TRANSACTIONAL WORKFLOW FOR TELECOMMUNICATION SERVICE MANAGEMENT

QINZHENG KONG AND GRAHAM CHEN

***Abstract*—As workflow management systems become more complex and critical to an organisation's performance, there is an opportunity for a new generation of workflow management products. These products could greatly increase the scope of workflow management systems in target application areas. One of the key extensions to workflow management systems will be the use of transactional semantics to increase reliability.**

Traditional transaction processing technology can not be used directly for workflow management since applications in workflow management are different from data-oriented applications, such as funds transfer. Customised workflow definition and workflow management capability is required to capture business requirements and rules.

This paper discusses the requirements and characteristics of transactional workflow management, addresses architectural issues of transactional workflow and its role in the telecommunication service management platform, and examines applications of transactional workflow for telecommunication service management.

***Source of Publication*—An early version of this manuscript was published in a poster session in the IEEE/ IFIP Network Operations and Management Symposium 1996.**

1 INTRODUCTION

1.1 WORKFLOW

A *workflow* is a concept closely related to re-engineering and automating business and information processes in an organisation. It can be defined as a collection of tasks organised to accomplish such processes.

A *task* can be performed by one or more software systems, one or more humans, or a combination of these. Human tasks include those with or without direct interaction with computer systems. Examples of tasks include updating a document or a database, generating a report, contacting a customer, and laying a cable (in telecommunication service provisioning applications).

In addition to a collection of tasks, a workflow defines the task invocation order and conditions, task synchronization, and information flow.

1.2 WORKFLOW MANAGEMENT

A *workflow management system* is defined by the Workflow Management Coalition (WfMC) as "a system that completely defines, manages and executes workflow processes through the execution of software whose order of execution is driven by a computer representation of the workflow process logic".

Workflow management is a technology supporting the re-engineering of business and information processes. It involves:

- defining workflow, in other words, describing those aspects of a process that are relevant to controlling and coordinating the execution of its tasks
- managing workflow, for example, starting tasks according to the order and condition specified in a workflow, keeping track of all the tasks, and keeping a record of workflow processes
- executing automatic tasks when required
- providing for fast design and implementation of the processes as business needs and information systems change.

2 REQUIREMENTS FOR TRANSACTIONAL WORKFLOW MANAGEMENT

2.1 EXAMPLE OF A TRANSACTIONAL WORKFLOW

Figure 1 below depicts a workflow representing a travel request process for attending an international conference.

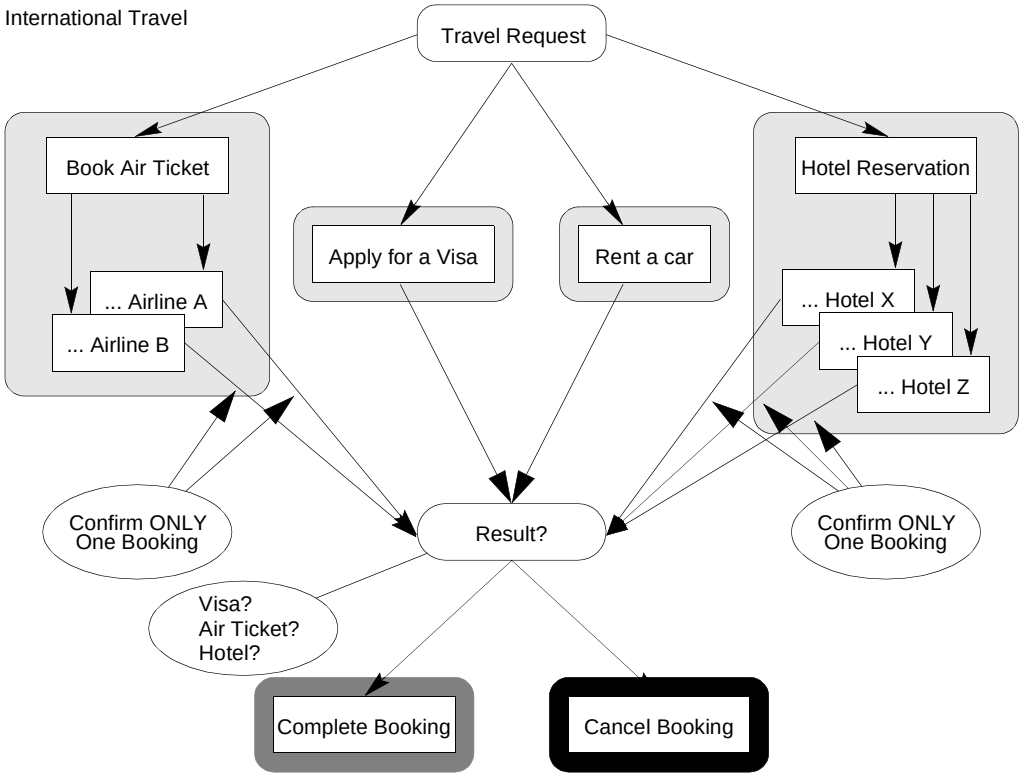

Figure 1: An Example Workflow

The workflow consists of the following tasks:

- Apply for a visa—If a visa is required, then obtain a visa before departure.
- Book air tickets—Air tickets must be booked. Sometimes, air ticket booking can be done with more than one airline at the same time. When one booking is confirmed, all the others need to be cancelled. If no seat is available in any airline, then the trip has to be cancelled.
- Book hotel room—As for air ticket booking, parallel bookings can be done with a number of hotels and when one booking is confirmed, all the others need to be cancelled. If all the selected hotels are fully booked, then the trip has to be cancelled.
- Rent a car—This is optional. If car rental cannot be arranged before departure, it may be arranged on arrival at the destination airport, or other transport may be used.
- Collect all the results and make a decision—If the trip has to be cancelled (e.g. no seat is available in any airline), all the confirmed bookings have to be cancelled. (No action is needed for the visa).

This process requires some transactional workflow features, such as:

- ability to specify complex task dependencies, such as “sequential”, “parallel”, “alternative” and “conditional” dependencies. See Section 2.2.
- customised specification of transactional tasks and their rollback behaviours
- “compensation” (see Section 3.3) of certain tasks

2.2 CHARACTERISTICS OF TRANSACTIONAL WORKFLOW

Transactional workflow has the following characteristics:

- It brings reliability, correctness and consistency into workflow management.
- It allows complex task dependencies to be specified. For example:

  - *sequential* tasks—One task starts after the successful completion of another task.
  - *parallel* tasks—A number of tasks can be started at the same time and performed concurrently. A follow-up task is started only after the successful completion of all the parallel tasks.
  - *alternative* tasks—A number of tasks are started at the same time and performed concurrently. Only one of the tasks is allowed to complete. All other tasks must roll back to a predefined stage. Preference can be specified among the tasks.
  - *conditional* tasks—The task started next depends on the previous task result.
- It involves some long-lived transactions, which means that resources cannot be locked for the whole life time of such transactions. The traditional ACID (atomicity, consistency, isolation and durability) features of transaction processing need to be extended to include:
  - *compensational transactions*—which will be executed when a rollback is required for a committed transaction. In a normal TP environment, a transaction cannot be rolled back after it has been committed. However, in many cases, a "compensation" can be defined and applied to a committed transaction to reverse the effect of the transaction.
  - *customised transaction specifications*—which allows users to define their own transactions, and the rollback and compensation behaviours of transactions

2.3 BUSINESS REQUIREMENTS FOR TRANSACTIONAL WORKFLOW

Business requirements for transactional workflow fall into the following categories:

- *enterprise requirements*—Coordination of activities at an enterprise level requires support for defining and implementing task dependencies, business rules, and temporal information across different business domains.
- *distribution requirements*—The move to distributed systems requires that workflow management systems be integrated into distributed, heterogeneous environments.
- *reliability requirements*—When workflow management systems are applied to more critical areas of business operations, the ability to guarantee the integrity and reliability of the processes and data becomes significantly more crucial to the success of such systems.
- *security requirements*—As organisations implement workflow management systems for managing their financial and other sensitive data, the security of the data becomes of critical importance. This is particularly the case where the systems operate over distributed locations and possibly different organisations.
- *flexibility requirements*—The move by organisations to use computer systems for strategic advantage rather than just cost control imposes strong requirements on the flexibility of those systems. The basic requirements of these systems change as organisations regularly "re-invent" the form and distribution of their products and services.

2.4 TECHNOLOGY REQUIREMENTS FOR TRANSACTIONAL WORKFLOW

The introduction of transaction functionality in workflow applications requires some extra support in addition to the existing functionality offered by standard transaction technology. In particular, the following features are required:

- *nested transactions*—to support multi-level transactions that allow committed tasks to be rolled back if required
- *rollback behaviour definition*—to allow customised definition of rollback semantics
- *compensation actions*—to support multi-level transactions that allow committed tasks to be rolled back if required—an alternative mechanism to nested transactions
- *specification of semantic dependencies*—to allow extra semantic constraints to be specified and enforced for a workflow
- *object orientation*—to utilise the functionality provided by distributed object technology and in particular the transaction processing support for workflow systems.

Section 3 discusses these features in detail.

3 FEATURES OF TRANSACTIONAL WORKFLOW

3.1 NESTED TRANSACTIONS

Most existing transaction processing technologies, such as those defined by ISO and X/Open, are based on a flat transaction model with ACID property, which does not support nested transactions.

Nested transactions contain subtransactions. The concept of *subtransactions* is gaining more acceptance and is particularly useful for workflow applications. It supports commitment of a subtransaction within a transaction and allows the subtransaction to be rolled back later.

Nested transactions are useful for the following purposes:

- *failure containment*—In the flat ACID transaction model, failure in any of a global transaction leads to the rollback of the whole transaction. This is very inflexible behaviour, and very inefficient for workflow applications. In many cases, failure in a task should be contained locally, that is, within the smallest subtransaction where the failure occurs. This prevents the whole workflow being rolled back. The workflow manager should be allowed to re-initiate the same task at a different time (or with different parameters) or to initiate an alternative task.
- *alternative subtransactions*—This feature relates to failure containment. It is useful when the workflow manager wishes to start an alternative task after a task performing a similar function has failed; or when the workflow manager wishes to initiate a set of competing tasks in order to achieve a commitment according to certain semantic criteria.
- *parallelism*—This feature allows the workflow manager to explore maximum parallelism by initiating multiple tasks in parallel as subtransactions.

Figure 2 shows the features of nested transactions.

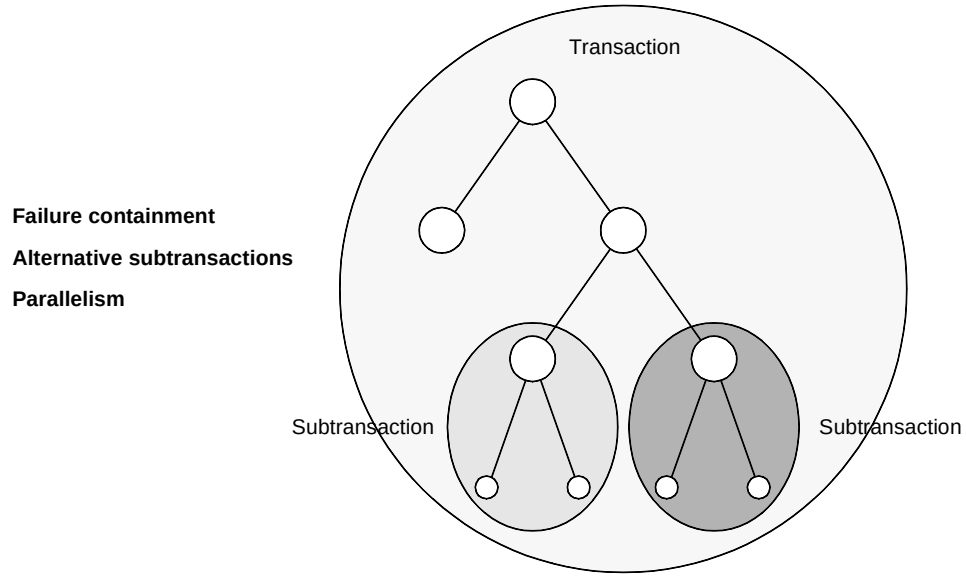

Figure 2: Features of Nested Transactions

3.2 ROLLBACK BEHAVIOUR DEFINITION

The existing OSI and X/Open transaction model is based on ACID property, that is, a transaction is defined as a set of tasks forming an *atomic* action. This *atomicity* of a transaction must be maintained at all times. The property of atomicity means that any data (bound data) operated upon by the actions within a transaction can only have two states: an *initial* state at the commencement of the transaction, before any operation is performed; and a *final* state, when all required actions are successfully performed. No other state of data is ever allowed to exist to the outside world of a transaction.

The semantics of a transaction is defined according to these two states. A transaction is *committed* when the bound data produces a final state. A transaction is *rolled back* when the final state is not produced or is discarded: in this case the bound data must be restored to its initial state.

The property of atomicity is very desirable and relatively easy to achieve in database-oriented applications. The atomicity of a transaction is the guarantee of data integrity. However, in many applications involving non-database activities, such as human tasks, and control of processes and equipment, it is very difficult to follow this model. Examples that do not fit the model include:

- A provisioning task may involve workers digging a trench in order to lay a cable. When a rollback is required, it is very undesirable to restore to the initial state.
- An operation resulting in certain behaviour of a piece of equipment (e.g. printing a cheque) may be very hard to rollback to the initial state.

In both these cases, it is desirable to allow users to define rollback behaviour of objects and operations. The reasonable behaviour for rolling back the task of laying a cable may entail the work completing, and an update to the asset database to record that the customer premises cabling equipment exists, although the provisioning workflow may have been rolled back. In the second example, the operation to destroy the cheque may be regarded as an acceptable rollback definition.

3.3 COMPENSATION ACTIONS

In a different scenario from the nested transaction and the definition of rollback behaviour, compensation actions are important. Compared with rollback behaviour which deals with how to roll back a task, *compensation* deals with committed tasks. There are many tasks which do not support nested transaction semantics, that is, after they are committed, it is impossible to roll them back when the whole workflow needs to be rolled back. One option in this case is to start another task (subtransaction) whose actions compensate for the previous committed task. An example of compensating for an entry in a directory is to remove the record from the directory.

Compensation can be used as an alternative mechanism for supporting rollback of committed subtransactions within a global transaction. The ACID property cannot always be guaranteed in this case. Similar to rollback behaviour definition, the compensation needs to be specified as part of the task specification of the workflow.

Figure 3 shows a sequence of actions, and compensation when a confirmed action needs to be rolled back.

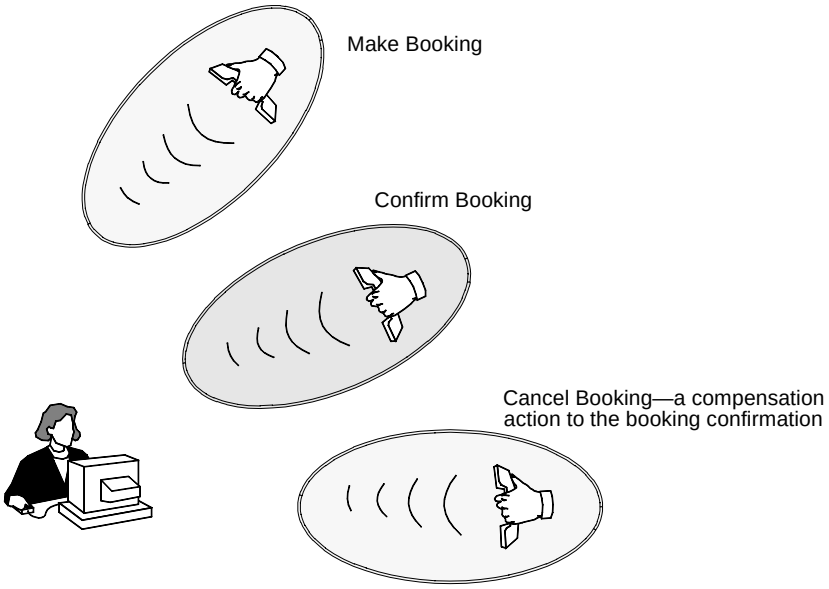

Figure 3: Compensation Actions

3.4 SPECIFICATION OF SEMANTIC DEPENDENCIES

There is no provision in the existing OSI and X/Open transaction model to define temporal ordering between different tasks within a transaction. Also, there is no mechanism to specify semantic dependencies between groups of operations in a transaction. Both features are important to workflow applications.

A workflow normally has a strong temporal logic associated with its tasks. Certain tasks cannot commence before some tasks have been committed or rolled back. An important part of the design of a workflow is to specify this temporal logic.

There are some semantic dependencies between workflow tasks, Some tasks must be performed in sequential order; some tasks are performed as alternative but competing tasks, that is, only one successful result is used; and some tasks are perform in parallel to achieve maximum performance.

All these features need to be integrated with the transaction semantics of a workflow and they can be best supported with the concept of subtransaction and a definition of rollback behaviour. The specification of these constraints form a major part of workflow design and the enforcement of these constraints form a major part of the workflow execution manager.

Figure 4 shows some dependencies among tasks.

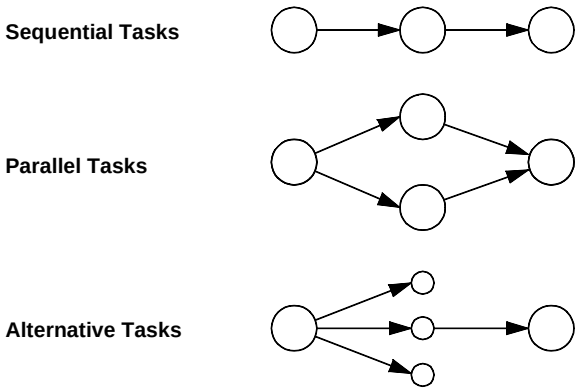

Figure 4: Task Dependencies

3.5 OBJECT ORIENTATION

Due to their distributed nature, workflow applications are best supported by using distributed object technology. In workflow applications, distributed object technology provides support for:

- heterogeneous communications in a networked environment
- transparent object definition and implementation of tasks
- transparent location and reconfiguration of tasks in a distributed environment
- migration to a new implementation technology

Distributed object technology provides encapsulation for underlying communication, location, implementation, and migration, and the separation of service access and service implementation. This helps workflow design concentrate on high-level semantics and it helps workflow execution utilise underlying support provided by the distributed infrastructure.

The transaction requirements for workflow can be well supported by distributed object technology. The object request broker (ORB) is an ideal mechanism for inter-task communication and propagation of transaction control and management over tasks and subtransactions. OMG CORBA, for instance, provides an environment in which components of workflow systems can be defined as building blocks offering specialised services. These components can be integrated with the environment using a well defined integration mechanism. The OMG has recently specified transaction services which allow transaction processing technology to be integrated with the CORBA environment to provide transaction support, including support for subtransactions.

Figure 5 shows the concept of object-oriented transaction services.

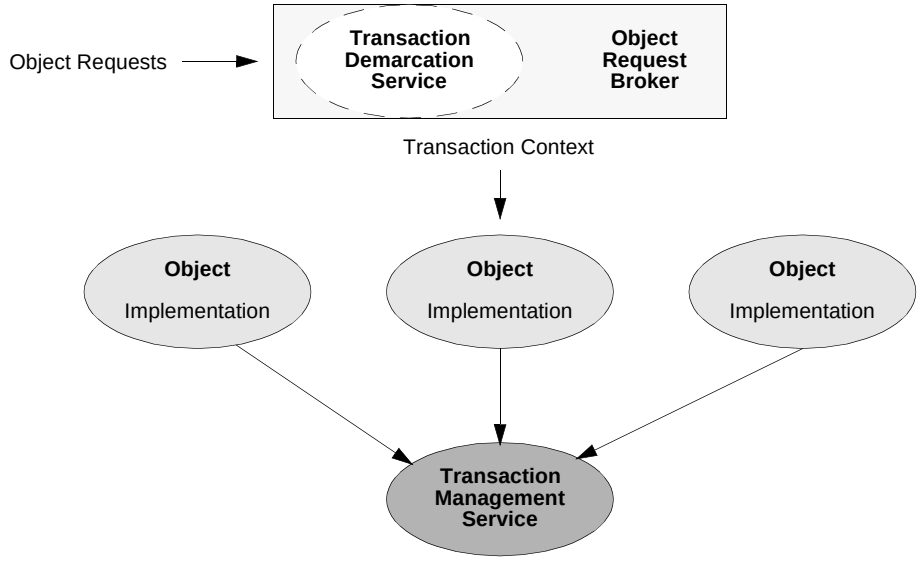

Figure 5: Object-Oriented Transaction Services

4 TRANSACTIONAL WORKFLOW MANAGEMENT ARCHITECTURE

Transactional workflow management can be built on top of available middleware technologies. These technologies provide the basic communication capability in a distributed environment, and support standard interfaces to applications and more than one underlying communication protocol. Some technologies also support interoperability between heterogeneous systems. Figure 6 shows a desirable architecture for transactional workflow management.

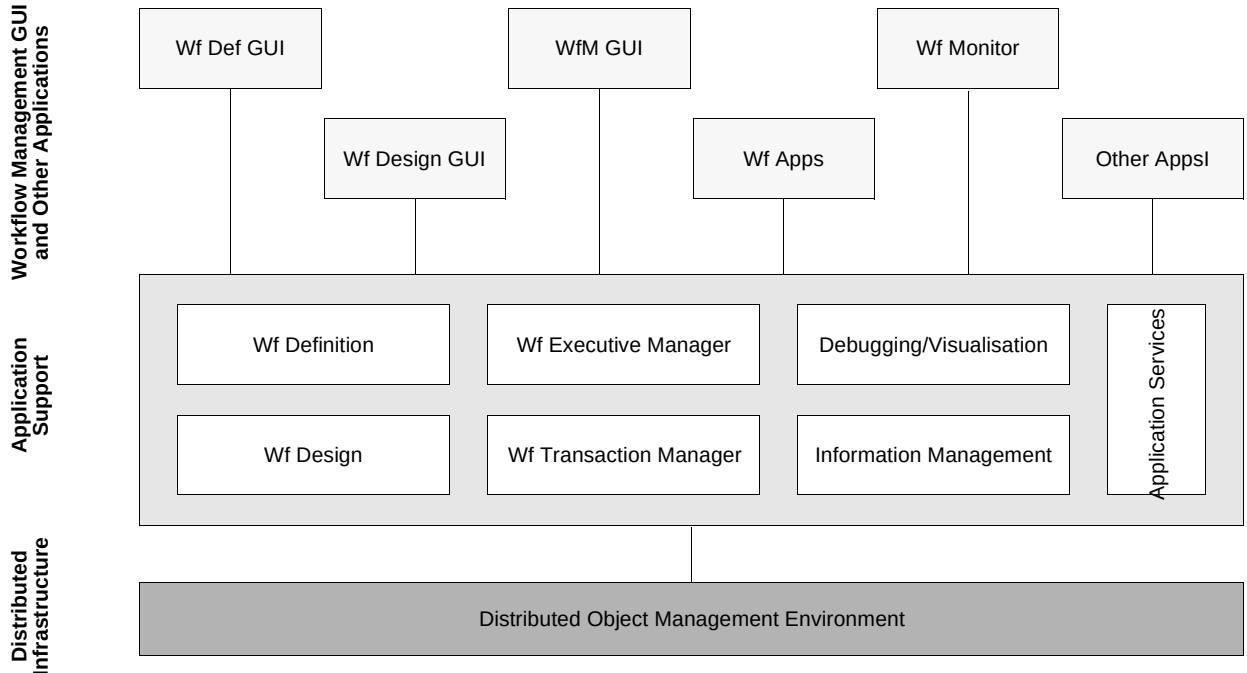

Figure 6: Transactional Workflow Management Architecture

A set of workflow management modules are the kernel part of the architecture. These modules include:

- A *workflow definition tool* that provides basic support for defining commitment, concurrency and rollback (CCR) semantics of workflow tasks, compensation operations for non-CCR tasks, and nested transactions.
- A *workflow design tool* for designing the workflow components and the temporal ordering and dependencies for each component. Using this tool, existing workflow specifications can also be used as components of larger workflow specifications.
- A *workflow execution manager* which coordinates the execution of instances of a specified workflow.
- A *workflow transaction manager* which focuses on the management issues of transactional tasks.
- A *debugging and visualisation facility* which allows users to view the status of the workflow in progress, to monitor and track workflow tasks (both local and remote), and to locate and correct problems in workflow specifications.
- A set of GUIs to provide a rapid application development environment and an easy user access environment.

5 TRANSACTIONAL WORKFLOW AND TMN SERVICE MANAGEMENT

In this section, we discuss how transaction workflow provides a useful technology framework for the tasks in the TMN management domain.

5.1 SCENARIO

Figure 7 depicts a scenario of service provisioning represented as a workflow.

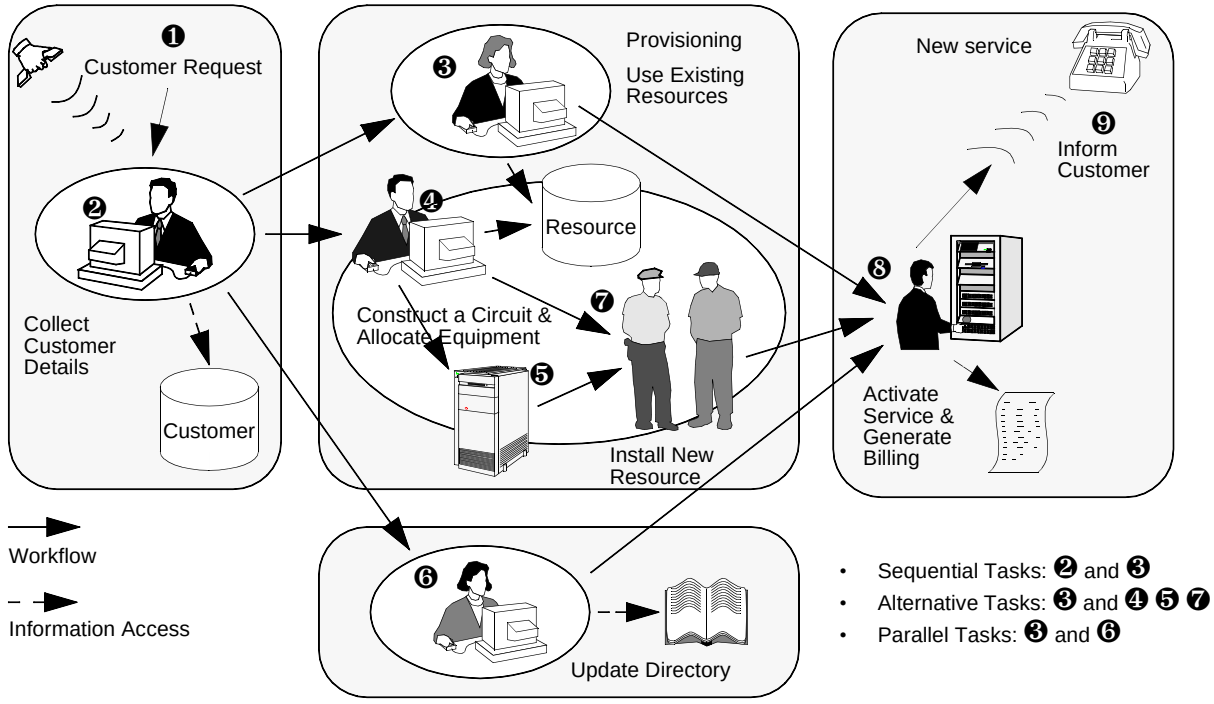

Figure 7: A Service Management Workflow

This workflow consists of the following tasks:

- Collecting information, which includes:
  - collecting information from the customer
  - verifying the accuracy of the information collected
  - creating a corresponding service order record
- Service provisioning, which involves constructing a circuit from a customer location to the appropriate telephone switch and allocating equipment to connect the circuit. This activity includes:
  - locating existing resources, such as lines and slots in switches
  - installing new resources if no existing resource is available

- Updating the telephone directory.
- Activating and testing the new service, informing customer that the new service is available and generating bills.

5.2 FEATURES

The workflow exhibits the following features:

- *task dependencies*—Examples of task dependencies in the workflow are:
  - sequential tasks—A service order is created after the customer information is collected.
  - parallel tasks—Service provisioning activities and changing directory action can be started at the same time. When both tasks are completed, next task can be started.
  - alternative choices—Two service provisioning activities can be started at the same time, but only one is allowed to complete. If an existing resource is available and the provisioning is successful, this resource will be used and the other provisioning task is rolled back. The rollback semantics can be user defined.
- *rollback behaviour*—Among two service provisioning activities, only one task can complete. If the existing resource is available for the service, the task which intends to use the existing resource should commit. Concurrently, the task which installs new resources should roll back. However, the rollback behaviour of the second task can be defined to complete the activity and record the newly created resources in the resource database for later use.

Transactional semantics is required to guarantee the correctness of such workflows. Figure 8 illustrates such semantics in a service provisioning workflow.

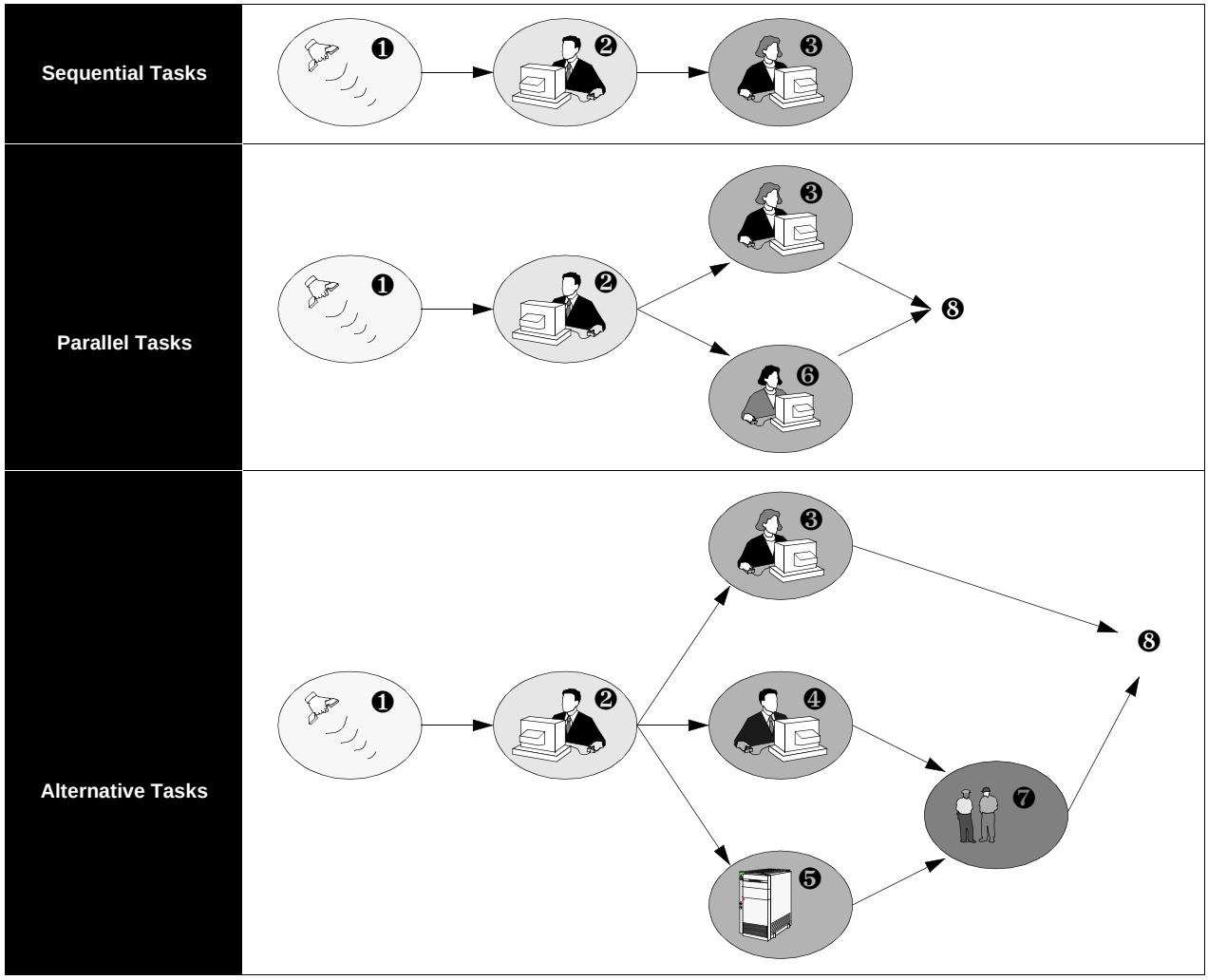

Figure 8: Transactional Semantics in a Service Provisioning Workflow

5.3 TRANSACTIONAL WORKFLOW IN A SERVICE MANAGEMENT PLATFORM

Transactional workflow management tools can be built on top of telecommunication service management platforms as building blocks for specific application support.

The basic components of a service management platform are:

- *distributed TMN management platform*—which provides the basic support for general network and systems management activities. It also provides environment support for application integration and interoperability between different business domains.
- *service management functions*—which support telecommunication service management functions defined in TMN, such as fault management and accounting management functions, and provides support for service integration. Service management functions enable new services to be installed rapidly and existing services to be integrated.
- *application support*—which provides a set of facilities useful for development of service applications. A transactional workflow management system can be one of the components to provide support for service provisioning, creation, activation etc.

Figure 9 shows the architecture of transactional workflow in a service management environment.

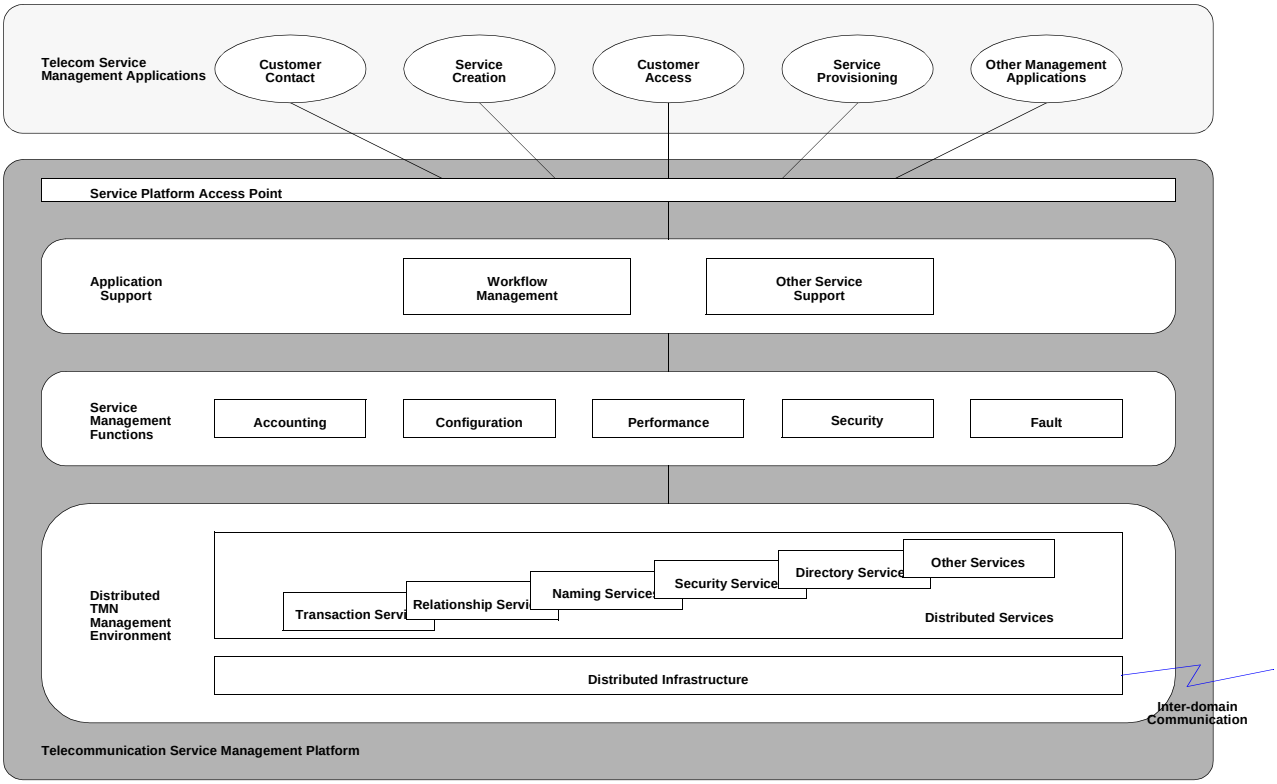

Figure 9: Transactional Workflow in Service Management Platform

6 CONCLUSION

Transactional workflow management is a useful technology which combines the features of process control and process automation of workflow with the power of transaction processing. In a process control and automation environment, we demonstrated that additional transactional features is desired.

Transactional workflow management technology is a very useful technology framework for TMN service management. It provides a framework to support telecommunication business process such as ordering and provisioning.

7 ACKNOWLEDGEMENT

The authors would like to thank the CiTR R&D group for their useful comments.