

A Foundation Integration Model for Industrial Data

Matthew West, Shell Services International, UK

1. Introduction

The community working to develop standards in ISO TC184/SC4 – Industrial Data (SC4) has achieved significant success in recent years developing standards for technical engineering information that meet industrial requirements. However, in doing so incompatibilities have developed between the data models in the different standards, for example, between ISO10303 – Standard for the Exchange of Product model data (STEP), ISO 13584 – Parts Library, and ISO15926 – Data Sharing for Oil & Gas and the Process Industry. Even within STEP there can be problems with interoperability between the data models in different Application Protocols (AP's).

In order to overcome these difficulties, SC4 established two Preliminary Work Items. The first is to develop a modularisation methodology to achieve interoperability between STEP APs, the second is to develop a new SC4 Data Architecture to support the integration and co-operative use of the existing initiatives. This paper presents some of the work towards the second initiative.

2. Requirements

The underlying requirement that has been identified is to develop an environment for the integration and sharing of data created according to different SC4 data standards, and also with non-SC4 data standards. This should be done in such a way as does not require (or forbid) the redevelopment of existing standards. An architecture has been developed¹, and the following components have been identified as necessary to achieve this:

- An integration data model able to support the different data requirements,
- A mapping methodology to define how data is migrated from one model to another,
- A consolidation methodology that defines how data sets are merged to eliminate duplicate information about the same object,
- Extensions to EXPRESS to support the development of a data model that crosses levels of abstraction,
- A low level data language capable of describing data and data models across different data modelling languages, i.e. languages in addition to EXPRESS.

This paper presents some of the work towards developing an integration model.

3. Approach to Developing the Integration Model

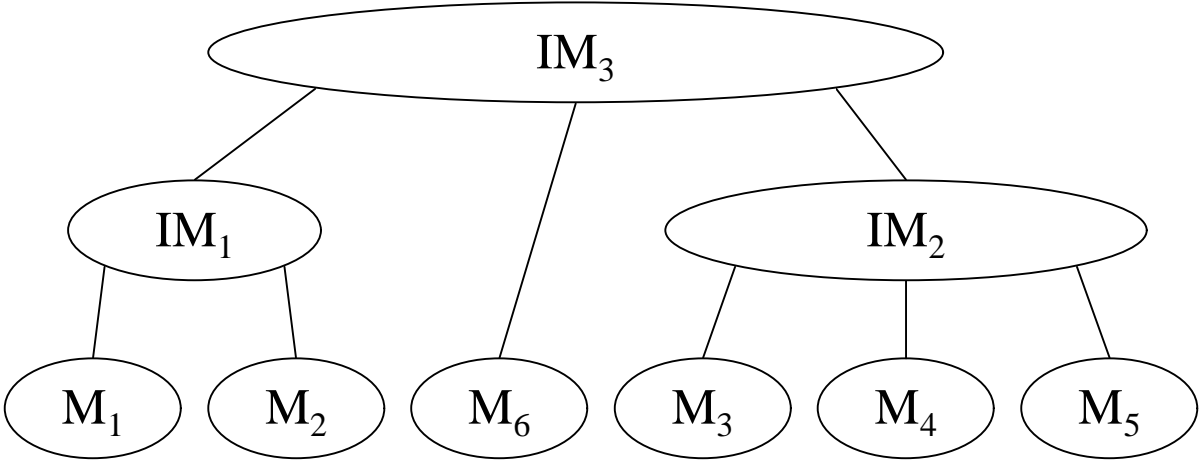


Figure 3.1: Data Model Integration

The ANSI/SPARC architecture for data models shows that for a data model, it is possible to construct views on it that are subsets, or constrained subsets of the original model. In the integration of data models, this process is reversed, that is a model is created for which the initial models are subsets, or constrained subsets. This model is an integration model with respect to the initial models, in that it is capable of holding the data from either or both of the original models. This is illustrated in Figure 3.1 above. This approach works provided that such a unified model can be created. A unified model is possible provided the world we live in is subject to a single order that we understand. Difficulties in creating such a model point to a gap in human knowledge about that order (or the lack of an order).

Note that data models that have been created as an integration model, can themselves be integrated. However, what is really desired is an integration model that is stable in the face of new models that are being integrated. Here stable means that the existing model does not need change, though extensions may be necessary when a new model is integrated. This is a design objective for an effective integration model. Much work has been done on this subject by EPISTLE (European Process Industries STEP Technical Liaison Executive)².

In principle, an integration model will consist of some irreducible or base concepts.

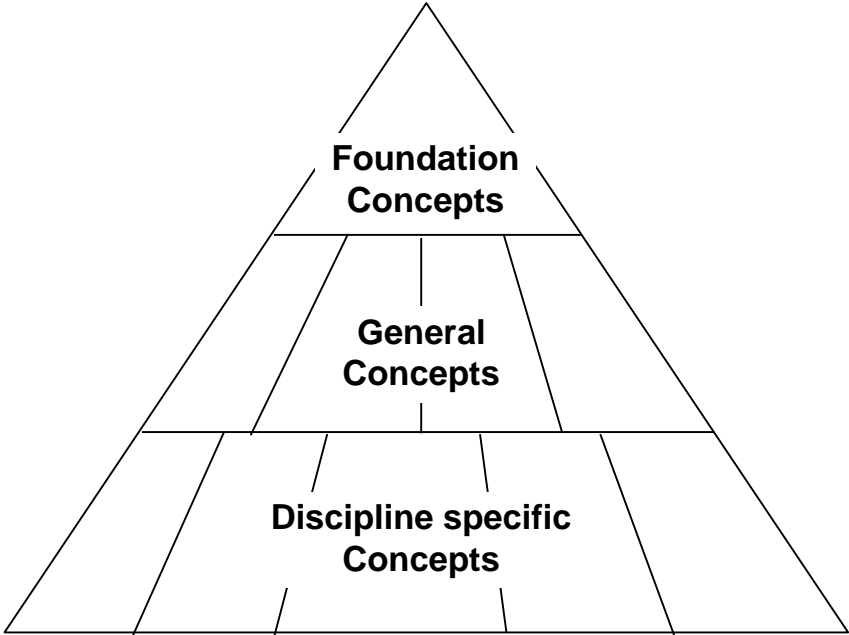


Figure 3.2: Base Concepts

These concepts are represented in Figure 3.2 above. At the top level are foundation concepts like classification, connection and composition. At the next level, there are general concepts like those of physics, and finally discipline specific concepts that are limited in their range of application. Whilst all the concepts are base concepts, those lower down the face are dependent on those higher up the face, that is the existence of the lower concept relies on the existence of one or more higher concepts. For example, without the concept of classification, there is relatively little that can be said about anything.

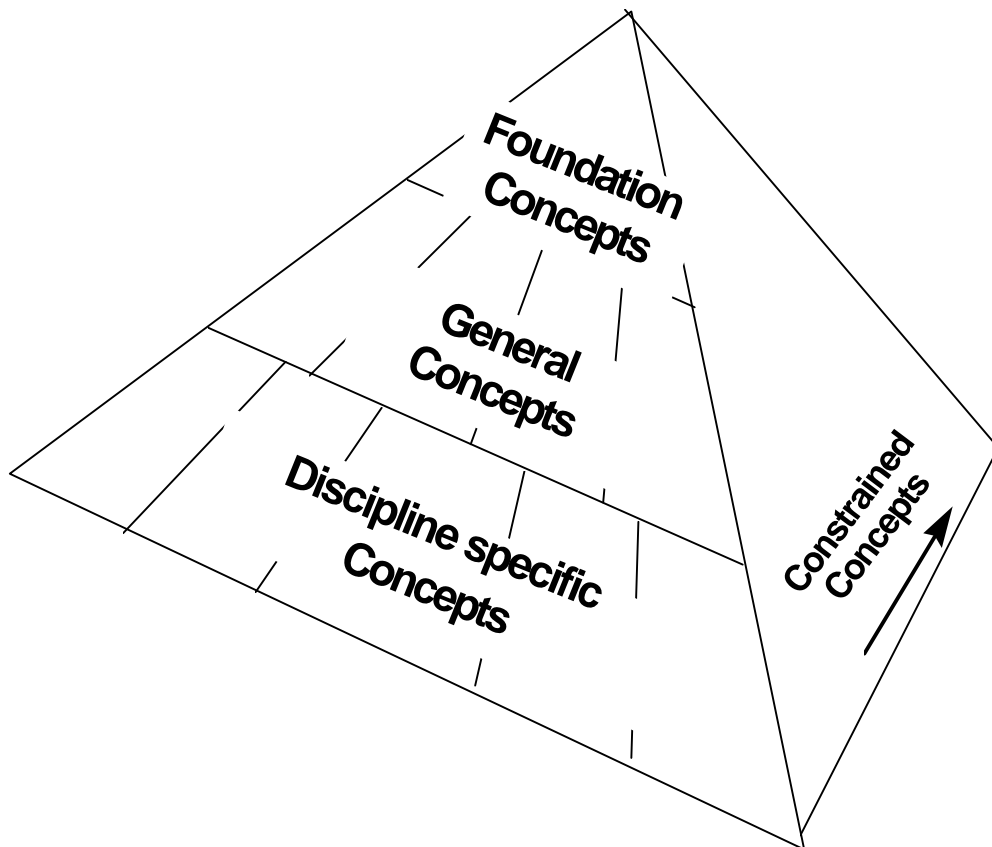


Figure 3.3: A Full Integration Model

A full integration model, as illustrated in Figure 3.3 is more than just base concepts, it includes useful and valid combinations of those concepts. A combination is formally the intersection of two or more classes.

For example, base concepts might include planned things, actual things, and material things. However, we might be interested in planned aspects of material things, or actual aspects of material things. These derived concepts can be created by intersection of the base concepts.

A key issue arises in managing change to the model. Here change is distinguished from extension. A change means either that an improved model is required to represent an improved understanding of the world, or that an error has been found that needs correcting.

In the case of an improved model, it should be the case that the new model is able to explain the old model that was used previously. This means that in principle we go through a model integration process with the old model and the new one. Some concepts that were originally base concepts will become derived concepts, and the special circumstances in which the old model holds will be identified.

In the case of an error, it is necessary to remember that the model is not the concepts modelled, but a representation of the concepts modelled. An error in representation does not necessarily reflect an error in the concept modelled. This implies that some change management process is necessary.

Whilst the issue of change management have been identified here, it is not further dealt with in this paper.

The model presented here is presented in EXPRESS. However, in developing an integration model, rather than a model to support a particular purpose, EXPRESS has some limitations.³ In particular it is not possible to show that one entity in a model is a member of another entity. This gives rise either to integration models that are very abstract, or that contain duplication between entities and entity instances. Further, part of the requirement to be met is to be able to integrate data models developed in different data modelling languages. This almost certainly requires a lower level language than EXPRESS is.

Thus whilst the model presented here is in EXPRESS, and this has the advantage of familiarity, it is likely that it will eventually be presented in a different, more basic, language.

4. Ontological Foundations

The development of the base concepts has a foundation in formal ontology, which is the branch of metaphysics in philosophy that looks at the nature of things. Some of this is relevant to the development of integration models^{4,5,6}, as well as some existing data modelling work with an ontological foundation⁷

In this section I shall outline the basic view of the universe on which the models are based. I shall not, however, attempt to justify the view by comparison with alternative positions that could be taken (after all this is a paper, not a book). The position is stated so that it is explicit, and can be challenged. It is worth noting that nothing presented here is original, although there is more than one source.

The first assumption is that we live in an ordered universe. That is to say there are particular things about us that have existence for some period of time, but that there is a pattern or order to existence which is timeless or universal. So whilst each person is an individual and separate person, there is some common pattern or order that enables me to say that some thing is a person. A key to understanding the order of things is the conditions that the order requires. For example, there are some conditions a thing must have for it to be a person, or a hydrogen atom. The order in the universe is of two types: that which is naturally occurring, and that which is imposed on top of the naturally occurring order. The naturally occurring order is the subject of study in science, whilst human endeavour often seeks to impose a further measure of order on the state of things.

The things that we see around us are continuants. They are characterised by having a unique spatio-temporal extent, or if you like a unique swept volume in space-time. If two continuants have the same spatio-temporal extent, they are the same thing. On the other hand, for some period of time two continuants can be coincident. For example, Matthew West and the Chairman of EPISTLE were coincident for a period. However, the whole spatio-temporal extent of Matthew West, and the whole spatio-temporal extent of the Chairman of EPISTLE are not identical, so they are not the same thing.

Occurrences are the changes in the world. Events mark the start or end of some individual (which may be a state) and have zero duration. Activities are aggregates of events, as such they appear to have a duration (from the first to last event) but in practice do not (if you add a lot of zeros together you still have zero).

A relation is something that one thing has to do with another (or itself). Base relations are timeless and are therefore universals. Relations that appear to be time based are actually complex objects that can be decomposed into temporal parts and timeless relations (so they are not base concepts). A relation is itself the classification of a tuple. Only binary relations/tuples are base objects. Relations/tuples of higher order can be decomposed into binary relations/tuples.

5. Entity Relationship Model

The model here is presented in EXPRESS-G for convenience. As a result, the model is presented at the level of relations, rather than as the classification of tuples, which is the base understanding. The model

is broken into a number of separate schemas, and in some cases the use of EXPRESS-G is extended beyond the standard.

5.1 Formal Level

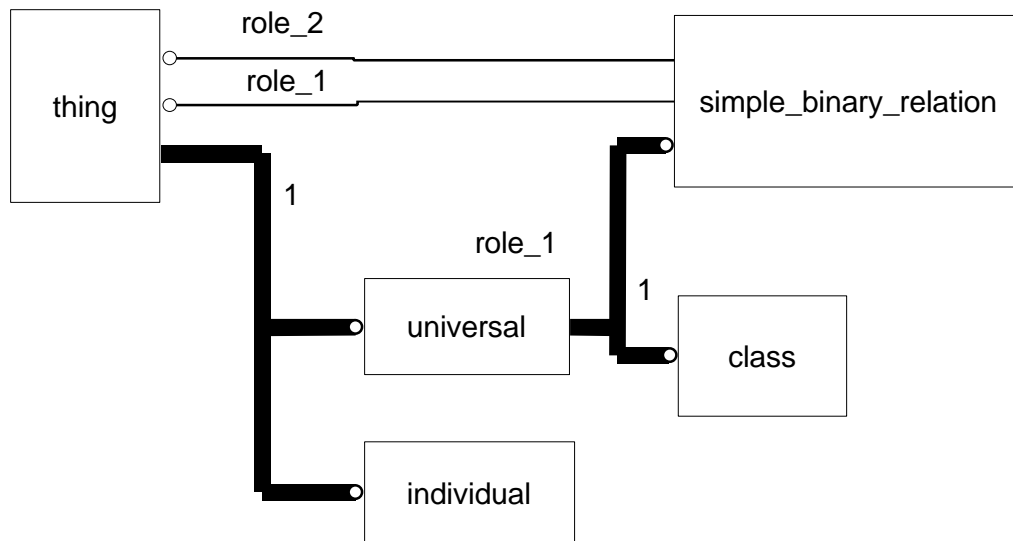


Figure 5.1: SCHEMA Level0

A *thing* is anything that exists, real or imagined. *Things* are either *universals* or *individuals*. An *individual* is something that exists in space time, e.g. my car. A *universal* is something that is timeless e.g. the temperature of 21 Degrees Celsius. A *universal* may be either a *class* or a *simple_binary_relation*. A *class* is a type of thing that is the common nature that is shared by its members, e.g. pump. A *class* may have zero, one, or many members. A *simple_binary_relation* is a type of *universal* that indicates what one *thing* has to do with itself or another *thing*. A *simple_binary_relation* is itself the classification of a tuple, but this is not modelled here.

The attributes for subtypes of *simple_binary_relation* are subtypes of *role_1* and *role_2*. This will be shown as attributes of the subtypes in subsequent models.

5.2 Individual - 1

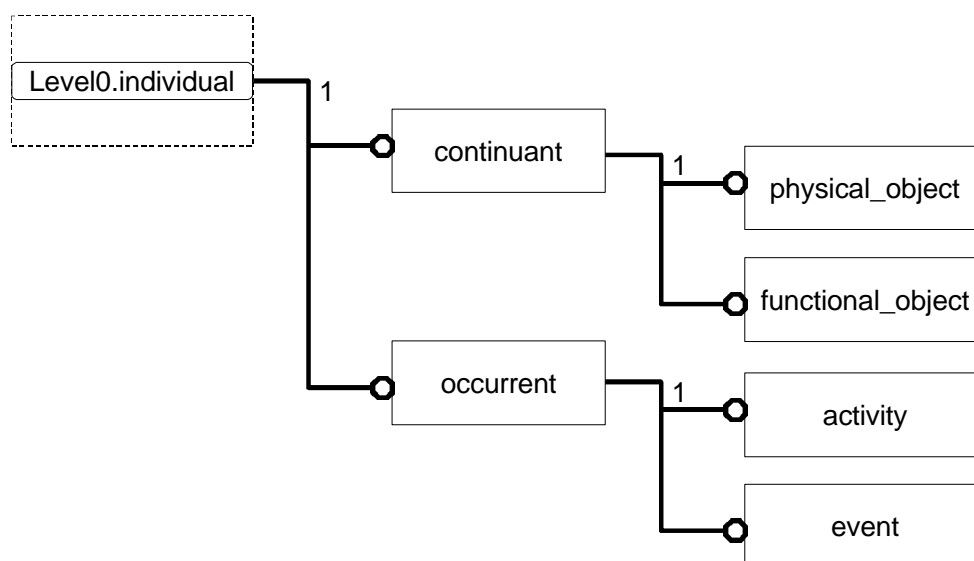


Figure 5.2: SCHEMA Individual1

An *individual* is either a *continuant* or an *occurrent*. A *continuant* is a type of individual that exists over time. That is to say it has a spatio-temporal extent. A *continuant* may be either a *physical_object* or a *functional_object*. A *physical_object* is a type of *continuant* that has essential physical continuity as its identity basis e.g. the car N199FOW. Essential physical continuity means that at least not all the parts change at once. It also allows for periods of non-existence, e.g. when a watch is dismantled, and reassembled it is still the same watch. A *functional_object* is a type of *continuant* that has essential functional continuity as its identity basis e.g. the offside front wheel of my car. A *functional_object* consists of the temporal parts of *physical_objects* that are installed to provide the function.

An *occurrent* is a type of *individual* that is a change rather than something static. An *occurrent* can be either an *activity* or an *event*. An *event* is a type of *occurrent* with zero duration that marks the start or finish of existence of one or more *continuants*. An *activity* is a type of *occurrent* that is an aggregate of event-effects.

5.3 Individual - 2

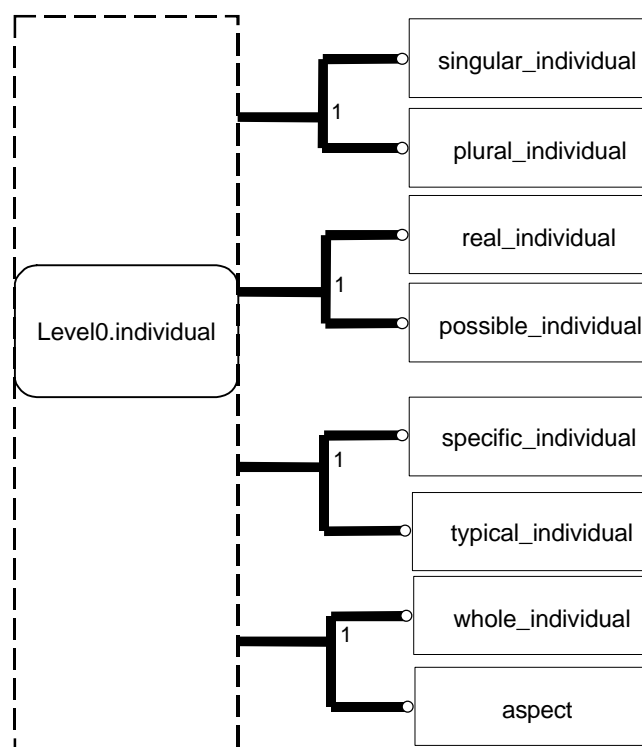


Figure 5.3: SCHEMA Individual2

An *individual* is:

- either a *singular_individual* or a *plural_individual*, and or,
- either a *real_individual* or a *possible_individual*, and or,
- either a *specific_individual* or a *typical_individual*, and or,
- either a *whole_individual* or an *aspect*.

It should be noted that any object **is** a member of one of each of these sets of subtypes, but you cannot guarantee to **know** that it is.

A *singular_individual* is a type of *individual* that is a coherent object, rather than an unorganised collection e.g. a pump. A *plural_individual* is a type of *individual* that is an unassembled collection. A

plural_individual is precisely the sum of the parts. No account is taken of any relationship between the parts, e.g. a bag of bolts, all the milk in the world.

A *real_individual* is a type of *individual* that is a part of the world we live in, as opposed to a *possible_individual* that is a type of *individual* that is e.g. fictional or hypothetical rather than one in the real world.

A *specific_individual* is a type of *individual* that has specific existence, i.e. it can be pointed at, as opposed to a *typical_individual* that is a type of *individual* that is a creation of the mind to represent some ideal, example, or average.

A *whole_individual* is a type of *individual* that has existence independent of any thing it might be a part of, e.g. a cup, a computer, as opposed to an *aspect* that is a type of *individual* that is an intrinsic part of another *individual*, yet can be considered separately, e.g. the shape of a cup, the temperature of a piece of metal.

5.4 Aspect - 1

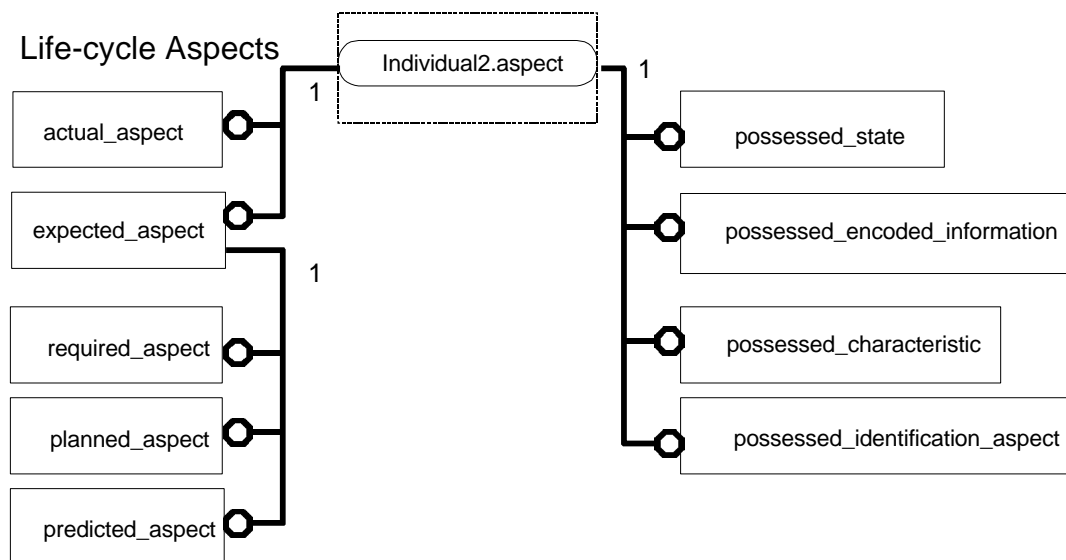


Figure 5.4: SCHEMA Aspect1

There are a large number of types of *aspect*. This schema identifies a few important ones.

All information about an *individual* relates to one or other of its life-cycle *aspects*. The *actual_aspect* is a type of life-cycle *aspect* that is the actual existence of an individual, whilst an *expected_aspect* is a type of life-cycle *aspect* that is what is supposed to be for an *individual*. An *expected_aspect* is either a *required_aspect*, a *planned_aspect*, or a *predicted_aspect*. A *required_aspect* is what is needed to be. For example, if I am drowning in a river, I have a requirement to be rescued. A *planned_aspect* is about what is intended to be done. For example, a person standing on the river bank, recognising my requirement to be rescued, might formulate a plan of how he intends to rescue me from the river. *Planned_aspects* are for things within our control. Finally, a *predicted_aspect* is for things that are outside our control. There is no point in planning customer orders, there is no control you can exert over the outcome. However, it is reasonable to predict customer orders.

Lifecycle *aspects* are orthogonal to the other aspects presented here. The remainder are only a sample of the most important *aspects*. A *possessed_state* is a type of *aspect* that is a temporal part of an *individual* for which some conditions are met. For example, a door being open. A *possessed_encoded_information* is a type of *aspect* that is a pattern in the object that can be decoded to provide information, e.g. the text on this page, a DNA code. A *possessed_characteristic* is a type of *aspect* that is a characteristic possessed by an *individual*, e.g. the temperature characteristic of my hand. The value of a *possessed_characteristic* may vary with time. Temporal parts of

Possessed_characteristics can be ordered and mapped onto a numeric scale. A *possessed_identification_aspect* is a type of *aspect* that is used to identify a *thing*. It is a combination of *possessed_characteristics*, *possessed_encoded_information* or other *aspects* by which an *individual* is known.

5.5 Types of Relation

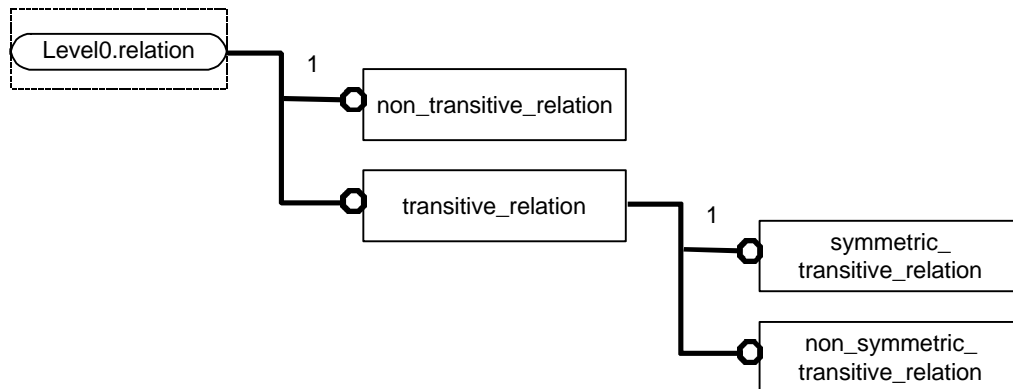


Figure 5.5: SCHEMA Relation1

A *Relation* is of one of some basic types, based on the behaviour of the *relation*. They can be either *transitive_relations* or *non_transitive_relations*. A *transitive_relation* is a type of relation for which if A is related to B and B is related to C then it is implied that A is related to C in the same sense. For example on this diagram, because *symmetric_relation* is a subtype of *transitive_relation* and *transitive_relation* is a subtype of *relation* it is implied that *symetric_relation* is a subtype of *relation*. A *non_transitive_relation* is a type of relation for which if A is related to B and B is related to C then a relation between A and C in the same sense is not implied. *Transitive_relations* can be either *symmetric_transitive_relation* or a *non_symmetric_transitive_relation*. A *symmetric_transitive_relation* is a type of *transitive_relation* for which if A is related to B, then B is related to A in the same sense, e.g. if A is connected to B, then B is connected to A. A *non_symmetric_transitive_relation* is a type of *transitive_relation* for which if A is related to B, B is not implied to be related to A in the same sense. For example if A is a part of B, B is not a part of A.

5.6 Class Member

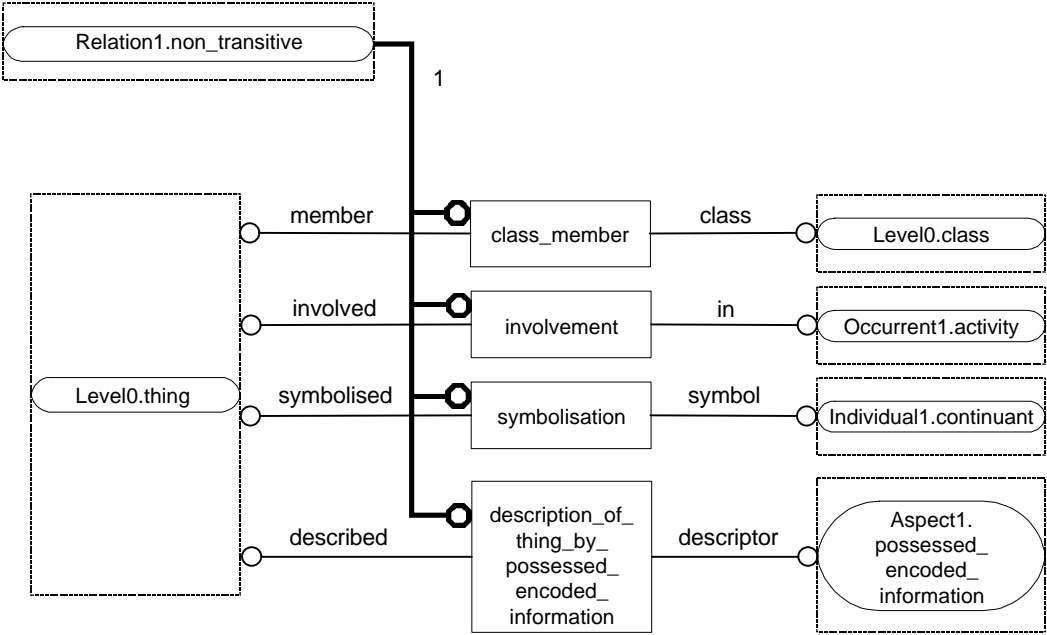


Figure 5.6: SCHEMA Non_transitive1

The most basic type of *non_transitive_relation* is the *class_member* relation. A *class_member* is a type of *non_transitive_relation* that indicates a *thing* is a member of a *class*. For example, the thing I am writing this paper on is a member of the class computer. Since *relations* are timeless, then a *class_member* is only valid when it is true for all time. Thus if a *thing* is only a member of a class for a period, then a temporal part of the *thing* is classified rather than the whole *thing*.

An *involvement* is a type of *non_transitive_relation* that indicates a *thing* is involved in an *activity*. For example I am involved in an activity of writing.

A *symbolisation* is a type of *non_transitive_relation* that indicates that an *individual* stands for a *thing*. For example, in a computer system a record may represent me.

A *description_of_thing_by_posessed_encoded_information* is a type of *non_transitive_relation* that indicates a *posessed_encoded_information* describes a *thing*. For example, a 3D model describes the product it is of.

5.7 Symmetric Relations

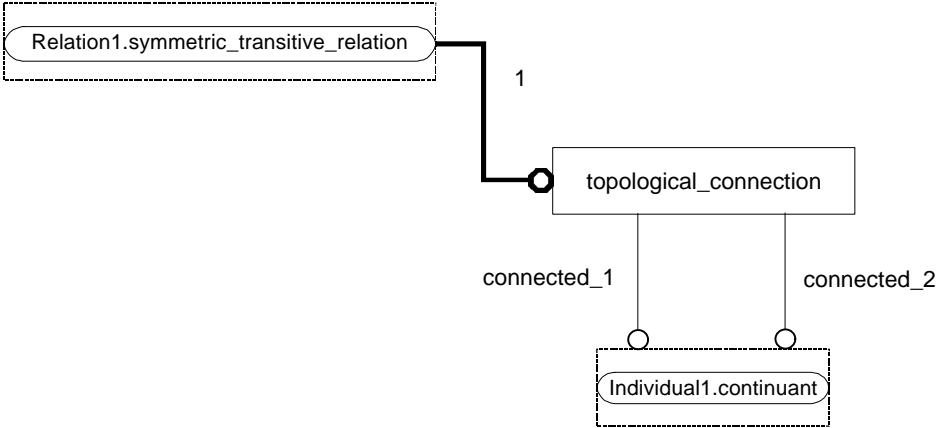


Figure 5.7: SCHEMA Symmetric1

A *topological_connection* is a type of *symmetric_transitive_relation* that indicates one *individual* is physically connected to another *individual* (or itself). For example, Pipe1234 is connected to Pipe 1235.

5.8 Non-Symmetric Transitive Relations

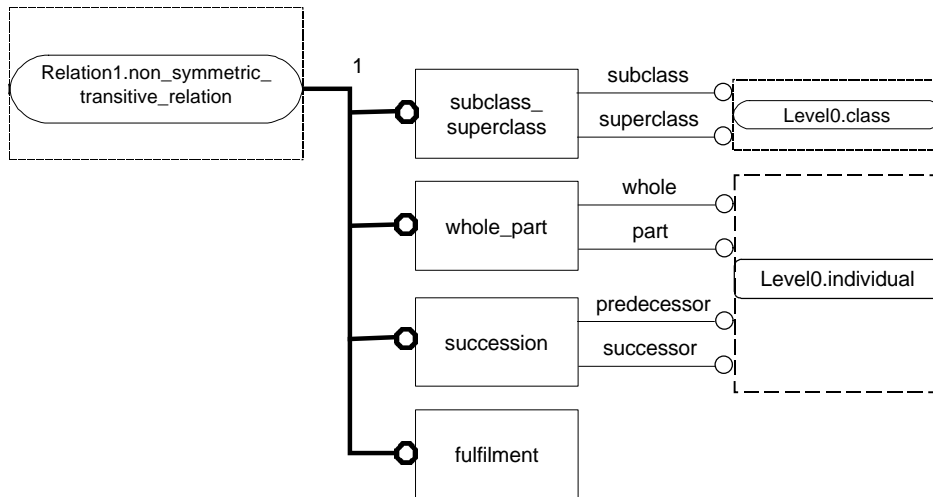


Figure 5.8: SCHEMA Non_symmetric1

A *subclass_superclass* is a type of *non_symmetric_transitive_relation* that indicates all members of the subclass are members of the superclass. For example, all things that are members of the class *continuant* are also members of the class *individual*.

A *whole_part* is a type of *non_symmetric_transitive_relation* that indicates that one *individual* is a part of another *individual*. For example, the keyboard I am typing on is part of my computer. Some types of whole part relation are given below.

A *succession* is a type of *non_symmetric_transitive_relation* that indicates one *individual* comes after another. There is no overlap in existence. For example, the 2nd World War was a successor to the 1st World War.

A *fulfilment* relation is a type of *non_symmetric_transitive_relation* that indicates one life-cycle *aspect* fulfils another. Some types of *fulfilment* are given below.

5.9 Whole-Part Relations

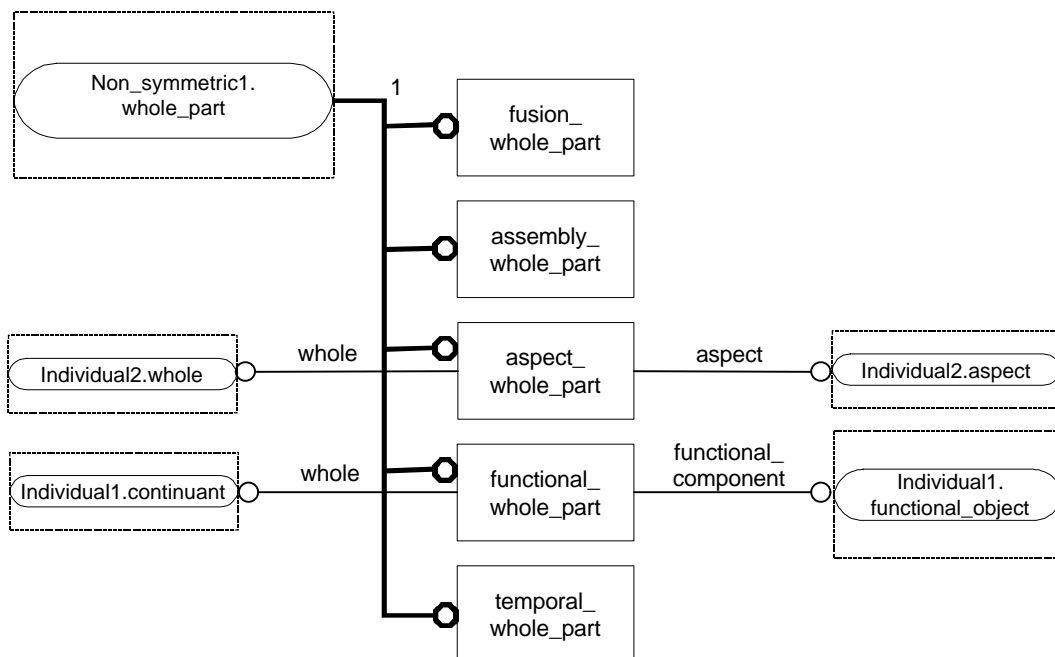


Figure 5.9: SCHEMA Whole_part1

A *whole_part* relation may be a *fusion_whole_part*, an *assembly_whole_part*, an *aspect_whole_part*, a *functional_whole_part*, or a *temporal_whole_part*.

A *fusion_whole_part* is a type of *whole_part* that indicates the whole is precisely the sum of the parts, i.e. the arrangement or connectedness of the parts is irrelevant to their being a part of the whole, e.g. a bag of bolts, all the milk in the world.

An *assembly_whole_part* is a type of *whole_part* relation that indicates the whole is something more than the sum of the parts. Assembly happens at a number of different levels e.g.

- electrons, protons, and neutrons into atoms,
- atoms into molecules,
- molecules into pieces or organisms,
- pieces into assemblies.

An *aspect_whole_part* is a type of *whole_part* that indicates that an *aspect* is an intrinsic part of a *whole* that can be considered separately from it. For example, the shape of my computer is an aspect that can be considered separately. Note: the attributes are specialisations of the whole/part attributes of *whole_part*.

A *functional_whole_part* is a type of *whole_part* relation that indicates a *functional_object* is a functional component of an assembly independent of the physical component that is there from time to time. For example, the No 1 engine on the destroyer Dartmouth is independent of the serial number on the turbine casing which identifies the particular turbine (T1234) installed. Note: the attributes are specialisations of the whole/part attributes of *whole_part*.

A *temporal_whole_part* is a type of *whole_part* that indicates the part is a temporal part (time slice) of the whole. For example, the turbine T1234 whilst it was installed as No1 engine on the destroyer Dartmouth.

5.10 Fulfilment

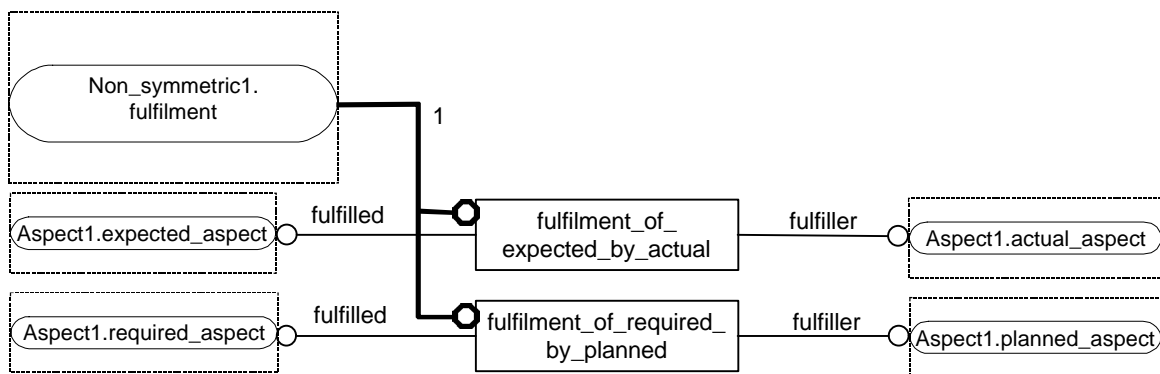


Figure 5.10: SCHEMA Fulfilment1

A *fulfilment* may be either a *fulfilment_of_expected_by_actual* or a *fulfilment_of_required_by_planned*.

A *fulfilment_of_expected_by_actual* is a type of *fulfilment* that indicates an *actual_aspect* fulfils an *expected_aspect*. Fulfilment implies satisfaction. For example, my planned holiday in Istanbul was fulfilled by my actual holiday in Istanbul.

A *fulfilment_of_required_by_planned* is a type of *fulfilment* that indicates that a *planned_aspect* is intended to fulfil a *required_aspect*. Fulfilment implies the *planned_aspect* is satisfactory. For example, the commissioning plan for Plant 1 satisfies the requirement to start production on 1st October.

6. Conclusions and Further Work

An entity relationship model has been presented that:

- has an explicit ontological foundation,
- is expressed in terms of base concepts,
- is modular,
- is extensible.

As such it provides an appropriate foundation for a model to integrate information from a number of sources.

The model is not complete, and is never likely to be. This is one reason why extensibility is crucial. However, it will be able to meet requirements covered by its degree of completeness at any point in time.

Further work that is required includes:

- extension of the model into other important areas,
- investigating how to manage change to the model,
- development of a meta-model.

7. References

¹ West, M.R. "Integration of Industrial Data for Exchange Access and Sharing: A Proposed Architecture", 1998, ISO TC184/SC4/WG10/N195.

² West, M.R. Fowler, F. "Developing High Quality Data Models, Version 2", 1996 <http://www.stepcom.ncl.ac.uk/>

³ West, M.R. "Data Integration Architecture Requirements for EXPRESS", 1999, ISO TC184/SC4/WG10/N249

⁴ Guarino, N. "Some Organizing Principles for a Unified Top Level Ontology" 1997, <http://www.ladseb.pd.cnr.it/infor/ontology/ontology.html>

⁵ Guarino, N. "Some Ontological Principles for Designing Upper Level Lexical Resources", 1998, <http://www.ladseb.pd.cnr.it/infor/ontology/ontology.html>

⁶ Hart, H. "Understanding our World", 1984, University Press of America, ISBN 0-8191-4258-1.

⁷ Partridge, C. "Business Objects, re-engineering for reuse" Butterworth Heinemann, 1996, ISBN 0-7506-2082-X