DCMs and Data Types

Thomas Beale, for San Diego, Sep 2011

Assumptions

- DCMs are based on an underlying model (ULM), rather than each being an independent model (e.g. Classes, RBD tables) for domain definitions
- DCMs are not themselves part of the software (some generated artefact might be)
  - This is the raison d'être for DCMs – to get out of the mess of endlessly growing and unmaintainable software and databases

Based on an underlying ... means

- The underlying model provides the ‘primitives’ needed for DCM modelling
- DCMs don’t have to redefine these primitives
- Therefore, such primitives are commonly required patterns for doing DCMs
- Insufficient patterns in the underlying model
  - → DCM authors continually re-invent basics
  - → multiple authors / orgs will re-invent them in different, non-interoperable ways

Based on an underlying ... means

- What relationship of DCMs to the ULM?
- We assume that the ULM provides a shared definition of data and (some) semantics, i.e.
    - Basis of at least data interoperability
    - And potentially software interoperablity
- Therefore… DCMs cannot ‘break’ the ULM

Based on an underlying ... means

- Possible mathematical relationships that allow this have a notion of formal conformance
- Including:
  - Constraint
  - Extension
- Where in all cases the DCM entity cannot invalidate a data instance of the ULM entity

In other words...

- The golden rule is that:
  - Every instance of a DCM element is also a valid instance of the corresponding ULM element
- Breaking this rule
  - → non-interoperable DCM instances
  - → No assumptions can be made by software

Based on an underlying ... means

- However...
  - The definitions provided for DCM purposes do not need to be full implementable definitions!
  - Instead, they only need to consist of those data elements that need to be specifically constrained in DCMs
  - And that guarantee data interoperability
- This should reduce the complexity of the ULM DTs
- We can think of these as model patterns

DT Concrete requirements

- The underlying model is often considered to consist of:
  - Data types (DTs)
  - Reference Model (RM) – higher structures
- In fact it would make more sense to just talk about 'reference model', but ... too late!
- The DTs and RM should consist of patterns that allow good DCM modelling

openEHR

About data types

- Data Types are the most basic patterns required
- Usually understood to mean ‘clinical data types’, i.e. We already assume basic computing types (ISO 11404 or similar):
  - Integer, Real, String, Boolean, Character
  - Octet / Byte
  - Decimal / BCD
  - List<T>, Array<T>, Set<T>, Hash<T,K>
  - With UTF support assumed in Strings / chars

About data types

- We will also assume
  - Date/Time definitions from ISO 8601:2004, represented as Strings, whose syntax allows expression of:
    - Date, Time, DateTime, Duration
  - URI, as a syntax-based String type
  - Oid ,
  - UUID/GUID

Clinical data types

- The need for clinical DTs is well-known in health informatics, types such as:
  - Identifiers
  - Text, coded text
  - Various quantity types
  - Ordinals
  - Dates, times, durations
  - Time specification types
  - Multimedia / encapsulated data
  - Esoteric types

Starting points for DCM

- Existing published models?
  - ISO 21090
  - HL7v3
  - openEHR
  - Grahame Grieve's RFH
- A proprietary model, brought into the open?
  - Intermountain Health
- A de novo model we build for DCM

Published models

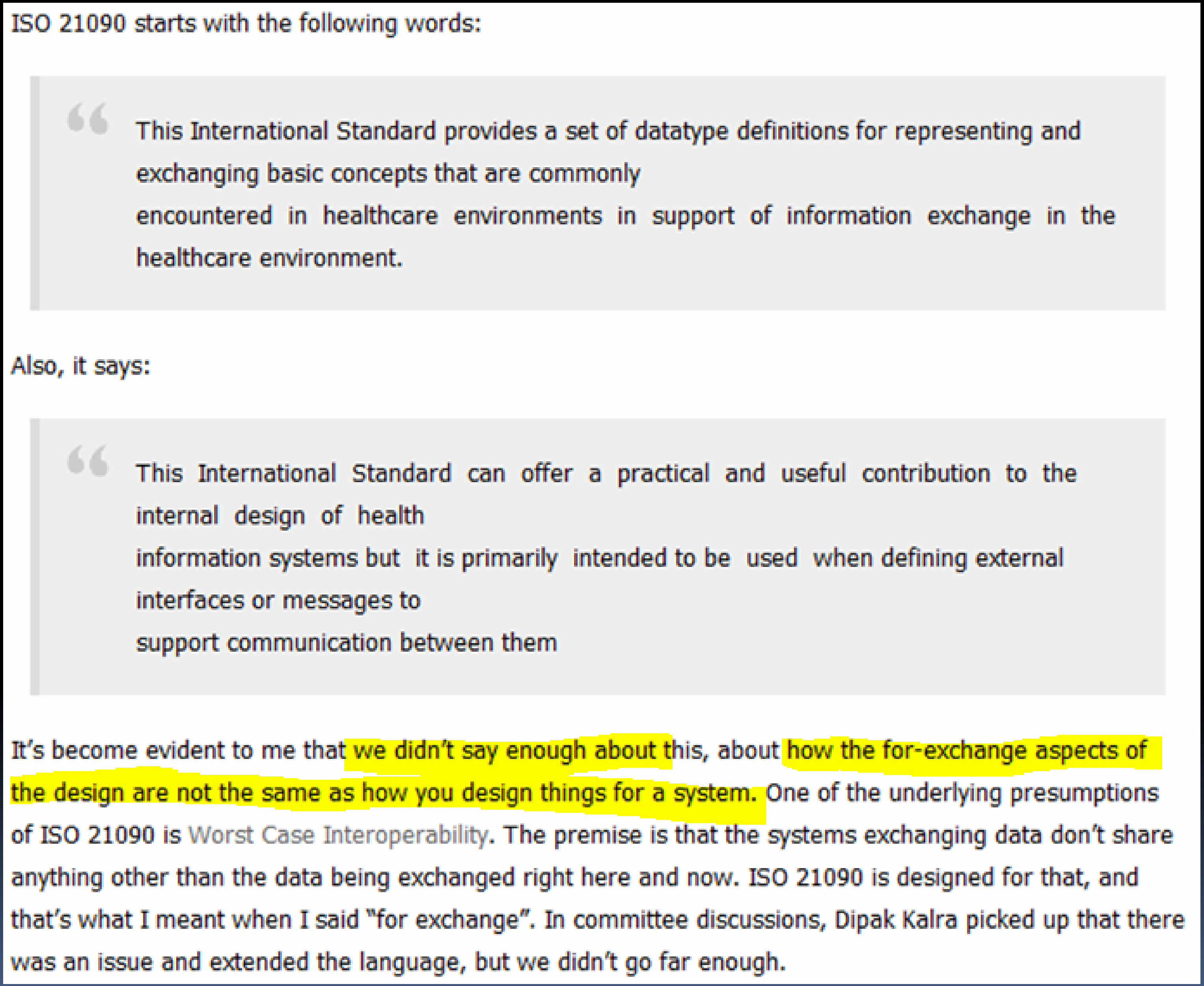
- Ideally we would choose the agreed standard for the domain
- Unfortunately there isn't one
  - There are various 'standards', but they contain major problems
  - The published standards do not have an appropriate scope

ISO 21090 / HL7v3

- For the purposes of DCM, we will treat these models as being the same, since they differ only in details
- Scope: HL7v3 messages
- The model is completely normalised to this use and no other…

ISO 21090 / HL7v3 - opinions

- GG: If you run into ISO 21090 on the fly from some wider perspective, and have to implement a little bit of it, then it's not going to make you happy; the density of the standard (particularly the 'design by condensation' discussed in the next post) is mostly going to be painful, and it's comprehensiveness, along with the value that can be leveraged from a solid implementation – that's going to be irrelevant to you. (HealthIntersections.com.au)



ISO 21090 starts with the following words:

> This International Standard provides a set of datatype definitions for representing and exchanging basic concepts that are commonly
> encountered in healthcare environments in support of information exchange in the healthcare environment.

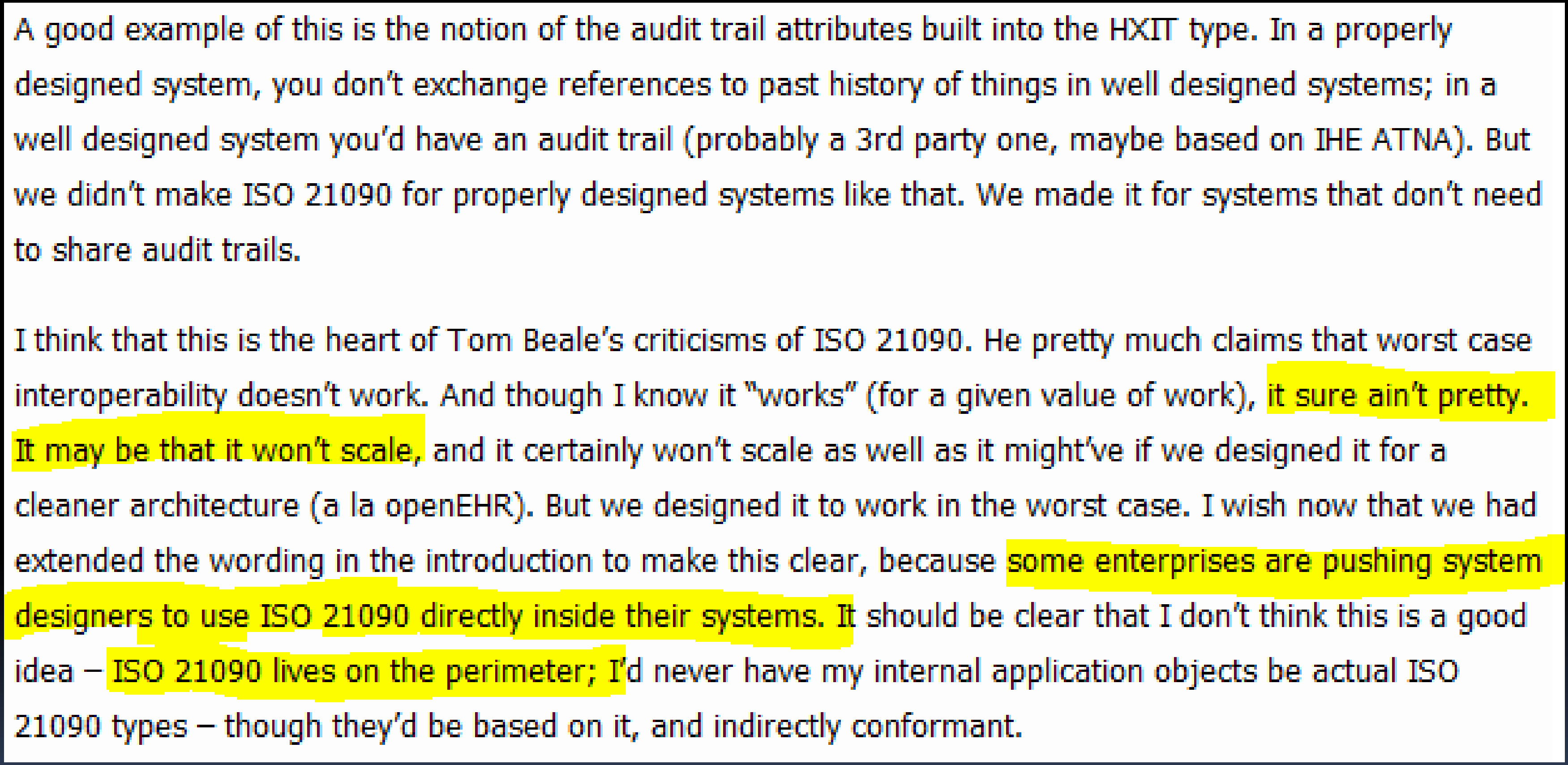
Also, it says:

> This International Standard can offer a practical and useful contribution to the internal design of health
> information systems but it is primarily intended to be used when defining external interfaces or messages to
> support communication between them

It's become evident to me that we didn't say enough about this, about how the for-exchange aspects of the design are not the same as how you design things for a system. One of the underlying presumptions of ISO 21090 is Worst Case Interoperability. The premise is that the systems exchanging data don't share anything other than the data being exchanged right here and now. ISO 21090 is designed for that, and that's what I meant when I said "for exchange". In committee discussions, Dipak Kalra picked up that there was an issue and extended the language, but we didn't go far enough.

openEHR

ISO 21090 / HL7v3 – GG view



A good example of this is the notion of the audit trail attributes built into the HXIT type. In a properly designed system, you don't exchange references to past history of things in well designed systems; in a well designed system you'd have an audit trail (probably a 3rd party one, maybe based on IHE ATNA). But we didn't make ISO 21090 for properly designed systems like that. We made it for systems that don't need to share audit trails.

I think that this is the heart of Tom Beale's criticisms of ISO 21090. He pretty much claims that worst case interoperability doesn't work. And though I know it "works" (for a given value of work), it sure ain't pretty. It may be that it won't scale, and it certainly won't scale as well as it might've if we designed it for a cleaner architecture (a la openEHR). But we designed it to work in the worst case. I wish now that we had extended the wording in the introduction to make this clear, because some enterprises are pushing system designers to use ISO 21090 directly inside their systems. It should be clear that I don't think this is a good idea – ISO 21090 lives on the perimeter; I'd never have my internal application objects be actual ISO 21090 types – though they'd be based on it, and indirectly conformant.

openEHR

ISO 21090 / HL7v3 – my view

- Problems:
  - Formally:
    - It contain systemic design flaws
  - Practical evidence
    - Recognised to be over-complex
    - Difficult to implement
    - Generally MODIFIED by implementers, resulting in private non-standards!!!

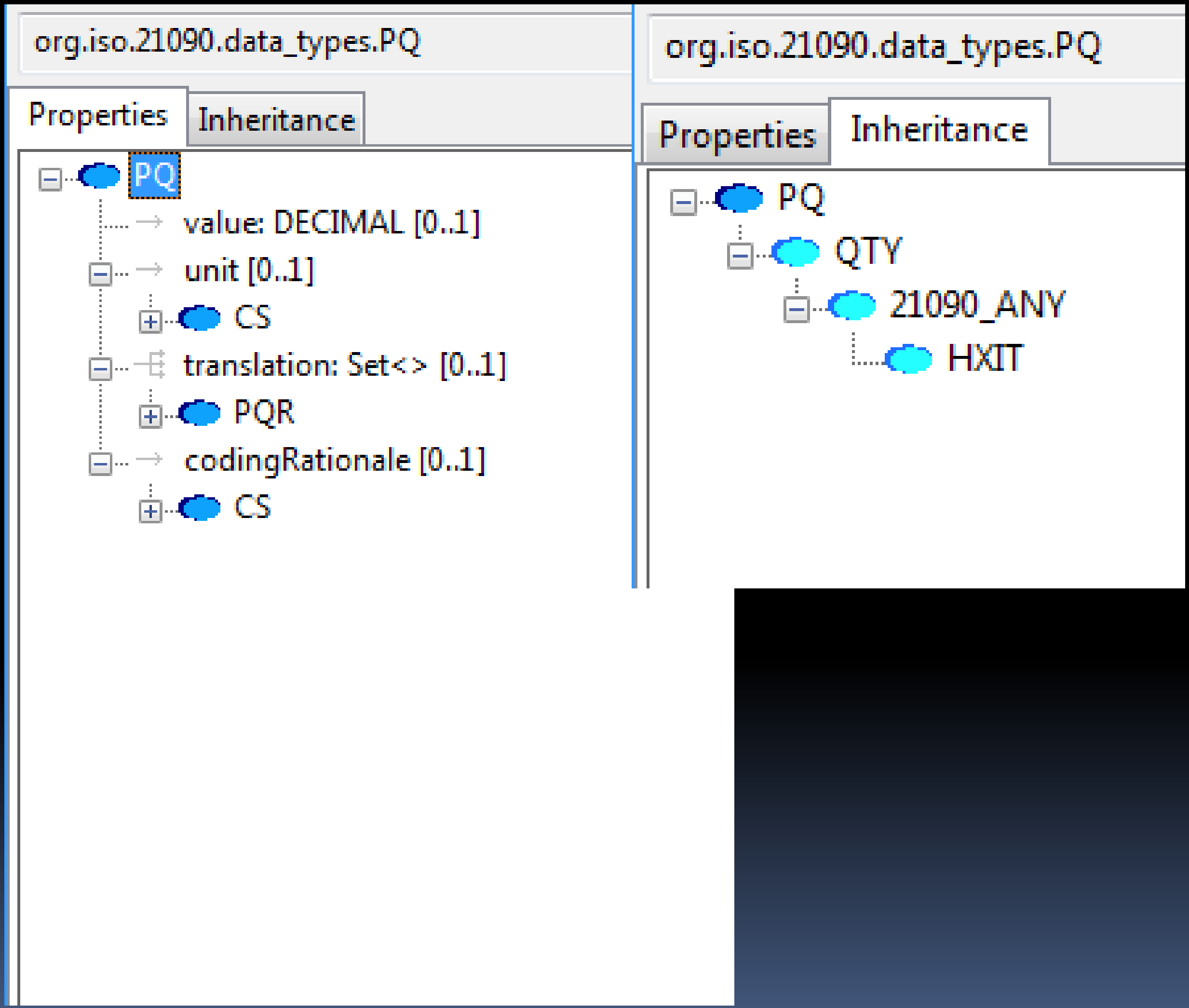
ISO 21090 / HL7v3

- The design choices break basic OO modelling rules and prevent
  - Normal OO implementation
  - Easy constraint / extension by DCMs
- Grahame argues that it is a question of ‘normalisation’ for a particular scope
- Regardless, there is no way to implement a common ‘PQ’ that isn’t full of HL7v3 message attributes, without ‘profiling’

ISO 21090/HL7v3 – problem #1

Illustration:

PQ definition



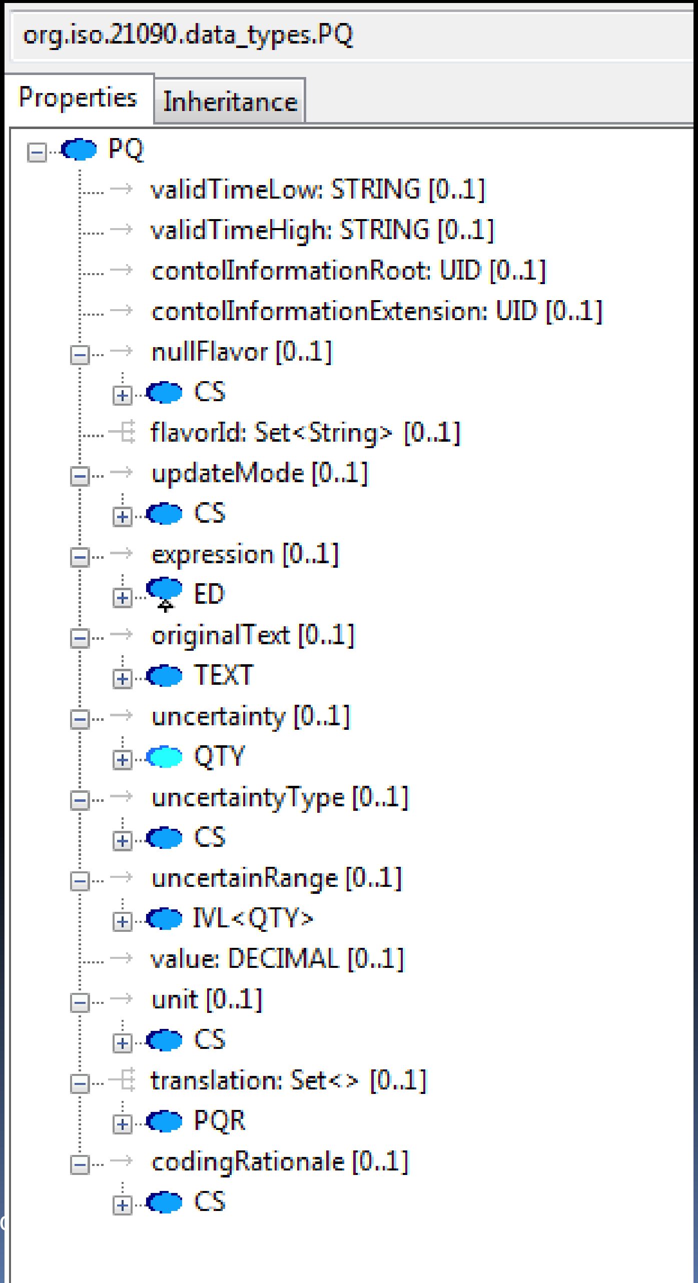
```
org.iso.21090.data_types.PQ

Properties | Inheritance

PQ
  value: DECIMAL [0..1]
  unit [0..1]
    CS
  translation: Set<> [0..1]
    PQR
  codingRationale [0..1]
    CS
```

```
org.iso.21090.data_types.PQ

Properties | Inheritance

PQ
  QTY
    21090_ANY
      HXIT
```

ISO 21090/HL7v3

Illustration:

PQ flat form (the real data)



org.iso.21090.data_types.PQ

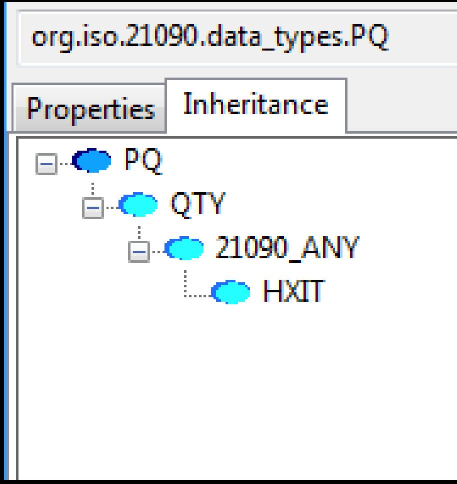
```
PQ
    validTimeLow: STRING [0..1]
    validTimeHigh: STRING [0..1]
    contolInformationRoot: UID [0..1]
    contolInformationExtension: UID [0..1]
    nullFlavor [0..1]
        CS
    flavorId: Set<String> [0..1]
    updateMode [0..1]
        CS
    expression [0..1]
        ED
    originalText [0..1]
        TEXT
    uncertainty [0..1]
        QTY
    uncertaintyType [0..1]
        CS
    uncertainRange [0..1]
        IVL<QTY>
    value: DECIMAL [0..1]
    unit [0..1]
        CS
    translation: Set<> [0..1]
        PQR
    codingRationale [0..1]
        CS
```

openEHR

ISO 21090 / HL7v3

PQ is forced to pick up all these attributes from higher classes, including ANY and HXIT

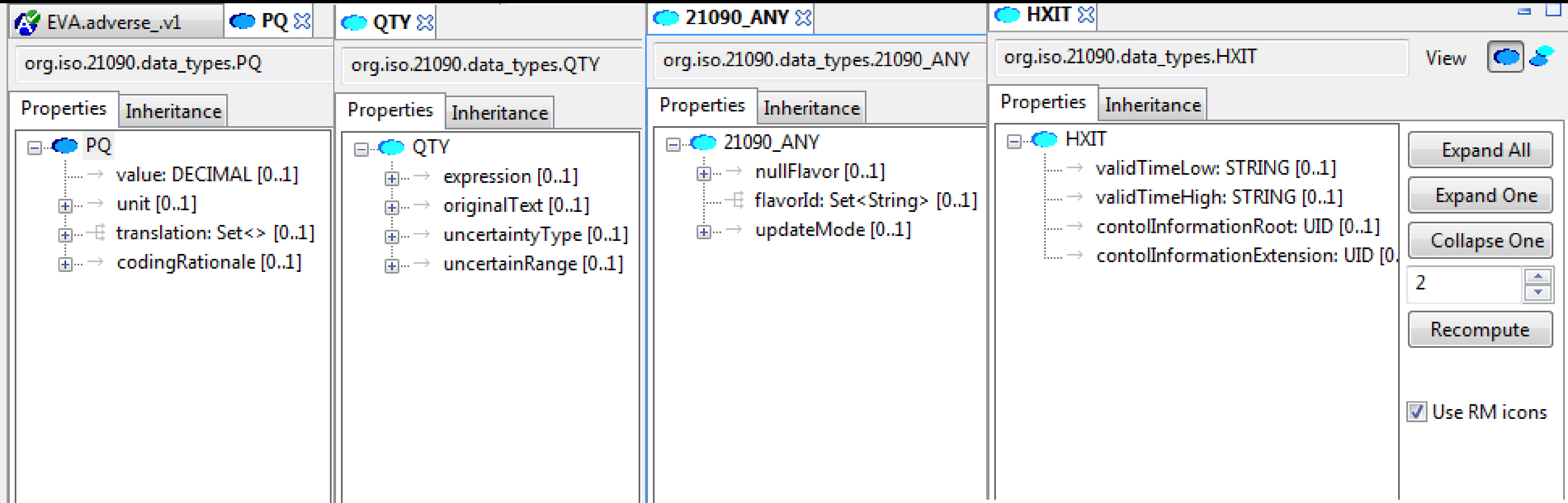
But these are HL7v3 message-specific classes and attributes



org.iso.21090.data_types.PQ

Properties | Inheritance

```
PQ
  QTY
    21090_ANY
      HXIT
```



EVA.adverse_v1 | PQ | QTY | 21090_ANY | HXIT

org.iso.21090.data_types.PQ
- PQ
  - value: DECIMAL [0..1]
  - unit [0..1]
  - translation: Set<> [0..1]
  - codingRationale [0..1]

org.iso.21090.data_types.QTY
- QTY
  - expression [0..1]
  - originalText [0..1]
  - uncertaintyType [0..1]
  - uncertainRange [0..1]

org.iso.21090.data_types.21090_ANY
- 21090_ANY
  - nullFlavor [0..1]
  - flavorId: Set<String> [0..1]
  - updateMode [0..1]

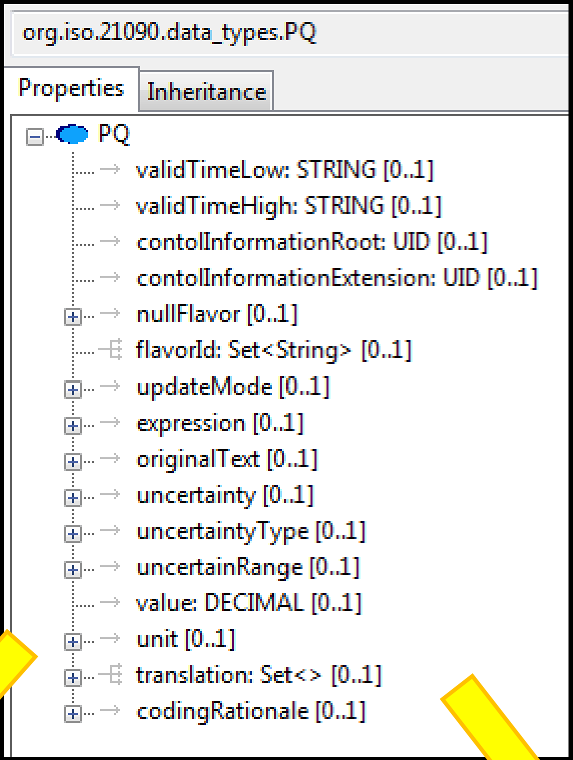
org.iso.21090.data_types.HXIT
- HXIT
  - validTimeLow: STRING [0..1]
  - validTimeHigh: STRING [0..1]
  - contolInformationRoot: UID [0..1]
  - contolInformationExtension: UID [0.

ISO 21090 / HL7v3

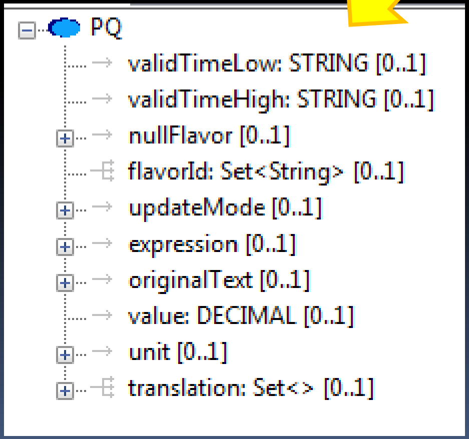
- All of 21090 is built like this (same for HL7v3 data types and the RIM)
- Obtaining the data types required for any particular context is done by subtractive profiling
- Different developers can do this differently

The effect of localised profiling...

And it is happening in real locations

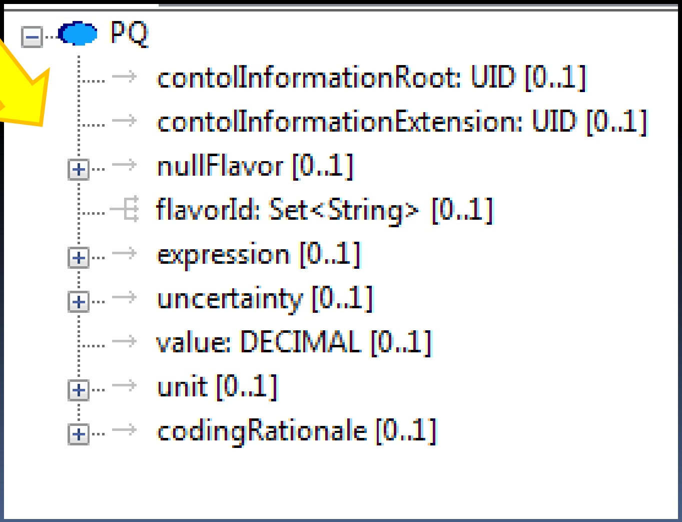


```
org.iso.21090.data_types.PQ
Properties | Inheritance
PQ
  validTimeLow: STRING [0..1]
  validTimeHigh: STRING [0..1]
  contolInformationRoot: UID [0..1]
  contolInformationExtension: UID [0..1]
  nullFlavor [0..1]
  flavorId: Set<String> [0..1]
  updateMode [0..1]
  expression [0..1]
  originalText [0..1]
  uncertainty [0..1]
  uncertaintyType [0..1]
  uncertainRange [0..1]
  value: DECIMAL [0..1]
  unit [0..1]
  translation: Set<> [0..1]
  codingRationale [0..1]
```



```
PQ
  validTimeLow: STRING [0..1]
  validTimeHigh: STRING [0..1]
  nullFlavor [0..1]
  flavorId: Set<String> [0..1]
  updateMode [0..1]
  expression [0..1]
  originalText [0..1]
  value: DECIMAL [0..1]
  unit [0..1]
  translation: Set<> [0..1]
```

Non-interoperable



```
PQ
  contolInformationRoot: UID [0..1]
  contolInformationExtension: UID [0..1]
  nullFlavor [0..1]
  flavorId: Set<String> [0..1]
  expression [0..1]
  uncertainty [0..1]
  value: DECIMAL [0..1]
  unit [0..1]
  codingRationale [0..1]
```

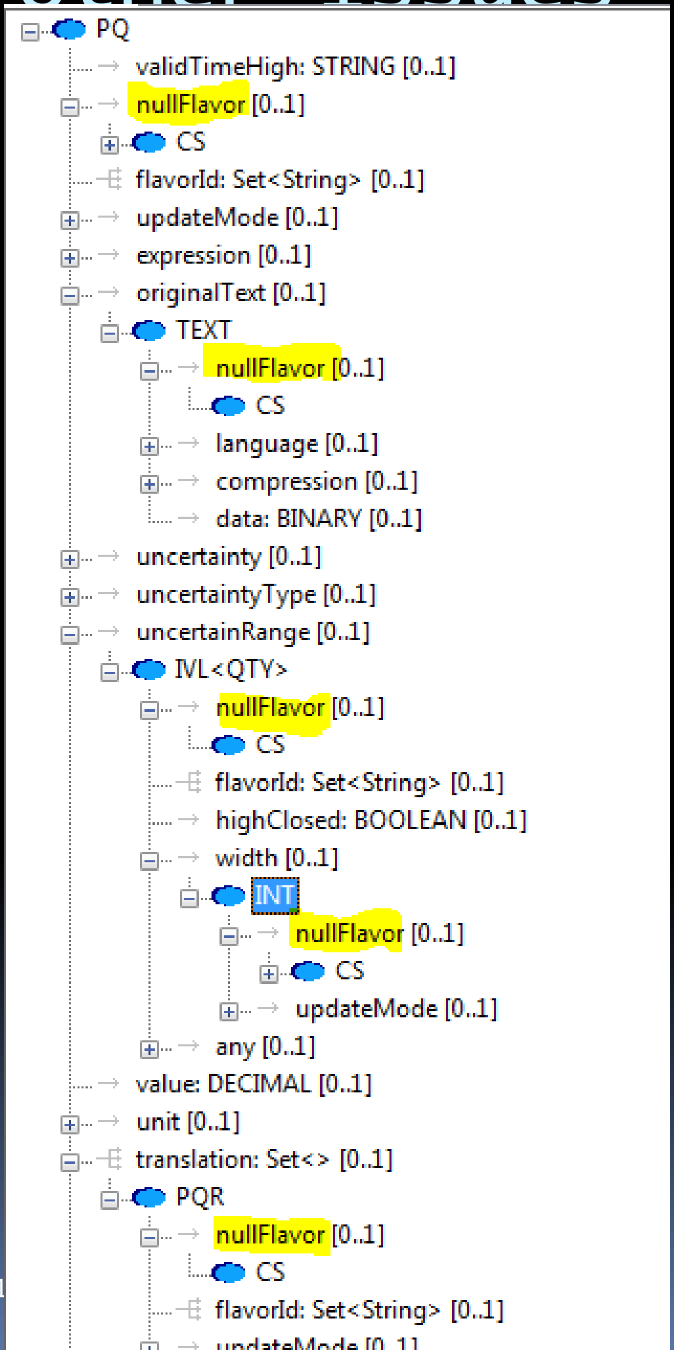
openEHR

ISO 21090 / HL7v3

- Controlled profiling might help
- But the underlying problem is that any derivatives (profiles) are not guaranteed by the modelling approach to be interoperable, only by other agreements
- Normal OO modelling doesn't have this problem
- This is why 'HL7 profiling' is not used anywhere in industry
- See my blog for details

ISO 21090/HL7v3 – other issues

Null flavours everywhere

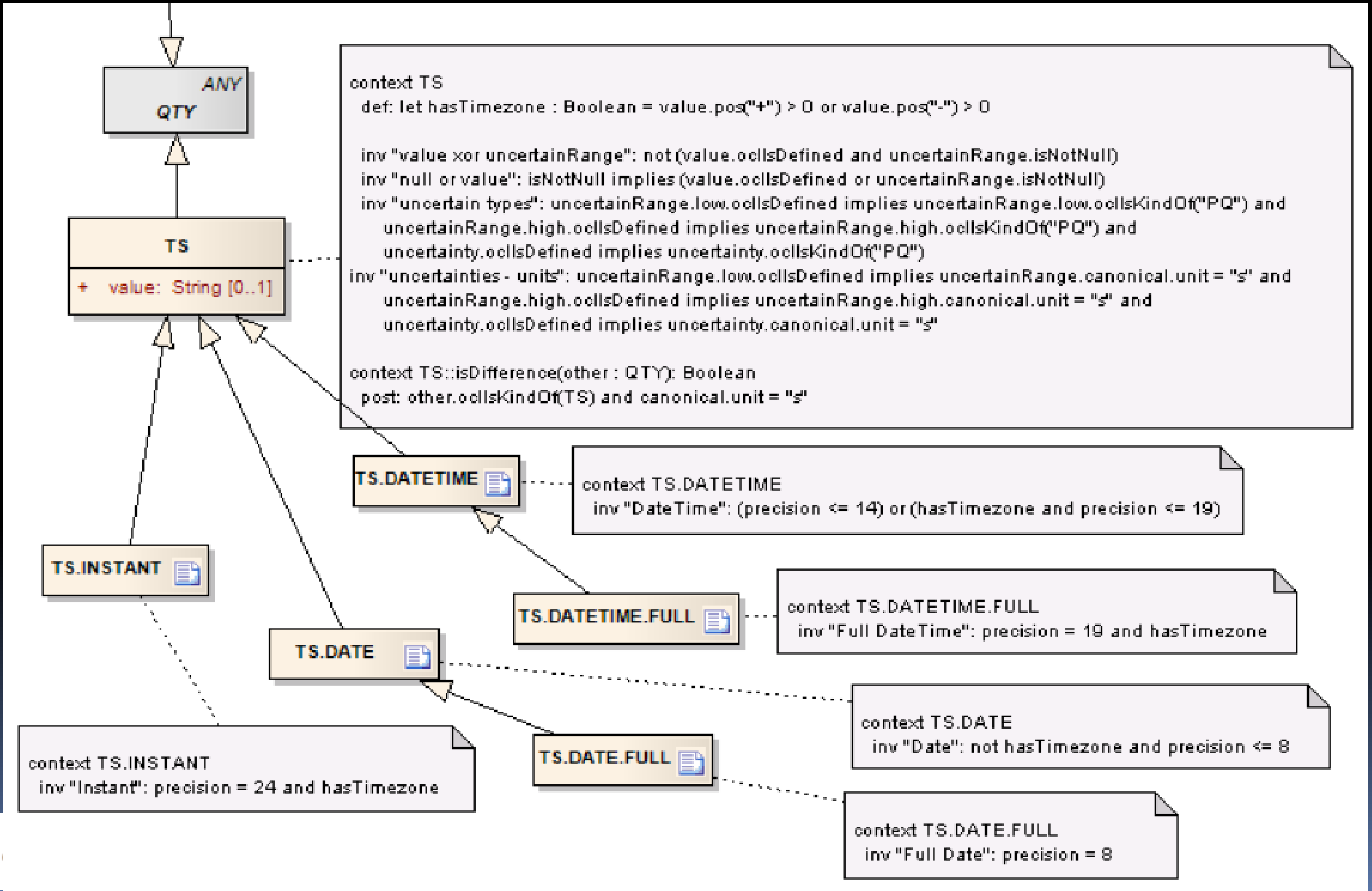


```
PQ
  validTimeHigh: STRING [0..1]
  nullFlavor [0..1]
    CS
  flavorId: Set<String> [0..1]
  updateMode [0..1]
  expression [0..1]
  originalText [0..1]
    TEXT
      nullFlavor [0..1]
        CS
      language [0..1]
      compression [0..1]
      data: BINARY [0..1]
  uncertainty [0..1]
  uncertaintyType [0..1]
  uncertainRange [0..1]
    IVL<QTY>
      nullFlavor [0..1]
        CS
      flavorId: Set<String> [0..1]
      highClosed: BOOLEAN [0..1]
      width [0..1]
        INT
          nullFlavor [0..1]
            CS
          updateMode [0..1]
      any [0..1]
  value: DECIMAL [0..1]
  unit [0..1]
  translation: Set<> [0..1]
    PQR
      nullFlavor [0..1]
        CS
      flavorId: Set<String> [0..1]
      updateMode [0..1]
```

ISO 21090/HL7v3 – other issues

- Conceptually simple types have become complex…

Basic types like DATE are now restrictions on TS



```
context TS
  def: let hasTimezone : Boolean = value.pos("+") > 0 or value.pos("-") > 0

  inv "value xor uncertainRange": not (value.oclIsDefined and uncertainRange.isNotNull)
  inv "null or value": isNotNull implies (value.oclIsDefined or uncertainRange.isNotNull)
  inv "uncertain types": uncertainRange.low.oclIsDefined implies uncertainRange.low.oclIsKindOf("PQ") and
      uncertainRange.high.oclIsDefined implies uncertainRange.high.oclIsKindOf("PQ") and
      uncertainty.oclIsDefined implies uncertainty.oclIsKindOf("PQ")
inv "uncertainties - units": uncertainRange.low.oclIsDefined implies uncertainRange.canonical.unit = "s" and
      uncertainRange.high.oclIsDefined implies uncertainRange.high.canonical.unit = "s" and
      uncertainty.oclIsDefined implies uncertainty.canonical.unit = "s"

context TS::isDifference(other : QTY): Boolean
  post: other.oclIsKindOf(TS) and canonical.unit = "s"
```

```
context TS.DATETIME
  inv "DateTime": (precision <= 14) or (hasTimezone and precision <= 19)
```

```
context TS.DATETIME.FULL
  inv "Full DateTime": precision = 19 and hasTimezone
```

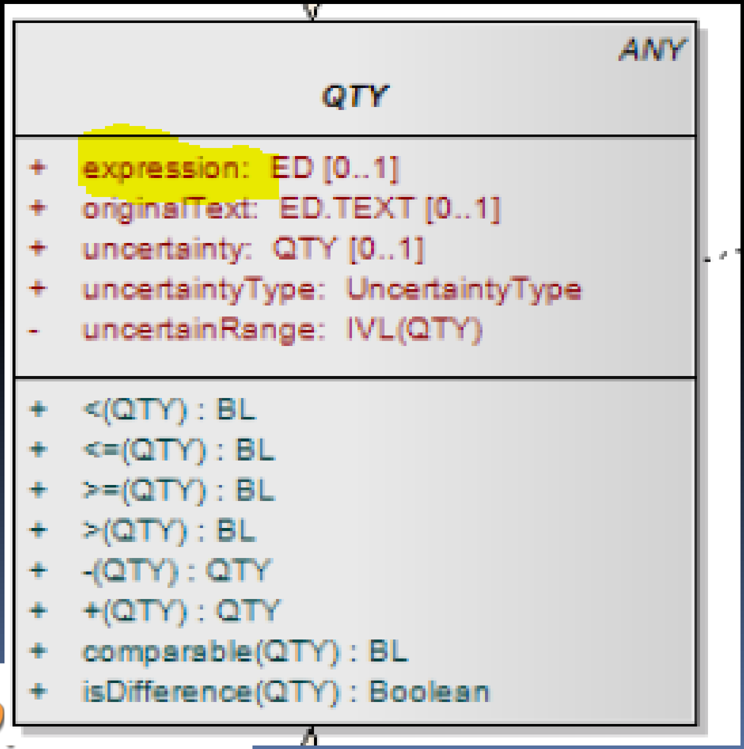
```
context TS.DATE
  inv "Date": not hasTimezone and precision <= 8
```

```
context TS.INSTANT
  inv "Instant": precision = 24 and hasTimezone
```

```
context TS.DATE.FULL
  inv "Full Date": precision = 8
```

ISO 21090/HL7v3 – other issues

- Due to modelling approach that tries to put attributes in classes for EVERY POSSIBLE USE CASE, many general classes have attributes of extremely narrow applicability


ANY
QTY
+ expression: ED [0..1]
+ originalText: ED.TEXT [0..1]
+ uncertainty: QTY [0..1]
+ uncertaintyType: UncertaintyType
- uncertainRange: IVL(QTY)
+ <(QTY) : BL
+ <=(QTY) : BL
+ >=(QTY) : BL
+ >(QTY) : BL
+ -(QTY) : QTY
+ +(QTY) : QTY
+ comparable(QTY) : BL
+ isDifference(QTY) : Boolean

7.8.2.3.1 expression : ED: An expression that can be used to derive the actual value of the quantitive given information taken from the context of use.

For example expression can be used for expressing dosage instructions that depend on patient's body weight.

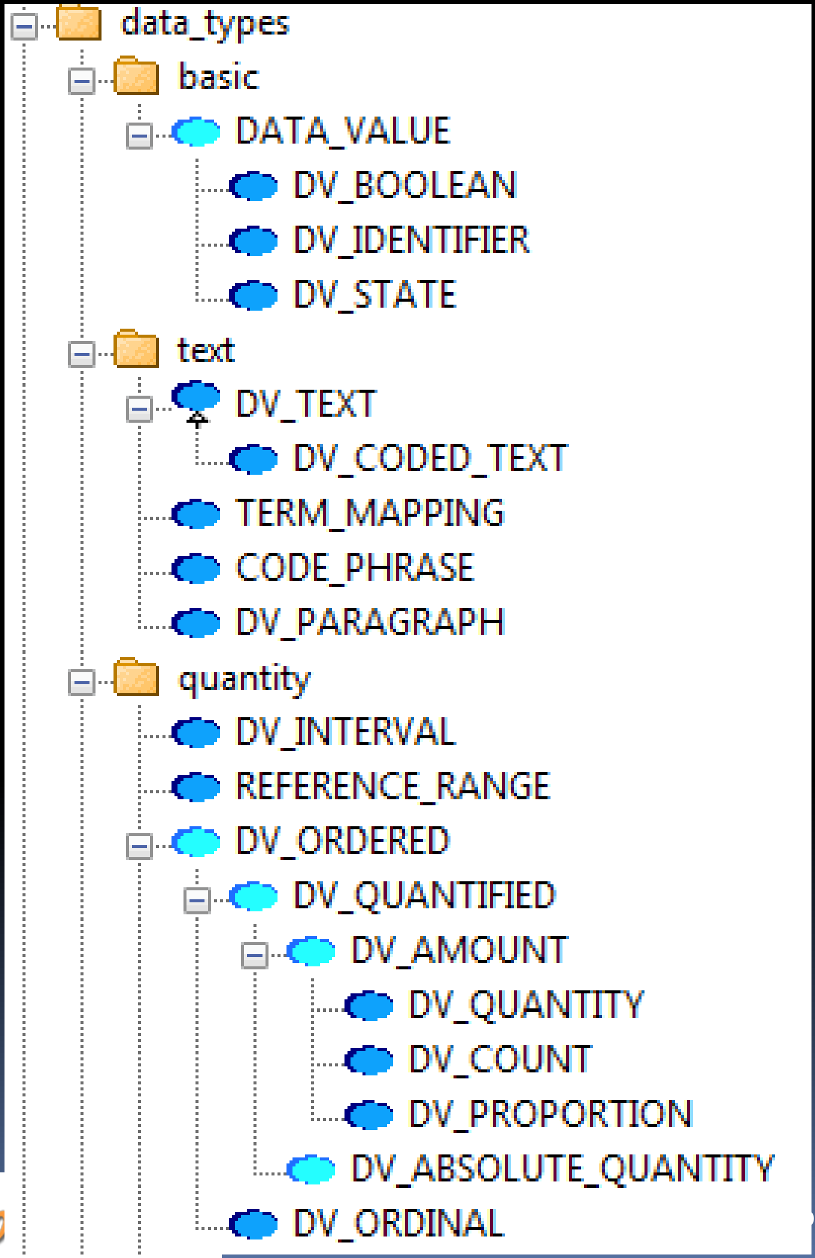
21090 - consequences for DCM

- It is not easy to separate basic 'patterns' from the mass of HL7v3 messaging attributes
- Inheritance appears upside-down
- If they were used, a 'DCM' profile would be needed.
    - This means work, and how would it be synchrhonised with other 21090 profiles?

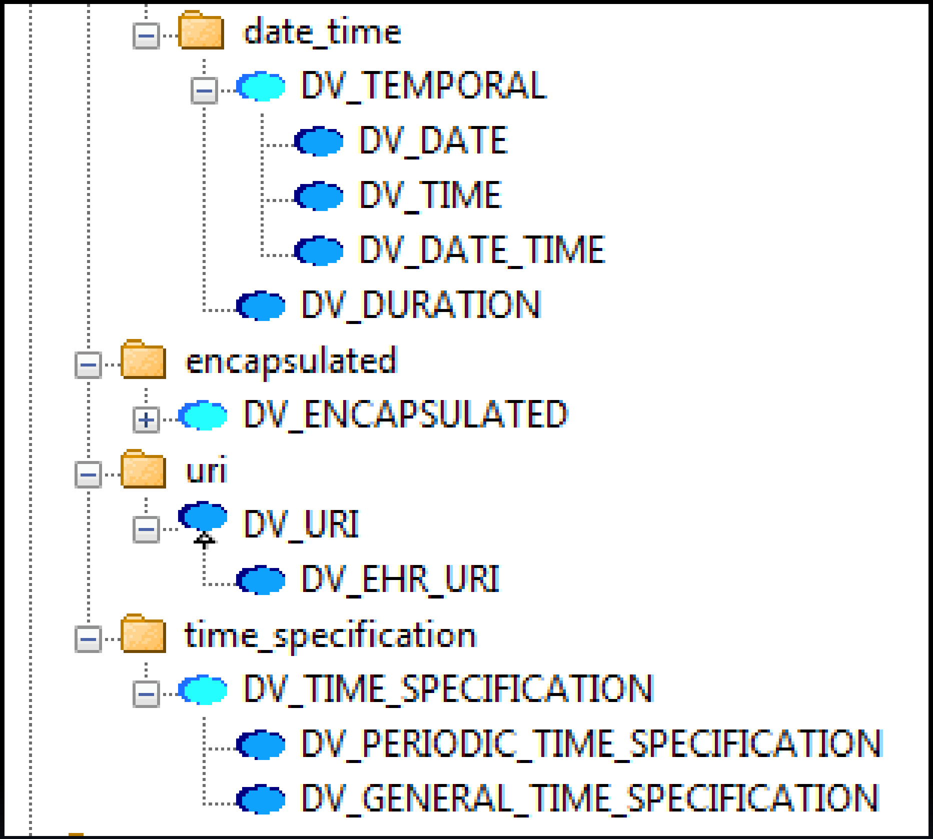
Data types – other possibilities

- openEHR?
  - Follows normal OO modelling rules
  - Uses HL7 types like ED, GTS
- One thing that is different is that every data type has been verified to be needed in archetypes already.
- → could provide a candidate starting structure

Data types – openEHR



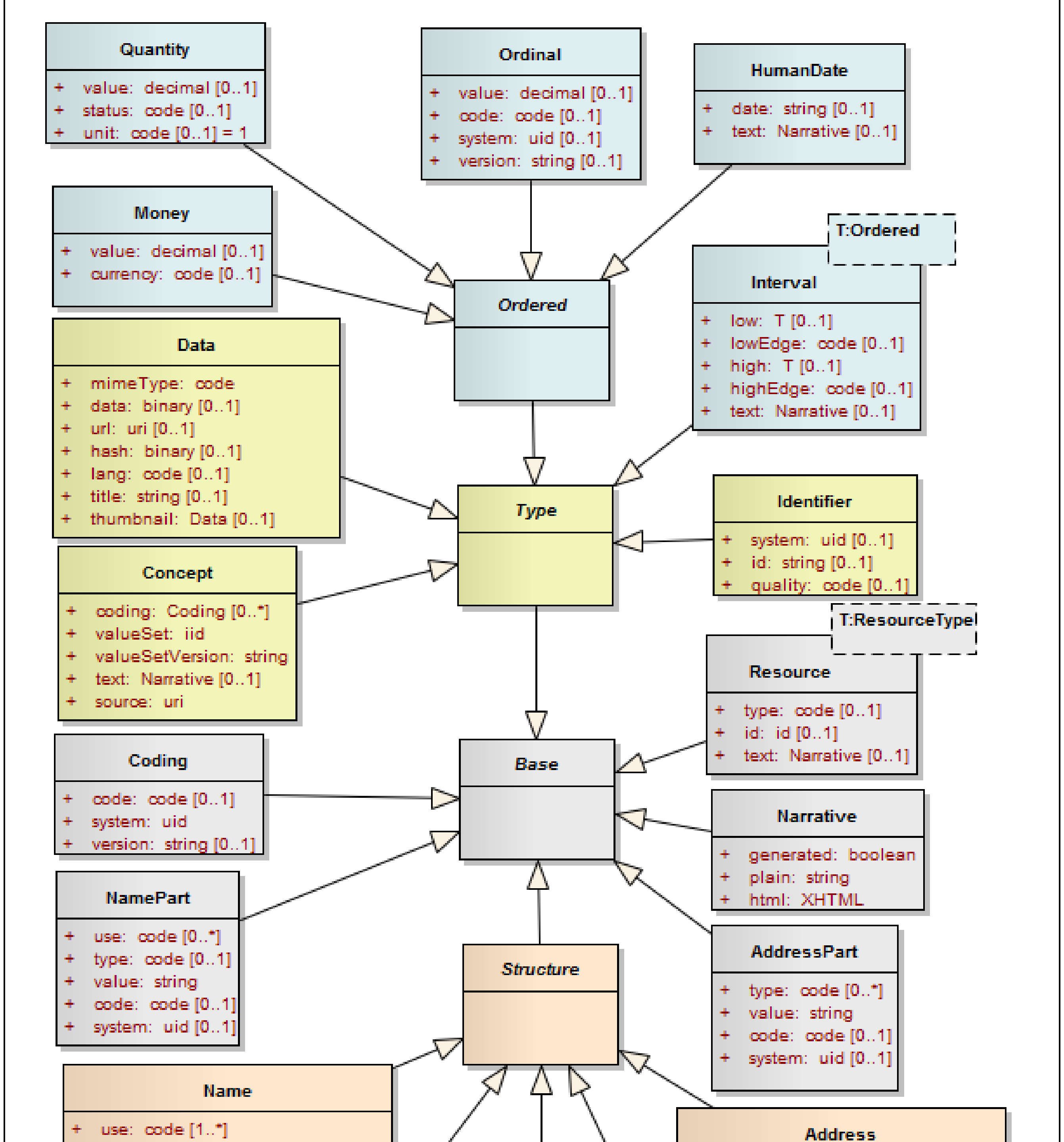
- data_types
  - basic
    - DATA_VALUE
      - DV_BOOLEAN
      - DV_IDENTIFIER
      - DV_STATE
  - text
    - DV_TEXT
      - DV_CODED_TEXT
    - TERM_MAPPING
    - CODE_PHRASE
    - DV_PARAGRAPH
  - quantity
    - DV_INTERVAL
    - REFERENCE_RANGE
    - DV_ORDERED
      - DV_QUANTIFIED
        - DV_AMOUNT
          - DV_QUANTITY
          - DV_COUNT
          - DV_PROPORTION
        - DV_ABSOLUTE_QUANTITY
      - DV_ORDINAL



- date_time
  - DV_TEMPORAL
    - DV_DATE
    - DV_TIME
    - DV_DATE_TIME
  - DV_DURATION
- encapsulated
  - DV_ENCAPSULATED
- uri
  - DV_URI
    - DV_EHR_URI
- time_specification
  - DV_TIME_SPECIFICATION
    - DV_PERIODIC_TIME_SPECIFICATION
    - DV_GENERAL_TIME_SPECIFICATION

Data types – HL7 Fresh Look

- HL7 Fresh Look – has led to Grahame Grieve’s Resources for Health
- Including a new Data Types proposal for HL7


Quantity
+ value: decimal [0..1]
+ status: code [0..1]
+ unit: code [0..1] = 1
Ordinal
+ value: decimal [0..1]
+ code: code [0..1]
+ system: uid [0..1]
+ version: string [0..1]
HumanDate
+ date: string [0..1]
+ text: Narrative [0..1]
Money
+ value: decimal [0..1]
+ currency: code [0..1]
Ordered
T:Ordered
Interval
+ low: T [0..1]
+ lowEdge: code [0..1]
+ high: T [0..1]
+ highEdge: code [0..1]
+ text: Narrative [0..1]
Data
+ mimeType: code
+ data: binary [0..1]
+ url: uri [0..1]
+ hash: binary [0..1]
+ lang: code [0..1]
+ title: string [0..1]
+ thumbnail: Data [0..1]
Type
Identifier
+ system: uid [0..1]
+ id: string [0..1]
+ quality: code [0..1]
Concept
+ coding: Coding [0..*]
+ valueSet: iid
+ valueSetVersion: string
+ text: Narrative [0..1]
+ source: uri
T:ResourceType
Resource
+ type: code [0..1]
+ id: id [0..1]
+ text: Narrative [0..1]
Base
Coding
+ code: code [0..1]
+ system: uid
+ version: string [0..1]
Narrative
+ generated: boolean
+ plain: string
+ html: XHTML
NamePart
+ use: code [0..*]
+ type: code [0..1]
+ value: string
+ code: code [0..1]
+ system: uid [0..1]
AddressPart
+ type: code [0..*]
+ value: string
+ code: code [0..1]
+ system: uid [0..1]
Structure
Name
+ use: code [1..*]
Address

openEH

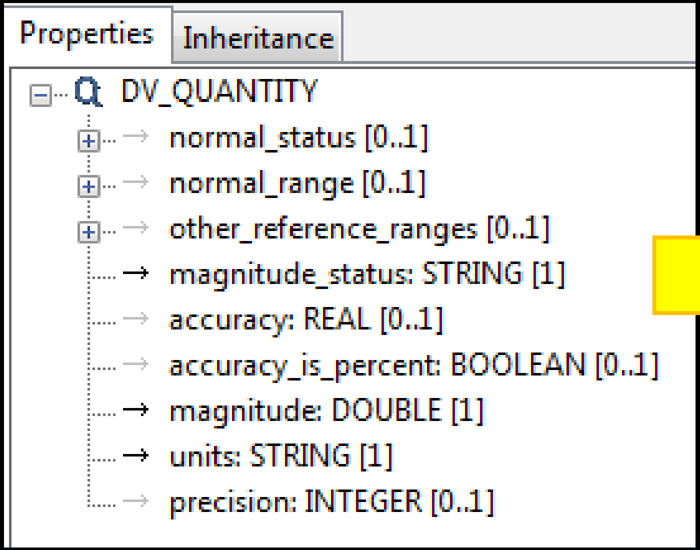
My recommendation

- Decide on a starting point that everyone can at least agree as the *starting point*!
  - E.g. Grahame's DTs
- Work on this to ensure it covers required types
  - E.g. Some missing ones from openEHR – DV_PROPORTION
  - Missing types from HL7/21090
- Then…. Determine a minimal definition of each class required for DCMs

openEHR

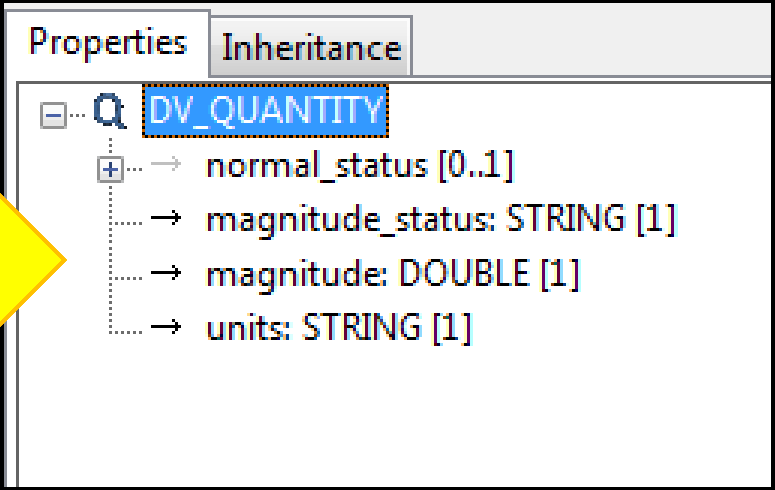
My recommendation

- E.g. Minimal definition of openehr Quantity



Properties | Inheritance

```
DV_QUANTITY
  normal_status [0..1]
  normal_range [0..1]
  other_reference_ranges [0..1]
  magnitude_status: STRING [1]
  accuracy: REAL [0..1]
  accuracy_is_percent: BOOLEAN [0..1]
  magnitude: DOUBLE [1]
  units: STRING [1]
  precision: INTEGER [0..1]
```



Properties | Inheritance

```
DV_QUANTITY
  normal_status [0..1]
  magnitude_status: STRING [1]
  magnitude: DOUBLE [1]
  units: STRING [1]
```

- Only these attributes ever get archetyped

My recommendation

- The work for this activity will be far less than trying to 'profile' 21090 or v3 data types.
- Only the core types have to be done initially, e.g.
  - Text, CodedText, Code
  - Quantity, Count, Ordinal
  - Date, Time, DateTime, Duration
  - Boolean

Resources

- openEHR ADL Workbench
- HealthIntersections – Grahame Grieve
  - ISO 21090
  - Null flavor
  - Resources for Health