

Reflective Transaction Service

Weihai Yu and Randi Karlsen

Department of Computer Science

University of Tromsø

Norway

{weihai|randi}@cs.uit.no

1 Motivation

Traditional transaction processing, flat or nested with two-phase locking [1], is unsuitable for applications with long transactions, special security or real-time requirements, or heterogeneous transaction management subsystems [2]. Many advanced transaction models have been proposed [2] in the last fifteen years, some of which have been successfully applied in various database applications. The CORBA Object Transaction Service (OTS) [3] addresses heterogeneity issues by adapting heterogeneous transaction management subsystems to standard object service interfaces defined in CORBA IDL. There are still needs for support for various transaction models.

The next generation middleware should be reflective for QoS sensitive and resource demanding applications [4]. In a large-scale dynamic environment, the transaction service should also be reflective to dynamically provide the necessary support for various transaction models in accordance to user requirements, changing environment, and so on.

Next, we provide a motivating scenario. Note that similar scenario can be common for other transaction models as well. An application starts a transaction that accesses multiple databases. The transaction may convert transparently from flat into (closed) nested when there is need for extra internal concurrency or independent recovery. If the transaction takes too long, it may convert into open nested to release resources immediately. Converting a transaction into open nested requires inspection of the corresponding subsystems to verify that either the resources are equipped with compensation capabilities or the application provides explicit compensation logic. The conversion may occur either locally to individual databases or globally to the entire transaction. It may happen either automatically within the transaction service, or requiring approval of the user (when the compensation has consistency or economical consequences).

2 Open Bindings for Reflective Transaction Service

We propose a reflective transaction service architecture with open bindings [4] (Figure 1).

In the architecture, a transaction is regarded as an explicit open binding where OTS stubs and skeletons provide local bindings to applications (transactional clients) and databases (transactional objects) whereas the OTS core connects the application and the involved databases in a distributed setting. The transaction binding consists of sub-bindings that associate individual databases with the transaction service. The binding, sub-bindings and their components provide meta-interfaces for inspection and change of their behavior.

In our scenario, the components need be inspected to verify, for instance, if a resource is compensatable or if an application provides compensation subroutines. A compensatable resource is recoverable, and in addition it logs object-level operations that can be logically undone. The availability of a compensatable resource allows a sub-binding to convert immediately into open

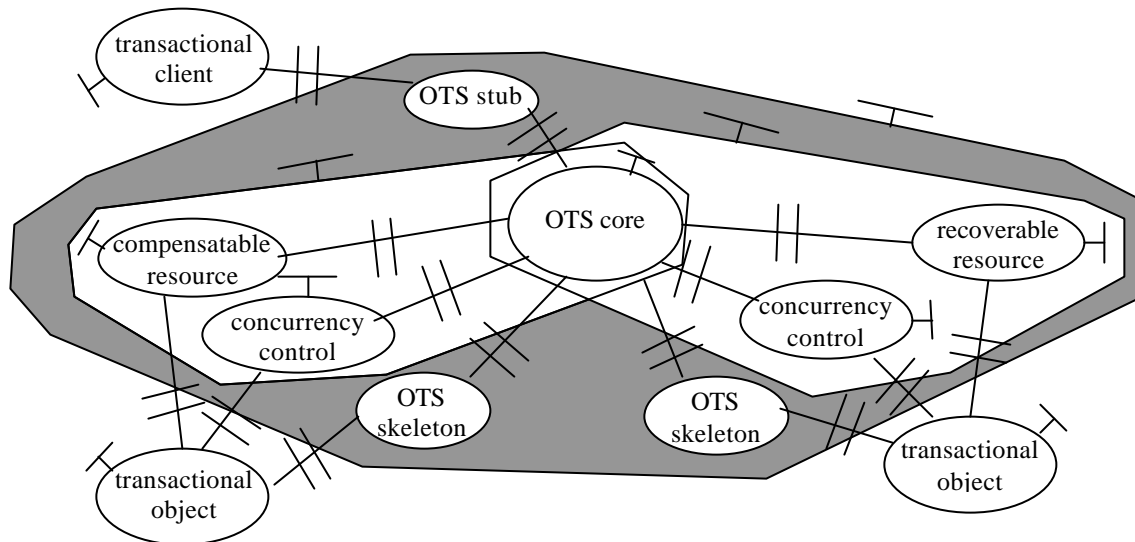


Figure 1. A reflective OTS architecture with open bindings

nested without intervention of the application. Conversion of transaction models can be initiated at sub-bindings or at the entire transaction binding and then be propagated recursively to all its sub-bindings.

3 Requirements on Reflection Support

Three meta-spaces, namely the encapsulation, composition and environment meta-spaces, are proposed in [4] for reflection support in a middleware. The support of all these meta-spaces is required for a reflective transaction service. Individual components like the recoverable resource may be inspected to verify if it is compensatable. A transactional client may be asked to provide compensation sub-transactions dynamically. A transaction open binding can be regarded as a composite object whose life-time is the duration of the transaction. The composition may be changed when, for example, a sub-transaction releases resources early or when a compensation sub-transaction must be started upon the abortion of the entire transaction. The underlying environment must be inspected to see, for example, if the transaction takes too long due to a network partition.

4 References

- [1] Gray, J. and A. Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman Publishers, 1993.
- [2] Elmagarmid, A. (eds.), *Database Transaction Models for Advanced Applications*, Morgan Kaufman Publishers, 1991.
- [3] Object Management Group, *CORBA Services Specification*, Chapter 10, Transaction Service Specification, December, 1998.
- [4] Blair, G., G. Coulson, P. Robin and M. Papatthomas, "An Architecture for Next Generation Middleware", In *Proceedings of Middleware '98*, 1998.