

# ACTA: A Comprehensive TransAction Framework for Extended Transactions

Panos K. Chrysanthis

Dept. of Computer Science  
University of Pittsburgh  
Pittsburgh, PA 15260

Krithi Ramamritham

Dept. of Computer Science  
University of Massachusetts  
Amherst, MA 01003

Although powerful, the transaction model adopted in traditional database systems is found lacking in *functionality* and *performance* when used for new applications, such as CAD/CAM, and design environments. Various extensions to traditional transaction model have been proposed to address these drawbacks<sup>1</sup>. In order to analyse these *ad hoc* extensions and in search for a good implementation support for the new applications, we have developed a comprehensive transaction framework, called *ACTA*<sup>2</sup>

ACTA characterizes the semantics of interactions in terms of different types of dependencies between transactions and in terms of transactions' effects on objects. Through the former, one can specify relationships between significant (transaction management) events, such as Begin, Commit, Abort, Delegate, Split, and Join, pertaining to different transactions. For instance, here is the definition of some often occurring dependencies: In these definitions,  $H$  denotes a history that indicates the order in which the events associated with a set of transactions occur. The predicate  $\epsilon \rightarrow \epsilon'$  is true if event  $\epsilon$  precedes event  $\epsilon'$  in history  $H$ ; false otherwise.

- $(Commit_{t_i} \in H) \Rightarrow ((Commit_{t_i} \in H) \Rightarrow (Commit_{t_j} \rightarrow Commit_{t_i}))$ . This says that if both transactions  $t_i$  and  $t_j$  commit then the commitment of  $t_i$  precedes the commitment of  $t_j$ . This **Commit-Dependency** is indicated by  $(t_j \text{ CD } t_i)$ .
- $(Abort_{t_i} \in H) \Rightarrow (Abort_{t_j} \in H)$  i.e., if  $t_i$  aborts then  $t_j$  aborts, states the **Abort-Dependency** of  $t_j$  on  $t_i$  ( $t_j \text{ AD } t_i$ ).

<sup>0</sup>This material is based upon work supported by the National Science Foundation under grant IRI-9109210.

<sup>1</sup>Special Issue on Unconventional Transaction Management, *Bulletin of the IEEE Technical Committee on Data Engineering*, March 1991.

<sup>2</sup>P. Chrysanthis and K. Ramamritham. "A Formalism for Extended Transaction Models," *17th International Conference on Very Large Databases*, Spain, 1991.

Conditions under which events may occur can be specified precisely. For instance, one can state the conditions (a) necessary for a transaction to *commit*, i.e., make permanent, (some of) its changes, or (b) sufficient to force a transaction to *abort*. Transactions' effects on objects are specified by associating two entities, namely *view* and *conflict set*, with each transaction and by stating how these are affected when a transaction invokes operations on objects or when significant events occur. A transaction's view specifies the state of objects visible to that transaction. A transaction's conflict set contains those operations against which conflicts have to be determined.

Inter-transaction dependency relationships, the view and conflict set of transactions, along with the conflict relationships defined between operations on objects capture the *visibility*, *consistency*, and *recovery* properties of transactions. ACTA has been used successfully to answer the following questions.

- What properties does a model possess vis a vis *visibility*, *consistency*, *recovery*, and *permanence* properties?
- In what respects does one transaction model differ from another? Can aspects of one model be incorporated within another?

Currently, we are exploring the practical application of ACTA for identifying the mechanisms suitable for managing extended transactions. Our goal is to design mechanisms supporting a particular transaction model given its ACTA specifications. Towards this end, we are developing programming language structures to express the type of specifications afforded by ACTA. We are also examining the correctness of a particular implementation by first formalizing the properties of the specific mechanisms used in the implementation and then showing that they maintain the desired correctness properties.