

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
18 April 2002 (18.04.2002)

PCT

(10) International Publication Number
WO 02/31688 A2

(51) International Patent Classification⁷: **G06F 17/30**

Holstlaan 6, NL-5656 AA Eindhoven (NL). **ALSAFADI, Yasser**; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(21) International Application Number: PCT/EP01/11152

(22) International Filing Date:

26 September 2001 (26.09.2001)

(74) Agent: **COHEN, Julius, S.**; Internationaal Octrooibureau B.V., Prof Holstlaan 6, NL-5656 AA Eindhoven (NL).

(25) Filing Language:

English

(81) Designated States (*national*): JP, KR.

(26) Publication Language:

English

(84) Designated States (*regional*): European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

(30) Priority Data:

09/686,401

10 October 2000 (10.10.2000) US

Published:

— without international search report and to be republished upon receipt of that report

(71) Applicant: **KONINKLIJKE PHILIPS ELECTRONICS N.V.** [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(72) Inventors: **TIRADO-RAMOS, Alfredo**; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL). **HU, Jingkun**; Prof.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.



WO 02/31688 A2

(54) Title: UML MODEL AND XML REPRESENTATIONS OF DIGITAL IMAGING AND COMMUNICATIONS IN MEDICINE STRUCTURED REPORTS (DICOM SR)

(57) Abstract: A modeling technique is presented for converting the DICOM SR specification into a UML-based Object-Oriented representation of the specification, and for converting DICOM SR reports into UML-compatible representations. Using this methodology, open exchangeable representations using the World Wide Web Consortium XML DTD and Schema specifications are also provided. This methodology for representing the DICOM SR specification provides a clear and comprehensive view of DICOM's semantically rich framework, its structures, and constructs. By providing a mapping between DICOM SR and UML, XML DTD and Schema, developers, analysts, and system architects will be better able to understand the DICOM SR specification, and better able to define and develop DICOM SR-aware applications.

UML model and XML representations of digital imaging and communications in medicine structured reports (DICOM SR)

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of modeling and data representation, and in particular to the modeling and representation of medical reports, via the use of DICOM SR
5 relational data.

2. Description of Related Art

The Digital Imaging and Communications in Medicine (DICOM) Structured Reporting (SR) standard, and the SR Documentation Model upon which it is based, improves
10 the expressiveness, precision, and comparability of documentation of diagnostic images and waveforms. DICOM SR supports the interchange of expressive compound reports in which the critical features shown by images and waveforms can be denoted unambiguously by the observer, indexed, and retrieved selectively by subsequent reviewers. Findings may be expressed by the observer as text, codes, and numeric measurements, or via location
15 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 supports not only the reporting of diagnostic observations, but the capability to document fully the evidence that
20 evoked the observations. This capability provides significant new opportunities for large-scale collection of structured data for clinical research, training, and outcomes assessment as a routine by-product of diagnostic image and waveform interpretation, and facilitates the pooling of structured data for multi-center clinical trials and evaluations.

The DICOM SR is based on a relational data technology, and has been
25 standardized by the National Electrical Manufacturers Association (NEMA). *Supplement 23: Structured Reporting Storage SOP Classes* to the DICOM Standard, published by the DICOM Standards Committee, 1300 N. 17th Street, Rosslyn, VA 22209 USA, introduces the SR Service-Object Pair (SOP) Classes for transmission and storage of documents that describe or refer to any number of images or waveforms or to the specific features that they

contain, and is incorporated by reference herein. This standard is expected to be adopted by the medical equipment manufacturers and providers at large, to provide text, image, and waveform content in a structured reporting format.

Although the DICOM SR standard provides for a consistent reporting and recording scheme, the use of the information contained in a DICOM SR is limited to DICOM compliant applications that can process this information using the DICOM specific format. Application developers must be DICOM literate, and a methodology for deploying applications that interoperate with other applications outside the DICOM domain has not yet been developed.

In the computer industry, progress has been made in the use of standardized languages and methodologies that facilitate the use of information from a variety of sources by a variety of applications. The Object-Oriented processing paradigm is particularly well suited for facilitating the sharing of information, as well as the sharing of processes that use the information, among heterogeneous systems. The Object Management Group (OMG) has formulated the Unified Modeling Language (UML) for exchanging information that is particularly well suited for Object-Oriented processing. The UML is a language for specifying, constructing, visualizing, and documenting objects produced within software-intensive systems, as well as for business modeling and other non-software systems. The UML is independent of any particular programming language, but provides a formal and consistent basis for expressing, exchanging, and processing information within the context of the specified object-oriented model.

Another standard language that is well suited for object-oriented processing is the World Wide Web Consortium Extensible Markup Language (XML), which is derived from the Standard Generalized Markup Language (SGML), and is designed to structure data so that it can be easily transferred over a network and consistently processed by the receiver. Because XML is used to describe information as well as structure, it is particularly well suited as a data description language. The XML is a metalanguage that allows one to describe other languages, wherein the other languages include the definitions of how information in a document will appear in an application that displays it, and other details. One of XML's particular strengths is that it lets entire industries, academic disciplines, and professional organizations develop sets of Document Type Definitions (DTDs) that can serve to standardize the presentation of information within those disciplines.

BRIEF SUMMARY OF THE INVENTION

Although UML and XML are relatively new and specialized languages, it can be expected that more programmers and other computer professionals will be familiar with UML and XML than those who are familiar with DICOM. Additionally, it can be expected
5 that more general-purpose utilities and applications will be available for use on UML and XML encoded information than will be available for use on DICOM SR encoded information.

It is, therefore, an object of this invention to provide a method and system that facilitates the creation of UML representations of DICOM SR representations and associated
10 information. It is a further object of this invention to provide a method and system that facilitates the creation of XML representations of DICOM SR representations and associated information.

These objects and others are achieved by providing a modeling technique to convert DICOM SR information from the DICOM relational model into an Object-Oriented
15 representation. A methodology is presented for converting the DICOM SR specification into a UML-based Object-Oriented representation of the specification, and for converting DICOM SR reports into UML-compatible representations. Using this methodology, open exchangeable representations using the World Wide Web Consortium XML DTD and Schema specifications are also provided. This methodology for representing the DICOM SR
20 specification provides a clear and comprehensive view of DICOM's semantically rich framework, its structures, and constructs. By providing a mapping between DICOM SR and UML, XML DTD and Schema, developers, analysts, and system architects will be better able to understand the DICOM SR specification, and better able to define and develop DICOM SR-aware applications.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is explained in further detail, and by way of example, with reference to the accompanying drawings wherein:

FIG. 1 illustrates an example flow diagram for converting the DICOM SR
30 specification into UML and XML representations that facilitate program development in accordance with this invention.

FIG. 2 illustrates an example flow diagram for converting the DICOM SR reports into UML and XML representations for processing by UML and XML enabled applications in accordance with this invention.

FIG. 3 illustrates an example DICOM SR to UML converter in accordance with this invention.

FIG. 4 illustrates an example flow diagram for converting a DICOM SR attribute to a UML attribute in accordance with this invention.

5 FIG. 5 illustrates an example flow diagram for converting a DICOM SR recursive data element to a set of UML elements in accordance with this invention.

FIG. 6 illustrates an example UML representation of a DICOM SR Information Object Definition (IOD) Module in accordance with this invention.

10 FIG. 7A illustrates an example XML Document Type Definition (DTD) representation of a UML representation of a DICOM SR Information Object Definition (IOD) Module in accordance with this invention.

FIG. 7B illustrates an example XML Schema representation of a UML representation of a DICOM SR Information Object Definition (IOD) Module in accordance with this invention.

15 Throughout the drawings, the same reference numerals indicate similar or corresponding features or functions.

DETAILED DESCRIPTION OF THE INVENTION

20 FIG. 1 illustrates an example flow diagram for converting the DICOM SR specification 110 into UML 130 and XML 170 representations that facilitate program development in accordance with this invention. As noted above, although applications can be developed that utilize DICOM's relational structured reporting scheme directly, it can be expected that the number of programmers and other computer professions who are familiar with UML and XML and object-oriented technologies and techniques will be substantially
25 greater than those who are familiar with DICOM and relational technologies and techniques. This invention is based on the premise that DICOM-related application programs can be more efficiently developed as UML and XML enabled applications. This efficiency is gained by presenting the DICOM SR structure in the context of a UML structure, or an XML structure, thereby eliminating the learning curve and context-switch difficulties typically
30 encountered when dealing with new languages. As is known in the art, the assimilation of the rules, conventions, idiosyncrasies, etc. of a language generally comes with time and experience. In like manner, the appreciation of the interrelationships among data items and entities is highly dependent upon an appreciation of the interactions and dependencies that are implied by the modeling language used to express these interrelationships. By presenting

the DICOM SR specification as a UML or XML representation, the experiences of the UML or XML programmer, systems analyst, technical writer, engineer, manager, etc. are advantageously used, thereby potentially reducing the costs associated with an application program development.

5 As illustrated in FIG. 1, a converter 120 converts the elements of the DICOM SR specification 110 into elements of a UML structure 130. This UML structure 130 can be used to facilitate the development 140 of an application program 150 that uses the UML structure 130, or it can be further converted, via converter 160, into elements of an XML Document Type Definition (DTD) and/or an XML Schema, collectively illustrated at 170.
10 The XML DTD and Schema 170 facilitates the development 180 of an XML-enabled application program 190.

 Note that the UML application program development 140 and the XML application program development 180 can be expected to require fewer resources than an application program development that uses the DICOM SR specification directly. A number
15 of UML and XML utility routines can be expected to be available for use in this development, based on the increasing use of UML and XML in the computer industry. Also, both UML and XML are object-oriented languages, and an objective of the object-oriented paradigm is to facilitate the transport and re-use of object-oriented software.

20 FIG. 2 illustrates an example flow diagram for converting the DICOM SR reports 210 into UML 230 and XML 270 documents for processing by the aforementioned UML 150 and XML 190 enabled applications, in accordance with this invention. That is, the programs 150, 190 whose development was facilitated by the DICOM converters 120, 160 of this invention, or other UML and XML programs, can be used to process individual DICOM
25 SR reports 210, merely by converting the reports 210 into corresponding UML 230 and XML 270 documents. Note that the term "document" is used herein for ease of reference, and includes any of a variety of embodiments, including data that is temporarily or permanently stored in a file, data that is stored in memory, or on a disk, data that is communicated among processing systems, and so on. The UML 150 or XML 190 applications may include, for
30 example, applications that render 280 the content of the individual SR report 210 for viewing by a diagnostician, applications that transfer the content of the individual SR report 210 to processing facilities that do not have DICOM SR enabled applications, applications that collect the information from individual SR reports 210 and perform analyses for clinical studies, and so on.

FIG. 3 illustrates an example DICOM SR to UML converter 300. Example DICOM entities 311- 317 are illustrated on the left of the converter 300, and corresponding UML entities 321-327 are illustrated on the right of the converter 300. In accordance with this invention, particular rules are associated with the various DICOM SR entity types to effect the conversion.

The DICOM SR includes five "Information Entities (IE)" 311: Patient, Study, Series, Equipment, and Document. The rule for creating corresponding UML entities is to create a UML "Class" 321 corresponding each IE 311, wherein the class assumes the same name as the corresponding IE, except that the UML class corresponding to the Patient IE is named "Patients", because, as discussed below, another DICOM entity also has the name "Patient", whereas UML requires a unique naming of each class. Alternative techniques for generating unique names, such as appending an incrementing numeral to a name, are common in the art.

The DICOM SR includes nine "Information Object Definition (IOD) Modules" 312: Patient, Specimen Identification, General Study, SR Document Series, General Equipment, SR Document General, SR Document Content, and SOP Common. The rule for creating corresponding UML entities is to create a UML "Class" 322 corresponding each IOD Module 312, wherein the class assumes the same name as the corresponding IOD module 312, except that blank spaces, if any are removed from the corresponding UML class name, because UML requires a single contiguous name for each class. Alternatively, a substitute character can be used to replace blank characters.

The DICOM SR includes "Macros" 313. The rule for creating corresponding UML entities is to create a UML "Class" 323 corresponding each Macro 313, wherein the class assumes the same name as the corresponding Macro 313, except that blank spaces, if any, are removed from the corresponding UML class name, as well as the postfix "Macro", as illustrated in FIG. 3, wherein the DICOM SR "SOP Instance Macro" 313' is converted to a UML Class named "SOPInstance" 323'. Alternatively, a substitute character can be used to replace blank characters, and/or the Macro postfix can be retained.

The DICOM SR includes "Attributes" 314. The rule for creating corresponding UML entities is to create a UML "Class Attribute" 324 corresponding each Attribute 314, respectively.

FIG. 4 illustrates an example flow diagram for converting DICOM SR Attribute names to UML Class Attribute names. The DICOM SR attribute name is initially

copied to form the UML attribute name, at 410, and converted to lower case, at 420. Via the loop 430-480, each blank and hyphen character is identified, at 440, and replaced with an underscore character, at 450, and each apostrophe or bracket character is identified, at 460, and deleted from the UML attribute name, at 470. The resultant UML attribute name is stored, at 490.

Returning to FIG. 3, each DICOM SR sequence attribute 314' is mapped to a corresponding UML class attribute 324', using the rules illustrated in FIG. 4. The sub-attributes 315 under a specific sequence 315' are mapped into class attributes 325 in a class 325' with the same name as the sequence, except that the postfix "Sequence" is removed. For example, the DICOM SR "Verifying Observer Sequence" 315' includes sub-attributes "Verifying Observer Name" and "Verifying Organization". These sub attributes form class attributes "verifying_observer_name" and "verifying_organization", respectively, in a class that is named "VerifyingObserver" 325'. In this manner, the class 325' that is formed from the DICOM SR sequence 315' has a unique name that is different from the attribute name 324' that is formed from the DICOM SR sequence attribute 314'.

DICOM defines a value representation (VR) for each attribute that offers a DICOM-internal primitive data type. The DICOM value representations include: Application Entity (AE), Age String (AS), Attribute Tag (AT), Code String (CS), Date (DA), Decimal String (DS), Date Time (DT), Floating Point Single (FL) and Double (FD), and Integer String (IS). For DICOM SR atomic attributes, such value representations are used; for DICOM SR composite attributes, such as sequence types, the associated classes serve as their types.

The DICOM SR includes a variety of "Constraints" 317, including mandatory, conditional, and other varieties. The rule for creating corresponding UML entities is to map mandatory and conditional DICOM SR constraints to the UML stereotypes of "required" and "conditional", respectively, and to map all other DICOM SR constraints into the UML stereotype of "optional". Alternatively, these DICOM SR constraints may be mapped to the UML via representations that are provided the Object Constraint Language (OCL) and others.

The DICOM SR enables Content Items unlimited recursion. FIG. 5A illustrates this recursion in the SR Document Content module 500, via the Document Relationship Macro 520. As illustrated, the Document Relationship Macro 520 may include a Content module 530, and this Content module 530 may contain a document relationship 532 that includes another Content module 540, and so on. In accordance with this invention, as

illustrated in FIG. 5B, the SR Document Content UML representation 560 includes elements DocumentRelationship 570 and DocumentContent 590, and the Document Relationship representation 570 and the Content representation 580 each reference each other, thereby effecting a recursive relationship.

5

Using the above rules for mapping the elements of DICOM SR into corresponding elements of UML, DICOM SR specification documents and DICOM SR content documents are converted into a modeling language that is more often used by computer professionals, thereby providing the opportunity to ease the task and cost of producing application programs that can be used for processing DICOM related material.

10

FIG. 6 illustrates an example UML representation of portions of DICOM SR "SR Document General" Information Object Definition (IOD) Module, using the above described rules for converting DICOM SR elements into UML representations. Note that the representation of FIG. 6 is not an exhaustive representation of the DICOM SR "SR Document General" IOD Module, but is presented for illustration purposes. The upper-level block 610 illustrates the basic definition of the DICOM SR "SR Document General". At this upper-level 610, the UML "SRDocumentGeneral" class contains a number of sequences that are subsequently referenced as other UML classes 620, 630, and 640 that are related to the upper-level class 610 via the UML "uses" operator. The classes may be nested, as illustrated by the class "ReferencedSeries" 650 which is used by the "SOPInstanceReference" 630. As also illustrated in FIG. 6, a class 660, 670 may be used by a plurality of classes (620, 650) and (620, 640) respectively.

15

20

25

To further ease the task and cost of producing DICOM-related application programs, the UML representation that is provided by the converter 300 in accordance with this invention is further converted into an XML DTD and Schema representation, as shown in FIGs. 1 and 2, and further detailed below with regard to FIGs. 7A and 7B.

30

FIG. 7A illustrates an example XML Document Type Definition (DTD) 701, and FIG. 7B illustrates an example XML Schema 702, each corresponding to the example UML "SRDocumentGeneral" class 610 in FIG. 6. The XML representation corresponding to a DICOM SR document is converted into an XML DTD 701 and XML Schema 702 via the following rules:

- All UML Classes are mapped to XML DTD Elements.

- All UML Class Attributes are mapped to XML DTD Elements.
- All UML Association and Uses relationships are mapped to XML DTD

Element Definition components.

- All atomic attributes are mapped to an XML Element with four attributes:

5 coding_scheme, code_id, type, and value.

The foregoing merely illustrates the principles of the invention. It will thus be appreciated that those skilled in the art will be able to devise various arrangements which, although not explicitly described or shown herein, embody the principles of the invention and are thus within the spirit and scope of the following claims.

CLAIMS:

1. A method for mapping a DICOM SR document (110, 210) into a UML document (130, 230), comprising:
 - mapping each DICOM Information Entity (311) in the SR document (110, 210) into a corresponding UML class (321) in the UML document (130, 230),
 - 5 mapping each DICOM IOD Module (312) in the SR document (110, 210) into a corresponding UML class (322) in the UML document (130, 230),
 - mapping each DICOM Macro (313) in the SR document (110, 210) into a corresponding UML class (323) in the UML document (130, 230), and
 - mapping each (314) in the SR document (110, 210) into a corresponding UML
 - 10 attribute (324) in the UML document (130, 230).
2. The method as claimed in claim 1, wherein
 - each DICOM Information Entity (311) has an IE name, and
 - each DICOM Information Entity (311) is mapped into the corresponding UML
 - 15 class (321) with a class name that corresponds to the IE name,
 - except that,
 - if the IE name is "patient", mapping the DICOM Information Entity (311) into the corresponding UML class (321) with a class name that is unique and different from "patient".
 - 20
3. The method as claimed in claim 1, wherein
 - each DICOM IOD Module (312) has a module name, and
 - each DICOM IOD Module (312) is mapped into the corresponding UML class
 - (322) with a class name that corresponds to the module name, except that space characters
 - 25 are removed from the class name.
4. The method as claimed in claim 1, wherein
 - each DICOM Macro (313) has a macro name having a "Macro" postfix, and

each DICOM Macro (313) is mapped into the corresponding UML class (323) with a class name that corresponds to the macro name, except that space characters and the "Macro" postfix are removed from the class name.

- 5 5. The method as claimed in claim 1, wherein
 each DICOM Attribute (314) has a DICOM attribute name, and
 each DICOM Attribute (314) is mapped into the corresponding UML attribute
 (324) with a UML attribute name that corresponds to the DICOM attribute name,
 except that,
10 each upper case letter is replaced in the UML attribute name with a
 corresponding lower case letter,
 each blank space is replaced in the UML attribute name with an underscore
 character,
 each hyphen character is replaced in the UML attribute name with an
15 underscore character,
 each apostrophe character is removed from the UML attribute name, and
 each bracket character is removed from the UML attribute name.

6. The method of claim 1, further comprising:
20 mapping each DICOM SR recursive element (500) into a pair of UML
 elements (570, 580) that reference each other.

7. An electronic document for processing via a computer system that corresponds
 to the UML document (130, 230) produced by the method of claim 1.

25

8. A computer program, configured to be executed on a computer system, that
 includes the UML document (130, 230) produced by the method of claim 1.

9. A method for mapping a DICOM SR atomic attribute into a UML class
30 attribute, comprising:
 copying (410) each lower case letter from the atomic attribute to the class
 attribute,
 copying (420) each upper case letter from the atomic attribute to the class
 attribute as a corresponding lower case letter,

copying (450) each blank space from the atomic attribute to the class attribute as an underscore character, and

copying (450) each hyphen character from the atomic attribute to the class attribute as an underscore character.

5

10. A method for creating an XML-compatible representation (701, 702) that corresponds to a DICOM SR document (110, 210), comprising:

mapping the DICOM SR document (110, 210) into a corresponding UML document (130, 230), and

10 mapping the UML document (130, 230) into a corresponding XML document (170, 270) that forms the XML-compatible representation (701, 702).

11. The method of claim 10, wherein the corresponding XML document (170, 270) includes at least one of: an XML Document Type Definition (DTD) (701) and an XML
15 Schema (702).

12. An electronic document for processing via a computer system that corresponds to the XML document (170, 270) produced by the method of claim 10.

20 13. A computer program, configured to be executed on a computer system, that includes the XML document (170, 270) produced by the method of claim 10.

14. A DICOM SR to Object-Oriented-representation converter, comprising:
an SR document (110, 210) to UML document (130, 230) converter that is
25 configured to:

map each DICOM Information Entity (311) in the SR document (110, 210) into a corresponding UML class (321) in the UML document (130, 230),

map each DICOM IOD Module (312) in the SR document (110, 210) into a corresponding UML class (322) in the UML document (130, 230),

30 map each DICOM Macro (313) in the SR document (110, 210) into a corresponding UML class (323) in the UML document (130, 230), and

map each DICOM Attribute (314) in the SR document (110, 210) into a corresponding UML attribute (324) in the UML document (130, 230).

15. The converter of claim 14, further comprising:

a UML document (130, 230, 701) to XML document (170, 270, 702)

converter that is configured to:

map each UML class in the UML document (130, 230) into a corresponding

5 XML DTD element.

map each UML class attribute in the UML document (130, 230) into a

corresponding XML DTD element,

map each UML association and uses relationship in the UML document (130,

230) into a corresponding XML DTD element, and

10 map each UML atomic attribute class in the UML document (130, 230) into a
corresponding XML element.

16. The converter of claim 14, wherein

each UML atomic attribute (314) is mapped to a corresponding XML Element

15 (324) with four attributes: coding_scheme, code_id, type, and value.

17. The converter of claim 14, wherein the SR document (110, 210) to UML
document (130, 230) converter is further configured to:

map each DICOM recursive element (500) into a corresponding pair of UML

20 elements (570, 580) that reference each other.

1/8

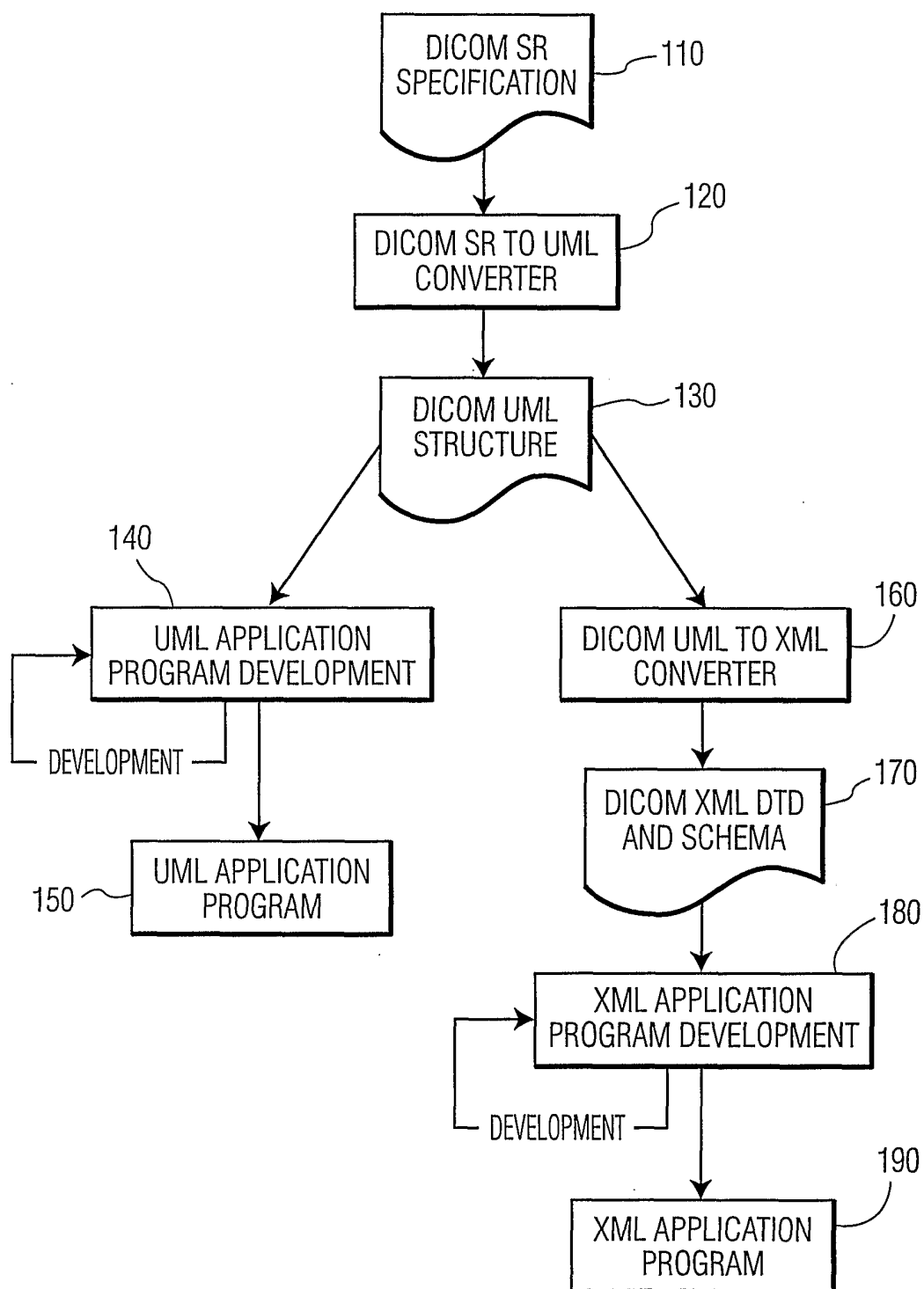


FIG. 1

2/8

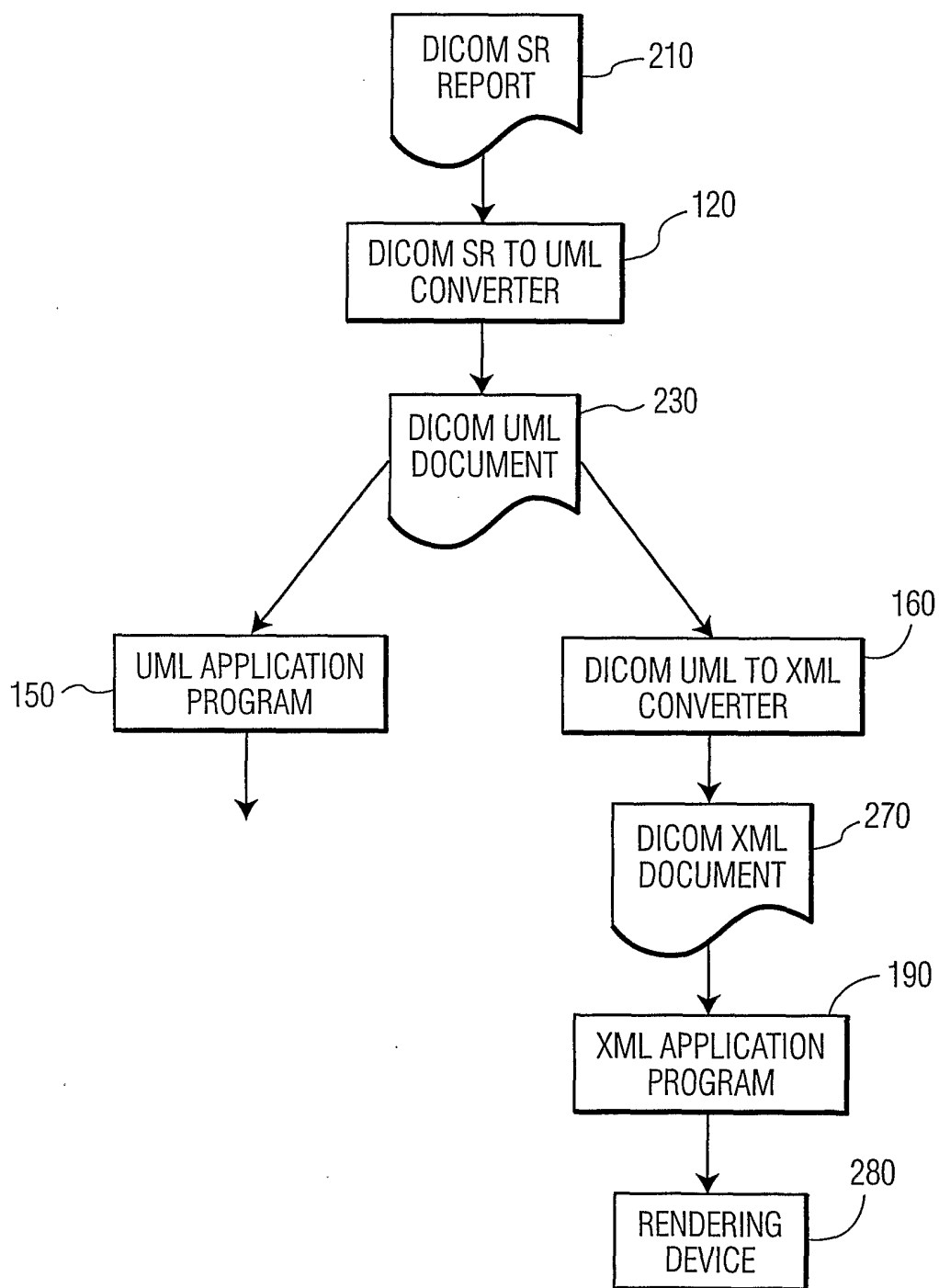


FIG. 2

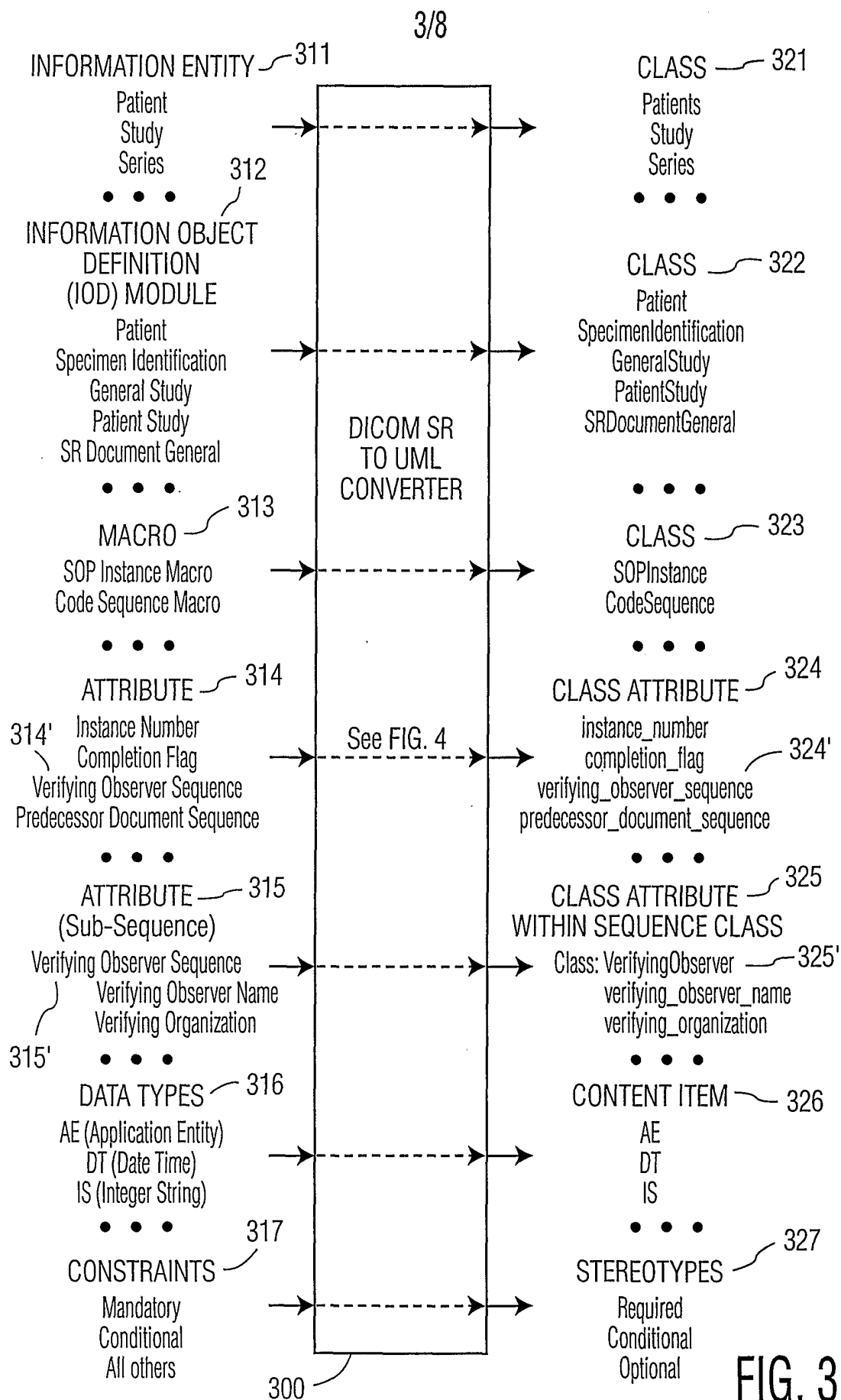


FIG. 3

4/8

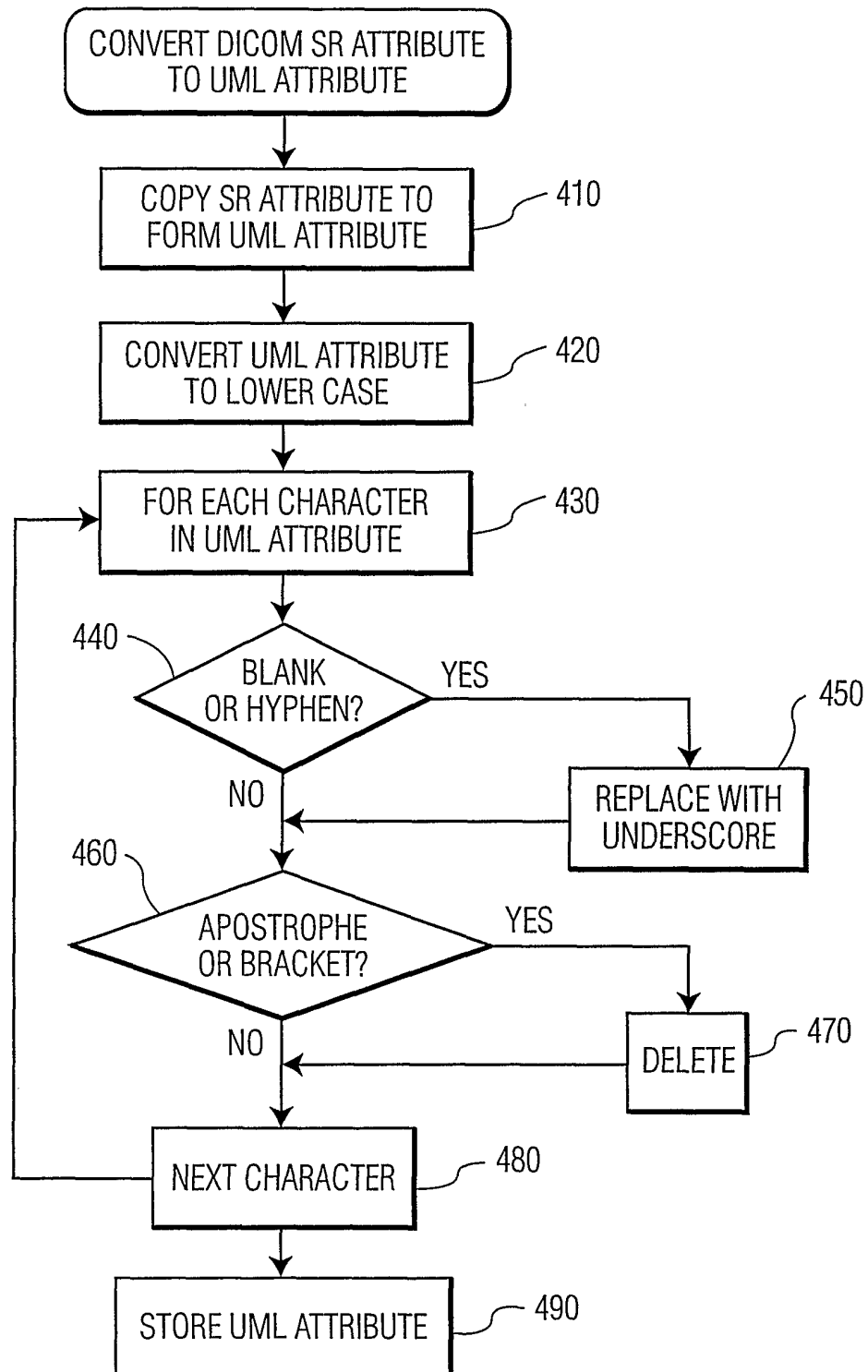
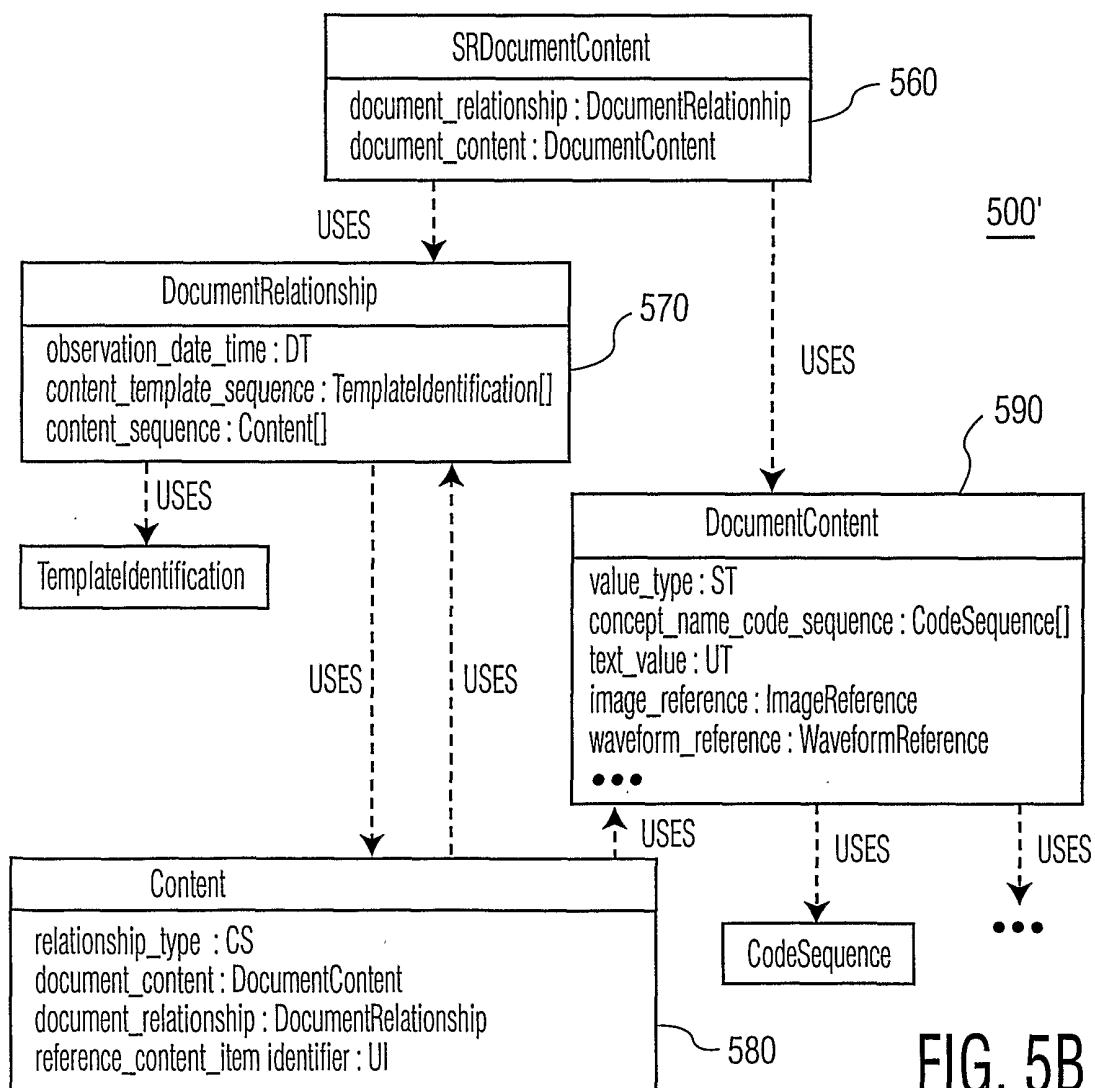
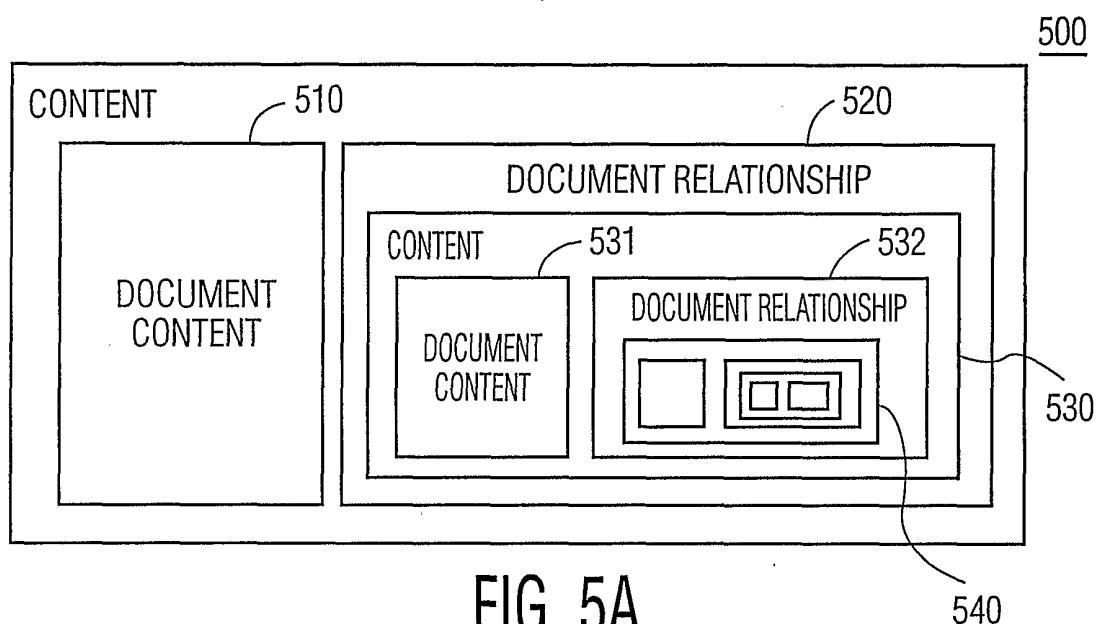


FIG. 4

5/8



6/8

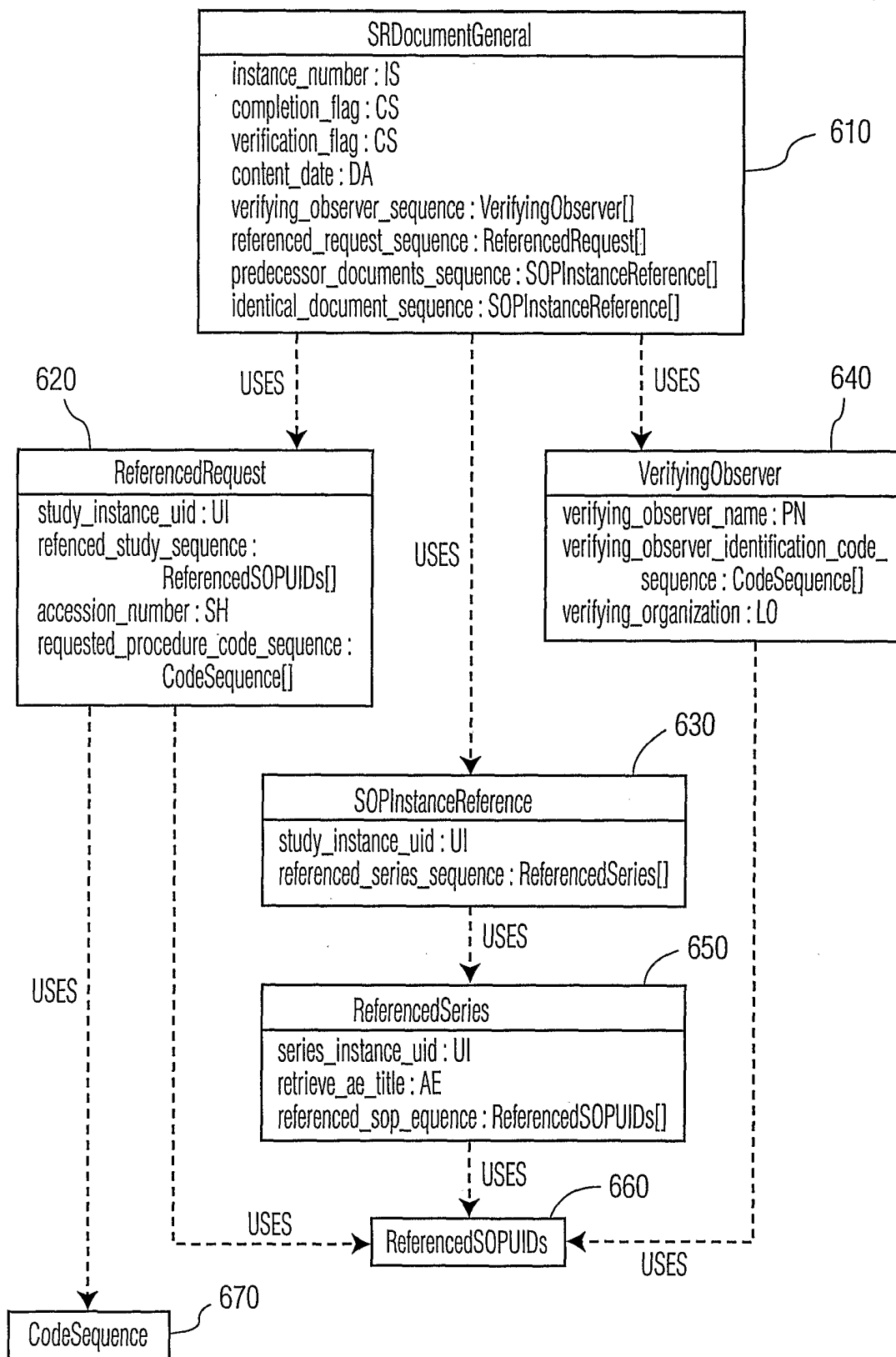


FIG. 6

7/8

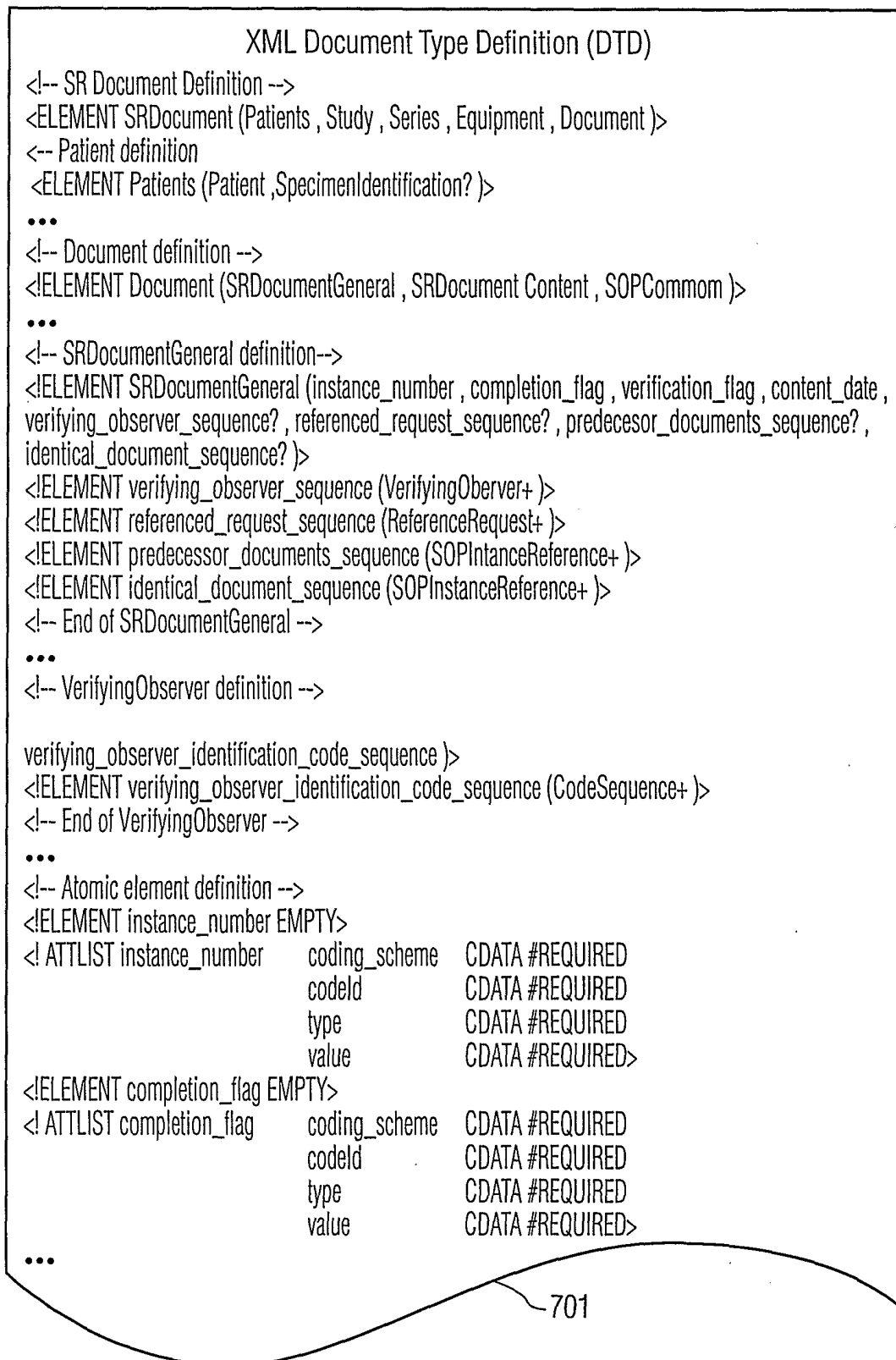


FIG. 7A

8/8

XML Schema

```

<?xml version = "1.0"?>
<schema targetNamespace = "sr-model3_jhu.dtd"
        xmlns = "http://www.w3.org/1999/05/06-xmlschema-1/structures.xsd">
...
<element name = "Document">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "SRDocumentGeneral"/>
            <element ref = "SRDocumentContent"/>
            <element ref = "SOPCommon"/>
        </group>
    </type>
</element>
<element name = "SRDocumentGeneral">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "VerifyingObserver" minOccurs = "1" maxOccurs = "*" />
            <element ref = "ReferenceRequest"/>
            <element ref = "SOPInstanceRefernece" minOccurs = "1" maxOccurs = "*" />
            <element ref = "Item" minOccurs = "1" maxOccurs = "*" />
        </group>
    </type>
</element>
...
<element name = "verifying_observer_sequence">
    <type content = "elementOnly">
        <group order = "seq">
            <element ref = "VerifyingObserver" minOccurs = "1" maxOccurs = "*" />
        </group>
    </type>
</element>
...
<element name = "instance_number">
    <type content = "empty">
        <attribute name = "coding_scheme" minOccurs = "1" type = "string"/>
        <attribute name = "codeID" minOccurs = "1" type = "string"/>
        <attribute name = "type" minOccurs = "1" type = "string"/>
        <attribute name = "name" minOccurs = "1" type = "string"/>
    </type>
</element>
...

```

702

FIG. 7B