Date of acceptance Grade

Instructor

Cross-organizational and distributed EBT support for workflow management applications

Michael Duku-Kaakyire

Helsinki November 30, 2009

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Michael Duku-Kaakyire

Työn nimi — Arbetets titel — Title

Cross-organizational and distributed EBT support for workflow management applications

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	November 30, 2009	20 pages + 0 appendices

Tiivistelmä — Referat — Abstract

For some time now, workflow-management systems have become an accepted technology to support automation in process-centric environments. Traditional simple transaction model provided by current DBMS is inadequate for workflow management applications as they require advanced transaction management functionality. Lately there has been a trend to distribute workflow executions, which requires an even more advanced transaction support system that can handle this distribution. Furthermore, organizations outsource supporting processes to other organizations and concentrate mostly on their core business processes. Organizations form a virtual enterprise where provider organizations offer e-services to consumer organizations. To offer support for these cross-organizational processes, current workflow management technologies would need to be extended. Transaction support, already considered an important issue in intra-organizational workflow management systems, must be extended to deal with the crossorganizational aspects as well. The WIDE approach is an orthogonal two-layer transaction model that supports both high-level and low-level workflow semantics. The model supports two independent transaction modules that each of which manages one layer of the model. This paper presents WIDE and two extensions, the first discussing the extension to the WIDE upper layer providing support for the distributed execution of workflow processes, i.e. distributed global transactions, and the second offering support for cross-organizational processes.

ACM Computing Classification System (CCS):
A.1 [Introductory and Survey],
I.7.m [Document and text processing]

Avainsanat — Nyckelord — Keywords
transaction, subtransaction, workflow,process,distributed,abort,commit,compensation

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Applications running on workflow managements require advanced transaction management that goes beyond what traditional relational database management systems provide. On one hand these applications require relaxed notion of transactionality to allow cooperativeness between workflow tasks, on the other hand stricter transactional notions are required to model business transactions that usually involve complex process structures but require ACID properties such as atomicity and isolation.

Lately there has been a trend to distribute workflow executions, which requires an even more advanced transaction support system that can handle this distribution. Furthermore, organizations outsource supporting processes to other organizations and concentrate mostly on their core business processes. Organizations form a virtual enterprise where provider organizations offer e-services to consumer organizations.

There have been a large number of transaction models to cater for long running transactions. Examples include nested transactions and the Saga model where long lived transactions are used to handle transactions that require a substantial amount of time[GMS87].

In the WIDE project, a transaction management system is designed that offers a two-layer transaction model. The upper layer offers a global transaction for long running process. This layer extends Saga to provide flexible rollbacks. The lower layer offers strict transactional semantics i.e. isolation and atomicity as offered by traditional database management systems.

This paper presents WIDE and two extensions, the first discussing the extension to the WIDE global transaction layer i.e. providing support for the distributed execution of workflow processes, i.e. distributed global transactions, and the second discussing a three-level transaction model for cross-organizational workflow management called X-transaction model.

The paper is organized as follows, in the first chapter we review the WIDE workflow model and architecture, in section two,SAGAs of which WIDE global trnsaction is based on id briefly discussed, then the basic WIDE two-layer transaction management follows. Section 3 discusses detailed distributed global transaction and cross-organizational supports.

2 WIDE Workflow model and architecture

WIDE, workflow on intelligent distributed database environment is a distributed workflow management system that offers scalability by using a distributed object model, client/server architecture and a distributed workflow server architecture[CGS97]. The unique feature about WIDE is that it is implemented on top of a commercial database management system offering advanced transaction management and active rule support.

The design of the architecture is ruled by three major design decisions[CGP+96]:

- the database management functionality should be orthogonal to the workflow management functionality,
- the transaction support functionality should be orthogonal to the rule support functionality,
- both extended database functionality and workflow management functionality should be independent from the underlying database management system.

2.1 WIDE model

The model of WIDE has a distinguishing characteristic of separating the description of the organization from the workflow process specification. Workflows are activities involving the coordinated execution of multiple tasks performed by different processing entities to achieve a common business goal.In business related workflows, a task defines some work to be done by a person, by a software system or by both of them.

Specification of a workflow (WF) involves describing those aspects of its component tasks (and the processing entities that execute them) that are relevant to control and coordinate their execution, as well as the relations between the tasks themselves.

We define workflow schema as a coordinated set of Work Tasks (or activities) which are connected in order to achieve a common goal according to a predefined sequence of execution. Each workflow schema can be instantiated in several instances of the workflow schema, called cases.

The WIDE model (Figure 1) maps two models which allow the specification of which agents may perform given roles

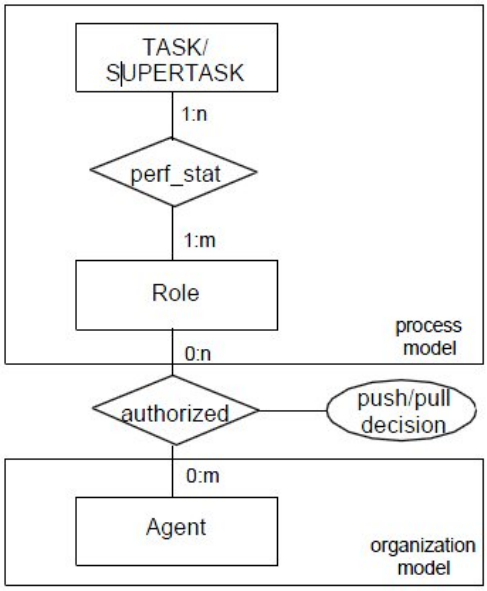

Figure 1: Wide models [CGS97]

2.1.1 Organizational model

In the organizational model, the organization is defined in terms of different types of agents, including individuals, specifying the (possibly complex) structure of the organization. The following enity types are defined

- an actor that can be an individual or an automatic entity
- a group that allows actors to be grouped based on their organizational properties
- organization function that defines a function performed by group(s) or actors.

2.1.2 Workflow process model

The workflow process model consists of the activities that must be performed and the order in which those activities must be performed. The order between the activities is specified using different kinds of control connectors and the entire ordering of

activities is called the control flow of the workflow process model. With the control connectors, it is possible to specify that the execution of activities must be done in sequence, or in parallel, or that a choice has to be made between activities (alternative activities), or that certain activities need to be executed more than once (iterative activities)

Elements of the workflow schema are defined in the process model. As can be seen from Figure 1, a role is defined to have a relationship with task or supertask. Each task can be associated to one or more roles.

The authorized relationship permits to associate agents modeled in the organization to one or more roles modeled in the process.

Tasks are atomic units of work, i.e., their effects have an all-or-nothing character with respect to resources under control of the workflow management system, depending on success or failure of task execution. Failed task executions are automatically undone by the workflow management system[CGP+96].

Connections describe interactions among tasks, i.e., the process flow; in WIDE, the goal is to provide advanced features for definition of semantically rich connections between tasks. Connectors may be connected to tasks or to other connectors, for a greater flexibility in specifying workflow schemas.

Supertasks(STs) have features of both workflows and tasks. Like WFs, they are internally decomposed into tasks (and possibly other STs); each ST has one start symbol and several stop symbols. Supertasks can be in turn hierarchically decomposed into supertasks. The action part is not defined for supertasks, as the job the ST performs is in effect decomposed into smaller jobs performed by the component tasks.

Start and Stop Symbols

Start and stop symbols enable the creation and completion of WF instances (cases). Each WF schema has one start symbol and several stop symbols; the start symbol has one successor symbol and each stop symbol has one predecessor symbol.

Split

Splits can be categorized as follows

- AND-split: after the predecessor ends, all successors are ready for execution.
- OR-split: each successor is associated with a condition; after the predecessor ends, conditions are instantaneously evaluated and only successor tasks with

a true condition are ready for execution.

- Conditional with mutual exclusion: only one condition can be true; thus, after the predecessor ends, if no condition or more than one conditions are true, an exception is risen; otherwise, one of the successors is ready for execution.

Join

AND-join: the successor becomes ready only after the end of all predecessors. k-AND-join: the join is associated with a value k; the successor becomes ready after the end of k predecessor tasks with the same activation number. Cycle (also called OR-join): the successor becomes ready every time a predecessor task ends

2.2 WIDE architecture

The WIDE architecture is shown in (Figure 2). The model is divided into layers. The lowest uses a relational database management system. Next to the RDBMS is the basic access layer(BAL) that shield the upper layers from the DBMS by providing a mapping of object oriented database access of its clients to relational interface. This mapping is provided by a translator.

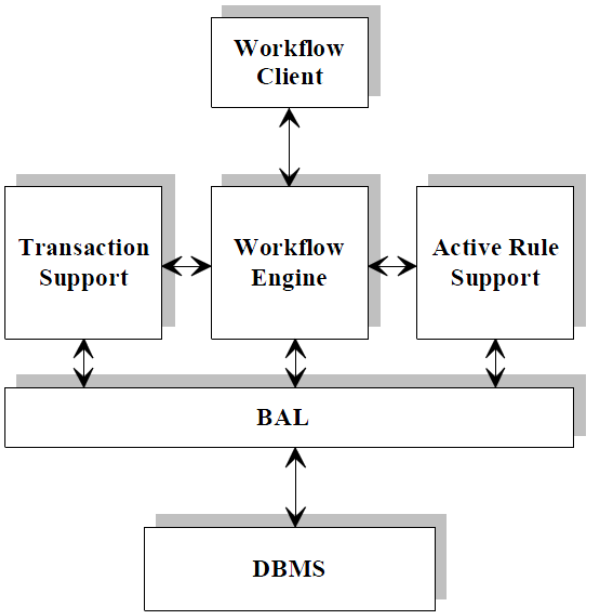

Figure 2: Wide architecture [CGS97]

Next to the basic access layer is the server layer. The server layer provides transaction and active rule support as well as a workflow engine that provides the heart of the workflow management system. It uses the BAL for database access, the transaction support module to provide advanced transactional contexts for its operation, and the active rule support module to handle reactive workflow behavior. Finally the topmost layer is the workflow client module which provides the users of the workflow management system an interactive interface. Note that the workflow client communicates directly with the workflow engine in the server layer.

2.3 Transaction management in WIDE

Workflow processes are hierarchical by nature i.e. they can be broken down to subprocesses up to individual tasks which are the basic units. The transaction model in WIDE is a two-layer model. This model is the focus of this paper. It is described next.

3 WIDE transaction model

3.1 SAGAs

Transaction support is a necessary functionality that is required by workflow management systems. These systems run long running processes that require adavcned trnsaction semantics beyond traditional ACID transactions. Thus different transaction model needs to be designed to support these processes.

A Long lived transaction(LLT) is a transaction whose execution, even without interference from other transactions, takes a substantial amount of time, possibly on the order of hours or days[GMS87]. An LLT has a comparatively long duration compared to most other transactions. The reason being that it accesses many database objects, has a lengthy computation or pauses for inputs from users. A consequence of this is the fact that LLTs presents serious performance issues.

In normal transaction execution context, when a transaction is being executed the system locks the objects used by this system and only releases these objects after the transaction has committed.

For an LLT because of the duration it takes to execute, other transactions that need to use these objects suffer a locking delay.

It is apparent that there is no solution that eliminates the issues caused by LLT, even if a mechanism is used to ensure the atomicity of LLTs, the long delays will still remain. However, for specific applications it may be possible to alleviate the problems by relaxing the requirement that an LLT be executed as an atomic action In other words, without sacrificing the consistency of the database, it may be possible for certain LLTs to release their resources before they complete, thus permitting other waiting transactions to proceed.

A LLT is a saga if it can be written as a sequence of transactions that can be interleaved with other transactions[GMS87]. The database management system guarantees that either all the transactions in a saga are successfully completed or compensating transactions are run to amend a partial execution[GMS87].

3.2 The WIDE transaction model

We now have all the tools with which to define our transaction model. As was described in section 1, in the WIDE project an extended workflow model and language are developed with advanced process primitives like multitasks, various join operators, exceptions etc.

In the process hierarchy, the top level is formed by a complete workflow process and the bottom consists of individual tasks which are the basic atomic units of work i.e. these tasks are not further divided. The separation between the higher and lower levels is formed by the notion of business transaction in the workflow application. Process levels representing business transactions and their subprocesses have strict transactional semantics; their superprocesses have relaxed semantics[GVBA97].

An example business process is shown in (Figure 3). This example is an online computer store where customers can order computers by choosing different parts that suites their requirements. The process starts when the customer goes online and configures the computer by selecting different components such as CPU speed, memory, hard disk size etc. Depending on the components chosen, the cost is computed. The customer might decide to cancel the order. If the order is made, the computer is assembled. Before packaging, the assembled computer is first tested and then delivered to the customer.

In the figure all the rectangles represent business transactions, the dotted rectangle finance represents supertask above the level of business transactions.

The WIDE transaction model is a two layer transaction model where the lower layer

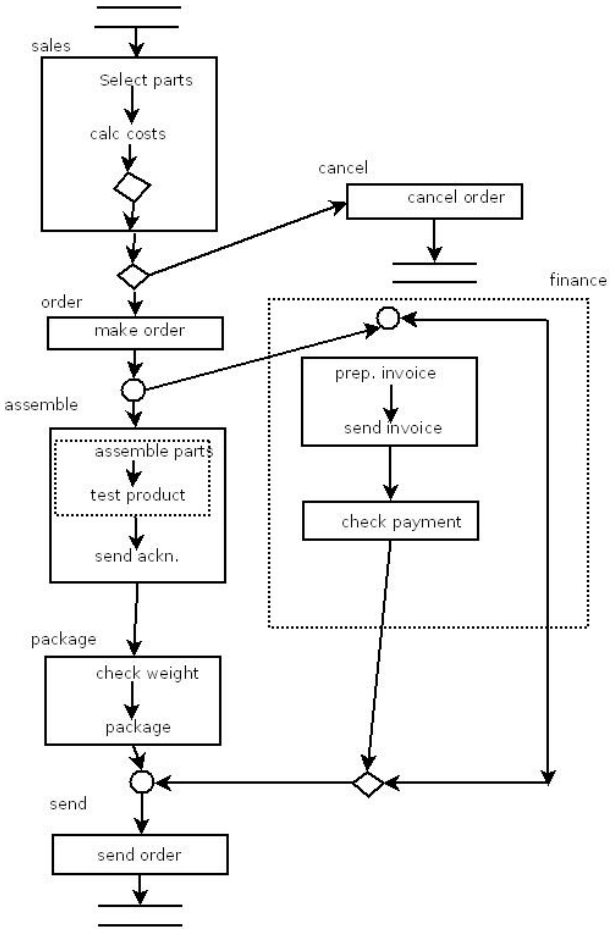

Figure 3: An example business process

is mapped onto the process levels associated with business transactions and below and the upper layer is mapped onto non-atomic supertasks.

The local transaction layer of the WIDE model is used to support business transaction semantics in workflow processing. A local transaction is composed hierarchically of subtransactions and basic actions. Whereas a subtransaction is a nested transaction of a parent(local transaction) and therefore can be made up of subtransactions and/or basic actions , a basic action cannot be broken down into transactions.

An example local transaction is shown in (Figure 4). In the figure local transaction 'assemble' consists of two subtransactions 'send ackn' and 'build product'. Whereas 'build product' is further decomposed of two subtransactions 'assemble parts' and 'test product', 'send ackn' is a basic action itself. Also in the figure subtransaction has been marked as a non-critical transaction which means that if it is aborted, it will not have any effect on the parent transaction 'assemble'. On the other hand sub-

transaction build product has been marked as a critical subtransaction i.e. a failure in building a product will have any effect on the business transaction 'assemble'.

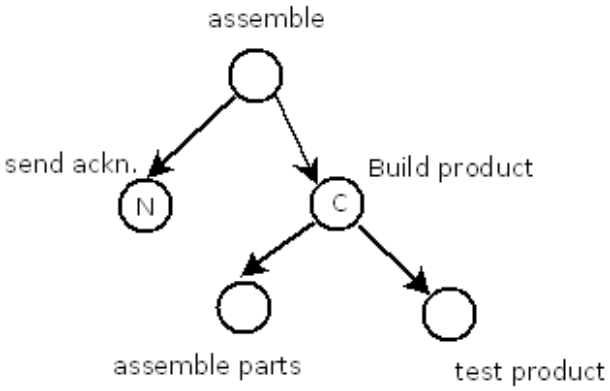

Figure 4: Local transaction

The upper layer of the two-layer transaction model consist of global transaction. The global transaction layer offers relaxed notions of atomicity and isolation to cater for the needs of workflow process above business transactions. A global transaction is a rooted directed graphed where each node is a local transaction. We say local transaction are steps in a global transaction graph[GVA01]. Furthermore local transactions are black-steps in this graph.

The graph can have only one starting point and an arbitrary number of ending points.

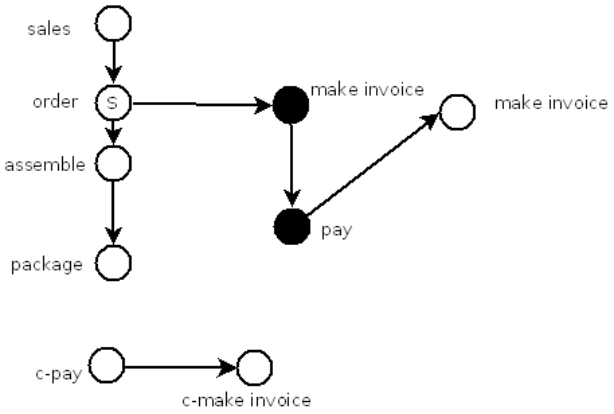

Figure 5: Specification and Execution graphs

The graph represents all possible execution orders of a workflow process. i.e. it consists of all the possible ways of accomplishing a business goal. We define a

specification graph as a graph as defined above. A execution graph however is an instance of a specification graph i.e. consider the global specification graph obtained from figure as shown in figure 5 a, the execution graphs figure 5 b,c are two instances of this specification. The first one is obtained after payments has been bounced back between 'pay' and 'make invoic'e until payment has been accepted and the computer finally sent to the customer. The second execution graph occurs after the customer has cancelled the order he made.

Let's consider how a rollback is implemented. Figure 6 shows a partial execution order of the execution graph of figure 5. In this figure the black circles have committed and step 'order' has been marked as a savepoint. Suppose running transaction 'make invoic'e causes an error. Because 'order' has been marked as a savepoint, a dynamic compensation transaction will be created as shown in figure 6.

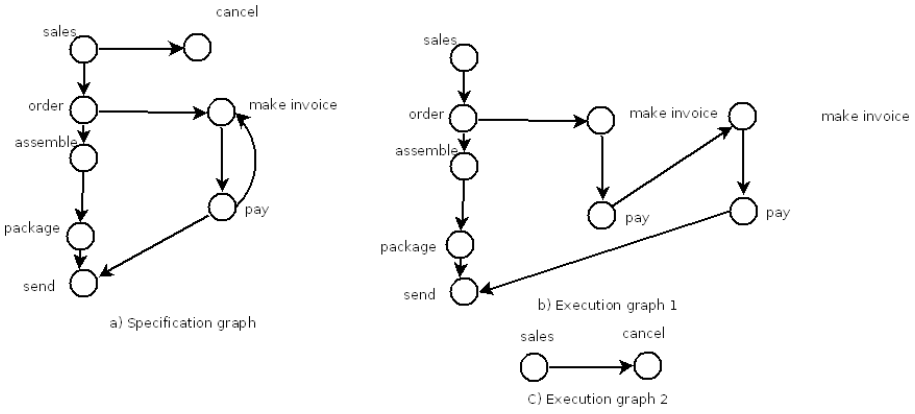

Figure 6: Partial execution graph with compensation

3.3 Transaction management architecture

The architecture for transaction management is partly based on the WIDE architecture discussed in the first section by looking into the transaction support component of the server layer.

Two orthogonal submodules are similarly used for extended trnsaction support. The first submodule is used to support global transactions and second module local transactions.

Local transaction support is provided in two layers(7). The top layer contains the local transaction manager(LTM) that is responsible for manipulating local transaction objects. The bottom layer contains the local transaction interface(LTI) which

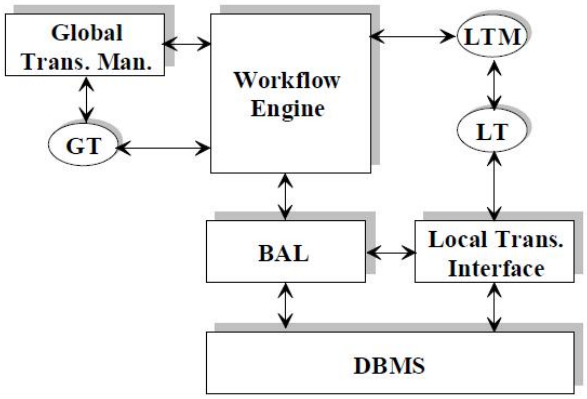

Figure 7: Transaction management architecture[CGS97]

is responsible for mapping logical transaction operations to physical transaction operations[CGS97].

The upper layer consists of the local transaction manager (LTM), which manipulates local transaction (LT) objects. The upper layer is independent. The LTI is considered part of the BAL since it addresses the databas directly.

Global transaction support is discussed in the next section.

4 Distributed and cross-organizational support

4.1 Distributed transaction model

The previous section outlined the basic transaction model for the WIDE distributed workflow management system. It consisted of a two-layer transaction model where the upper layer was made up of global transactions and the lower layer was made up of local transactions.

A distributed transaction model provides support for workflow management applications that have workflow management systems distributed in possibly different geographical locations.

Consider an organization that has it's workflow management systems distributed

in different business units. Each business unit has its own workflow management system. A business unit can delegate a subprocess to another business unit. We thus have a delegation model where different WFMS can collectively accomplished a business goal. (Figure 8)] shows a distributed WFMS with global transaction support. In the figure there are 4 business units BS 1 ...4. BS 1 has delegated subprocess to BS 2 and BS 3. BS 3 has delegated a subprocess to BS 4. Each WFMS can use a non-distributed transaction support(GTS)[VGBA99]. Only the distribution aspect requires an extension to the global transaction support system to handle the delegation of subprocesses.

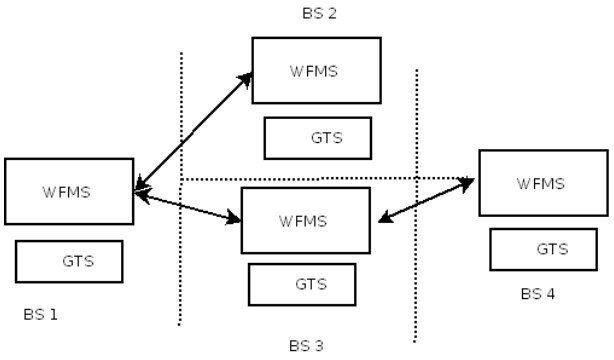

Figure 8: Distributed WFMSs and global transaction [VGBA99]

We now discuss global transaction support for a distributed workflow process.

As described before, the upper layer of the transaction model developed in the WIDE project relies on compensating actions to semantically undo the effects of completed global transaction steps in case of failures. Compensating an entire workflow, thereby undoing all the work that has been done so far is called a complete abort. A complete abort is usually too strict and undoing only part of the process would suffice[VGBA99]. For this purpose the global savepoint concept is introduced to allow for flexible, partial aborts.

In a distributed WFMS, each business unit acts as a workflow management system and can delegate a subprocess to another business unit. Transaction support for this unit can use the non-distributed transaction scenario model described in section 2.

In the distributed architecture however an additional communication protocol is required that decides which other sites need to abort as well and in which abort mode(partial or complete abort). It then signals those other sites to actually start an abort in the correct abort mode[VGBA99].

Compensating an entire workflow, thereby undoing all the work that has been done so far is called a complete abort. A complete sabort is usually too strict and undoing only part of the process would suffice. For this purpose the global savepoint concept is introduced to allow for flexible, partial aborts.

During the dynamic execution, any abort request should specify which abort mode to use. To illustrate the differences between forward and backward aborts, consider the delegation of subprocesses and transaction state of a four-site WFMS as shown in Figure 9.

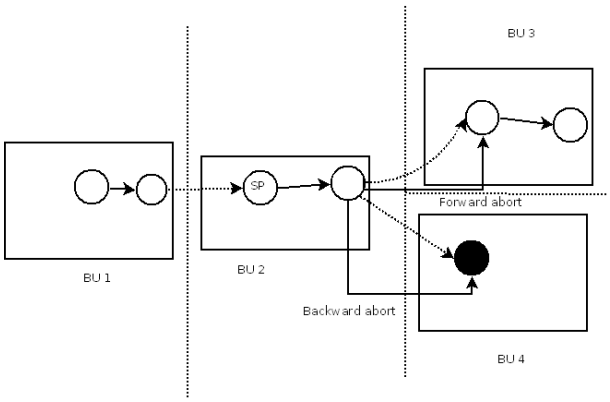

Figure 9: Forward and Backward abort. Adapted from [VGBA99]

Suppose the black circle step fails in BU 4, because no step in BU 4 has been marked as savepoint, the transaction system in BU 4 cannot roll back to any step in the subprocess. The transaction system in BU 4 will issue a backward abort to parent site(BU 2) to handle. In BU 2 the first step has been marked as a savepoint. The transacton needs to be roll back to the parent site step marked as a savepoint. Note that BU 2 has delegated a subprocess to BU 3.

BU 2 cannot do rolling back action since it has delegated a subprocess to BU 3 it will thus issue a forward abort to the transaction system in the child site BU 3. A forward abort implies that the site receiving the forward abort request must abort in complete abort mode as all the actions done at that site are dependent on an action that has been rolled back and are therefore invalid.

4.2 Global Transaction Support architecture

The non-distributed Global Transaction Support (GTS) consists of two main parts, i.e. the Global Transaction Manager (GTM) and the Global Transaction (GT)

objects. The GTM is responsible for the handling of compensation requests and the construction of compensating global transactions. The GT objects perform the administration of running global transactions. Each global transaction is managed by a GT object that is dynamically created at the start of the global transaction. The GT object is signaled by the workflow engine about events that may change the state of a global transaction, e.g. start and end of a global transaction step.

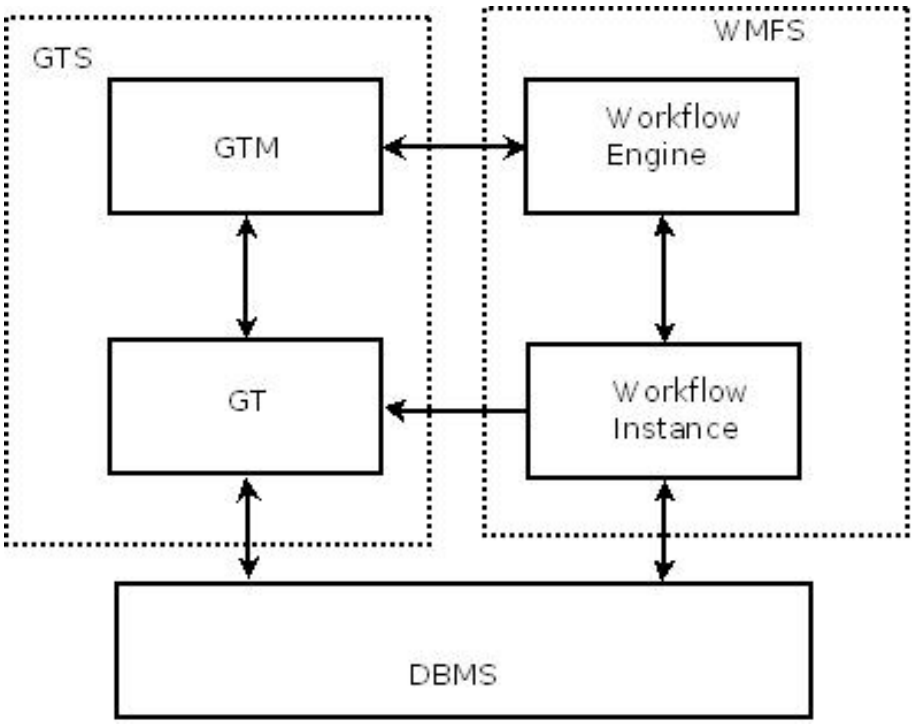

Figure 10: GTS - WFMS architecture[VGBA99]

The main task of the GTM is to construct compensating global transactions GTM is to construct compensating global transactions as discussed in Section 5.2. The GTM is activated by the workflow engine when the engine raises a global rollback event. It uses the appropriate GT object to obtain information on the current status of the global transaction, most notably the current execution graph and the compensating counterparts of executed steps. After the compensating global transaction has been constructed, it is passed to the GT object, which makes the information persistent.

4.3 X transaction model

Lately, organizations concentrate more and more on their core business processes while outsourcing supporting processes to other organizations, thereby forming vir-

tual enterprises. For this reason, it is important that workflow management systems ensure that these primary business processes are executed in a reliable and consistent manner. This can be accomplished by incorporating transaction semantics in the processes.

As would be seen later, the global transaction model taken as a basis for the work presented here, is based on the global transaction layer of the WIDE transaction model.

4.3.1 Virtual Enterprises

When an organization wants another organization to perform part of its process on its behalf, the organizations first need to be brought together so that they can form a virtual enterprise. Organizations find each other in an electronic marketplace where organizations put their services on offer, or search for services that are offered in the marketplace (business-to-business or B2B e-commerce)[VG03]. Using a match-making facility or trader, compatible organizations form a virtual enterprise, the cooperation in which is specified in an electronic contract.

An electronic contract specifies an e-service. An e-service is the electronic equivalent to a regular service offered by some organization.

An individual organization operating alone uses an intra-organizational process model. To apply workflow management, the business processes of an organization must be modeled in workflow process models.

When organizations form virtual enterprises, the business processes of the separate involved organizations need to be interconnected. The intra-organizational model described above cannot be used to model this new cross connection. A new model, cross organizational model is used. In this model, two organizations are involved in the execution of the cross-organizational workflow process. The organization that provides the service is called the service provider and the organization that consumes the service is called the service consumer.

The service provider organization executes a workflow process on behalf of the service consumer.

The service provider doesn't want to reveal its internal process to the consumer. To hide its internal process, it only presents an abstract view of is process. This is needed because the consumer organization does not want to know the specific details of the workflow process but an abstract view of it.

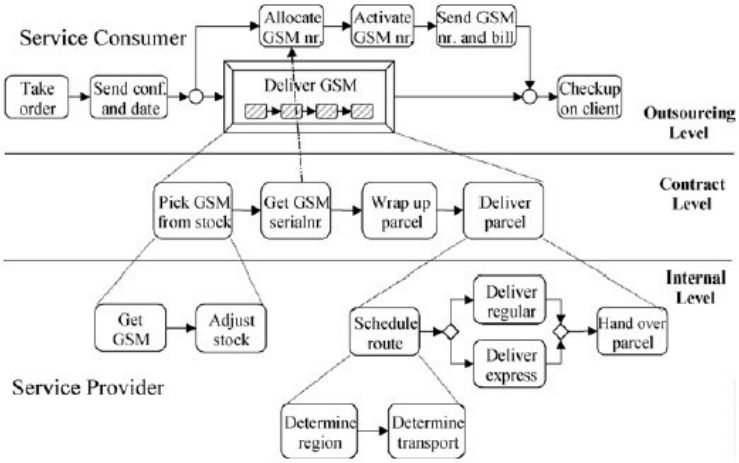

Figure 11: An example virtual enterprise[VG03]

Thus the two organizations create a view that is common to both. This common view which is included in the electronic contract is called the contract level process. The encapsulated process, i.e. the detailed process as it will actually be executed by the provider organization is called the internal level process and is thus not visible to the consumer organization. The process that is executed by the consumer organization is called the outsourcing level process[VG03]. Thus a cross-organizational process model consists of three levels as described above.An example cross-organizational process is shown in Figure 11.

In this example a phone company takes orders from customers. The 'deliver GSM' process has been outsourced to a logistics company. The logistics company has abstracted the process of delivering the GSM in four processes in the contract level as can be seen from the figure. The internal processes of the service provider actually performing the contract level processes are invisible to the service consumer.

4.3.2 Transaction support

Transaction model for the cross-organizational workflow processes consist of three transaction levels. Before we define these first we give some preliminary overview of transaction support that the cross organizational model inherits from the WIDE transaction model.

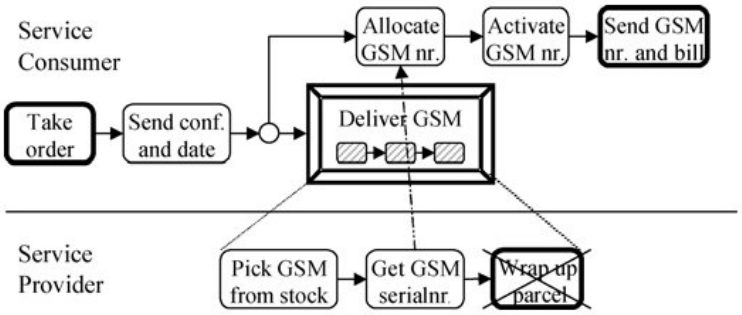

Figure 12: An example rollback [VG03]

We define an I-step(intra-organizational step) as an atomic piece of work that adheres to the ACID transaction properties. These steps are short running process steps that are part of long-running workflow processes of intra-organizational workflow processes. It can be noted that each I-step has a specified compensation counterpart that undoes the effect of the I-step on case of a failure in the execution of the I-step[VG03].

On the other hand the contract level process is divided into smaller steps that each commit their results when the step finishes and are compensated in case they need to be undone. Because these smaller steps relate to cross-organizational workflow processes these steps are called X-steps and because the contract level activities encapsulate the internal level activities, an X-step corresponds to one or more I-steps[VG03]. So, committing the result of an X-step is in fact done through the underlying I-steps, and an X-step thus has relaxed atomicity characteristics. An X-step is not executed in isolation because the intermediary results produced by the underlying I-steps are committed even though the X-step itself is not yet completed.

In the same way to undo an X-step in case of a failure, each step should have a corresponding compensation counterpart specified for it. All X-steps are specified in the electronic contract.

The X-transaction model, combines the three transactional levels previously, consisting of the I-steps and X-steps, into one transaction model, i.e. a three-level transactional workflow process model. The X-transaction model offers the required

loose transaction properties for the intra- as well as cross-organizational workflow processes. An X-transaction consists of all X-steps and I-steps of the cross-organizational workflow process, together with the corresponding compensating steps.

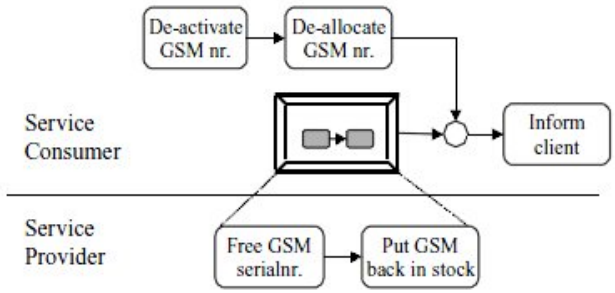

Figure 13: Compensating activities of rollback[VG03]

The X-transaction model offers a flexible rollback mechanism that allows rollbacks to take place at any of the three different transactional levels:

- Outsourcing level. A rollback on the outsourcing level is performed entirely by the consumer organization. If the outsourced process needs to be compensated, the compensating activity corresponding to the placeholder is executed.
- Contract level. A rollback on the contract level will involve only the X-steps of the cross-organizational process by executing the compensating activities that correspond
- Internal level. A rollback will only involve I-steps that are internal to the provider process and are thus not visible to the consumer.

Let's consider an example of how the X-trnsaction model can be used to ensure reliability when there is a failure in the execution of a process. Consider the partial execution(See (Figure 12). The crossed step indicates that there was a problem in the execution of that step. Because this step is in the service provider's end, it will start to rollback to a consistent state. Furthermore since the GSM number is linked

to the GSM serial number, it is stated in the contract that the rollback scope is cross organizational and that the provider should rollback in complete rollback and the consumer should rollback in partial abort mode.

The resulting compensation action is shown in Figure 13. The consumer organization will concurrently 'de-activate the GSM nr', 'de-allocate GSM nr' and finally 'inform the client' which are the compensating activites for 'activate GSM nr', 'allocate GSM nr' and 'Send conf. and date' respectively.

5 Conclusion

Thw WIDE transaction model is an advanced tranction model that solves workflow specific transaction requirements that traditional database systems are unable to solve. It was seen that by using the concept of a business transaction, tasks were divided into non-atomic supertasks that allowed loose transaction semantics and atomic supertasks that had strict transaction semantics. This transaction model and architecture is orthogonal to the workflow management system thus it is easily applicable to areas that deal with long running processes. The result is a distributed workflow management system offering distributed transaction support.

However, these approaches are only applicable to intra-organizational workflow processes, as cross-organizational aspects are not considered.

In the X-transaction model, a static intra-organizational infrastructure consists of a layer that incorporates the local WfMSs and a layer that is formed when two organizations come together to form a virtual enterprise. The prototype was been built in the CrossFlow project and tested using real-world scenarios[GAHL00].

References

CGP+96 Casati, F., Grefen, P., Pernici, B., Pozzi, G., Sánchez, G. and Pernici, B., Wide workflow model and architecture, 1996.

CGS97 Ceri, S., Grefen, P. and Sanchez, G., Wide-a distributed architecture for workflow management. *Research Issues in Data Engineering, International Workshop on*, 0, page 76.

GAHL00 Grefen, P., Aberer, K., Hoffer, Y. and Ludwig, H., Crossflow: Cross-

organizational workflow management in dynamic virtual enterprises, 2000.

GMS87 Garcia-Molina, H. and Salem, K., Sagas. *SIGMOD Rec.*, 16,3(1987), pages 249–259.

GPS99 Grefen, P., Pernici, B. and Sanchez, G., editors, *Database Support for Workflow Management: The Wide Project*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

GVA01 Grefen, P., Vonk, J. and Apers, P., Global transaction support for workflow management systems: from formal specification to practical implementation. *The VLDB Journal*, 10,4(2001), pages 316–333.

GVBA97 Grefen, P. W. P. J., Vonk, J., Boertjes, E. and Apers, P. M. G., Two-layer transaction management for workflow management applications. *DEXA '97: Proceedings of the 8th International Conference on Database and Expert Systems Applications*, London, UK, 1997, Springer-Verlag, pages 430–439.

GVBA99 Grefen, P., Vonk, J., Boertjes, E. and Apers, P. M., Semantics and architecture of global transaction support in workflow environments. *IFCIS International Conference on Cooperative Information Systems. CoopIS*. IEEE, 1999, pages 348–359, URL `http://doc.utwente.nl/19096/`.

VG03 Vonk, J. and Grefen, P., Cross-organizational transaction support for e-services in virtual enterprises. *Distrib. Parallel Databases*, 14,2(2003), pages 137–172.

VGBA99 Vonk, J., Grefen, P. W. P. J., Boertjes, E. and Apers, P. M. G., Distributed global transaction support for workflow management applications. *DEXA '99: Proceedings of the 10th International Conference on Database and Expert Systems Applications*, London, UK, 1999, Springer-Verlag, pages 942–951.