6

Using Workflow to Enhance Security in Federated Databases

Martin S Olivier
Department of Computer Science, Rand Afrikaans University
PO Box 524, Auckland Park, Johannesburg, 2006 South Africa
Email: molivier@rkw.rau.ac.za

Abstract

A workflow system automates processes that occur during the daily operation of an organisation. The description of such a process inherently includes information about *who* needs to perform each step. It also includes information about *when* this subject has to perform that step. This information can be used to enhance security in the system. In particular the *when* information can be used to dynamically adapt security according to the current state of the workflow process.

This paper considers the issues involved in using workflow to enhance security. We propose an architecture that uses the information contained in workflow scripts to dynamically modify security specifications. This information can be used to increase confidentiality, integrity and availability.

The paper is primarily interested in secure interorganisational workflow. The proposed system is based on a secure federated database shared amongst the participating organisations.

Keywords

Distributed Systems—Distributed databases; Heterogeneous Databases; Security and Protection (Keyword Codes: C2.4; H2.5; K6.5)

1 INTRODUCTION

A workflow system automates processes that occur during the daily operation of an organisation. The system automatically routes information to the next person involved in the process, who deals with that information, before it is sent on to whoever has to act on it next. So, for example, can the claims procedure in an organisation be automated such that the claimant initiates the process by submitting a claim form. This claim form is routed to the individual's superior who approves it. Finally this document is sent to a finance clerk who arranges transfer of the money to the claimant's account and records the transaction in the organisation's books.

This paper considers secure use of such a workflow system when it is used between organisations. As an example of such an interorganisational workflow system consider a

P. Horster (ed.), *Communications and Multimedia Security II*

system where an order flows through a client's system until it is fully approved; then it flows to the supplier's system who uses it as basis for execution of the order, after which it may flow back to the client's system where the client may use it to check delivery, and for financial accounting purposes. We will argue that the knowledge contained in the system about business processes can be used to enhance security of the system, while the additional security requirements posed by such a workflow system does not differ significantly from those posed by an interorganisational federated database.

In this paper we will assume that the workflow system is implemented on top of a federated database. A federated database is a distributed database characterised by relatively autonomous nodes. Since the business processes we are interested in involves more than one organisation, we assume that the nodes of the federated database each supports its own security policy, but that the database management systems on the various nodes are compatible.

The main aim of the paper is to propose an architecture for the security components of a workflow system. It is assumed that the federated database on which the architecture is based supports the underlying access control mechanisms to be used by the workflow security components. It is also assumed that the user interface of the workflow system falls outside the security component—it is, normally, not the responsibility of the security system to inform a user that some action is required.

The architecture is proposed in section 4. Section 2 provides background information, section 3 discusses the relationship between workflow and security and section 5 contains the conclusion of the paper.

2 BACKGROUND

In this paper we will use the term *workflow process* to refer to a business process that has been automated in a workflow system. The events in a workflow process are specified with a workflow script.

A workflow process can be compared to a transaction—it is also a unit of work that has to be completed in its entirety. However, a workflow process differs from a transaction in the following ways:

1. More than one subject is usually responsible for completion of a workflow process; and
2. A workflow process that cannot be completed cannot simply be rolled back—consider for example an order process that reaches a point where it cannot continue, but after the (physical) ordered goods have already been dispatched.

See Bock and Marca (1995:105–8) for background information about the use of workflow.

This paper addresses interorganisational workflow. This implies that more than one system, owned by more than one organisation, will be involved in the system. Two possibilities are available to form the basis of this system: messaging or a shared database. In a pure messaging system the concerned documents will be sent to the next party in the process who has to deal with them. However, this makes the documents unavailable for other parties involved in the system; in particular, this makes it difficult for other parties to keep track of the status of a workflow process. In addition, messaging cannot easily

accommodate cases where the next participant in the process is not uniquely identified, but can be any one of a number of parties. Consider, for example, a system where the first buyer to accept the price quoted by a seller gets the goods; here the process proceeds from the seller to all potential buyers, but only one buyer succeeds. A messaging system would have to broadcast such an offer and then add concurrency control mechanisms already available in a database. We therefore assume that a shared database is used to support the workflow system

Although a centralised database can be used to support an interorganisational workflow system, it introduces ownership, management and scalability problems. In this paper we therefore assume that a distributed database is used—in particular, a federated database. A federated database is a database that provides a relative high degree of site autonomy. See Özsu and Valduriez (1991) for a discussion of distributed databases and Özsu and Valduriez (1991:81,89) and Sheth and Larson (1990) for a discussion of federated databases in particular. See Thuraisingham (1994) for a discussion of the issues that must be considered when designing a secure federated database.

A model for a secure federated database has been described by Olivier (1995) that allows the use of different security policies at the different sites. This allows organisations to share a (distributed) database, but allows each site to implement the security policy it requires for data owned by it. The model ensures that such information will be protected according to this policy even if it is accessed from another site or relocated to another site. This model, SPO (Self-protecting Objects), will form the basis of our proposal in the current paper.

In SPO a common core of code executes at all sites of the federated database. This core is known as the TCC (*trusted common core).* The TCC is responsible for implementing the federated security policy—that is the security policy that applies to all members of the federation. As a minimum this policy specifies that all local security policies will be honoured by all members of the federation. Each site can then associate *trusted extension* (TE) methods with entities owned by that site to implement the local security policy. TE methods are initiated when the protected entity is accessed and perform the necessary access checks before access is allowed to the entity. TE methods accompany an object if the object is relocated, so that the protection is also enforced at the new site. When a TE method requires further information, it requests this information from a module at it owning site—the *trusted local extension* (TLE). See Olivier (1995) for further details about SPO, Olivier (1996a) for a discussion of support for local security policies and the possibilities that exist for such local policies and Olivier (1996b) for a consideration of integrity issues in an SPO database. In this paper TE methods will also be referred to as *security methods.*

This paper assumes the use of object-oriented databases and the terminology associated with such databases will therefore be used. See Kim (1995) for a description of object-oriented databases and Lunt (1995), Olivier and Von Solms (1994) and Rabitti *et al* (1991) for a discussion of security in such databases.

The security mechanisms described in this paper are intended to simplify the task of specifying authorisations for subjects (since such information forms an inherent part of the description of a workflow process). In addition they allow a more precise statement of such access restrictions, since the logical state of the system can influence authorisation—in other words logical time can be taken into account when such authorisations are specified.

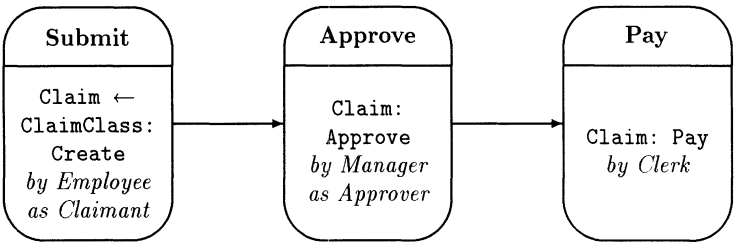


Figure 1 Claim workflow process

3 WORKFLOW AND SECURITY

A workflow system incorporates knowledge about *when* and *by whom* the components are to be used. Such knowledge can be used to increase the confidentiality, integrity and availability of information in a system. So, for example, can the knowledge about when a particular individual has to work on a given document be used to restrict access for that individual to the necessary times, thus increasing confidentiality. Similarly, integrity can be increased by, for example, structuring workflow processes such that changes to information cannot be made at inappropriate times. And, lastly, if it is known at what site an object is likely to be required next, the object can be relocated to that site timeously, not only to increase response time when the object is accessed, but also ensuring that it is available when required—even if communication lines are down at that point.

A workflow process consists of a sequence of actions. Since this paper assumes an object-oriented environment, the actions will be initiated by sending a message. Some actions can only commence once other actions have completed—the claim form, for example, can only be approved once it has been submitted. However, in other cases the sequence of some actions are immaterial: If the claim form has to be approved by two individuals, for example, it may not make any difference who approves it first. Both approvals can, however, only occur after the claim has been submitted, and transfer of funds cannot commence before both approvals have been completed. Yet other actions are alternatives for one another: the claim can either be approved or denied.

The actions described in the previous paragraph are significant for the workflow system since other actions depend on them, and they therefore have to occur for the workflow process to make progress. However, not all actions have to occur—some actions are optional: In the claim workflow process, the claimant may be allowed to access the claim at any time to determine the status of the process. Such an access by the claimant does not have any influence on the remainder of the process. We will refer to those actions that *are* necessary for the process to make progress as *events*.

A workflow process can be structured such that some unique event initiates the process and that some distinct event terminates the process. All other events in the process then have one or more preceding events and one or more succeeding events.

Consider the simple workflow process depicted in figure 1. This process consists of three events: *Submit*, *Approve* and *Pay*. The message that initiates an event is described

for each event. Note that the parameters of the messages are not given; however, in the case of `Claim` ← `ClaimClass: Create` the arrow indicates that the object returned by this particular message will be referred to as *Claim* later in the workflow script. Also note the assumption that the payment transaction is initiated by sending the `Pay` message to the `Claim` object; it is also possible that this transaction can be structured so that it is initiated by sending a message to some account object, with the `Claim` object as a parameter; in such a case the event may be specified as follows:

```
Account:TransferMoney Claim by Clerk
```

Here `Claim` is given as a parameter of `TransferMoney` since it is significant.

The potential senders of messages (and therefore the potential initiators of events) are naturally described as part of the process description. These senders can be identified individuals, such as a user John or a user Jane; however, usually it will be more natural to use roles—in figure 1 the roles *Employee*, *Manager* and *Clerk* have been used. In the model proposed in section 4 this information will be used to enhance access control and availability.

Note that the individual who acts on behalf of the designated role may be given special rights and/or obligations during the rest of the workflow process. In figure 1 any employee may submit a claim; however, it is likely that only the specific employee who submitted the claim should be authorised to track the status of the claim processing (apart from managers and other roles who have this authority in any case). To handle this, a special role can be created for any individual who acts in a workflow. In figure 1 the clause *as Claimant* is intended to indicate that the individual who initiates this process will be in the *Claimant* role for the duration of this process. If roles are properly structured it also becomes possible to specify that the approval event can be initiated by a manager of the claimant (that is, someone in the role *Manager(Claimant)*), excluding the possibility that a manager (who is also an employee) can submit and approve a claim.

In addition to the events that form the basis of a workflow process, additional actions may occur, that do not influence the process itself. In the current claim example, the claimant is allowed to track the status of the claim. Assume that this can be accomplished by sending a `GetStatus` message to the `Claim` object. Such actions may be allowed for the entire duration of the process or only between specified events. The right that the claimant has can be specified as follows:

```
Allow from Submit to Pay Claim:GetStatus by Claimant
```

Note that the duration of this permission is the entire process.

In addition to such permissions, integrity constraints can also be specified to be applied to the objects involved in the workflow process. We use the notation used by Olivier (1996b); two types of constraint can be given: *Prevent* and *Verify*. To illustrate, if the `Claim` object has `ChangeAmount` and `GetAmount` methods, the following specifications will allow the claimant to change the amount claimed until the claim is approved, but not afterwards. Using *Prevent* constraints, this can be done as follows:

```
Allow from Submit to Approve Claim:ChangeAmount by
   Claimant

Prevent from Approve to Pay Claim:ChangeAmount
```

Using *Verify* constraints, this can be specified as follows:

```
Allow from Submit to Approve Claim:ChangeAmount by
    Claimant

Verify from Approve to Pay Claim':GetAmount =
    Claim:GetAmount
```

The last constraint specifies that the given condition should hold whenever the claim object is accessed—in other words, whenever a message is sent to it; `Claim'` refers to the state of the object before such a message is processed, while `Claim` refers to it after the message has been processed. Because alternative mechanisms may exist to update the state of an object, *Verify* constraints are preferred over *Prevent* constraints—especially in multipolicy federated databases. See Olivier (1996b) for details, including an enforcement strategy.

More than one individual can act in such a workflow process role. If the simple process in figure 1 is extended by including a second *Approve* event after the first, the following specification will ensure that two distinct managers have to approve it and that both can determine the status of the request after approval. Both the *Approve1* and *Approve2* events will contain the specification:

```
Claim:Approve by Manager as Approver
```

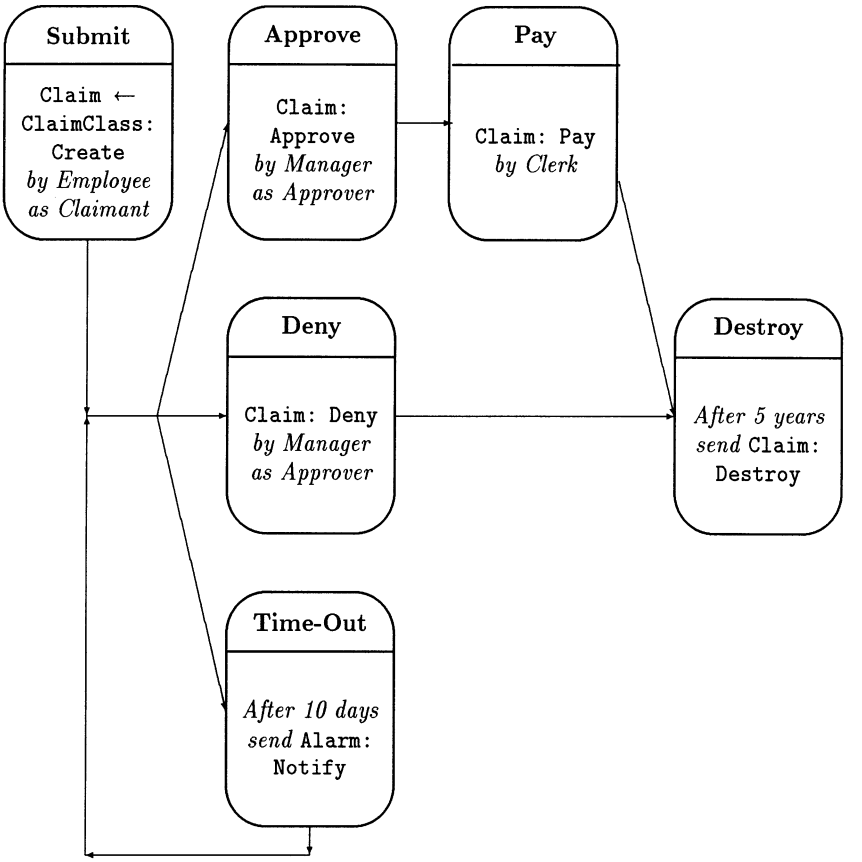
The following constraints will provide the required guarantees:

```
Prevent from Approve1 to Approve2 Claim:Approve by
    Claimant

Prevent from Approve2 to Approve2 Claim:Approve by
    Approver

Allow from Approve1 to Pay Claim:GetStatus by Approver
```

The first *Prevent* clause ensures that the claimant cannot approve the claim; this is only significant if the claimant happens to be a manager. The second *Prevent* clause ensures that the first approver cannot also be the second approver (since the subject who acted as first approver is the only subject in the *Approver* role at this point). The *Allow* clause will allow both approvers (from the point where each approved the claim) to determine the status of the claim.

Another factor that influences the integrity of workflow systems is time. If a claim is submitted, but not approved within some acceptable time the system cannot be deemed to control the workflow correctly. To handle such cases an event may also be initiated by expiry of some period; here it is also useful to enable the workflow system itself to send a message. To illustrate this concept in a simple workflow process consider figure 2 which extends the claim workflow of earlier to allow documents to be archived (and still be queried) for a period of five years after it has been paid. Note the `Notify` message sent to some `Alarm` object if a claim is not approved or denied within ten days: Normally it is not the function of the security system to inform users that it is their turn to take some action. However, if the action is overdue it does become an integrity issue.

An architecture that manages security based on workflow concepts is given in the next section.




```
Allow from Submit to Destroy Claim:GetStatus by Claimant

Allow from Approve to Destroy Claim:GetStatus by Approver

Allow from Submit to Approve Claim:ChangeAmount by
   Claimant

Verify from Destroy to Pay Claim':GetAmount =
   Claim:GetAmount
```

Figure 2 Time-based events

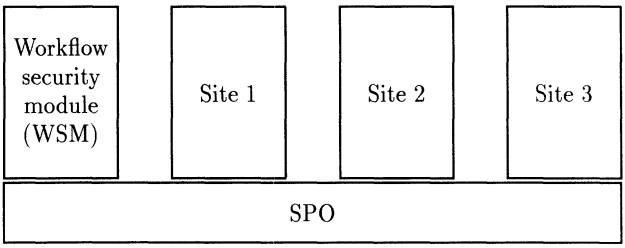


Figure 3 Federated workflow model

4 MULTISITE WORKFLOW

Multisite workflow poses problems not encountered in single-site workflow, especially if the sites are not administrated by the same authority. Without a single authority adherence to the sequence of events imposed by the workflow script is not automatically ensured. More generally, expectations that any organisation has about the behaviour of business partners in the workflow system can be violated. Some mechanism is therefore required to ensure that workflow rules are adhered to. In fact, a general mechanism is required to enforce all constraints that follow from the interoperation.

Since the workflow system can be relatively complex it cannot be given the sole responsibility to implement security. We therefore propose to implement it on top of a trusted (federated) database. The workflow scripts can then be used by the system security officer to determine the security requirements of the system: such static analysis can indicate who needs to access the system and in what manner. We will return to static analysis of workflow scripts and its use in security in the conclusion.

In this paper we are more interested in the dynamic contribution workflow scripts can make to security. Whereas a static analysis can indicate *who* needs to perform a specific operation, dynamic use of such scripts can, for example, also be used to control *when* they are allowed to perform such an operation.

Figure 3 illustrates the model we propose to dynamically make use of knowledge contained in the workflow scripts. As can be seen the model uses the SPO model described in section 2 as basis. This model will be referred to as the Workflow Security Module (WSM). The figure depicts four sites—three of these sites are 'normal' database sites associated with the various business partners who share the federated database. The fourth site controls the workflow activities. In fact, this 'site' does not have to be a separate site—it can share the physical facilities of one or more of the other sites. However, it does have a distinct role in the federation and makes use of the facilities provided by the SPO model just like any 'normal' site does. Note that the WSM does not form part of the trusted components in the SPO model; the workflow site therefore does not increase the complexity of the SPO model, which is used to enforce security in the final instance (and therefore requires a high level of trust). In particular, sites that do not make use of the workflow facilities do not have to be aware that a WSM exists in the federation.

The WSM forms an additional layer of security enforcement: It cannot violate any

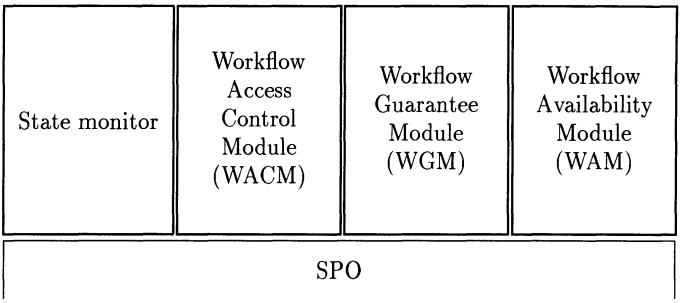


Figure 4 Architecture of the workflow security module

security enforced by the SPO model, but can itself enforce additional security. To this extent, it has to be trusted by the users of the workflow facilities.

In order to use the system the (non-workflow) access requirements are determined and entered into the base SPO system. If all processing on the federation occurs as part of workflow processes, this step can be automated by analysing the workflow scripts. (Note that, if only a few ad hoc operations can occur, workflow processes can be defined for those operations, where each process consists of a single such operation.) The system is now ready to use the workflow information dynamically.

Figure 4 illustrates the architecture of the WSM. The state monitor keeps track of the events that have already occurred in a workflow process. The workflow access control module maintains user information to prohibit subjects to access the objects at inappropriate times. The workflow guarantee module ensures that mutual expectations about the use of concerned objects are met. The workflow availability module attempts to relocate (and replicate) objects such that they are available for subjects who next have to deal with them.

Before each of these modules are discussed in more detail, it is necessary to point out that the workflow site, as described here, is only responsible for security during the workflow process. Compilers to translate the workflow scripts from their external representations, messaging systems to inform participants that they next have to deal with an event in a workflow process and the user interfaces through which entities involved in a workflow process are manipulated all fall outside the scope of the site as described here.

4.1 State monitor

As explained earlier, security specifications in a workflow system apply from some specific event to some other specific event—consider, for example, the claim workflow process earlier where the approver was authorised to determine the status of the claim from the *approval* event up to the *destroy* event.

Changes in the security checks are therefore associated with the events in the system. The role of the state monitor is to react to events by informing the access control, guar-

antee and availability modules of the workflow site of the events so that they can take the appropriate actions.

Most events are associated with a particular message. The SPO architecture allows the association of a security method with any other method, such that the security method is also executed when the 'protected' method is executed. This facility can be used to inform the state monitor that an event has occurred: simply associate such a security method with the method that corresponds with such an 'event' message which will report the event.

The state monitor is also responsible to initiate time-based events when necessary, and to send messages specified in the workflow script in such a case.

4.2 Access control

Since the workflow access control module (WACM) forms an additional layer on top of the existing access controls, complete access control is divided amongst this module and the SPO base. Using the static analysis, subjects who, at some time, need access to a protected entity are given the necessary permissions using the facilities provided by the SPO base. Since the SPO base is more trusted than the workflow access control module, a high level of trust exists that only subjects who need access will ever be able to access the protected entities. As in any SPO database, the underlying access control mechanisms can be based on discretionary and/or mandatory mechanisms (Olivier, 1995, 1996a). Here we will use a role, group and identity-based approaches for the workflow level, although other possibilities exist.

The workflow access control module 'superimposes' dynamic security on this static security by maintaining an access control list for all entities specified in the workflow process—in this particular case the access control list will contain the identities of individuals or that of groups or roles who may access the associated entity. See Olivier (1996a) for a description of how a module such as the WACM can maintain such information.

Whenever an event is registered by the state monitor the WACM has to update the access control lists for the concerned workflow process. Since the protected entities remain the same, this only requires modification of the associated access control lists. Further, since the workflow process normally passes a task from one subject to the next, changes to authorisations occur gradually, so that maintenance of the additional access control list does not present a significant overhead.

The WACM is also responsible for maintaining information about roles created for the duration of a workflow process (such as the *Claimant* and *Approver* roles used earlier).

For the claim example of a previous section the entities concerned may be the `GetStatus`, `Approve`, `Deny` and `Pay` methods of the `Claim` object and the `Create` method of the `Claim` class. Initially, the WACM need only protect `Create` by limiting access to employees. (Note that this protection will actually already be performed by the base so that explicit protection by the WACM is unnecessary.) When the *Submit* event occurs, the *Claimant* role is created and this role is added to the access control list of the particular `GetStatus` method. Simultaneously, the *Manager* role is added to the access control list of the `Approve` and `Deny` methods. When either event occurs, these roles are again removed and the access control lists further updated as required. This process continues until the workflow process eventually terminates.

4.3 Guarantees

As argued earlier, guarantees depend on the state of a workflow process: It may be perfectly acceptable for a claimant to modify a submitted claim before it has been approved, but definitely not after it has been approved.

Guarantee support in federated databases has been described by Olivier (1996b). There it was suggested that the user of a service requests a guarantee from a certification site, while the owner of the concerned entity provides the guarantee to the user via this certification site. In the workflow environment a similar procedure may be followed except that all guarantees required for the workflow process are specified in advance in the workflow script. The workflow guarantee module (WGM) can therefore request the necessary guarantee from the owner of the concerned entity when it is required. The WGM thus forms a centralised guarantee request module for a given workflow process.

4.4 Availability

Secure object relocation is a central objective of the SPO model. Object relocation can be used to enhance availability—especially access speed. (Note that SPO requires communication with owning sites to make access decisions; availability during network failures of an object relocated to a local site is not guaranteed. Disconnected operation is currently being investigated.)

In order to use this facility, the workflow availability module (WAM) needs to know the sites from which currently authorised subjects are likely to access an object. In the claim example, after submission of a claim, the `Claim` object may be relocated to a site at head office, if all authorised managers are located there. However, the fact that the claimant (at, say, the factory) can still obtain the status of the claim presents a number of interesting possibilities: It can be argued that approval is more important than obtaining the status and therefore relocation is recommended. (The status can still be obtained, albeit possibly slightly slower, unless the network fails.) Alternatively replication rather than relocation can be considered. Of course the size of relocatable objects, as well as the size of messages to access these objects, all influence the decision.

Since the primary role of the WAM will be to increase availability during disconnected operation, it is not discussed further in this paper.

4.5 SPO

In order to implement the WSM as described above, a number of assumptions have to be made about the SPO base. Can, for example, the WSM change access control for entities owned by other sites? If the WSM is trusted by all members of the federated database, SPO can be modified to support this. If not, attachment of security methods can cause problems—even if such security methods cannot override existing security methods.

Two alternative possibilities exist. Firstly, the workflow system can be defined such that all concerned entities are owned by the WSM. The WSM is then free to attach security methods to any object. The WSM then also acts as requester and provider of integrity guarantees. And the WSM has the final say about relocation of objects owned by it.

If ownership of all entities cannot be transferred to the WSM, The TLEs (see section 2) at the sites that participate in workflow processes can be modified to assign security

methods to entities owned by each such site on behalf of the WSM. This does, however, require that the participants in a workflow process trust one another's TLEs.

5 CONCLUSION

This paper described a system that uses the concept of workflow processes to increase the security of the system. It has been shown that workflow information can be used to increase the confidentiality, availability and integrity of information in the system.

The architecture that has been proposed is intended to be used in conjunction with an SPO secure federated database. The architecture uses workflow scripts to dynamically control security as a workflow process proceeds.

Static analysis of workflow processes has been mentioned and used in the paper, but the potential of static analysis has not been investigated fully yet. Such static analysis has the potential to enhance integrity by providing information to use as basis during deadlock prevention or deadlock resolution and other concurrency control issues. This requires further investigation.

The principles described in this paper are currently being investigated for use during disconnected operation. Their use in mobile computing seems promising.

6 REFERENCES

Bock, G.E. and Marca, D.A. (1995) *Designing Groupware: A Guidebook for Designers, Implementors, and Users.* McGraw-Hill, New York, New York.

Kim, W., ed. (1995) *Modern Database Systems: The Object Model, Interoperability and Beyond.* ACM Press, New York, New York.

Lunt, T.F. (1995) Authorization in Object-Oriented Databases, in *Modern Database Systems: The Object Model, Interoperability and Beyond* (ed. W. Kim) ACM Press, New York, New York, 130–45.

Olivier, M.S. and Von Solms, S.H. (1994) A Taxonomy for Secure Object-oriented Databases. *ACM Transactions on Database Systems*, **19**, 1, 3–46.

Olivier, M.S. (1995) Self-protecting Objects in a Secure Federated Database. Ninth IFIP WG 11.3 Conference on Database Security, Rensselaerville, New York.

Olivier, M.S. (1996a) Supporting Site Security Policies for Members of Federated Databases. Fourth European Conference on Information Systems, Lisbon.

Olivier, M.S. (1996b) Integrity Constraints in Federated Databases. Submitted.

Özsu, M.T. and Valduriez, P. (1991) *Principles of Distributed Database Systems.* Prentice-Hall, Englewood Cliffs, New Jersey.

Rabitti, F., Bertino, E., Kim, W. and Woelk, D. (1991) A Model of Authorization for Next-Generation Database Systems. *ACM Transactions on Database Systems*, **16**, 1, 88–131.

Sheth, A.P. and Larson, J.A. (1990) Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, **22**, 3, 183–236.

Thuraisingham, B. (1994) Security issues for federated database systems. *Computers & Security*, **13**, 509–25.