Some Industrial Experiences in the Development and Use of Ontologies

Matthew West

Principal Consultant, Shell Information Technology International, Shell Centre, London, SE1 7NA, United Kingdom
matthew.west@shell.com

Abstract. Ontologies have been part of developing information systems in Shell for some twenty years, taking the form of data models and reference data used within information systems. A problem in reusing or integrating systems is the context that they assume, which may not be valid beyond the scope of an implementation. Lessons learnt include trying to ensure that the context is explicit, and that what are really local rules in a global context are not defined as global rules in a local context. These lessons have been applied in the development of International Standards to provide an architecture for integration and a data model that includes both a foundation ontology that has been developed on a well defined and consistent basis, and provides a framework for extension of the ontology through reference data.

Introduction

Ontologies have been part of the development of information systems in Shell for at least twenty years, taking the form of data models and reference data used within information systems. More recently the need to share and exchange information with customers and suppliers has become evident, sometimes on a very large scale as with the acquisition of the design data for an oilrig or refinery from a design contractor.

This paper presents some experiences and learnings that have arisen from the development of data models and reference data aimed at achieving integration across a business or between businesses in and around Shell. A number of data models have been developed along this journey, culminating for the author in the development of ISO 15926-2 [1]. This paper presents some of the learnings that drove the design decisions embodied in this data model.

The work presented here is not solely that of the author, but of the data management team in Shell, the EPISTLE consortium, and members of ISO TC184/SC4, all of which the author has been a leading technical contributor to.

Experiences and Learnings in Shell

Historically Shell has been a diverse group consisting of more than 100 different companies, operating in more than half the countries of the world. Shell has been a user of Information Technology to support the business since the 1960s. However, with the introduction of commercial Relational Database Management Systems (RDBMS) in the 1980s there was an explosion of development of information systems to support the operation and management of the business. Since Shell was a devolved company, these developments happened locally rather than centrally.

It was not long before it was noticed that a system to perform the same function was being developed in many different places. This was clearly a waste, and so work began to see if some of these systems could be reused in other parts of Shell rather than being redeveloped. However, attempts at reuse usually failed. The reasons for this were investigated to find:

- why different Group companies in the same business had developed different data models and systems in the same area,
- why attempts to share these systems failed, and
- how to improve data models so that sharing became easier.

Two main causes of problems were found in data models:

1. Constraints in the data structure – sometimes deliberate, sometimes inadvertent – prevented data from being held.
2. Only a current snapshot was held, change and history could not be managed.
3. Classes that were essentially the same or overlapped were not recognized as such. So customers and suppliers would be managed separately, without recognising that a customer could also be a supplier.

One way of considering these shortcomings was to see that the local context had been built into the system and meant that the system could not be shared. On the other hand, if the context could be discovered and made explicit, then integration could become possible.

The approach to tackling this was to discover what traps to look out for in data models, and how to improve your data model to make the data model more reusable. These were documented in [2].

An Initial Paradigm

Looking at the traps that could restrict the reusability of data models also lead to the development of principles for developing high quality data models – that did not fall into the traps we found [3]. This in turn lead to a Generic Entity Framework that, for Shell, represented an initial attempt at what might now be described as an upper ontology [4],[5].

In principle there are an infinite number of ways that the world can be modelled. This is evidenced in part by the experience Shell had in the development of its early

database systems, where there were found many different data models and database designs for the same or similar requirements – none developed independently were found to be the same. The initial Shell Generic Framework was developed on an entirely intuitive basis, without any reference to, or knowledge of, any relevant work in ontology. The response, particularly to wanting to manage history consistently, was to develop a data model where all objects (classes, relationships, and individuals) were considered to exist for a lifetime. This gave a powerful and consistent approach to managing change.

An example of applying the principles and framework

This example is taken from three different systems that were found on the same site to have overlapping data in order to support the functions of those systems. The first system was in personnel, and dealt with some basic employment data, the second was a telephone directory, and the third was for administering security badges for gaining access to the site, see Figure 1. There was some data that was common to each of these systems (in bold).

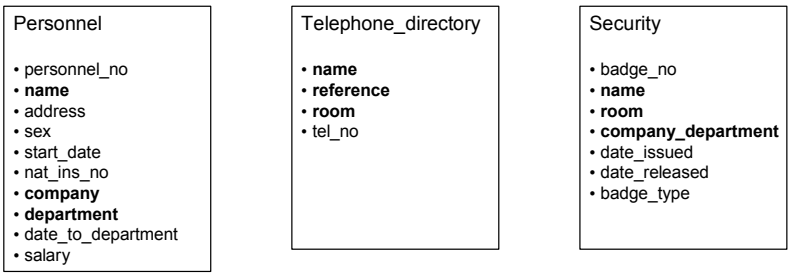

Fig. 1. Tables from three different systems holding similar data

Each of these data models supports just the requirements of the functionality to be supported, required the same data to be input into different systems, and imposes constraints on holding historical data. For example, you would loose the history of an employee's movement between departments, and his salary history.

A first remodelling of key aspects was undertaken using the principles and framework as a guide. The results are shown in Figures 2, 3, and 4. The data models are presented using the EXPRESS-G data modelling language [13].

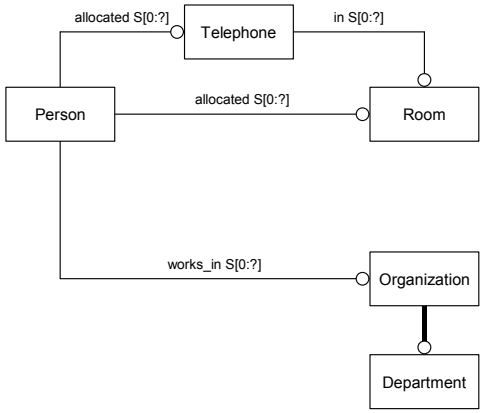

Fig. 2. Applying the principles and framework to the telephone example.

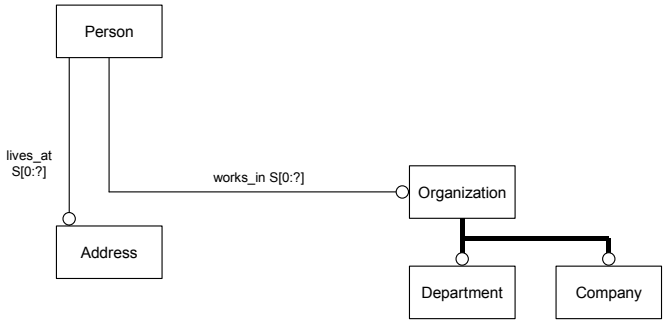

Fig. 3. Applying the principles and framework to the personnel example.

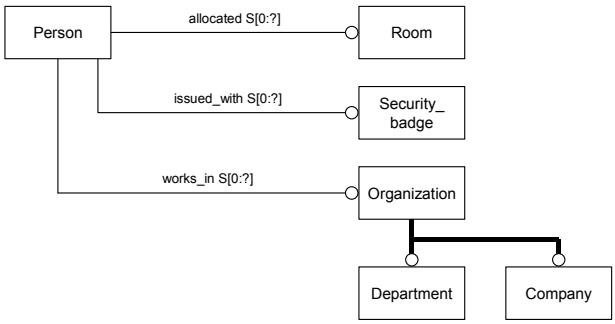

Fig. 4. Applying the principles and framework to the security example.

As a result of applying the principles and framework these three pieces of data model could be integrated into a data model for a system that could support all the requirements without redundancy of information between them, see Figure 5 below. A key learning from this was that improving data models individually can make the resulting data models easier to integrate.

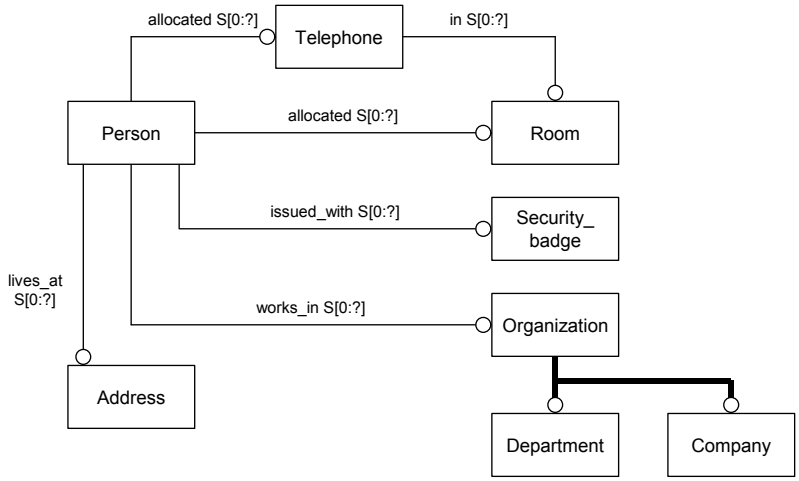

Fig. 5. The model resulting from putting the different pieces together.

Taking ideas into the wider world

If integration is a significant issue within Shell, then integrating with business partners is even more challenging. One area of particular importance to Shell is the handover of the design data for major assets like offshore platforms and refineries from the engineering contractors who design and build these facilities to operators and maintainers. To support this requirement we became involved in industry consortia such as PISTEP[1], USPI-NL[2], and EPISTLE[3] to develop ISO standards within ISO TC184/SC4 – Industrial Data[4].

This stream of work took the Shell Generic Entity Framework as its start point, from a number of alternatives, and further developed it as the EPISTLE Framework V2.0 [6], which has been the basis for a number of further developments and implementations:

[1] http://www.pistep.org.uk/
[2] http://www.uspi.nl/
[3] http://www.epistle.ws/
[4] http://www.tc184-sc4.org/

- It was simplified to become the POSC/Caesar Product Model, Snapshots A – E.
- It was the basis for developing ISO 10303-221 – Functional and schematic data for process plants.
- It was the start point for developing ISO 15926 [1].
- It was used in the PIPPIN (Pilot Implementation of a Process Plant information Warehouse) ESPRIT project [7].
- It has been adapted and implemented in a number of industrial situations [8],[9],[10],[11],[12].

However, it was found that different people could interpret some key parts of the data model differently. An example of this is given in the following case study.

An example of differing interpretations

A key concept in the EPISTLE Framework V2.0 is that of *facility*. This is illustrated in Figure 6 below. The definition provided for *facility* is as follows:

> A *facility* is a *functional thing* and is the capability to perform a function.
>
> A *facility* is a service to be provided, duty to be performed, functional view of, placeholder for, or a logical view of something.
>
> For example, tag number P1102A is a placeholder for, say, a boiler feedwater pump. If the particular pump that is fulfilling that function becomes defective, we can swap out the specific pump that is installed and replace it by another. The tag number does not change.
>
> A *facility* will often be implemented by something man-made, but it could equally be implemented by a natural *physical_thing* (a natural utility[5]).
>
> Examples of a *facility* are :-
>
> - P10, the pump service at the bottom of a column
> - 21T103, the measuring facility at the top of a column
> - The TAMANA Platform TEP-A
> - C: drive on your PC
> - The Shuaiba formation of the oil reservoirs in the Fahud field in Northern Oman
> - The Atlantic Ocean (taken, for example, as a transportation medium)

[5] An earlier version of the Framework included *natural utility* as a separate subtype.

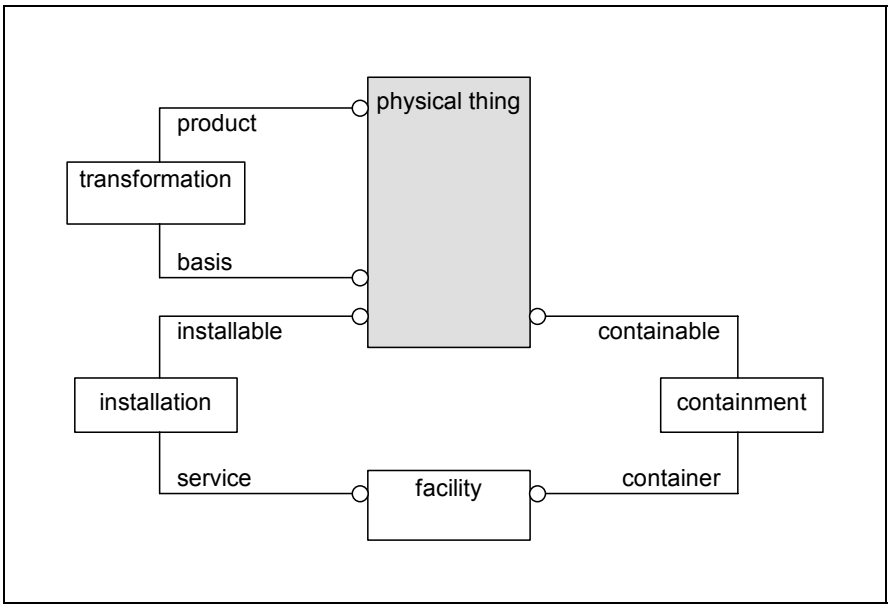

Fig. 6. Facility and some related concepts – taken from the EPISTLE Framework V2.0.

Within the subtype/supertype hierarchy *facility* is a subtype of *functional thing* which is in turn a subtype of *logical thing* which is defined as "A logical abstraction used to manage the world we live in."

Figure 6 illustrates, using the EXPRESS-G [7] language, some key related concepts. Most notable of these is *physical thing,* which is defined as follows:

A *physical thing* is a thing to which the laws of physics apply.
Examples of *physical thing* are :-

- Batch 101 (of Naphtha).
- Pump with serial no. 1234.
- Batch 165 of Stainless Steel.
- John Brown's body.
- The energy that is transferred between two materials in a heat exchanger.
- A void in a rock formation.
- The bore of an oil well.

Finally, a key relation that associates *physical thing* with *facility* is *installation* which is defined as follows:

The *installation* entity is a subtype of association. It has the following attributes :-

installable	The *physical thing* installed to provide the service
service	The facility of which the service is required that the installable has been installed to provide

An *installation association* indicates that a *physical thing* (the installable) has been installed to provide the *service* required of some *facility*.

Examples of *installation* are:-

- The *installation* of pump serial number 1234 to provide service as P1102.

The purpose of this part of the model was to allow the management of replaceable parts of significant value, i.e. the equipment replaced would be tracked in its own right. This is illustrated below in Figure 7 below.

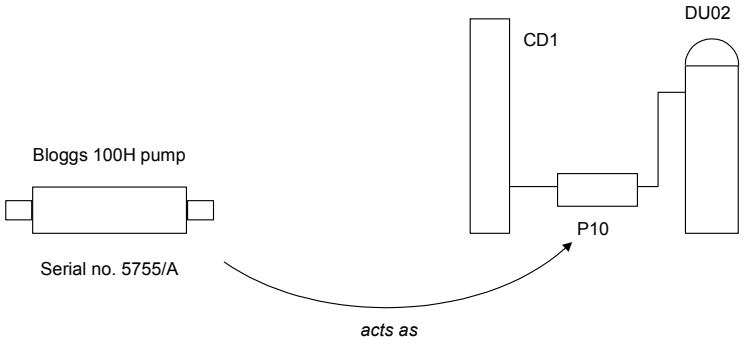

Fig. 7. Installation of a pump to act as a facility

We found that when some people implemented *facility* in the design environment they used it to carry the specification of the pump. This was not what had been intended. A *class* should have carried the specification of the *facility*, which in turn the *facility* would have been a member of. Perhaps the reasons for this is that a *facility* is abstract – it cannot be touched – and designers combined this abstract concept with that of *class* as a matter of convenience.

This misuse of the model turned out not to be fatal within the particular projects that took this view. However, it did mean that projects that had taken a different view were incompatible with each other.

Evolution of the Paradigm

A survey was carried out to establish the range of paradigms that were documented, and if any would help us resolve the issue of different interpretations and different ways to say the same thing.

Given the plethora of different database designs it was a surprise to find that in the literature two main ontological foundations for individuals could be found that had a respectable following:

1. A continuant/occurrent based foundation, where continuants endure through time, but are wholly present at each point in time when they exist, and occurrents that exist in time as well as space.

2. A four-dimensional approach that sees all objects as spatio-temporal extents [14],[15].

The former paradigm was found to be counter-intuitive to the approach we already had of all things having a lifetime. The latter paradigm, introduced to us by Partridge, on examination:

1. Mapped easily with the approach we already had to managing change.
2. Had the possibility of a clear and unambiguous identity basis based on spatio-temporal extent.
3. Reflected better the intuitions in the existing data model.
4. Provided a rigorous approach for analysis.
5. A classical mereology, which is both strong and simple is sufficient even when dealing with changing objects.

The EPISTLE Data Modelling Team decided to rework the EPISTLE Core Model using this 4D paradigm. The result can be found in ISO 15926-2:2003 [1]. After the fact, on comparing our results with the work of Sider [14], we found that our models were consistent with his preferred paradigm. In the following sections some key consequences of applying this approach are developed and explained.

Individuals as spatio-temporal extents

Individual is used here to mean an object that exists in the space-time continuum, whilst universal is used to apply to objects, like sets, that exist outside the space-time continuum. There are a number of ways that individuals can be defined. A spatio-temporal definition says that an individual is a chunk of the space-time continuum that may be extended in both space and time. An option that gives considerable clarity is to choose that any two individuals are the same if all the parts of each are also parts of the other, i.e. the identity basis is extensional in space-time. This is a one-size-fits-all definition that covers both activities and physical objects that in some approaches have a different identity basis.

A convenient tool for representing individuals and the patterns they form in space-time, is the space-time map. For representation in two dimensions, the three spatial dimensions are collapsed into the vertical dimension, and time is represented on the horizontal dimension. Different types of individual, with different identity criteria can be illustrated and analysed using this type of diagram. For example, for "ordinary" physical objects we normally allow that the object continues its existence if some parts change, but not if all parts change.

A physical object may have temporal parts, that is, all the spatial parts of an object for a period of time. We call these states; an illustration is given in Figure 8 below.

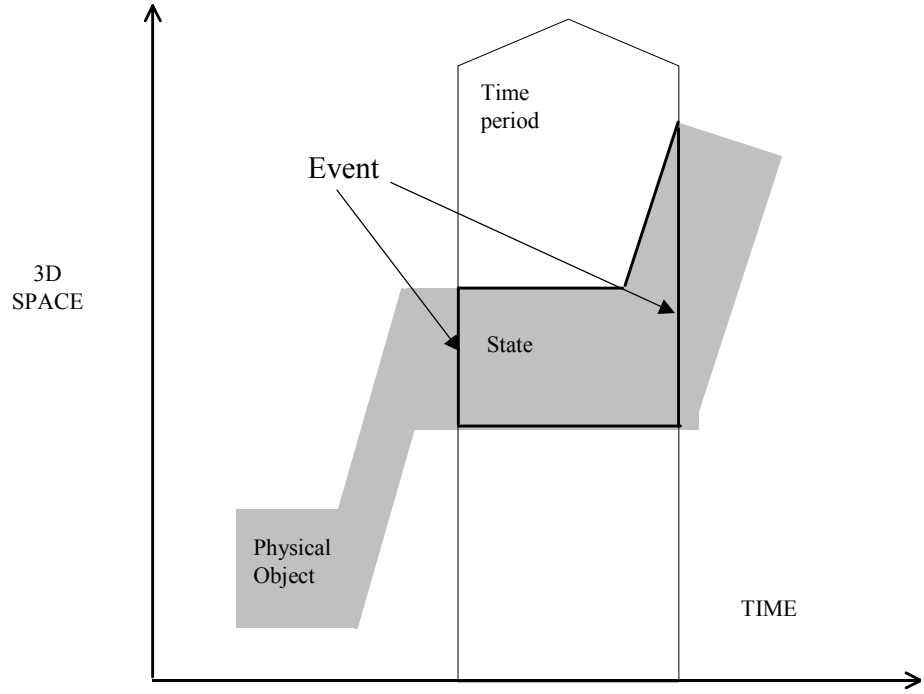

Fig. 8. State

Some properties or relationships may be true for a state that are not true for the whole life of the physical object. For example, my car may originally have been red, but now it is blue. In this case there is a state of the car that is red, and a state of the car that is blue.

The temporal boundaries of states we call events. This is quite a restricted use of the term. It reflects a state coming into or going out of existence. It is not necessarily a transition from one state to another.

This approach supports a classical mereology, as described in [16], extended into four dimensions [19]. A 3D mereology only remains classical when change in individuals is ignored. When change in individuals is taken into account a 3D mereology becomes quite complex as Simons describes [16].

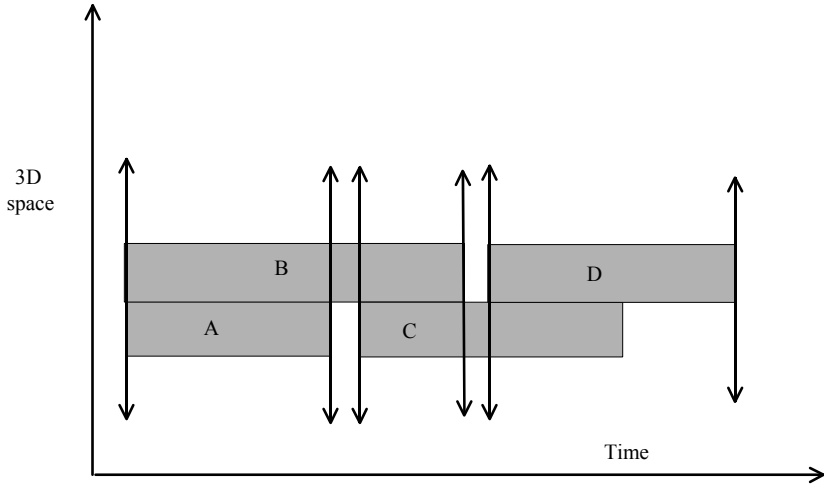

Fig. 9. A space-time map for an "ordinary" physical object

Figure 9 illustrates this for a simple example like a broom that has a head and a handle. At some point in time the old head is replaced, and at another time the old handle is replaced, but the identity of the broom is allowed to persist through these changes.

Not all physical objects take this form. Take for example the President of the United States. This is an object that exists in space-time, but it does not observe the identity criteria for "ordinary" physical objects, because from time to time all the parts change. This is illustrated in Figure 9. Because of this, some paradigms do not recognize this sort of object (similar to facility from the example above) as a physical object at all, but you can quite definitely talk to the President of the United States, and you can see the extent of this object in space-time. The interesting thing about this pattern is that some temporal parts of the President are also temporal parts of other objects, in this example Bill and George.

A similar concept can be found in Masolo et al [17]. Here it is called a *social individual* and in particular a *figure*. The idea is even introduced in 4 dimensional terms thus: “Figures could be considered as some kind of mereological fusion of the player-stages of a given role …” However, they then go on to redefine the concept in 3D terms which requires the introduction of *Qua Entities*. There is at least one additional difference. They say “But one may argue that the *figure* exists even when the corresponding role is ‘empty’ (not played by any entity), and thus the need to introduce it as a new entity is further justified.” I would argue that when the role is empty the *figure* is simply going through a period of non-existence – which would explain why they were unable to fulfil any duties at that time. You can of course refer to objects whilst they are non-existent, for example we have no trouble referring to historical figures.

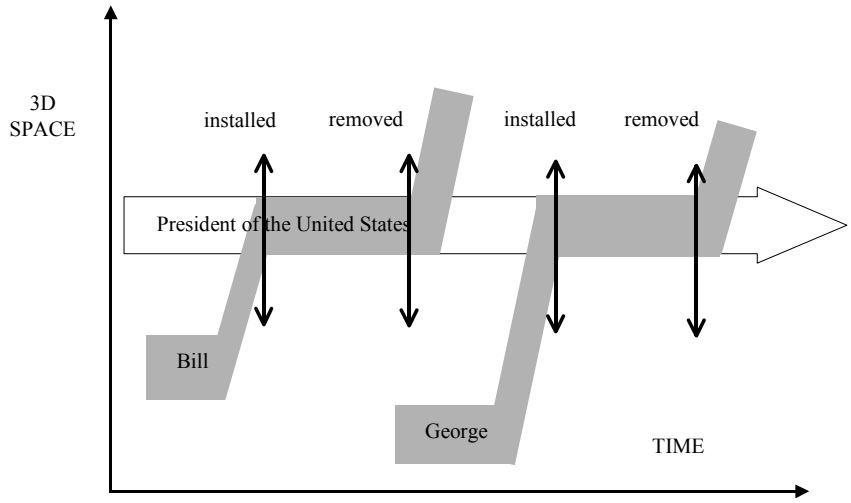

Fig. 10. Space-time map for President of the United States

Classes, physical properties, and set theory

One of the difficulties with some traditional approaches to ontology is managing change. If my car is red at one time and blue at another, not only has my car changed, but so also has the membership of the classes blue and red. This needs to be taken into account in the way that class membership works, and in particular means that classes are not sets, since the membership can change.

On the other hand, if a spatio-temporal paradigm is adopted, then we talk in terms of states. The state of my car that is red is always red, and the state of my car that is blue is always blue (even when looking into the future) and as a consequence classes become sets with unchanging membership. Similarly, physical properties, e.g. the particular degree of hotness that maps to the number 20.0 on the Celsius scale, are also sets, and some states of individuals will be unchanging members of that set.

Having concluded that sets can play a central role, the question arises as to which set theory to adopt. Classical set theories, such as Zermelo-Fraenkel set theory, allow a set to be defined by any predicate. They are also well-founded. This means that sets cannot be members of themselves. This was a constraint that was introduced into standard set theory as a result of Russell's Paradox, which defined as a predicate the set of all sets that were not members of themselves, a set that cannot be constructed, and hence causes a contradiction. This requires some objects that are not sets to be defined, or else some severe restrictions on what can be said.

Another approach that can be taken, is not to insist that there is necessarily a set for the evaluation of every predicate, but only that sets can be constructed. These are non-well-founded sets [18].

Figure 11 illustrates the allowed construction of sets in this case with the boxes representing sets, and the arrows showing set membership. As long as you can draw the set of arrows that defines the members of a set (at least in principle) then the set is considered valid. For well-founded sets no loops are allowed as e.g. between C and Z, and from P to itself.

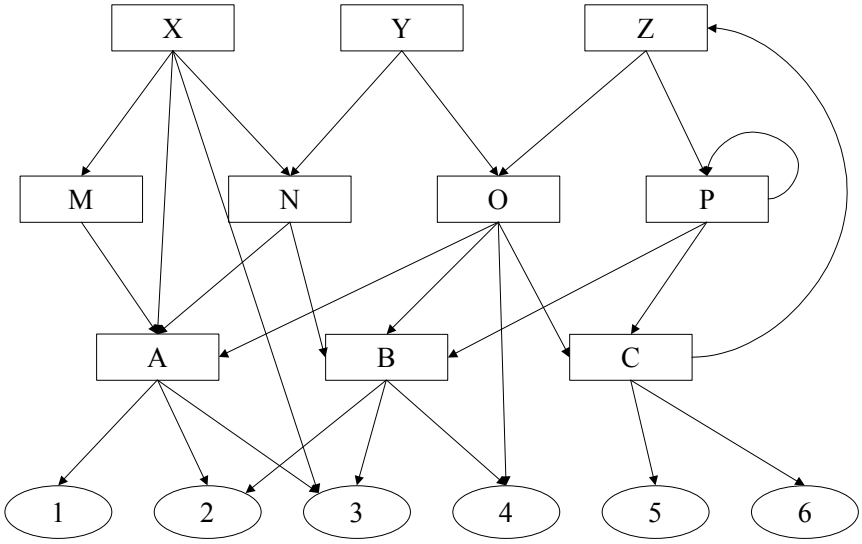

Fig. 11. Non-well-founded sets

For a complete ontology some things that need to be said are that *class* is a *class*, and *thing* is a *class*. These statements are not allowed in well-founded set theory, but are allowed in non-well-founded set theory, removing the need for some additional structures/concepts to handle this, so we chose to adopt it.

Whilst we identified the need to adopt a non-well-founded set theory, having discovered that the work had already been done, we follow the work of Aczel [18].

Conclusions

Although the ISO 15926-2:2003 data model is a foundation ontology, and is highly principled in its content and structure, this has not been a matter of fiat, but of evolving experience, trying out ideas to see if they work, and adopting new ideas when they prove their worth in practice.

References

[1] ISO 15926-2:2003 Integration of lifecycle data for process plant including oil and gas production facilities: Part 2 – Data model.

[2] Ottmann, B. West, M.R. Fyfe, A. *Reviewing and improving data models* 1992, Shell International

[3] West, M.R. *Developing High Quality Data Models: Principles and Techniques* 1994, Shell International, IC94-033

[4] West, M.R. 1994, Shell International, IC94-034 *Developing High Quality Data Models: The Generic Entity Framework V1.0*

[5] West, M.R. *Developing High Quality Data Models: Data Model Templates* 1994, Shell International, IC94-035

[6] Dziulka, P. Angus, C. editors *EPISTLE Framework V2.0 Issue 1.22* EPISTLE, 1998 available from the world wide web from http://www.epistle.ws/

[7] Thomson, A. *The PIPPIN Project* European PDT Days 1998.

[8] West, M.R. *STEP into the real world,* European PDT Days 1998.

[9] Walters, A. *The world's first operational POSC/Caesar data warehouse* European PDT Days 1998.

[10] Magnus, B.H., Park, A. *Life-cycle information data warehousing in a major offshore development project* PDT Days 1999.

[11] Wevik, H. *Generic data modelling by use of ISO 15926,* PDT Europe 2002.

[12] Wevik, H. *Lifecycle information management in generic data modelling,* PDT Europe 2003

[13] ISO 10303-11:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.*

[14] Sider, T. *FourDimensionalism: An Ontology of Persistence and Time*, Clarendon Press, 2001, ISBN 0-19-926352-3

[15] Partridge, C. *Business objects: re-engineering for re-use*, Butterworth-Heinemann, 1996, ISBN 0-7506-2082-X

[16] Simons, P. *Parts: a study in ontology*, Oxford University Press, 1987, ISBN 0-19-924146-5

[17] Masolo, C. Vieu, L. Bottazzi, E. Catenacci, C. Ferrario, R. Gangemi, A. Guarino, N. *Social Roles and their Descriptions,* Knowledge Representation, 2004

[18] Aczel, P. *Non-well-founded sets* CSLI Publications, 1988, ISBN 0-937973-22-9

[19] Stell, J.G., West, M.R. *A 4-dimensionalist mereotopology* to be published.