XML Representation of Data Types.

Gunther Schadow

Regenstrief Institute for Health Care

Why this separate presentation?

- This is not about the specifics of HL7 version 3 data types.
- It discusses some ideas of a proposed version 3 data type XML mapping.
- The essentials are applicable to other data type systems as well.

Contents

1. HL7 version 3 data types distilled.
2. The problems with XML-based HL7 messages.
3. The problems with DTDs.
4. XML mapping basics, FleXML, and HyperFleXML.
5. HyperFleXML demo on data type for coded concept.
6. So what?

HL7 Version 3 Data Types

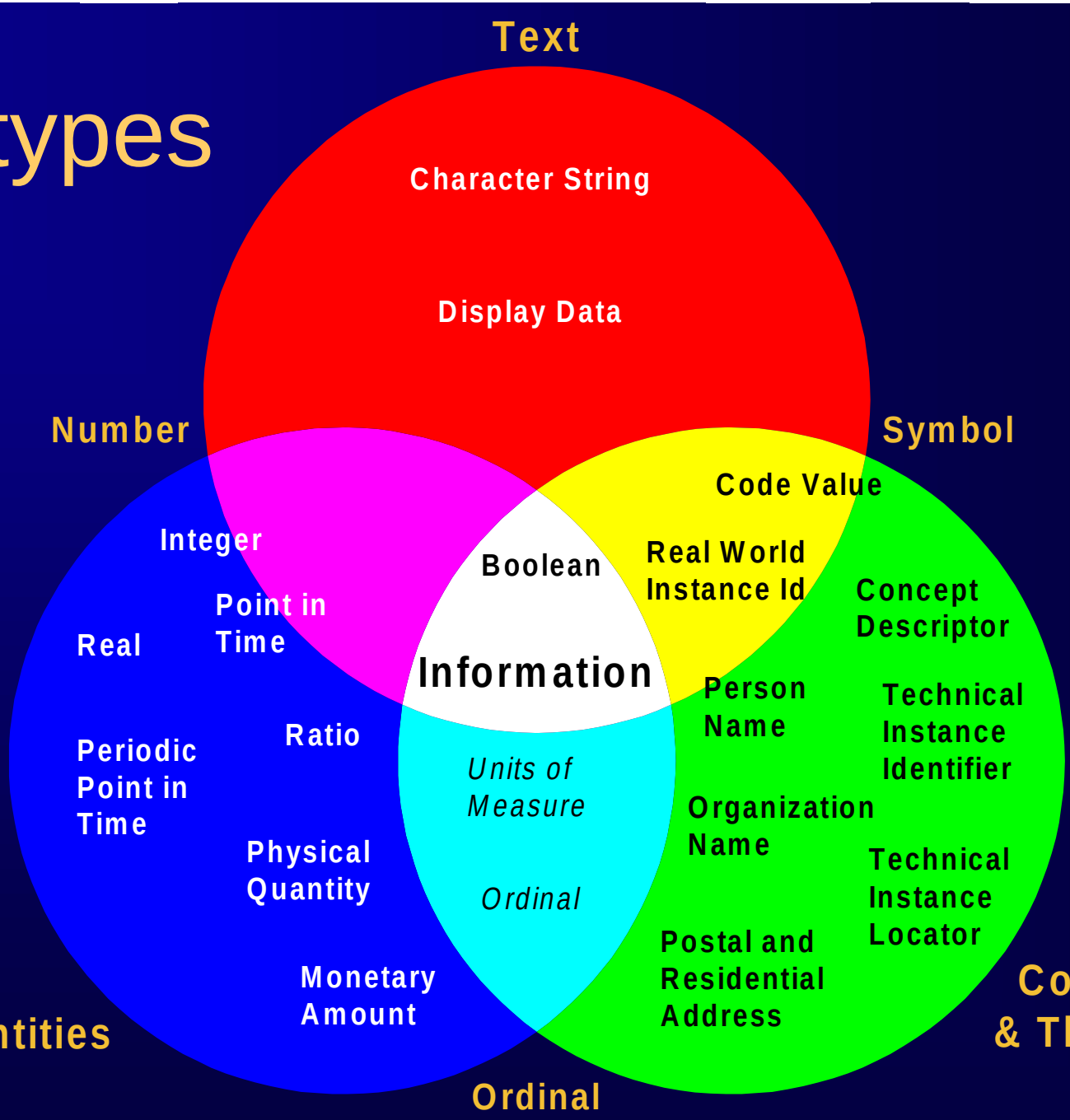
Quick overview in 6 slides

Semantics first … 1

- Data types are the fundamental constituents of all health care information.
- Share meaning across different technologies.

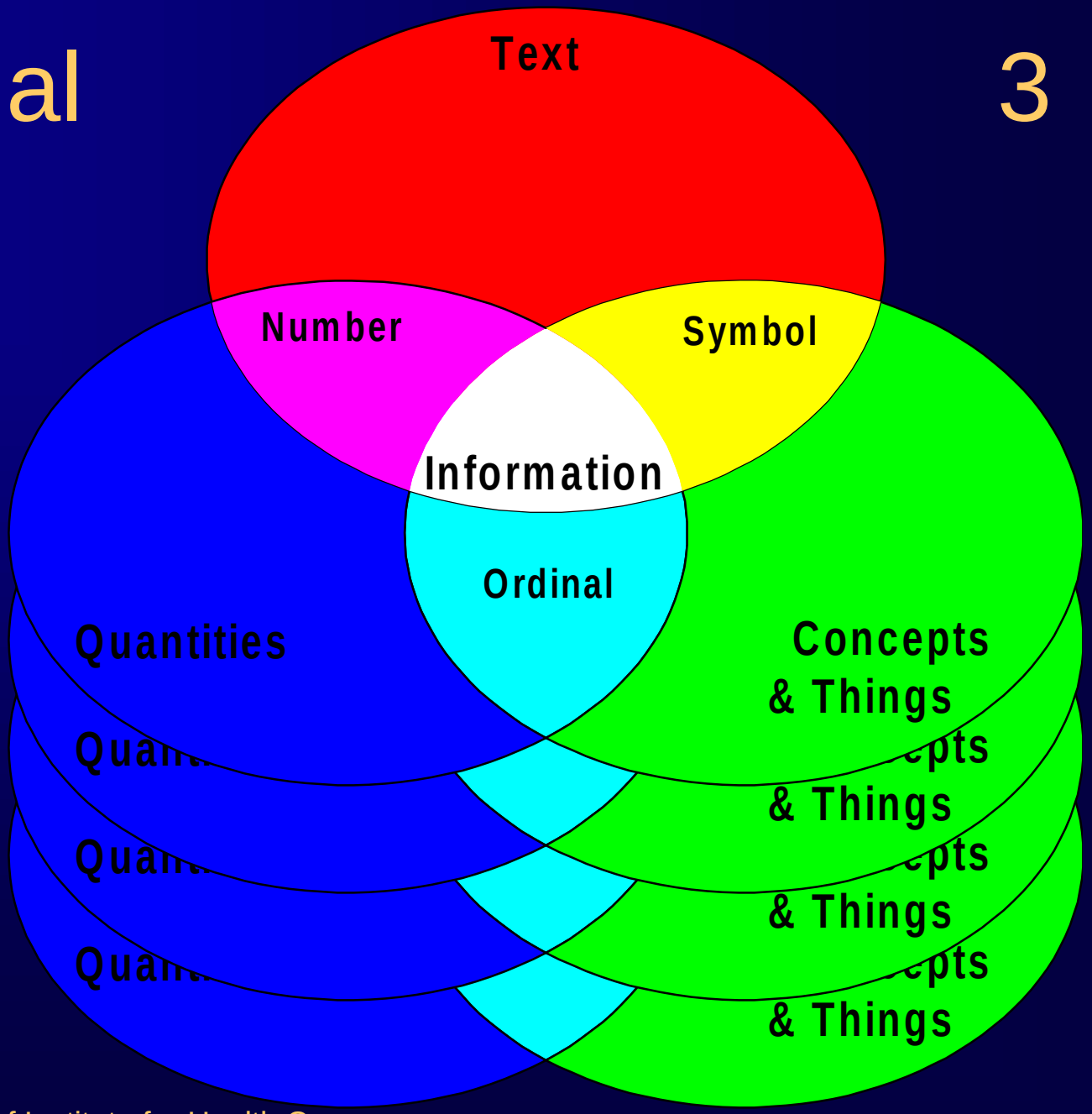
- … representation later
  - Every representation is possible that leaves information content unchanged.
  - Point in time (timestamp) is a concept with operations not just a character string format.
  - Interval (range) has two bounds and a width that may be represented by only two major components.
  - Real numbers have precision that can hide in the representation.

Basic types



Orthogonal issues

- Collections
  - set, list, bag
  - interval
- Incomplete information
  - null values
- Uncertainty
  - probability distributions
- History
  - linear, cyclic



Summary

	18 basic types		18 basic types
+	6 internal types		
+	10 generic types	×	10 generic types
=	34 defined types	=	∞ useable types

- No additional types expected in the future.
- Emphasis on semantics suggests implementation as intelligent data structures (objects.)
- XML can’t provide the semantics.

Data type vs. common class 5

- A data type is just another class.
  - In C++/Java you declare variables of class-types
  - in UML you can use classes as data types
- Fundamental building blocks are needed.
  - Information modeling is aggregation of smaller types into higher types, you need to start somewhere.
- Basic data types are everywhere
  - ISO 11404, Schema-2, SQL, C, Pascal, BASIC, …
  - They are all slightly different.
  - Greatest common denominator is, the bit?
  - … or the character string?

Why “our own” data types? 6

- Need for a standard
  - must be technology independent (not just XML.)
  - fundamental building blocks of meaning.
- Need for powerful types
  - a message is more than a bunch of strings
  - every type represents a fairly well understood semantic field (e.g. numbers, quantity, time.)
  - data types are more than a bunch of components
  - operations are much more important (and easy.)
  - orthogonal issues: e.g., uncertainty, precision

➨ Current standards are not sufficiently powerful.

XML Representation

- Regular DTD-able XML
- FleXML
- HyperFleXML

Why are data types different?

¿ If “everything is a type” why is XML mapping of data types a special issue ?

- Only data types can be “primitive.”
  - e.g., string, integer, ...
- Data types are small compounds of tightly related components.
  - An average information model class has 15 attributes.
  - Data types often have only 2 or 3 components.
- Data type specification reuses itself.
  - Types can comprise 3 or more layers of nesting.

So what's the problem?

- Bulky messages.
  - Message size ***does*** matter (a bit.)
  - XML markup contributes a lot to that size.
  - XML users have experienced penalties of network throughput while using XML.
- Information gets lost in the markup.
  - Human readable XML instances?
  - Self-documenting messages?
- A regular markup is often not intuitive.
  - Humans prefer lexical contractions.

Example data in HL7 v2.3

```
MSH|^~\&|LABGL1||DMCRES||199812300100||ORU^R01|LABGL
    1199510221838581|P|2.3|||NE|NE
PID|||6910828^Y^C8||Newman^Alfred^E||19720812|M||W|2
    5 Centscheap Ave^^Whatmeworry^UT^85201^^P|
    |(555)777-6666|(444)677-7777||M||773789090
OBR||110801^LABGL|387209373^DMCRES|18768-2^CELL
    COUNTS+DIFFERENTIAL TESTS
    (COMPOSITE)^LN|||199812292128||35^ML|||||||IN297
    3^Schadow^Gunther^^^^MD^UPIN|||||||||^Once|||||
    CA 20837^Spinosa^John^^^^MD^UPIN
OBX||NM|4544-3^HEMATOCRIT (AUTOMATED)^LN||45||39-
    49||||F|||199812292128||CA20837
OBX||NM|789-8^ERYTHROCYTES COUNT
    (AUTOMATED)^LN||4.94|10*12/mm3|4.30-
    5.90||||F|||199812292128||CA20837
```

~~Same~~ data in XML Less!

```
<Labrs3P00 T="Labrs3P00">
    <Labrs3P00.PTP T="PTP">
        <PTP.primrPrsnm T="PN">
            <fmn T="ST">Newman</fmn>
            <gvn T="ST">Alfred</gvn>
            <mdn T="ST">E</mdn>
        </PTP.primrPrsnm>
    </Labrs3P00.PTP>
    <Labrs3P00.SIOO_L T="SIOO_L">
        <SIOO_L.item T="SIOO">
            <SIOO.filrOrdId T="IID">LABGL110801</SIOO.filrOrdId>
            <SIOO.placrOrdId T="IID">DMCRES387209373</SIOO.placrOrdId>
            <SIOO.InsncOf T="MSRV">
                <MSRV.unvSvcId T="CE">18768-2</MSRV.unvSvcId>
                <MSRV.svcDesc T="TX">CELL COUNTS+DIFFERENTIAL TESTS (COMPOSITE)</MSRV.svcDesc>
            </SIOO.InsncOf>
            <SIOO.SRVE_L T="SRVE_L">
                <SRVE_L.item T="SRVE">
                    <SRVE.name T="CE">4544-3</SRVE.name>
                    <SRVE.svcEvntDesc T="ST">HEMATOCRIT (AUTOMATED)</SRVE.svcEvntDesc>
                    <SRVE.CLOB T="CLOB">
                        <CLOB.obsvnValu T="NM">45</CLOB.obsvnValu>
                        <CLOB.refsRng T="ST">39-49</CLOB.refsRng>
                        <CLOB.clnRlvnBgnDtm T="DTM">199812292128</CLOB.clnRlvnBgnDtm>
                    </SRVE.CLOB>
                    <SRVE.spcmRcvdDtm T="DTM">199812292315</SRVE.spcmRcvdDtm>
                </SRVE_L.item>
                <SRVE_L.item T="SRVE">
                    <SRVE.name T="CE">789-8</SRVE.name>
                    <SRVE.svcEvntDesc T="ST">ERYTHROCYTE COUNT (AUTOMATED)</SRVE.svcEvntDesc>
                    <SRVE.CLOB T="CLOB">
                        <CLOB.obsvnValu T="NM">4.94</CLOB.obsvnValu>
                        <CLOB.refsRng T="ST">4.30-5.90</CLOB.refsRng>
                        <CLOB.clnRlvnBgnDtm T="DTM">199812292128</CLOB.clnRlvnBgnDtm>
                    </SRVE.CLOB>
                    <SRVE.spcmRcvdDtm T="DTM">199812292315</SRVE.spcmRcvdDtm>
                </SRVE_L.item>
        </SIOO_L.item>
    </Labrs3P00.SIOO_L>
</Labrs3P00>
```

The numbers

Traditional encoding

- 607 total bytes
- 152 markup bytes
- 455 data bytes
- 75% data fraction

XML encoding

- 1224 total bytes
- 1023 markup bytes
- 201 data bytes
- 16% data fraction

➨ Straightforward XML encoding is 5 times larger.

- This is probably an optimistic estimate.
- Example does not even have extensively nested data types.

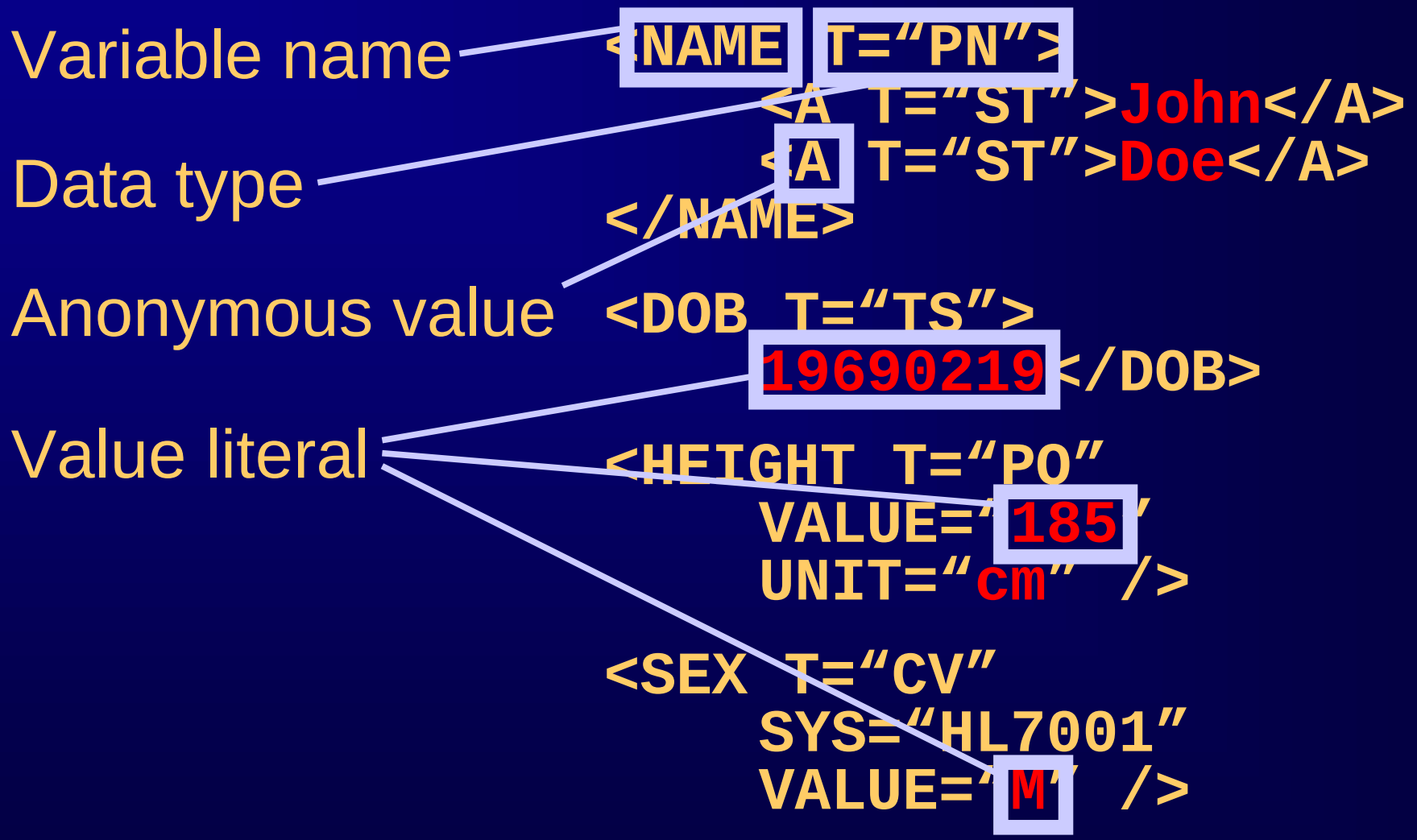
What is wrong with the DTD?

- XML “attributes” and “elements”
  - both stick labels to data, but in a different way,
  - typesetting legacy but inapplicable for data.
- DTD enforces unreasonable decision between XML attribute and element
  - ignites useless but vigorous discussions.
- DTD ignores *names* vs. data *types*
  - a distinction that is on the basis of computing,
  - name/type distinction must be made by style conventions defined outside XML/DTD.

DTD is not powerful

- DTD enforces element sequence.
  - which we hoped to overcome through tagging!
- DTD does not have scopes.
  - A person's "name" can not coexists with an observation "name" having different content models.
  - W3C namespaces were done wrong
- XML syntax is totally free of semantics.
  - Not even the simplest macro expansion is possible
  - Functions added by special programs from outside
- DTD can not express semantic constraints.

XML mapping basics



All “names”

- XML tags are only used for “names”
  - MIM attribute name
  - Component name
  - Variable name

```
<NAME T=“PN”>
      <A T=“ST”>John</A>
      <A T=“ST”>Doe</A>
</NAME>

<DOB T=“TS”>
      19690219</DOB>

<HEIGHT T=“PQ”
      VALUE=“185”
      UNIT=“cm” />

<SEX T=“CV”
      SYS=“HL7001”
      VALUE=“M” />
```

All data types

- Type attribute should be given
- if given type is different from expected type.
- Type attribute can be omitted to increase data fraction.

```
<NAME T="PN">
      <A T="ST">John</A>
      <A T="ST">Doe</A>
</NAME>

<DOB T="TS">
      19690219</DOB>

<HEIGHT T="PQ"
      VALUE="185"
      UNIT="cm" />

<SEX T="CV"
      SYS="HL7001"
      VALUE="M" />
```

All anonymous values

- Elements of collections
  - set
  - list
  - bag
- Use the “A” tag convention

```
<NAME T=“PN”>
     <A T=“ST”>John</A>
     <A T=“ST”>Doe</A>
</NAME>

<DOB T=“TS”>
     19690219</DOB>

<HEIGHT T=“PQ”
     VALUE=“185”
     UNIT=“cm” />

<SEX T=“CV”
     SYS=“HL7001”
     VALUE=“M” />
```

All value literals

- Primitive types in XML are literals.
- Literals are typed values not just strings.
- Even composite types can have literals.

```
<NAME T="PN">
     <A T="ST">John</A>
     <A T="ST">Doe</A>
</NAME>

<DOB T="TS">
     19690219</DOB>

<HEIGHT T="PQ"
     VALUE="185"
     UNIT="cm" />

<SEX T="CV"
     SYS="HL7001"
     VALUE="M" />
```

FleXML

- Overcomes the distinction between XML elements and attributes.
- A simple mapping of attributes to elements does the job.
  - As a SAX or DOM front-end.
  - DSSSL or XSL transformation.
- Can squeeze all components with primitive values to XML attributes.
  - Saves redundancy of the closing tag.

All element vs. Most attribute

```
<PATIENT>
  <DOB T="TS">
    19690219
  </DOB>
  <SEX T="CV">
    <SYS T="ST">
      HL7001
    </SYS>
    <VALUE>M
          </VALUE>
  </SEX>
</PATIENT>
```

```
<PATIENT
  DOB="19690219"/>

  <SEX T="CV"
    SYS="HL7001"

    VALUE="M"/>


</PATIENT>
```

Be verbose vs. Use default

```
<PATIENT
  DOB="19690219"/>

  <SEX T="CV"
    SYS="HL7001"

    VALUE="M"/>

</PATIENT>
```

```
<PATIENT
  DOB="19690219"/>

  <SEX T="CV"

    VALUE="M"/>

</PATIENT>
```

Use content to HyperFleXML

```
<PATIENT
  DOB="19690219"/>

  <SEX>M</SEX>




</PATIENT>
```

```
<PATIENT
  DOB="19690219"

  SEX="M"




/>
```

Regular vs. HyperFleXML

```
<PATIENT>
  <DOB T="TS">
    19690219
  </DOB>
  <SEX T="CV">
    <SYS T="ST">
      HL7001
    </SYS>
    <VALUE>M
          </VALUE>
  </SEX>
</PATIENT>
```

```
<PATIENT
  DOB="19690219"
  SEX="M"/>
```

HyperFleXML

- Revokes difference between XML attributes and elements (FleXML.)
- Makes extensive use of defaults.
  - Only provide data that is unusual or unexpected.
  - Expected information is redundancy!
- One component of a composite may be turned into *content* of the XML element.
- An element with only content can be turned into an attribute of the parent element.

...

Yet more flex ...

- Different components can be the content.
  - E.g. code value ...

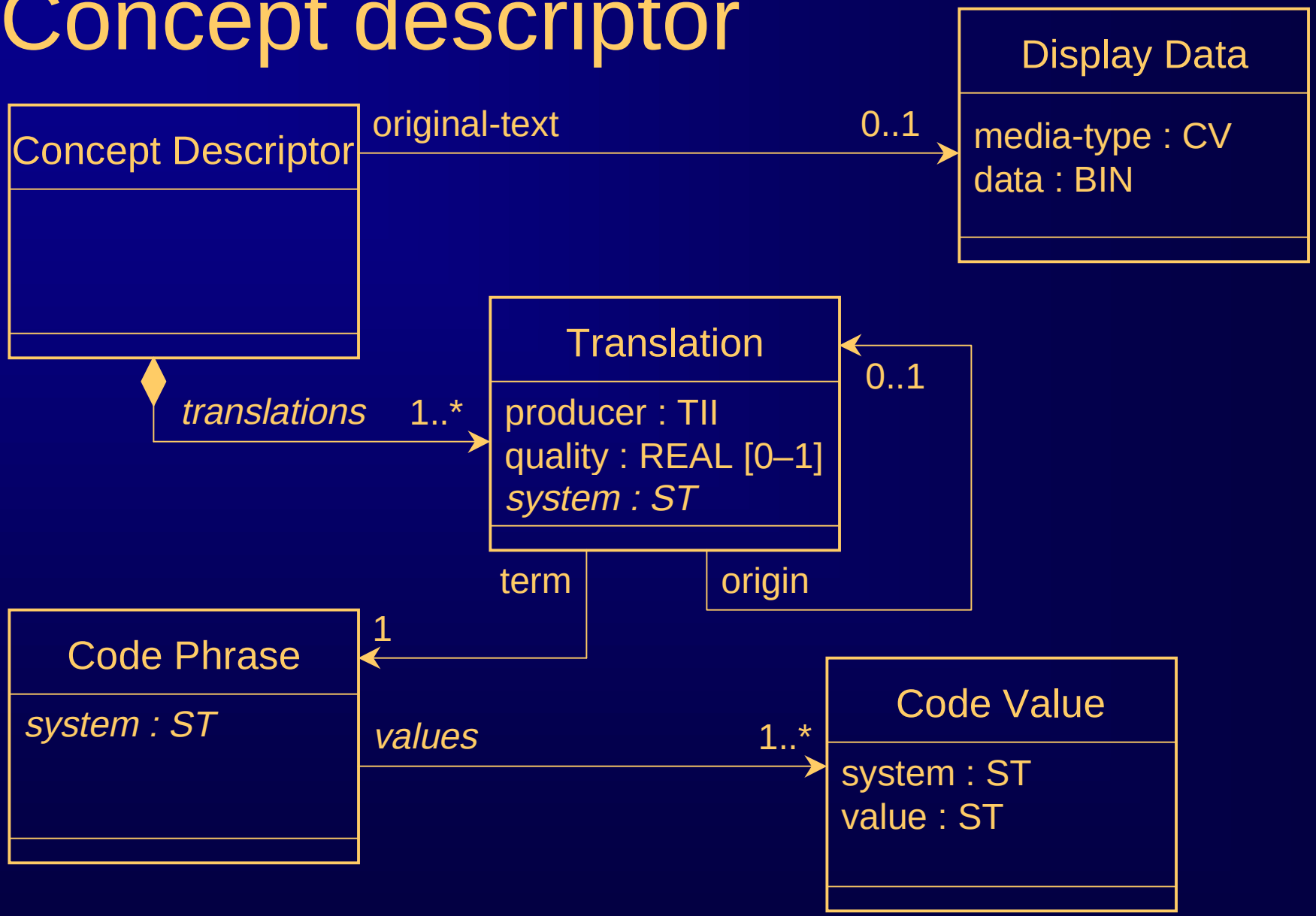
    <SEX PRINTNAME="MALE">M</SEX>

    <SEX>M</SEX>

    SEX="M"
  - … or the print name

    <SEX VALUE="M">MALE</SEX>
- Print name as content feels as if the code were added to a free text document
  - Converging to document-oriented philosophy.

Concept descriptor



Example for concept descriptor

Original text:

"… the patient's hair had an ashy-blondish color …"

Code translations:

"*AB*" for "*ash blond*" in local coding system "*99hcc*"

"*10.2*" for "*pale blond*" in standard "*ICHC*" system

"*B001, G002, H001*" for "*blond, slight gray, homogenous*" in multi-axial coding system "*PILS-AVACC*"

All elements

```
<HAIR-COLOR T="CD">
  <TEXT T="DD">
    <MEDIA T="CV">
      <VAL T="ST">text/plain</VAL>
      <SYS T="ST">HL7-9999</SYS></MEDIA>
    <ENC T="CV">
      <VAL T="ST">TEXT</VAL>
      <SYS T="ST">HL7-9998</SYS></ENC>
    <DATA>the patient's hair had an ashy-blondish color</DATA></TEXT>
  <TRANSLATIONS T="SET.CDXL">
    <A T="CDXL" ID="0">
      <VAL T="LIST.CV">
        <A T="CV">
          <VAL>AB</VAL><SYS>99hcc</SYS><PNM>ash blond</PNM></A></VAL></A>
    <A T="CDXL" ID="1" ORG="0">
      <VAL T="LIST.CV">
        <A T="CV">
          <VAL>10.2</VAL><SYS>ICHC</SYS><PNM>pale blond</PNM></A></VAL></A>
    <A T="CDXL" ID="2" ORG="1">
      <VAL T="LIST.CV">
        <A T="CV">
          <VAL>B001</VAL><SYS>PILS-AVACC</SYS><PNM>blond</PNM></A>
        <A T="CV">
          <VAL>G001</VAL><SYS>PILS-AVACC</SYS><PNM>slight gray</PNM>
        <A T="CV">
          <VAL>H001</VAL><SYS>PILS-AVACC</SYS><PNM>homogeneous</PNM></A></VAL></A>
  </TRANSLATIONS>
</HAIR-COLOR>
```

FleXML

```
<HAIR-COLOR T="CD">
  <TEXT T="DD">
    <MEDIA T="CV" VAL="text/plain" SYS="HL7-9999"/>
    <ENC T="CV" VAL="TEXT" SYS="HL7-9999"/>
    <DATA>the patient's hair had an ashy-blondish color</DATA></TEXT>
  <TRANSLATIONS T="SET.CDXL">
    <A T="CDXL" ID="0">
      <VAL T="LIST.CV">
        <A T="CV" VAL="AB" SYS="99hcc" PNM="ash blond"/></VAL></A>
    <A T="CDXL" ID="1" ORG="0">
      <VAL T="LIST.CV">
        <A T="CV" VAL="10.2" SYS="ICHC" PNM="pale blond"/></VAL></A>
    <A T="CDXL" ID="2" ORG="1">
      <VAL T="LIST.CV">
        <A T="CV" VAL="B001" SYS="PILS-AVACC" PNM="blond"/>
        <A T="CV" VAL="G002" SYS="PILS-AVACC" PNM="slight gray"/>
        <A T="CV" VAL="H001" SYS="PILS-AVACC" PNM="homogenous"/>
       </VAL></A>
  </TRANSLATIONS>
</HAIR-COLOR>
```

Use defaults and implicit typing

```
<HAIR-COLOR>
  <TEXT>
    <MEDIA VAL="text/plain"/>
    <ENC VAL="TEXT"/>
    <DATA>the patient's hair had an ashy-blondish color
    </DATA></TEXT>
  <TRANSLATIONS>
    <A ID="0">
      <VAL>
        <A VAL="AB" SYS="99hcc" PNM="ash blond"/></VAL></A>
    <A ID="1" ORG="0">
      <VAL>
        <A VAL="10.2" SYS="ICHC" PNM="pale blond"/></VAL></A>
    <A ID="2" ORG="1">
      <VAL SYS="PILS-AVACC">
        <A VAL="B001" PNM="blond"/>
        <A VAL="G002" PNM="slight gray"/>
        <A VAL="H001" PNM="homogenous"/>
       </VAL></A>
  </TRANSLATIONS>
</HAIR-COLOR>
```

Put data into element content

```
<HAIR-COLOR>
  <TEXT>
    <MEDIA>text/plain</MEDIA>
    <ENC>TEXT</ENC>
    the patient's hair had an ashy-blondish color</TEXT>
  <TRANSLATIONS>
    <A ID="0"><VAL><A SYS="99hcc">AB</A></VAL></A>
    <A ID="1" ORG="0"><VAL><A SYS="ICHC">10.2</A></VAL></A>
    <A ID="2" ORG="1"><VAL SYS="PILS-AVACC">
              <A>B001</A><A>G002</A><A>H001</A></VAL></A>
  </TRANSLATIONS>
</HAIR-COLOR>
```

Turn into parent’s attributes

```
<HAIR-COLOR>
  <TEXT MEDIA=“text/plain” ENC=“TEXT”>
    the patient's hair had an ashy-blondish color
  </TEXT>
  <TRANSLATIONS>
    <A ID=“0”><VALUES><A SYS=“99hcc”>AB</A></VAL></A>
    <A ID=“1” ORG=“0”><VAL><A SYS=“ICHC”>10.2</A>
                                        </VAL></A>
    <A ID=“2” ORG=“1”><VAL SYS=“PILS-AVACC”>
       <A>B001</A><A>G002</A><A>H001</A></VAL></A>
  </TRANSLATIONS>
</HAIR-COLOR>
```

Collections elements into content

```
<HAIR-COLOR>
  <TEXT MEDIA="text/plain" ENC="TEXT"
    the patient's hair had an
    ashy-blondish color
  </TEXT>
  <A ID="0"           SYS="99hcc"><A>AB</A></A>
  <A ID="1" ORG="0" SYS="ICHC"><A>10.2</A></A>
  <A ID="2" ORG="1" SYS="PILS-AVACC">
      <A>B001</A><A>G002</A><A>H001</A></A>
</HAIR-COLOR>
```

Elements to parent NMTOKENS

```
<HAIR-COLOR>
  <TEXT MEDIA="text/plain" ENC="TEXT"
    the patient's hair had an
    ashy-blondish color
  </TEXT>
  <A ID="0"           SYS="99hcc" VAL="AB"/>
  <A ID="1" ORG="0"   SYS="ICHC"  VAL="10.2"/>
  <A ID="2" ORG="1"   SYS="PILS-AVACC"
                      VAL="B001 G002 H001"/>
</HAIR-COLOR>
```

HyperFleXML at its best

```
<HAIR-COLOR>
 the patient's hair had an
 ashy-blondish color
 <A ID="0" SYS="99hcc" VAL="AB"/>
 <A ID="1" ORG="0" SYS="ICHC"
                   VAL="10.2"/>
 <A ID="2" ORG="1" SYS="PILS-AVACC"
                   VAL="B001 G002 H001"/>
</HAIR-COLOR>
```

The numbers

Regular DTD-able XML

- 768 total bytes
- 629 markup bytes
- 139 data bytes
- 9%–18% data fraction

HyperFleXML

- 181 total bytes
- 105 markup bytes
- 76 data bytes
- 41% data fraction

➨ Straightforward XML encoding is 4.5 times larger than HyperFleXML encoding

- Data fraction with nested types may drop from 16% to 9%!
- HyperFleXML can recover data fraction to be close to traditional HL7 encoding rules (41% vs. 75%.)

Summary of HyperFleXML

- Huge reduction in message size.
- Small is beautiful!
- Looks like marked-up document (PRA)
- HyperFleXML transformations might be the link between PRA and HL7 messages.

```
<HAIR-COLOR T="CD">
  <TEXT T="DD">
    <MEDIA T="CV">
      <VAL T="ST">text/plain</VAL>
      <SYS T="ST">HL7-9999</SYS></MEDIA>
    <ENC T="CV">
      <VAL T="ST">TEXT</VAL>
      <SYS T="ST">HL7-9998</SYS></ENC>
    <DATA>the patient's hair had an ashy-blondish color</DATA></TEXT>
  <TRANSLATIONS T="SET.CDXL">
    <A T="CDXL" ID="0">
      <VAL T="LIST.CV">
        <A T="CV">
          <VAL>AB</VAL><SYS>99hcc</SYS><PNM>ash blond</PNM></A></VAL></A>
      <A T="CDXL" ID="1" ORG="0">
        <VAL T="LIST.CV">
        <A T="CV">
          <VAL>10.2</VAL><SYS>ICHC</SYS><PNM>pale blond</PNM></A></VAL></A>
      <A T="CDXL" ID="2" ORG="1">
      <VAL T="LIST.CV">
        <A T="CV">
          <VAL>B001</VAL><SYS>PILS-AVACC</SYS><PNM>blond</PNM></A>
              <A T="CV">
          <VAL>G001</VAL><SYS>PILS-AVACC</SYS><PNM>slight gray</PNM>
        <A T="CV">
          <VAL>H001</VAL><SYS>PILS-AVACC</SYS><PNM>homogeneous</PNM></A>
        </VAL></A>
  </TRANSLATIONS>
</HAIR-COLOR>
```

```
<HAIR-COLOR>
  the patient's hair had an ashy-blondish color
  <A ID="0"          SYS="99hcc"       VAL="AB"/>
  <A ID="1" ORG="0" SYS="ICHC"        VAL="10.2"/>
  <A ID="2" ORG="1" SYS="PILS-AVACC" VAL="B001 G002 H001"/>
</HAIR-COLOR>
```

So what?

- How do W3C schemata pertain?
  - Can schema accommodate a similar results?
  - Will schema data types interfere with V3DT?
- Is HyperFleXML implementable?
  - FleXML is a no-brainer but HyperFleXML has many more transformations.
  - Will XSL-T do the transformations?
  - Will HyperFleXML introduce ambiguities?
- Will HL7 choose HyperFleXML?
  - Many are uncomfortable about HyperFleXML (yet).
  - Potential of converging PRA and HL7 messages.

Thank you.

For more information please visit
http://aurora.rg.iupui.edu/v3dt