DCMs and the Reference Model

Thomas Beale, for San Diego, Sep 2011

[note]

- Slides 3-9 identical to DCM_and_datatypes slides

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

openEHR

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

openEHR

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

About the Reference Model

- Data Types are the most basic patterns required
- The Reference Model is just higher-level patterns

openEHR

Therefore...

- Rather than debating what reference model among published EHR and other models should be used, we should...
- Identify the key patterns needed for creating real DCMs
- And define our DCM-RM based on that

Therefore...

- The DCM-RM should also provide good semantics for computing with data
- It if doesn't, it means there is a gap between how data are logically represented and how they are processed – this should be avoided where possible

What is an RM pattern?

- A well-known one is the Actor-participation-entity one
  - Ubiquitous in systems,
  - Described by Martin Fowler in 1997 book "Analysis Patterns"
  - Used in HL7 RIM & derivatives (inc. CDA), EN13606, openEHR and many other places
- We are looking for things like this, expressed in a clean, clear, minimalistic way

How to find patterns

- Method #1
  - Sit down for years and think really hard
- Method #2
  - Create a model with some patterns and try to build DCMs from it
- Method #3
  - Comb existing models, literature etc for good patterns and steal them…

How to find patterns

- All methods work
- We have limited time now
- But a lot of experience with existing models

openEHR

How openEHR did it...

- Building an initial RM
- By trying to build archetypes over 10 years...
- And changing the RM so provide the required patterns

openEHR

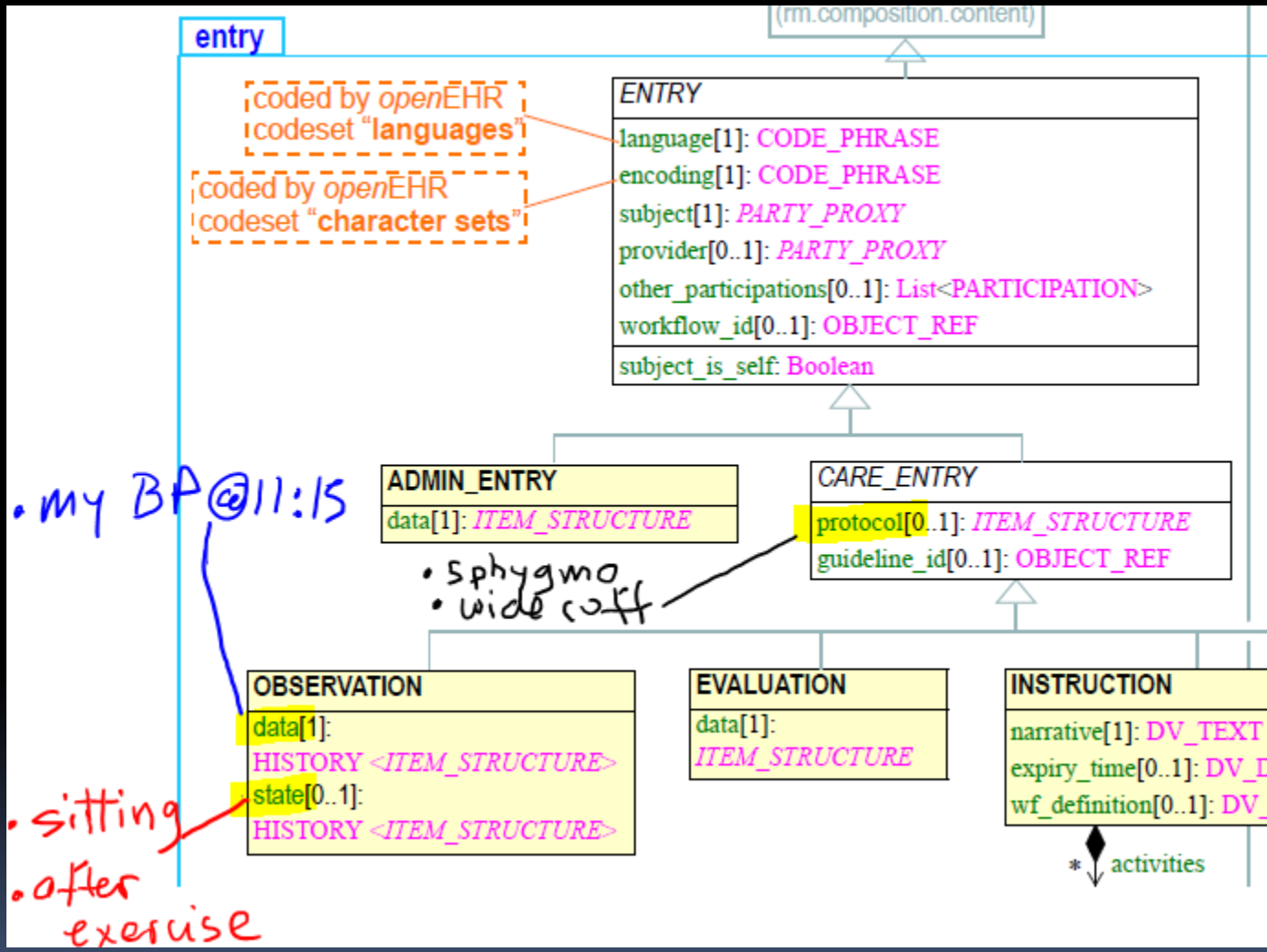
openEHR RM patterns

Pattern	Description
+ data / state / protocol (/reasoning)	In observation data, separate out data (actual datum being recorded e.g. BP) from patient state (e.g. lying, standing) and protocol (cuff type, instrument type)
+ History of events	Provide a structure containing 1..* Events, allowing data and patient state at each one, supporting intervals, point events, and math functions, e.g. ave/delta/max/min
+ Tree structure	Generalised free-form tree for containing clusters of data items, e.g. the 5+1 Apgar items, numerous microbiology result items, etc.
++ Order state machine	A way of recording current state in progression through a standard state machine applying to any 'order'
++ Composition / document	An aggregation concept acting as a 'bucket' for information recorded by a professional at a given time for a given subject of care.
+ Participation	A pattern defining the connection between parties (people, organisations, devices) and other information.
+++ Party / role / accountability	A pattern defining relationships between parties, including those that are roles played by some underlying actor.

#1 – Observation data/state/protocol

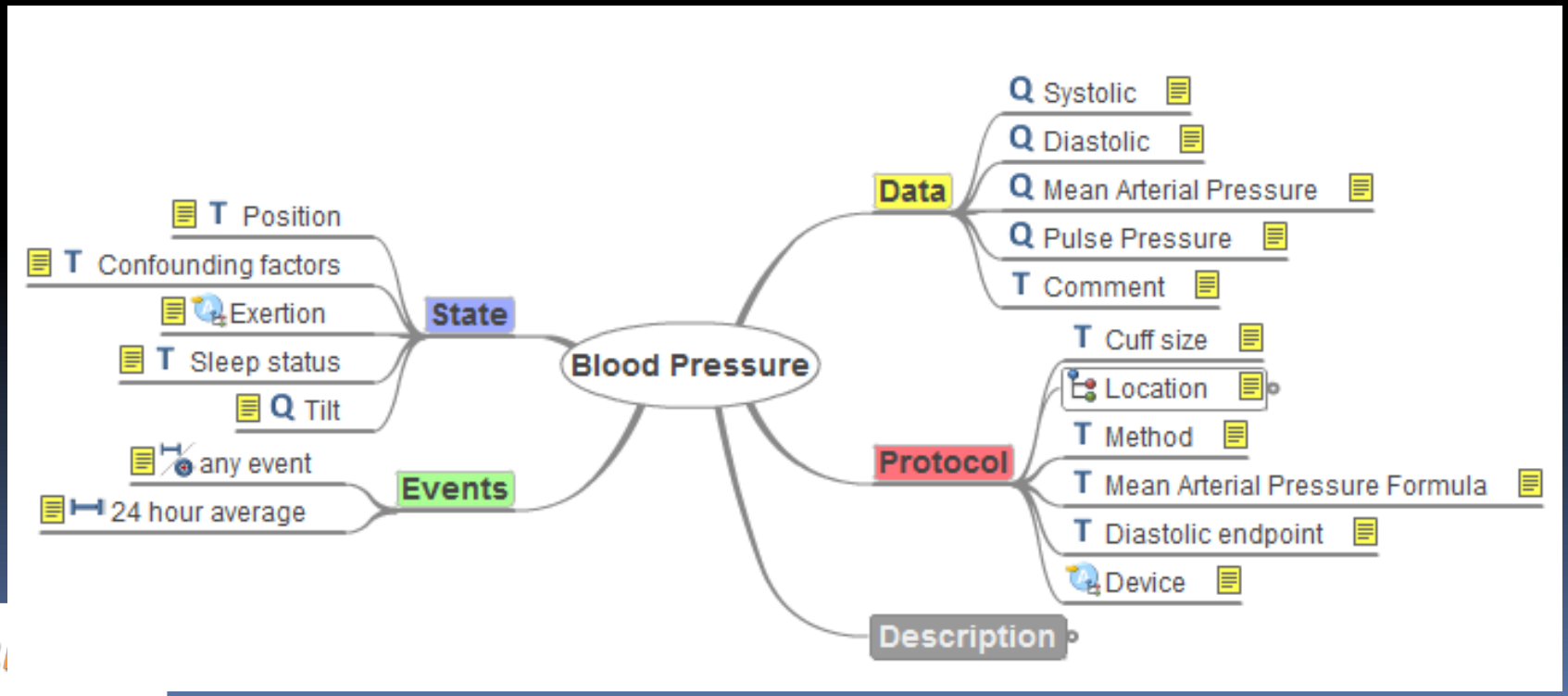
These 3 things potentially apply to nearly every scientific observation

But if mixed up, make the data hard to compute with



Why it's useful

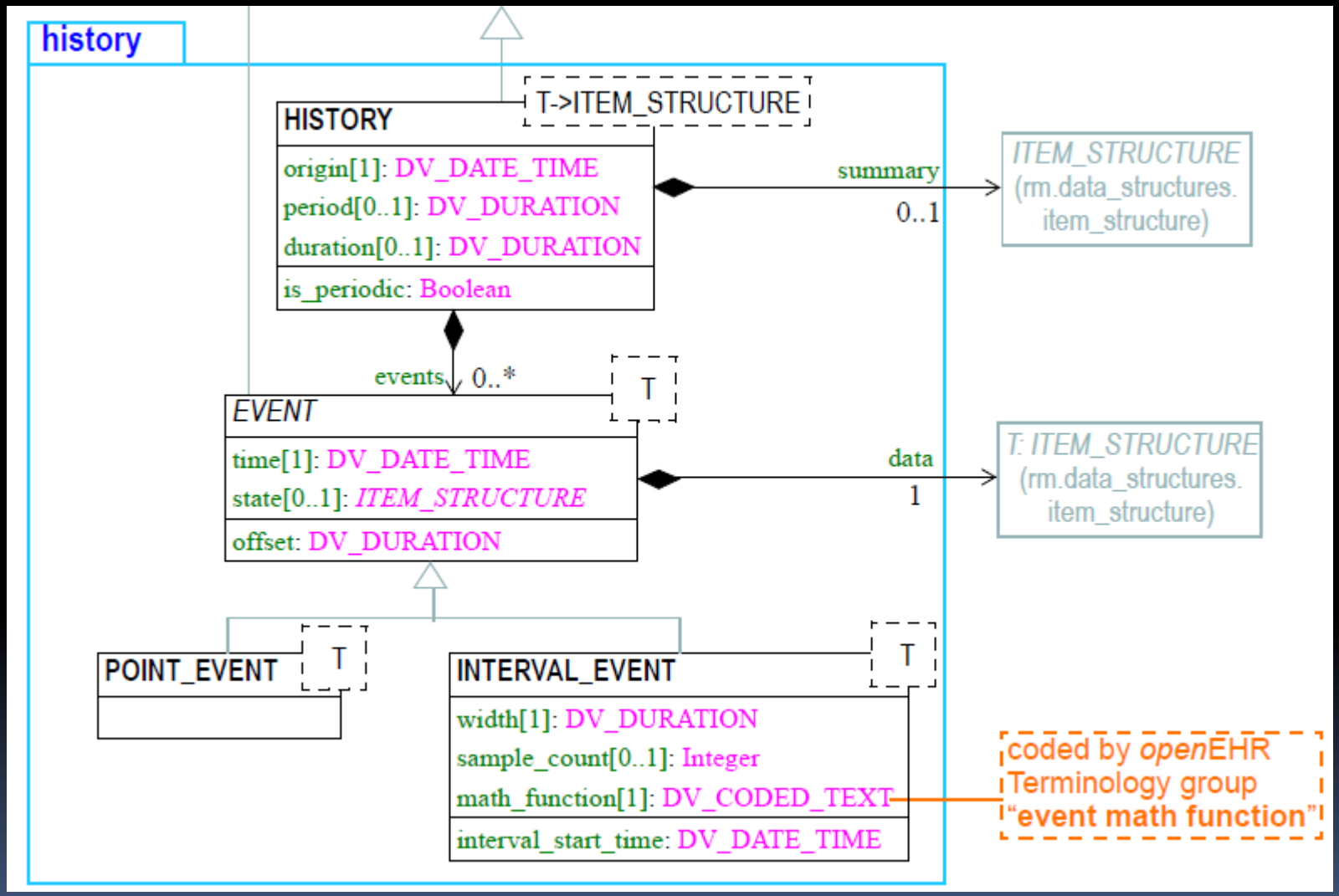
- Helps separate things like 'exercise', 'cuff size' and 'mean arterial pressure'
- Designers know where to put specific data points



Why it's useful

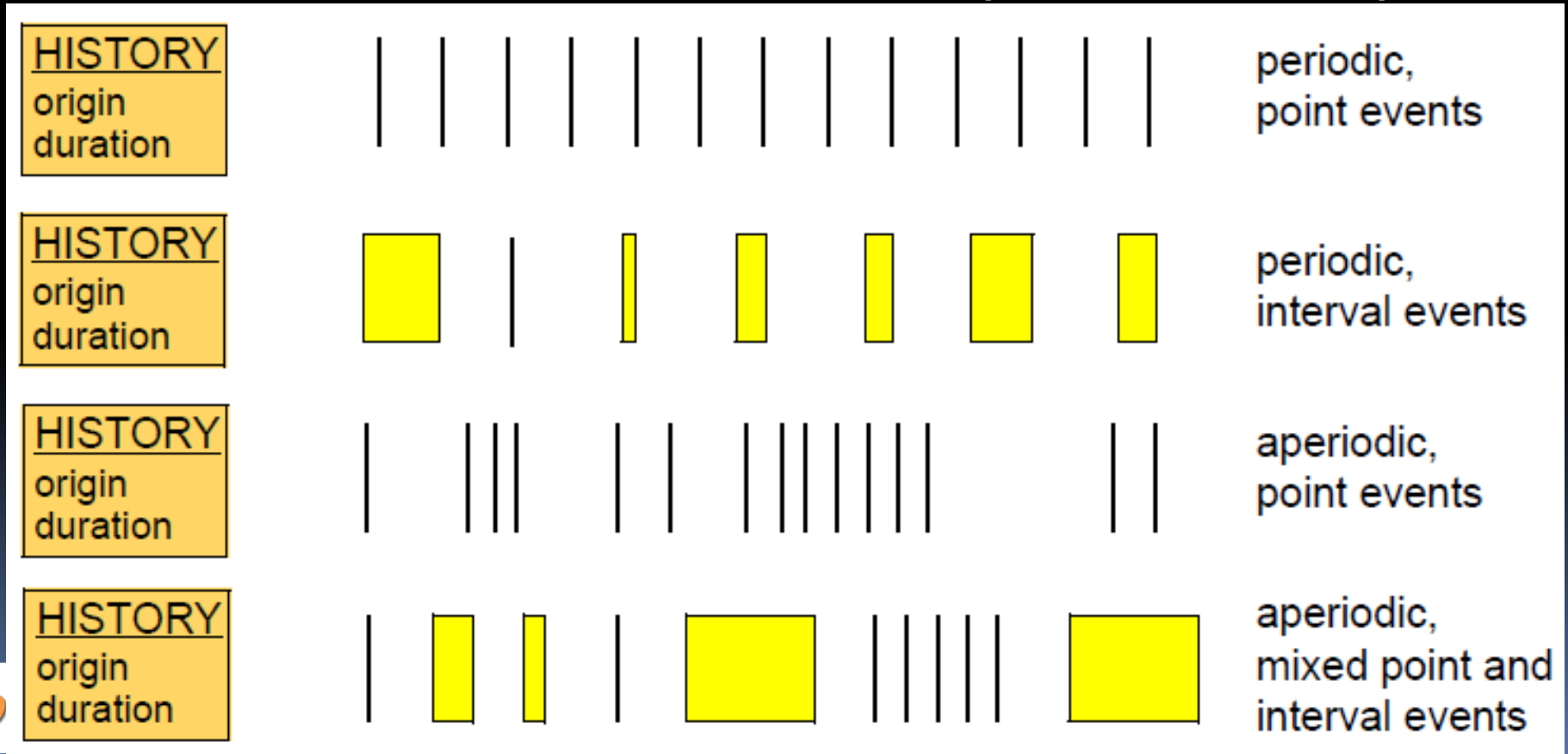
- UI designers know what is *needed* on the screen (data + state) and what can be optionally displayed (protocol)
- Developers know where to find the actual data e.g. to draw a trend – systolic pressure will never be mixed up with patient position or instrument type…

#2 – History of Events



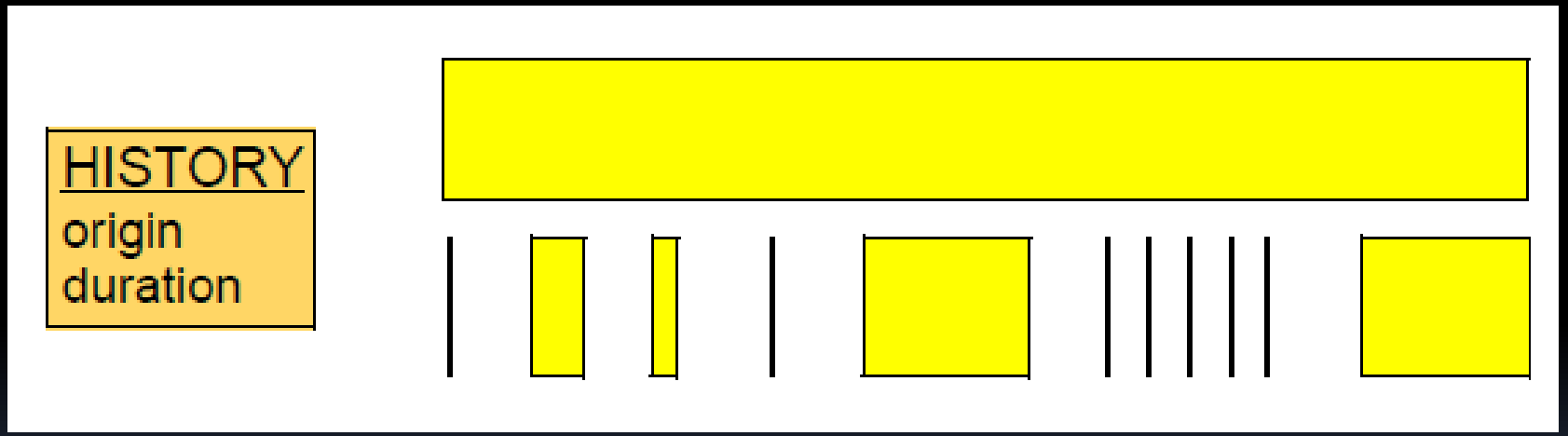
Why it's useful

- Supports Point and Interval Event types, periodic, aperiodic
- Allows software to treat 1 sample like N samples



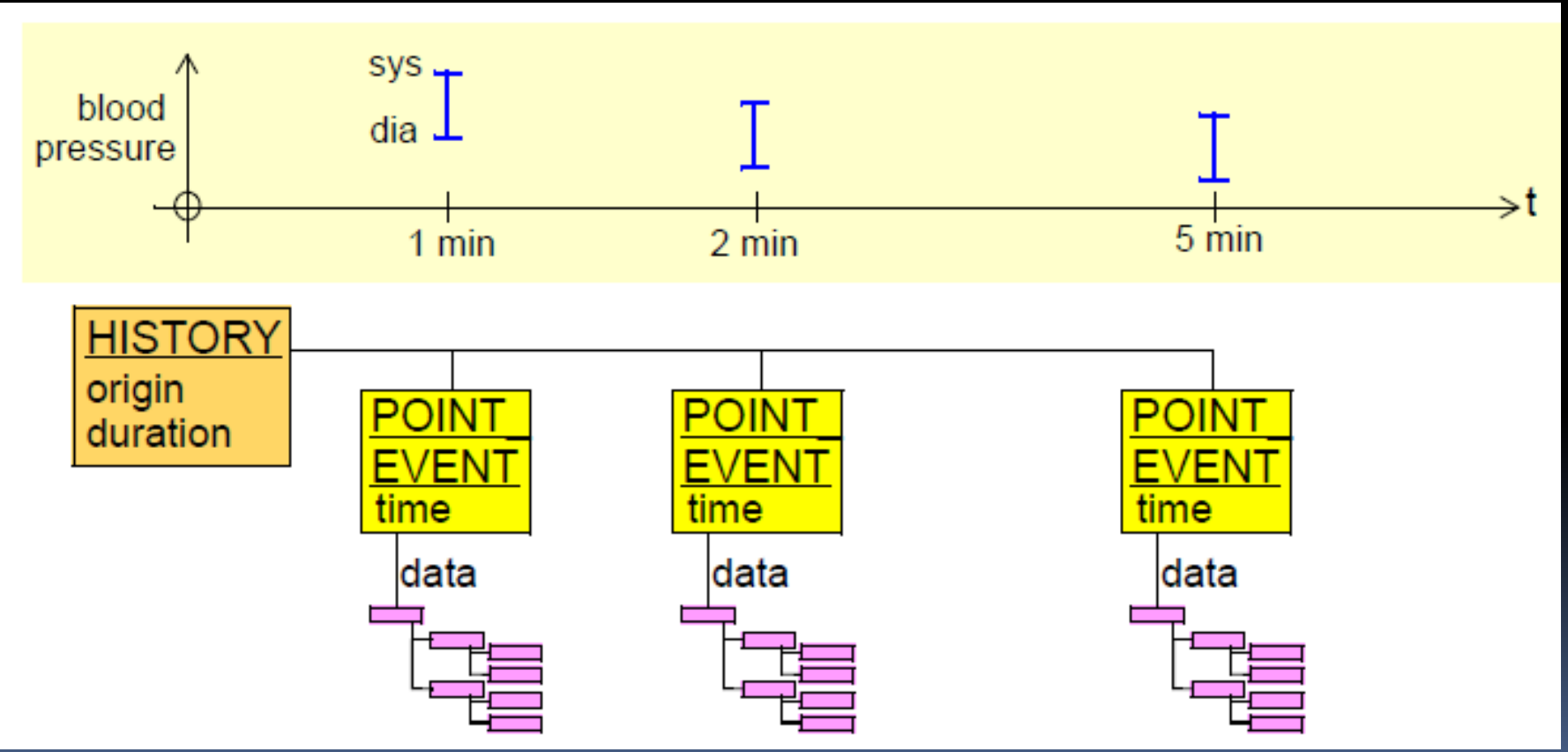
Why it's useful

- Supports overlapping events



Why it's useful

- Supports any complexity of data at each sample

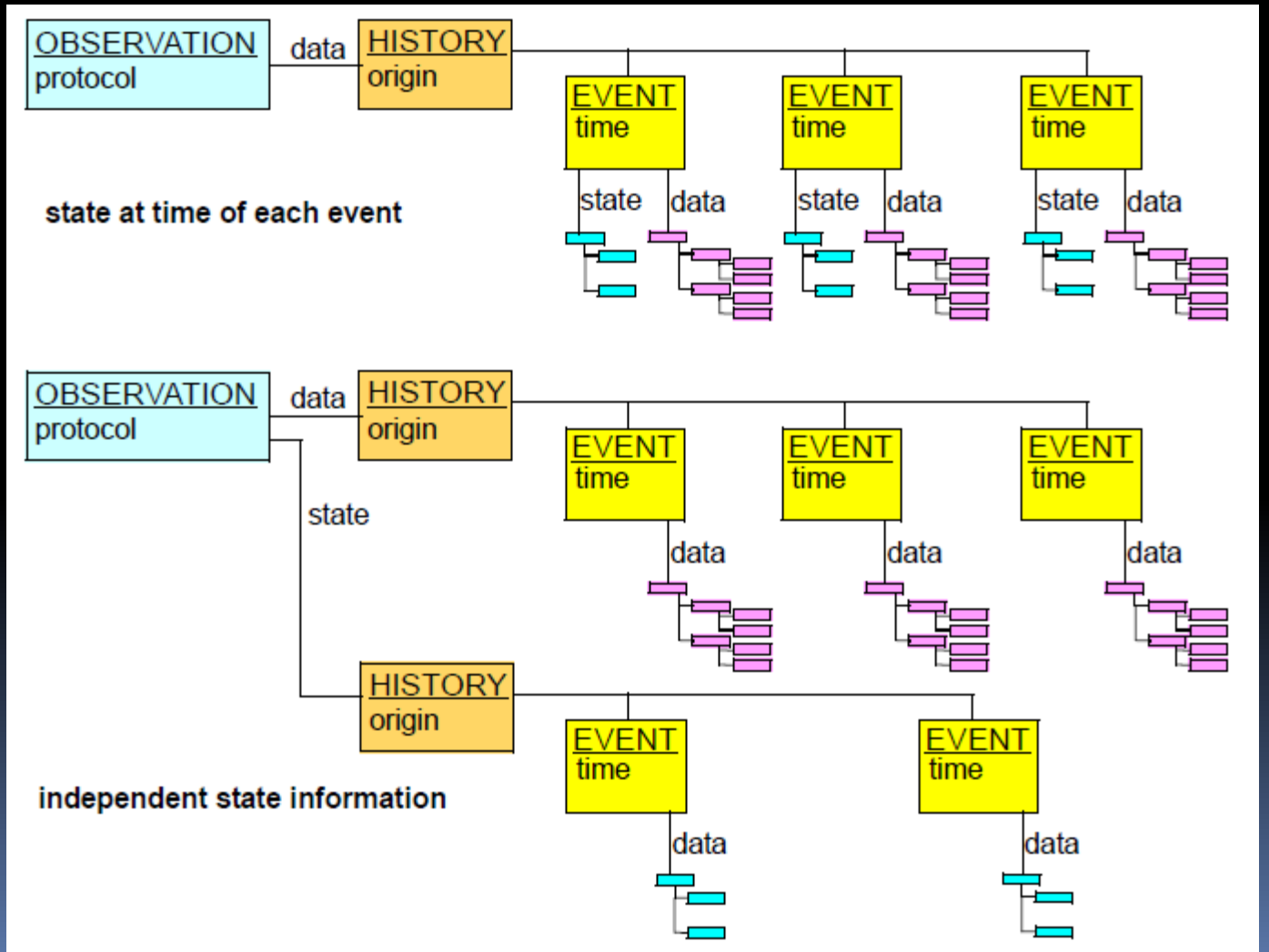


Why it's useful

- Supports per-sample or separate history for state information

- Good for OGTT →

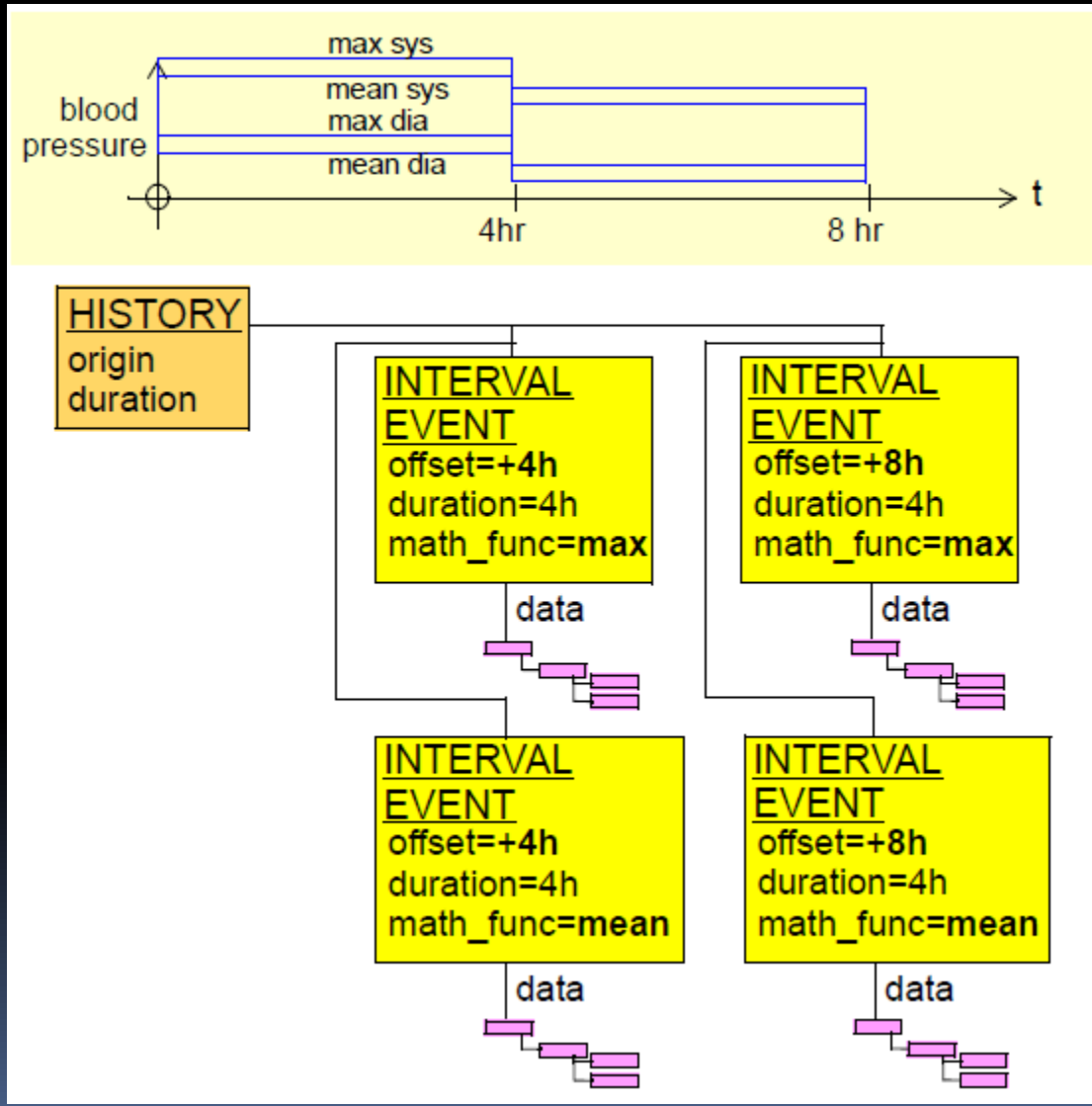
- Good for sports medicine →



openEHR

Why it's useful

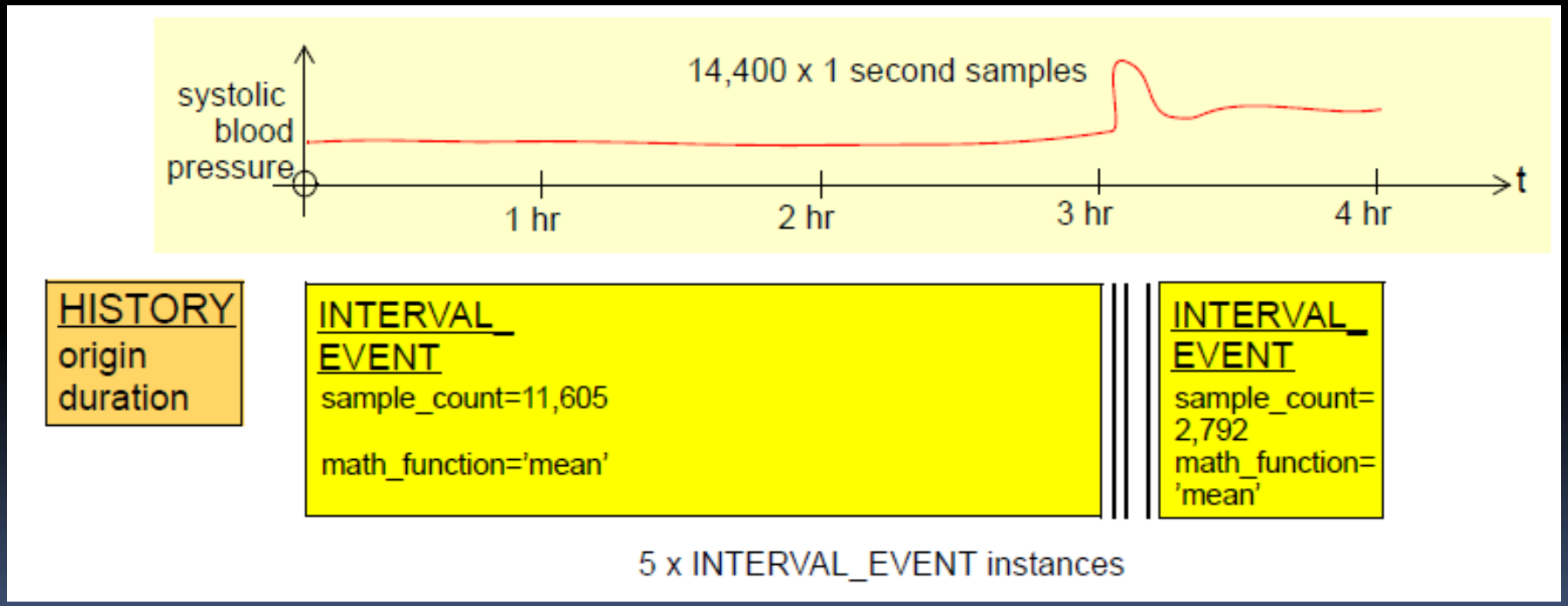
- Supports math functions like max, min, ave, diff



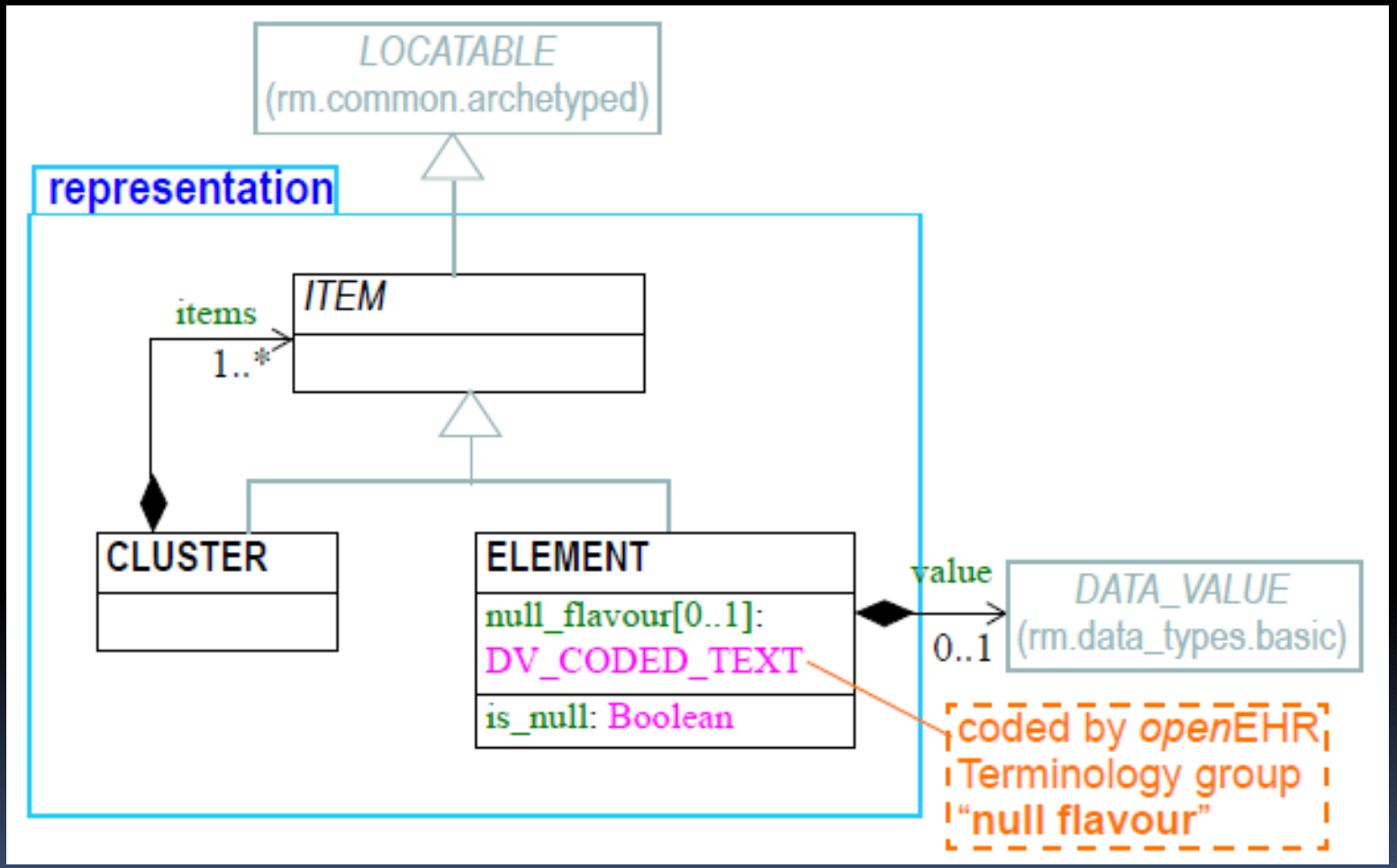
openEHR

Why it's useful

- Supports efficient compression of high-frequency device data

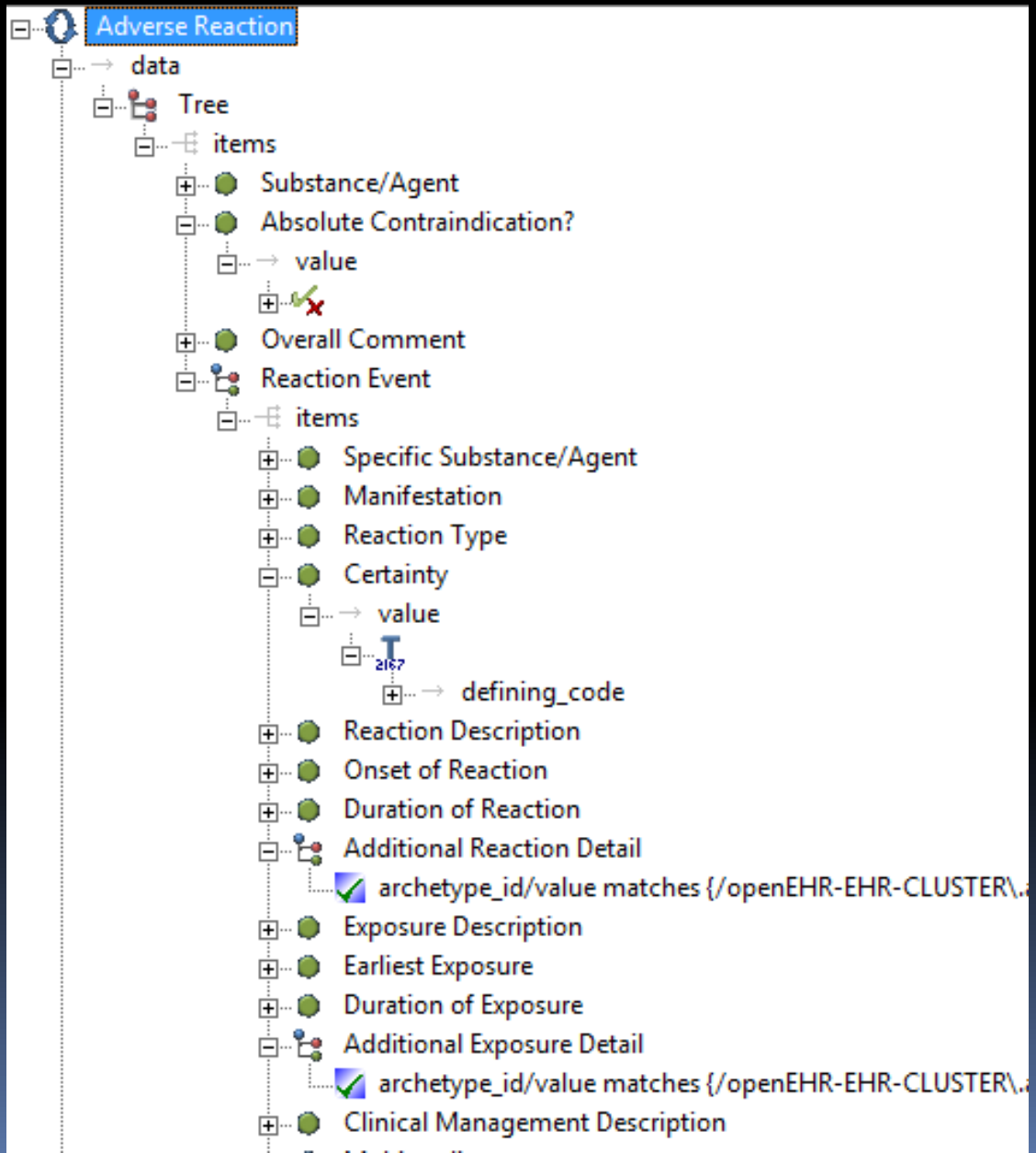


#3 – Basic tree structure



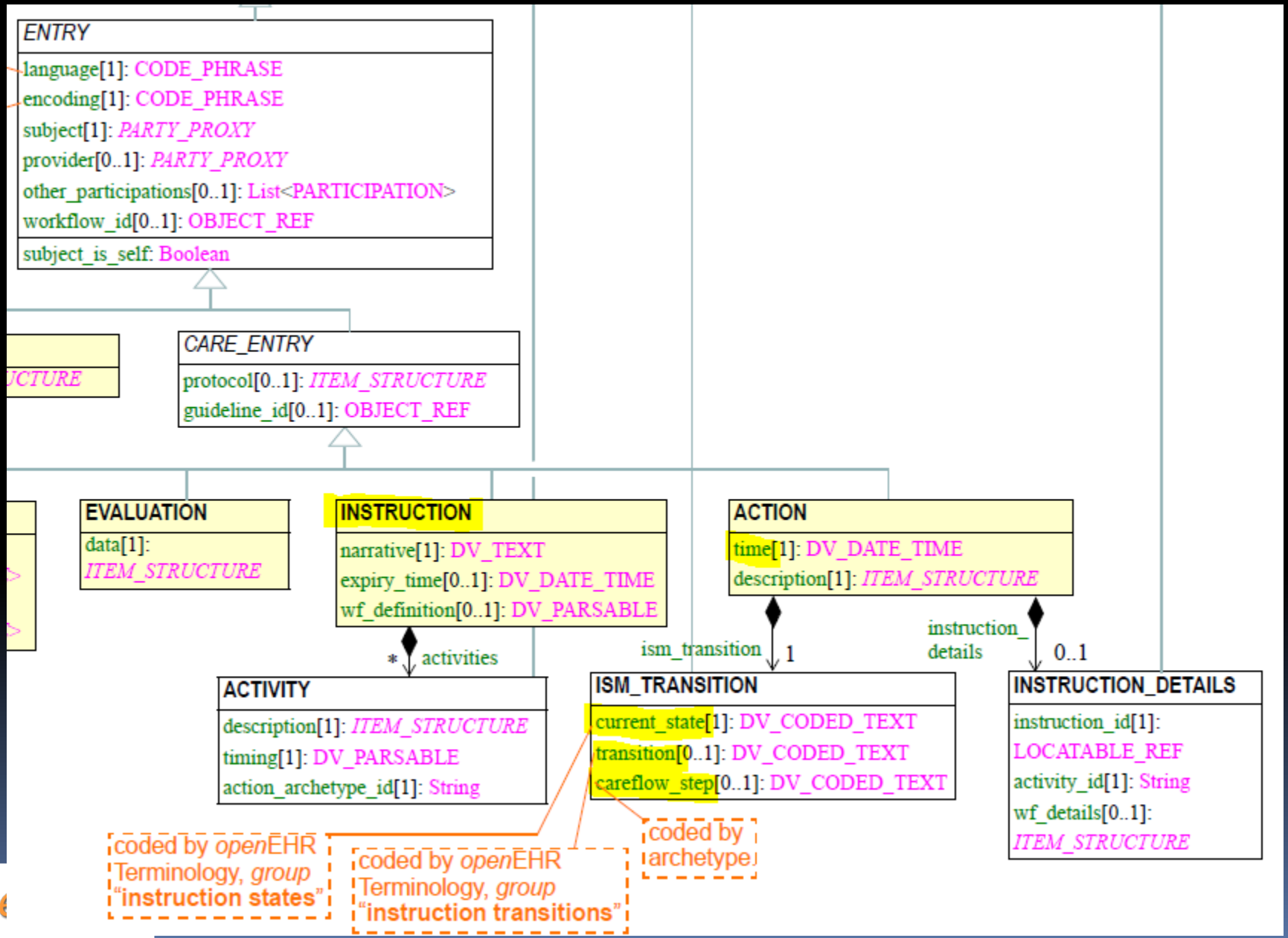
Why it's useful

- Every model has it ☺
- Real data are fractal

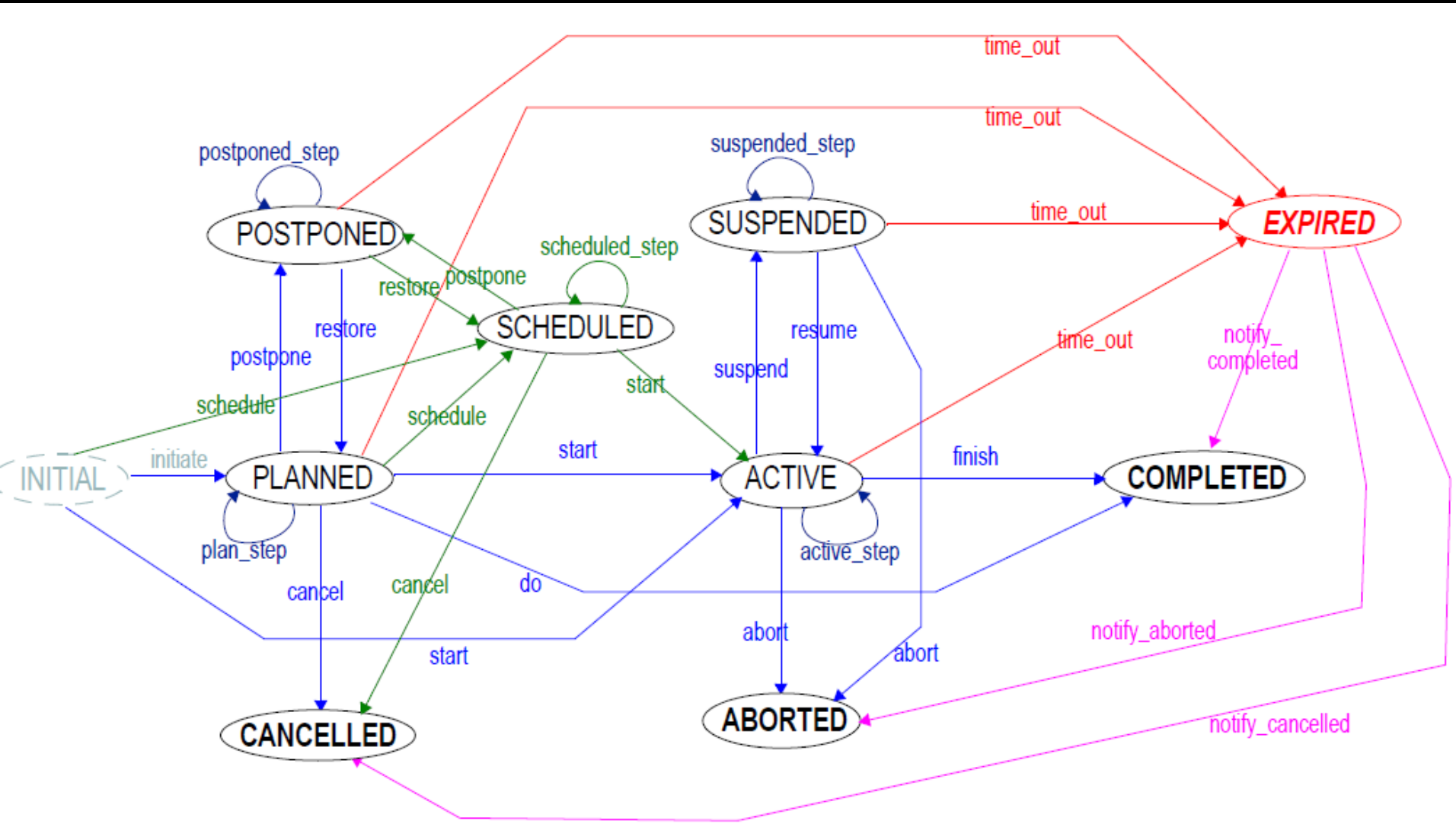


openEHR

#4 – Order state machine

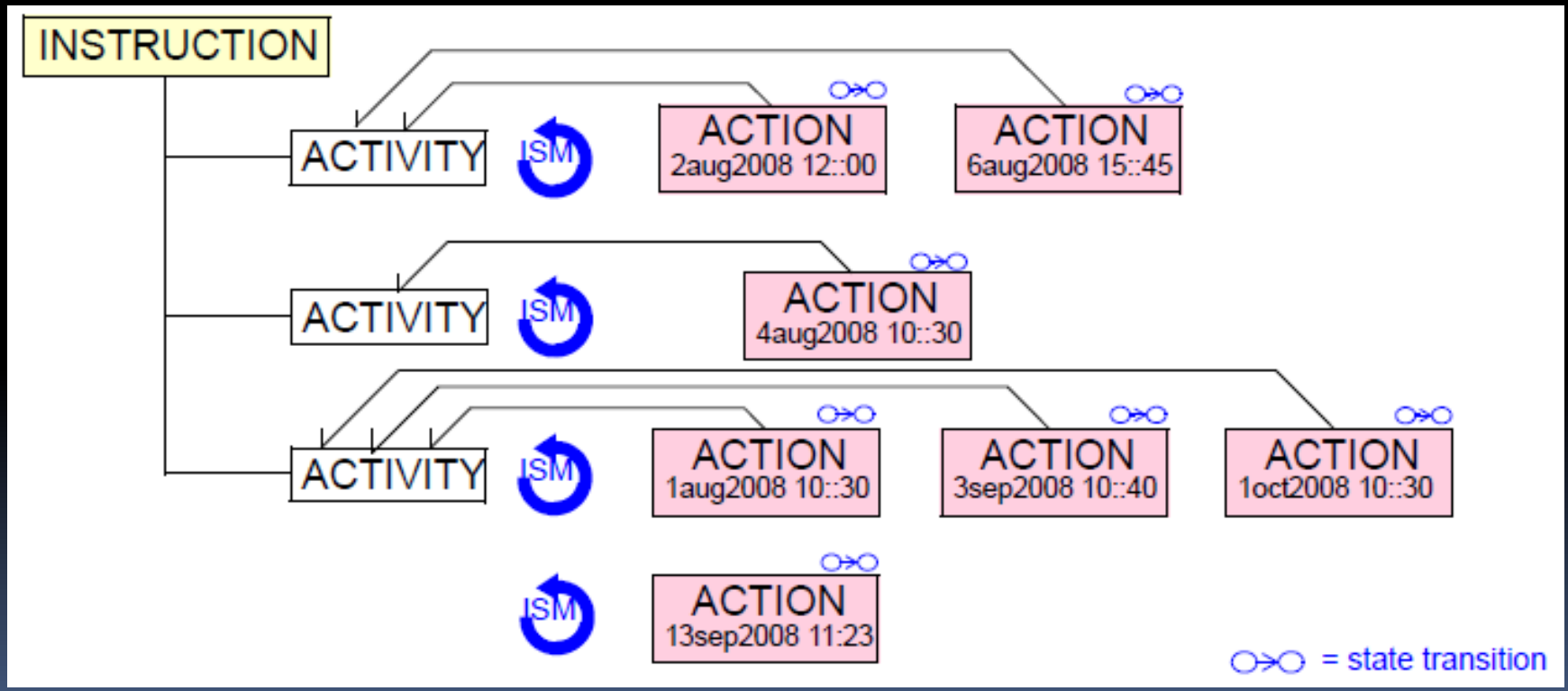


#4 – Order state machine



Why it's useful

- Supports Action tracking over time

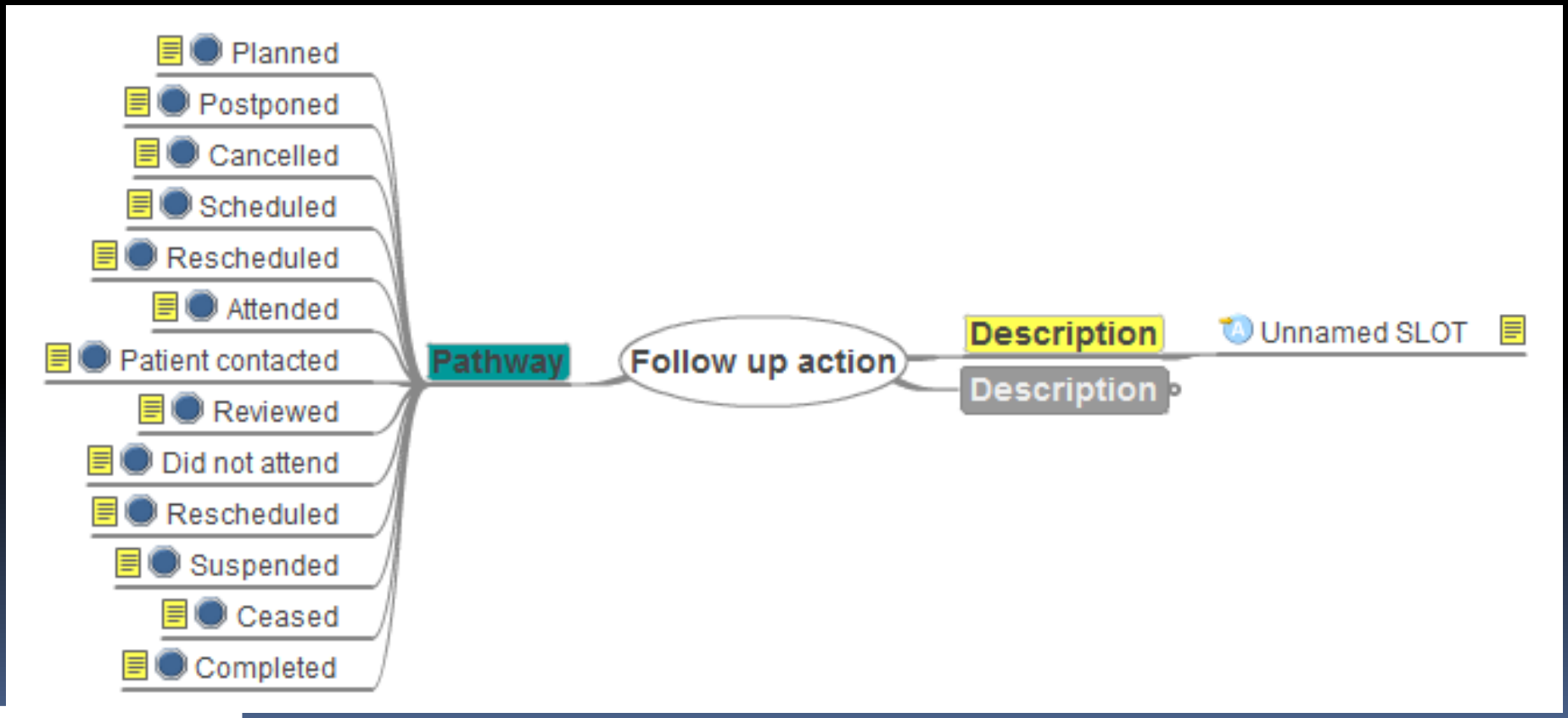


Why it's useful

- Supports 'careflow' steps that are specific to order type and maps them to standard states to support standard querying for:
  - What is active?
  - What is suspended?
  - What is booked?
  - What is stopped / cancelled / …?

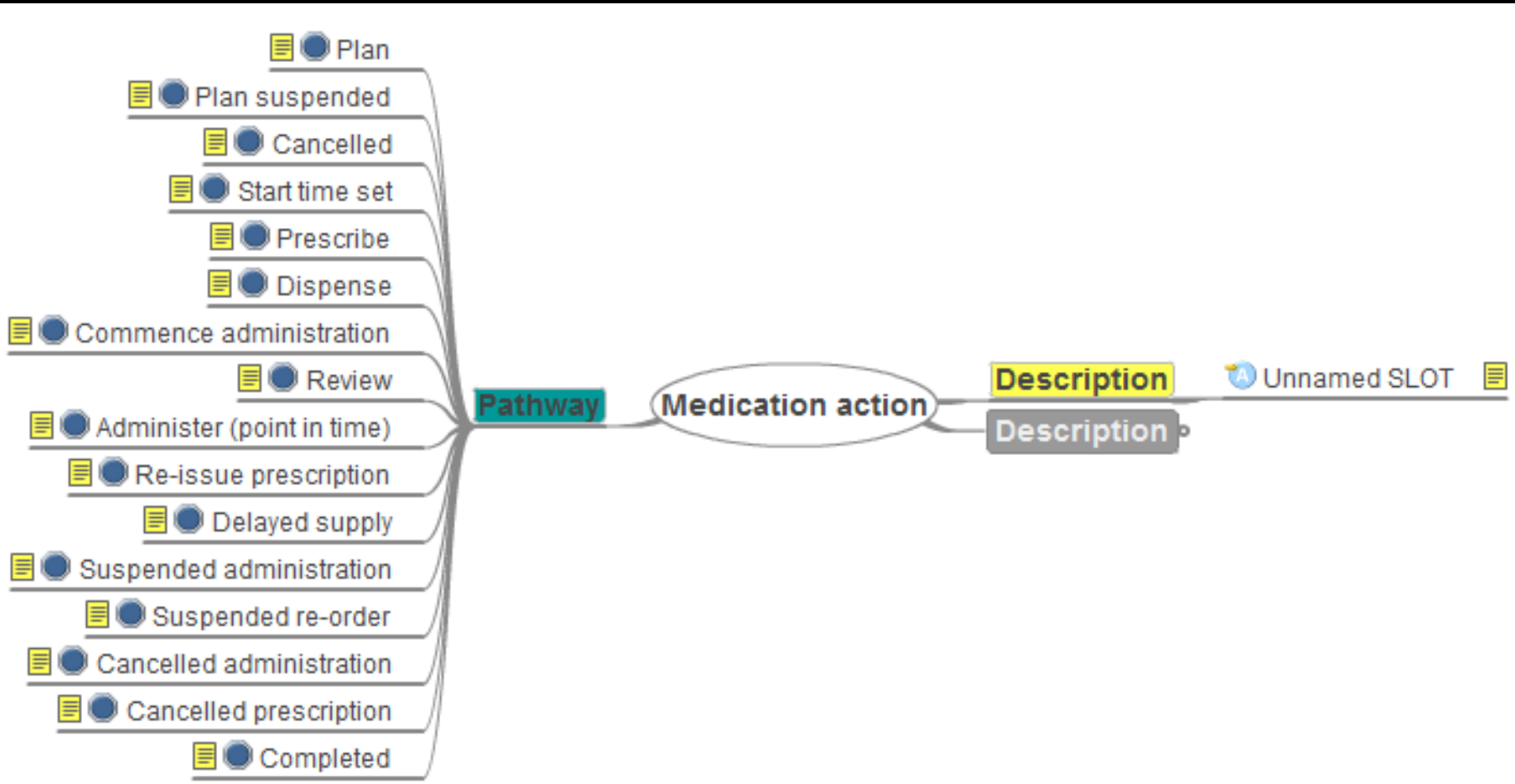
Why it's useful

- Careflow steps – follow-up action

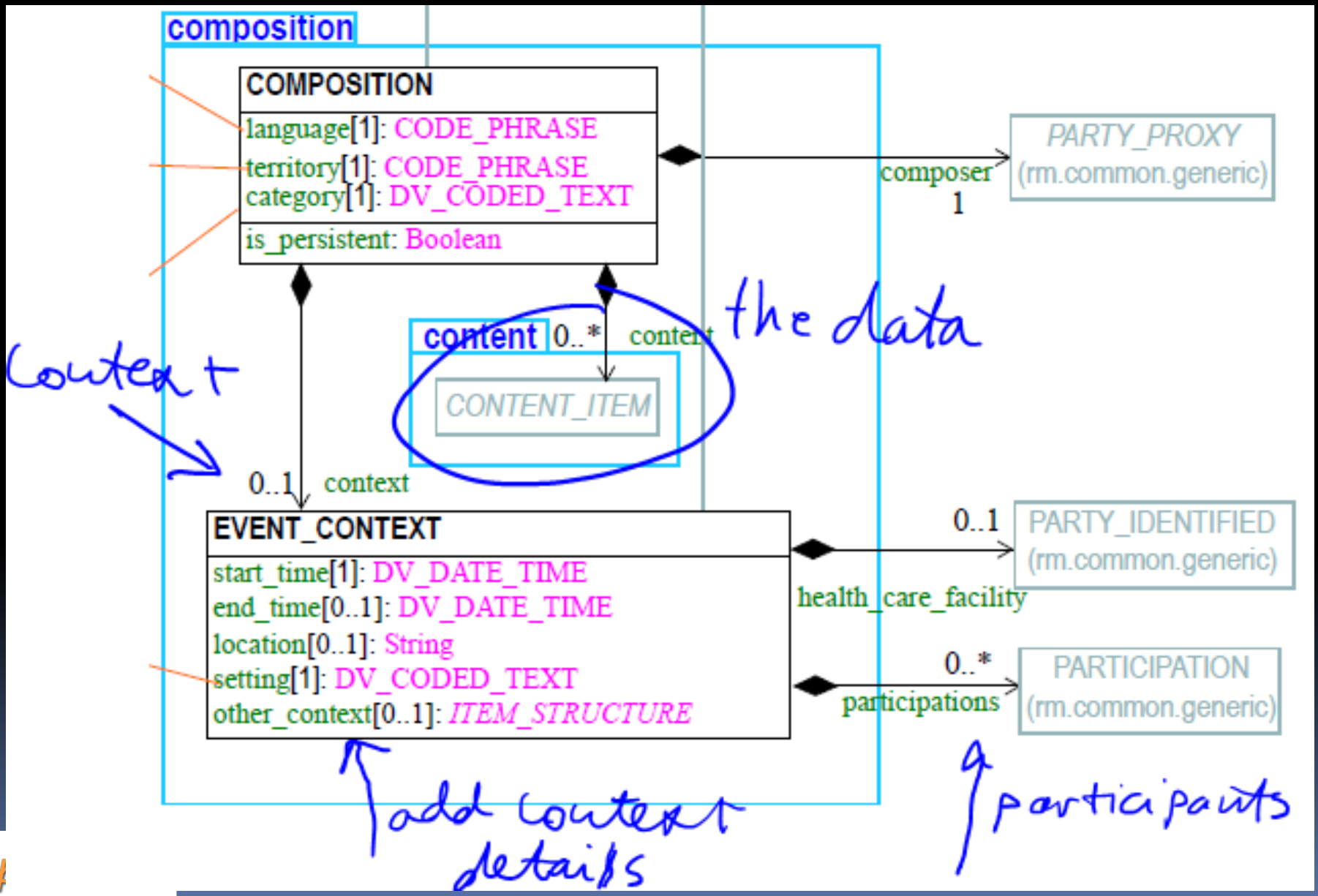


Why it's useful

- Careflow steps – medication action



#5 – Composition / document

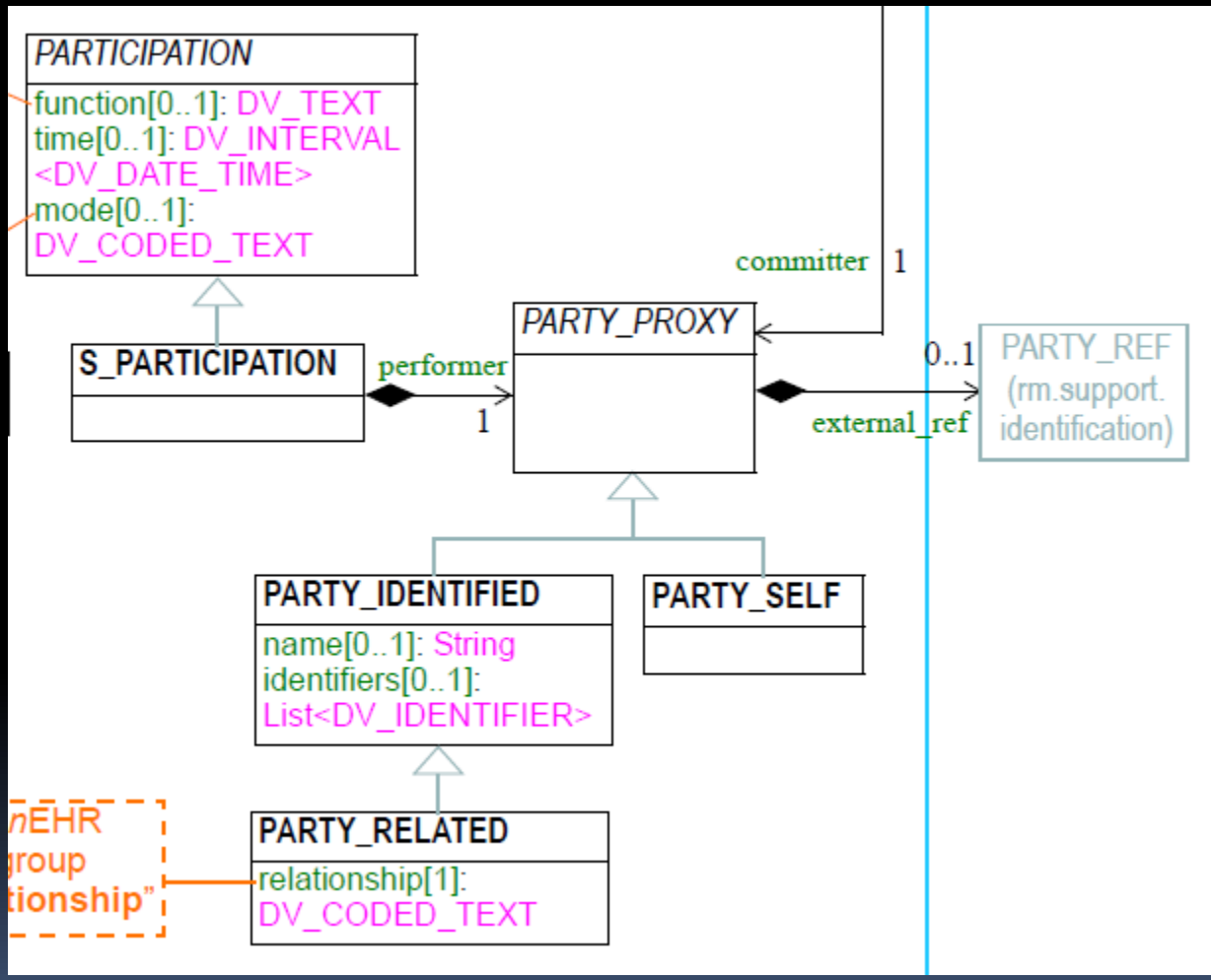


Why it's useful

- Defines the container in which data items for a given event are captured
- Coupled with versioning (supplied elsewhere in openEHR), defines a fully version-controlled health record document

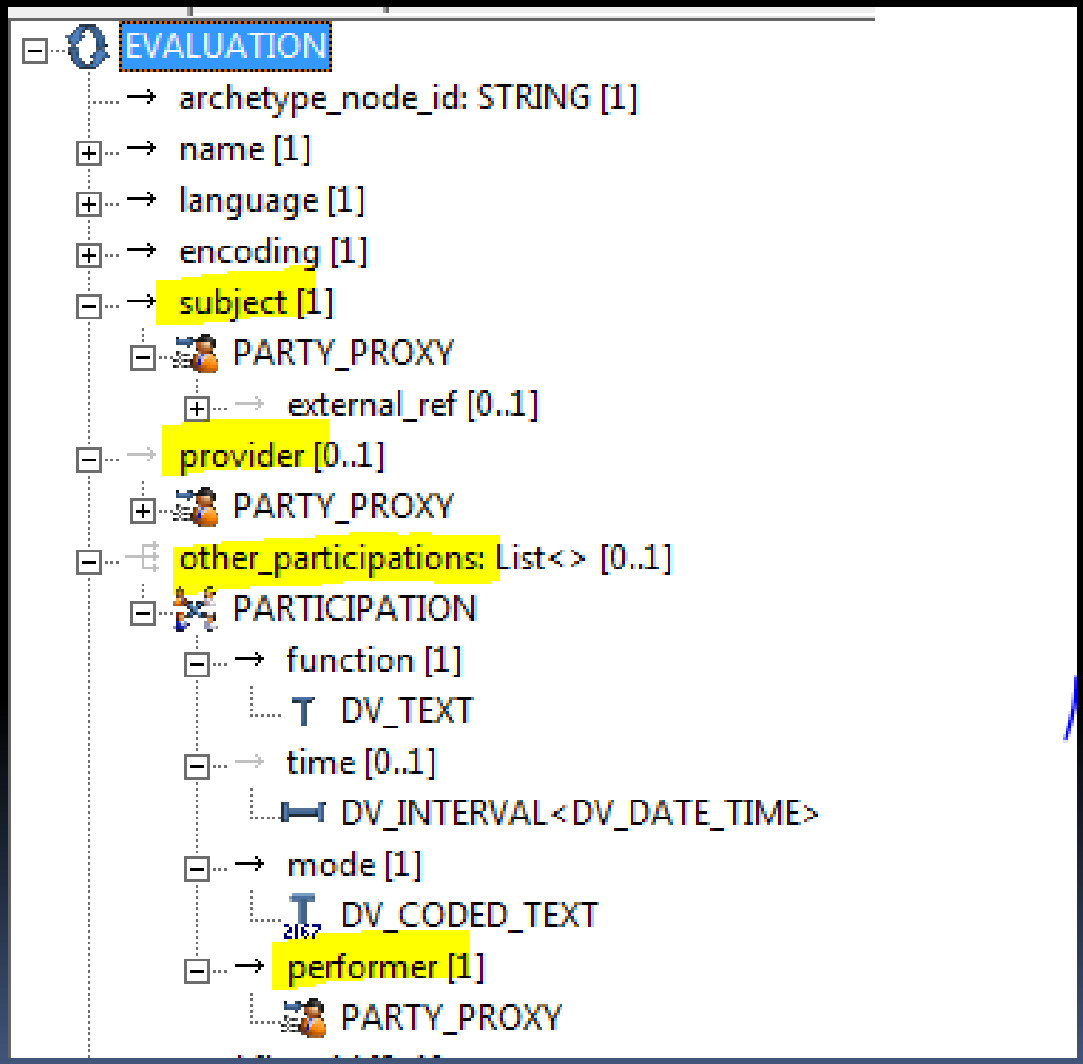
#6 – Participation

A standard model of participation is needed...



#6 – Participation

... That can be reused in the rest of the model

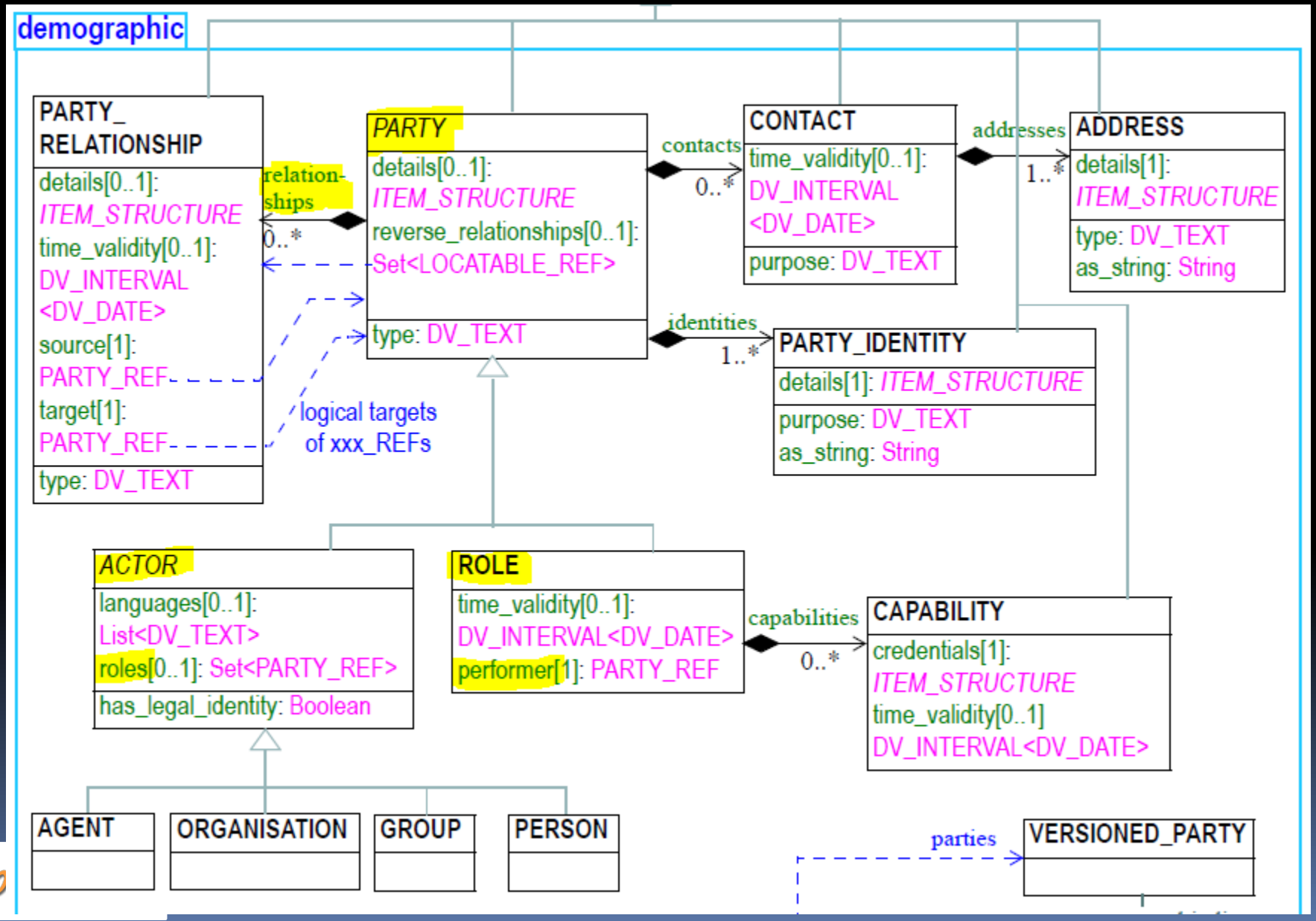


openEHR

Why it's useful

- (nearly) every action is performed by some agent, or 'participant'
- Some form of this pattern is found in HL7 RIM, CDA, EN13606, openEHR

#7 – Demographic relationships



Why it's useful

- Where demographic concepts like families, teams, employment etc are required
- Separates out actors, roles and posts.

openEHR

Conclusions

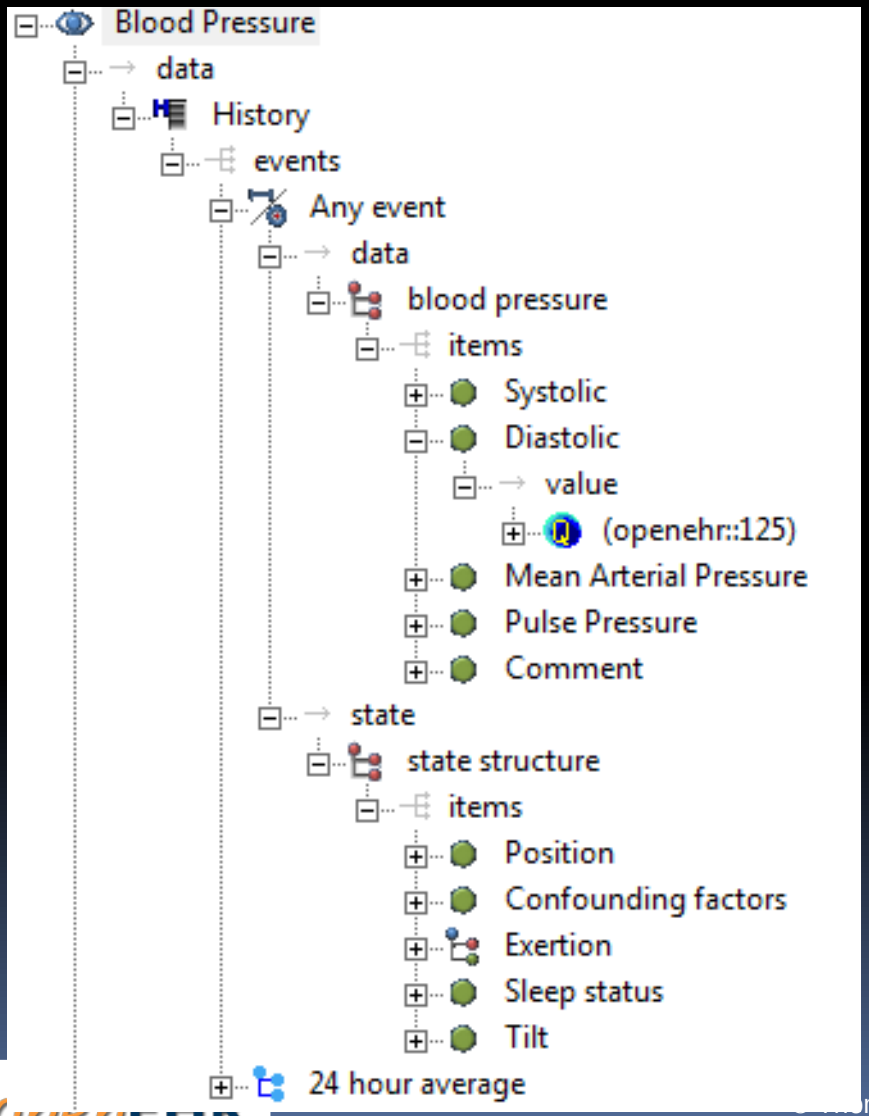
- openEHR's patterns are not the only ones, and are not perfect
  - In fact we have found a new variant of the 'tree' pattern that is needed for health data
  - Participation could be improved; see e.g. Singapore LIM
- But archetypes based on them provide a useful guide to their utility

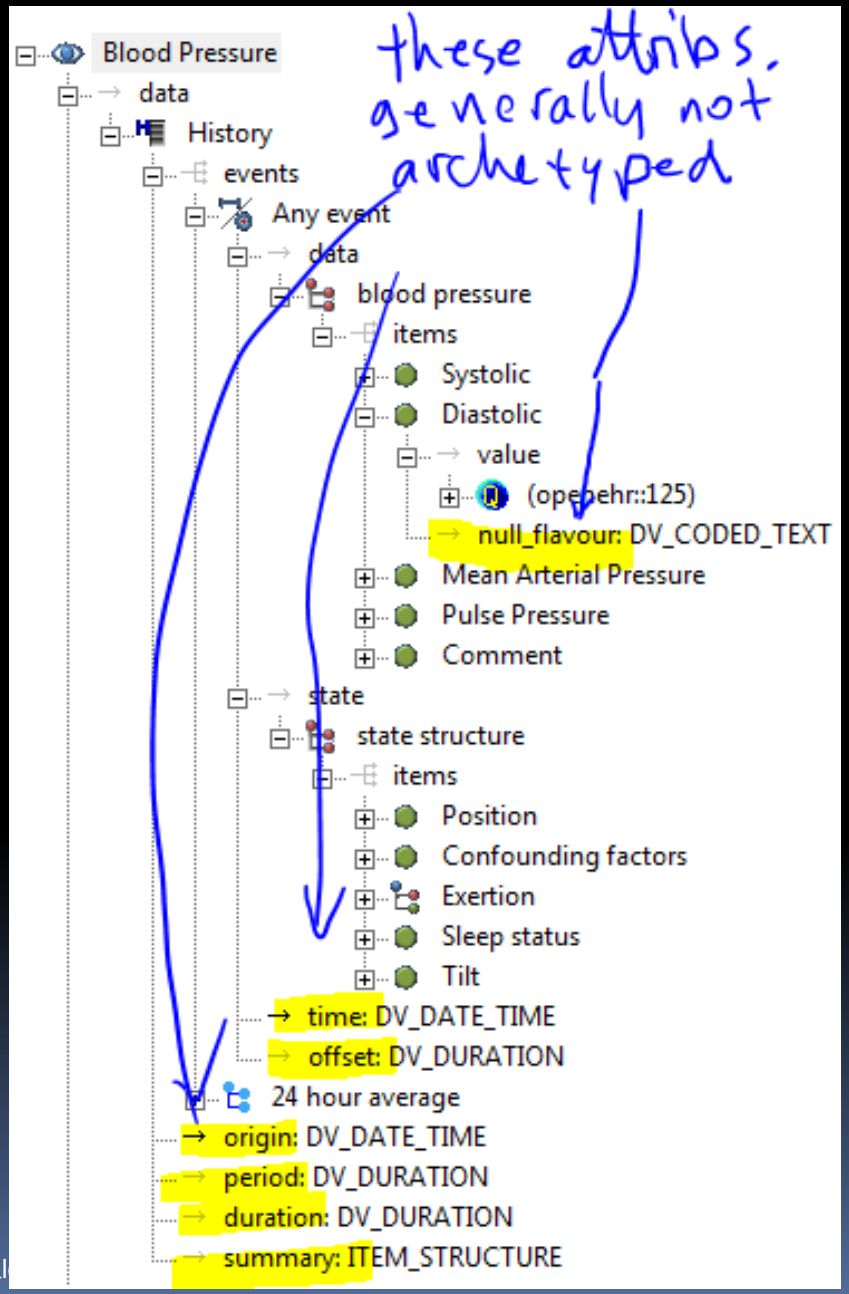
Conclusions

- Remember that for the purposes of DCM building, not all aspects of a 'reference model' as published are required
  - Because not all elements need to be archetyped

openEHR

Conclusions





My recommendations

- A. The key is to define an RM consisting of the key patterns that need to be archetyped / constrained in DCMs, leaving out details of messaging etc
  1. Some of openEHR's RM is potentially directly usable for this purpose, due to the archetype history
  2. Some pieces of other models also useful – see e.g. Singapore LIM, various CDA patterns etc

My recommendations

- B. Don't start 'building' this DCM-RM as a separate exercise; instead, define some key archetypes to be built and use these to determine what bits of the RM are needed
- C. Convertability of DCMs based on the DCM-RM to real world RMs has to be considered.

Resources

- openEHR ADL Workbench
- openEHR specifications

openEHR