

Integration and Sharing of Industrial Data

Matthew West, Shell Information Services Ltd, UK

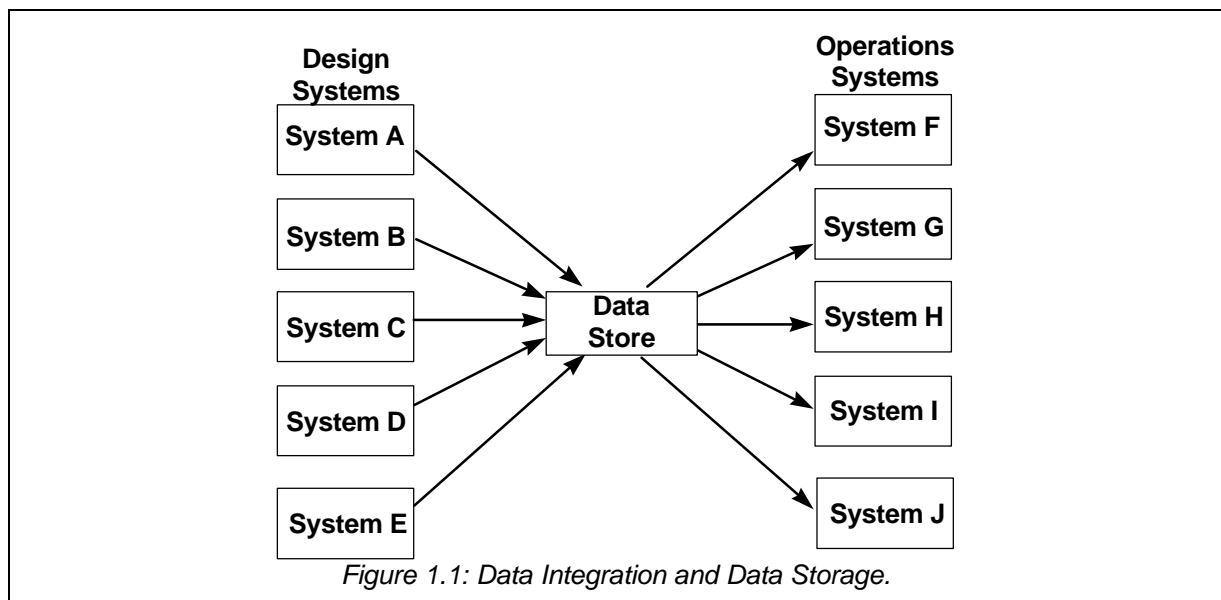
1. Introduction

Standardisation efforts, such as STEP, have so far concentrated on the exchange of design engineering data between applications. Eventually this information needs to be handed over to the owner of the asset. However, the design information will typically be held in many design systems, with no guarantee of consistency of information between them. Thus it is desirable, before handing this information over, to integrate the data from the different systems. On receipt, the asset owner may have a number of systems that he wishes to populate with the design data. The ESPRIT project 20506, PIPPIN (Pilot Implementation of a Process Plant Information system) is looking at this specific problem for the process industry.

This paper gives an overview of some of the work done by PIPPIN in the context of ISO TC184/SC4/WG10¹ to review and improve the architecture of the SC4 standards. It identifies requirements for standards to support industry for the integration and sharing of industrial data, within the SC4 standards, between the SC4 standards, and with other standards, and discusses key concepts relevant to the effective and efficient satisfaction of these requirements.

2. Problem Statement

The current focus of STEP is the exchange of Product Data, ensuring that it is meaningful in a particular context to a receiving application. However, rather than being used directly by some receiving application, the data may be integrated with data from other sources, stored, and the whole, or some subset of it, may then be used by an application. This is illustrated below.



In addition, the data may not come from a STEP AP, but from some other standard such as PLib, EDIF, POSC, or something SGML based. It is also not necessarily product data, but about some other aspect of the business. Thus it is not possible even to assume that the data model is defined in EXPRESS.

3. Requirements

To solve the above problem in a standardised way, the following requirements need to be satisfied:

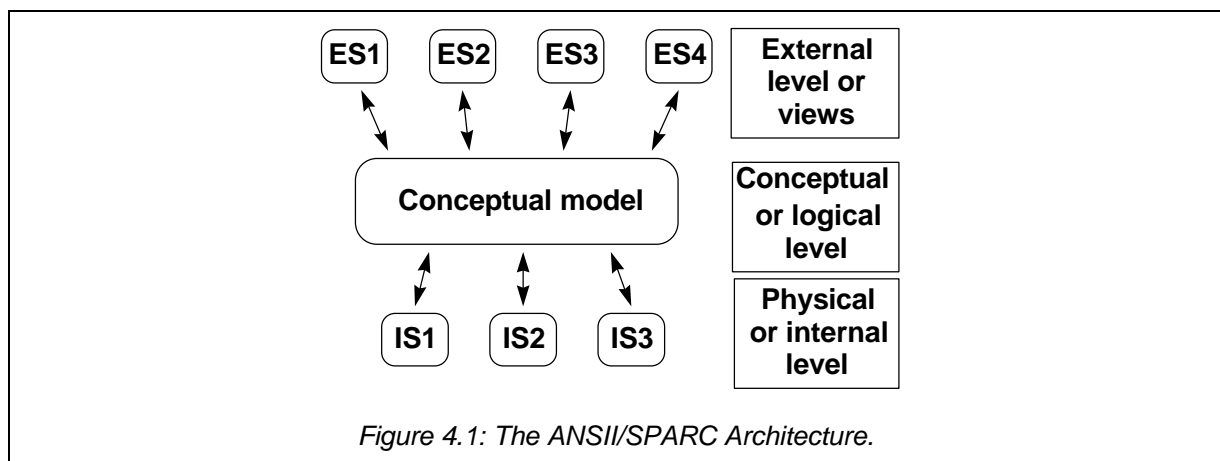
1. There is a requirement for a data model able to hold data from different sources independent of the usage of the data.
2. There is a requirement to be able to integrate product data with other industrial and business data.
3. In order to have the minimum number of data models for integrating data from different sources, these models are required to be extensible and modular.
4. There is a requirement to create two way mappings between the application level data models and the integration level data models, for both the creation and exchange of data sets, and for being able to view and update the data directly.
5. There is a requirement to handle data models in languages other than EXPRESS.
6. There is a requirement to identify the commonality between different data models in a consistent and manageable manner.
7. There is a requirement to consolidate data sets that contain data about the same objects. In particular this means being able to define when you are satisfied two records are about the same object.

4. Discussion

This section discusses a number of key concepts that are relevant to satisfying the requirements stated above.

4.1 Data Integration

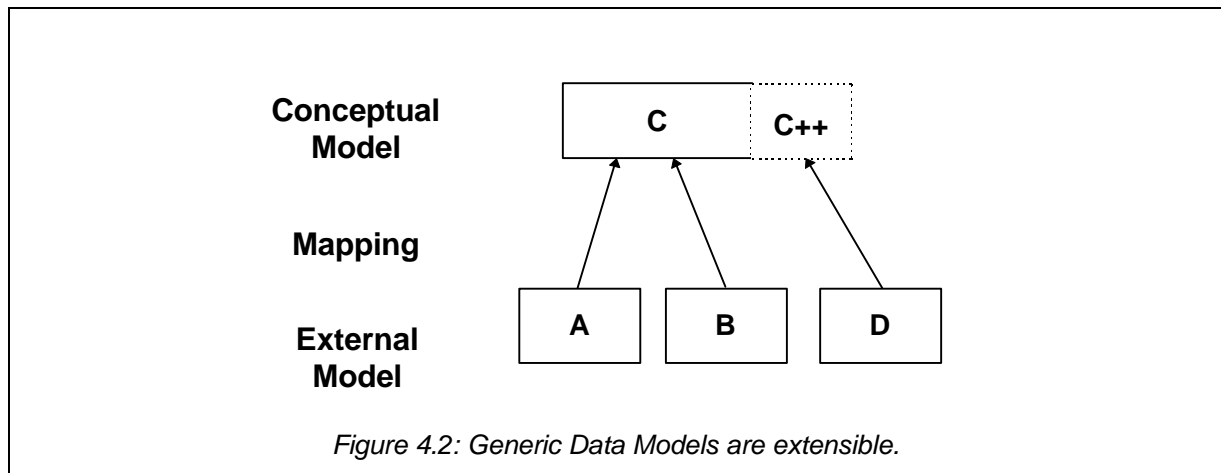
The primary requirement is to support the integration of industrial data from different data sets, according to different data models, into a single data set, and a single data model. The data in this model might then need to be viewed from a perspective defined by yet another data model.



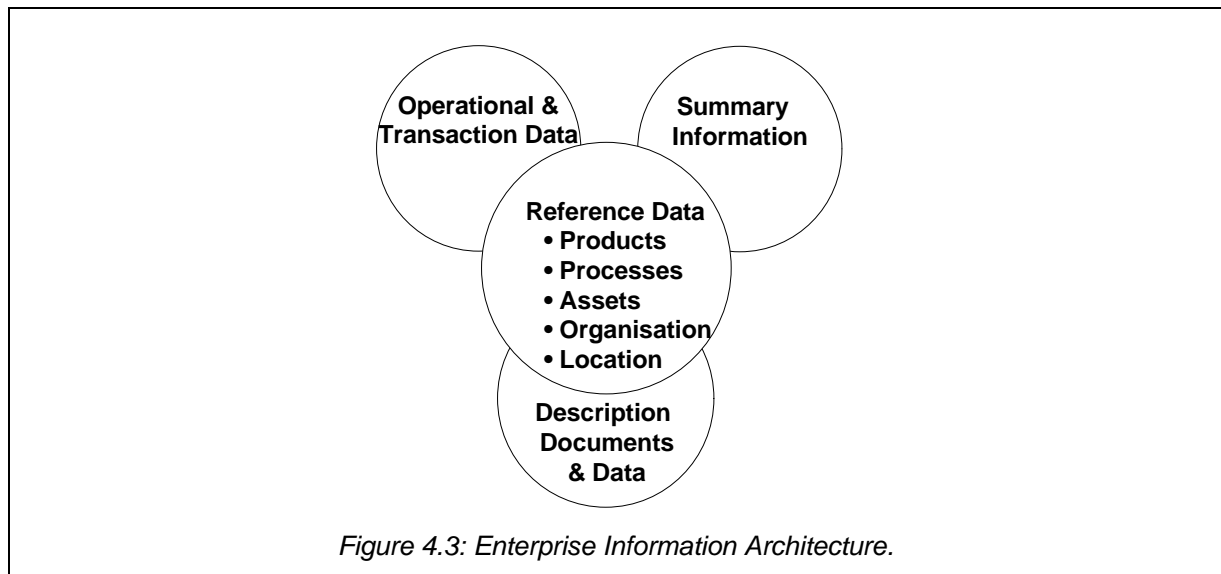
ANSI/SPARC introduced the idea that a data model that could support the data for a number of data models was a conceptual data model, see figure 4.1 above. This term will be used here, but to describe the relationship between one data model and another. Thus one data model might be conceptual relative to two or more others, i.e. it can support all of their data requirements (but not necessarily constraints).

Whilst it is well understood how to develop a conceptual data model from some predefined set of external models, this could give rise to a large number of conceptual data models for the different combinations that arise. It would be desirable to have only one, or a few conceptual data models.

However, since all the data models that need to be integrated will not be known at the start of the process, it is important that any conceptual data model is extensible in the face of additional information requirements, rather than requiring change to the existing model.



This will be particularly important to be able to integrate product data models with data models of other parts of a business. Product data is a small, but important part of an enterprise's data, and is part of the basis for the integration of an enterprises' information¹, as shown below.



Development of such an extensible conceptual data model is expected to be a technical objective. Such data models are called here Generic Data Models. The requirements for Generic Data Models have been established² as should:

- √ meet the data requirement,
- √ be clear and unambiguous to all (not just the authors),
- √ be stable in the face of changing data requirements,
- √ be flexible in the face of changing business practices,

- √ be reusable by others,
- √ be consistent with other generic data models covering the same scope, and
- √ be able to integrate data from different data models.

Fortunately, principles exist³ which, if followed, allow these requirements to be met. They are:

1. *Candidate attributes should be treated as representing relationships to other entity types.*
2. *Entities should have a local identifier within a database or exchange file. These should be artificial and managed to be unique. Relationships should not be used as part of the local identifier. (External, or global, identifiers are objects in their own right, see principle 1.)*
3. *Activities, associations and event-effects should be represented by entity types (not relationships or attributes).*
4. *Relationships (in the entity/relationship sense) should only be used to express the involvement of entity types with activities or associations.*
5. *Entity types should represent, and be named after, the underlying nature of an object, not the role it plays in a particular context.*
6. *Entity types should be part of a subtype/ super type hierarchy of generic entity types in order to define a universal context for the model.*

Developing data models that follow these principles has been found to lead to data models that satisfy the above requirements.

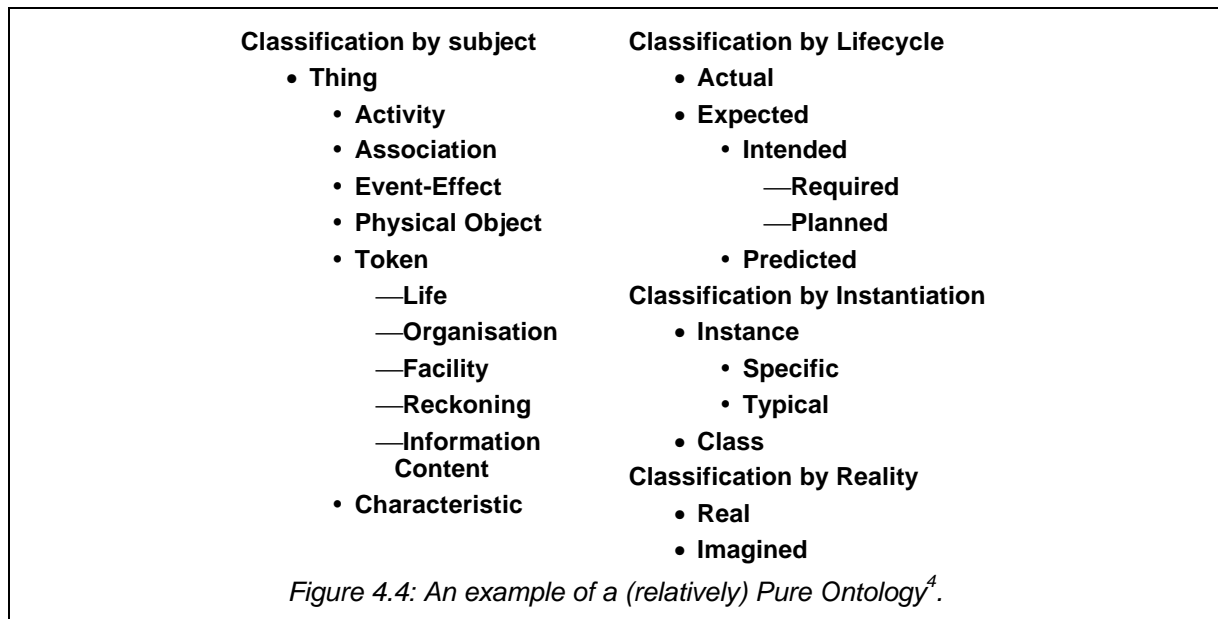
4.2 Partial Integration

Another form of integration is partial integration of two data models where they overlap. Here a single data set is not achieved, but the overlap data is mapped/viewed in both data models. This might be appropriate when some data was only ever relevant in a particular environment. Full integration can always be achieved at a later date. This is equivalent to the qualified external resource approach proposed by Guy Pierra.

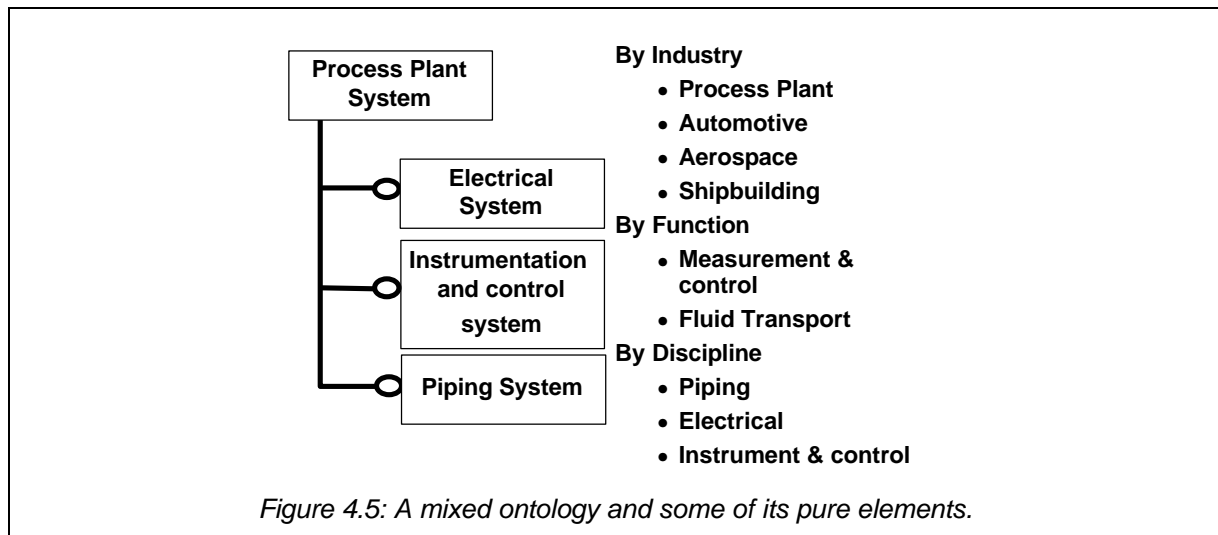
4.3 Ontologies

When you develop Generic Data Models, much of the semantic information is taken out of the data model. If this is not to be lost then it needs to be held somewhere else. An appropriate place to hold this information is in an ontology.

An ontology here is taken to be a classification scheme, and the relationships between classification schemes that represent our knowledge about some part of the universe. Two types of ontologies are recognised:



1. Pure ontologies, here the basis for classification is the same throughout the classification hierarchy. Such ontologies can be expected to be orthogonal. Here orthogonal means that membership of a class in one hierarchy is independent of membership of another hierarchy, and that classes at any level will be mutually exclusive. Figure 4.4 gives an example.
2. Mixed ontologies, these are the combinations of classes of thing we are interested in for a purpose. They are generally of more practical use, but can easily overlap with each other. The overlaps can be managed through the elements that make up the mixed ontologies coming from pure ontologies. Figure 4.5 below shows an example taken from ISO 10303-227.



Pure ontologies tend to be concise, whereas the mixed ontologies that can be derived from them are relatively large. This means that pure ontologies can be a useful tool for managing the mapping between data models. An entity type in a data model can be positioned (or not) within each of the pure ontologies. Doing so defines very precisely the meaning of an entity type, in a way that is consistent. Thus if the pure ontologies are considered to be an adjunct to the generic data model, then a consistent and extensible basis exists for integrating data from different data models. Further, if a data model is developed that can be driven by ontologies, then the data model no longer needs to be extensible, only the ontologies (using the same principles). Thus extensibility becomes a matter of managing additions to data rather than data models.

Alternatively, the ontologies can be used to specialise the data model to make greater detail explicit, should this be required. It is important to understand that declaring something as a member of an entity type is semantically identical to classifying it as being a member of a class, where the definition of the class and the entity type are the same. They are just different representations/presentations of the same information.

A data model designed to be driven by ontologies and to hold any data about any thing, is given in Annex A.

4.4 Formal mappings between data models

Mapping involves two things:

1. it takes the (implicit) context of an external data model, and maps it to/from explicit elements in the conceptual data model,
2. it takes the explicit elements in the external data model and maps them to/from equivalent elements in the conceptual data model.

This gives the basis for mapping data elements according to the external data model to and from the conceptual data model. This process can also be adopted privately for mapping application data structures to standardised data models. Below is given an example of mapping two data models. The examples are for products taken from an oil industry context.

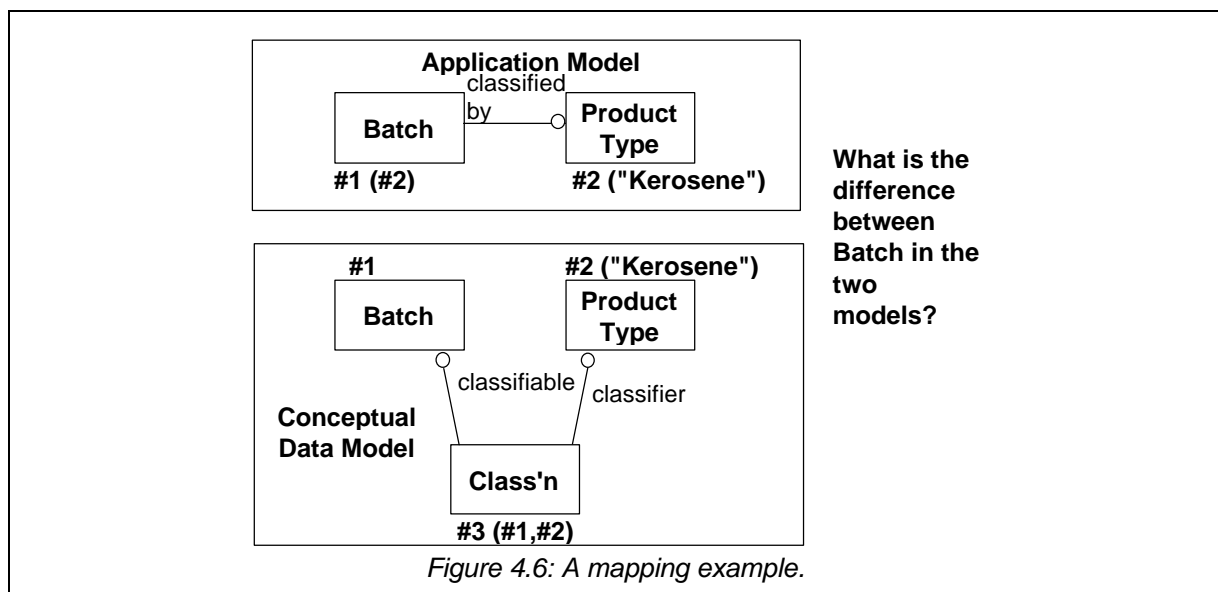


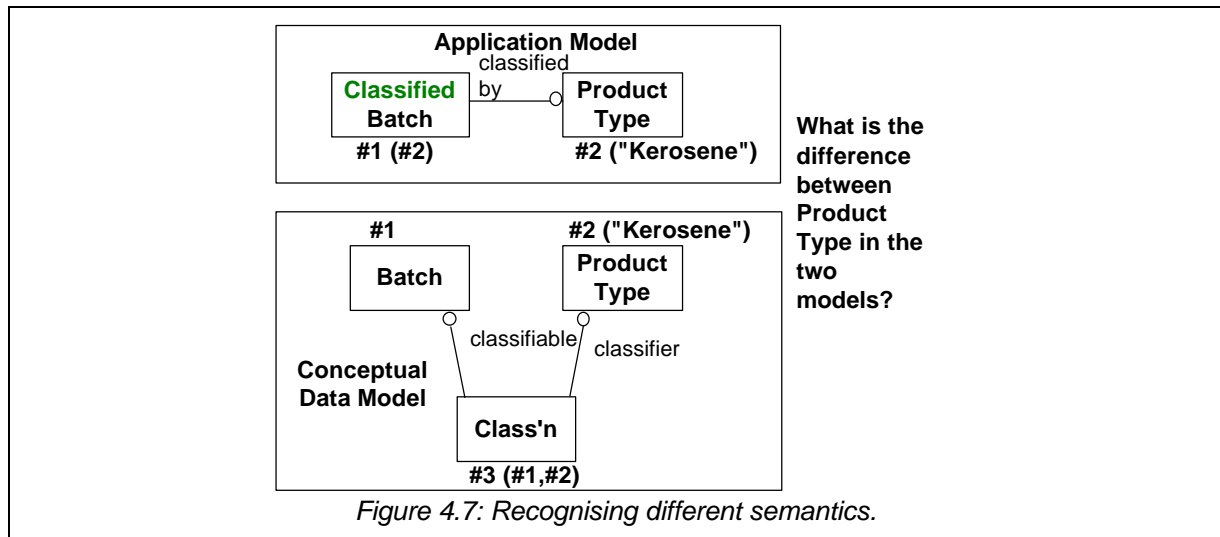
Figure 4.6 shows two data models that need to be mapped. First let me describe the two data models.

The top data model is one that is used in a particular context to identify the *product type* which represents the primary specification a *batch* of hydrocarbons is made to. The model is annotated with an example that is given in partial part 21 file format (the entity type names have been omitted because you can see them on the diagram). So *product type* #2, Kerosene, classifies *batch* #1. However, a *batch* can never have more than one *product type*, and must be classified by at least one *product type* if you are to record any information about it in this context.

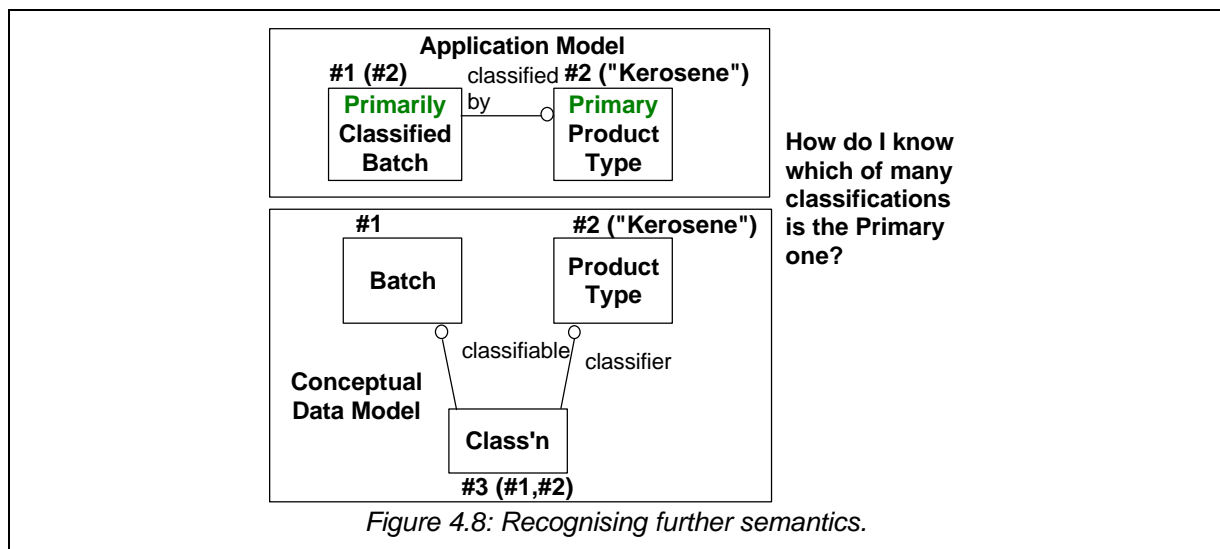
The bottom data model is designed to be conceptual, i.e. able to hold *classification* data by *product type* about *batches* in any context. The *classification* association indicates that a *batch* is classified as being of a *product type*. *Batches* do not have to be classified, and may be classified as many times as is desired. So, for example, *classification* #3, classifies *batch* #1 as *product type* #2, kerosene.

When we compare the two models we discover that *batch* means something different in the two data models. In the top data model it means *batches* that are classified, in the bottom data model it means any *batch*.

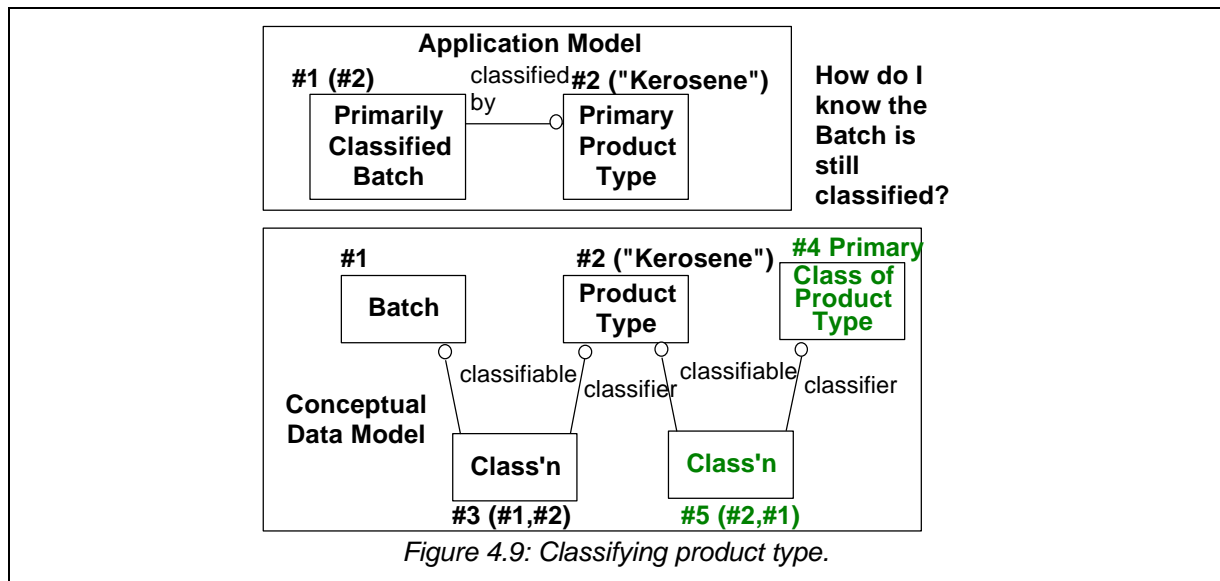
For the purposes of this example, the names of entity types will be changed to reflect the difference in semantics. However, this is not to suggest that this is necessary to map two data models. It is only necessary to discover and record the relationship between the two models. The result of this is shown in figure 4.7 below.



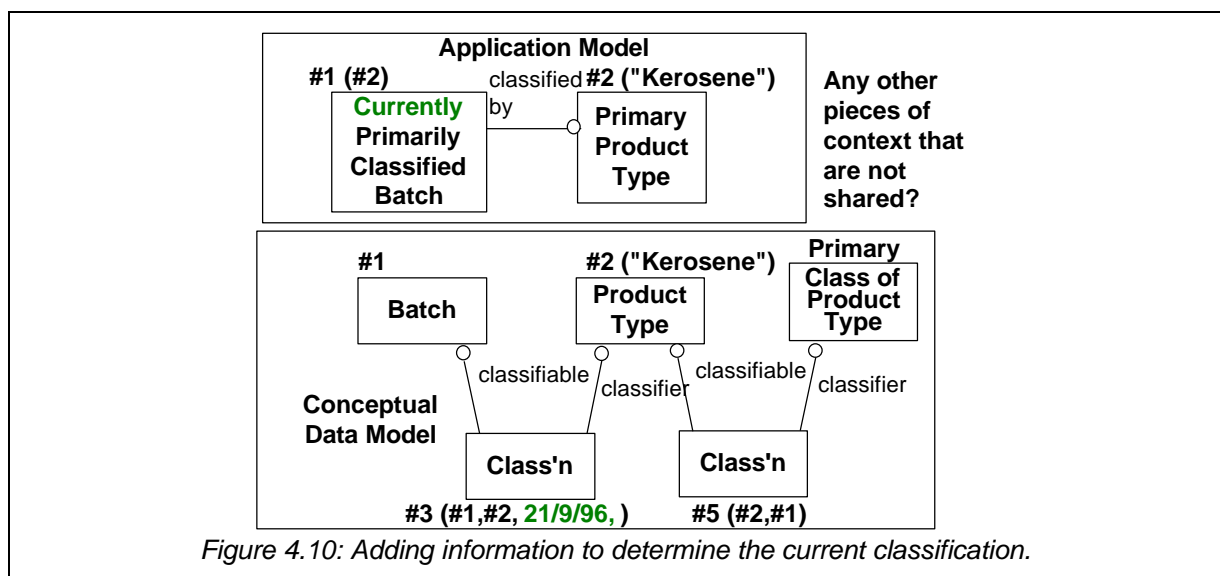
If there is a difference in the *batch* entity types, perhaps there is also a difference in the *product types*. This is quite likely, since the *batches* in the top model can only have one *product type* so any others it might have are not relevant in this context. There is no evidence just from the model to guide us, so we will assume that in the top model it is the *primary product type* that is of interest (in real life don't assume, ask!). This will also mean that *batches* that are classified, but not by a *primary product type* will not be of interest. If we incorporate this into the model then the result is shown in Figure 4.8 below.



The problem now is that when I populate the top model from instances of the bottom model, I don't know which of several classifications of a batch is the primary one. Thus we need to add something to the bottom data model to capture these semantics. Here we take a simple approach and add a further *classification* association, and introduce a *class of product type* entity type. This is illustrated in Figure 4.9 below. (Note: other approaches are possible.)



Now we know which *product types* are primary, so we can pick up the relevant *classification* associations. If we assume that the *primary product types* are mutually exclusive, then there will only be one *classification* that is current. However, the *primary product type* may change over time, in this case there will be more than one *classification* association with a *product type* that has a *classification* association with the *class of product type* primary. So we need to know which one is current. One way to do this is to hold the date effectivity of the association. A simple way to do this is to have the start and end date of the association as attributes. Of course this also adds to the semantics of the top model, which becomes a *currently classified batch*. This is shown in Figure 4.10 below.

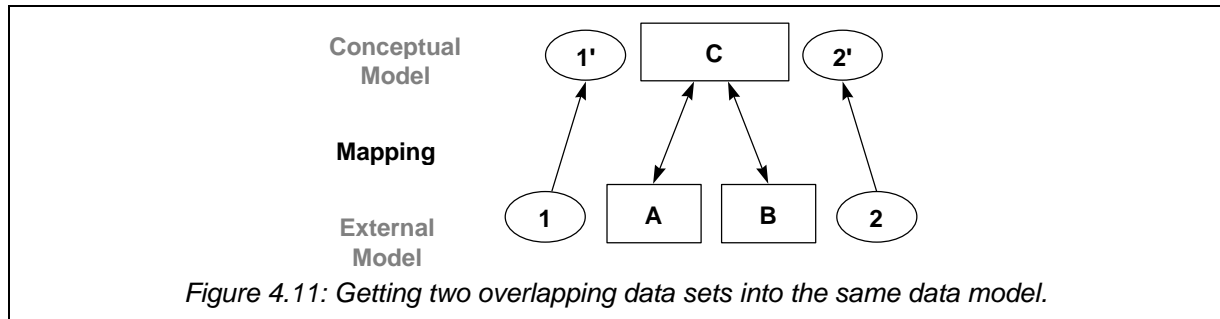


Whether this is the end, depends on the other models to be integrated and their context. Perhaps, for example, a different department has a different primary classification used in a different application model.

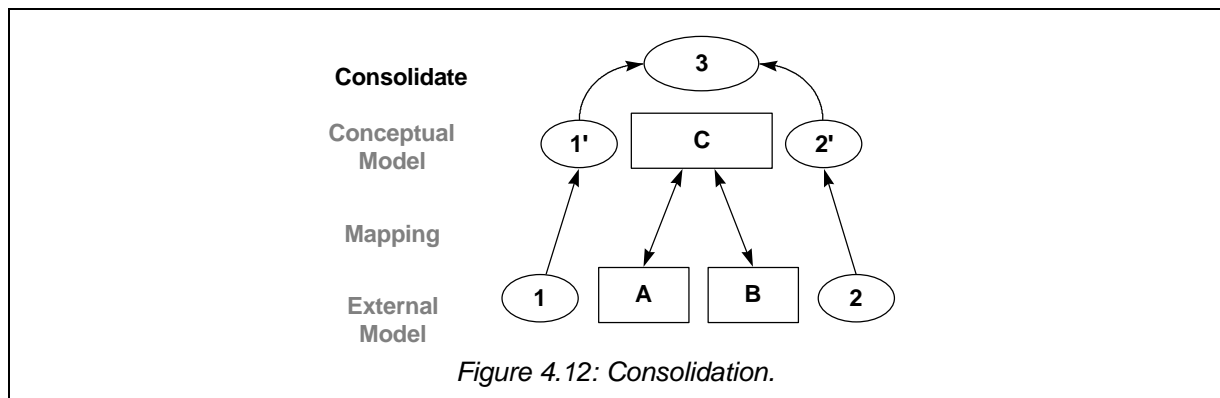
The key point here is that for a two way mapping, at least all the relative semantics must be discovered, and this assumes that the usage of the model conforms to the semantics. If not the semantics of the usage must be discovered too.

4.5 Data Consolidation

Having developed the conceptual data model, and the mappings from (say) two external data models to the conceptual data model. It is now possible to migrate target data sets from the external models to the conceptual data model. The meaning of the data set 1 and 1' is the same, but it is being said using a different data model. I can do the same with data set 2.



However, just because the two data sets are data sets according to the same data model, does not mean that the data is integrated. It is necessary to go through a process of consolidation. In this step it is necessary to identify what data is about the same thing. External identifiers, and other relationships that can be used for identification will be helpful in this. However, it is possible that situations will arise when it is uncertain whether two pieces of data are about the same thing or not, because insufficient care has been taken. In this case human intervention is necessary. The result is shown below.



It should be noted, that in this process, the same object will probably have had different local object identifiers in the different data sets, and that the consolidation process is largely about ensuring that each object has only one internal identifier.

5. Alternative Approaches

In looking at how to satisfy the requirements identified above, there are basically two approaches that can be taken. The first is to change and improve what we have in STEP to satisfy the needs for data integration and sharing across an enterprise and between enterprises. The second is to establish a companion standard to STEP that provides the same capability. However, whichever approach is taken, there is considerable development outlined above which will need to be undertaken. Only the key differences are outlined below.

5.1 Improve STEP

The STEP Generic and Integrated Resources would need considerable change. These changes would need to:

- move from a product data focus to an enterprise data focus,
- move from a data exchange focus, to a data exchange and sharing focus,
- move from being part conceptual model and part a set of templates, to a fully instantiable conceptual model,
- move from identifying the context, to the context being the role one thing plays relative to others.

The advantage of taking this approach is that all the work is carried forward. The disadvantage is that considerable change, and hence cost, would be required by those who had already implemented STEP. On the other hand considerable change seems to be necessary at present anyway.

5.2 Introduce a Companion Standard to STEP

This route involves introducing a companion standard to STEP that deals specifically with data sharing and data integration, including integrating STEP with other standards.

The advantage here is not being constrained by what has gone before, and equally not requiring change to what has gone before. The disadvantage is that if it is not STEP, then STEP will have to be supported in the development of the new standard.

6. Next Steps

1. Identify projects that want to establish a data sharing and data integration standard.
2. Initiate a Preliminary Work Item to provide a focus for work on satisfying requirements for data integration and data sharing as a project under WG10.
3. Establish which strategy is best for achieving data integration and data sharing, extending STEP or developing a companion standard.
4. Develop, and standardise where appropriate, a methodology and architecture for data integration and data sharing standards.

7. References

¹ "Integration of Industrial Data for Exchange Access and Sharing (IIDEAS)" , M.R. West, Shell Information Services Ltd, ISO TC184/SC4/WG10/N85, 1996.

² "Developing High Quality Data Models Volume 1: Principles and Techniques", M.R. West, Shell International Ltd, IC94-033, 1994.

³ "Developing High Quality Data Models - Version 2", M.R. West, EPISTLE, 1996.

⁴ "EPISTLE Framework V2.0", Chris Angus, EPISTLE, 1996.