A.E.B.Ruano, P.J.Fleming & D.I.Jones (1992), Connectionist approach to PID auto-tuning, *IEE Proceedings - Part D*, Vol.139, No.3, pp.279-285.

F.G.Shinskey (1979), "*Process Control Systems*", 2nd edition, New York, McGraw-Hill.

R.W.Swiniarski (1990), Novel neural network based self-tuning PID controller which uses pattern recognition technique, *ACC*, pp.3023-3024.

K.Warwick, G.I.Irwin & K.J.Hunt (Eds.) (1992), *Neural networks for control and systems*, IEE Control Engineering Series, Peter Peregrinus, London, UK.

M.Yuwana and D.E.Seborg (1982), "A new method for on-line controller tuning", *AIChE J.*, Vol. 28, No.3, pp. 434-440.

C.Valdebenito, D.Sbarbaro & J.P.Segovia (1995), "Gaussian networks for pattern based adaptive control", *Proceedings of the 3rd European Control Conference*, Vol.2, pp.1185-1190, Rome, Italy, September.

M.J.Willis & G.A.Montague (1993), Auto-tuning PI(D) controllers with artificial neural networks, *Proceedings of the IFAC 12th World Congress*, Sidney, Australia, Vol.4, pp.61-64.

J.G.Ziegler & N.B.Nichols (1942), "Optimum settings for automatic controllers", *Transactions of ASME*, Vol.64, No.11, pp.759-768.

J.G.Ziegler & N.B.Nichols (1943), "Process lags in automatic control circuits", *Transactions of ASME*, Vol.65, pp.433-444.

A REFERENCE MODEL FOR THE PROCESS CONTROL DOMAIN OF APPLICATION

Nirvani Dhevcharran, Prof. A L Steenkamp and Dr V Ram

Abstract

The increasing complexity of large process control systems impedes efforts to construct, co-ordinate and monitor these systems effectively. The object oriented paradigm and associated mechanisms greatly assist in software development in such domains. A reference model suitable for the process control domain and embodying object orientation, is described.

Introduction

The development of software systems is a complex undertaking, involving both technical and managerial considerations. In the case of large and complex application domains, development projects are conducted by following a team approach. The diverse perspectives of team participants call for a frame of reference which facilitates understanding of the specific aspects of software development.

A project for researching all relevant aspects when developing software systems, the Object Oriented Information Systems Engineering Environment (ISEE), has been formulated by Steenkamp[1995]. A general framework of four reference models are investigated: Development Process Reference Model, Quality Reference Model, Technology Reference Model and Target System Reference Model.

Process control deals with the technical, economic and safety dimensions of applications such as those found in chemical, oil and gas refineries and municipal water and sewage treatment. Each of these industries has data to be acquired, data to be disseminated, decisions to be processed, communications to be performed and reports to be generated. The challenge is to develop a control system that addresses the tremendous but unique complexity of each application without reinventing a thousand new wheels each and every time as described by Beam[1993].

According to Brown [1992], a reference model is a conceptual and functional framework which helps experts to describe and compare systems. It allows experts to work productively and independently on the development of standards for each part of the reference model. A reference model is thus not a standard itself; it should not be used as an implementation specification, nor as the basis for the conformance of actual implementations.

This paper concentrates on developing a reference model for the process control domain of application within the context of The Target System Reference model. The following aspects have been identified for the reference model: Environment, Information, Software Engineering and Systems Engineering.

The environment aspect : The Environment Aspect occupies the highest level within any methodology. For process control systems this aspect considers the structure of the process control environment in terms of its interface to the real world as well as its interface to physical devices.

The systems engineering aspect : The main focus of this aspect is on those engineering principles relevant at the operating and networking level. There are a number of principles involved in the process control domain, viz., distribution transparency, dependability, performance optimisation, scaling and language integration.

The software engineering aspect : This aspect expresses the target system in terms of the interacting object metaphor. The intra-object structure, inter-object relationships, the functionality of the objects

and the dynamic behaviour of objects are addressed. More specifically, the timeliness issue, the dynamic internal structure, the reactiveness issue, the concurrency issue and the distributed issue of the process control domain are described. Further, formalisms are required to model the structure of the system, of the data and of control.

The information aspect : The manner in which information is used within the process control domain is modelled. Typically, information has to be represented, manipulated and interpreted. The representation of data in this domain is normally in a real time database. This data has to be logically and temporally consistent. The latter arises from the need to preserve the temporal validity of data items that reflect the state of the environment that is being controlled by the system.

The process control domain is intrinsically very complex. It operates in real-time with a number of processes operating concurrently. The Object Oriented (OO) paradigm for software development supports the development of more flexible, more easily maintainable and less error-prone process control systems.

By making optimal use of the powerful OO features and further by establishing a reference model for the process control environment, a structured framework for software development is created.

Reference Model for the Process Control Environment

Figure 1 represents the process control meta model conceptually in terms of the relevant meta primitives for each of the four aspects that have been identified.

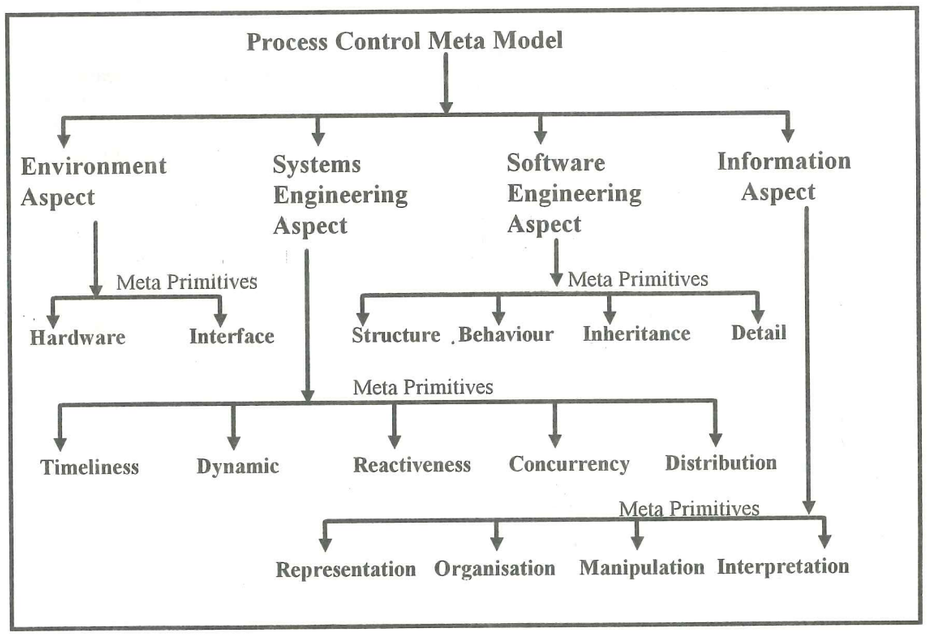

Figure 1. Process Control Meta Model

Meta Primitives of the Environment Aspect

Careful consideration must be given to configuring the environment within which a process control target system will reside in (hardware considerations) as well as to its interfaces (interface considerations) to the real-world. It is necessary to focus on the environment because it forms the basis for constructing a model of the behaviour of a system. The meta primitives are depicted in Figure 2.

Hardware considerations : Process control applications operate in real-time. Time is a critical factor in these systems and there are a few hardware issues that can affect the response/timing of an event, e.g. event scheduling and synchronising of clocks. A difficult problem is the determination of tight timing information on instructions and code sequences for contemporary computers; pipelining, caching and a host of other performance-enhancing features seen to hinder timing predictability. By selecting appropriate granularities for the higher level language elements, these and other hardware issues, such as exactly how and where to incorporate worst-case effects of memory and bus contention, can be handled practically.

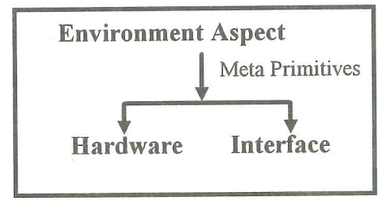

Figure 2. Environment Meta Model

Interface Considerations : The interfaces to a process control application can be either humans or devices. An example of a device is a PLC. There is continuous interaction between the application and the PLC which the system must be capable of handling. Simultaneously, one or more operators may be attempting to access the application. There is therefore, a need for the application to be capable of handling all the PLC as well as the operators transactions, i.e., need for concurrency. The interface between the operator and the system is on-line and interactive. Hence, the response time must be fast. If the same data is being accessed/updated by both the PLC and the operator, the access/update must be synchronised. If there is no synchronisation there is a potential for inconsistent data to be stored on the system.

Meta Primitives of the Systems Engineering Aspect

This aspect focuses on those engineering principles relevant at the operating system and networking level [Steenkamp1995]. The space/time trade-off, representing a particular set of options in the form of parameters, must be determined.

The salient properties which form the meta primitives, depicted in Figure 3, characterising the systems engineering aspect of process control systems are:

Timeliness of function : This is the most common attribute of real time systems of any kind. By definition, a real time system is required to perform its function "on time", whatever that happens to mean in a particular context.

Dynamic internal structure : Many real time systems are required to exercise control over an environment whose properties vary with time. This requires that the system components dealing with

the particular aspects of the environment must be dynamically reconfigured to match the dynamic of external environments. Because of limited resources (memory, processor capacity), this typically entails the dynamic creation and destruction of software components.

Reactiveness : A reactive system is one that is continuously responding to different events whose order and time of occurrence are not always predictable.

Concurrency : This is a feature of the real world in which a real time system is embedded. At any given time, multiple simultaneous activities can be taking place in the real-time system. When this is combined with the need for real time response, the usual result is that the real-time system itself must be concurrent.

Distribution : A distributed computing system is one in which multiple computing sites co-operatively achieve some common function. Distribution is either inherent (as is the case for communications systems) or it may be driven by the need to increase throughput, availability or functionality [Selic1994].

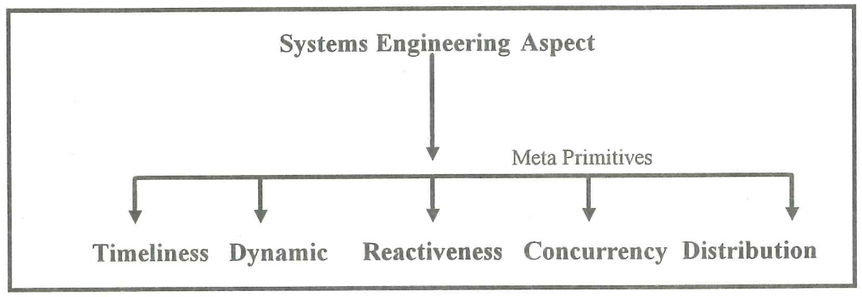

Figure 3. Systems Engineering Meta Model

Meta Primitives of the Software Engineering Aspect

This section is loosely based on Selic [Selic1994], who has developed a real-time OO modelling technique called ROOM.

This is the one of most important aspects within the process control domain. The success of a software system is highly dependent on the selection of the appropriate software engineering method and supporting techniques. The meta primitives of the software engineering aspect are depicted in Figure 4.

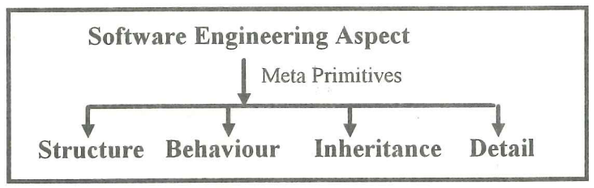

Figure 4. Software Engineering Meta Model

High level structure modelling : At this level the concept of reusability is fundamental to the OO paradigm. ROOM has adopted this reusability strategy at the high level structure dimension. A special concept defined as an actor is its basic modelling concept. In order to achieve reusability, each actor has to have one specific purpose with well defined interfaces to other components. An actor can have more than one interface as it can communicate with different types of actors. With this structured architecture, actors can co-exist with other actors concurrently in its domain. The process control domain is composed of concurrent objects and it is this functionality which makes ROOM a suitable choice. An actor makes use of encapsulation to maintain its primary purpose. The reasons for using encapsulation is that it is necessary to ensure that the coupling between actors is restricted to only the interactions across the interfaces.

High level behaviour modelling : This domain is characterised by the presence of two complex and difficult phenomena, concurrency and distribution. An event which is basically a message that is related to time, i.e. it has temporal properties, is generated from outside the ROOM environment. There are two general approaches to the handing of events: run-to-completion and pre-emptive. ROOM has opted for the run-to-completion approach. The two approaches are depicted in Figure 5.

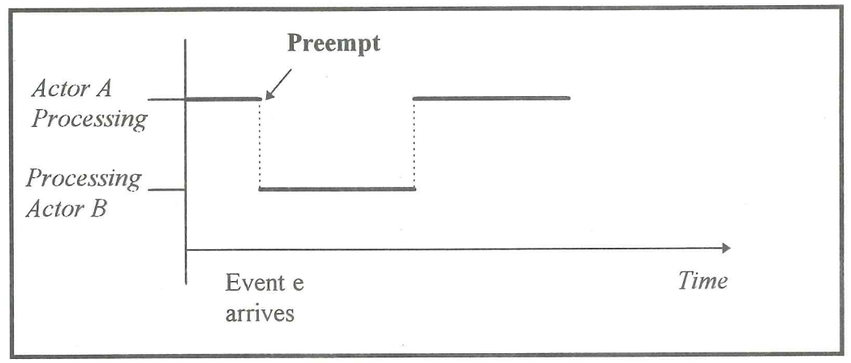

Figure 5. Pre-emption and Run-to-Completion

In order to achieve a short execution time of e1, protracted event processing sequences can be broken up into a number of shorter chunks. However, this can significantly complicate the implementation.

In terms of priorities, the ideal would be to have one single priority. However with the distributed process control environment this is not possible. It is thus recommended to have some level of urgency to be provided. This would cater not only for the high priority tasks but also the low priority tasks. Frequently, the low priority tasks are disregarded during busy periods.

High level inheritance : ROOM views inheritance primarily as an abstraction mechanism that helps to deal with complexity by allowing detail to be introduced gradually. With the abstraction based approach, abstract classes play a pivotal role. The definition and preservation of the integrity of abstract classes is the primary concern in dealing with inheritance. Each abstract class should symbolise a well-defined abstraction with a clear meaning.

In ROOM, inheritance plays a fundamental role, since all designs are specified as classes. Three different class hierarchies are supported: the actor hierarchy, the protocol hierarchy and the data object hierarchy. The inheritance rules of data objects are determined by the Detail Level language used in modelling. Actor classes allow both high level structure and high level behaviour to be subclassed, providing for a much higher form of reuse than is available in traditional OO programming languages. Only single inheritance is supported for actor classes and exclusion and overriding of attributes is allowed [Selic1994].

The detail level : This section examines the issues and techniques used to specify the Detail Level aspects of a ROOM model. The detail level deals with fine grained actions and fine-grained objects. All elements of concurrency and high level architectural issues are filtered out and dealt with at higher abstraction levels. As a result the complexity of the specifications at this level is greatly reduced. Detail level actions capture the behaviour that occurs during transitions of a state machine from one state to another. Fine-grained objects are used either to capture the extended state of a state machine or as information units that can be transferred between actors [Selic1994].

Meta Primitives of the Information Aspect

Process control systems utilise real time databases. A real time database system provides database features such as data independence and concurrency control while at the same time enforcing real-time constraints that applications may have. The meta primitives of the information aspect are depicted in Figure 5.

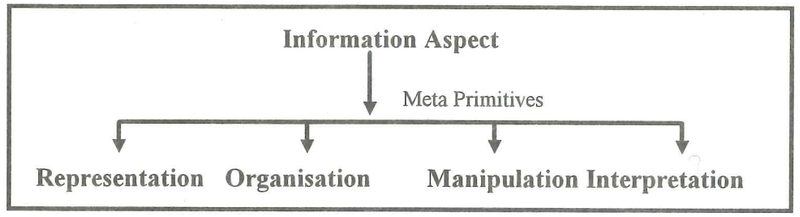

Figure 5. Information Meta Model

Representation of information : There are 3 key issues that affect the representation of information, data independence, binding and efficiency

There is generally a single globally accessible shared data area, whose organisation is defined by the data definition facilities of the chosen programming language; every task in the system that shares data is compiled with the definition of the entire shared area and thus has access to all the data at any time. This scheme has proven to be problematic. A key objective of data management technology is the achievement of data independence; that is each task should be unaware of the storage structure and access mechanism of the data accessed. If data independence is achieved, stored data can be re-organised with no impact on the code of each task. Data management technology achieves data independence by interposing one or more layers of system software between the application task and access to the data itself. In this way, a call is made to a system software task which then uses a data description to access the data item. These "additional" layers of code may introduce inefficiencies into the implementation, which causes problems in a process control system.

The efficiency and independence objectives are clearly in conflict. The two objectives can be studied together by introducing the concept of binding. In the case of a single globally shared data area whose definition is compiled into each process, the names are bound to locations at compile time or at link/load time. However, the use of a procedure to access a data item will delay the binding until run time. It is the execution of the binding at run time that causes the use of data management technology to be less efficient yet more independent [Ward1985].

Data organisation : There are two faces to data organisation implementation; the view that is presented to the designer and programmer and the actual physical organisation of the data on the chosen storage media. The designer and the programmer will want to perform operations on the data structures according to an external view of how the data is grouped and organised; this can be separated from the internal data structure chosen for the data. It is the business of any data

management software to provide mechanisms to manipulate the external data structure which can then be translated into operations on the internal data structure.

However support for real-time data base systems must take into account the following: firstly, not all data in a real-time database are permanent; some are temporal and secondly, since timeliness is sometimes more important than correctness, in some situations, precision can be traded for timeliness.

Manipulation of information : Process control transactions are characterised along three dimensions: the manner in which data is used by the transactions, the nature of the time constraints and the significance of executing a transaction by its deadline or more precisely, the consequence of missing specified time constraints.

Further temporal consistency requirements of the data can lead to some of the time constraints for the transaction.

Real-time database systems employ all three types of database transactions, i.e., write-only transactions, update transactions and read-only transactions. This classification can be used to tailor the appropriate concurrency control schemes.

Some transaction time constraints come from temporal consistency requirements and some arise from requirements imposed on system reaction time. The former can typically take the form of periodicity requirements: For example, every 5 seconds, sample the reservoir level.

System reaction requirements typically take the form of deadline constraints imposed on aperiodic transactions. For example, if Water_Flow > 100
add Chlorine within 10 seconds.
In this case, the system's action in response to the high Water_Flow must be completed by 10 seconds.

Transactions can also be distinguished based on the effect of missing a transaction's deadline. As discussed in the section on the Temporal paradigms the terms of hard, soft and firm transactions have been discussed. This categorisation shows the value imparted to the system when a transaction meets its deadline. The processing of real-time transactions must take their different characteristics into account.

Interpretation of information : In the process control domain, a real time database consists of a set of data objects representing the state of an external world controlled by a real-time system. The data objects are interpreted as two types: continuous and discrete. Continuous data objects are related to external objects continuously changing in time. The value of a continuous data object can be obtained directly from a sensor (image object) or computed from the values of a set of image data objects (derived object) within a regular period. Continuous data objects are related with the following additional attributes: a timestamp indicating when the current value of the data object was obtained, an absolute validity duration is the length of time during which the current value of the data object is considered valid and a relative validity duration is associated with a set of objects used to derive a new data object. Discrete data objects are static in the sense that their values do not become obsolete as time passes, but they are valid until update transactions change the values.

Summary

The environment, information, software engineering and system engineering aspects cannot exist in isolation. There is a close relationship between them as their boundaries are not well defined and there is clearly an overlap between each aspect's discipline.

The Reference Model proposed is meant to serve as a guideline in the development of process control applications. Due to the applications' inherent complexity, it is necessary to have a frame of

reference that encompasses all the relevant issues in its development. Further, by using the object oriented paradigm, there is an increase in the reuse of software in the automation software area, especially if the same applications with probably even the same configuration are to be used at various plants.

References

Beam K: Object Technology Object-Oriented Programming. I/S Analyser, December 1993, Vol. 31, Iss. 12, pp. 1 - 15.

Brown AW, Earl AN & McDermid JA: Software Engineering Environments. McGraw Hill, 1992.

Selic B, Gullekson G & Ward PT: Real-Time Object Oriented Modelling. John Wiley & Sons, 1994.

Steenkamp L: Object Oriented Information Systems Engineering Environment. May 1995, pp. 1-20.

Ward PT & Mellor SJ: Structured Development for Real-Time Systems. Volume 1: Introduction and Tools, Yourdon Press, 1985.

THE PEARL ALGORITHM AS A METHOD TO EXTRACT INFORMATION OUT OF A DATABASE.

J. W. Kruger
KRGER-JW@SOREX.VISTA.AC.ZA
Vista University - Soweto Campus
P. O. Box 359, Westhoven 2142
Tel: (011) 938-1701 x 238(W), (011) 477-2078 (H)

Abstract

Big databases have been built up over time. The information in these databases can be converted by the Pearl algorithm [Pearl 1988] from data to information. When a promotional drive is initiated, do we know who the potential customers are? The Pearl algorithm gives the belief that an individual will purchase. This means that the database can be used to select the individuals that are most likely to purchase. The breakeven point, where the cost of the promotional contact and the expected return are the same, can be calculated. The Pearl algorithm uses Bayes' probabilities to propagate belief through a tree.

The Correlation between variables

The Pearson correlation between the numeric different fields on a flatfile (or relational) database can be calculated. Non-numeric fields, like yes/no answers can be converted to numeric by allocating a one to yes and a zero to no. The Pearl algorithm was developed for binary fields, and the conversion, mentioned here, give good results. In my personal experience market researchers without a statistical background tend to have too many non-numeric fields in their questionaires. These fields are sometimes difficult to analyse and the data is then merely stored for possible future reference.

Star decomposability

If we have three variables (Nodes A, B and C), then they are said to be star decomposable if a latent structure [Lazarfeld 1966] exists that is the common cause to the three variables. The latent structure can be found as a hidden node. If the hidden variable is called W, then the correlation to the hidden variable can be determined (See figure 1).

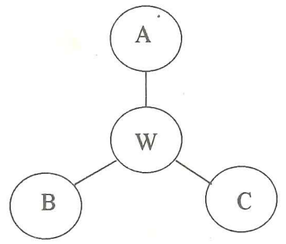

Figure 1. A star formation:

The latent structure, indicated by node W, links the leaf nodes A, B and C.

The correlation to W can be calculated from:

$r_{AB} = r_{AW} \cdot r_{WB}$

$r_{AC} = r_{AW} \cdot r_{WC}$

$r_{BC} = r_{BW} \cdot r_{WC}$

By solving:

$$r_{AW} = \sqrt{\frac{r_{AB} \cdot r_{AC}}{r_{BC}}}$$

The correlation to the hidden node, W, can be calculated.

Causal Structure

To add a node, D, to this structure, one of the following correlation equations must hold:

1. $r_{AD} \cdot r_{BC} = r_{AB} \cdot r_{CD}$
2. $r_{BD} \cdot r_{AC} = r_{AB} \cdot r_{CD}$
3. $r_{AD} \cdot r_{BC} = r_{AC} \cdot r_{BD}$

In equation 1, the node D must link to the arc between B and W.
In equation 2, the node D must link to the arc between A and W.
In equation 3, the node D must link to the arc between C and W.

By continuing with this reasoning a causal structure that is an acyclical graph (Tree / belief network) can be built up.

Joint-Occurence probabilities

For the time being, assume that we are working with binary variables and that W is the central node between the leaf nodes: X_1, X_2 and X_3:

Define the seven joint-occurence probabilities as [Lazarfeld 1966]:
$p_i = p(x_i = 1)$
$p_{ij} = p(x_i = 1, x_j = 1)$
$p_{ijk} = p(x_i = 1, x_j = 1, x_k = 1)$

The standard deviation of a Bernoulli variable is given by: $\sigma_i = [p_i (1 - p_i)]^{½}$

and the correlation coefficients: $\rho_{ij} = (p_{ij} - p_i p_j) / \sigma_i \sigma_j$

Link Matrix

Define $f_i = p(x_i = 1 \mid w = 1)$ and $g_i = p(x_i = 1 \mid w = 0)$; Now we can solve the elements of the link matrices [Bhat 1984] or transition probability matrix.

The prior probabilities of the node W, multiplied by the transition matrix, gives the prior probabilities for the node X_i.

Let $S_i = \pm [(p_{ij} - p_i p_j) (p_{ik} - p_i p_k) / (p_{jk} - p_j p_k)]^{½}$

$\mu_i = (p_i p_{ijk} - p_{ij} p_{ik}) / (p_{jk} - p_j p_k)$

$K = S_i / p_i - p_i / S_i + \mu_i / (S_i p_i)$

and $\alpha = t^2 / (1 + t^2)$, where t is the solution to $t^2 + Kt - 1 = 0$

then $f_i = p_i + S_i [(1 - \alpha) / \alpha]$
and $g_i = p_i - S_i [(1 - \alpha) / \alpha]$

Anybody interested in the manipulations can look at the theorem by Lazarfeld [Lazarfeld 1966, Pearl 1988].

Non-binary variables

The Pearl algorithm was developed for binary variables. For variables in more states or for continuous variables the causal structure can still be found.

To generalise, a method to calculate the transition matrices must be found. In general we must solve $3n(n - 1)$ unknown parameters (elements) to find the transition matrices, to connect three leaf nodes to the belief network (The tree). A generalisation to a continuous state space scenario is needed.

A heuristic to solve the elements of the link matrices with more than two states has been developed . [Kruger 1996a]

Belief Propagation

In a single chain structure (see Figure 2):

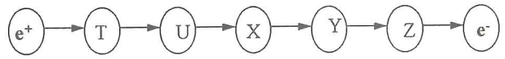

Figure 2. A causal chain with evidential data at its head (e^+) and tail (e^-)

To propagate the prior probability down away from the root, the prior probability vector must be pre-multiplied to the transition matrix.

To propagate the likelihood towards the root, the likelihood must be post-multiplied to the transition matrix.

For the tree however:

Consider a typical node x, with m children $y_1, y_2, \ldots, y_m$, and parent u:

Belief updating in a tree:

As with the chain structure above; Using the causal support, $\pi(x)$, and diagnostic support, $\lambda(x)$, the belief distribution of x is:

$BEL(x) = \alpha P(e^+ \mid x) P(x \mid e^-) = \alpha \lambda(x) \pi(x)$

α is a normalizing constant to make $\Sigma BEL(x) = 1$.

Note: The multiplication of these vectors must not be confused with the scalar product of vectors. By multiplication we mean that the corresponding co-ordinates are multiplied, giving a vector of products, each co-ordinate corresponding to the belief that the state is true.

The difference comes in the way that $\lambda(x)$ is calculated.

$\lambda(x) = P(e^- \mid x) = P(\text{evidence from the children} \mid x) = \prod_{i=1}^{m} \lambda(y_i \text{ of } x)$. because x separates its children and the siblings are conditionally independent.

$$\pi(x) = \sum_{u} P(x \mid u)\ \pi_x(u) = \pi(u)\ P(x \mid u)$$

Bottom-up propagation:

Node x uses the $\lambda(x)$ message from the children to compute a new message $\lambda(x \text{ of } u)$ to send to its parent u.

$$\lambda(x \text{ of } u) = \lambda(x)\ P(x \mid u)$$

Top-down propagation:

The new π message sent to by node x to its j-th child y_j is:

$$\pi(y_j) = \alpha\ \pi(x) \prod_{k \neq j} \lambda(y_j \text{ of } x)$$

but,

$BEL(x) = \alpha\ \lambda(x)\ \pi(x)$, therefore $\pi(y_j) = \alpha\ BEL(x) / \lambda(y_j)$.

Readers interested in Belief propagation in more general networks are referred to Pearl [1996]. As the Pearl algorithm produces a causal tree, belief propagation in more general networks are not included here.

From the above it is evident that we only need the prior probabilities of the root and the likelihood functions of the leaf nodes to calculate the belief of all the nodes.

The prior probabilities for the leaf nodes can be used as likelihoods. The prior probability for the root must still be found. The easiest is to make the dependent variable the root. This means that certain transition matrices will have to be inverted. Of the dependent variable is the root, then the prior probabilities can be found empirically.

Application

In a database, the variable, of whether the client will purchase or not, can be found from previous experience (data). Last time a promotion went out, who purchased. It can also be found from a pilot study. The causal structure with other fields in the database can be found by the Pearl algorithm.

Read the clients in the database sequentially. Instantiate the known fields for each client on the database.

Now these likelihoods and the prior probability for the root can be propagated through the belief network, to get the belief of all the undefined variables in the belief network.

Kruger [Kruger 1996a] gives a method to evaluate the different possible belief networks created. Kruger also compares the accuracy of belief networks on data, used to create the networks, with applying the algorithm to other data.

The belief of a state like purchase can be multiplied by the expected monetary gain of a purchase, to get the expected gain of contacting this client. If the gain is more than the cost, then it will be wise to go ahead. If the gain is less than the cost, then of course no contact must be made.

With this method qualitatively seemingly unrelated fields can assist in the highly competative field of retail sales. Fields giving biographical data like gender, home language, marital status can now be used to predict sales.

Personell Psychologists use biographical information to decide on the the selection of applicants. They create a weighted biographical information index, test the validity of this index and then base their decisions on this. The Pearl algorithm is a much more refined method to obtain a valid selection criterium out of a database [McCormich 1992, Kruger 1996b].

Acknowledgement

I want to thank Prof. David Lubinsky from the University of the Witwatersrand for the help he gave me in understanding the Pearl algorithm and Bayes' methods.

References

1. Bhat, U. N., "Elements of Applied Stochastic Processes", John Wiley & Sons, Inc., 1984
2. (a) Kruger, J. W., "Generalizing the number of states in Bayesian Belief Propagation, as applied to Portfolio Management", Masters Research Report, University of the Witwatersrand, 1996.
3. (b) Kruger, J. W., "Reliability and Validity of selection criteria into courses and of tests and exams.", Proceedings of the South African Computer Lecturers Association Conference at Pilansberg, 1996.
4. Lazarfeld, P.F., "Latent structure analysis, in measurement and prediction", eds. S. A. Stoufer, L. Guttman, E.A. Suchman, P.F.Lazarfeld, S.A.Star and J.A.Classen, Wiley, New York, 1966
5. McCormick E. J., and Ilgen D, "Industrial and Organisational Psychology" 8th edition, Scotprint Ltd, Musselburgh, ISBN 0-415-09452-6, 1992
6. Pearl, J., "Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference" Morgan Kaufmann, San Mateo, California 1988

THEORY MEETS PRACTICE: USING SMITH'S NORMALIZATION IN COMPLEX SYSTEMS

Alta van der Merwe & Prof. Willem Labuschagne
Department of Computer Science and Information Systems
University of South Africa
Pretoria

Abstract

In the middle 80s Smith introduced a new way of deriving fully normalized tables. His method was based on the use of an dependency list and a dependency diagram to produce fifth normal form directly. We introduce the use of end-line, begin-line and in-line bubbles to simplify the diagram for large-scale applications.

Introduction

It is usual to follow a procedural methodology when designing a schema for relational database management systems (RDBMS). During the design phase an entity relationship (ER) model is used to represent the entities and relationships. The designer then transposes the entities' associated data fields into a set of candidate tables and, finally, all candidate tables are transformed step-by-step to fifth normal form by applying normalization guidelines (Date, 1986). The apparent simplicity of this ER-model approach is deceptive inasmuch as the model uses the semantics of entities and not the fields. The designer of the system must have a complete understanding of the normal-form guidelines and any real-world information applicable to the system when transposing the ER-model to candidate tables (Codd, 1970).

In 1985 Henry C Smith introduced a method to compose fully normalized tables from a rigorous dependency diagram. This method is based on field-related semantics. Because the semantic meanings of its dependency chains are specifically tailored to normal-form guide-lines, the model directly and immediately synthesizes into fully normalized tables with completely specified primary keys (PK), foreign keys (FK), and join paths (Smith, 1990).

Because the dependency diagram plays the key role in Smith's method, it is essential that the diagram should be readable. Readability tends, of course, to diminish in proportion to the size and complexity of the application. In this paper we adapt Smith's method for application to complex large-scale systems.

Basic conventions

When designing a dependency model using Smith's method (SM), three documents are used: a dependency model, a schema synthesized from the model and an enterprise data dictionary. We focus on the dependency model and the schema synthesized from the model.

The dependency model is developed in accordance with the following rules:

- Every field name used in the dependency model must be atomic or non-decomposable and may appear only once in the dependency model.
- A single valued dependency (SVD) is diagrammed by drawing a single bubble around the field and using a single-headed arrow to the field dependent on it. See figure 1 for a single dependency A→ B (B is determined by A). The originator (A) is called the determinant and each attribute dependent on it is called a singleton.
- Multi-valued dependencies (MVD) arise when a field A determines many values of field B. An MVD is diagrammed by drawing bubbles around field A and B and using a double-headed arrow from bubble A to B (Figure 2). A is called the determinant and B is called the end-multiple bubble. B is called an end-multiple bubble if it contains a field whose multiple values are determined by A; a double-headed arrow points to B and no arrows point from it. Values of A and B must be not null and the combination A,B must be unique.

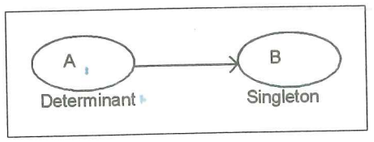

Figure 1 : Single-valued dependence

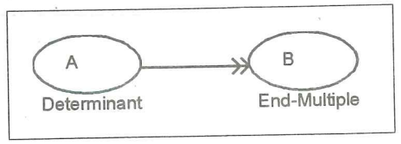

Figure 2 : Multi-valued dependence

- If MVDs exist between A and B and A and C (with A the determinant) they are diagrammed as in Figure 3.
- If MVDs exist from A to B and from B to C they form an MVD chain and are modelled as in Figure 4. Values of A, B and C must be not null and the combination A, B, C must be unique. A is called the uplink-determinant bubble, B the determinant bubble and C the end-multiple bubble.

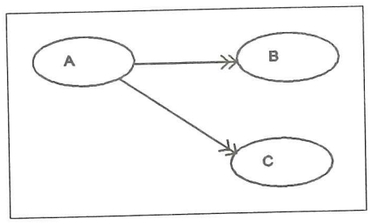

Figure 3 : Independent MVDs

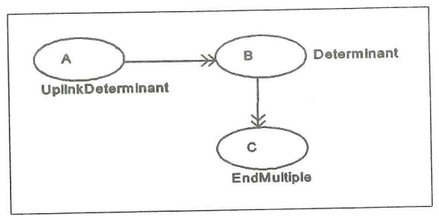

Figure 4 : An interdependent MVD

- SVDs can be uplinked to an MVD or MVD chain.
- No transitive dependencies are allowed.
- A double bubble is used when a field is a determinant of two different or conflicting dependencies. E.g. if A determines B but A also determines C where C and B do not have anything to do with each other, a double bubble are used to diagram these facts (figure 5).
- [n] at the bottom of fields is used to show that these fields share the same domain values (n=1..).
- <m> at the bottom of a singleton field means that more singleton fields can be found at number m in the dependency list.
- A singleton chain is a series of interlinked bubbles that ends with one or more singleton bubbles, i.e. a bubble with one or more single-headed arrows pointing to it and no arrows pointing from it.
- A end-multiple chain is a series of interlinked bubbles that end with an end-multiple bubble, i.e. a bubble that has one or more double-headed arrows pointing to it and no arrows pointing from it.

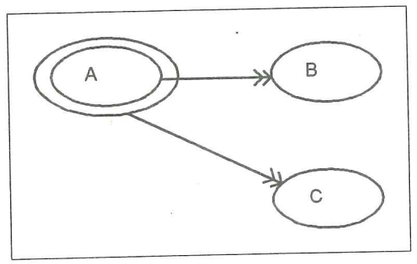

Figure 5 : Conflicting dependencies

Smith's algorithm to create 5NF relations

1. Consider all the singleton chains. All the fields within all singleton bubbles linked to the same determinant bubble are the non-key attributes of the table. The field within that determinant bubble plus field(s) within all uplink-determinant bubbles form the table's primary key. Mark the line between the last determinant and all the singletons linked to it, as worked (√).
2. For an end-multiple chain, work upwards from the end-multiple bubble : the fields within the end-multiple bubble, the determinant bubble and all uplink-determinant bubbles form the table's primary key. Mark the line between the last determinant and the end-multiple bubble linked to it, as worked (√).

Example

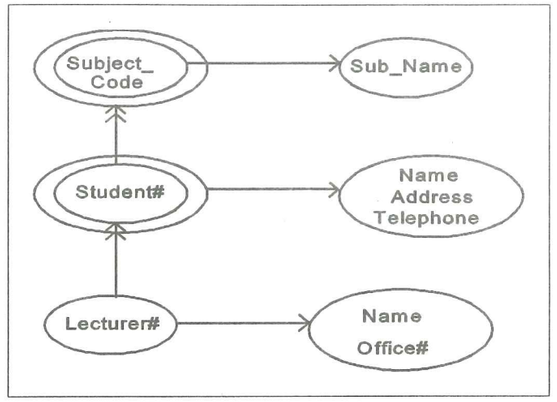

Figure 6 : A simple dependency diagram depicting a student system

Dependency List

1. For each student a Student Number (Student#), Name, Address and Telephone must be stored.
2. For each Lecturer a number, Name and Office Number (Office#) must be stored.
3. Each Subject is identified by an Subject_Code and described by a name (Sub_Name).
4. Each Lecturer can have more than one student and each student can take more than one subject.

5NF relations

Student(Student#, Name, Address, Telephone)
Lecturer(Lecturer#, Name, Office#)
Subject(Subject_Code, Sub_Name)
Class(Student#,Lecturer#, Subject_Code)

Large-scale applications

When one applies Smith's method on big complicated systems, the readability of the dependency diagram can be adversely affected in two principal ways, namely by lines that cross one another and by a proliferation of double-bubbles. We will use the MetAIS (Metrological Air Information System) system developed by Netsys International to illustrate the problems.

The MetAIS system was developed for a customer that provides operational flight information to the aviation community. The MetAIS system receives the data from various sources including satellite feeds and renders it available with a good interface in useful format. A typical use might be for a pilot to ask what the weather will be from point A to point B. The system must then gather all the information applicable to the route and draw the pilot's attention to potential problems in regard with meteorological phenomena as well as operational problems.

Representing multiple dependencies

The dependency list for the MetAIS system consisted of about 70 different dependencies of importance. To represent all the different dependencies explicitly in one diagram, and still keep it readable, was impossible. We used the rule: *<m> at the bottom of a singleton field means that more singleton fields can be found*

at number m in the dependency list to simplify the diagram. Although this helped, the readability of the diagram remained affected by the crossings of lines from one side of the diagram to another (figure 7).

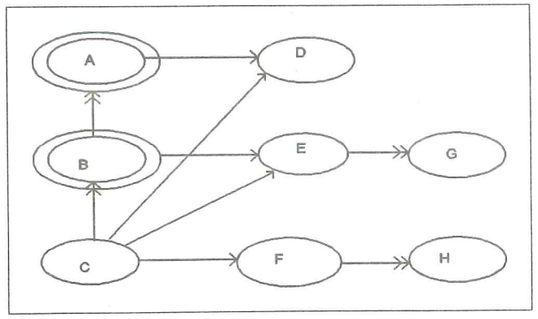

Figure 7 : Lines crossing to represent different relationships

An end-line bubble (ELB) and begin-line bubble (BLB) were created to eliminate the crossing of lines. Let's start with an example to illustrate the use of ELBs and BLBs. Consider dependency list number 7:

7. For Case A every value of field C will uniquely determine field D.

Our problem is that field C is positioned at the bottom lefthand corner and field D at the top righthand corner of the diagram. Linking the two requires crossing various other lines. By making use of an end-line bubble (figure 8) from C to link it to its begin-line bubble counterpart (figure 9), we simplify the diagram a great deal.

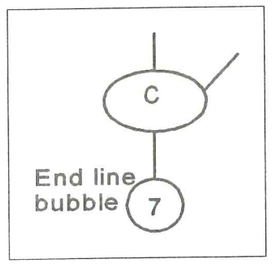

Figure 8 : End-line bubble

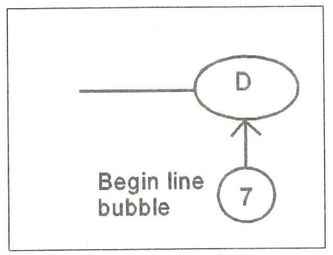

Figure 9 : Begin-line bubble

The following conventions govern the use of ELBs and BLBs:

- each ELB and BLB has a number written in it. This number is the same number as the dependency list number associated with it. The ELB and BLBs that are linked must have the same number.
- each ELB must have at least one BLB.

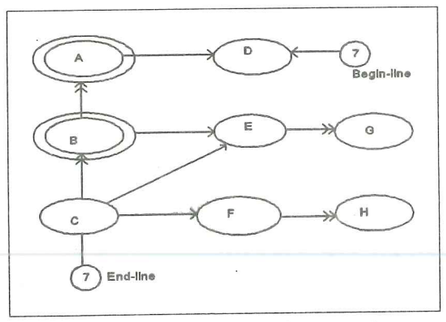

Figure 10 : ELB and BLB dependencies

Figure 10 illustrates the use of an ELB and a BLB for the example given in figure 7. It is clear that using ELBs and BLBs to eliminate the crossing of lines renders the diagram more readable.

Double-bubble representation

Double-bubbles are used to represent different or conflicting dependencies (Figure 5). The following is an extract from the MetAIS system.

29. Each NSWC has a TTAAii with more than one YYGGgg. For each YYGGgg more than one CCCC and BBB exist and the combination of the four has one ValidTo and one set of Data.
30. Each SWC has a TTAAii with more than one YYGGgg. For each YYGGgg more than one CCCC and BBB exist and the combination of the four has one ValidTo and one set of Data.
36. A low level forecast has a combination of TTAAii, CCCC, BBB, YYGGgg that is unique and identifies a single ValidTo and Data field.
44. For each WC a TTAAii with more than one YYGGgg is stored. For each YYGGgg more than one CCC and BBB exist and the combination of the four has one Flight_Level, ValidTo and one set of Data.

The diagram for the four dependencies in the dependency list is given by Figure 11.

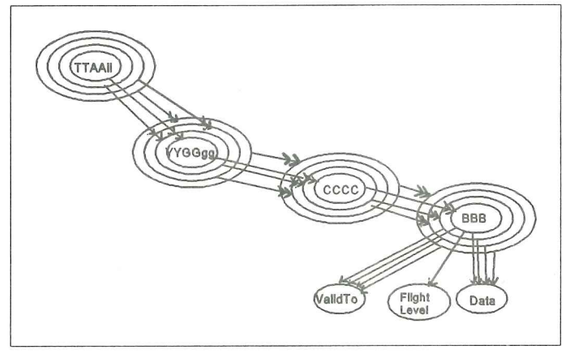

Figure 11 : Double-bubbles used in the MetAIS system

The fields TTAAii, YYGGgg, CCCC, BBB, ValidTo and Data are repeated for four different conflicting dependencies. We call this a repetitive dependency of degree 4. In general the double-bubble representation of repetitive dependencies of degree greater than 2 impedes the comprehension of the diagram. We propose the insertion of in-link bubbles as an alternative way to represent repetitive dependencies.

Consider the example in figure 12 with the following dependency list:

1. For Case 1 S# can have more than one D# and for D#,S# there exists only one O#.
2. For Case 2 S# can have more than one D# and for D#,S# there exists only one O#.
3. For Case 3 S# can have more than one D# and for D#,S# there exists only one O#.
4. For Case 4 S# can have more than one D#.

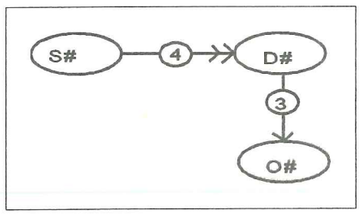

Figure 12 : Conflicting dependencies

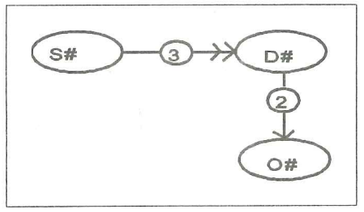

Figure 13 : Diagram after 1 iteration

An in-link bubble (ILB) is created between two fields to show the degree of the link. The value 3 in the in-link bubble between D# and O# means that, irrespective of any links to D# or O# and irrespective of any links that exit from D# and O#, this specific link must be used in exactly three iterations of the normalizing algorithm. Each iteration reduces the values inside the in-link bubbles by one. After the first iteration on the example in figure 12, we are left with the diagram in figure 13 and the relation CASE1 (S#, D#, O#) is derived. After the second and third iteration the following relations are derived: CASE2 (S#, D#, O#) and CASE3 (S#, D#, O#)

We are left with the diagram in figure 14.

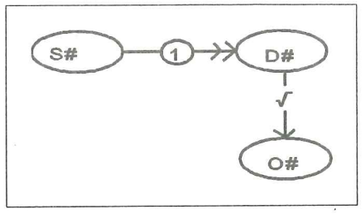

Figure 14 : Example after third iteration

Note that the link between D# and O# is marked off as worked (√). A link is marked 'worked' when it was used in an iteration and no longer needs to be taken into account in the normalization process. Only the repetitive dependency between S# and D# (of degree 1) is of any further importance and the relation CASE4 (S#,O#) is derived according to the rules.

To make provision for in-link bubbles a small modification must be made to the way of deriving the normalised tables.

1. Consider all the singleton chains. All the fields within all singleton bubbles linked to the same determinant bubble are the non-key attributes of the table. The field within that determinant bubble plus fields within all uplink-determinant bubbles form the table's primary key. If an in-link bubble exists between two fields, reduce the degree by one in the in-link bubble. In case of a degree of 0, mark the link as worked (√).
2. For an end-multiple chain, work upwards from the end-multiple bubble : the fields within the end-multiple bubble, the determinant bubble and all fields in the uplink-determinant bubbles form the primary key. If an in-link bubble exists between any two fields, reduce the number by one in the in-link bubble. In case of a degree of 0, mark the link as worked (√).

It should be borne in mind that any link to a singleton or end multiple without an in-link bubble is taken to be of degree 1. (Such a link must be used in one and only one iteration between the two fields that are linked).

The simplified diagram for the MetAIS system is given in figure 15.

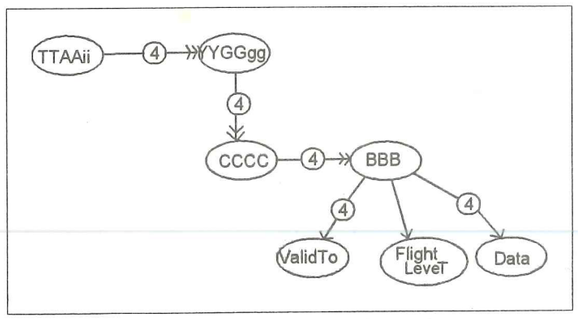

Figure 12 : MetAIS system simplified using ILB.

The relations derived for the system are the following:

NSWC (TTAAii, YYGGgg, CCCC, BBB, ValidTo, Data)
SWC (TTAAii, YYGGgg, CCCC, BBB, ValidTo, Data)
LLF (TTAAii, YYGGgg, CCCC, BBB, ValidTo, Data)
WC (TTAAii, YYGGgg, CCCC, BBB, ValidTo, Data, Flight_Level)

Note that it is not strictly necessary to show that the degrees of the links between TTAAii, YYGGgg and CCCC are 4. TTAAii, YYGGgg and CCCC are uplink-determinants of BBB and according to Smith's algorithm will be repeated automatically for each determinant - end multiple (or singleton) repetition.

Conclusion

Many systems are designed over a long period, and new programmers may be added to the team at a late phase of the development. One of the most difficult things for these programmers is to understand the data processes and how the data interlink. The use of diagrams for the system designer may sometimes entail tedious effort, but as a documentation tool for the user and programmers added to teams, the advantages are worth the effort, provided only that the diagrams remain readable. By employing end-line and begin-line bubbles the crossing of lines can be eliminated, and by inserting in-line bubbles the representation of repetitive dependencies can be simplified, thus improving the readability of diagrams involved in large-scale applications.

References

Codd, E.F. "A Relational Model of Data for Large Shared Databanks," Communications of the ACM 13(6):June 1970.

Date, C.J. An Introduction to Database Systems, Volume I, 4th Ed. Reading Mass.: Addison-Wesley, 1986.

Smith, H.C. "Database Design: Composing Fully Normalised Tables From a rigorous Dependency Diagram," Communications of the ACM 28(8): 826-838, August 1985.

Smith, H.C. "Dependency Modelling for Relational Database Design", Database Programming & Design, 3(9) : 39-49, September 1990.

A COMPARISON OF TRANSACTION MANAGEMENT SCHEMES IN MULTIDATABASE SYSTEMS

Karen Renaud and Paula Kotzé
Department of Computer Science and Information Systems,
University of South Africa, P.O. Box 392, Pretoria, 0001.
Email: {renaukv,kotzep}@risc1.unisa.ac.za

Abstract

In many applications today multiple pre-existing database systems are integrated into a single multiple database system called a multidatabase system. There are various characteristics of these systems which distinguish multidatabase systems from one another — autonomy, heterogeneity and distribution. One of the biggest problems with respect to transaction management in a multidatabase system is the question of how to maintain global concurrency control. Researchers have proposed various transaction management models for multidatabase systems. In this paper a core group has been formed by choosing the work of seven different research groups. Their respective approaches to transaction management will be appriased and the concurrency control methods they employ will be briefly discussed as well.

Introduction

A *multidatabase system* (MDBS) (also called a *multidatabase* (MDB)) is a system which is composed of autonomous or semi-autonomous, pre-existing *database systems* (DBSs), together with a *multidatabase management system* (MDMS) software layer, built on top of them, to control access to all data in all the component databases from the point of view of the user of the multidatabase system, while the component database systems still function independently.

Transactions submitted to the MDBS are called global transactions, while transactions submitted to the component DBSs are called local transactions. The MDMS facilitates access and manipulation of data at local sources (the component databases), possibly distributed among nodes of a computer network, by both users at the *local database systems* (LDBSs) as well as users of the MDBS. Transaction management in an MDB environment has many functions. It must undertake to ensure global consistency by ensuring that local and global transactions can execute concurrently without violating MDB consistency, and maintain freedom from deadlocks. This must be done in the presence of local transactions and in the face of the inability of local *database management systems* (DBMSs) to coordinate execution of MDB transactions, under the assumption that no design changes are allowed in local DBMSs [Bright 1992]. This paper will examine the maintenance of MDB consistency during concurrent transaction execution.

One cannot, and indeed may not, assume that the LDBSs are aware of the existence of other participating LDBSs, and thus one cannot assume any communication between them. As a result, while local DBMSs can be depended upon to preserve consistency of the local database, it cannot be expected to ensure that the concurrent execution of local and global transactions preserve MDBS consistency.

What is meant by MDBS consistency? In most centralized DBMSs the *serializability* correctness criterion is used to ensure that the consistency of a database is maintained while concurrent transactions carry out operations on the database. Achieving serializability in an MDBS is not a trivial task. Some researchers have felt that global serializability is too stringent an expectation for MDBSs. Various researchers have proposed alternative approaches to MDB consistency [Barker 1990, Du 1989, Elmagarmid 1990, Rastogi 1993].

If global transactions were to be executed serially in the MDBS, one would not have a problem. The challenge in an MDBS is to achieve the maximum degree of concurrency while maintaining MDBS consistency. To do this, a software module, referred to as the *global transaction manager* (GTM), which is part of the MDBS, coordinates the execution of global transactions in an MDBS environment [Breitbart 1995]. A correctness criterion is chosen for the MDBS and the GTM will employ mechanisms to ensure that concurrent execution of transactions will not violate these criteria.

In order to give an idea of how various researchers have addressed the problem of MDBS transaction management, a core group of seven different transaction management schemes will be introduced. A brief synopsis of the intricacies of their mechanisms will be given, as well as details about the concurrency control mechanisms being employed. To set the stage, the characteristics of MDBSs will be discussed, identifying the autonomy characteristic as the one which can be used to differentiate schemes, and to evaluate the "goodness" of a proposed transaction management scheme. We will then take a look at each of the transaction management schemes which comprise our core group and then, after evaluation, put forward the optimal scheme which scores the best on the autonomy stakes.

Characteristics of Multidatabase Systems

There are three features which characterize MDBSs: *distribution, heterogeneity* and *autonomy* [Özsu 1990]. Distribution and heterogeneity are self-explanatory but autonomy, the most relevant characteristic, deserves a short discussion. Autonomy indicates the degree to which individual component databases in an MDBS can operate independently.

The preservation of local autonomy is both desirable and necessary in an MDBS since applications which have been developed prior to integration should continue to run afterwards too. Local DBMSs should also have the same measure of control over local databases after integration, while participating DBSs should be added to, or removed from, the MDBS with ease [Bradshaw 1993, Rastogi 1993]. Barker [Barker 1994] has set out a number of guidelines for the quantification of autonomy in MDBSs. He proposes a method for quantifying the autonomy violation of a particular transaction management scheme and arriving at a single value indicating the autonomy violation. A zero value would indicate no violation, while the ceiling value of 3 would indicate that autonomy had been violated to the maximum extent.

Using these measurements, the terms *fully autonomous, semi-autonomous* and *non-autonomous* can be defined.

1. An MDBS is said to be *fully autonomous* if the individual DBSs making up the system are stand-alone DBSs systems that are unaware of the existence of other component DBSs. The value for the autonomy violation would be 0.

2. An MDBS is said to be *semi-autonomous* if the component DBSs can operate independently but have decided to participate in an MDBS in order to make their local data shareable. They typically require certain changes to be made to their DBMSs or to the data in order to participate in the MDBS. The autonomy violation would probably be midway between 0 and 3.

3. An MDBS is said to be *non-autonomous* if a single image of the entire database is available to any user who wants to share the information which may reside in the MDBS. The individual DBSs will typically not operate independently. The overall autonomy violation would probably be close to 3.

Integrating Various Concurrency Control Methods

The various local DBMSs integrated by the MDMS may, and must be assumed to, use different concurrency protocols. The concurrency control mechanism in an MDBS has to be able to synchronize global transactions with purely local, autonomous transactions which are under the control of the local DBMS and ensure that the consistency of the database is maintained. Global transactions will typically address more than one of the component DBSs. The global transaction is therefore broken up into a set of subtransactions, each of which operates on a single DBS. To maintain consistency, either *all* these subtransactions must commit or *all* must abort.

The autonomy requirement means that once the global transaction submits a subtransaction to the local DBMS, it effectively relinquishes control over it. The local DBMS will assume all responsibility and will decide whether to commit or roll-back the transaction. Hence, some local DBMS could commit one subtransaction and another could abort another subtransaction of the same global transaction, thereby destroying the atomicity of the global transaction and compromising the consistency of the MDBS. One of the biggest problems arises when a global transaction and a local transaction work on the same data item. The local DBMS will handle the conflict by causing the transactions to execute serially, but this could cause the resulting execution of the global transactions to be non-serial — as is shown in the following example.

Example
In this example the following notation will be used: GT_n signifies global transaction n, $r_n(x)$ denotes a read from data item x by local transaction n, L_n denotes local transaction n, LDB^n indicates LDBS number n, $w_n(x)$ denotes a write to data item x by transaction n, and c_n denotes that transaction n has decided to commit. LS_n refers to the local schedule at DBS number n — showing the order in which transaction operations were executed at the local system.

Consider the following two non-conflicting global transactions:

$GT_1 : r_1(d); r_1(s);$ $\qquad$ $GT_2 : r_2(e); r_1(t);$

In addition, consider the following local transactions; L_3 executing at LDB^1 and L_4 executing at LDB^2.

$L_3 : w_3(d); w_3(e);$ $\qquad$ $L_4 : w_4(s); w_4(t);$

Now consider a history in which transaction GT_1 first executes at sites LDB^1 and LDB^2 followed by the execution of transaction GT_2 at both LDB^1 and LDB^2. It is possible for the local transactions L_3 and L_4 to execute in such a manner that GT_1 is serialized before GT_2 at LDB^1, while GT_2 is serialized before GT_1 at LDB^2. For example:

$LS^1 : r_1(d);\ c_1;\ w_3(d);\ w_3(e);\ c_3;\ r_2(e);\ c_2;$
$LS^2 : w_4(s);\ r_1(s);\ c_1;\ r_2(t);\ c_2;\ w_4(t);\ c_4;$

As far as the GTM is concerned, global transactions GT_1 and GT_2 are executed serially. At LDB^1, the resulting execution is serial: GT_1, L_3 and GT_2 . At LDB^2, the resulting execution is also serial: GT_2, L_4 and GT_1.

However, the global execution is non-serializable because, to be serializable, GT_1 should always precede GT_2, or vice versa.

In order to integrate heterogeneous local concurrency control algorithms to satisfy the chosen correctness criterion, two problems have to be considered [Yun 1993] — how can a global transaction be processed when it violates either serializability or some alternative correctness criterion, and how can an indirect conflict introduced by a local transaction be managed.

The traditional approaches to integrating heterogeneous concurrency control algorithms can be classified into two groups — *bottom-up* and *top-down* approaches [Breitbart 1995]. The *bottom-up*(optimistic) approach collects local information from each local site at the global level and thereafter checks for global serializability or for an alternative criterion. The *top-down*(pessimistic) approach maintains a global serialization order at local sites, which has been determined already at the global level.

Mullen *et al* [Mullen 1992] has proved that it is impossible to synchronize local and global transactions while still preserving local autonomy. In other words, some autonomy has to be sacrificed in order to maintain MDBS consistency in the presence of concurrently executing transactions. One either has to restrict the types of global transactions allowed to execute, or impose restrictions of some sort on the structure of the local concurrency control mechanisms.

In line with this, most existing MDBSs support data retrieval only — they only allow retrieval of data by global transactions and do not have any concurrency control schemes to

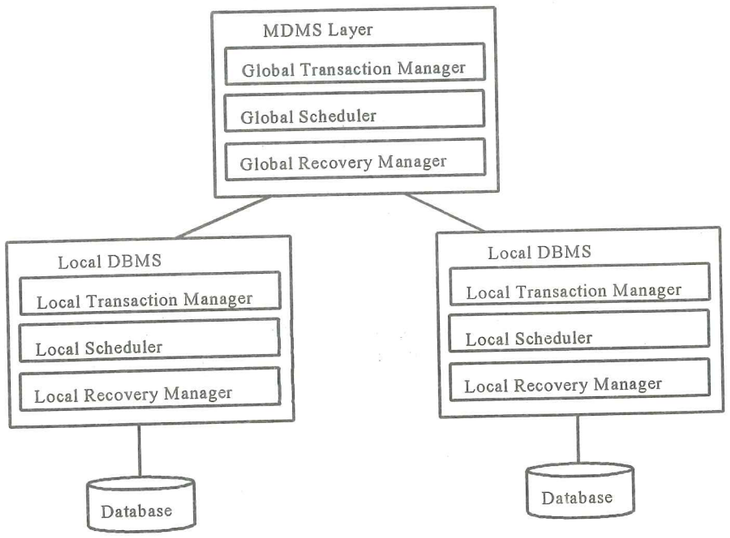

Figure 1: Components of an MDB in Barker and Özsu's model

[Barker 1990, p10]

co-ordinate the execution of global transactions. Even with this restriction, the problem of dirty, or unrepeatable reads, must be addressed.

Transaction Management Approaches

This section introduces the research done into transaction management in a selected core group of schemes. This is by no means a complete list but serves to give an indication of related work done in the area of transaction management in MDBSs.

Barker and Özsu's basic MDB model

Barker and Özsu [Barker 1990, Barker 1991, Özsu 1991] propose a very basic MDB model which is illustrated in Figure 1. This model serves to give a good basic understanding of MDBS architecture. The MDBS consists of various LDBSs, each having its own DBMS, each of which manages a possibly different type of LDBS. The MDMS provides a layer of software that runs on top of these LDBSs and allows users to access the various DBSs. The MDMS layer consists of a GTM, a global scheduler and a global recovery manager. Each DBMS has its own transaction processing components: the local transaction manager, the local data manager, and a local scheduler. The MDMS is simply seen as another user from which transactions are received and to whom results are presented.

Barker [Barker 1990] proposes a new correctness criterion called *m-serializability* which is an extension of serializability theory. MDB serialization graphs are developed to make it easy to determine when an MDB history is m-serializable. Barker and Özsu require strict schedules to be generated. The autonomy quantification for Barker and Özsu's model is shown in Figure 2.

Modification Dimension		
System	0.75	Reliability protocol requires strictness
Data	0	Data remains untouched
Design	X	Not discussed

$\bar{m} = \sqrt{0.75^2 + 0^2} = 0.75.$

Execution Dimension		
Local Transaction	0	Execute as usual
Global Transaction	0.5	Handshake required at commit point

$\bar{e} = \sqrt{0^2 + 0.5^2} * \sqrt{1.5} = 0.6124.$

Information Exchange Dimension		
Execution	0.25	Failures require MDMS to query DBMS
Data	X	Not discussed
Schema	0	External Schema only

$\bar{i} = \sqrt{0.25^2 + 0^2} = 0.25.$

The overall autonomy violation is $\sqrt{m^2 + e^2 + i^2} = \sqrt{0.75^2 + 0.6124^2 + 0.25^2} = 1.0$

Figure 2: Autonomy Quantification for Barker and Özsu's Scheme

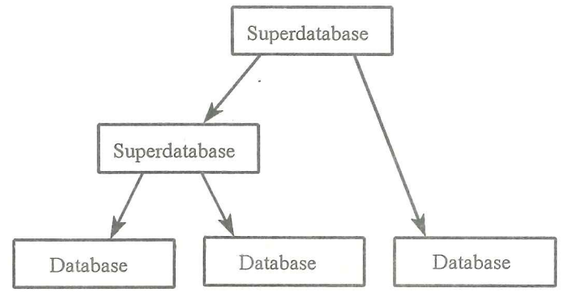

Figure 3: Pu's MDB transaction processing model

[Pu 1988, p146]

Pu's hierarchy of superdatabases

Pu [Pu 1988] describes MDBs in terms of a *hierarchy of superdatabases* as illustrated in Figure 3. Each participating DBS (called an element database) can be pictured as the *leaves* on a tree and each internal node as a superdatabase that manages the element databases in a hierarchical structure. Each element database operates independently but global activities are managed at the node level of the tree. Transactions that cross multiple element databases are called super-transactions and are posed against a superdatabase. Superdatabases utilize serializability as a transaction correctness criteria. Serializability is ensured by having each element database provide the superdatabase with information about the ordering of its local transactions by means of what he calls an O-element. Pu claims this is not necessary when element databases provide strict schedules. He assumes that each local DBMS can be modified to return the serialization order of each global transaction executed at the local site to the GTM. The GTM then uses the serialization orders from all the local sites to validate the execution of a global transaction. The approach provides a high level of concurrency at the expense of local autonomy.

There are some problems with Pu's approach. Formation of the O-element ordering requires that the local site keep the superdatabase informed of decisions made locally. This violates local autonomy. The superdatabase, and not the local databases, makes arbitrary decisions about whether transactions may commit, which also violates autonomy. The overall autonomy viola-

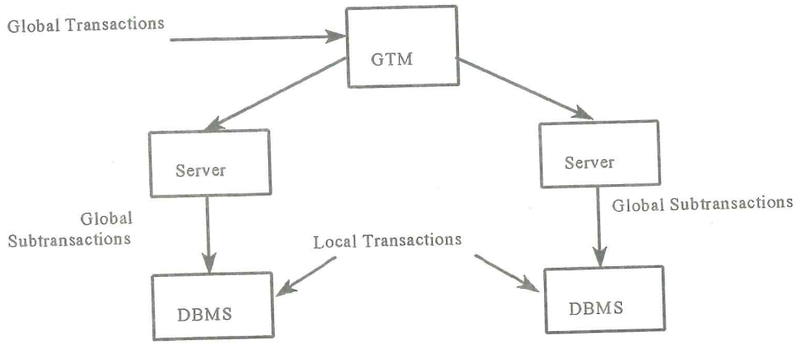

Figure 4: Breitbart *et al*'s MDB transaction processing model

[Breitbart 1995, p576]

tion for Pu's scheme is 1.25.

Breitbart *et al*'s work

Breitbart *et al* [Breitbart 1995] have done research into many aspects of MDBSs. Their approach is based on splitting up data into mutually exclusive groups of *locally* and *globally updateable* data. Breitbart and co-workers propose a transaction processing model where the software module includes a GTM at the MDB site, and a set of *servers*, one associated with each participating DBS. The MDB transaction processing model is illustrated in Figure 4. The GTM submits operations to the local DBMSs through the server, which acts as a liaison between the GTM and the local DBMS. Operations belonging to a single global subtransaction are submitted to the local DBMS by the server as a single local transaction.

A major drawback of this scheme is that it requires local transactions not to update certain data items [Mehrotra 1993, Rastogi 1993]. The overall autonomy violation for Breitbart *et al*'s scheme is 1.0897.

Elmagarmid *et al*'s work

Elmagarmid *et al* [Elmagarmid 1986a, Elmagarmid 1987, Elmagarmid 1986b] have done much work in the area of transaction management in MDBSs. They have defined a correctness criterion called *quasi-serializability* which is a weaker form of serializability. They also present a framework for designing concurrency control control protocols using a top-down approach [Elmagarmid 1987]. This means that the global serialization order of global transactions must be determined at the global level before their being submitted to the local sites. They present two mechanisms for ensuring global serialization at the local sites. The first controls the submission of global subtransactions by using a stub process, and the second controls the execution of global subtransactions by modifying local schedulers. The overall autonomy violation in the stub approach is 1.116 while in in the modification of the local scheduler approach the autonomy violation is 1.73.

Chen *et al*'s distributed MDMS

Chen *et al* [Chen 1993] extended Elmagarmid *et al*'s work by proposing a *distributed* MDMS which is not vulnerable to failures and thus addressing the reliability and recovery aspects of the model. The regular architectures which have been described up to now have a central node

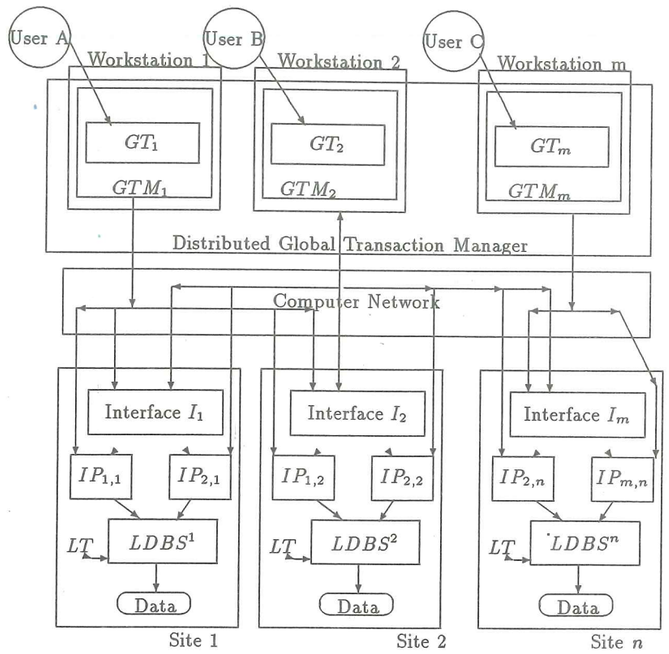

Figure 5: Chen, Bukhres and Sharif-Askary's MDMS architecture

[Chen 1993, p.95]

which, if it fails, incapacitates the whole system. The MDMS described by Chen *et al* consists of a GTM and a set of *interfaces* located at each site. The GTM controls execution of all MDMS transactions. For each MDMS transaction T_i, a GTM process GTM_i, which is responsible for the consistent and reliable execution of T_i, is issued. GTM_i is therefore coincident with the life cycle of T_i. The interface accepts and schedules the execution order of subtransactions on the local system where it resides and creates a *server procedure* for each subtransaction in the system. The server is coincident with the lifecycle of the subtransaction.

The architecture of Chen *et al*'s model is illustrated in Figure 5. Before a transaction is executed, it requests all corresponding MDMS interfaces to arrange the scheduling order of its subtransaction on the corresponding LDBSs so as to prevent any MDMS inconsistencies its execution may cause. When executing a global transaction, a GTM process only interacts with relevant MDMS interfaces, without the need to communicate with other GTM processes. This scheme makes no assumptions about the characteristics of the underlying LDBSs but rather exploits those characteristics in order to achieve global database consistency.

The disadvantages of this approach are that performance is lowered by additional network delays caused by the additional network traffic. This scheme also reduces concurrency compared to other algorithms. The network delays can be alleviated by high speed networks and the reduced concurrency is offset by the fact that no global transactions will be aborted due to deadlocks or nonserializable executions [Bukhres 1993]. The overall autonomy violation for Chen *et al*'s scheme is 0.829.

Kang and Keefe's decentralized GTMs

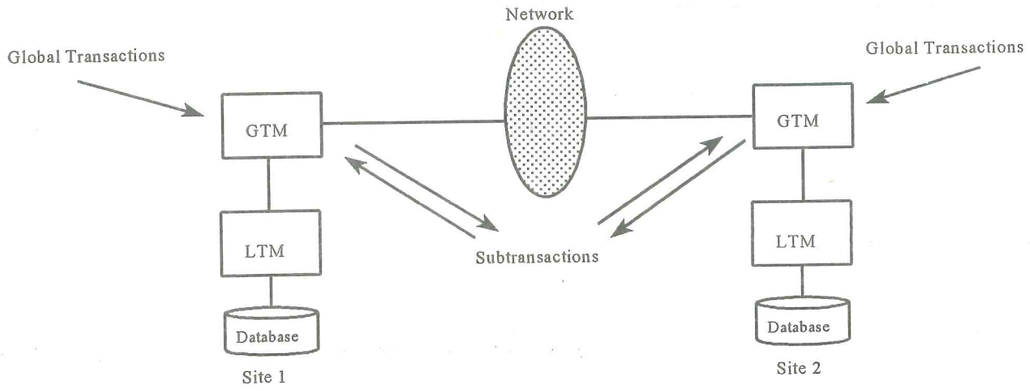

Figure 6: Kang and Keefe's MDB architecture

[Kang 1993, p458]

Kang and Keefe [Kang 1993] propose an MDB model where the GTM is totally decentralized. Their model caters for multiple versions and is illustrated in Figure 6. This scheme differs from Chen *et al*'s scheme [Chen 1993] because in their scheme the GTM is located at the site where the transaction is submitted and *that* GTM communicates with interfaces at all the other local database sites, whereas Kang and Keefe's scheme locates a GTM at each local site which does all the work — there are no servers. At each site there is a GTM accepting global transactions from users and receiving subtransactions from other GTMs via the network. The GTM maintains a global directory and therefore can determine the appropriate sites at which a global transaction will execute.

Kang and Keefe's scheme also partitions data items which once again violates local autonomy. They propose a *distributed strict timestamp ordering scheme* (DSTO) which is globally serializable. In DSTO, each global transaction is assigned a unique global timestamp when it starts. Each subtransaction carries the parent's timestamp. The GTM at each site executes strict *timestamp ordering* (TO). Strict TO blocks transactions attempting to read or write an object until the transaction that previously wrote it has either committed or aborted. The GTM ensures that conflicting operations are executed at the local site in global timestamp order by aborting transactions whose operations arrive too late. All global subtransactions are required to take-a-ticket (ticket being an object not updateable by local transactions). They only assume that the local data manager at each site outputs serializable and cascadeless schedules. Kang and Keefe prove that the DSTO scheme produces globally serializable histories in the face of failures and also prove that the scheme is deadlock free. The overall autonomy violation for Kang and Keefe's scheme is 0.93.

Garcia-Molina and Salem's sagas

Garcia-Molina and Salem [Garcia-Molina 1987] have proposed a nested transaction model intended to deal with long lived transactions. Their model uses nested transactions called *sagas*, with only two levels of nesting. A saga is not executed as an atomic unit. This means that the results of a subtransaction's execution are visible as soon as it commits and not only after commitment of the entire saga. Sagas are written so that they are interleavable with any other transactions, which makes concurrency control at the saga level unnecessary. Because of this design factor, the introduction of local transactions does not cause any concurrency control problems. In this model two assumptions are made which make this approach unsuitable in MDBS. Firstly, the model is not applicable to all MDB environments since it may be too restrictive to require that sagas be interleavable with other transactions. Secondly, it may not always be possible to write the compensating transactions that this model requires. The overall autonomy violation is 0.93.

Researcher	Description	Concurrency Control	Correctness Criterion	Autonomy Violation
Barker and Özsu	Basic MDB model	Serializability graphs	M-serializability	1.0
Pu	Superdatabases	Violates local autonomy	Serializability	1.25
Breitbart *et al*	Server model ticket method	Optimistic	Serializability	1.0897
Elmagarmid *et al*	Stub approach	Serialization events	Serializability	1.116 / 1.73
Chen *et al*	Distributed MDMS	Two phase locking & Linear ordering of resources	Serializability	0.829
Kang and Keefe	Decentralized GTMs	Distributed strict timestamp ordering	Serializability	0.93
Garcia-Molina and Salem	Sagas	No scheme needed	N.A	0.93

Table 1: Synopsis of characteristics of various transaction management schemes

Evaluation and Appraisal

The transaction management and concurrency control scheme proposed by the researchers in each of the research efforts which comprise the core group is summarized in Table 1. Assessing this table reveals the optimal approaches for transaction processing and concurrency control in MDBs.

The optimal MDB transaction processing model

After due consideration of the research done in this field, we have decided on the distributed GTM model presented by Chen *et al* [Chen 1993] as the most promising transaction management scheme. The reasons for this are that the GTM is distributed, failure resistant, and allows considerable leeway in how global transactions are handled. The interface can either send transaction operations to the local DBMS an operation at a time, or send through predetermined service requests to the DBMSs at the sites. It is also important to note that no assumptions are made about the local sites involved in the MDBS. Most importantly, LDBS autonomy is maintained and new DBSs are very easily added to and removed from the MDBS. A strong recommendation is that this scheme has been successfully implemented in the InterBase system at Purdue University.

Global concurrency control

We have reviewed various concurrency control schemes in this paper. Most of them suffer from one or more problems. They expect specific conditions to be satisfied in the LDBSs, or they are not failure resilient, or they violate local database autonomy to a lesser or greater degree, or they allow local transactions to only update a portion of the available data items, or they generate unacceptably high overhead, or they expect users to specify correct interleavings of subtransaction operations, or they could result in global deadlock, or they relax the atomicity requirement of transactions.

We propose the use of the customized global concurrency control algorithm as outlined in [Chen 1993], since it was designed for the transaction management scheme designed by Chen *et al* and because it achieves global serializability with the minimum violation of local autonomy. The algorithm utilizes the semantics of global transactions and the concurrency control strategies of the underlying LDBSs to customize a global concurrency control algorithm. It guarantees that the *relative serialization order* (RSO) of global transactions on different sites is consistent with their pre-determined relative scheduling order, ensuring that sites have the same RSO at all sites. It has been proved by Breitbart and Silberschatz [Breitbart 1988] that when global transactions have the same RSO at all sites, global serializability is preserved in the presence of local transactions.

Summary

In this article the transaction management schemes in the core group of research into transaction management and global concurrency control in MDBSs have been appriased, and the relative strengths and weaknesses of each scheme have been outlined. The optimal transaction processing model which scores well in the autonomy violation stakes, and is also reliable and failure resistant, has been proposed.

References

[Barker 1990] BARKER K. 1990. Transaction Management on Multidatabase Systems. Doctor of Philisophy Thesis. University of Alberta.

[Barker 1991] BARKER K & ÖZSU M T. 1991. Reliable Transaction Execution in Multidatabase Systems. In: *Proceedings of the First International Workshop on Interoperability in Multidatabase Systems,* edited by Y Kambayashi, M Rusinkiewicz & A Sheth. Los Alamitos: IEEE Computer Society Press.

[Barker 1994] BARKER K. 1994. Quantification of autonomy on Multidatabase systems. *Journal of Systems Integration,* 4(2):151-169.

[Bell 1992] BELL D & GRIMSON J. 1992. *Distributed Database Systems.* Great Britain. Addison Wesley.

[Bradshaw 1993] BRADSHAW D P, LARSON P A & SLONIM J. 1993. Concurrency Control in Multidatabase Management Systems. *Research Report CS-93-34,* Department of Computer Science, University of Waterloo.

[Breitbart 1988] BREITBART Y & SILBERSCHATZ A. 1988. Multidatabase Update Issues. *SIGMOD Record.* 17(3):135-142.

[Breitbart 1995] BREITBART Y, GARCIA-MOLINA H & SILBERSCHATZ A. 1995. Transaction Management in Multidatabase Systems. In: *Modern Database Systems,* edited by: Won Kim. New York: Addison Wesley.

[Bright 1992] BRIGHT M W, HURSON A R & PAKZID S. 1992. A Taxonomy and Current Issues in Multidatabase Systems. *Computer,* 25(3):50-60.

[Bukhres 1993] BUKHRES O, CHEN J, DU W, ELMAGARMID A & PEZZOLI R. 1993. Interbase: An execution environment for global applications over distributed, heterogeneous, and autonomous software systems. *IEEE Computer.* August 1993.

[Chen 1993] CHEN J, BUKHRES O A & SHARIF-ASKARY J. 1993. A Customized Multidatabase Transaction Management Strategy. In: *4th International Conference, DEXA '93. Database and Expert Systems Applications,* edited by: Vladimír Mařik, Jiři Lažanský, Roland R Wagner. Prague, Czech Republic, September 1993 Proceedings.

[Du 1989] DU W & ELMAGARMID A K. 1989. Quasi serializability: a correctness criterion for global concurrency control in InterBase. In: *Proceedings of the Fifteenth International Conference on Very Large Databases,* edited by P M G Apers & G Wiederhold. Palo Alto: Morgan Kaufmann.

[Elmagarmid 1986a] ELMAGARMID A K & HELAL A. 1986. Heterogeneous database systems. Technical Report TR-86-004. The Pennsylvania State University, University Park, Pennsylvania 16802.

[Elmagarmid 1986b] ELMAGARMID A K & HELAL A. 1986. Supporting updates in heterogeneous distributed database systems. In: *Proceedings of the fourth International Conference on Data Engineering.*

[Elmagarmid 1987] ELMAGARMID A K & LEU Y. 1987. An optimistic concurrency control algorithm for heterogeneous distributed database systems. *Q. Bull IEEE TC on Data Engineering.* 10(3):26-32.

[Elmagarmid 1990] ELMAGARMID A K & DU W. 1990. A paradigm for concurrency control in heterogeneous distributed database systems. In: *Proceedings of the Sixth International Conference on Data Engineering.* Los Alamitos: IEEE Compter Society.

[Garcia-Molina 1987] GARCIA-MOLINA H & SALEM K. 1987. Sagas. In: *Proceedings of ACM-SIGMOD. 1987 International Conference on Management of Data.* San Francisco.

[Kang 1993] KANG I E & KEEFE T F. 1993. Supporting reliable and atomic transaction management in multidatabase systems. In: *Proceedings of the 13th International Conference on Distributed Computing Systems.* Los Alamitos: IEEE Computer Society Press.

[Mehrotra 1993] MEHROTRA S. 1993. Failure-Resilient Transaction Management in Multidatabase Systems. Doctor of Philosophy Dissertation. University of Texas at Austin.

[Mullen 1992] MULLEN J G, ELMAGARMID A K, KIM W & SHARIF-ASKARY J. 1992. On the Impossibility of Atomic Commitment in Multidatabase Systems. In: *Proceedings of The Second International Conference on Systems Integration.* edited by: P A Ng, C V Ramamoorthy, L C Seifert & R T Yeh. Los Alamitos: IEEE Computer Society Press. p625-634.

[Özsu 1990] ÖZSU M T & BARKER K. 1990. Architectural Classification and Transaction Execution Models of Multidatabase Systems. *Lecture Notes in Computer Science,* 468(5):285-294.

[Özsu 1991] ÖZSU M T & VALDURIEZ P. 1991. *Principles of Distributed Database Systems.* Englewood Cliffs, New Jersey. Prentice Hall.

[Pu 1988] PU C. 1988. Superdatabases for composition of heterogeneous databases. In: *Multidatabase Systems: An Advanced Solution for Global Information Sharing,* edited by A R Hurson, M W Bright & A Pakzad. Los Alamitos: IEEE Computer Society Press. Los Alamitos, California.

[Pu 1991] PU C, LEFF A & CHEN S F. 1991. Heterogeneous and Autonomous Transaction Processing. *Computer.* Dec 1991.

[Rastogi 1993] RASTOGI R R. 1993. Concurrency Control in Multidatabase Systems. PhD Thesis. University of Texas.

[Yun 1993] YUN H & HWANG B. 1993. A Pessimistic Concurrency Control Algorithm in Multidatabase Systems. In: *Database Systems for Advanced Applications '93. Proceedings of the Third International Symposium on Database Systems for Advanced Applications. Taejoon, South Korea,* edited by: S Moon & H Ikeda. Singapore: World Scientific. p379-386.

COMPUTER-BASED APPLICATIONS FOR ENGINEERING EDUCATION

A C Hansen and P W L Lyne
Department of Agricultural Engineering
University of Natal
Pietermaritzburg

Abstract

There is a rapidly growing awareness in the world of the need to modify the educational process at all levels of human development. In South Africa the situation is made worse by a substantial backlog in education. Technology in the form of computers presents itself as a valuable teaching resource. In the USA conclusive results indicate that the use of computers in training accelerates learning by up to 40%. However, a study of the pedagogical aspects of computer based education shows that there is a great complexity involved in learning and it is impossible to expect that a computer can fulfil all the requirements in an effective learning environment. The inclusion of images, video or sound in a computer-based application can enhance the learning experience significantly as well as increase self-motivation. In the USA, increasing use of multimedia is being made in the field of engineering in both tertiary education and extension. This paper presents some methods for classifying instructional materials. Examples of applications in the different classifications are used to illustrate some dynamic interactive tools that are available for educating engineers.

Introduction

From the time that computers came into being, people have predicted that computer-based education would revolutionise education and training with respect to both effectiveness and efficiency. In spite of these expectations very few of the systems developed over the past three decades have lived up to the promises made for them (Reeves, 1994). However, a study of the literature in the engineering education field indicates that the situation is changing rapidly with much effort being expended in the development of properly designed computer-based instructional materials. Coupled with this development is a greater awareness of the need to incorporate sound pedagogical objectives into the system and to perform appropriate and substantial evaluations in an iterative manner during the development process. Engineers, who in the past were perhaps ignorant of the learning strategies that should be followed, are now seeking the assistance of education specialists in providing instructional materials in the engineering field.

Some of the advantages in making use of computer-based instruction as compared to traditional classroom tuition include interactivity, learner control, consistency, motivation, reduced administration and greater cost-effectiveness (Sheldon *et al.*, 1993). In addition, conclusive results in the USA indicate that the use of computers in training accelerates learning by up to 40% (Andrews, 1994). However, a major disadvantage of computer-based instruction is that it can erode interpersonal relationships between teachers or facilitators and learners. The computer remains an inanimate tool. The technical difficulties inherent in building student models and simulating human-like communications have been greatly underestimated by proponents of intelligent tutoring systems (Reeves, 1994). In relation to the teacher or trainer, computers are more efficient and reliable sources of content while human instructors are better providers of individual guidance, encouragement, and detailed explanation (Reeves, 1994). Hence the teacher's role in the development and implementation of computer-based instruction cannot be neglected and has to be evaluated very carefully.

The objectives of this paper are firstly to put forward a set of educational objectives that should underpin any development strategy for computer-based instructional materials, and secondly to provide examples of applications that fall into different categories that address different objectives and learning styles.

Educational Objectives

The development of effective computer-based instructional materials can be achieved only if the learning objectives of such materials are clearly understood at the beginning of the process. Montgomery and Fogler (1996) state that a well designed package can supplement the presentation of basic course concepts, test their understanding, provide feedback for their efforts, and generally increase both the quantity and the depth of their learning. However, a poorly designed program can lead to massive expenditures of time by both teachers and students without enhancing learning at all.

A system suggested by Montgomery and Fogler (1996) for classifying software in terms of potential effectiveness in a given classroom is based on answers to the following three questions:

a) What thinking skills is the software designed to challenge?
b) What student learning styles does the software accommodate?
c) What are the intended roles of the software?

The well-known and widely used Bloom's taxonomy of educational objectives is composed of the following six skill levels (Montgomery and Fogler, 1996):

a) *Knowledge:* the rote repetition of memorised information. Can the problem be solved simply by defining terms and by recalling specific facts, trends, criteria, sequences, or procedures? This is the lowest intellectual skill level.
b) *Comprehension:* repetition with understanding. Given a familiar piece of information, can the problem be solved by recalling appropriate information and using it? This is the first level of understanding.
c) *Application:* recognition of which set of principles, ideas, rules, equations, or methods should be applied, given all the pertinent data. Once the principle is identified, the problem is solved using skill levels 1 and 2.
d) *Analysis:* process of breaking the problem into parts so that the hierarchy of sub-problems or ideas and their relationships are made clear. Missing, redundant and contradictory information are identified and after the analysis each sub-problem can be solved using skills 1,2 and 3.
e) *Synthesis:* creation of a new whole from a number of parts such as in a design project. Once the various parts are synthesised, each part or problem requires the use of skill level 4 to continue towards a complete solution. This level relies on creative thinking.
f) *Evaluation:* qualitative and quantitative judgements are made about the extent to which the materials or methods satisfy internal and external criteria. This level emphasises critical thinking.

Undergraduate courses typically focus on the first three levels only. Properly formulated computer-based applications represent one way of allowing students to exercise higher level thinking skills.

After the desired skills have been identified, it is necessary to take into account the learning styles of the students in order to reach them more effectively and ensure that these skills are exercised (Montgomery and Fogler, 1996). The issue of learning styles has become a major consideration in the teaching/learning process. Rosati and Felder (1995) suggest that a student's learning style is the collective way in which a student preferentially takes in and processes information.

A learning style classification system proposed by Felder and Silverman (1988), cited by Rosati and Felder (1995) has been applied by a number of researchers and was identified by Montgomery and Fogler (1996) as being the most useful model in helping students understand their learning needs and preferences. This system comprises five dimensions as shown in Table 1. Results of assessment of learning styles of engineering students by Montgomery (1995) and Rosati and Felder (1995) indicated that approximately 67% of students learn best actively, 57% are sensors, 69% are visual and 28% are global.

The combination of Bloom's taxonomy of educational objectives and the learning style classification system provide a means of determining both the skills being exercised by the software and the mode of interaction with the student. These two considerations can be used in the assessment of the role of computer-based applications in engineering education.

Table 1 Dimensions of the inventory of learning styles (Montgomery and Fogler, 1996)

Dimension	Range	Comments
Perception	Sensing/Intuitive	Sensors prefer data and facts, intuitors prefer theories and interpretations of factual information.
Input	Visual/Verbal	Visual learners prefer charts, diagrams and pictures, while verbal learners prefer the spoken or written word.
Organisation	Inductive/Deductive	Inductive learners prefer to infer principles from facts and observations, while deductive learners deduce consequences from underlying principles.
Processing	Active/Reflective	Active learners learn best by doing something physical with the information, while reflective learners do the processing in their heads.
Understanding	Sequential/Global	Sequential learners make linear connections between individual steps easily, while global learners must get the "big picture" before the individual pieces fall into place.

Classification of Software

A vast range of software, available at present, is able to contribute to the education of engineers in various ways. Montgomery and Fogler (1996) provide a classification scheme that can be used to categorise this software by its role in addressing specific educational objectives and learning styles. The following four categories are presented by Montgomery and Fogler (1996):

a) *Presentation:* emphasis is on the knowledge, comprehension and application levels of Bloom's taxonomy. The software focuses on the delivery of technical material. Montgomery and Fogler (1996) list the ways in which these materials can be presented with their corresponding learning style dimensions:
   - Display of text material (verbal)
   - Access to expanded explanation of text material through hot keys (active, sequential)
   - Visual and graphical representation of material (visual, sensing, global)
   - Use of animation to display phenomena (global), or manipulate equations (active)
   - Display of video clips to illustrate industrial situations (global, visual, sensing).
b) *Assessment:* student is tested on mastering of material, such as through the use of multiple choice questions, which are often closed-ended and tend to focus on the first four levels of Bloom's taxonomy, although the upper two levels of synthesis and evaluation can be reached. Certain types of assessment can cater for active (through interaction), sequential (orderly) and sensing (if it deals with real situations) learners.
c) *Exploration:* allows users to better understand the role of various parameters on the performance of a given process through exploration of the process. These are exploratory simulations within a confined parameter space which provide students with a variety of problem definition alternatives and solution pathways to follow. Active learners appreciate the chance to manipulate parameters, visual learners benefit from graphical representations of phenomena, deductive learners can practice drawing their own conclusions, and sensors and global learners get to experience a real process, or at least a simulation of it. This type of software focuses on application and analysis in Bloom's taxonomy.
d) *Analysis:* includes software packages that allow students to enter equations and parameter values for any system. These packages include spreadsheets and equation solvers which allow users to create and solve new models and the corresponding sets of equations very easily. Software in this category gives students, particularly active, deductive and visual students, practice of the higher levels of Bloom's taxonomy, synthesis and evaluation.

Development Process

In the development of a Farm Tractor and Machinery Certification CAI (Computer Aided Instruction) program, Sheldon *et al.* (1993) identified six phases:

a) Discovery: this phase involved a task analysis, collection of resources, and generation of ideas.
b) Design: in the design phase, performance objectives were developed from the steps of the discovery phase. Prerequisite skills were identified and an outline of the program structure was developed.
c) Development: the development phase of the model involved specifying learning activities and sketching lesson displays on paper. Then the entire program was outlined on paper.
d) Coding: the actual computer programming took place in this phase.
e) Documentation: both internal documentation which explains the function of coding and external documentation which includes user guides and exercises are important components of this phase of the model.
f) Delivery: this stage includes pilot testing, validation, and field testing of the materials.

Grayson (1996) emphasised that the development of effective computer-based instructional materials should be iterative and interactive with the students. The following steps were put forward by Grayson (1996) as desirable components in a model for instructional software development:

a) Select topic (content area) and target group;
b) Identify student conceptual difficulties;
c) Write preliminary version of program;
d) Try it out with students - observe what they do, and how they interact with the program, test what they learn, and solicit students' comments and suggestions;
e) Modify program on the basis of step 4;
f) Try it out with students;
g) Repeat step 5 and 6 until the program is reasonably robust and fairly stable.

Examples of Applications

Some examples in each of the software categories are presented in this section. These examples are drawn mainly from the range of software applied in the Department of Agricultural Engineering, University of Natal.

Presentation

Many examples can be found in this category. One such interactive program called ChainSaw was developed by Lown *et al.*(1993) from the University of Tennessee in the USA to teach chain saw and tree-cutting safety . Seventy screens in ten sections containing chain saw and tree-cutting facts, figures, procedures, photos, simple animations, and sounds are arranged in a notebook format. Figure 1 illustrates the opening screen or first page of the program. The information can be accessed sequentially page by page or particular topics can be referred to by selecting the appropriate label in the index on the right hand side of the page in Figure 1. Four of the five methods of delivering technical material mentioned earlier are employed in this software.

A major avenue being explored by software developers in the presentation category is the World Wide Web, which incorporates support for multimedia presentation of material. Relatively little interaction has been possible until more recently with the development of Java applets and ActiveX controls. The opportunities and potential for delivery of instructional materials via "the Web" appear to be enormous and hence it can be expected that there will be tremendous expansion in this area subject to connection speeds.

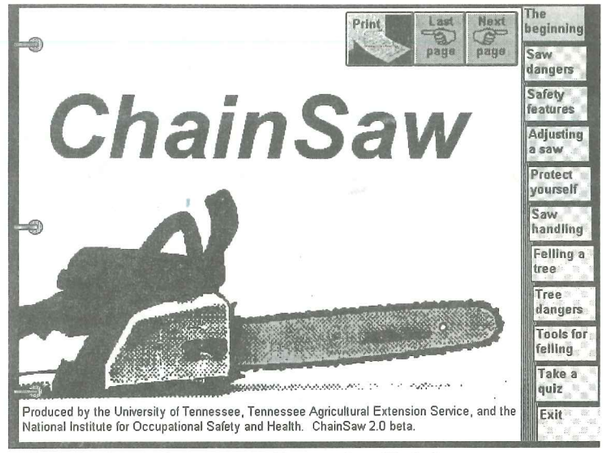

Figure 1 First screen of interactive ChainSaw program

Assessment

Packages in the assessment category allow students to test their knowledge of the material. The ChainSaw program described above includes a quiz in which students are required to select a correct answer to a question from a list. The program keeps track of the number of questions answered correctly.

An example presented by Montgomery and Fogler (1996) involves a program that helps students prepare for an undergraduate laboratory experiment on pumps. Some short questions as illustrated in Figure 2 give students a chance to practise their laboratory skills. Sensing and active learners benefit from this type of interaction, as do reflective learners, in that it allows them to gain confidence in tackling the actual equipment.

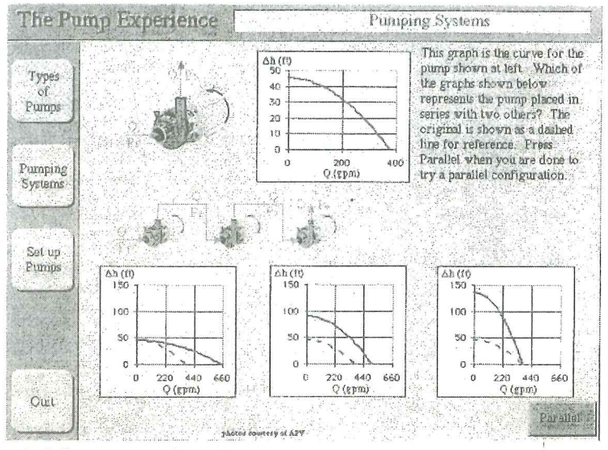

Figure 2 Interaction in pumps for assessment of understanding of laboratory equipment (Montgomery and Fogler, 1996)

Exploration

Computers offer great opportunities for developing simulations. One example of such a simulation was reported by Hansen *et al.*(1995), in which a simple PC-based diesel engine simulator was developed with