A Natural Basis for Interoperability

Nick Rossiter[1], Michael Heather[1], and David Nelson[2]

[1] Computing, Engineering and Information Sciences, Northumbria University, NE2 1XE, UK, nick.rossiter@unn.ac.uk
[2] Computing and Technology, University of Sunderland, SR6 0DD, UK, david.nelson@sunderland.ac.uk
WWW home page: http://computing.unn.ac.uk/staff/CGNR1/

Keywords: organisational interoperability, category theory, naturality, adjointness, Godement calculus

1 Need for Formal Natural Multi-level Type Systems

Interoperability needs natural techniques to deal with levels of types. To handle (non-local) interoperability, formality (for reliability and predictability), naturality (for reality) and multi-level types (for types of types) are all required. Categorical methods should replace classical models because models are local and interoperability is non-local. Categorical methods provide formal definitions of levels (as categories), mappings between levels (functors between categories) and comparison of one mapping between levels with another (natural transformation between functors). Categorical techniques are also natural: an arrow within a category is defined as unique up to natural isomorphism.

In areas such as ontologies an informal approach has been taken to naturality which can follow the categorical approach in style, if not in complete formality. Thus in enterprise ontology three levels may be defined: construction model, process/information model and action model [5]. Such ontologies are defining the existence of an object in the context of multiple levels, which is close to the spirit of category theory. In object-based applications MDA (Model-Drive Architecture) has been developed, which separates business and application logic from the underlying platform technology [13]. MDA is based on MOF (Meta-Object Facility) with considerable bias towards UML. An aim of MDA is platform-independence of object-based applications, rather than interoperability between general systems.

Category theory has been used before in information system applications. [7] used the monoid calculus in an attempt at standardising the querying

of different collection types. [10] applied sketches to entity-relationship and relational modelling and [6] to object databases. None of these approaches though have been multi-level. Sketches are also strictly outside category theory as they permit diagrams that do not commute but they may be mapped onto categories by a model functor.

One of the most important features of category theory is adjointness, which gives a degree of measurement of the extent to which the mappings between two categories are equivalent [3]. If the arrows between the categories are functors F, G, then the reverse logic gate $F \dashv G$ is conventionally used to represent adjointness. It is the phenomenon of naturalness. F is left adjoint to G and G is right adjoint to F. The unit of adjunction η and counit of adjunction ϵ measure the extent to which the result from composing the functors differs from the starting point: η_l is the unit of adjunction $1_l \longrightarrow GF(l)$ and ϵ_r is the counit of adjunction $FG(r) \longrightarrow 1_r$.

2 Multi-level Data Structures

The four-level architecture in Figure 1 has orthogonal types with the relationships between the levels expressed as categorical adjunctions, as already applied to structures in GRID data processing [9]. Categorical adjunctions relate one level to another. The relationship between levels is measurable by the unit of adjunction. For instance the adjunction $Policy \dashv MetaMeta$ indicates that the free functor $Policy$ is left adjoint to the underlying functor $MetaMeta$. The unit of adjunction is given by $\eta_{cpt} : 1_{cpt} \longrightarrow MetaMeta \circ Policy(cpt)$. The terms used have their normal meaning. In the downward direction, a collection of data structuring concepts (abstractions) are mapped through policies to a collection of constructions (for example classes, tables) which are in turn mapped through organisation to a collection of data types (for example, schema definitions) which are finally mapped through instantiation to named data values. In the opposite direction, the named data values are mapped through classification to schema types, which are in turn mapped through metadata to constructions and through metameta data to concepts.

3 Basis for Interoperability

As mentioned earlier, there are three areas of interoperability that our architecture must satisfy: data structures, constraints and data manipulation. Each is covered in turn.

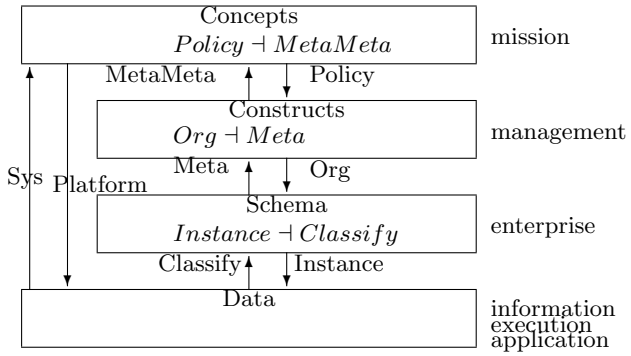

Fig. 1. Interpretation of Levels as Natural Schema in General Terms

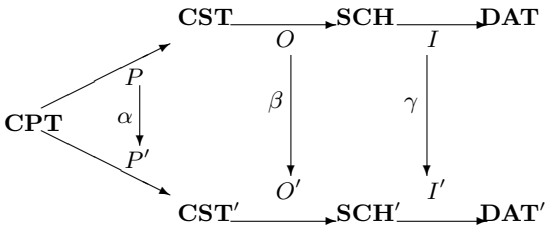

Fig. 2. Comparison of Mappings in two Systems

3.1 Natural Transformation as Data Structures

In category theory four levels are needed to define an arrow as unique up to natural isomorphism: 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.

The relationships between one four-level architecture and another can be constructed as in Figure 2, the expanded view of Figure 1. Here for simplicity the mappings are viewed in one direction only. Two systems are compared, one involving categories **CPT**, **CST**, **SCH** and **DAT**, the other **CPT**, **CST′**, **SCH′** and **DAT′**, representing concepts (**CPT**), constructs (**CST**), schema (**SCH**) and data (**DAT**) from Figure 1. **CPT** is the same in both systems as there is one universal type for concepts. As usual the functors relate the categories. We have now though added natural transformations to relate the mapping between one functor and another. It needs to be emphasised that none of these categories are discrete: all have an internal arrow-based structure so the natural transformations are non-trivial [14]. The functors need to be of the same variance and involve categories of the same underlying type for a meaningful natural transformation to exist between them. This is the case for α, β and γ: the functors are all contravariant as described later and each natural transformation involves two underlying categorical types.

An arrow comparing natural transformations is itself a natural transformation. Some categorists use an older terminology with degrees of 'cell' and describe the identity arrow as 0-cell, an arrow in a category as 1-cell and an arrow between arrows as 2-cell [11]. An arrow from one natural transformation to another gives a composition of the natural transformations, not a new level (([3], 1st ed., at p.85); [15]). This means that four levels are needed to give the natural closure [9].

It may be asked what the levels are going to comprise and what is the nature of the mapping between the levels. Are these constructions essentially arbitrary or do they have definitions, which naturally fall into place? Fortunately the latter seems to apply if we are working in category theory with its property of adjointness. Lawvere [12] in his study of adjointness showed that the relationship between intension and extension is contravariant, indicating that the mapping from the category representing the extension, say **DAT**, to the category representing the intension, say **SCH**, is from codomain in **DAT** to domain in **SCH** and domain in **DAT** to codomain in **SCH**.

For matching across the levels in a contravariant manner, the intension **SCH** should be defined with arrows of the form, name $\longrightarrow$ type, and the extension **DAT** with arrows of the form: value $\longrightarrow$ name. Both these arrows are functions since associated with each value is one name and associated with each name is one type. Mapping from extension to intension then maps the codomain *name* in **DAT** onto the domain *name* in **SCH** and the domain *value* in **DAT** onto the codomain *type* in **SCH**. This mapping effectively embeds values in types in the context of a name.

The alternative covariant mapping would be from domain to domain and codomain to codomain. The arrow in **SCH** then needs to be reversed to name $\longrightarrow$ value for the two levels to be related. However, name $\longrightarrow$ value is not a function so a covariant functor from **DAT** to **SCH** lacks naturality.

The four levels of Figure 2 can now be viewed as the two intension-extension pairs in Figure 3. The pairs are for **CPT/CST** (concepts/constructs) and **SCH/DAT** (schema/data). For interoperability purposes, it has been shown by the fundamental nature of category theory that four levels are sufficient for all purposes [16]. Further levels are possible but unnecessary. To maintain the coherence of the present approach it would be necessary to go up to six levels as the next step to maintain the intension-extension pairings.

The table in Figure 4 shows the four levels of concepts, constructs, schema and data with the functors between them of P' (metameta), O' (meta) and I' (classify). The arrows shown for the functors indicate the contravariant nature of the mapping with domain to codomain and codomain to domain. The three examples, from left to right, are for a property, aggregation with relational tables and encapsulation with an abstract data type (ADT). The latter shows the mapping from a binary-tree object named aTree through the class BST and the ADT construction to the encapsulation concept.

3.2 Adjoints for Constraints

In category theory there is a unique solution if adjointness holds between two functors as mentioned earlier. The construction can be readily extended to handle four levels, shown in Figure 5, as the composition of adjoints is natural. In Figure 5, with categories and functors as in Figure 3, there are six adjoints [9], one for each functor and its mapping in the opposite direction (1-3), one for each pair of adjacent functors and its opposite (4-5) and one for all three functors composed together and its opposite (6). These adjoints are defined in detail in the following six expressions:

$$< P, P', \eta_{cpt}, \epsilon_{cst} >: \mathbf{CPT} \longrightarrow \mathbf{CST} \tag{1}$$

η_{cpt} is the unit of adjunction $1_{cpt} \longrightarrow P'P(cpt)$ and ϵ_{cst} is the counit of adjunction $PP'(cst) \longrightarrow 1_{cst}$

$$< O, O', \bar{\eta}_{cst}, \bar{\epsilon}_{sch} >: \mathbf{CST} \longrightarrow \mathbf{SCH} \tag{2}$$

$\bar{\eta}_{cst}$ is the unit of adjunction $1_{cst} \longrightarrow O'O(cst)$ and $\bar{\epsilon}_{sch}$ is the counit of adjunction $OO'(sch) \longrightarrow 1_{sch}$

$$< I, I', \bar{\bar{\eta}}_{sch}, \bar{\bar{\epsilon}}_{dat} >: \mathbf{SCH} \longrightarrow \mathbf{DAT} \tag{3}$$

$\bar{\bar{\eta}}_{sch}$ is the unit of adjunction $1_{sch} \longrightarrow I'I(sch)$ and $\bar{\bar{\epsilon}}_{dat}$ is the counit of adjunction $II'(dat) \longrightarrow 1_{dat}$

$$< OP, P'O', P'\bar{\eta}_{cst}P \bullet \eta_{cpt}, \bar{\epsilon}_{sch} \bullet O\epsilon_{cst}O' >: \mathbf{CPT} \longrightarrow \mathbf{SCH} \tag{4}$$

$P'\bar{\eta}_{cst}P \bullet \eta_{cpt}$ is the unit of adjunction $1_{cpt} \longrightarrow P'O'OP(cpt)$ and $\bar{\epsilon}_{sch} \bullet O\epsilon_{cst}O'$ is the counit of adjunction $OPP'O'(sch) \longrightarrow 1_{sch}$

We have retained the symbol $\bullet$ indicating vertical composition [11] as distinct from horizontal composition indicated by the symbol $\circ$ which is normally, as here, omitted altogether.

$$< IO, O'I', O'\bar{\bar{\eta}}_{sch}O \bullet \bar{\eta}_{cst}, \bar{\bar{\epsilon}}_{dat} \bullet I\bar{\epsilon}_{sch}I' >: \mathbf{CST} \longrightarrow \mathbf{DAT} \tag{5}$$

$O'\bar{\bar{\eta}}_{sch}O \bullet \bar{\eta}_{cst}$ is the unit of adjunction $1_{cst} \longrightarrow O'I'IO(cst)$ and $\bar{\bar{\epsilon}}_{dat} \bullet I\bar{\epsilon}_{sch}I'$ is the counit of adjunction $IOO'I'(dat) \longrightarrow 1_{dat}$

$$\begin{aligned} < IOP, P'O'I', P'O'\bar{\bar{\eta}}_{sch}OP \bullet P'\bar{\eta}_{cst}P \bullet \eta_{cpt}, \\ \bar{\bar{\epsilon}}_{dat} \bullet I\bar{\epsilon}_{sch}I' \bullet IO\epsilon_{cst}O'I' >: \mathbf{CPT} \longrightarrow \mathbf{DAT} \end{aligned} \tag{6}$$

$P'O'\bar{\bar{\eta}}_{sch}OP \bullet P'\bar{\eta}_{cst}P \bullet \eta_{cpt}$ is the unit of adjunction $1_{cpt} \longrightarrow O'I'IO(cpt)$ and $\bar{\bar{\epsilon}}_{dat} \bullet I\bar{\epsilon}_{sch}I' \bullet IO\epsilon_{cst}O'I'$ is the counit of adjunction $IOO'I'(dat) \longrightarrow 1_{dat}$

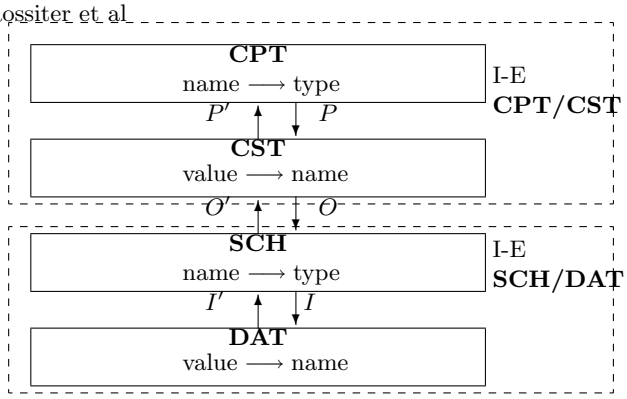

Fig. 3. Defining the Four Levels with Contravariant Functors and Intension-Extension (I-E) Pairs

Level	Template	Property	Relational Database (aggregation)	Abstract Data Type (encapsulation)
CPT	name ⟶ type	attribute ⟶ property	table ⟶ aggregation	ADT ⟶ encapsulation
P'	↗ ↖	↗ ↖	↗ ↖	↗ ↖
CST	value ⟶ name	registration_no ⟶ attribute	birth_type ⟶ table	BST ⟶ ADT
O'	↗ ↖	↗ ↖	↗ ↖	↗ ↖
SCH	name ⟶ type	car_reg ⟶ registration_no	birth_record ⟶ birth_type	aTree ⟶ BST
I'	↗ ↖	↗ ↖	↗ ↖	↗ ↖
DAT	value ⟶ name	'x123yng' ⟶ car_reg	<'Smith', 25 mar 1980, 'Torquay' > ⟶ birth_record	instance of tree (nodes/links) ⟶ aTree

Fig. 4. Examples of Levels in the Four-Level Architecture

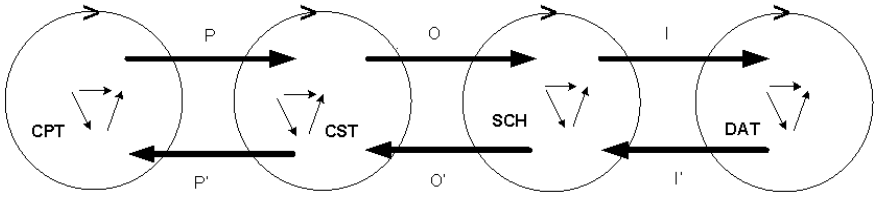

Fig. 5. Composition of Adjoints is Natural

The expressions above specify the conditions to be satisfied if adjointness occurs in all possible cases in Figure 5. From these constraints we derive values for the various units of adjunction η and counits of adjunction ϵ. If a unit of

adjunction is $\perp$, that is for example $1_{cpt} = P'P(cpt)$, then the application of functors P and P' in turn returns the initial object (1_{cpt}). If a counit of adjunction is $\top$, that is for example $PP'(cst) = 1_{cst}$ then the application of functors P' and P in turn returns the terminal object (1_{cst}). These are special cases. In other cases of adjointness η measures retrospectively the difference between the starting and finishing points after applying in turn the free and underlying functors. ϵ measures the difference between the starting and finishing points after applying in turn the underlying and free functors.

3.3 Natural Calculus for Data Manipulation

Looking at Figure 2, we have three types of mapping: within a category (for instance from a name to a value), from one category to another (for instance the functor P' from **CPT** to **CST**′) and from one functor to another (for instance the natural transformation α from P to P'). Following the constructive principles of category theory, the composition of these arrows is natural, giving rise to a natural calculus first expounded by Godement [8] and now available at ([3], 1st ed., pp 94-97) in the form of rules governing composition.

With natural closure a categorical approach ensures that the various arrows of different types can be composed associatively with each other, irrespective of their level. Equations representing an equality of paths, can be solved for unknown components that can be determined from an evaluation of the known properties. For instance in comparing methods with the path IOP from **CPT** $\longrightarrow$ **CST** $\longrightarrow$ **SCH** $\longrightarrow$ **DAT** defining one approach, then the path $I'O'P'$ from **CPT** $\longrightarrow$ **CST**′ $\longrightarrow$ **SCH**′ $\longrightarrow$ **DAT**′ might define an alternative approach if P' maps onto constructs in the category **CST**′.

The diagram in Figure 6 shows the application of the Godement calculus to handle semantic interoperability, defined as the interoperation of one system with another at the level of meaning of the data, that is at the metadata level.

The composition of the top line of functors $I_r \circ O_r \circ P$ gives the mapping from concepts to data for say a relational system r. The composition of the middle line of functors $I_{or} \circ O_{or} \circ P$ gives the mapping from concepts to data for say an object-relational system *or*. The composition of the bottom line of functors $I_{oo} \circ O_{oo} \circ P$ gives the mapping from concepts to data for say an object-oriented system *oo*. Comparing these compositions gives a framework for interoperability. For instance the natural transformation α' compares how the mapping is performed from constructions to schema in a relational system r with that from constructions to schema in an object-relational system *or*. The natural transformation β'' compares how the mapping is performed from schema to data in an object-relational system *or* with that from schema to data in an object-oriented system *oo*. The advantage of the Godement approach is that arrows at any level may be composed with each other.

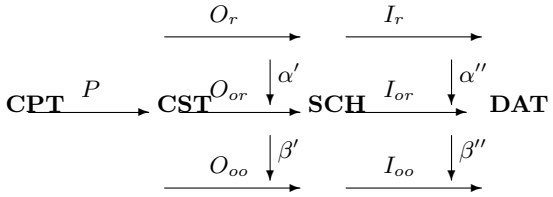

Fig. 6. Semantic Interoperability in terms of Godement

To extend the categorical framework to handle organisational interoperability, defined as the interoperation of systems at the business process level, we need to vary the functor P for each environment so that the metameta level is variable. The required diagram is shown in Figure 7.

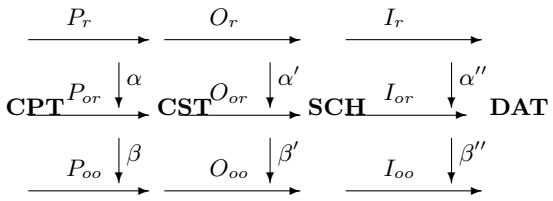

Fig. 7. Organisational Interoperability in terms of Godement

Applying the rules given at ([3], 1st ed., pp 94-97), the following compositions hold in Figure 7 according to the Godement calculus:

$$(\beta' \circ \alpha')(\beta \circ \alpha) = (\beta'\beta) \circ (\alpha'\alpha) \tag{7}$$

$$(I_{or} \circ O_r)\alpha = I_{or}(O_r\alpha) \tag{8}$$

$$\alpha'(O_r \circ P_{or}) = (\alpha' O_r)P_{or} \tag{9}$$

$$I_r(\beta' \circ \alpha')P_{or} = (I_r\beta' P_{or}) \circ (I_r\alpha' P_{or}) \tag{10}$$

$$\alpha''\alpha' = (\alpha'' O_{or}) \circ (I_r\alpha') = (I_{or}\alpha') \circ (\alpha'' O_r) \tag{11}$$

A number of general principles in composition are shown by the equations. Equation 7 indicates that of commutativity (the interchange law); equations 8. . . 9 indicate that of associativity; equation 10 indicates that of permutation of paths. The last equation, 11, shows the production of simultaneous equations representing different paths through the diagram. This is an important feature as it facilitates the solution for an unknown mapping. For example,

in equation 11 above, if the values α', α'' and I_{or} are known, then O_r is the only unknown and a solution can be found for it. That is if it is known how the mapping from constructions to schema and from schema to data varies between a relational system r and an object-relational system or and what the mapping is between schema and data in an object relational system or, then the mapping between constructions and schema in the relational system r can be derived.

4 Discussion

One of the purposes of developing a formalism for a problem area is to provide a rationale in which standards can be planned and discussed. It is perhaps only in the ideal world that standards are based entirely on a theoretical basis. Nevertheless some of the idiosyncrasies and inconsistencies of SQL have been attributed to not rigorously applying axiomatic set theory to the standard [4].

Category theory is a promising candidate as a formalism to assist in the preparation of an interoperability standard because of its pedigree as a workspace for relating different mathematics. The work here has shown that it can indeed perform this role with information systems and cover three critical areas of data structuring, constraints and manipulation (process) in an integrated manner. Recent advances in category theory are likely to improve its match with reality: 2-categories enable some of the strict criteria for composition and associativity to be relaxed to some extent [2].

The approach developed here is close in a number of respects to the IRDS standard for a reference model (ISO/IEC 10027, 13238). IRDS was based too on a multi-level approach with intension-extension pairs. However, IRDS has had limited success and we would attribute this to its reliance on set theory. This has made it difficult to handle multiple levels and has given an emphasis on data structure over important aspects such as process. Implementation has therefore been difficult.

Another standards approach which appears to have been used more is OSI which has a reference model containing seven layers: Application, Presentation, Session, Transport, Network, Data and Physical. OSI clearly covers more aspects of information systems than IRDS and is expressed at a lower level conceptually so it is easier to implement a complete system. OSI has omissions in security and business processes which are very important in current distributed web-based applications. Of direct interest is the final draft proposal (ISO/FDIS 19439) for enterprise modelling which attempts to standardise constructs for enterprise modelling including business process modelling. Without a formal basis, such a standard will be difficult to apply non-locally.

There are two basic tensions that arise with the use of standards: variety and naturality, manageable when local, but irreducible in the non-locality of globalisation. In simple examples uniformity arising from a fixed and narrow standard can result in a loss of variety on account of stringent reductionism.

By Ashby's law of requisite variety [1] a system is driven down if it lacks the necessary variety to provide a source for development, originality and creativity. Two or more interoperable systems require a sufficient interacting variety to operate, otherwise they will be driven down, that is seize up. The use of naturality in a formal context, as in the work presented here, is seen as a step forward in raising the quality of interoperability in the real world.

References

1. Ashby, W.R.,: Principles of the Self-Organizing Dynamic System, *Journal of General Psychology* **37** 125-128 (1947).
2. Baez, J., & Dolan, J.,: Higher-dimensional algebra III: n-Categories and the algebra of opetopes *Adv Math* **135** 145-206 (1998).
3. Barr, M., & Wells, C.,: *Category Theory for Computing Science*, Prentice-Hall (1990, 1995), Centre de Recherches Mathématiques, Montréal (1999).
4. Date, C J, & Darwen, Hugh, *Foundation for Future Database Systems: The Third Manifesto* 2nd Ed, Addison Wesley (2000).
5. Dietz, Jan L.G.,: *DEMO Modelling Handbook* version 2.0 (1999).
6. Diskin, Z., & Cadish, B.,: Algebraic Graph-Based Approach to Management of Multidatabase Systems, *NGITS'95* 69-79 (1995).
7. Fegaras, L., & Maier, D.,: Towards an Effective Calculus for Object Query Languages *Proc 1995 ACM SIGMOD* 47-58 (1995).
8. Godement, R.,: *Théorie des faisceaux*, Hermann, Appendix I (1958).
9. Heather, M.A., & Rossiter, B.N.,: The Anticipatory and Systemic Adjointness of E-Science Computation on the Grid, *Computing Anticipatory Systems* Liège, Dubois, D M, (ed.), AIP Conf Proc **627** 565-574 (2002).
10. Johnson, M., Rosebrugh, R., & Wood, R.J., Entity-Relationship-Attribute Designs and Sketches, *TAC* **10** 94-111 (2002).
11. Kelly, G.M., & Street, R.,: Review on the Elements of 2-categories, Proceedings Sydney Category Theory Seminar 1972-73, ed. G.M. Kelly, *Lecture Notes in Mathematics*, Springer-Verlag **420** 75-103 (1974).
12. Lawvere, F.W.,: Adjointness in Foundations, *Dialectica* **23** 281-296 (1969).
13. OMG, *Model Driven Architecture* http://www.omg.org/mda/ (consulted December 2005).
14. Rossiter, N.,: From Classical to Quantum Databases with Applied Pullbacks, *78th Meeting Peripatetic Seminar on Sheaves and Logic* February (2003).
15. Rossiter, N., & Heather, M.,: Four-level Architecture for Closure in Interoperability, *EFIS2003, Fifth International Workshop on Engineering Federated Information Systems*, Coventry, UK, 17-18 July 83-88 (2003).
16. Rossiter, Nick, & Heather, Michael,: Conditions for Interoperability, *7th ICEIS* Florida, USA, May 2005, 92-99 (2005).