Constructing Standards for Cross-Platform Operation

B.Nick Rossiter
Computing Science
Newcastle University, NE1 7RU, UK
email: B.N.Rossiter@ncl.ac.uk; tel: +44 (0)191 222 7946

Michael Heather
Sutherland Building
University of Northumbria at Newcastle, NE1 8ST, UK

1 Abstract

A universal representation is developed, based on the ISO standards for Information Resource Dictionary System (IRDS), with the aim of providing a complete definition of an information system from the physical data values held to the concepts employed for data and function description and real–world abstractions. It is discussed how such a multi–level model can be used to control the evolution of information systems by creating an environment where heterogeneous systems can be compared. Current trends towards more structured programming techniques and more disciplined software engineering environments lead to the potential for considerable benefits from using an IRDS approach. This potential, however, will only be realized if a formal underpinning of the IRDS standard is achieved and then only reliably if the *formal* is constructive and the underpinning is enabling not just supporting..

The application of standards to interoperable systems is frustrated by their complexity. Consistency across integrated levels in open environments can be satisfied by a single reference model but only if it has natural closure. Standards can usually be relied on when applied to closed systems but there is not the same guarantee in mixed systems based on different standards. The purpose of a reference model is to support corporate information systems by integrating the different standards but formal verification may need to go beyond first-order logic. The ISO standard Information Resource Dictionary System (IRDS) can be shown to have the appropriate formal basis to perform the role of a universal reference model

About the author

Nick Rossiter is lecturer in the Department of Computing Science with particular interests in databases and systems analysis.

Michael Heather is senior lecturer in law where he has been engaged on research into computers and law since 1979.

Suggested Keywords

Heterogeneous Systems, Computerized Information, Cross-platform Compatibility, Client/server Integration, Data Consistency, Category Theory, Natural Transformations, Policy Closure, Reference Model Standards.

2 Introduction

Interoperability is not simple, linear or Boolean. It is multi-level, distributed and usually made up of heterogeneous embedded micro-kernels. For consistency, security and safety, highly-reliable local subsystems may be inadequate. For overall assurance there must also be reliable interoperable connections at the global level.

There is both a theoretical and a very practical dimension. If a company's information system fails, the company fails. Liquidation of the whole enterprise often quickly follows. Robustness is therefore critical particularly in high-consequent computing. The cross-platform software must be able to cope with demands that may not be specifiable in advance. The functionality may be supporting or it may be enabling. If it is supporting it is self-organizing and naturally reacts to the unforeseen. If merely enabling it can only respond as pre-specified. For reliance across levels in the real-world the linking software needs to provide either a natural closure or full information of all limiting constraints.

Examples at the message level of a top layer of awareness needed to deal with the widest possible type of client/server system can be seen in relational databases. Transformations within the relational model are generally performed using the ISO standard query language SQL but there are different flavours of SQL. With object databases the situation is more difficult with no ISO standard emerging although attempts are being made at formulating an industrial consensus[2]. This gives rise to difficulties in relating complex objects between different object-oriented methods[1] like the BM (Bosch Method), OMT (Object Modelling Technique) and OOSE (Object-Oriented Software Engineering) which cannot be easily unified across heterogeneous platforms. Standards like CORBA[4] (The Common Object Request Broker Architecture) are therefore to be used with care. Although it is a standard for interface specification in support of interoperable distributed computing, CORBA does not, for example, provide an inherent security specification. Problems experienced with Java provide another striking example[5][2].

The question is how to unify the methods not just the language[11]. These difficulties show that a logical level is incomplete without an associated formal reference level to describe the representation of real-world entities, properties, relationships, typing, etc. It was for this type of purpose that the ANSI Standard Reference Model (X3.138) was developed in the 1980s, emerging in 1993 as an associated part of the ISO standard for a framework for the Information Resource Dictionary System (IRDS).

The aim of this paper is to investigate whether there is a formal model, for the ISO standards for Information Resource Dictionary System (IRDS), and to see if it can provide a complete definition of an information system from the physical data values held to the concepts employed for data and process representation and client/server architecture. We shall see how such a multi-level model can be used to control the evolution of information systems by facilitating the comparison of heterogeneous systems. For current trends towards more structured programming techniques and more disciplined software engineering environments

[1] UM (the Unified Method) was confidently predicted for the near future at OOPSLA95 but at *Software Development 96* it was admitted that it was only the language and not the methods that were ready for unification. A draft of the unified notation is available at [22].

[2] Although Java checks that a code is valid it then expects all code that passes the validation tests to be properly constructed under the rules. This provides an opportunity for rogue applets to carry into a system any type of computer virus.

are leading to considerable benefits by the use of an IRDS approach. This potential, however, can only be realized if a formal underpinning of the IRDS standard is achieved[19].

Interoperable systems belong to a class of complex objects that require advanced considerations when developing standards. They even raise the question of standards for standards in certain issues: for example, if a system is to be autonomous it must be able to select its own standard. For the system that is truly decentralized there cannot even be a choice imposed by some overall command. There is the need to go beyond the simple concept of a standard to that of the reference model. It seems that the only universal standards are *natural* ones as found in mathematics. There is the particular problem of how to deal with transitions across levels where there are traditional difficulties in selecting appropriate formal methods. The argument here is that only a constructive formalism is adequate for computational systems and this paper takes a constructive approach relying on recent results from category theory.

Current developments from VLSI to ULSI (ultra large scale integration) in hardware with CMOS technology moving to 0.15μ m processing means that in a short time it will be possible to mix diverse functions like analogue and digital logic, power capabilities and non-volatile memory technologies all within a single integrated chip of less than one centimetre square – all calling for internal cross-platform support. Furthermore the sheer complexity and colossal problems facing designers means that previous designs will have to be reused. Plug-compatibility of design macro cells will need to be developed. Verification for the use of recycled designs for business process re-engineering can only be possible by the use of cross-platform multilevel formalisms[20].

One immediate advantage of a formal representation of the IRDS is for use to control transformations in reusable or recycled systems, written in either a database, network or programming paradigm, with consistency and integrity. Indeed IRDS merges these quite different paradigms.

3 Rationale

There is a problem in dealing with layers or levels and transitions between levels in developing reference models. Interoperable systems require reference models rather than just standards. This does not mean that standards are superseded but rather subsumed. For example the ISO family of OSI standards[16] are widely accepted and successfully used as a convention for cooperative work but their value is limited to the syntactical level. For while there is internal consistency in the standard there is no guarantee that the application of the standard will result in a self-consistent system. This need not cause too much concern for implementers in a local system where everything is under their own control. A programming language or a LAN protocol is a close enough approximation to a closed system where little difficulty may be experienced in practice. However, as soon as any kind of openness or independent autonomicity is introduced, another level appears requiring closure at an even higher level. In terms of logic, higher-order is needed: first-order logic will just not do to connect between levels. The result is a need for a meta-meta level. Otherwise cross-platform operation is inadequate.

What is needed is a reference level in its most abstract form which can give this provable ultimate closure. Mathematics gives us this third-level closure through constructive methods for defining the reference model for applications like client/server systems. This need has

been recognized to a limited extent by standards bodies who have produced reference models which relate local standards across a number of levels. However, true reference models are still few and far between.

It is important to bear in mind what is meant by a reference model[3]. If we consider the client/server context as an example, the reference model for OSI is not itself the set of protocols for a communications system. It is a framework for the identification and design of protocols for existing or for future communications systems. It enables engineers to identify and relate together different areas of standardization. OSI does not imply any particular technology or methods of implementing systems. Put simply, the reference model helps engineers to design protocols for computing communications systems.

This multi-level context of the reference model also raises problems in the development of the theory which is always needed to give underlying confidence in matters of consistency and reliability, etc. Formal methods where limited to set-based approaches with axiomatic sets and first-order logic may well be satisfactory in theory for localized systems. However, in practice first-order predicate logic involves considerable mental effort, is difficult to learn and is viewed as an academic language whose application is to some extent an art[10, 12]. Relational and functional theories are much more suited to representation of real-world activities and relationships[17] but have been rather neglected.

Current formal methods lend themselves to partial solutions which may breakdown when their limits are tested; they involve complex manipulations because of their reliance on first order logic; such logic is not natural to practitioners; methods involving relations and functions may be better building blocks as ordering. It may therefore be appropriate to question the faith in formal methods that are not constructive. The interim UK defence standard DEFSTD 00-55 makes the use of formal methods mandatory and this has been followed in other countries. However, experiment has shown that use of formal notation does not lead inevitably to improving the quality of specifications. Unless they are solidly based on empirical evidence, formal methods can be dangerous and costly[6]. It should be emphasized in passing that this need for an empirical basis for formal methods arises because the formal methods employed are axiomatic. Constructive formal methods should be reliable without the support of empirical data which may not be always available.

In addition theories of distributed cross-platforms need characteristics like self-referencing and multilevel closure. It is therefore necessary to look to mathematical operations like functors, adjunctions and natural transformations as found in the constructions of category theory[1].

In this paper we seek to show that the IRDS has a constructive formal basis and can be relied on for this reason. It is therefore a valid method for formalizing a cross-platform perspective of the construction and implementation of information systems.

4 Levels in the Universal Reference Model

As indicated above, it is always necessary to go one level up to address the physical level from a logical standpoint. A minimum requirement is to know names, constraints, typing, etc. In modern client/server systems for example, this is the function of the system catalogue which for a relational database system, for instance, includes descriptions of the relation names, attribute names, attribute domains or types, key i.e. identifying attributes, other

constraints, views, storage structures and indexes. In a network database, information held in the catalogue includes descriptions of record types, set types and physical organization. In a COBOL program, the data division is held in a catalogue giving information on record types, field types and their pictures. In C++ or Java, classes define data structures, methods and their abstractions.

For client/server systems, the specification includes:

- the overall layout of the system including mainframe-centric or network-centric organization;
- hardware components;
- the segmentation and allocation of functions between client and server;
- distributed database design;
- participating client/server operating systems (Unix, Windows 95, Windows 5.x, Windows NT, XWindows, AIX, OS/390, OS/2, etc);
- selection of middleware (CORBA, ODBC, COSE, GUI builders, CDE, etc);
- declaration of network technology (SNA, ISDN, LAN, internet, ETN, TCP/IP, EDIFACT, etc);
- client and server program structure (e.g. documentation of Scripts).

For instance nearly all system catalogues today are *active* in the sense that they are a dynamic automatic source of naming and typing information for programs accessing the system (rather than a passive static reference). This makes the catalogue the cornerstone of the information system, often being subjected to a very high rate of searching activity. Clearly a major advantage of active catalogues is that system type changes are made once in the catalogue rather than replicated through many source programs.

However, a logical level is incomplete without an associated formal reference level to describe how it represents real-world entities, properties, relationships, typing, etc. This reference level deals with meta-meta data. Another way of looking at this level is to view it as dealing with the policy of the model: why is a particular construction in the model available and what semantic capability does it handle?

This comparison of logic and principles is performed against a reference or neutral layer which contains a superset of the facilities for a particular platform or application model. The reference layer contains as data the principles for data and activity representation and for each application the concepts which capture these principles. There is therefore a mapping from principles to schema intension for each application model in turn.

In relational systems with all data held in tables, the meta-meta level is implicitly part of the model. The freeness of the object-oriented paradigm, on the other hand, means that the meta-meta level needs to be constructed explicitly to control the representation of classes, objects, properties, inheritance, composition, methods, etc. In a COBOL system, *wrapper* constructions are increasingly used to provide encapsulation of programs with pre-determined interfaces to emulate object constructions. Other examples are the class of shell scripts and

functions like Korn shells and similar patches for unix or motif widgets in XWindows. However, these shells and wrappers are at the meta level: the meta-meta level would describe the purpose of wrappers and the facilities used for their construction. This also seems to be the crux of the problem with Java mentioned above. Applets are at the meta level and it is the metameta level which gives the overall reliability.

These are particular examples of the message level which provides a top layer of awareness which has to be developed further to deal with the widest possible type of information systems. It was for this kind of purpose that the ANSI Standard Reference Model (X3.138) was developed in the 1980s[8], emerging in 1993[14] as an associated part of the standard for a framework for Information Resource Dictionary System (IRDS) developed in 1990 and 1993[13]. Among early language bindings in IRDS were specifications for Pascal, COBOL, C and SQL. Language bindings can in principle be defined for any language. However little use seems to have been made of the IRDS but this may be because only recently has there been much need to cross levels in systems work.

In the meantime attention has been concentrated on the OSI seven-layer model for open systems because it provides a potential means for commercial suppliers to provide compatible components for different systems. However, while these may be compatible, there is no guarantee that they are consistent. This is because the OSI set of standards together provide a reference model but not a universal reference model. The IRDS on the other hand has the internal characteristics of a universal reference model.

The four levels of the IRDS can be shown to capture the universal nature of the reference model because it can be demonstrated that these can be constructed formally. Before we attempt to show this we should examine closer the four levels and their inter-relations to see how they bear out the discussion so far.

5 The Information Resource Dictionary System IRDS

Before embarking on a full formal description of the IRDS, some understanding and informal insight into its interpretation would be useful. Reference should be made to the diagram in the original standard[13]. The IRDS is constructed on four levels. Each level taken with its adjacent level acts as a *level pair* so that there are three level pairs across the four levels. This means that each point at each level is directly related to a point at the other level in the *level pair* as shown in Figure 1.

level	terminology	instances	interpretation	example (relational model)
1	meta-meta /policy level	usual abstractions (aggregation, composition, inheritance, association, etc)	mission (natural concepts)	real-world abstractions
2	meta/ operational level	abstractions at policy level	organizational (high-level operational/analytical tools)	available constructs
3	Intension level	network features and functions	formal declaration, schema and labels	data names
4	Extension level	data values satisfying the intension	the information itself	data values

Figure 1: Interpretation of Levels in the IRDS

The top level is the Information Resource Dictionary Definition Schema (IRDDS), in which concepts relating to policy and philosophy are defined. For example, object-oriented abstractions are to be declared at this level. In principle, only one instance of an IRDDS need be defined for a platform. In a coherent system there can be only one collection of such concepts. With the open-ended nature of object-oriented structures, however, some extensibility may be required.

The second level is the Information Resource Dictionary Definition (IRDD) in which schema facilities are defined. The context of each platform will have its own IRDD definition. For example a COBOL IRDD would declare that record-types were an aggregation of single- or multi-valued data field-types while one for SQL would declare that table-types were an aggregation of single-valued data fields.

The third level is the Information Resource Dictionary (IRD) which defines the intension for an application, giving names and constraints. There will clearly be many intensions defined in an organization, one for each application. Names, types and other constraints will be given to data objects, network connections, protocol names and signatures, server and client functions, etc.

The fourth level is the Information Resource Data (APP) which gives the extension, the data values. There will be one extension for each intension, the values being consistent with the names and constraints of the intension. Data values may be simple objects as in SQL or complex objects as in computer-aided design and multimedia systems.

The whole diagram represents one platform, paradigm or model. Take as an example the relational model. Level 1 would be real-world abstractions; level 2 would be the constructs available; level 3 would be the data names; and level 4 would be the data values. The four levels with their terms, instances, interpretation and the corresponding components of the relational model can be summarized in the table of Figure 2.

Between each level the mapping, between the *level pair* enables data at one level to be related to data at a lower level. For the same example of the relational data model, the level pair between levels 1 and 2 would be the association of an abstraction (say aggregation), with a constructive facility in the model (say table in this case); the level pair between 2 and 3 would be the association of a construction in the model (say table) with the name of a table; and the level pair between 3 and 4 would be the association of a data name with a data value. Figure 2 shows schematically the mappings between the various levels.

level	IRDS standard	mapping	example of level-pair
1	concepts		aggregation
	↓	Policy	↓
2	constructs		table
	↓	Org	↓
3	intension		name of table
	↓	Data	↓
4	extension		data value

Figure 2: Mappings between Levels in the IRDS

Between each level the mappings are strictly defined by their starting and terminating points in the respective levels. These may not be immediately obvious in the original standards diagram but are brought out in the informal diagram of Figure 3 together with more explicit interpretations of the levels. In particular it should be noticed that the interpretations of the mappings can only be appreciated by considering both directions for each respective mapping. Starting from the top level in the downward direction we can further expand the mappings shown in Figure 2.

- Between levels 1 and 2 (IRDDS and IRDD), there is the mapping *Policy* acting as a level pair. This level pair exists only in IRDS-type systems in which constructive facilities in a system are related to real-world abstractions. For example, *Policy* would indicate how a network-centric capability is made available in a particular approach.
- Between levels 2 and 3 (IRDD and IRD), there is the mapping *Org* acting as a level pair. This level pair provides a standard data dictionary function of, for instance, saying which tables are available in a relational system or which servers are available on a network.
- Between levels 3 and 4 (IRD and APP), there is the mapping *Data* acting as a level pair. This level pair can be thought of as the state of the art of an information system: to link values to names so that data can be addressed by name rather than by physical location.

- Between levels 1 and 4 (IRDDS and APP), there is the mapping *Platform* acting as a level pair. This level pair short-circuits the navigation through four levels by giving a direct mapping from real-world abstractions to data values. The use of this mapping is described later.

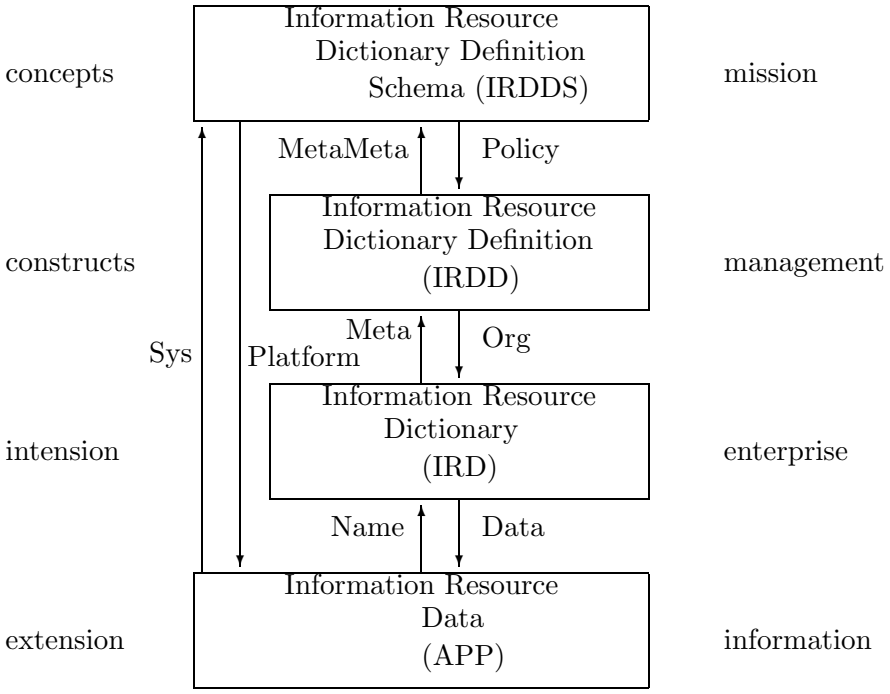

Figure 3: Interpretation of IRDS in Schematic Form

The IRDS standard is the basis for relating heterogeneous systems across platforms, that is systems based on different paradigms. While there is only one instance of the top level (the IRDDS), this level is extensible and new concepts and abstractions can be added as desired. From the point of view of client/servers, the IRDS provides the ability to run an organization with many different paradigms all integrated through the type of structure shown in Figure 3. The critical mapping is *Platform*, that is the arrow from IRDDS to APP, 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 accomodated and run in an integrated fashion. The way that federated database systems are managed[7] is an example of this approach.

The next task is to formalize the diagram in Figure 3 so that a sound scientific basis can be developed for the IRDS model to handle heterogeneous systems.

6 Formalizing the IRDS

Constructive mathematics attempts to develop logically what works in practice and can provide the necessary universality for interoperability of heterogeneous data systems with consistency and quality assurance in the real-world. Category theory[1] is particularly appropriate for modelling multi-level relationships for it is essentially concerned with links between objects.

Category theory provides a universal construction for formalizing information systems. Rather this should read *the* universal construction for the theory shows that there is only one. It is this uniqueness that provides the universalness that provides 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 model. In this approach, each class definition can be identified as a collection of arrows (functions) forming a category **IRD** and each family of object values conforming to a particular class definition as a category **APP**. The mapping from the intension (class definition) to extension (object values) is made by a functor *Data* which enforces the various constraints specified in **IRD**. Category **IRD** is the intension corresponding to the third level in IRDS and **APP** is the extension corresponding to the fourth level.

In reality the intension category **IRD** is a family of categories, representing definitions of classes, associations (relationships) and coproduct structures indicating inheritance hierarchies. The arrows within it may be methods as in object-based systems, network connections between clients and servers, logical connections as in network databases, or functional dependencies as in relational database schemas. It should be emphasised that categorical approaches naturally include procedures and functions through the underlying arrow concept ensuring that both structure and activity can be modelled in a multi-level manner. The category **APP** is also a family of categories, representing object values and association instances. The functor *Data* mapping from the intension to the extension not only connects a name to its corresponding set of values but also ensures that constraints specified in the schema, such as functionalities of relationships, membership classes and functional dependencies, all hold in the extension.

It is relatively straight-forward in category theory to extend the intension and extension two-level structures in a universal manner to handle the four levels of IRDS. In categorial terms each of the four levels of IRDS is defined as a category. 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 **IRDDS** which defines the various abstractions available for modelling real-world data. The next level is a category **IRDD** defining the various construction facilities available for representing abstractions and data in a particular system. There is therefore, for one instance of **IRDDS**, many instances of **IRDD**, one for each type of model (relational, network, COBOL, etc).

The data functor (level pair) *Policy* maps target objects and arrows in the category **IRDDS** to image objects in the category **IRDD** 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 **IRDD** with the names in a particular application in **IRD**. Combining these new constructions with the product ones above gives the direct and universal representation

of IRDS as shown in Figure 4

In category theory, this diagram is a composition of functors with $Platform$ as the overall functor from **IRDDS** $\longrightarrow$ **APP**, such that for each type of information system:

$$Platform = Data \circ Org \circ Policy$$

and

$$Sys = MetaMeta \circ Meta \circ Name$$

In order to relate concepts across platforms, we need to compare the functors $Platform$: **IRDDS** $\longrightarrow$ **APP** for each of our COBOL, network, relational, object-oriented systems, etc. This comparison is a natural transformation.

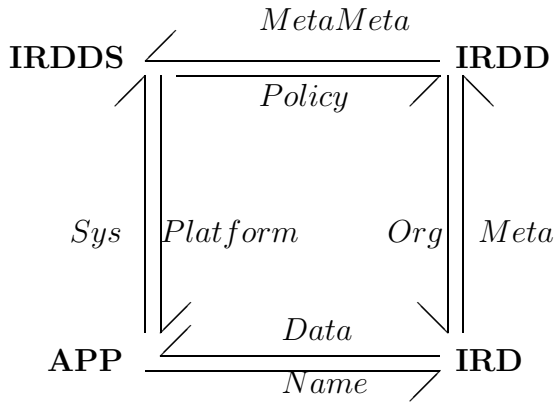

Figure 4: IRDS Levels in Functorial Terms

7 Implementation

As part of this work, an example prototype information system was developed called the Categorical Product Data Model (CPDM) which can formalize, for instance, the object-relational model[18]. The purpose of this prototype was to test the theoretical representation given in section 6 and to examine the ease with which such a representation could be implemented on a computer.

A prototype[15] of CPDM was implemented using the platform of P/FDM[9], an implementation of the functional database model in Prolog, from Aberdeen University. Categories and functors were implemented so that the two categories **IRD** and **APP** could be defined together with the mapping *Data* between them. In the actual work described in [15] the categories **IRD** and **APP** are called **INT** (for intension) and **EXT** (for extension) respectively. It is relatively straight-forward in category theory to extend the product data model CPDM, defined already as a two-level structure, in a universal manner to handle the four levels of IRDS. A further demonstrator project is planned to illustrate the advantages of our approach and to explore further the management of the four-level architecture for assisting in the control of cross-platform operations.

8 Discussion

A formal universal description, based on the ISO standards for Information Resource Dictionary System (IRDS), can therefore be developed and implemented to provide a complete definition of an information system from the physical data values held to the concepts employed for data and function representation and real–world abstractions. Such a multi–level formalism model can be used to control the evolution of information systems by creating an environment where heterogeneous systems can be compared for cross-platform performance. Current trends towards more structured programming techniques and more disciplined software engineering environments can very usefully exploit the IRDS approach. This potential which the IRDS possesses, however, will only be realized for consistent interoperability if a formal underpinning of the standard is achieved.

The constructions of the IRDS in systems offer a certain path for improving our ability to deal with heterogeneity. The universal basis provides a predictable and provable behaviour which is such an important aspect if heterogeneous information systems are to be reliable and productive. Essentially the formal basis for IRDS assists in developing a reference model to relate policy statements in different models and environments with consistency and integrity. This is the crux of interoperability.

A categorial IRDS achieves greater power than that envisaged in the original standards. For instance Spurr[21] comments on the difficulties of using the IRDS standard dictionary with CASE tools because of the lack in IRDS of a natural way of modelling object structures. Our work[15] shows the natural correspondence between categorical databases and object structures making possible a complete and faithful representation of the object-oriented paradigm across the four levels.

A possible difficulty is that this approach may require a complete knowledge of all features of every type of system available. There are systems developed by current procedures of software engineering which are fully documented in machine-readable form at a conceptual level. However, some other systems developed under *ad hoc* procedures are likely to be documented incompletely and in a non-conceptual manner. The top level IRDDS is always available but an IRDD may not be available for all paradigms. There is clearly a limit to the range of paradigms that could be covered in IRDD structures. This limit is the extent to which a paradigm can be conceptualized. IRDD are likely to be defined in reasonable detail provided the paradigm used is not too idiosyncratic, has clearly-defined schema facilities and these have been used in the applications. Old COBOL programs, where data and program divisions are defined, should present no problem here. Assembler programs with *ad hoc* data definition structures are much harder because data structures and typing information only appear as embeddings in character strings.

Even with a clearly-defined IRDD paradigm, there may still be difficulties if, in the original system, there are problems with incomplete or conflicting schema information or code has been written bypassing the conceptual tools of a system. Nevertheless there is no difficulty that cannot be solved (at least in principle) by going to higher levels because of the infinite closure property of the four-level approach.

We therefore see these techniques being more useful in coping with well-defined systems, even if they are using outdated methods, than poorly-engineered historical software where the case for a complete rewrite is obviously stronger. The improvements in languages, for instance Microfocus COBOL and Visual BASIC, and the greater emphasis on formal schema design

in current software engineering both indicate that future problems will be more amenable to the four-level approach.

It is, for this reason, suggested that current trends towards more structured programming techniques and more disciplined software engineering environments should be leading to greater use of this four-level IRDS standard with its formal underpinning. Our conclusion is that theory shows that, for situations where there are heterogeneous systems as in cross-platform operation, client/servers, legacy software, etc, which have to be coherently designed and implemented, there is no other way forward than by this four-level closure path or an equivalent framework to this categorical approach. For such systems presuppose universal techniques to ensure consistency and integrity across time and distribution.

References

[1] Barr, M, & Wells, C, *Category Theory for Computing Science*, Prentice-Hall (1990, 2nd edition 1995).

[2] Cattell, R, *The Object Database Standard. ODMG Release 1.2*, Morgan Kaufmann (1996).

[3] Clements, Alan, *Standardization for Information Technology,* BSI Catalogue no: PP 7315, at page 26 (1987).

[4] *The Common Object Request Broker: Architecture and Specification*, Object Management Group, Framingham, Mass. (1993).

[5] Dean, D, Felten, E W, & Wallach, D S, Java Security: From HotJava to Netscape and Beyond, *Proc. Symp. Security and Privacy*, IEEE CS Press, 190-200 (1996).

[6] Fenton, N, & Pfleeger, S l, Can Formal Methods Always Deliver?, IEEE Computer **30**(2) 34 (1997).

[7] Fiddian, N J, Gray, W A, Ramfos, A, & Cooke, A, Database Meta-Translation Technology: Integration, Status and Application, *Database Technology* **4**, 259-263 (1992).

[8] Gradwell, D, Developments in Data Dictionary Standard, *Computer Bulletin*, September 1987.

[9] Gray, P M D, Kulkarni, K G, & Paton, N W, *Object-Oriented Databases: A Semantic Data Model Approach*, Prentice Hall (1992).

[10] Gries, D, The Need for Education in Useful Formal Logic, *IEEE Computer* **29**(4) 29-30 (1996).

[11] Heather, M A, & Rossiter, B N, Content Self-awareness in Distributed Multimedia Publishing: the Need for a Unifying Theory, in: Third International Workshop on Principles of Document Processing (PODP'96), ed. Nicholas, C, & Wood, D, *Lecture Notes in Computer Science* **1293** Springer-Verlag 35pp (1997).

[12] Holloway, C M, & Butler, R W, Impediments to Industrial Use of Formal Methods, *IEEE Computer* **29**(4) 25-26 (1996).

[13] Information technology - *Information Resource Dictionary System (IRDS) framework*, Standard ISO/IEC 10027 (1990); 10728 (1993).

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

[15] Nelson, D A, & Rossiter, B N, Prototyping a Categorical Database in P/FDM. *Proceedings of the Second International Workshop on Advances in Databases and Information Systems (ADBIS'95)*, Moscow, 27-30 June 1995, Springer-Verlag Workshops in Computing, edd. J. Eder and L.A. Kalinichenko, ISBN 3-540-76014-8, 432-456 (1996).

[16] Standards dealing with OSI (Open Systems Interconnection) include BS ISO/IEC TR 9571 to 9596 and BS ISO/IEC TR 10162 to 10183.

[17] Parnas, D L, Mathematical Methods: What we Need and Don't Need, *IEEE Computer* **29**(4) 28-29 (1996).

[18] Rossiter, B N, Nelson, D A, & Heather, M A, *The Categorical Product Data Model as a Formalism for Object-Relational Databases,* Technical Report, Computing Science, Newcastle University, no.505, 41pp (1995).

[19] Rossiter, B N, & Heather, M A, *Data Modelling for Migrating Information Systems*, chapter 1, in: Legacy to Client/Server – Have You Chosen Wisely?, ed. Booth, A, Unicom, London 1–12 (1996).

[20] Sangiovanni-Vincentelli, Alberto, The Methodology of Formal Verification, *33rd Design Automation Conference,* Las Vegas Convention Center (1996).

[21] Spurr, K, CASE Tools, Does the ISO Standard IRDS provide Sufficient Support? in: *Fourth Generation Systems*, ed. S.Holloway, Chapman and Hall, Unicom 36-47 (1990).

[22] UML 1.0 at web address http://www.rational.com/uml/