DICOM Structured Reporting:
An Object Model as an Implementation Boundary

David Clunie
Development Director, Imaging Products
ComView Corporation
Overview

- What are the features of a DICOM SR?
- What are the applications?
- What can be learned from Internet?
- Representation-independent APIs:
  - Object model
  - Event stream
- Re-use DOM or SAX or SR-specific?
Chest X-ray Report:
Recording Observer: Clunie^David^A^Dr.
History: malignant melanoma excised 1Y
Findings:
- finding: multiple masses in both lung fields
- best illustration of findings:
Conclusions:
- conclusion: cannon-ball metastases
- conclusion: recurrent malignant melanoma
Diagnosis Codes:
- diagnosis: 172.9/ICD9
- diagnosis: 197.0/ICD9
DICOM SR: Object Model

DICOM SR Features

• Tree of “nodes” (content items)
• Each node is a “name-value” pair
• Types of values:
  • Code, text, numeric, dates & times, names, coordinates, references (images, etc.)
• Representation is binary DICOM tags
Sup 23: Structured Reporting

Hierarchical structure, codes, image references
“Recording Observer” = “Clunie^David^^Dr^”
“Study Instance UID ...” = “1.2.3.4.5.6.7.100”
“… Acquisition Subject” = “Homer^Jane^^^”
“Finding” = “Mass”
“diameter” = “1.3” “cm”
“margination” = “infiltrative”
“Baseline” =
“Conclusions”
“Specific Image Findings”
“best illustration of findings” =
“Views” = “PA and Lateral”

“Chest X-Ray Report”

“Context”
“Contains”
“Context”
“Contains”
“Context”
“Contains”
“Modifier”

Context
Contains
Contains
Contains
Contains
Modifier

Properties
Properties

1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8

Infd From

Infd From

Infd From

1.6.1.1
1.6.1.2
Application Requirements

- Create and edit SR reports
- Render SR reports
- Trans-code into other standards:
  - HL7 2.x ORU messages/OBX segments
  - Clinical Document Architecture (CDA)
- Print formatted text
- Archive and query report content
Implementation Requirements

- Creation of structured content
  - GUI
  - Natural language parsing (NLP)
- Encoding/parsing of “serialized” representation (as DICOM object)
- Trans-coding & rendering of same content
Expertise Requirements

- GUI and/or NLP
- DICOM
- Rendering and formatting

Methodology: separate different domain expertise by well-defined API boundaries.
Internet Lessons

• Structured content:
  • XML
  • APIs: DOM & SAX

• Trans-coding:
  • XSL-T

• Rendering:
  • HTML, CSS, JavaScript …
DICOM SR: Object Model

Same Model: Different Representation

DICOM

Internal

XML

(0x0040,0xa491) <COMPLETE>
(0x0040,0xa493) <VERIFIED>
(0x0040,0xa730) Content Sequence
(0x0040,0xa010) <HAS OBS CONTEXT>
(0x0040,0xa040) <PNAME>
(0x0040,0xa043) Concept Name Code Sequence
(0x0008,0x0100) <000555>
(0x0008,0x0102) <LNdemo>
(0x0008,0x0104) <Recording Observer>
(0x0040,0xa123) <Smith^John^^Dr^>
...

<contentsequence>
<contentitem>
<contentlabel>1.1</contentlabel>
<relatioshiptype>HAS OBS CONTEXT</relatioshiptype>
<conceptname>
<codesequence>
<codevalue>000555</codevalue>
<codingcodesignator>LNdemo</codingcodesignator>
<codemeaning>Recording Observer</codemeaning>
</codesequence>
</conceptname>
</contentitem>
</contentsequence>
DICOM SR: Object Model

Same Model: API Boundary

DICOM

Internal

XML

API Boundary

(contentsequence>
  (contentitem>
    (contentlabel>1.1</contentlabel>
    (relationshiptype>HAS OBS CONTEXT</relationshiptype>
    (conceptname>
      (codesequence>
        (codevalue>000555</codevalue>
        (codingschemedesignator>LNdemo</codingschemedesignator>
        (codemeaning>Recording Observer</codemeaning>
      </codesequence>
    </conceptname>
    (valuetype>PNAME</valuetype>
    (personname>Smith^John^^Dr^</personname>
  </contentitem>
)
Report of Chest X-Ray (PA and Lateral Views)

Patient Jane Homer
Study # 123456
 Recorded by Dr. John Smith

The finding is a mass measuring 1.3 cm in diameter with an infiltrative margination.
The finding is a mass measuring 1.3 cm in diameter with an infiltrative margination.
DICOM SR: Object Model

Trans-coding == Tree Re-writing

Specific Application
DICOM SR: Object Model

Trans-coding == Tree Re-writing

Generic Application

Rules in pattern language
Representation-independent APIs

- Object model
  - Internal representation of tree
  - Methods to traverse tree
  - Methods to get/set names/values

- Event stream
  - Start and end node “events”
  - Register call-backs or over-ride methods
Object Model v. Event Stream

CONTAINER: (,, “Procedure”)
contains TEXT: (,, “Description”) = “PA, lateral”
contains DATE: (,, “Date”) = “20010218”
DICOM SR: Object Model

Object Model v. Event Stream

CONTAINER: Procedure
contains

DATE: Date = 20010218

TEXT: Description = PA
Object Model v. Event Stream

CONTAINER: Procedure

contains

DATE: Date = 20010218

TEXT: Description = PA
Node node=Document.getRootNode();
node.getName();
for (node=node.getFirstChild();
    node!=null; node.getNextSibling()) {
    node.getName();
    node.getValue();
}
DICOM SR: Object Model

Object Model v. Event Stream

```
startDocument();
startElement();  // 1:
getAttribute();  // name=procedure
startElement();  // 2:
getAttribute();  // name=description
gGetAttribute();  // value=PA
endElement();    // 2:
startElement();  // 3:
...             // 3:
endElement();    // 3:
endElement();    // 1:
endDocument();
```
Which to choose?

- **Object model:**
  - May be traversed in different order
  - May be edited in place
  - Document size may be constrained

- **Event stream:**
  - Not constrained by storage (memory)
  - Faster if traversal order matches need
  - Application may need to preserve state
DICOM SR: Object Model

Re-using Internet Tools for SR

• W3C Document Object Model (DOM)
• Simple API for XML (SAX) events

• Robust, fast XML parsers
• XSL-T engines
Re-use of XML Tools

- Two choices:
  - literal representation of SR in XML
  - “virtual” XML model or event stream
- Either approach requires actual or virtual XML definition of SR (DTD or Schema)
Parser parser = new XMLParser()
Document document = parser.getDocument(XMLFile);

or

Parser parser = new DicomParser()
Document document = parser.getDocument(DicomFile);
Re-use of XML Tools

• Two choices:
  • literal representation of SR in XML
  • “virtual” XML model or event stream

• Either approach requires actual or virtual XML definition of SR (DTD or Schema)
Choices of XML Equivalent

```xml
<ContentItem>
  <ValueType>NUMERIC</ValueType>
  <RelationshipType>HAS PROPERTIES</RelationshipType>
  <Value>13</Value>
  <Units CSD="UCUM" CV="mm">millimeter</Units>
</ContentItem>

<ContentItem VT="NUMERIC" RT="HAS PROPERTIES" V="13"
  U_CSD="UCUM" U_CV="mm" U_M="millimeter"/>

<item>
  <tag G="0040" E="A010" N="RelationshipType" V="HAS PROPERTIES"/>
  <tag G="0040" E="A040" N="ValueType" V="NUMERIC"/>
</item>
```
Limitations of XML Tools for SR

- XML and SR are similar but different
- XML:
  - Elements lack “name-value pair” concept
  - “value”: plain characters or nested content
    • alleviate by using attributes of elements
  - Few constraints on PCDATA content
    • alleviate by using Schema rather than DTD
  - No relationships (implicit containment)
Value Types

- TEXT
- CODE
- NUM
- PNAME
- DATE
- TIME
- DATETIME
- CONTAINER
- UIDREF
- COMPOSITE
- IMAGE
- WAVEFORM
- SCOOORD
- TCOORD
Nodes linked by Relationships

Parent Node

Relationships

Child Nodes
DICOM SR: Object Model

Relationships

- Contains
- Has Properties
- Inferred From
- Has Observation Context
- Has Acquisition Context
- Has Concept Modifier
- Selected From
SR Specific Object Model (SR-OM)

- Re-use DOM (or SAX) principles
- Define methods specific to SR
- Simplifies SR-aware application
- Limits re-use of such tools as XSL-T
- Solve by providing automatic SR-OM ↔ DOM tools
- Avoids need for virtual XML DTD
SR-OM Classes/Interfaces

- Not just *Node* but:
  - *TextNode*
  - *CodeNode*
  - *ContainerNode*
  - *ImageNode*
  - etc.
SR-OM Node Methods

• Re-use all DOM traversors/iterators:
  • `getNodeName()`, `getNextSibling()`, etc.

• Add accessors for SR-specific content:
  • `getNodeName()`
  • `getNodeType()`
  • `getRelationshipType()`

• More content than `#PCDATA`
  • Coded entries (names, code values, units)
Conclusions

• Separate domain expertise by API boundaries
• Separate model of information from representation
• Re-use XML tools and lessons
Conclusions

• Three approaches
  • Document Object Model (DOM)
  • Simple API for SAX (SAX)
  • SR-specific versions of above

• Choice depends on
  • Need for interaction with other tools
  • Level of abstraction of SR details