Transaction Decomposition in Object-Oriented Models

R.J. Wieringa*

Conceptual modeling is that part of requirements determination that is concerned with specifying a precise model of observable system behavior, that can be used by implementors to realize the system (without making additional decisions about observable system behavior) and that can be used by domain specialists and users to validate the utility of the system. There is currently a need for conceptual modeling methods that adopt the best elements of modern object-oriented methods *and* traditional functional methods, as well as bridges the gap between informal domain understanding and formal system specification. The Method for Conceptual Modeling (MCM) reported elsewhere [6, 5] is composed from elements of traditional and modern methods and tries to bridge the gap between formal and informal understanding. It can be used in conjunction with a formal specification language such as LCM [3], Troll [4], Oblog [1] or Albert [2].

MCM takes a simple view of observable system behavior as a set of *transactions*, each of which is atomic (has no intermediary state). To represent transactions in an object-oriented way, MCM recommends making a *transaction decomposition table* that correlates transactions to object classes. The columns of the table represent transactions and the rows represent object classes. Each transaction is decomposed into one or more local events in the life of objects.

	borrow	return	renew	reserve	cancel	res_borrow
LOAN	create	return	renew			create
DOCUMENT	borrow	return	renew			res_borrow
MEMBER	borrow	return	renew			res_borrow
RESERVATION				create	delete	delete

In this library example, res_borrow is a transaction in which a library member borrows a document that s/he has reserved earlier. It decomposes into a creation event for a LOAN instance, a deletion event for a RESERVATION instance, and two borrow events, one in the life of the borrowed DOCUMENT instance and one in the life of the MEMBER instance. Each transaction is thus either a single local event or a synchronous communication between local events. All events in the life of an object must appear in the corresponding class row in the table. It is assumed that other parts of the model contain a specification of the effect of local events on the local state of objects as well as a specification of the life cycle of objects and of the preconditions of events. This could be done in a formal specification language such as LCM, Troll or Albert, or in a semi-formal style such as in the Fusion method. It is recommended in MCM to make a function decomposition tree of the system whose root represents the overall system function and whose leaves represent

*Faculty of Mathematics and Computer Science, Free University, De Boelelaan 1081a, 1081 HV Amsterdam. email: roelw@cs.vu.nl.

the system transactions. The nodes one level above the transactions are called *system services* in MCM.

The transaction decomposition table has a number of advantages. First, it can be used to define two modularizations of the system model. The system sefvices partition the behavior of the system into functional modules. This modularization is literally orthogonal to the object-oriented modularization of the system into classes. Note that function decomposition does not descend "into" the system, as it does in data flow modeling.

Second, the table provides a link between formal and informal understanding of the model. The function decomposition tree is highly meaningful to domain specialists and users, but has no formal meaning; the classes and the transaction decompositions can be given a precise meaning in a formal model, that has (virtually) no meaning to the domain specialists and users. Construction of the table is therefore a crucial step in the transition from an informal to a formal model, as well as in the validation of the formal model against the informal model. The table plays an essential role in the communication between the implementor and the user.

Third, the meaning of object-oriented models is usually contained in the communication structure, and the communication structure is usually very dense. The table gives a means to untangle this ravioli by decomposing the communication graphs into chunks that correspond to the system services.

Fourth, the table is well-known from traditional methods. Information Engineering uses tables with a similar structure and meaning. In Data Flow modeling, we can use the table to trace the response of the system to a stimulus (this corresponds to walking down a column of the table and making a DFD of what happens along the way). Some methods for ER modeling direct the discovery of entity types by starting from the transactions and listing for each transaction the entities that are created, deleted, read or updated. All of this means that the table is easily explained to analysts who use the old methods.

Finally, the transaction decomposition table is a simple technique that can be understood and used by people with little or no formal background. It has been used on several student projects as part of MCM and as a preliminary to write LCM specifications, by students with no training in formal specification, dynamic logic or process algebra.

References

[1] J.F. Costa, A. Sernadas, and C. Sernadas. *OBL-89 User's Manual, version 2.3.* Instituto Superior Técnico, Lisbon, May 1989.

[2] E. Dubois, P. du Bois, and M. Petit. O-O requirements analysis: An agent perspective. In O. Lehrmann Madsen, editor, *European Conference on Object-Oriented Programming (ECOOP'92)*, pages 458–481, Utrecht, June 29 – July 3 1992. Springer. Lecture Notes in Computer Science 615.

[3] R.B. Feenstra and R.J. Wieringa. LCM 3.0: a language for describing conceptual models. Technical Report IR-344, Faculty of Mathematics and Computer Science, Vrije Universiteit, December 1993.

[4] R. Jungclaus, G. Saake, T. Hartmann, and C. Sernadas. Object-oriented specification of information systems: The TROLL language. Technical report, Abt. Datenbanken, Tech. Universität Braunschweig, P.B. 3329, Braunschweig, Germany, 1991.

[5] R.J. Wieringa. A method for building and evaluating formal specifications of object-oriented conceptual models of database systems (`MCM`). Technical Report IR-340, Faculty of Mathematics and Computer Science, Vrije Universiteit, December 1993.

[6] R.J. Wieringa and R.B. Feenstra. The university library document circulation system specified in LCM 3.0. Technical Report IR-343, Faculty of Mathematics and Computer Science, Vrije Universiteit, December 1993.