A Language for Describing Office Information Systems

Julian Lebensold

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

December 1982

ABSTRACT

A LANGUAGE FOR DESCRIBING
OFFICE INFORMATION SYSTEMS

Julian Lebensold

We examine various definitions of the office as reported in the computer science literature, and propose to view an office as the information processing and generating component of an organization. As a result of this view, we propose eight requirements of a good modelling tool. ABL (Alternative Based Language) and three tools representative of those currently available for systems analysis are examined in the light of the stated requirements. We explore the use of ABL, developed at Concordia University by other researchers, as a modelling tool for office information systems. This language has powerful facilities for describing parallelisms, and is suitable for stepwise refinement.

To test the utility of ABL as a modelling tool, we use it to describe part of an existing, functioning

architectural office. Examples of both sequential and parallel office activity, as well as structured and unstructured tasks, are highlighted in the model.

Concurrent activities occur naturally in the office. Furthermore, complex office procedures are frequently broken into sub-tasks which can be independently executed. If powerful workstations form part of a high-bandwidth local area network, and if the granularity of the sub-tasks is large enough, then those sub-tasks can be executed in parallel by otherwise unoccupied workstations.

We present an algorithm for the assignment of these sub-tasks to available workstations. The algorithm minimizes inter-task communication cost, and is based on an ABL model of the complex activity. An inherent feature of such a model is its representation of complex procedures as interrelated sub-tasks. We also use ABL as a helpful tool to derive meaningful execution and communication costs for both a coarse-grain example, a sub-task in the architectural office, and a fine-grain case, Quicksort. We conclude that ABL is a powerful and useful tool for describing office information systems.

ACKNOWLEDGEMENTS

I am most indebted and deeply grateful to my supervisor, T. Radhakrishnan. Since the first time I walked into his office, he has been a source of guidance and inspiration. Throughout my graduate studies he has encouraged me to arrive at my own conclusions, and has stressed the importance of recognizing the validity of other points of view. He deepened my appreciation for conscientious research, scrupulous methodology, and accurate reporting. His quest for a harmonious and workable solution to problems, both academic and personal, and his desire for unity have served as valuable examples. Working with Krishnan has made my studies both an enjoyable and a rewarding experience.

It goes without saying that the pioneering work of Wojciech Jaworski in developing ABL plays a large part in my thesis. Over and beyond that, however, was the personal interest he took in my studies. He patiently explained unfamiliar concepts until I managed to grasp them, and was always willing to listen to and discuss any well-reasoned idea.

My fellow student and colleague, Cliff Grossner, was of great assistance in explaining some of the technical aspects of CUENET. His enthusiasm for CUENET and its applications has become contagious. My discussions with Libero Ficocelli greatly increased my comprehension of ABL, and led to the development of algorithm MERGE.

Financial support for my studies came from Concordia University, both through a University Fellowship and teaching duties, and the Government of the Province of Quebec, through a Bourse d'etudes from Les fonds F.C.A.C.

From my dear parents, Fred and Ruth, I received constant encouragement and support as I embarked on a second career. They expressed genuine interest in my work, even though it was outside the scope of their experience. It is largely as a result of my upbringing that I was able to focus my energies in a new direction.

Finally, this work is dedicated to my loving wife, Suzanne. Serving as a critical judge of half-baked ideas, she has continued to encourage and support me, even when we both thought my studies would never end. She has willingly borne with the frustrations of sometimes having everything else in our lives take a back seat to my studies. She has never let me lose sight of the fact, however, that computers must serve people, and not vice versa, a concept all too easily forgotten in this technological age.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

FIGURES

TABLES

LIST OF SYMBOLS

A_{ij} = **Alternative** i in **Step** j.

ah = The number of **Alternatives** in **Step** h.

ai = The number of **Alternatives** in **Step** i.

aj = The number of **Alternatives** in **Step** j.

ak = The number of **Alternatives** in **Step** k.

am = The number of **Alternatives** in **Step** m.

B_{ij} = The total probability of branching to **Step** j from **Step** i.

C_i = **Step** i.

C_0 = **Step** 0; i.e, the terminal **Step**.

c = The number of **Steps** in an ABL **System**.

D_{ij} = The average number of times the **Next_step** of **Step** i is **Step** j.

E_{ij} = The total cost of communicating between **Step** i and **Step** j.

F = The number of processors available for problem solution.

$FLAG_k$ = A subscripted variable which is added to an ABL **System** in order to implement algorithm MERGE.

G = The total communication cost within an ABL **System**.

g = An index for subscripted variables.

H = The total execution cost within an ABL **System**.

i = An index for subscripted variables.

j = An index for subscripted variables.

$K(C_{ij})$ = The total cost of communicating from **Step** i to **Step** j.

K(x) = Notation to denote the cost of expression x.

k = An index for subscripted variables.

L = A value which may be greater than, less than, or equal to 0.

lg x = The logarithm of x to base 2.

m = An index for subscripted variables.

N_{ij} = The **Next_step** of **Alternative** i in **Step** j.

n = An index for subscripted variables; also, the number of elements in a set of data.

P_i = **Precondition** i.

P_{ij} = **Precondition** i in **Step** j.

P(x) = The probability of the event denoted by expression x occurring.

p = The number of **Preconditions** in an ABL **System**.

Q = The ABL representation of a problem or program; i.e., an ABL **System**.

s = The ratio of the amount of time one processor will take to execute an algorithm to the amount of time several processors working in parallel will take; the speedup.

T_i = **Action** i.

T_{ijk} = **Action** i in **Alternative** j in **Step** k.

$T_{last,j,k}$ = The sequentially last **Action** to be executed in **Alternative** j in **Step** k.

t = The number of **Actions** in an ABL **System**.

U_{1i} = The i^{th} entry in the first row of matrix U; the average number of times C_i is executed.

V(x) = The truth value of the expression denoted by x.

x^q = The value of the expression denoted by x during the q^{th} iteration of algorithm ASSIGN.

x^{q+1} = The value of the expression denoted by x during the $(q+1)^{th}$ iteration of algorithm ASSIGN.

ϕ = The required fraction of the total cost of an ABL **System** to be represented by execution cost alone; the required processor utilization.

CHAPTER I

THE OFFICE OF THE FUTURE

The business community has developed a well-understood concept of the "office" as a place where certain kinds of activity take place. Computer scientists, however, see the office with its computer-based office products, local area networks, and portable intelligent terminals as an integrated system of communication. In an effort to bridge the gap between business people and computer people, the "office worker" of the traditional office has become the "knowledge worker" of the "office of the future" (UHLIG76). For the purposes of this study, the terms "office worker", "knowledge worker" and "one who works in an office" are synonymous.

In what follows, we take the point of view of the computer scientist. We first examine some of the prevailing concepts of what constitutes an office, and then propose our own definition. It is intended that this definition serve to disassociate the place of the office from the concept of the office. Our interest lies

primarily with the latter. In the remainder of this chapter we examine some of the research that has been done on "casual users" of computer-based systems, and relate that and the problems of user interfaces to office workers.

In chapter II we discuss modelling office information systems. It is generally accepted that both modelling and simulation play an important part in designing and restructuring systems in general, and office systems in particular (NUTT81). We present eight requirements for a modelling tool, including in our scope of modelling the concept of simulation. We also describe some existing modelling tools in order to help the reader understand the discussion that follows.

Chapter III introduces ABL (Alternative Based Language) and then examines ABL, SSA (Structured Systems Analysis) (MENDE80), BDL (Business Definition Language) (HAMME77), and ICNs (Information Control Nets) (ELLIS79) in the light of the requirements proposed in chapter II. This examination is done by examining how well each of the tools satisfies each of the requirements. Finally, we note some of the strengths and weaknesses of ABL.

As an application of the use of ABL as a modelling tool, we present in chapter IV the model of part of an architectural firm. We explain our choice of office to model, explain the model, and draw some conclusions from our experience.

In chapter V we discuss the questions of concurrency and problem decomposition as they relate to the office. We present an algorithm that draws on some of the features of ABL in order to provide an assignment of various sub-tasks to various parallel processors, based on minimizing inter-process communication. Each of these processors can be considered machines, individual knowledge workers, or departments in an organization without affecting the workings of the algorithm. We also discuss the question of costing office activity, and present two examples, one drawn from the architectural office, and the other from the computer science literature.

In chapter VI we present our conclusions, and indicate some possible areas for future research.

1.1 The present situation

The microelectronic revolution is well underway. Microprocessors are having a profound effect on the way in which many people work. Particularly affected is the traditional office worker. Various microprocessor-based office tools, from the simplest limited-memory electronic typewriter to the most sophisticated integrated workstation, are coming into widespread use. These tools are being linked together using local area networks to provide a working environment that is not limited by the physical constraints of an office building.

This new and impressive hardware is useless without adequate software to drive it. Furthermore, there is a growing concern about who is using computer-based systems and how the software interface to this hardware should be designed (MORAN81). The necessity for office managers to be able to assess both software and hardware aspects of the emerging technology and deciding which functions to implement raises the issue of being able to build an adequate picture of how the office functions now, and how it will function with the new technology in place.

1.2 The development and definition of OIS

Most people have an intuitive idea of what is meant by the term "office." When we deal with a detailed analysis of office activity, however, it is necessary that our terms be as precise and well-defined as possible.

One definition of the office is "a place where people read, think, write and communicate; where proposals are considered and plans are made; where money is collected and spent; where businesses and other organizations are managed" (GIULI82). Another definition considers the office as the physical place where people work, receiving, dealing with, classifying, remembering, finding and sending information (SCHEU81). It is important to remember that this transmission of information is done both orally and on paper. Uhlig et al. consider office workers to be "the

people who deal primarily in information" and point out that besides processing information, they also generate new information (UHLIG79). Of note is the Structured System Analysis definition of a business "as a logical set of functions which exists to provide a product or service..." (MENDE80). An office has also been defined as "a place where people gather to communicate with associates and customers, gain access to information and facilities, and perform their assigned tasks" (SEKEL82).

Uhlig et al. consider an "automated office" as "an office in which interactive computer tools are put in the hands of individual knowledge workers, at their desks, in the areas in which they are physically working" (UHLIG79). Ellis views "the typical large office of today ... as a complex, highly parallel, interactive information processing system." He goes on to say that "one definition of the office says that it is primarily an information processing and control system for an organization. Analogous to other systems, it has input, output and internal processing, but the individual activities tend to be simple, and the data structures tend to be complex" (ELLIS79).

Of the activities that take place in an office, we should note that, "a major process in any office is planning" (UHLIG79). Studies also indicate that only 20% of a secretary's time is spent typing (DRISC79). Other

major processes include the allocation of resources, monitoring the execution of plans, and making decisions (UHLIG79). Managers build relationships, persuade others and resolve conflicts (DRISC79). "Strategic level managers operate primarily by interfacing with people. They do this because each problem they deal with is unique, requiring unique information. Obtaining that information requires interacting with individuals" (UHLIG79). However, in the automated office, clerks are in a different category from managers and "are more used by computers than actually users of computers" (STEWA76).

Faced with these varying considerations of what is meant by "office," I propose a definition which disassociates the place of an office from the concept of an office: An office is that part of an organization into which comes information, from which comes information, and in which information is transformed in such a way that it can be used to produce products, services and money. Of particular importance is the consideration that the information that enters an office may come from many different sources, some of which might be considered non-traditional from the point of view of computer system input. There can be both structured and non-structured input data entering the office through such different channels as speech, gestures and mannerisms.

Given this interpretation, we can then characterize

offices by the kinds of information that enter them, the kinds that leave, and the sorts of transformations that take place within them. Also, we can look at the types of tools that are used both in the input/output operations and in the transformative operation.

Another consequence of the above definition is that it includes "preindustrial, industrial and information-age" offices within its scope (GIULI82). The preindustrial office relies heavily on individual performance. Workers perform their duties without the benefit of machines or a "systematic work organization." They deal with information according to their own predisposition, and are loosely gathered together into an organization. In the industrial office, workers are organized to serve the demands of a rigid system of production and its associated machinery. They handle information according to well-defined rules within a tightly-knit organization. The information-age office combines systems and machines so that both individual workers and their clients benefit. Individuals handle information as they see fit, while still working within a well-defined organization.

When "there is a computer terminal in every home, every office, and at every street corner ..." (HILL79) the physical location of the hardware and the software associated with office activity will not matter. Then a

view such as that taken here which places emphasis on the concept of the office rather than the place will become appropriate.

1.3 Office hardware

In the preindustrial office, office hardware consists largely of paper (which includes printed forms such as ledger sheets and letterhead as well as "scrap" paper used for recording ephemeral information), pencils, pens, filing cabinets, and postage stamps, all located within one well-defined physical area. The industrial office is modelled after an industrial assembly line, and includes such additional hardware as photocopiers, postage machines, electric and electronic typewriters and local telephone exchanges (PBX). The information-age office, however, takes advantage of current technology to minimize the amount of paper flowing through the office, and to distribute the logical components of the office in space.

In the information-age office, individual work stations may be composed of word processors, microcomputers, intelligent copiers, specialized graphic display units or high-speed printers. These individual stations can be installed in one physical location, clustered together and connected to each other, or they can be physically distributed and only logically interconnected. Local area networks (LANs), with their

speed, capacity, ease of installation and ready availability of data transmission paths, permit interconnection topologies that can be specific to particular office requirements (DIGIT82). Such LANs as Ethernet (SHOCH82) and CUENET (GROSS82) permit the logical interconnection of office components to be independent of their physical location. It is also possible to include special purpose hardware that will give the necessary degree of protection required in an office.

Already there are companies which are strongly committed to the information-age office. The low cost of microcomputers has allowed many companies to equip their office workers with simple, basic workstations. "Throughout the entire company worldwide, there are no secretaries in the traditional sense and only a few dozen typewriters. Virtually every one of our three thousand employees has an Apple computer on his or her desk" (KILLI82). Killins goes on to point out that the physical location of both the information resources and the employee are immaterial to the office workers.

1.4 Office software

In both preindustrial and industrial offices it makes no sense to talk of office software. However, organizational tools that ease the task of the office worker can be identified in both these kinds of offices.

It is by dealing with individuals on an individual basis that the office worker is able to transform the information that he receives from outside the office. Essentially, the program that determines how office employees function is resident both within them and their clients.

Workers in the industrial office are assigned very specific and limited tasks. As long as the information they are dealing with falls within the limits of their responsibility, they are able to deal with it. Anything outside their area of responsibility has to be referred to another level of management; that is, to another office worker. In this case, the "software" resides completely within the individual office worker.

In the information-age office, the typical office worker "wants to access others' data and programs, interchange documents, messages, etc. He would like an interconnected set of intelligent systems" personalized to his own needs (UHLIG79). He should have at his disposal a wide range of tools to assist him in his task of transforming information.

Bair considers the following typical tools (BAIR79): an interactive interface with on-line assistance; text and document preparation facilities, including typesetting; distributed teleconferencing; electronic mail delivery, cataloging, storage and retrieval; a hierarchical information structure with facilities to link with other

nodes; personal information management in the form of calendars, private notes, etc.; organizational information management; and facilities to enable custom-building subsystems. Word processing systems allow the office worker to transform information coming from many different sources into documents, forms and other written material rapidly. Data base management systems (DBMS) and data dictionaries permit him to store and retrieve information. Electronic mail systems (EMS) and teleconferencing networks assure him of ready and secured access to other office workers, even if they are located far from his own location. Finally, he has a host of software tools available to him to allow him to manipulate information so that he can make decisions appropriate to the task at hand. To point to one specific example, Cherry comments on some programs being used to help writers prepare documents: PARTS (which is used to analyze English text), STYLE (which produces readability indices), PROSE (which provides suggestions for improving the text), and REWRITE (which highlights "bad sentences") among others (CHERR82).

It is certainly true that "the real benefits of office information tools will become apparent when the tools form parts of the same environment" (TSICH80). Indeed, that may be the time when the "office of the future" would become the "office of the present."

1.5 The user interface and OIS

Considering only "knowledge workers," 37% of the United States work force works in an office (BARCO81). Including all those who manage information rather than produce goods, however, the figure rises to 60%. As a result, the problem of effective use of office tools becomes significant. In order to use a tool efficiently the user must be properly trained in its use, and the tool itself must be well-designed. Although Barcomb presents an excellent survey of the hardware tools available, he does not discuss the user interface. "The computer science profession must realize that the interface, not the technology, is the key to successful automated office information systems" (MARYA81). Thus, when looking at OIS, we are led to consider the user interface. After all, "it is people, not machines, who make an office what it is" (UHLIG79).

1.6 The casual user versus the office worker

There is a significant body of literature that concerns the "casual user." Researchers are starting to examine some of the characteristics of such a user, and are making suggestions as to how interfaces should be designed to suit his or her needs. We should note, however, that the worker in an office is not a "casual user." The

question of how closely these two kinds of user resemble each other will not be dealt with at length in this thesis. Nonetheless, some characteristics of the "casual user" that seem appropriate to the design of interfaces in the office environment will be highlighted.

In general, it seems clear that among the users of office hardware those users who have little ability or desire to write programs will dominate in the future. They look to computer systems as tools to be used, and are not interested in learning how to program them. They want to access each others' data and programs, interchange documents and messages, and so on. They would want interconnected intelligent systems to be personalized to their particular needs (UHLIG79).

There are various definitions of the casual user, some of which fit the characteristics of the office worker. Codd's characterization of the casual user as "one whose interactions ... are irregular in time and not motivated by his job or social role" exclude the majority of information-age office workers (CODD74). Their interactions are a direct result of their jobs. Kennedy defines the casual user as one who "...interacts with the system only rarely and who therefore requires a great deal of help in performing the action he intends" (KENNE74). This definition, too, seems to exclude office workers who would have a high degree of interaction with an automated

system. However, Kennedy's recommendations with regard to the design of user interfaces are certainly applicable to the office environment.

Cuff comes up with an appropriate definition for our purposes: Professionals in non-computing fields (CUFF80). He lists the following characteristics of the casual user: (1) Forgets major and minor details of a system; (2) Has a high error rate; (3) Expects to be forgiven for errors made; (4) Lacks typing ability; (5) Unable, or unwilling, to undergo a long training period; (6) Prefers on-line to off-line documentation; (7) Will not tolerate a strictly formal database query language; (8) Expects the database to be constructed according to his/her mental model; (9) Expects a non-legalistic interpretation of a database query; (10) Expects the context of the session to be maintained; (11) Expects the "system's end of any dialogue to be coherent"; and (12) Expects a polite system.

Office workers differ from Cuff's "casual users" in these repects: (1) They forget only those parts of the system that they do not use regularly; (2) Their error rate for unfamiliar parts of the system is high; (3) Secretaries and word-processing operators type well, but managers and clerks have varying degrees of typing ability; and (4) Managers, in particular, have no time for training, whereas clerical staff can be expected to undergo a training program. With respect to the other chracteristics, the

office worker has the same expectations as the "casual user."

With regard to the regular user, as opposed to the casual user, Eason points out that "the problem of the regular user is that he is likely to develop and change his needs and abilities and the computer has to be capable of satisfying many levels of ability" (EASON79). He also points out that they "progressively broaden their need for knowledge as their task requirements change."

Since "the user develops a conceptual view of the system from the total behaviour of the system" (MORAN81), and it is through the interface that the user observes how the system behaves, the design of the interface is extremely important. This problem is well summed up by Stewart: "... interface must be determined by the characteristics of the potential users" and their tasks. He concludes that the hardware and the software of the human must match the hardware and the software of the computer, or vice versa (STEWA76).

1.7 Interface design

As part of the consideration of proper interface design, we have to consider what modes of communication exist in the office, and which ones can (or should) be automated and which cannot. While some of the issues raised by this question are ethical and, touching on the

structure of society as a whole, will not be dealt with here, others are purely technological and will be briefly discussed.

In order to design effective man-computer interfaces, we need a "complete understanding of person-to-person communication" (CHAPA79). At the outset, we note that the simplest means of inter-human communication is through gestures, facial expressions and signs. Natural language, or speech, is a slightly more difficult means of communication, followed by writing or drawing, and typing. It is significant that the easiest way to communicate with a computer is by typing. It is more difficult for the computer to both understand and produce hand-writing and drawings. Intelligible speech can be produced by computers, but understanding free-flowing human conversation is still beyond the capabilities of computers. Finally, computers are totally incapable of understanding and generating gestures. That is, the order of ease of communication for computers is the exact opposite to that of humans (CHAPA79).

Communication in the office can be broadly divided into two categories: Formal and informal. Within each category there are three general methods used to communicate: Forms, speech and gestures (We include text and graphics as part of forms in this case). The communication of forms of all kinds can, and in some

instances is, being automated (TSICH80). Formal verbal communication can also be automated (HILL79) but it seems unlikely that communication using informal utterances or gestures will be automated in the near future.

Stewart points out the dual nature of the interface: a link and a barrier. The interface joins the physical and psychological aspects of the user to the hardware and software of the computer. It keeps the user from aspects of the system that are too complex for him to understand, or that are unnecessary for his current task. He suggests that the facilities should change as the user develops more expertise in his use of the system (STEWA76).

An extensive list of guidelines for the "humanizing" of human-computer interaction, is presented in (STERL74). The major concern is that each system be humanized so that "the system increases the kinship between men." Bair makes suggestions that relate to the work envoronment: Workers should have extensive on-line time; there should be a flexible "workstation" designed for all "knowledge workers," with a customized interface; and before installing a teleconferencing network, there should be a need for intercommunication within the user community, defined as a group having a common ground among its members (BAIR79).

With regard to a text-based interface, Kennedy makes the following suggestions: Communication be in a concise

natural language; the rate of exchange between system and user should lie in the user's "stress-free working range"; entries to the system should be in a free format; error messages should be polite, meaningful and informative; the system should allow the user to learn how to use it by himself; there should be on-line assistance; the command language should be logically consistent and simple; control "must appear to belong to the user;" and "the system should adapt to the ability of the user" (KENNE74).

Cuff, addressing the question of database interface design, suggests that the system actually help the user determine what it is that he wants to do (CUFF80). In order to facilitate the user's task, he suggests explicit and constrained choices (menus, prompting messages, and so on) with a "small set" of primitives. He also advocates a natural language interface, while recognizing some of the problems that can arise: Straying from the semantic bounds of the database, the limited word range of some systems, restricted grammars, spelling, syntax, loose queries, etc.

It is clear that office workers naturally use speech as part of their communication repertoire. Chapanis underlines the importance of speech to problem-solving activity (CHAPA79). The question then becomes one of determining how speech can be added to information storage, retrieval, and transformation. In this context, we must differentiate between voice input and voice output. Output

is cheap, of good quality, supports a large vocabulary, and can be produced in many languages. Input, however, has almost the exact opposite qualities. As a possible application of voice input, the system could ask for a user's password to be spoken, and thus could adjust itself to the user's voice (SCHEU81). Hill notes that by "restricting speech input devices to the identification and use of words spoken in isolation, it is possible to build cost-effective devices for the voice control of machines. Such devices may reasonably be called voice buttons..." (HILL79). If we are content to have a system which will only recognize one or two different voices, then voice input "may prove effective" even with microprocessor-based systems (EASON79). Such a system would only be used where it can prove to be cost-effective.

We can thus expect a typical work station to have voice input as well as a mouse, joy stick, touch screen, and keyboard. With the declining cost of personal computers, such hardware is becoming readily available in the home, and it is only a matter of time before it spreads to the office.

CHAPTER II

REQUIREMENTS FOR AN OIS MODELLING TOOL

2.1 The case for modelling

Until as recently as 1981, the approach being suggested to office professionals desiring to automate their offices was what could best be called the "trial and error" method (BARCO81, pp. 23-27). In fact it is claimed that the progression from prototype design through pilot installation to final implementation is a "strategy gaining widespread support" (BARCO81). One reasonable explanation for adhering to such a strategy is the lack of an adequate tool to model the various activities in an office.

Uhlig et al. recognized that if we view an office as a communication system, we will be able to quantify the processes that transpire in the office (UHLIG79). Then, given the appropriate tools, we will be able to model and formally describe office activities.

Ellis stressed the necessity of modelling office systems in order to help in the planning of new offices and the reorganization of existing ones. He proposed using

mathematical models of offices to gain insights into how the office functions, and possibly even to develop general theories about offices (ELLIS79). On the contrary, Hammer and Kuhn point out that "custom software must be produced for each office information system" (HAMME80). In order to develop such software, it is necessary to analyze what goes on in the office, and assess what is needed. Nutt and Ricci maintain that office managers need appropriate tools so that they can understand how their offices are structured. Such tools should allow them to "represent the structure of the office, to specify the information requirements of the office, to analyze the information flow, and to predict the implications of office reorganizations" (NUTT81). As Tsichritzis aptly points out, "there is a need to portray the flow of documents, the coordination requirements, and the structure of the office information systems." According to him, "there is a need for requirements specification tools, design tools, and modeling and analysis tools" (TSICH82).

Although it seems that "successful automated systems will require much modelling and analysis to be assured of correct operation" (TSICH80), we do not feel that "an all encompassing office procedure specification language will be rather complicated in order to be powerful" (TSICH80). In fact we will show that a relatively simple language, namely ABL, can be used for the specification of an office

environment.

Before examining exactly what we should look for in a modelling tool for OIS, it will be instructive to take a look at some of the characteristics of OIS that distinguish them from other systems.

2.2 Characteristics of OIS

Gorges et al. note that the typical office worker spends 10% of his time thinking, 20% reading and analysing, and 70% in some kind of verbal exchange. This verbal communication takes place over the telephone 20% of the time, in meetings 24% of the time, and in face-to-face communication 26% of the time (GORGE81). It is clear that the primary office activity is communication, and the majority of that communication is verbal. The communication may be highly structured, as in the case of form-handling, or it may be as unstructured as a remark between two office workers in an elevator. Tsichritzis, in describing OFS (Office Form System) recognizes that "there is much other activity not associated with forms which we cannot mechanize let alone automate" (TSICH80). The question we will try to answer is whether this activity can be modelled.

Another characteristic of OIS is what would be called exception handling in a computer system. One of the major talents that an office worker has to develop is the

ability to respond to unforseen events, since such events form an integral part of office activity. Barber explains this ability: "Office workers are able to handle unexpected contingencies in their daily work because they know the goals of the office work and because they know what actions are needed to achieve the goals of the office work" (BARBE82).

Parallelism is another aspect of an OIS. It is common for more than one office worker to be working on a given problem at the same time. This may result from having a team of people trying to resolve the problem, or it may come about because of management's desire to have different departments or subsidiaries compete against each other to be the first to arrive at a solution. Also, files are often shared among many workers. It is not uncommon to have different departments handle different copies of the same original document, making changes to their own copies which then become transferred to a master copy.

Data storage and retrieval in the non-automated office may not be as structured as the system modeller would like. From a small (sample size of 10) study of the way in which office workers organize their desks and offices, it was found that they like to store printed or written information in piles on the tops of desks and tables, in files in drawers and filing cabinets, and arranged either neatly or haphazardly on shelves (MALON82).

Given these characteristics and peculiarities of the office worker, it becomes a real challenge to effectively model an OIS. However, with appropriate tools, our task could become easier.

2.3 Proposed requirements for an OIS modelling tool

This leads us to consider what, exactly, are the desirable characteristics of a tool to model office activity. Recognizing that "a powerful office procedure specification language is an important aspect of an OIS" (TSICH82) we have formulated the following requirements of a modelling tool for an office information system:

1) Be simple to learn and use. Ideally, the modelling tool should be able to be used not only by the OIS professional, but also by the average knowledge worker and manager. In general, the kind of modelling tool we are looking for will be an automated one. That is, it will be a computer-based tool. The user interface to the software needs to be "friendly" and should follow the guidelines presented above, in section 1.7. The modelling tool should allow the models that are developed to be understood by decision-makers within the organization. Neither clerical workers, who may be called upon to build a model, nor managers, who will have to interpret the model, can devote a large amount of time to learning a complex tool.

2) Represent reality closely. One of the major

problems with many modelling tools is that once a model has been established, it is not a straightforward task to realize the system being modelled. One cannot simply scale up a model into a project and hope that it will work as specified. Similar problems can occur when we try to translate a model into reality. A model that could be directly implemented would be the ideal. However, given that it is unlikely that we will be able to automate every activity in the office, the modelling tool should at the very least have the capability of easily correlating the model with reality. The description of those parts of the office that are amenable to automation should be directly executable from the model. In this way, if the actual system proves to have flaws, then we would be able to relate them to the model and correct the errors in the system by going back to the model. The process of system design then becomes interactive and dynamic. This approach represents an improvement over the usual method which runs as follows: 1) develop the model; 2) translate the model into a set of programs that can be executed; 3) run the programs; 4) if the system does not perform as specified then, either the model is at fault or the translation is not faithful to the model, or both. At this point we need to be an expert in both the translation process and model building. By eliminating, or greatly diminishing the translation step, we are able to concentrate on the flaws

in the model itself.

3) Be consistent at various levels of the office hierarchy. Business organizations are traditionally structured hierarchically. Because of that structure, "it is appropriate that the specification language be multi-tiered, with the topmost level expressing the implementation-independent structure of the office and only the more detailed level serving to identify the particular way in which the general structure is being instantiated" (HAMME80). However, we should remember that at various levels of the organizational tree, there can be functions and procedures that are not hierarchical in nature. Some organizations have a more lateral structure; for example, many research establishments are organized laterally within a general hierarchy. Without resorting to different tools, the model should be able to reflect both hierarchical and heterarchical structures.

4) Express concurrent events. As Uhlig et al. point out, "processes occur in parallel, and clustered together, represent the various organizational functions" (UHLIG79). The model should be able to represent office activity adequately, whether it is sequential, parallel, or a combination of both. There can be various different activities going on in the office, but the synchronization necessary occurs using a "handshake" protocol. A common example is one worker asking for some information from

another worker, and either proceeding with other work while waiting for a response, or not being able to proceed until a response has been obtained. It is only in pre-industrial and industrial age offices that we find synchronous activity: The starting and stopping times of work. Thus the modelling tool must be able to represent the kind of asynchronous, sequential and parallel activity found in the office. Furthermore, the model should reflect the many kinds of interrupts that occur within an office. In a typical office environment there are many activities going on at the same time, each interrupting the other, and each one having a different priority. Some of these activities represent "informal" communication, and as such are difficult to model; however, others are very structured, and can be described.

5) Have a sound theoretical base. In order to facilitate the analysis of whatever systems are being described, the modelling tool should have a sound basis in theory. It would be preferable if this basis could be quantified mathematically. That way, the verification of certain properties of the system being modelled would be much easier. If a new language is being proposed as the basis for such a modelling tool, then it can be viewed as a formal language (HAMME80).

6) Represent both "document model" and "processor model." An office can be looked at in two quite different

ways. One is to consider the kinds of activities that occur, such as filing, mailing and so on. The other is to examine the kinds of data that travel through the organization, such as reports, documents and forms. The model should be capable of representing an office in either fashion, or, indeed, in both at the same time. In an actual business office, neither the documents nor the processes that deal with the documents are separated from each other. The model should be able to deal with either of the two approaches.

7) Handle incomplete specifications. One of the major uses of an OIS modelling tool is to plan new information systems and modify existing ones. It is not unusual for the specifications to be incomplete. Often this incompleteness becomes apparent after part, or in the worst case, all of the implementation has been done. A good modelling tool should help to bring out inadequacies in the specifications or in the implementation. In addition to this, Barber points out that it is necessary to add new information about the actions, choices, and situations present in the office environment "in an incremental fashion" (BARBE82).

8) Allow incremental change of models. It should be easy to make changes to an existing model as soon as any inadequacies are discovered. Rather than reconstructing the model from the beginning, it would be better to modify

it, one step at a time, and verify its performance at that stage before proceeding.

2.4 Some existing modelling tools

The existing tools for modelling office systems fall into four overlapping categories: Those based on Petri nets (PETER77), those based on form flow, those based on management techniques, and those that are a combination of techniques. Our objective in presenting some tools in the following pages is merely to describe them. We will leave a discussion of the merits of some representative tools to chapter III.

Petri nets form the basis for ICNs (Information Control Nets) which are graphical representations of the "set of related procedures" that makes up an office (ELLIS79) (figure 2.1). An ICN is a flow model that has a rigorous mathematical description. This formal description is presented in detail in (ELLIS79). The graphical representation is built up using circles to denote activities, squares to show data storage facilities, solid arrows indicating which activity comes before which other activity and dashed arrows to indicate the storing and retrieval of information. Arrows coming from nowhere are points of initiation, and arrows going nowhere are points of termination. Parallelism is indicated through the use of "AND nodes" which are solid circles from which come

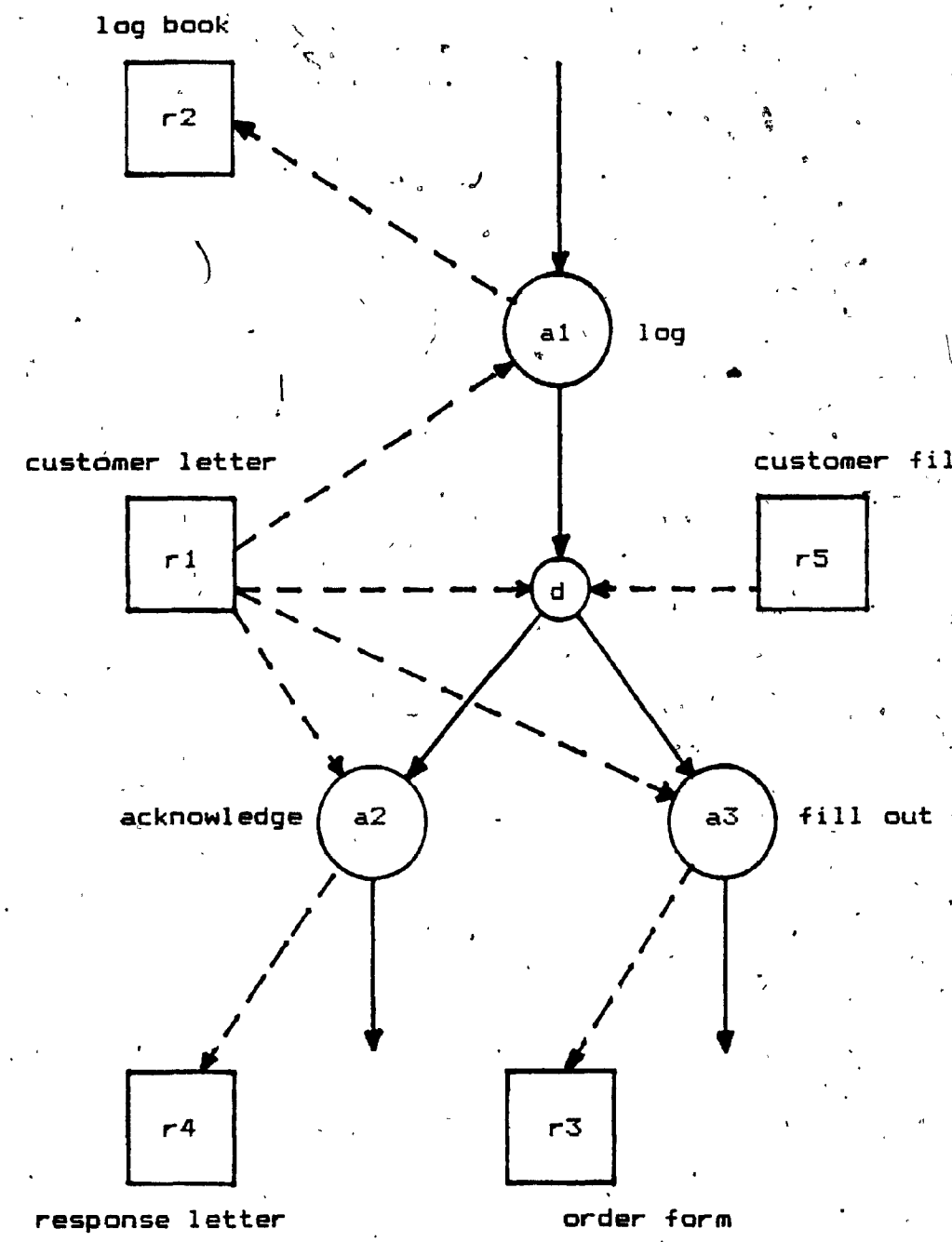


Figure 2.1

Information Control Net
(From ELLIS79)

arrows pointing to the various parallel activities. Finally, there are "decision nodes" which are circles into which come dashed arrows, and from which come solid arrows pointing to the activities among which a choice is to be made. The ICN is modular. Any circle can be further represented by another ICN. Similarly, a complete ICN can be represented by a single circle or node in another ICN.

Nutt and Ricci present Quinault, an automated tool for the analysis and construction of office models using ICNs (NUTT81). Ellis and Bernal describe ICNs used as part of the modelling subsystem of OfficeTalk-D, developed at the Xerox Palo Alto Research Center (ELLIS82). This subsystem permits the simulation of office activity, which can be distributed around an office using the Ethernet, "each machine acting on behalf of a particular set of people." An important aspect of the modelling subsystem is that one can "replace a simulation at a machine by a human interacting with the system via OfficeTalk-D" (ELLIS82).

The justification for modelling office activity using form flow seems to be that forms represent what is important to the organization, and that, since forms are so widely used in offices, the way in which they travel through the office presents a good picture of what is going on in the office. Form flow appears in several different tools, such as SSA (Structured Systems Analysis) and BDL (Business Definition Language), to be described below.

In OFS (Office Form System), a prototype OIS designed at the University of Toronto, modelling is done exclusively using forms, even though Tsichritzis points out that forms may not neccessarily be the best way of dealing with OIS (TSICH82). Treating voice as a form, and then adding that kind of form to OFS, Lee is able to use the form template to give a context to the output so that it becomes meaningful (LEE82). For example, the number 1982 in a "date" field will be spoken as "nineteen eighty-two," but in an "amount" field it will be uttered as "one thousand nine hundred and eighty-two dollars," and in a "part number" field as "one nine eight two."

Zloof has developed OBE (Office-by-Example) as an extension to the data base management system QBE (Query-by-Example) (ZLOOF81, ZLOOF82). Office workers use OBE to create and define objects (letters, graphs, charts, tables, etc.) on a two-dimensional display in a similar fashion to how they are created on paper. Although Zloof does not discuss this possibility, one might extend this strategy to include the specification of entire offices.

One OIS modelling tool is SSA (Structured Systems Analysis). It was developed as part of an effort by Exxon Corporation primarily to help improve software design and implementation. Mendes describes SSA as a "business modelling and communication technique" that is used by both the systems analyst and the business person. SSA concerns

itself with both the actual model that is built and the process of building the model (MENDE80) (figure 2.2). The SSA model consists of a "global model" which is a hierarchy diagram (figure 2.3), a "function matrix" which is an NxN array (figure 2.4), an "information flow diagram" which is a network diagram (figure 2.5), a "detail activity model" which is an annotated hierarchy diagram, a "data structure diagram" which is another annotated hierarchy diagram, and a "glossary of business terms" which is an English language narrative description.

The global model describes how the various functions in the organization are logically related to each other; the function matrix defines the responsibilities associated with each function; the information flow diagram represents how information travels through the organization; the detail activity model describes the "lowest-level functions in the Global Model"; the data structure diagram describes the view that the business person has of the information in the organization; and the glossary defines whatever terminology is relevant to the business being modelled. There are various "grammatical" rules which constrain the model builder, and assist him in producing an accurate model (MENDE80).

A language which appears to address the particular problems of the office is BDL (Business Definition Language). It was designed to be a problem-oriented,

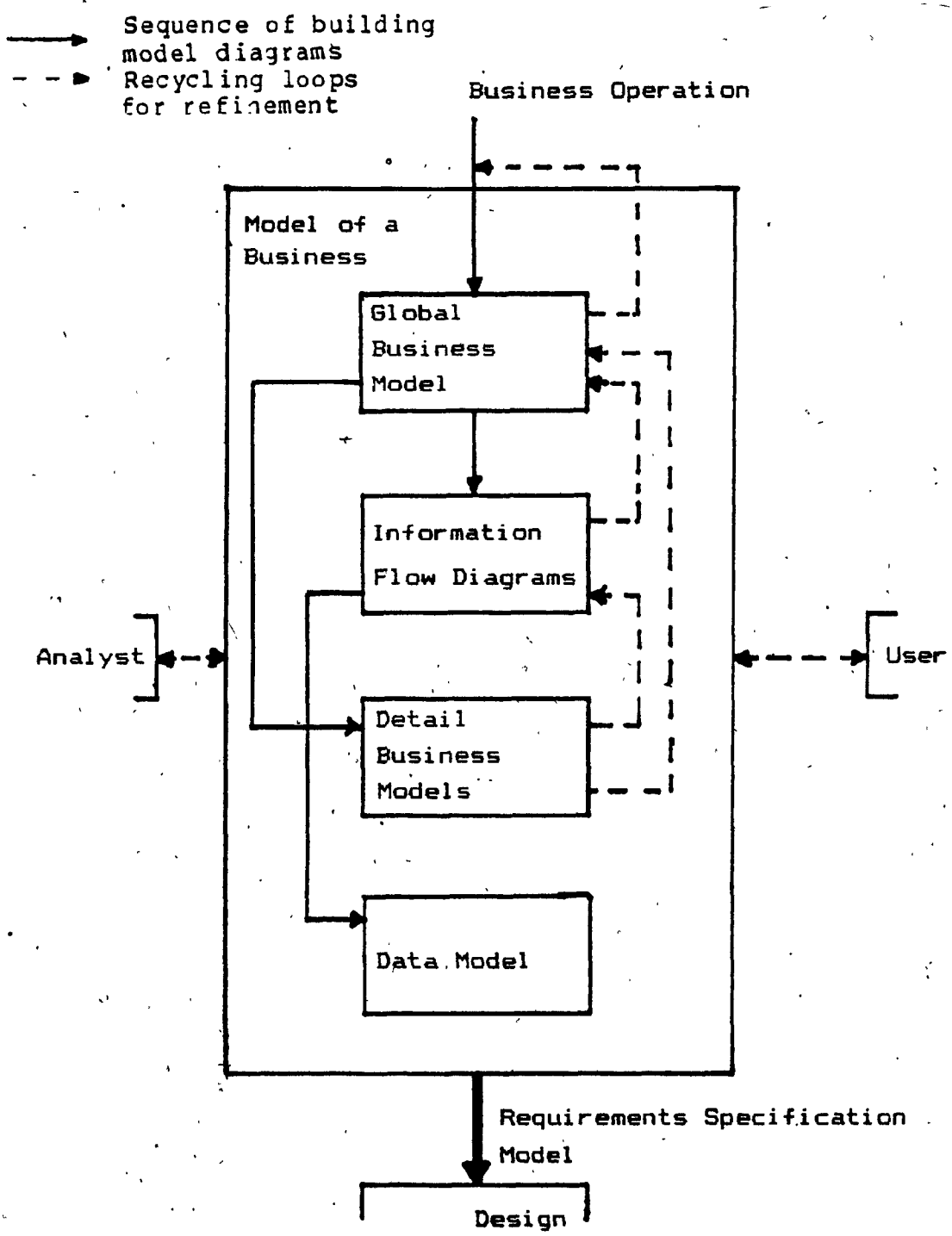


Figure 2.2

The Process of SSA
(from MENDE80)

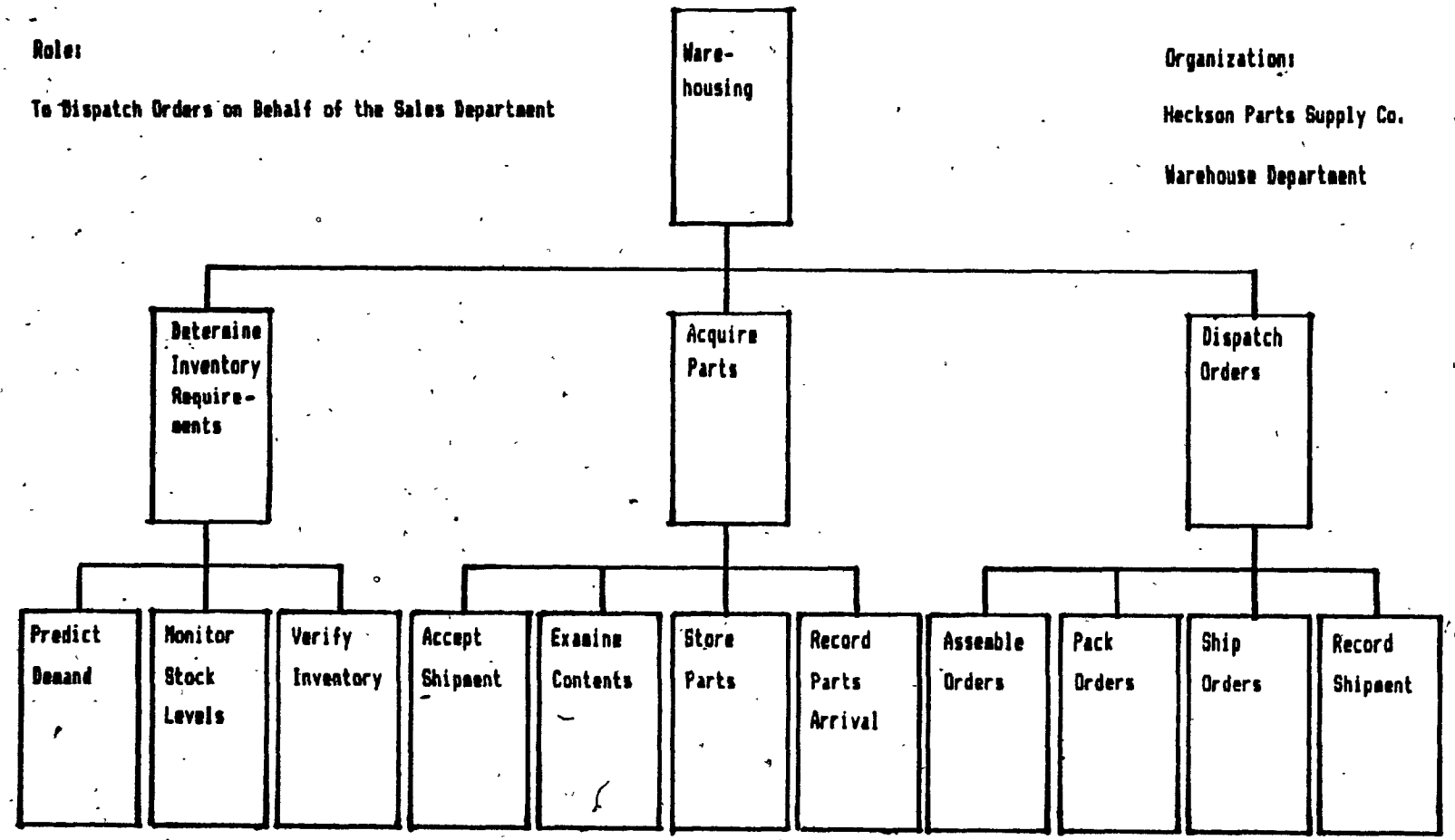


Figure 2.3

SSA Global Model
(from MENDE80)

Responsibility / Function	Warehouse Manager	Receiving Clerk	Forklift Operator	Shipping Clerk	Picker	Stock Clerk
Warehousing	X					
Determine Inventory Requirements	X					
Predict Demand						X
Monitor Stock Levels						X
Verify Inventory						X
Acquire Parts		X				
Accept Shipment			X			
Examine Contents		X				
Store Parts			X			
Record Parts Arrival						X
Dispatch Orders				X		
Assemble Orders					X	
Pack Orders				X		
Ship Orders			X			
Record Shipment						X

Figure 2.4

SSA Function Matrix
(from MENDE80)

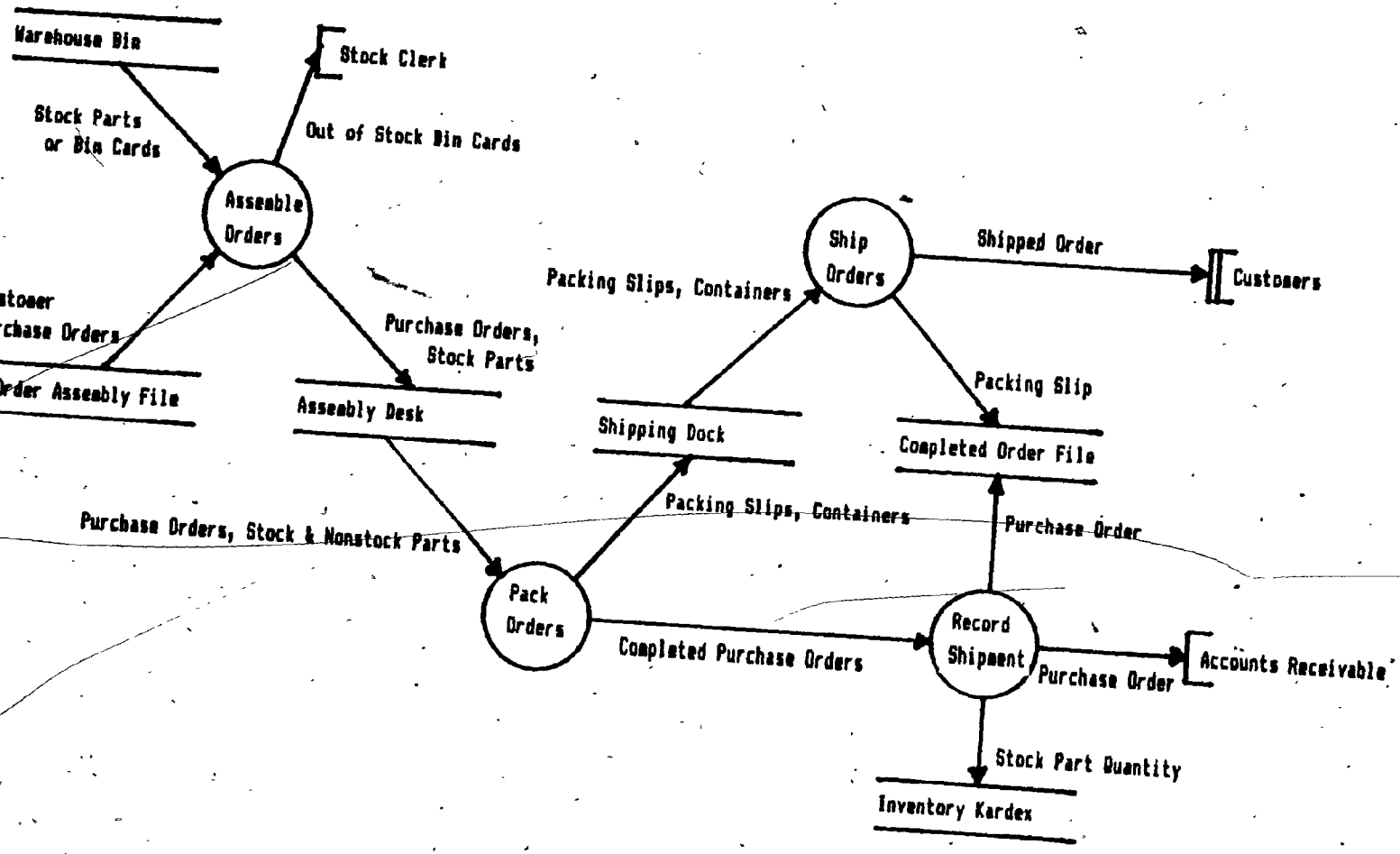


Figure 2.5

SSA Information Flow Diagram
(from MENDE80)

readable, easily modified "data processing" descriptive language (HAMME77). It was to be a means of communicating not only with computers, but also with people. BDL follows the structure of offices very closely, mirroring the flow of data through an organization.

There are four objects that BDL recognizes: documents, steps, paths and files. A document is the basic data structure, "simply data filled out on a form." Steps correspond to the "organizational units of the system being described," and they show, at the lowest level, how input documents are transformed into output documents. Paths represent the data flow. Documents flow over paths to get from one step to another. Files are "permanent repositories of documents." Since a document only exists between steps, it is necessary to have such a facility for storing information permanently.

Three of the major components that make up BDL are presented in (HAMME77). They are the Form Definition Component (FDC), the Document Flow Component (DFC) (figure 2.6), and the Document Transformation Component (DTC). Rather than there being one language which is used throughout BDL, there is a language specific to each of the separate components. Also, BDL programming requires an interactive terminal with some graphic facilities.

The FDC in general describes the forms that are used in the system. Hammer et al. define forms as "templates

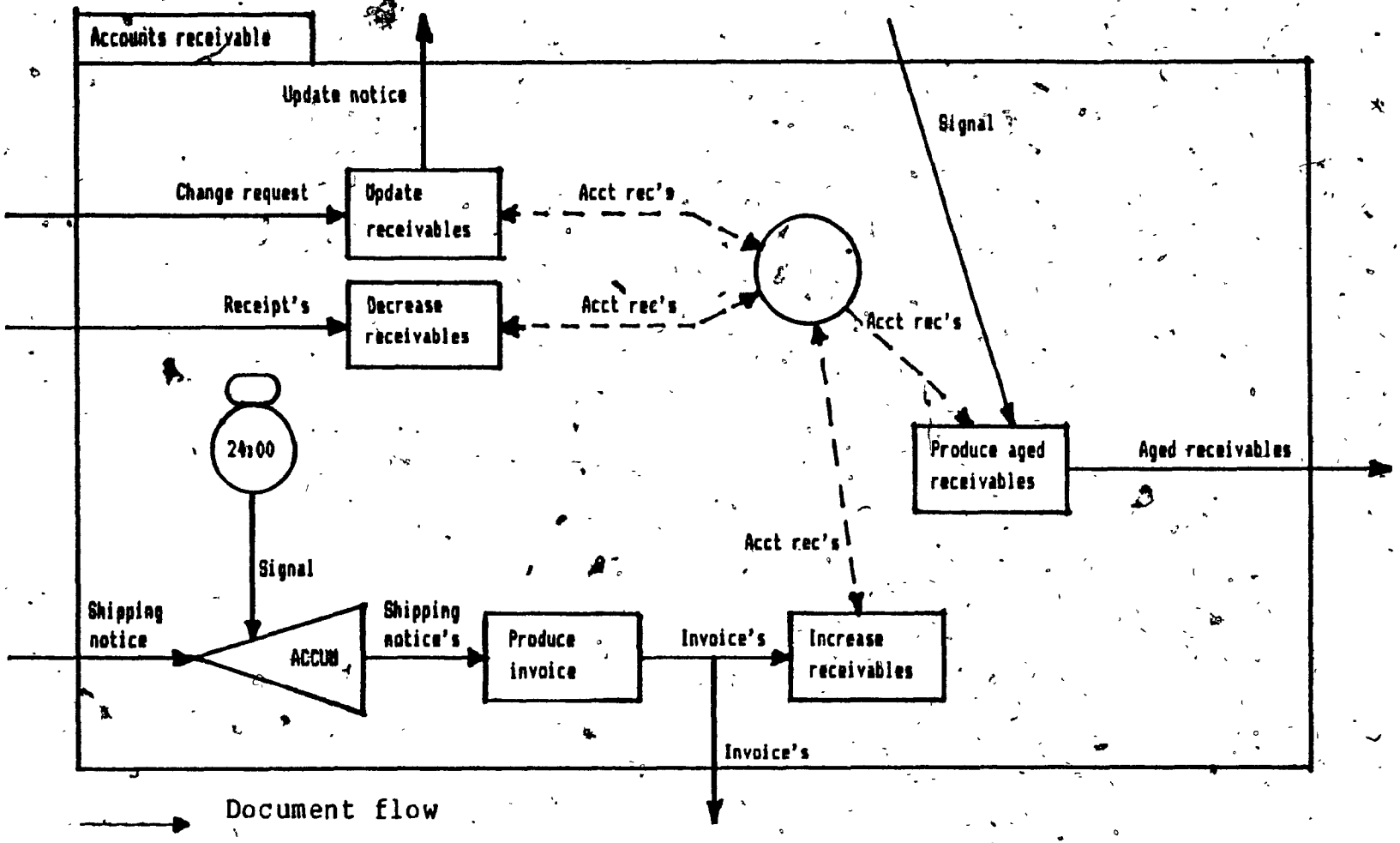


Figure 2.6

The Document Flow Component of BDL
(from HAMME77)

with which specific documents are produced." Many documents can use the same form; but they will all have the same structure and restrictions in both their use and the kinds of quantities they can represent. The language for forms definition is essentially one of interactive graphic use; that is, the user points to certain components on a screen, and then using those components he builds up and delimits the form.

The DFC uses rectangles, arrows and circles to describe the hierarchical structure of various activities within a system as well as their interrelationships. Rectangles are used to represent steps, solid arrows indicate which way documents flow (with the name of the document indicated beside the arrow, an "'s" indicating a group of documents), dashed arrows indicate file accessibility, and triangles indicate places in the system where documents are accumulated before being passed on for further processing.

The DTC shows how the various documents of a system function within that system. The language used here is a very high level language oriented towards dealing with groups of documents. One noteworthy aspect of the DTC language is that a "general approach to writing data processing programs" (HAMME77) is embodied directly in the language. Thus it is a highly structured language with built in control structures.

A more recent development is OSL (Office Specification Language), which has a "complex and intricate inherent structure, which may require more effort to learn but which should greatly enhance its usability" (HAMME80). "Associated with OSL will be a methodology for conducting office analyses and writing specifications" (HAMME80). "While OSL recognizes the importance of forms and people as individual units of office activity, it does not structure a procedure description around them; its orientation is towards the objects in an office. Objects in this context are the entities that are the focus of office activities and that form the basis for a description of office functions; the office as a whole is described in terms of the evolving history of its objects" (HAMME80). These entities are the things that the office workers deal with, and that the office is concerned with.

The interest in modelling tools is increasing. Bailey et al. describe TICOM-II, a modelling tool designed specifically for auditing purposes (BAILE82). As what could be part of a larger modelling tool, Arthurs and Stuck present a technique for finding the bounds of the mean rate of throughput and the mean delay in an abstraction of an office (ARTHU82). Taking an approach from the field of artificial intelligence, Barber uses a "knowledge embedding language called Omega" to "embed knowledge of the organization into an office worker's workstation in order

to support the office worker in his or her problem solving" (BARBE82).

CHAPTER III

AN EXAMINATION OF FOUR MODELLING TOOLS

3.1 ABL as a modelling tool

ABL (Alternative Based Language) was developed out of a concern for the need to have reliable and provable programs (FANC076, HINTE81). There are structural similarities between ABL, decision tables (MCDAN70) and guarded commands (DIJKS75), although they differ in many respects. In addition to providing the software designer with a powerful programming tool, ABL allows the system model builder to describe clearly and concisely how his system behaves. The complete syntax of ABL is presented in appendix A. In the following pages, we introduce ABL briefly, representing syntactic units in bold type, whereas in appendix A they are enclosed in angular brackets. The semantics of ABL constructs will become clear when we present a case study in chapter IV.

Briefly, an ABL description of a **System** consists of **Abstract_machines** and **Abstract_programs** (figure 3.1). An **Abstract_machine** is composed of **Actions**, **Preconditions**

```
                       PROGRAM                        MACHINE

         1  2  3  4  5  6  7  8  9 10 11 12       ALTERNATIVE NUMBER

C  1     V  .  .  .  .  .  .  .  .  .  .  .  V    C  1 Initialization.
C  2     .  V  .  .  .  .  .  .  .  .  .  .  V    C  2 Interaction with the client.
C  3     .  .  V  V  V  .  .  .  .  .  .  .  V    C  3 The project team.
C  4     .  .  .  .  .  V  V  V  .  .  .  .  V    C  4 Project start-up.
C  5     .  .  .  .  .  .  .  .  V  V  V  V  V    C  5 Wait.

                                                  CONTROL <DATA_OBJECT>S

                                                  <PRECONDITION>S
P  2     -  -  -  -  -  -  -  -  -  Y  -  N  -    P  2 Authorities contacted?
P  3     -  -  N  Y  -  -  -  -  -  -  -  -  -    P  3 Team complete?
P  4     -  -  -  -  -  -  -  -  N  Y  -  -  -    P  4 Design complete?
P  5     -  -  Y  Y  N  Y  Y  N  Y  Y  N  Y  -    P  5 Cost/performance ratio appears correct?

                                                  INPUT <DATA_OBJECT>S

                                                  <ACTION>S
A  1     4  .  .  .  .  .  .  .  .  .  .  .  .    A  1 Obtain a letter of intent
A  2     2  1  .  .  .  .  .  .  .  .  .  .  .    A  2 Type a document
A  3     .  3  .  .  .  .  .  .  .  .  .  .  .    A  3 Help client define his needs
A  4     .  .  .  .  .  .  1  .  .  .  .  .  .    A  4 Develop conceptual design
A  5     3  2  .  .  .  .  .  .  .  .  .  .  .    A  5 Interact with client
A  6     .  .  .  .  .  1  .  .  .  .  .  .  .    A  6 Contact local authorities
A  7     .  .  .  1  .  .  .  .  .  .  .  .  .    A  7 Select job captain
A  8     .  .  1  .  .  .  .  .  .  .  .  .  .    A  8 Select member of the team
A  9     1  .  .  .  .  .  .  .  .  .  .  .  .    A  9 Select partner-in-charge
A 10     .  4  .  .  .  .  .  .  .  .  .  .  .    A 10 Select project architect
A 11     .  .  .  2  .  .  .  .  .  .  .  .  .    A 11 Assign team to project
A 12     .  .  .  .  .  .  .  .  1  .  .  1  .    A 12 Question team
A 13     .  .  .  .  1  .  .  1  .  .  1  .  .    A 13 Question accountant ( Interrupt AC )

                                                  OUTPUT <DATA_OBJECT>S

                                                  EXCEPTION <DATA_OBJECT>S

                                                  <POSTCONDITION>S
P  1     -  Y  -  -  -  -  -  -  -  -  -  -  -    P  1 Needs defined?
P  2     -  -  -  -  -  Y  -  -  -  Y  -  Y  -    P  2 Authorities contacted?
P  3     -  -  -  Y  -  -  -  -  -  -  -  -  -    P  3 Team complete?
P  4     -  -  -  -  -  -  Y  -  Y  Y  -  -  -    P  4 Design complete?

NEXT     2  3  3  4  3  5  5  4  5  0  5  5  0

XCPT     1  2  0  3  0  0  0  0  0  0  0  0  0
```

Figure 3.1

An ABL System

and/or **Postconditions** and **Data_objects**. **Actions** are executed sequentially. Any particular set of **Preconditions** with its associated **Actions** is called an **Alternative**. **Alternatives** are grouped into clusters called **Steps**. An **Abstract_program** determines which **Alternatives** within a given **Step** are to be executed, based on the evaluation of the **Preconditions** in that **Step**. If more than one **Alternative** is selected for execution, then all those **Alternatives** are executed in parallel. After an **Alternative** is executed, the **Postconditions**, if present, are evaluated, and control is transferred either to the **Next_step** or to an **Exception_step**. The selection of **Alternatives** is then made again. This process continues until the **Abstract_program** stops which is indicated by a transfer to **Step** 0. It is important to note that of the **Actions** that compose an **Abstract_machine** some can be expressed in object code suitable for machine execution, and others can only be processed by humans. Further examples of the use of ABL to model simple systems may be found in (LEBEN81) and (LEBEN82). An excellent introduction to the use of ABL for a large-scale project is (HORVA82). An engineering application is found in (MORGA81). Linares developes ABMPL (Alternative Based MicroProgramming Language) from ABL (LINAR82) as a software tool to aid the microprogrammer.

3.2 Three other representative tools

In chapter II we mentioned several tools for describing office procedures. We selected three tools to evaluate in the light of the requirements presented in section 2.3. They represent the major categories referred to in section 2.4. ICNs are based on Petri nets, SSA has its roots in management techniques, and BDL is a combination of various techniques. Also, both SSA and BDL have components which model the flow of forms through the office. An exhaustive comparison of these tools with each other and with ABL is beyond the scope of this thesis. One of the reasons for not comparing these tools was their lack of availability to us. In the following section we briefly evaluate how well the three tools and ABL satisfy our proposed requirements.

3.3 ICNs, SSA, BDL and ABL in the light of the requirements

We will now examine the four modelling tools in the light of the requirements proposed in chapter II.

1) Simplicity of learning and use: An ICN can be understood by anyone with a minimum of graph theoretical knowledge. Whether they can be used as easily by non-mathematicians to model and describe existing office structures depends entirely on the user interface, since it becomes tedious to draw and modify complex graphs. Even

the simplest office procedure results in a complex network, and, as Ellis points out, complex office structures lead to extremely complex graphs. In order to reduce the complexity of an ICN, it is possible to consider connected subcomponents of it as another ICN, and then replace that ICN by a single node in the original ICN. This modular treatment permits the modelling of a complete organization which, by definition, has a high degree of complexity.

Creating a model of the business using SSA "is a cooperative effort between analyst and user" (MENDE80). The office worker who desires to use SSA either has to work with an SSA analyst, or has to be such an analyst himself. A wide variety of techniques are used in the process of developing the model. Each level of the design process uses a different technique complete with its own notations and symbols. Notwithstanding the difficulty associated with learning SSA, because the techniques are an integrated hierarchy, they foster a top-down approach to system design.

BDL too requires a "BDL application specialist" (HAMME77). Furthermore, the language requires both special hardware (a graphics terminal), and knowledge of several dissimilar languages. Without extensive training the typical office manager will have difficulty building a model of even the simplest office.

ABL has been used by undergraduate students taking a

first computer course for both problem analysis and solution. Students who have no preconceived notions about computer programming have an easier time understanding and using ABL than those who have been exposed to conventional programming languages like FORTRAN, COBOL or Pascal (JAWOR81). A formal study on the relative ease of use of ABL for problem solving is lacking, however, and would be helpful in comparing ABL to other programming languages. As an example of its ease of use for large-scale projects, we cite the use of ABL to model and implement a system for monitoring truancy in a high school system (HORVA82). Work is in progress to develop a "friendly" user interface to ABL for model building (EDDY82). At the outset we find that ABL is easy to use.

2) Machine interpretation of the model: An ICN is not intended to lead to the realization of the model. A one-to-one correspondence between the model and its equivalent representation in a programming language does not exist. Since an ICN is an uninterpreted model, it lacks the semantic content so important to its application to the real world. The lack of interpretation does facilitate the formal analysis of office environments, but for these environments to be meaningful to office personnel the models must be interpreted. Ellis points out that there are "pragmatic problems" with trying to specify an office completely, because of "exceptional conditions" and

"informal interactions" (ELLIS79).

SSA is not intended for direct implementation. It is useful for diagnosing problems, identifying opportunities for computerization and developing a "Requirements Specification Model." The process of model verification involves discussing the SSA model with all its potential users (MENDE80).

In the case of BDL, the Document Flow Component gives an overall view of what actually happens in an organization. The Document Transformation Component performs whatever calculations are necessary. However no provision seems to have been made for the two components to work together, nor for their implementation. BDL describes the document, or form, flow within an organization very well, however it does not represent the processing activities adequately.

By successively refining an ABL **System**, it is possible to maintain the same structure in a model as in the implementation of that model. **Systems** are successively refined to the atomic level of **Host_language_code** or **Narrative.** The **Host_language_code** can be executed by a machine, and the **Narrative** can be interpreted by a human being.

3) Consistency throughout the office: An ICN is modular in nature, and thus can be used at all levels of office activity. The ability to represent complex

activities by simple nodes facilitates the hierarchical construction of the model. Also, the fact that parallelism can be adequately shown enables one to represent lateral organizational structure. Data objects, however cannot be treated in the same way: "This model ... does not have facilities for arbitrary levels of data abstraction" (ELLIS79).

SSA reflects the hierarchical nature of the organization. It tends to look at business activities in a natural way; that is, as a hierarchy of information flowing at different levels. However, the model is the same no matter what kind of business is being described, and there seems to be no provision for laterally structured organizations. Furthermore, since no one of the tools that make up SSA is sufficient for a complete analysis, it becomes important to know which tool to use at which level of the organization.

BDL is strongly tied to the way existing businesses function and thus it could adequately represent the kinds of organizations found in the real world. The Document Transformation Component not only encourages, but actually forces top-down design. However, the design process in OIS is not exclusively top-down or bottom-up.

ABL permits the user to partition a System into several parts. Both the relationships that exist between the parts, and the parts themselves, can be displayed in

various ways, depending on the choice of user interface. Thus it becomes possible to look at an office environment both from a "low-level" point of view and from a "high-level" one. The OIS modeller can focus on any aspect of the office environment.

4) Expressing concurrent events: One of the strongest features of ICNs is their ability to express parallel activities. The concept of time concurrency is an integral part of an ICN. Although it can be modelled, Ellis suggests that the kind of asynchronous parallelism that is found in an office may be difficult to represent in a conventional programming language (ELLIS79).

In an SSA description of an organization, it is only at the lower levels of analysis that one can model parallel activities. This is done using the Detail Activity Model. However, no such tools exist for the same kind of modelling on a global basis.

In BDL much of the document flow diagram notation is derived from Petri nets. As a result BDL can represent concurrent events. Even so, the model is particularly well suited to the batch processing of documents. BDL tries to impose a structure on both structured and unstructured information, a process that may not always be successful. Hammer et al. claim that a telephone call can be viewed as "a stylized document carrying certain information" and can be represented using BDL terminology. In my opinion, it is

unrealistic to expect to be able to capture all the information conveyed in a telephone conversation in this manner, with the exception of very "structured" conversations.

ABL offers a rich variety of constructs to handle concurrency, asynchronous processing, and interrupts. The construction "(a,b)" in the syntax definition of ABL shows the capabilities of ABL for dealing with concurrency. An **Alternative**, on the other hand, is a sequence of **Actions**. An interrupt is a particular kind of **Action**, and the state of the system at the time it occurs is a **System**. This **System** is a **Data_object** which can be stored for recall and activation after the interrupt has been serviced.

5) Sound theoretical base: Because ICNs are based on Petri nets, there is a considerable body of analytical theory that the model builder can draw on while creating and analyzing a representation of an office. It is because of the rigorous mathematical definitions underlying ICNs that it is possible to "prove certain properties of offices" (ELLIS79); however, because of the complexity, it is difficult to formally analyze large offices.

Each technique that SSA uses, with the exception of the English language narrative used in the Glossary of Business Terms, has a sound theoretical basis. However, because of the wide variety of techniques and models used throughout SSA, it becomes a major task to verify any given

model mathematically.

Some components of BDL have a theoretical basis lending themselves to formal analysis, but others do not. Document Flow Components are based on network theory, and Document Transformation Components find their roots in formal language theory. The Form Definition Component, however, uses a variety of methods to describe a form. Thus, it is unclear how one would go about verifying that the form flow description is correct, or analysing the form requirements of the system being modelled.

The underlying theory of ABL is not yet formalized, however it could grow out of decision table theory (GANAP73, SHWAY74, SCHUM76), and formal language theory.

6) Representations of documents and processors: Both "processors" and "documents" can be seen in an ICN model; however, the inability to perform any kind of data decomposition severly hampers the model in its ability to serve as a data flow analysis tool. The assumption that all data is global does not bear any relation to reality, since in any office environment there are always locally defined data. In particular, since Ellis considers one of the information repositories "people's heads" (ELLIS79), it is hard to see how such a repository could be global in scope.

Both the flow of documents through the organization, and the kind of processing activity that takes place in the

system can be represented in SSA, but only by using different tools. The Information Flow Diagram shows how information ("documents") moves through a business, and the Detail Activity Model represents the processing activity. Because the two tools are so different, and because the interactions are not clearly defined, it is difficult to see how the two flows interact. Furthermore, the overall model itself remains static no matter what kind of organization is being described.

BDL allows the modeller to see both the document flow and the processing flow at the same time in the Document Flow Component; however it does not separate the two flows.

ABL gives the OIS designer complete flexibility in terms of being able to represent either the activities in an office, or the data that flows through it. The flexibility is embodied in the syntax of the language (see appendix A): The **Flow** that is part of an ABL **System** may be either a **Data_flow** or an **Abstract_program**, or both. Although the means of displaying and manipulating the **Data_flow** have not been completely developed, figure 3.2 serves to highlight the **Data_flow** component of the **System** presented in figure 3.1. Thus, for example, we see that we need the partner-in-charge, job captain, team, project, and team member to evaluate any of the **Preconditions** in **Steps** C3, C4 and C5. The project architect, D5, is necessary in

```
                                PROGRAM                          MACHINE

C  1  V  .  .  .  .  .  .  .  .  .  .  .  V  C  1 Initialization.
C  2  .  V  .  .  .  .  .  .  .  .  .  .  V  C  2 Interaction with the client.
C  3  .  .  V  V  V  .  .  .  .  .  .  .  V  C  3 The project team.
C  4  .  .  .  .  .  V  V  V  .  .  .  .  V  C  4 Project start-up.
C  5  .  .  .  .  .  .  .  .  V  V  V  V  V  C  5 Wait.

                                                 CONTROL <DATA_OBJECT>S
D  1  .  .  X  X  X  X  X  X  X  X  X  X  .  D  1 Partner-in-charge
D  3  .  .  X  X  X  X  X  X  X  X  X  X  .  D  3 Job captain
D  4  .  .  X  X  X  X  X  X  X  X  X  X  .  D  4 Team
D  5  .  .  X  X  .  .  .  .  .  .  .  .  .  D  5 Project architect
D  6  .  .  .  .  .  .  X  .  X  X  .  .  .  D  6 Conceptual design
D  8  .  .  X  X  X  X  X  X  X  X  X  X  .  D  8 Project
D 11  .  .  X  X  X  X  X  X  X  X  X  X  .  D 11 Team member

                                                 <PRECONDITION>S

                                                 INPUT <DATA_OBJECT>S
D  1  X  X  .  .  X  .  .  X  .  .  X  .  .  D  1 Partner-in-charge
D  2  X  X  .  .  .  .  .  .  .  .  .  .  .  D  2 Client
D  4  .  .  .  X  .  X  X  .  X  .  .  X  .  D  4 Team
D  5  .  .  X  X  .  .  .  .  X  .  .  X  .  D  5 Project architect
D  8  X  X  .  X  .  .  .  .  .  .  .  .  .  D  8 Project
D  9  X  X  .  .  .  .  .  .  .  .  .  .  .  D  9 Typist

                                                 <ACTION>S

                                                 OUTPUT <DATA_OBJECT>S
D  1  X  .  .  .  .  .  .  .  .  .  .  .  .  D  1 Partner-in-charge
D  3  .  .  .  X  .  .  .  .  .  .  .  .  .  D  3 Job captain
D  5  .  X  .  .  .  .  .  .  .  .  .  .  .  D  5 Project architect
D  6  .  .  .  .  .  .  X  .  .  .  .  .  .  D  6 Conceptual design
D  7  X  X  .  .  X  .  .  X  .  .  X  .  .  D  7 Document
D 10  .  .  .  .  .  X  .  .  .  .  .  .  .  D 10 Report
D 11  .  .  X  .  .  .  .  .  .  .  .  .  .  D 11 Team member

                                                 EXCEPTION <DATA_OBJECT>S
D  1  .  X  .  X  .  .  X  .  X  X  .  .  .  D  1 Partner-in-charge
D  2  .  X  .  .  .  .  X  .  X  X  .  .  .  D  2 Client
D  3  .  .  .  X  .  X  X  .  X  X  .  X  .  D  3 Job captain
D  4  .  .  .  X  .  X  X  .  X  X  .  X  .  D  4 Team
D  5  .  X  .  X  .  X  X  .  X  X  .  X  .  D  5 Project architect
D  6  .  .  .  .  .  .  X  .  X  X  .  .  .  D  6 Conceptual design
D  7  .  X  .  .  .  .  X  .  X  X  .  .  .  D  7 Document
D  8  .  X  .  .  .  .  X  .  X  X  .  .  .  D  8 Project
D 10  .  .  .  .  .  .  X  .  .  .  X  .  X  D 10 Report

                                                 <POSTCONDITION>S

NEXT  2  3  3  4  3  5  5  4  5  0  5  5  0

XCPT  1  2  0  3  0  0  0  0  0  0  0  0  0
```

Figure 3.2

Data_flow for an ABL System

Step C3 to determine the appropriate **Alternative.** In **Step** C4, the team, D4, produces either a report, D10, or a conceptual design, D6.

7) Incomplete specifications: The modularity of an ICN permits modelling incomplete systems and dealing with incomplete specifications. However, because the specifications are uninterpreted, we must be careful not to draw any conclusions from them.

Because SSA involves the use of what are essentially templates (the different "Models") for the description of various stages of organizational activity, it can be a useful tool for identifying missing elements in the organization.

Since BDL allows the specification of various components of the system without any notion of how they are going to work together, it is well-suited to describing a system from incomplete specifications.

Because ABL permits complete freedom in dealing with the various levels of an organization, incomplete specifications are easily handled. The "ELSE" clause in decision tables is the underlying power of this facility to deal with incomplete specifications (MCDAN70). This capability is implemented as a transfer of control to an **Exception_step. Postconditions** that are pre-defined by the modeller are evaluated after the execution of each **Alternative**, and control is subsequently transferred either

to a normal **Step** or to an **Exception_step**, depending on the state of the **Postconditions**.

8) Incremental change: ICNs, especially when supported by an interactive utility such as is found in the Quinault system, can be changed easily at any time in the development cycle. Since SSA is used by a trained analyst working interactively with the workers in the business, omissions in the model become readily apparent and can be incorporated in the next version. BDL also relies on interactive development of the model, both between the modeller and the users and between the modeller and the tool. It too permits the easy modification of an existing model. An ABL **System** is easy to modify at specification, design or implementation stages. **Actions**, **Preconditions**, **Data_objects**, **Alternatives**, or **Steps** can be added or deleted at any time during the development of a model.

3.4 The current state of ABL

ABL has been under development for several years. It has evolved to its present state largely as a result of many student projects. At the time of this writing there is a limited display facility, and an interpreter for ABL **Systems** that use Pascal as their **Host_language_code**.

As part of their studies, both graduate and undergraduate students have developed ABL interpreters using FORTRAN, BASIC, Pascal and assembly language as the

Host_language_code. The display program which has been developed is written entirely in ABL. It produces displays similar to those of figures 3.1 and 3.2. An interactive ABL editor is nearing completion, and plans are being made to develop a comprehensive ABL programming environment which will run on a microcomputer-based system.

The question of the theoretical soundness of ABL is, however, still open. Because of the formal basis for ICNs, Ellis can "prove certain properties of offices" such as the equivalence of office structures or the existence of a minimal data flow organization (ELLIS79). A formal theory in such a sense is yet to be developed for ABL.

The present lack of a comprehensive, screen-based, pictorial display is also a handicap. In order to build a model, the various components of a **System** have to be coded in special formats, and then a display can be produced. Simulation of the model requires different formats. If the **System** is to be changed, then the original coding has to be changed. The model-builder should be able to interact directly with whatever he sees displayed.

Although ABL is being used as a pedagogical tool to teach programming concepts, it lacks the various tools necessary for its own instruction. At the present time ABL is taught by a limited set of people. Courses, instruction manuals or text-books, and a system on which to develop and execute ABL models are all necessary. Cases drawn from

both management literature and real organizations need to be presented, analyzed, modelled, and simulated using ABL to the fullest.

The case study presented in the next chapter serves to show how ABL can be used to model an existing office. In (LEBEN81) and (LEBEN82) some ABL models of hypothetical offices are presented, and in (HORVA82) a model and implementation of an information system is developed using ABL.

CHAPTER IV

AN ARCHITECTURAL OFFICE MODELLED USING ABL

4.1 Selecting an Office to Model

We decided to test the ability of ABL to model an office. As a result we decided to build such a model. We were faced with the following four candidates: A hypothetical office, an office in a now-defunct organization, an existing office in the university, and an existing office outside the university.

To model a hypothetical office, we would first have had to develop a description of it. While such an exercise has merit, and would test ABL's ability to specify new offices, we would have had great difficulty in assessing the ensuing specification. As a result we decided to model a real office. We also had to decide between modelling an office which no longer exists and one which is currently operating. Modelling the former could have led to problems of acquiring information. We would have had to rely heavily on former employees' memories, and there would have been no way to directly observe the office activity. Thus

we decided to model a functioning office. Finally, we had to select an office to which we had ready access and whose employees would be cooperative. Such an office was available both within the university (a departmental office), and in the business world (an architectural office). We decided to test ABL outside the academic world so that we would be in a better position to assess its utility in a normal business environment.

4.2 The Architectural Office

Since we are dealing with a real office, we will protect the identity of the firm by calling it TAO (The Architectural Office). TAO is owned by five architects in partnership, with its head office in a major Canadian city, and other design offices in several Canadian and U.S. cities. TAO designs and supervises the construction of office buildings, theaters, community centers, and residential buildings. It is also involved in urban planning, acting as both a consultant to and designer for various levels of government. In addition to the head office, there is a "site office" at each project under construction. The number of employees varies, depending on the number and size of the projects being handled by TAO at that time. Furthermore, when there are very few projects being worked on, it is common to close the various branch offices and conduct all business from the head office.

There are two major components to TAO: The architectural or design component, and the financial or business component. Organizationally, each project that enters the office is placed under the direction and supervision of one of the senior partners of TAO, referred to as the "partner in charge". Depending on its size and relative development, a project may also have a "job captain" and a design team assigned to it. Individuals ("architects" and "draftsmen") who design and supervise projects devote varying amounts of time to the various projects that TAO is involved in. It is quite common for the various offices to work independently of each other in developing designs and monitoring the progress of their individual projects. The financial aspects of TAO, however, are centralized in the head office. The accounting department consists of one accountant and two bookkeepers, and a variety of forms is used for recording financial information. Copies of some of these forms can be found in appendix B of the thesis.

Informal communication can occur amongst TAO staff members, including the partners, between staff members and clients, and between staff members and outside consultants. It is this communication which forms the basis for many design decisions. As a result, office activity can take place in a partner's office, in transit between physical offices (one in which office equipment such as tables and

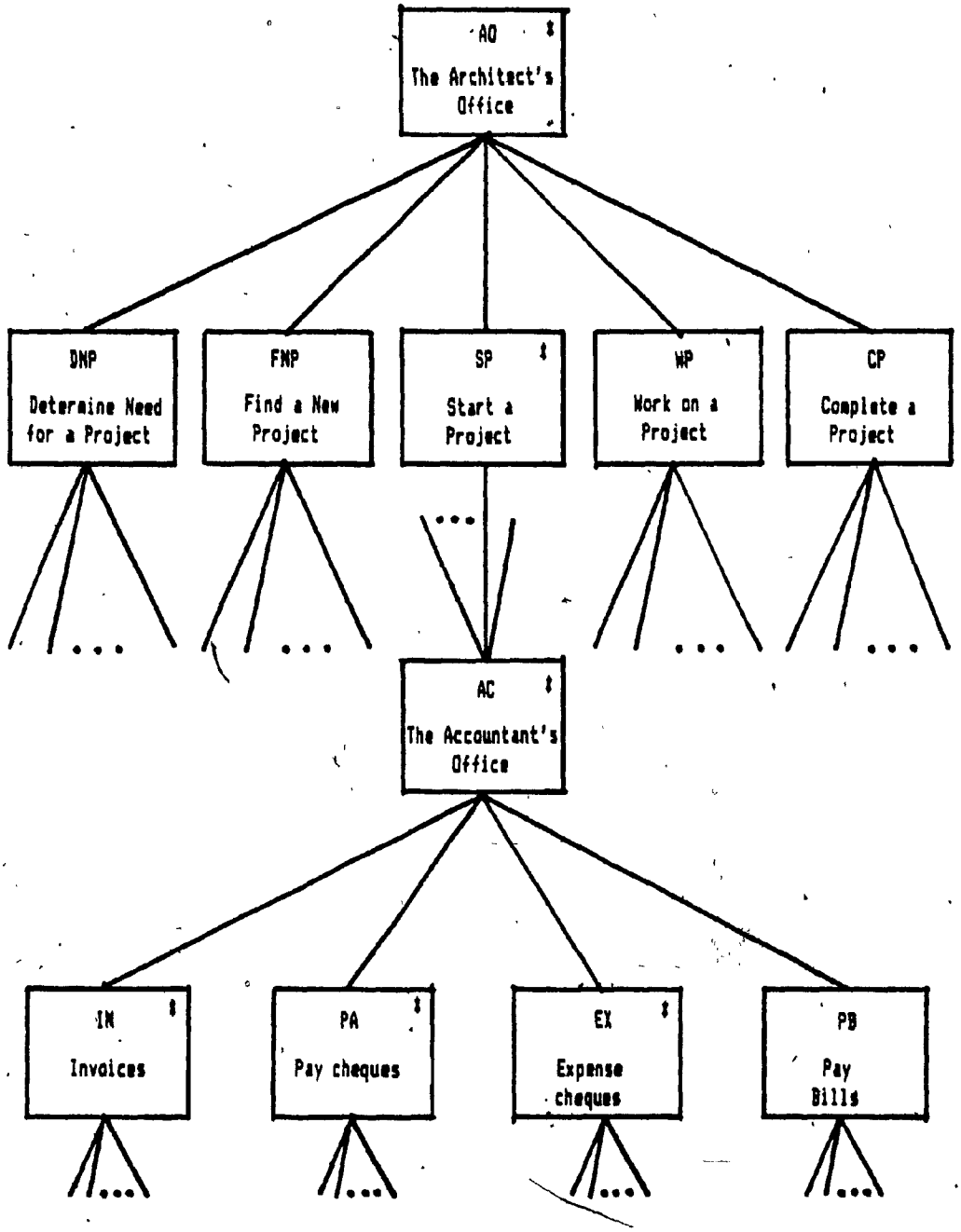
chairs is located), on a construction site, or in a physical TAO office.

4.3 The ABL Model

For a number of reasons, we decided to model only some of the aspects of TAO. TAO is an extremely complex organization, physically distributed across North America. We did not have the resources necessary to model the entire organization. Even at the local office level, the complexity of TAO is such that it would require a large amount of time to complete a detailed analysis of the activity in the office. The model that will be presented below, however, amply demonstrates some of the advantages and disadvantages of using ABL as an OIS modelling tool.

Figure 4.1 shows how the various ABL **System**s are related to each other. The **System**, AO (The Architect's Office) presents the most general view of the office. It calls DNP (Determine Need for a Project), FNP (Find New Project), SP (Start a Project), WP (Work on a Project) and CP (Complete a Project). SP issues interrupts to AC (The Accounting Office) which communicates back to SP before continuing. AC calls IN (Invoices), PA (Pay cheques), EX (Expense cheques), and PB (Pay bills), each of which calls other, lower level routines.

In figure 4.2, one form of the **System** AO is displayed (we shall refer to this form of display as the



* These <System>s are described in the text.

Figure 4.1

Structure of the model

STEP DESCRIPTION	NEXT STEP
1.0 Determine whether a new project is necessary.	
1.1 The need for a new project has been determined.	2
2.0 Find a new project.	
2.1 A new project has been found.	3
2.2 No project was needed.	1
3.0 Start up a new project.	
3.1 A new project has been started.	4
3.2 A new project has not been started.	2
4.0 Perform all the ongoing work associated with a project.	
4.1 A project is being worked on.	5
4.2 A project is not being worked on.	3
5.0 Finish a given project.	
5.1 A project is completed.	1
5.2 A project is not being worked on.	4

Figure 4.2

AO: Narrative Form

"narrative form"). There are five **Steps** that one of the senior partners of TAO identified. These are identified here as 1.0, 2.0, etc. The **Narrative** associated with each **Step** is taken directly from the description used by the partner. As part of each **Step** there is displayed the **Narrative** for each **Alternative** within that **Step.** Thus, the two **Alternatives** that form **Step** 3.0 are 3.1 and 3.2; i.e., "A new project has been started" and "A new project has not been started." Finally, the **Next_step** to be executed after the completion of each **Alternative** is indicated. For example, after "a new project has been started" (**Alternative** 3.1), the next **Step** to be executed is to "perform all the ongoing work associated with a project" (**Step** 4.0). If "a new project has not been started" (**Alternative** 3.2), then it is necessary to "find a new project" (**Step** 2.0). Since none of the **Alternatives** in this **System** have **Step** 0 as their **Next_step**, the **System** does not stop. This condition reflects the observation by the partner that "we are always trying to find new projects, we are always looking for work."

Figure 4.3 presents the same **System**, but in a more detailed form (we shall call this form the "matrix form"). The **Abstract_machine** is listed on the right, and the **Abstract_program** is represented by the matrix on the left. The **Abstract_machine** consists of four **Preconditions**, labelled P1, P2, etc., and five **Actions**, identified as A1,

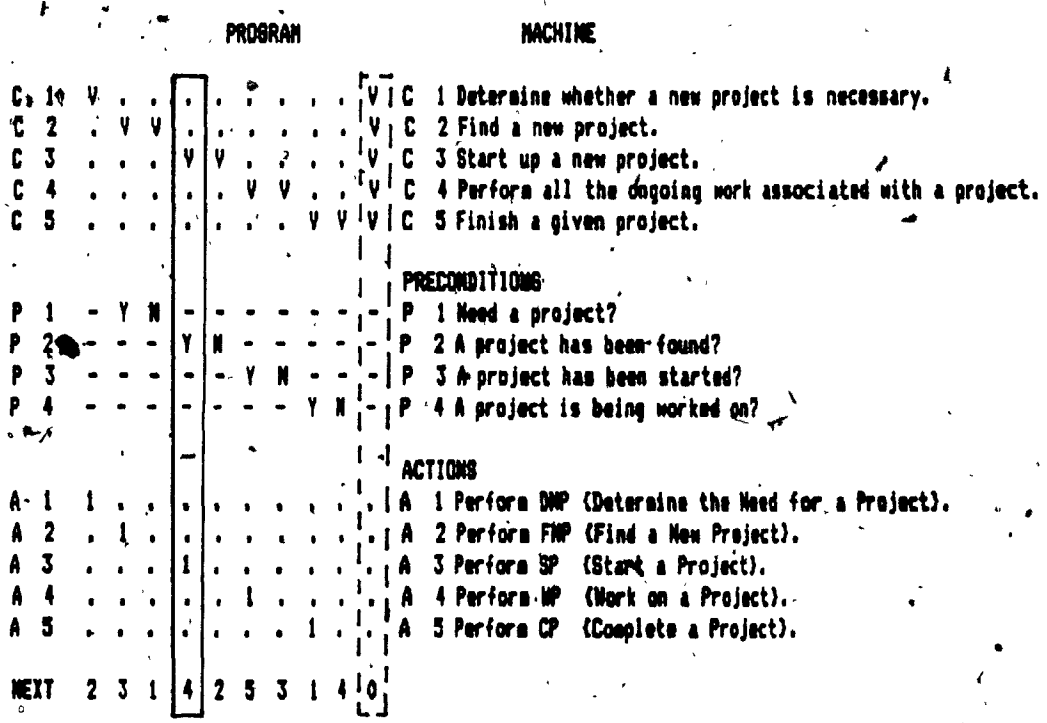


Figure 4.3

AO: Matrix Form

A2, etc. In this case, each of the **Actions** is a call to another ABL **System.**

In the matrix on the left, each row identified by a "C" corresponds to a **Step.** Thus, for example, **Step** 3.0, is identified as C3 in the matrix. Each column in the matrix corresponds to one **Alternative,** and membership of an **Alternative** in a given **Step** is indicated by a "V" at the matrix entry corresponding to the **Step** row and **Alternative** column. Thus, **Alternative** 3.1 is the column enclosed in a solid box in figure 4.3. This **Alternative** is selected for execution if **Precondition** P2 evaluates to TRUE; that is, if "a project has been found." If that is the case, then the **Actions** numbered below, are executed in the order in which they are numbered. In this case, only A3 is executed. Finally, when all the **Actions** have been executed, a transfer is made to the **Next_step** indicated by the identifier in the row labelled "NEXT," in this case **Step** 4.0.

We can summarize **Alternative** 3.1 as follows: If, when we are about to start a new project (**Step** 3.0), a project has been found (**Precondition** P2), then we should start a project (**Action** A3) and then "perform all the ongoing work associated with a project" (**Next_step** 4).

We should note that there is one **Alternative** that is considered to be part of all **Steps.** This **Alternative** is enclosed in a dashed box in figure 4.3. If, for any

reason, a situation arises such that none of the **Preconditions** can be satisfied within a given **Step**, then this **Alternative** is executed. As is clear from figure 4.3, no **Actions** are executed, and the **Abstract_program** comes to an orderly halt. This **Alternative** corresponds to the "ELSE" rule in limited-entry decision tables.

Moving to the next level of detail, we can examine what happens when a project is started. Figure 4.4 presents the narrative form of the **System** SP. The interpretation of this display is identical to that presented in figure 4.2. In this case, however, the **System** terminates at a given time; that is, the **Next_step** for **Alternative** 5.2 is 0.

The matrix form of SP is presented in figure 4.5. Again, the interpretation is similar to that of figure 4.3. Of interest here, however, are the two **Alternatives** enclosed in a solid box in the figure. Since the **Precondition_set** for both **Alternatives** is the same, and the requisite value of the **Precondition**, P5, is the same in both cases, the two **Alternatives** are to be executed in parallel. This reflects the parallel activities of developing a conceptual design and contacting the authorities in the place where the project is to be developed. Subsequent synchronization takes place in **Step** C5. An example of sequential execution occurs in the first **Alternative** (enclosed in a dashed box). **Action** A9 is

STEP DESCRIPTION	NEXT STEP
1.0 Initialization	
1.1 A letter of intent has been obtained.	2
2.0 Interaction with the client	
2.1 The client's needs are defined.	3
3.0 The project team	
3.1 A member of the staff has been assigned to a team.	3
3.2 A team is complete and assigned to a project	4
3.3 Financial state of the project is verified.	3
4.0 Project start-up	
4.1 Local authorities have been contacted.	5
4.2 The conceptual design is completed.	5
4.3 Financial state of the project is verified.	4
5.0 Wait	
5.1 Wait for conceptual design to be completed	5
5.2 Ready to continue	0
5.3 Financial state of the project is verified.	5
5.4 Wait for contact with local authorities to be complete	5

Figure 4.4

SP: Narrative Form

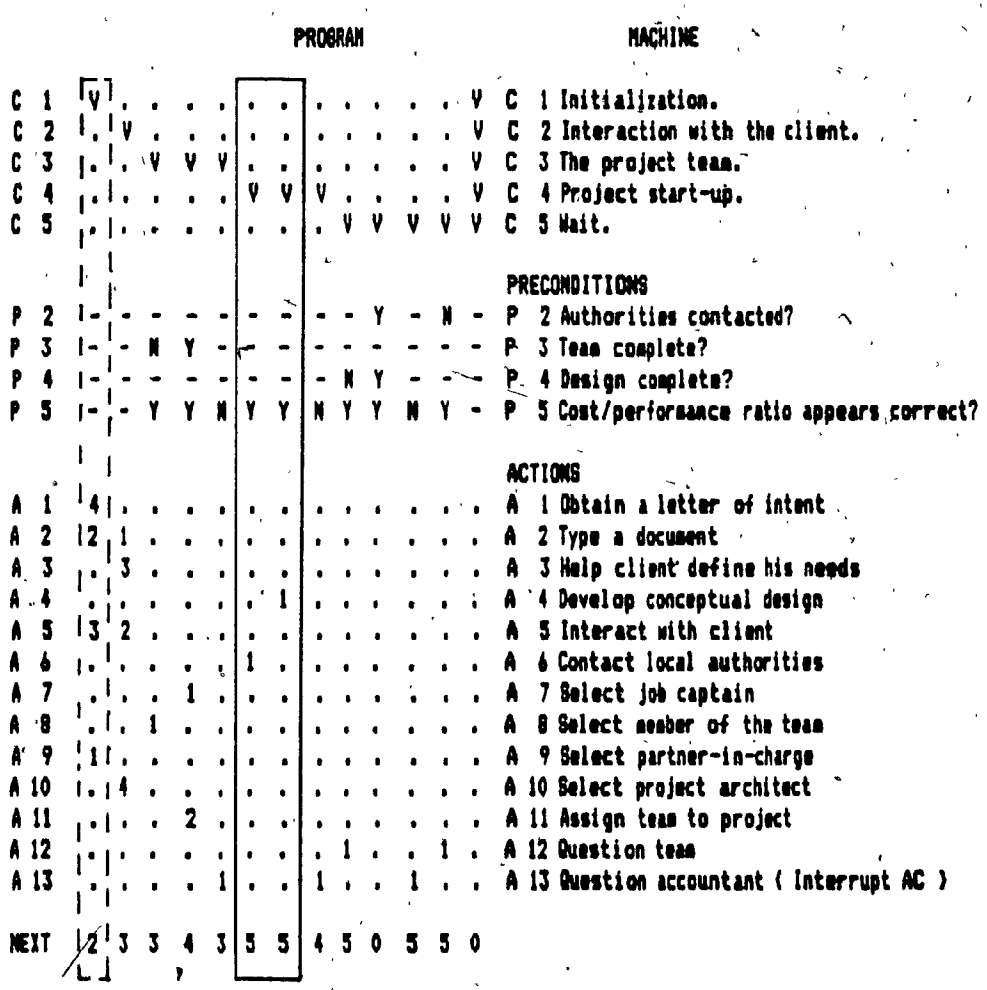


Figure 4.5

SP: Matrix Form

executed, followed by A2, A5 and A1. Note also, that at any time that it is not the case that the "cost/performance ratio appears correct" the accountant is questioned (**Action** A13) by interrupting the **System** AC.

Figure 4.6 is the narrative form of the **System** AC, the accounting office. In addition to the normal accounting activities of any business, the TAO accounting operation is responsible for keeping track of whether or not any one of the several projects being handled by the firm is getting out of financial control. That is, if the ratio of expenses so far to the expected total value of the project is consistent with the amount of time spent on the project.

The **Abstract_machine** for AC consists of both **Narratives** and what can easily become **Host_language_code** (figure 4.7). **Actions** A2 through A10 are presented in a form very similar to standard Pascal, however **Action** A11 is in ordinary English. Each of the various categories in **Actions** A3 through A8, such as "fees", refers to an item recorded on the job cost card (figure B.1). Although the **Abstract_program** specifies in detail how the "cost_to_date" is calculated, it does not detail how a report is to be presented (**Actions** A11 and A12).

The preparation of invoices is modelled by the ABL **System** IN. We should note here that this **System** corresponds to **Action** A13 in the **System** AC. Both forms of

STEP DESCRIPTION	NEXT STEP
1.0 Perform on-going accounting work, responding to partner interrupts.	
1.1 Invoices, expense cheques and bills have been dealt with.	1
1.2 Everything has been done.	0
1.3 Pay cheques have been dealt with.	1
1.4 A summary has been prepared.	2
1.5 Invoices, pay cheques, expense cheques and bills have been dealt with.	1
2.0 Report the status of a project.	
2.1 Project reported out of control.	1
2.2 Project reported under control.	1

Figure 4.6

AC: Narrative Form

		PROGRAM								MACHINE	
C	1	V	V	V	V	V	.	.	V	C	1 Perform on-going accounting work, responding to partner interrupts.
C	2	.	.	.	.	.	V	V	V	C	2 Report the status of a project.
											PRECONDITIONS
P	1	N	N	N	Y	N	-	-	-	P	1 Partner requests job status report?
P	2	Y	N	N	-	Y	-	-	-	P	2 End of the month?
P	3	N	N	Y	-	Y	-	-	-	P	3 Pay week?
P	4	-	-	-	-	-	N	Y	-	P	4 Cost_to_date / contract_value <= time_to_date / projected_time?
											ACTIONS
A	1	.	.	.	1	.	.	.	.	A	1 Get job cost card.
A	2	.	.	.	2	.	.	.	.	A	2 Cost_to_date := 0.
A	3	.	.	.	3	.	.	.	.	A	3 Cost_to_date := Cost_to_date + salaries.
A	4	.	.	.	4	.	.	.	.	A	4 Cost_to_date := Cost_to_date + fees.
A	5	.	.	.	5	.	.	.	.	A	5 Cost_to_date := Cost_to_date + consultant costs.
A	6	.	.	.	6	.	.	.	.	A	6 Cost_to_date := Cost_to_date + travel costs.
A	7	.	.	.	7	.	.	.	.	A	7 Cost_to_date := Cost_to_date + printing costs.
A	8	.	.	.	8	.	.	.	.	A	8 Cost_to_date := Cost_to_date + miscellaneous.
A	9	.	.	.	9	.	.	.	.	A	9 Overhead := Cost_to_date * 0.1.
A	10	.	.	.	10	.	.	.	.	A	10 Cost_to_date := Cost_to_date + Overhead.
A	11	.	.	.	.	.	.	1	.	A	11 Report project under control.
A	12	.	.	.	.	.	1	.	.	A	12 Report project out of control.
A	13	1	.	.	.	1	.	.	.	A	13 Perform IN (Prepare Invoices).
A	14	.	.	1	.	2	.	.	.	A	14 Perform PA (Issue Pay cheques).
A	15	2	.	.	.	3	.	.	.	A	15 Perform EX (Issue Expense cheques).
A	16	3	.	.	.	4	.	.	.	A	16 Perform PB (Pay bills).
NEXT		1	0	1	2	1	1	1	0		

Figure 4.7

AC: Matrix Form

IN are presented in figure 4.8. The client card referred to in **Action** A7 is shown in figure B.2 of appendix B.

Making out the pay cheques is more involved, and the process is modelled by the **System** PA, shown in figures 4.9 (narrative form) and 4.10 (matrix form). The employee time sheet is shown in figure B.3, and the partner's time sheet is presented in figure B.4.

The narrative form of EX, the **System** modelling the preparation of expense cheques, is presented in figure 4.11. When dealing with a **System** that has as many **Steps** as this one, it can be helpful to have a graphical representation of the **Abstract_program.** Such a display is presented in figure 4.12, where each numbered box corresponds to a **Step** in the **System** EX. Figure 4.13 shows the matrix form of the same **System**, and the auto expense report, general expense report, and travel expense report are shown in figures B.5, B.6, and B.7, respectively.

4.4 Experience in the use of ABL

After using ABL to try to model part of the activity in a real office, it is useful to assess the practicality of ABL as a modelling tool. Because ABL allows the system modeller to record **Actions**, **Preconditions** and/or **Postconditions** and **Data_objects** independent of any control structures, it was very easy to translate the seemingly random comments made by members of the TAO staff into

STEP DESCRIPTION	NEXT STEP
1.0 Get the next client.	
1.1 All invoices have been done.	0
1.2 Job cost card has been retrieved.	2
2.0 Determine state of the project.	
2.1 An invoice is required for this project.	3
2.2 No invoice required for this project.	1
3.0 Prepare the invoice.	
3.1 Invoice prepared using outstanding reports.	1
3.2 Invoice prepared using only the job cost card.	1

PROGRAM MACHINE

```
C  1  Y  Y  .  .  .  .  Y  C  1 Get the next client.
C  2  .  .  Y  Y  .  .  Y  C  2 Determine state of the project.
C  3  .  .  .  .  Y  Y  Y  C  3 Prepare the invoice.

                              PRECONDITIONS
P  1  N  Y  -  -  -  -  -  P  1 More invoices?
P  2  -  -  N  Y  -  -  -  P  2 Project dormant?
P  3  -  -  -  -  N  Y  -  P  3 Posting up to date?

                              ACTIONS
A  1  .  1  .  .  .  .  .  A  1 Get job cost card.
A  2  .  .  .  .  3  1  .  A  2 Add up costs.
A  3  .  .  .  .  4  2  .  A  3 Type invoice.
A  4  .  .  .  .  5  3  .  A  4 Mail invoice.
A  5  .  .  .  .  1  .  .  A  5 Get outstanding time sheets.
A  6  .  .  .  .  2  .  .  A  6 Get outstanding expense reports.
A  7  .  .  .  .  6  4  .  A  7 Post invoice to client card.

NEXT  0  2  3  1  1  1  0
```

Figure 4.8

IN: Narrative and Matrix Forms

STEP DESCRIPTION	NEXT STEP
1.0 Get the next employee.	
1.1 No more employees to process.	0
1.2 Another employee is being processed.	2
2.0 Issue pay cheque.	
2.1 Partner has been paid, fees have been posted.	1
2.2 Regular employee has not been paid.	1
2.3 Regular employee has been paid.	1
2.4 Partner has been paid, fees not posted.	1
2.5 Partner's secretary has been contacted for time sheet.	1

Figure 4.9

PA: Narrative Form

		PROGRAM								MACHINE	
C	1	V	V	.	.	.	.	.	V	C	1 Get the next employee.
C	2	.	.	V	V	V	V	V	V	C	2 Issue pay cheque.
											PRECONDITIONS
P	1	N	Y	-	-	-	-	-	-	P	1 More employees?
P	2	-	-	Y	N	N	Y	Y	-	P	2 Employee is a partner?
P	3	-	-	Y	N	Y	Y	N	-	P	3 Employee time sheet available?
P	4	-	-	Y	-	-	N	-	-	P	4 End of the month?
											ACTIONS
A	1	.	1	.	.	.	.	.	.	A	1 Get employee name.
A	2	.	.	9	.	.	1	.	.	A	2 Get partner's time sheet (week 1).
A	3	.	.	10	.	.	2	.	.	A	3 Get partner's time sheet (week 2).
A	4	.	.	1	.	.	.	.	.	A	4 Get partner's time sheet (week 3).
A	5	.	.	2	.	.	.	.	.	A	5 Get partner's time sheet (week 4).
A	6	.	.	11	.	7	3	.	.	A	6 Find total_hours / job.
A	7	.	.	12	.	.	.	.	.	A	7 Find total_fees / job.
A	8	.	.	.	.	8	.	.	.	A	8 Find total_salaries / job.
A	9	.	.	3	.	.	4	.	.	A	9 Find total_hours / week.
A	10	.	.	.	.	.	5	.	.	A	10 Find total_hours (week 1 + week 2).
A	11	.	.	4	.	.	.	.	.	A	11 Find total_hours (week 3 + week 4).
A	12	.	.	5	.	3	6	.	.	A	12 Calculate gross_pay.
A	13	.	.	6	.	4	7	.	.	A	13 Calculate deductions.
A	14	.	.	7	.	5	8	.	.	A	14 Net_pay := gross_pay - deductions.
A	15	.	.	8	.	6	9	.	.	A	15 Issue cheque for net_pay.
A	16	.	.	.	.	.	.	1	.	A	16 Contact partner's secretary.
A	17	.	.	13	.	9	.	.	.	A	17 Post total_hours / job to job cost card.
A	18	.	.	14	.	.	.	.	.	A	18 Post total_fees / job to job cost card.
A	19	.	.	.	.	10	.	.	.	A	19 Post total_salaries / job to job cost card.
A	20	.	.	.	.	1	.	.	.	A	20 Verify total hours.
A	21	.	.	.	.	2	.	.	.	A	21 Calculate total amount per entry.
NEXT		0	2	1	1	1	1	1	0		

Figure 4.10

PA: Matrix Form

STEP DESCRIPTION	NEXT STEP
1.0 Get the next employee.	
1.1 No more employees to process.	0
1.2 Another employee is being processed.	2
2.0 Process auto expense report.	
2.1 No auto expense report was submitted.	3
2.2 Auto expense report has been processed.	5
3.0 Process general expense report.	
3.1 No general expense report was submitted.	4
3.2 General expense report has been processed.	5
4.0 Process travel expense report.	
4.1 No travel expense report is expected.	1
4.2 Secretary has been contacted and claim is being held.	1
4.3 Receipts have been added up.	6
5.0 Process unaccounted-for amounts.	
5.1 All amounts have been accounted for.	1
5.2 Unaccounted-for amounts have been posted.	1
6.0 Verify receipts.	
6.1 Receipts and expenses do not balance.	1
6.2 Expenses balance receipts.	7
7.0 Process over- or under-payment.	
7.1 Balance due employer not paid.	5
7.2 No outstanding balances.	5
7.3 Balance paid to employee.	5
7.4 Balance paid to employer.	5

Figure 4.11

EX: Narrative Form

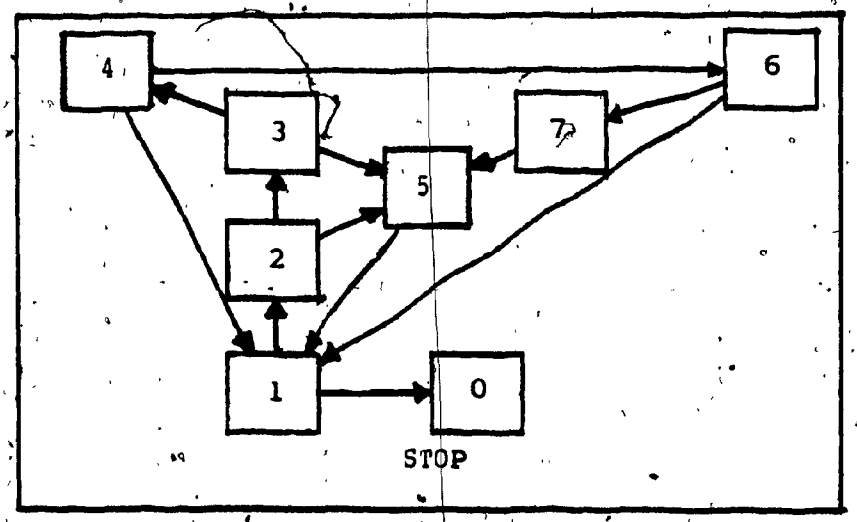


Figure 4.12

EX: Graph Form

	PROGRAM																					MACHINE		
C	1	V	V	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	V	C	1	Get the next employee.
C	2	.	.	V	V	.	.	.	.	.	.	.	.	.	.	.	.	.	.	V	C	2	Process auto expense report.
C	3	.	.	.	.	V	V	.	.	.	.	.	.	.	.	.	.	.	.	V	C	3	Process general expense report.
C	4	.	.	.	.	.	.	V	V	V	.	.	.	.	.	.	.	.	.	V	C	4	Process travel expense report.
C	5	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	.	.	.	V	C	5	Process unaccounted-for amounts.
C	6	.	.	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	.	V	C	6	Verify receipts.
C	7	.	.	.	.	.	.	.	.	.	.	.	.	.	V	V	V	V	V	V	C	7	Process over- or under-payment.
																							PRECONDITIONS	
P	1	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P	1	More employees?		
P	2	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P	2	Auto expense report submitted?		
P	3	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	P	3	General expense report submitted?		
P	4	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	P	4	Travel expense report submitted?		
P	5	-	-	-	-	-	-	N	Y	Y	-	-	-	-	-	-	-	-	-	P	5	Employee returned from trip?		
P	6	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	P	6	Amounts still unaccounted for?		
P	7	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	P	7	Totals on form = total of receipts?		
P	8	-	-	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	P	8	Balance due employee?		
P	9	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	N	-	Y	-	P	9	Balance due employer?		
P	10	-	-	-	-	-	-	-	-	-	-	-	-	-	N	-	-	Y	-	P	10	Payment enclosed?		
																							ACTIONS	
A	1	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	1	Get employee name.		
A	2	.	.	.	1	.	.	.	.	.	.	.	.	.	1	.	.	.	.	A	2	Amount := $ 0.22 * mileage.		
A	3	.	.	.	2	.	1	.	.	.	.	.	.	1	.	.	.	.	.	A	3	Enter total by job number on form.		
A	4	.	.	.	3	.	2	.	.	.	.	.	.	2	.	.	.	.	.	A	4	Post total amount / job to job cost cards.		
A	5	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	A	5	Post amounts to general expenses in general ledger.		
A	6	.	.	.	4	.	3	.	.	.	.	.	.	.	.	1	.	.	.	A	6	Issue cheque.		
A	7	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	A	7	Contact appropriate secretary.		
A	8	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	A	8	Add up receipts.		
A	9	.	.	.	.	.	.	.	2	.	.	.	1	.	1	.	.	.	.	A	9	Put claim on hold.		
A	10	.	.	.	.	.	.	.	.	.	.	.	2	.	2	.	.	.	.	A	10	Query employee.		
A	11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	A	11	Deposit payment.		
NEXT		0	2	3	5	4	5	1	1	6	1	1	1	7	5	5	5	5	0					

Figure 4.13

EX: Matrix Form

Abstract_machines. Once that was done, determining the **Abstract_program** for that **Abstract_machine** was an easy task.

In some cases, it became necessary to modify both the **Abstract_machine** and the **Abstract_program** to reflect new information. At the outset, for example, in **System** SP (figures 4.4 and 4.5) the verification of the state of a project did not appear. It was only after discussions with the accountant that we were able to determine that his operation was being interrupted. We were then able to determine where the interrupt came from, and as a result added **Precondition** P5 and **Action** A13 to SP (figure 4.5). We also added **Alternatives** 3.3, 4.3, and 5.3 (figure 4.4).

Because we were able to use ABL to model both the entire office (figures 4.2 and 4.3) and the details associated with the preparation of expense cheques (figures 4.11, 4.12 and 4.13) there was no need to use different techniques for different aspects of the model. We did not need to try to determine the best technique at each level of the office hierarchy. Although only one instance of concurrency occurs in the model (figure 4.5), it is evident that ABL can represent it adequately. We did not test the ability of the model to be directly executed; however, it is clear that the kind of calculations that appear in the accounting office (figure 4.7) can easily be programmed into a computer.

Although work is in progress for the development of suitable user interfaces to ABL, much still remains to be done before the average office worker will be able to build, interpret and modify ABL **Systems** easily. Thus, although this researcher has every reason to believe that the model presented in section 4.3 reflects the true state of affairs in the TAO office, the displays we produced could not be interpreted by the TAO staff without extensive explanation. As was expected, they were too busy with their own work to take the time to try to understand the model (this observation confirms the requirement of simplicity of use, presented in section 2.3). The model builder should be able to use a "model editor" to build and display his model, and then use a "model simulator" to simulate execution of the model, without having to worry about the internal form of representation of the model.

A final problem is that of adequately representing the **Data_flow** component of an ABL **System**. The display presented in figure 3.2 is of limited utility, since it is very difficult to follow any particular **Data_object** through the **System**. A display similar to that of figure 4.12 would be of more use.

CHAPTER V

CONCURRENT ACTIVITIES IN THE OFFICE

Computer-based tools have been finding their way into the office environmnt because of a "need to improve the productivity of both clerical and managerial office employees" and because of a need "to deal with information processing in increasingly complex and rapidly changing organizational environments" (OLSON82). These tools can be broadly classified as either information processing, or information transmission (communication) tools. With the advent of inexpensive microprocessors and reliable secondary storage media, workstations that offer a high degree of computational power are readily available. By providing the individual office worker with processing capabilities that were only available in the past on large centralized computers, the workstation functions can be "tailored to different roles; e.g., managerial, professional, secretarial, or even to individuals" (OLSON82).

Telephone calls, memos, and face-to-face meetings

are the traditional means of transmitting information in the office environment. If individual workstations can communicate with each other through a communication subnetwork, then much of the inter-office and intra-office communication can be automated. When the processing ability of a computationally powerful workstation is integrated with reliable inter-workstation communication "office work will be performed more efficiently" (OLSON82).

Local area networks (LANs), private automatic branch exchanges (PABXs) and hybrid networks combining both LANs and PABXs are being considered and marketed as solutions to the integration of processing and communication in the office (BERNH82). The different communications hardware available can be divided into three classes: Those providing high bandwidth over short distances, those providing low bandwidth over long distances, and those providing medium bandwidth over medium distances. The terms "high", "medium", and "low" are relative, and the actual hardware chosen for a particular office environment will depend on three factors: 1) The type of communication expected within the office environment (analog, digital, continuous, sporadic, etc.); 2) the physical distribution of individual workstations (all on one floor, spread throughout a building, scattered accross a city, world-wide); 3) and the cost and availability of the hardware (LANs have "one foot in the laboratory", PABXs use

the "old reliable" telephone system (BERNH82). It is probably the case that a large organization will require several different kinds of interconnected networks, each having its own specific application.

In the office, both the processes that operate on data objects, and the data objects themselves may be shared and used simultaneously. At some level of abstraction in the model of the office, these parallel aspects will have to be captured. If the network is an LAN with a sufficiently high bandwidth, such as CUENET (GROSS82), then it will be possible to simulate this parallelism. Each task or activity can be subdivided into smaller subtasks so that the subtasks can be executed by those workstations that otherwise would be idle.

5.1 Shared data objects in the office

Whether we were to examine one office in an organization, or several offices throughout the firm, we would find many data objects being simultaneously used by different individuals. The simultaneous access may be for "read only" purposes, or for "read and modify" operations. The question of data sharing in general has been well-studied (e.g., (ULLMA80)), and will not be dealt with here. However, in order to see how data sharing can arise in the office we provide an example.

Suppose I need 100 widgets. I fill out a purchase

request, noting the cost is $10.00 per widget. I send the request to the purchasing department, keeping a copy of it for my records. When my manager asks me to give him an accounting of my expenses, I retrieve my purchase requests, add them up, and give him the total. At the same time that I am using the purchase request for this purpose, the purchasing agent is combining my request for 100 widgets with similar requests from other departments. We are simultaneously accessing the purchase request in "read only" mode. As a result of combining requests from several departments, the purchasing agent is able to obtain the widgets for $9.50 each. He now modifies all the requests he has (a "read and modify" operation), and then instructs the issuers of the purchase requests to do the same. Once all the changes have been made, the integrity of the purchase request has been restored, since all the copies are the same.

5.2 Shared activities in the office

In addition to sharing data, however, it is possible for activities to be shared in the office. For example, the process of retrieving a file can be undertaken by several different individuals. The general concept of shared processes has also been well-studied (e.g., (LISTE79)) and will not be discussed here. We again, however, provide an example drawn from the office

environment.

A department is producing a report on its activities. Once the report has been written in draft form, it is necessary to have it typed. Because of the need for speed, rather than have one typist produce the entire report, the document is divided into sections and each section is given to a different typist. The size of the sections is determined by the number of typists available, their relative speed, and the number of logical divisions in the report. We should beware of giving too little (e.g., a page) to each typist to do. If the typists finish their tasks too soon, they will be spending most of their time in communicating; that is, in getting more work to do.

In general, we can consider that there is some activity to be performed (e.g., typing the report). This activity or task can be subdivided in such a way that each of the sub-tasks can be performed or executed by one of several "processors" (individuals or departments in an office) working in parallel. Each processor requires a certain amount of time and certain data objects to execute the sub-task assigned to it. We call this time and resource requirement the execution cost for that processor. Also, processors communicate between each other, transmitting intermediate results to each other. Communication between two processors takes a certain amount

of time. This time is made up of time taken by one processor to transmit, time for the transmission, and time taken by the other processor to receive. The total time is called the communication cost between the two processors. (We note in passing that when a processor is transmitting or receiving, it is not executing). If we sum the execution costs for all the processors we obtain the total execution cost for the activity. If we sum, the communication costs between all the processors, we obtain the total communication cost for the activity. Since the concern is to utilize the processors (e.g., typists) to the maximum a decomposition that will guarantee that the total communication cost is much less than the total execution cost is desirable. Thus, we are faced with the problem of taking a problem and decomposing it into several smaller sub-problems (each of which constitutes a process) so that the entire problem can be solved with minimum inter-process communication.

5.3 An algorithm for problem decomposition

Using ABL, it has been possible to develop an algorithm which resolves the question of problem decomposition, under certain constraints. In general, we assume that the problem can be expressed as an ABL **System**, and that any ABL **Step** can be processed on any of the available processors on a network. For the purposes of

office activity, these processors need not be machines, but may in fact be human beings. For the purposes of this algorithm we assume that it is possible to determine the cost of individual **Actions** in the **Abstract_machine**, and the probability of selecting a given **Alternative** for execution at each **Step**. Finally, it is assumed that the **Control_flow** of the **Abstract_program** satisfies the Markov property. This is a property of a stochastic process such that the probability of the process being in any given state of its state space is independent of the states in which it was previously (KNUTH73), (ISAAC76).

We can summarize this algorithm, ASSIGN, as follows: After developing an ABL representation of the problem, determine the costs of **Actions**, **Preconditions**, and inter-**Step** communication and the probabilities of each **Alternative** being chosen for execution from among the **Alternatives** in the same **Step**. Based on these values, determine the cost of each **Step**. Using Markov chain analysis find the expected number of times each **Step** is executed, and then use that information to determine the total communication cost of the **System**. This cost can then be compared to the total execution cost of the **System**. The criteria for termination are as follows: 1) The number of **Steps** is less than or equal to the number of free processors available for the problem, and 2) The execution cost compared to the total cost (execution cost plus

communication cost) is greater than a specified value. The second criterion is a measure of the relative utilization of the processors. If the algorithm does not terminate, then merge two **Steps** together, re-calculate the costs and probabilities, and start over. In selecting which **Steps** to merge, choose those **Steps** that communicate the most (Algorithm MERGE performs the merge). (ABL representations of algorithms ASSIGN and MERGE are presented in appendix F.)

ALGORITHM ASSIGN

{Assign clusters to processors}

Given: A problem stated in a natural language, or a program in a computer language, and F processors. It is assumed, for the purposes of this algorithm, that all processors are identical, and can process any part of a problem.

Step 1. [Translate the problem into ABL] Develop an ABL representation, Q, of the problem or program. This process is described in (HORVA82), (LEBEN81), and (MORGA81). From Q we immediately obtain the following: c, the number of **Steps** in Q; ai, the number of **Alternatives** in **Step** i, where $1 \leq i \leq c$; p, the number of **Preconditions** in Q; t, the number of **Actions** in Q; and N_{ij}, the **Next_step** of **Alternative** i in **Step** j,

where $1 \leq j \leq c$.

Step 2. [Determine initial values] Use external knowledge to determine the following: (this knowledge can come from observation, measurement, or simulation, depending on the application)

determine $P(A_{ij})$; where $1 \leq i \leq aj$, $1 \leq j \leq c$.

determine $K(T_{ijk})$; where $1 \leq i \leq t$, $1 \leq j \leq ak$, $1 \leq k \leq c$.

{Note that these costs may differ from **Alternative** to **Alternative**.}

determine $K(P_{ij})$; where $1 \leq i \leq p$, $1 \leq j \leq c$.

{Note that this may differ from **Step** to **Step**.}

determine $K(C_{ij})$; where $0 \leq i \leq c$, $0 \leq j \leq c$.

{We use the convention that C_0 is the terminal **Step**.}

Step 3. [Determine the cost of each **Alternative**]

$K(A_{ij}) = \sum_{k=1}^{t} K(T_k)$; where $1 \leq j \leq c$, $1 \leq i \leq aj$.

Step 4. [Determine the cost of each **Step**]

The cost of each **Step**, is

$K(C_i) = \sum_{j=1}^{ai} (K(A_{ji}) * P(A_{ji})) + \sum_{j=1}^{p} K(P_{ji})$; where $0 \leq i \leq c$.

Step 5. [Create a transition-probability matrix] Let B be a square matrix of dimension c+1. Then the B_{ij} entries represent the sum of $P(A_{ki})$ such that $N_{ki} = j$, for $1 \leq k \leq ai$. That is,

$B_{ij} = \sum_{k=1}^{ai} P(A_{ki})$, such that $N_{ki} = j$.

{B_{ij} represents the total probability of branching to

Step j from **Step** i.}

Step 6. [Create substochastic matrix] Let M be a square matrix of dimension c+1. {In order that the substochastic matrix be in the correct form for step 7 of this algorithm, the entries of B have to be rearranged so that the entries corresponding to final states are in the last row and column. Since C_0 is the terminal **Step** in an ABL **System**, the reordering is necessary. M is the reordered matrix.} The elements of M_{ij} are determined as follows:

$M_{ij} = B_{ij}$; where $1 \leq i \leq c$, $1 \leq j \leq c$.

$M_{ij} = 0$; where $i = c+1$, $1 \leq j \leq c+1$.

$M_{ij} = B_{i0}$; where $j = c+1$, $1 \leq i \leq c$.

Step 7. [Find the average number of times the **Steps** are executed] Let $U = (I-M)^{-1}$. Then U exists, and U_{1i} is the average number of times C_i is executed during the execution of Q (Lemma III.4.1 in (ISAAC76), section 2.3.4.2 in (KNUTH73), and in (FORMA81)). Note that in both U and M the $(c+1)^{th}$ entry corresponds to the terminal **Step**.

Step 8. [Find the average number of times the **Next_step** of C_i is C_j] Let D be a square matrix of dimension c+1. Then D_{ij}, the average number of times the **Next_step** of C_i is C_j, is given by $B_{ij} * U_{1i}$.

Step 9. [Find the inter-**Step** communication costs] Let E be a square matrix of dimension c+1. Then E_{ij} is the

total cost of communicating between C_i and C_j during the execution of Q where

$E_{ij} = D_{ij} * K(C_{ij}) + D_{ji} * K(C_{ji})$; where $0 \leq i \leq c$, $0 \leq j \leq c$, $i \neq j$ and

$E_{ij} = 0$; where $0 \leq i \leq c$, $0 \leq j \leq c$, $i = j$.

{Note that we assume the $K(C_{ij})$ remain constant throughout the execution of the program}.

Step 10. [Find the total communication cost] Let G be the total communication cost during the execution of the program. Then,

$$G = \sum_{i=0}^{c-1} \sum_{j=i+1}^{c} E_{ij}$$

Step 11. [Find the total execution cost] Let H be the total cost of execution within Steps. Then,

$$H = \sum_{i=1}^{c} (K(C_i) * U_{1i})$$

Step 12. [Compare costs] If $c \leq F$ and $H/(G+H) > \phi$, where ϕ represents the desired level of processor utilization, then STOP.

Step 13. [Find those Steps with maximum inter-Step communication costs] Determine i and j for which E_{ij} is a maximum, where $i < j$. {Note that since $E_{ij} = E_{ji}$, this is not a restriction}. If there is more than one such a pair, then choose the one with maximum $[K(C_i) + K(C_j)]$.

Step 14. [Merge C_i and C_j into C_k] Call ALGORITHM MERGE. {Note that after MERGE, $C_k = c + 1$, $ak = ai + aj$, $t = t + 2$, $p = p + 1$}

Step 15. [Adjust probabilities] Calculate the probabilities of each **Alternative** in C_k, based on the probabilities in C_i and C_j.

$P(A_{hk}) = P(A_{hi}) * U_{1i}/(U_{1i}+U_{1j})$; where $1 \leq h \leq ai$,

$P(A_{ai+1,k}) = P(A_{hj}) * U_{1j}/(U_{1i}+U_{1j})$; where $1 \leq h \leq aj$.

Step 16. [Calculate the inter-**Step** communication costs.] For $1 \leq h \leq c$, for $0 \leq g \leq c$, and for $1 \leq m \leq ah$, if $N_{mh}=g$ then there are the following four cases to be considered:

{Case 1: The **Next_step** of an **Alternative** in one of the merged **Steps** is one of the merged **Steps**} C_h and C_g are both in $\{C_i, C_j\}$:

$K(C_{hk}) = 0$

$K(C_{kg}) = 0$

{Case 2: The **Next_step** of an **Alternative** in a merged **Step** is not one of the merged **Steps**} C_h is in $\{C_i, C_j\}$ and C_g is not:

$K(C_{hk}) = 0$

$K(C_{kg}) = K(C_{ig}) + K(C_{jg})$

{Case 3: The **Next_step** of an **Alternative** in a non-merged **Step** is a merged **Step**} C_g is in $\{C_i, C_j\}$ and C_h is not:

$K(C_{hk}) = K(C_{hi}) + K(C_{hj})$

$K(C_{kg}) = 0$

{Case 4: The **Next_step** of an **Alternative** in a

non-merged **Step** is not a merged **Step**] Neither C_h nor C_g is in $\{C_i, C_j\}$:

$K(C_{hk}) = 0$

$K(C_{kg}) = 0$

Step 17. [Insert C_k] Replace C_i by C_k, adjusting the probabilities, costs and **Next_steps** accordingly.

Step 18. [Decrease the number of **Steps**] Since two **Steps** have been merged into one, c = c - 1.

Step 19. [Repeat the cycle] Go to step number 3.

ALGORITHM MERGE

{Merge together C_i and C_j into a new **Step** C_k}

Step 1. [Determine the number of **Alternatives** in C_k]

ak = ai + aj.

Step 2. [Create set of **Preconditions** and assign values]

Add a new **Precondition** to the set of **Preconditions** in Q.

$P_{hk} = P_{hi}$ OR P_{hj}; where $1 \leq h \leq p$.

Let $V(P_{hmk})$ be the value of **Precondition** h in **Alternative** m of **Step** k. Then,

$V(P_{hmk}) = V(P_{hmi})$; where $1 \leq m \leq ai$

$V(P_{h,m+ai,k}) = V(P_{hmj})$; where $1 \leq m \leq aj$

$P_{p+1,k}$ has meaning "$FLAG_k$ = TRUE?"

$V(P_{p+1,m,k})$ = "Y"; where $1 \leq m \leq ai$

$V(P_{p+1,m+ai,k})$ = "N"; where $1 \leq m \leq aj$

p := p + 1

Step 3. [Create set of **Actions**] Add two new **Actions** to the set of **Actions** in Q.

$T_{hgk} = T_{hgi}$; where $1 \leq h \leq t$, $1 \leq g \leq ai$

$T_{h,g+ai,k} = T_{hgj}$; where $1 \leq h \leq t$, $1 \leq g \leq aj$

Step 4. [Add appropriate **Actions** to all the **Alternatives**] In the following, the notation $T_{last,h,k}$ = "x" means the last **Action** performed in **Alternative** h in **Step** k is the **Action** denoted by the expression x.

{Case 1: The **Next_step** of an **Alternative** in the first merged **Step** is itself} When $N_{hi} = i$; where $1 \leq h \leq ai$:

$T_{last,h,k}$ = "$FLAG_k$ = TRUE"

{Case 2: The **Next_step** of an **Alternative** in the first merged **Step** is the other merged **Step**} When $N_{hi} = j$; where $1 \leq h \leq ai$:

$T_{last,h,k}$ = "$FLAG_k$ = FALSE"

{Case 3: The **Next_step** of an **Alternative** in the second merged **Step** is the other merged **Step**} When $N_{hj} = i$; where $1 \leq h \leq aj$:

$T_{last,h+ai,k}$ = "$FLAG_k$ = TRUE"

{Case 4: The **Next_step** of an **Alternative** in the second merged **Step** is itself} When $N_{hj} = j$; where $1 \leq h \leq aj$:

$T_{last,h+ai,k}$ = "$FLAG_k$ = FALSE"

{Case 5: The **Next_step** of an **Alternative** in a non-merged **Step** is the first merged **Step**} When

$N_{hm} = i$; where $m \neq i$, $m \neq j$, $1 \leq h \leq am$:

$T_{last,h,m}$ = "$FLAG_k$ = TRUE"

{Case 6: The **Next_step** of an **Alternative** in a non-merged **Step** is the second merged **Step**} When $N_{hm} = j$; where $m \neq i$, $m \neq j$, $1 \leq h \leq am$:

$T_{last,h,m}$ = "$FLAG_k$ = FALSE"

{Case 7: The **Next_step** of an **Alternative** in any **Step** is not a merged **Step**} For all other cases, do not add a T_{last}.

Step 5. [Modify **Next_step** indicators]

{Case 1: The **Next_step** of an **Alternative** in the first merged **Step** is a merged **Step**} When $N_{hi} = i$ or $N_{hi} = j$; where $1 \leq h \leq ai$:

$N_{hk} = k$

{Case 2: The **Next_step** of an **Alternative** in the first merged **Step** is not a merged **Step**} When $N_{hi} \neq i$ and $N_{hi} \neq j$; where $1 \leq h \leq ai$:

$N_{hk} = N_{hi}$

{Case 3: The **Next_step** of an **Alternative** in the second merged **Step** is a merged **Step**} When $N_{hj} = i$ or $N_{hj} = j$; where $1 \leq h \leq aj$:

$N_{h+ai,k} = k$

{Case 4: The **Next_step** of an **Alternative** in the second merged **Step** is not a merged **Step**} When $N_{hj} \neq i$ and $N_{hj} \neq j$; where $1 \leq h \leq aj$:

$N_{h+ai,k} = N_{hj}$

{Case 5: The **Next_step** of an **Alternative** in a non-merged **Step** is a merged **Step**} When $N_{hm} = i$ or $N_{hm} = j$; where $m \neq i$, $m \neq j$, $1 \leq h \leq am$:

$N_{hm} = k$

{Case 6: The **Next_step** of an **Alternative** in a non-merged **Step** is not a merged **Step**} For all other cases, N_{hm} remains the same; where $m \neq i$, $m \neq j$, $1 \leq h \leq am$.

Step 6. [Adjust the number of **Actions**] $t = t + 2$

Step 7. STOP

Algorithm ASSIGN converges to a solution under certain conditions. We should first note that at each iteration the communication cost decreases. This decrease, however, is not sufficient to guarantee convergeance. If the execution cost decreases from iteration q to the next iteration, q+1, as well, then we can guarantee convergeance if the following two conditions are met: 1) the decrease in execution cost must be less than the decrease in communication cost, and 2) the execution cost at iteration q must be greater than the communication cost at the same iteration.

To show that ASSIGN converges, we need to show that in step 12 of the algorithm, both the following inequalities hold:

$$c^{q+1} < c^q \qquad (1)$$

and

$$H^{q+1}/(G^{q+1}+H^{q+1}) \geq H^{q}/(G^{q}+H^{q}) \qquad (2)$$

where q is the number of the iteration of the algorithm.

Proof: (1) follows directly as a result of step 18 of algorithm ASSIGN.

In order to prove (2), we make the following assumption,

if $H^{q+1} < H^{q}$ then (a) $H^{q} - H^{q+1} < G^{q} - G^{q+1}$, and

(b) $G^{q} < H^{q}$

and establish the following Lemma:

Lemma: $G^{q+1} < G^{q}$.

Proof of Lemma:

$G^{q} = \sum_{i=0}^{c-1} \sum_{j=i+1}^{c} E_{ij}^{q}$; by step 10 of algorithm ASSIGN.

$= E_{01}^{q} + \ldots + E_{nm}^{q} + \ldots + E_{c-1,c}^{q}$

Without loss of generality, let n=i and m=j in step 13 of algorithm ASSIGN. Then E_{nm}^{q} is a maximum.

Therefore, by step 9 of algorithm ASSIGN,

$E_{nm}^{q} = D_{nm}^{q} K(C_{nm})^{q} + D_{mn}^{q} K(C_{mn})^{q} \neq 0$.

Therefore, either $K(C_{nm})^{q}$ or $K(C_{mn})^{q}$, or both are non-zero.

If $K(C_{nm})^{q} \neq 0$, then there is at least one **Alternative**, h, such that $N_{hn} = m$.

Then, by step 16 of algorithm ASSIGN, case 1,

$K(C_{nk})^{q} = 0$ and $K(C_{km})^{q} = 0$.

Then, by step 17 of algorithm ASSIGN,

$K(C_{ni})^{q+1} = 0$ and $K(C_{im})^{q+1} = 0$.

That is,

$K(C_{nn})^{q+1} = 0$ and $K(C_{nm})^{q+1} = 0$.

Similarly, if $K(C_{mn})^{q} \neq 0$, then $K(C_{mn})^{q+1} = 0$.

Since the only inter-Step communication cost in the ABL System that has been changed is that between C_m and C_n, the following is true:

$E_{ij}^{q+1} = E_{ij}^{q}$; where $i \neq m$, and $j \neq m$.

By step 16 of algorithm ASSIGN,

$E_{ij}^{q+1} = 0$; where $i=m$, or $j=m$.

Thus, $G^{q+1} = \sum_{i=0}^{c-1} \sum_{j=i+1}^{c} E_{ij}^{q+1}$

$= E_{01}^{q+1} + \ldots + E_{nm}^{q+1} + \ldots + E_{m,m+1}^{q+1} + \ldots + E_{c-1,c}^{q+1}$

$= E_{01}^{q+1} + \ldots + 0 + \ldots + 0 + \ldots + E_{c-1,c}^{q+1}$

$= E_{01}^{q} + \ldots + 0 + \ldots + 0 + \ldots + E_{c-1,c}^{q}$

$= G^{q} - \sum_{j=1}^{c} E_{mj}^{q} - \sum_{i=1}^{c} E_{im}^{q}$

Thus, $G^{q+1} \leq G^{q}$.

We note that we can express the relationship between H^{q+1} and H^{q} as:

$$H^{q+1} = H^{q} + L,$$

where L is either >0, <0 or =0.

Proof of expression (2):

The proof is by contradiction. Suppose

$$H^{q+1}/(G^{q+1}+ H^{q+1}) < H^q/(G^q+ H^q) \quad (3)$$

then there are three cases to be examined:

Case 1: L = 0:

Then, $H^{q+1} = H^q$ (4)

Applying (4) to (3) we obtain

$$H^q/(G^{q+1}+ H^q) < H^q/(G^q+ H^q)$$

$$G^q + H^q < G^{q+1} + H^q$$

$$G^q < G^{q+1} \quad (5)$$

But (5) contradicts the Lemma.

Therefore, for Case 1, (3) is false.

Case 2: L > 0:

Then, $H^{q+1} > H^q$ (6)

From (3) we obtain

$$H^{q+1}G^q + H^{q+1}H^q < H^qG^{q+1} + H^qH^{q+1}$$

$$H^{q+1}G^q < H^qG^{q+1}$$

That is,

$$H^{q+1}/H^q < G^{q+1}/G^q$$

By (6), the L.H.S (Left Hand Side) is greater than 1; but by the Lemma, the R.H.S (Right Hand Side) is less than 1.

Therefore, for Case 2, (3) is false.

Case 3: L < 0:

Then, $H^{q+1} < H^q$ (7)

From (3) we obtain

$$H^{q+1}G^q < H^qG^{q+1}$$

$$H^{q+1}/H^q < G^{q+1}/G^q$$

$$(H^q-H^q+H^{q+1})/H^q < (G^q-G^q+G^{q+1})/G^q$$

$$1 - (H^q-H^{q+1})/H^q < 1 - (G^q-G^{q+1})/G^q$$

From the Lemma and (7), we obtain,

$$(G^q-G^{q+1})/G^q < (H^q-H^{q+1})/H^q$$

$$(G^q-G^{q+1})/(H^q-H^{q+1}) < G^q/H^q$$

But, by assumption (a), the L.H.S. is greater than 1 and, by assumption (b), the R.H.S. is less than 1.

Therefore, for Case 3, (3) is false.

Therefore (2) is true.

Therefore, the algorithm will converge under the following condition: If the execution cost decreases from one iteration to the next, then that decrease should be less than the corresponding decrease in communication cost, and the communication cost for the first iteration should be less than the execution cost for that iteration.

5.4 The algorithm applied to an example

For the purposes of demonstrating how the algorithm can be applied, we will take as an example the TAO function of issuing expense cheques (figures 4.11, 4.12, and 4.13). We will assume that these seven **Steps** are to be performed by three people; that is, we need to have each person

perform more than one of these **Steps**. Based on interviews with TAO personnel, we were able to arrive at the probabilities presented in table 5.1, and the costs presented in tables 5.2, 5.3, and 5.4. These values are estimates only, and are used to demonstrate the application of this algorithm.

One of the measures that can be used to evaluate a decomposition is what we call the "processor utilization efficiency." If we consider the total cost of a **System** to be the sum of the execution cost and the communication cost, then we define the processor utilization efficiency to be the percentage of the total cost that is represented by the execution cost. When we applied algorithm ASSIGN to the **System** EX, using the figures obtained from the TAO personnel, we obtained the results summarized in table 5.5.

After the first iteration of the algorithm, we obtain the **System** shown in figure 5.1. The **Preconditions** and the **Actions** that have been added as a result of applying algorithm MERGE are outlined in the figure. Similarly, figures 5.2, 5.3, and 5.4 represent the state of the **System** after successive iterations. Again, the added **Preconditions** and **Actions** are outlined.

We can interpret the final version of the expense cheque operation, presented in figure 5.4, as follows: Assuming all our cost information is correct, and the relative frequency of **Alternative** execution is as in table

TABLE 5.1

PROBABILITIES OF ALTERNATIVES IN PROGRAM EX

Alternative	Probability
1.1	0.0133
1.2	0.9867
2.1	0.8000
2.2	0.2000
3.1	0.9467
3.2	0.0533
4.1	0.8933
4.2	0.0133
4.3	0.0934
5.1	0.9500
5.2	0.0500
6.1	0.0010
6.2	0.9990
7.1	0.0480
7.2	0.0050
7.3	0.7980
7.4	0.1490

TABLE 5.2

COSTS OF ACTIONS IN PROGRAM EX

Action	Cost (minutes)
A1	1.0
A2	0.5
A3	5.0
A4	10.0
A5	10.0
A6	3.0
A7	5.0
A8	60.0
A9	30.0
A10	6600.0
A11	5.0

TABLE 5.3

COSTS OF PRECONDITIONS IN PROGRAM EX

Precondition	Cost (minutes)
P1	1.0
P2	1.0
P3	1.0
P4	1.0
P5	2.0
P6	1.0
P7	0.5
P8	0.5
P9	0.5
P10	0.5

TABLE 5.4

COMMUNICATION COSTS IN PROGRAM EX

From Step	To Step	Cost (minutes)
1	0	0.0
1	2	0.5
2	3	0.5
2	5	2.0
3	4	0.5
3	5	2.0
4	1	1.0
4	6	1.5
5	1	1.0
6	1	1.5
6	7	2.0
7	5	2.0

TABLE 5.5

RESULTS OBTAINED FROM APPLYING ASSIGN TO PROGRAM EX

Iteration	Figure	Execution cost (minutes)	Communication cost (minutes)	Processor utilization efficiency (%)
0	4.13	2967.60	233.898	92.6941
1	5.1	3282.81	182.956	94.7210
2	5.2	3800.83	127.961	96.7430
3	5.3	5235.54	78.897	98.5150
4	5.4	5438.09	28.552	99.4780

	PROGRAM																		MACHINE
C 1	V	V	V	V	V	.	.	.	.	.	.	.	.	.	.	.	.	V	C 1 Next employee, travel report.
C 2	.	.	.	.	.	V	V	.	.	.	.	.	.	.	.	.	.	V	C 2 Process auto expense report.
C 3	.	.	.	.	.	.	.	V	V	.	.	.	.	.	.	.	.	V	C 3 Process general expense report.
C 4	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	.	.	V	C 4 Process unaccounted-for amounts.
C 5	.	.	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	V	C 5 Verify receipts.
C 6	.	.	.	.	.	.	.	.	.	.	.	.	.	V	V	V	V	V	C 6 Process over- or under-payment.
																			PRECONDITIONS
P 1	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P 1 More employees?
P 2	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	P 2 Auto expense report submitted?
P 3	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	P 3 General expense report submitted?
P 4	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	P 4 Travel expense report submitted?
P 5	-	-	N	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	P 5 Employee returned from trip?
P 6	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	P 6 Amounts still unaccounted for?
P 7	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	P 7 Totals on form = total of receipts?
P 8	-	-	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	P 8 Balance due employee?
P 9	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	N	-	Y	-	P 9 Balance due employer?
P 10	-	-	-	-	-	-	-	-	-	-	-	-	-	N	-	-	Y	-	P 10 Payment enclosed?
P 11	Y	Y	N	N	N	-	-	-	-	-	-	-	-	-	-	-	-	-	P 11 Flag 8 = TRUE?
																			ACTIONS
A 1	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A 1 Get employee name.
A 2	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	A 2 Amount := $ 0.22 * mileage.
A 3	.	.	.	.	.	.	2	.	1	.	.	.	1	.	.	.	.	.	A 3 Enter total by job number on form.
A 4	.	.	.	.	.	.	3	.	2	.	.	.	2	.	.	.	.	.	A 4 Post total amount / job to job cost cards.
A 5	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	A 5 Post amounts to general expenses in general ledger.
A 6	.	.	.	.	.	.	4	.	3	.	.	.	.	.	1	.	.	.	A 6 Issue cheque.
A 7	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A 7 Contact appropriate secretary.
A 8	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	A 8 Add up receipts.
A 9	.	.	.	2	.	.	.	.	.	.	.	1	.	1	.	.	.	.	A 9 Put claim on hold.
A 10	.	.	.	.	.	.	.	.	.	.	.	2	.	2	.	.	.	.	A 10 Query employee.
A 11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	A 11 Deposit payment.
A 12	.	.	1	3	.	.	.	.	.	1	2	3	.	.	.	.	.	.	A 12 Flag 8 := TRUE.
A 13	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	A 13 Flag 8 := FALSE.
NEXT	0	2	1	1	5	3	4	1	4	1	1	1	6	4	4	4	4	0	

Figure 5.1

EX: After 1 iteration of ASSIGN

	PROGRAM																				MACHINE		
C	1	V	V	V	V	V	V	V	.	.	.	.	.	.	.	.	.	.	.	V	C	1	Next employee, travel, auto reports.
C	2	.	.	.	.	.	.	.	V	V	.	.	.	.	.	.	.	.	.	V	C	2	Process general expense report.
C	3	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	.	.	.	V	C	3	Process unaccounted-for amounts.
C	4	.	.	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	.	V	C	4	Verify receipts.
C	5	.	.	.	.	.	.	.	.	.	.	.	.	.	V	V	V	V	V	V	C	5	Process over- or under-payment.
																							PRECONDITIONS
P	1	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P	1	More employees?
P	2	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	P	2	Auto expense report submitted?
P	3	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	P	3	General expense report submitted?
P	4	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P	4	Travel expense report submitted?
P	5	-	-	N	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P	5	Employee returned from trip?
P	6	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	P	6	Amounts still unaccounted for?
P	7	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	P	7	Totals on form = total of receipts?
P	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	P	8	Balance due employee?
P	9	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	N	-	-	Y	-	P	9	Balance due employer?
P	10	-	-	-	-	-	-	-	-	-	-	-	-	-	N	-	-	Y	-	-	P	10	Payment enclosed?
P	11	Y	Y	N	N	N	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P	11	Flag 8 = TRUE?
P	12	Y	Y	Y	Y	Y	N	N	-	-	-	-	-	-	-	-	-	-	-	-	P	12	Flag 7 = TRUE?
																							ACTIONS
A	1	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	1	Get employee name.
A	2	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	A	2	Amount := $ 0.22 * mileage.
A	3	.	.	.	.	.	.	2	.	1	.	.	.	1	.	.	.	.	.	.	A	3	Enter total by job number on form.
A	4	.	.	.	.	.	.	3	.	2	.	.	.	2	.	.	.	.	.	.	A	4	Post total amount / job to job cost cards.
A	5	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	A	5	Post amounts to general expenses in general ledger.
A	6	.	.	.	.	.	.	4	.	3	.	.	.	.	.	1	.	.	.	.	A	6	Issue cheque.
A	7	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	7	Contact appropriate secretary.
A	8	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	8	Add up receipts.
A	9	.	.	.	2	.	.	.	.	.	.	.	1	.	1	.	.	.	.	.	A	9	Put claim on hold.
A	10	.	.	.	.	.	.	.	.	.	.	.	2	.	2	.	.	.	.	.	A	10	Query employee.
A	11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	A	11	Deposit payment.
A	12	.	.	1	3	.	.	.	.	.	1	2	3	.	.	.	.	.	.	.	A	12	Flag 8 := TRUE.
A	13	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	A	13	Flag 8 := FALSE.
A	14	.	.	2	4	.	.	.	2	.	2	3	4	.	.	.	.	.	.	.	A	14	Flag 7 := TRUE.
A	15	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	15	Flag 7 := FALSE.
NEXT		0	1	1	1	4	2	3	1	3	1	1	1	5	3	3	3	3	0				

Figure 5.2

EX: After 2 iterations of ASSIGN

		PROGRAM																		MACHINE	
C	1	V	V	V	V	V	V	V	V	V	.	.	.	.	.	.	.	.	V	C	1 Next employee, travel, general, auto reports.
C	2	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	.	.	V	C	2 Process unaccounted-for amounts.
C	3	.	.	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	V	C	3 Verify receipts.
C	4	.	.	.	.	.	.	.	.	.	.	.	.	.	V	V	V	V	V	C	4 Process over- or under-payment.
																					PRECONDITIONS
P	1	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P	1 More employees?
P	2	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	P	2 Auto expense report submitted?
P	3	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	P	3 General expense report submitted?
P	4	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	P	4 Travel expense report submitted?
P	5	-	-	N	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	P	5 Employee returned from trip?
P	6	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	P	6 Amounts still unaccounted for?
P	7	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	P	7 Totals on form = total of receipts?
P	8	-	-	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	P	8 Balance due employee?
P	9	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	N	-	Y	-	P	9 Balance due employer?
P	10	-	-	-	-	-	-	-	-	-	-	-	-	-	N	-	-	Y	-	P	10 Payment enclosed?
P	11	Y	Y	N	N	N	-	-	-	-	-	-	-	-	-	-	-	-	-	P	11 Flag 8 = TRUE?
P	12	Y	Y	Y	Y	Y	N	N	-	-	-	-	-	-	-	-	-	-	-	P	12 Flag 7 = TRUE?
P	13	Y	Y	Y	Y	Y	Y	Y	N	N	-	-	-	-	-	-	-	-	-	P	13 Flag 6 = TRUE?
																					ACTIONS
A	1	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	1 Get employee name.
A	2	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	A	2 Amount := $ 0.22 * mileage.
A	3	.	.	.	.	.	.	2	.	1	.	.	.	1	.	.	.	.	.	A	3 Enter total by job number on form.
A	4	.	.	.	.	.	.	3	.	2	.	.	.	2	.	.	.	.	.	A	4 Post total_amount / job to job cost cards.
A	5	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	A	5 Post amounts to general expenses in general ledger.
A	6	.	.	.	.	.	.	4	.	3	.	.	.	.	.	.	1	.	.	A	6 Issue cheque.
A	7	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	7 Contact appropriate secretary.
A	8	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	A	8 Add up receipts.
A	9	.	.	.	2	.	.	.	.	.	.	.	1	.	1	.	.	.	.	A	9 Put claim on hold.
A	10	.	.	.	.	.	.	.	.	.	.	.	2	.	2	.	.	.	.	A	10 Query employee.
A	11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	A	11 Deposit payment.
A	12	.	.	1	3	.	.	.	.	.	1	2	3	.	.	.	.	.	.	A	12 Flag 8 := TRUE.
A	13	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	A	13 Flag 8 := FALSE.
A	14	.	.	2	4	.	.	.	2	.	2	3	4	.	.	.	.	.	.	A	14 Flag 7 := TRUE.
A	15	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A	15 Flag 7 := FALSE.
A	16	.	3	3	5	.	.	.	3	.	3	4	5	.	.	.	.	.	.	A	16 Flag 6 := TRUE.
A	17	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	A	17 Flag 6 := FALSE.
NEXT		0	1	1	1	3	1	2	1	2	1	1	1	4	2	2	2	2	0		

Figure 5.3

EX: After 3 iterations of ASSIGN

		PROGRAM																			MACHINE
C	1	V	V	V	V	V	V	V	V	V	V	V	.	.	.	.	.	.	.	V	C 1 Next employee, all reports, unaccounted amounts.
C	2	.	.	.	.	.	.	.	.	.	.	.	V	V	.	.	.	.	.	V	C 2 Verify receipts.
C	3	.	.	.	.	.	.	.	.	.	.	.	.	.	V	V	V	V	V	V	C 3 Process over- or under-payment.
																					PRECONDITIONS
P	1	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P 1 More employees?
P	2	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	P 2 Auto expense report submitted?
P	3	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	P 3 General expense report submitted?
P	4	-	-	-	N	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P 4 Travel expense report submitted?
P	5	-	-	N	Y	Y	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P 5 Employee returned from trip?
P	6	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	-	-	P 6 Amounts still unaccounted for?
P	7	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	-	-	P 7 Totals on form = total of receipts?
P	8	-	-	-	-	-	-	-	-	-	-	-	-	-	N	Y	-	-	-	-	P 8 Balance due employee?
P	9	-	-	-	-	-	-	-	-	-	-	-	-	-	Y	N	-	Y	-	-	P 9 Balance due employer?
P	10	-	-	-	-	-	-	-	-	-	-	-	-	-	N	-	-	Y	-	-	P 10 Payment enclosed?
P	11	Y	Y	N	N	N	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P 11 Flag 8 = TRUE?
P	12	Y	Y	Y	Y	Y	N	N	-	-	-	-	-	-	-	-	-	-	-	-	P 12 Flag 7 = TRUE?
P	13	Y	Y	Y	Y	Y	Y	Y	N	N	-	-	-	-	-	-	-	-	-	-	P 13 Flag 6 = TRUE?
P	14	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	-	-	-	-	-	-	-	-	P 14 Flag 5 = TRUE?
																					ACTIONS
A	1	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A 1 Get employee name.
A	2	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	A 2 Amount := $ 0.22 * mileage.
A	3	.	.	.	.	.	.	2	.	1	.	.	.	1	.	.	.	.	.	.	A 3 Enter total by job number on form.
A	4	.	.	.	.	.	.	3	.	2	.	.	.	2	.	.	.	.	.	.	A 4 Post total amount / job to job cost cards.
A	5	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	A 5 Post amounts to general expenses in general ledger.
A	6	.	.	.	.	.	.	4	.	3	.	.	.	.	.	.	1	.	.	.	A 6 Issue cheque.
A	7	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A 7 Contact appropriate secretary.
A	8	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A 8 Add up receipts.
A	9	.	.	.	2	.	.	.	.	.	.	.	1	.	1	.	.	.	.	.	A 9 Put claim on hold.
A	10	.	.	.	.	.	.	.	.	.	.	.	2	.	2	.	.	.	.	.	A 10 Query employee.
A	11	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	A 11 Deposit payment.
A	12	.	.	1	3	.	.	.	.	.	1	2	3	.	.	.	.	.	.	.	A 12 Flag 8 := TRUE.
A	13	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	A 13 Flag 8 := FALSE.
A	14	.	.	2	4	.	.	.	2	.	2	3	4	.	.	.	.	.	.	.	A 14 Flag 7 := TRUE.
A	15	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	A 15 Flag 7 := FALSE.
A	16	.	3	3	5	.	.	.	3	.	3	4	5	.	.	.	.	.	.	.	A 16 Flag 6 := TRUE.
A	17	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	A 17 Flag 6 := FALSE.
A	18	.	4	4	6	.	2	.	4	.	4	5	6	.	.	.	.	.	.	.	A 18 Flag 5 := TRUE.
A	19	.	.	.	.	.	.	5	.	4	.	.	.	.	3	1	2	2	.	.	A 19 Flag 5 := FALSE.
NEXT		0	1	1	1	2	1	1	1	1	1	1	1	3	1	1	1	1	0		

Figure 5.4

EX: After 4 iterations of ASSIGN

5.1, then if we only have three employees, and the activities they are required to perform can be modelled by the version of EX presented in figure 4.13 which requires seven employees, then we can improve the overall efficiency by assigning one person to doing the tasks represented by **Step** 1 in figure 5.4, another to receipt verification (**Step** 2), and the third to processing over- or under-payments (**Step** 3).

5.5 The problem of costing and ABL

The success of algorithm ASSIGN depends heavily on acquiring a good estimate of costs and probabilities. Such information is by no means easy to obtain in a real office, but ABL can be of assistance in acquiring and organizing this information. In order to demonstrate ABL's usefulness in this regard, we will use a well-studied example, Quicksort (WIRTH76).

Figure 5.5 is the ABL matrix form of Quicksort. We have taken the Pascal program as presented in (WIRTH76) and translated it into ABL without modifying either the control structures or the individual statements. No attempt has been made to optimize or alter the **Abstract_program.** Figure 5.6 is the narrative form of the same **System**, and it is provided for reference.

Before determining the various costs, we had to decide on a suitable environment. We chose the Motorola

```
                    PROGRAM                                    MACHINE
C  1  V . . . . . . . . . . . . . . . . . V  C  1 Initialize.
C  2  . V V . . . . . . . . . . . . . . . V  C  2 Scan from the left.
C  3  . . . V V . . . . . . . . . . . . . V  C  3 Scan from the right.
C  4  . . . . V V . . . . . . . . . . . . V  C  4 Exchange.
C  5  . . . . . . . . V V . . . . . . . . V  C  5 Sort.
C  6  . . . . . . . . . V V V V . . . . . V  C  6 Stack requests to sort.
C  7  . . . . . . . . . . . . V V . . . . V  C  7 Partition an interval.
C  8  . . . . . . . . . . . . . . V V V  V  C  8 Get the next request.

                                                    PRECONDITIONS
P  1  - Y N - - - - - - - - - - - - - - - -  P  1 a[i].key < x.key ?
P  2  - - - Y N - - - - - - - - - - - - - -  P  2 x.key < a[j].key ?
P  3  - - - - - Y N - - - - - - - - - - - -  P  3 i <= j ?
P  4  - - - - - - - - Y N - - - - - - - - -  P  4 i > j ?
P  5  - - - - - - - - - - Y N - - - - - - -  P  5 i < r ?
P  6  - - - - - - - - - - - - - Y N - - - -  P  6 l >= r ?
P  7  - - - - - - - - - - - - - - - Y N - -  P  7 s = 0 ?
P  8  - - - - - - - - - - - - Y N - - - - -  P  8 l < j ?
P  9  - - - - - - - - - - Y Y N N - - - - -  P  9 j-l < r-i ?

                                                    ACTIONS
A  1  1 . . . . . . . . . . . . . . . . . .  A  1 s := 1
A  2  2 . . . . . . . . . . . . . . . . . .  A  2 stack[1].l := l
A  3  3 . . . . . . . . . . . . . . . . . .  A  3 stack[1].r := n
A  4  4 . . . . . . . . . . . . . . . 1 . .  A  4 l := stack[s].l
A  5  5 . . . . . . . . . . . . . . . 2 . .  A  5 r := stack[s].r
A  6  6 . . . . . . . . . . . . . . . 3 . .  A  6 s := s-1
A  7  7 . . . . . . . . . . . . . 1 . 4 . .  A  7 i := l
A  8  8 . . . . . . . . . . . . . 2 . 5 . .  A  8 j := r
A  9  9 . . . . . . . . . . . . . 3 . 6 . .  A  9 x := a[(l+r) div 2]
A 10  . 1 . . . 4 . . . . . . . . . . . . .  A 10 i := i+1
A 11  . . . 1 . 5 . . . . . . . . . . . . .  A 11 j := j-1
A 12  . . . . . 1 . . . . . . . . . . . . .  A 12 w := a[i]
A 13  . . . . . 2 . . . . . . . . . . . . .  A 13 a[i] := a[j]
A 14  . . . . . 3 . . . . . . . . . . . . .  A 14 a[j] := w
A 15  . . . . . . . . . 1 . 1 . . . . . . .  A 15 s := s+1
A 16  . . . . . . . . . 2 . . . . . . . . .  A 16 stack[s].l := l
A 17  . . . . . . . . . 3 . . . . . . . . .  A 17 stack[s].r := r
A 18  . . . . . . . . . 4 1 . . . . . . . .  A 18 r := j
A 19  . . . . . . . . . . . 2 . . . . . . .  A 19 stack[s].l := l
A 20  . . . . . . . . . . . 3 . . . . . . .  A 20 stack[s].r := j
A 21  . . . . . . . . . . . 4 1 . . . . . .  A 21 l := i

NEXT  2 2 3 3 4 5 5 6 2 7 7 7 7 8 2 0 2 0
```

Figure 5.5

Quicksort: Matrix Form

STEP DESCRIPTION	NEXT STEP
1.0 Initialize.	
1.1 The stack has been initialized, and the array partitioned.	2
2.0 Scan from the left.	
2.1 Scanning continues.	2
2.2 Scanning has stopped.	3
3.0 Scan from the right.	
3.1 Scanning continues.	3
3.2 Scanning has stopped.	4
4.0 Exchange.	
4.1 Elements have been exchanged.	5
4.2 Elements have not been exchanged.	5
5.0 Sort.	
5.1 Left pointer has passed right pointer.	6
5.2 Left pointer has not passed right pointer.	2
6.0 Stack requests to sort.	
6.1 Stack request to sort right partition.	7
6.2 Continue sorting left partition.	7
6.3 Stack request to sort left partition.	7
6.4 Continue sorting right partition.	7
7.0 Partition an interval.	
7.1 This partition is sorted.	8
7.2 This partition has been partitioned.	2
8.0 Get the next request.	
8.1 No more requests on the stack.	0
8.2 Top request has been popped from the stack.	2

Figure 5.6

Quicksort: Narrative Form

6809 microprocessor for two reasons: 1) It is a popular microprocessor, and 2) the environment in which we would ultimately be able to carry out the simulation, CUENET (GROSS82), is designed around the 6809 microprocessor.

In order to arrive at reasonable costs for **Actions** and **Preconditions**, we followed the following strategy: We assumed that each **Action** and each **Precondition** was translated into Motorola 6809 assembly language instructions (LEVEN81). We then counted the number of machine cycles it would take to execute the given **Action** or **Precondition** and used that figure as our cost. The details of this translation are presented in appendix C. Table 5.6 lists the cost of each **Action** and table 5.7 lists the cost of each **Precondition**. We also assumed that all transfers of control from one **Step** to another within Quicksort had negligible cost, and thus assigned a cost of 0 machine cycles to each communication path within Quicksort. This is equivalent to assuming that the entire program operates in a strictly sequential manner, or that branches can be undertaken with no cost. The determination of this communication cost is highly dependent on the way in which an ABL interpreter for 6809 assembly language is built.

Using the mathematical analysis of Quicksort (WIRTH76), we were able to determine the probabilities of the various **Alternatives** in the ABL **Abstract_program**, independent of the number of items, n, being sorted. Table

TABLE 5.6

COSTS OF ACTIONS IN PROGRAM QUICKSORT

Action	Cost (machine cycles)
A 1	11
A 2	18
A 3	19
A 4	27
A 5	28
A 6	6
A 7	10
A 8	10
A 9	34
A10	6
A11	6
A12	19
A13	23
A14	19
A15	6
A16	27
A17	28
A18	10
A19	27
A20	28
A21	10

TABLE 5.7

COSTS OF PRECONDITIONS IN PROGRAM QUICKSORT

Precondition	Cost (machine cycles)
P 1	54
P 2	54
P 3	29
P 4	29
P 5	29
P 6	29
P 7	27
P 8	29
P 9	42

5.8 lists these probabilities, both for the general case and for specific values of n. We were then able to develop a cost for the execution of Quicksort applied to different sizes of data sets. These costs are compared to the theoretical values in table 5.9 and figure 5.7.

5.6 Problem decomposition

Having established a cost for sorting various numbers of items on one processor, it became of interest to examine how we might decrease the cost by using a network of 6809s configured as a parallel processor. (We should note that such a configuration is possible using CUENET (GROSS82).)

The general strategy we decided upon was simple: Have one processor (the "master") divide an array of items into two sub-arrays, send each sub-array to a different processor (a "slave") to be sorted using Quicksort, and then when both sub-arrays are sorted, merge them together into one array. The ABL **System** SPLIT & MERGE (figure 5.8) is a representation of the program that would run on the master processor.

The costs of **Actions** in SPLIT & MERGE are presented in table 5.10. Of note are the costs for A7 and A8. Since Quicksort is being applied to half the number of items in each case, the corresponding values from table 5.9 are used. The costs of the **Preconditions** in SPLIT & MERGE are

TABLE 5.8

PROBABILITIES OF ALTERNATIVES IN PROGRAM QUICKSORT

Alternative	Probability (independent of n)*	Probability (n = 16000)	Probability (n = 8000)	Probability (n = 4000)	Probability (n = 2000)
1.1	1.00000	1.00000	1.00000	1.00000	1.00000
2.1	0.50000	0.50000	0.50000	0.50000	0.50000
2.2	0.50000	0.50000	0.50000	0.50000	0.50000
3.1	0.50000	0.50000	0.50000	0.50000	0.50000
3.2	0.50000	0.50000	0.50000	0.50000	0.50000
4.1	0.33333	0.33333	0.33333	0.33333	0.33333
4.2	0.66667	0.66667	0.66667	0.66667	0.66667
5.1	2/lg n	0.14320	0.15430	0.16710	0.18238
5.2	((lg n)-2)/lg n	0.85680	0.84570	0.83290	0.81762
6.1	(lg n)/2n	0.00044	0.00081	0.00150	0.00274
6.2	(n-(lg n))/2n	0.49956	0.49919	0.49850	0.49726
6.3	(lg n)/2n	0.00044	0.00081	0.00150	0.00274
6.4	(n-(lg n))/2n	0.49956	0.49919	0.49850	0.49726
7.1	(lg n)+1)/n	0.00094	0.00175	0.00324	0.00598
7.2	(n-1-(lg n))/n	0.99906	0.99825	0.99676	0.99402
8.1	1/((lg n)+1)	0.06682	0.07160	0.07713	0.08357
8.2	(lg n)/((lg n)+1)	0.93318	0.92840	0.92287	0.91643

* We make the following assumptions (WIRTH76): Total number of passes is lg n; the total number of comparisons is n lg n; the total number of exchanges is (n lg n)/6; and the maximum stack size is lg n.

TABLE 5.9

COST OF QUICKSORT

	Value of n			
	16000	8000	4000	2000
Cost ($\times 10^6$)	37.63	17.63	8.20	3.79
n lg n ($\times 10^4$)	22.35	10.37	4.79	2.19

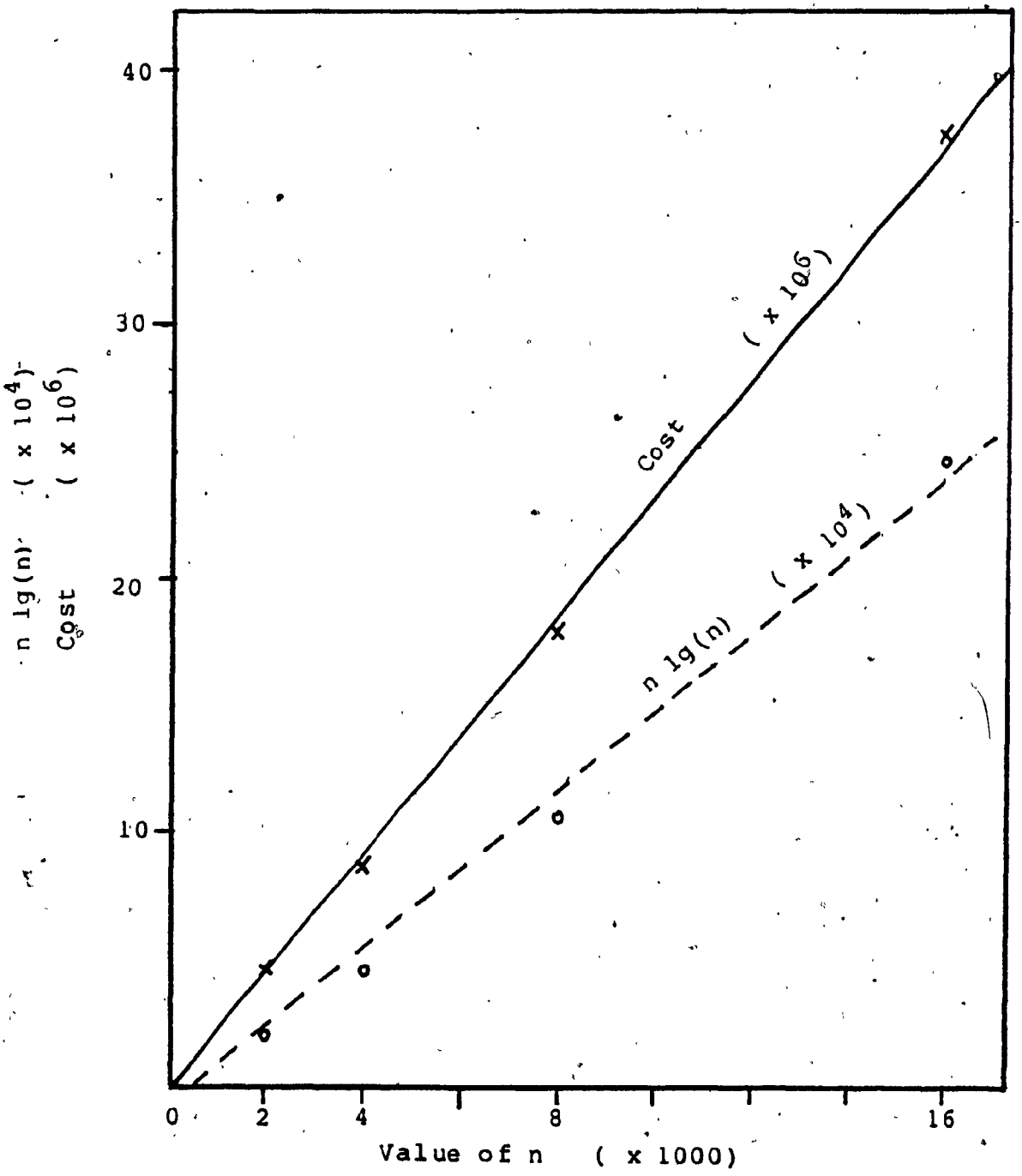


Figure 5.7

Cost of Quicksort

		PROGRAM												MACHINE	
C	1	V	.	.	.	.	.	.	.	.	.	.	V	C	1 Initialize.
C	2	.	V	V	V	.	.	.	.	.	.	.	V	C	2 Split.
C	3	.	.	.	.	V	V	.	.	.	.	.	V	C	3 Sort.
C	4	.	.	.	.	.	.	V	V	V	V	V	V	C	4 Merge.
															PRECONDITIONS
P	1	-	N	Y	-	-	-	-	-	-	-	-	-	P	1 i > m ?
P	2	-	N	N	Y	-	-	-	-	-	-	-	-	P	2 i > 1 ?
P	3	-	-	-	-	-	-	Y	Y	Y	N	N	-	P	3 j <= n ?
P	4	-	-	-	-	-	-	Y	Y	N	Y	N	-	P	4 k <= 1 ?
P	5	-	-	-	-	-	-	Y	N	-	-	-	-	P	5 b[j].key < c[k].key ?
															ACTIONS
A	1	1	.	.	.	.	.	.	.	.	.	.	.	A	1 m := n div 2
A	2	2	.	.	.	.	.	.	.	.	.	.	.	A	2 l := n - m
A	3	3	.	.	1	.	.	.	.	.	.	.	.	A	3 i := 1
A	4	4	1	.	.	.	.	.	.	.	.	.	.	A	4 b[i] := a[i]
A	5	5	2	1	.	.	.	.	.	.	.	.	.	A	5 c[i] := a[i+m]
A	6	6	3	2	.	.	.	3	3	3	3	.	.	A	6 i := i+1
A	7	.	.	.	.	1	.	.	.	.	.	.	.	A	7 QUICKSORT (b,m)
A	8	.	.	.	.	.	1	.	.	.	.	.	.	A	8 QUICKSORT (c,l)
A	9	.	.	.	2	.	.	.	.	.	.	.	.	A	9 j := 1
A	10	.	.	.	3	.	.	.	.	.	.	.	.	A	10 k := 1
A	11	.	.	.	.	.	.	1	.	1	.	.	.	A	11 a[i] := b[j]
A	12	.	.	.	.	.	.	.	1	.	1	.	.	A	12 a[i] := c[k]
A	13	.	.	.	.	.	.	2	.	2	.	.	.	A	13 j := j+1
A	14	.	.	.	.	.	.	.	2	.	2	.	.	A	14 k := k+1
NEXT		2	2	2	3	4	4	4	4	4	4	0	0		

Figure 5.8

SPLIT & MERGE: Matrix Form

TABLE 5.10

COSTS OF ACTIONS IN PROGRAM SPLIT & MERGE

Action	Cost (machine cycles)		
	16000 numbers	8000 numbers	4000 numbers
A 1	14	14	14
A 2	16	16	16
A 3	11	11	11
A 4	28	28	28
A 5	39	39	39
A 6	6	6	6
A 7	1.7628×10^7	8.2023×10^6	3.7923×10^6
A 8	1.7628×10^7	8.2023×10^6	3.7923×10^6
A 9	11	11	11
A10	11	11	11
A11	28	28	28
A12	28	28	28
A13	6	6	6
A14	6	6	6

TABLE 5.11

COSTS OF PRECONDITIONS IN PROGRAM SPLIT & MERGE

Precondition	Cost (machine cycles)
P 1	29
P 2	29
P 3	29
P 4	29
P 5	63

summarized in table 5.11. The detailed development of both costs is presented in appendix D. The probability of **Alternative** selection is presented in table 5.12, both in the general case, and for specific numbers of items to be sorted. Since **Alternatives** 3.1 and 3.2 are executed in parallel, they have the same probability, and for costing purposes they can be considered to be equivalent to one **Alternative** having a cost equal to that of the **Alternative** having the greater cost.

The communication costs for SPLIT & MERGE are given in table 5.13. These costs were arrived at by considering the amount of time it takes to transmit blocks of data between processors on CUENET. This transmission time is not yet reported in the literature, but was made available to us (GROSS82a). The detailed development of these costs is presented in appendix E. In this way we were able to devlop execution and communication costs, as well as total costs for sorting 16000 items, using two or three processors.

An interesting figure of merit in any parallel processing scheme is the "speedup." This is the ratio of the amount of time one processor will take to execute an algorithm to the amount of time several processors working in parallel will take. By altering the **System** SPLIT & MERGE so that instead of calling Quicksort, it called a copy of itself (figure 5.9), we were able to examine the

TABLE 5.12

PROBABILITIES OF ALTERNATIVES IN PROGRAM SPLIT & MERGE

Alternative	Probability (independent of n)	Probability (n = 16000)	Probability (n = 8000)	Probability (n = 4000)
1.1	1.00000	1.00000	1.00000	1.00000
2.1	(n-4)/n	0.99975	0.99950	0.99900
2.2	2/n	0.00013	0.00025	0.00050
2.3	2/n	0.00013	0.00025	0.00050
3.1*	1.00000	1.00000	1.00000	1.00000
3.2*	1.00000	1.00000	1.00000	1.00000
4.1	(n-2)/(2n+2)	0.49991	0.49981	0.49963
4.2	(n-2)/(2n+2)	0.49991	0.49981	0.49963
4.3	1/(n+1)	0.00006	0.00013	0.00025
4.4	1/(n+1)	0.00006	0.00013	0.00025
4.5	1/(n+1)	0.00006	0.00013	0.00025

* Since these alternatives are performed in parallel, for the purposes of costing, they are replaced by one alternative with a cost equal to the maxmimum of each alternative.

TABLE 5.13

COMMUNICATION COSTS IN PROGRAM SPLIT & MERGE

From Step	To Step	Cost (machine cycles)		
		number of items		
		8000	4000	2000
1	2	0.0	0.0	0.0
2	3	1408088	704088	352088
3	4	1408000	704000	352000

TABLE 5.14

COMMUNICATION COSTS AND EXECUTION COSTS FOR PROGRAM SPLIT & MERGE

Number of numbers	Number of processors	Cost (10^6 machine cycles)		
		Execution	Communication	Total
4000	8-15	4.6961	0.7041	5.4002
8000	4-7	10.0125	1.4081	11.4206
	8-15	7.2104	1.4081	8.6185
16000	2-3	21.2748	2.8163	24.0911
	4-7	15.0670	2.8163	17.8833
	8-15	12.2646	2.8163	15.0809

		PROGRAM												MACHINE	
C	1	V	.	.	.	.	.	.	.	.	.	.	V	C	1 Initialize.
C	2	.	V	V	V	.	.	.	.	.	.	.	V	C	2 Split.
C	3	.	.	.	.	V	V	.	.	.	.	.	V	C	3 Sort.
C	4	.	.	.	.	.	.	V	V	V	V	V	V	C	4 Merge.
															PRECONDITIONS
P	1	-	N	Y	-	-	-	-	-	-	-	-	-	P	1 i > m ?
P	2	-	N	N	Y	-	-	-	-	-	-	-	-	P	2 i > 1 ?
P	3	-	-	-	-	-	-	Y	Y	Y	N	N	-	P	3 j <= m ?
P	4	-	-	-	-	-	-	Y	Y	N	Y	N	-	P	4 k <= 1 ?
P	5	-	-	-	-	-	-	Y	N	-	-	-	-	P	5 b[j].key < c[k].key ?
															ACTIONS
A	1	1	.	.	.	.	.	.	.	.	.	.	.	A	1 m := n div 2
A	2	2	.	.	.	.	.	.	.	.	.	.	.	A	2 l := n - m
A	3	3	.	.	1	.	.	.	.	.	.	.	.	A	3 i := 1
A	4	4	1	.	.	.	.	.	.	.	.	.	.	A	4 b[i] := a[i]
A	5	5	2	1	.	.	.	.	.	.	.	.	.	A	5 c[i] := a[i+m]
A	6	6	3	2	.	.	.	3	3	3	3	.	.	A	6 i := i+1
A	7	.	.	.	.	1	.	.	.	.	.	.	.	A	7 SPLIT & MERGE (b,m)
A	8	.	.	.	.	.	1	.	.	.	.	.	.	A	8 SPLIT & MERGE (c,l)
A	9	.	.	.	2	.	.	.	.	.	.	.	.	A	9 j := 1
A	10	.	.	.	3	.	.	.	.	.	.	.	.	A	10 k := 1
A	11	.	.	.	.	.	.	1	.	1	.	.	.	A	11 a[i] := b[j]
A	12	.	.	.	.	.	.	.	1	.	1	.	.	A	12 a[i] := c[k]
A	13	.	.	.	.	.	.	2	.	2	.	.	.	A	13 j := j+1
A	14	.	.	.	.	.	.	.	2	.	2	.	.	A	14 k := k+1
NEXT		2	2	2	3	4	4	4	4	4	4	0	0		

Figure 5.9

SPLIT & MERGE: Recursive Matrix Form

speed-up obtained by using more than three processors. The structure of the parallel processing system is presented in figure 5.10.

In order to perform the simulation, we had to derive costs for applying SPLIT & MERGE to 4000 and 8000 items. The costs we obtained are listed in table 5.14. Table 5.9 gave us the amount of time one processor would take to execute Quicksort, and by using the figures presented in table 5.14, we were able to determine speedup for our simulation. These figures are compared to the theoretical speedup in table 5.15 and figure 5.11. The theoretical values are obtained by using the following formula:

$$1/s = 1/F + (2 - (\lg F)/F)/\lg n,$$

where lg is the logarithm to base 2, s is the speedup, F is the number of processors, and n is the number of items being sorted (GEHRI82).

5.7 The implications for OIS

The above discussions about speedup and parallel processing have relevance to the office environment. In an office a large task may be subdivided into several smaller sub-tasks, each of which can be performed more or less independently. By connecting several office workstations together via a local area network, the processing power of the different workstations can be pooled together and shared. If the bandwidth of the LAN and the granularity of

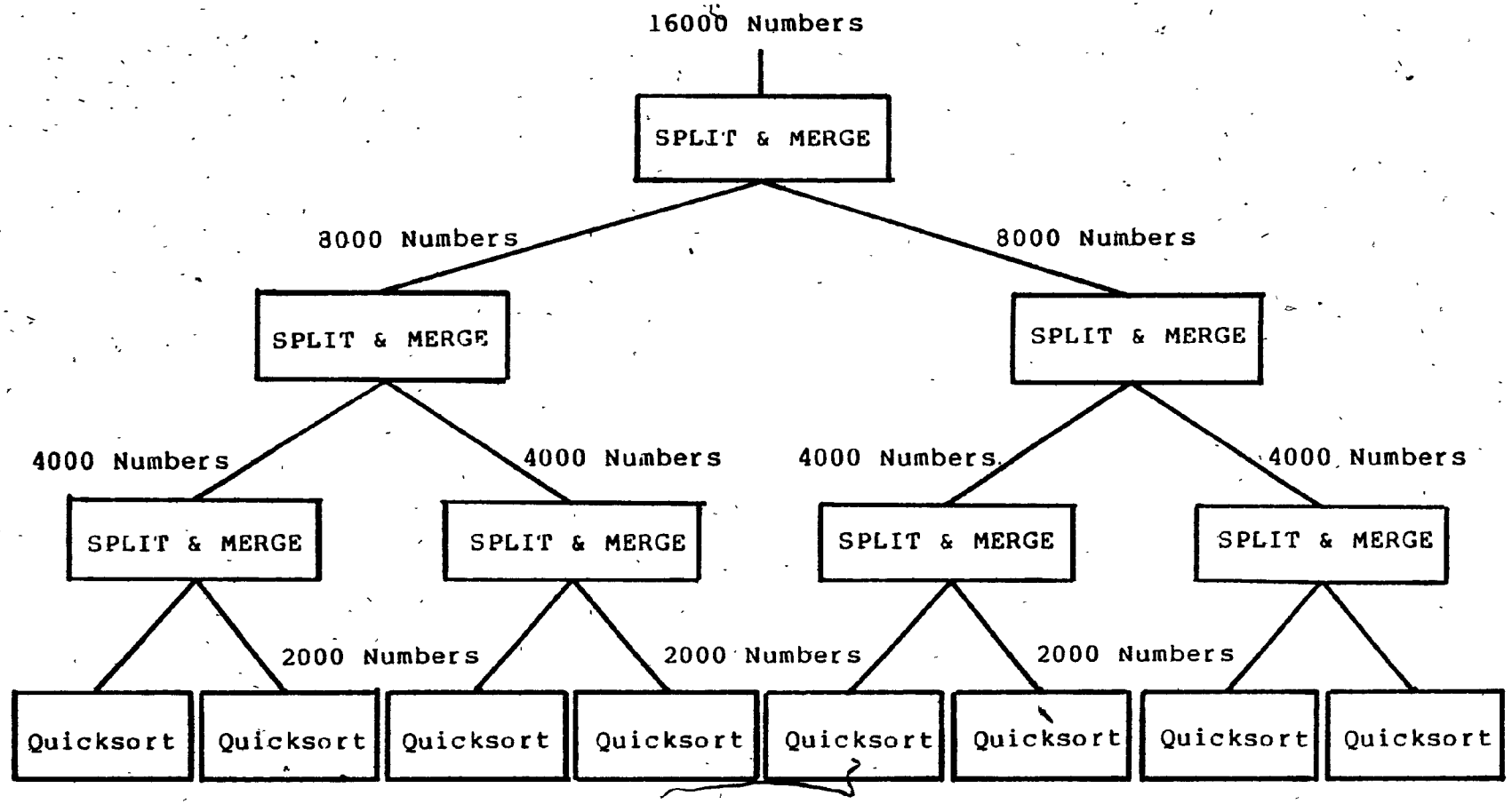


Figure 5.10

Structure of the parallel processor

TABLE 5.15

THEORETICAL AND SIMULATED SPEEDUP FOR QUICKSORT APPLIED TO 16000 ELEMENTS

Number of processors	Theoretical speedup*	Simulated speedup
1	1.000	1.000
2	1.866	1.562
3	2.558	1.562
4	3.109	2.105
5	3.554	2.105
6	3.919	2.105
7	4.221	2.105
8	4.474	2.495

* These values have been obtained by using the following formula: $1/s = 1/F + (2 - (\lg F)/F - 2/F)/\lg n$ (GEHRI82), where s is the speedup, F is the number of processors, and n is the number of elements.

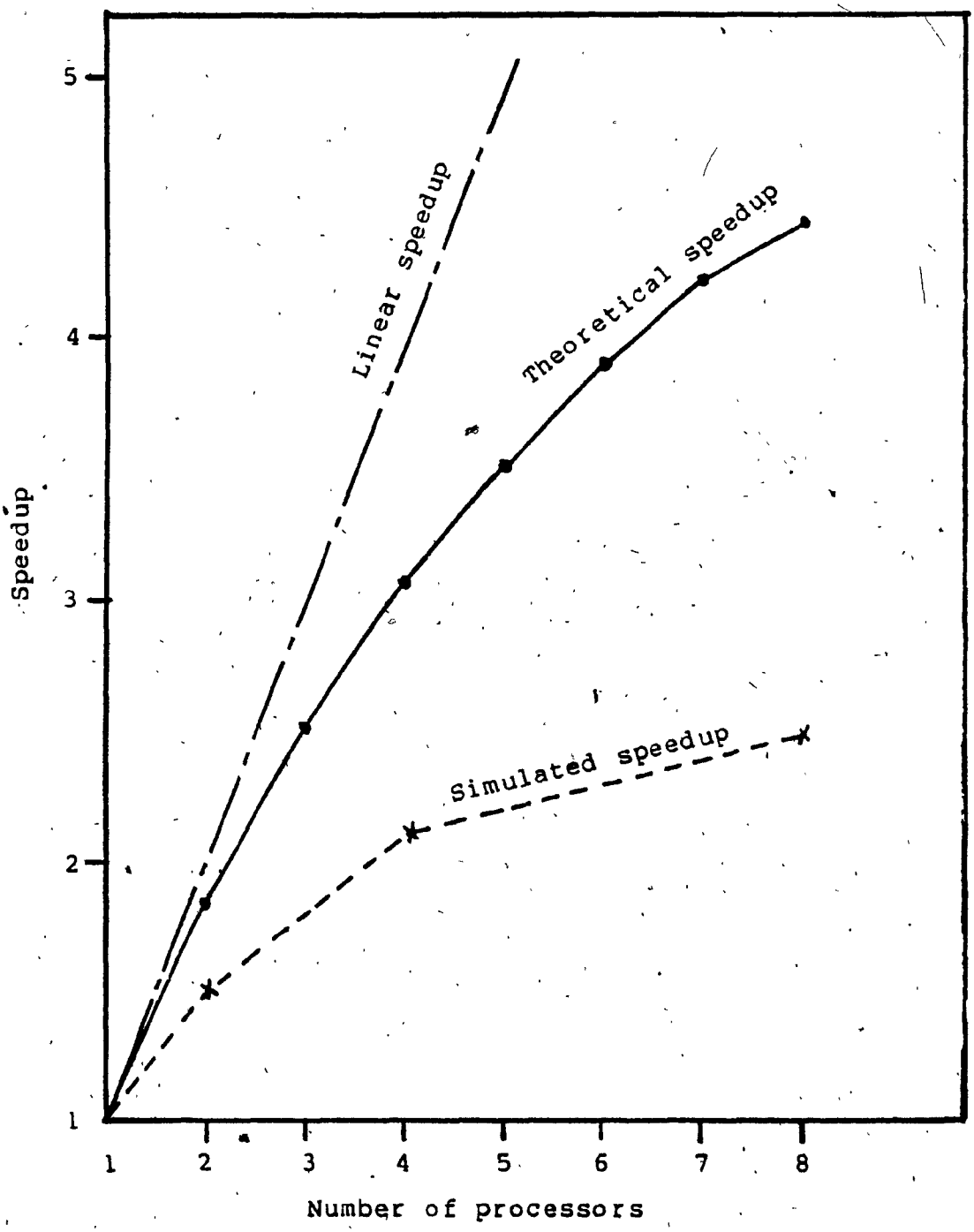


Figure 5.11

Theoretical and simulated speedup

the subdivision of the task are appropriate, then it would even be possible to process these subtasks in parallel using the different workstations on the network. The ABL representation of a large task is well-suited to its subdivision into such small sub-tasks. When performing the subdivision of tasks for the purposes of parallel processing on an LAN, it is important to take into account the cost of communicating between various sub-tasks. Of course, such a cost depends on the bandwidth of the network and the granularity of the subdivision.

Algorithm ASSIGN, described in section 5.3, is one method of allocating sub-tasks to free workstations on the network. This algorithm takes into account the communication cost between sub-tasks, and combines smaller sub-tasks into larger sub-tasks if necessary so as to minimize the overall communication cost to a reasonable level.

CHAPTER VI

CONCLUSIONS AND FURTHER RESEARCH

6.1 Summary and conclusions

After examining various definitions of the office proposed in the literature by different researchers, we have characterized the office as an information processing and generating subsystem of an organization. This view of an office has enabled us in turn to develop the requisite characteristics for a tool used to describe office procedures. This thesis examines the potential of ABL to be used as such a tool.

Although there are several tools reported in the literature that could be used for this purpose, in order to gain some insight into the characteristics of such tools, the following three were considered: SSA (Structured Systems Analysis), BDL (Business Definition Language), and ICNs (Information Control Nets). Our conclusion is that none of the four tools meet the requirements.

For the detailed study of the application of ABL to model an office we have selected an architectural office as

a case study. This office was chosen because it was a real, functioning office to which we had ready access. However, a full description of all the activities of this office is beyond our scope. Hence, we modelled part of the organization, and presented and explained the model. In the course of developing our model, we found that ABL permitted us to organize in a systematic manner the various information we received from the architectural office personnel.

We can also report that ABL is able to represent both sequential and parallel activities. Furthermore, our studies indicate that an architectural office provides a suitable environment in which to test the capabilities of an OIS modelling tool, providing as it does a wide range of both structured and unstructured activities, both parallel and sequential activities, and complex data objects. With regard to unstructured activity, we were able to specify under what conditions such activity would take place, without having to describe the form of the activity itself. The action of querying a employee (A 10 in figure 4.13), for example, may take different forms, and the activity itself may not be structured.

When powerful communication subnetworks are used to interconnect various workstations in an office, it becomes quite possible to process different segments of a large task in parallel. In this regard, we studied the problem

of decomposing a complex activity into smaller sub-tasks. We have found that the use of ABL to describe such a complex activity enables us to develop an algorithm for the decomposition problem. It is not claimed, however, that algorithm ASSIGN is the ultimate solution to this problem.

We demonstrated the use of the algorithm, taking an example from the architectural office. We showed how to derive total execution costs for a coarse-grain example, namely the issuing of expense cheques in the architectural office, and a fine-grain example, Quicksort. Finally, we showed how to use the execution costs and the communication costs to determine the speedup attainable by using several processors operating in parallel. Our studies indicate that ABL can be used to develop meaningful costs for both fine-grain and coarse-grain activities. These costs are necessary inputs to any solution to the decomposition problem. In conclusion, ABL is a useful tool for modelling office activities.

6.2 Future research

The importance of the user interface to any computer-based system cannot be over-emphasized. In developing a model of the architectural office it was not possible for the TAO employees to confirm that the ABL representation of the office activity conformed to their descriptions. This was because the way in which the model

was presented was unfamiliar to them. The amount of effort required to develop and understand a detailed model using the present limited ABL user interface seems out of proportion to the potential benefits that may be accrued. This gap can be reduced significantly by designing a better interface.

However, we should note that even for so well-studied a task as text editing, there is no agreement on what constitutes a better interface (WHITE82). The question of how to design adequate user interfaces for all the various systems present in an office remains open.

Before deciding what kind of interface should be designed, it is important to consider the users of the system. When we consider office workers, there may be many different categories of users. These categories may be represented by separate user profiles. If we can develop these profiles, then we may be able to adapt the interfaces to individual user groups (ROUSE75).

With regard to modelling tools, it is clear that it is difficult to compare them. One possible solution would be to model a real office to a desired level of activity using different tools, and then compare both the process of building the model, and the model itself. The problem that can arise, however, is that the office selected may not test the capabilities of all the tools equally. Thus there would have to be benchmark test cases which would permit a

comparison of tools.

Card et al. (CARD80) present a model of human activity which may have application in the office. The GOMS model suggests that people think in terms of "a set of Goals, a set of Operators, a set of Methods for achieving the goals, and a set of Selection rules for choosing among a goal's competing methods." Although their results indicate the validity of the model for manuscript-editing, it may be possible to look at office activity in terms of goals, operators, methods and selection rules. In particular, the accuracy of predicting user behavior averaged 90% for coarse-grain tasks; that is, tasks in which less detail was being modelled. Office activity can be characterized as a set of tasks "in which variability in behaviour along routine lines is demanded" (CARD80), and thus it would be worthwhile to study the use of the GOMS model for describing office activities. A superficial examination of the GOMS model indicates that there are similarities between it an ABL. A detailed comparison of GOMS and ABL would also be worthwhile.

It remains to be seen whether algorithm ASSIGN is practical in its implementation. Using CUENET, one should be able to determine experimental values for various decompositions and compare the theoretical costs with measured values. Also, it would be of great interest to use CUENET as a test-bed for office workstation design.

Providing, as it does, an environment hospitable to the dynamic reconfiguration of a network, it should be possible to develop a prototypical integrated office information system. Such a system would permit one to experimentally determine characteristics of office workers, develop and test user interfaces, and model and simulate office activities.

REFERENCES

ARTHU82 Arthurs, E. and Stuck, B.W. "Bounding Mean Throughput Rate and Mean Delay in Office Systems," IEEE Transactions on Communications, COM-30, 1 (January 1982), pp. 12-18.

BAILE82 Bailey, Andrew D., Jr., Gerlach, James and McAfee, R. Preston. "An OIS model for internal control evaluation," in Proceedings Supplement, SIGOA Conference on Office Information Systems (Philadelphia, Pennsylvania, June 1982), pp. 13-24.

BAIR79 Bair, James H. "Strategies for the Human use of a Computer-based system," in Proc. NATO ASI on Man/Computer Interaction, B. Shackel (ed). Noordhoff International Publishers, Leiden, Netherlands (1979), pp. 347-377.

BARBE82 Barber, Gerald. "Supporting Organizational Problem Solving with a Workstation," in Proceedings Supplement, SIGOA Conference on Office Information Systems (Philadelphia, Pennsylvania, June 1982), pp. 33-45.

BARCO81 Barcomb, David. Office Automation, Digital Press, Bedford, Massachusetts, 1981.

BERNH82 Bernhard, Robert. "The quandary of office automation," IEEE Spectrum, Vol.19, No.9 (September 1982), pp. 34-39.

CARD80 Card, Stuart K., Moran, Thomas P., and Newell, Allen. "Computer Text-Editing: An Information-Processing Analysis of a Routine Cognitive Skill," Cognitive Psychology, Vol.12, No,

1 (January 1980), pp. 32-74.

CHAPA79 Chapanis, A. "Interactive Human Communication: Some Lessons Learned from Laboratory Experiments," in Proc NATO ASI on Man/Computer Interaction, B. Shackel (ed), Noordhoff Intl. Publ., Leiden, Netherlands, 1979, pp. 65-114.

CHERR82 Cherry, Lorinda. "Computer Aids for Writers," SIGPLAN Notices, June 1982, pp. 62-67.

CODD74 Codd, E.F. "Seven steps to RENDEZVOUS with the Casual User," in Klimbie, J.W. & Koffeman, K.L., Eds, Data Base Management, North-Holland Publ., Amsterdam, 1974, pp. 179-199.

CUFF80 Cuff, Rodney N. "On Casual Users," International Journal of Man-Machine Studies, 12 (1980), pp. 163-187.

DIGIT82 Digital Equipment Corporation. Introduction to Local Area Networks, 1982.

DIJKS75 Dijkstra, Edsger W. "Guarded Commands, Nondeterminacy and Formal Derivation of Programs," Communications of the ACM, Vol.18, No.8 (August 1975), pp. 453-457.

DRISC79 Driscoll, James W. "People and the Automated Office," Datamation, 25, 12, (November 1979), pp. 106-112.

EASON79 Eason, K.D. "Man-Computer Communication in Public and Private Computing," in Man/computer communications, Infotech State of the Art Report, Vol.2, Infotech International, Maidenhead, England, 1979.

EDDY82 Eddy, Diane. Personal communication.

ELLIS79 Ellis, Clarence A. "Information Control Nets: A Mathematical Model of Office Information Flow," 1979 Conference on Simulation, ACM Proc. Conf. Simulation, Modeling and Measurement of Computer Systems, Aug. 1979, pp. 225-240.

ELLIS82 Ellis, Clarence A. and Bernal, Marc. "OfficeTalk-D: An Experimental Office Information System," Proceedings SIGOA Conference on Office Information Systems, June 1982, pp. 131-140.

FANCO76 Fancott, T. and Jaworski, W.M. "Primitive Logic

Constructs Considered Harmful in Structured Programs," Conference Proceedings, Canadian Computer Conference, Session '76 (Montreal, 1976).

FORMA81 Forman, Ira R. "On the Time Overhead of Counters and Traversal Markers," IEEE Fifth International Conference on Software Engineering (March 9-12, 1981), pp. 164-169.

GANAP73 Ganapathy, S. and Rajaraman, V. "Information Theory Applied to the Conversion of Decision Tables to Computer Programs," Communications of the ACM, Vol.16, No.9 (September 1973).

GEHRI82 Gehringer, Edward F., Jones, Anita K., and Segall, Zary Z. "The Cm* Testbed," Computer, Vol.15, No.10 (October 1982), pp. 40-53.

GIULI82 Giuliano, Vincent E. "The Mechanization of Office Work," Scientific American 247, 3(September 1982), pp. 149-164.

GORGE81 Gorges, P., Zanetti, C., Conrath, D., Marcus, M. & Khoury, E. "Method kayak pour l'insertion des outils services burotiques; presentation de la methode d'enquete d'etat actuel," Bulletin de Liason de la Recherche en Informatique et Automatique, no.70, 1981, pp. 16-18.

GROSS82 Grossner, Clifford. "The Design and Implementation of CUENET: A Reconfigurable Network of Loosely Coupled Microcomputers," M. Comp. Sci. Thesis, Concordia University, September 1982.

GROSS82a Grossner, Clifford. Personal communication.

HAMME77 Hammer, M., Howe, W.G., Kruskal, V.J., and Wladawsky, I. "A Very High Level Programming Language for Data Processing Applications," Communications of the ACM, Vol.20, No.11 (November 1977), pp. 832-840.

HAMME80 Hammer, Michael and Kuhn, Jay S. "Design principles of an office specification language," in Proceedings AFIPS 1980 National Computer Conference, AFIPS Press, Arlington, Virginia, pp. 541-547.

HILL79 Hill, D.R. "Using Speech to Communicate with Machines," in Man/computer communications, Infotech State of the Art Report, Vol.2, Infotech International, Maidenhead, England, 1979.

HINTE81 Hinterberger, H. and Jaworski, W.M. "Controlled Program Design by use of the ABL Programming Concept," Angewandte Informatik, 7/81, pp. 302-310.

HORVA82 Horvath, Anthony. "Modeling and Implementation of an Information System for the Control of Truancy in the Quebec Comprehensive High School," M. Comp. Sci. Thesis, Concordia University, June 1982.

ISAAC76 Isaacson, Dean L., and Madsen, Richard W. Markov Chains, Theory and Applications, John Wiley & Sons, New York, 1976.

JAWOR81 Jaworski, W.M. Personal communication.

KENNE74 Kennedy, T.C.S. "The Design of Interactive Procedures for Man-Machine Communication," International Journal of Man-Machine Studies, 6 (1974), pp. 309-334.

KILLI82 Killins, David. "Personal computer may axe secretaries, typewriters, offices," ComputerData, 7, 6 (June 1982), pp. 24-30.

KNUTH73 Knuth, Donald E. The Art of Computer Programming, Volume 1, 2nd. edition, Addison-Wesley Publishing Company, Reading, Massachusetts, 1973.

LEBEN81 Lebensold, J. and Radhakrishnan, T. "Modelling and Implementation of Office Information Systems: A Comparison of Approaches," Computer Science Department Technical Report, Concordia University, 1981. Available from the authors.

LEBEN82 Lebensold, J., Radhakrishnan, T. and Jaworski, W.M. "A Modelling Tool for Office Information Systems," Proceedings SIGOA Conference on Office Information Systems, June 1982, pp. 141-152.

LEE82 Lee, Dik Lun. "A Voice Response System for an Office Information System," Proceedings SIGOA Conference on Office Information Systems, June 1982, pp. 113-121.

LEVEN81 Leventhal, Lance A. 6809 Assembly Language Programming, Osborne/McGraw-Hill, Berkeley, California, 1981.

LINAR82 Linares, Juan. "A Comprehensive Support System for Microcode Generation," M. Comp. Sci. Thesis, Concordia University, August 1982.

LISTE79 Lister, A.M. Fundamentals of Operating Systems, 2nd. edition, The Macmillan Press Ltd., London, 1979.

LOUGH77 Lough, D.E. and Burns, A.D. "An Analysis of Data Base Query Languages," Master's Thesis, Naval Postgraduate School, Monterey, California, 1977.

MALON82 Malone, Thomas W. "How Do People Organize Their Desks? Implications for the Design of Office Information Systems," in Proceedings Supplement, SIGOA Conference on Office Information Systems (Philadelphia, Pennsylvania, June 1982), pp. 25-32.

MARYA81 Maryanski, Fred. "Office Information Systems," Computer, 14, 5 (May 1981), pp. 11-12.

MCDAN70 McDaniel, Herman. Applications of Decision Tables, Brandon/Systems Press, Inc., Princeton, 1970.

MENDE80 Mendes, Kathleen S. "Structured Systems Analysis: A Technique to Define Business Requirements," Sloan Management Review, Summer, 1980.

MORAN81 Moran, Thomas P. "An Applied Psychology of the User," Computing Surveys, 13, 1 (March 1981), pp. 1-11.

MORGA81 Morgan, Alan H. "An Engineering Approach to Problem Analysis," M. Comp. Sci. Thesis, Concordia University, May 1981.

NUTT81 Nutt, Gary J. and Ricci, Paul A. "Quinault: An Office Modelling System," Computer, Vol. 14, No. 5 (May 1981), pp. 41-57.

OLSON82 Olson, Margrethe H., and Lucas, Henry C., Jr. "The Impact of Office Automation on the Organization: Some Implications for Research and Practice," Communications of the ACM, Vol.25, No.11 (November 1982), pp. 838-847.

PETER77 Peterson, James L. "Petri Nets," Computing Surveys, Vol.9, No.3 (September 1977), pp. 223-252.

ROUSE75 Rouse, William B. "Design of Man-Computer Interfaces for On-Line Interactive Systems." Proceedings of the IEEE, Vol.63, No.6 (June 1975), pp. 847-857.

SCHEU81 Scheurer, B., Viarnaud, M.L., Mantoux, G., Berber, D. & Querard, B. "Le buroviseur, poste de travial dans le bureau du futur," Bulletin de Liason de la Recherche en Informatique et Automatique, no.70, 1981. pp.2-8.

SCHUM76 Schumacher, Helmut, and Sevcik, Kenneth C. "The Synthetic Approach to Decision Table Conversion," Communications of the ACM, Vol.19, No.6 (June 1976), pp. 343-351.

SEKEL82 Sekely, George F. "Preparing Your Staff For The Office Of The Future," CIPS Review, 5, 6 (March/April 1982), p. 25.

SHOCH82 Shoch, John F., Dalal, Yogen K., Redell, David D. and Crane, Ronald C. "Evolution of the Ethernet Local Computer Network," Computer, Vol. 15, No. 8 (August 1982), pp. 10-27.

SHWAY74 Shwayder, Keith. "Extending the Information Theory Approach to Converting Limited-Entry Decision Tables to Computer Programs," Communications of the ACM, Vol.17, No.9 (September 1974), pp. 532-537.

STERL74 Sterling, Theodor D. "Guidelines for Humanizing Computerized Information Systems: A Report from Stanley House," Communications of the ACM, 17, 11 (November 1974), pp. 609-613.

STEWA76 Stewart, T.F.M. "Displays and the Software Interface," Applied Ergonomics, 7, (1976) pp.137-146.

TSICH80 Tsichritzis, D. "OFS: An Integrated Form Management System," in CSRG-111, Computer Systems Research Group (1980), University of Toronto. Also in Proceedings of the Sixth International Conference on Very Large Data Bases, IEEE (1980).

TSICH82 Tsichritzis, D. "Form Management," Communications of the ACM, Vol. 25, No. 2 (July 1982), pp. 453-478.

UHLIG79 Uhlig, Ronald P., Farber, David J., and Bair, James H. The Office of the Future, North-Holland Publishing Company, Amsterdam, 1979.

ULLMA80 Ullman, Jeffrey D. Principles of Database Systems, Computer Science Press, Inc., Potomac, Maryland, 1980.

WHITE82 Whiteside, J., Archer, N., Wixon, D., and Good, M. "How Do People Really Use Text Editors?", Proceedings SIGOA Conference on Office Information Systems, June 1982, pp. 29-40.

WIRTH76 Wirth, Nicklaus. Algorithms + Data Structures = Programs, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

ZLOOF81 Zloof, Moshe M. "QBE/OBE: A language for Office and Business Automation," Computer, Vol. 14, No. 5 (May 1981), pp. 13-22.

ZLOOF82 Zloof, M.M. "Office-by-Example: A business language that unifies data and word processing and electronic mail," IBM Systems Journal, Vol. 21, No. 3 (1982), pp. 271-304.

APPENDIX A

BNF DESCRIPTION OF ABL

In order to describe ABL succinctly we shall use a modified form of Backus-Naur Form. In the following description, "<Name>" represents an ABL construct, "::=" means "is defined as" and "|" means "or". The construction "(<Name1>, <Name2>)" which appears in the definition of <Step> is used to indicate a selection of one or more of <Name1> and <Name2> without any regard to whether that selection be made in a sequential manner or in a parallel fashion.

<System> ::= <Abstract machine> <Flow> | <Flow> <Abstract machine>

<Abstract machine> ::= <Precondition set> <Abstract machine> | <Action set> <Abstract machine> | <Postcondition set> <Abstract machine> | <Data set> <Abstract machine> | EMPTY

<Flow> ::= <Abstract program> <Flow> | <Data flow> <Flow> | EMPTY

<Abstract program> ::= <Guard> <Data modification> <Guard>
<Next step> <Abstract program> | EMPTY

<Data flow> ::= <Input data set> <Data modifier> <Output
data set> | EMPTY

<Guard> ::= <Alternative selection> | EMPTY

<Data modification> ::= <Data modifier> <Data modification>
| <Data modifier>

<Next step> ::= <Step> | <Exception step> | EMPTY

<Alternative selection> ::= <Evaluate> <Step>

<Data modifier> ::= <Action> | <Alternative> | <Step> |
<System>

<Exception step> ::= <Step>

<Step> ::= (<Alternative>, <Step>) | EMPTY

<Alternative> ::= <Input data set> <Precondition set>
<Action set> <Output data set> <Postcondition
set>

<Action set> ::= <Action> <Action set> | EMPTY

<Evaluate> ::= Implementation dependent selection according
to an evaluation of <Precondition set> or
<Postcondition set>.

<Precondition set> ::= <Precondition> <Precondition set> |
EMPTY

<Postcondition set> ::= <Postcondition> <Postcondition set>
| EMPTY

<Input data set> ::= <Data set> | EMPTY

<Output data set> ::= <Data set> | EMPTY

<Data set> ::= <Data object> <Data set> | EMPTY

<Precondition> ::= <Description> | <System>

<Postcondition> ::= <Description> | <System>

<Action> ::= <Description> | <System>

<Data object> ::= <Description> | <System>

<Description> ::= <Host language code> <Description> |
<Narrative> <Description> | EMPTY

<Host language code> ::= Set of instructions executable on a hardware machine.

<Narrative> ::= Set of statements in a natural language.

APPENDIX B

FORMS IN THE ARCHITECURAL OFFICE

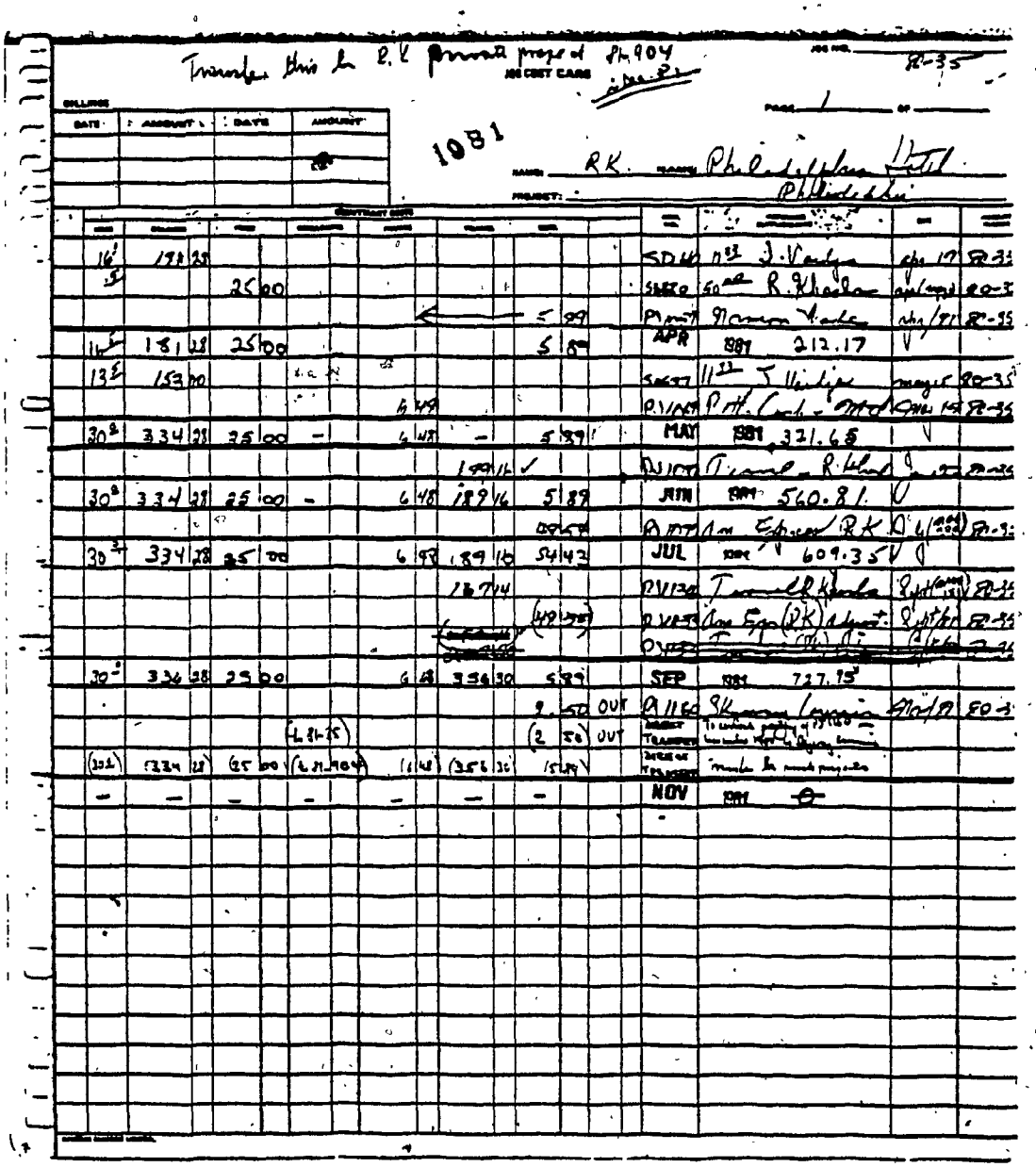


Figure B.1
Job cost card

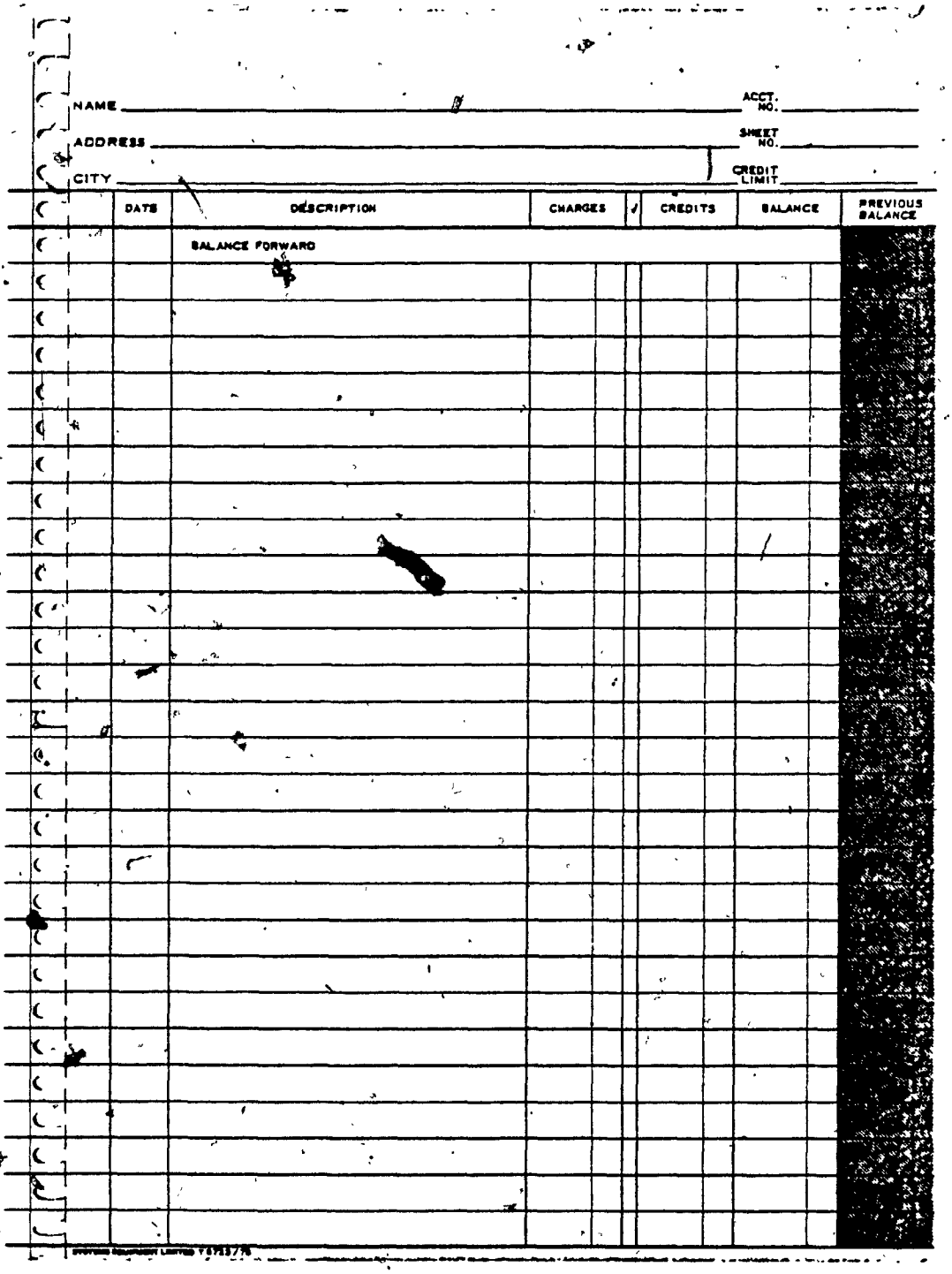


Figure B.2
Client card

BI-WEEKLY TIME AND DISTRIBUTION REPORT

Employee Name (print)		1st PERIOD WEEK ENDING.....							2nd PERIOD WEEK ENDING.....									
	Description	Job No.	Hours worked per day							Hours worked per day							Total Hours	Rate / Amount
			S	S	M	T	W	T	F	S	S	M	T	W	T	F		
1																		
2																		
3																		
4																		
5																		
6																		
7																		
8																		
9																		
10																		
	Total Hours Worked																	

Employees Signature

Supervisor

Regular

Overtime

To be submitted to the Accounting Department NOT LATER THAN FRIDAY of the 2nd Period.

QPP	UIC	MED	F.I.T.	P.I.T.	B.C.	LIFE	LTD	MISC.	NET PAY

Figure B.3
Employee time sheet

PARTNERS TIME SHEET

NAME: _____

WEEK ENDING: FRIDAY _____

JOB NO.	PROJECT NAME	SAT.	SUN.	MON.	TUE.	WED.	THURS.	FRI.	TOTAL

Figure B.4
Partners time sheet

MONTHLY AUTOMOBILE EXPENSE REPORT

NAME: _____ DATE: _____

Date	Mileage	Amount	Job No.	Gen.
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
TOTAL				

OTHER EXPENSES

	Amount
Repairs: _____	$
Parking: _____	$
Insurance: _____	$
Misc. Specify _____	$
Misc. Specify _____	$
Misc. Specify _____	$
Misc. Specify _____	$
Monthly Allowance: _____	$
TOTAL	$
Non-Recoverable Gas: _____	
Auto Expense Acct.No. _____	$
Job Expense Job No. _____	$
Job Expense Job No. _____	$
Job Expense Job No. _____	$
Job Expense Job No. _____	$
Posted _____ Ref. _____	$

Mileage end of Month: _____

Mileage Beginning of Month: _____

Remarks: _____

Signature: _____

Approved: _____

Figure B.5
Automobile expense report

GENERAL EXPENSE REPORT

NAME _____ DATE: _____

DATE	DETAILS	Job Number	Recoverable	Non Recoverable

TOTALS	$	$
TOTAL EXPENSES		$

ACCOUNTING DEPARTMENT

Charge Job/Acct.No. _____ $ _____
Charge Job/Acct.No. _____ $ _____
Charge Job/Acct.No. _____ $ _____
Charge Job/Acct.No. _____ $ _____
Charge Job/Acct.No. _____ $ _____

TOTAL $ _____

Posted _____ Ref. _____

Signature

Approved

Figure B.6
General expense report

TRAVEL EXPENSE REPORT

NAME:_____ Date:_____ Job No.:_____

Trip to:_____ Job Name:_____

Purpose of Trip:_____

TRANSPORTATION: via	Charged or Credit Card NAME	AMOUNT	CASH AMOUNT
Date_____ from_____ to_____			
Date_____ from_____ to_____			
TAXIS			
Date_____ from_____ to_____			
Date_____ from_____ to_____			
Date_____ from_____ to_____			
Date_____ from_____ to_____			
CAR RENTAL			
Date_____ from_____ to_____			
OWN CAR			
Mileage_____ miles @ .22/mile $_____			
Tolls_____ Parking_____			
Misc. (specify)_____			
HOTEL: Name_____			
No. of days_____			
Meals: Date:_____			
Date:_____			
Date:_____			
ENTERTAINMENT:			
Date:_____ Guest:_____ N.R.___ R.___			
Date:_____ Guest:_____ N.R.___ R.___			
Date:_____ Guest:_____ N.R.___ R.___			
ADVANCE $_____	TOTALS	$_____	$_____
CASH EXPENSE $_____			
Bal. due me $_____ Rec'd_____			
Bal. due $_____ Rec'd_____			
Signature_____			
Approved_____			

PROPER RECEIPTS MUST BE ATTACHED

* If not enough space list on reverse side and carry total forward.

ACCOUNTING DEPARTMENT

Charge Job/Acct.No._____ $_____

Charge Job/Acct.No._____ $_____

Charge Job/Acct.No._____ $_____

Charge Job/Acct.No._____ $_____

Accts. Pay. - Travel $_____

Posted_____ Ref._____

Figure B.7
Travel expense report

APPENDIX C

TRANSLATIONS FOR QUICKSORT

In order to translate Pascal statements into Motorola 6809 assembly language, we have to make some assumptions about how the data is stored, and how some aspects of an implementation of an ABL **System** would be done. (We should note here, that none of the **Systems** have been implemented, and consequently the timings presented here are theoretical and not experimentally derived.)

We assume that all variables are in available memory (i.e. not disk resident), and are 16 bits long (two bytes). The variable "stack" is a series of contiguous memory locations, each one containing a pointer to a pair of memory locations which contain "stack.l" and "stack.r". The variable "a" is also a series of contiguous memory locations, each one containing a pointer to a location which contains the "key". We also assume that the elements that are being sorted are 16 bit integers.

We also need three locations that are used to determine how the **System** is executed. ALTMAP is a 16 bit

variable, each bit corresponding to a **Precondition.** When a bit is set (= 1) in ALTMAP, the corresponding **Precondition** is considered to be TRUE, otherwise it is FALSE. NXTCLS is a 16 bit variable in which only one bit is set at a time. The position of that bit indicates the **Next_step** for the **Alternative** being executed. FLAG is another 16 bit variable, the position of each bit set corresponding to the number of the Flag (used in algorithm MERGE) that is TRUE.

We call "$Flag_k$ = TRUE?" a system **Precondition** and "$Flag_k$ = TRUE" and "$Flag_k$ = FALSE" system **Actions.** Thus the system **Precondition** can be implemented as follows:

```
        LDD     FLAG
        STB     F1
        LDD     ALTMAP
        ANDA    FLAG
        ANDB    F1
        STD     ALTMAP
```

for a cost of 27 machine cycles, and a system **Action** that takes care of both cases can be implemented by:

```
        LDD     NXTCLS
        STB     N1
        LDD     FLAG
        ORA     NXTCLS
        ORB     N1
        STD     FLAG
```

which costs 23 cycles. Since this sequence does both, we assigned a cost of 11.5 cycles to each system **Action** that is included.

The equivalent code for each **Action** follows:

```
A1:     s := 1
        CLRA
        CLRB
        INCB
        STD     SS      11 cycles
```

```
A2:     stack[l].l := l
                LDX     STACK
                CLRA
                INCB
                STD     [,X]    18 cycles

A3:     stack[l].r := n
                LDX     STACK
                LDU     N
                STU     [2,X]   19 cycles

A4:     l := stack[s].l
                LDX     STACK
                LDD     SS,X
                LDX     [,D]
                STX     L       27 cycles

A5:     r := stack[s].r
                LDX     STACK
                LDD     SS,X
                LDX     [2,D]
                STX     R       28 cycles

A6:     s := s - 1
                DEC     SS       6 cycles

A7:     i := l
                LDX     L
                STX     I       10 cycles

A8:     j := r
                LDX     R
                STX     J       10 cycles

A9:     x := a[(l+r) div 2]
                LDD     L
                ADDD    R
                ASRA
                RORB
                LDX     AA
                LDX     D,X
                STX     X       34 cycles

A10:    i := i + 1
                INC     I        6 cycles

A11:    j := j - 1
                DEC     J        6 cycles

A12:    w := a[i]
                LDX     AA
                LDX     I,X
```

```
                STX     W       19 cycles

A13:    a[i] := a[j]
                LDX     AA
                LDD     J,X
                STD     I,X     23 cycles

A14:    a[j] := w
                LDX     AA
                LDD     W
                STD     J,X     19 cycles

A15:    s := s + 1
                INC     SS       6 cycles

A16:    stack[s].l := i
                LDX     STACK
                LDD     SS,X
                LDU     I
                STU     [,U]    27 cycles

A17:    stack[s].r := r
                LDX     STACK
                LDD     SS,X
                LDU     R
                STU     [2,U]   28 cycles

A18:    r := j
                LDX     J
                STX     R       10 cycles

A19:    stack[s].l := l
                LDX     STACK
                LDD     SS,X
                LDU     L
                STU     [,U]    27 cycles

A20:    stack[s].r := j
                LDX     STACK
                LDD     SS,X
                LDU     J
                STU     [2,U]   28 cycles

A21:    l := i
                LDX     I
                STX     L       10 cycles
```

The equivalent code for each **Precondition** is as follows:

```
P1:     a[i].key < x.key
                LDX     AA
```

```
                LDU     I,X
                LDX     [,U]
                LDU     X
                CMPX    [,U]
                BGE     next statement
                LDD     ALTMAP
                ORB      %00000001
                STD     ALTMAP
                BRA     next statement          54 cycles

P2:     x.key < a[j].key
                LDU     X
                LDX     [,U]
                LDD     AA
                LDU     J,D
                CMPX    [,U]
                BGE     next statement
                LDD     ALTMAP
                ORB      %00000010
                STD     ALTMAP
                BRA     next statement          54 cycles

P3:     i <= j
                LDX     I
                CMPX    J
                BGT     next statement
                LDD     ALTMAP
                ORB      %00000100
                STD     ALTMAP
                BRA     next statement          29 cycles

P4:     i > j
                LDX     I
                CMPX    J
                LE      next statement
                LDD     ALTMAP
                ORB      %00001000
                STD     ALTMAP
                BRA     next statement          29 cycles

P5:     i < r
                LDX     I
                CMPX    R
                BGE     next statement
                LDD     ALTMAP
                ORB      %00010000
                STD     ALTMAP
                BRA     next statement          29 cycles

P6:     l >= r
                LDX     L
                CMPX    R
```

```
                BGT     next statement
                LDD     ALTMAP
                ORB      %00100000
                STD     ALTMAP
                BRA     next statement          29 cycles

P7:   s = 0
                LDX     SS
                CMPX     0
                BNE     next statement
                LDD     ALTMAP
                ORB      %01000000
                STD     ALTMAP
                BRA     next statement          27 cycles

P8:    1 < j
                LDX     L
                CMPX    J
                BGE     next statement
                LDD     ALTMAP
                ORB      %10000000
                STD     ALTMAP
                BRA     next statement          29 cycles

P9:    j-1 < r-i
                LDD     J
                SUBD    L
                ADDD    I
                CMPD    R
                BGE     next statement
                LDD     ALTMAP
                ORA      %00000001
                STD     ALTMAP
                BRA     next statement          42 cycles
```

```
                LDD     ALTMAP
                ORB      %00000001
                STD     ALTMAP
                BRA     next statement          29 cycles

P2:     i > l
                LDX     I
                CMPX    L
                BLE     next statement
                LDD     ALTMAP
                ORB      %00000010
                STD     ALTMAP
                BRA     next statement          29 cycles

P3:     j <= m
                LDX     J
                CMPX    M
                BGT     next statement
                LDD     ALTMAP
                ORB      %00000100
                STD     ALTMAP
                BRA     next statement          29 cycles

P4:     k <= l
                LDX     K
                CMPX    L
                BGT     next statement
                LDD     ALTMAP
                ORB      %00001000
                STD     ALTMAP
                BRA     next statement          29 cycles

P5:     b[j].key < c[k].key
                LDX     C
                LDX     K,X
                LDX     [,X]
                LDU     BB
                LDU     J,U
                CMPX    [,U]
                BLE     next statement
                LDD     ALTMAP
                ORB      %00010000
                STD     ALTMAP
                BRA     next statement          63 cycles
```

APPENDIX E

COMMUNICATION COSTS FOR SPLIT & MERGE

In order to determine the communication costs between **Steps** in SPLIT & MERGE, it was necessary to determine the number of 6809 machine cycles it would take to communicate between two processors connected by CUENET. Based on a message buffer size of 256 8-bit bytes, it was determined that it required 22 machine cycles per byte, on the average, to prepare the message to be sent, to send the message, and to receive and decode the message (GROSS82a).

Since the only communication costs we are interested in are those involving sending information to other processors, we can assume that the internal communication costs are 0. That is, the cost of communicating from **Step** 1 to **Step** 2 is 0.

In communicating from **Step** 2 to **Step** 3, we have to deliver the following variables: b, c, m, and l. Each element in the arrays b and c requires 4 bytes; 2 for an address, and 2 for a value (see appendix C for a discussion of how the variables are represented). The other variables

each require 2 bytes. Thus the total number of machine cycles required to deliver the variables to **Step** 3 is given by 4 x 22 x (number of items being delivered + 1).

Similarly, only the arrays b anc c are delivered from **Step** 3 to **Step** 4, and thus the number of machine cycles is given by 4 x 22 x (number of items being delivered). These results are summarized in table 5.13.

APPENDIX F

ABL REPRESENTATIONS OF ASSIGN & MERGE

	PROGRAM												MACHINE	
C	1	V	.	.	.	.	.	.	.	.	.	V	C	1 Initialize.
C	2	.	V	V	.	.	.	.	.	.	.	V	C	2 Communication costs.
C	3	.	.	.	V	V	.	.	.	.	.	V	C	3 Check for termination.
C	4	.	.	.	.	.	V	V	.	.	.	V	C	4 Merge <Step>s.
C	5	.	.	.	.	.	.	.	V	V	V	V	C	5 Adjust costs.
														PRECONDITIONS
P	1	-	N	Y	-	-	-	-	-	-	-	-	P	1 <Next step> of a <Step> is itself?
P	2	-	-	-	Y	-	-	-	-	-	-	-	P	2 Number of <Step>s <= number of available processors?
P	3	-	-	-	Y	N	-	-	-	-	-	-	P	3 Total execution cost/Total cost > desired threshold?
P	4	-	-	-	-	-	Y	N	-	-	-	-	P	4 More than one pair of <Step>s?
P	5	-	-	-	-	-	-	-	Y	Y	N	N	P	5 From <Step> is merged <Step>?
P	6	-	-	-	-	-	-	-	Y	N	Y	N	P	6 <Next step> is merged <Step>?
														ACTIONS
A	1	1	.	.	.	.	.	.	.	.	.	.	A	1 Translate problem into ABL.
A	2	2	.	.	.	.	.	.	.	.	.	.	A	2 Determine probabilities for each <Alternative>.
A	3	3	.	.	.	.	.	.	.	.	.	.	A	3 Determine cost of each <Action>.
A	4	4	.	.	.	.	.	.	.	.	.	.	A	4 Determine cost of each <Precondition>.
A	5	5	.	.	.	.	.	.	.	.	.	.	A	5 Determine communication costs.
A	6	6	.	.	.	.	.	5	5	5	4	.	A	6 Cost of <Alternative> := sum of costs of <Action>s.
A	7	7	.	.	.	.	.	6	6	6	5	.	A	7 Cost of <Step> := sum of <Precondition> costs + sum of <Alternative> cost times <Alternative> probability.
A	8	8	.	.	.	.	.	7	7	7	6	.	A	8 Matrix B has entries = branching probabilities between <Step>s.
A	9	9	.	.	.	.	.	8	8	8	7	.	A	9 M := reordering of B.
A	10	10	.	.	.	.	.	9	9	9	8	.	A	10 Create U := inverse of (I-M). Entries in first row are average number of times a <Step> is executed.
A	11	11	.	.	.	.	.	10	10	10	9	.	A	11 D := B(ij) * U(1i). D is the average number of times one <Step> follows another.
A	12	.	1	.	.	.	.	.	.	.	.	.	A	12 Total communication cost between two <Step>s = average number of times one <Step> follows another times communication cost.
A	13	.	.	1	.	.	.	.	.	.	.	.	A	13 Total communication costs := 0.
A	14	.	2	2	.	.	.	.	.	.	.	.	A	14 Total communication cost := sum of communication costs.
A	15	.	3	3	.	.	.	.	.	.	.	.	A	15 Total execution cost := sum of cost of <Step> times average number of times <Step> is executed.
A	16	.	4	4	.	.	.	.	.	.	.	.	A	16 Total cost := total execution cost + total communication cost.
A	17	.	.	.	.	1	.	.	.	.	.	.	A	17 Find two <Step>s that have maximum communication cost.
A	18	.	.	.	.	.	1	.	.	.	.	.	A	18 Find two <Step>s that have maximum total execution costs.
A	19	.	.	.	.	.	2	1	.	.	.	.	A	19 Merge two <Step>s into a new <Step>. Call MERGE.
A	20	.	.	.	.	.	3	2	.	.	.	.	A	20 Calculate <Alternative> probabilities in new <Step>.
A	21	.	.	.	.	.	.	.	1	.	.	.	A	21 Communication cost := 0.
A	22	.	.	.	.	.	.	.	1	.	.	.	A	22 Communication cost := sum of communication costs from both merged <Step>s to non-merged <Step>.
A	23	.	.	.	.	.	.	.	.	1	.	.	A	23 Communication cost := sum of communication costs from non-merged <Step> to each merged <Step>.
A	24	.	.	.	.	.	.	.	2	2	2	1	.	A 24 Replace lowest numbered merged <Step> by new <Step>.
A	25	.	.	.	.	.	.	.	3	3	3	2	.	A 25 Adjust probabilities, costs and <Next step>s.
A	26	.	.	.	.	.	.	.	4	4	4	3	.	A 26 Decrease number of <Step>s by 1.
NEXT		2	3	3	0	4	5	5	3	3	3	3	0	

Figure F.1

Algorithm ASSIGN

```
                PROGRAM                    MACHINE

C  1  V  .  .  .  .  .  V  C  1 Initialize.
C  2  .  V  V  .  .  .  V  C  2 <Precondition>s and <Action>s.
C  3  .  .  .  V  V  V  V  C  3 <Next step>s.

                                           PRECONDITIONS
P  1  -  Y  N  -  -  -  -  P  1 <Alternative> from first merged <Step>?
P  2  -  -  -  Y  N  N  -  P  2 <Next step> of <Alternative> is first merged <Step>?
P  3  -  -  -  N  Y  N  -  P  3 <Next step> of <Alternative> is second merged <Step>?

                                           ACTIONS
A  1  1  .  .  .  .  .  .  A  1 Number of <Alternative>s in new <Step>
                                  := sum of the number of <Alternative>s in both merged <Step>s.
A  2  2  .  .  .  .  .  .  A  2 Set of <Precondition>s for new <Step> := union of sets of <Precondition>s for
                                  each merged <Step> and the <Precondition>, "FLAG(k) = TRUE?"
A  3  3  .  .  .  .  .  .  A  3 Value of each <Precondition> for each <Alternative> in the new <Step>
                                  corresponds to its value in the corresponding <Alternative> in the merged <Step>.
A  4  .  1  .  .  .  .  .  A  4 Value of "FLAG(k) = TRUE?" = "Y".
A  5  .  .  1  .  .  .  .  A  5 Value of "FLAG(k) = TRUE?" = "N".
A  6  .  2  2  .  .  .  .  A  6 Sequence of <Action>s in each <Alternative> in new <Step> is the same
                                  as the sequence in the corresponding <Alternative> in the merged <Step>.
A  7  .  .  .  1  .  .  .  A  7 Last <Action> in <Alternative> is "FLAG(k) := TRUE".
A  8  .  .  .  .  1  .  .  A  8 Last <Action> in <Alternative> is "FLAG(k) := FALSE".
A  9  .  .  .  2  2  .  .  A  9 <Next step> of the <Alternative> is the new <Step>.
A 10  .  .  .  .  .  1  .  A 10 <Next step> of the <Alternative> is the same as the <Next step>
                                  of the corresponding <Alternative> in the non-merged <Step>.
A 11  .  .  .  3  3  2  .  A 11 Increase the number of <Action>s by 2.

NEXT  2  3  3  0  0  0  0
```

Figure F.2

Algorithm MERGE