Modelling a transaction manager using parallel decision processes.

Drs. S.C. van der Made-Potuijt
Erasmus University Rotterdam

Dr. L.P.J. Groenewegen
State University Leiden

November 1990

1 Preface

This is the second report concerning transaction management in the database environment. In the first report the role of the transaction manager in protecting the integrity of a database has been studied [MaNe]. In this report a model will be given for the transaction manager as a parallel decision process. To that purpose a modelling method for parallel behaviour, Paradigm, will be introduced. This method uses parallel decision processes for modelling parallel phenomena. Not all the notions of Paradigm will be discussed, a more extensive explanation can be found in [Groe]. After this short introduction to Paradigm, the method will be applied to a simple transaction manager. In future research more complex transaction managers will be modelled.

- 1984 CR catagories: G.1.0, H.2.4, H.2.7.
- 1980 Mathematics subject classification (1985 revision): 68N15, 68P15.
- Keywords & Phrases: Parallel Algorithms, Transaction Processing.

2 Paradigm.

2.1 Introduction.

Parallel decision processes have been used successfully in the realm of parallel programming. Formulating a parallel programming problem in terms of a parallel decision process, provides an object-oriented description of the problem. Solutions have been offered to several problems, including the critical section [Groe]. A model has been offered for the kernel of the Unix operating system and it has been implemented in an object-oriented manner [Stee]. This approach has been extended from parallel programming problems to general parallel phenomena. Parallel decision

processses can offer an object-oriented description for any parallel phenomena and these descriptions can be translated quite naturally into an object-oriented program.

In the database environment the object-oriented approach usually refers to the description of entities and the operations of the entities [BHGK]. Here object-orientedness is usually connected to a static description of the system: the organisation of the data and the possible operations. Not much attention is given to an object-oriented approach for the description of algorithms used for the database management. In this study we will focus on the algorithm of a transaction manager as a parallel decision proces in order to give an object-oriented approach for such an algorithm. This also underlines the importance of an object-oriented approach with respect to the dynamic properties of a system rather than the static ones. The notion of decision process is a key notion in the operational research [Hind]. Parallel decision processes and the modelling method for parallel behaviour are concepts developed in [Groe].

2.2 Decision Processes and Parallel Decision Processes.

The dynamic properties of a system can be brought into a decision process by modelling its behaviour in the following way. At each time instant the system is in exactly one state. When entering a state x, a strategy $\pi \in S$ selects an action $\alpha \in A$ based on the history h up to that instant. The transition mechanism F selects a next state y based on h and α. The sojourn mechanism G selects a sojourn time in state x based on h, α and y. The behaviour of a system is the sequence, summed up in chronological order of the states that are visited, the actions that are chosen and the time instants at which a transition occurs. The reward function makes it possible to compare behaviours. These are the descriptions of the seven components that have to be specified in a decision process:

1. T is a set of instants of time, ordered by a $<$ relation.
2. X is a nonempty set of states.
3. $A = \bigcup_{x \in X} A_x$ is a set of actions where for each $x \in X$ the set A_x is nonempty; this A_x is called the set of actions available in x.
4. S is a nonempty set of strategies.
5. F is a transition mechanism that determines which state of X will be entered from state x, given the history of the system up to now and the available action $\alpha \in A$.
6. G is a sojourn mechanism that determines how long it will take before the next state (selected by the transition mechanism) will be entered, given the history, the present state, the action chosen and the state to be entered.
7. The reward function, which maps the behaviour onto real numbers.

Definition 1 *A* decision process *is a tuple* $DP = (T, X, A, S, F, G, r)$ *with time space* T*, state space* X*, action space* A*, a set of strategies* S*, a transition mechanism* F*, a sojourn mechanism* G *and a reward function* r*.*

A formal definition can be found in [Groe]. These seven components make it possible to describe decision processes in detail . For many purposes it is not necessary to use all the components. For example the sojourn mechanism can often be neglected by stating that the sojourn times are always equal.

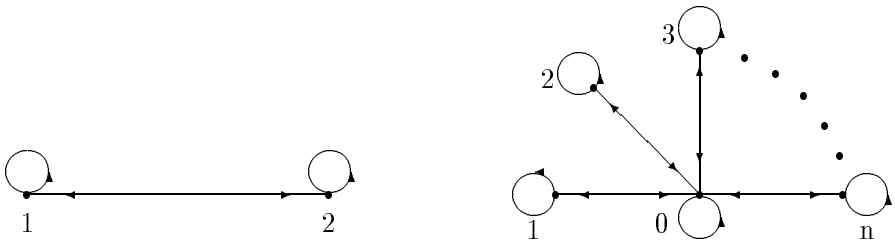
A decision process can model the sequential behaviour of a system. In order to model a set of decision processes including their mutual interaction the notion of a parallel decision process is necessary. This can be seen as a vector of decision processes, where the behaviour of one process can depend directly on the behaviour of other processes.

Definition 2 *A parallel decision process denoted as* $PP = [CP^1, CP^2, ..., CP^n]$*, is a construct where all* $CP^i, i \in 1, ...n$ *are decision processes, called constituent processes, with the same time space. The selection of actions, transitions and sojourn times in a constituent process can depend directly on the history of any set of the constituent processes up to that time instant.*

An important property of each parallel decision process is, that it is just a special case of a decision process. To be somewhat more precise on this point, a state of the parallel decision process PP is considered as the vector $(x^1, ..., x^n)$ where x^i is a state of CP^i. Because a parallel decision process is a decision process all properties of decision processes also hold for parallel decision processes. Parallel decision processes are often referred to as *parallel processes* and decision processes are often called *processes.*

A parallel process can be represented by a collection of graphs, each one representing a constituent process. States are denoted by numbers, transitions by directed edges. Each transition corresponds to a unique action in the state where the edge points from.

The definitions will be illustrated with the following example. We will describe a meeting of n people with one chairman. The processes are represented by the graphs in figure *A meeting.*



Process Member Process Chairman

Figure 1: A meeting

Remark: If transitions are possible from state x^i to state x^j and from state x^j to state x^i this is noted as one edge having two directions, representing the both transitions.

The Member process can be modelled by means of the following states and transitions:

1 The member is not speaking.

2 The member is speaking.

1→1 The member continues not to speak.

1→2 The member starts speaking.

2→2 The member continues to speak.

2→1 The member stops speaking.

The chairman can be modelled by means of the following states and transitions:

0 The chairman is allowing no one to speak.

i The chairman is allowing member(i) to speak.

0→0 The chairman continues giving no one permission to speak.

0→i The chairman gives member(i) permission to speak.

i→i The chairman continues allowing member(i) to speak.

i→0 The chairman withdraws member(i)'s right to speak.

T has not to be specified, it can be the the set of natural numbers. For each constituent process, the set of states, X is specified (mostly only two states). The set of actions can be determined by the table specifiying the transitions. S has been specified indirectly: each sequence of available actions may be selected, as if there exists no special dependency between the Chairman and the Member processes. The transition mechanism has been indicated. The sojourn mechanisme has not to be specified as all the sojourn times are supposed to be equal. Because it is not necessary to compare behaviour we will not define reward functions. So only the state space and the transitions are represented, specifying actions and strategies too. Each member can be modelled as a decision process, the chairman can be modelled as a seperate decision process with the same time space. Therefore the Chairman process together with the n Members process forms a parallel process.

2.3 Subprocesses and trap processes.

Dependencies between constituent processes imply that processes communicate with each other. In general when a process CP^1 is dependent on the behaviour of process CP^2 and no communication has yet taken place, CP^1 will be restricted in its behaviour from a certain point on, until the communication has taken place. In our example the behaviour of a member has to be restricted by the behaviour of the chairman. These behaviour limitations are modelled by means of so called *subprocesses.*

Definition 3 *Let $DP = (T, X, A, S, F, G, r)$ be a given decision process and let SP be a decision process $SP = (T, X, A, S'[X', A'], F, G, r)$ where $S'[X', A']$ is defined as follows:*

1. $\emptyset \neq X' \subseteq X$.
2. *For each state $y \in X'$, $A'(y) \subseteq A(y)$ is a non-empty set of available actions in y such that the transition corresponding to such an action is always to a state $z \in X'$.*
3. *For each strategy $s \in S'[X', A'] \subseteq S$ and for each $y \in X'$, only actions from $A'(y)$ are selected by strategy s in state y.*

Then SP is called a subprocess *of DP with proper state space X' and a proper action space $A' = \bigcup_{y \in X'} A'(y)$. A state $x \in X'$ is called a proper state. An action $\alpha \in A'$ is called a proper action.*

This definition will be illustrated with an extended example of the Meeting. This second model represents a refinement of the Member process in the previous situation. The new member process, called the Active Member process can be modelled by means of the states and transitions in figure *The Active Member(i).*

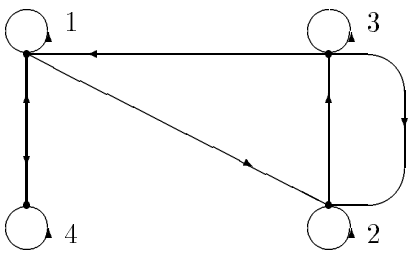


Figure 2: The Active Member(i)

1 The member is neither speaking nor expressing a wish to speak.

2 The member is raising his/her finger, indicating he or she wants to say something.

3 The member is speaking.

4 The member is indicating he or she has finished speaking.

1→1 The member continues neither to speak nor to express such a wish.

1→2 The member raises his/her finger.

2→2 The member keeps his/her finger raised.

2→3 The member starts speaking.

3→3 The member continues to speak.

3→2 The member renounces from speaking, but he or she immediately raises his/her finger.

3→1 The member stops speaking.

1→4 The member starts giving a sign to the chairman he or she renounces from speaking

4→4 The member continues to indicate he or she does not want to speak.

4→1 The member stops signing the chairman.

The Active Member Process can be divided into four subprocesses, I, II, III and IV.

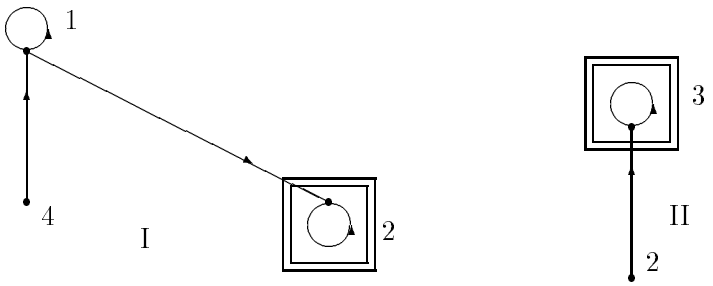


Figure 3: The subprocesses I and II of Active Member(i)

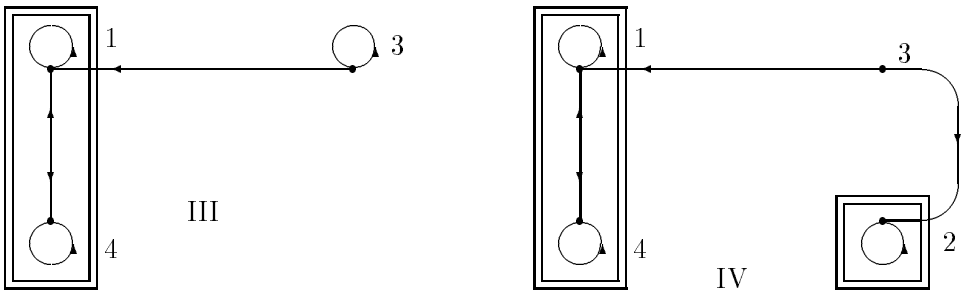


Figure 4: Subprocess III and IV of Active Member(i)

In representing a subprocess graphically, we shall always restrict ourselves to its proper states and the transitions corresponding to its proper actions.

I. The member has no permission to speak.

II. The member has permission to start speaking and to speak.

III. The member has permission to speak and to stop speaking.

IV. The member's permission to speak is withdrawn. If the member still wants to say something and therefore raises his/her finger a transition to state 2 has taken place, otherwise a transition to state 1 or 4.

In the next definition certain sets of subprocesses, that play an important role in the communication among processes, are dealt with.

Definition 4 *Let V be a collection of n different subprocesses SP^i of the decision process DP. Denote by X^i and A^i respectively the proper state space and proper action space of subprocesses SP^i, and let $\bigcup_{i=1}^{n} X^i = X$ and $\bigcup_{i=1}^{n} A^i = A$. Then V is called a partition of DP.*

So V = { I, II, III, IV } is a partition of the process Active Member(i).

It is easily seen that a subprocess actually is a restriction of the original process' full behaviour. The idea now is that such a behaviour restriction is (temporarely) prescribed by some other, as we shall see, constituent process. Let us assume that a decision process is inside the proper state space of its currently prescribed subprocess. As long as it obeys to this prescription, it cannot leave this proper state space. On the other hand, the decision process may wish to start behaving according to some other (prescribed) behaviour restriction. In order to express such a wish, the notion of trap is introduced.

Definition 5 *A trap of a subprocess DP, is a nonempty subset Y of the proper state pace of DP such that for each strategy π and for each state $y \in Y$, any selected proper action $\alpha \in A(Y)$ will always lead to a transition to a state $z \in Y$.*

Observe that a trap together with the relevant restrictions of the proper action space and the corresponding set of strategies, is a subprocess within a subprocess. In our example a trap of I is {2} , the set {3} is a trap of II and the set {1,4} is a trap of III. The sets {1,4} and {2} are traps of subprocess IV.

The notion of a *trap structure* makes it possible to control the changeover of one subprocess to other subprocesses.

Definition 6 *Let V be a partition, consisting of n different subprocesses SP^i of the decision process DP. Let TS be a set of subsets of the state space of DP with the property that for each subset $Y \in TS$, there exist subprocesses SP^i and SP^j such that Y is a trap of subproces SP^i and also that Y is a subset of $X^i \bigcap X^j$. Then TS is called a* trap structure *for the partition V, and Y is called a* trap from SP^i *to* SP^j.

In the Active Member process we can choose a suitable trapstructure TS for partition V with the following elements:

- Trap of I to II : { 2 }
- Trap of II to III : { 3 }
- Trap of II to IV : { 3 }
- Trap of III to I : { 1, 4 }
- Trap of IV to I : { 1, 4 }
- Trap of IV to II : { 2 }

The control of communication can be modelled by a seperate decision process, *a trap process*. A trap process is defined with respect to a decision process and a trap structure.

Definition 7 *Let DP be a decision process with partition V and trapstructure TS. Then, a* trap process of DP with respect to V and TS *is a decision process T^*, with state space the set V such that the following dependencies hold:*

1. *Whenever T^* is in state $v^i \in V$, the decision process DP will behave according to subprocesses v^i*
2. *after DP is trapped in a trap $S^{i,j} \in TS$, with $S^{i,j}$ a trap from v^i to v^j, T^* can transit to state $v^j \in V$, thereby forcing DP to behave according to subprocess v^j.*

Notice that the trap process takes the decision, chooses the action corresponding to a transition to another state, under the restriction that DP has been trapped. In this way the trap process controls the transitions between the subprocesses defined by the partition V and the trapstructure TS. We say that the parallel decision process $[DP, T^*]$ is a *control process* of DP with respect to V and TS. Because the trap process is one of the constituent processes of the control process (which is a parallel decision process) we will illustrate the definitions after we have introduced parallel subprocesses and trap processes.

2.4 Parallel subprocesses and trap processes.

The notions of subprocess, partitions and trap structures, trap processes and control processes as given above, can easily be extended to parallel processes. The strength of the next theorems is, that the notions for the parallel processes are of the same nature as for non-parallel processes. All theory, not presented in this report, that applies for decision processses will therefore also hold for parallel decision processes. Furthermore it allows for a uniform description of both types of processes.

Let PP be a parallel process denoted as $[CP^1, CP^2, ..., CP^n]$, where each $CP^k, k \in 1, ...n$ is a constituent process. For each $i \in 1, ..n$ let V^i be a partition of CP^i, with elements the subprocesses $CP^{i,1}, ..., CP^{i,m(i)}$ and $m(i)$ a positive integer. In [Groe] it has been shown that:

Theorem 1 *Each parallel decision process* $[CP^{1,j_1}, CP^{2,j_2}, ..., CP^{n,j_n}]$ *is a subprocess of* PP*, with* $j_k \in 1, .., m(k)$.

Theorem 2 *The set* V *defined as*

$$V \equiv [CP^{1,j_1}, CP^{2,j_2}, ..., CP^{n,j_n}]\ j_k \in 1, ..m(k),\ k \in 1, ...n$$

is a partition of the parallel decision process PP*.*

The partition V is said to be generated by PP and V^i , $i \in 1, ..n$.

Theorem 3 *For a parallel decision process* $PP = [CP^1, CP^2, ..., CP^n]$ *where* V^i *is a partition of* CP^i*, and* TS^i *a trap structure for the partition* V^i *for each* $i \in 1, .., n$ *the set* TS *defined as* $\prod_{i=1}^{n} TS^i$ *is a trap structure for the partition* V*.*

The trap structure TS for the partition V is said to be generated by PP and TS^i, $i \in 1, .., n$. Because parallel processes are also decision processes trap processes can be usefull for parallel processes. The structure of a trap process for parallel processes, indicated in the next corollary, is implied by theorem 3.

> **Corollary** *If in the situation as described in Theorem 3 the process* T^i *is a trap process of* CP^i *for* V^i *and* TS^i*, then the decision process* $T^* = [T^1, T^2, .., T^n]$ *is a trap process of* PP *for* V *and* TS*.*

A parallel control process will be defined as a trap process together with a parallel process.

Definition 8 *If* PP *is a parallel decision process* $[CP^1, CP^2, ..., CP^n]$ *and* T^* *is a trap process with respect to a given partition* V *of* PP *and given trap structure* TS *for* V *generated by* $V^1, ..., V^n$ *and* $TS^1, ..., TS^n$ *respectively, then the control process* $[PP, T^*]$ *is* a parallel control process *of* $CP^1, CP^2, ..., CP^n$ *with respect to* V *and* TS*.*

The parallel control process will be denoted as $[CP^1, CP^2, ..., CP^n, T^*]$

These theorems will first be illustrated with the example of The Meeting as described in figure 1. The Member process (not the Active Member Process) can be partitioned in the subprocesses I and II. Subprocess I reflects the situation that the

Figure 5: The partition of the Member process

member is speaking, subprocess II reflects the situation a member is not speaking. So V^i= [I ,II] is a partition for Member (i). We choose as a trapstructure TS^i state 2 as a trap of I to II and state 1 as a trap of II to I.

To be able to describe the control process for the individual members, we need to describe the trapproces. Let T^i be the trap process of Member with respect to V^i and TS^i. The trapproces T^*, defined as $[T^1, T^2, .., T^n]$ is the trapprocess with respect to the partition of all the Member processes and trapstructures. The state space of this control process T^* is $X_{T^*} = \prod_{i=1}^{n} \{I, II\}$, representing 2^n n-tuples with constituing values I and II. Not all the states of this control process T^* are acceptable however, as this would allow several members to be in subprocess I at the same time, meaning more than one member has the right to speak. Therefore we will restrict the state pace of T^* to be the set of the n-tuples having maximal one value equal to I. This set consists of exactly $n + 1$ n-tuples.

The $n + 1$ states of the chairman correspond exactly to this trapproces: n states describe the states in which exactly one member is allowed to speak, state 0 indicates no members are allowed to speak. Because of this correspondence between the Chairman Process and the trapprocess T^*, the Chairman Process can play the role of the control process: no extra control process is necessary. This parallel control process can be denoted as $[Member^1, Member^2, ..., Member^n, Chairman]$.

In the Active Member process however the Chairman process as described is not able to control all transitions. For example, it cannot distinguish between the transitions from subprocess II to III and from II to IV, whether a member renounces from speaking or that the right to speak has been withdrwan from the member. Therefore the Chairman process has to be refined. This Refined Chairman process is indicated in figure 6. Only the parts of the behaviour of the chairman towards member(1) and member(n) are represented. Notice that the Refined Chairman process is not in only one state of this graph, but in one for every member because for all the members it has to be indicated to what subprocess their behaviour has to be restricted.

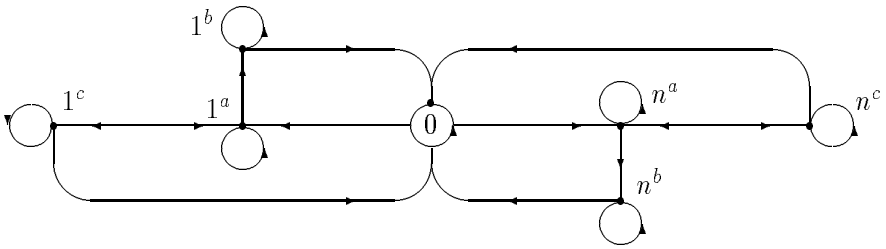


Figure 6: A part of the Refined Chairman process

The Refined Chairman can be modelled by means of the following states and transitions:

0 The chairman is allowing no one to speak, all the active members are restricted in their behaviour to subprocess I.

i^a The chairman is allowing active member(i) to start speaking, this active member is in subprocess II.

i^b The chairman is allowing member(i) to speak or to stop speaking if he or she is finished. The active member is in subprocess III.

i^c The right to speak has been withdrawn from member(i), this member is in subprocess IV.

$0 \rightarrow 0$ The chairman continues giving no one permission to speak.

$0 \rightarrow i^a$ The chairman gives member(i) the permission to speak.

$i^a \rightarrow i^a$ The chairman continues allowing member(i) to speak.

$i^a \rightarrow i^b$ The chairman is giving permission to the member to speak until he or she is ready with speaking.

$i^b \rightarrow i^b$ The chairman continues to allow member(i) to speak, or to stop speaking.

$i^b \rightarrow 0$ After the member has indicated he or she has stopped talking, the chairman withdraws the right to speak of this member.

$i^a \rightarrow i^c$ The chairman withdraws the right to speak of the member, whether he or she was ready or not.

$i^c \rightarrow i^c$ The chairman continues not to allow member(i) to speak. (If the right to speak of this member was withdrawn, but the member wanted to continu because he or she immediately raises his/her finger again, the chairman can allow him/her to speak again.)

$i^c \rightarrow i^a$ The chairman gives this member permission to speak as he or she had his/her finger raised.

$i^c \rightarrow 0$ After having indicated to the chairman the member agreed that his/her right to speak has been withdrawn, the behaviour of the member is restricted to subprocess I.

Since each Member consists of four subprocesses, a trap process for n Member processes has a state space containing 4^n different states, each state being an n-tuple prescribing one subprocess combination. However, most combinations are not to occur. The only relevant combinations are

- each Member in I: (I, I,..., I)
- exactly one Member in II, or in III or in IV:
  (I,..., I, II, I,...,I) ; (I,..., I, III, I,...,I) ; (I,..., I, IV, I,...,I);.

This results in $1 + 3n$ different states, exactly corresponding to the $1 + 3n$ different states of the Refined Chairman. So the Refined Chairman can play the role of trap process and the parallel control process can be denoted as
$[Active\ Member^1, Active\ Member^2, ..., Active\ Member^n, Refined\ Chairman]$.

3 Paradigm applied to simple transaction manager.

3.1 Introduction and definitions

In this section it will be shown that transaction managers for database management systems can be modelled using parallel decision processes. As a first, rather simple example we will consider a model for a single user environment. First we will present the used definitions, then a description of the transaction manager to be modelled will be given.

Definition 9 *The state of a database system is determined by data items and devices that have changeable values. There also exists a number of integrity constraints referring to these values. A database is in a* correct *state, if it is consistent and no integrity constraints referring to values in the database are violated.*

Definition 10 *The dynamic constraints determine wich transformations in the database states are permissable. A* transaction *is a permissable transformation of one correct state of the database to another correct system state.*

Definition 11 *A database management system is the collection of software modules that support the commands to access the database.* The transaction manager *is the part of the database management system that forms the interface between the user and the database system.*

If a transaction is submitted to the transaction manager, this manager should be able to bring the database from one correct state to another correct state according to the transaction. To this aim it supports the three basic operations of a transaction: Start, Commit and Abort. The transaction manager is responsible for concurrency control and recovery. Often it can also influence the buffer and cache manager of the operating system.

3.2 A model of a simple transaction manager

The transaction manager we describe is restricted to a single user environment to avoid complications due to concurrency. Although the DBMS is assumed to be single user, the CPU and main memory may be shared with other processes.

The transaction manager uses a log for recovery purposes. The logbuffers are written to stable storage periodically. Before the database buffers are written to secundary storage, all logbuffers pertaining information of these data are also transferred to stable storage. The recovery module does not use checkpoints.

The so-called immediate update protocol is used. This implies it can be neccesary to undo transactions that are rejected because some of the database buffers containing information about the new situation, may already have been written to secondary storage.

If a transaction is executed succesfully it will be called partially committed. This information is written in the log, but the logbuffers may not be written to stable storage yet. If the logbuffers are written to stable storage, the information concerning the transaction survives a system failure and a redo of the transaction can

take place. If however the logbuffers containing the information that the transaction has reached its partial commit are not written on stable storage, the transaction has to be undone in the process of recovery.

In figure 7 some situations are shown to illustrate the different operations the transaction manager must be able to perform.

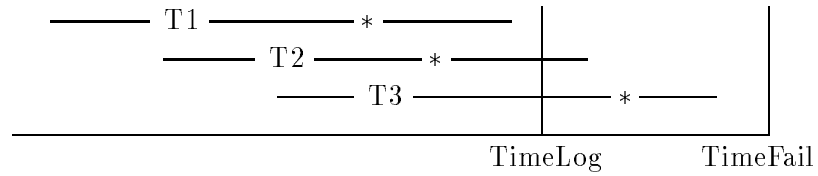


Figure 7: The states of transactions involved in recovery

The interpretation of this figure is as follows: A star, $*$, indicates a transaction has reached its partial commit, the end of a line indicates a commit. Both moments are registered in the log. The moment the logbuffers are written to stable storage is indicated by TimeLog and the moment of a system failure by TimeFail.

- Transaction T1 is completely succesfull: it has reached its commit, the logbuffers and the database buffers are written to stable resp. secundary storage. T1 will not be involved in the recovery process.
- Transaction T2 can be redone after the failure: it had reached its partial commit and the information has survived the system failure. For a redo it is not necessary to undo the transaction and execute it again because we are in a single-user environment: the transaction cannot have read incorrect values of other transactions.
- Transaction T3 has reached its partial commit after the logbuffers were written to stable storage. So it has to be made undone in the process of recovery.

3.3 A paradigm model for the transaction management

It will be shown that the state of a database during the execution of a transaction can be modelled as a parallel decision process with two constituent processes, representing the behaviour of the database during the execution of the transaction (DDE) and the transaction manager (TM).

For both processes only the state space and the transitions will be represented, specifying actions and strategies too. Because we do not want to compare behaviour we will not define reward functions. Time is kept implicit. The decision process DDE can be graphically represented as the directed graph in figure 8.

DDE can be modelled by means of the following 4 states:

1 No transaction has been submitted to the transaction manager. The database is in a correct state. If the transaction has been committed this is also reflected by state 1, in that case we say a new transaction has not been submitted yet.

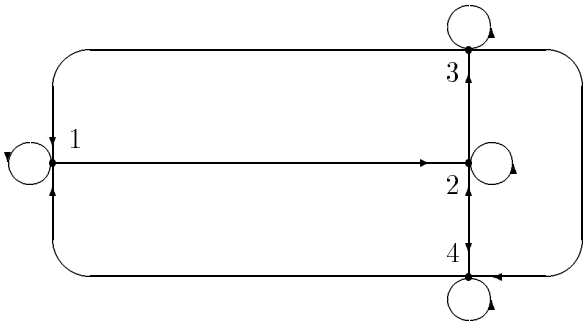


Figure 8: The behaviour of a database during the execution of a transaction

2 The transaction is being executed.

3 The transaction is partially committed.

4 The transaction is being rejected.

The interpretations of the transitions between the states is as follows:

1→1 The new transaction has not been submitted yet to the transaction manager.

1→2 The transaction is submitted to the transaction manager.

2→2 The transaction is being executed.

2→4 The transaction is rejected for some reason.

2→3 The transaction enters the state of being partially committed.

3→3 The transaction remains partially comitted.

3→4 The transaction cannot be committed.

3→1 The transaction can be committed and the transaction manager is ready with its task.

4→4 Information is gathered to support the decision whether to redo the transaction or to undo it.

4→1 The transaction will be undone.

4→2 The transaction will be redone.

The process DDE can be divided into four subprocesses:

I The transaction can be or has been submitted to the transaction manager or that a transaction can be submitted. This can be for the first time or because of a redo operation.

II The transaction has been executed succesfully, a partial comit has been reached and the transaction is waiting for the reply of the transaction manager.

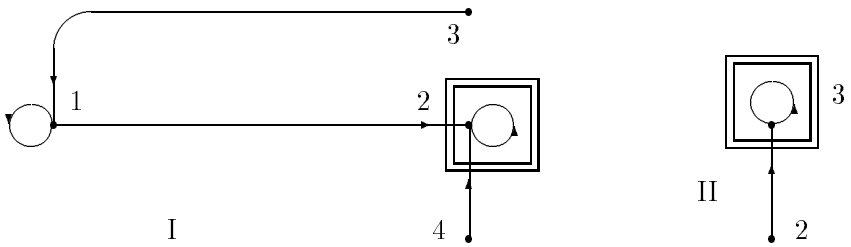


Figure 9: The subprocesses I and II

III The transaction manager has decided that the transaction has to be rejected.

IV The rejected transaction has to be made undone. (It is not the transaction manager who decides whether the rejected transaction will be executed again.)

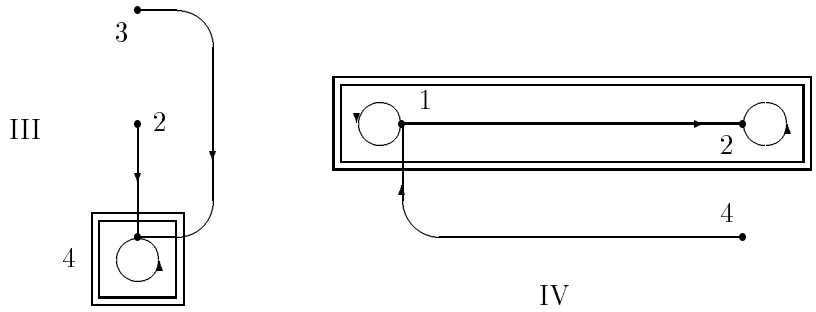


Figure 10: Subprocess III and IV of DDE

So V = { I, II, III, IV } is a partition of the process DDE. We choose a suitable trapstructure TS for partition V with the following elements:

- Trap of I to II : { 2 }
- Trap of I to III : { 2 }
- Trap of II to I : { 3 }
- Trap of II to III : { 3 }
- Trap of III to I : { 4 }
- Trap of III to IV : { 4 }
- Trap of IV to I : { 1, 2 }
- Trap of IV to II : { 2 }
- Trap of IV to III : { 2 }

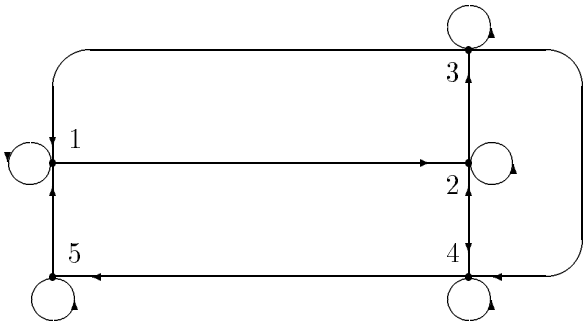


Figure 11: The transaction manager

The decision process representing the transaction manager as described in section **3.2** can be graphically represented as a directed graph in the figure *The transaction manager.*

Notice that this graph corresponds to a large extent with the graph representing the behaviour of a transaction. The only exception is state 5 where the transaction manager undo's a transaction. The reason for this similarity is, that in a single user environment the transaction manager can be dedicated completely to one transaction at the time. The interpretations of states and the transitions between the states is as follows:

1 Waiting for a transaction to be processed.

2 Executing the transaction.

3 The transaction is partially committed, the transaction manager prepares a commit (e.g. writing the log-buffers to stable storage and the db-buffers to disk).

4 The transaction has to be rejected, the transaction manager decides whether to undo or redo its results.

5 The transaction is made undone.

1→1 Waiting for a transaction to be processed.

1→2 A transaction has been submitted to the transaction manager.

2→2 The transaction manager is executing the transaction.

2→4 The transaction has to be rejected for some reason. It is still unclear whether the results have to be undone or that the transaction can be submitted again for execution.

2→3 The transaction has so far been executed successfully and has to enter the state partially-committed.

3→3 The transaction manager is preparing for a commit.

3→4 The transaction cannot be committed.

3→1 The transaction can commit and the transaction manager is ready with its task.

4→4 Information is gathered to support the decision whether to redo the transaction or to undo it.

4→5 The transaction must be undone.

4→2 The transaction can be redone.

5→5 The transaction is being made undone.

5→1 The transaction manager is finished.

From the two meeting examples and from other experiences, one can consider the transaction manager as a useful candidate for the trapprocess. To this aim we first indicate a correspondence between each state of the transaction manager and a subprocess of DDE. This results in the following combinations of such a state and such a subprocess, together with their interpretation. In this interpretation we also mention the various transitions from one combination to a following.

(1, I) The TM is ready to receive a transaction. If a transaction is submitted, a transition to (2, I) will follow.

(2, I) The TM is executing a transaction. If the execution is successfull to the extent that the transaction enters the state of being partially committed, a transition to (3, II) (TP in state 3) occurs. If it does not finish succesfully a transition to (4, III) takes place.

(3, II) The transaction has reached its partial commit and the TM is checking whether a commit is possible. A transition to (4, III) will follow if the transaction manager cannot commit the transaction, in case of success a transition to (1, I) will follow.

(4, III) The transaction has been rejected and the transaction manager is deciding wether to redo or undo the transaction. A redo implies a transition to (2, I), an undo implies a transition to (5, IV)

(5, IV) Indicates an undo takes place: a return to (1,I) will follow.

The list of these combinations, together with the various transitions between them can be graphically represented as follows, denoted as *The candidate trap process*.

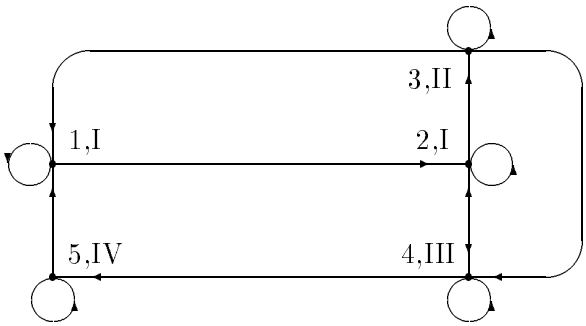


Figure 12: The candidate trap process

From the strong resemblance of the graphical representations of the transaction manager and the candidate trap process, we indeed give the transaction manager this trap process role. So it can control the transitions of the database during the execution of a transaction to other subprocesses, no extra trap process is necessary. This means that the transaction manager, called TM, and the database during the execution of a transaction, called DDE, can be modelled by the parallel control process [DDE, TM].

We now formulate the following conclusions. The DDE and the transaction manager both are relatively simple, but the corresponding Paradigm model of this situation is correspondingly simple. Therefore, this example gives us hope that also in more complicated and more realistic situations a Paradigm model of the various processes involved will result in a clear and precise specification thereof.

References

[Groe] L.P.J.Groenewegen, *Parallel Phenomena, a series consisting of technical reports. 1986-1990* Department of Computer Science, University of Leiden.

[Hind] K.Hinderer, *Foundations of non-stationary dynamic programming with discrete time parameter, 1970*.Springer, Berlin. Lecture Notes in Operation Research and Math. Econ. 33.

[Stee] M.R. van Steen, *Modelling Dynamic Systems by Parallel Decision Processes. 1988* Thesis, Department of Computer Science Leiden.

[MoGr] J.A. Morsink and L.P.J. Groenewegen, *Behavioural-oriented modelling if structural restrictions. 1990* Department of Computer Science, University of Leiden no 90-01.

[Gray] J.Gray, *The transaction Concept: Virtues and Limitations.1981* Lecture Notes in Computer Science, Conference Proceedings, The Netherlands 1981.

[MaNe] S.C. van der Made-Potuyt, *The necessity for check-on-commit in the protection of the integrity of a database, 1989* Department of Computer Science, Erasmus University Rotterdam, Report EUR-CS-89-05.

[MaTr] S.C. van der Made-Potuyt, *The transaction concept and its implementation. 1989* Proc. Benchmarking Database Functionality. Codd & Date Ltd., Berlin.

[VBD] F.Velez, G.Bernard and V. Darnis. *The O2 Object Manager: an Overview 1989* Proc. 15-th VLDB Conf., Amsterdam, pp 357-367.

[BHGK] J.Banerjee, Hong-TaiChou, J.F.Garza, W.Kim, D.Woelk and N. Ballou. *Data model Issues for Object-Oriented Applications. MCC and Hyoung-Joo Kim, 1987* ACM transactions on Office Information Systems 5(1): 3-26.

[Rish] T. Rish. *Monitoring Database Objects. 1989* Proc. 15-th VLDB Conf., Amsterdam, pp 445-453.