

Adapting Transactions to Exceptional Situations Using Structured Messages

Sozo Inoue
sozo@c.csce.kyushu-u.ac.jp

Graduate School of Information Science and Electrical Engineering,
Kyushu University
6-1 Kasuga-Koen, Kasuga-Shi, Fukuoka 816-8580, JAPAN

Mizuho Iwaihara
iwaihara@c.csce.kyushu-u.ac.jp

Abstract

We concentrate on the property that the progression of collaborative work involves communication among participants, and we consider a flexible method for managing transactions utilizing structured messages. In the literature, advanced transaction models for collaborative work environment have been proposed. However, since the models treat work process as a target of transaction, it is difficult to adapt to exceptions or dynamic reconfiguration of processes. In this paper, we argue about examples of handling exceptions in an environment involving workflow processes and database transactions, utilizing structured messages. We use the message transaction model for specifying structured messages. The model has ability of specifying workflow processes and various advanced transactions.

1 Introduction

Transaction management technique[5], which is a remarkable outcome of the efforts on database management technology, has been evolved to various kinds of advanced transaction models[14] in order to avoid the decrease of concurrency caused by long-lived transactions, such as those which involve human activities.

One of the requests remained for supporting collaborative work is adaptation to unexpected situations. The real world is inevitably surrounded by frequently changing environment and/or unexpected human behavior. As for process management technology, dynamic and adaptive workflow systems for managing exceptional situations have been proposed[9, 15]. As for transaction management technology, a formalization of split/join methods is proposed for dynamic restructuring of transactions for tree-structured human activity in [13]. However the formalizations of adaptation are discussed, there has been little consideration on uti-

lizing the semantics of human activity, such as policies for decision making and negotiation among organizations. In an environment incorporated with heterogeneous information systems and with advanced transactions, it is important to utilize semantics of human activity for managing transactions.

We concentrate on the property that the progression of collaborative work involves communication among participants[2, 4, 10, 11]. It is helpful for us to apply the information obtained by communication to transaction management. In order to apply the information to adaptive transaction management, we use structured messages[3], in which each message includes a formalized representation of the utterer's intention.

In this paper, we model a framework for adapting transactions to exceptional situations utilizing structured messages, standing on the several examples in which participants of collaborative work handle exceptions in a heterogeneous environment with workflow processes and database transactions. We use the message transaction model[6, 7] for specifying structured messages. The model has ability of specifying workflow processes and various advanced transactional models.

Since our model is based on messages exchanged in communication, it is possible to provide participants an interface for coordinating execution of transactions, and to incorporate the semantics obtained by human decisions to transaction management. The following functions can be realized using our model:

- Participants obtain guidelines for maintaining consistency which is implicitly described in the communication context.
- Participants interactively prevent data violation caused by concurrent transactions.
- When heterogeneous information systems are confronted with a complicated exceptional situation, the

- $e_1 < e_2$ (precedence) : If both event e_1 and event e_2 occur, then e_1 must precede e_2 . There is no constraint on a possibility of occurrence of e_1 or e_2 .
- $e_1 \rightarrow e_2$ (occurrence) : If event e_1 occurs, then event e_2 must also occur. There is no constraint on the order of occurrences between e_1 and e_2 .

The followings are examples of transactional dependencies.

- $CR(S_i, S_j)$: (commit root) $c(S_i) < t(S_j), t(S_j) \rightarrow c(S_i)$.
 S_j can be executed only after S_i commits.
- $EX(S_1, \dots, S_n)$: $c(S_i) \rightarrow a(S_j)$ for any pair $i, j (i \neq j)$.
Only one of S_1, \dots, S_n can commit.
- $AA(S_i, S_j)$: (abort dependency) $a(S_i) \rightarrow a(S_j)$
If S_i aborts, then S_j must abort.
- $CC(S_i, S_j)$: (commit dependency) $c(S_i) < c(S_j)$
If S_i commits and S_j commits, then S_i precedes S_j .
- $CO(S_i, S_j)$: (commit occurrence) $c(S_i) \rightarrow c(S_j)$
If S_i commits, then S_j must also commit.
- $AC(S_i, S_j)$: $(c(S_j) \rightarrow a(S_i)) \wedge (a(S_i) < c(S_j))$
 S_j can commit only after S_i aborts.
- $CA(S_i, S_j)$: $c(S_i) \rightarrow a(S_j)$
if S_i commits, S_j must abort.
- $TE(S_i, S_j)$: (terminate) $(c(S_i) < c(S_j)) \wedge (a(S_i) < c(S_j))$
 S_j can commit only after S_i finishes its execution.

Transactional dependencies have an ability to specify executional dependencies among m-groups and various kinds of advanced transaction models. For example, the transactional dependency CR from `propose` to `book_purchase` in Figure 1(a) denotes that `book_purchase` cannot be executed before `propose` commits. Moreover, CC and CO from `book_purchase` to `commit` in Figure 1(b) denote that if `order_accept` commits, then `book_purchase` must commit afterward.

AND-group, +-group, and *-group are the specifications how many and whether m-groups are permitted to be created. We omit the description since the discussion of the specifications is out of scope in this paper.

Arbitrary creations of transactional dependencies may cause situations such that there exists no sequence of transactional events which satisfy the imposed dependencies. For this problem, we can essentially apply the sufficient conditions for deciding satisfiability of transactional dependencies discussed in [8].

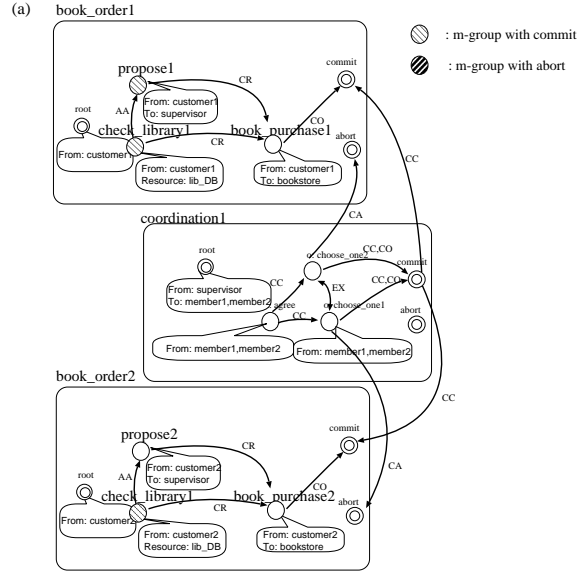
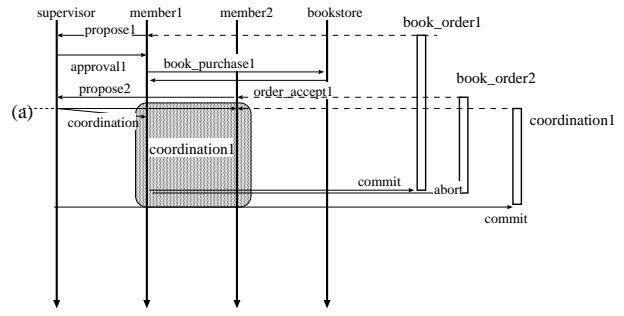


Figure 2. Event Diagram and m-groups of Scenario 1

2.2 Exceptional Situations in The Message Transaction

In the rest of the paper, we call the situation which is not assumed in the m-group templates' definition an *exceptional situation*. Exceptional situations in the message transaction model are classified as follows:

1. Modification on a child group:

Unexpected addition to or deletion from a child group leads to an exceptional situation when the modification is not assumed in the m-group template. For example, when a new compromise is found in a negotiation between two proposals, an m-group for compromise to the child group of the negotiation m-group is added. If the addition is not assumed in the m-group template of negotiation, an exceptional situation occurs.

2. Unsatisfiable transactional dependencies:

When a history of transactional events does not satisfy the existing transactional dependencies, an exceptional situation occurs, such that a post facto consent is taken, or that serial activities are executed in parallel.

3. Unexpected behavior of a transaction:

An exceptional situation occurs when a transaction does not perform in a predefined manner. An example is an abort of atomic transaction after that its commit event has been created.

4. Modification on a resource:

When a resource is to be deleted or replaced with another resource, an exceptional situation occurs if the resource is defined in the m-group template. For example, the situation when it is necessary to begin an argument with the provisional report in place of the final report leads an exceptional situation.

5. Modification on an m-group template:

Modification on an m-group template include the whole or a partial modification of 1–4. Moreover, a modified m-group template affects existing m-groups, and brings a more complicated exceptional situation. To overcome this problem, an approach of multiple versions for an m-group template is necessary, as discussed in [9] for evolving workflow schemas.

2.3 Scenarios for Exceptional Situations

Figures 2, 3, 4, and 5 illustrate several scenarios in which coordinations for exceptional situations are held among participants utilizing m-group templates shown in Figure 6, when each of member1 and member2 is to order a similar book to each other. We assume that the m-group templates shown in Figure 1 are usually used.

Scenario 1

In Figure 2, member1 proposes an order of a new book to the supervisor(propose1). The supervisor approves the proposal, and propose1 commits. Incidentally, member2 proposes a book which is similar to the book in propose1, and the supervisor induces member1 and member2 to choose one book by holding a coordination(coordination1).

During coordination1, the participants refer the following information:

1. The m-group book_purchase1 is already being executed, and order_accept1 has already committed.

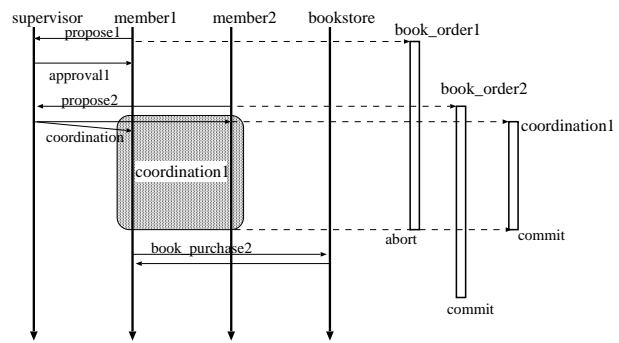


Figure 3. Event Diagram of Scenario 2

2. If book_purchase1 aborts, an exceptional situation occurs since order_accept has a dependency CO to book_purchase as in Figure 1(b).
3. In order to remove CO, an agreement with the bookstore is necessary. Moreover, the information of the removal should be informed to participants of the m-groups which share the resource in order_accept1, such as the log data of order acceptances.

As a result, the participants cancel propose2 of member2, the system aborts book_order2, and book_order1 continues its execution.

Scenario 2

Figure 3 illustrates the case where the proposal propose2 by member2 precedes the acceptance order_accept1 by the bookstore.

During coordination1, the participants refer the following information:

1. If book_order1 aborts, propose1 must abort.
2. In order to abort book_order1, the members of the m-groups which share the resource in book_order1 should be informed.
3. The rationale of executing book_order2. The participants of coordination1 refer the argument about creating the m-group book_order1 and book_order2.

Suppose that the conclusion of coordination1 is to choose book_order2. Then, the participants creates an m-group cancel as a child group of book_order1, and the system aborts book_order1. Moreover, book_order2 continues its execution.

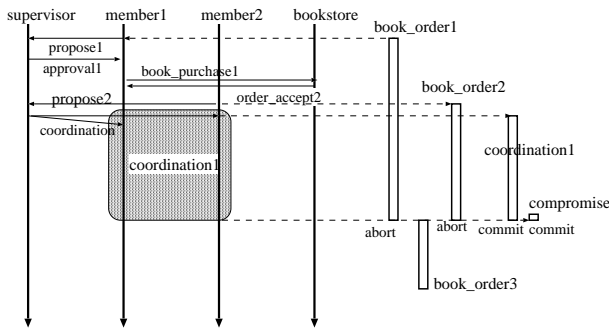


Figure 4. Event Diagram of Scenario 3

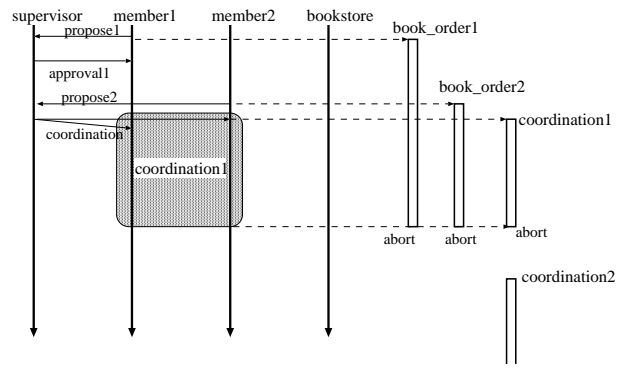


Figure 5. Event Diagram of Scenario 4

Scenario 3

Figure 4 illustrates the case where both `propose1` and `propose2` are not adopted in `coordination1`, and a new book is ordered as a compromise.

During `coordination1`, the participants refer following information in addition to the information in Scenario 2:

1. In order to adopt the compromise, the m-group compromise must be created, become a child group of `coordination1`, and commit. Moreover, the other conclusions `choose_one1` and `choose_one2` abort. Since this causes an exceptional situation around `EX` between `choose_one1` and `choose_one2`, `EX` must be removed.

As a result, the participants abort `book_order1` and `book_order2`, remove `EX`, and create a new m-group `book_order3` as the compromise. Moreover, when an exceptional situation occurs around `book_order3` later, someone may require the information that `coordination1` causes the aborts of both `book_order1` and `book_order2`, and the creation of `book_order3`.

Scenario 4

Figure 5 illustrates the case where an appropriate strategy is not found in `coordination1`, and `coordination1` is postponed.

During `coordination1`, the following information is referred in addition to those in Scenario 2 and 3:

1. The rationale of executing `book_order1` and `book_order2` is referred. The participants of `coordination1` refer the arguments about creating `book_order1` and `book_order2`.

Suppose that the participants of `coordination1` decide to postpone the conclusion. Then, the participants

abort `coordination1` and create a new m-group `coordination2`, which will be executed when the participants resume the postponed coordination, is created. Moreover, when an exceptional situation occurs around `coordination2` later, someone may require the information that `coordination2` is created as a result of the postponement of `coordination1`.

In the next section, we analyze functional requirements and model a framework for adapting transactions to exceptional situations by utilizing the message transaction model.

3 Modeling Adaptation of Transactions

3.1 Requirements for Supporting Exceptional Situations

The following observations can be obtained from the examples shown in the previous section:

1. **(Exception detection)** Detection of exceptional situation is divided to two categories: One is what a participant raises intentionally, such as the inductions of coordinations through Scenario 1 to Scenario4. The other is what subsequently occurs after other events, such as 2 in Scenario 1, and 1 in Scenario 3. In the latter case, a mechanism for detecting exceptions is necessary. We discussed the satisfiability problem of transactional dependencies in [8].
2. **(Invocation of communication)** In an exceptional situation, measures against the situations involve communication, such as a coordination or an agreement. Moreover, in some situations such as 3 in Scenario 1, the requirement is not to take the measure, but to know the area and the cost to take the measure.
3. **(Rationalization)** The rationale of an m-group is referred, such as 3 in Scenario 2, and as 1 in Scenario

4.

4. **(Interactive concurrency Maintenance)** When an m-group is to abort and the m-group uses a resource, the information should be informed or acknowledged to participants of the m-groups which share the resource, such as 1 and as 2 in Scenario 2.

5. **(Effect estimation)** When a transactional event is to occur, effects to other m-groups are estimated, such as 1 and 2 in Scenario 1. This means that a simulation mechanism for handling exceptions is required. Especially in subsequent exceptions such as the latter case in 1, avoiding cascade occurrences of exceptions by estimating costs is important.

3.2 Managing Exceptional Situations with Message Transactions

In the rest of this section, we model a framework for supporting adaptation of transactions to exceptional situations.

Suppose that S is an m-group, $e(S)$ is a transactional event of S , and P is a participant of S .

Exception Detection

We introduce the concept of *exceptional operation*, which is an operation causing an exceptional situation to occur. If an exceptional situation is raised directly by a participant, the operations by a participants are the exceptional operations. On the other hand, if raised subsequently after other events, the original operation and the subsequent events are the exceptional operations.

Invocation of Communication

The first step of this procedure is to decide *exception coordinators*, who are the participants to the invoked communication. The exception coordinators are the following participants:

- If the exceptional operations includes an transactional event $e(S)$, then the exception coordinators include the participants of each m-group which is an element of any transactional dependency whose elements include S , such as $CC(S_1, \dots, S, \dots, S_n)$.
- If an modification on a child group is included in the exceptional operations, then the participants of the child group are in the exception coordinators.
- If the exceptional operations includes an modification on a transactional dependency, then the exception coordinators include the participants of each m-group which is an element of the transactional dependency.

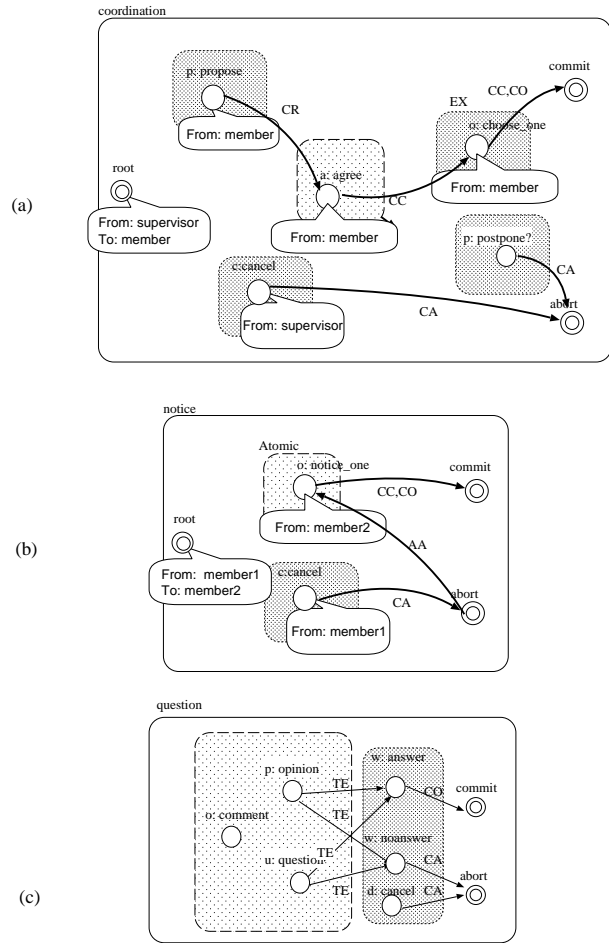


Figure 6. Examples of M-group Templates for Exceptional Situations

- If an modification on a resource of S is included in the exceptional operations, then the exception coordinators include the participants of S .

We describe system behavior for invoking communication for coordination.

1. The system displays the exception coordinators.
2. P selects a proper m-group S' to argue with the displayed coordinators. Figure 6 shows the examples of m-group templates.
3. The system creates S' , and adds the exception coordinators to participants of S' .
4. The system forbids the exceptional operations to work before the invoked communication committs.

Rationalization

We define $\text{producer}(S)$ for the purpose of detecting the argument for creating an m-group S . $\text{producer}(S)$ is a set of m-groups each element of which is the argument for creating S .

The followings are the mechanism of rationalization, where S' is an m-group created as the invocation of communication:

1. The system adds each element of an $\text{producer}(S)$ to a resource of S' . This operation enables S' to specify that the participants of S' may refer $\text{producer}(S)$.
2. When S' finishes its execution, the system adds S' to an element of $\text{producer}(S'')$, where S'' is an m-group which is created as a result or in course of S' .

Interactive Concurrency Maintenance

$\text{getSharing}(S)$ is a function for detecting m-group which shares a resource with S . $\text{getSharing}(S)$ is defined as a union of the following sets, where $e(S)$ is a transactional event on S :

1. A set of any m-group S' which has a resource R , where R is a resource of S , and S' has not finished its execution yet.
2. $\text{getSharing}(S')$ for S' in 1.

We describe system behavior for maintaining concurrency interactively among the transactions which share resources.

1. When an m-group S is to abort, the system displays the participants of each m-group which is obtained by $\text{getSharing}(S)$.
2. P selects a proper m-group S' to argue with the displayed participants, such as in Figure 6. P may select `notice` if the participants are less concerned with the shared resource, and may select `coordination`, if much so.
3. The system creates S' , and adds the members obtained in 1 to participants of S' .

Effect Estimation

$\text{getEffect}(S, e(S))$ is a union of the following sets, where $e(S)$ is a transactional event on S :

1. $\{e(S)\}$.
2. $\text{getEffect}(S, e'(S'))$, for any m-group S' assigned $e(S) \rightarrow e'(S')$.

This function provides users a primitive function for estimating effect of a transactional event $e(S)$. We plan to develop an advanced method for estimating effect in our future work, such as:

- Navigational queries as the effect simulation. For example, for a transactional event $e(S)$, the rationale and the participants of the m-groups which may share resources with S are retrieved.
- Temporal queries. For example, the range of the effect after an event is compared with those before the event.

The procedure described in this section enables participants to cooperate with other participants who get effected by entering the exceptional situation. Moreover, when they want to perform an exceptional operation, participants can be informed what to pay attention to.

4 Conclusion

We addressed the problem of handling exceptions in a collaborative work environment, utilized transactions specified in structured messages, and proposed a method for adapting the transactions to the situations which do not satisfy the dependencies of the environments.

The followings are the advantages of our method:

- Since the method is based on messages exchanged in communication, the method can easily incorporate the solutions obtained by human decisions to transaction management including exceptional situations.
- By introducing the concept of producer, the method provides an interface to utilize context of the arguments for the collaborative work, such as the rationale of executing an m-group. This leads to giving guidelines for the consistency maintenance to human decisions.
- By introducing the function **getSharing**, participants can interactively handle data violation from concurrent transactions.
- The method can apply to an distributed environment with heterogeneous information systems. Since the message transaction model can represent various dependencies in collaborative work environments by using transactional dependencies. Integrating with the order-constrained rules for workflow model and the dependencies for executing various advanced transaction models is possible.

References

- [1] P. Attie, M. Singh, A. Sheth, and M. Rusinkiewicz. Specifying and enforcing intertask dependencies. In *Proc. 19th Int'l Conf. Very Large Databases*, pages 134–145, 1993.
- [2] D. P. Bogia, W. J. Tolone, S. M. Kaplan, and E. de la Tribouille. Supporting dynamic interdependencies among collaborative activities. In *Proc. ACM Conf. Organizational Computing Systems*, pages 108–118, 1993.
- [3] J. Conklin and M. L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. In *ACM Trans. Office Information Systems*, volume 6, pages 303–331, Oct. 1988.
- [4] F. Flores, M. Graves, B. Hartfield, and T. Wingrad. Computer systems and the design of organizational interaction. In *ACM Trans. Office Information Systems*, volume 6, pages 153–172, April 1988.
- [5] J. N. Gray. The transaction concept: Virtues and limitations. In *Proc. 7th Int'l Conf. Very Large Databases*, Sept. 1981.
- [6] S. Inoue and M. Iwaihara. Structured message management for group interaction. In *Proc. Int'l Workshop. New Database Technologies for CSCW and Spatio-Temporal Data Management (NewDB'98)*, Singapore, Nov. 1998.
- [7] S. Inoue and M. Iwaihara. Dynamic reconfiguration of message structures for integrity maintenance in collaborative work. In *IPSJ Technical Report,99-DBS-119*, pages 255–260, July 1999.
- [8] M. Iwaihara, S. Inoue, and H. Matsuo. On unifying message and transaction management for collaborative design work. In *Proc. Int'l Symp. Digital Media Information Base (DMIB97)*, pages 300–304, Nara, Nov. 1997.
- [9] G. Joeris and O. Herzog. Managing evolving workflow specifications. In *Proc. of 3rd IFCS Int'l Conference on Cooperative Information Systems(CoopIS '98)*, New York, Aug. 1998.
- [10] S. M. Kaplan, A. M. Carrol, and K. J. MacGregor. Supporting collaborative process with conversationbuilder. In *Proc. ACM Conf. Organizational Computing Systems*, pages 69–79, 1991.
- [11] S. M. Kaplan, W. J. Tolone, D. P. Bogia, and C. Bignoli. Flexible, active support for collaborative work with conversationbuilder. In *Proc. ACM Conf. Computer-supported Cooperative Work*, pages 378–385, November 1992.
- [12] J. Klein. Advanced rule driven transaction management. In *Proc. COMPCON*, 1991.
- [13] L. Liu and C. Pu. Methodological restructuring of complex workflow activities. In *Proc. 14th Int'l Conf. Data Engineering*, pages 342–350, California, 1998.
- [14] K. Ramamritham and P. K. Chrysanthis. Advances in concurrency control and transaction processing. In *IEEE Computer Society Executive Briefing*, 1997.
- [15] M. Reichert and P. Dadam. ADEPT_{flex} – supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 1997.