FORMALIZING TYPES WITH ULTIMATE CLOSURE FOR MIDDLEWARE TOOLS IN INFORMATION SYSTEMS ENGINEERING

Nick Rossiter
School of Informatics, Northumbria University, NE1 8ST, UK Email: nick.rossiter@unn.ac.uk

David Nelson
CAT, University of Sunderland, SR6 0DD, UK Email: d.a.nelson@sunderland.ac.uk

Michael Heather
Northumbria University, NE1 8ST, UK Email: m.heather@unn.ac.uk

Key words: universal typing, reference models, category theory, meta objects

Abstract: A definition of types in an information system is given from real-world abstractions through data constructs, schema and definitions to physical data values. Category theory suggests that four levels are sufficient to provide ultimate closure for computational types to construct information systems. Examples of information systems are examined in terms of the four-level architecture including IRDS, the Grid, the semantic web and MOF/MDA.

1 INTRODUCTION

Interoperability is still a major problem in information systems. Most achievements have been with systems using a similar model or paradigm. Where heterogeneous systems are involved, progress has required much manual adjustment to mappings. Recently the development of the Grid has exposed the great difficulty of employing data held in formal database systems as opposed to operating system files (Watson, 2002). Reference models appear to offer the way forward as they provide four levels at which to address data, rather than the usual three in knowledge based systems today.

For a detailed discussion on interoperability and reference models the reader is referred to a previous paper (Rossiter, Nelson and Heather, 2001). This paper formalises the four-level architecture and discusses its use in various types of information system

2 FUNDAMENTAL LEVELS

The four fundamental levels were described in an earlier paper (Rossiter, Nelson and Heather, 2001) as the Information Resource Dictionary System IRDS. Our interpretation has been modified as shown in Figure 1 and is now approach neutral. To match the design process more closely, horizontal mappings have been added as follows.

Within level 3 there are horizontal mappings between the Design and Syntactic schema. The mapping of type *Realise* takes a diagrammatic notation and converts it into an underlying syntactical model. The mapping of type *Diagram* takes a syntactic model and converts it into a corresponding diagram. *Realise* is a standard function of many design tools; *Diagram* is in effect reverse engineering and is not so easy to achieve as some design information may have been made less explicit through the mapping *Realise*.

Within level 4 there are horizontal mappings between *Stored* and *Named*. The mapping of type *Place* takes a named value and places it on disk in storage format. The mapping of type *Name* associates a stored value with a name.

The four levels described above give the basis for relating heterogeneous types across platform systems, that is systems based on different paradigms. While there is only one instance of the top level type (concepts), this level is extensible and new concepts and abstractions can be added as desired. From the point of view of information systems, the four-levels approach provides the ability to run an organization with many different paradigms all integrated through the type of structure shown in Figure 1. The critical mapping is type *Platform*, that is the arrow from Con-

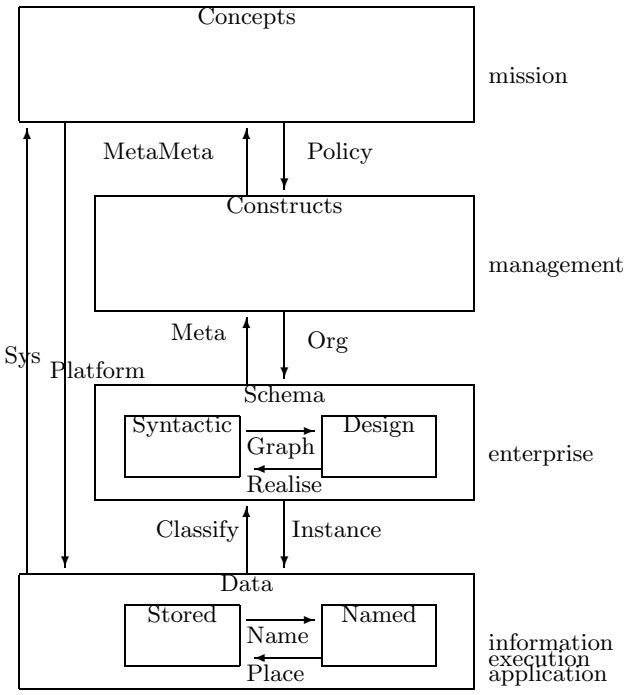

Figure 1: Interpretation of Fundamental Levels informally

cepts to Data, relating concepts to values. By determining this mapping for all types of system, the problems arising in re-engineering are avoided to some extent as all types of approach to information systems can be run in an integrated fashion.

The next task is to formalize the diagram in Figure 1 so that a sound scientific basis can be developed for this approach to the handling of heterogeneous systems. Initially only the vertical arrows in Figure 1 are considered. The horizontal arrows within Schema and Data are considered later.

3 FORMALIZING LEVEL TYPES

Category theory provides a universal construction for formalizing information systems with rigorous typing. It is this uniqueness that provides the universality to form the basis of a general consistent system. An example is now given for a prototype information system focusing on the aspect of a cross-platform system as a heterogeneous distributed database relying on the categorical product construct as a data-type model (Nelson and Rossiter, 1996). In this approach, each class definition can be identified as a collection of arrows (functions) forming a category **SCHEMA** and each family of object values conforming to a particular class definition as a category **DATA**. The type mapping from the class definition to object values is made by a functor *Instance* which enforces the various constraints specified in **SCHEMA**. Category **SCHEMA** is the intension corresponding to the third level type and **DATA** is the extension corresponding to the fourth level type.

It is relatively straight-forward to extend the schema/data two-level structures in a universal manner to handle the four fundamental levels. In categorial terms each of the four levels is defined as a category (i.e. a type). Between each level there is a higher-order function, a functor, which ensures that certain consistency requirements are met in the mapping between the source and target categories. The abstractions level (top) is a category **CONCEPTS** which defines the various abstractions available for modelling real-world data. The next level is a category **CONSTRUCTS** defining the various construction facilities available for representing abstractions and data in a particular system. There is therefore, for one instance of **CONCEPTS**, many instances of **CONSTRUCTS**, one for each type of system.

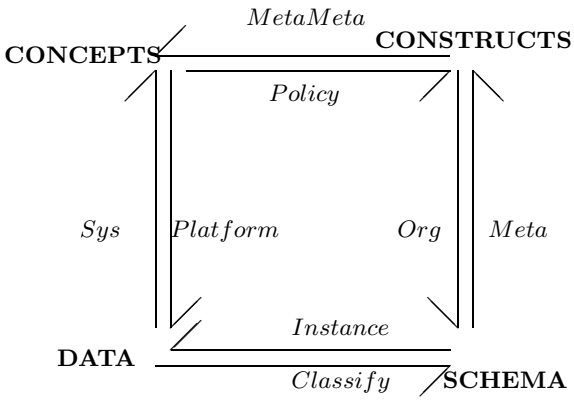

Figure 2: Four Levels in Functorial Terms

The data functor (level pair) type change *Policy* maps target objects and arrows in the category **CONCEPTS** to image objects in the category **CONSTRUCTS** for each type of system. This mapping provides at the meta-meta level the data for each kind of system, that is to say how each abstraction is to be represented. We also label the functor pair *Org* relating for each system the constructions in **CONSTRUCTS** with the names in a particular application in

SCHEMA. Combining these new constructions with the product ones above gives the direct and universal representation of the four levels shown in Figure 2.

The remaining functors $MetaMeta$, $Meta$ and $Classify$ are the duals of $Policy$, Org and $Instance$ respectively. It will be noted that in Figure 2 all the mappings are two-way and that two compositions emerge. Figure 2 is a composition of functors with $Platform$ as the overall functor from **CONCEPTS** $\longrightarrow$ **DATA**, such that for each type of information system the following compositions hold: $Platform = Instance \circ Org \circ Policy$ and $Sys = MetaMeta \circ Meta \circ Classify$

An obvious benefit is that we can relate concepts across platforms by comparing the functors $Platform$: **CONCEPTS** $\longrightarrow$ **DATA** for each of our types of system. However, for full type consistency we should consider the two-way mappings and ensure that composition holds in both directions. Such consistency is achieved by adjointness. The topic of adjunctions and their composition is therefore now discussed.

3.1 ADJOINTNESS BETWEEN CATEGORY TYPES

Adjointness characterises the unique relationship between cartesian-closed categories (that is categories of real-world objects). There is a lower-limit functor (F) that preserves co-limits and right-adjoint to (F) is an upper-limit functor (G) which preserves limits, the whole written $F \dashv G$. Further details on the concept of adjointness are available at (Barr and Wells, 1990).

3.2 COMPOSING ADJOINTS

The multi-level application shown in Figure 2 involves the composition of adjoints, that is an expression is derived in which two or more adjoints are adjacent to each other. It is part of the power of category theory that adjoints can be composed in the same way as other arrows. For example consider the adjoints shown in Figure 3 where **CC** is the category **CONCEPTS**, **CS CONSTRUCTS**, **SM SCHEMA** and **DT DATA**.

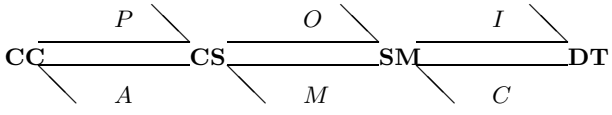

Figure 3: Composition of Adjoints

Then we may have six adjoints (if the conditions are satisfied):

1. $I \dashv C$
2. $O \dashv M$
3. $P \dashv A$
4. $I \circ O \dashv M \circ C$
5. $O \circ P \dashv A \circ M$
6. $I \circ O \circ P \dashv A \circ M \circ C$

where P is the functor $Policy$, O Org, I $Instance$, A $MetaMeta$, M $Meta$ and C $Classify$.

We can construct the 4-tuple to represent the composed adjunctions defined in Figure 2:

$< IOP, AMC, AM\bar{\eta}_{cc}OP \bullet A\bar{\eta}_{cc}P \bullet \eta_{cc}, \bar{\bar{\epsilon}}_{dt} \bullet I\bar{\epsilon}_{dt}C \bullet IO\epsilon_{dt}MC >$

If the conditions of this adjunction are met, we can represent the composed adjunction $Platform \dashv Sys$ by the 4-tuple
$< Platform, Sys, \eta_{cc}, \epsilon_{dt} >$: **CC** $\longrightarrow$ **DT**
where $Platform = IOP$, $Sys = AMC$, η_{cc} is the unit of adjunction, ϵ_{dt} is the counit of adjunction, cc is an object in **CC** and dt an object in **DT**.

This adjunction can be evaluated for each application giving a collection of 4-tuples. Comparison of these 4-tuples then gives the mechanism for computational type closure. The ability to compose adjoints naturally means that we can combine well together such diverse features as policy, organization and data in a single arrow. The advantage in deriving these compositions is that we have the ability to represent the mappings in either abstract or detailed form (Rossiter and Heather, 2000). The overall composition gives a simple representation for conceptual purposes; the individual mappings enable the transformations to be followed in detail at each stage and provide a route for implementation. The uniqueness of the components means that an adjunction can be resolved where there is a component missing.

3.3 COMPARING SYSTEMS

Adjunctions give the relationships between one level and another. We can also approach the problem by considering a direct mapping between one instance of the four-level architecture and another as in Figure 4. Here for simplicity the mappings are viewed in one direction only.

Two systems are compared, one involving categories **CC**, **CS**, **SM** and **DT**, the other **CC**, **CS′**, **SM′** and **DT′**. **CC** is the same in both systems as there is one universal type for concepts.

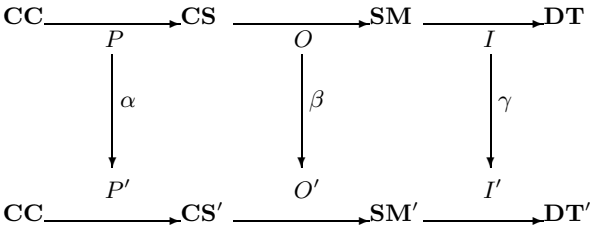

Figure 4: Comparison of Mappings in two Systems

As in Figure 3, the functors relate the categories. We have now though added natural transformations to relate the mapping between one functor and another. The functors need to be of the same type for a meaningful natural transformation to exist between them and this is the case for α, β and γ.

We therefore have three types of mapping to consider: within a category (for instance from a name to a value), from one category to another (for instance the functor P' from **CC** to **CS′**) and from one functor to another (for instance the natural transformation α from P to P').

If we follow the constructive principles of category theory, then the composition of these arrows is natural. The Godement calculus ((Godement, 1958); (Barr and Wells, 1990), pp 94-97) gives a number of rules governing the compositions. Rules G2 and G3 say that the composition of functors and natural transformations is associative so that for instance:

$$(I'O')\alpha = I'(O'\alpha)$$

$$\gamma(OP) = (\gamma O)P$$

Rule G3 says that natural transformations may be composed with each other:

$$\gamma\beta = (\gamma O) \circ (I'\beta)$$

$$\beta\alpha = (\beta P) \circ (O'\alpha)$$

The consequence of this for interoperability is that a categorical approach ensures that the various arrows of different types can be composed with each other, irrespective of their level in the system. Equations can be derived, representing an equality of paths, with unknown components that can be determined from an evaluation of the known properties. For instance with the path IOP from $\mathbf{CC} \longrightarrow \mathbf{CS} \longrightarrow \mathbf{SM} \longrightarrow \mathbf{DT}$ defining an object-oriented system, then the path $I'O'\alpha$ from $\mathbf{CC} \longrightarrow \mathbf{CS'} \longrightarrow \mathbf{SM'} \longrightarrow \mathbf{DT'}$ would define a relational representation if P' maps onto relational constructs in the category **CS′**.

3.4 FOUR LEVELS ARE SUFFICIENT

In category theory four levels are required to define an arrow as unique up to natural isomorphism. The four levels are: 1) object or identity arrow (within a category), 2) category (comparing objects), 3) functor (comparing categories) and 4) natural transformation (comparing functors). No more levels are required. An arrow comparing natural transformations is a natural transformation. The figures below show what happens when an arrow is placed between two natural transformations.

Two squares, derived from Figure 4, are shown below. Figure 5 must commute for each arrow $f : cc \longrightarrow cs$ if α is to be a natural transformation. Similarly Figure 6 must commute for each arrow $g : cs \longrightarrow sm$ if β is to be a natural transformation. Note that viewed in this way a natural transformation is not a layer above functors and functions. The levels are interwoven with natural transformations considering how every arrow defined at the lowest level is mapped.

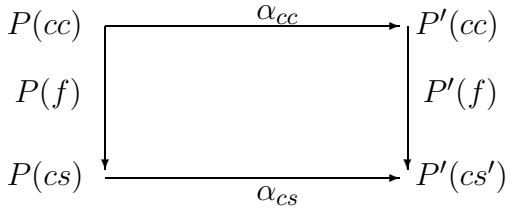

Figure 5: Commuting Target Square for Natural Transformation $\alpha : P \longrightarrow P'$, comparing policies in two systems

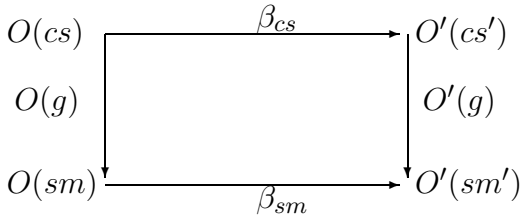

Figure 6: Commuting Target Square for Natural Transformation $\beta : O \longrightarrow O'$, comparing use of constructs in two systems

Now if we write the arrow: $\delta : \alpha \longrightarrow \beta$, we can see that δ is a composition $\beta \circ \alpha$ giving the commuting square shown in Figure 7 ((Barr and Wells, 1990), at p.85).

So an arrow from one natural transformation to another gives a composition of the natural transformations, not a new level ((Barr and Wells,

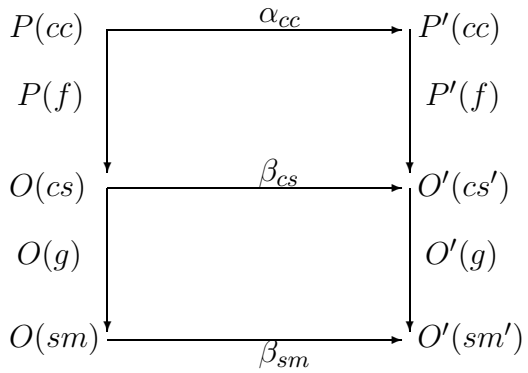

Figure 7: Commuting Target Square for Natural Transformation composition $\beta \circ \alpha$, comparing mapping from concepts to schema in two systems

1990), at p.85). The four levels of concepts, constructs, schema and data are viewed in Figure 2 as four categories connected by a composition of three functors. An alternative view, shown in Figure 8, is closer to the four levels inherent in category theory. The fundamental levels are considered to be data values, named values, classified values and contrasted representation corresponding in category theory to object, category, functor and natural transformation respectively. The natural transformations are now the duals of those shown earlier in Figure 4 as indicated by the * superscript. The earlier natural transformations were comparing the downward functorial mapping (towards data) while the current ones compare the upward mapping (away from data).

alternative fundamental levels	category theory levels	four levels of Figure 4
1. data values	objects (identity arrows)	id_{dt}
2. named values	category	**DT**
3. classified values	functor	C : **DT** $\longrightarrow$ **SM**
4. contrasted representation	natural transformation	$\alpha^* \circ \beta^*$

Figure 8: Alternative Interpretation of Levels in the Architecture

This view does not supersede the earlier one which remains more useful for system design and analysis as it has reassuring similarity to the ANSI/SPARC three-level architecture. The alternative view though may have some potential for interoperability where comparisons are an inherent part of the methodology and is also more in the spirit of category theory with natural transformations as ultimate closure.

It can be seen that the addition of further levels is possible but nothing is gained by it typewise. Thus addition of an extra level to the top of Figure 1 simply results in the top level (comparison of mapping from concepts to schema) being a composition of three arrows rather than two. Thus looking back at Figure 3, consider the addition of a new top level **PH** with the mappings F : **PH** $\longrightarrow$ **CC**, G : **CC** $\longrightarrow$ **PH** and $\alpha^{*\prime} : F \longrightarrow F'$ where $\alpha^{*\prime}$ compares the mappings F and F' in two different approaches. The adjoint is now $I \circ O \circ P \circ F \dashv G \circ A \circ M \circ C$. The level four of Figure 8 is now $\alpha^{*\prime} \circ \alpha^* \circ \beta^*$ and is still a natural transformation through the rules of composition. The practical consequence is that a fifth level is equivalent to an alternative fourth level. The meta-meta level gives ultimate closure of types.

It is an interesting question as to what constitutes a horizontal mapping as opposed to a vertical one. There is a type change in both cases but that in a vertical mapping is greater than in a horizontal one. Indeed the vertical mappings correspond closely to intension-extension mappings whereby the extension is populated with values corresponding to the intensional type. The horizontal mappings can be between variants on a theme (diagrammatic or syntactic schema) or more fundamental in linking names to values. The choice may be slightly arbitrary but in category theory (Mac Lane, 1998) the composition of horizontal and vertical mappings is natural, giving an equivalent result irrespective of the orientation of the mappings ((Barr and Wells, 1990), at p.95).

4 LEVELS IN APPLICATIONS

We now look at some existing approaches to see how they compare in giving a genuine four-level strategy for tackling the problem of interoperability.

4.1 INFORMATION RESOURCE DICTIONARY SYSTEM

For reasons like matching types across levels, the ANSI Standard Reference Model (X3.138) was developed in the 1980s (Gradwell, 1990), emerging in 1993 (ISO, 1993) as an associated part of the standard for a framework for Information

Resource Dictionary System (IRDS). Language bindings can in principle be defined for any language. Further information on the IRDS objectives has been compiled by Gradwell (Gradwell, 1990) and a basic scheme for the IRDS is shown in (Rossiter, Nelson and Heather, 2001).

The IRDS exhibits four levels in a manner similar to that shown in Figure 1 with the top level open to the addition of further abstractions. The IRDS can be considered as three composed level-pairs viewed in the direction of reducing abstraction, that is a type change from intension to extension. The IRDS therefore captures the arrows *Policy*, *Org* and *Instance* in Figure 2 but not $Classify$, $Meta$ and $MetaMeta$ which are in the direction of increasing abstraction. Since there are no two-way mappings the IRDS does not support adjointness as shown in Figure 3.

IRDS is a partial representation of a four-level reference model. Its use has been less than expected, perhaps because of the lack of theoretical underpinning and of a two-way mapping capability. Also some users are still reluctant to extend the levels beyond meta-data types. The tendency is to design flat systems where the need to cross levels is avoided.

4.2 THE GRID

The success of the Web for information has prompted the idea of the Grid for computing in e-science. To produce finite results, the Grid must have closure which the universal constructions of category theory show to be natural transformations. The categories involved in the Grid can be processors or connections on the Grid, that is objects or arrows. Figure 9 shows the levels ideally involved in the Grid and their interpretation. The double-headed arrows represent adjunctions. The overall adjointness at the level of the system as a whole is constructed as it arises and cannot be determined by any proactive planning at such a high level although the individual components can be efficiently organised from a local perspective. This overall adjointness is a four-level structure that can be resolved into the composition of its constituent lower-level adjoints.

To date though the Grid has almost entirely used operating system files with very little use of formal database systems (Watson, 2002). In effect the Grid is really only using the two bottom levels of Figure 1 of schema and data values with the omission of constructs and concepts. The need for an understanding of meta data in developing Grid database systems has also been raised with the possibility of using XML as a solution (Paton *et al*, 2002). It is clear that for successful interoperability the structure of the Grid has to be extended to provide a genuine four-level capability.

Level	Intension	↔	Extension
1	Grid Policy	↔	Program Purpose
	↕		↕
2	Grid Organisation	↔	Program Standard
	↕		↕
3	Grid Implementation	↔	Program Specification
	↕		↕
4	Grid Operation	↔	Program Execution

Figure 9: Interpretation of Levels in the Grid

4.3 DATA EXCHANGE LANGUAGES

Much work has recently centered on markup languages for data exchange. From initial workings with the eXtensible Markup Language (XML), the following basic constructions now exist (W3C, 2002):

1. The data values which are to be enhanced with the XML markup tags.
2. The XML document, data marked up with XML tags for identifying the content of a document.
3. the DTD (Document Type Definition), defining the XML document, structures, properties, relationships, rules and elements. Used to define the tags in an XML document.
4. The Schema, an alternative to DTD, enabling elements (objects), properties and relationships between elements to be defined.
5. The RDF (Resource Description Framework) (W3C, 2002) which aims to integrate a variety of web-based metadata activities. RDF defines triples <subject, predicate, object> where the components are typically single URIs (Uniform Resource Identifiers, web addresses to fragment level) but may also be literals, collections of URIs held in a container or another RDF.
6. The Semantic Web (Berners-Lee *et al*, 2001) aims to provide semantic interoperability between data sources mounted on the web, using RDF and ontologies. The meanings of terms

can be found by exploring ontologies given as URIs in the triples in RDF.

In terms of our four-level architecture, the data are the objects in **DATA** and the document is the objects in **SCHEMA**. RDF does not fit exactly into the architecture. It can be viewed as a relationship between one schema and another, represented categorically as a pullback (Nelson and Rossiter, 1996) of *object* over *subject* in the context of *predicate*. Figure 10 shows such a pullback with the subject as category **S**, the object as **O** and the predicate as **P**. The category **W** is the world-wide web and the pullback $\mathbf{S} \times_{\mathbf{P}} \mathbf{O}$ represents RDF relationships across the web. The arrow Σ ensures that single components of the pullback exist on the web and Π that quotients of components exist on the web.

The semantic web employs ontologies to enable agents to compare and contrast the use of terminology in one web site with that in another. The semantic web attempts to perform the function γ in Figure 4, enabling the mapping from $\mathbf{SM} \longrightarrow \mathbf{DT}$ (schema to data) in one system to be compared with that from $\mathbf{SM}' \longrightarrow \mathbf{DT}'$ in another. This view is summarised in the table in Figure 11.

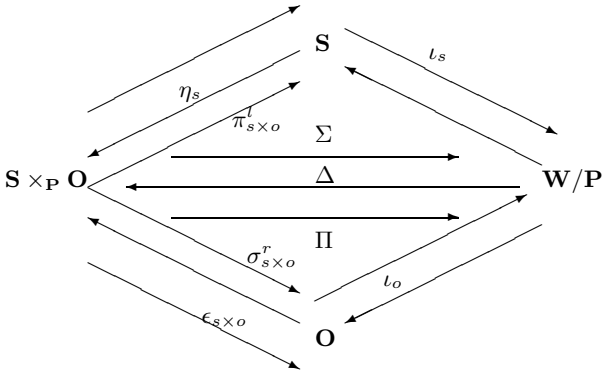

Figure 10: Pullback of Object (**O**) over Subject (**S**) in the context of Predicate (**P**) as a subcategory of the Web (**W**)

In terms of the pullback diagram, the semantic web depends on the composition $\Sigma \circ \Delta$. From resources in the web **W**, an RDF relationship is found through Δ in $\mathbf{S} \times_{\mathbf{P}} \mathbf{O}$. Further exploration of the web is then done through the mapping Σ for simple components or Π for container components. This process can be repeated as many times as necessary under the control of an agent. It needs to be stressed that in cases of mismatched ontologies there is no guarantee that the mapping γ can actually be achieved. It all depends on the capability of the agent.

W3C feature	function	four levels of Figure 4
Data	data	id_{dt}
Document	marked-up data	**DT**
DTD or Schema	definition of document (objects, etc)	**SM**
RDF	relating one schema to another	**SM** (a pullback $\mathbf{S} \times_{\mathbf{P}} \mathbf{O}$ within this category, as in Figure 10)
Semantic Web	Linking RDF to Ontologies	γ (if an agent can handle mismatched ontologies)

Figure 11: Interpretation of Levels in the W3C Proposals

4.4 METAOBJECTS IN THE MODEL-DRIVEN ARCHITECTURE

Considerable work in improving interoperability between object-based systems has been done by Crawley (Crawley *et al*, 1997). His group developed the Meta-Object Facility (MOF) which introduces an information layer above the layers of schema information and the information itself. In their terminology, meta-information is information about objects in the form of type schemas and meta-meta-information is information about meta-information in the form of meta-schemas containing types of types and types of relationships.

MOF provides a framework for managing meta-information and has been accepted by the Object Management Group (OMG) as a way forward in achieving interoperability between object systems. In particular it may be used for comparing UML constructions with those from other object-based conceptual modelling tools (Bezivin, 2001). The levels are effectively M3 (MOF), M2 (e.g. UML), M1 (a particular UML model) and M0 (a use of the UML model) corresponding in grammar terms to an extended BNF formalism, the C++ grammar, a C++ program and the execution of a C++ program respectively.

It is not clear that MOF has an open top level corresponding to real-world abstractions as found in say the IRDS above. In particular the top layer is said to be hard-wired (OMG, 2000). MOF certainly facilitates the connection of, say, one object system to another but appears to lack a top

neutral level relating constructs to universal abstractions, which would enable handling of the full range of heterogeneity, such as objects, relations, trees and data processing record-types. There is also some confusion in terminology with systems such as IRDS and relational databases using the term meta as a relationship between levels and the object systems using it as a level descriptor.

5 DISCUSSION AND CONCLUSIONS

The theory shows that four levels are needed to address an object's values with sufficient context to achieve non-local interoperability. More than four levels yields no better context and fewer than four levels yields a context that will only permit local interoperability.

The Information Resource Dictionary Standard (IRDS) did provide four levels but ANSI downgraded this standard and its influence has been less than anticipated. More recently ISO has begun again to value a four-level architecture with the consideration of a meta-meta *model* in work on comparing models (ISO, 1999). Systems developed recently, claiming to provide interoperability, such as MOF are able to provide considerable assistance within a paradigm but appear to lack the top level, mapping abstractions to constructions, necessary to achieve interoperability across paradigms. Both IRDS and MOF are data-driven approaches in a general sense. The semantic web takes a different approach, being partially data-driven through RDF but also relying on agent-based technology for resolving mismatches. The semantic web therefore appears to lack the two top levels of concepts and constructs but the use of ontologies and agents may compensate to some extent at least for some of this loss. The Grid also lacks the top two levels for data addressing and its potential will not be realised until this deficiency is tackled.

The generality of current techniques for interoperability is in doubt. Now with the categorical relationships described here, it should be possible to develop with greater confidence universal four-level systems. The definition of the four levels necessary for providing interoperability, the availability of the Godement calculus for composing mappings formed at different levels and the specifications of the adjointness between the levels and of pullback categories representing relationships, all add coherence through a categorical approach to interoperability. Practical difficulties should be less significant once a rigorous theoretical framework has been established. A prototype implementation is planned to test the formal architecture and the information systems described.

REFERENCES

Barr, M, & Wells, C, *Category Theory for Computing Science*, Prentice-Hall (1990).

Berners-Lee, T, Hendler, J, & Lassila, O,The Semantic Web, *Scientific American*, May 2001.

Bezivin, J, From Object Composition to Model Transformation with the MDA, *3rd ICEIS*, Setubal, invited paper (2001).

Crawley, S, Davis, S, Indulska, J, McBride, S, Raymond, K, Meta-meta is better-better!, *DAIS'97*, Cottbus, Germany 12pp (1997).

Godement, R, *Théorie des faisceaux*, Hermann (1958).

Gradwell, D J L, The Arrival of IRDS Standards, *8th BNCOD*, York 1990, Pitman 196-209 (1990).

Information technology - *Reference Model of Data Management*, Standard ISO/IEC 10032 (1993).

ISO, Study Report on the Feasibility of Mapping Modelling Languages for Analysis and Design Models.ISO/IEC JTC1/SC7 N2112, 1999/04/19 (1999).

Mac Lane, S, *Categories for the Working Mathematician*, 2nd ed, Springer-Verlag, New York (1998).

Nelson, D A, & Rossiter, B N, Prototyping a Categorical Database in P/FDM. *ADBIS'95*, Moscow, Springer-Verlag Workshops in Computing, 432-456 (1996).

OMG, Meta Object Facility (MOF) Specification Version 1.3 March 2000 http://www.ifs.univie.ac.at/lehre/WS2001/BWIEC/MOF-extract.pdf (2000).

Paton, N W, Atkinson, M P, Dialani, V, Pearson, D, Storey, T, Watson, P, Database Access and Integration Services on the Grid, UK e-Science Programme Technical Report Series UKeS-2002-03 (2002).

Rossiter, B N, Nelson, D A, & Heather, M A, A Universal Technique for Relating Heterogeneous Data Models, 3rd ICEIS, Setbal, I 96-103 (2001).

Rossiter, B N, & Heather, M A, Handling Incosistency with the Universal Reference Model, *MS'2000*, 611-618 (2000).

Watson, P, Databases and the Grid, Computing Science Technical Report no.755, University of Newcastle upon Tyne (2002), (16pp).

W3C Consortium, *semantic Web*, http://www.w3.org/2001/sw/ (October 2002).