Contract-Based Workflow Design Patterns in M-Commerce

V. K. Murthy
University of New South Wales, Australia

E.V.Krishnamurthy
Australian National University, Australia

INTRODUCTION

This overview describes an object-based workflow paradigm to support long and short duration transactions in a mobile e-commerce (or m-commerce) environment. In this environment, the traditional transaction model needs to be replaced by a more realistic model, called a "workflow model" between clients and servers (peers) that interact, compete and cooperate, realizing an intergalactic client-server program. The various types of task patterns that arise in m-commerce, for example, reservation, purchasing, require a "what if" programming approach consisting of intention and actions for trial-error design before an actual commitment is made. Such an approach enables us to take care of the unpredictable nature of connectivity of the mobile devices and the networks and also provide for the trial and error program design required in m-commerce.

BACKGROUND

A mobile computing environment consists of fixed host computers and mobile client computers that are linked together by a network (wired or wireless) so that these computers can communicate among themselves using messages. The mobile clients (MC) are capable of connecting to the fixed network via a wireless link. Fixed host (FH) provides mobile application services and information to mobile clients. MC supports query invoking and information filtering from FH to provide personal information service. We model a mobile computing environment as a collection of different types of objects by identifying each computer (mobile and stationary) as an object. This model is called Mobile Object Programming System (MOPS).

A mobile object programming system (MOPS) (Vitek & Tschudin, 1997) is interpreted as a collection of objects interacting through messages. Each object maintains its own share of data and has its own program piece to manipulate it. That is each object combines datastructure and functionality. The objects are active and behave like actors in a movie, each following its own script and interacting with other objects.

A task submitted from a mobile client is called a mobile transaction. It is a distributed task that can be executed partly within that mobile client (MC) as an internal transaction (Intran) and partly in other fixed hosts (FH) as external transactions (Extran). Each FH has a coordinator FHC that receives external transaction operations from mobile hosts and monitors their execution in the database servers within the fixed host. Similarly each MC has a coordinator MCC.

Conventionally, transactions are assumed (Bacon, 1993; Krishnamurthy & Murthy, 1992) to satisfy the ACID properties, namely:

- **Atomicity:** All or none of transaction happens;
- **Consistency:** A transaction preserves the consistency in database before and after its execution;
- **Isolation:** Intermediate results are not externally made visible until commitment;
- **Durability:** The effects are made permanent when a transaction succeeds and recovers under failure.

The ACID properties turn out to be restrictive for mobile transactions and need to be relaxed as illustrated by the following example on flight reservation.

Here, one needs to have the following tasks: select a suitable airline that offers cheaper fares, ensure there is vacancy, and make an advanced booking (that is to be confirmed later). These individual steps are really not traditional transactions, but well-defined program pieces that can be carried out concurrently and may need to follow a predefined partial order to satisfy certain predicates (e.g., seat availability) invariant criteria (number of seats is determined by the size of aircraft) and synchronization (one step requires the completion of other step). Therefore, the steps need not be atomic, need not be immediately consistent, and need not satisfy isolation property since intermediate non-commit states are to be made visible (called externalization). Furthermore, such steps are liable to be cancelled eventually and require a suitable rollback preserving some local states. Also, in a

cooperative mobile environment it may be necessary to allow data exchange between some transactions during their execution thereby necessitating the relaxation of the isolation property.

Thus in a mobile environment, we need a new model where:

(i) The isolation property is removed making the intermediate results visible.
(ii) The precedence order in execution and other dependencies are taken care by modifying the atomicity requirement.

This model is called "a workflow" between the MC and FH. The workflow model, its contract based formulation and the e-commerce patterns described here result from the basic work on contract model of Wachter and Reuter (1995), the design by contract by Meyer (1992a, 1992b) and the design patterns and contracts by Jezequel, Train, and Mingins (2000).

MAIN THRUST OF THIS ARTICLE

The main thrust of this article is to describe the workflow paradigm for applications in m-commerce. Unlike the transactional paradigm, which is severely restricted by the ACID properties, the workflow permits more general properties that are suitable for m-commerce applications. We illustrate some of the important applications and the design of suitable protocols. We also classify some workflow patterns arisng in e-commerce and the language support needed.

WORKFLOWS: EXTERNAL AND INTERNAL

A global workflow (we call it an external workflow or *Extran*) T(ij) is defined as a workflow between two objects O(i) and O(j); this consists of a message sent from O(i) to execute a desired workflow in O(j). This message is received by O(j). O(j) has a behavior specified by: Pre(T(ij)), G(j), C(j), Post (T(ij))S(j), where Pre() and Post() are respectively the pre and post states that are active before and after the workflow T(ij). G(j) is a guard of O(j) to signal when the required precondition is met, and C(j) is the command function; S(j) signals when the post condition is achieved. Here, the script specifies what message O(j) can accept and what actions it performs when it receives the message while in state Pre(T(ij)) to satisfy the post condition post(T(ij)). The Extran T(ij) can trigger in O(j) numeric, symbolic or database computations.

Each Extran T(ij) from object i to object j triggers a set of serializable computations in O(j) either in a total order or in a partial order depending upon whether parallelism, concurrency and interleaving are possible locally within O(j). If the object O(j) is "made up" of subobjects, we may have to execute a workflow consisting of several local workflows (called internal workflow or Intran). After executing Intran, each object reaches a new state from an old state using its internal command set C(j); before executing the commands, the required precondition is met, and after completion of the command set, the post condition is ensured in the new state. This is the design by contract approach (Meyer, 1992a, 1992b) and widely used in the language Eiffel. The precondition is specified by "require" and post condition by "ensure", see Meyer (1992a, 1992b).

We shall see in the next section that the Extran and Intran have more general properties than the ACID properties mentioned earlier.

CHEMICAL REACTIVITY-LIKE PROPERTIES

The Extran and Intran have more general properties called "chemical reactivity properties", since they resemble chemical reactions: molecularity, contractual obligation, opacity during a molecular action, and retry or rescue through a recovery protocol bringing the system back into the invariant state. These are defined as below:

(i) **Molecularity:** If there is a crash during a composite operation, all of the effects of the sub-operation are lost. If there is no crash, the composite or molecular operation is complete. That is, a molecule is synthesised fully or not at all.
(ii) **Contractual obligation:** Invocation of a single composite operation takes the program from one consistent state to another. This means precondition and post condition of a contract holds. Thus, conventional consistency is replaced by contractual obligation.
(iii) **Opacity:** The results of the sub-operations of composite operation should not be revealed until the composite operation is complete.
(iv) **Durability:** If a crash occurs and contract fails and a component cannot meet its obligation and fails in its contract, an exception is raised. Then we have three possibilities:
    a. Exception is not justified: It is a false alarm we may ignore.