On Handling Exceptions

Heikki Saastamoinen
Department of Computer Science
University of Colorado
Campus Box 430, Boulder CO 80309-0430
USA

Department of Computer Science and
Information Systems, University of Jyväskylä
P.O.Box 35, SF-40351 Jyväskylä
Finland

George M. White
Department of Computer Science
University of Colorado
Campus Box 430, Boulder CO 80309-0430
USA

Computer Science Department
University of Ottawa,
Ottawa K1N 6N5
Canada

ABSTRACT

The current literature of information systems has dealt extensively with all kinds of exceptions. There are several studies defining the concept of exception and even providing classifications. However, no studies provide a method for verifying the rules in order to handle exceptions and to achieve the goals set by an organization's rules. In this paper, a model employing a set of unique input/output (UIO) sequences is presented for verifying such rules. The model originally presented for Finite State Machines (FSM) has been modified to include concepts of exception handling and will be used to form a tool usable for verifying exception handling rules in OISs.

INTRODUCTION

Exceptions form an essential part of the behavior of offices [1] and they are a major component of office work [24]. When concepts of office and office work are defined, exceptions are often pointed out as a feature characterizing those concepts [1, 8, 14, 24]. Empirical studies point out that exception handling can sometimes take almost half of the total working time, and that the handling of, and recovering from, exceptions is expensive [20, 23].

Despite this, only a few OIS design methodologies, *e.g.*, [8, 9, 15] consider exceptions at all. Even these studies do not provide tools for exception handling in OISs. Even though there are many studies of exceptions see, for example [1-7, 10, 11, 13, 14, 16, 19-21, 23, 24] , they are more likely to focus on the general nature of exceptions or on some specific aspect related to them.

The concept of exception is closely related to the concept of rule [19]. Rules can be viewed as instruments of policies aimed at solving problems [26]. In a broad sense, *rule* is a general term including concepts such as precepts, regulations, rules of thumb, conventions, principles, guiding standards and even maxims [26]. Thus all office information systems can be viewed as rule based systems [19, 27]. *Exceptions* can then be defined as events for which no applicable rules exist [1, 21, 19].

Office work is highly goal oriented [25, 27]. Rules stating, for example, that all ledgers must always be kept in balance, that all incoming invoices are to be handled so that possible cash discounts can be achieved, that all received stock items have to be reported to an information system within twenty-four hours, are examples of definitions of such goals.

We can now consider an exception raised during invoice auditing where an invoice that partially includes items mentioned in a corresponding order, but also refers to products not found on the order. It is now highly likely that the rules of an organization do not define desired actions for this kind of a situation. In this case, an office faced with such an exception is probably familiar with the normal case and is well aware of the goal to be achieved: the real amount due must be determined. However, there is a remarkable gap between the real situation and the goal.

The literature of exception handling is well aware of this kind of exception. There are several studies defining the concept of exception and even providing classifications. However, no studies provide a method for verifying the exception handling rules and for achieving the goals set by an organization's rules. In this paper, a model employing a set of unique input/output (UIO) sequences is presented for constructing such rules. The model, originally presented by Rezaki, Ural and White [18] for Finite State Machines (FSM) is modified to include concepts of exception handling and can be used to form a tool usable for verifying exception handling rules in OISs.

In the next section, an overview of the nature of exceptions is presented. The third section focuses on the exception handling by considering the general process of event handling. In the fourth section a detailed discussion of organizations' rule bases is provided. This discussion is complemented by a section focusing on the dynamics of rule bases. The sixth section discusses the checking algorithm and goal verification. The seventh section presents a real world example followed by this study's conclusions.

COOCS 95 Milpitas CA USA © 1995 ACM 0-89791-706-5/9 5/08..$3.50

THE NATURE OF EXCEPTIONS

Since there are papers going into great detail on this matter, the discussion on the nature of exceptions presented in the following is only an overview of the phenomenon. For further details, please refer especially to [22]. Other papers [1, 19, 21, 23] address the real nature of exceptions as well.

The basic characteristic of exceptionality is the degree of difference when compared to the corresponding normal case. This dimension of exceptionality was first discussed by Auramäki and Leppänen [1] and was further defined in [19, 23]. This characteristic is derived from the rule base. It is important to see the difference between the existence of a single rule and the existence of a complete event handling rule. The latter guides the handling of an event as such, while the former is actually only apart of it.

An *established exception* is an event where appropriate event handling rules basically exist, but the rules of the organization are incomplete and they cannot pinpoint the exact set of rules to be applied.

An *otherwise exception* is an event where the organization has no applicable rules. However, the organization has rules for the handling of corresponding normal cases and knows the goal to be achieved as a result of handling this kind of exception.

A *true exception* is an event so unanticipated, that the organization has not been able to prepare for it at all. In other words, the organization can only recognize the case as an exception and knows neither the corresponding normal case nor the specific goal or state to be achieved as a result of handling such an exception.

Another important dimension of exceptions is their effect on organizations' rules. Like exceptionality, this dimension was first discussed by Auramäki and Leppänen [1] and further defined in [19, 23]. By using this criteria, three kinds of exceptions can be distinguished:

Exceptions with no effect on the rule base: the handling of such exceptions does not change the rules of the organization in any way.

Exceptions causing instance level updates: such updates are strongly related to the precise event to be handled, *e.g.*, to a specific incoming invoice and have no effect on other events handled or to be handled in the future.

Exceptions causing type level updates: such exceptions cause updates to rules applied to certain types of events. An example of this kind of update would be the setting up of new rules for handling incoming invoices from some specific company.

In addition, exceptions differ, *e.g.*, by their primary sources, acceptability, laboriousness, frequency, organizational influence area, and handling delays. Those dimension of exceptionality are beyond the scope of this paper and are not further discussed here.

THE HANDLING OF EXCEPTIONS

Office work is processed by office actors by handing *events* as they arrive. Here, we define an *event* [21] as a detected phenomenon that is to be handled by the information system. These may be high level events such as a change in the environment in which the organization is immersed or a low level event such as the arrival of a purchase order in the sales office. High level events are typically dealt with by the higher levels of the organization while the lower level events are handled by the office staff, usually in a predictable way within a well ordered structure.

Events are said to be handled by first identifying the event and then by selecting the appropriate procedures to process it. This may be very easy in the case of routine events that can be handled by clerical staff or it maybe very difficult and complex requiring long study and hard decisions by the senior levels of management. The basic mechanism for handing events is shown by the Petri-net of figure 1.

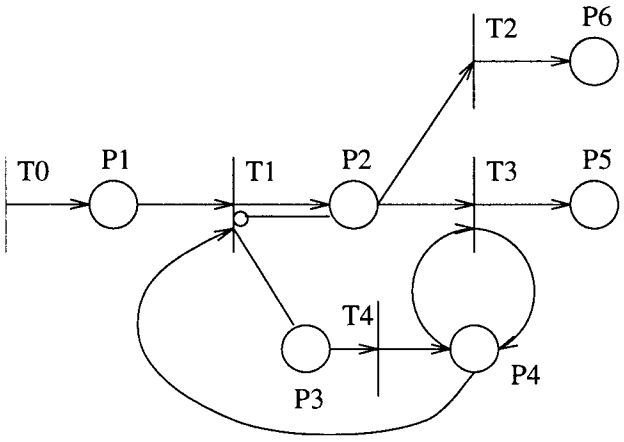

Fig. 1. The basic structure of event handling.

The event is first analyzed to see whether it is an instance of a recognized type. If it is, it is handled directly and dismissed. If not, it is analyzed to determine the degree by which it differs from a recognized type. This version of Petri-net is sometimes called the predicate/transition net, [12] augmented with explicit time dependencies as discussed in [17]

- T_0 models the entrance of events into the system. They are generated at random and have varying characteristics.
- T_1 takes these events in some order when the information system is ready to treat them.
- T_2 is a recognizer which analyzes the properties of the event and if it recognizes the event type, it handles the event according to the contents of its rule bases. If it cannot recognize the rule, it does not fire and does not handle the event.
- T_3 fires if T_2 has not previously fired, *i.e.*, T_2 has a priority over T_3 not visible in figure 1. In so doing it accepts the event, not recognized by T_2 and therefore an exception. It then either creates a new rule to handle the event or dismisses the event as unhandleable. In both cases, it passes the token to P_5 either as a processed event or an unprocessed one to be resubmitted.

- T_4 is a timed event, absorbing the token on P_3 and emitting a token to P_4 after some delay. It is the only transmission which does not fire instantaneously.

The initial marking has one token on P_4.

The transition T_3 is a complex one. It is enabled only when an event is not recognized by T_2 and therefore must either result in the creation of a new rule to handle the event or dismiss it as unhandleable. The decision process attached to the transition must consult the rule bases, discussed below, to decide how to dispose of the event, adding to the rules or modifying some part of the rule bases. Depending on the exact nature of the exception, this may involve either the individual actor changing an informal rule, or groups of actors changing their respective rule bases. This, in turn, may involve clerical staff or the higher levels of the organizations involved.

THE RULE BASE

The rules consulted by organizational actors while processing events are derived from a number of sources, both within and without the organization. Different actors consult different rule bases at different times for different purposes. The organizational memory and procedures are contained in these rule bases. They reflect the open nature of offices by being malleable in interpretation and subject to rapid and frequent changes with time.

There are three major classes of rules; the organizational rule base, the individual rule bases and the group rule bases. The three are derived from different sources and contain a rich variety of contradictions, both internally and with each other. It should be emphasized that the event handling rules, as contained in the rule bases, are different from those used to change the rules in the rule bases.

The Organizational Rule Base

This is the main rule base used by the entire organization. It has the following properties:

- There are formal ways of creating, amending, destroying and enforcing compliance with the rules. Examples of such rules are civil constitutions, laws, international treaties and governmental regulations. The ways of creating and changing the rules are generally well known.
- The rules are publicly available. Although the rules themselves may change in their interpretation from time to time due to court rulings, for example, the text of the rules can be obtained in publicly accessible data banks.
- The rules are (in principle) well known. Organizational actors are generally assumed to know of their existence. There are often formal channels of rule distribution.
- The rules come from multiple sources. In a legal sense, a commercial organization is bound by rules from senior and junior levels of government, international treaties and formal business agreements concluded with other organizations. These rules can conflict. This is not exclusively due to problems of interpretation. Often, different levels of jurisdiction have differing goals and the laws they make can be in direct conflict with each other.
- The rules are not created by those who must administer them. There are usually bodies created for the specific purpose of administering the rules, separate from the bodies that created them.

The Individual Rule Bases

These rule bases are the sets of rules used by individuals or small groups of individuals in the performance of their duties. They are largely specific to the individual concerned and when an individual changes jobs, the rules used by the new incumbent change accordingly. They have the following properties:

- The rules belong to the individual processing the event in question.
- The rules are not formally codified. The rules are usually not even created until some event occurs which requires a rule. It is then created on the spot and after application to the circumstance which required it, may be added to the individual rule base or may be forgotten.
- They are not publicly available. Their existence may not even be known by anyone other than the actor using them. The actor may even hide or deny their existence.
- The rules in this base are used rather infrequently. They are typically used to handle small numbers of exceptions occurring infrequently. If large numbers of similar exceptions occur, the rules used tend to become formalized and thus leave this classification.
- The rules are derived from one source, the individual performing the task. They are easily changed and because of this, they are not contradictory.
- The rules are made by the same person who implements them. It is problems in application of the other rule bases that cause them to be created. The rules frequently conflict with those in the other two classes of rule bases. They are usually created to resolve conflicts or uncertainties in the more formal rule bases.

The Group Rule Bases

The third class of rule bases consists of those developed by groups of actors to deal with exceptions that cannot be handled by other means. Typically this is done to handle the contradictions in the more formal organization rule base or to formalize procedures that were previously handled by actors using individual rule bases. They have the following properties.

- The rules apply to larger numbers of exceptions. An individual exception is often handled in an *ad hoc* manner by an individual actor. Large numbers of exceptions usually require a higher level of handling.
- The rules are not formally codified. If they were, they would be part of the organizational rule base. They may or may not be written down, or minuted but they don't constitute a formal set of rules.
- There is no procedure to ensure compliance with these rules.
- The rules are not publicly available. The actors themselves are usually made aware of the rules but there are no formal ways of making them known publicly. The actors may not want the rules known widely or their existence realized at all.
- The rules are often widely known or suspected. They may exist only very informally.
- The rules are very strongly influenced by the implementors. Often it is the implementors who create them in order to realize some local goal or to facilitate the handling of events.

Although the rule bases are conceptually distinct, there may be considerable overlap in practice within an organization. An individual actor may use rules from all three bases depending on whether or not that actor works alone on a task or as part of a group, and, on the nature of the task. The so-called "unwritten rules" of an organization appear as part of the group and individual rule bases. The distinctions between them are not always clear and different organizations may have their rules classified in different bases. Such things as dress codes may be found in the formal rules of one organization and in the group rules of another. In either case they can directly conflict with state legislation in such matters.

This view of the role and structure of group rule bases differs from that proposed by some other studies, *e.g.*, [27]. We emphasize that the group rule base is only quasi-structured and often does not appear in written form, making it qualitatively different from the organizational rule base.

The rule bases are dynamic. They change frequently. The changes are initiated by the economic and regulatory environment and by the nature of the events and exceptions that are detected. The nature of these changes is discussed in the next section.

THE DYNAMICS OF RULE BASES

The contents of the three rule bases change constantly due to changes in the operating environment or personnel.

When actors change responsibilities, the rules by which an event is handled change as well. This is due to the new set of individual rules accompanying the actor which will influence not only the performance of the actor in individual duties but also the performance of bodies on which that actor sits.

Even if there are no personnel changes, the rules migrate between classes and change their structure. An "individual" rule which is used and seen to work well tends to become part of the group rules. This is particularly true if the exception which caused the rule to be created increases in frequency. The individual rule will tend to diffuse and spread among other actors and related groups. Several things may happen:

- An "individual" rule may be deemed to be undesirable. The rule may be dropped or modified to appear more favourable.
- A "group" rule which is seen to work well may become codified by some competent authority and become part of the formal organizational rule base.
- Similarly, a rule which is thought to have undesirable effects may be prohibited by an authority.

The process by which the contents of the rule bases is changed runs in parallel with the processes implementing the regular business of the organization. The changes are created and implemented as side effects of regular business. These concepts are shown in figure 2.

GOAL VERIFICATION

During the processing of routine events the contents of the rule bases are not usually changed. If a change occurs, it is generally because of a change in the organization's operating environment rather than something due to the events themselves.

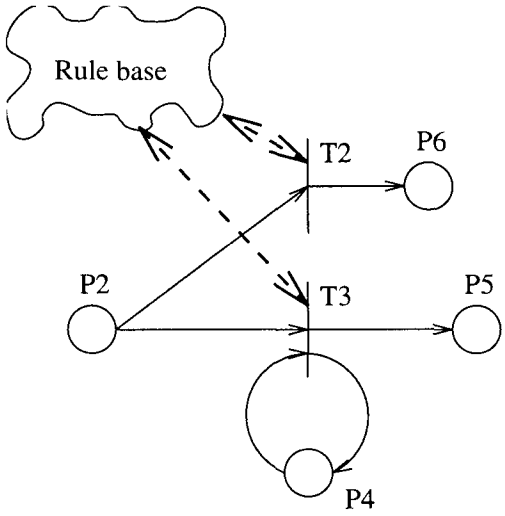

Fig. 2. The dynamics of rule base changes.

When a true exception is recognized, the rule bases are updated by the appropriate actor or actors, and the event is re-handled. When processed by the new rules, the event is either handled by the new rules or discarded. The question then arises whether or not the goal can actually be attained by the revised set of rules.

The result of the recognizer transition T_2 is either **true** or **false**. In practice, it is often found that the decision is calculated over a compound predicate that may contain a large number of conditions to be checked. For example T_2 may be a composite of several transitions shown in figure 3.

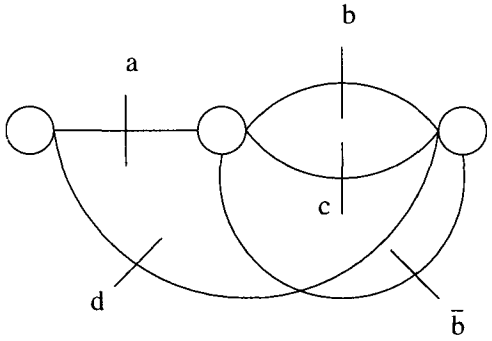

Fig. 3. T_2 as a composite transition.

If *a, b, c, d* are elementary predicates such as "the amount of the invoice is less than 500 dollars", or "the date of the submission was previous to Jan. 4, 1995", then the value of the predicate, when evaluated, indicates whether the amount of the invoice really was less than 500 dollars, etc. The goal state is attainable if some sequence of transitions forms a path leading to it. This the goal state in figure 3 is attainable if the event in question has properties satisfying the predicate

$$ab + ac + d + a\bar{b} = a + d$$

Note the redundancy with the first and last terms in the expression. This redundancy is deliberate and reflects the fact that, in this example, the processing of events at the stages where predicate b is considered is done by two different actors.

Changing the rule base during the handling of a true exception will not merely change the handling for the event in question, it may also change the way all subsequent events are handled. For example, if the order of processing events is modified such that a and b change position, the net of figure 4 is produced and the goal state is changed to

$$ab + bc + d$$

On the other hand, if the order is modified such that a and d change places, figure 5 is produced and the goal state is changed to $a + d$ the same as the goal state of figure 3.

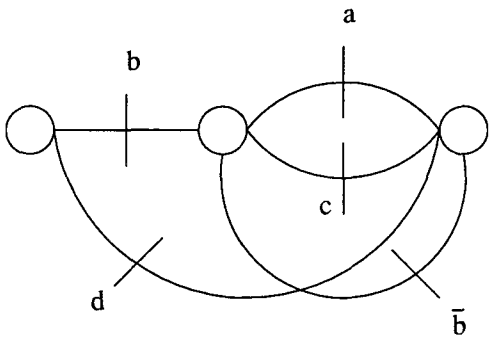

Fig. 4. a and b change positions.

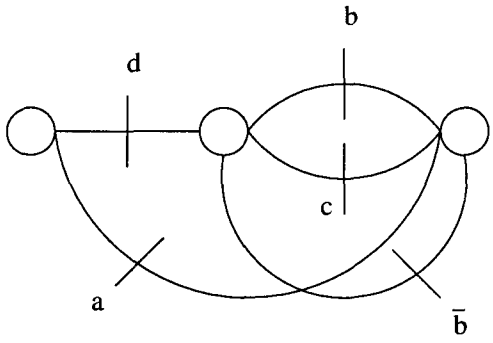

Fig. 5. a and d change positions.

Thus a modification of the rule bases changing the order in which the a and b predicates is evaluated would also change the value of the goal state while a similar change involving the a and d predicates would preserve this value. We would like to find a procedure which can determine whether a change to an office system in response to a change in the rule bases, will preserve the defined goal state.

In the rest of this section, we present an procedure for determining whether a change in the rule bases preserves the finite state diagram of the underlying net with a given initial marking. In so doing, we can guaranty that a given change in the rule bases will lead to the same goal state. This is accomplished by constructing a checking sequence for the FSM corresponding to the rule bases and to the net describing how they are used. This checking sequence is then applied to the net and its response is recorded. If this response corresponds to the response expected for the desired case, the rule bases will lead to the correct goal state.

In this treatment it is assumed that:

1) The recognizer's finite state machine M is minimal, deterministic and represented by a strongly connected directed graph. The system is deterministic in that all other things being equal, a given input will always produce the same output. It is indeterministic in that the choice of inputs is governed by higher level consideration and may appear random.
2) After a change in the rule bases, the resulting finite state machine M' has the same input set and, at most, the same number of states as M.
3) M accepts a reset input r which produces no output and sends the machine back to its input state.

The new machine M' is thought of as a black box and is not observed directly. It has limited controllability and observability. The procedure starts by calculating the unique I/O sequences, $UIOi$ for each state i, *i.e.*, the sequence of I/O symbols such that the response to the input portion is unique to that state.

After this, a second sequence, the E_α sequence, consisting of the UIO sequences $UIOi$ concatenated together, each one separated from the others by the reset input, is formed.

Thirdly, the E_β sequences are calculated. To form these sequences, the transition sequence leading from the initial state to each of the other states is prefixed to each member of the UIO sequences in turn, and the resulting augmented sequences are concatenated together separated by the reset sequence as before.

The fourth calculation is performed by considering each ordered pair of states in M. For each pair, the input sequence leading from the first member to the second member is prefixed to the UIO sequence for the second member. These sequences are concatenated together separated by the reset sequence. These are called the E_c sequences.

The fifth calculation uses the results from above. A new FSM is constructed using the states of M and the edges derived from the E_α, E_β and the E_c sequences. If a rural postman tour can be traversed on this graph, then the checking sequence for M is obtained by recording the input and output portions of the edges of the tour. If such a tour cannot be traversed, sufficient edges are added from the edges of M such that the tour can be made. The checking sequence is then formed as before.

By using the checking sequence as input to the graph M' and by observing the output sequence, it can be determined whether M' is similar to M or not. In particular, if the two are not the same, the goal states are not identical.

A detailed explanation of the procedure described above and an example can be found in [18].

EXAMPLE

In this section, a real world example [20] of one kind of purchasing process in a large paper machinery factory is provided:

The process is initiated by the engineering or manufacturing departments. When a need for a purchase arises, they normally request a purchase number from the computer system, then manually fill a purchase order and send it to a purchaser responsible for such items in the purchasing department. Even though the manufacturing departments have access to the purchasing system, they are not allowed to place orders. This task is left to purchasers, who know how to do it in accordance with organizational policy. A Petri-net representation of the process is shown in figure 6.

The purchaser first determines whether the order should be sent to tender or whether there is an appropriate agreement with some supplier about delivering such items. If no such supplier is available and there is plenty of time before the requested delivery, tenders are sent to potential suppliers. When offers are received, they are reviewed and the most appropriate supplier is selected. After this, the process goes

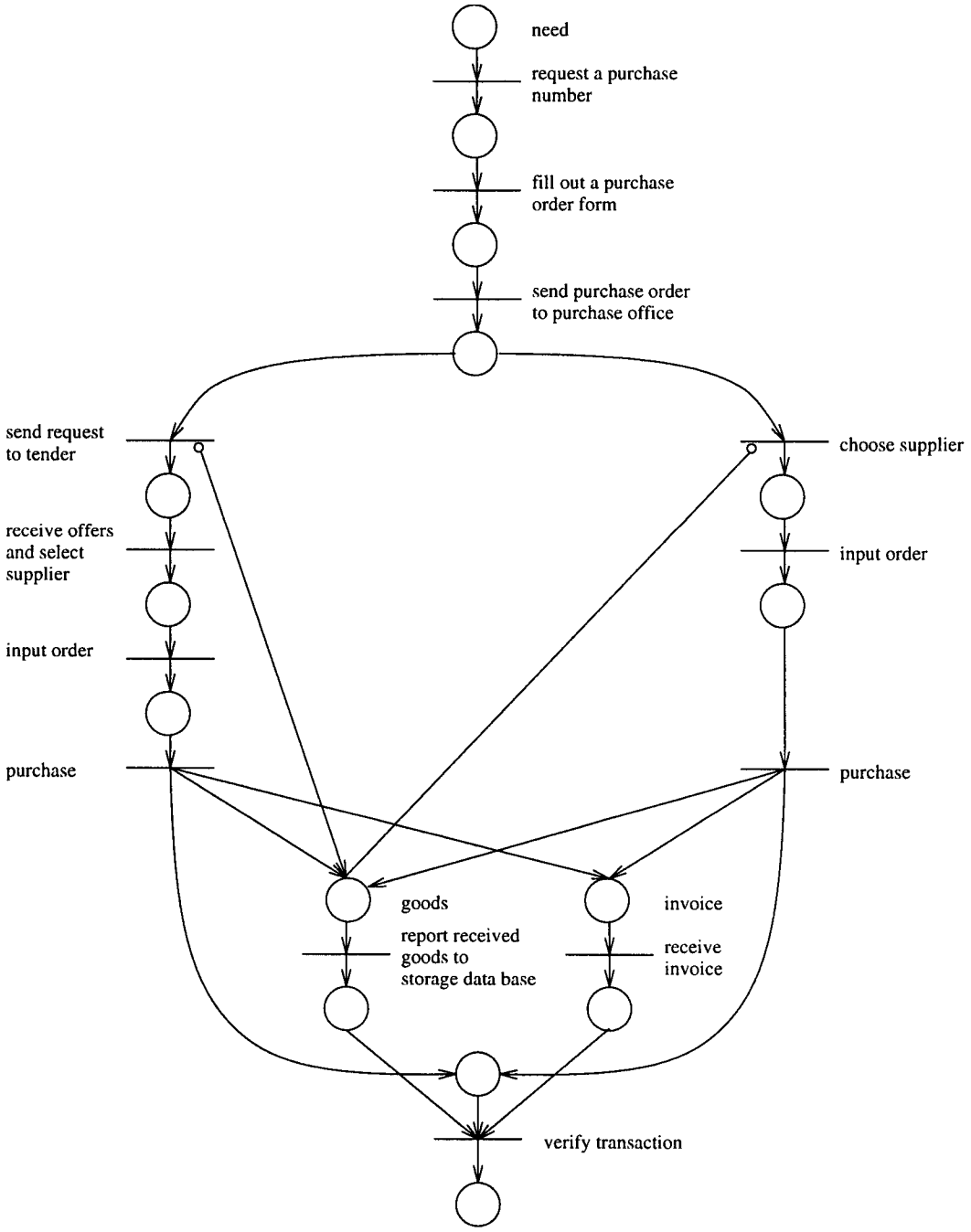

Fig. 6. Petri-net representation of purchasing system.

on as if there had not been a tendering process at all.

Next, an order will be produced and will be sent to the supplier. This updates the order database. When the requested goods arrive, information about them will be input to the storage database. When a corresponding invoice is received, its details will be input and transaction verification can take place. This is shown by the left branch of figure 6. If the order is not sent to tender, the path follows the right branch of figure 6.

Such a process involving several people from several organizational units is likely to suffer from exceptions. The following is a real world example of one of them detected during a case study:

In one of the manufacturing workshops a crucial tool breaks. Similar tools are used in other workshops, but they are all in heavy use. The foreman of the workshop knows a local hardware supplier carrying such tools and decides to pick one up there in order to keep the work going. Before going to the store, he requests a purchase number. The store will use this number as a reference when it later invoices the factory. When the foreman comes back, he notifies the purchasing department by filling out a purchase order. He sends the order to a purchaser. To make transaction verification possible, he also inputs his purchase to the storage database.

When the purchaser's secretary starts handling the purchase order, she is surprised to notice that the purchase number has already storage items attached to it. After a conversation with another secretary and the purchaser, they figure out what has happened. When the situation becomes clear to the purchasing department, they can start gathering more information and later are able to fill out a purchase order and thus make the forthcoming transaction verification possible. This requires a change to the organizational rule bases.

A change, however, may cause the system to work in unpredictable and inconsistent ways if the changes are not done carefully. The algorithm described earlier can be used to determine whether the changes in the rule bases brought about by resolution of the exception will cause undesirable effects elsewhere in the system.

The Petri-net of figure 6 is first reduced to its essential features by elimination of those transactions which provide no useful state space knowledge. The reduced net is shown in figure 7.

With one initial token in P_0, the finite state diagram, FSD, shown in figure 8 shows the behaviour of the net. To ensure net liveness, the output of the last transaction is fed back to place 0.

There are seven vertices in the net, $v0$, ... , $v6$ and 14 edges. Each vertex contains 8 digits denoting whether the corresponding position in the reduced Petri-net holds a token or not. Thus $v5$ contains the digits 00010011, indicating that positions 3, 6 and 7 hold tokens in that particular state. The 14 edges are labeled *input/output*. The *input* portion denotes the choice that can be made as to which way a token can be absorbed by a transition. There is a maximum of two choices at any point in this example, labeled a and b.

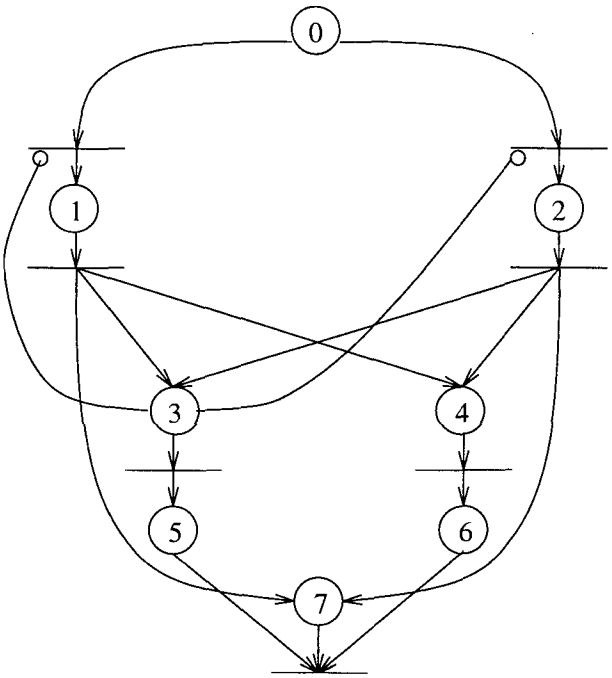

Fig. 7. Reduced Petri-net of figure 6.

The *output* portion denotes an observable result of a transition. This is typically a piece of paper or an electronic entry or something similar. To simplify matters they are represented here by the integers 0, ... ,9.

A redundant edge has been added from $v6$ to $v0$ emphasizing the continuous nature of the purchasing process studied here.

The UIO sequences for this example $a/0,a/1,a/2,a/3,a/4,a/5,a/6$ all have the same input portions, simplifying the analysis greatly. Using the calculation sequence described in [18] the value of the input portion of E_α is: $r\ a\ r$ where r is the reset input described earlier.

The input portions of the $E\beta$ sequences, when concatenated together, yield the string

aar bar aaar aaaar aabar aaaaar

The Ec sequences are calculated from the transition paths from each state to each successor state including those paths which are simple loops. There are 14 such paths and their input portions are:

aar bar bar aar bar aar aar bar bar aar bar aar aar bar

The next step consists in calculating $E_3 = E_a \cup E_\beta \cup E_c$. The graph formed from the set of nodes, V_1 and E_3 is then modified such that it is strongly connected by selecting additional edges, E'' , from the original FSD. This augmented set of edges, E', is used to form a new graph $G' = (V, E')$. This is shown in figure 9.

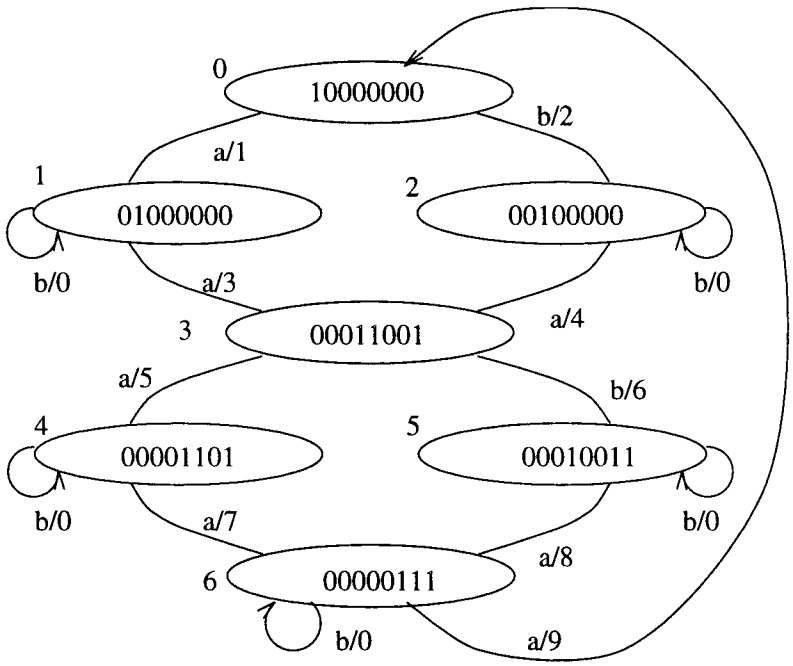

Fig. 8. Finite state diagram of figure 7.

The final step consists in calculating a tour on G', starting and ending at $V0$, and traversing all the edges E'. This is sometimes called a *rural postman tour.* The sequence of inputs required to implement the tour is the checking sequence which characterizes the system. For this example, it has the value

rar aar bar aaar aaaar aabar aaaaar abar bbar baar aaabar aabbar aabaar aaaaaar aaaabar

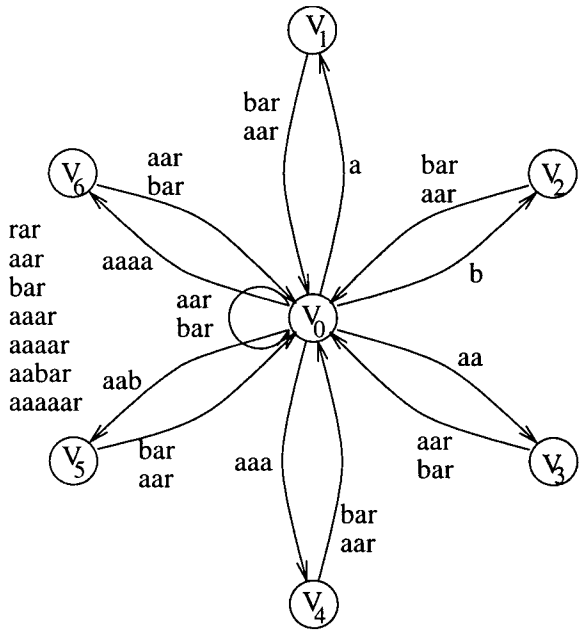

Fig. 9. The states and the E' edges and their input sequences.

and when it is input to the system, it produces an unique input output which characterizes it. A change which does not modify the output will not create any undesired or unexpected states.

CONCLUSION

The source of exceptions can be traced to a change of some kind in the operating environment of an organization or to some unexpected events whose handling may cause a change in the rule bases. These exceptions are extremely important in any study of OIS because of their potential cost and the possibility that their handling may obscure or inhibit the attaining of office goals.

The events that enter offices are analyzed to recognize their type and treatment and handled according to the rules in the office rule bases. These rule bases originate in the formal organization rules, the informal group rules and individual rules. These rules may conflict and overlap and they will change with time.

A procedure, based on the finite state machine representation of the Petri-net description of event handling using rules, was introduced to provide a method of determining whether the goal of the office procedures in question could still be attained after change. A positive result arising from this algorithm is sufficient, although not necessary, to ensure that the process goal can still be reached after a rule base change has been effected.

ACKNOWLEDGEMENT

We would like to thank Professor Clarence (Skip) Ellis for his suggestions and the hospitality offered to us at the University of Colorado at Boulder.

REFERENCES

1. Auramäki, E. and Leppänen M. Exceptions and Office Information Systems. In P. Pernici, A. A.. Verrijn-Stuart Ed. *Office Information Systems: The Design Process,* Elsevier Science Publishers B. V. (North-Holland), IFIP, 1989, pp. 167-182
2. Borgida A. Exceptions in Object Oriented Languages. *SIGPLAN Notices,* 21, 10 (1986), 107-119
3. Borgida, A. Language Features for Flexible Handling of Exceptions in Information Systems. *ACM Trans. Database Syst.* 10, 4 (Dec. 1985), 565-603
4. Broverman, C.A. Constructive Interpretation of Human-generated Exceptions During Plan Execution. *Ph.D. Dissertation Thesis, University of Massachusetts,* 1991
5. Cheriton, D. Making Exceptions Simplify the Rule and Justify Their Handling. *Information Processing 86,* H. J. Kugler Ed. North-Holland, 1986
6. Cox, B. Exception Handling and Object Oriented Programming. *Workshop-paper presented at ECOOP'91,* Geneva, July, 1991
7. Dony, C. Exception Handling and Object Oriented Programming: Towards a Synthesis. *Proceedings of OOPSLA/ECOOP '90,* ACM Press, New York, 1990
8. Ellis, C.A. Formal and Informal Models of Office Activity. *Information Processing 83,* R. E. A. Mason Ed. Elsevier Science Publishers B. V. (North-Holland), IFIP, 1983, pp. 11-22
9. Ellis C.A. Information Control Nets: A Mathematical Model of Office Information Flow. *Proceedings of ACM conference on Simulation, Measurement and Modeling of Computer Systems,* 1979, pp. 225-239
10. Galbraith, J. R. *Designing Complex Organizations,* The European Institute for Advanced Studies in Management, Addison-Wesley, Reading, Massachusetts, 1973
11. Galbraith, J. R. *Organization Design,* The Wharton School, University of Pennsylvania, Addison-Wesley, Reading, Massachusetts, 1977
12. Genrich, H. Predicate/Transition Nets. *Advances in Petri Nets, Lecture Notes in Computer Science, Vol. 254,* W. Reisig and G. Rozenberg Ed. Springer-Verlag, Berlin, 1987, pp. 207 - 247
13. Goodenough, J. Exception Handling: Issues and a Proposed Notation. *Comm. ACM,* 18, 12 (Dec. 1975), 683-696
14. Karbe, B. K and Ramsberger, N. G. Influence of Exception Handling on the Support of Cooperative Office Work. *ACM SIGOIS,* 11, 4 (Dec. 1990), 2-15
15. Kunin, J. *Analysis and Specification of Office Procedures.* Massachusetts Institute of Technology, Laboratory for Computer Science, MIT/LCS/TR-275, 1982
16. Liskov, B. A. and Snyder, A. Exception Handling in CLU. *IEEE Transactions on Software Engineering,* SE-5, 6 (Nov. 1979), 546-558
17. Masapati, G. H., White, G. M., TP Nets: A Computer Based Tool for Office System Design. *Proc of IFIP WG8.4 Working Conference on Office System Design,* Linz, Austria, Aug. 15 - 17, 1988, pp. 116-129
18. Rezaki, A., Ural, H., White, G. Construction of Checking Sequences Based on UIO Sequences. *Proc. of Ninth Int. Symp. on Computer and Information Sciences,* Antalya, Turkey, Nov. 7-9, 1994, pp. 319-326
19. Saastamoinen, H. T. Rules and Exceptions. *Information Modeling and Knowledge Bases IV: Concepts, Methods and Systems,* H. Kangassalo, H. Jaakkola, K. Hori, T. Kitashi Eds. IOS Press, Amsterdam, 1993, pp. 271-286
20. Saastamoinen, H. T. Significance of Exceptions in Office Information Systems - A Case Study in Valmet Paper Machinery, *preprint,* 1994
21. Saastamoinen, H. T., Savolainen, V. V. Exception Handling in Office Information Systems, *Proc. of the Third Intl. Conf. on Dynamic Modeling of Information Systems,* Noordwijkerhout, The Netherlands, 1992, pp. 345-363
22. Saastamoinen, H. T., Exceptions: Three Views and a Taxonomy, *preprint,* 1995
23. Saastamoinen, H. T., Markkanen, M. V., and Savolainen, V. V. Survey on Exceptions in Office Information Systems, *Univ of Colorado at Boulder Tech. Report CU-CS-712-94,* 1994
24. Strong, D. A. and Miller, S. M. Exception Handling and Quality Control in Office Operations, *Boston University School of Management Working Paper Number 89-16,* Boston, MA, 1989
25. Suchman, L. A. Office Procedure as Practical Action: Models of Work and System Design. *ACM Trans. on Office Information Syst.* 1, 4 (Oct. 1983), 320-328
26. Twining, W. and Miers, D. *How to Do Things with Rules, A Primer of Interpretation,* Weidenfeld and Nicolson, London, 1976
27. Williams, L. J. and Lochovsky, F. H., Supporting Knowledge Migration in Organizations, *Information Processing 89,* Ritter, G. X. Ed., Elsevier Science Publishers B.V., Amsterdam, 1989, 259-264