INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road Ann Arbor MI 48106-1346 USA
313 761-4700 800 521-0600

Order Number 9510746

Model-based information processing for internal control evaluation

Zhou, Ying, Ph.D.

City University of New York, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

MODEL-BASED INFORMATION PROCESSING FOR INTERNAL CONTROL EVALUATION

by

YING ZHOU

A dissertation submitted to the Graduate Faculty in Business in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1994

This manuscript has been read and accepted for the Graduate Faculty in Business in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

6-8-94
Date

Chair of Examining Committee

6-15-94
Date

Executive Officer

Dr. Douglas Carmichael

Dr. Martin Benis

Dr. Thomas Verghese

Dr. Nils Kandelin
Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

Model-Based Information Processing for
Internal Control Evaluation

by

Ying Zhou

Adviser: Professor Thomas Verghese

This research is motivated by the possibility of integrating descriptive and decision models of internal control evaluation. The purpose of this study is to explore the feasibility of using activity and event models for translating descriptive information to decision information.

Descriptive models provide a view of the transaction processing system that is close to the physical system. Decision models can use such information as inputs to the model. Current technology does not support the use of descriptive information in the decision stages of an audit; the description is used only to document understanding and not for analysis. The integration of descriptive and decision models requires a mechanism that can translate the objects and relations between the two types of models. The contribution of this study is to provide a computational theory for such model translation.

The computational theory for model translation is implemented by constructing a knowledge-base of extracting and structuring rules in propositional logic. Two analytical models based on activities and events, the reliability network model and the DEAS model, are chosen as target decision models. The information necessary for the target models is extracted from a source model based on diagrammatic descriptions of transaction processing systems. A set of algorithms is developed for extracting and structuring decision information.

The model translation theory and the algorithms are tested with an internal control evaluation problem. Results are consistent with the suggested solutions. To test the robustness of the translation mechanism, the source descriptive model has been altered to conduct a sensitivity analysis. Results from the sensitivity analysis show that the default rules work well when inconsistencies and incompleteness exist in the descriptive model.

General results of the study show that the integration of descriptive and decision models is computationally feasible and can be accomplished through knowledge-based methods. This study also demonstrates the usefulness of activity and event models in soliciting and organizing knowledge to be used in internal control analysis.

ACKNOWLEDGEMENTS

I would never have completed this dissertation without the guidance, encouragement, assistance and support of my dissertation committee.

I would like to thank Professor D.R. Carmichael, my dissertation committee chair, and Professor M. Benis, member of the committee, for their interest in this work, their challenging questions, and their willingness to offer help whenever it is needed.

My special thanks go to my dissertation advisor, Professor Thomas Verghese of the Department of Accountancy, the George Washington University, for everything an advisor could be: an exceptional role model, a teacher, a mentor, a colleague, an editor and a friend. He has introduced me to the area of information systems and intelligent decision support systems. His deep insights and perseverance in seeking the truth have always fascinated me. As my dissertation supervisor, he has guided my research by providing direction and detailed advice, and has been extremely generous with his time and patience. His constant attention, response, and counsel over the years to drafts

of my dissertation is incredible. This dissertation would not have been possible without his guidance.

I would also like to thank Professor Nils Kandelin of the Department of Accountancy, the University of Maryland, my outside reader, for his thoughtful comments on the final drafts.

I would like to thank the faculty and my fellow doctoral students in the Department of Accountancy at Baruch College for their support and friendship.

I would like to thank my colleagues in the Department of Accounting and Information Systems at Queens College, particularly, Professor A.J. Simon, the Department Chair, Professor A. Adelburg, and Professor M. Levine, who have continually recognized the importance of my academic pursuits. Professor Simon has allowed me tremendous latitude as I attempted to complete my degree while attending my professional duties.

I would like to thank the participants in the CUNY Dissertation Completion Program, especially Professor Susan Forman, our mentor in the program, for comments and suggestions on writing and completing a good dissertation and for their friendship.

I would like to thank Mr. Walls James for his assistance in polishing the final drafts.

I am grateful to Dr. Lillian Y.F. Hsu and Mr. Chao Y. Wang for their understanding and financial support.

Professor Richard P. Nathan and his wife Mary have provided me with all the support and encouragement that one would like to have when studying in a foreign country. I would like to thank them for everything they did for me to make me feel at home and to keep me going.

My parents have taught me to seek after truth and to persist when there appears to be no hope of succeeding. I thank them for implanting the seeds of learning within me and nurturing those seeds so well. My mother stayed with me and lent her hands when I was completing the final drafts. It was her strong belief in my ability to finish that kept my momentum.

Dissertation writing is a difficult, stress-ridden and lonely process. I would like to thank my best friend and spouse, Shichuan Lu, for having faith in me when I had none; for listening, understanding and supporting over the years, especially during crisis.

I would like to thank my daughter Sara for giving me a reason to finish and for her lovely and nurturing presence.

I would like to dedicate this dissertation to my parents, my husband and my daughter.

Table of Contents

List of Tables

List of Figures

Chapter 1 INTRODUCTION

1.1 Motivation

Understanding of a client's internal control structure is required by the second auditing standard of field work[1] (AICPA 1988, 1989). Analysis of a transaction processing system which includes analysis of an accounting system and control procedures contributes to the evaluation of internal control structure and can be divided into two stages: an information collection stage and a decision stage.

During the information collection stage, auditors study transaction processing systems, collect data about their functioning, and document their (auditors') understanding using diagrammatic symbols. The symbol set used depends on the particular descriptive model used by an auditor. SEADOC (Elliott 1983), INFOCUS (Grant Thornton 1992), and system flowcharts are examples of such descriptive models. During the decision stage, the auditor

---

[1] SAS 55 revised the second standard of fieldwork of the ten generally accepted standards as follows:

> A sufficient understanding of the internal control structure is to be obtained to plan the audit and to determine the nature, timing, and extent of tests to be performed.

translates understanding of the transaction processing system into a decision about its reliability, which then is incorporated into the decision about the control risk for one or more of the financial statement assertions.

Audit researchers have been studying audit decision making for decades and have developed various decision models and tools to aid auditors in their audit work. The long-term goal for such research has been to increase the effectiveness and efficiency of audits. Internal control analysis is one area where research efforts have been heavily invested. To date, a number of decision aids for internal control analysis have been developed (Elliott 1983; Gal 1985; Meservy 1985; Meservy *et al.* 1986; Grant Thornton 1992; Hamscher 1992). These computerized decision aids are knowledge-based systems that apply experts' experiential knowledge to internal control analysis. The models underlying these decision aids usually employ high-level concepts such as objects, processes, activities and controls.

In the past two decades, several internal control decision making models have been suggested (Yu and Neter 1973; Cushing 1974, 1975; Stratton 1981; Bailey *et al.* 1985; Knechel 1985; Verghese 1988; Srinidhi and Vasarhelyi 1989). These decision models all require descriptive information about transaction processing systems, yet

current audit technology does not support the use of descriptive information in the decision stages of an audit – the description is used only to document understanding and not for analysis (Hamscher 1992). Using descriptive information in computer-accessible form for the decision stages will result in efficiency in model acquisition, enhanced quality of decisions, and uniformity in judgment (Hamscher 1992; Grant Thornton 1992).

Implementing Hamscher's suggestions in a practical sense will require that decision information be extracted mechanically from computer-readable descriptions of transaction processing systems. At present, it is common practice to implement system documentation using computer systems and hence computer-accessible descriptive models are readily available.

Computer system documentation is qualitative in nature, describing the physical objects of a transaction processing system. The representation of qualitative information of this kind is best achieved with declarative propositions – statements that a particular kind of object exists in the system and that it bears a particular kind of relationship to other objects in the system.

If qualitative descriptive information is represented as propositions, then implementing Hamscher's suggestion amounts to the translation of descriptive propositions to

decision propositions. Since each kind of proposition is constructed from a domain of object classes and relationships, the extraction of decision concepts from descriptive concepts requires the translation of descriptive objects and relationships into decision objects and relationships.

The relationship between decision concepts and descriptive concepts often is neither explicit nor formal. The relationship is often implied in a shared context of knowledge between the description and the decision stage. During the translation process, information that is not explicitly available in the description must be added to generate the decision model.

The problem addressed in this dissertation is the mechanical extraction of decision information from descriptive information. This mechanical process includes translation and the addition of non-explicit information. The descriptive model chosen is the system flowchart. Two analytical models, based on activities and events, are chosen as the target decision models. The information necessary for both of the analytical models can be implemented using the propositional form.

The choice of source model is driven by non-proprietary reasons. Flowcharts are commonly used in text

books[2], have been promulgated as generic practice by the AICPA (1988, 1989), are based on physically observable entities (unlike SEACAS and INFOCUS), and are understood by all auditors.

The choice of target models is driven by several factors. First, the semantics of the two models is clearly defined – reliability for activity networks and probability for event networks. They are non-proprietary and are independent of audit practices, models, or transactions. The reliability computations from any subset of the networks to the entire network is formally defined. Finally, both models are described easily with the predicate grammar of formal logic.

The criterion for extracted information is that it should be intuitive, be based on the descriptive information, and be suitable for activity and event analysis.

1.2 Previous Research

In this section, the systems approach to internal control decisions is briefly reviewed. The major focus of

---

[2] System flowcharts are used as system documentation methods and introduced to students in many auditing textbooks. The following is an incomplete list: textbooks by Arens and Loebbecke (1988), Watne and Turney (1990), Kell and Boynton (1992), Konrath (1993).

this line of research is to identify system components and study how component behavior affects the system as a whole. Depending on the objects and events allowed in the modeled system, the specification of system behavior varies and is calculated as a function of the behavior of system components. A summary of these models is included in table 1.1 at the end of this section.

Yu and Neter (1973) first propose a decision model for computing system reliabilities. In Yu and Neter's model, the financial information system is viewed as consisting of many operating elements, defined as performance units, that can affect the state of error and quality of documents. The performance of each operating element is defined in terms of a probability that indicates the error-producing propensity of that element. The system behavior during transaction processing is modeled as a stochastic process (Markov). The quality of system output depends on the quality of the performance of its individual operating elements, each of which generates, compounds, or corrects errors in the accounting data as they are processed through the system. The total system's reliability is contingent upon the performance of each system component and the manner in which those components are configured.

Cushing (1974, 1975) adapts reliability theory to internal control analysis. The accounting system is

perceived as consisting of processing and control procedures. Processing procedures introduce errors into the system, whereas control procedures prevent or correct these errors. The system behavior depends on the interaction between the potential errors and the control procedures. Stratton (1981), Srinidhi and Vasarhelyi (1989), and Knechel (1985) extended and applied the Cushing and Yu and Neter approaches. Later research by Bailey *et al.* (1985) and Hamscher (1992) has introduced rich semantics into internal-control decision models.

Bailey *et al.* (1985) design a formal language to describe actions of agents and nature of tasks in modeling transaction processing and control systems. Hamscher (1992) extends Bailey *et al.*'s set of basic concepts, made a distinction between activities and actions, and defined relations among various actions. Verghese (1992) proposes a theory of transaction processing, in which the system behavior, its lawful and unlawful conditions and possible activities and events are formally defined. These later studies (Bailey *et al.* 1985; Hamscher 1992; Verghese 1992) have provided the foundation for this research. The semantics required for model translation is based largely on the results of these studies.

Table 1.1 Previous Research

Research Study	Model Objects	Events
Verghese 1992	activity events transaction	assigning value to transaction properties control: prevent unlawful assignment of values
Hamscher 1992	activity record repository agent	actions, *e.g.*, create, copy, compare records
Bailey 1985	agent, task, system-objects, repositories	assign, modify, destroy, transfer, wait for, put, get, copy
Knechel 1985	processing procedures, control procedures	error generation error correction
Stratton 1981	procedures	performance of procedures
Cushing 1974	processing procedures, control procedures	error occurrence, error correction
Yu & Neter 1973	operating elements,	add errors, compounding errors, correcting errors

1.3 Summary

To summarize, system documentation provides descriptive information about transaction processing systems. Decision models can use such information as input to the model. The problem addressed in this research is the mechanical extraction of decision information from descriptive information. Two analytical models based on activities and events are chosen as target decision models. The information necessary for the target models is to be extracted from a source model based on diagrammatic descriptions of transaction processing systems (a systems flowchart). All three models are implemented with predicate logic syntax.

The remainder of this dissertation is organized as follows: Chapter 2 provides analysis of models; Chapter 3 discusses the translation theory; Chapter 4 discusses algorithms; Chapter 5 presents examples and tests, and Chapter 6 draws conclusions.

Chapter 2 Analysis of Models

In this chapter, the information requirements for activity and event analysis of transaction processing systems are examined. Flowcharts are discussed and a flowchart model is described.

2.1 Activity-Based Analysis

Activity-based analysis of a system focuses on the reliability of system activities which is combined into an overall system reliability. Reliability is defined as the probability that an activity or activity sequence will complete without a predefined error occurring[3].

Stratton (1981), following Cushing's introduction (1975), demonstrates reliability analysis in a purchase transaction cycle application. The logical structure of a system's activities is placed in the form of a logical network and reliabilities are attached to each activity. Given the network and individual reliabilities the computation of overall reliability is a trivial computational exercise. However, forming the network and

---

[3] For an in-depth discussion of reliability network concepts, and their adaptation to the internal control systems, see Srinidhi and Vasarhelyi (1989).

assigning reliabilities are tasks requiring expertise and judgment.

Srinidhi and Vasarhelyi (1989) provide justification for the use of reliability theory for internal control evaluation. They divide such evaluation into three stages: 1) estimation, 2) aggregation and 3) interpretation. Included in the estimation stage are subtasks of identification of the relationship between the different components of the internal control system and estimation of the reliabilities of the individual components. The aggregation of system reliabilities and subsequent interpretation depend on the component reliabilities identified in the estimation stage.

Other activity-based research projects include TICOM by Bailey *et al.* (1985), the DEAS model by Verghese (1992), and SAVILE by Hamscher (1992). The TICOM model focuses on task and sub-task analysis, the causal relationships between various tasks and the results of their interactions. The TICOM model uses query satisfaction to search for sufficient conditions for the execution of a sequence of actions. The TICOM's ultimate goal is to help auditors identify control strengths and weaknesses in the system. The DEAS model is activity and event based, and focuses on the input and output resources of activities and possible events caused by the activities. SAVILE represents

accounting systems in terms of activities, records and repositories. Based on a description of correct systems behavior, SAVILE is able to enumerate potential failures that are associated with each activity. The set of failures provides a basis for subsequent risk assessment, control evaluation and test planning. Failure risk analysis used in SAVILE is complementary to the reliability analysis.

The requirements for performing a reliability analysis are descriptions of the logical structure of activities and reliabilities of each activity. The extraction of the logical structure from the observed physical structure of activities is the analytical problem to be solved.

The propositional form of a reliability network would consist of propositions describing the structure as well as propositions describing each activity node. The syntax of reliability network propositions will be:

```
statement(x,s).
connected(x,y).
reliability(x,z).
```

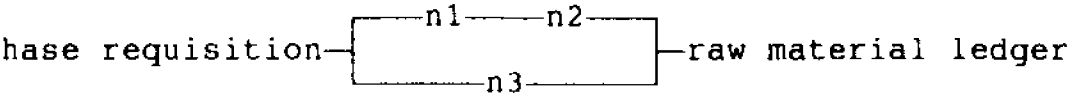
where

x and y are variables for nodes,
s is a statement that describes node x,
z is the reliability number,
connected and reliability are predicates indicating a relationship of their arguments.

For example, Stratton's simplified reliability network (1981, 63) is reproduced in figure 2.1. This network consists of three activity nodes, n1, n2, and n3. Nodes n1

and n2 are serially connected. Node n3 is structurally in parallel to n1 and n2.

Figure 2.1 Stratton's Reliability Network



The following propositions can be set up to represent Stratton's network:

```
statement(n1, "purchase agent prepares purchase order").
statement(n2, "receiving personnel prepare receiving
               report").
statement(n3, "accounts payable clerk reconciles vendor's
               invoice, purchase order, and receiving
               report").

connected(x,n1).
connected(n1,n2).
connected(n2,y).
connected(x,n3).
connected(n3,y).

reliability(n1,0.97).
reliability(n2,0.9533).
reliability(n3,0.92).
```

The probability that the purchase of raw materials is recorded correctly in the materials ledger will depend on the reliability of each of the three activity nodes and the structure with which these nodes are connected. Once the reliability network structure and the reliability of each node are known, the system reliability can be calculated using reliability theory.

2.2 Event-Based Analysis

Event based analysis of a system focuses on the multiple individual events that can occur in a system and assesses probabilities to each. The probability of complex events can then be computed if the structure of events is known.

Yu and Neter's internal control analysis model can be generalized as an event-based analysis. Their analysis, however, does not explicitly name the underlying events, but instead classifies events by their monetary effects.

Reliability analysis is a restricted event analysis where each activity can cause only one of two outputs, such as success or failure.

Forming the events and event sequences in a transaction processing system is clearly a task requiring expertise. In order to restrict the events in an event analysis we need a framework of how and what events occur in a transaction processing system. The next model to be discussed, the DEAS model provides this framework.

The propositional form of an event network would consist of propositions describing the structure as well as propositions describing the probability of each event. The event network will be described using the following:

```
statement(x,s).
connected(x,y).
probability(y,z).
```

where

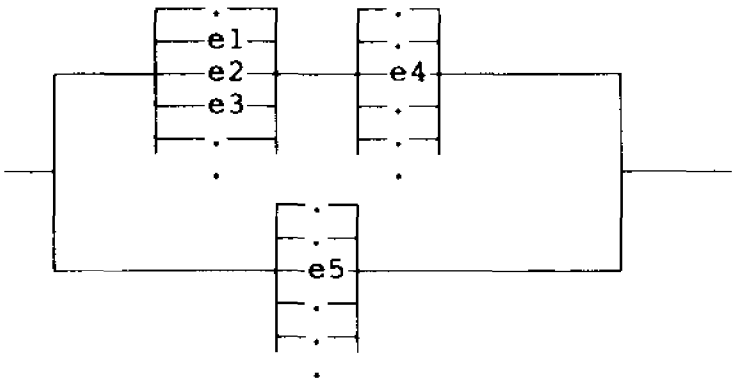
x and y are variables for nodes,
z is the probability number, $0 \leq z \leq 1$
connected and probability are predicates indicating a relationship of their arguments.

Consider once again Stratton's reliability network (figure 2.1) as an example. For each of its activity nodes, several events can be developed. The following propositions can be set up to represent part of the event network:

```
statement(e1, "purchase order matches purchase
               requisition").
statement(e2, "purchase order contains a mathematical
               error").
statement(e3, "purchase order is prepared without a
               purchase requisition").
statement(e4, "quantity received equals quantity
               purchased").
statement(e5, "vendors invoice matches receiving report").

probability(e1,0.997).
probability(e2,0.05).
probability(e3,0.02).
probability(e4,0.95).
probability(e5,0.985).
```

The partial event network for the five propositions is given in figure 2.2.

Figure 2.2 Partial Event Network



2.3 The DEAS Model[4]

DEAS model is activity based and designed to model transaction processing systems. Model objects include resources and activities. The activity is the basic model object.The activity object is defined in a manner that specifies the set of possible events which occur in a transaction processing system. Resources include transaction attributes, communication channels, and information repositories. Resources can be assigned to activities as inputs and outputs. A possible event set can be generated by analyzing an activity's input and output resources.

[4] This entire section is adapted from the DEAS model discussion by Verghese (1992).

The conditions for a successful execution of an activity are as follows:

C1: o = F(i).
C2: m = o.
C3: r = o.

where

o is the output attribute-instance for the activity
i is the input message for the activity,
F is the policy implemented by the activity,
m is the output message,
r is the record for the attribute.

The complex proposition describing the possible events generated by a transaction processing activity is shown in figure 2.3.

Figure 2.3 Possible Events Set Generated by a Transaction Processing Activity

	y=F(x)	m = y	r = y
e1:	t	t	t
e2:	t	t	f
e3:	t	f	t
e4:	t	f	f
e5:	f	t	t
e6:	f	t	f
e7:	f	f	t
e8:	f	f	f

where t represents a true value and f represents a false value of a condition.

Only e1 in the event set leads to a successful state of the processing component, that being that the transaction processed according to the transaction policy. The remaining events indicate failure states of the

processing component, either because the transaction is not processed according to the transaction policy (e5-e8), or the message communicated or the record created does not agree with the transaction attributes.

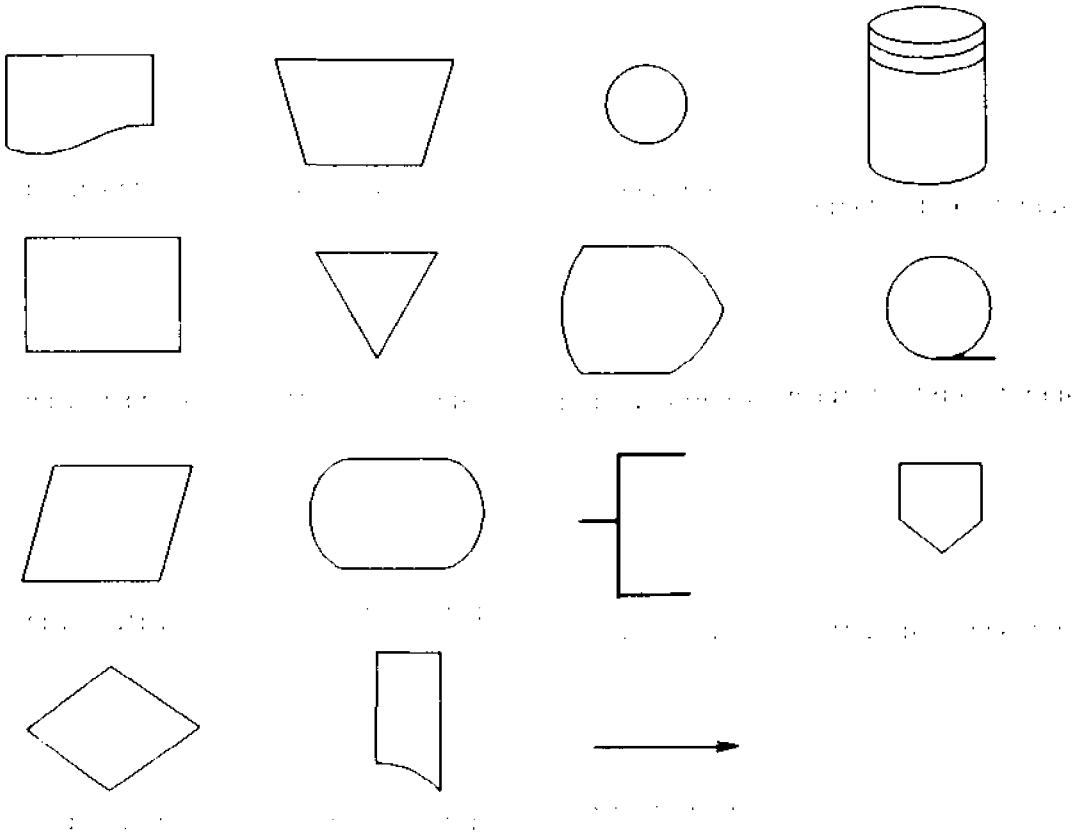
For example, in the case where a transaction is not processed according to the transaction policy, *i.e.*, when the condition y = F(x) takes a false value, the result would be that a sales order is prepared without a customer order.

2.4 Flowchart Model

Flowcharts were originally designed in computer programming as a means of describing the flow of control and computations in computer programs. Flowcharts were also used to describe the sequence of major processing operations and the data flow to and from the files used in processing. They were subsequently adapted in auditing for documenting the flow of transaction processing and the corresponding controls.

Standard flowcharting symbols and techniques are described in the American National Standard X3.5-1970 (Flowchart Symbols and Their Usage in Information Processing). Flowchart symbols used by auditors in developing flowchart models include either a subset or a

Figure 2.4 Commonly Used Flowchart Symbols



modified set of the standard flowchart symbols. The symbols presented in figure 2.4 have been collected from an informal survey of textbook flowcharts (Arens & Loebbecke 1988; Watne & Turney 1990; Wallace 1991; Kell & Boynton 1992; Konrath 1993; Pany and Whittington 1994).

The information in a flowchart includes both symbols and text annotation labels. Symbols represent entities while text describes the characteristics of entities of interest in internal control.

Flowchart symbols can be categorized into the following groups: 1) resource symbol (RESCSYM), 2) activity symbol (ACTSYM), and 3) auxiliary symbol (AUXSYM). This classification is general enough to allow for the addition of new, specialized symbols. Symbols in the resource category include those that represent data objects, such as a sales order; input/output devices used in EDP systems, such as display terminals; and data storage, such as filings of sales order, on magnetic disk, tape etc.. Symbols in the activity category represent activities such as data creation, data processing, and data transfer in the modeled world. Auxiliary symbols are language specific, including direction lines, annotations, on-page and off-page connectors. The auxiliary symbols do not represent model objects, but are included to show relations of model objects or are used for additional diagramming information.

For example, the annotation symbol is used to add attribute information to model objects. The direction line is a special symbol whose meaning depends on how it is used. It can be used to represent many different relations. For example, if it links the same document in two different sections, then it indicates the transportation of the document from one functional unit to another; if it links a manual process and a document, then it indicates that some action has been applied to the document, or the document is a product of the process; if it links a display terminal symbol and a document, then it means that some document information is being entered into or retrieved from the system; if it links a document with an off-line storage, then it means that the document is being filed; if it links two process symbols, it indicates the continuation of activities. The on-page and off-page connectors in essence are an extension of the direction line. These connectors are often used to indicate the transportation of documents from one functional unit to another.

Flowchart text can be perceived as attachments to flowchart symbols. It is used to describe attributes of model-based objects. For example, the agent attribute can be attached to processes through text, and an activity-type attribute can be inferred from text that describes the

activity and its object. Document names can be attached as an attribute to document symbols using text as well.

The set of flowchart symbols and their associated attributes used in the analysis are presented in table 2.1[5].

Table 2.1 Flowchart Symbols Used

Document (RESCSYM) -- Represents paper documents and reports of all types.
Attributes: Name.

Accounting Records (RESCSYM) -- Represents accounting journals and ledgers.
Attributes: Name.

Manual process (ACTSYM) -- Represents any process that is done manually and causes a change in value, form, or location of information.
Attributes: Agent, Act, Object.

Off-line storage (RESCSYM) -- Represents off-line storage of documents, records, and EDP files.
Attributes: Name, Filing Method.

Input/output (ACTSYM) -- Used to indicate information entering or leaving system.
Attributes: Agent, Act, Object.

Direction line (AUXSYM) -- Used to show the direction of processing or data flow.

System flowcharts contain information about transaction processing systems at the activity level.

---

[5] Based on flowchart symbols commonly found in auditing textbooks (Arens and Loebbecke, 1988, and Kell and Boynton, 1992)

Agents, activities and data objects are directly represented with flowchart symbols. Each agent is related with one or more activities, and each activity acts upon one or more data objects. The sequence of the activities, the transportation and repositories of data objects are also directly represented.

More detailed transaction information, however, is not directly represented and therefore is exogenous to the flowchart model. For example, transaction attributes related with each data object, such as quantity, price and date, are not directly represented. This presents the following problem: at what level of detail should the information be analyzed? The answer depends on the requirements of the target model. In this project, reliability network is chosen as the target model and it requires information at activity level – the relevant decision is to determine whether an activity is a control activity or a processing activity and whether the completion of an activity succeeds or fails. The model does not require detailed information such as quantity, price and date on a data object. As a result, these details are deemed unnecessary for the analysis.

The propositions describing a flowchart will be the following:

```
1. general functions:
     node_type(x,a).
     previous_node(x,y).
     next_node(x,z).


2. functions specific to resource symbols:
     name(x,a).

3. functions specific to activity symbols:
     agent(x,a).
     act_name(x,a).
     act_obj(x,a).
     activity_type(x,a).

4. other functions:
     connect(z,x,y).
     start_node(x).
     end_node(y).
     path(x,y,p).
```

where

x, y, z are variables for nodes,
a and p are variables for attributes.

These functions can be used to extract descriptive information in the source model and they translate the source descriptive information into decision information in the target model. For example, an output information repository is a target model resource, a translation rule that creates an output information repository in the target model can be written as follows:

```
resource(x),
name(x,a),
end_node(x)
   ->
      output_information_repository(x,a).
```

2.5 Summary

In this chapter both activity-based analysis and event-based analysis are introduced. Both the DEAS model and the flowchart model are examined to provide a basis for information extraction from such documentation. In the next chapter, a translation theory is developed as a guide for model conversion.

Chapter 3 THE TRANSLATION THEORY

3.1 Introduction

Descriptive models cannot be tailored to decision models since several decision models may be used in an audit while only a single descriptive model is practically possible. Furthermore, since only physical concepts can be verified objectively, descriptions of physical models becomes paramount in the primary documentation.

Since physically observed transaction processing systems differ from logical sequences of activities, the translation activity must transform physical concepts to multiple logical sequences of events. Since objects and relationships are observable and are named, the translation activity translates the names and attributes of observable objects into propositions describing activities, events, and the structure of activity and event networks.

One of the problems with mechanical translation is that the level of the generated decision information will depend upon the level at which a system is observed and described. It is reasonable to expect that auditors will describe systems at the level appropriate for control analysis.

It is not reasonable to expect auditors to follow a set of flowcharting rules with constant and complete rigor. In auditing, flowcharts are not used as formal languages with rigid syntax and semantics. They are looser forms of language with structured rather than formal syntax and semantics. As a consequence, the translation technique must take into account auditor omissions and mistakes in diagrams occasioned by the lack of formality in the syntax. In the technique described in this chapter, omissions are handled with defaults and mistakes identified by informal syntax checking.

In this chapter, I present a theory of structured translation from flowcharts to reliability networks and event networks in the form of solutions to the above.

3.2 Basic Extraction of Activities and Events

3.2.1 Rewrite Rules for Flowchart Symbol Classification

The basic elements in flowcharting are flowchart symbols. Flowchart symbols can be classified into three categories: 1) activity symbols, 2) resource symbols, and 3) auxiliary symbols. A set of rewrite rules for classification is included in Appendix A and an example is shown below:

node_type(x,manual_process) -> activity_symbol(x).

where

x is a variable for flowchart nodes,
manual_process represent the type of node,
node_type and activity_symbol are predicates that describe model objects.

This classification of flowchart symbols into general categories maintains the flexibility of the rewrite rules. When new and unknown symbols are encountered, as long as they can be grouped into one of the categories, the rewrite rules written for the general categories need not be changed.

3.2.2 Rewrite Rules for Translation of Direction Lines

There are several ways an activity can be represented in the flowchart. An activity can be directly represented through an activity symbol, such as a manual process or a computer process symbol, or it can be implied by a direction line. For example, a direction line that connects a document symbol and an off-line storage symbol implies a filing activity.

When an activity is directly represented by activity symbols, the nature of the activity is determined by the activity attributes, such as agent, activity type, and act obj[6].

---

[6] The term "act obj" is used to refer to the object of an action, for example, "sales order" in the phrase "sales clerk prepares sales order". It is

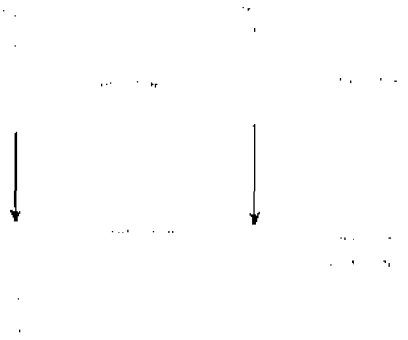
The analysis of the direction line depends on the symbols it connects. Four basic cases are identified and the rewrite rules for each are presented and discussed below.

1) The direction line connects a resource symbol to another resource symbol when a) the second symbol is not an end node, in which case the direction line represents resource transfer; and b) the second symbol is an end node, in which case the direction line represents information storage (see fig. 3.1 for diagrammatic representation examples). In both cases an activity object and the associated input and output resource objects are created and their values assigned. The difference between case a) and case b) is that in case a) the output communication channel is assigned a value, whereas in case b) the output information repository is assigned a value.

---

differentiated from the term "activity object", which refers to model objects, such as those in a DEAS model.

Figure 3.1 Examples of Diagrammatic Representation for Cases (1a) and (1b)

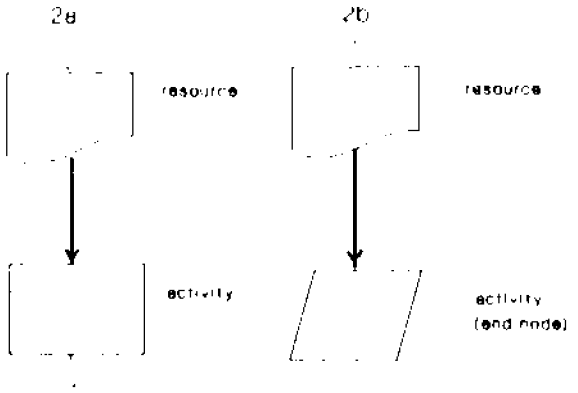


The rewrite rules for cases (1a) and (1b) are as follows:

```
1a) connect(z,x,y),
    resource(x),
    resource(y),
    auxiliary(z),
    name(x,a),
    name(y,b)
                -->
                    input_communication_channel(z,a),
                    output_transaction_attribute(z,b),
                    output_communication_channel(z,b),
                    output_information_repository(z,nil).

1b) connect(z,x,y),
    resource(x),
    resource(y),
    auxiliary(z),
    name(x,a),
    name(y,b),
    end_node(b)
              -->
                 input_communication_channel(z,a),
                 output_transaction_attribute(z,b),
                 output_information_repository(z,b),
                 output_communication_channel(z,nil).
```

2) The direction line connects a resource symbol with an activity symbol, in which case the resource symbol represents the input to the activity being modeled and the name of the resource is used to instantiate the input communication channel (case 2a). When the second activity symbol is an end_node, then the rewrite rule assumes that the transaction information is processed and to be communicated further (case 2b) (see fig. 3.2 for a diagrammatic representation).

Figure 3.2 Examples of Diagrammatic Representation for Cases (2a) and (2b)

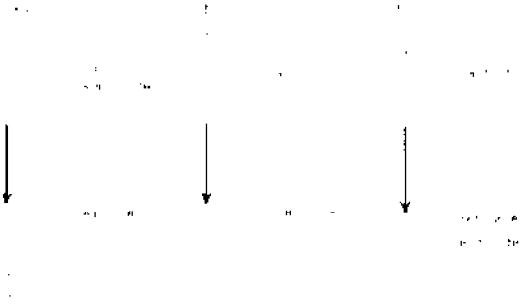


The two rewrite rules for cases (2a) and (2b) are shown below:

```
2a) connect(z,x,y),
    resource(x),
    activity(y),
    auxiliary(z),
    name(x,a)
              ->
                 input_communication_channel(y,a).

2b) connect(z,x,y),
    resource(x),
    activity(y),
    auxiliary(z),
    name(x,a),
    end_node(y)
              ->
                 input_communication_channel(y,a),
                 output_transaction_attribute(y,a),
                 output_communication_channel(y,a),
                 output_information_repository(y,nil).
```

3) The direction line connects an activity symbol to a resource symbol, in which case the resource symbol represents the output of the activity being modeled. The name of the resource is used to instantiate the output transaction attribute. The values for the output communication channel and the output information repository again depend on whether the resource symbol is an end node (cases 3b and 3c). If it is an end node, the output information repository is assigned a value with the name of the resource. Case (3a) represents a special case where the activity symbol is also a start_node, in which case the value of the input communication channel is assigned with the value of the act obj (see fig. 3.3 for a diagrammatic representation).

Figure 3.3 Examples of Diagrammatic Representation for Cases (3a), (3b) and (3c)



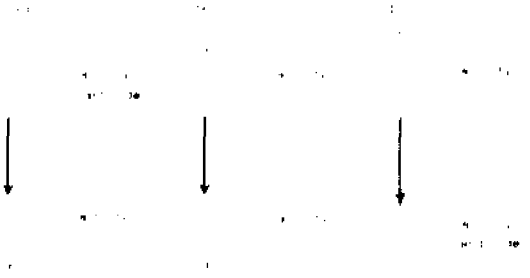
The rewrite rules for cases (3a), (3b) and (3c) are as follows:

```
3a) connect(z,x,y),
    activity(x),
    resource(y),
    auxiliary(z),
    name(y,a),
    start_node(x),
    act_obj(x,b)
              ->
                   input_communication_channel(x,b),
                   output_transaction_attribute(x,a),
                   output_communication_channel(x,a),
                   output_information_repository(x,nil).

3b) connect(z,x,y),
    activity(x),
    resource(y),
    auxiliary(z),
    name(y,a),
              ->
                   output_transaction_attribute(x,a),
                   output_communication_channel(x,a),
                   output_information_repository(x,nil).
```

```
3c) connect(z,x,y),
    activity(x),
    resource(y),
    auxiliary(z),
    name(y,a),
    end_node(y)
            ->
                    output_transaction_attribute(x,a),
                    output_communication_channel(x,a),
                    output_information_repository(x,nil),
                    input_communication_channel(z,a),
                    output_transaction_attribute(z,a),
                    output_information_repository(z,a),
                    output_information_repository(z,nil).
```

4) The direction line connects one activity symbol to another activity symbol, in which case the output of the first activity becomes the input of the second activity. Similar to case (3a), case (4a) represents a special case where the first activity symbol is a start node, in which case the value of the input communication channel takes the value of the activity object. Case (4b) represents a case where the second node is not an end node whereas the second node in case (4c) is an end node. The output resources for both activities are assigned values. (see fig. 3.4 for a diagrammatic representation).

Figure 3.4 Examples of Diagrammatic Representation for Cases (4a), (4b) and (4c)



The rewrite rules for cases (4a), (4b) and (4c) are as follows:

```
4a) connect(z,x,y),
    activity(x),
    activity(y),
    auxiliary(z),
    start_node(x),
    act_obj(x,a)
              ->
                   input_communication_channel(x,a),
                   output_transaction_attribute(x,a),
                   output_communication_channel(x,a),
                   input_communication_channel(y,a).

4b) connect(z,x,y),
    activity(x),
    activity(y),
    auxiliary(z),
    input_communication_channel(x,a)
              ->
                   output_transaction_attribute(x,a),
                   output_communication_channel(x,a),
                   input_communication_channel(y,a).
```

```
4c) connect(z,x,y),
    activity(x),
    activity(y),
    auxiliary(z),
    end_node(y),
    input_communication_channel(x,a),
              ->
                   output_transaction_attribute(x,a),
                   output_communication_channel(x,a),
                   output_information_repository(x,nil),
                   input_communication_channel(y,a),
                   output_transaction_attribute(y,a),
                   output_communication_channel(y,a),
                   output_information_repository(y,nil).
```

3.2.3 Discussion of Special Cases

The application of the rewrite rules assumes a single sequence of flowchart symbols, with each flowchart node representing a single resource, a single activity or an auxiliary node. Following are special cases that are often found in flowcharts and discussions on how symbols are treated in each case.

1) Due to the need to communicate transaction information to various agents, multiple copies of transaction documents are often used. A flowchart symbol that describes multiple copies of a document is treated as if it contains the same transaction information, unless a different name is assigned in the original flowchart to acknowledge additional processing on the copies. For example, a sales order approved is a name given only to those copies of the sales order that contain the credit

approval. In the case where a different name is assigned to different copies of the same document, they are treated as separate documents, *i.e.*, they contain different transaction information.

2) When a flowchart symbol includes multiple copies of multiple documents, it is split into several nodes each representing a single document regardless of the number of copies. This only occurs if each document leads to a different node. Otherwise, the multiple documents are treated as one node with a name that reflects all the documents.

3) When multiple activities are represented with only one activity symbol, the activity is split into several activity symbols each representing a separate activity.

4) When multiple direction lines lead to one activity, each direction line is processed separately, resulting in multiple input resources to the activity.

5) When multiple direction lines come from an activity, the output resources of the activity are assumed to be the same while each direction line is processed separately.

Connector symbols are special cases of the direction line. Each pair of connectors can be collapsed into a direction line through the following rewrite rule:

```
node_type(x,on_page_connector),
name(x,a),
end_node(x),
node_type(y,on_page_connector),
name(y,a),
start_node(y)
              ->
                   connect(z,x,y).
```

Similar rules can be written for off page connectors.

3.2.4 Rewrite Rules for Event Extraction

Since the DEAS model restricts the set of events that can happen with each activity, once the activities and the associated resources are defined, possible event sets can be computed. Thus event extraction is model driven. The main complex event is that the transaction attribute is a function of the input, and the output message and record are the same as the transaction attribute. The rewrite rule for extracting events from DEAS model objects is as follows:

```
activity(x),
input_communication_channel(x,a),
output_transaction_attribute(x,b),
output_communication_channel(x,c),
output_information_repository(x,d)
                                ->
                                  event(y),
                                     y = function(b,a),
                                  event(z),
                                     z = same_as(c,b),
                                  event(w),
                                     w = same_as(d,b).
```

The set of events can be explained as follows: that the transactions are being processed according to the

transaction policies, and the transaction information communicated and stored is the same as the transaction information generated.

3.3 Naming of Activities and Events

3.3.1 Naming of Activities

Naming is a special and very important aspect of translation since auditors will expect to see names for activities and events. In naming and describing an activity, the following questions need to be answered: 1) Who is the actor of the activity? 2) What is the type of activity? 3) What is the object of the activity? 4) If it is a document transfer activity, what are the source and destination of the transferring activity?

The name of the actor is needed so that a sales order received by the shipping department can be distinguished from a sales order received by the billing department. The type of activity is useful for distinguishing processing and control activities. The object of the activity is useful for determining where the transaction information is recorded or stored. The source and destination are needed to determine where the input comes from and where the output goes to when there is an information transfer either through physical transportation of documents or transfer through electronic means, such as a modem. The source and

destination attributes are needed only when the transfer of a document or transaction information is between agents or to and from agents outside of the organization.

An activity therefore is characterized by the following attributes: agent, activity type, act obj, source and destination. Naming of an activity is accomplished by listing out its attributes. For example, an activity can be named as "bookkeeper #1 sends approved sales order to bookkeeper #2". Included in this name are the five activity attributes with assigned values, namely, agent -> bookkeeper#1, activity type -> send, act obj -> approved sales order, source -> bookkeeper#1 and destination -> bookkeeper#2.

3.3.2 Naming of Events

Naming of events is based on the values assigned to the input and output resources in the DEAS model. For example, assuming the following values are assigned to the input and output resources:

input communication channel: approved sales order
output transaction attribute: shipping advice
output communication attribute: shipping advice
output information repository: shipping advice

Three basic events can be extracted:

e1: the shipping advice is processed according to the information on the approved sales order.

e2: Information on the shipping advice sent is the same as the shipping advice prepared.

e3: Information on the shipping advice stored is the same as the shipping advice prepared.

Each of the events listed is represented by a proposition that can be true or false. A complex event set can be generated based on the basic events to include combinations of the true state of any of the three events and the false state of any of the three events. The entire event set is restricted by the DEAS model as discussed in chapter 2.

In order to distinguish the sales order sent to shipping department and the sales order sent to billing, the activity attributes that are associated with each event again need to be analyzed. A more detailed representation of the basic events e1, e2 and e3 as discussed above is given as follows:

e1: shipping advice (prepared by shipping department) is processed according to the information on the approved sales order (received from credit department).

e2: Information on the shipping advice sent (to billing department) is the same as the shipping advice prepared (in shipping department).

e3: Information on the shipping advice stored (in shipping department) is the same as the shipping advice prepared (by the shipping department).

3.4 Defaults

3.4.1 When Defaults Are Necessary

An auditor is unlikely to represent systems with complete and correct syntactic efficiency. Thus the translation technique should include knowledge for constructing defaults given syntactic variations.

Good flowcharting techniques include some of the following as suggested by Arens and Loebbecke (1988): 1) use specified symbols, 2) use flowlines (direction lines), 3) show separation of duties, 4) include relevant controls, 5) include written comments and clarification, 6) show the source of every document in the flowchart, 7) use a process symbol for every document or record prepared, 8) show the disposition of every document in the flowchart.

Not using good flowcharting techniques will result in flowcharts with incomplete or invalid syntax. To facilitate discussion, the techniques as suggested by Arens and Loebbecke are grouped into three categories. In the following paragraphs, the different treatments for each group when the techniques are not followed will be discussed.

The first group includes 1) use specified symbols. Use of specified symbols is assumed in this dissertation for the flowcharts to be examined. If there is variation in the

use of flowchart symbols, it will be converted to the restricted set used in this dissertation.

With the development of computer technology, more flowcharts are prepared by using computer software than using manual methods. The set of symbols allowed in flowcharting is usually restricted in computer software. The possible variations will include using the same symbol to describe different entities. For example, the input/output symbol is sometimes used to describe accounting ledgers or journals. The translation process discussed in this dissertation assumes that the set of flowchart symbols used is restricted and conforms to the ANSI standard.

The second group includes 3) show separation of duties, 4) include relevant controls, and 5) include written comments and clarification. The analysis with regard to such information as segregation of duties and relevant controls assumes the nonexistence of such information if it is not shown in the flowchart. Since comments usually take free text style and the information provided is mostly for clarity and is of supplementary nature, the analysis of the annotation symbol is not dealt with in this dissertation and such omission is believed not to be detrimental to this study of translation.

The third group include 2) use flowlines (direction lines), 6) show the source of every document in the flowchart, 7) use a process symbol for each document or record prepared, and 8) show the disposition of every document in the flowchart. The suggested techniques in this group if not followed will result in flowcharts with incorrect or incomplete syntax, and such syntax is dealt with by using defaults.

3.4.2 Examples of Default Applications

Following are examples of flowchart segments with incomplete or incorrect syntax, and the default treatments.

1) No process symbol is used for document preparation. A document that is not immediately preceded by a document preparation activity may appear on a flowchart -- the default assumes a preparation activity that precedes the document, given that the document has not been mentioned in all of the previous activity sequences and no other indication that the document comes from outside.

2) Missing a direction line. When a document is created by one agent and is processed by another agent, but there is no direction line showing the transfer of the document, the default will assume that such transfer exists between the two agents.

3) Missing document symbols. A series of activity symbols are used one following the other, without explicitly showing the input and output for each activity. The default rule assumes the following: the first activity has the associated input resources and the last activity has the associated output resources. Except for the last activity, the output resources for all the other activities are calculated as follows: the output transaction attribute and the output communication channel are assigned value(s) of the object(s) generated by the activity. For example, if the activity is "prepare invoice", then the assigned value for the output resources would be "invoice". The default also assumes that the output resources of a preceding activity also become the input resources of the succeeding activity.

4) Not showing disposition of document. If a document is created and processed and never filed, the default will assign a null value to the output information repository associated with the activities.

5) Missing activity attributes. For an information transfer activity represented by a direction line, the default assumes the same agent as the previous activity since no activity attribute is attached to each direction line. The default also assigns that same agent to be the

source and the agent for the next activity to be the destination.

6) Similarly, for an information storage activity represented by a direction line, the default also assumes the same agent as the previous activity.

Model-based defaults can be supplied since DEAS restricts the structure.

3.5 Structuring Activities and Events

Once the activities and events are identified, the activity and events network can be constructed with relative ease.

The construction of an activity network requires that the following computational processes be completed: 1) establishing propositions about an activity, 2) determining whether the activity is a processing activity or a control activity, and 3) determining the logical sequence of the activities.

The transaction processing and control activities identified as the DEAS model associated with the flowchart can be categorized into two groups: 1) those that are explicitly represented in a flowchart, *e.g.*, activities identified through a processing node; and 2) those that are implied by the direction line on the flowchart, *e.g.*, a filing activity and a document transfer activity.

Propositions about the first type of activity can be set up by using the activity attributes directly. For example, the following is a possible proposition: sales order department approves sales order. The following computational functions are used to set up the proposition: agent, activity type, and act obj. For the second type of activity (*i.e.*, those represented by the direction lines), the agent, the activity type and the act obj need to be identified first. Default rules are used to determine the agent and the activity type. The agent is identified to be the same as that of the previous activity. The act obj is identified to have the value of the input communication channel that is associated with the current activity. Also as a default, the source for the information transfer activity is the agent of the previous activity, and the destination is the same agent as that of the activity that follows. The type of activity can be determined by analyzing the output resources. For an information transfer activity, the associated output communication channel has the value of the message to be sent and the associated output information repository has a null value. For a filing activity, the associated output communication channel has null value and the associated output information repository has the value of the record.

The next step in constructing an activity network is to distinguish between serial nodes which represent processing activities and parallel nodes which represent control activities.

A processing activity contains processing procedures that create records of transaction information, change information on an existing record, communicate information from one agent to another, and store information by filing records. Processing activities may generate errors during the process. A control activity contains control procedures that fall into one of the two groups: 1) those that are necessary for the transaction initiation, processing, completion and recording, and 2) those that add controls to other processing procedures and without which the transaction can still be initiated, processed, completed and recorded. Control procedures in the first group cannot be separated from other processing procedures and the lack of them results in invalid transactions being completed or recorded. Control procedures in the second group aim at reducing errors created and the lack of them has effects on the error propensity of the transaction processing system. It is the control procedures in the second group that constitute the parallel nodes on the activity network. The control procedures in the first group are treated as equivalent to processing procedures. In the rest of the

discussions, the term "control procedure" is used to refer only to those included in the second group, *i.e.*, those which constitute the parallel node in the activity network.

The key issue in constructing an activity network lies in the identification of control procedures, and hence, of control activities. A control activity can be identified through its type, such as a matching of documents, or a review activity. The document or transaction information must have been created already. The control activity is supposed to detect and correct errors, given that such errors exist. The following is a rule for determining whether an activity is a control activity:

```
activity(x),
act_name(x,review),
input_communication_channel(x,a),
output_transaction_attribute(x,a),
output_communication_channel(y,a),
                          ->
                              activity_type(x,control).
```

This rewrite rule states that if an activity exists, it is a review activity and the value of the output transaction attribute is the same as the message in the input communication channel, *i.e.*, there is no new transaction information created by this activity, and the transaction information is created and communicated by a previous activity. In that case, this activity is classified into the control activity category.

As a comparison, a rewrite rule for a processing activity will look as follows:

```
activity(x),
act_name(x,create),
input_communication_channel(x,a),
output_transaction_attribute(x,b),
                          ->
                             activity_type(x,processing).
```

This rewrite rule states that if an activity exists and the value of the transaction attribute is different from the message in the input communication channel, *i.e.*, new transaction information is generated by this activity, then this activity is classified into the processing activity category.

Once a control activity is found, the next issue is to determine the processing activities to which it is in parallel. As a general rule, the control activity should be placed in parallel to all the previous activities that had processed – either created or modified – the input resources accessible to the control activity.

The logical sequence of the activities can be determined by connecting all propositions that describe processing activities with logical *ANDs* and by placing control activities in parallel, *i.e.*, using logical *ORs* to connect to the series processing activities they control.

The event network is constructed on the basis of the activity network. Each node on the activity network

represents an activity and is exploded into a set of events in parallel. The set of events is restricted by the DEAS model.

3.6 Summary

In this chapter, a theory of translation is presented. Problems in the extraction of activities and events and in the construction of activity networks and event networks are discussed. In the next chapter algorithms for the translation based on the theory presented in this chapter is developed.

Chapter 4 ALGORITHMS

4.1 Flowchart Symbol Representation and Related Functions

Flowchart information is represented as follows: each flowchart node is assigned an ID number which uniquely identifies that node. Each node is identified with a node type, such as a processing node or a document node. The relation of any node to its adjacent nodes is determined by its previous and next node, which in turn defines a direction line. This adjacency relation is used to determine the direction of information flows and the location of information repositories. Text labels are formalized by using attributes for each entity. Flowchart symbols are grouped into three categories through rewrite rules: resource symbol, activity symbol and auxiliary symbol.

Functions such as node_type(x,y), act_name(x,y), act_obj(x,a), connect(z,x,y), etc. are used to extract information from the flowchart representation. Given the value of one argument, these functions return the value of the other argument. For example, given node n1, we can use function "node_type" to find out its type, and use function "connect" to find out which node is connected to node n2. When both arguments are given values, then these functions

serve as predicates returning a truth value about the relationship.

4.2 The Translation Process

The translation is carried out in two steps: first, the flowchart model is translated into a DEAS model and then the DEAS model is used to derive activity and event networks.

4.2.1 Translation from Flowchart Model to DEAS Model

The translation process starts with the identification of a start node on the flowchart. Then a search of paths in the flowchart is conducted and each flowchart symbol including direction lines along the path is analyzed based on the rewrite rules discussed in the previous chapter. This process continues until all the flowchart symbols have been reached and analyzed. If there is more than one start node on the flowchart, the process will start again with the next start node. The DEAS model objects are established by applying the rewrite rules. The assignment of values for the model objects is also accomplished by applying the rewrite rules.

The following discussion describes the various stages of the translation process. The first step is the transformation of auxiliary symbols such as the on-page and

off-page connectors. Then each flowchart node is mapped to a DEAS object, *i.e.*, either a type of activity or a type of resource. The main processing is carried out with the direction line. Start from any node. If there are multiple direction lines that run from that node to other nodes, then each line is processed separately until none is left. Otherwise, only a single direction line has to be analyzed. Rewrite rules for direction lines are applied based on the category of nodes that precede and follow each direction line. An algorithm for activity extraction is presented in Figure 4.1.

Figure 4.1 Algorithm for Activity Extraction

1. Applying rewrite rules to link all on-page connectors.
2. 2.1 Repeat until no more start nodes are available,

   2.2 Get an unprocessed start node.

   2.3 Repeat until no more next nodes are available,

   Get next node and the direction line that links the current node and the next node. Determine the category of each node.

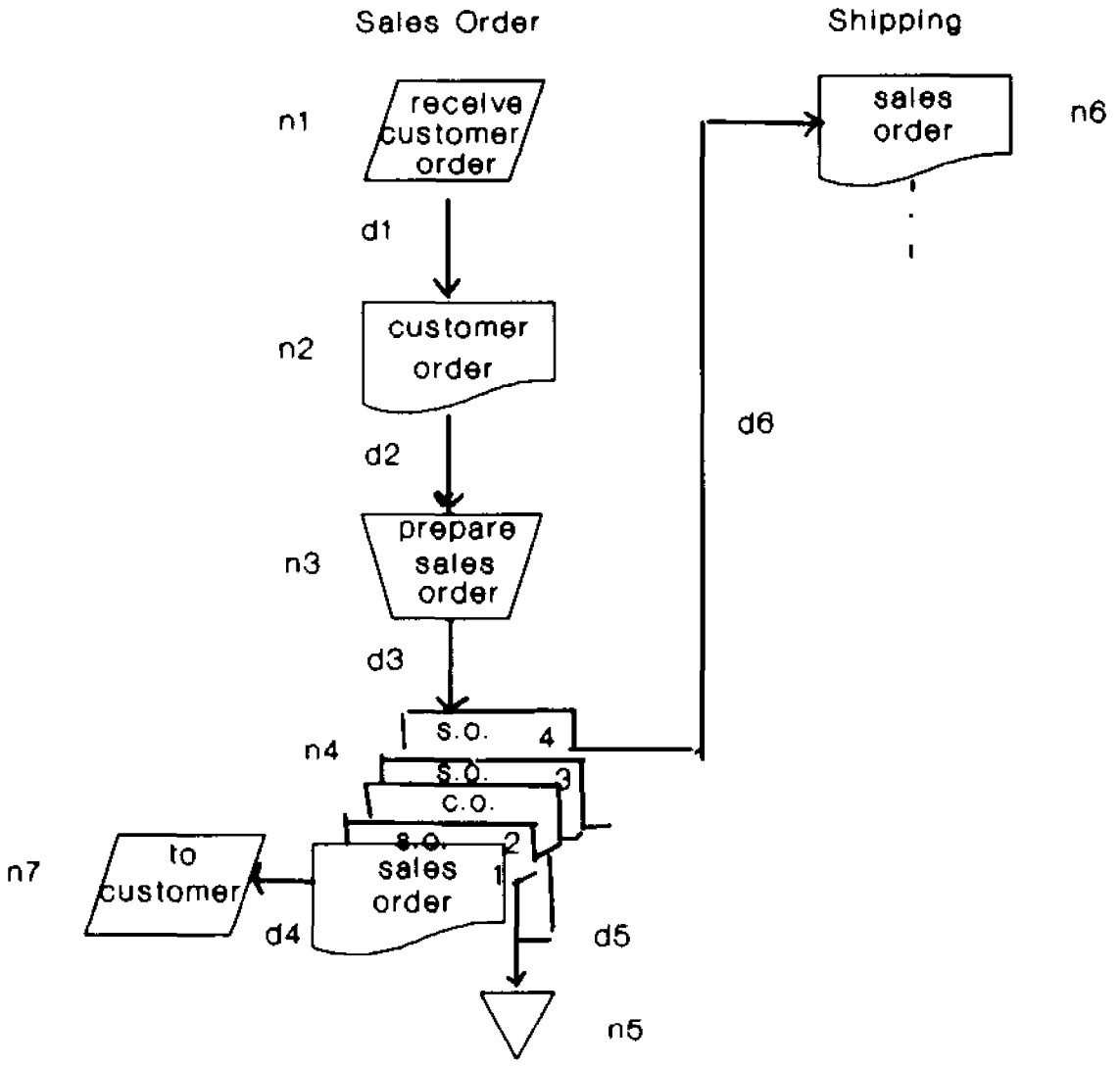
   If both nodes represent resources, then apply rewrite rule #1x (*i.e.*, rewrite rule 1a or 1b);
   if the first node represent resource and the second node represent activity, then apply rewrite rule #2x (*i.e.*, rewrite rule 2a or 2b);
   if the first node represent activity and the second node represent resource, then apply rewrite rule #3x (*i.e.*, rewrite rule 3a, 3b, or 3c);

```
        if both nodes represent activities, then apply
           rewrite rule #4x (i.e., rewrite rule 4a, 4b,
           or 4c);
        otherwise stop.

     Push the next node onto a queue.

2.4 Get a node from the queue,

        If it is unprocessed, go to step 2.3;
        otherwise, go to step 2.1.
```

Using the algorithm presented in fig. 4.1, all connectors are processed first and converted to direction lines. Next, each node is processed together with its successor node. The adjacent two nodes and the direction line that links them are analyzed to determine which rewrite rule for the direction lines applies (the rewrite rules are discussed in chapter 3). The rewrite rule is then applied and the program proceeds to the next unprocessed node until no more are left.

Naming of an activity is accomplished by forming propositions about the activity. Three examples are discussed in the following paragraphs. A flowchart segment is presented in fig. 4.2 and is used for all three examples.

Figure 4.2 Flowchart Segment of a Credit Sales System



1. For activities that have corresponding process nodes: functions are used to return activity attributes, *i.e.*, agent, activity type and act obj. The proposition for the activity is formed by joining the activity attributes as follows:

```
p1 = agent(n3,"sales order department") and
     act_name(n3,"prepare") and
     act_obj(n3,"sales order").
```

Name of the activity is obtained by concatenating the character strings used for naming the activity attributes: "sales order department prepare sales order".

2. For filing activities: filing activities are implied by direction lines and are identified by checking the value assigned to the output information repository. If it has non-null value, then the activity represents a storage of information. The proposition for a filing activity is constructed by default. The default agent is the same as the one in the previous activity, the default activity type is filing, and the default act obj is the value in the output information repository.

The following functions establish the proposition for a filing activity:

```
connect(d5,n4,n5).
resource(n4).
resource(n5).
agent(n4,"sales order department").
output_information_repository(d5,"sales order").
```

```
p2 = agent(d5, "sales order department") and
     act_name(d5, "file") and
     act_obj(d5, "sales order, customer order").
```

These functions are explained as follows: that the nodes n4 and n5 are connected by d5, both n4 and n5 are resource nodes. A filing activity is implied by the direction line d5 (rewrite rule 1b). The agent of the activity is "sales order department" and this activity creates a record of "sales order", which is being filed (a message is being stored in the output information repository). The proposition inferred from these functions includes the following: that the activity name is "file", the activity object is "sales order" and the activity agent is "sales order department". The name of the filing activity is obtained by concatenating the character strings used to describe the activity attributes: "sales order department file sales order".

3. For transaction information transfer activities: information transfer activities are those identified by direction lines and not by process symbols. The proposition for an information transfer activity is also constructed by default. The default agent is the same as the one in the previous activity; the default activity type is send; the default act obj is the value in the output communication channel. It is important to know the source and destination of the information transfer, so that the agent who sends

the information and the agent who receives the information are explicit. As a result, the source and the destination attributes are also added to the proposition by default. The default source is the same as the agent of the previous activity, and the default destination is the agent of the activity that follows. The following functions establish the proposition for a transaction information transfer activity:

```
connect(d6,n4,n6).
resource(n4).
resource(n6).
agent(n4,"sales order department").
agent(n6,"shipping department").
output_communication_channel(d6,"sales order").

p3 = agent(d6, "sales order department") and
     act_name(d6, "send") and
     act_obj(d6, "sales order") and
     source(d6, "sales order department") and
     destination(d6, "shipping department").
```

The functions are explained as follows: that the nodes n4 and n6 are both resource symbols and connected by d6. An information transfer activity is implied by the direction line d6 (rewrite rule 1a). A message is put to the output communication channel, and the transfer is from n4, the agent of which is "sales order department" to n6, the agent of which is "shipping department". The proposition inferred from these functions is that the activity name is "send", the activity object is "sales order", the activity agent is "sales order department", the source of the activity is

"sales order department" and the destination of the activity is "shipping department". Name of the information transfer activity is obtained by concatenating the character strings used to describe the activity attributes: "sales order department send sales order from sales order department to shipping department".

4.2.2 Translation from DEAS Model to Activity and Event Network

When each activity is named, there is one more step that needs to be accomplished, *i.e.*, to distinguish between processing and control activities. This step is necessary so that the logic of the activity network or the reliability network can be worked out. The algorithm for distinguishing processing activities from control activities and for constructing activity work is presented in fig. 4.3.

Figure 4.3 Algorithm for Activity Network Construction

```
1. Repeat until all activities are processed
     Get an activity.
     Match the activity type to the process activity list.

     If there is a match, push the activity to stack #1;
       otherwise, match the activity type to the control
          activity list.

        If there is a match, push the control activity to
          stack #2,

        otherwise stop.
```

2. Repeat until both stack #1 and stack #2 are empty.
   Pop a control activity from stack #2.
   Pop a processing activity from stack #1.

   Match the transaction attributes of both activities.

   If there is a match, join both activities with OR and push the resultant unit onto stack #3;

   if there is no match, pop from stack #3 and join it using AND with the processing activity, then push the result back onto stack #3.

The program keeps lists of known processing and control activity names. Each activity on the network is compared to the lists, first to determine if it is a processing or a control activity. Then, pairs of activities are compared to determine if they should be connected with logic AND or if a control activity is being processed. All the other processing activity nodes that are parallel to this control activity node are marked and together, these AND nodes join with the control activity node with a logical OR. This process continues until no more nodes are left unprocessed.

Once the activity network is established, it is not difficult to construct an event network. The algorithm for setting up propositions about events, naming each event and setting up the event network are discussed below:

1. For each processing activity, get the value of its associated transaction attribute (*e.g.*, sales order), and get the value of its associated input (for example, the

customer order), and assign the following value to event #1: that the value of the associated transaction attributes is a function of the value of the input.

2. For each processing activity, get the value of its associated transaction attribute, and the value for its output communication and its output information repository, then form event #2 and event #3 as follows: let event #2 be that the value of the output communication channel, given it is a non-null value, is the same as the value of the transaction attribute and let event #3 be that the value of the output information repository, given it is non-nil value, be the same as the value of the transaction attribute.

3. Given event #1, event #2 and event #3, generate a table of complex events, consisting of all possible combinations of the truth value of each of the events. The general format of the table was discussed in chapter two.

As a result, each activity node on the activity network is exploded into a set of at most eight parallel nodes representing possible events (see fig. 2.3 in chapter 2). The overall structure of the network remains the same.

The number of possible events can be less than eight, which is the complete set of events. When the value in the output communication channel or the output information

repository is null, parts of the event set will never assume a true value and therefore they become unusable.

4.3 Defaults

The default rules are applied at various stages during the translation.

Because there are no activity attributes connected with direction lines, whenever direction lines are translated into activities, defaults are used to supply values for the activity attributes. The following describes the algorithm for the application of the default.

If a direction line implies an information transfer activity, then the default rule will be such that the agent and the source of such activity take the value of the agent of its previous activity, the activity type will take the value of "send", the act obj will take the same value as the output communication channel of the previous activity and the destination will take the value of the agent of the next activity.

When direction lines are missing, the result is a break in the flow of the document or of the information. A document or an activity symbol will be incorrectly shown as a start node. To supply default values for such missing direction lines, some determination rules are applied whenever a start node is encountered in order to eliminate

the possibility of a false start node. A description of the algorithm for such an analysis is presented in figure 4.4.

Figure 4.4 Algorithm for Default Analysis

1. Get a start node.

   1.1 If it is a document node, determine if there is another document node with the same attribute (same name), that is not a node subsequent to the current node. If such node exists, then this node is not a true start node;

   otherwise, it is a start node.

   1.2 If it is an activity node, determine if the act obj exists as a separate node which is not a node subsequent to the current node. If such node exists, then this node is not a true start node;

   otherwise, it is a start node.

   1.3 If a node originally appeared as a start node and has been marked as not being a start node, perform the following procedure:

      1.3.1 For a document node which is a nonstart node, the default assumes that the document is sent by the agent who prepares the document: find the activity with the following attributes:

      activity type =
            "prepare" or equivalent
      act obj =
            the name of the document

      Create a direction line between the output node of the activity and the document.

1.3.2 For a document node which is a true start node, the source of the document has to be determined. The knowledge base will include a list of commonly found documents that come from outside. The name of the current document name is compared to the list. If there is a match, then an input/output symbol is created and added in front of the document node via a direction line.

1.3.3 For an activity node which is a nonstart node, match its act obj with the value of the output communication channel of some prior activity and let the input communication channel of the current activity node take the same value.

4.4 Summary

In this chapter, algorithms for extracting activity and event information from system documentation are discussed. Algorithms for applying default rules are also discussed. In the next chapter, an example will be provided for testing the theory and the algorithm.

Chapter 5 EXAMPLE AND TESTING

5.1 Introduction

In this chapter, the theory of mechanical extraction of activity and event information from system documentation is tested. The test is carried out with a flowchart given in a Certified Public Accountant Examination (Kell & Boynton 1992, 514-515). This flowchart describes a revenue and cash receipts transaction. CPA candidates were asked to identify control weaknesses in the system. The following reasons account for the choice of this flowchart in this dissertation: 1) all the information regarding the transaction accounting and control system is given in the flowchart; inferences are made based on this information; 2) the level of task difficulty is medium, given the time constraints on the exam; 3) a certain level of accounting and auditing knowledge is required of the candidates; 4) the flowchart describes a system with design weaknesses. As a result, testing on this flowchart will reveal to us 1) adequacy of the contents and structure of the knowledge base for model construction; 2) knowledge required to extract transaction information, and 3) the usefulness of the DEAS model in analyzing transaction activities and events. Results produced by computation will be compared to

the model answers supplied by the AICPA. The CPA flowchart is shown as Figure 5.1.

Since we allow auditors to have flexibility in drawing flowcharts, the translation program has to have the ability to deal with flowcharts with syntactic variations. To test the robustness of the default rules that deal with syntactically different inputs, a sensitivity analysis is conducted on the flowchart, given syntactic variations.

5.2 Using DEAS Model to Extract Activity and Event Information

The DEAS model implicit in the flowchart is first extracted from the flowchart. The DEAS model objects are constructed and object values assigned.

The results of this process are presented in table 5.1. The headings are explained as follows: DR LN lists all the direction lines that were processed. RL lists the rewrite rules that were activated during the translation process. ACT ID shows the symbol id used for each activity object in the DEAS model, and ACTIVITY NAME gives an abbreviated name for each DEAS activity object. IN-CC, OUT-TA, OUT-CC, OUT-IR list the values assigned to the input and output resources in the DEAS model.

Figure 5.1 Flowchart of Credit Sales and Cash Receipts Activities

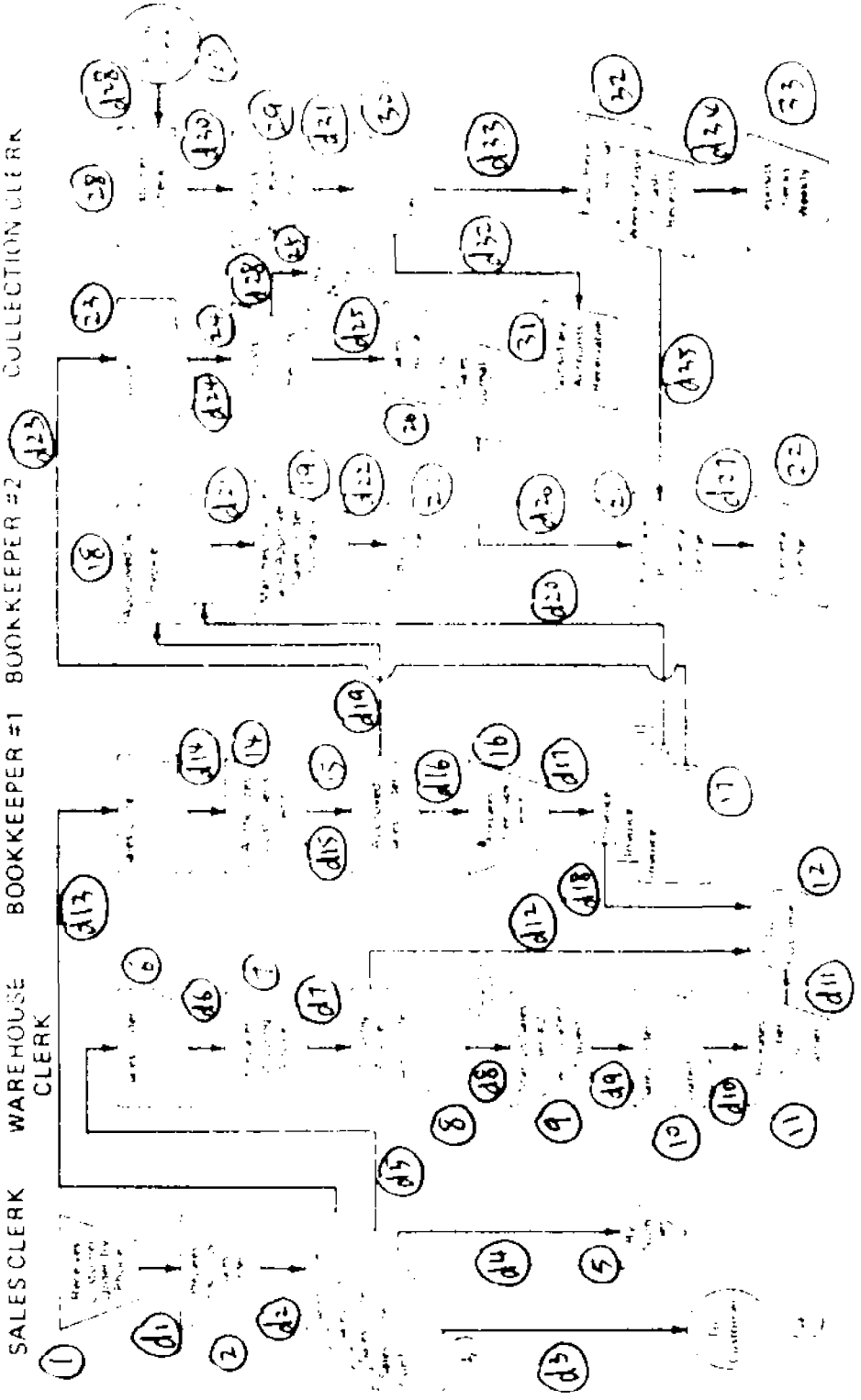


Table 5.1 DEAS Model Objects

DR LN	RL	ACT ID	ACTIVITY NAME	IN-CC	OUT-TA	OUT-CC	OUT-IR
d1	4a	n1	REC C/O	C/O	C/O	C/O	NIL
d2	3b	n2	PREP S/O	C/O	S/O	S/O	NIL
d3	2b	n4	SD S/O TO CST	S/O	S/O	S/O	NIL
d4	1b	d4	FILE S/O	S/O	S/O	NIL	S/O F
d5	1a	d5	SD S/O TO WH	S/O	S/O	S/O	NIL
d6	2a	n7	PREP SHIP	S/O	SHIP//S/O	SHIP//S/O	NIL
d7	3b						
d8	2a	n9	SHIP W DATE	S/O	S/O D	S/O D	NIL
d9	3b						
d10	2a	n11	RELSE LUMBER	S/O D	S/O D	S/O D	NIL
d11	4c	n12	SEND TO CST	S/O D*SHIP*INV	S/O D*SHIP*INV	S/O D*SHIP*INV	NIL
d12	2b						
d13	1a	d13	SD S/O TO BK1	S/O	S/O	S/O	NIL
d14	2a	n14	APPR CRDT	S/O	A S/O	A S/O	NIL
d15	3b						
d16	2a	n16	PREP INV	A S/O	INV	INV	NIL
d17	3b						
d18	2b						
d19	1a	d19	SD A S/O TO BK2	A S/O	A S/O	A S/O	NIL
d20	1a	d20	SD INV TO BK2	INV	INV	INV	NIL
d21	2a	n19	MCH INV & A S/O	A S/O*INV	A S/O*INV	A S/O*INV	NIL
d22	3c	d22	F A S/O & INV	A S/O*INV	A S/O*INV	NIL	A S/O*INV F
d23	1a	d23	SD INV TO CL CK	INV	INV	INV	NIL
d24	2a	n24	PST TO RCD	INV	SJ*DCPSJ*INV	SJ*DCPSJ*INV	NIL
d25	3b						
d26	2a	n21	FOOTS AND POSTS	SJ*DCPSJ*CRJ *WCPCRJ	G/L	G/L	NIL
d27	3c	d27	FILE G/L	G/L	G/L	NIL	G/L F
d28	3c	d28	FILE INV	INV	INV	NIL	INV F
d29	3a	n27	REC CST CHK	CST CHK	CST CHK	CST CHK	NIL
d30	2a	n29	FOR DEPST ONLY	CST CHK	CST CHK'	CST CHK'	NIL
d31	4a	n30	POST TO RECD	CST CHK'	CST CHK*SUB A/R *CRJ*WCPCRJ	CST CHK*SUB A/R *CRJ*WCPCRJ	NIL
d32	3c	d32	FILE SUB A/R	SUB A/R	SUB A/R	NIL	SUB A/R F
d33	3b						
d34	2b	n33	DPST CHK WKLY	CRJ*WCPCRJ	CRJ*WCPCRJ	NIL	CRJ*WCPCRJ
d35	2a						

Based on the DEAS model, an activity network is created. The propositions included in the activity network are presented in table 5.2. The activity network that shows the logical relations of the activity propositions is presented in figure 5.2. An event network is also created based on the activity network. A partial event network is presented in figure 5.3.

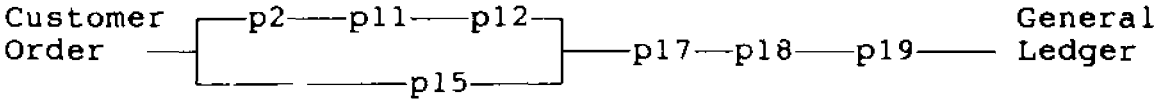
Table 5.2 Propositions about Activities

p1	Sales clerk receives customer order.
p2	Sales clerk prepares sales order.
p3	Sales clerk sends sales order to customer.
p4	Sales clerk files sales order.
p5	Sales clerk sends sales order to warehouse clerk.
p6	Warehouse clerk prepares shipping advice.
p7	Warehouse clerk stamps sales order with date shipped.
p8	Warehouse clerk releases lumber to carrier.
p9	Warehouse clerk / bookkeeper #1 send dated sales order, shipping advice and invoice to customer.
p10	Sales clerk sends sales order to bookkeeper #1.
p11	Bookkeeper #1 approves credit.
p12	Bookkeeper #1 prepares invoice.
p13	Bookkeeper #1 sends approved sales order to bookkeeper #2.
p14	Bookkeeper #1 sends invoice to bookkeeper #2.
p15	Bookkeeper #2 matches approved sales order with invoice.
p16	Bookkeeper #2 files approved sales order and invoice.
p17	Bookkeeper #1 sends invoice to collection clerk.
p18	Collection clerk posts sales to sales journal.
p19	Bookkeeper #2 foots [sales journal] and posts (to general ledger).
p20	Collection clerk files invoice.
p21	Collection clerk receives customer check.
p22	Collection clerk stamps the check for deposit only.
p23	Collection clerk posts (cash receipts to cash receipts journal and accounts receivable subsidiary ledger).
p24	Collection clerk deposits checks.

The structure of the activity network depends on auditor's interest of activities and/or events and will vary based on the chosen propositions. Let's suppose the auditor has chosen p19 to be the proposition he/she is interested in, *i.e.*, he/she is interested in sales journal being footed and general ledger being posted for sales. The following is

a possible activity network generated based on that choice.

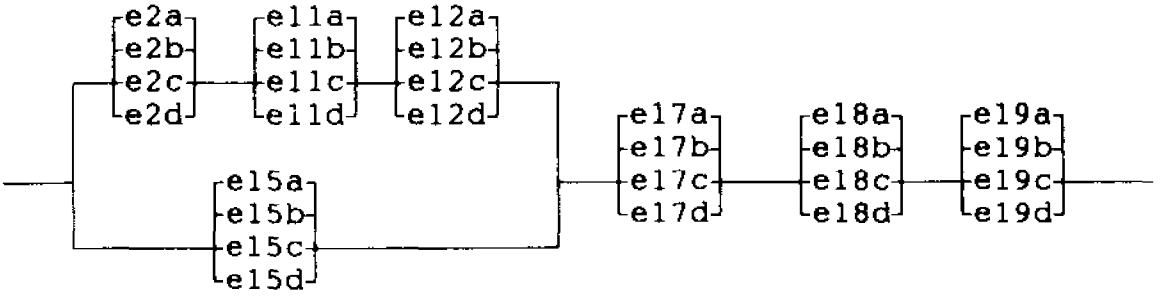
Figure 5.2 Activity Network



This computed network shows that in order for bookkeeper #2 to be able to foot the sales journal and post the sales to the general ledger (p19), sales clerk must have prepared sales order (p2), bookkeeper #1 must have approved credit (p11) and prepared invoice (p12), or bookkeeper #2 must have matched approved sales order with each invoice (p15), and bookkeeper #1 must have sent the invoice to collection clerk (p17), who must have posted sales to the sales journal (p18).

Figure 5.3 shows the event network generated from the activity network shown in figure 5.2.

Figure 5.3 Event Network Generated from the Activity Network



Following are some examples of the propositional statements attached to the events:

e2a : Sales order is prepared according to the customer order and sales order communicated is the same as the sales order prepared.

e15a : Transaction information on the invoice is the same as that on the approved sales order; or transaction information on the invoice is not the same as that on the approved sales order, but the error is corrected.

e17a : The invoice sent to the collection clerk is the same as the invoice received.

e18a : Transaction information on the invoice is correctly posted to the sales journal.

e19a : Transaction information in the sales journal is correctly posted to the general ledger.

5.3 The Use of Activity and Event Networks

The following discussion focuses on how activity and event networks can be useful to auditors.

The activity network has the same structure as a reliability network, and therefore, can be turned into a reliability network by finding the reliability of each processing unit that performs the activity. The activity network provides auditors with a set of relevant activities and their logical relations. The reliability of the processing unit of each activity has to be determined by the auditors. Once the reliability of each processing unit is found, then the reliability for the entire system can be computed from the reliability equations associated with the network structure.

The use of an event network is illustrated through an example. Suppose an auditor is interested in finding evidence for the following propositions:

1. sales order is approved.
2. goods are shipped.
3. sale is recorded.
4. recorded sales = goods shipped = goods ordered

Proposition #1 can be supported by a sales order with approval on it (APPSO). Proposition #2 can be supported by some shipping document (SHIP). Proposition #3 can be

supported by a record in the sales journal (SJ). Proposition #4 can be supported if the following equations hold true:

SJ = F(SHIP) and SHIP = F(APPSO).

The equations are interpreted as follows: that the recording of sales information in the sales journal is a function of the information on the shipping documents, which, in turn, is a function of the information on the approved sales order. Events such as those discussed above are of interest to the auditors and can be identified in the event network.

Several complex events (restricted to be at most eight for each activity by the DEAS model) can be generated from an activity network, but not all of them are possible events. For example, whenever the output information repository takes the value of nil and the output transaction attribute has a non-null value, those events become impossible that require the value in the information repository to be equal to the value of output transaction attribute (e1,e3,e5,e7)[7]. Similarly, whenever the output communication channel takes a null value, events e1,e2,e5,and e6 become impossible, which require the message in the output communication channel be the same as

---

[7] For the entire event set, refer to figure 2.3 in chapter two.

the value for output transaction attribute. Before the event network is set up, those impossible events will be identified and eliminated from the analysis. This elimination process reduces the number of events to be considered by the auditors. The event set can be further reduced depending on the auditor's interest. If an auditor is not interested in information being changed during a transfer, which is less likely in a manual system than in a computerized system, those events that deal with information transfer can be further eliminated from the events set.

Given the desired audit focus, it is useful to examine the results presented in table 5.1. Since the system is manual, we are not concerned about information change during a transfer process. Also assume that for all filing activities, the major concern will be that the document filed is the same as the document processed, *i.e.*, the value assigned to the output information repository is the same as the value assigned to the output transaction attribute. This leaves us with the following activities for consideration: n2, n7, n9, n14, n16, n24, n21, n30.

The common characteristics of these activities and associated events include the following:

1) They all have a null value in the associated output information repository. For these activities, therefore,

events e1,e3,e5,e7 become impossible because they require that r = o take the true value (*i.e.*, the record of the transaction attribute is the same as the transaction attribute).

2) The value assigned to the input communication channel is different than the value assigned to the output transaction attribute and the output communication channel. As a result, a basic event generated by these activities that is of interest to the auditors is that the value of the output transaction attribute is a function of the value in the input communication channel and be equal to the value in the output communication channel. Take activity n2 for example, the basic event is that the sales order is prepared according to the customer order and the sales order further communicated is the same as the sales order prepared. The other three events e4,e6,e8 are based on this basic event. The basic events for these activities are presented in table 5.3.

Table 5.3 Basic Events

Events for n2: the sales order is prepared according to the customer order and the sales order communicated is the same as the sales order prepared.

Events for n7: the shipping advice is prepared according to the sales order and the shipping advice communicated is the same as the shipping advice prepared.

Events for n9: sales order dated reflects the goods shipped and is based on the sales order, and the dated sales order communicated is the same as the one prepared.

Events for n14: approved sales order is based on the sales order and the approved sales order communicated is the same as the one approved.

Events for n16: invoice is prepared according to the approved sales order and the invoice communicated is the same as the invoice prepared.

Events for n24: sales journal and daily copy of sales journal is recorded based on the invoice and sales journal, daily copy of the sales journal and invoice are the same as the ones prepared.

Events for n21: general ledger is recorded according to the sales journal, daily copy of the sales journal, cash receipts journal and weekly copy of the cash receipts journal. General ledger communicated is the same as general ledger prepared.

Events for n30: subsidiary accounts receivable ledger is updated according to customer check with endorsement. subsidiary accounts receivable ledger communicated is the same as the subsidiary ledger prepared.

As discussed earlier, the auditor's interest is to determine whether recorded sales is equal to goods shipped, and also equal to goods ordered, or the equations

SJ = F(SHIP) and SHIP = F(APPSO) hold true. Based on the events generated, the event that sales recorded is equal to goods shipped can not be tested in this particular system because such an event is an impossible event in the system. The sales recorded in this system is only based on the invoice, which in turn is based on the approved sales order. The second equation can not be tested in this system either, because the event that goods are shipped according to approved sales is an impossible event in this system. Both the shipping advice and the dated sales order (which reflect the actual shipment) are based on the sales order, not on the approved sales order.

If this result be compared with the answer given in the CPA examination[8], the following conclusion can be drawn:

The activity network shows that no control activity is present in parallel to processing of shipping document, sales journal, cash receipts journal, accounts receivable ledger, and general ledger. The event network shows that 1) the invoice is prepared as a function of approved sales order (not shipping document), 2) shipment occurs as a function of sales order (not approved sales order), and 3)

---

[8] The CPA examination answer listed the control weaknesses found in the internal control structure of the lumber company (refer to Appendix B).

customer check (not remittance advice) is used for posting to accounts receivable subsidiary ledger and cash receipts journal. These results are compatible with the following answers given in the CPA examination:

* Warehouse clerk releases lumber prior to authorization.
* Bookkeeper #1 prepares and mails invoice without knowledge of what was shipped.
* Subsidiary accounts receivable ledger should be reconciled to general ledger.
* Deposit slips are not reconciled to cash receipts journal or general ledger.
* Remittance advice is not used as the basis for posting collections.

Several system's weaknesses deal with segregation of duties issues. For example, bookkeeper #2 who maintains general ledger should not be responsible for footing and crossfooting of journals (sales journals and cash receipts journals); collection clerk should not maintain sales journal; collection clerk should not maintain accounts receivable subsidiary ledger. Currently the DEAS model treats segregation of duties issue as an exogenous variable. The reliability network model incorporates this variable as affecting the reliability of the processing units of the activities. Because the main efforts of this research are to show the translation mechanism among models and not the reliability calculation, segregation of duties analysis is not included as a result. However, given that the activities and their agents are identified during the translation process, it is relatively easy to develop a set

of rules to perform the segregation of duties analysis. The main task of the rule set will be to identify any incompatible duties and to signal the user.

Two additional system weaknesses identified in the CPA examination deal with timing: 1) checks are not promptly endorsed by the mail clerk, and 2) cash receipts are not promptly deposited. Timing presents a control problem due to the possibility of loss of data or assets, which may result in an invalid transaction. The longer the wait between the two activities, the greater the chance of interruption or discontinuation in the information flow. Since DEAS model does not incorporate timing elements, these weaknesses are not identified. We assume that the wait between activities will not interfere with the information flow as documented.

5.4 Sensitivity Analysis

In order to test the robustness of the translation program, a sensitivity analysis has been performed. The following variations are used in the analysis: 1) omitted symbols, 2) discontinued flows, 3) incomplete labeling, 4) misuse of symbols.

Omitted symbols often result from simplification. For example, a process symbol should ideally show its input and output document as its preceding and succeeding symbols.

When processes are drawn one after another, some of the inputs to and outputs from the processes are omitted. When omitted symbols are detected, default rules will be applied first. If default rules fail, then any information that is not explicitly supplied or cannot be inferred by the default rules will be assumed nonexistent. Take fig. 5.1 for example. Assume n8 is omitted and assume a direction line linking n7 and n9. There is no problem with the input to the activity series, which is a sales order represented by n6. The output from the activity series presents a problem. The shipping advice can be inferred from the default rule by examining the act obj of process n7. However, unless the shipping advice is shown again as a symbol following n9, it will disappear, *i.e.*, neither be communicated nor be filed. To summarize, if a document symbol is omitted in between activities, the default rules will take care of it if it reappears later in the activity series. Otherwise, it will be shown as a transaction attribute but never be put into an output communication channel or into an information repository.

In addition to the omission of a document symbol (or a resource symbol in general), activity symbols could be omitted as well. One special case, where a document appears without preparation activity preceding the document, is taken care of by the default rule, which adds a document

preparation activity to the document. Following are some other cases where an activity symbol is omitted: 1) there is one document preceding the activity and a second document following the activity. Now with the activity omitted, a direction line is used to link the first document with the second document. In this case the default rule will create an activity, taking the first document as the input and the second document as the output of the activity. This rule is tested with n7 and n16 omitted.

Another case with an omitted activity symbol is when the activity is shown with an end node, representing either a document transfer or a document filing. When the activity symbol is omitted, the document symbol becomes the end node, in which case it is treated as a filing and not a transfer. As a result, removal of d5 and n5 has no effect on the interpretation of the DEAS model created, whereas remove of d4 and n4 will lead to a failure in creating a document transfer activity. If an end activity node is preceded by another activity node, and the end node is omitted, then the preceding activity node becomes the end node and is treated accordingly. Information in the original end node will be lost and thus not included in the DEAS model.

When a direction line is omitted where it should be shown, it results in a discontinued flow (of transaction

documents/transaction processing activities). When this happens, there will be more start and end nodes presented in the flowchart. In this case, the default rules will try to link the discontinued nodes together by either finding the same resource symbol used for a start node and an end node, or finding the resource symbol used as a start node and an activity that generates the resource, then link the output of that activity to the start node. When this fails, the information contained in the flowchart will only be partially processed and the DEAS model generated will be incomplete.

Currently, no mechanism exists to handle incomplete labels and misuses of symbols. The user will be signaled when certain labels and symbols cannot be processed. With current software technology, it should not be difficult to restrict the syntax of labels and the set of symbols to be used for flowcharting so that incomplete labels and misuse of symbols are kept to a minimum.

5.5 Summary

In this chapter, results from several tests performed on the feasibility of model translation were presented. Propositions found in the activities and events network were used to identify the weaknesses in the internal control structure. Segregation of duties and timing issues

are considered exogenous to the DEAS model and therefore were not included in it. A sensitivity analysis was conducted to test the robustness of the default rules. Situations where the default rules might fail were identified and discussed.

Chapter 6 SUMMARY AND CONCLUSIONS

6.1 Summary

This research focuses integrating descriptive and decision models of internal control evaluation. Descriptive models provide a view of the transaction processing system that is close to the physical system, *i.e.*, there is a one-to-one relationship between the observable physical objects and their relations as they are found in the business world, and the descriptive objects and relations as used in the model. Decision models use objects and relations at a higher and more abstract level. As a result, the decision objects and relations are not directly observable and require transformation from physical objects and relations. Current technology, which provides separate tools that support descriptive as well as decision models, does not, however, quite support the integration between descriptive and decision models.

The problem addressed in this research is the mechanical extraction of decision information from descriptive information. Descriptive information is first captured by identifying objects and their relations in the descriptive model. Then these descriptive objects and relations are mapped to decision objects and relations

through rewrite rules. Using declarative propositions to represent descriptive as well as decision objects and relations, integration of models has been accomplished through the translation of descriptive propositions to decision propositions.

Two target decision models are used: an activity-based model and an event-based model. An activity-based model is first established with the decision objects and relations. This model provides a mechanism with which the reliability of a transaction processing system can be calculated based on the reliabilities of the processing unit of each activity. An event-based model is then created based on the activity model. The event-based model provides a mechanism with which possible event sets can be analyzed and the probability of events in the system can be calculated.

The method used is not restricted by particular transaction cycles. As a model-based approach, the method aims at capturing the logic behind control analysis. It provides a mechanism to translate between formal models, thus facilitating the implementation of such models.

The model translation theory and the algorithms have been tested with an internal control evaluation problem. Results have been consistent with the suggested solutions. To test the robustness of the translation mechanism, the source descriptive model has been altered to conduct a

sensitivity analysis. Results from the sensitivity analysis show that the default rules work well when inconsistencies and incompleteness exist in the descriptive model.

It must be pointed out that the purpose of this research is not to replace auditors in decision making, but rather to provide support for gathering and structuring potentially relevant data and for modeling and experimenting with alternative strategies.

6.2 Conclusions

The translation theory and algorithm developed in this research demonstrate a model-based approach to support internal control analysis. By translating descriptive information about transaction processing and control systems into propositions used in decision models, auditors can document the system at a level where physical objects and their relations exist and are observable, and use decision models without incurring excessive costs to obtain model objects and relations.

Since the DEAS model provides semantics about transaction activities as well as events, the system analysis can be conducted at each level. This knowledge about activities and events also renders the DEAS model a promising intermediary model candidate for model integration.

The translation algorithm is not restricted by types of transactions, because the general translation theory applies to all systems. The naming of activities and events is the only part that differs from transaction to transaction.

6.3 Limitations and Future Research

The current translation algorithm do not include rules that deal with incomplete labeling (text). The problem is partly solved by pre-editing. Incomplete text information is treated similarly as the omission of symbols. Either it is assumed that the omitted information does not exist, or the missing information is supplied through pre-editing. To solve this problem, more knowledge about text analysis needs to be incorporated into the translation algorithm.

The current default rule set is fixed and not flexible, *i.e.*, the defaults do not change when unexpected circumstances are encountered. The translation program needs a learning element that can update the default rules as it gains experience from more cases.

Currently, the DEAS model is used to extract descriptive information from a single source model. It would be very useful if the DEAS model is enhanced to extract and integrate descriptive information from multiple source models, such as system flowcharts, internal control

questionnaires, and narratives. To solve this problem, knowledge about relations among different transaction types needs to be added to the knowledge base, and an algorithm of how to add and subtract a part of the DEAS model is also required.

So far the system analysis is restricted to the activity and event level. It can also be extended to agent level, where agent responsibilities and behavior are analyzed and the results of the analysis can be incorporated into the reliability or probability calculation of the DEAS model elements.

APPENDICES

APPENDIX A REWRITE RULES FOR FLOWCHART SYMBOL TRANSLATION

```
node_type(x,manual_process) -> activity_symbol(x).
node_type(x,computer_process) -> activity_symbol(x).
node_type(x,display_terminal) -> activity_symbol(x).
node_type(x,input_output) -> activity_symbol(x).

node_type(x,document) -> resource_symbol(x).
node_type(x,accounting_record) -> resource_symbol(x).
node_type(x,off_line_storage) -> resource_symbol(x).
node_type(x,transmittal_tape) -> resource_symbol(x).

node_type(x,direction_line) -> aux_symbol(x).
node_type(x,on_page_connector) -> aux_symbol(x).
node_type(x,off_page_connector) -> aux_symbol(x).
```

APPENDIX B ANSWERS TO THE CPA FLOWCHART PROBLEM

The weaknesses in Smallco Lumber's internal control structure are these:

a. Warehouse Clerk
   * Releases lumber prior to authorization, for example, approval of customer's credit.
   * Copies of shipping advice should be prepared and forwarded to Bookkeeper #1.
   * Lacks documentation that lumber was given to the carrier.

b. Bookkeeper #1
   * Credit authorized by bockkeeper and not a responsible officer.
   * Prepares and mail invoice without knowledge of what was shipped.

c. Bookkeeper #2
   * Bookkeeper who maintains general ledger should not be responsible for footing and crossfooting of journals, that is, sales and cash receipts journals.
   * Subsidiary accounts receivable ledger should be reconciled to general ledger.

d. Collection Clerk
   * Collection clerk should not maintain sales journal .
   * Collection clerk should not maintain accounts receivable subsidiary ledger.
   * Remittance advice not used as the basis for posting collections.
   * Checks are not promptly endorsed by the mail clerk.
   * Cash receipts are not promptly deposited.
   * Deposit slips are not reconciled to cash receipts journal or debits to general ledger.

REFERENCES

REFERENCES

American Institute of Certified Public Accountants. 1988. Consideration of the Internal Control Structure in a Financial Statement Audit. *Statement on Auditing Standards*. Auditing Standards Board, American Institute of Certified Public Accountants (April).

--. 1989. Proposed Audit and Accounting Guide for Consideration of the Internal Control Structure in a Financial Statement Audit. Exposure Draft. Audit Standards Board, American Institute of Certified Public Accountants (August).

Arens, Alvin A. and James K. Loebbecke. 1988. *Auditing: An Integrated Approach*, 4th ed. Englewood Cliffs, N.J.: Prentice-Hall.

Bailey, A.D, G.L.Duke, J.Gerlach, C.Ko, R.D. Meservy, and A.B. Whinston. 1985. TICOM and the Analysis of Internal Controls. *The Accounting Review* (April): 186-201.

Cushing, B.E. 1974. A Mathematical Approach to the Analysis and Design of Internal Control Systems. *The Accounting Review* (January): 24-41.

--. 1975. A Further Note on the Mathematical Approach to Internal Control. *The Accounting Review* (January): 151-154.

Elliott, Robert K. 1983. Unique Audit Methods: Peat Marwick International. *Auditing: A Journal of Practice & Theory* 2 (Spring): 1-12.

Gal, G. 1985. Using Auditor Knowledge to Formulate Data Model Constraints: An Expert Systems for Internal Control Evaluation. PhD dissertation. Michigan State University.

Grant Thornton. 1992. *INFOCUS: Information and Control Understanding System – User's Manual*. Grant Thornton.

Hamscher, Walter C. 1992. Modeling Accounting Systems to Support Multiple Tasks: A Progress Report. 10th National Conference on Artificial Intelligence. San Jose, Calif. (July): 519-524.

Kell, Walter G. & William C. Boynton. 1992. *Modern Auditing*. 5th ed. John Wiley & Sons, Inc.

Knechel, W.R. 1985. A Simulation Model for Evaluating Accounting System Reliability. *Auditing: A Journal of Practice and Theory* (Spring): 38-62.

Konrath, L.F. 1993. *Auditing Concepts and Applications: A Risk-Analysis Approach*. 2nd ed. West Publishing Company.

Meservy, R. D. 1985. Auditing Internal Controls: A Computational Model of the Review Process. PhD dissertation. University of Minnesota.

Meservy, Rayman David, Andrew D. Bailey, Jr. and Paul E. Johnson. 1986. Internal Control Evaluation: A Computational Model of the Review Process. *Auditing: A Journal of Theory and Practice* 6 (1): 44-74.

Pany, K, and O. R. Whittington. 1994. *Auditing*. Irwin.

Srinidhi, Bin N., and M.A. Vasarhelyi. 1989. Adaptation and Use of Reliability Concepts in Internal Control Evaluation. *Advances in Accounting* (Supplement 1): 141-158.

Stratton, W.O. 1981. Accounting Systems: The Reliability Approach to Internal Control Evaluation. *Decision Sciences* (January): 51-67.

Verghese, Thomas. 1988. A Formalization of Internal Control Evaluation. Ph.D. dissertation. Columbia University.

--. 1992. Automating Audit Reasoning About Testing Transaction Processing Systems. Working Paper. George Washington University.

Wallace, W. A. 1991. *Auditing*. 2nd ed. PWS-Kent Publishing Company.

Watne, D. A. and P.B.B.Turney. 1990. *Auditing EDP Systems*. 2nd ed. Prentice Hall.

Yu, S., and J. Neter. 1973. A Stochastic Model of the Internal Control System. *Journal of Accounting Research* (Autumn): 273-295.