The simplification in integration architecture that 4D supports

Ian Bailey
telicent

GCHQ UCL Brunel University London UNIVERSITY OF Southampton THE UNIVERSITY OF WARWICK

UNIVERSITY OF CAMBRIDGE

cdbb
Centre for Digital Built Britain

Ian Bailey BEng, PhD, CEng, FIET, FIMechE, FBCS

- Specialist in complex data integration problems
  - Working with international blue-chip companies and governments to solve difficult systems integration and migration problems
  - This has involved decades of work implementing 4D ontologies – initially in the oil & gas sector, later in Govt
- Subject matter expert in Enterprise Architecture
  - UK Govt advisor in EA, representing UK in standards bodies and international programmes
  - Advisor to EU
  - Tech lead on international EA programmes
- Data standards
  - Lead on international IDEAS 4D ontology
  - Editor of two ISO data standards
  - Tech lead on UK MODAF standard, and on NATO AF v4
  - Contributor to UML 2, SysML and UAF
  - Now tech lead on UK Govt IES v4 – a 4D data exchange standard
- Experienced start-up CTO
  - cyber (Ascot Barclay)
  - machine-learning (illumr)
  - data platforms (telicent)

Implementing 4D ontologies

Work started on implementing these kinds of ontologies back in the 90s…with varying success. Initial work was all focussed on RDBMS.

Shell Single-Table Model (early 90s)

- Not 4D, but laid the foundations for a lot of the later thinking around how to implement ontologies
- Approach is very similar to Martin Fowler's Reusable Object Models approach, though this predates his work by a decade.
- Data and data model all in the same database. All I the same table, in fact. Each record is a class, an instance or a property.
- Classes can be instances of classes (higher order)
- Table is bootstrapped by populating it with fundamental classes – class, relationship, type-instance, sub-supertype
- To anyone who knows RDF Schema, this will be very familiar
- Performance was OK with smaller datasets, but began to drop off as data grew – indexing was key to performance, but only took you so far
- Tech was eventually spun off into a start-up – Kalido – who were able to solve a lot of the performance issues
- Key players were Chris Angus, Andy Hayler, Bruce Ottman, and Matthew West

PIPPIN (mid 90s)

- EU funded programme to accelerate uptake of process plant standards – ISO10303-221 and ISO15926.
- A major part of the programme was about implementation of the standards using off-the-shelf technology
- Process companies were wedded to Oracle, as were most big organisations at the time so main focus was on implementing 4D in relational databases

- Lessons learned:
- Performance is tricky in RDBMS
- Conversion to 4D is relatively easy
- Conversion back to legacy formats is a real pain
- Reference data libraries can hard to maintain– a lot of planning is needed before starting
- Traditional data modelling languages (in this case ISO10303-11 EXPRESS) are poorly suited to ontology development
- Engineers (real engineers) get 4D pretty quickly
- Programmers struggle to get 4D
- Experienced data modellers are doomed never to get 4D

Shearwater (late 90s)

- A major off-shore oil and gas rig – Shell
- Approach taken was highly pragmatic, led by the lessons learned in PIPPIN and other R&D programmes
- Use of non-RDBMS storage technology (Quillion)
- Minimalist approach to reference data
- Mapping was one-way – the target was Quillion, and that's where the data stayed. A "data warehouse" approach

Image © Shell

Downstream One

- At the time, this was the biggest IT programme in the world
- The main thrust was a roll-out of SAP to all the Shell operating companies throughout the world
- But…
- Shell had traditionally allowed its operating companies to exercise a lot of local control
- This had resulted in wildly varying data standards and data quality across the enterprise
- The purpose of the Downstream One model was to reconcile the various data models and catalogues in use world-wide
- Decision was to use an existing data standard – ISO15926 and expand on that wherever possible
- Probably the first time the BORO methodology had been applied to oil & gas work

IDEAS (mid 2000s)

- A programme to align the systems architecture standards used by various national defence ministries – Australia, Canada, France, Sweden, UK, USA & NATO
- Aligning the underlying data models (meta-models) was proving very difficult.
- Decision was taken to try the BORO method as a way to get to the root of what was common between the models
- The result was a 4D ontology
- Modelled in UML – still not ideal, but better than EXPRESS for this work
- Implemented variously in RDF and relational databases
- Became the basis of US Dept of Defense DoDAF v2 standard, UK's MODAF, and NATO AF v4, then later the domain meta-model of the OMG UAF standard.
- Lessons learned
- 4D is difficult – teams not only need training, but they need experience. A good deal of brain re-wiring is required
- A little knowledge is a dangerous thing
- A data model in OWL is still a data model
- The BORO method ruthlessly uncovers problems in data models – people *really* don't like to be told their baby is ugly

UNIVERSITY OF CAMBRIDGE

National Defence

Australian Government Department of Defence

Ministry of Defence

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
MINISTÈRE DE LA DÉFENSE

NATO OTAN

cdbb
Centre for Digital Built Britain

IES – UK Govt Information Exchange Standard (2020)

- A data exchange standard developed between Police, MOD, Home Office & Intelligence Community
- Had grown in functionality over versions – initially just about sharing entity nominals and selectors, but had grown to include relationships and events
- By v3 it had its own meta-model and XML Schema encodings – repeating a lot of the W3C linked data stack but in a non-standard way
- Decision was made to develop an RDF Schema version of the standard

- This was also an opportunity to re-engineer into a 4D ontology – version 4 of IES
- Lessons learned:
  - RDF Schema is pretty good for modelling 4D ontologies from scratch
  - IES already had a lot of 4D ideas in it, but not all treated in a consistent way – BORO helped with this
  - Triplestores offer a great way to implement these ontologies
  - ...but are less useful for more transactional applications
  - ...hence we still need to implement this data in other storage paradigms – e.g. big-table, document, RDBMS, time-series, etc. for different applications

How does 4D make integration easier ?

Lego™ not Airfix™ *

IES is not like a traditional data model. It is made up of a few re-usable components that you put together in different ways to make your model. It's like Lego™ - the model might not exactly as you expected, but the parts move, and you can always change and extend it easily. You can also change small parts without breaking the rest of the model…

…I think that's enough metaphor stretching for one day…

Image wikimedia commons, author:Tangopaso

Image creative commons, NASA, Maria Werries

*Airfix is a UK model kit company – similar to Revell™, Heller™ or Tamiya™.

How does 4D make integration easier ?

Space & Time

IES is a 4D model. Any instance of an IES Element will be something that occupies space and time. The 4D approach allows us to say things about temporal chunks (states) of these Elements. The approach goes further though – extent is the criterion for identity – if two things occupy precisely the same space at the same time, they are the SAME THING. Understanding this is the key to understanding IES.

In the example above, Fred appears to have three different masses. However, each mass is associated with a different state of Fred – i.e. a different point in his life. We've also introduced yet another notation here – the space-time diagram.

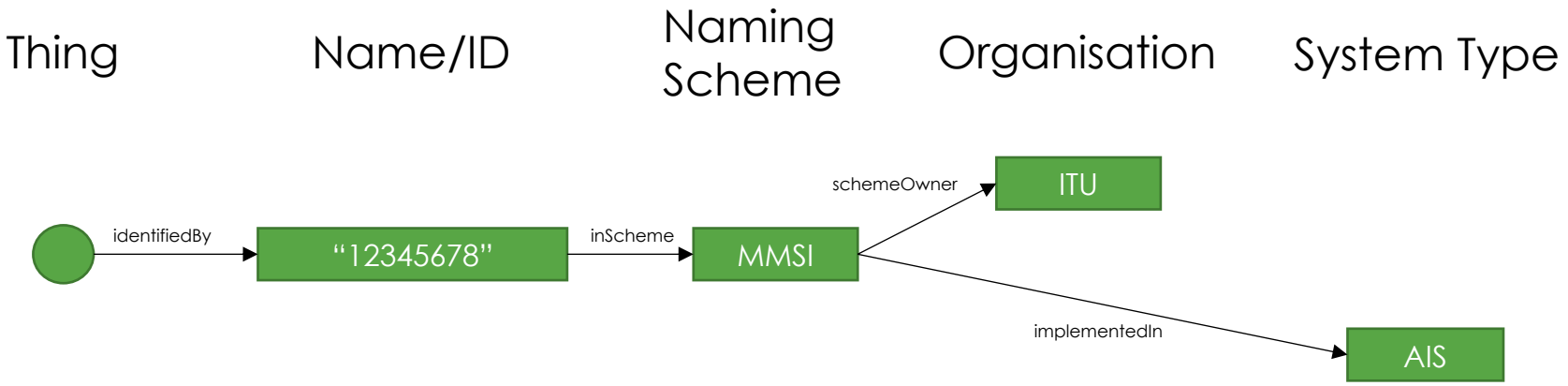
For more background on the 4D approach (formally, this is b-series four-dimensionalism), refer to:
"How Things Persist", Katherine Hawley
"Developing High Quality Data Models", Matthew West
"Business Objects: Re-engineering for Re-use", Chris Partridge

UNIVERSITY OF CAMBRIDGE

cdbb
Centre for Digital Built Britain

How does 4D make integration easier ?

[A: The BORO naming pattern]



How does 4D make integration easier ?

[A: Add new classes as you need them]

This is also true in a lot of other ontologies, but it's an important distinction as compared with traditional data models.

If you can extend your model this way, it's far easier to deal with new, unexpected data or changing business requirements.

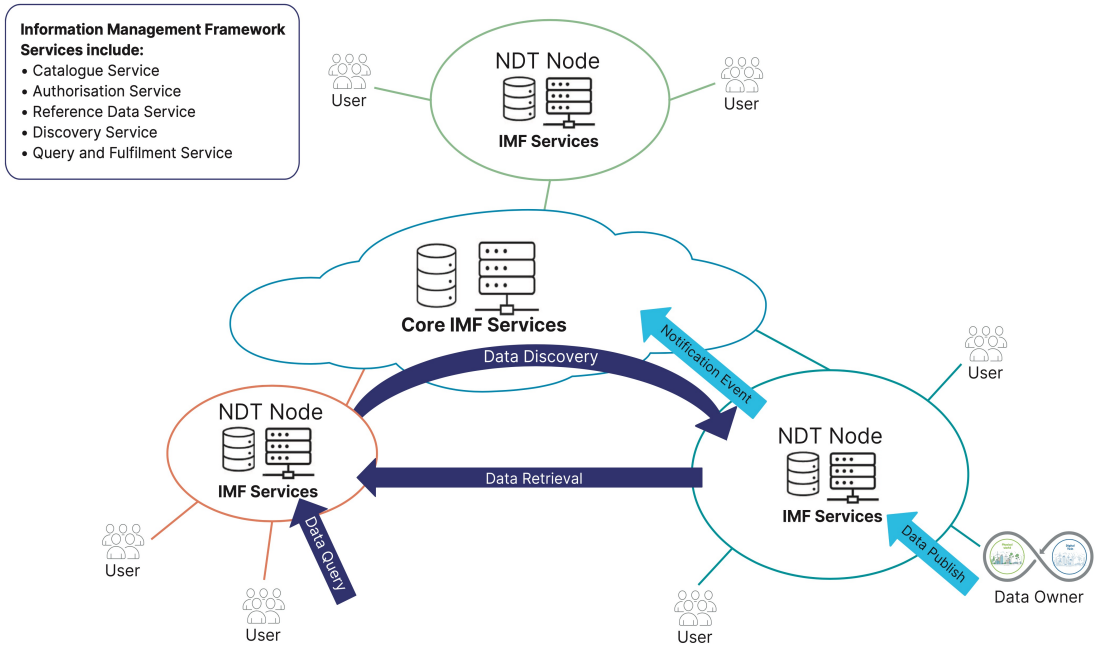
The model/data boundary was always artificial and arbitrary – usually at the whim of a data modeller or business analyst rather than anything to do with the facts or the requirement.

How does 4D make integration easier ?

[Summary]

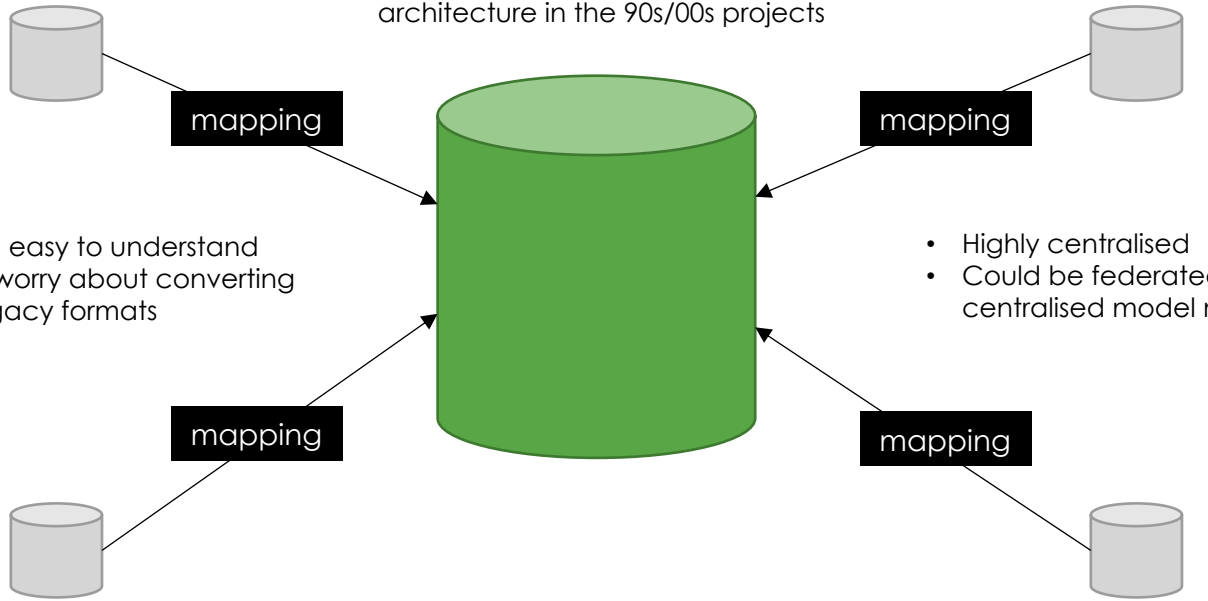
Increased Flexibility	Increased Consistency	Increased Quality	Increased Diversity	Increased Precision
Extensible	Single model for time	Exposes holes in data	Integrates multiple sources	Temporal model
Multiple identifiers	Criteria of identity	Exposes errors in data	Covers multiple domains	Specificity of vagueness
Additive approach (lego)	Simple repeatable patterns	Exposes inconsistencies	(using the same patterns)	Fine-grain

NDT Logical Architecture



Data Warehouse Model

All data sources contribute to a central store – this was the predominant architecture in the 90s/00s projects

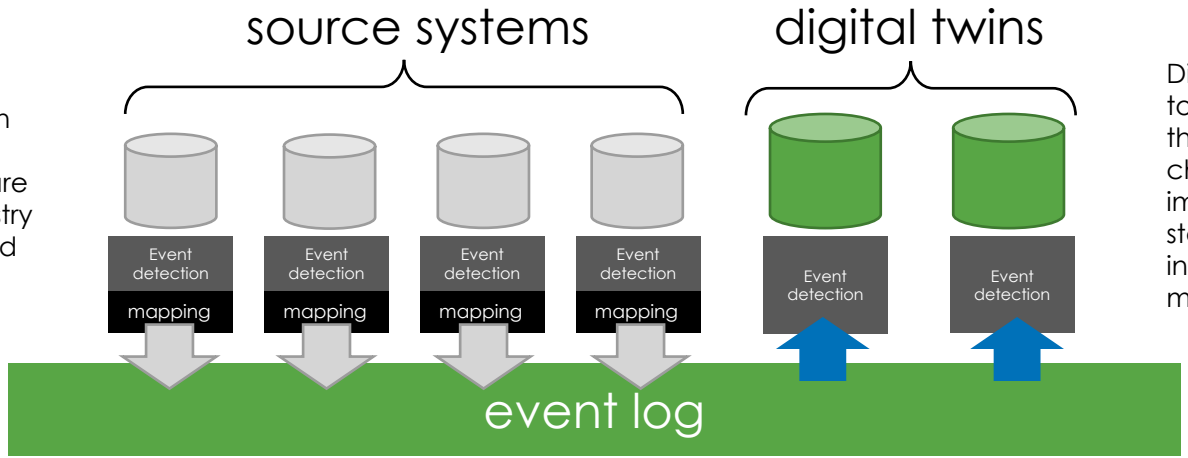


- Simple to build, easy to understand
- Don't have to worry about converting back to the legacy formats

- Highly centralised
- Could be federated, but fits a centralised model much better

Event Sourcing Model

Changes to data in source systems are detected. Deltas are converted to industry data model pushed to the log

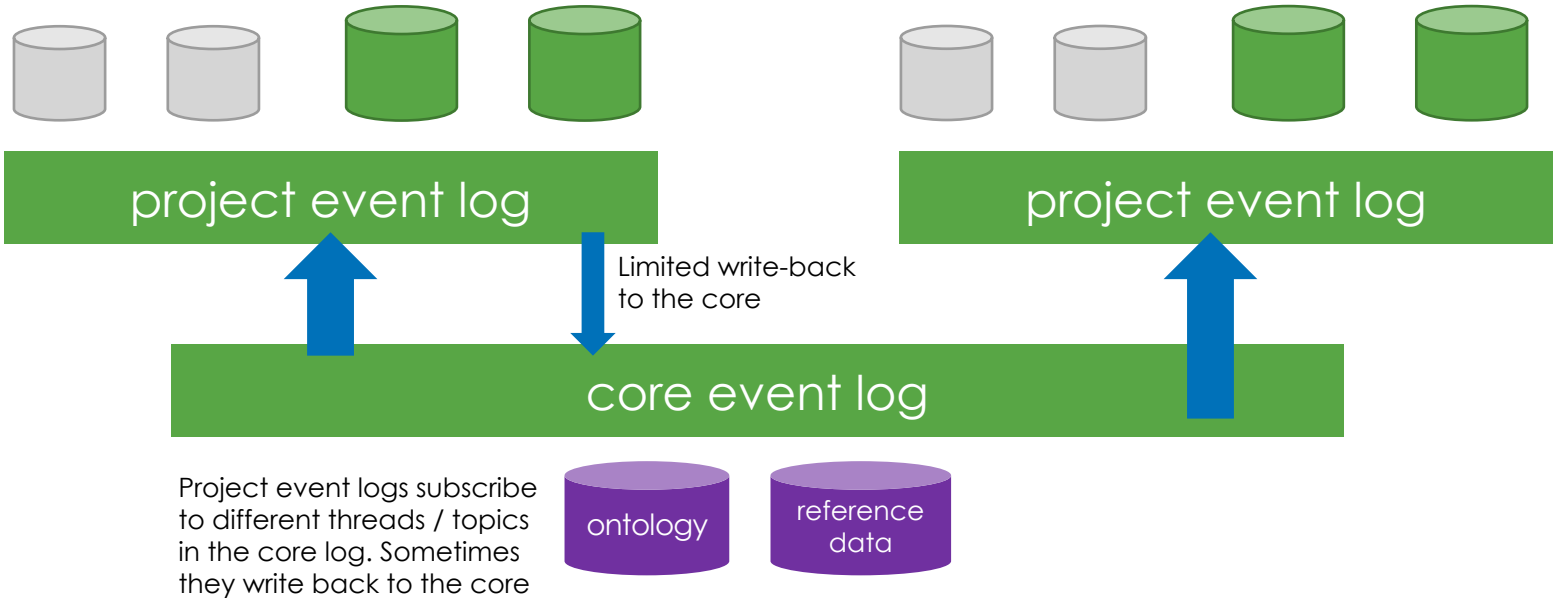


Digital twins stay up-to-date by watching the event log for changes. They implement the standard data model internally so require no mapping

The log is a sequential record of all changes to the data from all the supplying systems. New consumers (e.g. digital twins) can subscribe to the log and read in the data they need.

The log is divided into streams (or "topics" in Apache Kafka) which allows consumers to be selective about what they read.

“Federated” Event Sourcing



Summary

- Decades of experience of implementing 4D in the community
  - Spanning finance, oil & gas, defence, policing and construction
- A lot of lessons learned
  - How to work with ontologies in a pragmatic way
  - How to get the best out of 4D
  - How to manage reference data
- Enterprise Architecture – big picture
  - Aiming for a federated approach with some centralised functionality
  - Event-sourcing approach looks useful, and fits well with W3C RDF stack
- Looking forward
  - There will soon be an ontology to work with and test
  - We are building demonstrators based on existing 4D standards in the meantime

Questions?

Contact

Ian Bailey, CTO of Telicent
https://telicent.io
https://www.linkedin.com/in/ianbailey/

UNIVERSITY OF CAMBRIDGE

cdbb
Centre for Digital Built Britain

So, what did we learn from all of this ?

- Formal 4D ontologies are incredibly expressive – models that are closer to reality are better models
- They are flexible, and can adapt in-situ to changing business requirements
- The improvement in data quality is clear, but it's hard work to get there
- The W3C linked data stack (RDF & Triplestores) seems to work well for 4D
- For some applications though, we need to use different database tech

- Sometimes, the source data can't meet the base quality threshold to be useful in 4D – there will be a cost/benefit trade-off
- Formal ontology is hard to do right. Formal 4D ontology is even harder – practitioners need to serve their apprenticeships
- Converting legacy data to 4D is relatively easy, but the reverse is not true
- Reference data libraries can a pain to create and maintain – it will always be a cost/benefit trade-off

When to use 4D

- When the domain you're interested in changes or moves over time
  - …which you could argue is most domains, but in particular:
  - Built estate, process plant, police investigations, climate impact studies, logistics, etc.
- When the effort required to clean and restructure the data is justified
  - Data that is retained for long periods
  - Data pertaining to high value projects
  - Data crucial to saving lives

…and when not

- When the data is going to be thrown away after it's been looked at
  - Highly transactional data – e.g. individual purchases, sensor readings, etc.
- When the data quality is terrible, and the margins are tight
  - You may still want to do the work, if only to show how bad the data is
  - In many cases though, the cost of quality improvement doesn't outweigh the savings