  xmlns:xsd=http://www.w3.org/2001/XMLSchema
  xmlns:pms=http://www.philips.com/pms>
  <xsd:annotation>
    <xsd:documentation>SR IOD (Section A.35.1.1)</xsd:documentation>
  </xsd:annotation>
  <!--Start of external schemas-->
  <xsd:include schemaLocation="dataTypes.xsd"/>
  <xsd:include schemaLocation="typeDefs.xsd"/>
  <xsd:include schemaLocation="general_equipment_module.xsd"/>
  <xsd:include schemaLocation="general_study_module.xsd"/>
  <xsd:include schemaLocation="patient_module.xsd"/>
  <xsd:include schemaLocation="patient_study_module.xsd"/>
  <xsd:include schemaLocation="sop_common_module.xsd"/>
  <xsd:include schemaLocation="specimen_identification_module.xsd"/>
  <xsd:include schemaLocation="sr_document_content_module.xsd"/>
  <xsd:include schemaLocation="sr_document_general_module.xsd"/>
  <xsd:include schemaLocation="sr_document_series_module.xsd"/>
  <!--End of external schemas-->
  <xsd:element name="sr_iod">
    <xsd:complexType>
      <xsd:sequence>
```



```

        <xsd:element name="patient_ie" type="pms:patient_ie"/>
        <xsd:element name="study_ie" type="pms:study_ie"/>
        <xsd:element name="series_ie" type="pms:series_ie"/>
        <xsd:element name="equipment_ie" type="pms:equipment_ie"/>
        <xsd:element name="document_ie" type="pms:document_ie"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="patient_ie">
    <xsd:sequence>
        <xsd:element name="patient_module"
type="pms:patient_module"/>
        <xsd:element name="specimen_identification_module"
type="pms:specimen_identification_module" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="study_ie">
    <xsd:sequence>
        <xsd:element name="general_study_module"
type="pms:general_study_module"/>
        <xsd:element name="patient_study_module"
type="pms:patient_study_module" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="series_ie">
    <xsd:sequence>
        <xsd:element name="sr_document_series_module"
type="pms:sr_document_series_module"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="equipment_ie">
    <xsd:sequence>
        <xsd:element name="general_equipment_module"
type="pms:general_equipment_module"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="document_ie">
    <xsd:sequence>
        <xsd:element name="sr_document_general_module"
type="pms:sr_document_general_module"/>
        <xsd:element name="sr_document_content_module"
type="pms:sr_document_content_module"/>
        <xsd:element name="sop_common_module"
type="pms:sop_common_module"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

Appendix B. XML Representation of Numeric Measurement Macro

```

<?xml version="1.0" encoding="UTF-8"?>
<macro_attributes>
    <reference>C.18.1</reference>
    <attribute_name>NUMERIC MEASUREMENT Macro</attribute_name>
    <name>numeric_measurement_macro</name>
    <sequence_attribute minOccurs="0" maxOccurs="1">

```

```

    <attribute_name>Measured Value Sequence</attribute_name>
    <name>measured_value_sequence</name>
    <tag>(0040,A300)</tag>
    <type>2</type>
    <description>This is the value of the Content Item. Shall consist of
a Sequence of Items conveying the measured value(s), which represent integers or
real numbers and units of measurement. Zero or one Items shall be permitted in
this sequence.</description>
    <atomic_attribute>
        <attribute_name>Numeric Value</attribute_name>
        <name>numeric_value</name>
        <tag>(0040,A30A)</tag>
        <type>1</type>
        <vm>1</vm>
        <description>Numeric measurement value. Only a single value
shall be present.</description>
    </atomic_attribute>
    <sequence_attribute minOccurs="1" maxOccurs="1">
        <attribute_name>Measurement Units Code
Sequence</attribute_name>
        <name>measurement_units_code_sequence</name>
        <tag>(0040,08EA)</tag>
        <type>1</type>
        <description>Units of measurement. Only a single Item shall be
permitted in this sequence.</description>
        <include>
            <attribute_name>Code Sequence Macro</attribute_name>
            <name>code_sequence_macro</name>
        </include>
    </sequence_attribute>
</sequence_attribute>
</macro_attributes>

```

Appendix C. Generated Schema for Numeric Measurement Macro

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.philips.com/pms"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:pms="http://www.philips.com/pms">
    <xsd:annotation>
        <xsd:documentation>NUMERIC MEASUREMENT Macro (Section
C.18.1)</xsd:documentation>
    </xsd:annotation>
    <!--Start of external schemas-->
    <xsd:include schemaLocation="code_sequence_macro.xsd"/>
    <!--End of external schemas-->
    <!--Definition of numeric_measurement_macro-->
    <xsd:complexType name="numeric_measurement_macro">
        <xsd:sequence>
            <xsd:element name="measured_value_sequence" nillable="true">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element

```

```

name="measured_value_sequence_item"
  type="pms:measured_value_sequence_item"
  minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="VR" fixed="SQ"/>
    <xsd:attribute name="codingScheme" fixed="DCM"/>
    <xsd:attribute name="codeId" fixed="(0040,A300)"/>
    <xsd:attribute name="codeMeaning"
fixed="Measured Value Sequence"/>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

Appendix D. Type for Measured Value Sequence Elements

```

<xsd:complexType name="measured_value_sequence_item">
  <xsd:sequence>
    <xsd:element name="numeric_value" type="pms:numeric_value"/>
    <xsd:element name="measurement_units_code_sequence">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element
name="measurement_units_code_sequence_item"
  type="pms:measurement_units_code_sequence_item"/>
        </xsd:sequence>
        <xsd:attribute name="VR" fixed="SQ"/>
        <xsd:attribute name="codingScheme" fixed="DCM"/>
        <xsd:attribute name="codeId" fixed="(0040,08EA)"/>
        <xsd:attribute name="codeMeaning"
fixed="Measurement Units Code Sequence"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

```