24

A Transaction Service of an ODP compliant Engineering Platform

Smith R., Maroulis D.
KYROS
55 Evripidou St., Kallithea 176 74, Athens, Greece
Telephone numbers: +30-1-9680708, +30-1-9309908
Fax numbers: +30-1-8940022, +30-1-9309908
E-mail addresses: rsmith@compulink.gr dmarou@compulink.gr

Abstract

This paper is based on the approach taken by the EU's RACE II project EURSAF on formulating a feasible Engineering Platform, as viewed from the ODP engineering viewpoint perspective, which will support the development, implementation, execution and management of advanced telecommunication services. The paper outlines the concepts of the Engineering Platform's Transaction Service, whose role is to provide support for transaction processing of telecommunication application services.

The key objective of the Transaction Service is to provide distribution transparency in terms of transaction transparency, thus contributing to the creation and operation of an open heterogeneous environment, which is suitable to all of the players involved. A brief description of the requirements of distributed transaction processing in a telecommunications environment is given and the related actions that need to be supported are presented. Transaction processing in the telecommunications environment has some unique requirements, such as scalability, multidomain, multiparty, real time, extensive heterogeneity, billing, etc. The types of transactions supported and the issues for constructing the Transaction Service are discussed. The resolution of these issues leads to the design of the Transaction Service in outline form. Elements of the design are presented.

Keywords

ODP, Engineering Platform, Transaction Service, Engineering Objects.

F. A. Aagesen et al. (eds.), *Information Network and Data Communication*

1 INTRODUCTION

The role of the EURSAF Engineering Platform's Transaction Service is to provide transaction support for telecommunication application and for other Engineering Platform modules, which require such support. Transparencies constitute the foundation for such support. Distribution transparency removes the need for the service developer to be concerned that a transaction might be distributed. Transaction transparency removes the need for the service developer to be concerned with transaction issues. The Transaction Service ensures that a transaction is completed and it provides recovery mechanisms if it is aborted.

In the Information Technology (IT) environment, transaction processing is defined as a set of operations on data that are either all performed or non are performed, while at the same time the integrity of the data and the logic execution is ensured. Transactions follow well defined rules and are subject to policies with the goal to either modify data in a consistent manner or leave data unchanged in the event of failure, thus moving data from one consistent state to another consistent state. In the case of distributed transactions, bound data is not centralised anymore and the tasks that a transaction consists of can also be distributed, with the integrity and the consistency of the data and of the execution still guaranteed [5]. Before the required transaction engineering mechanisms can be formulated it is necessary to address transactions from an ODP computational viewpoint [1], where transactions can be viewed as consisting of several building blocks such as begin transaction, commit, abort, recovery, rollback, concurrency, end transaction.

In the context of Telecommunications systems, transaction processing has to take into account the additional issues of scalability, multidomain, multiparty, real time, extensive heterogeneity, billing, etc. The Transaction Service of the EURSAF Engineering Platform will take into account the associated building blocks that comprise a typical transaction, while at the same time it will render the service provision transparent to the transactions themselves.

A typical telecommunications service, executing as an application on the Platform, can comprise many transactions. The individual transactions which comprise a service and the boundaries and ordering of these transactions are computational viewpoint issues and outside of the scope of the Engineering Platform. The application of distributed processing designs and techniques is a realistic starting point in attempting to design a transaction service for such an environment, where the additional telecommunications requirements have to be taken into account and incorporated in the design.

2 THE COMPUTATIONAL VIEWPOINT

In distributed processing, a number of transaction cases need to be modeled and supported. Various actions can disrupt a transaction as it progresses through its normal Begin-Body-End path. Transaction requests and/or responses can be lost or delayed, additional parties might be parts of the transaction in responding to the same requests, etc. Figure 1 presents the computational view of a transaction, that incorporates the different cases.

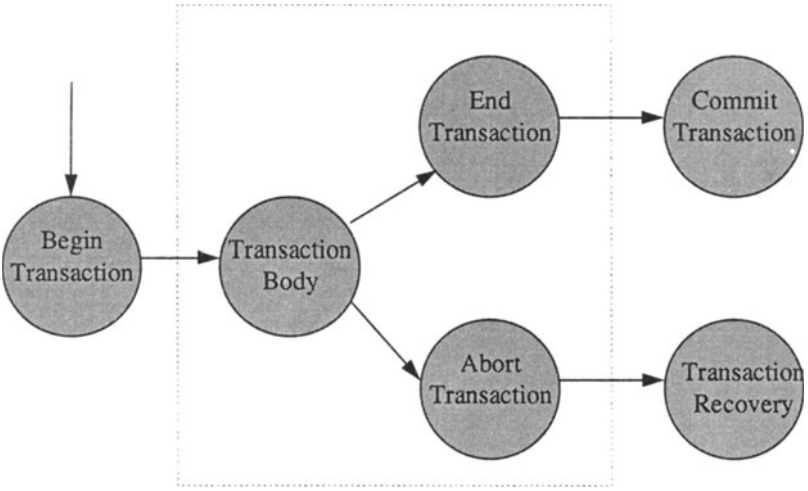

Figure 1 The Computational view of a transaction.

Thus, the first three fundamental actions Begin, Transaction Body, End Transaction, that cover the normal path of a transaction are expanded to include actions that deal with Commitment, so as to signify/stamp the successful completion of a given transaction, and Abort, to handle cases where transactions are unable to complete successfully, accompanied by the necessary Recovery actions [4].

3 REQUIREMENTS

The ISO/IEC 10026 [3] Recommendations specify the requirements for distributed transaction processing from the user needs and modeling perspectives. The addition of requirements that are particular to the telecommunications environment formulate the set that needs to be met by the design of the Engineering Platform as a whole and the Transaction Service in particular. The focus of the project during the design was on four particular issues which are briefly explained:

- Handling scalability issues. The Transaction Service should be capable of handling a very large number of service nodes, as well as service nodes of various sizes. Engineering techniques such as domain partitioning, avoidance of large routing tables and of centralised algorithms are among the things, which should be common to the Platform service as a whole, that should be included in the design of the Service.

- Supporting real time applications. The provision of the Transaction Service should be quick, avoiding introducing any delays to the application. The use of timers, fixed point commit states whenever possible, heuristic decisions, etc., will aim at meeting this requirement.
- Allowing migration. The DPE Service in general and the Transaction Service in particular should be designed in such a way as to handle existing older technology systems, which are common in Telecommunications, as well as future systems, without becoming obsolete. The modularity in the design and the inclusion of the proper interfaces are the steps to take to tackle this crucial problem of coexistence.
- Providing manageability. The DPE and its Services need to be manageable. In Telecommunications, this requirement is especially important, due to the inherent multidomain, multiparty, and heterogeneous nature of this environment. The Transaction Service aids in platform manageability by providing a transaction environment which includes flexible management interfaces allowing transaction management at various points in the progress of a transaction and by the various actors and components involved in the transaction.

4 TYPES OF TRANSACTIONS

The diversity and complexity of the telecommunications environment could lead to the identification and grouping of different transaction types that can be handled differently by the Transaction Service of the Engineering Platform. A non exhaustive set of the different transaction types follows:

- Flat (normal) transactions. These are the straight forward simple transactions that start with a Begin and finish with either a successful End accompanied by a Commit, or an unsuccessful Abort accompanied by a rollback.
- Long transactions. This type of transactions resembles the previous one except that the time it takes to execute them is longer than usual. Due to this additional requirement, there should be appropriate features in the design of the Transaction Service to support them (i.e. signify the nature of the transaction when it begins, disable the features that put time restrictions, etc.).
- Scripted Sequential transactions. In this case, the successful end of a transaction triggers the beginning of the next. The scenario is modeled as a workflow script. Special care should be given in designing the Transaction Service, to accommodate this type of transactions by acknowledging it during the initiation of the Service and setting up the appropriate commit and recovery features for addressing failures in the chain.
- Nested transactions and/or subtransactions. Again another type of transaction that needs to be accommodated in the design of the Service, during the initiation, commit, and recovery phases.
- Queued Transactions. Transactions required by applications, such as voice mail, where the transaction is submitted and queued for processing at a later time.
- Partial Transactions with fixed point. This type of transactions deviate from the traditional types and are more particular to Telecommunications. Certain types of transactions migh

not be able to be rolled back completely or it might not be a need to do so. So intermediate points of commit can be utilised. In such cases a transaction can be modeled as consisting of parts with fixed points of commit.. There will not be a need to roll back past that fixed point. In this case the transaction still abides by the ACID properties, but in a piecewise manner.

The types of transactions could be thought of as an open ended list. Any new transaction types could be checked against the list and a new entry can be made if needed. The Transaction Service must be designed so that these new transaction types can be added easily without affecting the operation of the existing transaction types.

5 THE TRANSACTION SERVICE

5.1 The Components of the Service

To be consistent with ODP [1] and OSA [6], the underlying EURSAF engineering architecture needs to be defined in terms of the Capsules, Clusters and Basic Engineering objects which comprise the Transaction Service. The communications mechanisms of these components with the nuclei have to be in place as well.

Of particular interest are the different transparencies that need to be part of the Service. The Platform, as perceived from the ODP Engineering viewpoint, has as a major objective the provision of the various distribution transparencies to the applications executing on the platform and to service developers writing applications for the platform. As such, a primary goal of the EURSAF Transaction Service, as perceived from the ODP Engineering Viewpoint, will be to provide distribution transparencies for transaction support. In addition, a key engineering objective of the Transaction Service will be to provide transaction transparencies to service applications and developers, thus hiding the complexity of transaction mechanisms from the service developer, since the intricacies of transaction processing are outside the scope of knowledge of the typical telecommunications service developer. Four levels of transparencies for transactions will be supported. These are:

- Full Transparency - there will be instances where the service designers are totally unaware that an action they perform requires the service of the transaction service. This will be implemented via objects which have been internally inherently structured as transaction objects.
- Partial Transparency - this can be considered the normal transaction service. As such, it will be considered the default transaction transparency. From a computational perspective this would be seen as start transaction, various modify processing actions, abort and commit transaction and so on.
- Programmable Transaction Transparency (Translucency) - there will be times when the service designer will need to control some aspects of a transaction. For example, the service designer might need to implement a service using a combination of roll-back and roll-forward facilities.

- Coexistence Transparency - the future ISE environment will consist of multiple platforms. The Transaction Service will have to inter-operate with the transaction services of these platforms. The purpose of this transparency is to hide from the service developer and other EURSAF modules the fact that a transaction involves nodes with non EURSAF transaction services.

Within the Transaction Service additional transparencies are provided to assist in realising the Service, such as access, concurrency, location, migration, replication, resource, failure and federation transparencies.

5.2 Transaction Service Operation

As shown in Figure 2, the EURSAF Transaction Service is part of the minimal service machine, residing on each service node in the Service Network. Thus the Transaction Service can assume that the service exists on each service node involved in a transaction and will not need to have mechanisms to invoke an instance of the transaction service on remote nodes.

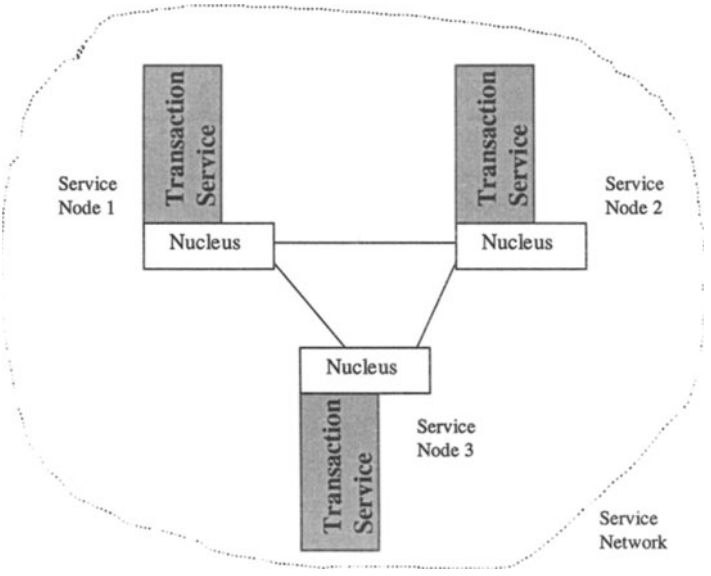

Figure 2 The Transaction Service as part of the Service Network.

In the EURSAF Engineering Platform environment, an atomic transaction is achieved through cooperation between a client object, which initiates the transaction, and instances of the EURSAF Transaction Service. The client specifies the series of actions that comprise the transaction and the Transaction Service guarantees to preserve the atomic properties of the entire sequence. All operations involved in a transaction involve a master and slave Transaction Service mechanism. The client contacts the Transaction Service of the node it is

residing on. This instance of the Transaction Service becomes the master and serves as the overall transaction coordinator. The master Transaction Service is responsible to contact and coordinate all the other Transaction Services required to complete the transaction. Thus, when having an operation between two service nodes x and y, with service node x requesting a transaction which involves an object on service node y, the nucleus on service node x will make a transaction request to its Transaction Service, which will serve as the master for the transaction. After determining, via the naming service, that the requested object is on node y, the Transaction Service will bind with the Transaction Service on node y. The Master Transaction Service will instruct the Slave Transaction Service residing on the local node that has the proper transaction mechanisms to set up, to properly service the transaction.

In the event that the naming service might determine that the requested object is on the same service node, the Transaction Service on node x will function as both the master and slave. The recovery and concurrency mechanisms which comprise a transaction will be under the control of the slave instance of the Transaction Service.

The support of nested transactions will require that the Transaction Service also take on the role of controlling the nested part of a transaction. The master-slave transaction model will be extended to support nested transactions. In its nested role, the Transaction Service will have to provide the recovery and concurrency mechanisms required for objects involved in the subtransaction portion of a nested transaction.

Coexistence and interoperability with non-EURSAF nodes will be a key platform requirement. The Transaction Service will have to interoperate with non-EURSAF transaction services. There are two cases which have to be supported, one where the EURSAF Transaction Service is the master and one of the slave nodes is non-EURSAF, and the other where the EURSAF node is a slave in a transaction initiated by a non-EURSAF node.

5.3 Outline Formulation of the Transaction Service

All transactions commence with a Begin Transaction request and either proceed normally and terminate with an End Transaction or abort and enter the appropriate recovery process. Transactions can be aborted either by the object requesting the transaction or by the Engineering Platform.

In order to formulate the Transaction Service, there is a need to address the Engineering Parameters involved in the Service. The Transaction Engineering Parameters are the parameters which control the engineering mechanisms which constitute the Transaction Service. The Transaction Engineering Parameters control all aspects of how a transaction is implemented, providing flexible support for the diverse telecommunications environment. These parameters are specified at the beginning of a transaction and may be changed as the transaction progresses by accessing various management interfaces.

The Transaction Engineering Parameters are a complex set of parameters. The average service designer might not need to get deeply involved in the manipulation of these parameters in order to add transaction support to his application. The transaction transparency aspect of the EURSAF Transaction Service hides the complexity of these parameters from the service designer not requiring transaction flexibility, while allowing the manipulation of these Parameters to the service designer requiring such flexibility.

The Transaction Engineering Parameters operate on the concept of defaults. When a service node is booted and the Transaction Service capsule initiated, the Transaction Engineering Parameters will be set based on the capsule template for that node. The Transaction Engineering Parameters defaults will vary from service node to service node depending on the engineering resources which compose an individual service node. As a service node's resources change, management interfaces can be used to change the Transaction Engineering Parameters defaults. A Begin Transaction request, issued by a service designer desiring full transaction transparency, will use the Transaction Engineering Parameters defaults on the master transaction service to control the transaction. Service designers, wishing to alter certain Parameters, will include in their Begin Transaction request the Transaction Engineering Parameters they wish to change from the defaults. Query interfaces will allow the service designer to determine the current Transaction Engineering Parameters defaults used at a service node. The Transaction Engineering Parameters used in a Begin Transaction command will be position independent and need only include the parameters to be changed. This allows the flexibility of adding new parameters, as new transaction engineering mechanisms become available, to the Transaction Engineering Parameters list without effecting existing transactions.

The Transaction Engineering Parameters are organised into different categories representing the different aspects of a transaction. Such categories are communications, recovery, commit, concurrency and general.

The next major issue in the formulation of the Transaction Service is the identification of the clusters needed for the operation of the Service. A brief description of these clusters follows [2]:

- Transaction Coordinator Cluster
  It serves as an entry point to request and process any type of transaction service. It is responsible for the overall coordination of all the engineering mechanisms (Basic Engineering Objects combined into Clusters) involved in a transaction. It acts as the Capsule Manager for the Transaction Service. It controls a transaction from beginning to end. It establishes a unique transaction number for each transaction (this ID will have to have a naming scheme which will identify the node where the transaction is anchored). It serves as a single point to manage the transaction in its entirety. It checks to see if all objects to be involved in the transaction are available and ready. It will keep the status of the current state of all active transactions. It is responsible to ensure that all transactions are either committed or aborted and that there will be no lost transactions.
- Locator Coordinator Cluster
  It is responsible for locating needed objects, maps name to address, provides location transparency. In providing migration transparency, it will maintain an active channel to the modules used for location. It tracks the migration of located objects by updating the name to address location information. It will be used by all modules which need access to objects via the name and it will be permanently bound to the name service at initiation.
- Concurrency Coordinator Cluster
  It is responsible for controlling concurrent access to local node objects involved in transactions. On abort, after recovery is completed, it issues requests to the nucleus to release all current active concurrency mechanisms on objects involved in the transaction.

- Recovery Coordinator Cluster
  It is responsible to set up the recovery mechanisms specified in the Begin Transaction parameters. In the event that a transaction is aborted this cluster must ensure that all objects involved in the transaction maintain the ACID properties they had before the start of the transaction It is also responsible for instructing the appropriate recoverer to begin recovery.

As an example, Table 1 lists the engineering objects contained in the Transaction Coordinator Cluster.

Table 1 The Engineering Objects of the Transaction Coordinator Cluster

Master Transaction Registrar	Nested Transaction Registrar
Slave Transaction	Transaction Scripter
Transaction Archive	Transaction Billing
Transaction Time Reference	Transaction Garbageman
Transaction Security	Capsule Manager
Cluster Manager	Transaction Parameter
Recovery Enforcer	Commit Enforcer
Transaction Batch Scheduler	Slave Transaction Actions
Class1 and Class2 Transaction Adapters (for handling non-EURSAF nodes)	Nested Transaction Actions

The details of all the objects as well their interrelationships and the mechanisms involved can be found in [2].

6 CONCLUSIONS

This paper presented the concepts involved in the EURSAF Engineering Platform's Transaction Service, whose role is to provide support for transaction processing of telecommunication application services.

The key objectives and the requirements of the Transaction Service, as they apply to the telecommunications environment, were identified and the actions that need to be supported were described and explained. The types of transactions supported and the issues for constructing the Transaction Service were discussed. The resolution of this issues leads to the design of the Transaction Service. The aim of this Service is to provide support for distributed transaction processing in a telecommunications environment. This support is transparent to the relevant applications and it is designed to be provided in real time, while at the same time it can handle multiple systems and large domains.

The design of the Transaction Service is based on RM-ODP and current research has gone down to the detail of specifying the needed clusters, the engineering objects within the clusters and their inter-relationships. The mechanisms involved in the operation of the Service are also described [2] and the issue of handling non-EURSAF nodes is addressed with the provision of appropriate objects and mechanisms. Specifications of the object interfaces are in the plans to be produced in IDL, thus bringing the Service closer to actual implementation.

7 REFERENCES

[1] Basic Reference Manual of Open Distributed Processing, ODP Engineering Language (1994) ITU-T X.903-ISO 107046.

[2] EURSAF Deliverable D-8 (1995) The EURSAF Engineering Platform, EURSAF RACE II R2124.

[3] ISO/IEC 10026 Information Technology - OSI Distibuted Transaction Processing Parts 1-3, 1992

[4] Goscinski, A. (1992) Distributed Operating Systems, The Logical Design, Addison Wesley.

[5] Lacoste, G. (1994) Distributed Transaction Processing in the IBC, Towards a Pan-European Telecommunication Service Infrastructure, IS&N'94, Aachen.

[6] Trigila, S. (editor), (1995) Open Services Architectural Framework for Integrated Service Engineering, CASSIOPEIA RACE II Deliverable R2049 /FUB/SAR/DS/P/023/ b1.

8 BIOGRAPHIES

Mr. **Ron Smith** received his MSc(Electrical Engineering) from the State University of NY, USA. He worked in the US and Europe with multinationals in the capacity of customer engineer, project manager as well as senior consultant (Hewlett Packard, Wang, Sierra Research) in the areas of customer support in H/W, S/W systems and telecommunications. Mr. Smith also taught and at Troy State University from 1982 to 1987. He is currently involved in EU's RACE and ACTS projects in the areas of telecommunications and multimedia with the Greek research company KYROS.

Dr. **Dimitris Maroulis** received his M. Eng. from Rensselaer Polytechnic Institute, USA in 1975 and his Ph.D. from the State University of N.Y. at Buffalo USA in 1979, in Electrical Engineering. He served as an instructor at SUNY/B and he taught numerous computer courses at the University of Maryland, European Division. He has participated in EU projects in various programs, such as COST-11bis and ter, ESPRIT and AIM. Dr. Maroulis is currently the Technical Director of KYROS and he is involved as a team leader in EU's RACE, ACTS, as well as Libraries projects. His current interests are in the fields of computer and telecommunications networks and their security, and multimedia.