Distributed Data Interfaces - the Lessons of IMDAS

Ed Barkmeyer, National Bureau of Standards

Background: Sharing Data in a Manufacturing Complex

In modern manufacturing systems, two developments are paramount: flexible automation - computer systems controlling and monitoring the physical processes, and Computer Integrated Manufacturing (CIM) - direct data sharing among production control systems and the engineering and administrative systems that support them. The resulting data systems engineering task is to provide access from many systems to the many sources of manufacturing data.

In most industrial facilities, existing control, engineering and administrative systems operate on computer systems from many different manufacturers and use many different data systems. They have existing independently designed, and therefore overlapping, databases, further complicated by logical and physical differences in the representation of the same real-world objects from database to database. But the most important characteristics of these systems are that they are *the* data repositories for the current manufacturing enterprise, that the existing application programs depend on them, and that the validity of their data depends on those application programs. The ideal integrated data system, therefore, co-operates with the existing applications on the existing databases, while enabling new application programs to be built which use these databases in new or improved ways, expanding the capabilities of the enterprise, and which are insulated from accidental distinctions in data location, representation and access mechanisms.

The Integrated Manufacturing Data Administration System (IMDAS) [Bark86, Su86b, Kris87] is a prototype of such a system. It allows application programs to access existing databases distributed over many computer and database systems for both update and retrieval, via a single common interface (Figure 1).

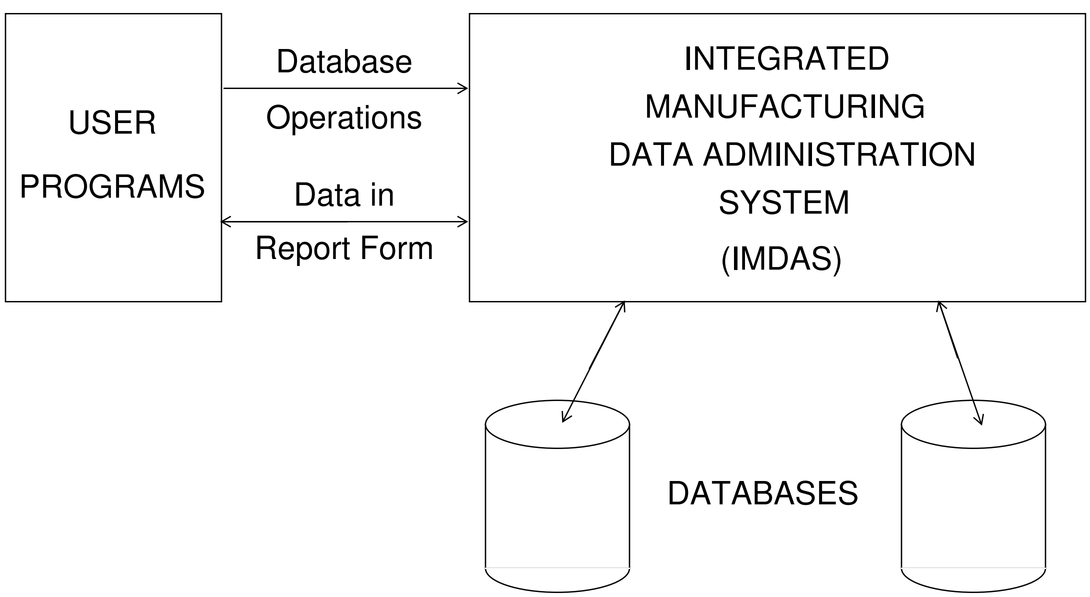

Figure 1: IMDAS Concept

IMDAS Features

IMDAS was developed to support the NBS Automated Manufacturing Research Facility (AMRF) [Nanz84] - a testbed for small-batch manufacturing automation and in-process measurement, funded by NBS and the Navy Manufacturing Technology Program. Certain aspects of the IMDAS design were significantly influenced by characteristics of the AMRF itself: use of a powerful integrating model, development of a flexible control architecture with modular components and standard interfaces, and expeditious motion of data.

IMDAS uses the Semantic Association Model - SAM* [Su83, Su86a] - as the integrating model. A SAM* model is a semantic network, capable of representing the complex structures and relationships and many integrity constraints found in the manufacturing enterprise. IMDAS uses SAM* to model the *conceptual* data, the abstract objects and information units, and a *fragmentation schema* to map the modelled data objects to the actual data units stored in the various underlying databases.

The user program phrases data operations in a data manipulation language (DML) which superficially resembles SQL [ANSI86a], but since the model being manipulated is not purely relational, the syntax and semantics of the IMDAS DML contain significant modifications. In particular, a user operation establishes a viewpoint in the semantic network from which all the data referenced can be seen as a "generalized relation", the elements of which can be simple types, or structured data, or sets or embedded tables. The interpretation of a DML command is then expressed in terms of these generalized relations. In addition, the user program can specify that any particular data element (or set or table) used in the operation is to come from, or be delivered to, a file local to the user, or a shared memory area available to both the user and the data system, and the external form in which the generalized relation or data unit is (to be) represented.

Internally, the IMDAS represents a user transaction as a tree of elementary operations on the generalized relations which result from the transaction viewpoint being imposed on the semantic network. It then maps this tree onto the underlying "relations" or "sets" in the databases which actually contain the corresponding data. Technically, the "tree" may at some point become a "forest", when maintaining global data integrity requires related transactions to be spawned, or when the distribution of data requires the transaction to be decomposed into subtransactions sent to different systems.

Existing database systems are "front-ended" by IMDAS modules supporting an "interchange query form", which is a representation of these transaction trees, and an "interchange data form", which is a representation of the generalized relations themselves. The front-end modules, called "Command and Data Translators", or "CTs", translate the transactions from the interchange form to whatever language and interface the particular database management system supports. The CT also translates inbound data from the interchange form to the form expected by the data manager and outbound data from the form the data manager delivers to the IMDAS interchange form.

Like the AMRF manufacturing complex, the IMDAS is a hierarchical control system, and within it, control is separated from data. Control, in the form of commands and status, flows through the hierarchy, but data flows directly between data repositories as directed by the commands. User data areas (files and shared memory) mentioned in user commands are simply additional data repositories to and from which data can flow, and for them, "automatic" replication of certain data is also supported. The objective of this feature is to move data directly from producer to consumer, with as little overhead as possible, thereby facilitating "real-time" application [Mitc84, Libe85].

IMDAS Architecture

The nominal architecture of IMDAS is the 4-level hierarchy shown in Figure 2.

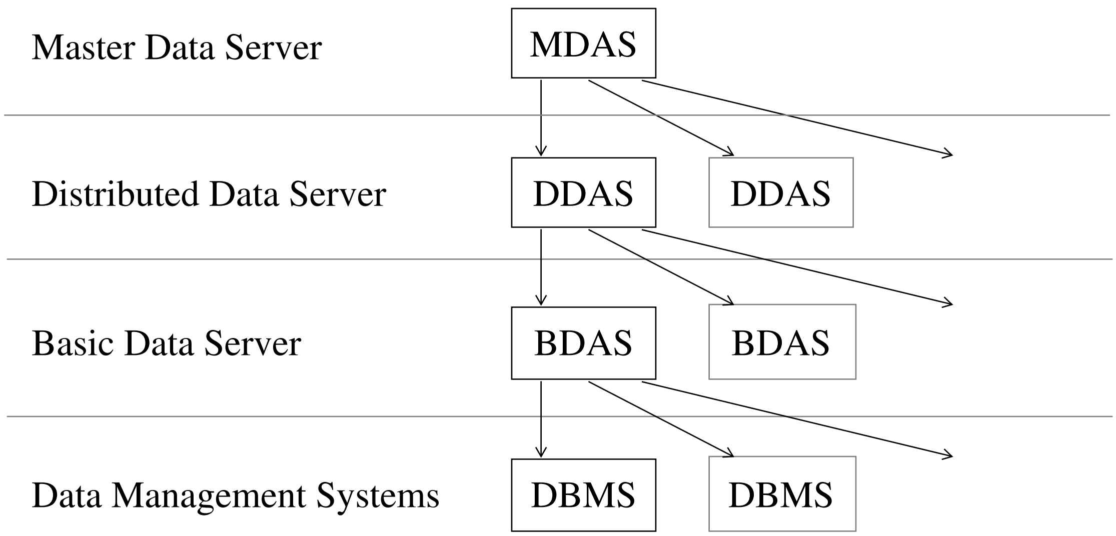

Figure 2: The IMDAS Hierarchy

The lowest level of architecture comprises the data repositories - databases, files, controller memories - managed by commercial DBMS, file systems, home-grown application-specific servers, etc. These are the existing data systems on which the IMDAS depends. Each computer system in the enterprise has a Basic Data Server (BDAS), which provides the interface between the local repository managers and the integrated data system. It contains the front-end processes which provide the standard interfaces for the local DBMSs. The BDASs, and the DBMSs, are the elements which *execute* the data manipulations.

The Distributed Data Servers (DDAS) perform the query processing and transaction management functions. Each DDAS provides the query processing interface to all application programs within a cluster of computer systems which are its segment of the enterprise, and logically integrates the collection of data repositories managed by the BDASs in that cluster into a corresponding segment of the global database. The DDASs *manage* the data manipulations.

The Master Data Server (MDAS) integrates the separately managed segments into the global database, and manages transactions which cross segment (i.e. DDAS) boundaries. The MDAS is a utility used by the distributed controllers to resolve the global model and provide concurrency control for transactions which involve multiple DDASs. It does not *manage* the fully distributed system so much as it *coordinates* it.

The IMDAS modular architecture permits several "distributed data system architectures" to be built from the same components. A system with exactly one DDAS and one BDAS is essentially a centralized system, while a system with one DDAS and several BDASs is a distributed system with centralized control. A system with multiple DDASs is a distributed system with distributed control. The implied growth path was exactly that of the AMRF IMDAS.

IMDAS Operation

An application program issues a transaction to the IMDAS in the SQL-like data manipulation language, specifying source and destination data areas. The DDAS representing the IMDAS in that cluster (Figure 3) accepts the transaction and the query processor converts the transaction, expanding application-specific views, into elementary operations on the conceptual generalized relations. If the resulting transaction tree can be executed entirely within the segment managed by that DDAS, it is passed to the DDAS transaction manager; otherwise, it is sent to the MDAS transaction manager.

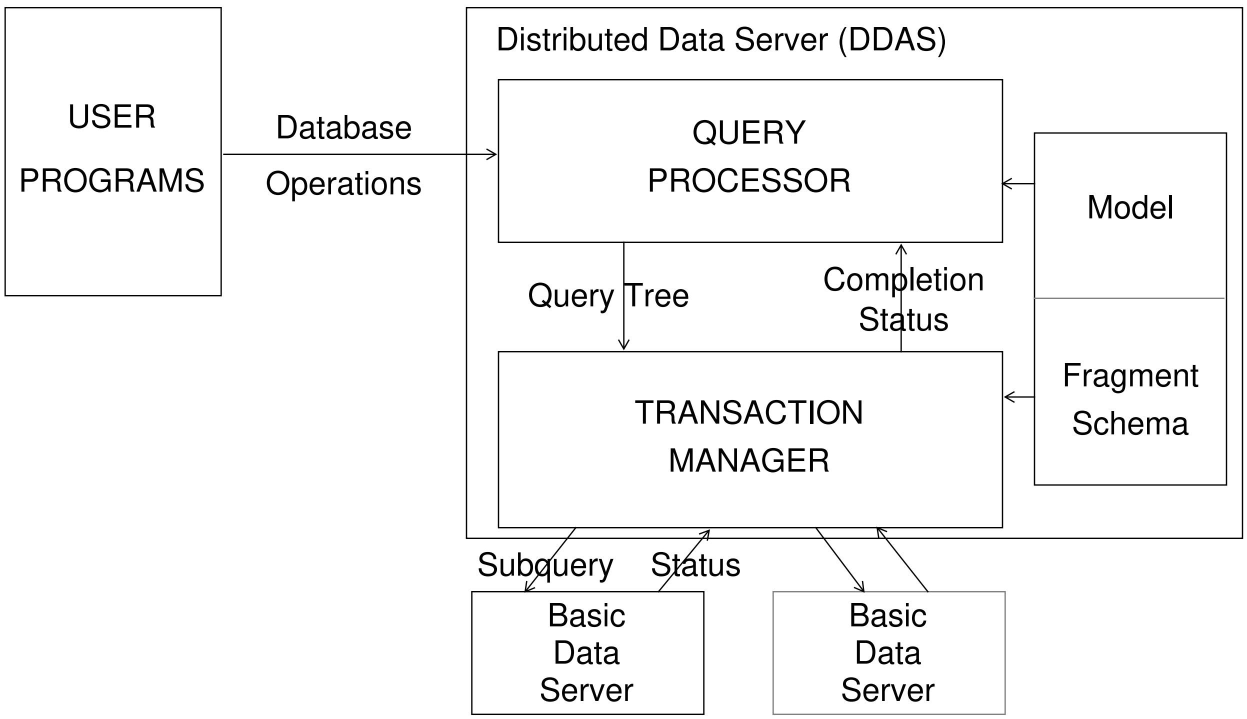

Figure 3: Distributed Data Server

The query mapping service in the DDAS transaction manager consults a fragmentation schema describing the "physical" distribution of its segment of the global model, and maps the transaction into a set of "subqueries", each of which operates on elements of the global database managed by an individual DBMS. The mapping algorithm takes into account the capabilities of the target DBMS, and operations which exceed the capabilities of the repository DBMS are routed to a sufficiently capable DBMS. or to a special-purpose DBMS front-end called the "Data Assembler", with some of the "base relations" specified to be the generalized relations output from other repository systems. This is also the mechanism by which information units from multiple DBMSs are integrated. The forest of subqueries is then passed to the scheduler in the transaction manager for scheduling, sequencing and dispatch to the affected BDASs. The transaction managers use two-phase locking to prevent integrity problems from read/write or write/write conflicts for access to the same "subrelations" and nominally use a two-phase commit protocol for distributed updates. When the whole transaction is completed, the transaction manager reports completion status to the query processor, which in turn reports to the originating user program.

An affected BDAS (Figure 4) receives subqueries from the DDAS in the interchange form, specifying the operations to be performed on the local data repositories and the sources and destinations of the associated data. The Basic Service Executive (BSE) accesses referenced local input user data areas (files or shared memory), converting between the user-specified representation and the IMDAS interchange form. The BSE also accesses required remote data areas, by direct communication with the remote BSE. When all of the data required for the transaction is available locally in interchange form, the BSE releases the subquery to the Command Translator front-ending the designated database.

The CT converts the transaction to the form appropriate to the local DBMS, and any input data from the interchange form to the DBMS form, and passes the operations to the DBMS by whatever interface the DBMS demands. If the transaction produces data, the CT converts the results back to the interchange form in the area designated by the BSE and reports completion. If this is the final output, the BSE will then convert the results to the user-specified form in the user-specified area, locally or remotely by cooperation with a remote BSE. When the subquery and any final deliveries are completed, the BSE reports completion to the DDAS transaction manager.

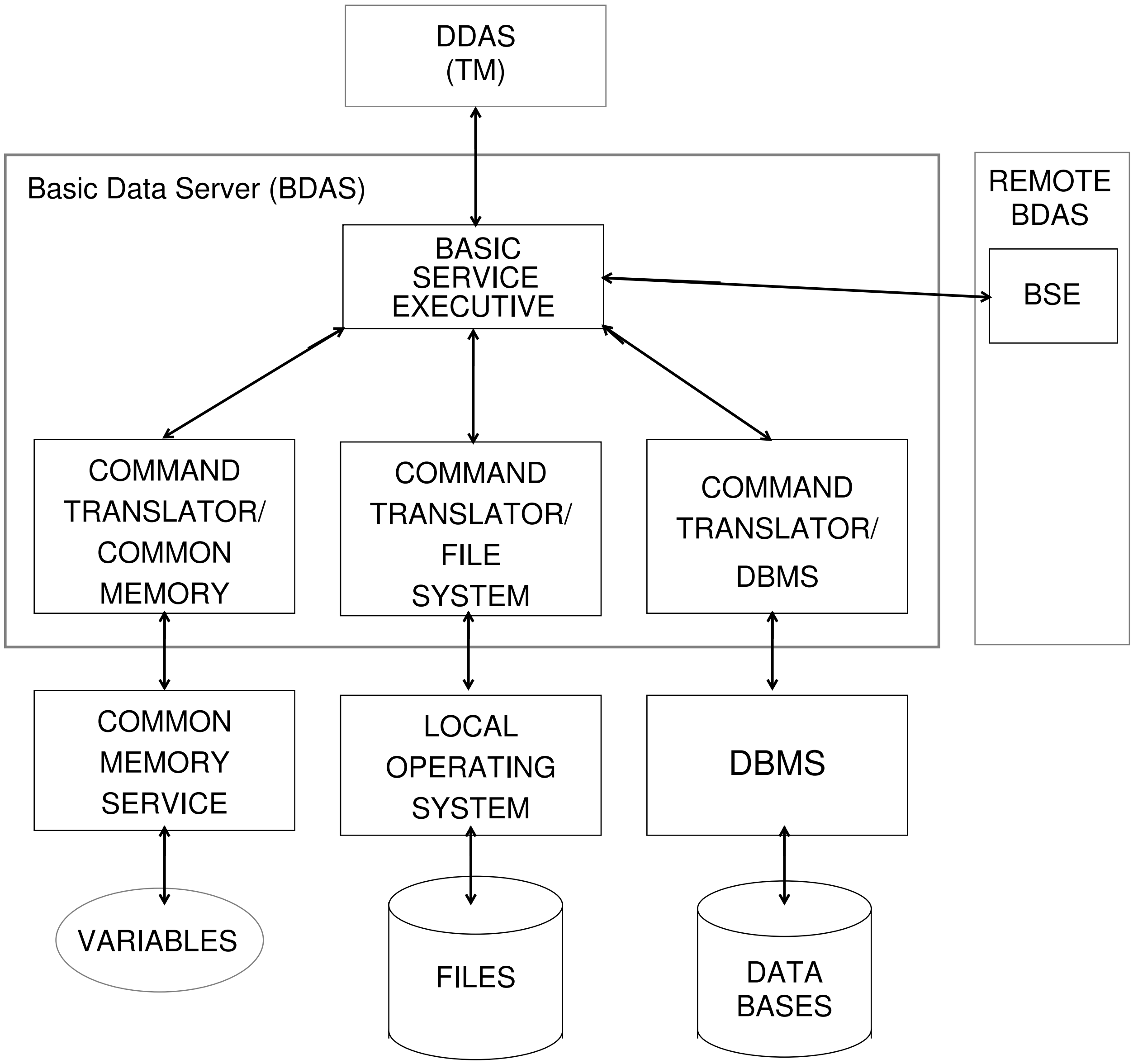

Figure 4: Basic Data Server

When there is more than one operating DDAS in an enterprise, it becomes necessary to create an MDAS (Figure 5). The MDAS is essentially a DDAS transaction manager with a fragmentation schema which describes the distribution of the global model over the DDASs, instead of the underlying DBMSs. The MDAS accepts transactions from, and reports status to, the individual DDAS query processors; and it sends subqueries to, and receives status reports from, the individual DDAS transaction managers (Figure 6). Since the MDAS is a clone of the DDAS transaction manager, it can be instantiated in any station which has a DDAS and thus readily replaced in the event of failure.

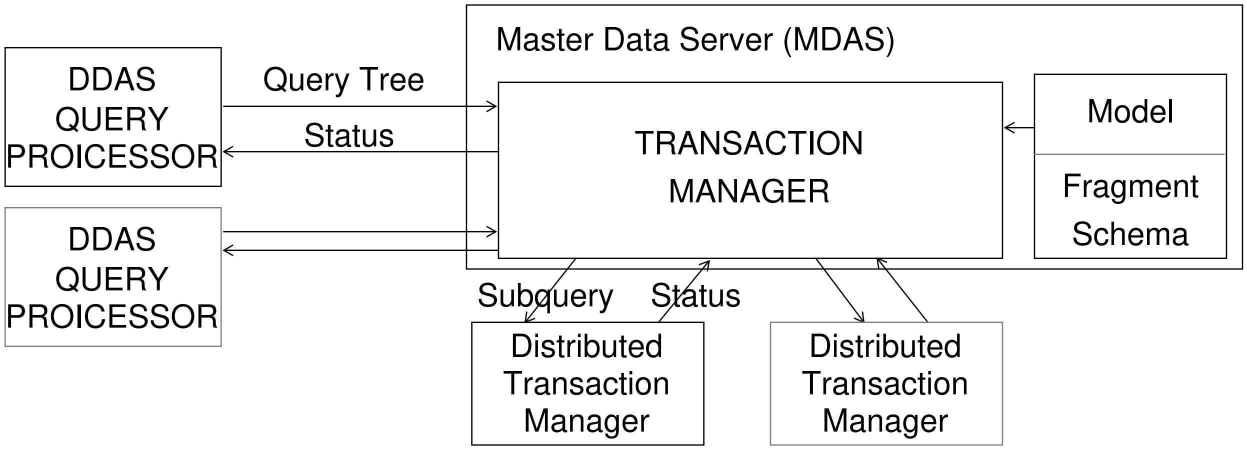

Figure 5: Master Data Server

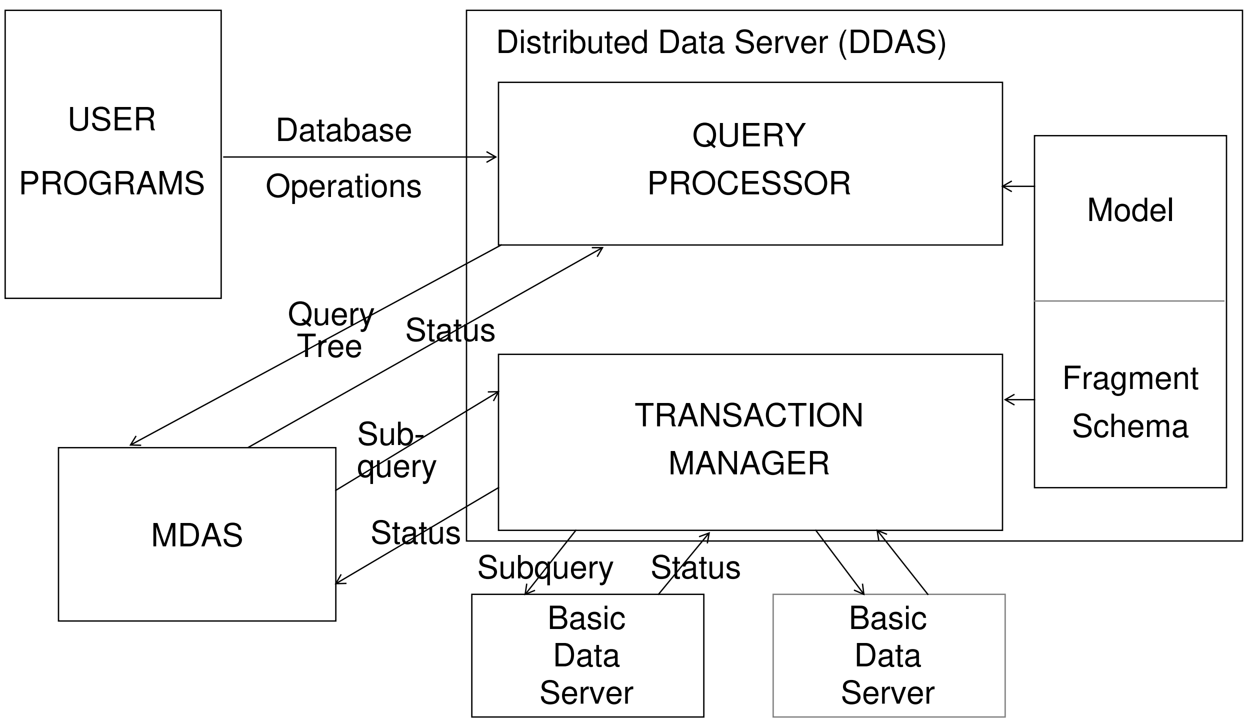

Figure 6: MDAS Transaction Flow

IMDAS Lessons for Interface Standardization

The IMDAS, like any distributed data system, has the following four visible interfaces:

1) the user/data-server command and status interface;
2) the user/data-server data input and output interface;
3) the distributed-server/local-server command and status interface; and

4) the distributed data interface.

The IMDAS has one more visible interface, which is peculiar to its architecture and therefore unrelated to standardization issues:

5) the query-processor/transaction-manager command and status interface.

It is common to bind the user data interface to the user command and status interface and, similarly, to bind the distributed data interface to the distributed command and status interface. The SQL standard, for example, attempts to do the former, and the proposed Remote Database Access Protocol [RDA87] clearly does the latter. Such a binding is neither necessary nor desirable. In the first place, the IMDAS is an example of a viable architecture in which data *does not travel the same paths* as command/status; so for IMDAS, binding data to command/status is wholly unworkable. But even for those systems in which the data does travel the same path as command/status, the nature of the objects in the two interfaces and the functions of those objects are *entirely different*. This means that the nature of the services which develop and use these information units cannot be the same, even if they are co-resident. We note that the proposed Manufacturing Messaging Service (MMS) [EIA87], by comparison, does not make this mistake: it separates the transmission of data, such as a numerical control program, from transmission of the command to use it, precisely because it permits the data to come from a different service, or a different site, or at a different time. The IMDAS view and the MMS view are based on the same manufacturing automation principle: To obtain maximal power and flexibility, you must separate data from control.

A second common failing, which appears in the SQL and NDL [ANSI86b] standards and in the RDA proposal, but not in the MMS, is extensive specification of the command language accompanied by a very weak specification of status reporting. This is based on the view that all "successful" results are the expected ones and all "failures" are equally unexpected. In the particular case of data service requests, the fact that a data object is not present, or already present, or multiple records, may be an expected, or at least recoverable, occurrence. But while a user at a terminal can be expected to understand any reasonable English phraseology of such a result, an application program needs a specific value in a specific place to distinguish not only "success" from "failure", but also the nature of the failure, or in some cases the nature of the "success", if it is to be recovered. Thus any command/status protocol must completely specify the "status" language as well as the command language. We therefore refer to these interfaces as "command/status" interfaces and not just "command" interfaces.

In implementing the user/data-server command and status interface, the IMDAS intentionally departs from the conventional wisdom. Most data systems preprocess SQL or NDL statements embedded in application programs into local data manipulations and subroutine calls to the external entry points for the data service. The IMDAS accepts and processes the DML string at run time. The user codes a subroutine call to whatever local communication service delivers the string to the IMDAS. While this method is less efficient, it has the advantage of allowing the same language in the same form to be used in COBOL programs in the administrative areas, in LISP or FORTRAN or Pascal programs in the engineering areas, and in FORTH or BASIC or Ada or C programs in the controllers! Thus the integrated database is made instantly accessible to application programs anywhere in the manufacturing facility without requiring development of a preprocessor for each particular system and language. Preprocessors, by comparison, are available for only a small number of systems and languages - so a data system depending on preprocessors is automatically isolated from *most* of the systems in a manufacturing complex. There are simply too many systems and languages for this approach to be generally workable. Particularly for controllers, some form of DML which can be encapsulated in a character string and nonetheless provide data location information must be provided. The shared memory and shared files features, which are provided for in the proposed MMS and built in to the AMRF, may be the key to a workable solution.

Contributions to the Current Standardization Effort

The most critical part of the user/data-server command interface is the data manipulation language. The existing SQL and NDL standards provide adequate solutions for the relational and navigational data models, respectively - two of the three common data organization models. The IMDAS is representative of the new breed of data systems based on semantic network models. There are still serious problems in the design and definition of data manipulation languages for such models; so any worthwhile standard in this area is still several years in the future. Some immediate improvement could be made to the definition of status languages, and a "unified" status language is probably possible. The existing IMDAS status language, among others, may have something to offer to that effort.

The user data interface cannot be standardized. Each user program must have the ability to require external data formats to be something which is comfortable for that program and language to manipulate. Location information can have at most standard keywords; the actual data exchange mechanism is necessarily system-dependent.

For the distributed-server/local-server command and status interface, there are few opportunities for standardization. The proposed Remote Database Access protocol may serve the purpose, but it is currently completely specified only for systems which are based wholly on a pure relational organization model. Moreover, it was not originally intended as a distributed service protocol, but rather as a method of transmitting an operation on a database to another system. As a consequence, it has a problem in associating views with operations. A distributed data server expects its operations to associate directly to the remote conceptual model, while a user expects his operations to associate to his external views of the conceptual model. Since these are two distinct levels of interface and since they directly affect the interpretation of the transferred operation, the proposal should address this issue, and does not currently do so. In addition, the RDA needs work on separation of data from control, on status language formalization, and on distributed update features - concurrency control definition and commitment protocol. But all of these are actually ongoing, so a useable near-term standard is likely.

Although a place has been reserved in the RDA for navigational or semantic network data models, no effort toward that standard has yet been expended. The IMDAS "interchange query form" is very closely coupled to the IMDAS model and approach, and is not a candidate for standardization in any way. Unlike the thoroughly tested relational model, operating directly on a semantic model is an untested concept which is currently the subject of much research in data systems built around object models [Ege87, Su88] and semantic network models [Mark87]. Standardization in this area must await an accepted methodology. And the pure relational version of the Remote Database Access proposal *will not be appropriate* for data systems based on navigational or semantic models.

On the other hand, IMDAS data interchange form may be a good straw-man for a database *data* interchange standard. This is primarily because the generalized relation can accommodate arbitrarily complex data structures and be readily mapped to and from relational, navigational and object-oriented data organizations. It may therefore be expected to apply to arbitrary underlying data systems. In the interchange form, the generalized relation is prefixed by a syntactic definition of the data to follow, and then encoded following the Basic Encoding Rules for the ISO Abstract Syntax Notation 1 [ISO87], thus corresponding to existing international standardization efforts for presentation of arbitrary data units in communication.

References

[ANSI86a] American National Standards Institute, X3.135: "Database Language - SQL", December, 1986.

[ANSI86b] American National Standards Institute, X3.133: "Database Language - NDL", December, 1986.

[Bark86] Barkmeyer,E., Mitchell,M., Mikkilineni,K., Su,S.Y.W. and Lam,H., "An Architecture for an Integrated Manufacturing Data Administration System", NBSIR 863312, National Bureau of Standards, Gaithersburg, MD, January 1986.

[Ege87], Ege, A., and Ellis, C., "Design and Implementation of GORDION: an Object Base Management System, IEEE Computer Society, Proceedings of the Third International Conference on Data Engineering, Los Angeles, California, February, 1987.

[EIA87] Electronic Industries Association, Standardization Project 1393A: "Manufacturing Messaging Standard Service Specification and Protocol", draft 7, August 1987, unpublished.

[ISO87] International Standard ISO 8824 "Information processing systems- Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), and
International Standard ISO 8825 "Information processing systems- Open Systems Interconnection - Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), International Organization for Standardization, 1987.

[Kris87] Krishnamurthy, V., Su, S.Y.W., Lam, H., Mitchell, M., Barkmeyer, E., "A Distributed Database Architecture for an Integrated Manufacturing Facility", Proceedings of the International Conference on Data and Knowledge Systems for Manufacturing and Engineering, pp. 4-13, October 1987.

[Libe85] Libes, D., "User-Level Shared Variables," Proceedings of the Summer 1985 USENIX Conference, Portland, Oregon, June 1985.

[Mark87] Mark, L., "The Binary Relationship Model - 10th Anniversary", University of Maryland Institute for Advanced Computer Studies, Technical Report UMIACS - TR-87-50, October, 1987.

[Mitc84] Mitchell, M. and Barkmeyer, E., "Data Distribution in the NBS AMRF", Proceedings of the IPAD II Conference, Denver, CO, April, 1984.

[Nanz84] Nanzetta, P., "Update: NBS Research Facility Addresses Problems In Set-ups for Small Batch Manufacturing," Industrial Engineering, pp 68-73, June 1984.

[RDA87] International Organization for Standardisation, ISO/TC97/SC21/WG3 - N1926, "Remote Database Access", 3rd working draft, July, 1987, unpublished.

[Su83] Su, S.Y.W., "SAM*: A Semantic Association Model for Corporate and Scientific/Statistical Databases", Journal of Information Sciences #29, 1983, pp. 151-199.

[Su86a] Su, S.Y.W., "Modeling Integrated Manufacturing Data Using SAM*", Proceeding of GI-Conference on Database Systems for Office, Engineering and Science, Karlsruhe, Federal Republic of Gernany, March 1985, reprinted in IEEE Computer, Vol. 19, No.1, January, 1986.

[Su86b] Su, S.Y.W., Lam, H., Khatib, M., Krishnamurthy, V., Kumar, A., Malik, S., Mitchell, M., Barkmeyer, E., "The Architecture and Prototype Implementation of an Integrated Manufacturing Database Administration System", Spring COMPCON 1986.

[Su88] Su, S.Y.W., Krishnamurthy, V., Lam, H., "An Object Oriented Semantic Association Model (OSAM*) ", to appear as a chapter in *AI in Industrial Engineering and Manufacturing: Theoretical Issues and Applications*, edited by Kashyap, R.L., Kumara, S., and Soyster, A.L., American Institute of Industrial Engineers, 1988.