

**Some thoughts and questions in connection with the conference**

***Open Forum on Metadata Registries,  
Berlin, april 2005<sup>1</sup>***

**(2005-05-09)**

**Stig Berild**

**(Santa Anna IT Research Institute AB)**

---

<sup>1</sup> Rapport framtagen under e-Society II-projektet ([www.skriver.nu/esociety](http://www.skriver.nu/esociety)) vid Santa Anna IT Research Institute ([www.santaanna.se](http://www.santaanna.se)) samt med stöd från Stiftelsen SISU.

## First of all

The Open Forum conference was in many respects informative and nicely accomplished. As a newcomer into the fields of SC32/WG 2 and TC 37 I had the privilege to get a lot of insight into the work performed by the two groups as well as into different metadata application areas. At the same time some thoughts and issues came to my mind, probably as a result of my data modelling background. Some of them are discussed below. The intention is not to criticise but rather to give a view from a perhaps slightly different perspective. Furthermore, please regard the document just as a number of slightly edited notes from the conference listed in arbitrary order.

## 1 ‘Meta’ as a prefix

There has always been a lot of confusion in connection to ‘meta’. I have discussed some of this in a report <http://www.skriver.nu/esociety/archives/Metadata%20engelsk.PDF>.

The recent Open Forum conference added new fuel to the subject. This is my view of ‘meta’ and different levels of ‘meta’:

### 1.1 *The IRDS standard and its levels of abstraction as a starting point*

Some of the confusion can be derived from the ISO standard ISO/IEC 10027:1990 “Information technology -- Information Resource Dictionary System (IRDS) framework” (by the way produced by JTC 1/SC32). This standard was a product of its time about 15 years ago when Case tools to support systems design still had a share of the market and of the general interest. These tools were supposed to manage the modelling data in repositories or Information Resource Dictionaries (IRD) as they were called at the time. The IRDs were databases storing information about all the different aspects and phases to take into consideration when developing applications, preferably database applications.

This information came in the form of conceptual or information models expressing different views of an application during different stages of its development (and sometimes during later stages of its life-cycle). IBM was at the forefront with its AD/Cycle Information Model, for its time – and still – an impressively advanced information model for systems design.

The different types of models (object models, activity diagrams, relational schemas, ...) were the data to be managed. This data belonged to and was accommodated on the *IRD Level*.

Obviously this data had to be managed according to one or several conceptual models as in any other type of application. If UML had existed at that time, UML would have been a candidate for such set of conceptual models. To distinguish these conceptual models from the IRD Level models, the former were said to exist on the *IRD Definition Level*, the “next level up”, if you want.

If we want to share information about IRD Level models and want to be able to share that information no matter which IRD Definition Level conceptual models we may be using (i.e. with possible reference to IRD Definition Level concepts), we need a language capable of

handling both levels of concepts. To be generally applicable that language must include concepts that represent abstractions of the different types of entities that might exist on the IRD Definition Level. Those concepts form a conceptual model for the language. Once again, to distinguish this language conceptual model level from the other two levels of models, it was said to exist on the *IRD Definition Schema Level*.

As an example (please disregard the implementation dependent flavour of it); suppose we have a repository implemented as a relational database and managing activity diagrams and Entity-Relationship models (IRD Level). The content is structured according to some relational schema (IRD Definition Level) based on SQL (IRD Definition Schema Level). This is nothing different from managing information about employees in an employment database except that the information of interest – the Universe of Discourse (UoD) – is different.

To be “complete” the IRDS standard added the *Application Level* as the basic or starting level. This level is supposed to enclose all the information in the application database representing the UoD. All together four levels of abstraction. But again, the operational focus is on the three upper levels, the different IRD associated levels.

Compare with a database storing data about all car models ever produced on earth including a number of relationships among them such as ‘based on’, ‘limited edition of’, ‘same engine as’, ..... An ordinary database application. What about all the millions of individual cars, each of which is related to its specific car model? They are probably not even noticed and should not be taken into consideration given the UoD at hand, even if information about them could be said to exist on the Application Level. The same should be true when operating model data.

In my view, there are always only three abstraction levels to talk about; the information of interest level, the conceptual/information model level, and the modelling language conceptual model. The latter form the conceptual base for the language used for managing the information of interest. These three levels may be applied to any context representing any UoD on any abstraction level. In the context of the IRDS standard the pure Application Level is in fact the IRD Level.

By the way, the Application Level of IRDS is almost always associated only with instances of the created object model (or some ER-model), very seldom, except for those working on execution management issues, with process instances from any processing model. Suppose our UoD was only about state diagrams, activity diagrams, goal models, flow models and the like - not object models. Would the Application Level then ever be brought up - bringing confusion around an envisioned fourth level (or basic/first level, depending on where you start counting)?

## **1.2 Entering ‘meta’ on the scene**

The IRDS standard is not using ‘meta’ to reference different abstraction levels. IRDS has an explicit name for each level with the intent to give meaning to the level within the given context – information resource management. The IRDS standard furthermore tried to approach abstractions by grouping levels in instance – abstraction level pairs, in an effort to limit the confusion concerning abstraction levels. These pairs were Application Level Pair, IRD Level Pair, and IRD Definition Level Pair.

‘Meta’ came into use extensively by OMG during the work with MOF, XMI and other standards. ‘Meta’, ‘meta-meta’ and even ‘meta-meta-meta’ levels came to be discussed. Not surprisingly, more confusion than clarity was established. At each meeting these meta levels had to be explained to the participants in an attempt to clarify, probably with limited success.

Trying to excel in ‘meta levels’ is a pure way to plant confusion. It doesn’t help to name the levels M0 – M3. In the OMG view, meta-data populate the M1 (model) level, meta-meta data the M2 (meta-model) level and meta-meta-meta data the M3 (meta-meta-model) level, though they reformulate this M3 level data as data on the meta-meta model level to avoid three meta in a row.

It doesn’t get better when the OMG view is taken up by the so called metadata community focusing on information about digital resources on the web. Here even the data about a digital resource instance, i.e. on the Application Level (M0), is called meta-data. Consequently data on M3 level should have four ‘meta’ prefixes.

Let’s continue up the levels. Suppose our UoD is conceptual modelling languages, perhaps with the purpose to compare them concerning expressive power or to define possible concept mapping between them. Using the OMG level naming, this information is on level M2 modelled by some conceptual model on M3 and finally using modelling language concepts expressed on a new level M4. We now have five modelling levels, all prefixed by different repetitions of meta.

In the military and also in some other application areas, there is often a need to add information to messages, to sets of values and even to individual values or data element instances. Information of interest might be the exact time the value was delivered or preferably when the observation behind the value was made, by whom, under which conditions, the probability for its correctness, and other types of information.

One could say that we now reach the M-1 level or at least adding one more level to the other five somewhere, perhaps as a start of a new side track level hierarchy. (In a way this level is close to the one dealing with information about digital resources on the web.) This view is probably the closest to the common definition of metadata as “data about data”. The other meta levels deal with different abstraction levels, which is something else.

Great, isn’t it..... Just continue up and down the levels depending on the context of interest.

Talking about this definition of metadata; A book, even in digital format is in my view primarily something to be read and hopefully have a good time with – not data, in the same way that a car is something used for transportation or as a hobby rather than a piece of steel. Their roles are in most cases what’s of interest, not their representations.

Another example; We are seldom interested in humans as just a piece of blood and flesh. Even a surgeon hopefully sees patients to be cured, not just as neutral bodies. Humans are involved in thousands of roles in their active live. These roles are of interest to share information about. (Even when we study physiology we do it for a reason, with a purpose – the role again.) The same is true for the different things existing on the web.

In the same way; to model modelling concepts is nothing inherently different from modelling any other things real or imagined. To model values, messages, value sets, subsets of a database or other combinations is nothing different either from modelling any other things real or imagined. And so on.

The UoD is determining. Meta is relative – if you still want to use the term.

### **1.3 *Mixing abstraction levels?***

It is sometimes, even within a specific UoD, hard to distinguish between different abstraction levels.

Take a UoD involving cars, for example. A database for a used car company might include information about each car as well as information about car models. A specific car is described by its registration number, the asked price, number of miles driven, number of owners as well as a reference to the car model information. Is that car model information data or metadata? Most of us would say data. (By the way, ‘information’ and ‘data’ are used in a rather sloppy way in this document.)

Car model information includes name, model year, horsepower and mileage. Car model information is generic for each specific car pointing at it. Is the car model information then metadata? Is a conceptual model defining the types of data elements to be used to describe individual cars metadata? What about the conceptual model for car model information? Is it the same metadata because the information about individual cars and car model information is in the same database?

In my view, the individual car and car model information both exist on the Application Level if this is what’s intended with the UoD. Then the conceptual model on the IRD Level has to incorporate all abstracted aspects of this information.

By the way, even if only individual cars were stored in the database, car model information would not be part of the IRD Level because this level is not about describing car models but about what types of information to be used to describe cars. Probably car might be the concept to use and perhaps Honda Car as a specialization (or even Honda Civic Car) in case these specific types of cars are documented partly by their own description elements.

But isn’t car model information at least a little more meta than car information .....? ☺

Finally, is the database schema to be seen as metadata or is just the implementation independent conceptual model allowed to take this role? Why not call them schema (to mean an implementation dependent model) and conceptual model respectively and leave ‘meta’ out of it?

### **1.4 *Meta and digital resources***

Now, let us move the same discussion to an auction house specializing in books. We have individual books of interest as well as general information about books and so on in the same way as with used cars. Is there any difference in what should be considered data or metadata?

No, why should it? It is all data. Suppose this auction house also works with digital books. Is the prefix 'meta' now moving closer to be used? No, why should it?

Why is it that things of interest in digital or printed form often are seen as so essentially different from all other things that information about it has to have its own name (metadata)? I am confused. Why not just data?

### **1.5 Back to meta and cars**

The car database is later exposed on the web. To give interested car buyers an attractive interface the database is extended to include information about the information available for description of car models. This is found to be important as the number of types of data elements differs a lot between different model years and between different brands. Users might for instance only be interested in cars having a car model description including horsepower.

Is this information about information to be classified as data, metadata or perhaps meta-meta data (meta-meta because it is a conceptual model about car model information which in turn gives generic information about individual cars)? Perhaps only 0.5 or 1.5 meta because this is not a conceptual model about an information base as a whole but more specifically about a number of subsets of the information base, in fact about each individual object on that level in the information base?

Again, my view on this is that all the information in the information base reflects a context of interest for some user group to share information about (UoD. It is all data. The allowed content in that information base is expressed by the user group in the form of a conceptual model (some may want to call it metadata). On the other hand, if cars of a specific car model is (partly) described differently than cars of another car model, this is a subject for an IRD Level conceptual model to describe in an appropriate specialization structure.

## **2 Conceptual modelling languages**

### **2.1 Background**

The term 'conceptual model' has been around for decades. A number of books were written about conceptual modelling languages and conceptual modelling methods in the 1970s and 1980s.

Different types of binary modelling languages were defined in the first part of the 1970s. They have recently been reinvented by World Wide Web Consortium (W3C) in Resource Description Framework (RDF), as it seems without knowledge or interest in the former. A new wheel again ....

A number of different types of so called Entity-Relationship languages were developed and competing in their ability to express things, starting with the ER-model by Peter Chen in 1976. NIAM is one good example of a more advanced language. In the late 80s and the first part of the 90s came the different so called guru based models, in part reinventing inferior alternatives of already existing languages. They were to a great extent very similar even if the gurus at the time didn't accept that fact. However, OMG later on pushed them to come up

with one unified language, UML. UML also incorporates facilities from those object modelling languages that has been around for quite some time in the programming world.

Recently OWL has been defined by W3C as a language for the vision of a semantic web. OWL is a mix of the old binary and ER-modelling languages added with a flavour from the AI field. OWL (as well as RDF) is heavily based on the use of URIs as representation mechanisms. This is almost taken as an axiom, though this approach indeed can be debatable.

## **2.2 Application areas**

Things of interest to share information about (UoD) might be anything. It might be physical things like cars, humans, lakes, roads, books, ... It might be business related things like orders, invoices, customers (a role to be distinguished from a human), companies, .... It might be feelings, wishes, thoughts, visions, .... It might be specifications of systems under development or under execution. And many other things - some of them mentioned above. They may all be of interest as individuals or as generic entities. Some of them may come in different flavours and in different representations.

This is for instance the case with entities that might be printed on paper. Many of them might also be represented digitally in different formats. When it comes to written text, many different languages and formats are available. Sometimes these digital entities represent something else (for instance a picture of my car used as a designation of my car). Sometimes they are just something of interest in itself (for instance a picture of my car as a picture of a car).

Add to this different purposes or contexts, given the same area of interest. For instance, some contexts deal only with general information while others deal with very detailed information. Contexts differ also depending on perspective. All together the UoD represent a specific combination of all these different aspects. Add also the fact that different information sharing groups have different views or preferences.

More on this subject can be found in  
<http://www.skriver.nu/esociety/archives/Conceptual%20modeling.PDF>.

## **2.3 One or many languages?**

An obvious conclusion from the last section is that one language is not enough for every UoD and each information sharing group. Each language has its own capability in expressing semantics. Some languages work with a number of different concepts, while others (e.g. binary languages) only have a few. They all have their weak and strong parts. Most of them have probably found their niches. These niches are based on a combination of the different aspects mentioned above. The application area at hand is only one parameter to be considered. Most of the languages are in fact rather application independent and thus freely applicable on any UoD as long as its capabilities are enough for the purpose.

But, among other things for interoperability reasons, it is important not to develop a new language “just for fun” or to avoid spending time and interest in looking for and evaluating other already existing languages. Defining a modelling language is not an easy task, especially if the intent is to include new types of semantic constructs not found in other

languages. There is also a tendency to arbitrarily view the own needs as unique. This is often more a rule than an exception. We already have more languages than we really need.

Some of the modelling languages in existence today could probably be merged together. What we don't need are a number of different languages differentiating only in the terms used for concepts or where one language is a pure subset of the former. Subsets can and should be specified based on the superset rather than as a separate language.

The first and obvious step in choosing a language is to define the UoD of interest and to do that in a way so that all participants have a chance to understand and accept it. It doesn't mean that this UoD has to be static for all times. However, without an agreed upon, interpretable UoD specification, things soon get messy with endless discussions and different opinions and preferences, badly influencing the productivity of the work to be performed.

Next, look for an applicable modelling language. If it exists, recommend it and use it. Point. If such a language is not available, find languages that as much as possible resemble what is needed. Try to get involved in revising one or several of these languages according to own preferences. As a last resort do the language specification yourself and do it starting from an existing one. Expand or revise its expressiveness as needed, at all times looking for influences from other standard languages, de facto standard languages and broadly used languages. Keep interoperability in mind at all times.

The language so defined is possibly useful also for other application areas following the hypothesis, once again, that most modelling languages are application area independent.

How is the metadata community acting in this respect? For start, is the UoD for the new or revised directions (e.g. semantic interoperability) defined?

Or, should we just heavily compete and let the market and other influencing factors decide along the way? Are we sure convergence will ever occur? Is it worth the extra time this takes and all the confusion? If so, do we need the standardization efforts? Are we sure to end up with the most useful languages? ..... I am somewhat ambivalent.

## **2.4 Modelling languages for TC 37 and SC 32?**

Several presentations at the conference talked about or gave references to modelling languages - often UML. At the same time a number of modelling concepts, non-existing in traditional modelling languages, were presented. In fact, most often only terms were mentioned. This is fine as long as a listener is guided with information about the approached UoD and as long as the concept is defined as part of a conceptual model. Reference to concepts/terms in known conceptual modelling languages would also help, especially if similarities and differences are discussed.

SC 32 is doing a good job in the ISO-11179 standard. The UoD is clearly defined. So are the conceptual models. The standard even states what modelling language it is using (UML) for the conceptual models to help the reader interpret the different graphical figures. That's why it is easy to understand the meaning behind the concepts in the models. Everything is explicit and clear. Debating the models can always be performed in a productive way. No hiding behind vague expressions in non-defined or perhaps even non-existing modelling languages.

SC32 has taken a strong position towards modelling constructs for the explicit distinction between concepts and values (Ogden's triangle). This distinction is often lacking in modelling languages. SC32 has in this respect done some piece of pioneering work. The next step is to incorporate these modelling constructs into "ordinary" modelling languages. An important task that should be performed without any specific focus on metadata as such.

Ogden's triangle by the way; The triangle explains the need to keep concepts and symbols/terms as distinct entities. It has been around as guidance for modellers for a long time now. However, it doesn't take into account the possibility of indirect designations. A concept may be semantically defined by its relationships to other concepts or by a combination of concepts and properties. No symbol or only parts of the symbols building the designation are in that case directly related to the concept. The designation is rather the symbols of indirect concepts participating in the definition. This is important for modelling languages to cover and for modellers to keep in mind.

The ISO 1087 view of a designation as "representation of a concept by a sign which denotes it" is in that respect a simplification. Take 'marriage'. It is designated by two individuals as concepts each designated by a Social Security Number in combination with a Marriage Date. Each Ssno designates an individual. The marriage is designated indirectly by the two individuals as concept entities in combination with the direct designation component Marriage Date.

## **2.5 An approach?**

Thus, one important work to be performed is to add the distinction between concepts and symbols in ordinary modelling languages, i.e. a mix of some ER-model and the work of SC 32 in ISO 11179.

A subgroup within the Swedish Standards Institute 1991 defined a conceptual modelling language (Stanli) for modelling concepts, primarily to be used in areas dealing with geographical information management and interchange. As it turned out, this language was not at all constrained to geographical information but was generally applicable for conceptual modelling. The language came to be used and is used extensively in Sweden for conceptual modelling. The main purpose was on conceptual modelling (as a way to grasp and trim concepts within organizations) rather than creating data models during database systems development.

This is perhaps one reason for its capability to distinguish between concept and symbol. It is now an old language (defined a number of years before UML) but still well suited for its purpose. Which by the way is no surprise. The language was based on a long tradition of knowledge sharing and debates about good and bad modelling languages features that took place at international conferences, in magazines and other forums in the 1980s.

Unfortunately the Stanli specification only exists in the Swedish language, but I have made a translation of the conceptual model for the language into English. It is shown in the figure below. Hopefully most of the model is interpretable by the graphics. It is specified using its own concepts.

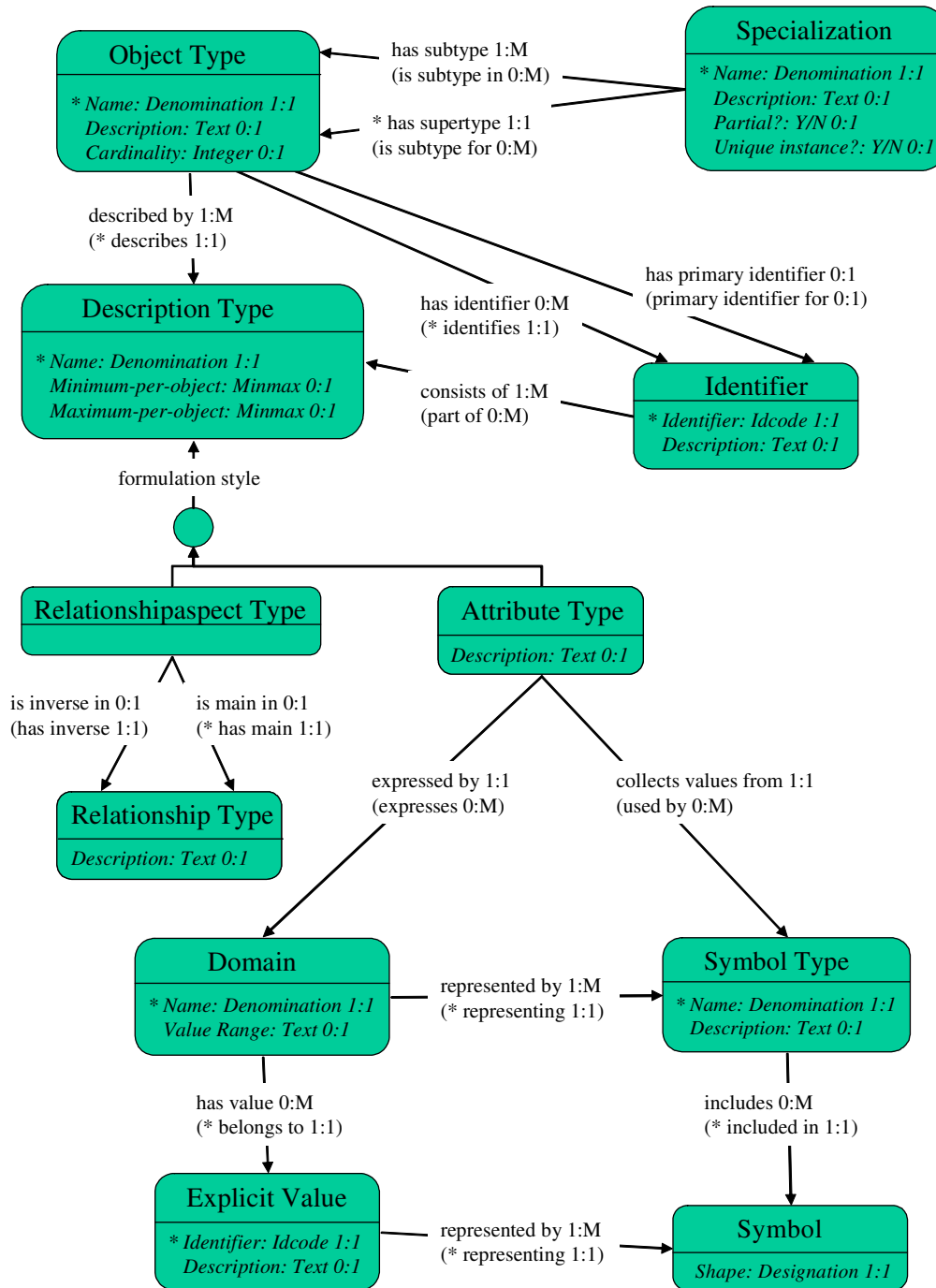


Figure 1

In Stanli the combination of asterisk marked entities together form a unique reference ('primary identifier') for an Object Type. For instance, with reference to figure 1 as a conceptual model, a 'Symbol Type' is referenced by a combination of its 'Name' and the reference of the 'Domain' it is 'representing'. A 'Description Type' is referenced by its 'Name' together with the reference for the 'Object Type' it 'describes'. And so forth.

Stanli lacks concepts to express and map different enumeration detail levels, for instance the LOM standard domain ‘Typical Age Range’ with mapping between three different Symbol Type range subdivisions ‘standard school level’ (primary, secondary, senior, college, ...), ‘numeric age range’ (1-4, 5-9, 10-15, ...) and ‘numeric age’ (1,2,3,4,5, .....). Also completeness rules in Domain – Symbol Type relationships are lacking. On the other hand, no other language seems to have that capability either. Unfortunately.

### **3 A conceptual model for the TC 37 modelling concepts?**

Interoperability in combination with content/information management, conceptual models, modelling languages and even metadata seem to be some of the connecting points between the two standards groups (SC 32/WG2 and TC 37). Are they in fact overlapping in a way that integration efforts would be a possible next step? Are the differences merely caused by their origin in different application areas? Or are they in fact currently not approaching interoperability (UoD)?

The TC 37 terminology work seems to focus more on terms and their relationships bringing in concepts more as a constituent in the definitions while the SC 32/WG 2 metadata management work focuses on concepts and their relationships and bringing in terms for definition completeness. If so, maybe the overlap is fairly minor. Of which may follow that integration is not that hard, if the two communities believe it is worthwhile?

TC 37 is using a set of concepts and/or terms from linguistics. This is not any problem per se, rather the opposite, I assume. However, a conference participant not familiar with the field would have a much better chance to understand the work performed if the UoD was clearly stated for each TC 37 WG and if the modelling concepts/terms used were defined in a conceptual model.

Take a feasible conceptual modelling language for this conceptual modelling. UML or MOF would do. Probably also RDF. Use the language to define your modelling concepts and the meanings behind them will be delivered pure and simple. If that work has already been performed, I would be eager to take part of its outcome.

A TC 37 conceptual model should incorporate all the modelling concepts used by TC 37, e.g. term, (superordinate, comprehensive, partitive, ..) concept, context, content entity, conceptual entity, designation, property, characteristic, (hierarchic, associative, partitive, generic, ...) relation, synonym, mononym, .....

If interoperability is part of the focus, this model has to be developed in close contact with the intended users and expressed in a way that has a chance to get broad acceptance. As a consequence the choice of language is also of vital importance. Given this conceptual model a much more productive discussion between interested parties can take place. We often state the importance of capturing concepts, terms and their relationships into models as a prerequisite for information sharing and management in different domains. What about “practice what one preaches”?

For the moment the concepts used by the groups seem to differ a lot, at least the different terms used to designate the concepts. In fact, almost no term can be found in both groups.

With conceptual models defined by the two groups, comparisons like the following one from a TC 37 presentation will hopefully disappear.

First the term (concept?) definitions:

*11179: data element: unit of data for which the definition, identification, representation and Permissible Values are specified by means of a set of attributes*

*TC 37: data category: unit used to store, retrieve, and manipulate minimal items of information in digital resources; these are viewed as virtual objects*

Then the imagined difference:

*Difference in point of view: TC 37: what a data element is in the real world; 11179: what a data element is in a metadata registry*

Notice the data category definition as a ‘unit to store, retrieve, manipulate information in digital resources’. Is that something in the real world? Furthermore ‘unit used to store items of information ..’ indicates that this unit is some type of operational mechanism working with information items. Certainly not the intention but it indicates the risk of having only free text definitions of concepts. And, what is a ‘virtual object’? It must be something well defined as it is used in defining something else. I am waiting for the UoD and the conceptual model.

Also, data category seems to come in different flavours: Complex, simple, closed, lexical, terminological, linguistic, .... Are they pure specializations or just similar meanings? Why not put up a conceptual model of all the concepts and their interrelationships, if any? Are there an infinite number of data categories or just a predefined set of them? If the latter, who has the defining authority? Do they have an existence by their own right or just as part of some other linguistic entity or as part of some application area or ...?

Another example of some possible confusion: ‘Concept system’ (TC 37) and ‘conceptual model’ (used by many) seem to be almost synonyms. Are they? If they are, one early step in the joint SC 32/TC 37 work would be to clearly state the fact and bring it out to the world. An even better outcome would be if the two groups could agree on just one of the terms. Both ‘system’ and ‘model’ is about some whole built from related parts in some way. ‘Model’ also means abstraction of something. This is important as the concepts of interest are abstractions of entities in the UoD. So ‘conceptual model’ is my favourite. If, on the other hand, there is a difference, it should be clearly expressed for the interested community to notice.

By the way; Information Interoperability, Information Management, Conceptual Modelling, etc. may very well be rather different UoDs requiring their own sets of concepts and conceptual models. Perhaps also their specific conceptual modelling languages.

Tough job waiting.

Even if these two groups at some time come up with a common language, the same problem exists but on a broader scale. The combined group has to argue with the rest of the modelling world about the necessity and uniqueness of the own work, as well as the necessity for a new modelling language (if that is the intention). This argumentation will certainly have to be articulated; especially if there is a general agreement that metadata is nothing but data in a special context. By the way,, even if this agreement is not on the table. Again, keep interoperability in mind.

Adding mapping specifications to and from an own language is a necessity when the language already exists, but can never be a motivation for allowing own language development. Mapping is always hard to define in a way so that semantics is not disappearing in one or both mapping directions. Or, if 100% mapping can be guaranteed, this is the best proof only one of the languages should be in use. (I am talking about mapping on the concept level, not the term level.)

## 4 Conceptual models

### 4.1 *Conceptual models versus conceptual modelling languages standards*

Difference should be made between activities focused on standardizing conceptual models (e.g. Dublin Core, LOM) and activities focused on standardizing modelling languages (e.g. Express, UML, RDF, OWL, ...). The former belong to what the IRDS standard calls the IRD Level (M1 in OMG terms), the latter belong to the IRD Definition Level (M2 in OMG terms). We sometimes seem to refer to Dublin Core at the same time as for instance UML indicating that they are just two solutions to the same type of problem. That is dangerous. Dublin Core could easily be defined using UML, NIAM or some other ER-modelling language while the opposite is not at all true.

However, this distinction is not always obvious following the discussion above that the UoD at hand should be guiding what should be part of which model level.

### 4.2 *Conceptual model standards need conceptual modelling languages*

The standards efforts, focussing on conceptual models at the IRD Level, should always start their work by formally identifying the modelling language to be used for the purpose. The chosen language should be based on the specific needs of the current context and purpose (UoD). In the general case the chosen languages should be based on the need for semantic expressiveness rather than based on a specific view within the specific application domain. A presentation of the language is well suited to guide a reader when approaching the model features. Less risk for misunderstandings.

For example, the Dublin Core and LOM (Learning Object Metadata) standards were initially based on a modelling language just offering a list of attributes for some type of entity. LOM also includes description (data element) hierarchies and categories. See example in figure 2.

- 4 Technical
  - 4.1 Format
  - 4.2 Size
  - 4.3 Location
  - 4.4 Requirement
    - 4.4.1 OrComposite
      - 4.4.1.1 Type
      - 4.4.1.2 Name
      - 4.4.1.3 Minimum Version
      - 4.4.1.4 Maximum Version
  - 4.5 Installation Remarks
  - 4.6 Other Platform Requirements
  - 4.7 Duration

Figure 2

The specification rules for LOM could be expressed as a conceptual model like the one in figure 3, using Stanli notation.

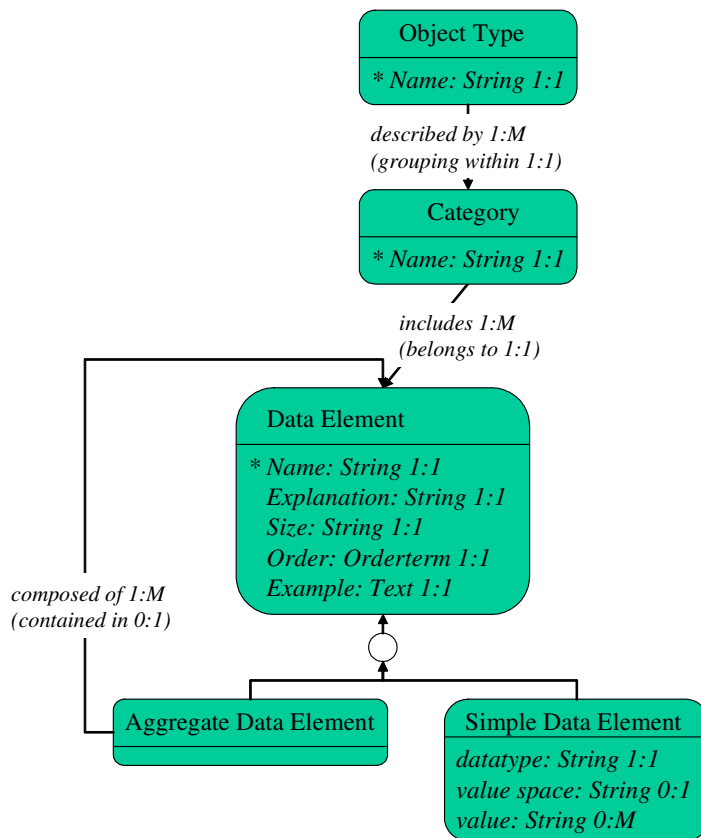


Figure 3

Maybe this is to make things more formal than needed but I still believe this formal representation has its advantages for ease of understanding. Probably even more so when it comes to required extensions of the standard. An initially simple model often soon becomes more complicated. Extensions under full control and more concrete discussions will result.

Also, having a formal definition makes it easier to compare and evaluate other model languages if and when extensions and/or mappings are of interest.

By the way, Dublin Core is also more than a list of elements. For instance, the standard includes specification of the attributes to be used to describe those elements. Also, qualifiers may be defined, among other things.

Perhaps some day a more general conceptual modelling language would be a better choice instead of adding more features to the own one. Not the least for interoperability reasons.

SC 32 has taken that approach by using UML in the ISO 11179 standard for specification of its conceptual model for managing terms and concepts. Sure, DC and LOM have their application areas on the Application Level while ISO 11179 data is on the IRD Level. But that doesn't change the fact that one is using a standard conceptual modelling language for its UoD while the other two have chosen to define their own structuring rules for their UoD. For good reasons?

By the way; ISO 11179 may equally well be at the IRD Level, abstracting data on the Application Level if the UoD is Information Management rather than a conceptual modelling language to be used for database applications, where the distinction between concept and term is put at the forefront in the guiding conceptual model (and in the database).

This becomes obvious when dealing with the Administration and identification conceptual model as part of ISO 11179. It has no chance to exist on the IRD Definition Model if we accept that UoD is defining the abstraction levels (the dynamic approach). However, based on the static IRDS way of looking at things, it might very well exist at the IRD Definition Level together with all other modelling languages defining interesting aspects of some system under development or in production.

Another thing; why has OMG chosen to develop a separate language for specification of conceptual models (being data) when they have UML available “next door”? Are there unique requirements to care for? Hopefully. Or the other way around; If, in the work of specifying conceptual models, new modelling concepts are found to be needed, why not add them to UML? Or, is the reason that UML also includes an object model flavour (expressing behaviour) of no interest in modelling models? Or .....?

### ***4.3 One standard conceptual model for digital resources?***

There seem to be some conception that it is possible to specify one and only conceptual model for digital resources. This is in my understanding a mistake. There might be many different needs to express information views on and about digital resources, each purpose requiring its specific capabilities. Who wants to argue that we only need one model to specify views of cars or persons? This is why for instance LOM has failed to be generally useful. It doesn't take into account different UoDs.

Working with subsets is not the solution as user groups might want partly or completely other concepts to work with for their specific UoD. This is furthermore the reason why for instance qualifiers have been added to Dublin Core, I think. More importantly, these approaches lack recognition for the need to start the specification by specifying the modelling language to be

used. If that “platform” is weak or arbitrary even the best of intentions about the models to be created using this language will fail.

## 5 Semantic Interoperability

Semantic Interoperability is a popular term for something that different people probably put different concepts behind. This is natural when it comes to new trends. But in order to discuss different views and approaches we need to express what we mean in a way that at least give some hints to what meaning we put behind the term. Which is important because interoperability as a capacity in itself is becoming more and more important. Also, this is an area of urgent need for standards.

I have put some thoughts into this area. If you are interested, a report in English gives an overview of a soft architecture based on a binary modelling approach. It can be found at <http://www.skriver.nu/esociety/ba.pdf> . Binary modelling languages are easy to understand and work with – an important aspect in a global environment. Furthermore they easily open up the freedom to use other more flexible means of messaging than those based on XML.

## 6 Registry or Repository

What is the real difference between the concepts behind the terms Registry and Repository? Are they two different terms for the same concept? Is the difference mainly that they manage different types of contents, i.e. being two specializations of the concept we usually designate ‘database’? Both manage information about things. Some years ago the term Repository was used for storage of application specifications. Case tools were the main users.

For instance, is a complete specification of an application made in Rational Rose a Registry or Repository? It includes among other things an object model that can be used as a schema or model for data in an object database. Suppose the data in that object database also include message specifications expressed in the XML Schema language. Is it then no longer a database or repository because the content is a model of a message type? Especially if the message type is the only message type of interest and as such at the same time can be considered to express the conceptual model of the actual application (sending, receiving and storing messages of the type)? Suppose the purpose was only to use a digital storage (database) for testing and discussing different message types among a group of designers?

Please guide me on the difference and the reason for keeping the difference, if any.

## 7 The term “Knowledge”

TC 37 is using the term Knowledge in a number of combinations, e.g. ‘Knowledge Engineering’, ‘Knowledge Representation’, ‘Knowledge Ordering’, ‘Knowledge Management’, The term Knowledge is surrounded by a number of different meanings all the way back to Protagoras and Platon. This indicates that the use of the term in specific contexts has to be supplemented by a definition of the intended meaning to avoid misinterpretations.

Knowledge is in my view something closely connected to humans, something created during processing of information in our brains and digesting signals through our senses. I can't see how a computer program generates knowledge. It operates on data, perhaps generating other data. So please give me some explanation of the concepts behind the terms "Knowledge engineering" and others.

As professionals working in the terminology field we should be careful in our use of terms and even more careful in creating new terms without explaining them in a way that lowers the risk for misinterpretations. This is especially important when talking to people in other fields of expertise.

Someone might have knowledge but that doesn't mean that the symbols used by this person in expressing that knowledge in itself is knowledge. The expressed symbols are data or designations. Or?

By the way, "lexicographical data" sounds interesting. Is it a synonym for morphology? Does it mean something else? Something more? Anything being a rule for forming combinations of words? Please, explain the meaning and the rationale for it or give a reference to where that information exists.

## 8 Bargmeyer's apple.

Fun example. And fun discussion. A few thoughts, though:

An easy start is to model this UoD as figure 4 (using the Stanli language). One reason for the use of PLU codes is to link a price per volume or weight to it.

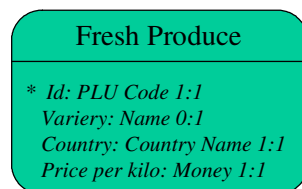


Figure 4

But this is a simplification, and in fact wrong. The label of the apple is not identifying the real individual apple but an entity being a specific type of apple produced in Canada. This became evident by the possibility to move the label around on any apple of the same kind produced in Canada. By the way, do we really know if the individual apple had Canada as origin. Or the thing labelled, indeed is an apple? Could be put on a cucumber. ☺

Anyway, the same PLU Code can be used for fresh produce coming from different countries or different districts. Each such delivery may be of interest to keep separate, often to allow different pricing. The model is modified accordingly.

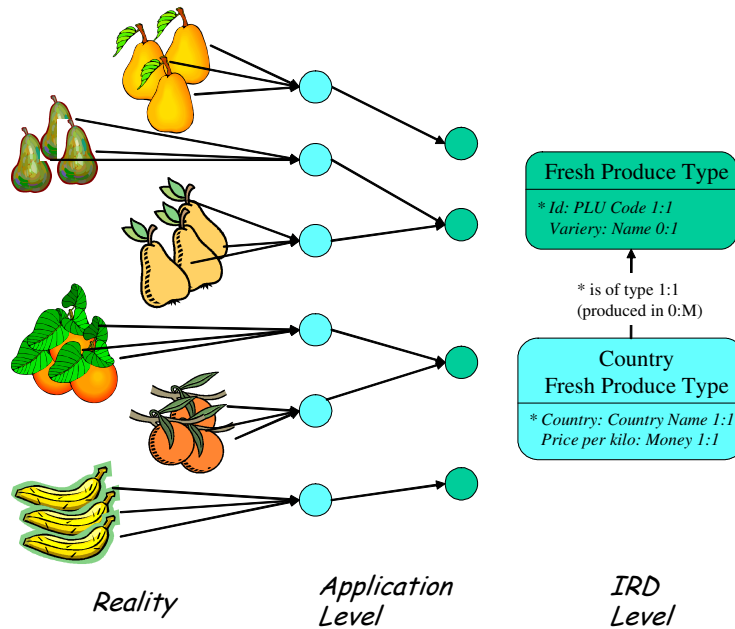


Figure 5

Still, this is not right. The label is only attached to each individual fruit to make it easier for the cashier at the register. Just put the bag of fruit on the balance and enter the code (and perhaps the country?) to get the right price for the bag. A label on the bag alone would do the same job. Figure 6 gives a more accurate view of the situation.

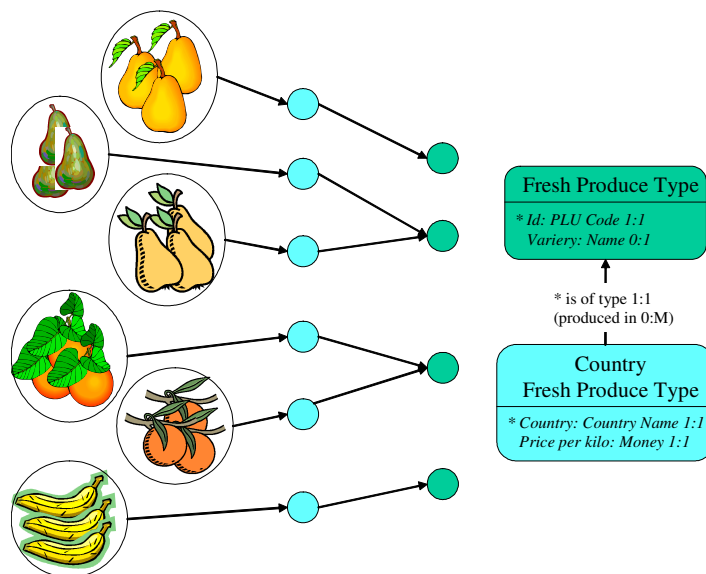


Figure 6

Some PLU codes are reserved for each retailer's own use. One such use could be to identify a unique instance of fresh produce in order to give it a unique price at the store. A store has two huge pumpkins for sale. To sell them for the ordinary kilo price would make them extremely

expensive. The store is therefore giving each a unique PLU code (and a unique price). See figure 7.

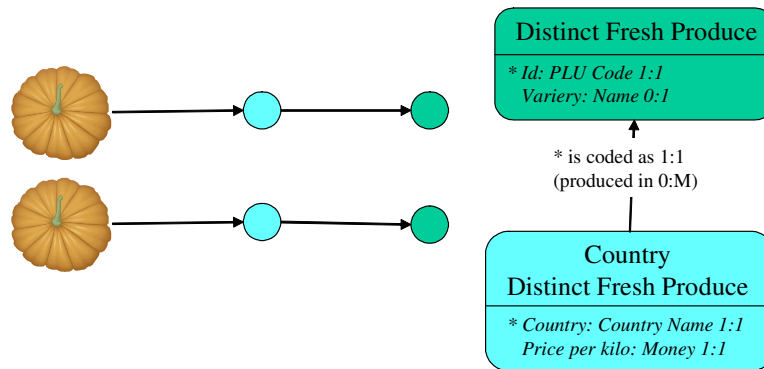


Figure 7

The model is given some new names to reflect the new use of the codes and partly new meaning of the concepts behind these terms.

But the PLU codes are enough for this purpose. Country origin is not of interest. Figure 8 is better.

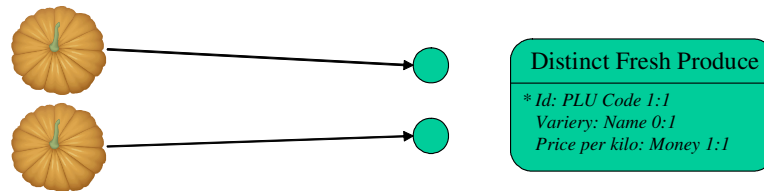


Figure 8

Suppose we have a store selling only very unique and expensive types of fruits. They want to keep track of each individual fruit it sells. One reason is to keep track of the age of the fruit to guarantee its freshness when sold. However each fruit of the same origin and type has the same price per weight.

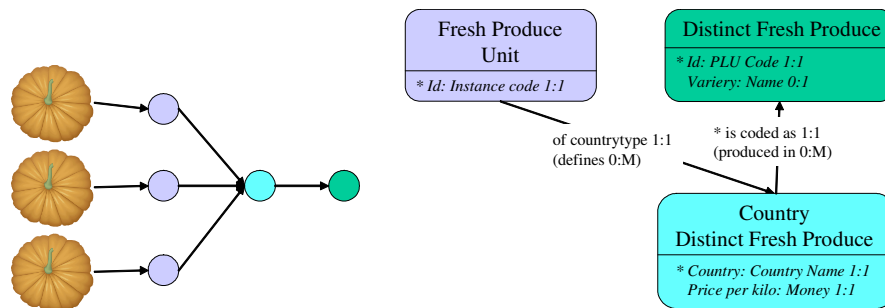


Figure 9

Now the Instance code has to be on each label while the PLU Code + Country name gets the price.

Suppose there is a store that has a mix of all these aspects to take care of. How to integrate them into one model? Who said modelling is easy?

By the way, aren't Distinct Fresh Produce and Country District Fresh Produce on a slightly more meta level than the Fresh Produce Unit? ☺