X^{2}TS: Unbundling Active Object Systems

C. Liebig, M. Malva, A. Buchmann
Darmstadt University of Technology
{chris,malva,buchmann}@dvs1.informatik.tu-darmstadt.de

Workflow Management Systems exhibit task management and coordination functionality at the application level. Separating coordination of activities from their realization is the basis for a flexible and extensible software architecture for workflow management applications. We are interested in applying concepts of workflow management to distributed object systems, CORBA in particular, where the business logic of applications is implemented by composed objects. Our research is motivated by the requirements of a concrete application, an Integrated Tower System (ITS) that is part of the new German Air Traffic Control System (ATC). The ATC-ITS has workflow-like requirements and shall be built in a component-oriented manner on a CORBA platform. A primary requirement is to ensure reliable execution and provide support for rich failure semantics and exception handling, like compensation and contingency actions. As part of the ongoing research we are investigating how to incorporate advanced transaction management in CORBA by unbundling concepts of active DBMSs and how to integrate those concepts with the CORBA Object Transaction Service and a notification service. The resulting service implements an event-action model with transactional behavior and rich coupling modes and is used as a basic building block for reliable components.

1. Introduction

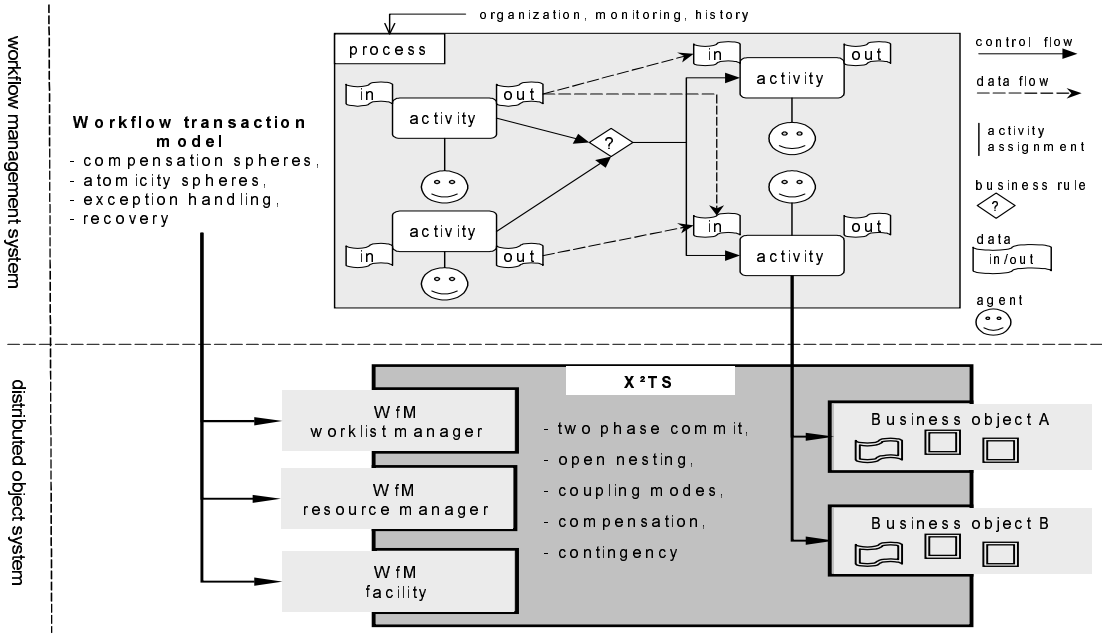
Workflow management systems (Wfms) have attracted a great deal of attention both in industry and research. The practical success of Wfms is derived from the fact that Wfms enable programming in the large, providing support for organizational aspects, application integration, monitoring, distribution and heterogeneity [2,21]. A workflow is a collection of activities organized to accomplish some business process. When defining a workflow, activities represent *what* to do and link to the application or business object that implements the functionality. The control and data flow between activities is specified separately, so is the aspect of staff assignment. Separation of concerns is the basis for a flexible and extensible software architecture for workflow management applications. It is evident that workflow management systems and component-oriented systems have similar goals and can benefit from each other [1,25,34,35]. Moreover, a broad range of workflow management applications in distributed heterogeneous environments will be based on distributed object systems like CORBA [9,32,36,37,40]. Consequently, both the Wfms and the implementation of activities may be realized in a component-based manner while making (re)use of services provided by the middleware. The distinction between enterprise-wide programming in the large and composing CORBA objects and applications from components, i.e. programming in the small, allows to build flexible and extensible systems with high potential for reuse [24,34,35]. For the architecture of such a system two levels of abstraction can be identified: on the top level we apply the concepts of workflow management systems whereas at the bottom level activities and the runtime components of the Wfms itself are implemented using composed CORBA objects. Similarily, we distinguish between reliable execution and support for rich failure handling and recovery in workflows through *workflow transaction models* [3,11,13,23,24,38,41] on the top level, and reliable execution encompassing distributed objects through an *object transaction model* (e.g. [6]) on the object system level.

Reported research and prototypes in this area apply event-driven architectural styles to realize the Wfms runtime component which controls the workflow and enforces task interdependencies [1,9,16,20,23]. Reliable execution over distributed heterogeneous objects is typically achieved by applying transaction concepts, ranging from the traditional ACID model to extended transaction models [1,6,14,22]. The relationship of a generic workflow management facility

to existing CORBA services and the reuse and the integration thereof needs further investigation [36,37]. These observations motivate our work to integrate and extend the pertinent CORBA services, namely Object Transaction Service [31] and a (simplified) Event/Notification Service [30]. Our goals is to provide middleware needed to build reliable components of workflow management runtime systems, as well as, business objects that implement functionality of workflow activities. In particular we suggest to unbundle concepts from *active object systems* [5,6] and to realize an event-action model with transactional behavior and rich coupling modes. In fact, several research prototypes realize various subsets of active object systems [7,12,16,23] tightly integrated in the Wfms runtime but do not unbundle the functionality to provide added-value services.

Our research is particularly motivated by the need for design and development of a next-generation *integrated tower system* in air traffic control (ATC-ITS), especially a workflow management system to support gate-to-runway control. The ATC-ITS is to be built on a CORBA platform and encompasses the integration of distributed and heterogeneous component systems. When evaluating the above approach for the ATC-ITS scenario, we found that the traditional ACID transaction model at the object system layer is too restrictive and inflexible for the ITS environment. Thus we suggest to unbundle concepts from active object systems [4,5,6] and incorporate features like open nesting, contingency and compensating actions into an extended CORBA transaction service (X^2TS). In X^2TS we integrate a simple notification service with CORBA transactions to provide event-driven interactions with support for coupling modes and transaction context propagation. The use of X^2TS is twofold: it provides failure atomicity and isolation for a component-oriented implementation of the functionality that is linked to an activity. Additionally, the implementation of a distributed workflow engine itself, encompassing components like process engine, resource manager and worklist manager may be built using X^2TS. Consequently, the use of the transactional event-action service provides the glue between the execution of activities and the invoked CORBA objects, and the workflow engine that controls both levels. We follow the principle of the OMA [27], that "Object Services are the basic building blocks for distributed object applications... They can be used to construct higher level facilities and object frameworks...". In that way, a CORBA workflow management facility [32] can benefit from the proposed extended transaction coordination services to make flexible failure handling and recovery feasible through facilitating system services.

Figure 1. Concepts and architecture



In this paper we focus on the design and prototypical implementation of X^2TS, developed in our research group, and on the integration of a subscription-based notification service within X^2TS. The subscription-based notification service with subject-based filtering is particularly well-suited for ATC applications as shown in [26]. The impact of organizational aspects, collaboration and the workflow transaction model and definition of the *ATC tower workflow* are part of the ongoing work and will be presented elsewhere.

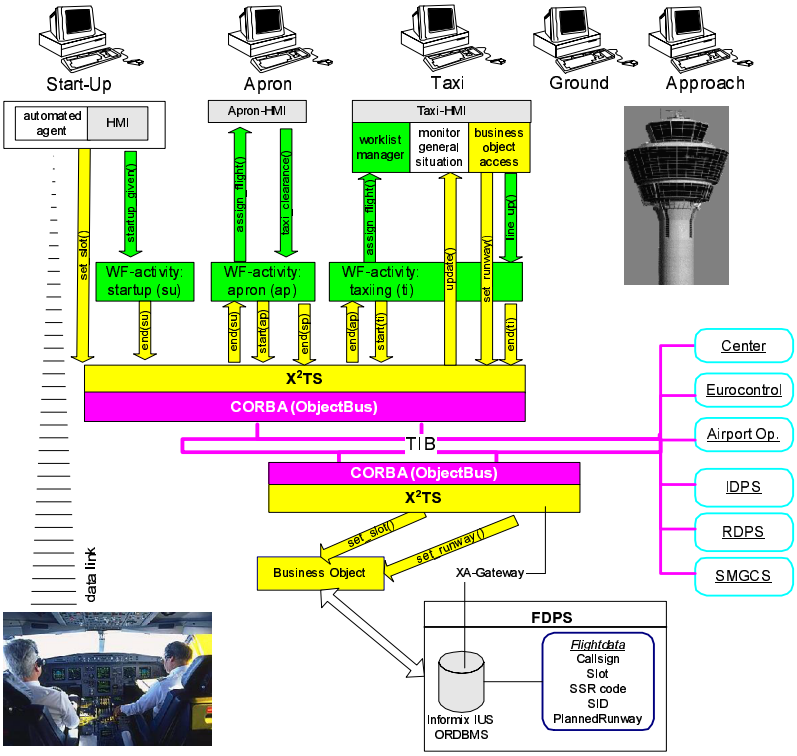
The rest of this paper is organized as follows. In Section 2 we introduce the ITS scenario and the overall system architecture. Section 3 presents the features and some implementation issues of X^2TS. Section 4 concludes the paper.

2. Integrated Tower System

German ATC authorities identified the airports as likely bottlenecks in handling the expected increase in air traffic. Next generation tower systems must reflect the collaborative manner of operational procedures, support the controller in making the right decisions and exploit opportunities for automated processing while preserving correct and reliable operation. Each of the 17 airports currently operated by the DFS (Deutsche Flugsicherung) requires different procedures imposed by the geographical layout of runways, taxiways, departure routes and organizational differences. Moreover, today's *external* IT systems, e.g. airport operator facilities, airline IT systems, en-route ATC center, national and supra-national flight plan processing systems (e.g. CFMU at Eurocontrol), will be tightly integrated with next-generation ITS [15]. The same holds for an aircraft's on-board computer which is connected by wireless communication networks, known as *datalink*, and allows to implement, for example, semi-automatic departure clearance functions. It is not the responsibility of the ITS to autonomously make decisions but to provide the controller with current situation information for planning, scheduling and guiding aircraft movements, as well as the movements of auxiliary equipment (fuel trucks, buses, etc.). Additionally, the ITS must support the collaboration between controllers with interfering areas of competence. The DFS stated clearly that the ITS may never limit the possibilities of a controller but inform him about probable conflicts. To accommodate the required flexibility and extensibility, a workflow like approach for the purpose of coordinating the various tasks of ATC procedures is proposed. It offers the required flexibility to cope with the different airport situations and organizational configurations, because the general workflow can be customized to fit the situation at hand.

The most complex work in the airport tower is to coordinate the on-ground gate-to-runway and runway-to-gate traffic. The workflow for departure clearance procedures typically involves several controllers with different roles. Initially positioned at the terminal block, the crew of an aircraft negotiates with the *startup controller* the flight plan data, such as call sign, category of aircraft, slot, destination and standard instrument departure route (SID). As shown in Figure 2. the startup clearance can be either performed by a startup controller via the appropriate HMI and voice radio or by an automated agent via the data link. Depending on the earliest slot time, current terminal block and taxi conditions, the startup controller acknowledges the startup request and hands over to the *apron controller*, who will guide the push-back and taxiing off the apron. The apron controller in turn hands over to the responsible *taxi controller* who guides the aircraft to the runway and lines it up according to the preassigned slot times and local restrictions. Along with the general status information about weather, runways, taxiways, ILS, beacons etc., each controller monitors the flight plan data of the aircraft currently in his competence zone as well as flight plan data (FPD) of aircraft in adjacent areas. Final line-up and take-off clearance is the responsibility of the *ground controller*. Collaboration is needed with *approach control* as take-off clearance for aircraft must be carefully coordinated with approaching aircraft. Typically, controller in the tower, apron, approach and en-route controller are situated in different organizational units and locations.

Figure 2. Integrated Tower System



The main components of the ITS are the HMI at the controller's working position, integrating the worklist manager, access to the business objects, and the situation monitoring. The Flight Data Processing System (FDPS) provides persistent storage for flight plan business objects. Additionally, various external systems like ATC center, radar data processing systems (RDPS), weather information systems (IDPS), surface guidance and control systems (SGCMS), and logging facilities are connected and integrated. Figure 2. shows the overall architecture of the ITS system.

We must consider the heterogeneous and distributed nature of the involved technical facilities when implementing an ITS. Typical business objects are realized by local, regional and supra-regional systems, such as technical facilities for ATC (radar, beacons etc.), airport operators, aircraft, bounding ATC centers, flight plan data processing systems, central flow management units, airline operators, etc. Some business objects may provide native support for transactions, for example, flight plan data objects and SID configurations, while others merely wrap legacy systems or external technical facilities. For example, the transponder of the aircraft is configured with a so called SSR code to identify the call sign to the terminal radar. A currently unused SSR code must be requested from a pool of SSR codes at the terminal radar system. This operation cannot be rolled back but can only be compensated. In a component-oriented system, business objects are constructed from smaller, possibly distributed, components. Isolation of computations is an issue, because the situation monitoring and planning services must not see inconsistent results or at least must be notified of rolled back or compensated transactions. All steps of a process must be logged for auditing purposes, some must be published to external systems to allow billing, flow control and planning procedures to be updated. It is important that the process status captured by the workflow engine is consistent with the actions happening in the real world.

The workflow is enforced by an event-driven transactional workflow system. At its heart lies the transactional event-action service (X^2TS), which propagates workflow events (activity/process status change, exceptions, agent assignment) as well as application events (e.g. state change notifications for display purposes at multiple working positions) and realizes the triggering of reactions and follow-up activities. The workflow engine is designed following a fully distributed architecture, similar to the approach suggested in [9].

Using X^2TS as the glue between business objects and workflow management system components - process manager, task manager, resource manager and worklist manager - provides for seamless transaction context propagation, spanning units of atomicity and isolation across the implementation boundaries. Using flexible coupling of transactions

allows to provide a highly integrated architecture with respect to reliability and failure tolerance. Coupling modes determine the execution of triggered actions relative to the transaction in which the triggering event was published [4,5,10]. Based on X^2TS, a workflow transaction model may be realized that supports atomic spheres and contingency spheres for collection of activities borrowing ideas from [1,19,25]. An atomic sphere is a collection of activities that can be backed out by undoing all business object executions in an atomic (all-or-nothing) way and control flow reenters at the initiating activities. Obviously, the applicability of this concept not only depends on the semantic of the affected activities at the workflow model level, but also on their implementation, i.e. the transaction support of linked business objects. An application of atomic spheres to the ITS scenario is the subworkflow that is executed in order to change the direction of a runway. Such a situation arises when the weather conditions impose the inversion of the *planned runway*. Changing the planned runway direction encompasses an activity that adopts the SIDs for all FDPs appropriately, activities to reconfigure the ILS and beacon systems and most important the notification of all affected controllers. Either all activities must execute successfully or none of the effects may remain, i.e. in that case we need to backout all activities.

In the ITS scenario there is a common requirement for more flexible exception handling at the workflow model level. Compensational spheres and subworkflows for exception handling are more adequate in situations, when the execution of activities have affected the real world in a way such that backout at the IT system level is not appropriate and forward recovery (at the workflow level) is needed. Note, that even in this case, partial backward recovery at the business object system level may be required. On the other hand, there are cases, where business objects do support transactional semantic but still at the workflow level compensational activities must be executed.

We believe that event-driven computing combined with transactional coupling is a suitable approach to compose business objects out of smaller units of functionality and that the implementation of a workflow transaction model for flexible and reliable workflow executions can benefit from such middleware services.

3. X^2TS

The goal of unbundling active object systems [4,5,6] is to provide a CORBA service for event-driven interactions that also supports transaction context propagation and coupling modes. The extended object transaction model incorporates features like open nesting, contingency and compensating actions.

An active business object may subscribe to events of interest, optionally specifying a coupling mode. Thereby event-action rules can be realized that are automatically triggered and executed within a certain transaction context. Additionally, the triggering of the event-action rule depends on the status and/or outcome of the triggering transaction as requested with the subscription.

An active business object may register a compensating object for a resource. The compensation method is invoked automatically by X^2TS if either the transaction with which the resource (and compensation) is registered or any of its ancestors is rolled back. This mechanism allows to realize SAGAs [8] and incorporate component systems that do not cooperate in atomic commitment.

X^2TS ensures that the subscribed objects are automatically triggered and executed within the specified transaction context and coupling. Thus X^2TS offers the possibility to realize reliable event-driven object invocations and supports rich failure semantics and exception handling.

The dimensions in defining a coupling mode are [4,5,10]:

- when the reacting object should be notified,
- the transaction context within which the object should react, and
- the commit(abort)-dependency of the triggered action with respect to the triggering transaction

An object may either be notified

- immediately,
- deferred,
- sequentially on commit,
- sequentially on abort or
- sequentially on termination.

The triggered action may execute its operations either

- within the transaction context of the triggering transaction,
- within a child of the triggering transaction,
- within a sibling of the triggering transaction or
- within its own top transaction.

The triggered actions are either

- commit-dependent,
- abort-dependent,
- non-dependent,
- vital or non-vital.

X^2TS integrates and extends the CORBA Object Transaction and Notification Services [30,31]. We argue that the basic mechanisms provided by an object transaction service (OTS) - indirect context management, implicit context propagation and commit callback handling - are a suitable basis for incorporating the features described above. Our X^2TS prototype is based on an extended object transaction service implementation and a simple push-push notification service leveraging multicast enabled messaging middleware [26].

The CORBA OTS provides a framework to manage transactional contexts and orchestrate the two-phase-commit processing. OTS neither provides failure atomicity nor isolation itself but delegates the implementation of recovery and isolation properties to the participating recoverable servers. Isolation can be either implemented by the transactional object itself or by use of the Concurrency Control Service [29]. OTS supports closed nested transactions. Interoperability with X/Open resources through an XA-gateway [43] is straightforward and has been implemented for the X^2TS prototype - an extension that is essential when accessing a commercial database system.

A CORBA recoverable server object must agree upon a convention of registering callback objects with the OTS `Coordinator` which drives the 2PC through invocation of callback methods. The callback methods provided by OTS are shown in Figure 3.

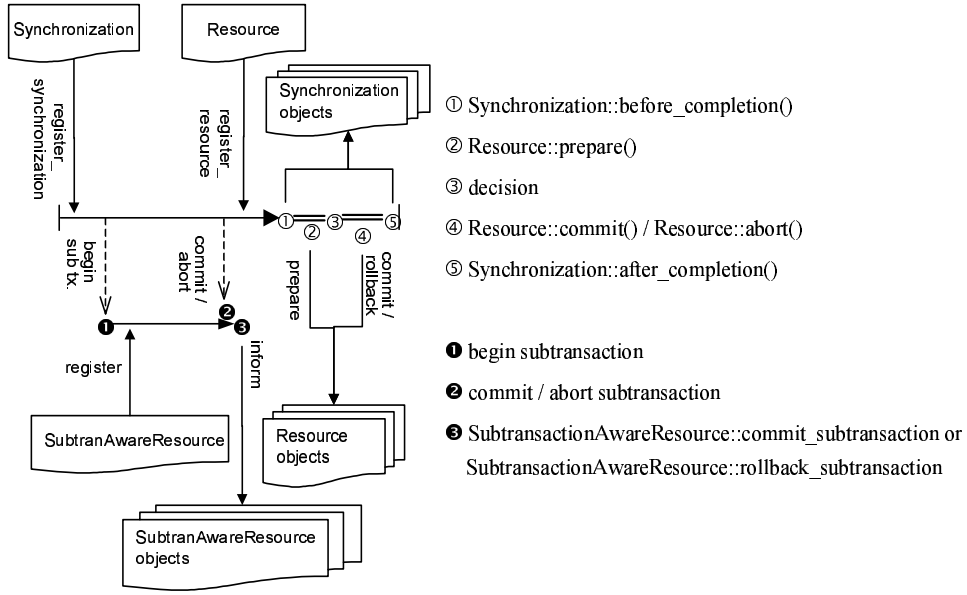


Figure 3. Callbacks in OTS

The basic mechanisms of an OTS that we exploit in X^{2}TS are: interposition, indirect context management and implicit context propagation. In the following we outline how theses features are implemented in our prototype.

The OTS specification suggests the use of interposition when implementing a transaction service: when a transaction reaches a new domain, a subordinate coordinator is created locally and registers its participation in the transaction with its superior coordinator [17]. Through the subordinate's registration, all callbacks on behalf of the superior's 2PC processing are relayed to the subordinate and in turn forwarded to all `Resource`, all `SubtranAwareResouce` and all `Synchronization` objects. In order to realize interposition, pass-by-value semantic for `Control` objects - which encapsulate the transaction hierarchy is needed. The following steps must be implemented:

- transmit the state of the superior coordinator along with the object reference,
- construct a subordinate coordinator out of the transmitted data in the target domain,
- use the local subordinate coordinator instead of the remote superior coordinator for all locally resolvable calls,
- register the subordinate coordinator with the superior coordinator.

In X^{2}TS we realize the above behavior by modifying the generated stubs and skeletons. The registration of subordinate coordinators with their superior coordinator takes place on return from the method invocation. No additional remote calls are needed as the necessary data is added by dedicated interceptors to the request/reply packets of the transactional method call.

Indirect context management means that clients of an OTS do not access the `Control` object directly but use the `Current` pseudo object which manages the thread transaction association.

Implicit context propagation implies that the transaction context is not passed explicitly as a parameter with a method invocation but is to be transmitted transparently whenever a method on a transactional object is invoked. In our implementation transactional objects are declared by policy and there is no need to derive from the `TransactionalObject` dummy interface, in accordance with [28]. The implementation is based on CORBA interceptors. We distinguish client interceptors and server interceptors. A client interceptor is invoked before and after a request is marshalled but before the ORB transmits the request from the client site to the server site. A server interceptor is invoked before and after the

request is unmarshalled but before the implementation object will be called by the ORB. Thus interceptors can be used to transparently piggyback the transaction context with a method invocation.

Our Notification Service features a simple push-push model with subject-based filtering. The event supplier interacts with a consumer proxy which resides on the publisher site. On each consumer site resides a supplier proxy with which the event consumer interacts. The event supplier will publish events to the consumer proxy which will forward (push) them to all supplier proxies. In X^2TS the publication of an event is mapped to a multicast method invocation to the group of supplier proxies that are subscribed to the specific event subject [26,39]. Finally, the supplier proxies will forward (push) the event to all consumers which are subscribed to the incoming event.

We argue that context management, context propagation and callback handling are a suitable basis for incorporating extended transaction coordination and to realize transaction dependencies as suggested by the DOM transaction model [6]. In order to support coupling modes, the notification service and the object transaction service must be integrated. The following issues arise:

(i) implicit transaction context propagation with event publication,

(ii) event notification dependent on transaction status and outcome,

(iii) enforcing the commit dependencies between triggering and triggered transaction,

(iv) ensuring the successful completion of all vital triggers and

(v) realizing compensations.

As the publication of an event is mapped to a multicast method invocation, the techniques for interposition and context propagation can be leveraged:

- the transaction context of the triggering transaction is implicitly shipped with the event to the subscriber's site,
- an interposed coordinator is constructed and
- the interposed coordinators are registered on return from the multicast method invocation with the coordinator of the triggering transaction.

The supplier proxy performs event notification depending upon transaction status and outcome. It does so by buffering the events until the transaction is in the required state or outcome. The supplier proxy uses the Synchronization interface to be notified about the status and outcome of a top-level transaction. As shown in Figure 3. a Synchronization object is invoked prior to the start and after the termination of the two-phase commit protocol. Only if theses callbacks are invoked the buffered events are forwarded to the subscribed objects. To be notified about the transaction outcome via the Synchronization interface the supplier proxy must register a Synchronization object with the triggering transaction. If, for example, the supplier proxy is notified about the commit of a transaction the events will be forwarded to all causally dependent subscribers. In case of coupling on a subtransaction the SubtransactionAwareResource is used.

In order to establish the commit(abort) dependencies between a triggering transaction A and a triggered transaction B, the outcome of the 2PC of B must depend upon the outcome of A. Therefore, a special Resource object is registered with B. This Resource object votes during the 2PC of B according to the coupling mode and the outcome of A. Again the Synchronization and the SubtransactionAwareResource interface can be used to propagate the outcome of transaction A to the special resource.

Compensation is realized by establishing an abort dependency with the transaction and all its ancestor transactions in the transaction tree.

In the sketched approach, the interposed coordinators are registered on return from the multicast method invocation. If vital actions are to be triggered, this approach is not suitable because it does not guarantee atomic event delivery to *all* vital actions. Therefore, if active objects with vital actions are subscribed, we register a dedicated resource with the triggering transaction before publishing the event. This dedicated resource is configured with the group of subscribers that registered vital event-actions. During 2PC it contacts each member of the group of interposed subcoordinators and thereby enforces atomicity with respect to all vital triggers.

4. Current Status and Future Work

X^2TS is a complete reimplementation of the basic XTS system [18] with many extensions and integrated with a simple push-based notification service. X^2TS incorporates a powerful framework to realize an extended object transaction model with support for open and closed nesting, contingency transactions, and compensating transactions for undoing committed subtransactions in an open nested model. Because of the more powerful transaction model some of the basic assumptions made in OTS (and XTS), such as the assumption of presumed abort, must be revised. In this case an expanded logging facility is required. To make the two phase commit more efficient, it may be implemented using a multicast mechanism instead of the conventional unicast. At present we implemented the basic coupling modes. An ongoing effort is the adaptation of XTS's concurrency control service and the implementation of the full range of coupling modes. Additionally, we are investigating, how coupling modes and transactional subscriptions should be specified in accordance with the QoS policy mechanisms found in [28,30]. Some open issues remain with respect to recovery.

At the level of the workflow management system, we are researching models to handle workflows with exceptional behavior and to support workflow recovery based on the ideas found in [19,20,25]. Such a model must be integrated in the workflow engine using X^2TS, and the semantics of our workflows will be validated against the requirements of the ITS application.

5. References

[1] G. Alonso, C. Hagen and H.-J. Schek and M. Tresch. *Distributed Processing over Stand-alone Systems and Applications*. VLDB Conference 1997, Athens, Greece, 1997.

[2] G. Alonso, D. Agrawal, A. El Abbadi and C. Mohan. *Functionality and Limitations of Current Workflow Management Systems.* IEEE Expert 1997.

[3] G. Alonso, D. Agrawal, A. El Abbadi, M. Kamath, R. Guenthoer and C. Mohan. *Advanced Transaction Models in Workflow Contexts*. Proc. 12th International Conference on Data Engineering, New Orleans, February 1996.

[4] H. Branding, A. Buchmann, T. Kurdass and J. Zimmermann. *Rules in an Open System: The REACH Rule System*. In Proceedings of Intl. Workshop on Rules in Database Systems (RIDS 93), Edinburgh, Scotland, September 1993.

[5] A. P. Buchmann. *Active Object Systems.* In A. Dogac, M.T. Özsu, A. Biliris, T. Sellis: Advances in Object-Oriented Database Systems; Springer Verlag, 1994.

[6] A. Buchmann, M.T. Özsu, M. Hornick, D. Georgakopoulos and F.A. Manola. *A Transaction Model for Active Distributed Object Systems*. In [14].

[7] S. Ceri, P.W.P.J. Grefen and G. Sanchez. *WIDE: A Distributed Architecture for Workflow Management*. Research Issues in Data Engineering (RIDE), Birmingham, England, 1997.

[8] P.K. Chrysanthis and K. Ramamritham. *ACTA: The SAGA Continues*. In [14].

[9] S. Das, K. Kochut, J. Miller, A. Sheth and D. Worah. *ORBWork: A Reliable Distributed CORBA-based Workflow Enactment System for METEOR_2*. Technical Report UGA-CS-TR-97-001, Department of Computer Science, University of Georgia, 1997.

[10] U. Dayal, A. Buchmann and D. McCarthy. *Rules are Objects too: a knowledge model for an active, object-oriented database system*. 2nd Intl. Workshop on Object-Oriented Database Systems, Lecture Notes in Computer Science 334, Springer, 1988.

[11] U. Dayal, M. Hsu and R. Ladin. *A Transaction Model for Long-Running Activities*. In Proc. of the 17th VLDB Conference, September 1991.

[12] J. Eder, H. Groiss and H. Nekvasil. *A Workflow System Based on Active Databases*. Chroust G.,Benczúr A. (eds.): Proc. Connectivity-94: Workflow Management - Challenges, Paradigms and Products CON 1994, Oldenbourg Verlag, Wien, München, 1994.

[13] J. Eder and W. Liebhart. *Workflow Transactions*. In P. Lawrence (ed.) Workflow Handbook, Wiley & Sons 1997.

[14] A.K. Elmagarmid (Edit.). *Database Transaction Models for Advanced Applications*. Morgan Kaufmann, 1992.

[15] Eurocontrol, *EATMS Operational Concept Document*. Ver. 1.1. Eurocontrol Brussels, http://www.eurocontrol.be/projects/eatchip/ocd/

[16] A. Geppert and D. Tombros. *Event-based Distributed Workflow Execution with EVE*. Proc. of IFIP Intl. Conf. on Distributed Systems Platforms and Open Distributed Processing (MIDDLEWARE'98), September 15-18, The Lake District, England, 1998.

[17] E. Grasso. *Implementing Interposition in CORBA Object Transaction Service*. 1st Intl. Enterprise Distributed Object Computing, Gold Coast, Australia, 24-26 October 1997.

[18] E. Grasso. *An Extended Transaction Service for Real-Time and Telecom Object Request Brokers*. 2nd Intl. Workshop on Distributed Object Oriented Computing, Frankfurt, Germany, 11. October 1996.

[19] P. Grefen, J. Vonk, E. Boertjes and P.Apers. *Semantics and Architecture of Global Transaction Support in Workflow Environments*. Intl. Conf. on Cooperative Information Systems (CoopIS 99), September 1999.

[20] C. Hagen and G. Alonso. *Beyond the Black Box: Event-based Inter-Process Communication in Process Support Systems*. 19th Intl. Conf. on Distributed Computing Systems (ICDCS), Austin, Texas, USA, June 1999.

[21] S. Jablonski and C. Bussler. *Workflow-Management: Modeling Concepts, Architecture and Implementation*. International Thomson Publishing, Bonn, 1996.

[22] S. Jajodia and L. Kerschberg (Edit.). *Advanced Transaction Models and Architectures*. Kluwer Academic Publishers, 1997.

[23] N. Krishnakumar and A. Sheth. *Managing Heterogeneous Multi-system Tasks to Support Enterprise-Wide Operations*. Distributed and Parallel Databases, 3 (2), April 1995, Kluwer Academic Publishers, 1995.

[24] F. Leymann. *Supporting Business Transactions Via Partial Backward Recovery In Workflow Management Systems*. Datenbanksysteme in Büro, Technik und Wissenschaft (BTW), Dresden, Springer, March 1995.

[25] F. Leymann and D. Roller. *Workflow-based applications*. IBM Systems Journal, Vol 36 No. 1, IBM, 1997.

[26] C. Liebig, B. Boesling and A. Buchmann. *A Notification Service for Next-Generation IT Systems in Air Traffic Control*. GI-Workshop *Multicast - Protokolle und Anwendungen*, Braunschweig, Germany, May 1999.

[27] Object Management Group (OMG). *Object Management Architecture Guide*. 3rd Edit., R.M. Soley (ed.), John Wiley&Sons, Inc., New York, 13. June 1995.

[28] Object Management Group (OMG). *CORBA Messaging*. OMG Document orbos/98-05-05, Famingham, MA, May 1998.

[29] Object Management Group (OMG). *Concurrency Control Service Specification*. In *CORBA Services Specification*, Famingham, MA, May 1998.

[30] Object Management Group (OMG). *Notification Service Specification*. OMG Document telecom/98-06-15, June 1998.

[31] Object Management Group (OMG). *Transaction Service Specification*. In *CORBA Services Specification*, Famingham, MA, May 1998.

[32] .Object Management Group (OMG). *Workflow Management Facility*. OMG Document bom/98-07-07, Famingham, MA, July 1998.

[33] B. Oki, M. Pfluegl, A. Siegel and D. Skeen. *The Information Bus - An Architecture for Extensible Distributed Systems*. In Proceedings of SIGOPS 93, December 1993.

[34] F. Ranno, S.K. Shrivastava and S.M. Wheater. *A System for Specifying and Coordinating the Execution of Reliable Distributed Applications*. International Working Conference on Distributed Applications and Interoperable Systems (DAIS'97), Cottbus, Germany, September 1999.

[35] S. Schreyjak. *Coupling of Workflow and Component-Oriented Systems*. Proc. of the Second International Workshop on Component-Oriented Programming (WCOP '97), Jyväskylä, Finnland, 1997.

[36] W. Schulze. *Fitting the Workflow Management Facility into the Object Management Architecture*. 3rd Workshop on Business Object Desing and Implementation, OOPSLA'97, Atlanta, Georgia, USA, 1997.

[37] W. Schulze, C. Bussler and K. Meyer-Wegener. *Standardising on Workflow-Management - The OMG Workflow Management Facility*. SIGGROUP Bulletin, Vol 19 (3), April 1998.

[38] A. Sheth and M. Rusinkiewicz. *On Transactional Workflows*. Data Engineering Bulletin 16(2) 1-4, June 1993.

[39] TIBCO Software Inc., *TIB/Active Enterprise*. http://www.tibco.com/products/active_enterprise/index.html, TIBCO Software Inc., Palo Alto, USA.

[40] S.M. Wheater, S.K. Shrivastava and F. Ranno. *A CORBA Compliant Transactional Workflow System for Internet Applications*. Proc. of IFIP Intl. Conf. on Distributed Systems Platforms and Open Distributed Processing (MIDDLEWARE'98), September 15-18, The Lake District, England, 1998.

[41] D. Worah and A. Sheth. *Transactions in Transactional Workflows*. In [22].

[42] Workflow Management Coalition. *WfMC Terminology and Glossary.* Technical Report TC1011 Issue 3.0, February 1999.

[43] X/Open DTP. *Distributed Transaction Processing: Reference Model, The XA Specification.* Reading, Berkshire, England, X/Open Ltd., 1991.