Regular Paper

A Framework for Secure Distributed Workflows

Vlad Ingar Wietrzyk,† Katsuya Tanaka†† and Makoto Takizawa††

Workflow Management Systems (WFMSs) provide an automated framework for managing intra- and inter-enterprise business processes. Workflow is a critical enabler in today's hot technologies, such as portals and e-business. This paper describes the design of a model as well as the architecture to provide support for distributed advanced workflow transactions. We discuss the application of transaction concepts to activities that involve the integrated execution of multiple tasks over different processes. These kinds of applications are described as transactional workflows. A distinguishing feature of the workflow transaction support system proposed is the ability to manage the arbitrary distribution of business processes over multiple workflow management systems. We choose to develop a formal framework for a secure distributed workflow architecture since interworkflow is anticipated as a major supporting mechanism for Business-to-Business Electronic Commerce. We strive to develop a practical logical characterisation of multilevel secure (MLS) distributed workflow for the first time using the inherently difficult concept of non-monotonic reasoning.

1. Introduction

Many technical and non-technical issues hinder enterprise-wide workflow management. Workflow types cannot always be fully predefined, therefore they often need to be adjusted or extended during operation. Distributed workflow execution across functional domains is necessary, but distribution transparency is currently impossible because, different types of Workflow-Management-Systems (WFMSs) implement different WFMS metamodels.

One possible way to enable distributed workflow execution is to build a workflow-management infrastructure integrating different and heterogeneous WFMSs. Users would have access to total funcionality because they access the workflow-management underlying infrastructure, not individual WFMSs. The resulting architecture is general and can accommodate as many WFMSs as required.

Transaction concepts have begun to be applied to support applications or activities that involve multiple tasks of possibly different types - including, but not limited to transactions, and executed over different types of entities - including DBMSs. The architect of such applications may specify inter-task dependencies to define task coordination requirements, and additional requirements for isolation, and failure atomicity of the application. Generally we will refer to such applications as multi-system transactional workflows.

The recent trend to distributed workflow executions requires an even more advanced transaction support system that is able to handle distribution.

To summarise, the new aspects of our approach to security in distributed workflow database management systems include the following research contributions. The novel approach to the development of a practical logical characterisation of multilevel secure (MLS) distributed workflow for the first time using the inherently difficult concept of non-monotonic reasoning. A distinguishing feature of the workflow transaction support system proposed is the ability to manage the arbitrary distribution of business processes over multiple workflow management systems. We chose to develop a formal framework for a secure distributed workflow architecture since we are actively involved in building a prototype of such a system. We also derived a general theorem which must be active when classifying every item of information. We have planned the presentation of the current research as follows. We first present a brief introduction to work on workflow transaction models and discuss extended-relaxed approach to handle workflow transactions in Section 2. Section 3 covers related aspects of workflow distribution and heterogeneity. A number of relaxed transaction models in workflow contexts have been defined recently. These models are per-

† School of Computing, University of Western Sydney
†† Department of Computers and Systems Engineering, Tokyo Denki University

mitting a controlled relaxation of the transaction isolation and atomicity to better match the requirements of various workflow applications. There are discussed in Section 4. In Section 5 we develop a formal model. Also some axioms related to the multilevel secure distributed workflow model are given from which the theorems regarding secure workflow database models are derived. Section 6 provides review of the fundamental concepts of multilevel security and multiversion serialisability theory. In Section 7 the implementation of the secure distributed workflow is covered in greater depth by providing relevant details describing the workflow run-time environment. Section 8 supplies the details related to evaluating the quantitative effects of the workflow system. Section 9 concludes the paper with a summary and a short discussion of future research.

2. Related Work

Some known examples of extended-relaxed transaction models are reported in Refs. 3) and 4).

In the WIDE project [5], a workflow is supported at two transaction levels: global and local. At the global level, the SAGA-based model offers relaxed atomicity through compensation and relaxed isolation by limiting the isolation to the SAGA steps. At this level of granularity, the workflow activities are defined and therefore the grouped workflow activities follow the strict ACID properties. However, the flexibility in assigning transaction properties to workflow activities is limited to the extended SAGA or nested transaction model.

Support for long-duration applications has been independently extended by practitioners and researchers focusing on workflow systems and transaction systems. Extended transaction systems structure a large transaction into sub-transactions and execute them with additional constraints on the individual sub-transactions. Some researchers in workflow systems have proposed the notion of transactional workflow [6]. In a transactional workflow environment, additional correctness requirements can be specified on top of traditional workflow specifications.

The Workflow Management Coalition has specified a standard interface to facilitate the interoperability between different WFMSs [7]. However, they do not address transactional issues with the exception of writing an audit log.

The transaction model used in the Exotica project [8] is based on the SAGA model, but relies on statically computed compensation patterns. As a result, its functionality is limited compared to the work presented in this research paper.

Finally, most commercial products are designed around a centralised database. This database and the workflow engine attached to it—in most cases there is a single workflow engine presenting a point of failure which quickly becomes a bottleneck and is not capable of providing a sufficient degree of fault tolerance.

To summarise, databases and workflow management systems are complementary tools within the corporate computing resources. Databases address the problem of storing and accessing data efficiently. Workflow management systems address the problem of monitoring and coordinating the flow of control among heterogeneous activities.

Very often, a WFMS processes data for which high standards must be set with respect to privacy and data security. Most of the workflow transaction management theory for multilevel secure database systems has been developed for workflow transactions that act within a single security class. In our research work, we look at workflow transactions that act across security classes, that is, the workflow transaction is a multilevel sequence of database commands, which more closely resemble user expectations. We propose a formal model and semantics for interpreting security issues in a workflow architecture which can incooperate a multilevel deductive database.

2.1 An Architecture for Multilevel Secure Workflow Interoperability

Global information management strategies based on a sound distributed architecture are the foundation for effective distribution of complex applications that are needed to support ever changing operational conditions across security boundaries. What we need is a new MLS distributed computing paradigm that can assist users at different locations and at different security levels to cooperate.

A user can initiate a distributed workflow transaction at any site. If access to objects stored at remote sites is required, the distributed workflow transaction initiates a subtransaction at the remote site. To guarantee a correct execution of distributed workflow transactions, each site in the distributed workflow database is under the operation of a concur-

rency control protocol and an atomic commit protocol.

We present the fully distributed architecture for implementing a Workflow Management System (WFMS). An MLS workflow management system consists of a set $\mathcal{N}$ of sites, where each site $N \in \mathcal{N}$ is an MLS database. The sites in the workflow system are interconnected via communication links over which they can communicate. The WFMS architecture operates on top of a Common Object Request Broker Architecture (CORBA) implementation. A CORBA's Interface Definition Language (IDL) can be used to provide a means of specifying workflows. Also we assume that communication links are secure—possibly using encryption. This distributed workflow transaction processing model describes mainly those components necessary for the distribution of a transaction on different domains.

A domain is a unit of autonomy that owns a collection of flow procedures and their instances. In practical terms, a domain might define the scope of a department or division in an organisation. Therefore, flows are grouped by domains and each domain also manages a set of flow procedures installed in the domain. A domain is not defined or limited by networks, processors, or peripherals. The manager of resources can, however, be designed in any fashion, they are exclusively responsible for the ACID properties on their data records. Solely the interface to the components of the distributed workflow model must exist.

If a transaction should be distributed on several domains—a global transaction, in every domain, there must exist the following components, (see **Fig. 1**).

- TM—Transaction-Manager. The transaction manager plays the role of the coordinator in the respective domain. If a transaction is initiated in this domain, the TM assigns a globally unique identifier for it. The TM monitors all actions from applications and resource managers in its domain. In every domain involved in the distributed workflow transaction environment, there exists exactly one TM.
- CRM—Communication-Resource-Manager. Multiple applications in the same domain talk with each other via the CRM. This module is used by applications but also other management components for inter-domain communication. CRM is the most important module with respect to the transactional support for distributed workflow executions. Our model specifies the T*RPC as a communication model, which supports a remote procedure call (RPC) in the transactional environment.
- RM—Resource-Manager. An accountable performer of work. A resource can be a person, a computer process, or machine that plays a role in the workflow system. This module controls the access to one or more resources like files, printers or databases. The RM is responsible for the ACID properties on its data records. A resource has a name and various attributes defining its characteristics. Typical examples of these attributes are job code, skill set, organisation unit, and availability.
- AMS—Administration-Monitoring-Service. The monitoring manager is used to control the workflow execution. In our approach, there is no centralised scheduler. In the figure, each Task Manager—designated as TSM, is equipped with a conditional fragment of code which determines if and when a given task is due to start execution. The scheduler communicates with task managers using CORBA's asynchronous Interface Definition Language (IDL) interfaces. Task managers communicate with tasks using synchronous IDL interfaces as well. AMS module is also responsible for the coordination of the different sites in case of an abort that involves multiple sites. Individual task managers communicate to the monitoring manager their internal states, as well as data object references - for possible recovery.

The distributed architecture suits the inherent distributional character of workflow adequately in a natural way.

This approach also eliminates the bottleneck of task managers having to communicate with a remote centralised scheduler during the execution of the workflow. This architecture also posseses high resiliency to failure—if any one node crashes, only part of the workflow is affected.

3. Distributed Workflow Concepts

Workflow distribution introduces additional levels of requirements. Distributed workflow execution across heterogeneous WFMSs is currently not possible in a transparent way, there-

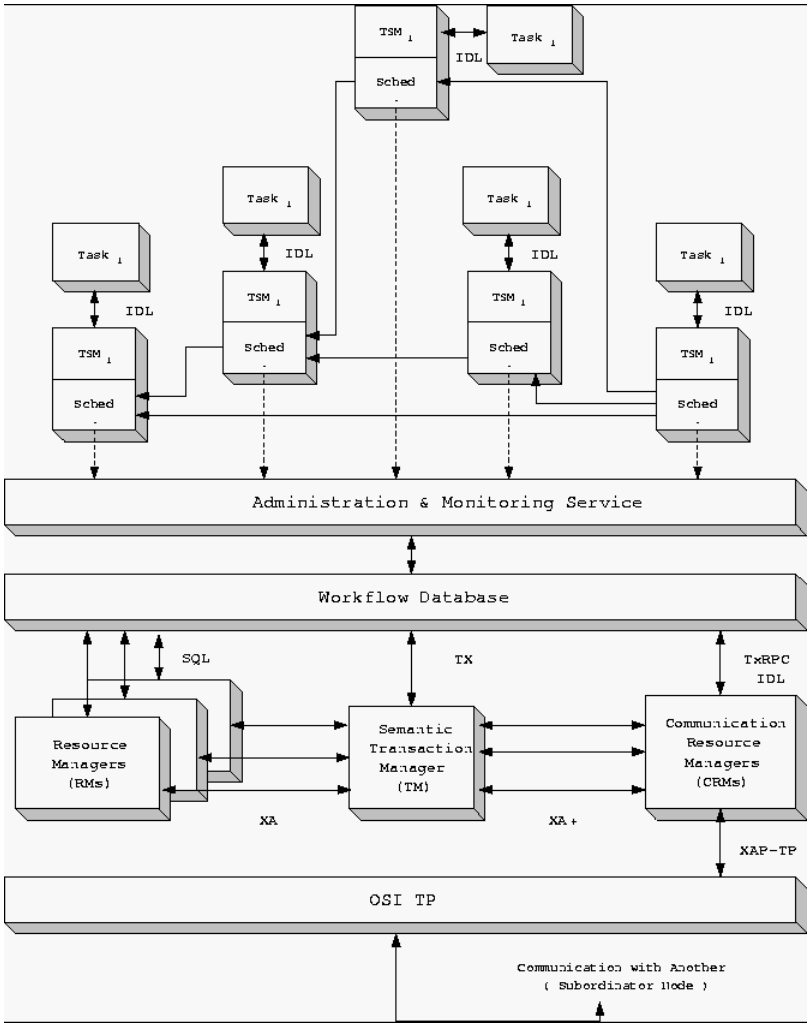

Fig. 1 Distributed workflow architecture.

fore we must consider the problem of workflow funcionality isolation. To efficiently define a work breakdown structure a functional decomposition by means of subworkflows is required. This in turn requires version and variant management, because each reused-workflow-definition alteration might lead to a new variant or configuration of the reusing workflow definition, if it necessitates to stay related to the old version. Dynamic changes of running workflows require a workflow's functional decomposition to change. This might also involve replacing elementary workflow tasks with other composite workflow definitions. Workflow distribution is called homogeneous if the associated WFMSs are of the same type, heterogeneous otherwise. A workflow is distributed when at least two of its objects reside in two different WFMS installations. This is relevant to workflow definitions as well as workflow instances. An often-cited situation is subworkflow distribution, where subworkflows are subject to execution on remote WFMSs. Some variants are possible, such as executing a subworkflow synchronously or asynchronously to the invoking workflow. One of the typical variants involves executing some part of a workflow on one WFMS, and continuing on another (see **Fig. 2**).

If the associated WFMSs, do not know about each other, then it is indirect distribution. In this case, the WFMSs do not implement distribution natively and the system designer must attach distribution functionality to the associated WFMSs. A recognised way is to establish communication buffers between the WFMSs,

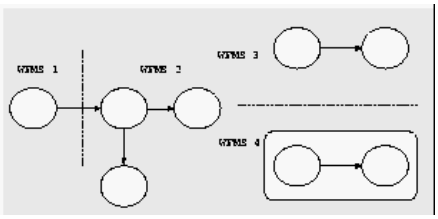

Fig. 2 Workflows division across different WFMSs.

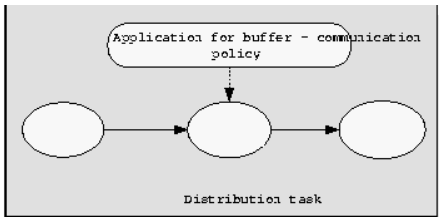

Fig. 3 The distribution task invokes an application for buffer communication.

such as a database or persistent file stores. **Figure 3** shows an example workflow definition with one distribution task. The distribution task invokes an application for buffer communication. Typically, workflow types can be distributed, too.

4. Relaxed Transaction Models in Workflow Contexts

A number of relaxed transaction models have been defined recently that permit a controlled relaxation of the tranaction isolation and atomicity to better match the requirements of various workflow applications. Usually, we will refer to such applications as multi-system transactional workflows. This area has been also influenced by the concept of long running activities.

The intention is to merge advanced transaction technology and workflow management systems to support business processes with well-defined failure semantics and recovery features. Our work is based on an interpretation of the workflow operations from the databases point of view.

4.1 Transactional Workflows

Support for workflow applications has been addressed by researchers focusing on workflow systems and transaction systems. Extended transaction systems structure a large transaction into sub-transactions and execute them with additional precedence requirements between start, commit, abort of the individual sub-transactions. Our approach falls in the category of transactional workflows [6] where additional correctness requirements can be specified on top of traditional workflows specifications. These requirements specify additional constraints on workflow execution schedules. Workflow management systems coordinate the execution of applications distributed over networks. The need for the coordinated execution of workflow steps arises from application as well as data consistency requirements. Flexible transactions work in the context of heterogeneous distributed multidatabase workflow environments [10]. In such workflow environments, each database acts independently from the others. Because a local database can unilaterally abort a transaction, it is not possible to enforce the commit semantics of global transactions. Therefore, flexible transaction were designed to address this problem. The traditional transactions are usually characterised by the atomicity, consistensy, isolation and durability requirements, called the ACID properties of transactions. To better support workflow operational environments, the flexible transaction model relaxed the isolation and atomicity properties. This approach is the direct result of our belief, that tying a workflow system to a particular transaction model, will result in major restrictions that will limit its applicability and usefulness as a workflow tool.

4.2 A Formal Model of Flexible Transactions

From a user's point of view, a transaction is a sequence of actions performed on data items in a database. Flexible transaction models proposed for the distributed workflow environment will increase the failure resiliency of global transactions by allowing alternate subtransactions to be executed when a local database fails or a subtransaction aborts. The approach supports the concept of varied transactions allowing compensatable and noncompenstable subtransactions to coexist within a single global transaction. This transactional environment allows a global transaction to have a weaker (relaxed) form of atomicity, termed semi-atomicity, while still maintaining its correct execution in the workflow. In a workflow multidatabase environment, a local transaction is a set of subtransactions, where each subtransaction is a transaction accessing the data items at a single local site. The concurrency control of global transactions require, that each global transaction has at most one subtransaction at each local site [11]. Following Refs. 10) and 12), the definition of flexible transactions takes the form of

a high-level specification. The flexible transaction model supports flexible execution control flow by specifying two kinds of dependencies among the subtransactions of a global transaction:

- Execution ordering dependencies between two subtransactions.
- Alternative dependencies between two subsets of subtransactions.

In what follows, we shall formally describe the flexible execution control in the flexible transaction model.

Let $\Omega = \{t_1, t_2, \ldots, t_n\}$ be a collection of subtransactions and $\Pi(\Omega)$ the collection of all subsets of Ω. Let t_i, $t_j \in \Omega$ and T_i, $T_j \in \Pi(\Omega)$. Two types of control flow relations are defined on the subsets of Ω and on $\Pi(\Omega)$, namely:

- precedence $t_i \prec t_j$ if t_i precedes t_j $(i \neq j)$;
- preferece $T_i \rhd T_j$ if T_i is preferred to T_j $(i \neq j)$. If $T_i \rhd T_j$, we also declare that T_j is an alternative to T_i.

Both of the above relations, precedence and preference are irreflexive and transitive or more formally, for each $t_i \in \Omega$, $\neg(t_i \prec t_i)$; and for each $T_i \in \Pi(\Omega)$, $\neg(T_i \rhd T_j)$. If $t_i \prec t_j$ and $t_j \prec t_k$, then $t_i \prec t_k$; if $T_i \rhd T_j$ and $T_j \rhd T_k$, then $T_i \rhd T_k$.

From the above definitions, we can see then, the precedence relations determines the correct parallel and sequential execution ordering dependencies among the subtransactions, while the preferece relation determines the priority dependencies among alternate sets of subtransactions for selecting in completing the execution of Ω.

Now a flexible transaction can be defined as follows:

Definition 1. *Flexible transaction A flexible transaction Ω is a set of related subtransactions on which the precedence ($\prec$) and preference ($\rhd$) relations are defined.*

The semantics of the precedence relation refers to the execution order of subtransactions. For example, $t_1 \prec t_2$ may imply that t_2 cannot start before t_1 finishes or that t_2 cannot finish before t_1 finishes. By the same token, the preference relation defines alternative choices and their priority. For example, $\{t_i\} \rhd \{t_j, t_k\}$ may imply that t_j and t_k must abort when t_i commits or that t_j and t_k should not be executed if t_i commits. In this environment, $\{t_i\}$ is of higher priority than $\{t_j, t_k\}$ to be chosen for execution.

We consider that a workflow database state is consistent if it preserves workflow database integrity constraints. As it is the case for traditional transactions, the execution of a flexible transaction as a single unit should map one consistent multidatabase workflow state to another. We designate the relation $(T_i, \prec_i)$ as a partial order of subtransactions. $(T_i, \prec_i)$ is a representative partial order, if the execution of subtransactions in T_i represents the execution of the entire flexible transaction Ω. From the above it is clear that, if $(T_i, \prec_i)$ is a representative partial order, then there are no subsets T_{i1} and T_{i2} of T_i such that $T_{i1} \rhd T_{i2}$. Because each global transaction has at most one subtransaction at a local site, each representative partial order of a flexible transaction must have at most one subtransaction at a local site. In our workflow execution environment, for flexible transactions, the above definition of consistency requires that the execution of subtransactions in each representative partial order must map one consistent workflow multidatabase state to another.

4.3 Scheduling of Flexible Transactions

Since the flexible transaction model was proposed, much research has been devoted to its application. The availability of visible prepare–to–commit states in local database systems is the basic assumption underlying this work. In such an operational environment, the preservation of the semi-atomicity of flexible transactions is relatively easy. As we mentioned in the previous subsection, failures of subtransactions in a flexible transaction are tolerated by taking advantage of the fact that a given function can frequently be accomplished by more than one database system. Also, time used in conjunction with a subtransaction and global transaction can be exploited in transaction scheduling.

A schedulable subtransaction may be submitted for execution to the transaction module. The scheduler first has to check for satisfaction of the preconditions for execution of each subtransaction—it determines whether a subtransaction is schedulable. This entails the specification of the execution dependency among the subtransactions of a global transaction. Execution dependency [6], is a relationship among subtransactions of a global transaction which determines the legal execution order of the subtransactions.

Under normal operational circumstances, the transaction execution state is used to keep track

of the execution of the workflow subtransactions. It is also used to determine if a global workflow transaction has achieved its objectives. When a subtransaction t_i completes the corresponding execution state, x_i is set to S if the subtransaction has achieved its objective, and to F, otherwise. At a certain point of execution, the objectives of the global workflow transaction may be achieved. At that point, the global transaction is considered to be successfully completed and can be committed.

To support the specification of the execution dependency, we define a transaction execution state as follows:

Definition 2. *The transaction execution state x for a global transaction T with m subtransactions, is an m–tuple $(x_1, x_2, \ldots, x_m)$ where:*

$$x_i = \begin{cases} E & \text{if } t_i \text{ is currently being executed;} \\ N & \text{if subtransaction } t_i \text{ has not been submitted for execution;} \\ S & \text{if } t_i \text{ has successfully completed;} \\ F & \text{if } t_i \text{ has failed or completed without achieving its objective;} \end{cases}$$

A number of approaches can be used to assure global serialisability which constitutes a satisfactory correctness criterion for concurrent execution of multidatabase workflow transactions, if there is a lack of additional information about their semantics. The objective of concurrency control is to assure that the serialisation order of multidatabase workflow transactions should be the same, at all sites they execute. It was shown in Refs. 10) and 13), that the above condition is sufficient to assure global serialisability. However, in our workflow operational environment this requirement can be relaxed to require that the relative serialisation order of Workflow Transactions should be the same only at those nodes where they conflict. This would lead to a weaker notion of serialisability; called WT-serialisability, which will be used as our correctness criterion for concurrent execution of Workflow Transactions. We define conflict among workflow transactions if they execute at the same (local) site, and they are not commutative. The conflict relation is transitive, and therefore determines a set of equivalence classes, which can be named as conflict classes. In our workflow environment they are used to determine the granularity of locking. In order to define workflow transaction serialisability; WT-serialisability, let us consider two workflow flexible transactions WT_α and WT_β, and conflict classes, i and j. A global schedule is WT-serialisable if for any subtransactions ST_i^α and $ST_j^\alpha \in WT_\alpha$, and ST_i^β and $ST_j^\beta \in WT_\beta$ such that conflict $(ST_i^\alpha, ST_i^\beta)$ and conflict $(ST_j^\alpha, ST_j^\beta)$, $ST_i^\alpha \prec ST_i^\beta \Rightarrow ST_j^\alpha \prec ST_j^\beta$, at all sites they conflict. In our workflow environment the $\prec$ relationship is defined in terms of local serialisability. WT-serialisability establishes a partial order among all workflow flexible transactions. The submission order at each system, can be used to determine the execution and, consequently, the serialisation order at each site. Therefore, the concurrency control mechanism of the local system will assure that the transactions that are submitted to the local system, will be executed correctly with respect to the local concurrency control. As a result, the lock held by a subtransation can be released as soon as the subtransaction completes its submission phase. Therefore, we will have several transactions that are executing concurrently at each local site.

5. A Formal Approach to Support Workflow Security

An MLS distributed workflow management system should support functionality equivalent to a single-level workflow management system from the perspective of MLS distributed workflow users who design, implement and utilise multilevel secure distributed workflows.

A number of models for secure workflow have been proposed. These models differ in many respects. Despite a heavy interest in building a model of secure workflow management systems, there is no clear understanding regarding what a multilevel secure data model exactly is.

5.1 A Logic-Based Semantics for Multilevel Secure Workflow

In a multilevel secure workflow management system users cleared to different security levels access and share a database consisting of data items at different sensitivity levels.

As a part of our research work, we introduce a belief-based semantics for multilevel secure workflow that supports the notion of a declarative belief and belief reasoning in multilevel security scheme (MLS) in a meaningful way. We strive to develop a practical logical char-

acterisation of MLS workflow for the first time using the inherently difficult concept of nonmonotonic reasoning.

Recent research shows that users in the MLS workflow model have an ambiguous view and confusing belief of data [14].

Multilevel security implements the policy of mandatory protection defined in Ref. 15) and interpreted for computerised systems by Bell and LaPadula [16]. In this research paper we assume the representation and execution of MLS rules obey the Bell-LaPadula "no read up, no write down" principles. Many multilevel data models have been proposed in the literature, just to mention a few: SeaView [17),18)]; also models proposed by Sandhu-Jajodia [20),21)]; and by Smith-Winslett [22)] and many others. Some of these models has its strong points (e.g., the belief-based semantics of the Smith-Winslett model, etc.). However, we argue that most of these proposals are not completely satisfactory, in particular, if the workflow database may be polyinstantiated.

5.2 Multilevel Workflow Database

The majority of proposals for multilevel workflow secure relational (MLS) databases have utilised various syntactic integrity properties to control problems that arise under very strict security, such as polyinstantiation and proliferation of tuples resulting from updates, with only some partial success. We propose modal logic as a natural vehicle for reasoning about security. Because much security is dependent on the concept of what a subject knows, logic allows us to reason about knowledge, one of the fundamental concept of computer security.

We are interested in our research in workflow databases which enforce the multilevel security policy. Lets designate by *Level* a finite set of security levels. The set *Level* is assumed to be a lattice associated with a partial order relation denoted by $<$. This directly implies that, the *least upper bound* and *greatest lower bound* are determined. To describe that, we shall employ two functions *lub* and *glb*. Assuming that l_1 and l_2 are two security levels, then $lub\,(l_1,\, l_2)$ and $glb\,(l_1,\, l_2)$ are respectively the upper bound and greatest lower bound of l_1 and l_2. There are also two distinctive levels, the one which is lower than all other levels, designated by $\perp$ and the other level which is higher than all other levels, designated by $\top$. We view the global multilevel database as a set of partitions, where each partition accomodates a single-level database associated with one particular security level. We can formally represent this as follows. A multilevel database DB is represented by a set of databases $\{DB_i,\, i \in Level\}$. Every DB_i is a partition containing a finite set of propositional formulae whose classifications are equal to i and which are satisfiable but not necessarily complete. We assume that the integrity constraints are classified at level $\perp$ because there is a single set of integrity constraints which is common to every single-level database $DB_i,\, i \in Level$. We wish to remove this restriction, therefore we have to consider that we partition the global set of integrity constraints into subsets I_i associated with each single-level database DB_i. For example, let us assume that the following integrity constraint i_1 is stored at the unclassified level:

- $\forall_x, \forall_y, Emp(x) \wedge Earn(x, y) \rightarrow y \leq 80,000$

i.e., an employee must not earn more than \$80,000.

However, let us assume that there are employees who can earn up to \$99,000 but this data must be kept secret. Inductively, we can proclaim the following integrity constraint i_2 at the secret level:

- $\forall_x, \forall_y, Emp(x) \wedge Earn(x, y) \rightarrow y \leq 99,000$

However, two different sets of integrity constraints I_i and I_j may be conflicting, i.e. $I_i^* \cap I_j^* = \emptyset$, therefore we might suggest using so called [25)] the trusted approach. We need to observe that data stored in each single-level workflow database generally corresponds to a partial view of the universe by users at the corresponding security level. This is induced from our assumption that each single-level workflow database DB_l only contains data classified at level l. Therefore, in the trusted approach, the view at a given level l is obtained by merging the single-level workflow database at level l with all the lower single-level workflow databases. For example, if a workflow database at level l_{k-1} is consistent with a workflow database at level l_k, then $DB_{l_{k-1}}$ can completely flow to level l_k—as in the additive approach [25)]. Lets describe, $View\_at\_level\_l$ as the view of the multilevel workflow database for users at level l. Therefore, we can use the trusted approach to derive the set of integrity constraints

$Integrity\_at\_level\_l_k$ which apply to the security level l_k:

- $(Integrity\_at\_level\_l_1)^* = I^*_{l_1}$
- $(Integrity\_at\_level\_l_k)^* = I^*_{l_k} \rhd (Integrity\_at\_level\_l_{k-1})^*$

To be realistic, we shall assume that the global workflow multilevel database may be polyinstantiated. We define this as follows: a workflow multilevel database DB is polyinstantiated if and only if there are two security levels i and j such that $DB^*_i \cap DB^*_j = \emptyset$.

Formally, a multilevel relation consists of two parts: scheme and instances, defined below.

Definition 3. *Relation Scheme* *Let $A_1, \ldots, A_n$ be data attribute names over domain D_i, each C_i is a* classification attribute *for A_i and TC is the tuple-class attribute. The domain of C_i is specified by a range $[L_i, H_i]$ which defined a sub-lattice of access classes ranging from L_i to H_i. Let the domain of TC be the range $[lub^{\star}\{L_i : i = 1, \ldots, n\}, lub\{H_i : i = 1, \ldots, n\}]$.*

Definition 4. *Relation Instances* *Let $R(A_1, C_1, A_2, C_2, \ldots, A_n, C_n, TC)$ be a multilevel relation scheme. This collection of state-dependent relation instances, one for each access class c in the given lattice is designated by R_c. Then each instance of a multilevel relation is a set of distinct and ordered tuples of the form $(a_1, c_1, a_2, c_2, \ldots, a_n, c_n, t_c)$ where each $a_i \in D_i$ or $a_i = null$, and $t_c = lub\{c_i : i = 1, \ldots, n\}$. If $a_i \neq \perp$ (null value) then $c_i \in [L_i, H_i]$. We also require that c_i be defined even if a_i is null - a classification attribute cannot be null or more formally, $c_i \neq \perp$ for $\forall\, a_i$.*

Similarly to classical relations, multilevel workflow relations are required to satisfy several integrity properties. Since multilevel workflow relations have different instances at different access classes, the definition of keys becomes unclear because a relation instance is now a collection of sets of tuples rather than a single set of tuples.

5.3 The Necessity for Semantics in Secure Workflow Databases

The problem of polyinstantiation arises because of different views of a single entity in the real world at different security levels by two subjects. Also the above problem generally occurs through the avoidance of a covert channel. If for example a user inserts a relation instance—tuple with key K_1, a user from a lower security level cannot be prevented from inserting a different tuple with key K_1 later on, as rejecting the later insertion would open a covert channel. As a direct result of this operation, MLS workflow relations can contain multiple tuples with the same key value—polyinstantiated tuples. This problem has been indicated in some previous models by means of syntactic integrity properties, which control the extent and nature of polyinstantiation–e.g., Jajodia and Sandhu [20),21)] and Jukic and Vrbsky [14)].

Our contention is that both these models of asserting user beliefs about security are incomplete and somewhat stringent.

The Jukic-Vrbsky model is too restrictive and has only fixed interpretations. On the other hnd, the Jajodia-Sandhu model is too basic where users are left to discover the truth. Users in these frameworks really do not have any reasoning capabilities as the interpretations are already given.

The paucity of attempts aimed at developing a logical characterisation for MLS models shows that MLS workflow deductive databases are really at their embryonic state. While there were proposals such as Ref. 19) that addressed the general issue of authorisation in a deductive framework, only Cuppens addressed the issue of querying MLS deductive databases [23)]. We believe a middle ground is warranted where the user is given the choice to reason and theorise about the beliefs of others and decide how he wants to believe information which is visible to him. To support that approach, we assert that users should be given linguistic tools to view data as well as to construct meaning of the visible data. In such an environment, the user may take a firm view of the data and insist that whatever is created at his security level only are correct and believable data. Thus lower level data are of no value.

5.4 Inference Control Theorems of MLS Workflow Database

We argue that any proposed model of MLS worckflow database, under either discretionary or mandatory security, should incorporate at least the following elements:

- A formally defined model of the MLS including all the security propeties that databases under this model will possess.
- Classification of any piece of information at any given classification level, should be en-

☆ Least upper bound.

forced by powerful inference control rules.
- A formal definition—semantics for databases under the proposed model, which can represent the beliefs about the state of the world held by the users at a chosen security level.

The axiomatics of the language $\mathcal{L}$, which we consider is based on classical axiom schemas of first order logic with equality, augmented with appropriate axioms of our theory related to the multilevel workflow object-relational database. The subset of our language $\mathcal{L}$ is universally consistent with any language based on first order logic with equality [23]. What follows is a set of some axioms, which are relevant to a set of integrity constraints to be enforced by the multilevel workflow object-relational database:

- If a is an attribute of the object o then o is an object.
  $\forall_a \forall_o, OA(o,a) \rightarrow Object(o)$ $(\boldsymbol{\mathcal{A}})$
- If m is a method of the class c then c is a class.
  $\forall_m \forall_c, Method(c,m) \rightarrow Class(c)$ $(\boldsymbol{\mathcal{B}})$
- If a is an attribute of the class c then c is a class.
  $\forall_a \forall_c, CA(c,a) \rightarrow Class(c)$ $(\boldsymbol{\mathcal{C}})$
- Any object attribute has a value.
  $\forall_a \forall_o, OA(o,a) \leftrightarrow \exists_v, Val(o,a,v)$ $(\boldsymbol{\mathcal{D}})$
- The value of an object attribute is unique.
  $\forall_a \forall_o \forall_v \forall_{v'}, Val(o,a,v) \wedge Val(o,a,v') \rightarrow (v = v')$ $(\boldsymbol{\mathcal{E}})$
- Any object is instance of at least one class.
  $\forall_o, Object(o) \rightarrow \exists c, Instance(o,c)$ $(\boldsymbol{\mathcal{F}})$
- If o is an instance of c then o is an object and c is a class.
  $\forall_o \forall_c, Instance(o,c) \rightarrow Object(o) \wedge Class(c)$ $(\boldsymbol{\mathcal{G}})$

In this section we also present the general constraints that should be enforced when classifying the workflow database content. Those constraints must be satisfied when classifying $Class - c$, containing objects o and attributes a at level l and $Class(c)$ at level $\hat{l}$. The language that we propose to represent the multilevel workflow database is an extension of the above defined language combined with the acclaimed Datalog language which is also augmented with the predicates of the Logic Data Language—LDL, resulting in a powerful combination of the expressive power of a high-level, logic-based language (such as Prolog) with the non-navigational style of relational query language, where the system is expected to devise an efficient execution strategy for it. For each predicate P of an arbitrary n used to represent the non-protected workflow database content, there is a predicate $\hat{P}$ of arity $(n+1)$ used to represent the MLS workflow database.

It is generally acknowledged that when classifying any piece of information at a given level, the following inference control rule must be active:

Definition 5. *Rule - 1 Let $x_1, \ldots, x_n$ be tuples of variables consecutively compatible with the arity of predicates $P_1, \ldots, P_n$. Let y be another tuple of variables compatible with the arity of Q. For simplicity we assume that each variable in tuple y appears in at least one of the tuples $x_1, \ldots, x_n$. therefore if:*

$$\forall x_1, \ldots, \forall x_n, P_1(x_1) \wedge, \ldots, \wedge P_n(x_n) \rightarrow Q(y)$$

is an axiom of the non-secure object oriented database, then by following the similar approach as in [23], we can derive the following theorem in relation to the multilevel workflow object oriented database:

$$\forall x_1 \ldots \forall x_n \forall l_1 \ldots \forall l_n \forall l, \hat{P}_1(x_1, l_1) \wedge \ldots \wedge \hat{P}_n(x_n, l_n) \wedge \hat{Q}(y, l) \rightarrow l \leq lub(l_1, l_2, \ldots, l_n)$$

If the above rule 1 is not complied with, then a subject cleared at level $\text{lub}(l_1, \ldots, l_n)$ can access every $P_i(x_i)$ and use the above defined axiom to derive $Q(y)$. On the other hand if the classification of $Q(y)$ is not lower or equal to $\text{lub}(l_1, \ldots, l_n)$, then an inference passage enabling prohibited information to be disclosed is opend. By combining the above derived rule 1 with some more axiomatic of our language, we can derive more useful theorems[☆].

For example by combining rule - 1 with axiom $(\boldsymbol{\mathcal{D}})$, we can derive the following theorem:

- $\forall_a \ \forall_o \ \forall_v \ \forall_l \ \forall'_l$,Val'(o,a,v,l) $\wedge \ OA'(o,a,l') \rightarrow (l')$ $(\boldsymbol{\mathcal{H}})$

Which can be described as follows: the sensitivity of "v is a value of the attribute a in object o" dominates the sensitivity of "a is an attribute of object o".

This model includes the possibility to hide some parts of the multilevel workflow database schema and to deal with rules in the database. Therefore, it may also be used as a formal semantics for multilevel workflow deductive databases.

[☆] Detailed demonstration on how similar theorems can be established can be found in Ref. 25)

When classifying any data of information at a given sensitivity level [24], the following control rule must be operational if one wants to protect the existence of secure information:

Definition 6. *Rule - 2 Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_p$ be tuples of variables consecutively compatible with the arity of predicates $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_p$ and let y be another tuple of variables. For simplicity we assume that each variable in tuple y appears in at least one of the tuples $y_1, \ldots, y_p$ and each variable in tuples $y_1, \ldots y_p$ appears in at least one of the tuples $x_1, \ldots x_n, y$. If:*

- $\forall\, x_1 \ldots \forall\, x_n,\ P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow \exists\, y,$
  $Q(y_1) \wedge \ldots \wedge Q(y_p)$ **$(\mathcal{L})$**

is an axiom of the non–protected workflow object–relational database, then, the following theorem can be derived related to the workflow multilevel object–relational database:

- $\forall\, x_1 \ldots \forall\, x_n\ \forall\, l_1 \ldots \forall\, l_n,\ P_1'\ (x_1, l_1)$
  $\wedge \ldots \wedge P_n'(x_n, l_n) \rightarrow \exists\, y\ \exists\, l_1' \ldots \exists\, l_p',$
  $Q'(y_1, l_1')$
  $\wedge \ldots \wedge Q'(y_p, l_p') \wedge lub(l_1', \ldots, l_p')$
  $lub(l_1, \ldots, l_n)$ **$(\mathcal{M})$**

In case, when Rule - 2 is not satisfied, then a subject cleared at level $lub(l_1, \ldots, l_n)$ can access every $P_i(x_i)$ and use the axiom **$(\mathcal{L})$** to derive the existence of the secure data (facts) $Q(y_1), \ldots, Q(y_p)$ some of them being classified higher than $lub(l_1, \ldots, l_n)$. As the result, effectively a signaling channel is created, which enables the existence of prohibited information within the workflow repository to be disclosed.

6. The System Model

We will very concisely review the fundmental concepts of multilevel security and multiversion serialisability theory. We refer the reader to Refs. 1) and 2) for more details related to the security model and to Ref. 28) for additional details relevant to multiversion serialisability. A different approach to the problem of devising secure concurrency control techniques is to adopt less restrictive correctness criteria than the ordinarily acknowledged one-copy serialisability [29]. It has been recognised for some time, that traditional models of concurrency and transactions are too restrictive for many different applications. In particular some difficulties arise for integrity constraints involving data at different security levels [30]. As a direct result of this situation, different correctness criteria weaker than one-copy serialisability can be used for many transactions. Modern, concurrency control algorithm in Trusted Oracle uses a combination of two-phase lockig and timestamping for secure concurrency control that generate histories which are not one-copy serialisable. To concur with Ref. 31) we consider any multilevel secure system as consisting of a set D of data items, a set T of transactions (streams) which manipulate these data items and lattice S of security levels, called the security lattice, whose elements are ordered by the dominance relation $\preceq$. If two security levels s_i and s_j are ordered in the lattice such that $s_i \preceq s_j$, then s_j dominates s_i. A security level s_i is said to be strictly dominated by a security level s_j, designated as $s_i \prec s_j$, if $s_i \preceq s_j$ and $i \neq j$. Each data item from the set D and every transaction from the set T is assigned a fixed security level. The following two conditions are necessary for a system to be secure:

- Transaction T_i is not allowed to read data element $x \preceq L(T_i)$.
- Transaction T_i is not allowed to write a data element x unless $L(x) = L(T_i)$.

The above two restrictions must be augmented with the guard against illegal information flows through signaling and covert channels for the system to be secure.

As we already stated before generally our approacch to security policy is based on the Bell-LaPadula model [27].

Now we present our relaxed form of correctness criterion, called level-wise serialisability. We refer the reader to[1),31)] for additional details related to the security model and for details relevant to multiversion serialisability. It can be used under those circumstances where integrity constraints are independent among data at various access classes.

Definition 7. *Level-wise serializability We say a multiversion history H over T is level-wise serialisable if:*

- *For each level $s \in S$, the subhistory H_s of H is one-copy serializable.*
- *Let T_i be a transaction with $L(T_i) = s$, and let $s' \prec s$. Then the subhistory $H_{T'}$ where $T' = \{T_i \in T : L(T_i) = s'\} \cup R_{s'}(T_i)$ is one-copy serialisable.*

The prevalent idea of level-wise serialisability is relevant to the notions of fragmentwise serialisability introduced by Garcia-Molina and Kogan [32]. The above first requireents of level-wise serialisability states that if we consider only

those transactions which are executing at a single level, then the concurrent execution involving these transactions is one-copy serialisable. The second requirement promise that whenever a high transaction reads data at a lower level, it only sees a consistent copy of the low data.

The concurrency control algorithm that has been implemented in Trusted Oracle [33], guarantees level-wise serialisability. It is secure and does not require a trusted scheduler. It is composed of the following steps:

(1) There is separate scheduler at each security level s.
(2) There is a shared, monotonically increasing hardware clock at system low that is guaranteed to return a new value every time it is read. This clock is used to assign a unique timestamp to each transaction T_i, denoted by $ts(T_i)$. Each T_i is also assigned a commit timestamp, $cts(T_i)$, when T_i reaches its commit point.
(3) If T_i is a read-only transaction at level s and wishes to read an item x at level $s^{'}$ such that $s \succeq s^{'}$, then step 7 is executed.
(4) If T_i is a read-write transaction at level s, the following steps are performed.
(5) At each security level s, strict two-phase locking is used for concurrency control. However, when T_i writes an item x, it creates a new version x_i. A write timestamp $wt(x_i)$ is assigned to x_i which equals the commit timestamp of T_i.
(6) When a transaction T_i at level s wishes to read an item x at level $s^{'}$ such that $s \succ s^{'}$, step 7 is executed.
(7) The timestamp $ts(T_i)$ is used to select the appropriate version of x. The version selected is x_k with the largest $wt(x_k)$ such that $wt(x_k) \prec ts(T_i)$.

Thus, the above stated Trusted Oracle DBMS fully supports transactional workflow securiy requirements.

7. Implementation of the Secure Distributed Workflows

The architecture of our multilevel secure workflow transaction processing system can be divided into a trusted, an untrusted, and a receptive module as shown in **Fig. 4**.

The trusted component consists of two functioning modules: Trusted Lock Manager and Trusted File Manager. The untrusted component is a collection of Transaction Managers (TMs), one for each security level. The receptive component is the Workflow Database - its File Store component, which may be physically partitioned according to the security levels.

Figure 4 shows the interfaces between the different components of our implementation. Except for the native interface between the RMs and the Workflow Database, all interfaces are defined as an API in the X/Open standard:

- TX interface ($WD \Leftrightarrow TM$): With this interface an application can demarcate the begin and the end of a transaction to the TM [34].
- XA interface ($TM \Leftrightarrow RM$): The TM coordinates the begin, the suspending and the commit protocol of a transaction to the registred RM in a domains - computer nodes [35].
- XA+ interface ($TM \Leftrightarrow CRM$): With these functions the TMs can communicate via the CRMs in a distributed transaction between different domains [36].
- TxRPC IDL ($AP \Leftrightarrow CRM$): The TxRPC [37] is based on the DCE RPC [38], which can use some other DCE services like the name service for the localisation of servers. The DCE interface definition language (IDL) is able o support the TxRPC. The CRMs communicate with each other via the OSI TP protocol.

The major advantage of this implementation is the standardisation of the interfaces. Due to the modular concept, all components and applications can be easily extended and replaced. So cooperation between components of different vendors is possible. In our sample environment the transaction monitors TUXEDO (Novell) and Encina (Transarc) can be involved in a single transaction and can even talk to each other via a CRM of a different vendor. The model supports changes in the configuration at runtime because the RM and CRM can also register dynamically at the TM. Multiple transactions initiated by different workflows applications are distinguished by the thread of control (ToC). It is a structure defined in the X/Open model and must be distinguished by the thread-ID or process-ID in the sense of POSIX. The TM manages the sequence of transactions by suspending them and resuming them at a later time.

Note that in order to accomodate relevant commercially available DBMS to support our approach, the assumption about Lock Manager

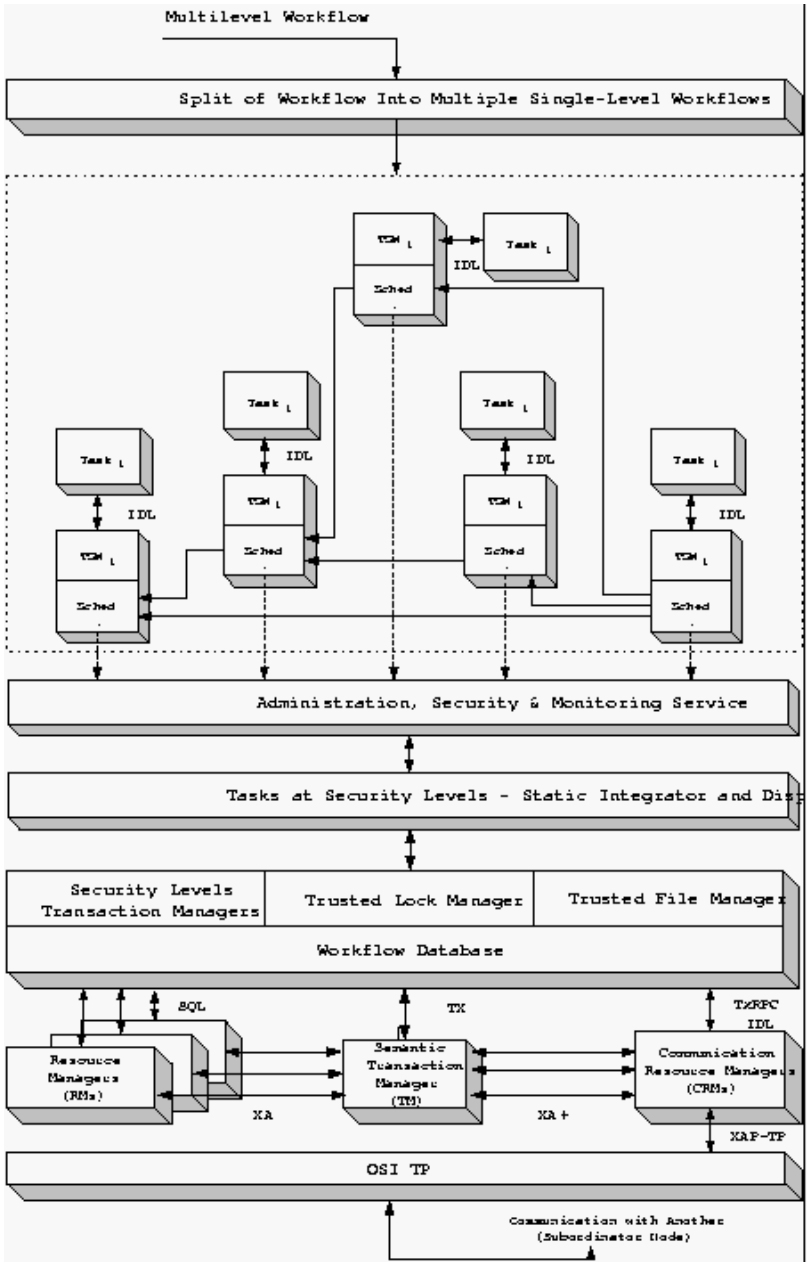

Fig. 4 The MLS distributed workflow architecture.

being trusted can be relaxed by providing one Lock Manager for each security level. If that particular option is taken, a Trojan Horse inside some untrusted Lock Manager can compromise the correct execution of Concurrent workflow transactions, but cannot violate security. Additionally, as the whole body of a standard Lock Manager, written with all the requisite defensive programming, exception habdlers, optimizations, deadlock detectors, etc. comes to about a thousand a thousand lines of actual code, it is easily verifiable. Thus our assumption of a Trusted Lock Manager, which jeopardises neither security nor integrity, is justified.

Composing an MLS workflow from multiple single-level workflows is the only practical way to construct a high-assurance MLS WFMS today. In this approach, the multilevel security of our MLS workflow does not depend on single-level WFMS but rather on the underlying MLS

distributed architecture. A transaction T_k at the security level s_i can read information stored at level s_i only if $s_j \preceq s_i$, and can write information only at level s_i [31]. The corresponding TM_i at the level s_i controls the concurret execution and recovery of those—and only those transactions that are at level s_i, therefore TM_i can be untrusted component of the architecture. In a multilevel secure workflow, tasks may belong to different security levels. Thus ensuring all the task dependencies, especially those from a task at a higher security level to that at a lower security level, may compromise security. This implies that it is important to understand that in a multilevel environment it is not possible to force the abort of a lower level task upon the abort of a higher level task.

The workflow Static Integrator and Dispatcher module examines the structure of workflow dependencies which also includes security levels of tasks and alters the workflows accordingly. Integration is done statically, therefore the Integrator and Dispatcher need not be trusted. As it is supported by our architecture, while the WFMS layer enforces all the dependencies existing among the various tasks in a workflow by submitting the tasks to the appropriate DBMS in a coordinated manner, the corresponding DBMS simply executes their respective tasks and sends the responses back to the WFMS. Similar implementations can be found in the context of extended transaction models, for example the reflective transaction framework in Ref. 39), etc.

Recall that, in a parallel workflow control architecture, the state of an individual workflow is on a single node while the global state is distributed across nodes. Hence, the extended solutions require an additional message passing the state and event information across nodes. Distributed workflow control requires an additional message since the state of an individual workflow is distributed across nodes. Our environment consists of a heterogeneous collection of nodes as shown in Fig. 4. Any of these nodes can act as agents or engines in central, parallel or distributed control architectures. Hence to achieve total portability across heterogeneous architectures, the workflow compiler and the architecture's run-time environment have been partially implemented using the Java programming language and it's development tools. One of the essential functions of the MLS workflow compiler is to divide an MLS workflow into multiple single-level workflows. These multiple single-level workflows will be executed on the underlying MLS distributed architecture that is dipicted on Fig. 4. The workflow run-time environment has to manage several workflow instances concurrently. Also, workflow instances may have two or more steps from concurrent branches executing simultaneously. These requirements have been efficiently achieved in the present run-time environment using the multithreading facility available in Java. The prototype system is client-server oriented. The backend server features a set of transactional activity management primitives, and is built on Transarc's transaction processing TP monitor Encina. The frontend features several web-friendly graphical user interfaces for activity control and administration at both specification and instance levels, and is built with a mixed use of Java applets, JavaScript, and HTML for portability and reusability. Our WFMS utilises a simple model DBMS from which workflows may be selected for execution. Workflows or components of workflows may be added to the model repository by specifying them in a workflow language. WFSL/TSL may be used to specify workflows [40]. The WorkFlow Specification Language—WFSL is a declarative rule-based language to describe the conceptual workflow specification, while the Task Specification Language TSL [40] is a language to specify simple tasks that run in our workflow systems environment. Once a workflow is selected from the repository, several steps are carried out that instantiate a workflow instance that is able to run within our execution environment.

8. Evaluating the Quantitative Effects of the Workflow System

We performed some experiments with the prototype in order to demonstrate the difference in operational efficiency between two methods. Experiments span between two geographically separated branches of the business enterprise. The experiment was carried out by the members of those two branches who usually are engaged by those operations as a normal part of their work duties. We calculated the average of several independent experiments in order to establish a more objective experimental environment. During the first experiment we measured the time needed to complete the whole operational cycle according to the busi-

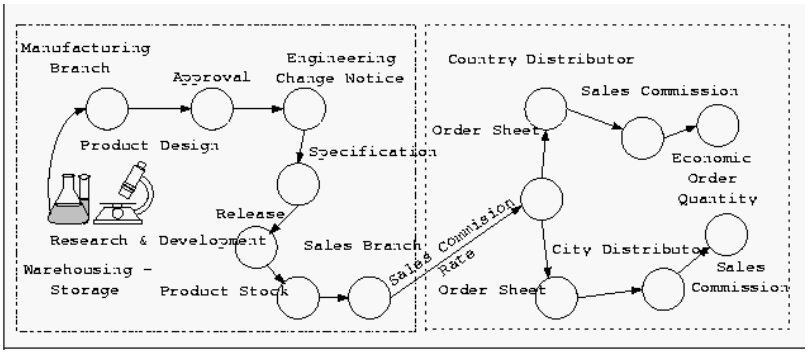

Fig. 5 The evaluated business process.

ness process being described by **Fig. 5**.

This experiment was performed by following the usual manual disconnected operation between the two physically separate business establishments. In this case each branch manages the internal workflow individually. Interoperation between organisational units is only supported by an E-mail system.

During the second set of experiments the business process depicted on Fig. 5 was carried out by the workflow engines interoperating with each other. Document exchange across the organisations is performed automatically by a messaging function supported by the protocol used by the workflow engines. All accompanying transactional activities were also supported in that way by the supporting workflow environment. The average values related to the execution time of the business process depicted on Fig. 5, namely: Manufacturing, Warehousing & Storage, Sales—City Distributor and Sales—Country Distributor. Note, that calculated time reflex only values related to the movement of business data within the system under investigation. In particular, it does not include the time consumed by various supporting activities being a part of miscelanous approval activities.

The results of those experiments show us (see **Fig. 6**) that savings in time amounts to about 59% in case of using interworkflow environment to support the business processes. From these experiments, we can draw the conclusion, that by deploying workflow operational environment, substantial amounts of time can be saved while supporting the busines operations.

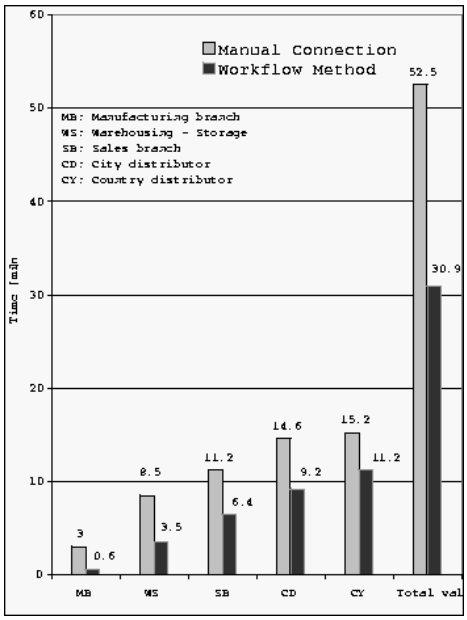

Fig. 6 Execution time of manual and workflow methods.

9. Conclusions

Well-structured process management has become an ingredient to modern information management as essential as data management. Consequently, workflow management systems have entered the arena of business computing as the cornerstone for business process or workflow management. The impetus for our current research is the need to provide an adequate framework for belief reasoning about security in MLS distributed workflow management systems. The notions of correctness for transaction processing that are usually proposed for multiuser databases are not necessarily suitable when these databases are parts of a multilevel secure workflow systems. We believe, that the best approach will depend upon the characteristics of the multilevel secure workflow and the applications. It is incumbent upon those who develop multilevel secure workflow systems to

ensure that the user's needs and expectations are met to avoid misunderstandings about the system's functionality.

The insight developed in the current research serves as the basis for a complete logical synthesis of SecureLog, the language which we are currently developing as an orthogonal extension of the work contained in this paper in the direction of F-logic [26].

We chose to develop a formal framework for a secure distributed workflow architecture since interworkflow is anticipated as a major supporting mechanism for Business-to-Business Electronic Commerce. We strive to develop a practical logical characterisation of MLS distributed workflow for the first time using the inherently difficult concept of non-monotonic reasoning. We also derived a general theorem which must be active when classifying every item of information.

Acknowledgments We wish to acknowledge the anonymous referees for their helpful comments, which led us to an improved presentation of this paper. A preliminary version of this paper appeared under the title "A Strategy for MLS Workflow", in *Proceedings of The Sixth ACISP International Conference on Information Security and Privacy*, Sydney, July 2001.

References

1) Wietrzyk, V., Takizawa, M. and Varadharajan, V.: A Strategy for MLS Workflow, *Information Security and Privacy in Proc. 6th International Conference, ACISP 2001*, Sydney (July 2001).
2) Wietrzyk, V., Takizawa, M., Orgun, M.A. and Varadharajan, V.: A Secure Environment for Workflows in Distributed Systems, *Parallel and Distributed Systems in Proc. 8th International Conference, ICPADS 2001*, KyongJu City (June 2001).
3) Wietrzyk, V. and Orgun, M.A.: A Foundation for High Performance Object Database Systems, *Databases for the Millennium 2000 in Proc. 9th International Conference on Management of Data*, Hyderabad (Dec. 1998).
4) Elmagarmid, A.: *Transaction Models for Advanced Database Applications*, Morgan-Kaufmann (Feb. 1992).
5) Grefen, G., Pernici, B. and Sanchez, G.: *Database Support for Workflow Management — The WIDE Project*, Kluwer Academic Publishers (Aug. 1999).
6) Rusinkiewicz, M. and Sheth, A.: On transactional Workflows, *Bulletin of the Technical Committee on Data Engineering* (June 1993).
7) The Workflow Management Coalition Interoperability Abstract Specification, *The Workflow Management Coalition* (June 1996).
8) Alonso, G. and Agrawal, D.: Advanced transaction Models in Workflow Contexts, *Proc. Int. Conf. on Data Engineering* (1996).
9) Georgakopoulos, D., Hornick, M. and Sheth, A.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, *Distributed and Parallel Databases*, Vol.3, No.2, pp.119–153 (1999).
10) Elmagarmid, A.K., Leu, Y., Litwin, W. and Rusinkiewicz, M.E.: A Multidatabase Model for Interbase, *Proc. 16th VLDB Conference* (Aug. 1990).
11) Gligor, V. and Popescu-Zeletin, R.: Transaction Management in Distributed Heterogeneous Database Management Systems, *Information Systems*, Vol.11, No.4 (1986).
12) Zhang, A., Nodine, M., Bhargava, B. and Bukhres, O.: Scheduling with Compensation in Multidatabase Systems, *CSD-TR-93-063*, Vol.11, No.4 (1993).
13) Ansari, M., Rusinkiewicz, M., Ness, L. and Sheth, A.: Executing Multidatabase Systems, *TM-TSV-019450* (1991).
14) Jukic, N.A. and Vrbsky, S.V.: Asserting beliefs in mls relational models, *Sigmod Record*, Ithaca, NY, pp.30–35, ACM Press (1997).
15) Department of Defense, National Computer Security Center: Department of Defense Trusted Computer System Evaluation Criteria, *DOD 5200.28-STD* (1985).
16) Bell, D.E. and LaPadula, L.J.: Secure Computer Systems: Mathematical Foundations and Model, Technical Report, MITRE Corporation (1974).
17) Denning, D., Lunt, T., Heckman, R. and Shockley, W.: A Multilevel relational data Model, *Proc. IEEE Symposium on research in Security and Privacy*, Oakland, April, pp.220–234, IEEE, New York (1987).
18) Lunt, T.: Multilevel Security for Object-Oriented Databases. Spooner, D.L. and Landwehr, C. (Eds.), *Database Security, III*, pp.199–209, Amsterdam, North-Holland (1990).
19) Candan, K.S., Jajodia, S. and Subrahmanian, V.S.: Secure Mediated Databases, *Proc. ICDE*, pp.35–55 (1996).
20) Jajodia, S. and Sandhu, R.: Toward a Multilevel Secure Data Model, *Proc. ACM SIGMOD*, Denver, Colo., May, pp.50–59, ACM, New York (1991).
21) Jajodia, S. and Sandhu, R.: Polyinstantiation Integrity in Multilevel Relations, *Proc. IEEE*

Symposium on Research in Security and Privacy, Oakland, May, pp.104–115, IEEE, New York (1990).
22) Winslett, M. and Smith, K.: Entity Modeling in the MLS Relational Model, *Proc. 18th International Conference on VLDB*, pp.199–210, VLDB Endowment (1992).
23) Cuppens, F.: Querying a Multilevel Database: a logical Analysis, *Proc. 22nd VLDB Conference*, VLDB Endowment (1996).
24) Boulahia-Cuppens, N., Cuppens, F., Gabillon, A. and Yazdanian, K.: Decomposition of Multilevel Objects in an Object-Oriented Database, *Proc. European Symposium on research in computer security*, Brighton, UK, Springer Verlag (1994).
25) Gabillon, A.: *Sécurité Multi-Niveaux dans les Bases de Données à Objects*, ENSAE (1995).
26) Kifer, M., Lausen, G. and Wu, J.: Logical Foundations for Object-Oriented and Frame-Based Languages, *Journal of the Association of Computing Machinery*, Vol.42, No.3, pp.741–843 (1995).
27) Bell, D.E. and LaPadula, L.J.: Concurrency Control and Recovery in Database Systems, Addison-Wesley, Reading, Mass. (1987).
28) Bernstein, P.A., Hadzilacos, V. and Goodman, N.: Decomposition of Multilevel Objects in an Object-Oriented Database, *Proc. European Symposium on research in computer security*, Brighton, UK, Springer Verlag (1994).
29) Kogan, B. and Jajodia, S.: Secure Concurrency Control, *Proc. Third RADC Workshop Multilevel Database Security*, Castille, N.Y. (June 1990).
30) Meadows, C. and Jajodia, S.: Integrity versus Security in Multilevel Secure Databases, *Database Security, Status and Prospects*, Landwehr, C.E. (Ed.), Amsterdam, North Holland (1988).
31) Jajobia, S., Elisa, B. and Atluri, V.: 1SR-Consistency: A New Notion of Correctness for Multilevel Secure, Multiversion Database Management Systems, *Proc. 19th Latin Am. Informatics Conf.*, Buenos Aires, Argentina (Aug. 1999).
32) Garcia-Molina, H. and Kogan, B.: Achieving High Availability in Distributed Databases, *IEEE Trans. Softw. Eng.*, Vol.14, No.7 (1988).
33) Maimone, W.T. and Greenberg, I.B.: Single-Level Multiversion Schedulers for Multilevel Secure Database Systems, *Proc. Sixth Ann. Computer Security Applications Conf.*, Tucson, Ariz. (Dec. 1990).
34) X/Open Ltd.: CAE Specification 1995: Distributed Transaction Processing, *The TX* (*Transaction Demarcation*) *Specification*, X/Open Company Ltd. (1995).
35) X/Open CAE Specification: Distributed Transaction Processing, *The XA Specification, X/Open* (1991).
36) X/Open Snapshot: Distributed Transaction Processing, *The XA+ Specification, X/Open, 1999*, *Version 2*, X/Open Company Ltd. (1999).
37) X/Open CAE Specification: Distributed Transaction Processing, *The TxRPC Specification, X/Open* (2000).
38) Schill, A.: DCE — Das OSF Distributed Computing Environment, *The DCE* (*Transaction Demarcation*) *Specification*, Verlag (2001), ISBN 3-540-55335-5.
39) Barga, R. and Pu, C.: A Practical and Modular Method to Implement Extended transaction Models, *Proc. VLDB* (1995).
40) Krishnakumar, N. and Sheth, A.: Managing heterogeneous multi-system tasks to support enterprise-wide operations, *Distributed and Parallel Databases*, Vol.3, No.2 (1995).

(Received February 28, 2001)
(Accepted October 16, 2001)

Vlad Ingar Wietrzyk obtained his M.Sc. degree from Prague University. EU and his Dip. in Computer Science from UTS, Sydney. Since 1999 he has been at the University of Western Sydney. Since 1997 until 1998 he had been a visiting researcher of Stuttgart and Mannheim Universities. He has publications in national and international conferences and workshops. In 1999 he was a visiting researcher at the Institute of Software Engineering, Montreal University. He has served on the program committees of international conferences like ICPADS, IDEAS, CIT, COMAD, ENTER. He has deliverd industrial seminars on computing to companies like VERSANT and ALCATEL. While at the Analytical Service Corporation, Sydney 1987–1995 he designed and implemented in software, a hierarchical clustering method which was the first to support the analysis of data based on groups and data exploration. His current research interests are: Object Distributed Databases, Various aspects of Information Systems Design Methodologies (including Distributed Systems), Transaction Processing in distributed systems, Concurrency Control, Distributed and Federated Database Systems, and Distributed Workflow Technology supporting Electronic Commerce. He is a member of IEEE and AIEA.

Katsuya Tanaka was born in 1971. He received his B.E. and M.E. degree in Computers and Systems Engineering from Tokyo Denki University, Japan in 1995 and 1997, respectively. From 1997 to 1999, he worked for NTT Data Corporation. Currently, he is an assistant in the Department of Computers and Systems Engineering, Tokyo Denki University. He received the D.E. degree from Dept. of Computers and Systems Engineering, Tokyo Denki University, Japan, in 2000. His research interests include distributed systems, transaction management, recovery protocols, and computer network protocols. He is a member of IEEE CS and IPSJ.

Makoto Takizawa was born in 1950. He received his B.E. and M.E. degrees in Applied Physics from Tohoku Univ., Japan, in 1973 and 1975, respectively. He received his D.E. in Computer Science from Tohoku Univ. in 1983. From 1975 to 1986, he worked for Japan Information Processing Developing Center (JIPDEC) supported by the MITI. He is currently a Professor of the Dept. of Computers and Systems Engineering, Tokyo Denki Univ. since 1986. From 1989 to 1990, he was a visiting professor of the GMD-IPSI, Germany. He is also a regular visiting professor of Keele Univ., England since 1990. He was a program co-char of IEEE ICDCS-18, 1998 and serves on the program committees of many international conferences. He chaired SIGDPS of IPSJ from 1997 to 1999. He is IPSJ fellow. His research interests include communication protocols, group communication, distributed database systems, transaction management, and security. He is a member of IEEE, ACM, and IPSJ.