Volume 19 Number 2 May 1995 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Guest Editor: Jerzy R. Nawrocki

Integration Multidatabase Model

Software Reliability Control

Disk Mirroring

Parallel Gaussian Elimination

Deep Knowledge

The Slovene Society Informatika, Ljubljana, Slovenia

Informatica **19** (1995) Number 2, pp. 177–290

1977

Informatica

An International Journal of Computing and Informatics

Basic info about Informatica and back issues may be FTP'd from ftp.arnes.si in magazines/informatica ID: anonymous PASSWORD: <your mail address>
FTP archive may be also accessed with WWW (worldwide web) clients with
URL: `http://www2.ijs.si/~mezi/informatica.html`

Subscription Information: Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 61000 Ljubljana, Slovenia.
The subscription rate for 1995 (Volume 19) is
– DEM 50 (US$ 35) for institutions,
– DEM 25 (US$ 17) for individuals, and
– DEM 10 (US$ 7) for students
plus the mail charge DEM 10 (US$ 7).
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LATEX Tech. Support: Borut Žnidar, DALCOM d.o.o., Stegne 27, 61000 Ljubljana, Slovenia.
Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.
Printed by Biro M, d.o.o., Žibertova 1, 61000 Ljubljana, Slovenia.
Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Department for Computer Science, Jožef Stefan Institute: Tel (+386) 61 1259 199, Fax (+386) 61 219 385, or use the bank account number 900–27620–5159/4 Ljubljanska banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

According to the opinion of the Ministry for Informing (number 23/216-92 of March 27, 1992), the scientific journal Informatica is a product of informative matter (point 13 of the tariff number 3), for which the tax of traffic amounts to 5%.

Informatica is published in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Franjo Pernuš)
Slovenian Artificial Intelligence Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
Slovenian Association of Technical and Natural Sciences (Janez Peklenik)

Referees: David Abramson, Azer Bestavros, David Cliff, Travis Craig, Hesham El-Rewini, Terrence Forgarty, Hugo de Garis, David Green, Inman Harvey, Rod Howell, Eric Johnson, Li-Shan Kang, Gilad Koren, Phil Laplante, Bud Lawson, Joseph Y-T. Leung, Raymond Lister, Doug Locke, Zbigniew Michalewicz, Daniel Mossé, Jerzy Nogieć, Stefano Nolfi, Warren Persons, Gary Preckshot, Michael Quinn, Felix Redmill, Bo Sanden, William Spears, Przemysław Stpiczyński, Jurij Tasič, Ken Tindell

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Cur. Cont. & Comp. & Math. Sear., Engineering Index, INSPEC, Mathemetic Reviews, Sociological Abstracts, Uncover.

The issuing of the Informatica journal is financially supported by the Ministry for Science and Technology, Slovenska 50, 61000 Ljubljana, Slovenia.

RIMM: A Reactive Integration Multidatabase Model

Niki Pissinou, Vijay Raghavan and Kanonkluk Vanapipat
The Center for Advanced Computer Studies
University of Southwestern Louisiana
pissinou@cacs.usl.edu

Keywords: multidatabases, E-C-A rules, temporal objects, mediators

Edited by: Xindong Wu

Received: August 5, 1994 **Revised:** November 7, 1994 **Accepted:** January 10, 1995

Traditional solutions to multidatabase systems mainly focus on resolving the structural and semantic incompatibility among the local databases to provide one or more frozen shared schemas to the users. However, the limitations of such approaches are evident in environments where changes might occur frequently. Specifically, it is impractical to assume that once set up, the global interface would remain valid and frozen in time. Over time the properties, behaviors, roles, and perhaps the identities of the objects in the local systems may change to reflect the evolution of the modeled universe. Additionally, the information requirements of the global users may change to reflect their needs. Such dynamism makes the functionalities of a multidatabase system obsolete. In this paper, we provide a paradigm for the dynamic interaction between local and global systems to resolve foreseeable conflicts that may occur over time, while supporting object consistency across databases and object relativism between local and global systems. Based on this paradigm, we develop a formal Reactive Integration Multidatabase Model (RIMM) that contains the expressiveness to represent the temporal changes, temporal conditions, and events in the real world. The model can be utilized as a framework to realize the multidatabase architecture that incorporates the event driven production rules to automatically react to anticipated changes of local databases and global users and dynamically reconfigure the global interfaces, so as to support interoperability over time. Our targeted application domains include geographic information systems, scientific database systems, and digital libraries.

1 Introduction

In today's information-oriented society, the need to share information is increasing rapidly. With advances in computer and communication technologies, sharing information across *heterogeneous database systems* [1] has become not only an obvious trend but a necessary task in many large organizations. While networking technology provides physical connectivity for these heterogeneous databases [12], the logical connectivity of such databases can be accomplished only when the heterogeneities of the data stored in these diverse databases are resolved. However, this is still yet a major challenge that remains to be realized.

The endeavors to answer the quest for logical connectivity to achieve meaningful information sharing among the heterogeneous and independent databases have appeared under various research topics, such as global information-sharing systems [5], interoperable systems [7, 4, 34], heterogeneous database systems [7, 34], federated database systems [34], and multidatabase systems [4, 23, 25]. In this paper, we make no distinction among these systems and use them to refer to a collection of *heterogeneous database systems* that are logically integrated to provide one or more

[1] We use the term *heterogeneous database systems* to indicate that these database systems are pre-existing and separately-developed, and they may be different in the data models, schemas, data definition and data manipulation facilities, data representations, operating systems, and hardware environments.

unified views of the databases to a set of users (called global users).

A multidatabase system includes two main components: a set of local databases and a number of global users/applications. In general the global users access data from local databases through global interfaces. Serving as go-between agents, global interfaces manipulate requests from the global users and data from the local databases. It has been recognized that achieving interoperability among many separately-developed systems is one of the most difficult tasks currently facing the database research community and industry.

Interoperability among heterogeneous databases has been a research issue for at least a decade. The primary objective of the research is to provide frameworks for database integration to support interoperability and information sharing over time. Database integration is defined as a tool to combine or interface data and functions from multiple disparate data sources (e.g., database systems) into a cohesive system so that the heterogeneity of the underlying systems and data transparent to the users [18]. Its goal is to provide access to data that managed by different systems, may be stored in different structures, and may have different semantics.

Traditional solutions to database integration can be generalized into two major approaches: tightly-coupling and loosely-coupling [13, 14, 16, 17, 34]. The tightly-coupling approach mainly focuses on resolving the structural and semantic incompatibility of the local databases to provide one or more integrated schemas (viz., federated schema) as the global interfaces to the global users. Hence, the data access is restricted to via the integrated schemas, and explicit source selection subjected by the users' preferences is not allowed.

The process of creating one or more integrated schemas is termed *schema integration* [3]. It provides users with one or more conceptual views of the participating database schemas while hiding the heterogeneity of the data models and representation and preserving the autonomy of the database sites. However, schema integration is an extremely tedious and labor intensive task, and it becomes more and more complex and error-prone as the number of schemas to be integrated increases. The inherent complexity of the schema integration suggests that the integrated schemas are expected to remain valid and frozen in time, once they are set up. Any changes to an integrated schema may require the complete process of schema integration to be repeated. In this respect, the tightly-coupling approach is likely to generate systems that are insusceptible to changes.

Instead of restricting the global users to the integrated-schemas, the loosely-coupling approach provides a data manipulation language to the global users and allows them to query the local databases directly. Thus, the users have unrestrained access to all local databases. Generally, each local database makes available to the global users an export schema to assist in query formulation. However, the semantic interoperability is only supported through the data manipulation language, and the data semantics are usually not obvious from the export schemas. Thus, it is the users' responsibility to identify, understand, and resolve the semantic conflicts. Such burden on the users is unrealistic and unacceptable in the environment where the data semantics may change over time.

There have been some recent research attempts to incorporate knowledge representation techniques into multidatabase systems[1, 9]. In these approaches, a global semantic model is constructed for integrating the different representations and meanings of the local databases. Thus in addition to the schematic details, the model also explicitly represents the underlying semantics of the databases. However, the support of the semantic heterogeneity is limited and mostly described in the form of mappings from the different semantic representations into a common interpretation in the global knowledge representation [12].

The research into the notion of context interchange [12, 36] further extends the support for the semantic heterogeneity. In addition to the heterogeneity and the autonomy of the local databases, the approach also recognizes the differences as well as the individuality of the global users. It focuses on the representation of disparate data semantics. Specifically, the data semantics of both the local databases and the global users are explicitly represented in the export and the import contexts (viz., export and import schemas [34]). Since the data semantics are clearly described for the local databases as well as the global users, the semantic conflicts can be automatically detected and resolved (i.e., conversion).

The automatic recognition and resolution supported by the context interchange is limited in the sense that it is lacking support for the temporal changes (e.g., the temporal semantics) that may be subjected to the temporal conditions. These temporal conditions require the additional capability of monitoring the situations or events of interest and executing some predefined actions in a timely manner when appropriate events are detected. As an example, a temporal change in the vehicle navigation systems can be described as following: "after construction has been completed, the highway will have a third lane." The temporal condition associated with this example would be the *after* completion of the construction, and the semantic (i.e., property) associated with the highway and subjected to the indicated temporal condition is that this highway will be three-lane.

Since global interfaces retrieve data from local databases and format the data according to the needs of the global users, the interfaces should respond to either changes in the databases or users' request. Thus, the global interfaces should be "dynamic", instead of "static". Moreover, there are applications that are inherently dynamic, such as vehicle navigation systems (a GIS application), scientific database systems, and hospital monitoring systems. These applications may need continuous, fast access to diverse databases. Correct and up-to-date information is critical to their operations, and changes (e.g. in vehicle navigation systems, a road may be blocked because of some accident, a tunnel may be flood, and construction of a new road) may affect these systems radically. Therefore, the role of the global interfaces becomes even more important, and these interfaces must be capable of capturing possible changes automatically.

A database with such a capability may be called an *active database*. Research in active databases has incorporated active rules into traditional databases to permit them to detect and respond to the events of interest automatically. The additional capability allows the database systems to deal with problems concerned with observation of objects and situation (e.g. the database state, temporal event, and application defined events) and executing some predefined actions in a timely manner when certain events are detected. Among the advantages of integrating active production rules into database systems include [21]:

1. Extending the modeling capabilities of a database to realize efficient integrity constraint enforcement through triggers and alerters. Thus, external integrity maintenance programs can be avoided.

2. Provide a uniform mechanism for derived data maintenance, security, and version control.

3. Provide a suitable framework for large and efficient knowledge-base and expert systems.

Incorporating the active production rule paradigm into a global information-sharing environment of heterogeneous databases provides a suitable platform for specification of global reactions to the dynamic changes of the local databases, as well as of the global users. Such rules extend the power of multidatabase systems to deal with observation of local (as well as user) objects and events. When situations or events of interest occur, the appropriate responses (e.g. updating the global knowledge or signaling the global user for new information) are triggered in a timely manner. This allows the explicit specification of the temporal conditions (i.e. situations or events which may cause conflicts to occur over time) to be monitored, and the automatic recognition and resolution of the monitored conflicts.

The existing multidatabase architectures are not equipped with mechanism to support these rules. This suggests a new architecture must be developed to integrate the active database rules into the multidatabases. To do so, the impact of space and time on the interoperability of multidatabases needs to be identified. However, little attention has been given to this pending problem.

As the world changes over time, it may cause the databases to evolve and the users to change task and information requirements. There are two kind of dynamics which may occur in the world: anticipated and unanticipated. The anticipated dynamics are the expected changes, and they are predictable. On the other hand, the unanticipated dynamics are the changes that are unknown and unpredictable. So far, little attention has been given to the impact of time on the interoperability of multidatabases. One of the problem facing the existing multidatabase solutions is their weak expressiveness for the representation of the temporal changes, as well as the temporal conditions, and the representation of events in the external world.

Since the databases and users are dynamically changing, static interfaces may become obsolete and incorrect over time. It is natural to expect that the dynamics of the local databases and the global users will necessitate the modifications to the global interfaces. However, existing global interfaces are passive in the sense that any changes to the local databases and global users must be manually propogated to the global interfaces. In other words, when the local databases and global users change over time, the global administrator would be contacted to register and make the changes to the global interfaces. Such traditional global interfaces are not adequate to applications, such as in geographical information systems and scientific database systems, that are inherently dynamic. The dynamic changes affect these systems radically. These applications require interfaces that are able to capture the possible changes, specifically anticipated changes automatically.

In this paper we emphasize the dynamic interactions among the global system and the local databases to maintain object consistency between local databases and object relativism between local and global systems. In particular, for the global interfaces to monitor, detect, and react to the anticipated evolution of the local databases and global users, they need *temporal* and *active* capabilities. In view of this, the strong interest of this research is to develop a reactive multidatabase architecture that is capable to automatically reacting to anticipated changes and dynamically reconfiguring the global interfaces to support interoperability over time. However, a major barrier in developing such an architecture is the lack of the underlying principle and foundation. Our work presents a new integration model that places special emphasis on the events which may cause objects to transform. It provides a new paradigm for dynamic interactions between the local database systems and the global users to resolve the *foreseeable* conflicts which may occur over time.

The remainder of this paper is organized as follows. In section 2, we review the literature in the heterogeneous multidatabases that are closely related to our research context. In section 3, we study the temporal objects in multidatabase environment to provide principles and foundations for our Reactive Integration Multidatabase Model (RIMM) that is presented in section 4. The last section contains our conclusion and the future plan for the research.

2 Research Context

The need for distributed systems that can share information from all participating sites makes multidatabase systems an important part of the advancing information technology. Over the past decade multidatabases have been an active research area in database community. These research efforts have addressed the need for interoperability among pre-existing autonomous systems, and a number of systems have been developed in both academia and industry [5, 4, 11, 15, 16, 17, 20, 24, 25, 26, 34, 37].

Traditional solutions to global information sharing mainly alleviate the syntactic heterogeneity problems and lack the techniques to solve the problems of semantic heterogeneity. Several current research activities have extended in the direction of intelligent interoperability in semantically heterogeneous multidatabase environments so that the global systems become intelligent agents and capable of examining the semantic differences of the participated local systems. To motivate the discussion of our research, in this section we shall overview the current trends in multidatabase research and projects which related to our work.

An important problem of semantic interoperability in multidatabase environment is maintaining the consistency of *interdependent data* [2] physically stored in multiple databases. Sheth, et al.,

[2]Sheth, et al., use the term *interdependent data* to imply that "two or more data items stored in different databases are related through an integrity constraint that specifies the data dependency and the consistency requirements between these data items" [35].

proposed a polytransaction framework to address such problem. In their framework, the interdatabase dependencies can be specified in the declarative fashion in the Data Dependency Descriptors (D^3) which constitute the Interdatabase Dependency Schema (IDS). There are three components to each of the data dependency descriptor: data dependency information (predicate), mutual consistency requirements (predicate), and consistency restoration procedures. When a transaction updates a data item in a database, the data dependency information stored in the IDS can be used to convert the transaction into *polytransactions* to perform the activities required to maintain consistency of the interdependent data.

One direction in which research efforts have proceeded toward to ensure semantic interoperability is to incorporate active database rules into multidatabase systems. Ceri and Widom used active database rules to maintain the consistency of interdependent data in multidatabases [8]. They considered the environments in which the presence of data in one database depend on the existence of related data in another database, and the value of data in one database depend on the value of related data in another database. They use active rules and persistent queues to address the problem of data consistency across semantically heterogeneous distributed systems. Their approach manages semantic conflicts by maintaining integrity constraints across database systems.

The Distributed Object Management (DOM) project at the GTE Labs [2] also utilizes the active database paradigm to model heterogeneous systems. In the DOM framework, the heterogeneous systems which participated in the global information sharing are modeled as a collection of *active objects*. An active object which incorporated with mechanisms to execute autonomously and asynchronously actions upon the detection of the monitored *events* and the predefined conditions associated with the events become true. Therefore, the participated heterogeneous systems are capable of monitoring events (either of their internal state or of the active object space) and reacting autonomously and asynchronously when predefined conditions become true.

Another direction in which current interest on multidatabase research and projects has also proceeded toward is developing system architectures and frameworks based on the mediator architecture. Sciore, et al., proposed a system architecture to support semantic interoperability [36]. In this architecture, the *context mediator* is the central component. A context mediator is defined as an "agent that directs the exchange of values from one component information system to another, and provides services such as inter-system attribute mapping, property evaluation, and conversion [36]." Their main idea to ensure semantic interoperability is to capture the semantics of the shared data as a unit of exchange called the *semantic values* and make these values sharable among the components. Semantic values can either be stored explicitly or defined in the *data environment* where the context mediator consults to determine the semantics of the data to be shared or exchanged. In addition, mapping knowledge which describes equivalences among the participated systems is contained in the component called the *shared ontology*, and a set of conversion functions is maintained in the *conversion library* to be used in converting semantic values from one context to another context. The context mediator architecture is incorporated into a relational system and called a relational Context-SQL (C-SQL) system.

The idea of *context mediator* is extended in the *context interchange* framework [12]to construct large-scale interoperable database systems where participated local data sources (e.g. databases) and global data receivers (viz. *global users*) may frequently enter and exit the system. The *context interchange* architecture mainly differs from the *context mediator* architecture [36] when there are multiple data sources and receivers in two ways. First, the *context interchange* allows multiple data source and receivers to have commonly-held assumptions, called *a supra-context*, as well as the assumptions which are peculiar to a particular source or receivers, called *micro-context*. Second, the *context interchange* provides two ways for the data receivers to query multiple data sources. One way is for the data receivers to query the data sources directly through their export schemas. The other way is for the data sources to query the data sources through the *external views* which may be defined on *federated schema* (viz. integration of two or more data source schemas).

Related to the concept of mediators, Papazoglou applied the techniques from the Distributed Artificial Intelligent (DAI), specifically the Multiagent Systems (MAS), in distributed databases to support intelligent and cooperative interoperability [28]. The DAI has mainly focused on techniques and tools for a collection of independent autonomous intelligent agents to solve mutual problems in a cooperative fashion. Therefore, to facilitate cooperation between agents and to coordinate the agent interaction sequences, concepts such as contracting and negotiating, multiagent planning, and case-base reasoning (use past experience to predict future interaction and solve complex problems) have been developed. The incorporation of these AI concepts into the distributed database technologies added intelligence to the internal mechanisms of the distributed systems and hence led towards the new generation of *distributed intelligent interoperable systems* [28].

However, our research takes a different view from these approaches. Our research addresses the temporal conceptual aspects of objects in the distributed environment where the databases are semi-autonomous (viz., interdatabase data dependency is possible) and heterogeneous. Over time objects in databases and users' task may dynamically evolve to reflect changes in the real world. Thus such changes must be incorporated into the global interfaces so that the share information always valid and meaningful. Our research concentrates on developing a formal reactive integration model that contains the expressiveness to represent the temporal changes, temporal conditions, and the events in the real world.

3 Temporal Objects in Multidatabase Context

A multidatabase system (MDBS) maintains interfaces to allow the global users to access or manipulate data from the diverse databases that participated in the information sharing. However, as the dynamic world changes either predictably or unpredictably, it may cause the following two changes:

1. Objects in the databases evolve to describe the changes in the world,
2. Users change task which may alter their information needs and/or information requirement format.

A typical local database models a part of the evolving universe where the state may change, both predictably and unpredictably, over time. Hence, objects in local databases may be modified to reflect the evolution of entities and their relationships in the universe which they model. Furthermore, the global users submit queries to access objects stored in heterogeneous databases through interfaces provided by the MDBS. Since the world is dynamic and changes over time, the global users may acquire different tasks over time as well. As a consequence, their information needs and object requirements may be altered.

Instead of being static, both local databases and global users should be considered dynamic in a multidatabase environment. Therefore, the interfaces supported by the MDBS should also be dynamic. To provide consistent and up-to-date information, the dynamic evolution of the databases and global users should be propagated to the global interfaces so that the global knowledge (repository of all local database and global user information, possibly involving the use of a dictionary, thesaurus, knowledge base, and/or an auxiliary database) can be corrected to reflect such evolution when occurring.

To capture the concept of objects in multiple databases as they evolve over time, the term "object" is used at different levels of granularity in this paper (e.g., databases, schemas, classes, and instances of a class are referred to as objects). We express objects in multidatabase context in a three dimensional view as shown in Figure 1. The space dimension represents the diverse preexisting and autonomous databases (possibly heterogeneous) in which objects may reside. Research and projects in multidatabases traditionally concentrated on integrating objects from different locations (space). However in our view, the semantics of time must also be considered for the evolution of objects in such context. We use the time axis to denote the discrete time interval in which objects exist in the different databases. Objects may evolve during any time chronon [3] in the time interval which the objects exist. By

[3]An instant is a time point on an underlying time axis, a time interval is the time between two instants, and a

incorporating the time dimension to the objects in multidatabases, the temporal object framework [29, 30, 31] is transferred into multidatabase context. Such extension provides a richer and more realistic view to address the objects in different pre-existing and possibly autonomous databases.

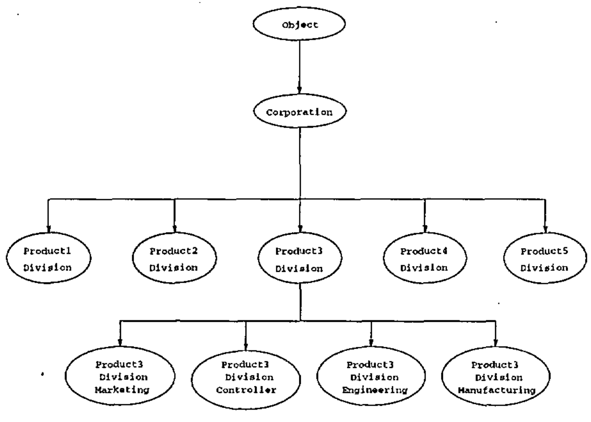

Object
Corporation
Product1 Division
Product2 Division
Product3 Division
Product4 Division
Product5 Division
Product3 Division Marketing
Product3 Division Controller
Product3 Division Engineering
Product3 Division Manufacturing

Figure 1: Three dimensional view of objects in MDBSs

Temporal objects in multidatabases include both space and time dimensions. The extension of time dimension allows us to specify and relate different time-varying aspects of the objects. We use the term *object state* to include the structure, operations (viz. methods), and constraints (e.g. integrity constraints and temporal constraints) defined on an object. An object composes of two parts: invariant object properties and variant object properties [29, 30, 31, 33]. Invariant object properties, denoted as *IOP*, are the time invariant properties associated with the object. Variant object properties, denoted as *VOP*, are the time variant properties associated with the object. In general, an object state contains both time-invariant and time-variant properties.

There are two significant problems in managing temporal objects in multidatabase context. The first problem has not been extensively addressed and concerns with the consistency of the inter-related objects (viz., objects which are related and dependent through integrity constraint [35]) which reside in multiple databases; the term *interdependent object* [35] is used to refer to such objects. When objects evolve over time, the MDBSs should posses the ability to sustain the consistency among the interdependent objects as much as possible, if not all.

Another important problem that must be addressed when considering the temporal objects in multidatabase environment is the maintenance and support of "object relativism" between local and global systems. We use the term "object relativism" to mean that the semantics, properties, and constraints of the shared objects are relative and consistent from local to global levels. As the temporal objects in the databases and global users evolve over time, the changes must be reflected in the the global interfaces which maintain the global knowledge (the repository of all the local database and global user information) so that the integrity of the objects is preserved in the global system. The problems of interdependent object consistency and local-global semantic relativism are depicted in Figure 2. The interdependent object consistency is local to local while the local-global semantic relativism is local to global.

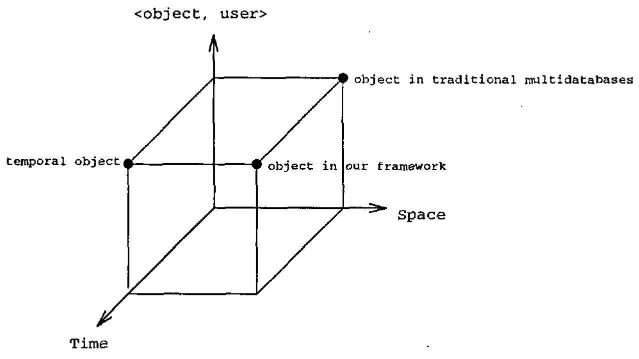


Figure 2: Interdependent Object Consistency vs. Local-Global Semantic Relativism

Some applications, such as certain GISs, are also sensitive to world changes, and up-to-date information is critical to their operations. Recently, an emergent class of database systems, called the *Active Rapidly Changing systems* (ARCS), manages databases which model a part of the universe where the state change rapidly [10]. Operating with poor and outdated information may result in the economic or even safety consequences to the users. Therefore, the integrity of a multidatabase system will be questionable unless its global interfaces reflect the world changes. In other words, the interfaces that provide services to global users to access data from local databases should be modified to respond to the evolution of objects in the local databases and global users.

chronon is a non-decomposable time interval of some fixed minimal duration [19].

In the rest of this section, we discuss the dynamic characteristics of objects in multidatabases addressed in our framework. The term "dynamic" is used in general to refer to the process of continuously changing in structure, meanings , behavior, or properties of objects in the multidatabase environment. We also introduce our concepts of time on objects in multidatabase domain. Special emphasis has been put on analyzing the semantic of time on the relations of objects in the context of local and global systems.

3.1 Dynamic Characteristics of A Multidatabase Environment

Dynamism in the context of multidatabases relates to the notion that a multidatabase system is composed of different databases managed by independent database management systems and different global users whose needs may change over time. In this section, we analyze the dynamic characteristics of objects in multidatabase environment. Particularly, we identify four dynamics of objects: network topology, spatial representation, temporal objects, and object constraint [40].

The *network topology* refers to the physical organization of the local databases which provide information and the global users who want to access information from these databases. The number of local databases participated in a MDBS may change over time [12]. This means that on the one hand, some existing databases may decide to terminate their participation in the system. On the other hand, new databases may desire to participate in the global information sharing as information providers, and they need to be added to the system. Similarly, the number of global users may change over time [12]. Some existing users may no longer require the services provided by the MDBS and decide to withdraw from the system. Also, some new users may request to be added to the system. Thus, the network topology of a MDBS may be dynamic.

The entities and their relationships in the universe may be modeled differently in different databases. Each database has its own representation, both in terms of structure and semantics, associated with the relevant objects in the universe in which it models. Therefore, the behavior of objects in the universe is dynamic with respect to databases (space). We use the term *spatial representation* to denote the different in structure, semantic, and behavior of the real-world objects in different databases at a given time chronon. In addition to the different representation of objects in different databases, each global user has her/his own set of requirements for the structure, semantic, and properties for the received objects, which may not be the same as those of the local databases. To ensure the interoperability of the objects, the differences between users and source databases must be recognized and handled. Normally information conversion and abstraction process is required. For example, in a financial application, the monetary unit of a source database might be in U.S. "dollar", but the monetary unit of a user might be in Japanese "yen".

As an example of *spatial representation* of object, the object "organization" in the universe can be represented in two databases (space) as shown in Figure 3 and 4.

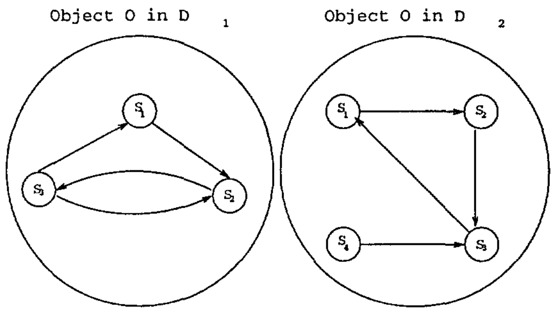


Figure 3: Functional form of object organization

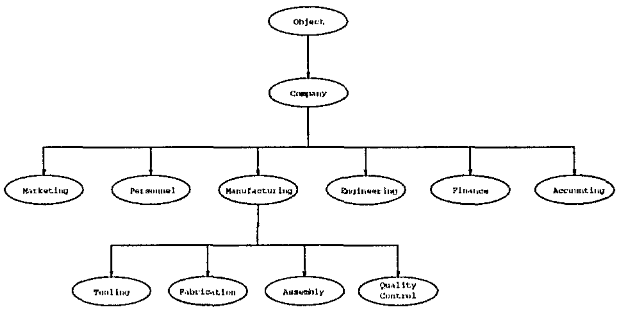


Figure 4: Product form of object organization

The object "company" shown in Figure 3 is used to model the entity "organization" in the universe. The "company" is subdivided according to the managerial functions, as shown as marketing, personnel, manufacturing, engineering, finance, and accounting. In Figure 4 the entity "organization" in the universe is modeled by object "corporation". However the "corporation" is subdivided according to the major products, as shown as product1, product2, product3, product4, and product5. For each of the products, it can be further subdivided using the managerial function. Both of the objects "company" and "corporation" model the same entity "organization" in the universe even though they may behave differently or have different representation schemas.

The *temporal objects* [29, 30, 31, 33] incorporate the notion of time to objects. The time-variant properties, behavior, and role of objects are considered. In a multidatabase context, objects in different local database may evolve over time to reflect the changes of their corresponding universal objects. As an illustration of the concept of the *temporal object*, let us consider a geographical example. In the midwest area, there is the threat of the New Madrid Fault to cause the "big" earthquake within the next several decades. Therefore, it is very important to monitor the geographical changes in this area. To simplify our example, let us assume that the observation for geographical changes has started at time t_i. At time t_j, a moderate earthquake has caused a piece of farmland, which may be represented in a database by a coordinate point (x, y), along the Mississippi River to become a part of the river. Then we say that during the time interval $< t_i, t_j >$, the location (x, y) has changed in its behavior, from a piece of land to be a part of the Mississippi River. It is obvious that either as a piece of farmland or a part of the Mississippi river, it is the same geographical location which behaves differently over time.

Object integrity constraints are derived from the local constraints and user constraints pertinent to the objects. Any changes in the objects may impact the set of constraints associated with the objects. Such constraints include the integrity constraints on objects which the databases require that they be enforced at the global level, temporal conditions, and global situation/action rules [38]. For example in a healthcare information system, with respect to the patients' privacy a user's ability to access patients records should be different. A physician should be able to access information concerning her/his patients more freely from a hospital database than an insurance company. Since it is inappropriate for a local system to keep track of global users in an ever changing environment, the enforcement of such constraints should be at the global level.

Based on the concept of spatial representation and temporal objects as we have described, we want to consider the consequences they have on the global interfaces. Unlike objects in a single database, objects in a multidatabase system have already existed in the local databases. The main functionality of multidatabase systems is to support interfaces for global users, so that they can access objects from these databases efficiently and effectively. In general, multidatabase systems maintain global knowledge which is the repository of all local database information. A part of this functionality includes correlating objects in one local database to the objects in another local database, so that if they model the same objects in a modeled universe, then the heterogeneities of information due to different space are hidden from the users, as in the global mapping knowledge shown in Figure 5.

Object $O_{(db_i, u_j)}$ denotes an object which models the concept u_j in the universe and is stored in database db_i. An arrow between any two objects in DB_1 and DB_2 indicate that the two objects are related, for they are modeling the same concept in the universe, and the relations between objects in the two databases are reflected in the global mapping knowledge.

To specify and relate different time-varying roles, behaviors, characteristics, and identities of objects in multidatabases, a framework to combine the space and time to objects is challenged. Such framework can essentially provide basic platforms to develop systems which are sensitive to changes as they happen in the real world. In the following section, we informally present a framework called a *temporal multidatabase object paradigm* which based on the dynamic representation and properties of objects in multidatabase environment as discussed in the above. The paradigm will be formalized in the section 4.

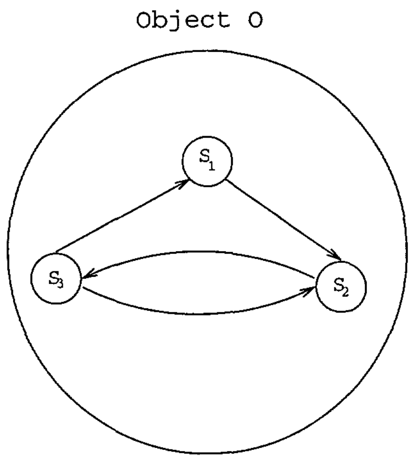


Figure 5: Correlation of objects in two database systems

3.2 A Temporal Multidatabase Object Paradigm

In our framework, we use the notions of *local object conception* and *global object unification* [29, 30, 31, 33, 40] to describe an object across its multiple representations in different databases (space). The *Local Object Conception* of objects, denoted by *LOC*, refers to the conceptual modeling of the universal objects in local databases, and the *Global Object Unification* of the objects, denoted as *GOU*, refers to a global interpretation or sense of the object in the universe. From the illustration in Figure 5, $O_{(db_1,u_1)}$ is the LOC of object u_1 in the local database db_1, $O_{(db_2,u_1)}$ is the LOC of object u_1 in the local database db_2, and U_1 in the global mapping knowledge is the GOU of the object u_1.

We can view GOU as a mechanism to correlate objects in different systems which represent the same concepts in the universe so that the problem of space-variant modeling of the universal objects in different databases can be hidden from the global users. In a way, we can view a GOU of objects as the aggregation of all the possible LOCs of the objects. As an example, the global object unification of the object "organization" could be defined as "a group of people who are united for some purpose," and its local object conceptions could be the objects "company" and "corporation" as shown in Figures 3 and 4.

To specify and relate different time-varying roles, behaviors, and identities of an object, the relations between objects in different databases must be dynamically altered over time as objects evolve. Therefore, the global knowledge should not only describe the time-invariant relations of objects but should also posses the ability to dynamically alter the relations it maintains to facilitate the evolution of objects in time. In our example, when $O_{(db_1,u_1)}$ has evolved from concept u_1 to concept u_2 in the universe, the relation maintained in U_1 between the object which previously was $O_{(db_1,u_1)}$ and the object $O_{(db_2,u_1)}$ is no longer valid and must be eliminated.

Using the notions of of LOC and GOU, we want to extend our abstract framework to bring together the notions of "temporal object modeling" and "multidatabases", viz., integrate the concepts of space and time for objects in multidatabase context. Our aim is to develop a model such that the anticipated dynamic changes in the behaviors, roles, characteristics, or even identities of objects in local databases and global users can be captured automatically in the global interfaces over time. We use the *Local Object Conception-to-Local Object Conception* (LOC-to-LOC) mapping, which we extend from the mappings introduced in [29, 30, 31, 33], to express the evolution of the objects in diverse databases.

LOC-to-LOC mapping: Objects may acquire new roles, behaviors, or possibly new identities over time to reflect the evolution of the concepts in the universe which they model. A LOC-to-LOC mapping is a relation that supports the temporal objects in multiple databases. It allows an object to change its characteristics, behavior, and role from one object state (or concept) into another object state. There are two types of LOC-to-LOC mapping: LOC_{d_i}-to-LOC_{d_i} and LOC_{d_i}-to-LOC_{d_j}. On the one hand, the LOC_{d_i}-to-LOC_{d_i} mapping allows intra-database object evolution where an object in a database may evolve from one state into another state within the same database. On the other hand, the LOC_{d_i}-to-LOC_{d_j} mapping allows inter-database object evolution where an object in a database may evolve into a state of an object in another database. As an example, the object $O_{(db_1,u_1)}$ in Figure 5 may evolve from concept u_1 to concept u_2 within the same database, and the object $O_{(db_1,u_3)}$ may evolve from concept u_3 in database DB_1 to concept u_k in database DB_2.

In general, a *LOC-to-LOC* mapping is a relation $\mathcal{R}_{loc-to-loc}$ whose domain contains the LOC of an object in a database at time t_k, and its range contains the LOC of the object at time t_l, for $t_k < t_l$. More formally, $\mathcal{R}_{loc-to-loc}$: $(d_i, \alpha, t_k) \rightarrow (d_j, \beta, t_l)$ where α and β denote objects such that $\alpha \neq \beta$, d_i and d_j denote local databases i and j and i may be equal to j, t_k and t_l denote time chronons such that $t_k < t_l$. To illustrate the LOC-to-LOC mapping, let us consider the geographical example which we have used to explain the notion of temporal roles of an object. We can express the evolution of the geographical location (x, y) in a database d_i which has evolved from a piece of farmland into a part of the Mississippi River in term of LOC-to-LOC mapping as following.

$$\mathcal{R}_{loc-to-loc} : (d_i, land_{(x,y)}, t_k) \rightarrow (d_i, river_{(x,y)}, t_l), \text{ for } t_k < t_l.$$

The LOC-to-LOC mapping supports interdependent objects in different databases. We want to extend our model to support the object relativism between local databases and global interfaces. To represent the relationships among the evolving temporal objects in local databases, the *Local Object Conception-to-Global Object Unification* (LOC-to-GOU) mapping is introduced to correlate these objects.

LOC-to-GOU mapping: A LOC-to-GOU is a relation $\mathcal{R}_{loc-to-gou}$ whose domain contains the LOC of an object, α, in the local database d_i, for $1 \leq i \leq n$ and n is the number of participated local databases, and its range contains the GOU of the object in global system G. More formally, $\mathcal{R}_{loc-to-gou} : (d_i, \alpha) \rightarrow (G, \beta)$.This relation maps different conceptions of α in local databases d_i, for $1 \leq i \leq n$, to a global object unification maintained by the global system G at any given time chronon.

GOU-to-GOU mapping: A GOU-to-GOU mapping is a relation $\mathcal{R}_{gou-to-gou}$ that supports the temporal objects in the global system. Similar to LOC-to-LOC mapping, it allows global objects to change its characteristics, behavior, and role from one object state (or concept) into another object state. Formally, $\mathcal{R}_{gou-to-gou} : (G, \alpha, t_k) \rightarrow (G, \beta, t_l)$. This relation allows an object α at time t_k in the global system G to evolve to another object β in the same global system G from time t_l. Such object evolution may permit α to assume a new identity.

GOU-to-LOC mapping: A GOU-to-LOC mapping is a relation $\mathcal{R}_{gou-to-loc}$ whose domain contains the GOU of an object, and its range contains the LOCs of the object in different databases. There are two purposes for the GOU-to-LOC mapping. First, the mapping is used to describe the information conversion and abstraction process which is normally used to resolve the structural and semantic differences of objects between local databases and global users. It maps a global object unification α in the global interfaces to a user's conception of the object so that objects can be abstracted and converted from the local databases to satisfy the object specification of the global users. Second, it may be used to support interdependent objects in different databases. In this situation, the local databases are not totally autonomous. The integrity constraints of the interdependent objects in different databases are explicitly specified in the global interfaces, and the global interfaces have the authority to enforce such constraints on the local databases. More formally, $\mathcal{R}_{gou-to-loc} : (G, \alpha) \rightarrow (d_i, \beta)$ where α denotes an object in the global system, β denotes an object in a database, d_i denotes a database.

To attempt to combine the concept of time with the concept of multidatabases to objects, it is necessary to incorporate to the functionalities of the global interfaces the ability to capture the new properties, behaviors, roles, or even identities of objects over time. The global knowledge of the local systems should be kept up-to-date so that it can provide the most current and valid information. To ensure that evolutionary changes are reflected in the global interfaces, there must be some active mechanism to forward the effect of the changes to the global system so that the knowledge maintained by the global interfaces can be updated. In our framework, when we anticipate evolutionary changes, the objects that might be involved in such changes are monitored (which is possible because we monitor events). The effect of these changes are propagated to the global system. To accommodate the flow of object evolution from local to global systems, we describe such evolution in terms of our mapping as follows.

1. (space) $\mathcal{R}_{loc-to-gou} : (d_i, \alpha) \rightarrow (G, \beta)$ and $\mathcal{R}_{loc-to-gou} : (d_i, \gamma) \rightarrow (G, \phi)$ at time t_k,
2. (time) When a monitored event occurs in MDBS at time t_l, for $t_k < t_l$, such that $\mathcal{R}_{loc-to-loc} : (d_i, \alpha, t_k) \rightarrow (d_j, \gamma, t_l)$, then $\mathcal{R}_{loc-to-gou} : (d_i, \alpha) \rightarrow (G, \phi)$ at the time chronon t_l where
   - α and γ denote objects n local databases (as well as global users) such that $\alpha \neq \gamma$,
   - β and ϕ denote objects in global interfaces such that $\beta \neq \phi$,
   - t_k and t_l denote time chronons such that $t_k < t_l$,
   - d_i and d_j denote local databases i and j for i may be equal to j,
   - G denotes global system.

Relation $\mathcal{R}_{loc-to-gou} : (d_i, \alpha) \rightarrow (G, \beta)$ in (space) correlates objects α in database d_i to its global object unification, maintained by the global system, when d_i initially participates in the MDBS at a given time chronon. A similar mapping is included in many multidatabase architectures; for example in some systems it is included as the "information mapping" component [34] while in other systems as "shared ontology" component which specifies the terminology mapping [36]. In our architecture, we extend this mapping to include the temporal modeling of objects so that the state of an object can evolve over time and is reflected in the global system.

Relation $\mathcal{R}_{loc-to-gou}$ in time allows the effects of object evolution to be forwarded from local databases (as well as global users) to the global interfaces. In our framework, the anticipated events which may cause the objects to evolve from one object state to another object state are monitored. If an event occurs in the system in such a way that an object evolves from one object state into another object state, such as from a piece of farmland into a part of the Mississippi River, the relation $\mathcal{R}_{loc-to-gou}$ in time reunifies the object to a new identity in the global interfaces. The relation $\mathcal{R}_{loc-to-loc}$ is used to express the changes in object, and the relation $\mathcal{R}_{loc-to-gou}$ propagates these changes from local databases (as well as the global users) to global interfaces so that knowledge maintained by the interface can be modified.

So far, we have presented our concepts and paradigm of objects in multidatabases with respect to space and time. In the next section, we will formalize the dynamic object model in multidatabases introduced in this section.

4 RIMM: Reactive Integration Multidatabase Model

A major aim of our research is to achieve an integration of the abstract concepts that are assumed to characterize temporal multidatabase objects. In order to achieve this goal, these abstract concepts are mapped to a simple Reactive Integration Multidatabase Model (RIMM), thus articulated as concrete concepts. This provides a specific context for our approach described in the section 3 and allows us to define a conceptual framework for the temporal evolability of multidatabase objects. The main purpose of our model is to provide a basic framework for "reactive" multidatabase models and systems. The model is based on a temporal distributed object (viz., temporal property and conditions) and the notion of event [19] in the real world.

In responding to operations and transactions that reflect changes in the real world, objects in multidatabases may change states. Objects can be characterized by their structure, sets of methods, and sets of constraints (e.g. semantic constraints, integrity constraints, and temporal conditions) [39]. We formally define an object state as the triple $(Struc, M, \Sigma)$ where

- Struc is the structure of the object.
- M is a set of operations defined on the object,

- Σ is a set of constraints associated with the object.

$Struc, M$, and Σ contain both IOP and VOP as defined in the section 3. Since the objects in the databases model the evolving real world entities, each object may be in the different states at the different time chronon. Thus over temporal element [4], an object may subsume more than one state. Let an object scheme represents the entire states of an object. Then an object scheme can be defined as $S = \{S_{t_i} = \{Struct_{t_i}, M_{t_i}, \Sigma_{t_i}\} | 1 \leq i \leq k\}$, where k is the maximum number of the object states in a database. For the sake of notation simplicity, hereon the subscript i implies t_i.

Figure 6 gives an example of object states. Object "O" in the figure has three states: S_1, S_2, and S_3. State S_1 of the object may transform to state S_2, state S_2 may transform to S_3, and state S_3 may transform to both S_1 and S_2.

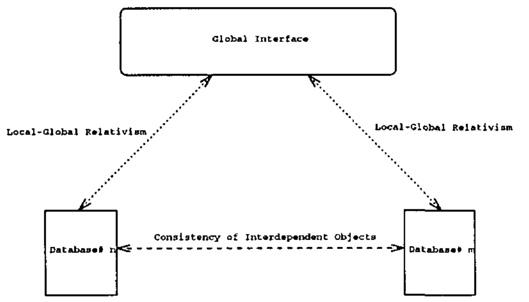


Figure 6: An object "O" with three states: S_1, S_2, and S_3

Furthermore, let $D = \{D_1, D_2, ..., D_n\}$ denote the set of "n" databases participated in a global information-sharing multidatabase domain. We define an object scheme S_{D_j} in database D_j, $1 \leq j \leq n$, to be $S_{D_j} = \{S_{(i,D_j)} = \{Struct_{(i,D_j)}, M_{(i,D_j)}, \Sigma_{(i,D_j)}\} | 1 \leq i \leq k_j\}$. Thus, with respect to the global system, the object scheme is the set of all the local object schemes. We define the global object scheme, denoted as S_G, as $S_G = \{S_{D_1}, S_{D_2}, ..., S_{D_n}\}$.

An object instance "O" of S_G, denoted Global Object Unification (GOU) [5], is a tuple $(O_{D_1}, O_{D_2}, O_{D_n})$ of objects on $S_{(i,D_j)}$. We denote the set of instances by $\mathcal{S}(S_G)$. Components of the tuple are called "Local Object Conception" (LOC) of O. If LOCs are disjoint (viz. no duplications between any two LOCs), $O_G = O_{D_1} \cup O_{D_2} \cup ... \cup O_{D_n}$.

Figure 7 pictures an example of the object states for an object "O". The object "O" has three and four states in databases D_1 and D_2 respectively, and these states are represented in our model as follows.

$S_{D_1} = \{S_{(1,D_1)}, S_{(2,D_1)}, S_{(3,D_1)}\}$
$S_{D_2} = \{S_{(1,D_2)}, S_{(2,D_2)}, S_{(3,D_2)}, S_{(4,D_2)}\}$
$S_G = \{S_{(1,D_1)}, S_{(2,D_1)}, S_{(3,D_1)}, S_{(1,D_2)}, S_{(2,D_2)}, S_{(3,D_2)}, S_{(4,D_2)}\}$

An object state $S_{(i,D_k)}$ is directly reachable from $S_{(i',D_l)}$ if the transition between the two states is legal. Object state transformations are specified in the transition system. We define a transition system to be a pair $TS = \{S_{(i,D_j)}, \{\xrightarrow{r} | r \in \mathcal{R}\}\}$ where $S_{(i,D_j)} \in S_G$, and $\mathcal{R}$ is a non-empty set of transition relations which models state transformation. TS $\subseteq S_G$ X S_G for state transition where $S_{(i',D_l)}$ is directly reachable from $S_{(i,D_j)}$ if $S_{(i,D_j)} \xrightarrow{r} S_{(i',D_l)}$.

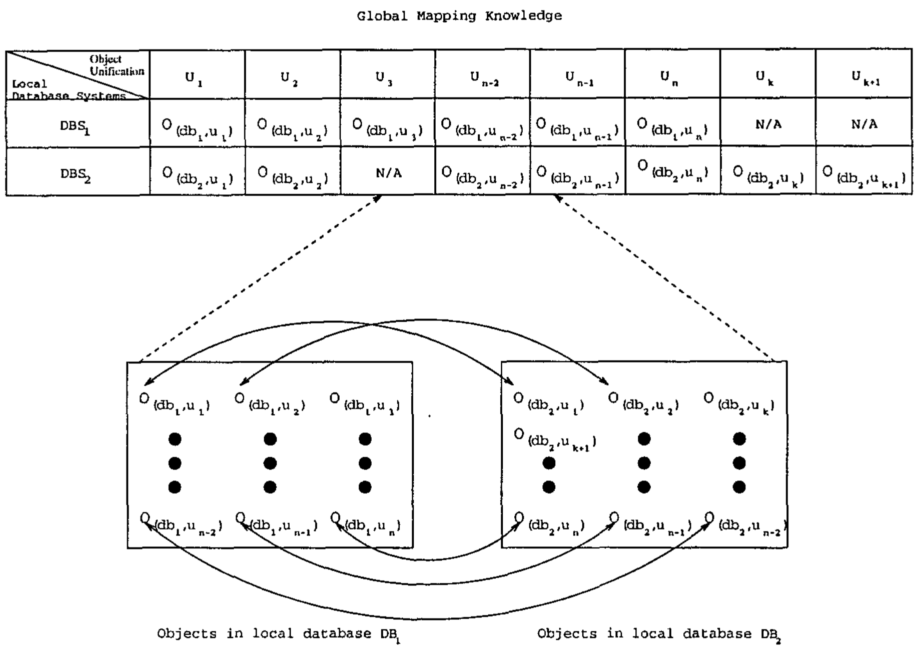
In addition, an object instance "O" from $\mathcal{S}(S_G)$ can be modified either using operation $m \in M_{(i,D_j)}$ which is applied to the LOC of O in D_j, denoted $LOC_{D_j}(O)$), or conditional transformation $TR = \{(\alpha, T)\}$. We use TR to model state change. A conditional transformation changes an object O only if $\models_O \alpha$ and $\models_{T(O)} \Sigma$. There are four types of transformations in TR; $T = \{\mathcal{R}_{loc-to-loc}, \mathcal{R}_{loc-to-gou}, \mathcal{R}_{gou-to-loc}, \mathcal{R}_{gou-to-gou}\}$.

- $\mathcal{R}_{loc-to-loc}$ is a set of operations which allows an object to transform from one object state to another object state, either within the same database or between two databases as described in the section 3.2.

- $\mathcal{R}_{loc-to-gou}$ supports the object relativism between the local and global systems. Since $S_G = \{S_{D_1}, S_{D_2}, ..., S_{D_n}\}$, the global system usually maintains the global knowledge which is a repository of all the local database information. $\mathcal{F}_{loc-to-gou}$ is a set of operations which allows an object state in a local database to modify the global knowledge information.

[4] A temporal element is a finite union of n-dimensional time interval [19]

[5] Definitions of GOU and LOC are as defined in the section 3.2.



Global Mapping Knowledge

Local Database Systems \ Object Unification	U_1	U_2	U_3	U_{n-2}	U_{n-1}	U_n	U_k	U_{k+1}
DBS_1	$O_{(db_1,u_1)}$	$O_{(db_1,u_2)}$	$O_{(db_1,u_3)}$	$O_{(db_1,u_{n-2})}$	$O_{(db_1,u_{n-1})}$	$O_{(db_1,u_n)}$	N/A	N/A
DBS_2	$O_{(db_2,u_1)}$	$O_{(db_2,u_2)}$	N/A	$O_{(db_2,u_{n-2})}$	$O_{(db_2,u_{n-1})}$	$O_{(db_2,u_n)}$	$O_{(db_2,u_k)}$	$O_{(db_2,u_{k+1})}$

Figure 7: States of an object "O" in databases D_1 and D_2

– $\mathcal{R}_{gou-to-loc}$ also supports object relativism between local and global systems. It is a set of operations which allows an object state in global system to transform an object instance in a local database from one state to another state.

– $\mathcal{R}_{gou-to-gou}$ is a set of operations which allows object transformation from one object state to the another object state within the global system.

Let $\mathcal{T}$ denote a set of time chronons.

$\mathcal{R}_{loc-to-loc}$: S X D X $\mathcal{T} \rightarrow$ S X D X $\mathcal{T}$,
$\mathcal{R}_{loc-to-gou}$: S X D $\rightarrow$ S X G,
$\mathcal{R}_{gou-to-loc}$: S X G $\rightarrow$ S X D
$\mathcal{R}_{gou-to-gou}$: S X G X $\mathcal{T} \rightarrow$ S X G X $\mathcal{T}$,

where $\mathcal{R}_{loc-to-gou}$ and $\mathcal{R}_{gou-to-loc}$ are inverse relations.

A dynamic object scheme can be defined as an object scheme, a set of object transformation, and a set of dynamic properties. Formally,

$DynObj = (S, TR, \Sigma_\triangle)$,
$DynObj_{S_G} = (S_G, TR, \Sigma_\triangle)$

where the dynamic properties $\Sigma_\triangle$ includes both the transition properties or temporal conditions. For any two instances $O_G = (O_{D_1}, O_{D_2}, ..., O_{D_n})$ and $O'_G = (O'_{D_1}, O'_{D_2}, ..., O'_{D_n})$ in $\mathcal{S}(S_G)$ where O'_G is derived by applying transformation $\tau \in TS$ to O_G, a transition property (α, β) is valid for (O_G, O'_G) if $\forall j$, $1 \leq j \leq n$ from $\models_{O_{D_j}} \alpha$ it follows that $\models_{O'_{D_j}} \beta$.

Our main objective in this paper is to develop a formal reactive integration model that contains the expressiveness to represent the temporal changes, temporal conditions, and events in the real world. In the section 3, we have defined the global objects to consist of LOCs. To unify the LOCs in diverse databases, we define the "Reactive Integration" of LOCs by a transition system, a set of local databases, a global system, a set of selectors, a composition operator, and a set of events. Formally,

$Integration_{Reactive} = (TS, D, G, \mathcal{H}, \Phi, Events)$
where

– TS, D, and G are as defined in this section.

– $\mathcal{H}$ is a set of selectors. An instance $O_G = (O_{D_1}, O_{D_2}, ..., O_{D_n})$ satisfies a set of selectors $\mathcal{H} = \{sel_1, ..., sel_n\}$ if $\forall j, 1 \leq j \leq n, O_{D_j} = sel_j(O_{D_1}, O_{D_2}, ..., O_{D_n})$ is valid.

– Φ is a conditional unification $(\gamma, \cup)$. Whenever LOCs satisfy the condition in γ, the Φ can be applied and the LOCs are unified as the result.

– Objects dynamics depend on events. An event is an occurrence of interest that happens at a point in time. Events are used to specify state changes of databases and may cause several state transformation sequences to be valid. In general, an event can either be a basic or composite event. A basic event is a pair <event occurrence, time instant>. The event occurrence is represented by some symbol e and is mapped to a time instant t on the system clock. The basic event (e, t) is said to occur at time t. A composite event can be created from basic or other composite events through the used of closed algebra [32].

Based on the formal reactive integration model presented in this section, the multidatabase architecture that incorporates event driven productions rules can be realized. The architecture will be capable to automatically react to anticipated changes of the local databases and the global users and dynamically reconfigure the global interfaces, so as to support interoperability over time.

5 Final Remarks

This paper focuses on combining the concepts of space and time in a multidatabase domain thus providing a dynamic framework to actively support interoperability of multidatabases. Specifically it identifies the kind of dynamism that local systems and global users exhibit over time and space, resulting in an expressive representation that can specify complex "real-world" situation. To this end, we extend the temporal concepts of objects[29, 30, 31, 33] to a multidatabase environment and study its consequences in such a context. Based on these abstract principles and foundation, we have developed a formal model, called Reactive Integration Multidatabase Model (RIMM), that contains the expressiveness to represent the temporal distributed objects.

As a realization of the RIMM model, we are currently developing an architecture, based on the mediators [12, 36, 41, 42, 43], that incorporates the event driven production rules to automatically react to anticipated changes of local databases and global users and dynamically reconfigure the global interfaces, so as to support interoperability over time. We argue that with respect to managing dynamic conflicts in a multidatabase environment, any chosen architecture should include active database rules, so that the management process can be automated as much as possible. Our future work will concentrate on the implementation of the system, for validation and simulation of our work. In addition, we are investigating how rules can be defined and generated and how to develop optimization techniques to improve performance of event detection therefore better providing support for geographical and real-time applications.

6 Acknowledgements

The authors are sincerely thankful to the anonymous referees for making several helpful comments and suggestions on earlier versions of the paper, and to Angelica Kokkinaki for the valuable discussions.

This work is partially supported by NSF/LaSER under grant number EHR-9108765, by LEQSF under grant number 94-RD-A-39, and by NASA under grant number NAG 5-2842.

References

[1] Y. Arens and C. A. Knoblock. "Planning and Reformulating Queries for Semantically-Modeled Multidatabase Systems", in *Proceedings of the 1st International Conference on Information and Knowledge Management*, Baltimore, Md., pp. 92-101, November 1992.

[2] A. P. Buchmana, et al. "A Transaction Model for Active Distributed Object Systems", *Advanced Transaction Models for New Applications*, A. Elmagarmid (ed.), Morgan-Kaufmann, 1992.

[3] C. Batini, M. Lenzerini, and S. B. Navathe. "A Comparative Analysis of Methodologies for Database Schema Integration", *ACM Computing Surveys*, Vol. 18, No. 4, pp. 323-364, December 1986.

[4] Y. Breitbart. "Multidatabase Interoperability", *SIGMOD Record*, Vol. 19, No. 3, pp. 53-60, September 1990.

[5] M. W. Bright, A. R. Hurson, and S. H. Pakzad. "A Taxonomy and Current Issues in

Multidatabase Systems", *IEEE Computer*, pp. 50-59, March 1992.

[6] O. A. Bukhres, A. K. Elmagarmid, and J. G. Mullen. "Object-Oriented Multidatabases: Systems and Research Overview", in *Proceedings of the ISMM International Conference CIKM-92*, Baltimore, MD, pp. 27-34, November 8-11, 1992.

[7] S. Ceri and G. Pelagatti. *Distributed Databases: Principles and Systems*, McGraw-hill, New York, 1984.

[8] S. Ceri and J. Widom. "Managing Semantic Heterogeneity with Production Rules and Persistent Queues", in *Proceedings of the 19th VLDB Conference*, Ireland, pp. 108-119, August 1993.

[9] C. Collet, M. N. Huhns, and W, M. Shen. "Resource Integration Using a Large Knowledge Base in Carnot", *IEEE Computer*, Vol. 24, No. 12, pp. 55-63, 1991.

[10] A. Datta. "Research Issues in Databases for ARCS: Active Rapidly Changing Data Systems", *SIGMOD Record*, Vol. 23, No. 3, pp. 8-13, Sep. 1994.

[11] U. Dayal and H. Y. Hwang. "View Definition and Generalization for Database Integration in a Multidatabase System", *IEEE Transaction on Software Engineering*, Vol. SE-10, No. 11, November 1984.

[12] C. H. Goh, S. E. Madnick, and M. D. Siegel. "Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment", *Submitted for publication.*

[13] A. R. Hurson, M. W. Bright, S. H. Pakzad, "Multidatabase Systems: An Advanced Solution for Global Information Sharing", *IEEE computer Society Press*, 1994.

[14] D. Heimbigner and D. McLeod. "A Federated Architecture for Information Management", *ACM Trans. Off. Inf. Syst.*, Vol. 3, No. 3, pp. 253-278, July 1985.

[15] D. K. Hsiao and M. N. Kamel. "Heterogeneous Databases: Proliferations, Issues, and Solution", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, No. 1, pp. 45-62, March 1989.

[16] D. K. Hsiao. "Federated Databases and Systems: Part I-A Tutorial on Their Data Sharing", *VLDB Journal*, Vol. 1, No. 1, pp. 126-179, July 1992.

[17] D. K. Hsiao. "Federated Databases and Systems: Part II-A Tutorial on Their Resource Consolidation", *VLDB Journal*, Vol. 1, No. 2, pp. 285-310, October 1992.

[18] S. Heiler and M. Siegel. "Heterogeneous Information Systems: Understanding Integration", in *Proceedings of the First International Workshop on Interoperability in Multidatabase systems*, Japan, pp. 14-21, 1991.

[19] C. Jensen, et al. "A Consensus Glossary of Temporal Database Concepts", *SIGMOD RECORD*, Vol. 23, No. 1, pp. 52-64, 1994.

[20] W. Kim and J. Seo. "Classifying Schematic and Data Heterogeneity in Multidatabase Systems", *IEEE Computer*, pp. 12-18, December 1991.

[21] W. Kim. *Modern Database Systems: The Object Model, Interoperability, and Beyond*, ACM Press, New York, 1995.

[22] M. Hsu, R. Ladin, and D. McCarthy. "An Execution Model for Active Database Management Systems", in *Proceedings of the 3rd International Conference on Data and Knowledge Bases*, June 1988.

[23] W. Litwin and A. Abdellatif, "Multidatabase Interoperability", *IEEE Computer*, Vol. 19, No. 12, pp. 10-18, December 1986.

[24] W. Litwin. "From Database Systems to Multidatabases Systems: Why and How", *Proc. Sixth British National Conference on Databases*, W. A. Gray, ed., British Computer Society, Cambridge Univ. Press, pp. 161-188, 1988.

[25] W. Litwin, L. Mark, and N. Roussopoulos. "Interoperability of Multiple Autonomous Databases", *ACM Computing Survey*, Vol. 22, No. 3, pp. 267-293, September 1990.

[26] A. Motro. "Superviews: Virtual Integration of Multiple Databases", *IEEE Transaction on Software Engineering*, Vol. SE-13, No. 7, pp. 785-798, July 1987.

[27] D. R. McCarthy and U. Dayal. "The Architecture of An Active Data Base Management System", *SIGMOD Record*, Vol 18, No. 2, pp. 215-224, June 1989.

[28] M. Papazoglou. "On the Duality of Distributed Database and Distributed AI Systems" in *Proceedings of the Second International Conference on Information and Knowledge Management*, Washington, D.C., USA, pp. 1 -10, Nov. 1993.

[29] N. Pissinou and K. Makki. "A Unified Model and Methodology for Temporal Object Databases", *The International Journal on Intelligent and Cooperative Information Systems*, Vol. 2, No. 2, pp. 201-223, 1993.

[30] N. Pissinou and K. Makki. "A Coherent Architecture for a Temporal Object Database Management System", *The International Journal of Systems and Software*, Vol. 27, No. 3, 1994.

[31] N. Pissinou and K. Makki. "Separating Semantics from Representation in a Temporal Object Database Domain", *The Journal of Computer Information Systems*, Vol. 34, No. 3, 1994.

[32] N. Pissinou, R. T. Snodgrass, R. Elmasri, I. S. Mumick, M. T. Ozsu, B. Pernici, A. Segev, B. Theodoulidis, and U. Dayal. "Towards an Infrastructure for Temporal Databases: Report of an Invitational ARPA/NSF Workshop", *SIGMOD Record*, Vol. 23, No. 1, pp. 35-51, March 1994.

[33] N. Pissinou and K. Makki. "A Representation of Temporal Object Roles in Object Oriented Databases", *Proceeding of the International Workshop on Temporal Representation and Reasoning*, Goodwin S. d. and Hamilton J. H. (editors) Florida, May 1994.

[34] A. P. Sheth and J. A. Larson. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys*, Vo. 22, No. 3, pp. 183-236, September 1990.

[35] A. P. Sheth, M. Rusinkiewica, and G. Karabatic. "Using Polytransaction to Manage Interdependent Data", in *Database Transaction Models for Advanced Applications*, edited by A. Elmagarmid, Mogan Kaufmann Publishers, San Mateo, California, pp. 555-581, 1992.

[36] E. Sciore, M. Siegel, and A. Rosenthal. "Using Semantic Values to Facilitate Interoperability among Heterogeneous Information Systems", *To appear in ACM Transactions on Database Systems*, June 1994.

[37] S. Spaccapietra and C. Parent. "View Integration: A Step Forward in Solving Structural Conflicts", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 2, pp. 258-274, April 1994.

[38] A. K. Tanaka. "On Conceptual Design of Active Databases", *Ph.D. Thesis*, Department of Computer Science, Georgia Institute of Technology, December 1992.

[39] B. Thalheim. "State-Conditioned Semantics in Databases", in *Proceedings of the 13th International Conference on the Entity-Relationship Approach*, Manchester, UK, December 1994.

[40] K. Vanapipat, N. Pissinou, and V. Raghavan. "A Dynamic Framework to Actively Support Interoperability in Multidatabase Systems", Accepted for publication in *Proceedings of the Fifth International Workshop on Research Issues in Data Engineering*, Taiwan, 1995.

[41] G. Wiederhold. "Mediators in the Architecture of Future Information Systems", *IEEE Computer* pp. 38-49, March 1992.

[42] G. Wiederhold. "Intelligent Integration of Diverse Information", in *Proceedings of the ISMM International Conference CIKM-92*, Baltimore, MD, pp. 1-7, November 1992.

[43] G. Wiederhold. "Intelligent Integration of Information", in *Proceedings of the ACM SIGMOD Conference*, pp. 434-437, May 1993.

Statistical Usage Testing for Software Reliability Control

Per Runeson
Q-Labs AB, IDEON Research Park,
S-223 70 Lund, Sweden
AND
Claes Wohlin
Dept. of Communication Systems,
Lund Institute of Technology, Lund University,
Box 118, S-221 00 Lund, Sweden

Keywords: statistical usage testing, software reliability, usage profile, operational profile, statistical quality control

Edited by: Jerzy R. Nawrocki
Received: June 21, 1994 **Revised:** January 19, 1995 **Accepted:** February 15, 1995

Software reliability is a frequently used term, but very seldom the reliability is under control during a software development project. This paper presents a method, Statistical Usage Testing (SUT), which gives the possibility to estimate and predict, and hence control software reliability. SUT is the reliability certification method described as a part of Cleanroom software engineering. The main objective of SUT is to certify the software reliability and to find the faults with high influence on reliability. SUT provides statistically based stopping rules during test as well as an effective use of test resources, which is shown by practical application of this and similar methods. This paper presents the basic ideas behind SUT and briefly discusses the theoretical basis as well as the application of the method.

1 Introduction

The software development community is not in control of the software reliability. This can be stated based on a quote from Tom DeMarco, [3]: "You can't control what you can't measure". It is not possible based on traditional development techniques to actually measure the software reliability hence the reliability is out of control.

Software reliability engineering is currently a fast growing area. Therefore the situation is not hopeless; the techniques are becoming available to control the reliability. The software process will become more and more controlled which means that methods to estimate, predict and certify the fault content and reliability will be introduced. This is the only way towards managing the process of actually engineering reliable software, instead of crafting unreliable software.

The testing techniques normally applied are aimed at finding faults. This is a negative point of view since it implicitly accepts that errors are made and that the testers have to remove them. This reasoning may be philosophical, but it is believed to be one of the key issues in controlling software reliability. One of the most important aspects in the development is the motivation and belief in being able to do something, in this case develop software with few or no defects. Therefore it is essential to provide techniques so that the software developers can believe in developing zero-defect software, which also includes methods to certify the actual reliability level.

Cleanroom Software Engineering [13], [4], [6] and [12] emphasizes the intellectual control in software development. Cleanroom is a collection of several sound management and engineering techniques, in particular it is emphasized that it is possible to develop nearly zero-defect software. One of the engineering techniques emphasized in Cleanroom is Statistical Usage Testing, which is a method for statistical control of the software reliability during the system or acceptance testing phase. The objective of this paper is to propose a method for Statistical Usage Testing and to illustrate its use. The requirement on the testing phase, when applying usage testing, is that it resembles the operational phase to be able to apply the techniques to actually remove failures most critical for the user and certify a particular reliability level. It may be difficult to forecast the exact usage, but it is important to try to understand how the software probably will be used. It is often, at least, possible to identify usage classes, and the exact probabilities are not that critical. One possibility is to certify the software for one or several scenarios, which then can be used in negotiations between developers and procurers.

Statistical Usage Testing and its opportunities to promote statistical control of the software reliability are discussed in this paper. The usage testing technique provides a basis to certify a reliability requirement, see section 2. The principles behind Statistical Usage Testing are discussed in section 3, while the techniques within usage testing are further described in section 4, i.e. usage specification and reliability certification. In section 5 a minor example is presented to illustrate the usage testing technique described. Section 6 presents some practical experiences with usage testing techniques. Finally in section 7 some conclusions are presented.

2 Reliability requirement

2.1 Requirements specification

The requirements specification contains normally both functional requirements and quality requirements; in particular reliability or availability requirements are put into the specification. The fulfilment of the functional requirements is evaluated through using the functions specified in the requirements, but the other requirements ought to be fulfilled as well. The methods for reliability certification have, however, not been available or the available techniques have not been applied. This must change, either it is no use formulating reliability requirements or methods to evaluate and certify the reliability must be applied.

Statistical Usage Testing is a method to actually certify the reliability requirement. This type of method must be applied, since it is not possible to keep applying traditional systematic testing techniques and then see the system fail in operation. This is a result of not certifying the reliability requirement. The society can not afford software system failures, neither in safety critical systems nor in other cost intensive systems. For some systems, certification of the required level is not possible, in these cases certification must be combined with correctness proofs of the software. Usage testing is thus not the solution to obtain high quality software, but it enables certification of a reliability level.

The reliability requirement aims at the reliability as perceived by the users when the software system goes into operation. Therefore usage testing must be applied, since reliability is not only the number of faults but also the actual location of them compared with usage of the software system.

The method being presented allows for certification of the reliability requirement, which means that we will be in control of the reliability before releasing the software product instead of being surprised as the system fails in operation. In particular, it provides an opportunity to formulate a stopping criterion for the testing, where the criterion is based on the fulfilment of the requirement.

2.2 Validity of certification

Usage testing as a means for reliability demonstrations is well-known and its superiority compared to coverage testing is discussed in [15], [4] and [2]. An argument often raised concerning usage testing is: it is not possible to determine the usage profile, therefore usage testing is not applicable. This is, however, not completely true. It will probably be difficult to determine the usage profile exactly, but it ought still to be possible to determine usage probabilities relative to each other. It is often well known which functions will be used most frequently in a software system. Thus by making a clever guess or an estimate of the probabilities this method will still achieve preferential results over coverage testing. Successful applications of a usage profile based approach have been presented in [11] and [1].

A simple example will show the gain in applying usage testing, even when the usage profile, estimated during testing, is not the profile observed as the system is put into operation.

Let us assume that we have a system consisting of three modules as illustrated in figure 1.

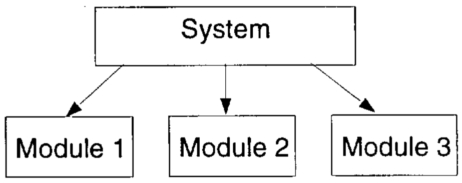


Figure 1: System architecture.

For simplicity it is assumed that the MTBF values for the three modules are all equal to 100 when the system testing begins. All times are given in some undefined time unit. The system test can be performed either as a coverage test or a usage test. In coverage testing the modules are tested equally, see table 1, and the MTBF values for each module improve identically, e.g. to 1000, see table 2.

It is necessary to use a particular usage profile to be able to perform a usage test. The profile is identified from experience and knowledge of the probable usage of the system being developed. Test cases are generated according to this usage profile, see table 1. Usage testing gives different MTBF values for the modules. In [2] and [4], it is shown in their particular case that based on a study of a number of projects that usage testing improved the perceived reliability during operation 21 times greater than that using coverage testing. Thus the MTBF for an arbitrary use in this example is assumed to be 21 000 (see table 3), i.e. 21 * 1000, where 1000 is the MTBF for an arbitrary use based on coverage testing. The MTBF values for the different modules are obtained based on the MTBF for an arbitrary use, and by assuming that the raise in MTBF is proportional to the probability of using a specific module. This reasoning leads to the figures presented in table 2.

The actual values are not particular interesting, but emphasis must be placed on the result after the system is put into operation and the increase obtained in perceived reliability using usage testing. An actual usage profile is obtained as the software is put into operation. The usage profile actually being experienced during operation may differ from the one applied during usage testing. Two examples are shown in table 1. The first example (Actual operation 1) shows a minor change or error in the usage profile, whilst the second profile (Actual operation 2) is an example of a rather large difference between the profile used during testing and the profile in actual operation.

The actual usage gives the perceived MTBF. The two testing techniques give different MTBF values in the two operation cases, see table 3. The values in table 3 are calculated by weighting the MTBF values for the three modules. The case "after usage testing (major difference)" is used to illustrate the calculations: MTBF = 0.1 * 121 + 0.15 * 314 + 0.75 * 21 230 = 15 982.

The MTBF estimates from testing are shown in the upper part of table 3. In the lower part of the table the perceived MTBF values during operation are shown.

It can be concluded that usage testing gives an improved MTBF during operation. Usage testing is of course dependent on the accuracy of the usage profile. Even if the profile is erroneous, the perceived MTBF is an improvement on that obtained with coverage testing, provided the probabilities for usage are in the correct order. In this particular example the probability of usage of the modules (from highest to lowest) are ranked 3, 2 and finally module 1. The MTBF is, however, overestimated during usage testing if the profile is erroneous, but still the result is better than that of applying coverage testing.

The overall conclusion is that usage testing improves the probability (as opposed to coverage testing) of locating faults that influence the reliability during operation. This conclusion is also supported by other sources including [2] and [4]. The example has also shown that even if the usage profile is erroneous during testing, the MTBF during operation will be higher than that obtained with coverage testing or other black box testing approaches.

3 Statistical Usage Testing

3.1 Cleanroom

Statistical Usage Testing (SUT) is the certification part of Cleanroom Software Engineering. Cleanroom is a methodology which consists of a set of software engineering and management principles as well as practices according to which software can be developed with very high quality and productivity [13], [14], [4], [12] and [5]. The methodology has proven to be very successful when

	Module 1	Module 2	Module 3
Coverage testing	1/3	1/3	1/3
Usage testing	0.001	0.01	0.989
Actual operation 1	0.002	0.05	0.948
Actual operation 2	0.10	0.15	0.75

Table 1: Probabilities for using the three different modules.

	Module 1	Module 2	Module 3
Before test	100	100	100
After coverage testing	1000	1000	1000
After usage testing	121	314	21230

Table 2: MTBF values per module.

applied in software development companies in Europe as well as in the USA [23], [19] and [24].

Cleanroom aims at development of almost zero-defect software with measurable reliability. The basic idea is to do things right from the beginning instead of first introducing and then correcting errors. Management and engineering techniques in Cleanroom are:

– The software is developed by small teams (3-5 people) with clearly defined responsibilities. There are three types of teams, specification, development and certification teams. The teams are jointly responsible for the produced results.

– Very much emphasis is put on rigourous specifications which are the basis for the development.

– The development is done in increments, each of which is executable. By partitioning the software into increments each increment may be handled by different teams and developed in parallel. Furthermore the increments are small enough to be held under intellectual control.

– The software is developed in small steps from specification to design according to a step by step algorithm. Each step is a refinement of the prior one. For each step more details are added and finally it ends up in executable code.

– Each of the development steps is rigourously verified towards the previous steps. The verification is mainly performed by reviews, supported by a theoretically based method called "functional verification".

– Traditional testing is replaced by certification of the software reliability by Statistical Usage Testing.

Most of the techniques are well-known but the combination of these and the management attitudes have given encouraging results concerning software quality as well as productivity and development time control. In this paper we concentrate on the certification part of Cleanroom, Statistical Usage Testing.

3.2 Usage based testing

Traditional testing is often concerned with the technical details in the implementation, for example branch coverage, path coverage and boundary-value testing, [18]. SUT on the contrary takes the view of the end user. The focus is not to test how the software is implemented, but how it fulfils its intended purpose from the users' perspective. SUT is hence a black box testing technique taking the actual operational behaviour into account. It treats the software as being a black box and is only concerned with the interfaces to the users.

SUT has two main objectives:

– To find the faults which have most influence on the reliability from the users' perspective.

ESTIMATION FROM TEST	MTBF
Coverage testing	1000
Usage testing	21000
PERCEIVED IN OPERATION	
After coverage testing (independent of actual usage profile)	1000
After usage testing (minor change)	20142
After usage testing (major difference)	15982

Table 3: MTBF estimates from test and operation environments.

- To produce data which makes it possible to certify and predict the software reliability and thus to know when to stop testing and to accept the product.

The latter implies that a usage profile is needed as it is not possible to certify and predict reliability in operation from other black box testing techniques.

3.3 Cost effectiveness

Studies show that usage based testing is an efficient way to find the faults which have most impact on the reliability [2]. The referenced study shows a gain with a factor 20. From seven software development projects at IBM it is concluded that 1.6% percent of the faults caused 58% percent of the failures during operation, while 61% percent of the faults caused only 2.8% percent of the failures. Thus it is more efficient to remove the 1.6% of the faults.

Software reliability depends not only on the number of faults in the software, but also on how the software is used. A fault in a part of the software which is frequently used has larger impact on the reliability than a fault in a less frequently executed part.

As the study by Adams shows, the most efficient way to improve software reliability is to remove the faults causing most of the failures, and not those which occur very seldom. In SUT test cases are selected to test according to the operational usage and are hence effective in order to find the faults which affect software reliability.

—subsectionSoftware reliability certification

The other objective of SUT is reliability certification, i.e. getting a reliability measure corresponding to the intended operational usage. To certify the software reliability, there is a need for a reliability model, which based on failure data from testing can estimate and predict the software reliability.

Most reliability growth models which can be used for reliability certification and prediction have a common prerequisite: usage based testing [7], [10] and [15]. This prerequisite has been overlooked during the years, but has come into focus during the last years [17] and [21].

3.4 Software acceptance

The software reliability measure obtained in usage based testing can be used as a criterion for software acceptance as well as a stopping rule for the testing.

The contract between a supplier and a purchaser often includes a software reliability requirement to be fulfilled at delivery. Neither the supplier nor the purchaser however can prove that the requirement is fulfilled or not. SUT is a possibility for both parts to get objective measures which may be used for judgement about the requirement fulfilment.

From the supplier's side a question of interest is when to stop testing. Large parts of a software development project costs are spent on testing. There is a need for saving testing costs. However it can cost money for a supplier to deliver bad products as well in terms of damages or bad reputation. This emphasizes a need for controlling software reliability by using reliability measures as stopping criteria for software testing, which are provided by SUT.

4 SUT models and methods

When applying SUT two kinds of models are needed, a model to specify the usage and a reliability model. In section 4.1 the usage specification is presented while the reliability models are treated in section 4.2. A method describing how to use the models during Statistical Usage Testing is presented in section 4.3.

4.1 Usage specification

The usage specification is a model which describes how the software is intended to be used during operation. Different types of models have been presented in the literature:

- Tree-structure models, which assign probabilities to sequences of events [17].
- Markov based models, which can specify more complex usage and model single events [25] and [21].

The primary purpose of a usage specification is to describe the usage to get a basis for how to select test cases for the usage based testing. It can however be used for analysis of the intended software usage as well, to plan the software development. Frequently used parts can be developed in earlier increments and thus be certified with higher confidence. Development and certification of increments are further discussed in [26].

In this paper the State Hierarchy (SHY) usage specification is briefly presented [21]. It consists of a usage model, which is the structural part, and a usage profile, which is the statistical part.

4.1.1 Usage model

The SHY usage model is a hierarchical Markov chain which copes with specification of the usage of large multi-user software systems. The basic concept of the SHY model is shown in figure 2. Examples below are taken from the telecommunications field.

The usage is specified as a hierarchy. The state on the top represents all the usage. The users can be divided into different user types or categories, for example for a small business exchange, secretaries, other employees and modem connections. Note that this example shows that a user must not be human.

For each of the user types, a number of individuals are specified on the user level, for example one secretary, four other employees and one modem connection.

Each user individual can use a number of services, which are specified on the service level, for example "basic call" and "call forward".

The usage of the services is then specified as plain Markov chains on the behaviour level.

The SHY model can be applied with different levels of detail depending on the application. The behaviour level can for example be excluded if less details are to be specified in the usage model.

4.1.2 Usage profile

The usage profile adds the probabilities for selection of the branches to the usage model. Probabilities are assigned to the transitions in the behaviour level Markov chains as well.

The probabilities are assigned based on measurement on usage of earlier releases or on expert knowledge. The SHY model makes it possible to analyse parts of the usage and assign probabilities for only that part of the model at a time, for example a user type.

The assignment does not have to be in absolute figures. Classes of usage frequency can be used, for example very frequently, frequently and seldom used. These classes can be assigned relative probabilities which may be an easier task than to assign every single probability.

4.2 Reliability model

To analyse the failure data collected during the statistical testing a reliability model is needed. Several models have been published over the last 20 years, see [8] for an overview. Models of different complexity and possibility to estimate the software reliability have been presented.

One very simple model which is suitable for software certification is the hypothesis testing control chart model [15]. It is based on a traditional quality control technique: sequential sampling [9].

The model is based on a control chart with three regions, reject, continue and accept, see figure 3. The control chart is constructed based on the required level of confidence in the estimation.

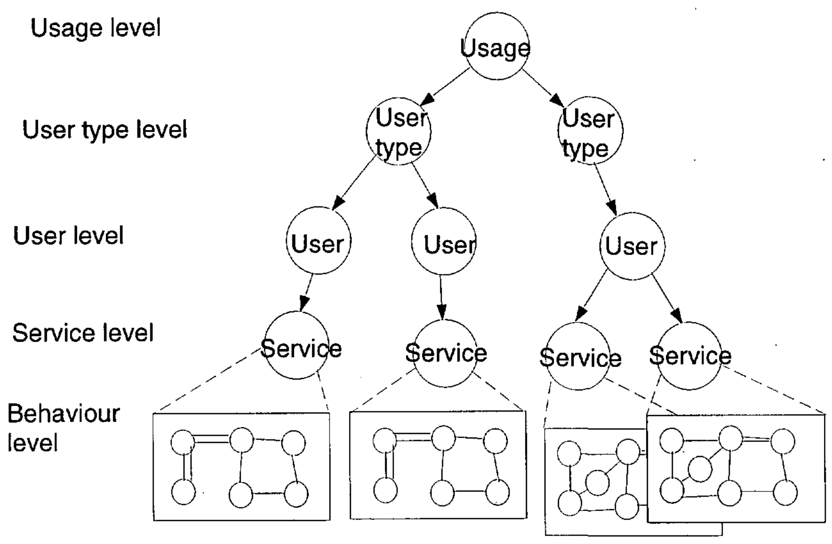


Figure 2: SHY model.

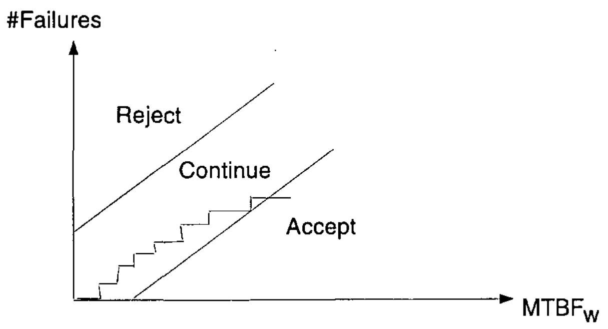


Figure 3: Hypothesis testing control chart.

The failure data are plotted in the chart, failure number towards weighted time between failures, where weighted time between failures means the measured time divided by the MTBF requirement. As long as the plots fall in the continue region, the testing has to continue. If the plots fall in the rejection region, the software reliability is so bad that the software has to be rejected and re-engineered. If the plots fall in the acceptance region, the software can be accepted based on the required MTBF with given confidence and the testing can be stopped.

Thus the hypothesis certification model provides a means for certifying the software and giving a reliability measure for the software as well as a means for controlling the testing effort.

4.3 SUT method outline

The models presented above can be applied according to the following method:

During specification:

- Produce the usage model.
- Assign the usage profile.

During test:

- Select test cases from the usage specification.
- Run the test cases and collect failure data.
- Certify the software.

During step 5 a decision is made based on the certification model outcome. If the failure data plots fall in the continue region, the method is repeated from 3 to 5 again. If the software is rejected, it is put back for redesign and finally if the failure data fall in the acceptance region, the certification is stopped and the software is accepted.

5 Example

This section contains an example which purpose is to make the models and methods presented in section 4 easier to understand and to apply. The method followed is the one presented in section 4.3. The subsections below are numbered according to the method outline. The focus is on usage modelling, see section 5.1, while section 5.2 to section 5.5 are more briefly described since the techniques for usage specification are less known than the other techniques.

The example on which the test method is applied is a private branch exchange (PBX) for a small office, see figure 4. Five human users are connected to the PBX, one secretary and four other employees. Furthermore there is one modem line. The connection with the outer world is through two lines. More details about the example specifications are given throughout the example.

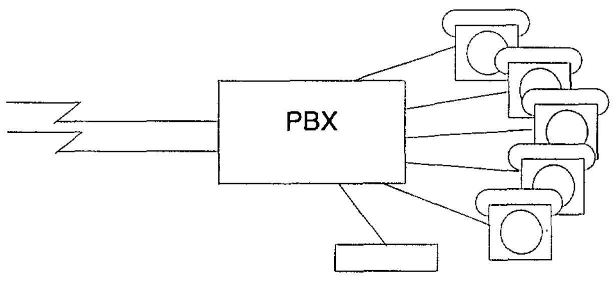


Figure 4: The example PBX system structure.

5.1 Produce the usage model

The SHY usage model, see figure 2, is produced in a number of steps, each of which is small and rather easy to perform. The steps are:

1. Identify the services available for the users.
2. Identify the user types and determine which services are available for each type.
3. Determine the number of individuals for each type.
4. Specify the behaviour Markov chain for each service.
5. Instantiate the services for the users.

In the example the first three steps are fully shown, while the last is only partially performed in this paper.

The usage model is produced starting with identification of the services available. The services are internal and external basic call, internal call forward, internal call transfer and call from the network.

The different types of users are identified. In this example there are four: Secretary, employee, modem and in/out line. The employees have internal and external basic call and internal call forward. The secretary has furthermore internal call transfer. The modem line has only external basic call. The in/out lines can only handle calls.

The different individuals of the user types are identified. There is one secretary, four employees, one modem and two in/out lines in the example.

In table 4 the first three steps in preparing the usage model for the PBX are summarised.

The behaviour level Markov chains for the services are then specified, see figure 5 and figure 6 with accompanying table 5 and table 6. For the internal basic call the stimuli are selected to be: Off Hook (OfH) (lift the receiver), Dial Internal Number (DIN), On Hook (OnH). There is also an asterisk (*) stimulus which means that the transition is forced by another behaviour level Markov chain. The state with dotted line is the start state. Note that the IBC service referenced in the link table is another instantiation of the service (another user).

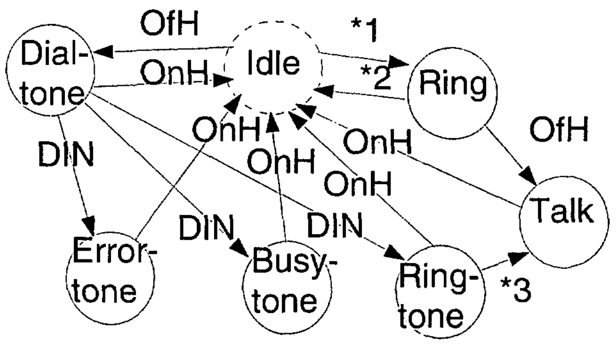


Figure 5: Behaviour Markov chain for internal basic call (IBC).

Link	Forced by
1	IBC: Dialtone-DIN
2	IBC: Ringtone-OnH
3	IBC, ICT: Ring-OfH

Table 5: Link table.

The behaviour level Markov chain for the internal call transfer service is described in figure 6 and table 6. The stimuli are: Activate Call Transfer (ACT), Dial Internal Number (DIN) and Transfer The Call (TTC). The linked transitions are forced by an instance of the Internal Basic Call (IBC) service.

In this manner all the services are specified; in this example however only these two are specified.

User type	Instances	Services
Secretary	1	Internal basic call (IBC)
		External basic call (EBC)
		Internal call transfer (ICT)
Employee	4	Internal basic call (IBC)
		External basic call (EBC)
		Internal call forward (ICF)
Modem	1	External basic call (IBC)
In/Out-line	2	Call (C)

Table 4: PBX usage.

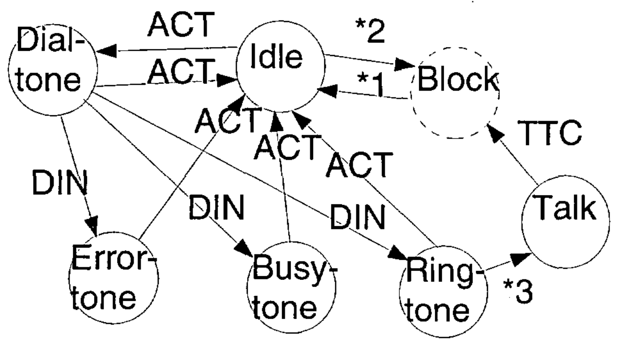


Figure 6: Behaviour Markov chain for internal call transfer (ICT).

Link	Forced by
1	IBC: Idle-OfH
2	IBC: OnH
3	IBC: Ring-OfH

Table 6: Link table.

Finally the state hierarchy model is compounded by its parts. The usage model for the example as a whole is presented in figure 7.

5.2 Assign the usage profile

When the usage model is produced the probabilities for the arcs have to be assigned, i.e. the usage profile is assigned.

The assignment starts on the user level which corresponds to what is well-known, at least in terms of relations between the usage. In the example it is assumed that one out of five of the events origin from each of the in/out-lines. Among the other users it is assumed that an event from the secretary is three times as probable as events from three of the employees and equally probable as events from the fourth employee. Modem line events are equally probable as events from one of the three least probable employees. These assumptions must come from market surveys, knowledge of existing and similar systems and expert opinions.

Based on this information, it is possible to derive probabilities for the usage model hence providing a basis for generating test cases which resemble the anticipated behaviour in operation. An equation can be set up which gives the absolute probabilities for the users (P_a):

$$\begin{aligned} P_a(In/out1) &= P_a(In/out2) = 0.2; & (1)\\ P_a(Secr) &= 0.18; & (2)\\ P_a(Empl1) &= P_a(Emp2) = \\ &= P_a(Emp3) = 0.06; & (3)\\ P_a(Emp4) &= 0.18; & (4)\\ P_a(Modem) &= 0.06. & (5) \end{aligned}$$

To apply these figures on the SHY usage model, they have to be divided on the user types and the user individuals. The user level probabilities (P_{ul}) are given by the relations between the individuals. Since the sum of the probabilities equals one, the calculations for the employees are:

$$\begin{aligned} P_{ul}(Empl1) &= P_{ul}(Emp2) = P_{ul}(Emp3) = \\ & \quad 0.06/(0.06+0.06+0.06+0.18) = \\ &= 0.166; & (6)\\ P_{ul}(Emp4) &= 0.18/(0.06+0.06+0.06+0.18) = \\ &= 0.5; & (7) \end{aligned}$$

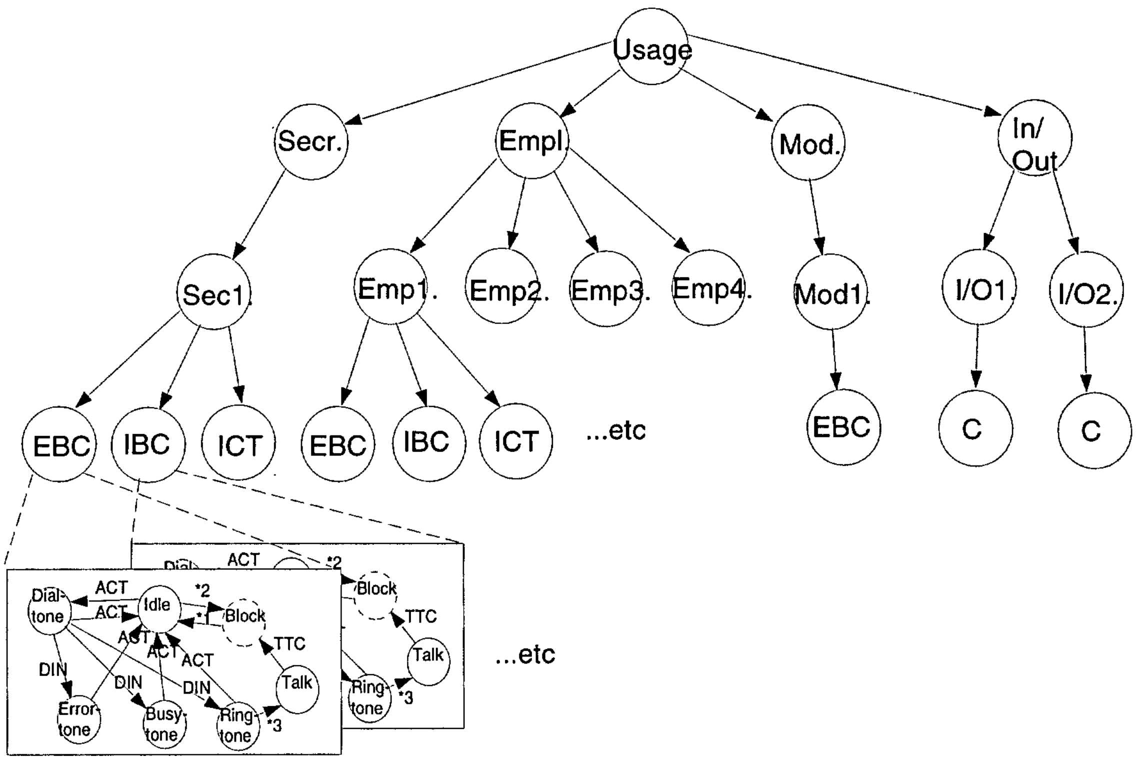


Figure 7: Usage model for the PBX example.

The sum of the absolute probabilities of a user type gives the user type level probabilities (P_{utl}):

$$P_{utl}(Emp) = 0.06+0.06+0.06+0.18 = 0.36; \quad (8)$$

When all of the calculations are performed the usage profile applied on the SHY model is according to figure 8:

The transitions in the behaviour Markov chains are assigned probabilities as well, except for the transitions forced by other services. This gives a complete usage specification from which test cases can be generated.

5.3 Select test cases

The test cases are selected from the usage specification by running through it beginning from the usage state and down to a single event. The actual path through the model is controlled by a random number sequence. For each event in a test case the specification is run through once. The beginning of an example test case is illustrated in table 7:

Event No.	Event
1	Emp3: Off Hook
2	Emp3: Dial Internal Number
3	In/Out2: Call subscriber
...	...

Table 7: Beginning of example test case.

5.4 Run test cases

The test cases are run as during any other type of testing. During testing, failure data are collected. The data form the basis for the certification, and are thus very important.

5.5 Certify the software

The failure data are input to the hypothesis testing model, see figure 3, which is used to certify a particular reliability level. The reliability is measured in terms of MTBF, and the certification is done with a given statistical confidence. The outcome from the certification is reject, continue or accept. In the case of reject, the software is sent back for redesign, in the case of continue,

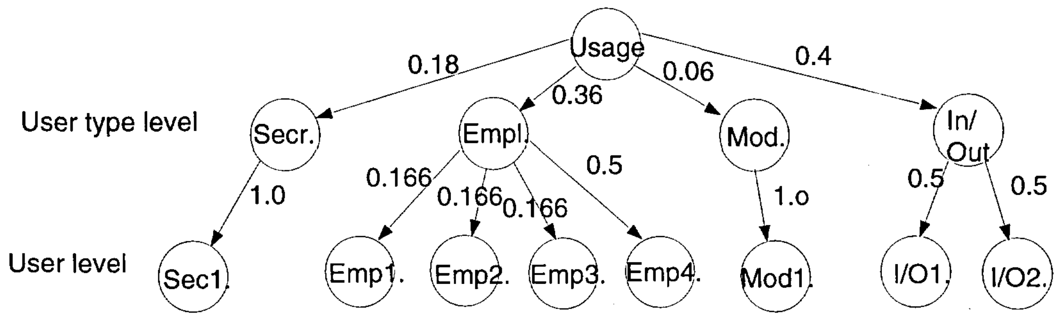


Figure 8: Usage profile for the upper levels of the example.

new test cases are selected and run, see sections 5.3 and 5.4 above. If the outcome is accept the reliability level is certified and the testing can be terminated.

6 Practical experience

Statistical Usage Testing can be applied at different phases in the software life cycle. The testing can be applied during system testing or acceptance testing, but it may also be applied on software components, [27], which then can be put into a repository for future reuse. The reuse of components is one important aspect in the future to cope with the cost of software development. Reuse requires that reliability measures of the reusable components are stored with the component. Reliability measures must be stored together with the usage profile which has been used in the certification process. Based on the reliability of components it must be possible to calculate the system reliability. This issue is further discussed in [20].

The new method presented is not fully developed, but it is beginning to be implemented and evaluated. Currently, a case study is conducted to evaluate the procedure. The study includes dynamic analysis, simulation and finally testing. The results from performing usage analysis during all these activities will form the basis for a thorough evaluation of usage analysis in the software life cycle. Furthermore, usage modelling with a hierarchical Markov chain (SHY), which then is transformed into a usage model in a high level design technique, has been used to generate usage test cases to the next release of a case tool, [22]. It is concluded from this application, that both the SHY model and the transformation into the high level design technique are useful concepts when generating usage cases. The main reason to transform the Markov model is to obtain tool support, which can be used to automatically generate test cases.

It can from this reasoning be concluded that usage testing is a useful technique, which can be applied at different phases in the life cycle with the common denominator that reliability certification is needed to stay in control of the reliability.

Application of Statistical Usage Testing or similar techniques have started at different companies, in particular in the USA. AT & T has reported that they have lowered the cost for system testing by 56% and the total cost in the project by 11.5% by applying Operational Profile Testing, [16], [17], [1] and [11]. The objective with Operational Profile Testing is the same as for Statistical Usage Testing even if some of the techniques to specify the usage are different. The usage testing technique is starting to spread in Europe as well [24] and [5].

7 Conclusions

It is a fact that reliability or availability requirements are formulated as a part of the requirements specification, but it is also clear that neither the developer nor the procurer of the software is capable of evaluating these requirements. This is not satisfactory; the society depends so heavily on the systems that it is of outermost importance to be able to certify the software systems. A failure in operation may cause injuries either in terms of humans or at least in terms of financial losses.

A model to specify the usage has been presented and a reliability model based on hypothesis testing control chart has been described briefly. These techniques together have made it possible to formulate a method, which can be applied during the testing phase to actually evaluate the reliability requirements. The application of the proposed method has been illustrated in an example.

Some practical experiences reported in the literature as well as experience obtained while applying the proposed techniques have been presented. The overall conclusion is, that the only way towards control of the reliability before releasing a software product is through application of usage testing techniques. It is the only technique that has shown to be able to certify the reliability requirement in the same time as it is cost effective. The application of the testing technique facilitates the formulation of a stopping criterion for software testing, i.e. the testing can terminate as the required reliability level has been reached.

The time has come to change the way of testing software. The objective must not be to find faults in general, but to show that the reliability requirements have been met. Usage testing aims at finding the faults influencing the reliability the most, instead of just removing arbitrary faults. The technique is mature enough to be used and those managing the transition first will probably be the ones delivering the products with the right reliability, which not necessarily is the highest.

Acknowledgement

We would like to thank the anonymous referees for providing valuable comments which have improved the article.

References

[1] S. R. Abramson, B. D. Jensen, B. D. Juhlin and C. L. Spudic, *International DEFINITY Quality Program*, Proceedings International Switching Symposium, Yokohama, Japan, (1992).

[2] E. N. Adams, *Optimizing Preventive Service of Software Products*, IBM Journal of Research and Development, January, 1984.

[3] T. DeMarco, *Controlling Software Projects*, Yourdon Press, New York, USA, (1982).

[4] R. H. Cobb and H. D. Mills, *Engineering Software Under Statistical Quality Control*, IEEE Software, pp. 44-54, November, (1990).

[5] H. Cosmo, E. Johansson, P. Runeson, A. Sixtensson and C. Wohlin, *Cleanroom Software Engineering in Telecommunication Applications*, Proceedings 6th International Conference on Software Engineering and its Applications, Paris, France, pp. 369-378, November (1993).

[6] M. Dyer, *The Cleanroom Approach to Quality Software Development*, John Wiley & Sons, (1992).

[7] A. L. Goel and K. Okumoto, *Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures*, IEEE Transactions on Reliability, Vol. 28, No. 3, pp. 206-211, (1979).

[8] A. L. Goel, *Software Reliability Models: Assumptions, Limitations and Applicability*, IEEE Transactions on Software Engineering, Vol. 11, No. 12, pp. 1411-1423, (1985).

[9] E. Grant and R. S. Leavenworth, *Statistically Quality Control*, Sixth edition, McGraw-Hill Int., (1988).

[10] Z. Jelinski and P. Moranda, *Software Reliability Research*, Statistical Computer Performance Evaluation, pp.465-484, (1972).

[11] B. D. Juhlin, *Implementing Operational Profiles to Measure System Reliability*, Proceedings 3rd International Symposium on Software Reliability Engineering, pp. 286-295, Raleigh, North Carolina, USA, (1992).

[12] R. C. Linger, *Cleanroom Process Model*, IEEE Software, pp. 50-58, March, (1994).

[13] H. D. Mills, M. Dyer, and R. C. Linger,*Cleanroom Software Engineering*, IEEE Software, pp. 19-24, September, (1987).

[14] H. D. Mills and J. H. Poore, *Bringing Software Under Statistical Quality Control*, Quality Progress, pp. 52-55, November, (1988).

[15] J. D. Musa, A. Iannino and K. Okumoto, *Software Reliability, Measurement, Prediction and Application*, McGraw-Hill Int., (1987).

[16] J. D. Musa, *Software Reliability Engineering: Determining the Operational Profile*, Technical Report AT & T Bell Laboratories, Murray Hill, NJ 07974, New Jersey, USA, (1992).

[17] J. D. Musa, *Operational Profiles in Software Reliability Engineering*, IEEE Software, pp. 14-32, March, (1993).

[18] G. J. Myers, *The Art of Software Testing*, Wiley Interscience, (1979).

[19] *The Cleanroom Case Study in the Software Engineering Laboratory - SEL 90-002*, Software Engineering Laboratory, (1990).

[20] J. H. Poore, H. D. Mills and D. Mutchler, *Planning and Certifying Software System Reliability*, IEEE Software, pp. 88-99, January, (1993).

[21] P. Runeson and C. Wohlin, *Usage Modelling: The Basis for Statistical Quality Control*, Proceedings 10th Annual Software Reliability Symposium, pp. 77-84, Denver, Colorado, USA, (1992).

[22] P. Runeson, A. Wesslén, J. Brantestam and S. Sjöstedt, *Statistical Usage Testing using SDL*, Submitted to 7th SDL Forum, Oslo, Norway, 25-29 September, (1995).

[23] R. W. Selby, V. R. Basili and F. T. Baker, *Cleanroom Software Development: An Empirical Evaluation*, IEEE Transactions on Software Engineering, Vol. 13, No. 9, September, (1987).

[24] L-G. Tann, *OS-32 and Cleanroom*, Proceedings 1st European Industrial Symposium on Cleanroom Software Engineering, Copenhagen, Denmark, October, (1993).

[25] J. A. Whittaker and J. H. Poore, *Markov Analysis of Software Specifications*, ACM Transactions on Software Eng. Methodology, Vol. 2, pp. 93-106, January, (1993).

[26] C. Wohlin, *Managing Software Quality through Incremental Development and Certification*, In Building Quality into Software, pp. 187-202, edited by: M. Ross, C. A. Brebbia, G. Staples and J. Stapleton, Computational Mechanics Publications, Southampton, United Kingdom, (1994).

[27] C. Wohlin and P. Runeson, *Certification of Software Components*, IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 494-499, (1994).

Performance Analysis of Disk Mirroring Techniques

Cyril U. Orji, Taysir Abdalla
School of Computer Science, Florida International University
Miami, Florida 33199
{orji,abdallat}@geneva.fiu.edu
phone: (305)348-2440; fax: (305)348-3549
AND
Jon A. Solworth
Department of EECS (M/C 154), University of Illinois at Chicago
Chicago, Illinois 60607-7053
solworth@parsys.eecs.uic.edu
phone: (312)996-0955; fax: (312)413-0024

Keywords: disk mirroring, chained & interleaved declustering, RAID

Edited by: Jerzy R. Nawrocki
Received: September 19, 1994 **Revised:** January 12, 1995 **Accepted:** January 26, 1995

Traditional mirroring maintains a logical disk image on two physical disks thereby improving reliability and read performance during normal mode operation. However, failure mode operation may be inefficient since the load on a single surviving disk could potentially double. Moreover, write performance during normal mode operation is inefficient since a single data item must be written to both disks in a mirror set. Interleaved and chained declustering improve load balancing during failure mode operation, while distorted mirror improves write performance.
This paper presents a comparative study of the performance of three mirroring schemes – traditional mirroring, distorted mirroring, and interleaved declustering under various operating conditions – normal, degraded and rebuild modes. Our findings show that using a disk track as a rebuild unit provides the best balance between response time and rebuild time. In addition, the performance of traditional and distorted mirrors during the rebuild process is adversely affected if incoming application requests are dominated by read requests while interleaved declustering is adversely impacted by write requests.

1 Introduction

Disk mirroring replicates a logical disk image on two physical disks. The two physical disks are called the *mirrored set*, and in *traditional mirroring* the disks in the mirrored set contain identical disk images.

Traditional mirroring improves reliability (Bates & TeGrotenhuis 1985) and performance (Bitton & Gray 1988, Bitton 1989) over a single disk system. But there are at least two problems associated with traditional disk mirroring. The first is that writes are expensive. Since every data block must be written by both disks, two randomly placed arms must seek to the same location, increasing the (maximum) seek distance for a random write to about $0.46n$ (Bitton 1989) instead of $0.33n$ in a single disk system. We refer to this as the *write* problem. The second is that traditional mirroring has poor, or in fact, no load balancing capability when one of the disks in a mirror set fails. The single surviving disk not only bears the workload previously shared by 2 disks, but also must use up additional bandwidth reconstructing the failed disk. This can degrade failure mode performance to unacceptable levels. We refer to this as the *load balancing* problem.

There are techniques for dealing with the write and load balancing problems of mirrored disks. By batching writes as proposed in (Solworth &

Orji 1990) disk mirroring with alternating deferred updates (Polyzois et. al. 1993) achieves improved write performance even with increasing write ratio. In (Orji & Solworth 1993, Solworth & Orji 1993), distorted and doubly distorted mirrors were used to improve small write performance by up to a factor of 2 over traditional mirrors. However, distortion techniques increase the (relative) cost of recovery vis a vis traditional techniques.

To address the load balancing problem, Teradata introduced Interleaved Declustering (Teradata 1985). In this scheme, data is partitioned over a number of disks in a cluster so that if a single disk fails, all the remaining disks in the cluster can share the load of the failed disk and also help in its reconstruction. In traditional mirroring, 2 disks in a mirror set are tightly coupled. This does not allow other disks in a network to help in load sharing when a disk fails. In chained declustering (DeWitt & Hsiao 1990) used in the GAMMA database machine (DeWitt et. al. 1990), 2 disks do not form a mirror set; instead mirrors are formed by a *chained* technique where the replica of disk i in an N disk network is maintained on disk $(i+1)$ *mod* N. This allows all $N-1$ disks to participate in load sharing by a dynamic reassignment of queries when one of the disks fails.

We have used simulation to study the behavior of these mirrored architectures under various operating conditions. This paper presents results obtained evaluating three of these schemes. The three schemes reported here are traditional mirroring, distorted mirroring and interleaved declustering. Traditional mirroring is considered the base mirroring scheme and provides a basis for comparative evaluation between other schemes. Distorted mirroring is an example of a scheme that addresses the write problem while interleaved declustering is an example of a scheme that addresses the load balancing problem. These three schemes are briefly overviewed in Section 2.

In (Menon & Mattson 1992) three operating modes were defined for drives in a disk array. These modes were used as a basis for evaluating the performance of drives in a RAID-5 (Patterson et. al. 1988) array.[1] We use similar modes for evaluating mirrored schemes. These are *normal*, *degraded* and *rebuild* operating modes. In normal mode, all disks in a mirror set are functional; in degraded mode, one of the disks in a mirror set has failed and no attempt is being made to reconstruct it, and all requests are serviced by the functioning disk(s); in rebuild mode, the failed disk is being reconstructed. If the second drive in the set fails while the system is in degraded mode or before the failed drive is fully reconstructed in the rebuild mode, the system will lose data.

[1]For completeness, we note that RAID is a taxonomy of data architectures that includes disk mirroring as level 1. However, in this paper, we refer to disk mirroring as a separate architecture. Our reference to RAID will imply RAID level 5 or rotated parity RAID.

A number of studies have evaluated and compared rebuild strategies in RAID-5 and traditional mirror schemes. We briefly overview some of these studies in Section 3. However, our study is motivated by the fact that none of these studies focused on the various forms of mirrored architectures. The studies implicitly assumed a traditional mirroring scheme. But, as evidenced by our findings, results from traditional mirroring scheme do not necessarily hold for other mirroring schemes. To evaluate the relative performance of the mirroring schemes under the various operating modes, we describe in Section 4 a simulator we constructed, and the types and parameters of workloads simulated. In Section 5 the rebuild algorithms for the various mirroring schemes are presented. Section 6 is devoted to the results obtained, and we conclude the paper in Section 7.

2 Mirroring Techniques

In this section we briefly overview traditional mirroring, distorted mirrors and interleaved declustering.

2.1 Traditional mirroring

Traditional mirroring maintains a logical disk image on two physical disks. The two physical disks are exact mirrors of one another. Figure 1 shows an example of traditional mirroring. Four disks are shown; disk 0 and disk 1 contain identical data and hence form a mirror set; similarly disk 2 and disk 3 form another mirror set.

Traditional mirroring improves reliability (Bates & TeGrotenhuis 1985) and performance (Bitton & Gray 1988, Bitton 1989) over a single disk system. Reliability is improved by replicating each component in the I/O subsystem. This allows continuous operation of the I/O subsystem even in the event of a single component failure. During normal operation, read requests can be serviced by any of the two disks that contain the requested data. As a result, the system throughput could be potentially doubled. In addition, a read request is efficiently serviced using a nearest free arm scheduling algorithm. For a disk with n cylinders, traditional mirroring reduces the average seek distance of a read request to about $n/5$ cylinders (Bitton 1989) This is in contrast to the average seek distance of about $n/3$ cylinders using a single disk. Intuitively, the 2 disk arms divide the total disk band into two regions.

Since both disks must write a block, two randomly placed arms must seek to the same location. The (maximum) seek distance for a random write is increased to about $0.46n$ (Bitton 1989) instead of $0.33n$ in a single disk system. By batching writes as proposed in (Solworth & Orji 1990) disk mirroring with alternating deferred updates (Polyzois et. al. 1993) achieves improved write performance even with increasing write ratio. However, this comes at the cost of reduced read efficiency since read requests do not take advantage of the two disk arms as suggested in (Bitton 1989).

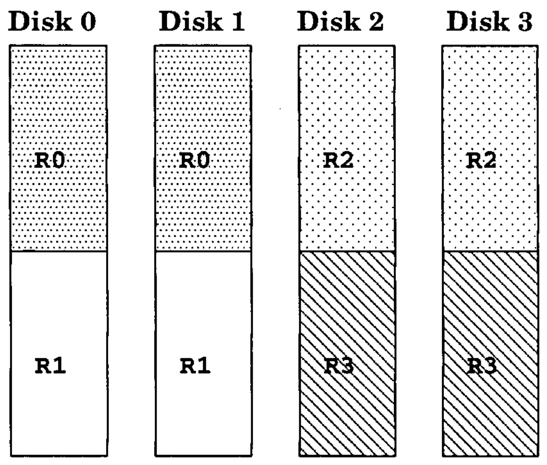


Figure 1: Traditional Mirroring. *Disk0 and Disk1 form a mirror set while Disk2 and Disk3 form another set.*

When one of the disks in a mirror set fails, all requests are serviced by the surviving disk until the failed disk is reconstructed on another drive. Disk reconstruction usually involves copying the survivor disk to a replacement disk.

2.2 Distorted mirrors

Distorted mirrors (Solworth & Orji 1991), like traditional mirrors, replicate a logical disk across the physical disks in a mirrored set. An example is shown in Figure 2. Four disks are shown; the four disks form two mirror sets as in traditional mirroring. Each physical disk is divided into two unequal parts, a *master* partition and a slightly larger *slave* partition. The two master partitions in a mirror set form a logical disk. For example in Figure 2, $M_0 \cup M_1 = 1$ logical disk. Each block in a logical disk is mastered on one disk and slaved on the mirror. Blocks are organized sequentially in the master partition and written arbitrarily in the slave partition. For example, in Figure 2 the blocks in M_0 are mastered and written in sequential order on disk 0. The same blocks are written in arbitrary free locations in the slave partition S_0 of disk 1.

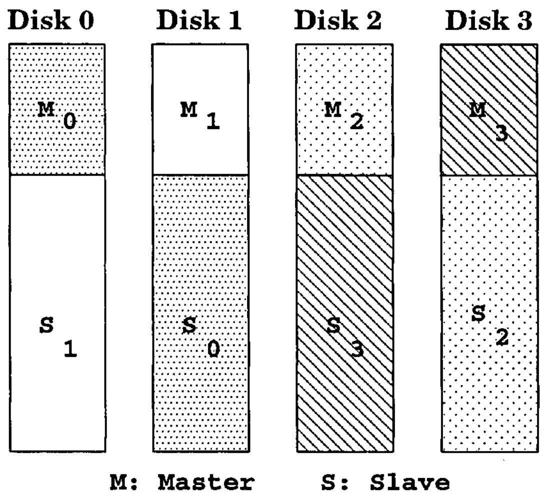


Figure 2: Distorted Mirrors. *Disk0 and Disk1 form a mirror set while Disk2 and Disk3 form another set. Blocks mastered on a disk as M_i are slaved on the partner disk as S_i. For example, the disk blocks mastered on disk0 as M_0 are placed in the slave partition of the partner disk1 as S_0.*

Distorted mirror uses a *diskfull factor (DFF)* to determine the amount of free space in the slave partition. DFF is the proportion of the physical disk that holds valid data. For example, a DFF of 80% means that 80% of the physical disk contain valid data and the remaining 20% are left free to facilitate efficient location of free blocks in the slave partition. Another way to look at this is that 40% of the physical disk is used as master partition, and the remaining 60% used as slave partition contain 40% worth of blocks mastered on the partner disk. We sometimes use the term *backup factor* to present the complementary information; precisely $DFF = 100\% - backup\ factor$.

During normal operation, single block read requests can be serviced by either the master or slave disk while sequential read requests are serviced by the master disk. Writes are serviced by both disks, but slave write is usually very efficient since blocks are written *anywhere* there is a free block on the slave disk. Like traditional mirroring, a surviving disk in a mirror set picks up the load when its partner disk fails. However, unlike traditional mirroring, disk reconstruction is more complex and time consuming since a disk-to-disk copy cannot be performed because of the distorted mapping technique employed.

2.3 Interleaved declustering

The Teradata database machine (Teradata 1985) uses interleaved declustering for fault tolerance and load balancing. A relation R is horizontally partitioned across N disks. The N disks are further grouped into clusters of C disks such that $N = kC$ for some integer $k > 1$. Figure 3 shows a cluster of four disks ($C = 4$). Disk i contains relation partition Ri as a primary copy. The backup copy of relation partition Ri is subdivided across the remaining disks in the cluster. For example, in Figure 3, $R0$ is subdivided into three backup fragments labeled $r0.0, r0.1$, and $r0.2$. These fragments are placed on the remaining $C - 1$ disks in the same cluster. This arrangement allows data availability to be maintained in the presence of any single disk failure.

During normal operation, read requests are directed to the primary copy and write requests are directed to both the primary and backup copies. If a failure occurs, an appropriate backup copy is promoted as primary copy and the workload of the failed disk is shared by the remaining $C - 1$ disks in the cluster so that their workload is increased by approximately $1/(C - 1)$. All $C - 1$ disks in a cluster participate in the reconstruction of a failed cluster member. This reduces the chance of overloading of a single disk; a situation that is common in other mirrored schemes.

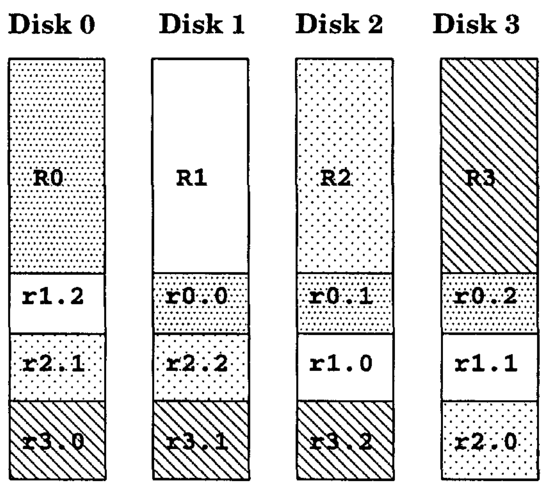


Figure 3: Interleaved Declustering. *The four disks form a cluster. 50% of each disk contains primary data and the remaining 50% is used to backu up part of the primary data of the other disks. For example, 50% of disk0 contains R0, the primary copy of relation 0, while the remaining 50% contains* $r1.2, r2.1$, *and* $r3.0$ *which are fragments of the other relations whose primary copies are stored in the other disks that belong to same cluster as disk0.*

3 Related Work

Studies have been performed to evaluate techniques for reconstructing a failed disk in high availability disk arrays. Using analytic models, Muntz and Liu (Muntz & Lui 1990) developed a *baseline copy* algorithm for rebuilding a failed disk in a RAID-5 array. The baseline copy algorithm is similar to a disk-to-disk copy algorithm except that in this case, the survivor disks are read and their data used to regenerate data in the failed disk. All read requests are directed to the survivor disk. If a read request maps to the failed disk, the data is regenerated from the survivor disks. There are two enhancements to the baseline copy algorithm — *redirection of reads* and *regenerated writes*. Redirection of reads allows read requests to be redirected to the replacement disk if the data has been reconstructed. With regenerated writes, if data is regenerated to satisfy a read request that maps to a failed disk, the data is also written to the replacement disk so it does not have to be regenerated again.

The studies of Muntz and Liu were theoretical. However, they were sound and formed a basis for subsequent work in this area. For example, Holland and Gibson (Holland & Gibson 1992) extended these algorithms and performed simulation studies to verify them. Using simulation, they were able to detect some non-intuitive behavior in some of the algorithms. Their simulation showed that redirection of reads and piggybacking may not always be beneficial especially when the survivor disks are not saturated.

In (Hou, Menon & Patt 1993), the effect of the *rebuild unit* on rebuild time and response time was investigated. Three rebuild units – block, track, and cylinder were studied and their impact on rebuild time and response time of incoming application requests investigated. It was found that rebuilding data one block at a time ensures that incoming application requests are minimally delayed, however, an increase in rebuild time results. At the other extreme, if a cylinder worth of data is atomically rebuilt, the response times of incoming application requests are significantly increased, however, the rebuild process takes a shorter time. Hou, Menon and Patt concluded that a track rebuild unit provided the best compromise between response time degradation and rebuild time. This finding was also supported by findings in a related study by Hou and Patt (Hou & Patt 1993). The study evaluated RAID-5 and similar capacity mirrored arrays using algorithms developed in (Holland & Gibson 1992, Muntz & Lui 1990).

In this paper, we do not study RAID-5 arrays. Our focus is on mirrored organizations. In Section 4 we describe a simulation model used to evaluate these schemes. We also describe the workload simulated.

4 Simulation Model

We wrote a simulator to evaluate the expected performance of traditional mirroring, interleaved declustering and distorted mirroring under various operating modes. We assumed an array of 16 disks; thus for the traditional and distorted mirrors, there are 8 mirror sets, with 2 disks in each set. The two disks contain identical data as described in Section 2. For the distorted mirror, 80% diskfull factor was used.[2] For the interleaved declustering, we partitioned the 16 disks into 2 clusters (clusters 0 and 1), with a cluster size (C) of 8 disks. Data distribution in the 8 disks is as described in Section 2.

The parameters of the disk drives modeled in the experiments are shown in Table 1. The parameters are identical to those of IBM 0661 Model 370 disk drive. Also shown in Table 1 is a summary of the workload parameters used in the experiments. The system may experience two types of workloads depending on its operating mode. Under normal and degraded operating modes, all requests are assumed to be initiated by application programs. The workload is modeled as a synthetically generated stream of disk requests characterized by its type (whether a read or a write). Requests are generated uniformly across the disk space and each request accesses 4KB (one block) of data. Other studies have used 75% read, 25% write as a representative mix for most on-line transaction processing (OLTP) applications. We also studied this mix, but in addition, we studied an all write workload (0% read), an all read workload (100% read) and a workload with an equal mix of read and write requests (50% read). We varied the I/O rate to the 16 disks from 100 I/Os to 350 I/Os per second.

When the system is in rebuild mode, in addition to the normal application requests, the system initiates rebuild requests. A rebuild request triggers a disk read on the surviving disk(s) and a corresponding disk write on the replacement disk. In all the schemes, requests generated by the application are given higher priority over rebuild requests. A rebuild request is serviced if and only if there is no outstanding application request; however, once started, a rebuild request cannot be preempted by an application request.

5 Rebuild Algorithms

In this section we describe the algorithms used during the rebuild phase in each mirroring scheme. The rebuild algorithms for the traditional mirror are discussed first, then the rebuild algorithms for distorted mirrors and interleaved declustering are also discussed.

[2]The implication is that in the distorted mirror, the logical disk is slightly smaller than the logical disk in the traditional mirror. However, we considered the effect of this to be minimal and is not considered any further in the experiments.

Disk Parameters	
No of cylinders	949
Tracks per cylinder	14
Blocks per track	6
Block size	4096 bytes
Average seek	12.50 ms
Rotational speed	4318 RPM
Seek cost function	$2.0 + 0.01(distance) + 0.46\sqrt{distance}$
Latency cost function	uniform
Workload Parameters	
Request Size	4 KB
Read Ratio	0 – 100%
Request Distribution	Uniform
I/O rate	100 to 350/sec.

Table 1: Disk and workload parameters

5.1 Rebuild algorithms for traditional mirror

The rebuild algorithm for traditional mirror is similar to the baseline copy algorithm in (Muntz & Lui 1990). However, the algorithm does not perform a sequential scan of the disk; instead, whenever there is no outstanding application request, a rebuild request is initiated for the "nearest" *unrebuilt unit* in the survivor disk. This policy minimizes arm motion in the survivor disk but could cause a substantial disk seek in the replacement disk. The reason is that the disk arms in the two disks are randomly positioned since all incoming application read requests are serviced by only the survivor disk. A variation of this algorithm would initiate the rebuild on the "nearest" unrebuilt unit in the replacement disk, thereby potentially increasing arm motion in the survivor disk. We preferred not to explore this alternative since the survivor disk is already highly loaded by normal application and rebuild requests.

In our experiments, three rebuild units were considered — block, track and cylinder rebuild units. Each unit defines the amount of data atomically read from the survivor disk and atomically written to the replacement disk. For example, if a track rebuild unit is in effect, a read request is initiated for the "nearest" unrebuilt track on the survivor disk.

In an enhancement to this algorithm, an application read request could be redirected to the replacement disk if the requested data has been reconstructed. If redirection is in effect, the choice of disk to service the request is made as in the normal mode of operation. The preferred choice is to schedule the request on an idle disk. If both disks are busy, a minimum service time model is used. Ties are randomly broken. If an application requests a write, the block is written to both disks and is considered reconstructed regardless of whether it was previously reconstructed or not.

5.2 Rebuild algorithms for distorted mirror

In distorted mirror a logical block has a fixed address on the master disk and a floating address on the slave disk. Blocks mastered on one disk are slaved on the mirror; consequently, a disk-to-disk copy algorithm cannot be used to reconstruct a failed disk. As in the other schemes, three rebuild units — block, track and cylinder were considered. With block rebuild unit, the algorithm simulated is exactly identical to the traditional mirror algorithm.

But the situation is different with track and cylinder rebuild units. With these rebuild units, we divided the reconstruction process into two phases. In the first phase, we reconstruct the slave partition of the replacement disk, and reconstruct the master partition in the second phase. This order of reconstruction has significant impact on system performance. The slave partition can be speedily reconstructed in less than 20% of the total reconstruction time. The reason for this will be obvious shortly; but given that 50% of the replacement disk blocks are reconstructed in such a short time, they can be made available for servicing incoming application requests.

Slave partition reconstruction involves reading data from the master partition of the survivor disk, and writing the data anywhere in the slave partition of the replacement disk. Since the replacement disk is initially empty, writing blocks anywhere in the slave partition incurs minimum cost because free blocks are easily located. Moreover, enough contiguous free blocks can be located (most of the time) to allow many sequential transfers. In addition, since data is sequentially written in the master partition, reading the survivor disk and writing the replacement disk can proceed at sequential access rate in this phase. This is why we reconstruct the slave partition before the master partition.

Master partition reconstruction involves reading data from slave partition of the survivor disk, and writing the data to the master partition of the replacement disk. Fetching slave blocks that map to contiguous addresses in the master partition requires random disk accesses on the slave disk. Assume cylinder rebuild unit is in effect; the first step in the rebuild process is to select the cylinder to reconstruct. The obvious choice is to choose a cylinder that minimizes the write cost on the replacement disk.[3] The blocks are randomly fetched from the survivor disk and sequentially written to the replacement disk. The master partition reconstruction is thus dominated by the random disk accesses to read the blocks from the survivor disk.

Another alternative algorithm which we explored for distorted mirroring is the use of two replacement disks (Polyzois et. al. 1993). In this technique, data is copied to two replacement disks and at the end of reconstruction, the survivor disk and failed disk are returned to the system as spares. The advantage of this technique is that during reconstruction, all incoming write requests are directed to the two replacement disks and the survivor disk is placed in read-only mode relieving it of all write operations. However, we did not observe any significant performance advantage over the single replacement disk algorithm, thus we do not discuss this any further in this paper. We focus only on the single replacement disk model for all the mirroring schemes.

Some points are in order here. In each distorted mirror experiment, we used a buffer that holds the number of blocks in the rebuild unit. For example, a buffer that holds just a block is used when the rebuild unit is block. An alternative that could improve master partition reconstruction time is to use write buffering techniques (Solworth & Orji 1990). In this technique, a large buffer is used to hold blocks and the disk transfer is delayed until enough blocks have accumulated in the buffer. Blocks that map to "nearby" locations on disk can be written together. We did not explore this alternative in the current study.

[3]Due to the random placement of blocks in the slave partition of the survivor disk, the cost of fetching the blocks is independent of the cylinder chosen.

Organization	**Algorithm**
TM	Rebuild *nearest* unrebuilt unit on *survivor* disk.
DM	
block rebuild	Same as in TM.
track & cyl. rebuild	1. Rebuild slave partition using *nearest* unrebuilt unit strategy as in TM. 2. Rebuild master partition using similar techniques.
ID	Rebuild *nearest* unrebuilt unit on *replacement* disk.
	All algorithms are optimized using redirection of reads.

Table 2: Summary of rebuild algorithms

5.3 Rebuild algorithms for interleaved declustering

The rebuild algorithm for interleaved declustering is similar to that of traditional mirroring. Because reconstructing data on the failed disk involves all the other disks in the same cluster, choosing the disk to fetch data from, at any point during the reconstruction process can impact performance.

In our experiments, we considered two disk selection options. In the first option, the survivor disk in the cluster that minimizes the fetch time for a unit of unrebuilt data is selected. This scheme leads to a minimal increase in disk utilization for the survivor disks, and also to a minimal increase in response time for incoming application requests. In the second option, we aim to minimize the (seek) cost of writing data to the replacement disk. We select a survivor disk that helps us achieve this objective. This second option ensures that the replacement disk performs optimally. The result is that rebuild time is shorter than in the first option. Results reported in this paper were obtained using this second option.

Block, track and cylinder rebuild units are also investigated. A summary of the rebuild algorithms is given in Table 2.

6 Results

In this section we present results of our experiments. The results are presented in three sections. In Section 6.1 we present results of the systems when operating in normal mode. In Section 6.2 we present degraded mode performance and present rebuild mode performance in Section 6.3. Due to lack of space, we are unable to present detailed results for all the read/write workloads studied. However, in Section 6.4 we briefly present some results representative of the various read/write ratios. But in most of this section, we focus on results for a 75% read (25% write) workload, which is typical of transaction processing environments.

The performance metrics used in our studies depend on the operating mode being investigated. Under normal and degraded operating modes, the response time of application read requests was of interest. Under rebuild mode, in addition to the response time of application read requests, we also collected data on the time taken to rebuild a failed disk (*rebuild time*). We note that all response time data are given as 90th percentile response time for application read requests.

6.1 Normal mode operation

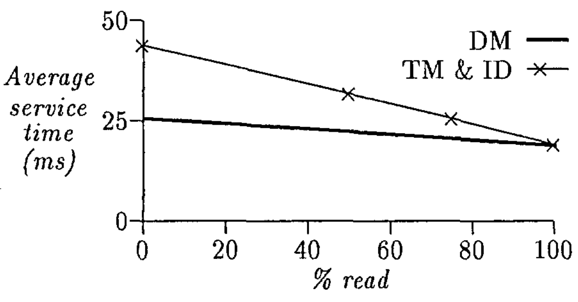


Figure 4: Read ratio vs. average service time (*I/O rate: 200 per second.*)

In Figure 4 we show for the three schemes, the service time of servicing a single random block request. The service time of a request consists of *seek, rotational latency*, and *transfer* time components. Thus, this is the time a disk arm is tied up servicing a request. Two curves are shown in the figure, one for distorted mirror and the other for both the traditional mirror and interleaved declustering. Note the superior performance of distorted mirror especially for high write rates. For example, with a 0% read (100% write) workload, distorted mirror shows a 70.8% superior performance over the other schemes. This performance advantage decreases with increasing read ratio; for example it is only 25.8% with 75% read. The three schemes have identical performance with 100% read workload, since each scheme uses only one disk to service a read. However, both disks in a mirror set must service a write request. Traditional mirror and interleaved declustering incur identical write service time costs on both disks in a mirror set. However, in distorted mirrors, the slave write is performed very efficiently. Using write anywhere technique in the slave partition, slave write is performed with no seek cost and with little or no rotational latency.

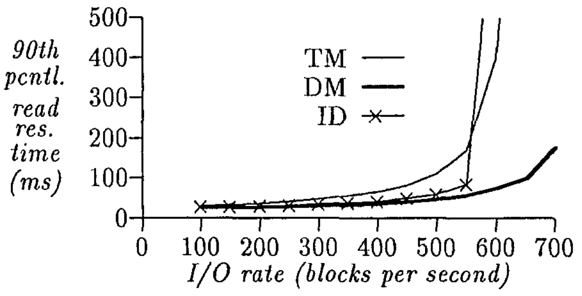


Figure 5: 90th percentile read response time vs I/O rate. *(75% read: Normal mode)*

Figure 5 shows the relative performance of the three schemes with respect to response time of read requests. At low I/O rates, the three schemes have comparable performance. We have shown very high I/O rates to demonstrate the ability of distorted mirror to handle such workloads. Note that neither traditional mirror nor interleaved declustering is able to handle such high I/O rates. These two schemes do not handle writes as efficiently as distorted mirror, hence every write request potentially adds to the queueing time of pending requests. In the rest of the paper, we do not discuss very high I/O rates any further.

We restrict our discussion to a maximum rate of 350 I/Os per second.

At 200 I/Os (or 12.5 I/0s per disk) per second, the 90th percentile read response time is 30.24 ms for interleaved declustering. The corresponding figures for traditional and distorted mirrors are 36.17 ms and 31.40 ms respectively. These performance numbers are also reflected in the disk utilizations; traditional mirror disks saturate faster than disks in the other schemes.

6.2 Degraded mode operation

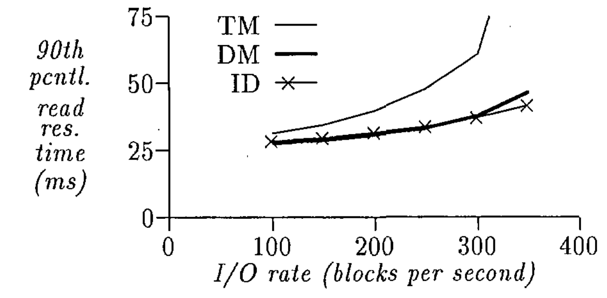


Figure 6: 90th percentile read response time vs I/O rate. *(75% read: The system is in degraded mode and response time is measured over all application read requests. This includes requests that map to the mirror set with a failed disk.)*

Figure 6 shows the response time of all application read requests when the system is in degraded mode. Interleaved declustering has the best performance, followed by distorted mirror. Traditional mirror saturates very quickly at high I/O rates. In Figure 7 we isolate and show the response time for requests that map to the set with a failed disk. At 200 I/Os per second, with the system in degraded mode, the response time in interleaved declustering increases by about 5.3% over normal mode response time. Increases were also observed in the other schemes. For distorted and traditional mirrors these were 73.8% and 76.4% respectively. These numbers demonstrate the ability of interleaved declustering to spread the workload of a failed disk over many disks.

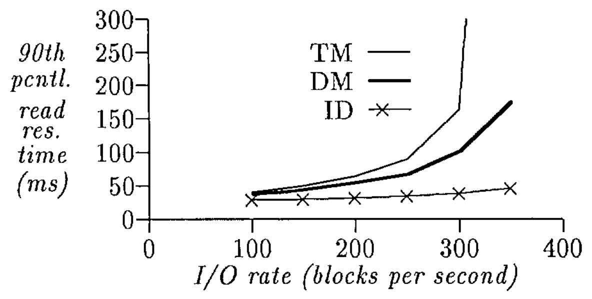


Figure 7: 90th percentile read response time vs I/O rate. *(75% read: The system is in degraded mode and response time is measured over only those application read requests that map to the mirror set with a failed disk.)*

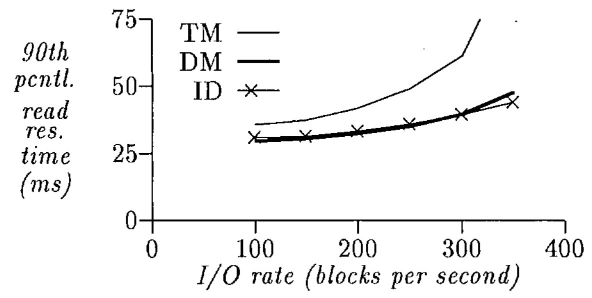


Figure 8: 90th percentile read response time vs I/O rate. *(75% read: The system is in rebuild mode and response time is measured over all application read requests. This includes requests that map to the mirror set with a failed disk. The rebuild unit is block.)*

6.3 Rebuild mode operation

We use two metrics for evaluating performance in rebuild mode – read response time and rebuild time. First we consider block rebuild unit, then track and cylinder rebuild units. Figure 8 shows the response time of read requests while the system is in rebuild mode. The graph shows rapid saturation of traditional mirror beyond about 300 I/Os per second. At 200 I/Os per second, there is a 10.4% increase over normal mode response time in interleaved declustering. Corresponding increases for distorted and traditional mirrors are 2.7% and 16.2% respectively. However, if we consider only requests that map to the failed set, the re-

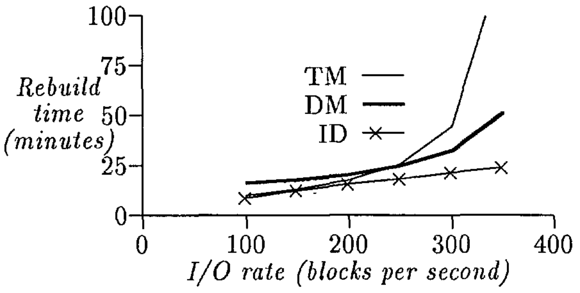


Figure 9: Rebuild time (in minutes) vs I/O rate. (*75% read: Total time to rebuild failed disk on a replacement disk using block rebuild unit.*)

sponse time increases for interleaved declustering, distorted mirror and traditional mirror are 16.5%, 88.5% and 89.3% respectively (graph not shown). This again demonstrates the ability of interleaved declustering to balance its load in failure mode.

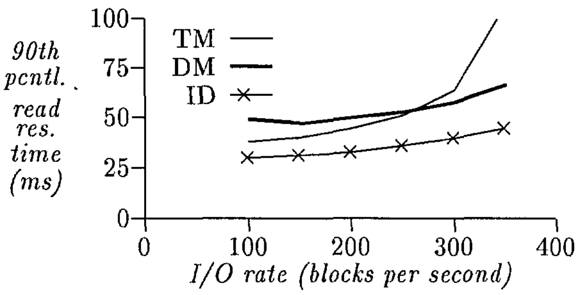


Figure 10: 90th percentile read response time vs I/O rate. *(75% read: The system is in rebuild mode and response time is measured over all application read requests. This includes requests that map to the mirror set with a failed disk. The rebuild unit is track.)*

Block rebuild time is graphed in Figure 9. At very low I/O rates, there is available bandwidth in all the schemes for rebuilding the failed disk; hence the performance is determined by the single disk being rebuilt and not by the other disks used in the rebuild process. Hence at low I/O rates, all the schemes have comparable performance. For the three schemes, an increase in I/O rate means reduced bandwidth for the rebuild process thereby increasing rebuild time. Traditional mirror saturates very quickly beyond 300 I/Os per second.

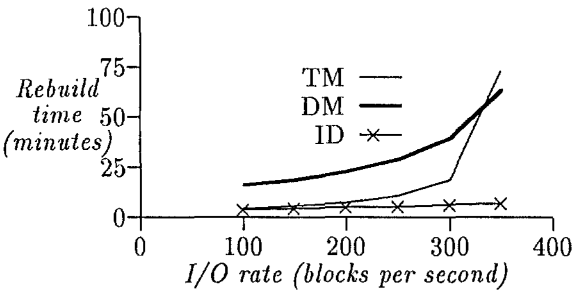


Figure 11: Rebuild time (in minutes) vs I/O rate. (*75% read: Total time to rebuild failed disk on a replacement disk using track rebuild unit.*)

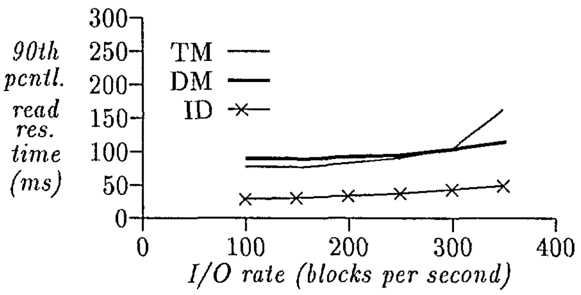


Figure 12: 90th percentile read response time vs I/O rate. *(75% read: The system is in rebuild mode and response time is measured over all application read requests. This includes requests that map to the mirror set with a failed disk. The rebuild unit is cylinder.)*

The graphs for track rebuild unit are shown in Figures 10 and 11 while the graphs for the cylinder rebuild unit are shown in Figures 12 and 13. The two sets of graphs have identical patterns. Considering the response time curves (Figures 10 and 12), we note that distorted mirror has the worst performance. This is due to the random disk accesses to rebuild a track or cylinder in the master partition of the replacement disk (see Section 5.2) which forces incoming application requests to queue up behind the rebuild request. Large rebuild units are clearly inappropriate for distorted mirrors during the rebuild phase.

The rebuild time graphs (Figures 11 and 13) are identical. For almost the entire range of I/O rates

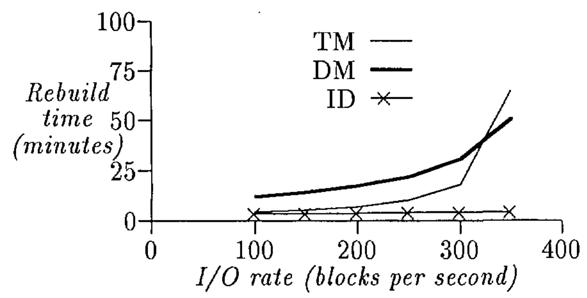


Figure 13: Rebuild time (in minutes) vs I/O rate. (*75% read: Total time to rebuild failed disk on a replacement disk using cylinder rebuild unit.*)

simulated, distorted mirror has the worst rebuild time for the same reasons explained above. Interleaved declustering rebuilds fastest. Traditional mirror has reasonable rebuild time at low I/O rates and starts to degrade as I/O rates increase.

In Tables 3 to 5 we summarize the response and rebuild time data for the three schemes to show the impact of redirection of reads. Two points can be made from the table. The first is that distorted mirror benefits most from redirection of read showing an average improvement of about 47.4% over the three schemes. Since the slave partition can be rebuilt in a relatively short time, there is benefit in making the blocks available for servicing incoming application requests. Traditional mirror has about 7.8% average improvement over the three schemes, while interleaved declustering has only a 1.3% improvement. In fact when we consider only those requests that map to the failed disk, interleaved declustering has a -12.8% improvement (or degradation) in response time with redirection of reads. The behavior supports the findings of Holland and Gibson (Holland & Gibson 1992); all algorithms do not benefit from redirection of reads. Interleaved declustering was able to balance its workload such that the disks were never saturated, hence there was no need to redirect read requests.

When the tradeoffs between response and rebuild times are considered, we arrive at conclusions similar to those in (Hou & Patt 1993). Using block rebuild unit minimally increases the response time of application read requests. This increase is significant when cylinder rebuild is used. However, cylinder rebuild offers the best performance if rebuild time is the primary consideration. Our experiments show that track rebuild is a good compromise that balances response time and rebuild time demands.

	TM		
	block	*track*	*cyl*
90% res. time (**NR**)	42.01	44.60	83.00
rebuild time (**NR**)	17.70	7.40	7.10
90% res. time (**R**)	39.10	41.50	74.98
rebuild time (**R**)	16.38	6.81	5.57
% improvement	6.9	6.9	9.6

Table 3: Response and rebuild time data (TM) at 200 I/Os per second. *75% read. Response time is in milliseconds while rebuild time is in minutes.*

	DM		
	block	*track*	*cyl*
90% res. time (**NR**)	32.25	49.94	93.22
rebuild time (**NR**)	20.59	22.73	17.56
90% res. time (**R**)	30.69	32.38	34.60
rebuild time (**R**)	19.22	15.48	11.83
% improvement	4.8	35.2	62.9

Table 4: Response and rebuild time data (DM) at 200 I/Os per second. *75% read. Response time is in milliseconds while rebuild time is in minutes.*

We note one important conclusion from Tables 3 to 5. It has to do with response and rebuild times in rebuild mode. Rebuild time is primarily important since it determines the amount of time in which response time is degraded. The response time for DM in rebuild mode with redirection of reads is only a few percent worse than in normal mode. Moreover the rebuild performance of DM is close to or better than normal performance of all other types. This is because rebuilds are write intensive, which DM excels at.

6.4 Other read/write ratios

In this section, we briefly show results that reflect behavior of the schemes at other read/write ratios. Due to lack of space, we discuss only rebuild time data using block rebuild unit. We also show only data at 200 I/Os per second.

	ID		
	block	*track*	*cyl*
90% res. time (**NR**)	33.38	33.02	34.59
rebuild time (**NR**)	15.50	4.97	3.83
90% res. time (**R**)	32.80	32.76	34.14
rebuild time (**R**)	16.28	5.20	3.97
% improvement	1.7	0.8	1.3

Table 5: Response and rebuild time data (ID) at 200 I/Os per second. *75% read. Response time is in milliseconds while rebuild time is in minutes. (In Tables 3 to 5, the system is assumed to be operating in rebuild mode. 90% response time is over all application read requests. Rebuild time is wall clock time in minutes to rebuild a failed disk. NR means no redirection of reads during the rebuild phase. R means reads were redirected to the replacement disk whenever it was possible and efficient to do so. % improvement is improvement in response time when read redirection was in effect. The capacity of each disk is approximately 320 mbytes.)*

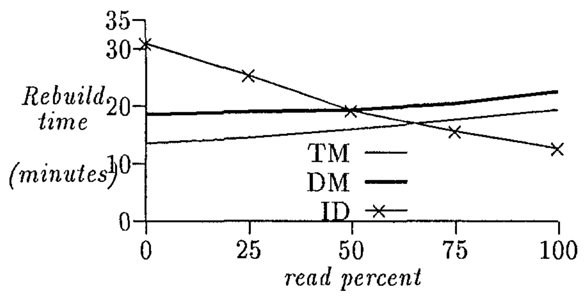


Figure 14: Rebuild time (in minutes) vs read ratio. (*I/O rate: 200 per second. Total time to rebuild failed disk on a replacement disk using block rebuild unit.*)

Figure 14 shows two different patterns for the three schemes; traditional and distorted mirrors have identical behavior while interleaved declustering has a different (opposite) behavior. While the results in Figure 14 were derived from the block rebuild unit data, we note that similar patterns were observed in both track and cylinder units.

The behavior of traditional and distorted mirrors is intuitive. An increase in read ratio will place high demands on the bandwidth of the survivor disk since the disk services all read requests. This degrades rebuild performance. Since interleaved declustering distributes the workload of the failed disk over many disks, we expected its performance to be relatively better than the performance of traditional and distorted mirrors. Our findings constradicted this intuition. With high write ratio, the performance of interleaved declustering was worse than that of the other schemes. For example, with 0% read (100% write), the rebuild time in interleaved declustering is 27.34 minutes; in traditional mirror it is 16.10 minutes and in distorted mirror it is 14.03 minutes.

The reason for this behavior can be explained as follows. 16 disks were used in the experiment. For traditional and distorted mirrors, this is equivalent to 8 mirror sets. Thus there is a 1:8 chance that a write request will map to the set with a replacement drive. While the request is being serviced, the drive is unavailable for rebuild operations. However, the 16 disks are divided into 2 clusters in interleaved declustering. Thus there is a 1:2 chance that a write request will map to the cluster with the replacement drive, increasing the probability that the replacement drive will be unavailable for rebuild operation. Thus, the higher the write ratio, the higher the chance that in interleaved declustering, the replacement disk would be unavailable for rebuild operations. We note that at high read ratios, interleaved declustering outperforms traditional and distorted mirrors.

7 Conclusion

We have examined the performance of three mirroring schemes under three operating conditions – normal, degraded and rebuild modes. Using a synthetic workload that closely models transaction processing environments, we studied various algorithms for rebuilding a failed disk in these mirroring schemes using three rebuild units identified in (Hou & Patt 1993). Most of our findings support findings in related studies.

- During normal mode operation, distorted mirror has best service time since the slave write is efficient.

- The response time for distorted mirror in rebuild mode with redirection of reads is only a few percent worse than in normal mode. Moreover the rebuild performance of distorted mirror is close to or better than normal performance of all other types. This is because rebuilds are write intensive, which distorted mirror excels at.

- Because writes are inefficient in traditional mirrors, disks used in this configuration saturate rapidly with increasing I/O rate.

- Interleaved declustering has the best performance for most of the workload studied. However, if the workload is dominated by writes, the performance of interleaved declustering in rebuild mode is worse than the performance of traditional and distorted mirrors.

- Redirection of reads is not beneficial in rebuild algorithms if the disks are not saturated.

- Track rebuild unit provides a good balance between response time and rebuild time demands.

There are many other possible algorithms, and variations to the algorithms studied here, that need to be investigated. We plan to pursue these as future extensions to this study. For example, if cost is a secondary issue and systems can maintain many spare disks on-line, it would be interesting to see how performance would differ when both disks in a mirror set are replaced when one of them fails. We explored this alternative for the distorted mirror and observed no significant performance improvements. But this scheme may not be straightforward (or actually feasible) with interleaved declustering; should all the disks in a cluster be replaced when one of them fails? We plan to explore this issue at least with a small disk size configuration. We also plan to see how beneficial the scheme is for traditional mirroring. Buffer size is another parameter of importance. In the experiments, the buffer size was fixed to the size of the rebuild unit. We would like to see how sensitive performance is to variation in buffer size. As suggested elsewhere in this paper, we suspect that distorted mirror could benefit from a large buffer space during rebuild operations.

References

[1] Bates K. & TeGrotenhuis M. (1985) Shadowing Boosts System Reliability. *Computer Design*, April 1985.

[2] Bitton D. (1989) Arm Scheduling in Shadowed Disks. *Proceedings of the IEEE Computer Society International Conference (COMPCON)*, pages 132–136, San Francisco, California, February 1989.

[3] Bitton D. & Gray J. (1988) Disk Shadowing. *Proceedings of the 14th International Conference on Very Large Data Bases*, pages 331–338, Los Angeles, California, September 1988.

[4] DeWitt D., Ghandeharizah S., Schneider S., Hsiao H. & Rasmussen R. (1990) The Gamma Database Machine Project. *IEEE Transactions on Knowledge and Data Engineering*, 2, No. 1:44–61, March 1990.

[5] DeWitt D. & Hsiao H. (1990) Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines. *Proceedings of the IEEE International Conference on Data Engineering*, pages 456–465, Los Angeles, California, February 1990.

[6] Holland M. & Gibson G. (1992) Parity Declustering for Continuous Operation in Redundant Disk Arrays. *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 23–35, October 1992.

[7] Hou R., Menon J. & Patt Y. (1993) Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array. *Proceedings of the Hawaii International Conference on System Sciences*, pages 70–79, 1993.

[8] Hou R. & Patt Y. (1993) Comparing Rebuild Algorithms for Mirrored and RAID5 Disk Arrays. *Proceedings of the International Conference of the ACM SIGMOD*, pages 317–326, Washington D.C., May 1993.

[9] Menon J. & Mattson D. (1992) Comparison of Sparing Alternatives for Disk Arrays. *Proceedings of the 19th International Symposium on Computer Architecture*, pages 318–329, May 1992.

[10] Muntz R. & Lui J. (1990) Performance Analysis of Disk Arrays Under Failure. *Proceedings of the 16th International Conference on Very Large Data Bases*, pages 162–173, Brisbane, Australia, 1990.

[11] Orji C. & Solworth J. (1993) Doubly Distorted Mirrors. *Proceedings of the International Conference of the ACM SIGMOD*, pages 307–316, Washington D.C., May 1993.

[12] Patterson D., Chen P., Gibson G. & Katz R. (1988) A Case for Redundant Arrays of Inexpensive Disks (RAID). *Proceedings of the International Conference of the ACM SIGMOD*, pages 109 – 116, Chicago, Illinois, June 1988.

[13] Polyzois C., Bhide A. & Dias D. (1993) Disk Mirroring with Alternating Deferred Updates. *Proceedings of the 19th International Conference on Very Large Data Bases*, Dublin, Ireland, 1993.

[14] Solworth J. & Orji C. (1990) Write-Only Disk Caches. *Proceedings of the International Conference of the ACM SIGMOD*, pages 123–132, Atlantic City, New Jersey, May 1990.

[15] Solworth J. & Orji C. (1991) Distorted Mirrors. *First International Conference on Parallel and Distributed Information Systems*, pages 10–17, Miami, Florida, December 1991.

[16] Solworth J. & Orji C. (1993) Distorted Mapping Techniques to Improve the Performance of Mirrored Disk Systems. *Distributed and Parallel Databases: An International Journal*, 1(1):81–102, 1993.

[17] Teradata (1985) Teradata Corporation Publication. *DBC/1012 Database Computer System Manual Release 2.0, Document No. C10-0001-02*, November 1985.

Data Consistency in Hard Real-Time Systems

Neil C. Audsley, Alan Burns, Mike F. Richardson and Andy J. Wellings
Real-Time Systems Group
Department of Computer Science
University of York, UK
Phone: +44 1904 432779, Fax: +44 1904 432767
burns@minster.york.ac.uk

Keywords: databases, real-time

Edited by: Jerzy R. Nawrocki
Received: April 21, 1994 **Revised:** March 29, 1995 **Accepted:** April 19, 1995

The incorporation of database technology into hard real-time systems presents an important but difficult challenge to any system designer. Notions of consistency and failure atomicity are attractive for architects of dependable hard real-time applications. Unfortunately, the hard real-time requirements of bounded access times are not easily imposed upon the concurrency control methods usually found in current database models. The model of data consistency presented here uses temporal consistency constraints in order to ensure that data values are sufficiently recent to be usable. Blocking is avoided by imposing the restriction that only one type of transaction can write to a particular data object, and by using a non-blocking access mechanism. Data itself is considered to be either perishable or non-perishable.

1 Introduction

Traditionally, databases are used in systems which either do not operate in real-time, or where quick response is desirable but not strictly necessary. Incorporating databases into hard real-time systems necessarily imposes stricter timing constraints, such that the failure to meet a constraint can have catastrophic consequences. Formally, the correct functioning of the system depends on timely operation as well as on functionally correct operation.

A number of approches have been proposed to improve the performance of conventional databases. Examples include: *multiple versions* [2][17], where older versions of data objects are maintained for read-only transactions; *semantic atomicity* [5], where the semantics of the transactions are exploited; *multilevel atomicity* [15] which is a generalisation of semantic atomicity; and *weak correctness* [18][19], where some transactions are allowed to see the intermediate results of other transactions. Schemes also exist which take advantage of typical database access patterns, so that transactions may proceed more speedily than otherwise, on the expectation that no conflicts with other transactions will arise; these are generally referred to as *optimistic* schemes. If such conflicts do occur, then transactions may have to be aborted and restarted. Such a scheme is described by Kung and Robinson [12]. Other investigation into real-time databases can be found in the references [1] [3] [4] [7] [9] [10] [13] [14] [22].

However, the above methods are characterised by the fact that the performance improvement is statistical in nature, and cannot be guaranteed. Hence, while they may be beneficial in (soft) real-time systems, they are not sufficient for use in hard real-time systems. In this paper, we present a database model which can be used in a class of hard-real time systems, and which takes advantage of the characteristics of those systems. The temporal requirements of the database are formalised, and lead to scheduling constraints on the transactions which access the database.

For the purposes of this paper, we define the term *database* as follows:

Defn (1): *A database is a structured collection* *of data items, where specified consistency constraints between data items must be maintained.*

Note that we do not include associated systems (such as data base management systems) within this definition.

Section 2 of this paper presents some characteristics of conventional and of hard real-time databases. Section 3 then proposes a model for hard real-time databases, and section 4 defines database consistency for this model. An outline of the way in which database consistency requirements can be transformed to transaction timing is given in section 5, followed by a simple example in section 6. Conclusions are presented in section 7.

2 Characteristics of DataBases

The types of systems in which conventional databases are used are generally different from those that contain hard real-time databases. In this section we outline some of these different characteristics.

2.1 Conventional Databases

- Conventional databases have very large numbers of data objects, and the number of data objects may not be known in advance. For example, a banking system may have hundreds of thousands of customer accounts, which may be created and deleted. In principal, the size of the database is unbounded.

- A particular type of transaction may access any (possibly unbounded) subset of the data objects. In the banking example, a transaction which transfers money between accounts may access any pair of accounts. Further, the particular subset may be dependent on parameters passed to the transaction.

- The arrival of transactions cannot, in general, be predicted. Although some banking transactions may occur at regular intervals, such as accrual of interest, the transfer transactions will be invoked as and when customers request them.

- The data held in the database is generally discrete and is changed in a stepwise manner. Further, there is usually at best limited scope for using anything other than the most up-to-date value for a particular data object. For instance, although a customer enquiry as to the state of an account might be satisfied by the value at the close of business the previous day, transfer and interest transactions would not.

- The databases are typically *event* driven, that is, changes are made in response to events in the environment (for example, the occurence of a transfer request).

- The *consistency* of the database, and the *correctness* of the transactions which operate on it, are defined purely over the values of the data objects. Hence, in a banking system, money must neither be lost nor created, but there is no requirement for a transaction to be performed within a particular limited time.

- The problem of scheduling multiple concurrent transactions is one of performing those transactions correctly. While it may be desirable that the transactions are performed as quickly as possible, this desire is secondary to the correctness. Hence, developments in scheduling aim to statistically improve throughput.

2.2 Hard Real-Time Databases

- Hard real-time systems have a relatively small number of data objects, limited perhaps to a few thousand. More importantly, the number is exactly known in advance, or at least is bounded. If this were not so then it would never be possible to guarantee any set of timing requirements.

- Each transaction in a hard real-time system will access a bounded subset of the data objects, with the particular subset being known in advance. For instance, a transaction might read satellite, inertial and sonar position values, to derive a more accurate calculated position.

- The design of the system may require that particular transactions are invoked with some regularity, and the implementation may allow the invocation of others to be made regular. Hence, a transaction which fetches a satellite position may have to execute in synchronisation with the satellite broadcast, while the transaction which combines the positions can be made to run with some sufficient frequency.

- Hard real-time systems are typically embedded as the control element within some system which is essentially continuous in nature. Although a system's calculated ship position might be updated at regular intervals, and hence step through a series of values, it is reflecting a continuous variable. Also, the system may be able to perform adequately provided that the values available for particular data objects are sufficiently fresh, where *sufficiently* is defined by the dynamics of the environment.

- The databases tend to be *state* driven, that is, the values stored in the data objects tend to reflect the state of the environment, for instance, the position of the ship.

- While the notions of consistency and correctness with respect to data values may still be applicable, there is an additional requirement concerning time. Hence, data objects may have to be updated at intervals, or in orders, that are temporally constrained. In the ship system, it might be desirable that the satellite, inertial and sonar position systems are sampled at about the same time, and that the calculated position has bounded freshness.

- The transactions must be scheduled such that both the data and the temporal constraints are met. Although this is a more complicated scheduling problem, especially as the time available in which to make scheduling decisions is also limited (unless the system is completely pre-scheduled), it is sufficient to find any schedule which meets the constraints. Statistical performance improvements beyond this do not necessarily confer any benefit.

2.3 Consequences for Hard Real-Time Databases

The above points have consequences, beneficial or otherwise, for hard real-time systems which embody databases. Note that the term *analysis* is used in the following to refer to the act of determining whether a system is guaranteed to meet all its timing requirements.

The necessity of providing timely response to events in the system's environment makes the scheduling problem much harder. Hence, although the general problem of deciding if a schedule for a conventional database is serialisable is NP-complete [11], there are many algorithms which generate subsets of all serialisable schedules in polynomial time, such as the two-phase locking protocol [6]. However, once timing constraints are added, even the problem of generating any feasible schedule becomes NP-complete [16].

On the other hand, since (a) the number of data objects is bounded with transactions acting on fixed subsets of objects, and (b) the frequency with which transactions are invoked is bounded, it is possible to establish worst-case circumstances on which analysis can be based. The ability to use sufficiently recent values of data objects helps to eliminate interference between components of the system. From a scheduling point of view, this allows a larger number of possible schedules.

It should be noted that there is an important distinction between a database whose correctness includes temporal constraints, and one that does not. In the latter, conventional case, a consistent database remains consistent even if no further transactions are executed. However, a real-time database may become temporally inconsistent if certain transactions **are not** executed.

3 A System Model

To analyse further the requirements of a hard real-time database it is necessary to postulate a general database model which provides applications with a suitable set of abstractions. We are not concerned with a database that has only a peripheral or archival importance. Rather, we focus on the use of a hard real-time database as the main structuring component of the system's architecture.

3.1 The Object Structure of the Database

Consider an embedded (possibly distributed) application that has N distinct streams from sensors or HCI (human-computer interface) inputs; and M streams to actuators (effectors) or HCI outputs. Suppose also that there are P data objects which contain intermediate, or *derived* [1], values used in the calculations involved in transforming the inputs to the outputs. The database holds $N + M + P$ *primary* objects.

Clearly, the input values represent measurements of quantities in the environment. The derived values can also be considered to correspond to quantities in the environment, which are, however, inferred rather than measured directly. In this paper, we use the term *real-world value* to refer to the actual value of the quantity in the environment (some derived data objects may appear to have little direct relation to the real world, for instance *time to reach destination*; however, this is often just a matter of a suitable coordinate transformation). We say that a value v *corresponds* to its real-world value if it is equal to the real-world value within a static and predefined error.

Ultimately, system timing requirements are expressed between inputs and outputs; i.e., a change in the value of an input must result in an output being appropriately updated within some specified response time. The response time is a function of the system being controlled and the precision with which that control must be exercised.

3.2 The Transaction Structure of the Database

All software components interact via the database; they read a subset of data objects and write to some other subset. A simple end-to-end timing requirement may thus involve the reading of a sensor to generate a value S in the database; the transmission of this value to a shadow S' on a different machine; the calculation of some derived value V; and the use of V to calculate an output value A. In this paper, we use the term *transaction* to identify these software components. This is a little different to the meaning used in conventional databases.

In a conventional database, an event in the environment may require the execution of a transaction which updates a number of data objects. Typically, this involves *locking* the objects, making the necessary updates, and then *unlocking* the data objects, at which point the updates become visible to other transactions. The locking is necessary in order to prevent conflicting concurrent update. However, the completion of the transaction does not ordinarily imply any further activity; all changes that must follow on from the event will have been made by the transaction which it triggers.

In the model presented here, a transaction which is triggered by an event in the environment need not update all data objects which are dependant on that event. It may update a subset of those data objects, the update of the remaining data objects being the function of further related transactions. In effect, the event triggers the first of a chain of transactions, where the completion of one transaction triggers the execution of the next. This reduces the potential for contention between transactions over access to data objects, and hence the potential for transactions to become *blocked.*

However, as a result of the transaction model, the hard real-time database may, by comparison to a conventional database, be functionally inconsistent for some interval of time. This is analagous to the notions of *eventual* and *lagging* consistency; a summary of these is given by Sheth *et al* [20]. The issue is addressed the section 4, where database consistency is defined for this model.

3.3 Distributed Databases

If the application is distributed then the database may not be centralised. In this case some primary objects will have corresponding *shadow* objects on remote processors. If the value of a primary object changes, then the change must be propogated to all of its shadow objects. We model the propogation as essentially *producer* driven, with an additional transaction on the same processor as the primary object being used to transmit the value. An alternative is a *consumer* driven model, in which an additional transaction is placed on the same processor as the shadow object; the

[1] This is the term used by Song and Liu [23], and will also be used here.

choice is essentially arbitrary.

However, we choose to model the propogation by using transactions since the propogation requirements can be treated as database consistency constraints, and the timing requirements on these transactions can be derived in the same manner as for any other transaction. This is elaborated in the sections 4 and 5.

3.4 Database Input and Output

As implied above, inputs and outputs are modelled by database objects. An input object is not considered to be written by any normal transaction, but is updated directly from the sensor. Similarly, an output object is not read by any transaction, but is passed directly to an actuator. This corresponds directly to the model described by Song and Liu [23]. Alternatively, inputs could be modelled as transactions which do not read any data objects, and outputs as transactions which do not write any objects. However, for the purpose of generating timing requirements, as described later, the database object input-output model is more convenient.

3.5 Perishable and Non-Perishable Data Objects

At this stage we introduce the notion of *perishable* and *non-perishable* data objects. All data objects in the database will fall into one of these two groups.

A data object is limited to taking values which are drawn from a finite set of possible values; in effect, the value is a state drawn from a finite set of possible states. Ideally, the data object should change states in a manner which follows changes in the corresponding real-world value. However, some real-world quantities will change value sufficiently rapidly that it is impossible or impracticable to keep the data object completely up to date; indeed, in some cases there may be no need for the data object to follow the real-world value exactly, even when it is possible. On the other hand, some will change sufficiently slowly that the data object can follow it, albeit with a slight delay due to the non-zero response time of the system.

We therefore define perishable data objects as follows:

Defn (2): *A real-world value can be mapped to a sequence of states which are its representation in the database. A data object is perishable if it cannot take all those states in time-ordered sequence; some states will inevitably be missed.*

The term *non-perishable* will refer to any data object which is not *perishable* by the above definition. These distinctions will be used later.

4 Maintaining DataBase Consistency

We now consider the issue of specifying temporal consistency in a hard real-time database. We start by defining two properties, namely *absolute* and *relative* temporal consistency. These terms are derived from the *absolute* and *relative coherence* described by Song and Liu [23]. Here, however, the term *consistency* is used in preference to *coherence*, as the properties under discussion are analogous to database *consistency.*

4.1 Absolute Temporal Consistency

In order that a transaction executes meaningfully, the transaction requires both that the values of the input data objects are accurate within some error bounds, and that the value stored in the data object is sufficiently recent. We quantify *sufficiently* by the *absolute* temporal consistency (abbreviated as *ATC*) of the data objects, as follows:

Defn (3): *If a data object X has an ATC a_x then, at any given time t_{now}, the value v_x stored in X must correspond to the real-world interpretation of X at some time in the interval $[t_{now}-a_x, t_{now}]$.*

We note at this point that, if a data object V is updated by a single transaction T, then the responsibility for maintaining the ATC of V falls solely on transaction T. Also, we will use the term *ATC expiry* of a data object to mean that the ATC requirement for that data object no longer holds.

4.2 Relative Temporal Consistency

If a transaction reads a set of data objects, and uses the values to calculate some result, then it may be necessary that the data objects contain values which correspond to the real-world at similar times. For instance, a combination of satellite, sonar and inertial position information may only be meaningful if they have been sampled sufficiently close together. This type of timewise consistency will be referred to as *relative* temporal consistency, and abbreviated as *RTC*. Formally:

Defn (4): *If a set of data objects $S = V_i$ has an RTC r_s, then the values v_i of the V_i must correspond to real-world interpretations of the V_i at times which fall within an interval of length r_s.*

It is likely that different input data objects, to some transaction, will themselves be updated by different transactions. In order to maintain the RTC of those objects, it is necessary to ensure the updating transactions are invoked at appropriate times. We will use the term *RTC set* to refer to a set of data objects over which there is a RTC requirement, and *RTC value set* to a set of values which satisfy the RTC requirement.

It should be noted that a RTC requirement on its own is not particularly useful. If the data objects for which a RTC requirement exists do not have any ATC requirements, then consistency could be maintained by leaving the data objects unchanged once a consistent set of values had been written. Where RTC is particularly useful is when the RTC requirement is significantly shorter than any ATC requirements which the group possesses (for example, sample a set of position sensors within 2ms (the RTC), but only every 2s (the ATC)).

4.3 Functional Consistency

If the update of some data object X requires a subsequent update to another data object Y, in the manner described in section 3, then there is a functional dependency of Y on X. This can be expressed either as $Y = f(X)$, or as $P(Y, X)$, where P is a predicate which must hold for the database to be consistent. The latter form is generally applied to conventional databases.

In our model, the function f may be executed some time after X is updated, and while the old value of Y is still visible. Hence, until f completes, the predicate P may not hold. We will therefore define functional consistency in the following manner:

Defn (5): *Suppose that some predicate P applies to data objects X and Y where the data objects have ATC requirements a_x and a_y, and Y has value v_y. Then X and Y are functionally consistent at time t_{now} if $P(v_y, v_x)$ holds, where v_x is some value held by X in the time interval $[t_{now} - (a_y - a_x), t_{now}]$.*

This is justified as follows. We assume that, in order for v_y to satisfy the ATC for Y at time t_{now}, it must have been calculated from a value v_x which itself corresponded to the real-world within the time interval $[t_{now} - a_y, t_{now}]$ [2]. In the worst case, the value v_x could have been read from X at time $t_w = t_{now} - (a_y - a_x)$ and, at the time it was read, the value v_x could have corresponded to the real-world value of X_i at a time a_x earlier. Hence, the value v_y could in the worst case be based on a value corresponding to the time $t_w - a_x$. This is equal to $t_{now} - a_y$, which is the ATC requirement of Y.

This definition can trivially be generalised to the case where Y is dependant on a set of data objects $S = \{X_i\}$. In this case, there may also be a RTC requirement on the values of the X_i.

4.4 Some Examples

In this section we present some simple examples. We write a_x to indicate the ATC requirement of data object X, and $r_{p,q,..}$ for a RTC requirements over the data objects $P, Q, ...$ All examples are based on an aircraft control system.

(a) Suppose that object B is an input from a barometric altimeter, and H is a derived data object containing altitude, corrected for barometric pressure at sea level. B might, for practical purposes, contain the instantaneous barometric altitude, hence $a_b = 0$; if the value in H must never be more than two seconds out of date, then $a_h = 2$. It is likely that H is perishable for any reasonable resolution and update rate, and hence the transaction which implements $H = f(B)$ must be executed periodically.

(b) Suppose that fuel is contained in two tanks, whose levels are available via derived data objects $T1$ and $T2$. Fuel is pumped between the tanks for trimming purposes. Also, the total fuel remaining must be periodically written to an output data object R for display to the pilot. It may be adequate to update the display every 10 seconds, hence $a_r = 10$, and we might choose $a_{t1} = a_{t2} = 5$.

[2] If the transaction could perform perfect forward interpolation, then it could use earlier values. However, in this paper we assume that forward interpolation is used only to better the ATC requirement, and not to meet it.

However, if a significant amount of fuel can be pumped from one tank to another within a 5 second period, we also require that the values in $T1$ and $T2$ correspond to similar times; hence, we might have $r_{t1,t2} = 1$.

(c) Suppose that input data object D indicates whether the cargo door is open, and that output data object W is a warning display in the cockpit, indicating that the door is open. W is a non-perishable data object, since the cargo door will not be opened and closed with any great frequency. We might have $a_d = 0$ and $a_w = 1$, so that the cockpit display never lags the door state by more than one second. The transaction which implements $W = f(D)$ might be executed periodically; however, sporadic execution with a suitable deadline would entail lower processor loading.

4.5 Transaction Scheduling and the Single-Writer Restriction

In order to maintain the consistency requirements described above, two conditions must be met:

- Each individual transaction must itself satisfy its corresponding predicate P. This is exactly the functional consistency of a conventional database.
- The transactions must be scheduled in a timely manner, such that definitions (3), (4) and (5) are maintained.

Note that, in a conventional database, the ATC requirement is effectively zero. If this is applied to definition (5), then the predicate P must hold over data values in the interval $[t_{now},t_{now}]$, that is, the current values. This is the conventional definition of functional consistency, and commonly leads to *serialisable* [11] schedules. Much database theory is directed to finding schedules which, in the interests of speed, overlap transactions, but which are equivalant to some serial schedule.

The primary obstacle to schedule generation is the need to provide mutual exclusion between transactions which access common data objects. This problem can, however, be alleviated, or even removed, if the single-writer restriction can be applied:

Defn (6): *The data base is single-writer if each data object, other than input data objects, is updated by a single transaction.*

Many databases conform naturally to this definition. Others can easily be transformed, to follow this model, by introducing new transactions that act as single servers for the multiply updated objects.

If this restriction is imposed, then mutual exclusion is only needed for the period during which a transaction writes to a data object. This is the case even for read-recompute-write access, since the data object cannot be changed by any other transaction during the computation. However, mutual exclusion schemes, such as locking [6] can be avoided using the algorithms described by Simpson [21]. Briefly, these allow shared access to data objects in constant time, and work on both uniprocessor and shared-memory multiprocessor systems. The most general of the algorithms has the additional useful property that data cannot be lost as a result of the failure of the writing transaction. Note that it is possible to implement multiple-writer access without locking, as described by Herlihy [8]. However, Herlihy's algorithm is much more complicated and computationally expensive, although still bounded.

The generation of transaction timing requirements from the ATC and RTC requirements is described in the next section, based on the single-writer restriction.

5 Transaction Timing Requirements

In this section, we derive some transaction timing requirements from the database timing requirements.We will use the notation $V(X)$ to denote the value of data object X. Also, $V_t(X)$ denotes the value at time t, and $V_{t_0}^{t_1}(X)$ to denote the set of values taken by data object X in the time interval $[t_0, t_1]$.

5.1 Periodic Transactions

Suppose that a periodic transaction τ_p, with period T_p and deadline D_p, reads a data object X, performs some calculations, and writes data object Y, such that $Y = f(X)$, with the corresponding predicate P. Also, assume that X and Y have ATC requirements a_x and a_y. By the definition of functional consistency (5), we require that, at any time t_{now}:

$$\exists v \in V^{t_{now}}_{t_{now}-(a_y-a_x)}(X) : P(v, V_{t_{now}}(Y))$$

That is, at any given time t_{now}, the value of data object Y must be based on a value stored in data object X in the interval $[t_{now}\text{-}(a_y\text{-}a_x),t_{now}]$.

Now, the maximum time interval between a change in the value stored in data object X, and the subsequent update of data object Y based on the new value, occurs over two successive executions of τ_p, such that τ_p reads X immediately on release of the first execution, and writes Y at the deadline of the second execution; call these times t_0 and $t_0+(T_p+D_p)$. In this case, a change in X immediately after t_0 will not be reflected in Y until $t_0+(T_p+D_p)$. Clearly, just before $t_0+(T_p+D_p)$, $Y \doteq f(V_{t_0}(X))$, so in order that functional consistency is maintained over this interval, we require:

$$T_p + D_p < a_y - a_x$$

This can be trivially generalised to cover the case of transactions which read and write multiple data objects. Suppose that the transaction reads data objects $X = \{X_1, ..., X_n\}$ and writes to data objects $Y = \{Y_1, ..., Y_m\}$. Then, in order that functional consistency is always maintained:

$$\forall i = 1, ..., n, j = 1, ..., m : T_p + D_p < a_{y_j} - a_{x_i}$$

and hence:

$$T_p + D_p < \min_{i,j}(a_{y_j} - a_{x_i}) \qquad (1)$$

5.2 Sporadic Transactions

A similar argument can be applied where $Y = f(X)$ is maintained by a sporadic transaction τ_s with deadline D_s. If we assume that the transaction is released as soon as X is updated, then the longest interval before the subsequent update of Y is D_s, and hence:

$$D_s < a_y - a_x$$

This can be generalised to multiple data objects in exactly the same manner:

$$D_s < \min_{i,j}(a_{y_j} - a_{x_i}) \qquad (2)$$

However, to facilitate any possibility of providing guarenteed schedulability, it is necessary to assign a mimimum inter-arrival time to the transaction τ_s. This corresponds to a minumum interval between changes to data objects.

5.3 RTC Requirements

The presence of a RTC requirement introduces transaction timing requirements over and above those derived in the previous two sections. Specifically, whenever the data objects in a RTC set are read by some transaction, a valid RTC value set must be available.

Consider a RTC group $S = (X_1, ..., X_n)$, where the data objects have ATC requirements a_{x_i}, the RTC requirement is r_s, and where the data objects are written by transactions $\tau_1, ..., \tau_n$. For simplicity, we assume that the transactions are periodic (with periods and deadlines T_i and D_i), and that they have a critical deadline at some time. Finally, assume that if τ_i writes to X_i at time t_i, then the value corresponds to the real world at a time not earlier than $t_i - \delta_i$, where δ_i is a function only of τ_i.

The latest time at which an execution of a transaction may write to data object X is at its deadline. Suppose that transaction τ_i has a deadline at $t = \epsilon_i$. Then, the earliest time at which the write could occur is $t = \epsilon_i - D_i$, and the earliest real-world time for the data value is $t = \epsilon_i - D_i - \delta_i$. Now, the RTC requirement will be met if the total span of these intervals does not exceed r_s. That is:

$$\max_i(\epsilon_i) - \min_i(\epsilon_i - D_i - \delta_i) \leq r_s$$

The window is smallest if the ϵ_i are all equal, and we can, without loss of generality set $\epsilon_i = 0$ for all i. Hence:

$$\max_i(D_i - \delta_i) \leq r_s \qquad (3)$$

If the above relationship holds, then a set of data values for which the RTC requirement holds will be available immediately after time $t = 0$. In general, a new set will be be produced at times $t = n.LCM_i(T_i)$, where n is an integer, and LCM_i is the Least Common Multiple function [3]. This behaviour is analagous to a single periodic process with period $LCM_i(T_i)$, and a deadline equal to $max_i(D_i)$. Hence, the last result above for periodic transactions can be used to ensure the ATC requirements of the data objects in S:

$$LCM_i(T_i) + \max_i(D_i) < \min_i(a_i - \max_j(a_{i,j})) \qquad (4)$$

[3]Intermediate sets may be produced between LCM intervals, however, this is relatively difficult to analyse and is ignored here.

where $a_{i,j}$ is the ATC requirement of the data object X_j read by the transaction τ_i.

If the periods T_i are not all the same (and hence equal to the LCM interval), then some transactions will generate values which are not members of any RTC value set. If this destroys the previous value, then the transactions which read the RTC group must be synchronised with the writing transactions, such that they read the RTC group after a valid RTC value set appears, but before any individual data object is changed. This is only guaranteed in the interval for $t = n.LCM_i(T_i)$ to $t = n.LCM_i(T_i) + min(T_i)$. This requirement places undue restrictions on the transactions which read the RTC set. Further, it means that a sporadic transaction cannot read the RTC group, since it may not execute in suitable synchronisation with the writing transactions.

In order to circumvent these problems, we arrange that data objects which are members of RTC sets hold multiple data values. Each data object X_i holds the $LCM_j(T_j)/T_i + 1$ most recent values written to the data object. Under these circumstances, it is always possible to select a set of value which comprise a RTC value set. Hence, transactions which read the RTC set need not be synchronised.

6 An Illustrative Example

A simple example is now presented in order to illustrate the ideas described above.

6.1 Example System

The system is a fuel and stability system for a boat. Fuel is stored in two tanks, port and starboard, and is used to provide ballast as well as feeding the engines. Fuel is pumped between the two tanks using a ballast pump. The tank levels are monitored via level gauges, and the total fuel remaining is displayed.

The software system is shown in Diagram(1), where circles represent data objects and squares represent tasks. An arrow from a data object to a task indicates that the task uses that data object. Note that those parts of the system concerned with controlling fuel flow to the engines have not been included. Table 1 summarises the appreviations that will be used in the analysis.

Table 1: An Example System

Object	Usage
pli(Port_Level_In)	port level input
sli(Star_Level_In)	starboard level input
pl(Port_Level)	Processed portlevel
sl(Star_Level)	Processed starboard level
fa(Fuel_Avail)	Calculated total fuel remaining
bd(Ballast_Diff)	Required ballast difference
bp(Ballast_Pump)	Ballast pump control output

Process	Usage
plp(Port_Level_Proc)	Process port level input
slp(Star_Level_Proc)	Process starboard level input
fl(Fuel_Level)	Calculates total fuel remaining
ba(Ballast_Adj)	Controls ballast pump&warning

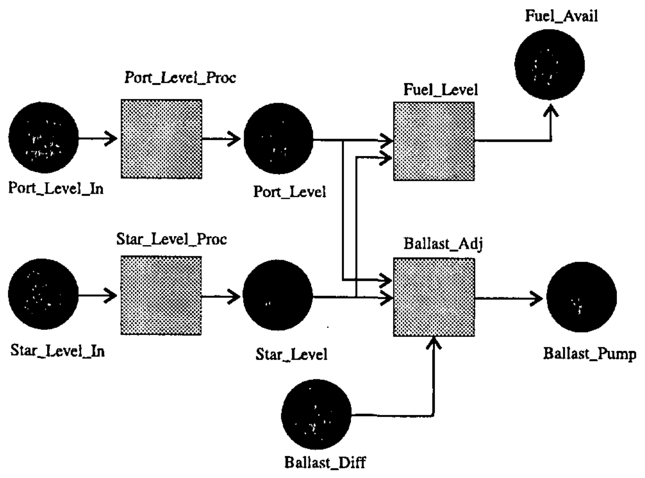


Diagram (1): An Example System

We assume (a) that the ballast pump can pump fuel in either direction at a maximum rate of 10gals/sec; (b) that the fuel available value must be accurate to within ±20gals (at the time to which the displayed value corresponds); (c) that the fuel available value must be not more than 30secs old; (d) that the ballast pump control must be set at least every 10sec; (e) that the ATCs of the input data objects are all zero; and (f) that the age associated with an object when written is the maximum age of the input objects to the writing task (ie., none of the tasks contain any predictive functionality)

(1) $a_{fa} = 30$

(2) $a_{bp} = 10$

(3) $a_{pli} = 0$

(4) $a_{sli} = 0$

(5) $a_{bd} = 0$

6.2 Task Timing Constraints

We now consider the requirements placed on each of the tasks. Also, for brevity, define $R_{task} = T_{task} + D_{task}$.

6.2.1 Fuel_Level

The calculated fuel level value may be up to 30secs out of date; however, it must be correct within 20gals. Given that the ballast pump can move 10gals/sec, the tank levels must be sampled within not more than 2 seconds of each other (between such samples, the level in the tank sampled second could change by 20gals; this ignores fuel pumped to the engines). Hence, from the ATC requirements and Equation (1):

(6) $R_{fl} \leq min(a_{fa} - a_{pl}, a_{fa} - a_{sl})$

(7) $r_{pl,sl} \leq 2$

¿From the RTC requirement, and Equations (3) and (4):

(8) $D_{plp} - \delta_{plp} \leq 2$

(9) $D_{slp} - \delta_{slp} \leq 2$

(10) $LCM(T_{plp}, T_{slp}) + max(D_{plp}, D_{slp}) < min(a_{pl} - a_{pli}, a_{sl} - a_{sli})$

6.2.2 Ballast_Adj

The ballast adjustment constraint is derived simply from the requirements on the Ballast_Pump output. Depending on the precision required of the ballasting operation, there might be a RTC requirement, however this is ignored here for simplicity. By Equation (1):

(11) $R_{ba} \leq a_{bp} - a_{pl}$

(12) $R_{ba} \leq a_{bp} - a_{sl}$

(13) $R_{ba} \leq a_{bp} - a_{db}$

6.2.3 Port_Level_Proc, Star_Level_Proc

In order to maintain the ATC of the Port_Level and Star_Level objects, the relationships below must hold.

(14) $R_{plp} \leq a_{pl} - a_{pli}$

(15) $R_{slp} \leq a_{sl} - a_{sli}$

Further timing requirements on these processes are implied by the earlier RTC requirement.

6.3 Task Timing Characteristics

Any set of timing characteristics for the tasks, which satisfy the requirements given in the previous section, and for which a feasible schedule exists, can be used to successfully implement the system. We will assume that all tasks are periodic.

If we set $a_{pl} = a_{sl} = 5$, then the ATC requirements imply the following:

(16) from (6) $R_{fl} \leq min(30 - 5, 30 - 5) = 25$

(17) from (11,12,13) $R_{ba} \leq min(10-5, 10-5, 10-0) = 5$

(18) from (14) $R_{plp} \leq 5 - 0 = 5$

(19) from (15) $R_{slp} \leq 5 - 0 = 5$

For the RTC requirements, assume that $\delta_{plp} = D_{plp} + a_{pli}$, and similarly for δ_{slp}. This is justified by assumption (f). Hence:

(20) from (8) $D_{plp} - (D_{plp} + a_pli) \leq 2$

(21) from (9)$D_{slp} - (D_{slp} + a_sli) \leq 2$

(22) from (10) $LCM(T_{plp}, T_{slp}) + max(D_{plp}, D_{slp}) < min(5 - 0, 5 - 0) = 5$

If we arbitrarily set $T_i = D_i$, then the following results satisfy 16-22.

(23) from (16) $T_{fl} = 12.5$

(24) from (17) $T_{ba} = 2.5$

(25) from (18) $T_{plp} = 2.5$

(26) from (19) $T_{slp} = 2.5$

6.4 Comments

For a given set of task timing constraints, there may be an arbitrary number of sets of task timing requirements; of these, not all may schedulable. Also, the choice of ATC requirements for derived data is not well defined; some choices may preclude the existance of a schedulable task set. To make the process usable, particularly for realistically sized systems, a practicable mechanism must be found. One possibility is *simulated annealing*; this is described by Tindell [24] in connection with allocating hard real-time tasks to a processors in a distributed system.

7 Conclusions

In this paper a database model is presented that is applicable to at least some hard real-time systems. The essential properties of the model are:

- some objects may have ATC values defined
- some sets of objects may have RTC values defined
- objects are either perishable or non-perishable

If a conventional database is used in a real-time system then two inter-related (NP-complete) problems need to be addressed: concurrency control over access to the database, and the scheduling of systems of tasks to meet timing constraints. The motivation behind the model presented here is that value consistency (which requires concurrency control) is transformed into temporal consistency so that the complete problem becomes one of scheduling only. To this end we have also considered restricting the database so that all objects are written to by exactly one task. This does not necessarily restrict functionality but may require that more objects and tasks are introduced. Again the motivation is to constrain the problem to one of scheduling. The database does not require locking primitives or abort facilities.

Given these properties it is perhaps worthwhile to question whether the resulting formulation is indeed a database. One means of addressing this issue is to consider the four properties often associated with databases, namely: atomicity, consistency, isolation (or serialisability) and durability. The single writer model does provide atomicity; isolation is maintained by the scheduling of accesses to the database; and although durability has not been addressed in detail it is achievable (indeed, by not making use of locking the problem is simplified). It is in the issue of consistency that the proposed model is unconventional. The model supports two forms of temporal consistency; value consistency is only achieved when it can be derived from the temporal consistency relationships. Thus the real-time model uses temporal consistency, whereas the conventional model uses value consistency. The parallels are such that the term database is still appropriate.

One of the motivations for this formulation is as a system modelling technique. Within the correct application domain it is possible to capture all timing requirements as temporal consistencies on a system-wide database. From these requirements the necessary tasks (and additional database objects) can be derived systematically, together with all their temporal attributes (period, deadline, etc). The resulting task set is of a form that is amenable to schedulability analysis. Hence the absolute and relative temporal consistencies can be guaranteed.

References

[1] R. Abbott and H. Garcia-Molina. Scheduling real-time transactions. *ACM SIGMOD Record*, March 1988.

[2] D. Agrawal and S. Sengupta. Modular synchronisation in multiversion databases: Version control and concurrency control. *1989 ACM-SIGMOD Intl. Conf.*, 1989.

[3] A.P. Buchmann, D.R. McCarthy, M. Hsu, and U. Dayal. Time-critical database scheduling: A framework for integrating real-time scheduling and concurrency control. In *Data Engineering Conference*, February 1989.

[4] M.J. Carey, R. Jauhari, and M. Livny. Priority in dbms resource scheduling. In *Proceedings of the 15th VLDB conference*, 1989.

[5] W. Cellary, E. Gelenbe, and T. Morzy. *Concurrency Control in Distributed Database Systems*. 1988.

[6] K.P. Eswaran, J.N. Gray, and R.A. Lorie. The notions of consistency and predicate locks in a database system. *Communications of the ACM*, 19(11), November 1976.

[7] J.R. Haritsa, M.J. Carey, and M. Livny. Dynamic real-time optimistic concurrency control. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, December 1990.

[8] M. Herlihy. Wait-free synchronisation. *ACM Trans. on Prog. Lang. and Systems*, 11(1):124–149, January 1991.

[9] J. Huang and J.A. Stankovic. Experimental evaluation of real-time optimistic concurrency control schemes. In *Proceedings of the 17th International Conference on VLDB*, pages 35–46, 1991.

[10] H.F. Korth, N. Soparkar, and A. Silberschatz. Triggered real-time databases with consistency contraints. In *Proceedings of the 16th VLDB conference*, Brisbane, Australia, August 1990.

[11] H.T. Kung and C.H. Papadimitriou. An optimal theory of concurrency control for databases. *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 111–126, 1979.

[12] H.T. Kung and J.T. Robinson. On optimistic methods of concurrency control. *ACM Trans. on Database Systens*, pages 213–226, June 1981.

[13] K.J. Lin. Consistency issues in real-time database systems. In *Proceedings of the 22nd Hawaii International Conference on System Sciences*, Hawaii, January 1989.

[14] Y. Lin and S.H. Son. Concurrency control in real-time databases by dynamic adjustment of serialization order. In *Proceedings of the 11th IEEE Real-Time Systems Symposium*, December 1990.

[15] N.A. Lynch. Multi-level atomicity - a new correctness criterion for database concurrency control. *ACM Transactions on Data Base Systems Vol 8 No 4*, pages 484–502, Dec 1983.

[16] A.K. Mok. *Fundamental Design Problems of Distributed Systems for Hard Real Time Environments.* PhD Thesis, Laboratory for Computer Science (MIT), MIT/LCS/TR-297, 1983.

[17] C.H. Papadimitriou and P.C. Kanellakis. On concurrency control by multiple versions. *ACM Trans. on Database Systems*, 9(1):89–99, March 1984.

[18] L. Sha, J.P. Lehoczky, and E.D. Jensen. Modular concurrency control and failure recovery. *IEEE Transactions on Computers*, 37(2):146–159, February 1988.

[19] L. Sha, R. Rajkumar, and J.P. Lehoczky. Concurrency control for distributed real-time databases. *SIGMOD Record*, pages 82–96, Vol, 17, No. 1, March 1988.

[20] A. Sheth and M. Rusinkiewicz. Management of interdependant data: Specifying dependency and consistency requirements. In *Proceedings of the Workshop on the Management of Replicated Data*, pages 133–136, November 1990.

[21] H.R. Simpson. Four-slot fully asynchronous communication mechanism. *IEE Proceedings on Computers and Digital Techniques*, 137(1):17–30, January 1990.

[22] S.H. Son, S. Park, and Y. Lin. An integrated real-time locking protocol. In *Proceedings of Data Engineering*, pages 523–534, 1992.

[23] X. Song and J. W. S. Liu. Performance of multiversion concurrency control algorithms in maintaining temporal consistency. Technical report, Dept. of Computer Science, Uni. of Illinois at Urbana-Champaign, Feb. 1990.

[24] K. Tindell, A. Burns, and A.J. Wellings. Allocating real-time tasks (an np-hard problem made easy). *Real Time Systems*, 4(2):145–165, June 1992.

Parallel Gaussian Elimination Algorithms on a Cray Y-MP

Marcin Paprzycki
Department of Mathematics and Computer Science
University of Texas of the Permian Basin
Odessa, TX 79762-0001, USA
Phone: +915 552 2258, Fax: +915 552 2374
paprzycki_m@gusher.pb.utexas.edu

Keywords: performance, parallel Gaussian elimination, Strassen's algorithm

Edited by: Jerzy R. Nawrocki
Received: September 18, 1994 **Revised:** January 24, 1995 **Accepted:** February 27, 1995

Various implementations of Gaussian elimination of dense matrices on an 8-processor Cray Y-MP are discussed. It is shown that when the manufacturer provided BLAS kernels are used the difference in performance between the best blocked implementations and Cray's routine SGETRF is almost negligible. It is shown that for large matrices Strassen's matrix multiplication algorithm can lead to substantial performance gains.

1 Introduction

We shall consider the solution of a system of linear equations

$$Ax = b,$$

where A is an $N \times N$ real dense matrix, using Gaussian elimination with partial pivoting. We are interested in a parallel solution based on the use of *BLAS* [9, 10, 15] primitives and blocked algorithms [1, 3, 6, 8, 12]. Since the release of Unicos 6.0 Cray provides a set of assembly coded parallel *BLAS* kernels as well as some additional routines; one of them is *SGETRF* which performs parallel Gaussian elimination with row interchanges. There exist situations [19, 23] in which Gaussian elimination with column interchanges is necessary, so it becomes important to know how much worse is the performance of the "home made" codes in comparison to *SGETRF*. It was shown [2, 14, 16] that using Strassen's matrix multiplication algorithm in the update step of the Gaussian elimination on a one-processor Cray Y-MP can lead to substantial time savings. We shall presently address the possibility of using parallel Strassen's matrix multiplication algorithm in the update step of parallel Gaussian elimination.

2 Level 3 BLAS based algorithms – implementation details

We compared the performance of different versions of Gaussian elimination on an 8-processor Cray Y-MP using manufacturer provided *BLAS* kernels. Since we used FORTRAN as the programming language we considered only three (column oriented) implementations out of the six possible versions of Gaussian elimination. We investigated the performance of *DOT*, *GAXPY* and *SAXPY* versions of Gaussian elimination as described in [6, 11, 16]. Each consists of three operations performed on submatrices (blocks) of A: (1) the solution of a linear system for the multiple right hand sides (level 3 *BLAS* routine *_TRSM*), (2) the block update (level 3 *BLAS* routine *_GEMM*), and (3) the decomposition of a block of columns. Since Cray provides assembly coded routines, *STRSM* and *SGEMM*, only the last operation needs to be implemented independently.

To decompose a block of columns each of the three unblocked versions of the column oriented elimination can be used. In [16] it was shown that for the one-processor Cray Y-MP, for the long blocks of columns, the unblocked *GAXPY* and *DOT* versions are the most efficient. At the same time for larger numbers of processors the unblocked *GAXPY* performs poorly [20]. Our experiments suggest that for multiple processors and for blocks of columns shorter than 1500 the unblocked *SAXPY* based routine is more efficient than the unblocked *DOT* whereas as the number of rows in the block increases the *DOT* routine seems to become more efficient. For that reason we have selected the *SAXPY* based decomposer for the blocked codes. To confirm this choice we have performed some experiments with the implementations using the unblocked *DOT* version of the decomposition step. For matrices of sizes up to 2600 (which was the largest size we have experimented with), all blocked codes with the *SAXPY* based decomposer have outperformed the codes with the *DOT* based decomposer. It should be added as the matrix size increased that the difference in performance tended to decrease. as the matrix size increases the percent of the time spent in the decomposition step relative to the time spent in other steps decreases [22]; thus for very large matrices, even if the *DOT* based decomposer would be slightly more efficient its effect would be negligible. It is only for smaller matrices that the efficiency of a decomposer really matters and in those situations *SAXPY* is the fastest.

3 Numerical results

Since in [16, 20] it was shown that when the assembly coded *BLAS* kernels are used the difference between the performance of the best versions of blocked and unblocked codes is almost negligible, in the first series of experiments we have investigated the parallel performance of the level 2 *BLAS* based codes. Figure 1 summarizes the 8-processor performance.

The effect of the system bus being saturated with data communication can be clearly observed. The *SAXPY* variant is competitive only for very small matrices, whereas the *DOT* version is the leader for medium size matrices. Interestingly, the *GAXPY* variant, which is slower than the others for most of the matrix sizes, becomes the performance leader for very large matrices. This gain, however, is only relative to the unsatisfactory performance of the other variants. A severe effect of memory bank conflicts can be observed

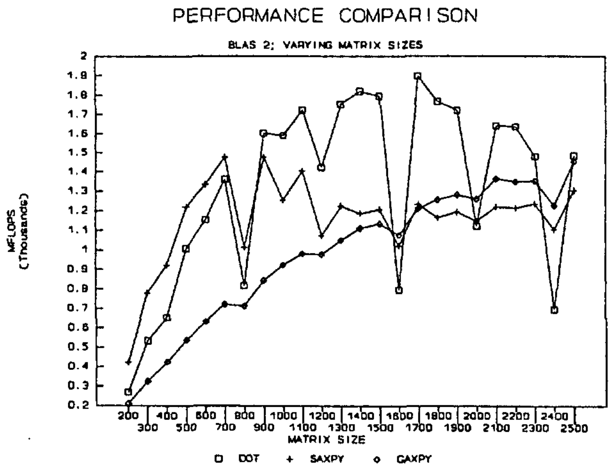


Figure 1: Comparison between the level 2 *BLAS* based codes; 8 processors; results in MFlops.

for matrices of size 800, 1600 and 2400.

The second series of experiments was designed to investigate the effect of change in the blocksize on the performance of the blocked algorithms. There are some attempts by researchers from the *LAPACK* project [4] to provide a theoretical basis for an automatic blocksize selection. For the time being, however, only the method of trial and error is available. Table 1 compares the performance of all three versions of the blocked algorithm for a variety of blocksizes when all 8 processors are used; the matrix size is 1600.

Blocksize	SAXPY	DOT	GAXPY
64	1.559	1.541	1.689
128	1.551	1.549	1.580
192	1.530	1.549	1.559
256	1.545	1.531	1.564
320	1.570	1.546	1.546
384	1.645	1.612	1.609

Table 1: Blocksize effect on the performance; 8 processors; time in seconds.

The best performance is obtained for blocksizes 192 and 256. In general, for large matrices the performance gain from blocking (over level 2 *BLAS* based codes) was approximately 30%; the performance gain from using the best blocksize was additional 3-4%. These results conform to what was presented in [20] for smaller matrix sizes. At the same time our experiments indicate that (1) as the number of processors increases (for a fixed matrix size) the optimal blocksize increases and (2) as the matrix size increases (for a fixed number of processors) the optimal blocksize decreases. These results are illustrated in Tables 2 and 3.

This may be explained by the fact that as the number of processors increases greater amounts of data need to be transferred to keep them busy, and so a larger blocksize is required. As the matrix size increases a smaller number of columns needs to be transferred to provide the fixed number of processors with the same amount of data to work with. This effect is, of course, mediated by the communication capacity of the system bus. Thus, contrary to what is common in practice, for blocked algorithms performed on parallel computers, there seems to exists an *optimal blocksize per processor* (rather than a *fixed* optimal blocksize). Unfortunately the optimal blocksize per processor will vary from architecture to architecture and needs to be established experimentally.

The final set of experiments was directed at the possibility of using Strassen's matrix multiplication algorithm in the update step of parallel Gaussian elimination. In 1969, Strassen [21] showed that it is possible to multiply two matrices of sizes $N \times N$ in less than $4.7N^{log_2 7}$ arithmetic operations. Since $log_2 7 \approx 2.807 < 3$, this method improves asymptotically over the standard matrix multiplication algorithm which requires $O(N^3)$ operations. Strassen's algorithm is based on the recursive division of matrices into blocks and subsequent performance of block additions, subtractions and multiplications.

When implemented, Strassen's algorithm involves certain trade-offs. The first one is between a gain in speed and an increase in required storage. In Cray's implementation (routine *_GEMMS*), the additional work array of size $2.34 * N^2$ is required [5]. The second is between the depth of recursion and parallelization. The deeper the level of recursion the greater is the one-processor performance gain. At the same time, however, in Cray's implementation where recursion is applied only up to block of size of approximately 64 (and parallel *SGEMM* is used to calculate the result), the effects of recursion are reduced [17].

There is one further important consideration concerning the stability properties of Strassen's algorithm since they are less favorable than those of the conventional matrix multiplication algorithm [7, 14]. In [14] Higham showed that for square matrices (where $N = 2^k$), if $\hat{C}$ is the computed approximation to C ($\hat{C} = AB + \Delta C$), then

$$||\Delta C|| \le c(N,N,N)u||A||||B|| + O(u^2)$$

where u is a unit roundoff, and

$$c(N,N,N) = (\tfrac{N}{2^k})^{log_2 12}((2^k)^2 + 5 * 2^k) - 5N.$$

(For non-square matrices the function c becomes more complicated, but the overall result remains the same.) Observe that the error bound for standard matrix multiplication is $Nu||A||||B|| + O(u^2)$. Therefore, the expected error growth should not be too serious in computational practice. This conclusion is also backed up by the results of our experiments.

Figure 2 presents the results of our experiments with Strassen's matrix multiplication in the update step of the Gaussian elimination for matrix sizes N = 600, ..., 2600. Following [2], the results are presented in calculated MFlops. (This allows for a clearer expression of the performance gains of the new algorithm even though Strassen's algorithm uses fewer operations.) We have decided to present only the *SAXPY* and *DOT* based implementations since when multiple processors were used (for the Strassen as well as non-Strassen based implementations) the *GAXPY* variant was much slower than the other two for all matrix sizes (this is primarily caused by the fact that the matrices involved in the update step are long and "thin"; see also [20]). We also present the performance of the Cray provided routine *SGETRF* and the practical peak performance. For both Strassen as well as non-Strassen implementations the blocksize 265 was used.

A number of observations can be made. There is no additional gain from using the Cray provided routine *SGETRF*. This is very important whenever Gaussian elimination with column interchanges needs to be employed (e.g. in algorithms using alternate row and column elimination strategy [19, 23]). In [17], it was argued that the practical peak performance for the 8-processor Cray

Matrix size	200	600	1000	1400	1800	2200	2600
Optimal blocksize	512	320	256	320	256	256	192

Table 2: Optimal blocksize for a fixed number of processors (8) and increasing matrix size.

Number of Processors	1	2	3	4	5	6	7	8
Optimal blocksize	64	256	256	256	384	256	384	256

Table 3: Optimal blocksize for a fixed matrix size (2500) and increasing number of processors.

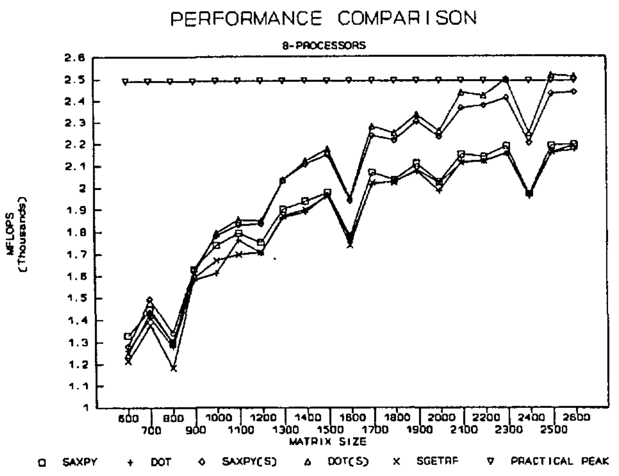


Figure 2: Performance comparison between Strassen and non-Strassen-based codes; 8 processors; results in MFlops; '(S)' marks codes using Strassen's update.

Y-MP is approximately 2490 MFlops. Therefore the blocked (non-Strassen) Gaussian elimination algorithms reach approximately 88% of this practical peak for large matrices. The time reduction obtained due to Strassen's matrix multiplication used in the update step of Gaussian elimination increases with the matrix size and reaches 13% for matrices of size 2600. Using Strassen's matrix multiplication is advantageous only for matrices of sizes larger than 1000. Approximate minimal matrix sizes for which the Strassen-based Gaussian elimination outperforms the non-Strassen versions for $1 - 8$ processors are summarized in Table 4.

The results in Table 4 indicate clearly that the cross-over point migrates toward larger matrix sizes as the number of processors increases. Finally, the effect of memory bank conflicts is much smaller for the blocked codes than it is for the unblocked ones. This can be attributed to a more efficient data transmission pattern.

4 Conclusions

In the paper we have studied the performance characteristics of the *BLAS* based blocked Gaussian elimination for large dense matrices on an 8-processor Cray Y-MP. We have shown that the "home made" implementations of *SAXPY* and *DOT* versions of Gaussian elimination perform very closely to the Cray provided routine *SGETRF*. We have suggested that the standard notion of blocksize used for single processor blocked algorithms should be replaced by the notion of blocksize per processor for shared memory multiprocessor systems. We have also shown that if stability is not a serious consideration then, for large matrices, using parallel Strassen's matrix multiplication algorithm in the update step leads to substantial time savings.

5 Acknowledgement

The computer time grant from Cray Research Inc., is kindly acknowledged. The author wishes to express his gratitude to Chris Hempel (Cray Research Inc., Austin) for his help in running the experiments as well as in obtaining the computer time grant, anonymous referees for helping to prepare a better paper and to Katarzyna Paprzycka for the help with English.

References

[1] Anderson, E., Dongarra, J., Evaluating Block Algorithm Variants in LAPACK, in: Dongarra, J., Messina, P., Sorensen, D.C., Voigt, R.G., (eds.) *Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1989

Number of Processors	1	2	3	4	5	6	7	8
Cross-over point	400	500	600	700	800	900	1000	1100

Table 4: Approximate cross-over point between the Strassen-based and non-Strassen Gaussian elimination for increasing number of processors.

[2] Bailey, D.H., Lee, K., Simon, H.D., Using Strassen's Algorithm to Accelerate the Solution of Linear Systems, *The Journal of Supercomputing*, 4, 1990, 357-371

[3] Bischof, C.H., Fundamental Linear Algebra Computations on High-Performance Computers, Technical Report MCS-P150-0490, Argonne National Laboratory, 1990

[4] Bischof, C.H., Lacroute, P.G., An Adaptive Blocking Strategy for Matrix Factorization, Technical Report MCS- P151-0490, Argonne National Laboratory, 1990

[5] Cray Research, Inc., Math and Scientific Reference Manual, SR-2081 5.0.

[6] Dayde, M.J., Duff, I.S., Level 3 BLAS in LU Factorization on Cray-2, ETA-10P and IBM 3090-200/VF, *The International Journal of Supercomputer Applications*, 3 (2), 1989, 40-70

[7] Demmel, J.W., Higham, N.J., Stability of Block Algorithms with Fast Level 3 BLAS, Numerical Analysis Report No. 188, University of Manchester, 1991

[8] Demmel, J.W., Higham, N.J., Schreiber, R.S., Block LU Factorization, Numerical Analysis Report No. 207, University of Manchester, 1992

[9] Dongarra, J.J., Du Croz, J., Duff, I., Hammarling, S., A Set of Level 3 Basic Linear Algebra Subprograms, Technical Report ANL-MCS-TM57, Argonne National Laboratory, 1988

[10] Dongarra, J.J., Du Croz, J., Hammarling, S., and Hanson, R.J., An Extended Set of FORTRAN Basic Linear Algebra Subprograms, *ACM Transactions on Mathematical Software*, 14 (1), 1988, 1-17

[11] Dongarra, J.J., Gustavson, F.G., and Karp, A., Implementing Linear Algebra Algorithms for Dense Matrices on a Vector Pipeline Machine, *SIAM Review*, 26, 1984, 91-112

[12] Gallivan, K., Jalby, W., Meier, U., Sameh, A., Impact of Hierarchical Memory Systems on Linear Algebra Algorithm Design, *The International Journal of Supercomputer Applications*, 2 (1), 1988, 12-46

[13] Gallivan, K., Plemmons, J.R., Sameh, H.A., Parallel Algorithms for Dense Linear Algebra Computations, *SIAM Review*, 32 (1), 1990, 54-135

[14] Higham, N.J., Exploiting Fast Matrix Multiplication Within the Level 3 BLAS, *ACM Transactions on Mathematical Software*, 16 (4), 1990, 352-368

[15] Lawson, C.L., Hanson, R.J., Kincaid, D.R., and Krogh, F.T., Basic Linear Algebra Subprograms for FORTRAN Usage, *ACM Transactions on Mathematical Software*, 5 (3), 1979, 306-323

[16] Paprzycki, M., Comparison of Gaussian Elimination Algorithms on a Cray Y-MP, *Linear Algebra and its Applications*, 172, 1992, 57-69

[17] Paprzycki, M., Parallel Matrix Multiplication – Can We Learn Anything New?, *CHPC Newsletter*, 7 (4), 1992, 55-59

[18] Paprzycki, M., Cyphers, C., Multiplying Matrices on the Cray – Practical Considerations, *CHPC Newsletter*, 6 (6), 1991, 77-82

[19] Paprzycki, M., Gladwell, I., Using Level 3 BLAS to Solve Almost Block Diagonal Systems, in: Dongarra, J., Kennedy, K., Messina, P., Sorensen, D.C., Voigt, R.V., (eds.), *Proceedings of The Fifth SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1992, 52-62

[20] Sheikh, Q., Performance of Block Matrix Factorization Algorithms and LAPACK on CRAY Y-MP and CRAY 2, in: Dongarra, J., Messina, P., Sorensen, D.C., Voigt, R.G., (eds.) *Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1989

[21] Strassen, V., Gaussian Elimination is not Optimal, *Numerical Mathematics*, 13, 1969, 354-356

[22] van de Geijn, R.A., LINPACK Benchmark on the Intel Touchstone GAMMA and DELTA Machines, Preliminary Report, University of Texas, 1991

[23] Varah, J.M., Alternate Row and Column Elimination for Solving Certain Linear Systems, *SIAM Journal on Numerical Analysis*, 13, 1976, 71-75

Multi-Grain Rendezvous

Stanislaw Chrobot
Kuwait University, Department of Mathematics,
P. O. Box 5969 Safat, 13060 Kuwait
chrobot@mcc.sci.kuniv.edu.kw

Keywords: distributed programming, message passing, rendezvous, interrupt

Edited by: Pavol Navrat

Received: November 21, 1994 **Revised:** March, 15, 1995 **Accepted:** May, 30, 1995

In many distributed languages, the select statement introduces a dynamic selection mode for a sequential process as a mode in which the successor for an operation is selected at run-time from a set of operations defined by the process program. It is an alternative to the static selection mode in which only one successor for an operation is defined by the program. The dynamic selection mode is used to accept the remotely invoked operations, called transactions in Concurrent C. The transactions themselves, however, execute in the static selection mode. This fact is a source of poor expressiveness and poor efficiency of the transactions. To improve the expressiveness and efficiency, we suggest that the transaction are subdivided into grains, and are not only accepted in the dynamic selection mode, but are executed in this mode as well. We also suggest that the implementation of the dynamic selection mode should be based on the interrupt feature.

1 Introduction

The [1] basic property of a sequential process is that it executes one operation at a time. In other words, each operation, except the last one, is succeeded by exactly one operation. In programming languages, there exist two modes of selection of a successor operation in a sequential process: static and dynamic selection mode. In the *static selection mode*, exactly one successor is defined by the text of the program and its data; in the *dynamic selection mode*, more than one successor can be defined by the text of the program and its data. The decision which of them is actually executed, is taken at the process execution time.

The first language construct that supported the dynamic selection mode was proposed by Dijkstra under the name of *guarded command* (Dijkstra 1976) as an element of a sequential language. To illustrate this command let us compare two instructions which find the maximum m of two values: x and y. Classical *if-then-else* defines one of the assignment statements in a static way:

```
if x >= y
then m:= x else m:= y;
```

Guarded *if-fi* introduces the dynamic selection mode. The statement does not define which of the assignments is executed when x = y:

```
if x >= y -> m:= x
|| y >= x -> m:= y fi;
```

Later on, input/output operations were admitted in the guards. In this form the guarded command was moved by Hoare to his CSP language (Hoare 1978), and was adopted as an essential element of almost all the well-known distributed programming languages like DP (Brinch Hansen 1978), occam (INMOS 1988), Ada (United 1985), SR (Andrews at al. 1988), and Concurrent C (Gehani at al. 1986).

For the concurrent computation analysis purposes, it is convenient to discuss the selection modes at the abstraction level of so called grain operations. The *grain* is a part of a sequential process which does not contain any synchronisation or communication delay operation and is separated from other grains by such delay operations. In the *static grain selection mode*, one successor grain is

[1]This work is supported by Kuwait University under the grant number SM 087

defined for each grain; in the *dynamic grain selection mode*, more than one successor grain can be defined for each grain by the process program.

In Concurrent C (and other rendezvous based languages), a server process enters the dynamic grain selection mode by executing the *select statement*. This statement defines a set of transactions acceptable for the server. (We will use the term *transaction* for the operations accepted by the *select* statement in other languages too). Once the transaction is accepted, the server abandons the dynamic mode to execute the transaction in the static selection mode. Each transaction can nest other transactions. The server, however, waits until the nested transaction is completed. After the completion, only one operation, the one following the nesting call statement, is possible for execution. The server returns back to the dynamic mode by executing another select statement. This switching back and forth between the static and the dynamic selection modes makes both the *expressiveness* and *efficiency* of the server much purer comparing with those of the monitor.

Let us analyse the expressiveness first. A monitor procedure can be delayed at any point of its execution in a non-blocking way by the mechanism called condition. There are two kinds of monitor conditions: Hoare's condition(Hoare 1974) with immediate resumption, and Mesa's condition (Mitchel 1978) with broadcast resumption. Due to the condition mechanism, more than one monitor call can be executed in parallel, and the synchronisation conditions can be expressed easily in the monitor procedure body.

A synchronous transaction, once started, has to be executed to completion. A transaction can be delayed in a non-blocking way only before it starts. Moreover, its synchronisation condition is evaluated on special rights, as a part of the select statement rather then as a part of the transaction itself. As a result, there can be at most one transaction in progress at a time, and special lexical entities (like *guards* or *suchthat clauses*) must be used to express synchronisation conditions.

The low efficiency of the server has its source in two kinds of overhead: the process management overhead and the guard evaluation overhead. To analyse the former one let us assume that the server is waiting for a client. When the client's message arrives, the client is blocked, and the server is resumed. *This requires a context switch and process rescheduling.* When the transaction is completed, the server is blocked, and the client is resumed. *This requires another context switch and process rescheduling.* It is worth stressing that this kind of overhead is extremely expensive on RISC processors where the context switch needs reloading the register windows.

Let us note that such overhead does not appear in monitor. If a process calls a monitor when its critical region is not occupied, neither the context switch nor the process rescheduling is needed.

Another source of overhead is the guard evaluation. The server has to re-evaluate the guards every time the select statement is started. This leads to a semi-busy form of waiting. In monitor, the synchronisation conditions are evaluated when there is a need, before a potential waiting or signalling.

In this paper we suggest that both the expressiveness and the efficiency of the server can be significantly improved, when the transactions are not only accepted in the dynamic selection mode but when they are executed in this mode as well. We propose a *multi-grain transaction* as a collection of grains. Such transaction can be delayed at any time of its execution, and its synchronisation condition can be coded as part of its body. At the same time both the process management and the guard evaluation overhead is significantly reduced.

As a result, the expressiveness and the process management overhead of the two modules: monitor and the server become comparable. These modules can be considered duals to each other unless other aspects of the processing, like load imbalance or communication overhead are taken into account.

In this paper we point out also that such a dualism can be found not only at the level of the monitor's and the server's primitives, but in the internal structures of these primitives as well. We suggest that the multi-grain transaction can be built on top of the two fundamental low-level concepts: *interrupt* and *needle* in a similar way as the monitor concept can be built on top of the *spin-lock* and the *coroutine* concepts (Wirth 1985), (Chrobot 1994).

Having the two tools: monitor and server comparable as far as their expressiveness and process management overhead is concerned we can freely choose between them to minimise other kinds of overhead like *communication overhead* and *load imbalance overhead* (LeBlanc et al. 1992).

The *communication overhead* in shared variable paradigm is, in general, bigger than that in the message passing paradigm. In the shared variable paradigm, each process can access each variable, but because of cache misses and non-local memory accesses, some data references are more expensive than others. In the message passing model, each process resides within its own address space. Data is associated with a process in a static way, thus, all the references are to local memory.

The *load imbalance overhead* occurs whenever a processor is idle when there is still work to be done. The load imbalance is almost completely eliminated in shared variable paradigm. A central *ready queue* is accessible evenly to all processors. Thus, no process can stay idle when the *ready queue* is not empty. In the message passing paradigm, each processor has its own ready queue. Thus, it is very likely that one processor is idle while the *ready queue* of another processor is not empty.

For presentation purposes, we have implanted our multi-grain transaction concept into the Concurrent C language. In Section 2, we present a description of the language constructs related to this concept. In the next Sections, we discuss some design and implementation issues of the multi-grain transaction. The last two Sections contain the review of the related works and conclusions.

2 The Multi-Grain Transaction Concept

2.1 Process Specification and Definition

To have a proper context, let us recall that in Concurrent C (Gehani at al. 1986), a *process definition* consists of two parts: a *type* (or *specification*) and a *body* (or *implementation*). An instant of a process definition is called a *process*. It runs in parallel with other processes. The *process type* specifies parameters sent to a process of this type and transactions which are accessible to other processes. Processes use transactions for their synchronisation and communication purposes.

```
process spec <process-type-name>
(<parameter-declaration>).
{<transaction-declarations>}
```

A *transaction* is an operation defined as a part of a process called *server process* and invoked by another process called *client process*. The transaction can access parameters and can return a value. A transaction declaration is like C function prototype, except that it is preceded by the keyword trans, and that the parameter names are explicitly specified.

```
trans <return-type> <tname>
(<parameter-declaration>);
```

A transaction name is only meaningful in the context of a specific process type; different process types can use the same transaction name with different meanings.

A new process is created and activated using the *create operator*

```
create <process-type-name>
(actual-parameters).
```

It returns a process value identifying the created process. The process value can be stored in a process variable of the same type as the process type and/or can be sent to other process as a parameter.

A client process can call the transactions using the *transaction call* expression:

```
<process-value>.<transaction-name>
(<actual-parameters>).
```

Like C function arguments, the transaction arguments are passed by value; the call expression has the type returned by the transaction. The transaction is executed jointly by the client calling the transaction and the server accepting it. Both the processes have to be synchronised before starting the transaction. The actual parameters of the transaction are passed to it by the client process. The transaction can access, as its external variables, the variables accessible for the server process. The mode of executing the transaction is called *rendezvous*. Both the client and the server can continue separately when the transaction has been completed. The transaction call expression can not appear either in a grain body or in a grain actual parameter list (see Section 2.2).

A *process body* contains the declarations and the statements that are executed by a process of this type. The process body attributes are not visible to other processes.

```
process body <process-type-name>
(<process-parameter-nam>)
{<compound statement>}.
```

Each process of that type will get its own set of automatic variables defined in the *compound statement* (if any). Process parameters are used in its body just as function parameters are used in the function body.

2.2 Single-Grain Transaction

A transaction is executed either as one grain operation or as a chain of grain operations (see Section 2.3). We will call such a transaction *single-* or *multi-grain transaction* correspondingly. Like a transaction, a grain can access parameters and can return a value. A grain declaration is like a transaction declaration, except that it is preceded by the keyword grain.

```
grain <return-type> <grain-name>
(<parameter-declaration>);
```

The first grain of the transaction takes its name and parameters from this transaction. If the transaction and its first grain have the same return types, the grain need not to be declared; the transaction declaration is considered the declaration of this grain. Other grains have to be declared in the process body.

A flag called *mask* is associated with each grain. Its value represents the state of the grain: *masked* or *unmasked*. The initial value of each mask is unmasked. Two statements:

```
mask <grain-name>;
```

and

```
unmask <grain-name>;
```

assign the value *masked* or *unmasked* to the mask associated with the grain specified by *grain-name* respectively.

A grain operation is defined by the accept statement:

```
accept <grain-name>
(<parameter-names>)
[{<compound statement>}
[<other statements>]]
```

The *compound statement* and *other statements* are optional. They define the grain body. The parameters of a grain are accessible in its compound statement only. An accept statement can only be used in a select statement.

The *select statement* enters the dynamic grain selection mode, and defines and opens for acceptance one or more grain operations

```
select
{<accept-statement>
 or    ...  or
 <accept-statement>}
```

A grain is acceptable if it has been invoked, opened, and unmasked. A grain invocations not accepted yet by the server are called *pending invocations.* If there are no acceptable pending invocations, the server blocks in the select statement; otherwise, it selects one of the acceptable pending invocations in an arbitrary way.

The grains of a given process are executed in a mutual exclusion mode, at most one grain at a time. The grain terminates when its execution has reached the end of its body. After the termination of the accepted grain, the server continues executing the select statement. All the grains defined by this statement remain opened, and can be accepted by the server until the *close_select statement* of the form

```
close_select;
```

is executed by a grain. This statement moves the server to the static grain selection mode. When the grain calling this statement is terminated, the server closes all the grains opened by the running select statement, terminates this select statement and executes the statement following it.

A transaction terminates when a grain belonging to this transaction's chain reaches the end of its compound statement or when it executes a treturn statement. After the transaction termination both the server and the client calling this transaction continue their jobs separately.

The *treturn statement* has the following form:

```
treturn [<expression>];
```

The value of the *treturn expression* is returned to the calling process and the current transaction is terminated. The type of the expression must conform to the return type of the corresponding transaction. If the return type is void then the expression is omitted and no value is returned to the calling process.

Example 1:
The *communication_buffer* Server

Let us analyse a distributed version of the *communication_buffer* used by two processes: producer and consumer. The buffer is specified as a server process with two transactions: *put* and *get*. The acceptance conditions for the grains are controlled by their masks.

The two client transactions are single-grain transactions. The put(c) grain is unmasked when
$(0 <= n \&\& n < max)$;
the get() transaction is unmasked when
$(0 < n \&\& n <= max)$.
These are the synchronisation conditions for the grains. They assert the buffer's invariant:
$(0 <= n <= max)$.

The *get* transaction issues the *close_select* statement when the character to be returned to the consumer is EOT. After the grain is completed, the select statement is terminated. The server completes its job by freeing the memory allocated for the buffer.

```
process spec
  communication_buffer (int max)
{ trans void put(char c);
  trans char get();
};
process body
  communication_buffer (max)
{ char *buf;
  int n = in = out = 0;
  buf = malloc(max); mask get;
  select
  { accept put(c)
    { buf[in] = c;}
      n++; in = (in+1) % max;
      if (n==1)  unmask get;
      if (n==max)  mask put;
    or accept get()
    { treturn  buf[out];}n--;
      out = (out+1) % max;
      if (n==0)  mask get;
      if (n==max-1) unmask put;
      if (c==EOT)  close_select;
  } }
  free((char *) buf);
}
```

The transactions defined by the server are single-grain transactions, thus no additional grains are declared in the process body. It is important that the server stays in the select statement after executing a grain. There is no loop at the program level. This property can be a source of significant program efficiency gains. Let us compare our algorithm with the one designed in genuine Concurrent C(Gehani at al. 1986).

```
process body
  communication_buffer (max)
{ char *buf;
  int n = in = out = 0;
  buf = malloc(max);
  for ( ; ; )
  select
  { (n < max):  accept put(c)
    { buf[in] = c; }
    n++; in = (in + 1) % max;
  or ( n > 0):  accept get()
    { treturn  buf[out];}
    if (buf[out] == EOT)  break;
    n--; out = (out + 1) % max;
  } }
  free((char *) buf);
 }
```

The classic definition of the rendezvous (Ada (United 1985), Concurrent C (Gehani at al. 1986) and others) assumes that the transaction is always executed by the server process. Thus, with the low frequency of the transaction calls, the server is resumed before, and delayed after executing each transaction. It is a source of two kinds of overhead. One is related to the context switching and the other one related to the process rescheduling. On some computers both operations are quite time consuming (e.g. context switching on the Sun SPARC computers).

Our definition does not specify that the transaction has to be executed by the server process; we say that the transaction is executed *jointly* by the client and the server. As a result, different implementations for such a rendezvous are possible.

When the server and the receiver share the same address space, and there is no locality problem, the grain can be executed in the context either of the server process or of the client process, depending on which of them comes to the rendezvous later. When the server is delayed in the select statement, there is no must for it to be resumed to execute the grain just invoked by the client. The grain can be executed in the context of the calling client process as well.

When the client and the server run in different address spaces, the grain invocation can be implemented by sending an interrupt signal to the server address space. If the server process is blocked in the select statement, the grain can be executed in the context of the interrupted process.

In both cases, the server process remains blocked. As a result, when there is low congestion, the transactions can be executed with no context switching and no process rescheduling.

In Concurrent C (and other rendezvous based languages), the guards are used to delay the acceptance of the transaction, when its synchronisation condition is not satisfied. It leads, however, to a semi-busy form of waiting; the server has to re- evaluate the guard every time the select statement is started. Introducing the masks eliminates this semi-busy waiting overhead. The guard expression is evaluated only when a state variable involved in the guard expression has changed its value.

2.3 Chaining the Transaction Grains

As it was mentioned above, the grains can be chained to create a multi-grain transaction. The statement of the form

```
chain <grain-name>
(<actual-parameter>);
```

terminates the calling grain's compound statement and invokes the grain specified by the *grain-name* with the parameters specified by the *actual-parameter*. The type of the actual parameter has to conform to the return type of the calling grain and the parameter type of the invoked grain. The invoked grain joins the *grain chain* which represents a multi-grain transaction invoked by a client. The grain chain of a given transaction is originated by the first grain of this transaction. The chain statement can only appear in the compound statement of the grain statement.

Example 2:
The *double-resource* Allocator

A set of client processes use two resources. To facilitate the deadlock prevention, both the resources can be allocated "in one go". Thus, the server provides the clients with three transactions to allocate the resources either separately or jointly and another two transactions to free each resource separately. The resources are numbered with 0 and 1. In case the double allocation is needed, the *allocate_1* grain chains the *allocate_both* grain.

```
 process spec double_resource(void)
{  trans void allocate_0(void);
   trans void allocate_1(void);
   trans void allocate_both(void);
   trans void free_0(void);
   trans void free_1(void);
}
process body double_resource()
{ int free0 = 1, free1 =1;
/* initially all the grains
   are unmasked */
  select
  { accept allocate_0()
    { treturn;}free0--;
      mask allocate_0;
      mask allocate_both;
    or accept allocate_1()
    { treturn;}
      free1--;  mask allocate_1;
    or accept allocate_both()
    { chain allocate_1;}
      free0--;
      mask allocate_0;
      mask allocate_both;
    or accept free_0()
    { treturn;}free0 ++;
      unmask allocate_0;
      unmask allocate_both;
    or accept free_1();
    { treturn;} free1 ++;
      unmask allocate_1;
   } }
```

The example illustrates the ability to define a multi-grain transaction. The multi- grain solution is useful when the transaction has to be delayed not before its start but in the course of its execution. In such a case, one grain of the transaction is terminated and the next one is invoked by the chain statement. When the chained grain is masked, its acceptance is delayed until another transaction unmasks it. The grain mask flag can reflect the value of the synchronisation condition. The flag is set to unmasked when the condition is satisfied, or it is set to masked otherwise. This mechanism is similar to that of Hoare's conditions(Hoare 1974). It is suitable when the synchronisation condition depends on the server's state variables, but not on the transaction parameters. A version of a double-resource monitor with Hoare's condition can look like this:

```
type double-resource =  monitor
    var free0, free1: boolean;
        f0, f1: condition;

procedure allocate_0;
begin if not free0 then f0.wait;
 free0:= false
end;

procedure allocate_1;
begin if not free1 then f1.wait;
 free1:= false
end;

procedure allocate_both;
begin if not free0 then f0.wait;
 free0:= false;
 if not free1 then f1.wait;
 free1:= false
end;

procedure free_0;
begin free0:= true; f0.signal end;

procedure free_1;
begin free1:= true; f1.signal end;

begin  free0:= true;
 free1:= true
end.
```

Chaining the grains is a form of asynchronous message passing. The chaining grain does not block after it has invoked the chained grain. Thus, we benefit from the main advantage of the asynchronous message passing, namely from the higher concurrency level of the transactions. At the same time, we avoid the main disadvantage of the asynchronous message passing which is the need of unbounded buffering of invocations. In our model, a grain can chain one grain only. Thus, the number of invocations of a given grain is limited by the number of transactions running in parallel. This number, in turn, is limited by the number of client processes accessing the server.

2.4 Deferring the Grains

A grain execution can be deferred by calling the *defer statement* of the form

```
defer;
```

The statement can be executed anywhere in a grain compound statement; it takes the current values of the grain's actual parameters as a new grain invocation and defers this invocation. The defer statement can appear in the grain body only. The resume statement in the form

```
resume <grain-name>;
```

removes one of the deferred invocations of the grain specified by the *grain-name* and re-invokes this grain with the parameters values saved at the defer time. After being accepted, the re-invoked grain starts executing from the beginning of its body.

The *broadcast statement*

```
broadcast <grain-name>;
```

resumes all the transactions deferred on the grain specified by the *grain-name.*

Example 3: The *multisemaphore* Server

The *multisemaphore* is a kind of semaphore where the waiting operation (*mwait*) needs consuming one or more synchronisation signals to pass, and the signalling operation (*msignal*) deposits one or more signals on the semaphore. The numbers of signals consumed or deposited by the operations are given by their parameters.

```
process spec multisemaphore(int i)
{ trans void mwait(int c);
  trans void msignal(int c);
}
process body multisemaphore(i)
{ int count;
  if (i > 0) count = i;
  else count = 0;
  select
  {  accept mwait(c)
     { if (count < c) defer;
       else count - = c; treturn;}
  or accept msignal(c)
     { count + = c; broadcast mwait;
       treturn;
     }
} }
```

Deferring a grain is useful way of delaying a transaction if the grain's synchronisation condition depends on the invocation parameters; each grain invocation has to check the fulfilment of the condition by itself. The masks are generally not useful in this case since they can represent the synchronisation conditions which depend on the server's state variables only. Deferring grains allows achieving similar results like the *suchthat* clause in Concurrent C or the *and* clause in SR. A multisemaphore algorithm in genuine Concurrent C could look like this:

```
process body multisemaphore(i)
{ int count;
  if (i > 0) count = i;
  else count = 0;
  for ( ; ; )
  select
  { accept mwait(c)
    suchthat (count >= c)
    { count - = c; }
  or accept msignal(c)
    { count + = c; }
} }
```

Our solution is, however, more flexible; the synchronisation condition evaluation is programmed as a regular part of the grain body. There is no restriction as far as side effects are concerned and the condition can depend on the current values of the grain parameters. In Concurrent C and SR, the synchronisation condition is evaluated before the transaction is started.

Moreover, the defer mechanism reduces the number of synchronisation condition re-evaluations considerably. The deferred grains re-evaluate their conditions after their resuming only. Both the *suchthat* clause in Concurrent C and the *and* clause in SR need the condition re-evaluation for each pending invocation every time the select statement is executed. The chaining and the deferring mechanisms of our rendezvous gives the functionality similar to that of Mesa conditions (Mitchel 1978).

2.5 Non-Blocking Transaction Call

The non-blocking transaction call is executed by the nest statement of the form

```
   nest <chained-grain-name>
   (<process-value>.<transaction-name>
   (<actual-parameters>))
```

The statement terminates the calling grain's compound statement and calls the transaction specified by the *transaction-name* in the process designated by the *process-value*. The current select statement remains running; the nested transaction executes in parallel with other acceptable grains opened by the current select statement (if any). When the nested transaction terminates, the grain specified by the *chained-grain-name* is invoked. The return value of the nested transaction is passed as an actual parameter to the chained grain.

Example 4: Hierarchical Resource Allocator

Two resources x and y are controlled by the xa and ya instances, respectively, of the *allocator* server. Resource t is used always along with ether x or y. The server *t_allocator* allocates or frees t and x or y together. The *t_allocator* server calls *allocator* to access x or y. This is a non-blocking nesting; while a transaction is waiting for either x or y resource, *t_allocator* can still accept other transactions.

```
process spec t_allocator
  (process allocator xa,
   process allocator ya)
{trans void allocate(int other);
 trans void free(int other);
}
process body t_allocator(xa,  ya)
{ grain void allocate_t();
  grain void free_t();
  int free = true;
  select
  { accept allocate(other)
    {if (other == X)
      nest allocate_t(xa.allocate());
      else
      nest allocate_t(ya.allocate());}
  or accept allocate_t()
      free_t--; mask allocate_t;
  or accept free(other)
    {if (other == X)
      nest free_t(xa.free());
      else nest free_t(ya.free());}
  or accept free_t()
      free_t++; unmask allocate_t;
   } }
```

3 Design Issues

3.1 Synchronous Versus Asynchronous Transactions

Our model of the multi-grain transaction is simple but powerful enough to illustrate the dynamic selection mode of concurrent computation. The transaction is a synchronous transaction implemented as a sequence of grains and nested transactions. It combines some aspects of both the synchronous and the asynchronous message passing features.

From the client process point of view, the transaction is synchronous; the client is blocked until it is notified that the transaction is terminated and the result is sent back to it. From the server point of view, however, the transaction grains are asynchronous; the chaining grain does not wait for the chained grain to complete. As a result, there can be many transactions running in parallel even though their grains are executed in mutual exclusion mode. To keep the number of the pending grain invocations limited, we decided that transactions cannot branch their grains (each grain can chain one grain only). As a consequence, the statements: *treturn, chain, defer* and *nest* terminate the compound statement of the calling grain.

In Section 2 we suggested that, as far as the expressiveness is concerned, the synchronous multi-grain transaction can be compared with the monitor procedure entry call. The model can be extended with *asynchronous multi-grain transaction*; it is similar to the synchronous multi-grain transaction with the exception that it does not return any value or acknowledgement to the calling client. A semi-blocking call can be a reasonable solution for the client call. The client is blocked until the server (but not the transaction) acknowledges buffering of the client's invocation. The transaction grains are still executed in a mutual exclusive mode and can be masked and/or deferred. Such asynchronous transaction invocation can be compared to a monitor process entry start.

Further extensions correspond to non-mutual exclusive calls:
- *remote procedure call*; it is a synchronous call corresponding to a class procedure entry call; procedures are not executed in a mutual exclusive mode,
- *remote process start*; it is an asynchronous call corresponding to start of a process declared as an entry attribute of a class.

3.2 Blocking Transaction Call

The nest statement defined in Section 2.5 allows calling a transaction which will execute in parallel with other acceptable grains opened by the running select statement. We named this kind of transaction call a non-blocking nesting. The optional solution - blocking nesting - makes the calling grain delay until the called transaction terminates.

The problem of choosing between the blocking and non-blocking mode for transactions nested in a grain is similar to that of the nested monitor calls. An operation of monitor A which calls an operation of another monitor B can either release or hold the critical region of monitor B. None of the solutions is ideal. We have chosen the non-blocking nesting as it is more compatible with the spirit of the multi-grain transaction concept. The blocking call is still possible outside the grain body (in the static selection mode).

3.3 Time-Outs and Termination

We have skipped the features related to time-outs at the client side (timed transaction call) and at the server side (delay statement as a select statement alternative). They are important features of the distributed programming language but are not necessary to illustrate the dynamic selection mode.

4 Implementation Issues

In this section, we focus on the implementation of the multi-grain transactions for the processes running on the multiprocessor systems with the UMA shared memory. Some remarks concerning the implementation of such transactions in the NUMA shared memory (Chaves et al. 1993) and distributed memory architecture are also formulated. This implementation can also be applied for the threads multiplexed on virtual processors (e.g. UNIX-like processes) which can share memory.

We structure the multi-grain transaction mechanism at two levels. The low-level module called DCORE defines elementary concepts of *coroutine, link*, and *needle.* This concepts, in turn, are used to build the high-level module DKERNEL which defines user level concepts: *process, mailbox*, and *grain.*

4.1 Low-Level Primitives

The low-level primitives we have introduced are an extension of the synchronisation primitives introduced by Wirth to Modula-2 (Wirth 1985). Wirth's primitives are meant for a single-processor computer working in a multiprogramming mode. The primitives are based on two concepts: *coroutine* and *interrupt.*

The coroutine is a kind of logical process running on a processor along with other coroutines in an interleaving mode, one at a time. The coroutines are created by the NEWPROCESS primitive. The control is passed explicitly from one coroutine to the other using the TRANSFER primitive. TRANSFER(a, b) passes control from coroutine *a* to coroutine *b*. A coroutine can synchronise with outside world using interrupts. The IOTRANSFER(a, b, i) primitive transfers control from *a* to *b* much like TRANSFER does. Later on, on arrival the interrupt *i*, the control is switched back from *b* to *a*. Mutual exclusion on entry to the monitor is achieved by disabling interrupts.

In (Chrobot 1994) we discuss how these single-processor related concepts can be expanded for multiprocessor shared memory system. The final conclusion is that a third low-level concept, *busy semaphore* (spin lock) has to be added; a global instance *mutex* of such a spin lock is used to achieve an inter-processor mutual exclusion. The most important finding is that the critical region assured by mutex must be released inside the transfer control primitive. As a result, a new primitive RELEASETRANSFER(a, b) is introduced. It releases *mutex* after saving the context of coroutine *a* and before restoring the context of coroutine *b*. On top of such primitives, monitor construct for multiprocessor can be built.

4.1.1 Fundamental Idea for Structuring the Message Passing Tools

Now we want to expand the same low-level primitive approach for the message- passing paradigm. The fundamental idea is that the message-passing paradigm synchronisation is to be built around the interrupt concept, in the same way as the shared-memory paradigm synchronisation is built around the spin lock.

The interrupt feature and the spin lock are both hardware supported features. They are one-out-of-many selection mechanisms. In case of spin lock, many processes compete, in a loop, for a signal deposited in a shared bit. The process which finds the signal resets the bit and passes the selection operation (traditionally called Test-and- set). The hardware support for this kind of selection is called *memory arbiter.*

In case of interrupt, one process checks, in a loop, many signal bits. Each bit is associated with a process to be selected. A process is selected, if it has sent a signal to its signal bit and if the selecting process has accepted this signal. Hardware support for this kind of selection is called *interrupt system.* The processor plays usually the role of the selecting process. The signal bits are called *interrupt sources*, and the processes associated with the signals are called *interrupt handlers.*

The spin lock can be considered a prototype of the mutual exclusion mechanism, while the interrupt - as a prototype of the rendezvous mode of execution.

On top of the interrupt concept, we build a hierarchy of message passing tools in a structured way (Chrobot et al. 1995), much the same as a hierarchy of shared variable paradigm tools are built on top of the spin lock concept. It is worth stressing that we use interrupts not only to synchronise the physical processors and devices in the system. We use them to synchronise the logical processes (threads) multiplexed on physical (or virtual) processors as well.

4.1.2 Synchronous Link and Acknowledgement Protocol

For communication between coroutines, we introduce a message passing concept called synchronous link. The *synchronous link (slink)* is a type of uni- directional, point-to-point communication entity to transmit simple values (words or pointers) between two coroutines running in the same or separate memory spaces. Two coroutines: sender and receiver access a synchronous link through its ports: *slink output port* and *slink input port.*

Synchronous link implements an idea of the zero-capacity bounded buffer accessible for two coroutines only. The input port can store a simple *data value* (word or pointer) and a *data signal bit.* The output port can store an *acknowledgement signal bit.* To transmit a simple value v via the link l with the output port *op* and the input port *ip*
- the sender sends the value v through the output port *op* calling the DPUT(op, v) operation, then it sends the data signal bit using the DSIGNAL(op) operation, and busy- waits for the acknowledgement signal by calling AWAIT(op); on the other end of the link l
- the receiver calls the DWAIT(ip) operation to wait for the data signal in the port *ip*, then it receives the data value to the variable x by calling DGET(ip, x), and invokes ASEND(ip) to send an acknowledgement signal back to the sender.

This basic acknowledgement protocol can be used to transmit messages between coroutines in different memory architecture systems as follows:
- in the shared memory with uniform access (UMA), it is enough to transmit the message pointers only; the receiver accesses the message contents directly in the sender's memory location,
- in the shared memory with non-uniform access (NUMA), the message pointer is transmitted to the receiver and then the message contents are copied from the sender to the receiver location before the message is acknowledged,
- in the distributed memory, the message contents are transmitted down the link word by word; the size of the message can be defined by the number of words in the message or by end-of-text (EOT) character transmitted in the message.

4.1.3 Interrupts and Interrupt Selection Modes

The DWAIT and AWAIT operations related to the synchronous link assure a busy-waiting synchronisation between coroutines. Such a form of synchronisation is acceptable, if each coroutine runs on its own processor. If many coroutines are multiplexed on one processor, a non-busy form of waiting is needed. To achieve a non- busy form of waiting we use link signals as interrupt signals. In technical terms, one could say that we 'connect the link signals to the interrupt system'.

Wirth's IOTRANSFER primitive allows a coroutine to wait for one interrupt at a time only. We have introduced a new primitive, MTRANSFER, which allows one coroutine to wait for many interrupts at a time. Each interrupt i, to be waited for, has to be, previously, attached to the coroutine and has to be associated with its handler H by the ATTACH(i, H) primitive. A *mask flag* is associated with each interrupt signal. The MTRANSFER(fg, bg) operation transfers control from the foreground coroutine *fg* to the background coroutine *bg* and opens all the interrupts attached to *fg*. From now on, if the interrupt system is enabled, the signals from the opened and unmasked interrupt sources are accepted. On each acceptance, the handler associated with the accepted interrupt is executed.

The interrupt handler starts without any context switching. It is executed on the stack of the interrupted process. When the handler is completed, the interrupted process is continued. The context switching overhead for this kind of operations is even lower than that for lightweight processes (threads); we call such an operation a *needle.*

During a needle execution, the interrupt system is disabled. Thus, at most one needle can be executed at a time. When the needle is completed, the MTRANSFER operation is still pending and other signals can be accepted. It is worth stressing that the sequence in which the needles are selected for execution is not defined by the foreground process program (and its data). It depends on the sequence of arrival of the interrupt source signals. If more than one signal has arrived at the same time, the selection is done in an arbitrary way by the interrupt system. We can see that the dynamic selection mode is based on the interrupt feature.

The foreground coroutine enters the dynamic needle selection mode by executing a MTRANSFER operation and stays in it until one of the needle handlers closes it explicitly by executing the CLOSESELECT operation. When the needle calling CLOSESELECT is completed, the control is transferred back from the background process *bg* to the foreground process *fg*. The foreground process enters the static selection mode and stays in it until the next MTRANSFER operation is executed.

4.2 High-Level Primitives

The high-level concepts: *process*, *mailbox*, and *grain* are built on top of the low-level primitives, in the module called DKERNEL. Each (virtual) processor has its own instance of this module.

DKERNEL consists of two sub-layers. The lower sub-layer, called *mailbox*, is permanent and configuration dependent. The upper sub-layer supports *user process* and *grain* concepts; it is volatile and application oriented.

4.2.1 Mailbox

The lower sub-layer - mailbox - consists of a coroutine called *message dispatcher* and *inter-processor channels* to connect the message dispatcher with its counterparts in other processors. Each inter-processor channel consists of two links transmitting data in the opposite directions. The inter-processor channel links and the interrupt system associated with them are hardware supported. The ports of the inter-processor channel links are associated with appropriate needle handlers which co-operate during remote message passing.

The mailbox is initialised at the system start time. It runs permanently and creates the environment for applications using multi-grain transactions.

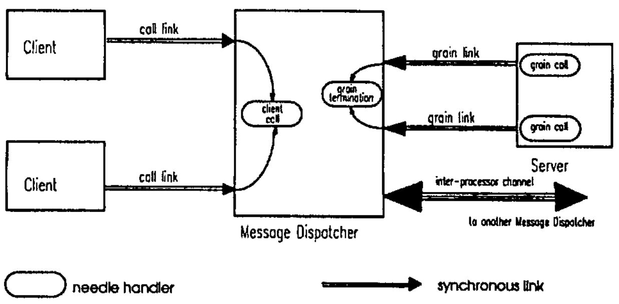
4.2.2 Processes and Grains

In the higher sub-layer of the DKERNEL module, the functions creating a *multi-grain transaction library* are implemented. The functions can be used to build language constructs related to multi grain transactions like processes and grains.

The *processes* are coroutines which are interconnected with the local message dispatcher coroutine with links. The processes are created by the initial process using the CREATEPROCESS(P) operation. The operation creates a new coroutine with the function P as its body and inserts it into *ready* queue.

There are two (not necessary disjoint) classes of processes: clients and servers. The *client process* is a coroutine which is interconnected with the local message dispatcher with so called *call link*. The client process is a sender for the call link, and the message dispatcher is its receiver. Each client process has one call link only. The message dispatcher associates the *client-call* needle handlers with all the call link output ports.

The *server process* is a coroutine which is interconnected with the local message dispatcher with so called *grain links*. One grain link is created for each grain defined in the server. The message dispatcher is a sender for the grain links while the server is the receiver for them. On the message dispatcher side, the *grain-termination* needle handler is associated with each grain link output port , and on the server side, the *grain-call* needle handler is associated with each grain link input port.



To be able to accept its transaction, the server attaches a grain handler *GH* to each grain link input port *gop* by calling ATTACHGRAIN(gop, GH), and enters the dynamic grain selection mode calling the OPENSELECT() operation. The *grain handler* is a function taking a pointer to a message and returning no value. The grain handler is invoked by the *grain-call* needle. The needle hides the protocol operations of the grain link from the user. The operation OPENSELECT(fg, bg) delays the current foreground process *fg*, moves from the ready queue a background process *bg* and uses MTRANSFER(fg, bg) to transfer control from *fg* to *bg*. The foreground process stays in the MTRANSFER operation and accepts grains until one of them invokes the operation CLOSESELECT. This operation completes MTRANSFER. Then the foreground process is resumed to complete the OPENSELECT operation. The background process is inserted into the ready queue, and the foreground process continues with the operation following OPENSELECT.

Client process invokes the transaction *tn* in the process server *sn* by preparing the parameters in its own message buffer pointed by the pointer *m*, and by calling the primitive TCALL(sn, tn, m). The message buffer is big enough to contain not only the arguments and results of the transaction but to pass messages between the grains chained in the transaction as well. TCALL stores (pushes) *sn*, and *tn* in the message buffer and transmits its address down the call link to the message dispatcher.

The *client-call* needle, at the message dispatcher side, retrieves *sn* and *tn* from the message and identifies the first grain in the transaction chain. If the grain is currently invoked, the new invocation (call link input port *cip*, and received message pointer *m*) is inserted into the queue od pending invocation *pen*; otherwise the new invocation becomes the current invocation and the message pointer *m* is send down the grain link to the server. If the grain input port is not masked the message pointer is accepted by the grain call needle; otherwise the acceptance is postponed until the grain is unmasked.

If the grain is completed, the *grain-call* needle at the server side sends an acknowledgement signal down the grain link. This invokes the *grain-termination* needle on the message dispatcher side. The needle retrieves the code of the *grain termination operation* invoked by the grain. The possible options are: TRETURN, CHAIN, DEFER, NEST.

In case of TRETURN, the needle simply sends an acknowledgement signal to the client through the cip port. In case of CHAIN, the chained grain is invoked according to the rules described for the first grain invocation. In case of DEFER, the current invocation is inserted into the deferred invocation queue *def*. In all the cases, if the queue *pen* for the terminated grain is not empty, one pending invocation is removed and sent to this grain.

The deferred invocations are moved to the pending invocations queue by the RESUME and BROADCAST operation. The former moves one invocation only while the later moves all the deferred invocations.

The high-level library hides the message dispatcher coroutine and both the message dispatching and process scheduling algorithms. Each client process can call any server process and the server is not aware of the client's name.

5 Related Works

Silberschatz (Silberschatz 1979) presents a set of a system level primitives and protocols for an abstract implementation of the CSP IO commands and the guarded statements (Hoare 1978). The primitives and protocols are similar to our low-level primitives and busy-waiting protocols. We, however, have shown that an efficient implementation of the non-busy form of waiting in such protocols can be based on the interrupt feature.

An extension of the Modula-2 low-level concurrency features related to interrupts is given in Modula-2 Standard Draft (2nd Committee 1992). It facilitates the implementation of selective acceptance protocol. However, it does not support the needle handlers, but the process level interrupt handlers only. The Draft considers the interrupt to be an IO device synchronisation feature only. It does not define any mechanism to either generate the interrupt signals by the processes or to transmit data between the processes.

Hills goes further with his proposal for Structured Interrupts (Hills 1993). He adds the facility to send interrupt signals by processes. In this way, the interrupts are considered as an interprocess synchronisation mechanism. The primitives proposed by Hills are of higher abstraction level than our primitives; they include the ready process queue management. His interrupt handlers are the process-level handlers.

Chaves et al. (Chaves et al. 1993) suggest that the interrupts can be used for kernel-kernel communication in a shared-memory multiprocessors with non-uniform memory access (NUMA). One of the alternatives for the kernel-kernel communication is the remote invocation. "The processor at node *i* sends a message to a processor at node *j*, asking it to perform the operation on its behalf. The operation may be executed directly by the message interrupt handler, or indirectly via wakeup of a kernel process"(Chaves et al. 1993). In our terminology the operation can be executed either by a needle handler, or by a process-level handler. The authors point out that the message interrupt handler cannot be used to execute an operation that may block. For such operations they suggest process-level handling.

Bjorkman (Bjorkman 1994) points out that substantial efficiency gains can be achieved if interrupt handlers are used instead of threads (process-level handlers) for network protocol processing. Typical message delivery includes one interrupt and two context switching on the receiving side. The context switching is extremely costly in RISC processors with large register sets. When the protocol processing is included into the interrupt handler, the context switching can be avoided.

The examples presented above point out that using interrupts in programs based on message passing paradigm can improve efficiency of such programs. All this attempts, however, remain under a substantial influence of the static selection mode for operations in concurrent sequential processes. In the Hill's idea, the interrupts are accepted in the dynamic mode, but handled in the static mode. Chaves et al., and Bjorkman suggest executing non-blocking handlers in the dynamic selection mode, but the blocking handlers have to be executed in the static mode as process-level handlers.

This paper makes a step further and proposes a model in which also the blocking message handlers can be executed in the dynamic selection mode. The blocking message handler takes a form of a multi-grain transaction.

6 Conclusion

In the classic implementation of the message passing primitives, the Concurrent C transactions or ADA entries are selected for execution by the *select* statement and are executed as a part of the server process. We have identified in this implementation two different modes of selection of operation in a sequential process. The *select* statement introduces *dynamic selection mode* for a sequential process. It is a mode in which the successor of an operation is selected at run-time from a set of operations defined by the process program. It is an alternative to the *static selection mode* in which only one successor for each process operation is defined by the program of the process. The dynamic selection mode is used to accept the transactions, but the transactions themselves execute in the static mode.

We have identified the dynamic selection mode as an inherent property of the message passing paradigm and the static selection mode as an inherent property of the shared variable paradigm and came to the conclusion that mixing the two modes in one paradigm can be a source of poor expressiveness and poor efficiency of the transactions. This observation leads us to the concept of *multi-grain transaction* which is not only accepted in the dynamic selection mode but executed in it as well.

Applying the multi-grain approach to the transactions which are executed in either the mutual exclusion or non-mutual exclusion modes and, at the same time, in either the synchronous or asynchronous way gives a full range of message passing primitives with the expressiveness comparable with analogous primitives in the shared variable paradigm. Moreover, the process management overhead of the message passing paradigm primitives can be reduced considerably.

Improved in this way, the message passing mechanism can be used as a basic IPC mechanism to reduce the communication overhead in pure distributed systems in the similar way as the shared variable mechanism can be used to reduce the load imbalance overhead in the pure shared memory systems. In the systems which combine the features of both the shared and distributed memories, a combination of these two paradigms can be used.

It is also interesting that the dualism at the monitor and server level described above can be observed in the internal structure of their mechanisms as well. The basic low-level mechanisms: spin lock and interrupt system can be considered dual to each other. It shows a very strong internal interrelationship between the two paradigms.

References

[1] Andrews, G. R., Ollson, R. A., Coffin, M., Elshof, I., Nilsen, K., Purdin, Townsen, .G. (1988) An Overview of the SR Language and Implementation, *ACM Transaction on Programming Languages and Systems*, Vol. 10 No. 1. pp. 51-86.

[2] 2nd Committee (1992) Draft of the Modula-2 Standard: CD 105114.

[3] Bjorkman, M., (1994) Interrupt Protocol Processing in the x-kernel. TR 94-14. Department of Computer Science, The University of Arizona.

[4] Brinch Hansen, P., (1978) Distributed Processes. A Concurrent Programming Concept. *Comm.ACM* Vol. 21. No. 11. pp. 934-941.

[5] Chaves, E. M., Das, P. Ch., LeBlanc, T. J., Marsh, B. D., Scott, M. L., (1993) Kernel-Kernel Communication in a Shared Memory Multiprocessor. *Concurrency: Practice and Experience*, Vol. 5, No. 3, pp 171-191.

[6] Chrobot, S. (1994) Where Concurrent Processes Originate. *Proceedings of the Conference on Programming Languages and System Architecture*, ETH, Zurich, Lecture Notes on Computer Science 782, pp 151-170.

[7] Chrobot, S., Stras, A., Stras, R., (1995) Structuring the Message-Passing Channels.*Applied Mathematics and Computer Science.* Vol. 5, No. 1, pp. 189-208.

[8] Dijkstra, E. W., (1976) A Discipline of Programming. *Prentice Hall*, Inc. Englewood Cliffs, New Jersey.

[9] Gehani, N. H., Roome, W. D. (1986) Concurrent C. *Software - Practice and Experience*, Vol.16, No. 9 pp. 821-844.

[10] Hills, T., (1993) Structured Interrupts. *Operating System Review*, Vol. 27, No. 1, pp. 51-69.

[11] Hoare, C. A. R. (1974) Monitors: An Operating System Structuring Concept. *Comm. ACM*, Vol. 17, No. 10. pp. 549 - 557.

[12] Hoare, C. A. R., (1978) Communicating Sequential Processes. *Comm. ACM*, Vol.21, pp. 66-677.

[13] INMOS (1988) Limited, occam2, Reference Manual, *Prentice-Hall.*

[14] LeBlanc, T. J., Markatos, E. P., (1992) Shared Memory vs. Message Passing in Shared-Memory Multiprocessors. *Proc. 4th IEEE Symposium on Parallel and Distributed Processing*, Dallas, Texas, pp. 254-263.

[15] Mitchel, J. G., (1979) Messa Language Manual, *Xerox PARC Technical Report CSL-79-3*,Palo Alto, Calif.

[16] Silberschatz, A. (1979) Communication and Synchronisation in Distributed Systems. *IEEE Transactions on Software Engineering.* Vol. 5, No. 6, pp 542-546.

[17] United (1983) States Department of Defence, Reference Manual for the ADA Programming Language, ANSI/MIL-STD 1815A.

[18] Wirth, N. (1985) Programming in MODULA-2. *Springer-Verlag.* Berlin.

Comparing Inference Control Mechanisms for Statistical Databases with Emphasis on Randomizing

Ernst L. Leiss
University of Houston, Department of Computer Science, Houston, Texas, 77004, U.S.A.
coscel@cs.uh.edu

AND
Jurij Jaklič
University of Ljubljana, Faculty of Economics, Kardeljeva pl. 17, 61000 Ljubljana, Slovenia
jurij.jaklic@uni-lj.si

Keywords: statistical databases, security, comparison, randomizing

Edited by: Jerzy R. Nawrocki
Received: July 27, 1994 **Revised:** February 24, 1995 **Accepted:** April 12, 1995

Statistical databases *are primarily collected for statistical analysis purposes. They usually contain information about persons and organizations which is considered confidential and only aggregate statistics on this confidential attribute are permitted.*
However, deduction of confidential data *(inference) is frequently possible. In order to prevent this possibility several security-control mechanisms have been developed, among them the* randomizing *method.*
We compare randomizing with other methods using several evaluation criteria. Evaluation shows that randomizing has several advantages in comparison with other methods, such as high level of security, robustness, low cost. On the other hand, the problem of bias for small query sets can be considerable for some applications.

1 Introduction

Databases often contain *confidential* information. Various security-control mechanisms deal with different kinds of security problems, such as encryption, identification of users, and access authorization. Here we will study inference, i.e., the deduction of confidential information from legal (i.e. permitted) statistical summary queries.

A *statistical database* (SDB) is a database where certain users are authorized to issue only aggregate (statistical summary) queries, such as sum, maximum, minimum, count, average, median, variance, standard deviation, and k-moment. These users (*researchers*) cannot retrieve information about an individual entry.

Typical examples are Census Bureau databases, salaries of individual persons in a company, and diagnoses of patients in a hospital.

We must enable these users to retrieve aggregate statistics, but prevent them from retrieving values of confidential attributes of individual records. In this way we protect the individual's right to privacy and on the other hand we can process information needed by the society [22].

Problems arise when certain users (*snoopers*) try to derive or infer confidential information from legal aggregate queries. If they are successful, we say that the SDB is *compromised.* There is more than one way how to compromise a SDB, for example the linear system attack [10] or using a tracker [8].

Several methods have been developed in order to protect SDB's against compromises. Most of them are described in [2], where they are also classified and evaluated. Randomizing security-control mechanisms such as the one described in [17] are just mentioned in [2]. Our goal is to evaluate this method and compare it with others.

2 Overview of the Methods

Two models offer frameworks for dealing with the problem of the SDB security [2]: one is called *the conceptual model* [5] and the other *the lattice model* [9]. They support the research in this field but do not offer methods for security control.

There are two approaches to the problem of inference for statistical databases:

- query restriction approach and
- perturbation approach.

Methods belonging to the same group have similar characteristics and are therefore easier to compare. In this section a brief overview of the two approaches is given.

2.1 Query Restriction Approach

The idea is that if we do not permit every aggregate query to be executed, we can achieve better security of the database. The proposed methods differ in the way in which it is decided whether the query is permitted.

The first method (*query set size control*) is to limit the query set size [13]. It has been noticed [10], that if the issued queries have many common entities (records) in their query sets, there is a higher possibility of compromising the database. So, the method *query set overlap control* proposes to restrict the number of records that any pair of issued queries can have in common. One of the methods which can provide a very high level of security is *auditing* [27]. Auditing keeps track of issued queries and checks for each new query whether the database can be compromised. The partitioning method ([4],[25]) and the cell suppression method [6] are other techniques.

2.2 Perturbation Approach

The perturbation approach includes two subgroups of methods. One subgroup replaces original data in the SDB with other data and uses these perturbed data to compute statistics (*data perturbation*) while the other subgroup computes statistics from the original data, but noise is added in one or another way during the computation of the statistics (*output perturbation*).

The *probability distribution methods* [19] replace the original SDB by another sample with the same (assumed) probability distribution. A variant of this method (the *analytical method*) approximates the data distribution by orthogonal polynomials (see [15]). The other proposed approach [23] is to replace the true values of a given attribute with the perturbed values once and forever (*fixed data perturbation*). There are two different methods, one for numerical and one for categorical attributes.

An example of the output perturbation methods is the *random sample queries* method [7]; a statistic of a randomly selected subset of the original query set is given as a response to the issued query.

In the *randomizing method* [17] a response to the given query is computed from a superset of the query set. Records are randomly selected from the database and added to the query set.

Let us assume that a user is interested in a certain statistic of a given query set of size k. Let $i_1, \ldots, i_k$ be the indices of the records in the query set, and let DK be the name of the confidential attribute in whose statistic the user is interested. Instead of computing the true statistic, another $v \geq 0$ entities of the database are selected randomly and added to the query set. Then the statistic of this superset of the original query set with $k+v$ records is computed and returned as the result. Thus for a query of type average we have the perturbed response

$$A = \frac{\sum_{j=1}^{k} DK_{i_j} + \sum_{j=1}^{v} \mathrm{DK}_{s_j}}{k+v} = \\ = \frac{k \cdot a + \sum_{j=1}^{v} \mathrm{DK}_{s_j}}{k+v},$$

where s_j is the index of the j-th randomly selected record and a is the true response $(\sum_{j=1}^{k} DK_{i_j})/k$.

v should be (much) smaller than k, otherwise the precision of the result can be very bad, although the security of this method increases with higher values for v. That yields a *security-accuracy trade-off*. Here we select v to be equal to 1, justified by the observation (see [17]) that even with small introduced noise, the relative error of the inferred values for DK will be considerable for large values of k.

3 Comparison

We use the evaluation criteria proposed in [2] to compare the different methods; they cover the important objectives of a good security control mechanism. Some of the criteria exclude each other, and thus an effort must be directed towards balancing them.

3.1 Security

We consider different kinds of disclosures:

- *Exact disclosure* is possible when a user can obtain the exact value of the confidential attribute.
- *Partial disclosure* is possible when a user can obtain an estimate DK_i' of the value of the confidential attribute DK for the i-th record, such that $Var(DK_i') < c_1^2$ (i.e. variance of the estimate DK_i' is less than c_1^2), where the parameter c_1 is set by the DBA (Data Base Administrator). For the case of categorical attributes a *partial disclosure* is possible when a user can infere that the confidential attribute *has not* a certain value.
- *Statistical disclosure* occurs when the same query is issued several times in order to obtain a small variance of the estimate of the true response - *filtering*.

Let us look first at the exact compromisability of an SDB under different protection methods. There is the possibility of exact disclosure for some methods belonging to the query restriction approach group, namely the query set size control and the query set overlap control.

Exact compromise cannot be done for a SDB protected by the auditing because of the nature of the auditing. This is also true for cell suppression and partitioning (see [6] and [26]).

For methods belonging to the perturbation group, including randomizing, there is no possibility for the exact disclosure, except for some rare cases for the analytical method (see [2]) and for rounding [1]. Conditions under which exact disclosure is possible for these two methods are very severe.

There are more possibilities for partial disclosure. Regardless of which method we use, it is possible to achieve partial disclosure. As for the most of the perturbation approach methods, it is possible to balance the security against the precision also for randomizing. The parameters which influence the security (and precision) and can be set by the DBA are: v, the number of records to be added to the query set and j, the parameter which is used in the method to solve the problem of accuracy (see Section 3.2).

The problem of statistical disclosure has to be considered only in the cases where answers to the same query issued several times differ. For these (perturbation) methods one can achieve a better estimate of the real answer to the query using the method of filtering. The idea is that the user repeats the same query several times. Let A_n denote the perturbed response to the n-th repetition of the query with the true response a. In general $A_i \neq A_j$ for $i \neq j$. Then the user can repeat the query m times and compute the average

$$a' = \frac{A_1 + A_2 + \ldots + A_m}{m}.$$

The result of this expression will converge to a certain value a^* with increasing m. If the perturbed response (after one repetition) is A then we have

$$Var(a') = \frac{Var(A)}{m},$$

if the answers to repeated queries are independent. Thus, we see that the more times one repeats the query, the better an estimate a' of the true response a can be achieved.

Let us see how many queries (m_r) we have to issue if we want to get statistical disclosure of the SDB protected by the basic randomizing method, if the criterion for the statistical disclosure is

$$Var(a') < c_1^2,$$

where c_1 is the parameter set by the DBA. Let us consider a fixed query of type average with the true value a. The perturbed value of that query is

$$A = \frac{a \cdot k + \mathrm{DK}_s}{k+1},$$

where k is the size of the original query set and DK_s is the value of the confidential attribute DK of the randomly selected record. If the random number generator we use is uniform, then we can expect that on the average the value for DK_s is equal to the average of the values over the database (DK*). Thus the expected value of A is

$$E(A) = \frac{a \cdot k + \mathrm{DK}^*}{k+1}.$$

Now we can compute the variance of A:

$$Var(A) = E\left(\frac{a \cdot k + \mathrm{DK}_s}{k+1} - \frac{a \cdot k + \mathrm{DK}^*}{k+1}\right)^2 = \\ = \frac{E(\mathrm{DK}_s - \mathrm{DK}^*)^2}{(k+1)^2} = \frac{Var(\mathrm{DK})}{(k+1)^2}.$$

Because the responses to the queries are independent of each other, we have

$$Var(a') = \frac{Var(A)}{m} = \frac{Var(\mathrm{DK})}{m \cdot (k+1)^2}.$$

As we expected, the variance of the estimate depends on the variance of the confidential attribute and it is smaller for larger query set size. Thus we have:

$$m_r \geq \frac{Var(\mathrm{DK})}{c_1^2 \cdot (k+1)^2}.$$

Even if the number of queries needed to compromise a SDB can be quite large, it is possible to do it. The reason is that with the increasing m the average introduced noise of the query always converges to the same value DK^*. In order to avoid this one can use the following method.

For each issued query two calls to the random number generator are made, and therefore two indices for the additional record are proposed. The selection of the single record to be added to the query set depends on the values of the confidential attribute of the records which are already in the query set. Let us say that the two proposed indices are x_1 and x_2. Then we choose $\max\{x_1, x_2\}$ if the value of the boolean expression

$$E = [(DK_{i_1} \leq DK_{i_2}) \oplus (DK_{i_2} \leq DK_{i_3}) \oplus \dots \\ \dots \oplus (DK_{i_{k-1}} \leq DK_{i_k})]$$

is true, and $\min\{x_1, x_2\}$ otherwise. Here $\oplus$ denotes the XOR operator. Thus the average introduced noise may differ for two different queries and therefore the database cannot be compromised.

3.2 Accuracy of Responses

The problem of the accuracy of responses occurs when we use perturbation methods. Using query restriction control methods, the responses to queries are always equal to the true responses.

We consider two criteria, namely *bias* and *precision*, i.e. variance of the estimator.

As stated in [2] the main disadvantage of the randomizing method in comparison with other output perturbation methods is the problem of bias. A bias occurs when

$$E(a|A = w) \neq w$$

where again a is the true response to a fixed query and A is the perturbed value for that query.

In our case of the randomizing method and for queries of type average, we have

$$a = \frac{A \cdot (k+1) - \mathrm{DK}_s}{k}.$$

¿From this we can compute

$$E(a|A = w) = \frac{(k+1) \cdot w}{k} - \frac{E(\mathrm{DK}_s|A = w)}{k}.$$

Since, the selection of the random record does not depend on the query (for the basic method), we can obtain the final result

$$E(a|A = w) = \frac{(k+1) \cdot w}{k} - \frac{\mathrm{DK}^*}{k} = \\ = w + \frac{w - \mathrm{DK}^*}{k},$$

where DK* is again the average over all database. It follows that in the limit, $k \to \infty$, the bias will be *zero*.

The variance of the perturbed value A for randomizing is $\frac{Var(DK)}{(k+1)^2}$. So, for large values of k (query set size) this method gives us quite good results, which means precise and without bias. The only parameter which influences the precision is the query set size; the variance of the perturbed value is proportional to the variance over all the database.

The problem of accuracy is that the maximal error introduced by the randomizing can be arbitrarily bad [17]. The average error is not so bad, but the maximal error can be very unpleasant, specially for smaller k.

This problem can be solved, if we do not permit that the additional value is very different from the values of the records from the query set. Thus, for a query of type *average* we stipulate that the chosen record satisfy the condition

$$avg - \frac{mx + mn}{2 \cdot j} \leq DK_s \leq avg + \frac{mx + mn}{2 \cdot j},$$

where s is the index of the selected record, avg, mx and mn are the average, minimal and maximal values of the confidential attribute in the specific query set, and j is a parameter set by the DBA. If the first selected record does not satisfy the condition, then we select another record, and so on. Of course we must set a limit on the number of repetitions. Here the selection of j is essential. If j is too small then we do not restrict randomizing; on the other hand if j is too large, possibly no record will satisfy the condition.

It is difficult to say which of the perturbation methods is more precise, because the precision depends to a great deal on the selection of parameters of a given method.

All data perturbation methods, except the fixed data perturbation method for categorical attributes, suffer from the problem of bias. On the other hand, among the output perturbation approaches only randomizing has this problem. This problem might be considerable for small databases with high variance of the confidential attribute.

3.3 Consistency

A security control method is *consistent* if there are no contradictions or paradoxical results, e.g., if we get different responses to the repetition of the same query, or when the response on the average query differs from the quotient of the sum and count queries over the same query set.

All query restriction methods are consistent. The only thing we have to take care about is the possibility that the same query is once restricted and once not (e.g. query set overlap control). Also data perturbation methods do not give contradictory results, but we can obtain some paradoxical results, such as negative salaries.

On the other hand, all probability distribution methods, random sample queries and varying output perturbation methods are inconsistent. Since the randomizing method belongs to the random sample queries methods it is inconsistent too. But we can overcome this problem if we use quasi-randomizing [18] instead of the basic method described in [17].

In order to select a random record to be added to the query set we use a random generator. Each random generator is a function of a parameter *seed*, and for the same seed the same sequence of random numbers is generated. So, when we want to generate a random index of a record, we can use as a seed a function of the query set; thus for the same query set the randomly selected index will be always the same. The requirement for the function which maps a query set into a seed is that it does not change for any permutation of the query set. A simple solution is the sum of the values of the confidential attribute. If

$$QS = \{DK_{i_1}, \ldots, DK_{i_k}\},$$

then

$$\mathrm{seed}(QS) = DK_{i_1} + \cdots + DK_{i_k}$$

or

$$s = g(\mathrm{Rand}(DK_{i_1} + \cdots + DK_{i_k}))$$

where s is the randomly selected index and g is some function which maps random numbers into the set of indices of the records in the database. Another advantage of this method is that statistical disclosure is not possible, because responses to the same query issued several times are always the same. The problem of paradoxical values for randomizing is not as severe as for the fixed data perturbation method.

3.4 Robustness

We say that a security control method is robust if supplementary knowledge does not help a user who wants to compromise the SDB. Supplementary knowledge is considered to be all the information about the database which a user knows from a source other than the system [2]. In general the robustness of the query restriction approach methods is very low, since the responses to queries for these methods are always correct. So with supplementary knowledge about the database one can easily compute other values. Robustness can be controlled for some methods such as partitioning [26]. Very severe is the problem of robustness for auditing because queries with very small query set sizes may be permitted.

The perturbation methods are more robust, clearly because perturbed answers are returned to a user. Their robustness varies from moderate (in most cases) to high for the data swapping method and can be usually controlled by the parameters of a particular method.

The robustness of the randomizing method is high. In fact, if the number of elements known by the user is small in comparison to the query set size, the SDB is still secure. As stated in [16, pp. 15] also for the number of known elements approximately equal or larger than k (query set size), the number of repetitions of queries one has to issue in order to compromise the database is quite large. This is even more pronounced if quasi-randomizing is used.

3.5 Cost

We consider the cost of the implementation of the security control mechanism, the processing overhead per query and the education of the user necessary to understand the responses.

For randomizing we can say that the initial implementation effort is very low for the basic method and a bit more complicated for the mechanisms which provide higher security (see Section 3.1), higher accuracy (see Section 3.2) and consistency (see Section 3.3).

Some methods have very low processing overhead, for example query set size control, cell suppression, all data perturbation methods and random sample queries. On the other hand there are methods such as query set overlap control and auditing, which are time and space consuming. Even more disturbing than the average complexity of comparisons is the fact that the processing overhead is much larger for the queries issued later than for the queries issued at the beginning.

The randomizing method has low processing overhead, the time overhead required per query is constant and there is no need for additional space. From this point of view all variants of randomizing are basically low in cost.

The randomizing method follows the majority of the methods regarding the cost of the education of the user, which means that it is low in cost - simple to understand.

3.6 Generality

Some of the methods described in [2] are not developed for all types of aggregate statistic queries. For example, the varying output perturbation is developed only for *sum*, *count* and *percentile* statistics.

The randomizing method can be used for almost all types of queries, with the exception of *count*. But there are obviously some differences between the usage of randomizing for statistics such as sum or average and usage of the same method for selector functions as median, min, max.

While the precision for queries of type average is good, i.e. $Var(A) = \frac{Var(DK)}{(k+1)^2}$, we cannot say the same for the selector functions. In the worst case for the max and min queries the difference between the true response and the randomized response can be as large as the difference between the maximal and minimal value of the confidential attribute in the database, $\max_{i=1,\ldots,N}|\mathrm{DK}_i| - \min_{i=1,\ldots,N}|\mathrm{DK}_i|$. On the other hand, a user can get also the true response. If we choose an arbitrary query set of size k, then the probability that the answer given to the user will be exact, is equal to

$$\sum_{m=k}^{N} \frac{\binom{m-1}{k-1}}{\binom{N}{k}} \cdot \frac{m}{N} \approx \frac{k}{N}.$$

In order to solve this problem one can use the restricted randomizing method. However, we know that the value of the perturbed response to a query of type *max* is greater or equal to the true response. Thus, the problem of bias is here more severe.

The situation is the same for queries of type *min*, but not for queries of type *median*: here the answer will change often, but the error is usually small and depends on the variance of the database and the selection of the query set.

Since there are few perturbation methods which can be used for all types of queries, we can consider randomizing as a general method, even though it is not equally good for all types of queries.

3.7 Suitability

It is desirable that a method be applicable for both numeric and categorical attributes. All query restriction approaches are able to deal with both. Two fixed data perturbation methods have been developed, one for numeric attributes [28] and the other one for categorical attributes [2]. The probability distribution and the random sample queries can be used for both. The randomizing method is not suitable for categorical, but only for numeric attributes.

In addition, some methods are suitable for more than one attribute and others for only one attribute. Again all the query restriction methods are basically suitable for more than one attribute.

The only problem is for auditing, where the processing overhead can become very high and this method may have no practical value. From the perturbation methods the following are suitable for more than one attribute: data swapping, analytical method, random sample queries, if the attributes are independent, also the varying output perturbation, as well as randomizing.

The last criteria is suitability to dynamic SDB. For some purposes a SDB can be static, for example a census database. For other purposes it is very important that the database be dynamic and on-line. For such a database the method used must provide security also in cases of changes in the database. Moreover, it must not require too much processing overhead per change in the database. Randomizing is completely suitable for on-line dynamic databases and does not require any additional effort for implementation nor any processing overhead per change in the database. There are some methods which are not suitable to on-line dynamic SDB at all: cell suppression, data swapping, and probability distribution. Note that all the output perturbation methods are suitable for on-line dynamic SDB.

3.8 Information Lost

Information lost is defined as "amount of non-confidential information that is *unnecessarily* eliminated, as well as, in the case of perturbation methods, the statistical quality of the information provided to the users" [2]. This means that in the case of perturbation methods the term information lost corresponds to precision. In other words: the higher the precision is the lower the information lost is. This applies also to the randomizing method.

The situation is different for the query restriction methods. Information lost can be very high for the query set size restriction and for the query overlap restriction approaches.

The auditing method restricts by its nature only queries that can lead to compromise. Following the definition of the information lost (see above) we can say that this method causes no information loss. However, the price that must be paid for this is very high (see Section 3.5).

It is more difficult to give an estimation of information lost for the partitioning and cell suppression methods because it depends on data in the database. Some empirical results (see [26]) show that information loss can be very severe for partitioning. In order to reduce it, *dummy records* have been proposed.

3.9 Conclusions

In summary we can say that the randomizing method has more than one advantage in comparison with other methods. It assures high security, robustness, and precision if the improvements described in this section are used. Another very important advantage is that the method is among the lowest in cost.

The disadvantages are: its bias (however it tends to be small for large sizes of query sets); it is not suitable for categorical data; it is not suitable for some types of queries.

References

[1] Achugbue J.O. and F.Y. Chin. "The effectiveness of output modification by rounding for protection of statistical databases." *INFOR Journal*, Vol. 17, No. 3, August 1979, pp. 209-218.

[2] Adam N.R. and J.C. Wortmann. "Security-Control Methods for Statistical Databases: A Comparative Study." *ACM Computing Surveys*, Vol. 21, No. 4, December 1989, pp. 515-556.

[3] Beck L.L. "A security mechanism for statistical databases." *ACM Trans. Database Syst.*, Vol. 5, No. 3, September 1980, pp. 316-338.

[4] Chin F.Y. and G. Özsoyoğlu. "Security in partitioned dynamic statistical databases." *In Proceedings of the IEEE COMPSAC*, 1979, pp. 594-601.

[5] Chin F.Y. and G. Özsoyoğlu. "Statistical database design." *ACM Trans. Database Syst.*, Vol. 6, No. 1, March 1981, pp. 113-139.

[6] Cox L.H. "Suppression methodology and statistical disclosure control." *Journal of American Statistical Association*, Vol. 75, No. 370, June 1980, pp. 377-385.

[7] Denning D.E. "Secure statistical databases with random sample queries." *ACM Trans. Database Syst.*, Vol. 5, No. 3, September 1980, pp. 291-315.

[8] Denning D.E., P.J. Denning and M.D. Schwartz. "The tracker: A threat to statistical database security." *ACM Trans. Database Syst.*, Vol. 4, No. 1, March 1979, pp. 76-96.

[9] Denning D.E. and J. Schlörer. "Inference control for statistical databases." *Computer*, Vol. 7, No. 16, July 1983, pp. 69-82.

[10] Dobkin D., A.K. Jones and R.J. Lipton. "Secure databases: Protection against user influence." *ACM Trans. Database Syst.*, Vol. 4, No. 1, March 1979, pp. 97-106.

[11] Fellegi I.P. "On the question of statistical confidentiality." *Journal of American Statistical Association*, Vol. 67, No. 337, March 1972, pp. 7-18.

[12] Friedman A.D. and L.J. Hoffman, "Towards a fail-safe approach to secure databases." *In Proceedings of the IEEE Symposium on Security and Privacy*, 1980.

[13] Hoffman L.J. and W.F. Miller. "Getting a personal dossier from a statistical data bank." *Datamation*, Vol. 16, No. 5, May 1970, pp. 74-75.

[14] Jaklič J. "Protecting statistical databases by randomizing and other methods: Comparison and simulation." *Master thesis*, Dept. of Computer Science, University of Houston, December 1992.

[15] Lefons D., A. Silvestri and F. Tangorra. "An analytic approach to statistical databases." *In Proceedings of 8th International Conference on Very Large Databases*, 1983, pp. 260-273.

[16] Leiss E.L. "Protecting statistical databases through randomizing." *Technical Report #UH-CS-81-07*, University of Houston, December 1981.

[17] Leiss E.L. "Randomizing, a practical method for protecting statistical databases against compromise." *In Proceedings of 8th International Conference on Very Large Databases*, 1982, pp. 189-196.

[18] Leiss E.L. "Principles of data security." *Plenum Press*, 1982.

[19] Liew C.K., W.J. Choi, and C.J. Liew. "A data distortion by probability distribution." *ACM Trans. Database Syst.*, Vol. 10, No. 3, pp. 395-411.

[20] Matloff N.E. "Another look at the use of noise addition for database security." *In Proceedings of the IEEE Symposium on Security and Privacy*, 1986.

[21] Morris J.L. "Computational methods in elementary numerical analysis." *John Wiley & Sons*, 1983.

[22] Palley M.A. and J.S. Simonoff. "The use of regression methodology for compromise of confidential information in statistical databases." *ACM Trans. Database Syst.*, Vol. 12, No. 4, December 1987, pp. 593-608.

[23] Reiss S.P. "Practical data-swapping: The first steps." *ACM Trans. Database Syst.*, Vol. 9, No. 1, March 1984, pp. 20-37.

[24] Reiss S.P. "Practical data-swapping: The first steps." *In Proceedings of the IEEE Symposium on Security and Privacy*, 1980.

[25] Schlörer J. "Information loss in partitioned statistical databases." *Comput. Journal*, Vol. 26, No. 3, 1983, pp. 218-223.

[26] Schlörer J. "Disclosure from statistical databases: Quantitative aspects of trackers." *ACM Trans. Database Syst.* Vol. 5, No. 4, December 1980, pp. 467-492.

[27] Schlörer J. "Confidentality of statistical records: A treat monitoring scheme of on-line dialogue." *Methods Inform. Med.*, Vol. 1, No. 15, 1976, pp. 36-42.

[28] Traub J.F., Y. Yemini and H. Wozniakowski. "The statistical security of a statistical database." *ACM Trans. Database Syst.*, Vol. 9, No. 4, December 1984, pp. 672-679.

Deep Knowledge and Domain Models

Jarmo J. Ahonen
Lappeenranta University of Technology, P.O.Box 20, 53851 Lappeenranta, Finland
jarmo.ahonen@lut.fi

Keywords: modeling, simulation, deep knowledge, domain models

Edited by: Jerzy R. Nawrocki
Received: September 30, 1994 **Revised:** Januray 26, 1995 **Accepted:** May 5, 1995

An approach to the concept of deep knowledge is outlined. The approach is based on the assumption that the depth of knowledge results from its explanatory powers. After considering some examples of deep and shallow knowledge and defining deep knowledge and robustness, an approach to the development of domain models based on deep knowledge is proposed. The proposed approach is based on the Salmonian concept of causal processes and it provides a uniform point of view to knowledge of physical domains and domain modeling. The approach is developed in order to incorporate structural and causal knowledge directly into numeric models because qualitative approaches seem to have philosophical problems.

1 Introduction

Since the Chandrasekaran and Mittal article (Chandrasekaran and Mittal, 1983) most writers concerned with so-called domain models have been using the term *deep knowledge*. Unfortunately, the term has not been defined very well and its use has depended on vague notations of its nature. The vagueness of the meaning of the term invites us to try to define it more clearly, and one approach to the concept of deep knowledge and its nature and use in domain modeling will be discussed later in this paper. The approach is developed because qualitative techniques seem to have some philosophical problems (Ahonen, 1994) and there obviously is a need for the incorporation of the best features of quantitative and qualitative approaches. In addition to this some of the most obvious consequences of the use of deep knowledge for the overall usefulness and structure of domain models will be discussed.

In fault diagnosis and simulation there is one obvious requirement for the domain model, namely that it should be able to function like the modeled phenomenon in every case. Unfortunately, how this can be achieved is not entirely clear, despite some results and authors speaking in favor of deep knowledge. The main problem with deep knowledge is that we do not know what it is although we know what it should do, i.e. provide better robustness for domain models.

The use of the term deep knowledge seems to be closely related to the idea that we cannot create practical AI systems without turning our attention to physical reality around us. Hayes proposed the use of models of the physical reality as a method of avoiding the problem of too simplistic problem domains (Hayes, 1979). The formalization of the physical world and models created from the physical world should, according to Hayes, have the following characteristics (Hayes, 1979):

- Thoroughness, i.e. it should be able to cover the whole range of everyday phenomena (the formalization will not, of course, be perfectly thorough).
- Fidelity, i.e. it should be reasonably detailed.
- Density, i.e. the ratio of facts to concepts should be high.
- Uniformity, i.e. there should be a common framework for the whole formalization.

In addition to the interest in physical domains, Hayes said that reasoning alone is not enough, from which it may be concluded that an intelligent system would include both reasoning and knowledge of physical reality. This dual view leads us to adopt the idea of *model based reasoning*, according to which the descriptions of domain and reasoning are kept separate.[1] The model based reasoning approach is more a methodology than a technique, because it does not determine the implementation (Saarenmaa, 1988). The model based approach seems to be especially useful in fault diagnosis and other fields that require the use of real-world models (see, e.g., de Kleer (1987), Koton (1985), Nardi and Simons (1986), Xiang and Srihari (1986), Rich and Venkatasubramanian (1987), Cross (1984), and Adams (1986)).

One interesting possible approach to deep knowledge is to consider the depth of knowledge possessed by human experts. The human performance is especially interesting because human experts seem to be able to estimate the functioning of a physical system even in cases that have not been encountered earlier and because studies of cognitive processes of human experts seem to suggest that they, i.e. human experts, use some kind of domain models in their reasoning (e.g. Sweller's (1988) results support this belief). In this paper we will consider the nature of deep knowledge from the perspective of scientific knowledge. Scientific knowledge was chosen because there are human experts succesfully applying scientific knowledge to tasks which require deep knowledge and robustness. Possible examples vary from engineers designing new equipment to doctors diagnosing patients. Hence scientific knowledge provides a promising starting-point for our considerations.

At this point it must be stressed that the considerations presented in this paper are applicable only to domain modeling. All other aspects like knowledge used by natural robust creatures are left out. In addition to that restriction, the discussion is limited to physical domains only. The domain we are interested in is physical reality, i.e. the concrete world, in which case pure theory building and abstract entities are omitted. The main reason to consider physical domains only is that there obviously is a need for a method which could be used to incorporate causal and structural knowledge into numeric simulation models because qualitative models do not work wery well with all physical domains (Ahonen, 1994).[2]

In the next section we will define some of the key terms and concepts.

2 Models

In order to make the following discussion clearer we will define several concepts, some of which have been already used. In this paper the term *physical reality* means the concrete world. Similarly, a *real-world phenomenon* or a *real-world object* is a phenomenon or an object which exists or can potentially exist in physical world. A *model* is a representation of physical reality. Such representations are simplifications of physical reality, and unfortunately they do not produce reliable information on every aspect of the phenomenon being modeled (Lewis and Smith, 1979, 2). Our definition of models differs from the normal definition in its clear nature as a representation of physical reality (see, e.g., (Futo and Gergely, 1990) for the usual approach). *Simulation* is the creation and execution of dynamic models employed for understanding system behavior (Fishwick, 1992). These definitions restrict the applicability of the following discussion.

Sometimes it may be the case that the model is used to simulate a situation that has not been considered during the development of the model. Such a *new situation* is a case in which the model is intended to be used without prior knowledge of the possibility that such a case may be encountered during the use of the model. A model is *robust* if it behaves like its real counterpart would behave in a similar new situation. In other words, a robust model produces right values to the variables used to represent the modeled real-world phenomenon or object.

Models should be designed in a way that they do not lose their modeling ability even in new si-

[1]Note that according to Clancey (1992) *all* expert systems are model based.

[2]One of the fundamental points to remember when dealing with physical domains is that scientific knowledge differs from everyday knowledge (or intuitive knowledge) (Tuchanska, 1992), and it would seem fairly strange to consider physical causality and intuitive mythical causality, which has been discussed by de Kleer and Brown (1984), to be the same. In addition to that, the philosophical discussion presented in (Ahonen, 1994) suggests that qualitative approaches are not very good for modeling some types of continuous physical systems.

tuations. The obvious way to achieve this goal is to develop models by using knowledge that is required to determine the behavior of the model in as many new situations as possible and do it despite the fact that models are simplifications (note that the models of human experts are also simplifications). We will call such knowledge deep knowledge because it can be said to go beyond the surface of the knowledge on which our models are based. *Deep knowledge* is knowledge of the features that define the structure and behavior of the phenomenon or the object considered. Deep knowledge provides the basis on which robustness may be built.

This definition of deep knowledge is usable, however it leaves two important questions open. The first question concerns the actual nature of deep knowledge, and the other question the structure of models that are based on deep knowledge.

Despite the superficial differences both AI systems and simulation systems include models, simulation systems by definition, and according to Clancey (1992) all AI systems include models. Hence the distinction between knowledge representation in the traditional AI sense and knowledge representation in the traditional simulation sense is not very clear, although some rough divisions may be made on the basis of the intended usage of the models (Miller et al, 1992). Because the connection between both types of models is very strong, we can proceed as if they were the same.[3] The sameness will be assumed because the approach we will outline later in this paper will be the same for both AI and simulation modeling. The approach outlined in this paper is based on the philosophical discussion and proposal presented in (Ahonen, 1994). In order to approach modeling tasks from a clearer point of view it is necessary to outline how modeling tasks proceed. One way to look at such tasks is to divide them into different steps or levels. For example Zeigler (1976) divides modeling tasks into the following levels:

- the "real system", a source of potentially observable data;
- the "experimental frames", a set of limited observation or manipulation cases for the real system;
- the "base model", a comprehensive model of the system in every experimental frame;
- the "lumped model", a simplified version of the base model, which is simplified in a way that still provides the reliability of the original base model in interesting cases;
- the "computer model", an implementation of the lumped model in a computer programming language on a machine.

[3]There is clearly a need for a representation which could really connect both approaches (Kuipers, 1993).

The above steps of traditional modeling include the assumption that there is no model which is *complete*, i.e. so thorough a replica of the modeled object or phenomenon that it could not be improved as a result of changes in knowledge about original or experienced defects in the developed model. This is clearly expressed by Zeigler (1976, 31) who concludes that:

> In any realistic modeling and simulation area, *the base model description can never be fully known.* (original emphasis)

The definition of robustness and the inherent incompleteness of models mean that no really robust models exist. But for pragmatic reasons it is a compelling and useful aim to make models more robust.

In the next section we will briefly look at the connection between shallowness, robustness and the explanatory powers of the knowledge used in the creation of models.

3 Shallow models, explanation, and robustness

In the field of the philosophy of science, one of the most active topics of interest is the explanative nature of scientific knowledge. For our purposes we will consider the explanatory aspect of scientific knowledge, and suppose that scientific knowledge consists mainly of explanations. (This assumption seems to be a safe one because the explanatory nature of scientific knowledge tends to be so widely accepted a proposition, see e.g. (Fetzer, 1981), (Niiniluoto, 1983), (Salmon, 1971), and (Salmon, 1984), that it is not seriously questioned at all.) In order to avoid a multitude of arguments we will assume that concepts are considered to be defined in such explanations.

To clarify the connection between robustness and the explanatory powers of models we will briefly consider so-called shallow models, which are often thought to be the opposite of deep models (see e.g. Chandrasekaran and Mittal (1983)). Although we will not define shallow models explicitly, we will briefly consider some of their features. In this paper the term *shallow-model* is interpreted in a way that makes clear that shallow models include primarily shallow knowledge acquired by, for example, empirical observations, which are then written into rules or equations that define the values of variables by relations between those values.[4] If we consider shallow models to include only empirical knowledge acquired by observing the surface behavior of an object or a phenomenon, then we can assume that the features of such models differ from the features of the respective deep-knowledge models. In order to emphasize the distinction we will briefly consider some simple examples of models that provide desired robustness or do not provide it.

Consider, as an example, a situation where a human being cannot get enough vitamin A. It is well known that too little vitamin A causes illness, and it is similarly well known that by giving more vitamin A to the person in question, his/her health can be improved. From a limited amount of knowledge it can be concluded that by giving more and more vitamin A the person's health will improve infinitely. It is, however, known that this is not the case. Knowledge of the biological low-level effects of vitamin A would have enabled the avoidance of the wrong conclusion. This example is, however, fairly weak because a fully covering statistical example could be created reasonably easily. Although a covering model could be created, that statistical model would suffer from the same shortcomings as the next example.

As an example of shallow models which completely lack explanatory power we can consider a model of the growth of pines. Having gathered a great number of observations, it is possible to generate a function, and say that

> 'According to our empirical observations we can give the statistical relationship between the height growth and the age of pines as
>
> $$h(t) = \frac{H}{1 + h_1 e^{-h_2 t}} \tag{1}$$
>
> where H (max. height), h_1 and h_2 are 21 m, 20.4 m and 0.064, respectively'

which is true. But by saying (as Oker-Blom et al (1988)) that

> 'The height growth was assumed to be independent of stand density and was modeled as a logistic curve (1). The values of H (maximum height), h_1 and h_2 were chosen as 21 m, 20.4 m and 0.064, giving a height development in accordance with existing growth and yield tables.'

and implicitly supposing that the equation explains something, we say something that is not true. If we say that the equation states the statistical relationship between h and t, we do not say anything that is untruthful but we do not provide any explanation, either. In addition to not being able to provide any explanations, the model is not robust. The lack of robustness can be easily pointed out by asking "What will happen to the growth if the climate really changes as they say?". The presented statistical model cannot answer that question.

Obviously Oker-Blom's model does not have great explanatory power in the sense required by the question:

> 'Why does the equation (1) hold for the height growth and the age?' (2)

The question (2) to which a human expert could easily provide an answer, cannot be answered by the model — although that question is one of the most obvious to be asked.

Although knowledge present in Oker-Blom's model can explicitly determine a model, it does not even implicitly answer the question (2). The fact that models of this type do not include the knowledge required for adequate explanations

[4] In other words, a shallow model is like a "black-box" which responds to external stimuli and produces values to interesting variables according to predefined mappings from the stimuli to the values. The inside of the box cannot be seen in the case of shallow models.

does not mean that explanations do not exist at all. The problem of non-existence of explanations in statistical relations has been noted earlier (Fetzer, 1981, 87; Salmon, 1984).

It is actually easy to find examples of cases in which obvious statistical relations have nothing to do with explanations or actual mechanisms of the case. We can, for instance, take Salmon's (1984, 268) example:

> '...there is a strict positive correlation between the amount of time required for clothes hung out on a line to dry and the distance required to get an airplane off the ground at the nearby airport. I take it as given that the higher the relative humidity, the longer the clothes will take to dry. Thus the phenomenon that requires explanation is the fact that increased relative humidity tends to make for greater takeoff distance, other things — such as the temperature, the altitude of the airport, the type of the plane and the load it is carrying — being equal...'

In the example chosen by Salmon there is no direct explanatory connection between the observed phenomena. The actual explanation gives the common reason for the phenomena. The interesting thing is that in addition to providing the necessary knowledge, the explanation hints at the possibility of creating a model of the concepts used in the explanation. Such a model could be used in a variety of cases which involve physical phenomena and objects modeled — in other words, such a model could be fairly robust. Note that in Salmon's example there is no way of creating a covering statistical model which could offer any insight into the correlation.

Although statistical analysis of shallow knowledge is often very useful, the unfortunate fact is that sometimes even strict correlations or conditional probabilities of 1 or 0 do not mean that there is an explanatory relation. It can, at first, seem strange to say that $P(A|B) = 1$ or $P(A|B) = 0$ does not mean that there is any connection between A and B, but it is fairly easy to find examples which show that the claim is true. A great deal of this is implied by the fact that statistical correlation is a symmetric relation while causality is not (Irtzik and Meyer, 1987). Obviously an alternative is requied in order to create robust domain models.

Considering the vitamin A example and Oker-Blom's equation and Salmon's airport-example and proper explanations for those cases[5], it is interesting to note that such explanations have to explain the observed phenomena by using concepts and entities which are not parts of the original description of the case. The same can be said of any physical phenomena which is explained using Newtonian mechanics because Newtonian mechanics is not present in the description of e.g. the famous apple often said to be connected with the late Newton. It seems to be the case that those explanations use deep knowledge, i.e. knowledge that goes beyond the observed surface features (i.e. the values of the actually interesting variables) of the case and the original description of the situation.

Problems with shallow models lead us to look for other approaches. As was said, one of the most important aspects of knowledge is to answer why-questions, questions which often imply some kind of causality. Unfortunately the concept of causality are very difficult to define, but we will, however, briefly outline causality and explanation. We will continue on to the assumption that causal relations exist.

Thus far we have briefly considered the connection between explanatory power and robustness. In the following section we will outine one possible approach to the conceptualization of physical reality. The approach differs somewhat from the usual object-phenomena -models.

4 Causality and explanation

The use of causality in different forms is actually very tempting. The concept of causal relations is acceptable at face value an and cognitively plausible. Actually, different levels of explanatory powers and probability in causal explanations seem to exist (see e.g. Sober (1984)), which implies the versatile nature of causal explanation.

Probably the most useful approach to define the features of the physical reality spoken about is proposed by Salmon (1984), who proposes

[5]The explanations are left out because such case-specific aspects are outside the scope of this paper.

the concept of causal processes instead of causal events. Of causal processes he says:

> 'Causal processes .propagate the structure of the physical world and provide the connections among the happenings in the various parts of space time. Causal interactions produce the structure and modifications of structure that we find in the patterns exhibited by the physical world. Causal laws govern the causal processes and causal interactions, providing the regularities that characterize the evolution of causal processes and the modifications that result from causal interactions' (Salmon, 1984, 132)

The concept of causal processes is not, however, very clear; many of their features have to be defined with more precision. Those features include mark transmission, structure transmission, the principle of causal influence, and causal interaction.

The most important criterion for a causal process is its ability to transmit a mark. By mark transmission we mean that a causal process can transmit a mark from point A to point B (and every point between them) without further interactions. The *mark transmission* (MT) is defined, in a more explicit way (Salmon, 1984, 148) , as:

> Let P be a process that, in the absence of interactions with other processes, would remain uniform with respect to a characteristic Q, which it would manifest consistently over an interval that includes both the space-time points A and B ($A \neq B$). Then, a mark (consisting of a modification of Q into Q'), which has been introduced into process P by means of a single interaction at point A, is transmitted to point B if P manifests the modification Q' at B and all stages of the process between A and B without additional interventions.

Note that marks in a process are, actually, changes in the process itself. Therefore we can say that a transmission of a mark is a transmission of the changed structure of the process transmitting the mark, and a process can always be said to transmit its own structure, changed by a mark or not. The principle of *structure transmission* (ST) can be formulated as follows (Salmon, 1984, 154):

> If a process is capable of transmitting changes in structure due to marking interactions, then that process can be said to transmit its own structure.

The fact that a process does not transmit a particular type of mark does not mean that it is not a causal process. Consider, as an example, the processes of a hard rubber ball and a particular beam of light (caused by a lamp and colored white). It is possible to paint a green mark on the surface of the ball but it is not possible to do the same to the beam of light, although it is possible to change the color of the beam to green by using a green filter. Marks must be consistent with the structure and properties of causal processes — a causal process cannot be marked by every method.

In accordance with the principle of structure transmission there must be a way to define how a causal process propagates causality from one space-time locale to another. The *principle of causal influence* (PCI) can be defined as (Salmon, 1984, 155):

> A process that transmits its own structure is capable of propagating a causal influence from one space-time locale to another.

Combined together the concepts MT, ST and PCI define what a causal process is. Although a causal process can be very effectively defined by them, no interactions between processes have been defined. Obviously processes interact and interactions constitute the actual structure of causal relations.

As can be deduced from the definitions of MT, ST and PCI, there exist, in addition to causal processes, a great number of non-causal processes. The existence of non-causal processes makes the definition of causal interaction (CI) between processes quite difficult task, and one proposition is made by Salmon (1984, 171), and is as follows:

> Let P_1 and P_2 be two processes that intersect with one another at the space-time point S, which belongs to the histories of both. Let Q be a characteristic that process P_1 would exhibit throughout an interval (which includes subintervals on both sides of S in the history

of P_1) if the intersection with P_2 did not occur; let R be a characteristic that process P_2 would exhibit throughout an interval (which includes subintervals on both sides of S in the history of P_2) if the intersection with P_1 did not occur. Then the intersection of P_1 and P_2 at S constitutes a causal interaction if:

1. P_1 exhibits the characteristic Q before S, but it exhibits a modified characteristic Q' throughout an interval immediately following S; and
2. P_2 exhibits the characteristic R before S, but it exhibits a modified characteristic R' throughout an interval immediately following S.

Salmon proposed that scientific explanations should use the concept of causal processes as the basis, and that actual explanations are linguistic descriptions of chains of causal interactions and mark transmissions. The discussion of how explanations are actually created by using the causal processes -concept is not presented in this paper — for that discussion we refer to (Salmon, 1984). In the following section we will consider a modeling approach that is based on the concept of causal processes and briefly discuss the intuitively tempting claim that models able to provide explanatory knowledge are robust.

5 Causal process models

In this section we will describe a domain modeling approach based on the Salmonian concepts of causality. Generally we will consider models that may be called explanatory models. An *explanatory model* is constructed according to the concept of causal processes and the model produces quantitative results, i.e. values to variables, based on processing the descriptions of the modeled causal processes. Note that we are considering models which are not qualitative in the meaning discussed in (Weld and de Kleer, 1990). Because qualitative models of the physical reality suffer from philosophical problems (Ahonen, 1994), we have to develop an alternative to normal qualitative and quantitative approaches. Our approach includes causal[6] and structural knowledge directly into numeric simulation models.

In order to prevent any confusion between explanatory inferences (i.e. the mechanism by which explanations are generated) and explanatory models (i.e. models of the causal structure of the reality) we can consider the sentence 'I see that there is snow outside and therefore I know that the temperature outside is below zero degrees celsius'. The sentence is not an explanatory model, and even the sentence 'I see that there is snow outside and therefore I know that because water can be snow only if the temperature is below zero, I am able to deduce that the temperature outside is below zero' is not an explanatory model. Writing that 'When the temperature is below zero° C, water molecules arrange into a combination in which they do not constitute a liquid' we are much nearer to having an explanatory model. Note that a transmission of energy causes water molecules to rearrange – an obvious case of causal interaction and mark transmission. In the next subsection we will outline a simple formalization of explanatory models.

5.1 The structure of explanatory models

If we consider any practically useful model, it is very probable that such a model would include several processes which may interact with each others. In addition to interacting with other processes from time to time, it is interesting to note that causal processes P_1 and P_2 can be in continuous interaction. Consider, as an example of continuous causal interactions, a car. The motor of the car always has several different causal interactions with other parts of the car, and there are, in the motor, parts which can be thought to be causal processes continuously interacting with each others. An explanatory model M (without the interactions) could now be said to be a set of

[6]At this point it must be stressed that in this paper we are discussing physical domains and physical causality only. This is partly because the approach presented is developed to overcome the problems encountered by qualitative modeling of physical reality, and the knowledge considered is our best knowledge about the physical world, i.e. scientific knowledge, not naive knowledge used in qualitative modeling approaches like the mythical causality discussed by de Kleer and Brown (1984).

causal processes, i.e.

$$M = \{P_1, P_2, \ldots, P_n\}$$

in which P_i denotes a causal process included into the model. In order to have interactions, as defined in the previous section, between various processes an explanatory model must be redefined as

$$M = (P, I)$$

in which P is the set of the modeled processes and I is the set of the possible interactions between them. Every member of I is a CI.

Another interesting feature of causal processes is that they are divisible. Consider, again, a car. For a pedestrian injured by the car, the car constitutes one process, and for a mechanic the car consists of a multitude of different causal processes. The example of the car shows that we can divide a process into several processes depending on what we know of it. A causal process can sometimes be divided into several subprocesses which have causal interactions and causal structures of their own.

Because some processes can be divided into subprocesses, a process should be clearly distinguishable from its subprocesses by means other than just having different characteristics. One possible method for this separation is the introduction of the concept of levels; on lower levels there would be the subprocesses of process P_j and P_j itself would be on the upper level. Now the process P_j can be formalized as

$$P_j = \{P_{j,1}, P_{j,2}, \ldots, P_{j,n}\}$$

in which $P_{j,i}$ is a *subprocess* of P_j.

In principle the number of levels of an explanatory model is not restricted. Different models require different numbers of levels, and the number of levels of a model is restricted by the limitations dictated by the computer on which the model is used or by the fact that sometimes the required accuracy can be achieved by using fewer levels, or by the fact that human knowledge of the problem may not be complete enough.

It is interesting to note that the concept of levels seems to a be natural solution for the modularity problem. Different processes existing on different levels fulfill the modularity requirements presented by Cota and Sargent (1992).

In order to define the characteristics of a processes there must be a way to identify that particular process. Therefore it is reasonable to consider the characteristics of a process to be identified by an intensional name (the name represents the intension of the intended real counterpart existing or potentially existing in the physical reality). The name can be considered to identify the variables and the values of the variables and the characteristic space-time functions which constitute the features according to which the process is defined and characterized. Now a process is formalized as

$$P_j = < N, S >$$

in which S is the set of subprocesses and N is the name of P_j.

A process is characterized by the values of the variables and the variables themselves and the space-time functions which are used to represent the process in question. The name of the process is used to clarify the recognizable features of the process in order to make the characterization more comprehensible. In order to include the actual process characteristics in the model, we can define a process as a tuple

$$P_j = < N, V, F, S >$$

in which N and S are as above, V is the set of the variables used to represent the characteristics of P_j, and F is the set of space-time functions which define the characteristics of P_j together with V. Note that a model of a causal process does not require any names in principle, because an actual process is characterized by the variables and the values of the variables and the space-time functions which may change the values of the variables (the space-time functions are required in order to enable specific time-related characteristics like Polonium^{218}'s tendency to lose mass over time). Processes are named in order to make the model more understandable and easier to use and construct. Now an explanatory model can be defined as a tuple

$$M = < P, I >$$

as above.

In this solution the variables in V and the functions in F are the method by which the characteristics of a given process are represented. These characteristics include the structure of the process and possible marks, i.e. changes to the structure of the process, and transmit the structure and the marks over space-time. If the process definition includes a variable for the color of a rubber ball, that ball will be *marked* by painting it and changing the value of that variable. That change would then be transmitted over the simulated space-time and the representation would fulfil the definition of MT. Similarly the "variables with functions" representation obeys the principle of structure transmission, and by being able to transmit their own structure the representations are able to propagate causal influence from one space-time locale to another and hence fulfil the definition of PCI. Naturally every $I_k \in I$ must fulfil the definition of CI.

Obviously the movement of a car, which is a causal process, has an effect on the doors of the car — at least the effect of movement. Or consider a stone hitting one of the doors. Obviously causal processes on different levels can have causal interactions with each others. It seems actually to be the case that there is no limit for the level number difference between processes which have causal interactions with each others.

In some cases the causal interaction is constant, but sometimes it is only a possibility. The possibility of the causal interaction exists between any cup and any table, but it actually exists only if the causal processes in question fulfill certain characteristics (one example of that kind of characteristics is the space-time locations of the processes). This makes I_k more a definition of possible causal interactions than the actual interaction.

It can be said that the definitions of interactions are intentions of interactions and actual interactions are their extensions. The general definition of an interaction could be thought to be a demon which causes a real interaction to happen when certain requirements are satisfied.

5.2 Remarks on explanatory models

In the subsection above we defined the structure of explanatory models and some other concepts. In this subsection we will consider some special semantic and syntactic features of our explanatory model structure, and try to clarify its intended interpretation.

In the explanatory model structure a causal process is a set of variables and functions. The set is named by a term, and the naming relation defines the characteristics of the process connected to the term. The mapping from the name to the set defines the characteristics of the process. The mapping represents the intension of the name, and the set of the variables and functions to which the mapping maps the name represents the extension of the name. This means that a certain process, i.e. the set of variables and space-time functions, can be named by several terms at a time and that set of terms can change over time. As the consequence of the possibility of multiple names, N must be defined as a *set of names*.

One result of this characterization is that causal interactions and characterizations themselves can make the process to fall under different characterizations at different times. Consider, as an example, a steel plate. If the plate is crushed into another form, it is not a steel plate any more although it is still steel and still the same process. Before crushing, the steel plate was named by "steel" and "plate", but after crushing that process is no longer named by "plate" although it is still the same process. Actual processes do not disappear although they can change into forms which may be very different when compared to the original forms. Physical relations are not static. A part of a machine can be removed, or a physical entity can be broken into several pieces. Causal interactions can change the physical relationships between processes. This requires great flexibility from the simulation control program which should allow such changes and maintain the reliability of the system despite those changes.

Two models of two different chairs have very much in common despite the fact that the chairs can be very different in some respects. A model of every physical entity can include knowledge of atoms and particles, and knowledge of atoms and particles is very similar in any case. The relation between knowledge used in two models could be said to mean that causal processes on deeper levels are more and more alike conceptually, although physically different. Two processes may have very similar structure and definition. If models T_1 and T_2 are both models of pines, then they have very similar knowledge on their lower levels. In other words, knowledge used in a model is more common to other models on lower levels than it is on upper levels.

In order not to give the impression that the only useful type of explanatory models includes knowledge of even subatomic relations, we have to point out a quite obvious way of determining where to stop modeling. In every case we start modeling by using a scientific theory according to which we model. During the deepening of the model we will always end with a level which cannot be defined without using a scientific theory that is different from the theory according to which we started modeling. Such a change of theories provides a suitable point in which we may stop modeling and still be able to provide required robustness. If we are modeling a mechanical device, e.g. a clock, we may be able to stop modeling when we have reached a level on which we have exhausted the means of Newtonian mechanics. Explaining that level would require entirely another kind of theory, and in many cases such theories are not necessary in order to achieve the required robustness or explanatory power. Note, however, that the number of scientific theories used in a model depends on the pragmatic aspects of the modeling project, and those aspects depend on the case considered.

There is often no need to compute the actual causal interaction on every level of the model. The actual amount of causal interactions required to be computed depends on the model and the use of the model. Consider, for example, the familiar fact that a metal table will support a coffee cup. It is clear that the explanation of this fact involves the details of atomic structure and the appeal to concepts as unfamiliar to everyday experience as the Pauli exclusion principle (the example is taken from (Salmon, 1984)). An explanatory model of the situation can be created, but a precise model would include the atomic structure, and the workings of the atomic structure would appear to be extremely difficult to be computed.

The computational problem can be avoided by hiding the atomic structure of the model from normal computations. Such hiding could be done by using the method described in (Ahonen and Saarenmaa, 1991). If computations are necessary, the causal processes and interactions needed could be created when required and disposed after the need has been met. The hiding of the more precise structure of a model in order to avoid computational problems seems to promise a solution to the question of computational effectiveness of causal models, see e.g. (D'Ambrosio et al, 1985).

5.3 The robustness of explanatory models

Although we would like to show that the structure of explanatory models does provide robustness, we have to admit that it is not easy. The best we are able to achieve is to discuss some features of explanatory models and the intuitively robust nature of those features.

A model should be able to function properly in new situations. From the nature of models it is clear that robustness is actually provided by the knowledge already present in the model, and the same seems to be the case with the robustness present in estimations made by human experts. This makes us to adopt a view very similar to Hacking's (Hacking, 1967, 319). According to Hacking you do not actually know something if you have not performed the necessary inferences required to deduce it.

If we assume that the robustness of models is due to knowledge which is used in a new way, we can understand the role of deep knowledge better. If our assumption of the growing similarity of deep knowledge is true, then it is tempting to conclude that new combinations of such basic knowledge, i.e. deep knowledge, provide the robustness. Since very deep knowledge of one model is very alike to similarly deep knowledge of another model, it may be the case that such knowledge can be easily arranged in combinations that provide the required robustness at the upper levels of the model.

We have considered the reasons for the robustness of models that use deep knowledge, but we have not paid any attention to the robustness of our explanatory models. We claim that explanatory models may provide a uniform approach to domain knowledge, explanatory power and robustness, because explanatory models employ only one concept — the concept of causal processes. The concept of causal processes enables the modeler to use uniform representations and a uniform way of thought throughout the development of the model and maintain the explanatory power of scientific knowledge. Causal processes provide a method by which more or less deep knowledge may have similar conceptualization and representation notwithstanding its depth.

The concept of causal processes has originally been developed to provide a uniform approach to the study of scientific knowledge and scientific explanation, and there seems to be no reason to assume that the same approach could not be used in domain modeling. The causal process approach with a suitable formalization may provide a very promising alternative by defining what kind of knowledge should be used and what kind of interactions models include. Considering the nature of causal processes it is reasonable to believe that representations of processes enable very different and unanticipated causal interactions and mark transmissions to occur.

In the next section we will briefly discuss the creation of explanatory models.

6 Building an explanatory model

In this section we will briefly outline how an explanatory model could be built. Althought the actual process of building such models is not covered here, we will outline some of the major issues and pitfalls.

6.1 Real systems and experimental frames

The choice of the domain to be modeled may seem to be an easy task. Unfortunately it is not — especially if we consider the requirements for the domain.

Earlier we restricted our interest to the physical domains, i.e. the concrete world. If we consider physical domains only, we have to drop the modeling of purely theoretical things like mathematics, arts and other non-concrete artifacts. In other words, the domain must be one you can take a piece off (probably not literally, but in the same meaning).

In addition to being limited to the physical world as his/her domain, the modeler has additional limitations to his/her view of the structure of the world. As discussed above, the modeler will consider *processes* in space-time continuum — and those processes must be *causal processes* in the Salmonian sense.[7] The most significant feature of causal processes is their different nature from normal conceptualization of processes in modeling, especially in AI modeling (for a more usual approach, see e.g. (Drabble, 1993)) in which processes are considered to be series of events occuring in the world of objects. This is not the case when the world is conceptualized as causal processes.

The causal process conceptualization of the physical world has no objects and *no* events in the traditional sense.[8] Every physical object or phenomenon is thought to be a *causal process* or a *causal interaction* between such processes. Processes are *not* series of events, they are the basic *structure* of the world on which everything else is based. Processes have characteristics which identify them and which make them behave in their specific ways, but they are not objects.

The fundamental difference between objects and causal processes is in their connection to the physical reality. It is much easier to decide that objects may be almost anything, not only something which really exists in the physical world. One of the main reasons for using causal processes when considering the world is that the definition of causal processes strictly defines what there is in the physical world. Normal object-oriented approaches do not impose such restrictions on the suitable domain. In order to make the nature of the modeled reality and the usage of knowledge unambiguous such restrictions are, in our opinion, required. Such restrictions are needed especially if we want to maintain a coherent structure and uniformly interpretable representation in our models.

The strict definition of the domain to be modeled and thorough thinking prior to commencing the actual modeling project will be started are

[7]Note that *all* qualitative approaches to continuous causal processes suffer from the problems of qualitative modeling discussed in (Ahonen, 1994). All qualitative approaches handle continuity in the same way (Bobrow, 1984), and hence approaches like the "Qualitative Process Theory" of Forbus (1984) suffer from the same problems. Therefore Qualitative Process Theory does not provide satisfactory means to achieve our goal, i.e. to be able to represent causal and structural knowledge in models which produce reliable numeric simulation results.

[8]This makes causal ordering and other considerations presented by Iwasaki and Simon (1994) unnecessary when dealing with physical systems. In this paper we do not, however, attempt to say anything about the usability of those considerations when modeling other domains.

even more necessary when modeling by using causal processes than other approaches. The exclusion of all features and concepts which do not exist in the strictly defined world of causal processes is a necessary step in order to make the creation of explanatory models successful. This makes it necessary to focus the possible experiments on the definition of the characteristics of causal processes existing or potentially existing in the modeled domain and the possible causal interactions between them. Such experiments may not be easily planned or carried out. We do, however, leave the discussion of the identification of causal processes existing in a specific domain out of this paper, for such considerations we refer to (Salmon, 1984).

In the next subsection we will briefly consider what specific features the usage of the causal process concept and explanatory models will require from the base model created from the chosen domain and the experiments done to it.

6.2 Base model

The definition of the domain and the possible experiments carried out in order to find out more about the domain are themselves the first step in the creation of the base model. It is very difficult and probably impossible to make a distinction between the choice of the domain, the planning of the experiments and the creation of the base model.

The definition of the domain to be modeled and the creation of the base model seems to be a circular process. The first step is to decide to model, for example, a tree and the second step is to start the creation of the base model of the tree. During the creation of the base model the modeler has to identify the causal processes and decide to model a specific subdomain of the original domain (a needle is a good example of such subdomains if the original domain is a pine).

The close relation between theory building and the choice of what experiments to conduct has been discussed in the literature of the philosophy of science, and it seems to be the case that it is very difficult to identify the order in which those phases appear. Hence we will leave that discussion out of this paper and only refer to the discussion in the field of the philosophy of science.

In the next subsection we will briefly discuss the creation of the lumped model from the previously defined base model. In addition to that, we shortly consider the implementation of explanatory models.

6.3 Lumped model and computer implementation

Unfortunately the base model of the domain cannot ever be fully known, and that makes it very difficult to produce robust models by simplifying theories which are already incomplete. Fortunately it seems to be the case that scientific knowledge is, in many cases, fairly robust.

Many scientific theories are fairly robust,[9] and in our opinion they are robust because they attempt to offer a testable explanation to the questions regarding the structure and behavior of reality[10]. Such theories are, of course, members of one type of models. The apparent use of theories in explanation may help us to reduce the possibly very large set of domain theories into an implementable lumped model.

As discussed above, the most viable point to stop modeling is when the modeler should change the theory of the domain. If an explanatory model of a mechanical device has been defined according to Newtonian theory and in order to deepen the model the modeler would need to use quantum mechanics, then that level would be an obvious level to stop modeling. In that way several layers of scientific theories of the domain could be dropped from the lumped model, although those theories are important parts of the base model. Note that the base model could, in our opinion, be created by using multiple levels of specification and theoretical accuracy — of course all the theories should be able to be used to model the structure and behavior of the domain according to the concept of causal processes.

The actual computer implementation of an explanatory model would not be an obvious task to be completed by using a conventional language like FORTRAN 77. The most interesting alter-

[9]This is because they are themselves conceptually or empirically testable, either directly or indirectly (Bunge, 1973, 27-43).

[10]This discussion is outside the scope of this paper and its fundamental aspects and problems are left to people who have specialized in those problems. In this paper we attempt only to use one specific point of view of the scientific knowledge, namely that scientific knowledge consists of *explanations*.

natives seem to be various object-oriented languages, but even with them there are some conceptual problems. The most difficult problem with object oriented languages is that they are mainly event-oriented and causal processes are, by definition, continuous and the simulation of causal processes should proceed over constantly flowing time, not as series of events.[11]

In order to make the task of implementing an explanatory model easier it might be a good idea to develop a specified programming environment and a specific language for the task. The language should directly support the concept of levels and other features of explanatory models. In addition to supporting the features of causal processes, the language should include methodologies to connect the implemented model structure and behavior to an explanation generation system of some type.

7 Discussion

It is plausible that the cognitive models used by humans are very robust because of the human ability to connect different items of knowledge in new ways. If we assume this, it is easy to conclude that robustness of domain models may be based on the same method. The obvious way to introduce robustness to domain models seems, in that case, to depend on the conceptualization and representation of domain knowledge. Such conceptualization should provide a versatile and uniform enough approach to domain knowledge and its representation. In addition to that, the conceptualization should be able to ease the discomfort that Iwasaki and Simon (1993) express about the subjective and arbitrary representation of causality.

The concept of causal processes provides a uniform view to domain knowledge, and it is not very difficult to develop a framework for the modeling of causal processes and their features. This should enable us to follow a clear path in our modeling activities, and it is reasonable to assume that models with enough levels and accurate descriptions of processes and the features of processes are robust. Unfortunately this is not obvious, although most of the writers that have discussed deep knowledge have adapted a similar point of view to modeling with deep knowledge.

The uncertainty concerning the robustness provided is due to that we do not actually know how human experts generate new combinations of knowledge. If it is done by methods that work like normal logic or straightforward simulation, then there should not be any problems. But if the only method is induction, our attempts to develop models robust enough to be comparable to the cognitive models used by human experts may be void.

Another hazard with the use of the concept of causal processes may be that the concept of causal processes seems to be extremely difficult or even impossible to formalize fully with descriptive methods. It may be possible that the nonlogical modalities and counterfactuals present in the definition of causal processes prevent exact formalization (Fetzer, 1987). This problem does not, however, make the use of the concept of causal processes impossible, it may only complicate it by giving more responsibility to the modeler.

The modeler's difficulties should, however, be manageable because the structure of explanatory models is surprising simple. By using that simple structure and uniform conceptualization of physical reality, the creation of robust domain models should be possible and resulting models would fulfil the requirements defined by Hayes (1979). Thoroughness is an inherent feature of causal processes because they are the basic structure of the physical reality, and fidelity and density are directly supported by the definition of the explanatory model structure. In order to make such models work, the amount of detail has to be fairly high and a working model would include relatively many facts when comparing their number to the number of concepts. Uniformity is, also, an inherent feature of explanatory models because everything existing in the physical reality is modeled by using the causal processes conceptualization.

One additional benefit of the concept of causal processes is that if quantitative models were developed according the approach, it could be possible to provide a natural connection between numeric simulation models and the representation of physical causality and structural knowledge. If

[11] A direct application of the methods used in qualitative simulations of continuous processes, e.g. QSIM (Kuipers, 1985), is not very promising because those methods are more or less directly connected with the philosophical assumptions behind qualitative modeling.

there were a method to naturally include causal and structural knowledge into numeric simulation models, it should be much easier to provide the connection requested by Clancey (1992) and Kuipers (1993). That connection could enable us to create numeric simulation models which would include the best features of both qualitative and quantitative approaches and which would not suffer from the shortcomings of qualitative approaches discussed in (Ahonen, 1994).

References

Adams, T. L. (1986). Model-Based Reasoning for Automated Fault Diagnosis and Recovery Planning in Space Power Systems. In *Proc. 21st Intersociety Energy Conversion Engineering Conference.* San Diego.

Ahonen, J. J. (1994). On Qualitative Modelling. *AI & Society*, 8. 17-28.

Ahonen, J. J. and Saarenmaa, H. (1991). Model-based reasoning about natural ecosystems: An algorithm to reduce the computational burden associated with simulating multiple biological agents. In *Proc. 6th Symposium on Computer Science for Environmental Protection.* 193-200.

Bobrow, D. G. (1984). Qualitative Reasoning about Physical Systems: An Introduction. *Artificial Intelligence*, 24. 1-5.

Bunge, M. (1973). *Method, Model and Matter.* D Reidel Publishing Company, Dordrecht.

Chandrasekaran, B. and Mittal, S. (1983). Deep Versus Compiled Knowledge Approaches to Diagnostic Problem Solving, *International Journal of Man-Machine Studies*, 22. 425-436.

Clancey, W. J. (1992). Model Construction Operators. *Artificial Intelligence*, 53. 1-115.

Cota, B. A. and Sargent, R. G. (1992). A Modification of the Process World View. *ACM Transactions on Modeling and Computer Simulation*, 2. 109-129.

Cross, S. E. (1984). Model-based Reasoning in Expert Systems: An application to enroute air traffic control. In *Proc. AIAA/IEEE 6th Digital Avionics Systems Conference.* Baltimore.

D'Ambrosio, B., Fehling, M. and Adams, T. (1985). Levels of Abstraction in Model Based Reasoning. In *Proc. ROBEXS '85 (Robotics and Expert Systems).* NASA. Johnson Space Center.

de Kleer, J. and Williams, B. C. (1987) Diagnosing Multiple Faults, *Artificial Intelligence*, 32. 97-130.

de Kleer, J. and Brown, J. S. (1984). A Qualitative Physics Based on Confluences. *Artificial Intelligence*, 24. 7-83.

Drabble, B. (1993). EXCALIBUR: a program for planning and reasoning with processes. *Artificial Intelligence*, 62. 1-40.

Fetzer, J. H. (1987). Critical Notice: Wesley Salmon's Scientific Explanation and the Causal Structure of the World, *Philosophy of Science*, 54. 597-610.

Fetzer, J. H. (1981). *Scientific Knowledge.* D Reidel Publishing Company. Boston.

Fishwick, P. A. (1992). An Integrated Approach to System Modeling Using a Synthesis of Artificial Intelligence, Software Engineering and Simulation Methodologies. *ACM Transactions on Modeling and Computer Simulation*, 2. 307-330.

Forbus, K. D. (1984). Qualitative Process Theory. *Artificial Intelligence*, 24. 85-168.

Futo, I. and Gergely, T. (1990). *Artificial Intelligence in Simulation.* Ellis Horwood. Chishester.

Hacking, I. (1967). Slightly More Realistic Personal Probability. *Philosophy of Science*, 34. 311-325.

Hayes, P. J. (1979). The Second Naive Physics Manifesto. In: Michie, D. (ed) *Expert Systems in the Micro-Electronic Age.* Edinburgh University Press. Edinburgh.

Irtzik, G. and Meyer, E. (1987). Causal modeling: New Directions for Statistical Explanation, *Philosophy of Science*, 54. 495-514.

Iwasaki, Y. and Simon, H. A. (1994). Causality and model abstraction. *Artificial Intelligence*, 67. 143-194.

Iwasaki, Y. and Simon, H. A. (1993). Retrospective on "Causality in device Behavior". *Artificial Intelligence*, 59. 141-146.

Koton, P. A. (1985). Empirical and Model-Based Reasoning in Expert Systems. In *IJCAI-85 Proceedings*. Los Angeles.

Kuipers, B. J. (1993). Qualitative simulation then and now. *Artificial Intelligence*, 59. 133-140.

Kuipers, B. J. (1985). The Limits of Qualitative Simulation. In: *Proceedings of International Joint Conference on Artificial Intelligence*. 128-136.

Lewis, T. G. and Smith, B. J. (1979). *Computer Principles of modeling and Simulation*. Houghton Mifflin. Boston.

Miller, D. P., Firby, R. J., Fishwick, P. A., Franke, D. W. and Rothenberg, J. (1992). AI: What Simulationists Really Need To Know. *ACM Transactions on Modeling and Computer Simulation*, 2. 269-284.

Nardi, B. A. and Simons, R. K. (1986). *Model-Based Reasoning and AI Problem Solving*. IntelliCorp Technical Article Reprint. IntelliCorp.

Niiniluoto, I. (1983). *Scientific Reasoning and Explanation* (in Finnish). Otava. Keuruu.

Oker-Blom, P., Kellomaki, S., Valtonen, E. and Vaisanen, H. (1988). Structural development of Pinus sylvestris Stands with Varying Initial Density: a Simulation Model, *Scandinavian Journal of Forest Research*, 3. 185-200.

Rich, S. H. and Venkatasubramanian, V. (1987). Model-Based Reasoning in Diagnostic Expert Systems for Chemical Process Plants, *Computers and Chemical Engineering*, 11. 111-122.

Saarenmaa, H. (1988). Model-Based Reasoning in Ecology and Natural Resource Management. In *Proc. Resource Technology 88: International Symposium on Advanced Technology in Natural Resource Management*. Fort Collins (Colorado).

Salmon, W. C. (ed) (1971). *Statistical Explanation and Statistical Relevance*. University of Pittsburg Press. Pittsburg.

Salmon, W. C. (1984). *Scientific Explanation and the Causal Structure of the World*. Princeton University Press. Princeton.

Sober, E. (1984). Common Cause Explanation, *Philosophy of Science*, 51. 212-241.

Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning, *Cognitive Science* , 12. 257-285.

Tuchanska, B. (1992). What is explained in Science. *Philosophy of Science*, 59. 102-119.

Weld, D. S. and de Kleer, J. (eds) (1990). *Qualitative Reasoning about Physical Systems*. Morgan Kaufmann Publishers, San Mateo, California.

Xiang, Z. and Srihari, S. N. (1986). Diagnosis Using Multi-level Reasoning. In *Proc. Expert Systems in Government Symposium*, McLean (VA USA).

Zeigler, B. P. (1976). *Theory of Modeling and Simulation*. Wiley. New York.

First Call for Papers

The Eighth Australian Joint Conference on Artificial Intelligence (AI'95)
13–17 November 1995

Hosted by
Department of Computer Science
University College, The University of New South Wales
Australian Defence Force Academy, Canberra, ACT 2600, Australia

About AI'95

AI'95 is the Eighth Australian Joint Conference on Artificial Intelligence. The last two conferences were held in Melbourne, Victoria (AI'93), and Armidale, New South Wales (AI'94). This annual conference is the largest Australian AI conference and also attracts many overseas participants. Over 45% of submitted papers to AI'93 and AI'94 came from overseas. The proceedings of previous conferences were published by World Scientific Publishing Co. Ltd.

The main theme of AI'95 is *"bridging the gaps,"* i.e., bridging the gap between the classical symbolic approach and other subsymbolic approaches, such as artificial neural networks, evolutionary computation and artificial life, to AI, and bridging the gap between the AI theory and real world applications. The goals of the conference are to promote cross-fertilisation among different approaches to AI and provide a common forum for both researchers and practitioners in the AI field to exchange new ideas and share their experience. Tutorials and workshops on various topics of AI will be organised before the main conference. A separate call for proposals for tutorials and workshops will be distributed.

Paper Submission

Authors are invited to submit papers describing both theoretical and practical work in any areas of artificial intelligence. (Papers accepted or under review by other conferences or journals are not acceptable.) Topics of interest include, but are not limited to:

Adaptive Behaviours
Artificial Life
Artificial Intelligence Applications
Automated Reasoning
Autonomous Intelligent Systems
Bayesian and Statistical Learning Methods
Cognitive Modelling Computer Vision
Distributed Artificial Intelligence
Evolutionary Learning
Evolutionary Optimisation
Fuzzy Systems
Group Decision Support Systems
Hybrid Systems
Image Analysis and Understanding
Intelligent Decision Support Systems
Knowledge Acquisition
Knowledge-Based Systems
Knowledge Representation
Machine Learning
Natural Language Processing
Neural Networks
Pattern Recognition
Philosophy of AI Planning and Scheduling
Robotics
Speech Recognition

Five hard copies of the completed paper must be *received* by the conference programme committee chair before or on **9 June 1994**. Fax and electronic submission are not acceptable. Papers received after 9 June 1994 will be returned unopened. A **best student paper award** will be given at the conference. The first author of the paper must be a full-time student, e.g., a PhD, MSc, or Honours student. A letter from the head of the student's department, confirming the status of the student, must be submitted along with the paper in order to be considered for the best student paper award. The award includes a $200 cheque and a certificate issued by the AI'95 Programme Committee. Send all paper submissions to:

Dr X. Yao
AI'95 Programme Committee Chair
Department of Computer Science
University College,
The University of New South Wales
Australian Defence Force Academy
Canberra, ACT 2600, Australia
Email: xin@csadfa.cs.adfa.oz.au
Phone: +61 6 268 8819
Fax: +61 6 268 8581

Preparation of Manuscript

All five hard copies must be printed on 8.5 × 11 (inch^2) or $A4$ paper using 12 point *Times*. The left and right margin should be 25mm each. The top and bottom margin should be 35mm each. Each submitted paper must have a separate title page and a body. The title page must include a title, a 300 – 400 word abstract, a list of keywords, the names and addresses of all authors, their email addresses, and their telephone and fax numbers. The body must also include the title and abstract, but the author information must be excluded. The length of submitted papers (excluding the title page) must be no more than 8 single-spaced, single-column pages including all figures, tables, and bibliography. Papers not conforming to the above requirements may be rejected without review.

Programme Committee

Dr. Xin Yao (Chair), UNSW/ADFA
Prof. Zeungnam Bien, KAIST, Korea
Mr. Phil Collier, University of Tasmania
A/Prof. Paul Compton, UNSW
Dr. Terry Dartnall, Griffith University
Prof. John Debenham, University of Technology, Sydney
Dr. David Dowe, Monash University
Dr. David Fogel, Natural Selection, Inc., USA
A/Prof. Norman Foo, University of Sydney
A/Prof. Matjaz Gams, Jozef Stefan Institute, Slovenia
Prof. Ray Jarvis, Monash University
A/Prof. Jong-Hwan Kim, KAIST, Korea
Prof. Guo-Jie Li, NCIC, PRC
Dr. Dickson Lukose, University of New England
Dr. Bob McKay, UNSW/ADFA
Prof. Zbigniew Michalewicz, UNC-Charlotte, USA
Mr. Chris Rowles, Telecom Research Laboratories
Dr. John Slaney, Australian National University
Prof. Rodney Topor, Griffith University
Dr. Chi Ping Tsang, University of Western Australia
Dr. Olivier de Vel, James Cook University of North Queensland
Dr. Geoff Webb, Deakin University
Dr. Wilson Wen, Telecom Research Laboratories
A/Prof. Kit Po Wong, University of Western Australia
Dr. Chengqi Zhang, University of New England

Organising Committee

Dr. Bob McKay (Chair), UNSW/ADFA
Dr. Jennie Clothier, Defence Science and Technology Organisation
Dr. Richard Davis, Commonwealth Scientific and Industrial Research Organisation (CSIRO)
Mr. Warwick Graco, Health Insurance Commission
Dr. Tu Van Le, University of Canberra
Mr. John O'Neill, Defence Science and Technology Organisation
Mr. Peter Whigham, UNSW/ADFA
Dr. Graham Williams, CSIRO
Dr. Xin Yao, UNSW/ADFA

Conference Location

AI'95 will be held at ADFA (Australian Defence Force Academy) in Canberra, the capital city of Australia. ADFA is located less than 5km from the CBD of Canberra.

Proposal Submission

Tutorials on various AI related topics will be organised on 13 and 14 November 1995 in parallel with pre-conference workshops. The length of each tutorial is 3 hours. Proposals dealing with any AI related topics are solicited. Application-oriented tutorials are particularly welcome, especially those relating to topics of interest to Canberra's large administrative sector. Topics of interest include, but are not limited to:

Three hard copies of the tutorial proposal must be *received* by the tutorial/workshop coordinator at the following address before or on **7 April 1995.**

Dr. J. R. Davis
AI'95 Tutorial/Workshop Coordinator
CSIRO Division of Water Resources
P O Box 1666
Canberra City 2601, Australia
Email: `richardd@cbr.dwr.csiro.au`
Phone: +61 6 246 5706
Fax: +61 6 246 5800

Further Information

Further information about AI'95 can be obtained by emailing the following address (preferred):

`ai95@adfa.edu.au`

or by contacting the organising committee chair Dr. Bob McKay at the following address:
Dr Bob McKay
AI'95 Organising Committee Chair
Department of Computer Science
University College, The University of New South Wales
Australian Defence Force Academy
Canberra, ACT 2600, Australia
Email: `rim@csadfa.cs.adfa.oz.au`
Phone: +61 6 268 8169
Fax: +61 6 268 8581

Workshop Proposal Submission

AI'95 continues the tradition of organising high standard pre-conference workshops. Some of previous workshops have resulted in either journal special issues or published proceedings. However, workshops which focus on informal talks and discussions are equally welcome. We invite potential workshop organisers to submit their proposals for pre-conference workshops to AI'95. The workshop organisers are only responsible for technical issues such as sending out their CFPs, reviewing submitted papers, inviting speakers, and preparing the programme. The conference organising committee will look after all the organisational issues such as venue booking, registration, accommodation, etc.

All workshops will be held on 13 and 14 November 1995 in parallel with tutorials. The length of a workshop should be at least half a day and at most two days. Three copies of the proposal must be received by the tutorial/workshop coordinator Dr. J.R. Davis before or on **7 April 1995.**

Important Dates

7 April 1995 Deadline for Workshop Proposals.
5 May 1995 Notification of Proposal Acceptance.
9 June 1995 Deadline for Paper Submission.
21 July 1995 Notification of Acceptance.
18 August 1995 Camera Ready Copy.
9 October 1995 Camera Ready Copy of Workshop Papers.
13–14 November 1995 Tutorials/Workshops.
15–17 November 1995 Conference Sessions.

This is the Final Call For Papers for a special journal issue of INFORMATICA on the topic:

MIND <> COMPUTER

[i.e. Mind NOT EQUAL Computer]

In this special issue we want to reevaluate the soundness of current AI research positions (especially the heavily disputed strong-AI paradigm) as well as pursue new directions aimed at achieving true intelligence. This is a brainstorming special issue about core ideas that will shape future AI. We are interested in critical papers representing all positions on the issues.

The first part of this special issue will be a small number of invited papers, including papers by Winograd, Dreyfus, Michie, McDermott, Agre, Tecuci etc. Here we are soliciting additional papers on the topic.

TOPICS: Papers are invited in all subareas and on all aspects of the above topic, especially on:

- the current state, positions, and advancements achieved in the last 5 years in particular subfields of AI,
- the trends, perspectives and foundations of natural and artificial intelligence,
- strong AI versus weak AI and the reality of most current "typical" publications in AI,
- new directions in AI.

TIME TABLE AND CONTACTS: Papers in 5 hard copies should be received by May 15, 1995 at one of the following addresses (please, no e-mail/FAX submissions):

North & South America:
Marcin Paprzycki
`paprzycki_m@gusher.pb.utexas.edu`
Department of Mathematics and
Computer Science
University of Texas of the Permian Basin
Odessa, TX 79762, USA

Asia, Australia:
Xindong Wu
`xindong@insect.sd.monash.edu.au`
Department of Software Development,
Monash University
Melbourne, VIC 3145, Australia

Europe, Africa:
Matjaz Gams
`matjaz.gams@ijs.si`
Jozef Stefan Institute, Jamova 39
61000 Slovenia, Europe

E-mail information about the special issue is available from the above 3 contact editors.

The special issue will be published in late 1995.

FORMAT AND REVIEWING PROCESS: Papers should not exceed 8,000 words (including figures and tables but excluding references. A full page figure should be counted as 500 words). Ideally 5,000 words are desirable.

Each paper will be refereed by at least two anonymous referees outside the author's country and by an appropriate subset of the program committee.

When accepted, the authors will be asked to transform their manuscripts into the Informatica LaTeX style (available from `ftp.arnes.si`; directory `/magazines/informatica`).

More information about Informatica and the Special Issue can be accessed through URL: `ftp://ftp.arnes.si/magazines/informatica`.

ERK'93
Electrotechnical and Computer Conference
Elektrotehniška in računalniška konferenca
September 25–27, 1995

Conference Chairman
Baldomir Zajc
University of Ljubljana
Faculty of Electr. Eng. and Comp. Science
Tržaška 25, 61000 Ljubljana, Slovenia
Tel: (061) 1768 349, Fax: (061) 264 990
E-mail: baldomir.zajc@fer.uni-lj.si

Conference Vice-chairman
Bogomir Horvat
University of Maribor, FERI
Smetanova 17, 62000 Maribor, Slovenia
Tel: (062) 25 461, Fax: (062) 212 013
E-mail: horvat@uni-mb.si

Program Committee Chairman
Saša Divjak
University of Ljubljana
Faculty of Electr. Eng. and Comp. Science
Tržaška 25, 61000 Ljubljana, Slovenia
Tel: (061) 1768 349, Fax: (061) 264 990
E-mail: sasa.divjak@fer.uni-lj.si

Programe Committee
Tadej Bajd
Saša Divjak
Janko Drnovšek
Matjaž Gams
Ferdo Gubina
Marko Jagodič
Jadran Lenarčič
Drago Matko
Miro Milanovič
Andrej Novak
Nikola Pavešić
Franjo Pernuš
Jurij Tasič
Borut Zupančič

Publications Chairman
Franc Solina
University of Ljubljana
Faculty of Electr. Eng. and Comp. Science
Tržaška 25, 61000 Ljubljana, Slovenia
Tel: (061) 1768 349, Fax: (061) 264 990
E-mail: franc@fer.uni-lj.si

Advisory Board
Rudi Bric
Dali Djonlagić
Karel Jezernik
Peter Jereb
Marjan Plaper
Jernej Virant
Lojze Vodovnik

Call for Papers

for the fourth **Electrotechnical and Computer Conference ERK'95**, to be held from September 25–27, 1995 in Portorož, Slovenia. Official languages are Slovene and English.

The following areas will be represented at the conference:

- *electronics,*
- *telecommunications,*
- *automatic control,*
- *simulation and modeling,*
- *robotics,*
- *computer and information science,*
- *artificial intelligence,*
- *pattern recognition,*
- *biomedical engineering,*
- *power engineering.*
- *measurement techniques.*

The conference is being organized by the **Slovenian Section of IEEE** and other Slovenian professional societies:

- Slovenian Society for Automatic Control,
- Slovenian Measurement Society (ISEMEC 93),
- SLOKO-CIGRE,
- Slovenian Society for Medical and Biological Engineering,
- Slovenian Society for Robotics,
- Slovenian Artificial Intelligence Society,
- Slovenian Pattern Recognition Society.

Authors who wish to present a paper at the conference should send three copies of full paper to Prof. S. Divjak. The paper should include:

1. the title of the paper,
2. author's address, telephone, fax and e-mail,
3. the paper's subject area.
4. the paper should be max. 4 pages long in Slovene or English.

Authors of accepted papers will have to prepare a four-page camera-ready copy of their paper for inclusion in the proceedings of the conference.

Time schedule:	Submission due	*July 24, 1995*
	Notification of acceptance	*August 24, 1995*

CALL FOR PAPERS
International Conference on Software Quality
ICSQ '95
November, 6 - 9 1995
Maribor, Slovenia

Organised by:
University of Maribor
Faculty of Electrical Engineering and Computer Science,
Faculty of Business and Economics,
Slovenia Section IEEE,
Association of Economics Maribor,
Slovene Society Informatika

Objectives

The aim of ICSQ '95 is to provide a platform for technology and knowledge transfer between academia, industry and research institutions in the software quality field, by:

- the introduction and discussion new research results in software quality,
- offering the practising quality engineers an insight into the results of ongoing research,
- acquainting the research community with the problems of practical application.

Topics

Some important topics for the Conference include, but are not limited to the following:

- quality management systems (QMS),
- metrics,
- process improvement,
- risk Management,
- methodologies,
- verification & validation methods,
- quality planning,
- QMS tools,
- total quality management (TQM),
- audits systems,
- human factors in quality management,
- standards.

Instructions for Authors

Four copies (in English) of the original work, not longer than 4000 words (10 pages), should be submitted to the Scientific Conference Secretariat before May 15th, 1995. Papers should include a title, a short abstract and a list of keywords, the author's name, address and title should be on a separate page. All papers received will be refereed by the International Program Committee. The accepted papers will be published in the Conference Proceedings and will be available to the delegates at the time of registration. The language of the conference will be English.

Important Dates

May 15th, 1995	Full paper
June 30th, 1995	Notification of final acceptance
October 1st, 1995	Camera Ready Copy

Conference Location

The town Maribor was founded in the 12th century. Today it is the second largest town of Slovenia, located in its North, close to the Austrian border. Many businessmen and tourists enjoy the variety of cultural, sports and gastronomic possibilities of the town. Maribor is surrounded by vineyards and has one of the largest wine-cellars in this part of Europe. Maribor is easily accessible by international air lines from the airport of Brnik (Slovenia) and from Graz (Austria).

Conference Organisation

Organising Chairperson:
Marjan Pivka
Faculty of Business and Economics
Razlagova 14, Maribor 62000
Slovenia
Tel.: +386 62 224 611
Fax.: +386 62 227 056
Email: `pivka@uni-mb.si`

Programme Chairperson:
Ivan Rozman
Faculty of Electrical Engineering and Computer Science
Smetanova 17, Maribor 62000
Slovenia
Tel.: +386 62 25 461, +386 62 221 112
Fax: +386 62 227 056
Email: `i.rozman@uni-mb.si`

Conference Secretariat:
Miss Cvetka Rogina
Association of Economics Maribor
Cafova ulica 7, 62000 Maribor
Slovenia
Tel.: +386 62 211 940
Fax.: +386 62 211 940

Programme Committee

Boris I. Cogan, Institute for Automation and Control, Vladivostok (Russia)
Sasa Dekleva, DePaul University (USA)
Matjaz Gams, J. Stefan Institute, Ljubljana (Slovenia)
Hannu Jaakkola, Tampere University of Technology (Finland)
Marjan Pivka, University of Maribor (Slovenia)
Heinrich C. Mayer, University of Klagenfurt (Austria)
Erich Ortner, University of Konstanz (Germany)
Ivan Rozman, University of Maribor (Slovenia)
Franc Solina, University of Ljubljana (Slovenia)
Stanislaw Wrycza, University of Gdansk (Poland)
Joze Zupancic, University of Maribor (Slovenia)

First Announcement and Call for Papers

Lukasiewicz in Dublin

University College Dublin
8-10 July 1996

The year 1996 marks the 40th anniversary of the death of one of this century's foremost logicians and historians of logic, Jan Lukasiewicz. In this year the Joint Session of the Aristotelian Society and the Mind Association will be held in Dublin, where Lukasiewicz spent his last years, and we intend to use this occasion to celebrate the work of Lukasiewicz at a conference on 8-10 July 1996 in University College Dublin, immediately after the Joint Session.

The conference is being organised by the Department of Philosophy UCD, and the European Society for Analytic Philosophy. The programme will consist of plenary lectures by a number of invited speakers and workshops consisting of papers of 30-40 minutes duration.

Speakers will include Czeslaw Lejewski (Manchester), Grzegorz Malinowski (Lodz), Witold Marciszewski (Warsaw), Graham Priest (Queensland), Peter Simons (Leeds), Timothy Smiley (Cambridge), Alan Weir (Belfast), and Jan Wolenski (Cracow).

We welcome submissions for papers (30-40 minutes) on topics related to any area of Lukasiewics's work. Authors should submit 3 copies of their abstract, 2 sides of A4 double spaced, by 30 November 1995.

All correspondence should be directed to:
Dr. Maria Baghramian
Dept. of Philosophy
University College Dublin
Dublin 4, Ireland
Tel: +353-1-7068125
Fax: +352-1-2693469
E mail: `Baghram@macollamh.ucd.ie`
Maria Baghramian
Dept. Of Philosophy
University College Dublin
Ireland

Announcement and Call For Papers

GLOCOSM – Global Conference on Small & Medium Industry & Business

3-5 January 1996
Bangalore, India

Organized by
SDM Institute for Management Development, India
and Indiana University Purdue University Fort Wayne, U.S.A.

Educators, executives and government officials from around the world are invited to participate in this unique conference whose theme is "Small & Medium Industry & Business in the New Global Environment: Prospects & Problems." Papers, abstracts or symposium/workshop proposals related to the theme are solicited. Submissions in accounting, commerce, economics, finance, human resources management/organizational behaviour, information systems, operations management, quantitative methods/statistics, strategy, environmental/ethical/legal issues, public sector management, social/cultural issues, technology management and other topics relevant to the global community of management professionals scholars are also welcome.

Deadline for submission of contribution: 15 April 1995.

The Address of General Chair

Dr. B.P. Lingaraj
General Chair - GLOCOSM
Department of Management & Marketing
Indiana University Purdue University
Fort Wayne, IN 46805, U.S.A
E-mail: `lingaraj@cvax.ipfw.indiana.edu`

For other details (instructions for the authors etc.) please contact the member of the Programme Committee:
Professor Ludvik Bogataj
University of Ljubljana
Faculty of Economics
Kardeljeva ploščad 17, P.O.Box 103
61000 Ljubljana
Slovenia
Phone: +386 (061) 168-33 -33
Fax: +386 (061) 301-110
E-Mail: `ludvik.bogataj@uni-lj.si`

Machine Learning List

The Machine Learning List is moderated. Contributions should be relevant to the scientific study of machine learning. Mail contributions to `ml@ics.uci.edu`. Mail requests to be added or deleted to `ml-request@ics.uci.edu`. Back issues may be FTP'd from `ics.uci.edu` in `pub/ml-list/V<X>/<N>` or N.Z where X and N are the volume and number of the issue; ID: anonymous PASSWORD: <your mail address> URL- `http://www.ics.uci.edu/AI/-ML/Machine-Learning.html`

THE MINISTRY OF SCIENCE AND TECHNOLOGY OF THE REPUBLIC OF SLOVENIA

Address: Slovenska 50, 61000 Ljubljana, Tel.: +386 61 1311 107, Fax: +386 61 1324 140.
WWW:http://www.mzt.si
Minister: Prof. Rado Bohinc, Ph.D.
State Secretary for Int. Coop.: Rado Genorio, Ph.D.
State Secretary for Sci. and Tech.: Ciril Baškovič
Secretary General: Franc Hudej, Ph.D.

The Ministry also includes:
The Standards and Metrology Institute of the Republic of Slovenia
Address: Kotnikova 6, 61000 Ljubljana, Tel.: +386 61 1312 322, Fax: +386 61 314 882., and
The Industrial Property Protection Office of the Republic of Slovenia
Address: Kotnikova 6, 61000 Ljubljana, Tel.: +386 61 1312 322, Fax: +386 61 318 983.

Scientific Research and Development Potential. The statistical data for 1993 showed that there were 180 research and development institutions in Slovenia. Altogether, they employed 10,400 people, of whom 4,900 were researchers and 3,900 expert or technical staff.

In the past ten years, the number of researchers has almost doubled: the number of Ph.D. graduates increased from 1,100 to 1,565, while the number of M.Sc.'s rose from 650 to 1,029. The "Young Researchers" (i.e. postgraduate students) program has greatly helped towards revitalizing research. The average age of researchers has been brought down to 40, with one-fifth of them being younger than 29.

The table below shows the distribution of researchers according to educational level and sectors (in 1993):

Sector	Ph.D.	M.Sc.
Business enterprises	51	196
Government	482	395
Private non-profit organizations	10	12
Higher education organizations	1022	426
Total	1,565	1,029

Financing Research and Development. Statistical estimates indicate that US$ 185 million (1,4% of GDP) was spent on research and development in Slovenia in 1993. More than half of this comes from public expenditure, mainly the state budget. In the last three years, R&D expenditure by business organizations has stagnated, a result of the current economic transition. This transition has led to the financial decline and increased insolvency of firms and companies. These cannot be replaced by the growing number of mainly small businesses. The shortfall was addressed by increased public-sector spending: its share of GDP nearly doubled from the mid-seventies to 0,86% in 1993.

Income of R&D organizations spent on R&D activities in 1993 (in million US$):

Sector	Total	Basic res.	App. res.	Exp. dev.
Business ent.	83,9	4,7	32,6	46,6
Government	58,4	16,1	21,5	20,8
Private non-p.	1,3	0,2	0,6	0,5
Higher edu.	40,9	24,2	8,7	8
Total	184,5	45,2	63,4	75,9

The policy of the Slovene Government is to increase the percentage intended for R&D in its budget. The Science and Technology Council of the Republic of Slovenia is preparing the draft of a national research program (NRP). The government will harmonize the NRP with its general development policy, and submit it first to the parliamentary Committee for Science, Technology and Development and after that to the parliament. The parliament approves the NRP each year, thus setting the basis for deciding the level of public support for R&D.

The Ministry of Science and Technology is mainly a government institution responsible for controlling expenditure of the R&D budget, in compliance with the NRP and the criteria provided by the Law on Research Activities. The Ministry finances research or co- finances development projects through public bidding, partially finances infrastructure research institutions (national institutes), while it directly finances management and top-level science.

The focal points of R&D policy in Slovenia are:

- maintaining the high level and quality of research activities,
- stimulating collaboration between research and industrial institutions,
- (co)financing and tax assistance for companies engaged in technical development and other applied research projects,
- research training and professional development of leading experts,
- close involvement in international research and development projects,
- establishing and operating facilities for the transfer of technology and experience.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are postgraduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of Slovenia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia
Tel.:+386 61 1773 900, Fax.:+386 61 219 385
Tlx.:31 296 JOSTIN SI
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA
AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS
INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of the original figures on separate sheets and the text on an IBM PC DOS floppy disk or by e-mail – both in ASCII and the Informatica LaTeX format. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 61111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than two years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: ....

Title and Profession (optional): ....

....

Home Address and Telephone (optional): ....

....

Office Address and Telephone (optional): ....

....

E-mail Address (optional): ....

Signature and Date: ....

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatica is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or Board of Referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board and Board of Referees are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Volume 19 Number 2 May 1995 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Contents: