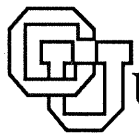
The Modelling and Analysis of Coordination Systems

Clarence A. Ellis and Gary J. Nutt

CU-CS-639-93 January 1993



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

The Modelling and Analysis of Coordination Systems

Clarence A. Ellis
Gary J. Nutt
Department of Computer Science
University of Colorado
Boulder, CO 80309-0430

I. Summary

This paper discusses the framework of ongoing research within the Collaboration Technology Research Group at the University of Colorado. It addresses an area of coordination technology called *workflow systems* --- systems which facilitate the description, modeling, enactment, and coordination of structured work processes by groups of people. These systems assist and mediate communication, interaction, understanding, and synchronization among collaborating processes within organizations. Workflow techniques have existed for decades but, despite progress in many areas, intelligent, industrial strength workflow systems are not well established; the models themselves are too restrictive and the systems lack flexibility, built-in intelligence, distribution, and a solid theoretic foundation. To advance the state of the technology, this research has two objectives briefly described in this paper:
(1) the evolution of a *mathematical family of models* for description and analysis of organizational activity, and
(2) the development of *system architectures* to support the enactment of organizational workflow.

The *models* must enable expression of goals, temporal constraints, dynamic change, and exception handling. The *systems* must enable execution of dynamic, goal-based models; provide coordination and assistance to users at each opportunity; and take advantage of distributed computer systems technology. The method to be employed for this research will include prototyping efforts, methodology development, and studies of work in selected offices. The final outcome of the research will be a solid foundation consisting of a family of models, architectures, and methodologies for the development of future workflow systems.

II. Project Description

2.1. Motivation

Contemporary organizations employ a vast array of computing technology to support their information processing needs. There are many successful computing tools designed as personal information aids (word processors, spreadsheets, etc.) but fewer tools designed for collaborating groups of people. These latter tools are called groupware. *Groupware* is defined as "systems that support groups engaged in a common task or goal, and that provide an interface to a shared environment." [Ellis91a] The potential pitfalls of groupware have been discussed in conferences, tutorials, and journals concerned with the new area of computer supported cooperative work (CSCW) [Grudin88, Poltrock91]. As opposed to much of the previous generation of office information systems, this literature addresses the inherently interdisciplinary nature of groupware. To successfully implement groupware in an organization requires well grounded technological development, with careful attention paid to the social and organizational environment into which the technology is being imbedded.

Many groupware products have recently been introduced to the market [Dyson92]. A few of these products capture knowledge of the organizational activity that they are assisting, but the vast majority do not. For example, a group document editor knows nothing about the organizational purpose of the document being edited. Organizationally aware groupware can potentially lead to significantly more powerful and useful systems. One class of organizationally aware groupware is workflow.

Workflow systems are designed to assist groups of people in carrying out work procedures, and contain organizational knowledge of where work flows in the default case. *Workflow* is defined as "systems that help organizations to specify, execute, monitor, and coordinate the flow of work items within a distributed office environment." [Bull92] The system contains two basic components: the first component is the workflow model, which enables administrators and analysts to define procedures and activities, analyze and simulate them, and assign them to people. Most workflow products have no model, so this component is called the "specification module"; usage of this module is typically completed before the flow of work tasks actually begins. Our research explores the hypothesis that a model of coordination is a useful entity in all phases of groupware use.

The second component is the workflow execution module (the workflow system) consisting of the execution interface seen by end users and the execution environment which assists in coordinating and performing the procedures and activities. It enables the units of work to flow from one user's workstation to another as the steps of a procedure are completed. Some of these steps may be executed in parallel; some executed automatically by the computer system. The execution interface is utilized for all manual steps, and typically presents forms on the electronic desktop of appropriate workers (users.) The user fills in forms with the assistance of the computer system. Various databases and servers may be accessed in a programmed or ad hoc fashion during the processing of any work step. Our research explores the hypothesis that the specification and execution modules need to be tightly interwoven.

The history of workflow application in corporate America has been mixed; more systems have silently died than been successful [Bair81]. It is found that organizations succeed only if people creatively violate, augment, or circumvent the standard office procedures when appropriate. People tend to work through goals rather than through procedures [Li90]. Opportunity exists for a leap forward in productivity, effectiveness, and satisfaction when workflow systems successfully incorporate and utilize knowledge of goals, constraints, and the social and organizational context into which they are embedded. Structured procedural work frequently has unstructured components. The mechanisms to help people do their necessary problem solving and exception handling are not available in today's workflow systems. This project is researching basic workflow issues which must be addressed for this vision to become a reality.

The set of issues and problems are challenging, but approachable. The time seems appropriate for this work because many contemporary organizations employ personal computers, workstations, and networks; also many are expressing a strong interest in workflow [Dyson92]. The Gartner group has predicted that workflow will be one of the primary areas of organizational productivity enhancement in the 1990s, and will mature around the year 2000 if some significant inhibitors can be overcome [Leung92]. Both of the Principal Investigators were involved in workflow research in the 1970s (it was then called office information systems) [Ellis80]. Workflow was unsuccessful at that time largely because the technology was not available, because personal computers in the office were not socially accepted, because vendors were unaware of the requirements and pitfalls of group technology, and for other reasons that we enumerate later in this document.

Although there is a clear challenge and demand, qualified CSCW researchers are not generally working in this area - there is a noticeable lack of workflow research in the CSCW conference proceedings [CSCW92]. Some CSCW research-

ers flatly reject the whole workflow approach as being too restrictive, inhumane, and intractable. We believe that if one takes the approach articulated in this document that workflow specification is simply a means of specifying the base default procedures from which to augment, evolve, and change; and that workflow is neither inhumane nor intractable. It can form the backbone for allowing non-local information to be available to workers at all levels in a readily understandable form to the right persons at the right time.

2.2. Objectives

The general objective of this proposal is to build a solid foundation for the development of future workflow systems. This is in keeping with the CTRG general long term objectives of understanding and creating a foundation for coordination theory and technology in general. This is an interdisciplinary endeavor, and our perspective is a computer science perspective. Computer Science areas such as distributed systems, artificial intelligence, and visual languages have much to contribute provided they are coupled with a deep understanding of the interdisciplinary issues.
We are studying models and systems that explore ideas of enhancement by incorporating concepts of goals and constraints into workflow. People are often hindered in a complex organization if they do not know the goals behind the specific tasks that they are doing.They ask "Why am I filling out this form in this manner?" Likewise a system to assist and augment these solutions must also have some awareness of the goals of the tasks. It must be able to answer the above question. Goals also provide a mechanism for synthesizing pieces of unstructured work within a structured procedure. By introducing new workflow modeling primitives, we provide a mechanism explicitly for exception handling and other activity that is not ordinarily captured by formalism.

Our philosophy is that the model and the system must co-evolve to handle the dynamic change and exception handling in a way which gives assistance (rather than hindrance) to the end user. Modeling should not be restricted to pre-execution, but should be a vehicle for continual system evolution as the organization evolves. In the midst of execution, the model can be useful for what-if scenarios, for question answering, and for making pro-active suggestions. Likewise the system should be available to the model so that pieces of a simulation can be evolved into enactment via incremental environmental relaxation (see later section.)

Distributed systems issues are important within the architecture of the system and the model. We believe that the model should embody a language that can be used by people to discuss aspects of their work, and to instruct their workflow system when they want it to do specific tasks for them. Thus we are exploring issues of distributed scheduling, concurrency control, and visual languages. Within the editor for our model, we are exploring new techniques for abstraction, for distribution, for construction, and for analyses. We expect to leverage off of our past and current experience; to build prototypes to expand our intuition; and to validate our work and prototypes by usage studies.

2.3. Related Work

There are many stages between research innovation and successful products. Our workflow research is primarily concerned with the earlier stages but our concerns are grounded in experiences with the later stages. Both Nutt and Ellis have had many years of product development experience in industry. Workflow product development inherently crosses many disciplines, perspectives, and interdisciplinary issues. Our workflow research is conceived from a computer science perspective, but is informed by the disciplines of information systems, artificial intelligence, organizational design, social/cognitive psychology, and cultural anthropology. The two authors were strongly influenced by related work performed during their years in the interdisciplinary office research group at Xerox PARC in the 1970s. At the University of Colorado, the PIs work and perspective continues to be influenced by Professors Gerhard Fischer [Fischer90] and Clayton Lewis [Freeman92] (AI / Institute for Cognitive Science), and with Professor Ilze Zigurs [Zigurs92] (Information Systems Program within the Business School.) Due to page constraints, many of these important influences are not explicitly documented in the short list of related work which follows.

2.3.1. Others' Research

Prominand (IABG)[Karbe91]. This European ESPRIT research project is closely related to our work since it investigates workflow architecture and design with an emphasis on exception handling. The Prominand system is based upon an "electronic circulation folder" paradigm. It is a system which circulates office tasks consisting of steps to be carried out by persons playing office roles. The system attempted to categorize all of the types of exceptions (e.g. lost information; wrong recipient; etc.), and provide handlers for exceptions (e.g. skip next step; return to sender; etc.). Their results illustrate that it is not feasible to identify and program handlers for all exceptions that will occur in the life of a workflow system a priori. In this sense, our approach to workflow architecture and exception handling is quite different (see section

2.5, research details.)
There has been other work which addressed systems aspects of workflow. Much of this thrust comes from the PIs early work on Officetalk (see subsection 2.3.2.). GMD has implemented several versions of Domino [Kriefelts84], a Petri net based prototype office information system. Usage reports detail numerous problems and reasons for user rejection of the system -- this typifies problems of current workflow. Other workflow efforts include the Xerox "Collaborative Process Model" [Sarin91], and the WooRKS workflow prototype within the ITHACA ESPRIT project [Ader92]. None of these systems consider goal based workflow.

AI based Workflow. Early work on this was performed by Fikes, also at Xerox PARC. His Odyssey system [Fikes81] was a single user inferencing system. He observed that a system which does inferencing and takes actions for the user without informing the user is confusing, even to a single user, especially as complexity increases. Li investigated the use of Shank's MOPS as an AI technique for office systems [Li90]; he used ideas of conceptual specialization to generate office procedures, but has not yet found applicability in realistic office systems. The most similar goal-based work to ours is Croft's work on Polymer; it is a goal based planning system to assist in the performance of multi-agent, loosely structured, underspecified tasks [Croft89]. This work is an excellent starting point for our current research, since it addresses problem solving and domain representation languages to assist in office work. He uses techniques of AI planning, inferencing, and backtracking. Polymer is not a workflow system, and as such, does not explore mathematical workflow models or distributed systems. However, the results of Croft's work will be quite useful in our efforts to incorporate goals into ICN workflow models. AI work related to office systems has been summarized by one of the PIs [Ellis88].
Other Workflow Studies. Some workflow studies have transpired in the Information Systems field and the Organizational Design field within business schools. Examples include Bair's TUMS [Bair91], Woo's SACT [Woo90], Hirshheim's model [Hirshheim85], the Society model [Ho86], Hammer's BDL and OAM [Sirbu84], the OSSAD model [Dumas91], etc. Several office models have emerged from concepts of discrete mathematics. These include Petri net based workflow models [Zisman77, Holt88, Li91], and graph theory based models [Luqi90]. There is also a set of models which have emerged out of the software engineering community. These could be classified as extended flowchart/state machine notations [Harel90], project management models [Kellner91], and process programming models [Osterweil88, Saeki91]. Office models are reviewed and contrasted in several articles including [Ellis80] and[Bracchi84].

The primary contrast of our model (the ICN model) to the above are three fold.

1. ICNs were originally designed specifically to model workflow type situations; other models were not.
2. ICNs are now goal based models; other models are not.
3. ICNs have been shown to be useful for end user understanding, and for formal mathematical analyses.
(We remark that ICNs have formed the basis of several workflow products [Leung92].)

Information Lens. We conclude with a few systems which are quite different from our thrust, but related. Several mail based systems have emerged as fledgling workflow systems. Several of these are based upon the Information Lens [Malone, 1987] and Object Lens [Lai88] research projects by Malone and his group at the MIT Sloan School. The Beyond Mail and Wijit products provide tools for specifying complex routings for mail messages. These can be characterized as semi-structured systems. After an organization is comfortable with the mail tool, individuals can write mail based workflow scripts to gradually automate the model of operation.
Speech Act Theory. One example of a company that started with a mail based tool is Action Technologies. Action Workflow [Dyson92] is based upon Winograd's Speech Act theory [Winograd86]. The theory and the system are built upon a well defined set of "language acts," and a notion of "conversations" which are coherent sequences of language acts with a regular structure. Within this approach, there are many workflows in an organization, each of which constitutes a basic unit of work. In each, a "performer" commits to produce "conditions of satisfaction" defined by a "customer," on or by a specified time. This system is a follow-on to their previous product; The Coordinator was a mail based product that did not have workflow, and which was a very controversial introduction of Speech Act theory. Clearly, the jury is still out concerning this approach. Although we are taking a much different approach, we have many goals in common. Hopefully research in each direction will continue, and will give feedback and perspective to each other and to the advancement of the field of workflow.
Role Interaction Nets (MCC) [Rein93]. RINs are a visual formalism for the design, specification, and enactment of work processes. This research is being worked upon by close colleagues of Ellis at MCC. The RIN formalism, which is based upon Organizational Role Theory [Roos68], describes processes as a collection of organizational role types and interactions among the role types. Although it does not embrace goals, the project has produced several interesting graphical

languages which try to carefully match the end user concepts to the multi-dimensional graphics. This work is seen as a good complement to ours because unlike our work, it emphasizes graphical end user computer interaction environments. This human-computer interface (i.e. workflow GUI) is never-the-less a very important aspect of any workflow system.
Comic (ESPRIT) [Rodden92]. The Comic Project, computer based mechanisms of interaction in co-operative work, is an ongoing ESPRIT basic research project which aims to develop the theories and techniques necessary to support the development of future CSCW systems. This is not a workflow project, but will be investigating some of the same issues of tightly coupled distributed systems. Comic is primarily interested in real-time synchronous groupware.
Commercial Products. FileNet's WorkFlo Business System [Davis91] is one of the oldest and most established of the commercial products. It includes scanning hardware and software to accommodate paper input to workflow, and a large, complex, proprietary language to program "scripts" for your particular application. Other workflow systems are now emerging, including Plexus corporation's Imageflow, Bull Systems Automatique's FlowPath [Bull92] (which employs ICNs as its model base), Workhorse Systems' Workhorse, IBM's WorkFolder Application Facility, Unisys' Event Manager, and FCMC's Staffware [Leung92].

2.3.2. PIs' Prior Research

Ellis and Nutt have had a sustained interest in workflow research for the last two decades, e.g., see [Ellis80] which surveyed and discussed issues of workflow. At Xerox PARC they were involved in office information systems research which led them to address various other aspects of office work. The original ICN model was also developed while they were at PARC [Ellis79a], which led to a family of related modeling studies. More recently, Nutt has conducted systems and evaluation work (the Olympus studies) based on the BPG variant of ICNs, while Ellis independently applied ICN technology to a commercial workflow system (see Bull FlowPath product). Ellis was also very active in defining groupware -- by building experimental prototypes and by helping to form the symposia on Computer Supported Cooperative Work [CSCW92]. We elaborate on these studies and work in the remainder of this subsection.

a) Officetalk Related Work

Officetalk was an experimental office information system developed in the Office Research Group at Xerox PARC [Ellis80]. Officetalk was the first system that we know of that provided a visual electronic desktop metaphor across end users' personal computers. It also provided a set of personal productivity tools for manipulating information, a forms paradigm, and a network environment for sharing information. This system was created, evolved, and used extensively within the research lab, and was also tested in selected sites outside of PARC.

Backtalk. This prototype facility was built to control the network loading on individual Officetalk workstations [Nutt79]. The Backtalk facility enabled one to configure a network of logical nodes to represent job stations in an office; some of the nodes were implemented by a workstation and a human user, while other nodes were a logical workstation with an algorithmic representation of how a human might execute certain classes of work. In the absence of a workflow model, Backtalk provided a facility for adding the algorithmic nodes into the network of workstations with human users, allowing one to experiment with a distributed system. For example, if a network of Officetalk workstations were to be introduced into an office, then each individual person would need to be trained how to do their work in the new automated environment; if all persons were to be trained at one time, then chaos would result, since each user is depending on other users to behave in the manner prescribed by the office procedure designer. Backtalk allowed one to automate all classes of workstations except the one being used to support the trainee; thus that person could learn how to use the system (in the absence of exceptional conditions!) without being dependent on the actions of another trainee.

OfficeTalk - D. The extension to Officetalk that graduated it to workflow was called Officetalk-D [Ellis82]. It added the ICN framework and graphical language to officetalk, enabling the specification and execution of procedures. At this evolutionary stage, the system had primitive knowledge of the organizational structure and procedures. This knowledge was stored in a database, and the user interface paradigm presented information as windows onto a database. (The D stands for database oriented.)

OfficeTalk - P. A further extension to Officetalk was embodied in Officetalk-P [Ellis79b], which used the implementation strategy, and the user metaphor of a system consisting of migrating processes. Thus, a user viewed his windows as intelligent forms which travelled from workstation to workstation to get their mission accomplished as specified by an ICN. Furthermore, the experimental system was constructed as forms processes, workstation processes, and overseer processes. (The P stands for process oriented.)

b) ICN Modeling Research

Quinault [Nutt81]. Quinault was a model specification system to support ICNs. The system provided a visual editor and visual simulation facilities, enabling a user to construct ICNs, then to animate token flow through them to obtain a qualitative understanding of the operation of the target system.

Strawberry [Nutt83]. This system was a generalization of Quinault. It was distributed across workstations and the Ethernet using low level IPC and a tailored message-passing protocol with a novel naming mechanism (essentially referencing objects by class name or instance name). It also distributed the function of the modeling and simulation facilities, and was the forerunner of Olympus. Part of the motivation for deriving a custom IPC protocol was to establish a logically homogeneous computing environment in a potentially heterogeneous physical network; each Strawberry node executed an agent process to interact with other agents, and to otherwise provide a consistent set of high-level facilities at each node.

Office Modeling Studies [Ellis79c] A large series of studies were undertaken at PARC during the phase of invention/evolution of the ICN model. These studies were jointly carried out by members of the Office Research Group, and of the Analysis Research Group. The goal was to evolve to a model which was easily understood by end users in the office, and that was mathematically tractable. Useful analyses and methodologies were explored and evolved, and problem areas were uncovered. Analyses that were found useful included dataflow analysis, throughput analysis, consistency analysis, timing analysis, deadlock analysis, parallelism analysis, and streamlining analysis [Cook80].

c) Olympus

The Olympus modeling system has been used to support a variety of different models, including multi-tiered models (see the section on Prior NSF Awards). Bilogic precedence graphs (BPGs) -- a variant of ICNs -- were the first model supported by Olympus, although subsequently Olympus was used to support various versions of Petri nets and ICNs.

The original Olympus research has wound down, leaving a legacy of tools to support workflow model experimentation, and a spectrum of ideas for supporting multiple sessions, hierarchies, and further distribution of the system in a network environment. As Olympus has served as the infrastructure project for distributed systems modeling work in the past, it will now be used as the infrastructure project for distributed workflow research. It provides a working workflow system with several interesting architectural features, including skeletal support for multiple users, alternative syntactic views of a common model, distribution of graph interpretation facilities, and a mechanism for easily changing underlying models and adding custom analysis capabilities.

The Olympus architecture is designed to meet modeling system requirements in such a manner that the human-machine session is modeless and so that the machine supports multiple users concurrently [Nutt89]. The current architecture is distributed across UNIX processes using the client-server model. The server implements parts of the modeling system that are shared among independent human users, e.g., the storage of a specific model, the state of the model as it is executed, etc. Each client implements a specific aspect of the modeling system, e.g., an editor or a specific analyzer.

The server backend is a natural mechanism to use as the storage for our workflow models and their execution state. Each frontend client implements functionality specific to the individual human user's requirements, e.g., a graph editor with a specific view of the logical model within a specific window system, a deadlock analyzer, etc.

The Olympus client-server protocol uses a transaction mechanism in which each client posts an operation to the server than waits for an acknowledgment from the server. When the server has completed the transaction, it broadcasts the acknowledgment to all clients, thus client use a "snooping protocol" to determine state changes in the server.

McWhirter's graph editor builder project grew from the Olympus work [McWhirter91; McWhirter92]. This system allows a system designer to quickly build a visual frontend to a modeling system by specifying the characteristics of the formal model and of the visual interface.

d) Groupware

During his 6 years at MCC, PI Ellis was head of the groupware research efforts. He was one of the instigators of several workshops, conferences, and journals that are still going very strong today, and that address CSCW issues [CSCW92]. Groupware is categorized into quadrants of same versus different time, and same versus different place [Ellis91a]. Workflow falls in the different time, different place category, but it is now recognized that useful groupware must span multiple quadrants [Baecker93]. Thus, since structured work sometime requires unstructured activities, workflow must be capable of supporting activities such as ad hoc real time group problem solving. At MCC, Ellis was involved in research on all of the above quadrants. Some of the systems that emerged from this work are briefly described here.

Nick [Ellis87]. Electronic meeting room system assisting face to face interaction among meeting participants. Several rooms were constructed with innovative technology including electronic blackboards, portable workstations, and interaction/interface software. These rooms were extensively instrumented and tested. Then formal validation experiments were performed [Ellis89].

Grove [Ellis90]. Real time group editor allowing distributed group discussion and simultaneous editing of a document. Based upon a hypothesis that groups would be much more synergistic if they were allowed to work in a more tightly coupled fashion, this system implemented a sophisticated algorithm allowing simultaneous editing with no locking. Users could work in private, shared, or public windows, seeing multiple views of a document; the system guaranteed consistency, and implemented a group user interface.

gIBIS / rIBIS [Ellis91b]. Issue based hypertext system for the capture of design rationale. This system implemented a graphical version of issue-based information systems (IBIS) that all design conversations can be cast in terms of issues, positions, and arguments. This hypertext system allowed a group to navigate jointly or independently through the space using a convenient graphical interface. The system was initially created for different time, different place, but was extended to other quadrants.

2.4. Results from Prior NSF Support

Technology Support for Collaborative Workgroups (PI: Gary M. Olson, 6/89-5/92). This grant was obtained from the NSF as one of the first round of awards of the solicitation "Special Initiative on Coordination Theory and Collaboration Technology." Professor Gary M. Olson, Department of Psychology, University of Michigan, was the principal investigator. PI Ellis was a participating scientist from MCC (the Microelectronics and Computer Technology Corporation) during the first portion of the grant duration, and acted as an advisor, and a supplier of groupware tools to be evaluated, and groupware implementation expertise. The project examined how a small group of collaborators work together to design software requirements, and what impact - good or bad - the use of groupware might have on this process. Scientists from the University of Michigan led the project. They cooperated with colleagues from the Software Technology Program of MCC and from Arthur Andersen & Co. The specific project activities represented the blend of skills in the building of computer systems, the empirical analysis of human behavior, and the application of theory both from technology and from cognitive science to the design and analysis of technology augmentations of work. The research pursued the following specific activities:

Studies of current practice:
1. Analysis of collaboration from design meetings data previously collected by video at MCC
2. Observational studies of collaboration at Arthur Andersen meetings
Studies of groupware supported collaboration:
3. studies of collaboration (student subjects) using MCC groupware prototypes
4. studies of collaboration (student subjects) using U of Michigan groupware prototypes.
MCC played a significant role during the early phases of the project in both technological and experimental research collaboration and assistance, but was required by MCC management to withdraw when MCC encountered serious financial difficulties. Relevant publications generated by MCC are listed in the bibliography.

Parallel Program Modeling and Evaluation (PI: Gary J. Nutt, 10/88-1/91, Grant No. CCR-880228). This study investigated approaches to assist in the design of parallel programs by studying tools to provide quantitative and qualitative feedback to program designers early in the design cycle. The environment was developed as a functionally distributed system that supported various formal models and independent views of the models. The general organization of the environment is described in [Nutt89a, Nutt90a, Nutt90b]. The environment uses the client-server model to support a specific formal graph model; BPG-Olympus, VISA, and Phred are systems built in the environment to support specific graph models. Each of system instance incorporates facilities to: create and edit visual representations of the formal graph model; store an internal representation of the graph model; interpretively execute the stored model (i.e., interpret the activity among nodes in the graph, and to execute procedures (node interpretations) associated with each node); provide animation facilities in scaled real time; provide detailed reports on the interpreted execution; and provide an interactive console to control the system.

The BPG-Olympus Modeling System supports Bilogic Precedence Graphs (BPGs) [Nutt89b, Nutt89c], providing a gen-

eral flow graph model simulation environment, that supports a group of designers with interactive graphic-based tools to create, maintain, modify, and exercise an interpreted, marked network model.

VISA supports ParaDiGM (Parallel Distributed computation Graph Model) [Demeure89a], a two-tiered model that represents functional aspects of a computation at the first tier (DCPGs) and partitioning and interprocess communication organizations in the second tier (PAMs). ParaDiGM allows one to compare different process architectures of distributed computations in a message-passing environment. It has been used to represent and analyze a variety of computations, including adaptive global optimization [Demeure89b], chaotic relaxation [Demeure90a], the client-server model [Demeure90b], and station message detail recording in a PBX.

Phred programs use graph models to define and analyze a parallel program such that one can statically analyze the program for determinacy [Beguelin90, Beguelin91a, Beguelin91b]. The language employs standardized programming constructs which will admit nondeterminacy, but the system identifies those aspects of the program that are not determinate. Phred takes advantage of the architecture to incorporate an asynchronous critic process to perform dynamic formal graph analysis [Beguelin91b].

2.5. Research Details

A workflow model is an abstraction capturing pertinent aspects of a work environment, and work processes within an organization or workgroup. An extension f the ICN model captures abstractions of goals, constraints, resources, processes, activities, repositories, roles, and actors. We will define this extended ICN model in Subsection 2.5.1. The workflow model relies on the existence of a workflow system, specifically a model specification system (including a model editor) which allows the creation, alteration, and simulation of workflow. This editor should present the model in a form which is understandable to the non-specialist, and easy to manipulate; this subsystem is discussed in Subsection 2.5.2. Besides being simple to understand, a good model should be mathematically tractable. Thus it should be precisely defined, and there should be meaningful analyses that leverage this mathematical notation. These analyses should be invoked and controlled via the model editor. In Subsection 2.5.3, we enumerate some useful analyses which have been performed using ICNs, and we present exception analysis as one of many novel analyses to be further explored in the future.

A *workflow system* is a computer system which helps to execute, coordinate, and monitor the flow of work within an organization or workgroup. Examples of procedures amenable to workflow include an "equipment order procedure" within an engineering company, a "claims processing procedure" within an insurance company, and a "document distribution procedure" in a technical support organization. The workflow system logically consists of a specification subsystem (mentioned above), an end user interface subsystem, and an execution environment.

Workflow systems should fit into an existing organization by providing ready access to personal productivity tools, and to external databases and servers when needed. Some of these accesses can be preprogrammed, but many will be ad hoc. Workflow should also enable a user to delegate work to the system when that is the appropriate action, but to use the system as a productivity tool in other cases. Thus the user interface subsystem must present an interface to these accesses as well as organize and present the current work to be done, or in progress, for an individual user at that user's workstation.

Office work has been characterized as either *structured* or *unstructured* work [Dumas91]. Structured work has the property that its underlying nature can be captured by an algorithm and then specified in a program specification. Unstructured work cannot generally be characterized by an algorithm. We base our research on the hypothesis that the model and the execution environment should support both structured and unstructured work; in Subsection 2.5.4, we discuss how our execution environment can handle both. The execution environment portion of the workflow system contains global knowledge about the goals, the algorithms, and the resources of the system. It schedules recurrent tasks (called procedures) whose basic work items (called activities) performed by various people (called actors) in certain roles. The work items are typically executed in a certain order (sequential or parallel.) The execution environment thus must synchronize processes, coordinate simultaneous access to data, monitor activities, and when necessary, and send reminders to actors.

Suppose that the most abstract characterization of work in an office is a *goal statement*; then work is accomplished by human(s) deciding how to achieve the goal, then by performing the work using various tools, collaborations, resources, functions, etc. provided by the workers total environment, including the execution environment. Some of the alterative subtasks may be structured work and can be cast as programs in some work specification language. Thus a workflow model can be used to represent the structured part of the work even when it is within the context of higher level unstructured work. In this case, workflow can be viewed as just another tool in the repertoire available to the information worker. Similarly, unstructured work may be identified as sub-parts of some larger structured unit of work. A clear example of

this is unexpected exception handling within routine work.
The careful co-development and integration of a workflow model and system allow the exploration of novel methodologies for the incremental introduction of workflow technology into organizations. This aspect of our research is explained in subsection 2.5.5.

Thus, our ongoing research work has five specific aspects:

(1) conceptual workflow model development,
(2) development and implementation of the specification subsystem to support the model,
(3) creation and refinement of mathematical analyses,
(4) design and prototyping of the execution environment, and
(5) methodology investigations.

2.5.1. Conceptual ICN Model

All of our workflow research is based upon a mathematical model called Information Control Nets (abbreviated ICNs). The ICN is a simple, but mathematically rigorous formalism created and designed specifically to model office procedures [Ellis79a]. We postulate a refined ICN that captures goals, control flow and data flow; relations indicate which goals apply to which procedures; which activities precede which other activities; other relations indicate which repositories are input or output for which activities. ICNs have been studied in universities [Dumas91], and used in industry [Bull92]. They have been shown to be valuable for capturing office procedures, for mathematical analyses, and for simulation. Some of the documented analyses of ICNs include throughput, maximal parallelism, and reorganization, and streamlining [Cook, 1980].

Experience with the extant ICN model identifies that workflow must be cognizant of organizational goals and structures, individual skills and preferences, as well as the sustaining social environment. These are the primary modeling research issues that must, in the longer term, be addressed. Motivation for much of this research can be seen by considering exception handling, which is a prevalent activity within offices. For example, when an exception arises such that the next activity cannot be executed, people who know the goals of their information processing, can derive alternative activities to attain the same goals. They can use their knowledge of the organizational structures to obtain proper approval of these alternative activities. They can use their social network to expedite information exchange if necessary. If the workflow system is aware of the goals and preferences, then it can assist in this process of exception handling. Thus, longer term research includes the imbedding of social and organizational sub-models within the ICN model. Coherent definitions of subgoals, of conflicting goals, of group goals, and of subgroup goals must also be incorporated.

As they have evolved, ICNs have been used to model both structured and unstructured activity, and high/low (detailed) level activities [Ellis83]. One clear lesson learned is that no one model can succinctly capture all of the quantities of potential interest to modelers. Our model is conceived, designed, and evolved to capture aspects of organizations that are most relevant to workflow. Even within the domains on which we have worked, there is a large variety of types of activities, resources, and constraints of potential interest. Thus, we define a family of models rather than a single model.

> Definition: An *Organizational Framework* is a tuple, F = [G, R] where G is a class of abstract objects (e.g. goals), and R is a class of non-procedural, concrete objects (e.g. resources and constraints.)

The exact choice of members of G and R determines the exact member of the family of models, and depends upon the modeling purpose, and level of modeling detail which is desired. Classes of abstract objects include group goals and individual preferences. Examples of classes of concrete objects are file cabinets, money, furniture, and telephones. Analysis within this goals and resources model is made feasible by attaching attributes to objects. For example, given the set of people available to do work within an office, it is useful to know each person's skill level, experience, pay expectation, location (if the organization is distributed), etc. In general, attributes are associated with classes, so all objects in the same class have the same set of attributes; e.g. all members of the class have a skill level attribute, but office furniture may not. Of course classes can be subdivided into subclasses, so for example in a university, people may be subclassed into categories such as students, faculty, etc., and each subclass has different attributes as in typical object oriented structures.
The second component of our ICN definition adds to the first (framework) part by introducing procedural objects to model structured activities and mappings to model refinement, precedence, and other relations.

Definition: An *Office Schema* is a tuple, S = [F, A, f_s] where F is an organizational framework, A is a class of procedural objects (e.g. activities nodes, goal nodes, continuation nodes), and f_s is a set of early binding mappings over F and A.

A particularly useful member of our family of models can be obtained by choosing specific classes of model nodes as the objects of study, and mappings of goal refinement, precedence between activities and input, output mappings between activities and repositories. Activities can be elementary or compound: compound activities can be either procedures or goals. Thus, the definitions of framework and schema can nestedly encompass each other reflecting the reality of the nesting of structured and unstructured work. This is in keeping with the ideas developed by Hong and colleagues of the "Society Model of Office Systems." [Ho86]
The third component of the extended ICN definition captures the dynamic nature of the phenomena which we want to model. This is a necessary component for us to study executions and dynamic structural change within organizations, and to analyze its impact.

Definition: An *Office Net* (or *Information Control Net*) is a tuple, ICN = [S, T, f_n] where S is an office schema, T is a class of tokens, and f_n is a set of late binding mappings over S and T.

The tokens introduced here are similar to Petri net tokens with attributes attached to them, and represent transactions or jobs that are flowing through the organization. One can picture the ICN model at any particular time as modeling a snapshot of the office with various transactions in various states of completion. Thus, the mapping from tokens to activities is a dynamic one, and models the current state of the system.

The extended ICN as we have described is capable of modeling structured as well as unstructured sets of activities. Another novel aspect of this ICN system is the capability of any office worker (and managers will particularly find this valuable) to see a static layered diagram of their activities, to simulate work flow by specifying hypothetical work loads, and finally to see a dynamic graphical view of real work in progress at the current instant throughout their organization.
In summary, our model recognizes that an organization, fundamentally speaking, is comprised of resources (people, money, etc.) and goals. These goals can be realized in structured or unstructured or mixed work. On top of this, we add a dynamic token function to allow us to describe a dynamic organization in action with work procedures and jobs at various stages of execution.

An office procedure may be described at the top level by identifying the goal of the procedure. In some cases there may be no further specification of the procedure; any worker that is charged with executing the procedure is expected to understand the goal, then to do the work to achieve the goal using the available organizational knowledge, tools, and resources. In this case, some of these facilities used in achieving the goal are structured work and other are unstructured work; the structured work can be defined by a sub-ICN and the unstructured work is expressed as a sub-goal.

Example: The Purchasing Cycle. In Chapter 9 of [Nutt92] an example is presented to explain how purchases are made in a company. Figure 1 is a simplification of the process to illustrate ICNs. In the example, we hypothesize a vendor and a customer, each with its own goals, i.e., the vendor and the customer perform unstructured work. Among the vendors tasks, there is structured work to fill an order; among the customer's tasks is structured work to order a part. The ICN represents these two pieces of structured work from the point of view of the customer, i.e., there are precedence arcs connecting the vendor's order-filling procedure with the customer's purchase procedure. Within the customer (buyer) procedure there are potentially many roles, e.g., purchaser, purchasing agent, shipping clerk, accountant, etc. Consider the activity labeled "Select vendor": this activity is likely to be unstructured work since it involves factoring in quality of the vendor's products, delivery schedule, price, warranty, etc. Thus this activity represents unstructured work within a structured framework. Other activities may be either totally manual, partially automated, e.g., "Supervisor approval?" or

entirely automated.

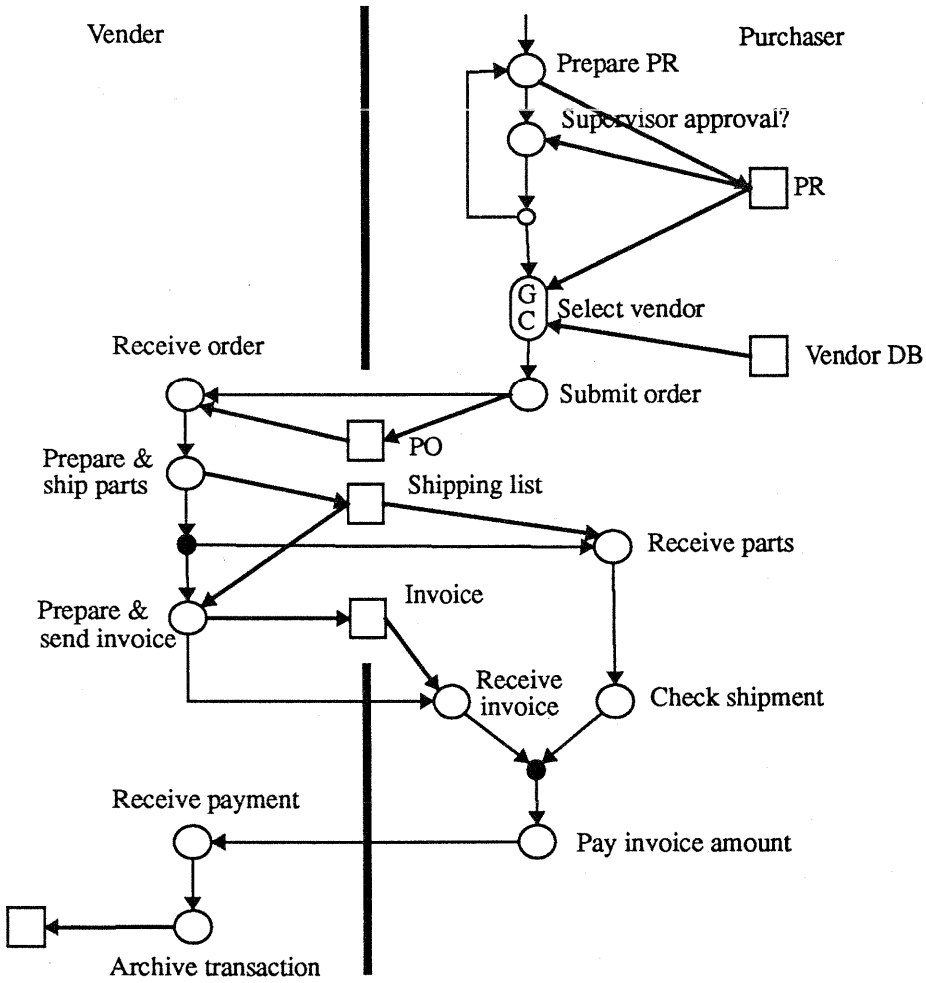


Figure 1: An ICN of the Purchase Cycle

Notice that our view allows work to be expressed at the top-level as a goal or a procedure. If the top level description is one of intent (*what* is to be accomplished), then it is expressed as a goal; if the top level description focuses on *how* the work is to be accomplished then it is expressed as a procedure or as a workflow model. Both means of defining work are acceptable and the authority that defines the work should be able to use the one most appropriate for the particular task.
Previous ICNs have been used to model structured and unstructured work; at detailed levels, and at high general levels. In this proposal, we have introduced goal and continuation nodes into ICNs. Structured work is represented by extant ICN constructs, and unstructured portions of the work are represented by a *goal nodes*. The interpretation of a goal node is that it represents unstructured work so its details are not generally representable by a workflow model. In a model of the procedure, the behavior of a goal node is explicitly an approximation to the behavior when the procedure is executed. At execution time, the goal node represents a point (possibly within a structured work specification) at which the machine hands processing off in an open manner to a human. Thus this model framework is intended to capture structured work procedures and to identify the context of unstructured work procedures whenever possible. The top-level unstructured work has no context, so it appears only as a goal without a structured context; the computation of deductible taxes is unstructured work that operates within a structured context.
In some cases, goals that appear within a structured context will be expected to *return* some kind of result, e.g., the goal to "Determine the amount of deductible taxes" should ultimately result in a number being computed and filled into some line within Schedule A. The structured work cannot be completed until the unstructured work has been completed (if the

unstructured work is never completed, then the structured work may never be completed). In these cases, the returned value constitutes a *continuation* of the scheduled work, and it is represented in the ICN by the appearance of a continuation node in the graph. Notice that a continuation is a condition for the structured work to operate, not a constraint on how a goal is treated. In parallel programming languages, this circumstance would generally be referred to as a synchronous procedure call paradigm.
In other cases, there is no requirement for the goal to return a result, or it may return many different results to many different (parts of) the ICN(s). That is, a goal node can ultimately cause zero or more tokens to appear on activities as a result of their behavior. The asynchronous procedure paradigm ("fork") is included in this class of behaviors as are many other forms of control flow.
Goal and continuation nodes are critical additions to the ICN model. They provide a means for interconnecting structured and unstructured parts of the work. The specification and execution environments address the dynamic change requirement by allowing enhanced ICNs to be changed at any time. By adding goal and continuation nodes to ICNs, the model and execution environment can address exceptions, as we will explain below.

2.5.2. The Model Support System

The original ICN model evolved with a set of companion field studies [Ellis79c]. This early experience made two points very clear: first, there is a strong need for a machine editor for physically drawing graphs; model construction is an iterative process in which most parts of the graph evolve throughout the design. While a whiteboard can be used to construct the graph, an interactive graphical editor provides superior support, and helps automate important functions. The Quinault system was a first attempt at providing machine support for ICNs. The second point is that the (structured) ICN model itself evolved into very well accepted language for describing structured procedures in the office; office workers were able to quickly grasp the language and focus on the procedures themselves.

Our previous experience with specification support systems (Quinault; Strawberry; Olympus) also indicates the overwhelming importance of a system that can provide both *quantitative and qualitative* feedback to the workflow designer in terms of animation of the model. The existing Olympus system provides such facilities for the existing ICNs, but the demands made for incorporating dynamism and goal and continuation nodes into the model will have significant implications about the nature of the supporting system. For example, how will the designer keep versions of the model organized? How can groups of designers interact on different versions of the same model? How can independent, but consistent, views be supported during a design session? How can visual interfaces scale to large models? This aspect of the research proposal is intended to address these and similar system design issues to understand and prototype model specification systems.

A system to support model specification should include facilities to:

- Create and modify models
- Save and retrieve models
- Analyze models
- Animate and simulate models

We have posed modifications to ICNs to address unstructured work and exceptions; we also require the specification system to handle change through dynamism at various levels of granularity of detail. We expect that this will require that we consider many different models within our family, and explore a variety of human-machine interaction paradigms The family of prototype specification subsystems will be derived from Olympus, since it establishes an experimental software environment in which we can experiment with different visual interfaces [McWhirter92], an environment in which it is possible to employ nonintrusive analysis tools [Beguelin90a], group model derivation [Nutt90], and initial work on scalable models. Because of space limitations, we elaborate only on scalability.

A common complaint about visual systems is their inability to scale to large problems. Preliminary experimentation with hierarchies has encouraged us to refine ideas and prototypes to attack the scaling problem by using hierarchy. We distinguish between the *visual* and *functional* hierarchies of a workflow model. The model is defined within a hierarchical framework, where any node in the model can be refined by another workflow model (among other things). Once a model has been defined within such a framework, a user's perception of the model can be defined in terms of how the model is presented on a display device (the visual aspect) or how it behaves when it is executed (the functional aspect). For example, node A could be refined by another model with nodes B_1, B_2, ... B_n; and B_i could be refined by another model with nodes C_1, C_2, ... C_m; etc. When any particular user views the model, it may be desirable to view the aspect of the model represented by node A as the single node, the B-model, or the B-model with node B_i expanded. Similarly, when one exe-

cutes the model, the user should define the level of detail at which the execution occurs within the hierarchy, e.g. when node A is executed, the executed the B-model without expanding node B_i with the C-model.

Assuming the modeling system supports multiple users: should there be one hierarchical view shared by all users? If not, is it necessary for users to "execute" the same functional level of abstraction? This leads to the notion of *sessions* in which a group of users share a specific functional and/or visual abstraction; other users that use the same model with a different functional abstraction need to have a unique session that captures the interaction with the model, e.g., model execution state. There are many systems issues related to supporting sessions in this context: should there be a session server in addition to the backend (model) server? If there are session servers, should models be cached? What are the model coherency semantics in this environment?

2.5.3. Mathematical Analysis of Exceptions

This subsection is concerned with the modeling and analysis of exception handling, and more generally, dynamic change. It briefly illustrates some aspects of the utility and analytic capability of our model. Our exposition here is short and informal; a more detailed, formal presentation of these ideas is available as a technical report [Ellis92b]. The notion of exception generally connotes a rare event. However, in studies of routine office work, exceptions to the routine are noted to occur frequently [Sasso87]. We define an exception as some condition which arises unexpectedly; thus, it is not a pre-specified part of the workflow model, by definition. It is frequently due to incorrect or incomplete information inputs, changes to information inputs or resources, special process requests, or errors in processing [Strong89]. We distinguish exception detection from exception handling. In our workflow system, exception detection may occur within any activity, and causes the activity to be refined as a sequence of activities including a goal node. The execution of this goal node allows open ended processing to perform exception handling; it may also result in tokens appearing at various other activity nodes of the ICN model, or new activity nodes may be created, or other changes. This is explained further in the subsection on execution environment.

Exceptions are an area of acknowledged importance which has not been well handled in most workflow systems. Most systems declare that exception handling is "outside of the scope of the system" which frequently results in systems which do not allow users the freedom needed to resolve exceptions[Kriefelts91]. At best, these systems do nothing about exceptions. Recall that the Prominand system attempts to list all possible exception and to provide exception handlers a priori [Karbe91]. The problem with this approach is that one cannot know and program around all exceptions in advance.
Since exceptions can have a variety of effects, our approach to modeling them is a general one in which we consider exception handling as a special case of dynamic change. The type of change we are referring to is not the changing of a data item value, but the change of basic structural elements in the model such as addition of an activity, or removal of an actor or resource. Note that a token moving from a goal node to some distant unconnected activity node is equivalent to a structural change that adds an arc to the ICN graph which extends from the goal node to the distant previously unconnected activity node.

Dynamic structural change implies that the system cannot be stopped and recompiled in order to make the changes, but that the changes must be applied while the system is in execution. Can we guarantee correct operation of the system after change? Can those transactions (i.e. tokens) which are in progress at the instant of change be salvaged? Will we introduce inconsistencies? This is a challenging problem within many types of computer systems, and in factories, software engineering, etc. Most mathematical models are not dynamic and not reflective - we will investigate both of these issues within our model. Most mathematical models are not capable of representing dynamic self change. We will investigate this issue extensively within our research, and hope that our findings will be useful to a wide range of applications. We present informally here a preliminary finding which states that under certain specified conditions, if the ICN model is known to be correct before the change, then correctness after the change will include the correctness of any transactions that are "in progress."

Dynamic structural change can be classified as temporary change (e.g., exception handling), or permanent change (e.g., organizational evolution.) Note that the addition of an arc for exception handling, as mentioned above, would be a change that would be in effect on a one time basis for some particular transaction. Thus, it is classified as temporary change. On the other hand, when the inevitable and continuous change of rules occurs within an organization, it might necessitate change of procedures which would be enacted on a permanent basis. These type of changes are important and are always occurring. Since our model is intimately bound to our system, recording of exception handling by the system might be a useful activity to assist with model evolution. If it is observed that certain exception handling activity occurs

frequently, then it could be incorporated as one of the standard available activities.
Dynamic change can be instantaneous or of some noticeable time duration. In the former case, we can assume that the structural change occurs between events of the system; in the later case, we must allow for the possibility that tokens in the system at the initiation of change may progress before the change is completed. This adds utility, but also significant analysis complexity. Dynamic change can be compound or elementary. If it is compound and non-instantaneous, then we must investigate the preferred order of change. If it is elementary change, then it is one of six types:

1. object insertion
2. object deletion
3. object editing
4. object movement
5. object merge
6. object split

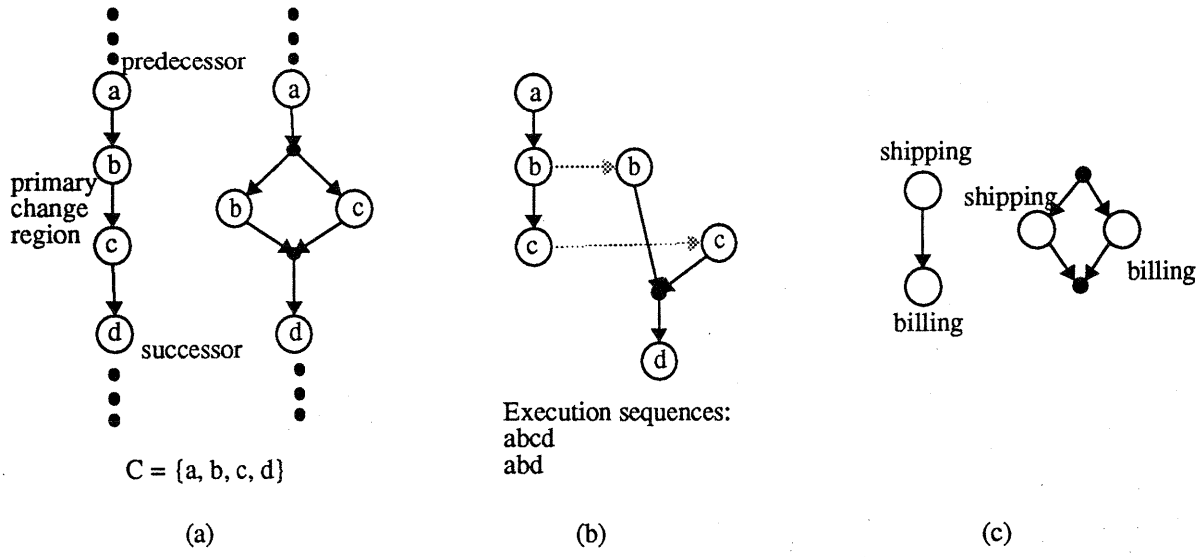


Figure 2: ICN Transformations

Given a procedure within an ICN, we can define a set of regular expressions of its activity names which we call its *correctness criteria*. An example is C = {aW*b +aW*c} which means that any execution sequence of this procedure from start to finish must begin with activity a, followed by any sequence of activities and must end with activity b or c. (* is iteration, W is the wild card that matches any string.) A change to an ICN procedure is called *stable* if it preserves correctness. Thus, given a correctness criteria set C, given a procedure P, and given a change f, we can sometimes state that the ICN procedure P' = f(P) after the change also fulfills the correctness criteria. In this case the change is stable. Notice that even if both P and P' fulfill the correctness criteria, it may occur that tokens in the system during the change violate the correctness criteria. In other words, guaranteeing the correctness of static change is not sufficient to guarantee the correctness of dynamic change (see Figure 2c). In general, this correctness depends upon what tokens are within what activities (i.e. the marking) when the dynamic change occurs. If we can guarantee this dynamic correctness for all markings, then we say that the change is *robust*.
Figure 2c shows an ICN example in which the shipping activity, followed by the billing activity, is changed to shipping and billing performed in parallel. This is a simple change that might greatly expedite transaction turnaround; the correctness criterion is simply that every customer transaction must somehow go through both shipping and billing. The correctness criterion is satisfied before and after the change, but since this is a dynamic change, a token (transaction) that was within shipping when the change occurred would skip the billing step, and the customer would never receive a bill.
The following definitions and theorem further explore the change robustness property: Given an elementary change to a procedure within an ICN, we can define the set of activities involved in the change as the *primary change region*. All predecessors to these activities are in the *upper change region*, and all successors are in the *lower change region*. Those uninvolved activities are in the outer regions. The primary change region plus its immediate predecessors and immediate successors are used to form a subgraph called the *local composite graph* by drawing lambda arcs from each node within

the primary change region of the old graph to its "equivalent" nodes in the new graph, G'. See Figure2b representing the local composite graph of the change depicted in Figure 2. Given these definitions, we can prove the following theorem.

Theorem: Given any stable elementary change to an ICN procedure P,
If the local composite graph is correct, then the change is robust.

Justification: All differences between P and P' are isolated within the local composite graph. This graph encapsulates all execution sequence possibilities for tokens which begin under control of the original ICN P, enter the primary change region, get permuted, and exit the primary change region under control of the changed ICN, P'. Thus if all of these meet the correctness criteria, then no incorrect transient behavior is possible.

Ramifications: In general checking an ICN to see if it meets a set of correctness criteria is an exponential computation. Computation time grows very rapidly as the size of the ICN grows. However, when a change is made, it is not necessary to recheck the entire ICN, and if we know that the before and after procedures are correct (static correctness), then this theorem says that a smaller checking computation will suffice to verify dynamic correctness.

Further questions posed by this investigation include
a. Can the static post-change checking be greatly reduced if we know that the ICN was correct before the change?
b. Can this result be tightened to and "if and only if" result? Can it be expanded to compound change?
c. If a system fails the robustness test, can the system give interactive guidance to humans concerning which markings, and which orderings of change are robust?
d. Can similar results be derived somehow for non-instantaneous changes?

2.5.4. Execution Environment Architecture

Model execution is distinguished from model development in the same sense as programming is distinguished from using a program to accomplish some work. It is our intent to blur this distinction in our research because these two environments will in the future merge into one. In interpretive systems that support model refinement (as in the specification system) there is little distinction between development and execution. In this aspect of the proposed research we consider issues that are specific to model execution. A primary technical distinction between the specification and execution environments is in how dynamic change and exceptions are handled.

The general requirement for the execution environment is to act as a resource manager for a network of actors assigned to activities, with the implicit goal of enabling as much concurrent operation as possible. In particular, the execution architecture portion of the research project focuses on facilities to schedule and dispatch work in which some of the activities are automated, and others are manual and others are mixed. Further, the system must be capable of providing reminders, notifications, help, information, and assistance when requested. By the nature of workflow, the system must also transfer data among actors as needed. It should also support real time collaboration, and allow flexible exception handling and dynamic change. This component should also be open in the sense that any user in the middle of executing any activity should be able to access external information if the protection rights permit.

Note that model and system interact in many ways. For example, administrators and end users can use it to see and manipulate the organizational procedures and structures during execution. This also means that an organizational specification can be simulated graphically, analyzed, and graphically demonstrated. The output of the drawings created by an office administrator can specify the set of procedures, and structures which define the default way in which jobs are handled in an organization. The execution environment can then read the model to know what activities should be executed in what order. This model needs to be dynamically changeable, while maintaining internal consistency

The execution environment interprets ICNs. Ordinary activity nodes are interpreted according to their specification in the model. If the component to be interpreted is a goal node, then the execution environment is responsible for dispatching the goal to an appropriate human actor. If the goal has a tightly coupled continuation node (the synchronous call case), then the execution environment is responsible for auditing the dispatched work, possibly issuing reminders to the actor that has been charged with fulfilling the goal. If the goal has no designated continuation, then reminders are inappropriate. Unattached continuation nodes are a mechanism for introducing external events into a structured part of the work; just as the goal node is a gateway from the structured world to the unstructured world, a continuation node is a gateway from the unstructured world to the structured world.

Exceptions can occur in any structured work, i.e., there is a specification of work that is inadequate to address some situation that arises in the course of execution of the ICN. In terms of an ICN, the occurrence of an exception within an activity means that the activity should be refined into three parts: that which has already occurred (and which is structured), the introduction of a goal node to *handle* the exception, and the representation of the remainder of the activity for those cases where exceptions do not occur. When an exception is identified, the execution environment replaces the activity in which the exception occurred with the subgraph containing the goal node. Figure 3 indicates how this situation might arise in the purchasing scenario; if the shipping list does not match the parts actually shipped, then there is an exception; notice that the parts that were received need to be marked as such but those that were not received must be identified separately and handled in a different manner. There are research issues: how should continuations be handled? Can the actor save the exception occurrence (graph) as a persistent graph? What should be the scope in which the change is saved? Here many of the issues merge the notion of specification subsystems and execution environment since we effectively treat an exception as a case in which the model changes and the unstructured part of the work is encapsulated in a goal node.

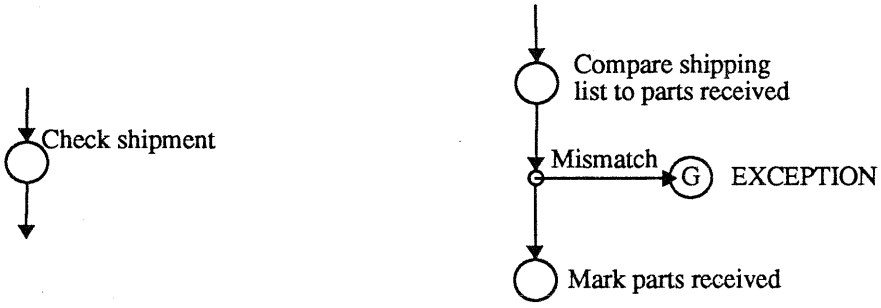


Figure 3: Handling Exceptions

A second aspect of this part of the project is to refine the capabilities of automated actors -- agents -- as was initiated in the Strawberry system. An agent should as much as possible simulate the behavior of a human actor attending to an activity; the specification of the agent should incorporate knowledge of the organization, while knowledge of the role and actor are tailorable items.

Commercial workflow systems typically utilize a client-server architecture in which resource management control information, such as the next task to be performed, is centralized in a single server. These systems also keep one centralized primary copy of all data. This approach makes the server a critical (and frequently a bottleneck) component of the system -- if the server fails then the entire system cannot operate. It also places a rigid bound on the amount that the system can grow, i.e., these systems tend not to be scalable.

The execution environment must allow the dynamic binding of actors to activities (or agents to activities), depending on the nature of the workload in the system at any given time. We see this research as being closely related to scheduling and dispatching in a network computation, and ultimately to load balancing. One direction we expect to explore is that of adaptive load balancing based on distributed simulation techniques (see [Nutt90]). The approach is based on the idea that the model can be easily replicated (e.g., by caching) at various agent locations; the agents can then be dynamically assigned activities based on the prevailing flow of work through the model. This will effectively amount to distributing the scheduler over the network.

The specification environment relies on interpretation (as opposed to translation and execution) of the workflow model. Performance or portability requirements often require that the model -- or parts of it -- be compiled. For example, it is easy to conceive of a workflow model in which parts of the system are to be executed on DOS machines, while other parts can be interpreted/executed in an arbitrary environment; this suggests that activity interpretations be translatable into source programs that (when executed) are compatible with the underlying execution environment. We expect that performance considerations will lead us to consider an approach in which parts of a model are translated and compiled,

yet still interact with the resource management part of the system at runtime.

Group computing support is another important issue. Once a specification has been derived for structured work in the office, it can only be distributed by partitioning the work into units and scheduling the execution of those units among different agents that can accomplish the work. By introducing collaboration to the workflow system, the system issues beg for more sophisticated solutions; we use an analogy to describe the added complexity of group computing: partitioning and scheduling traditional computing tasks are difficult problems in a uniprocessor environment, but are much more complex in a network computing environment; workflow for the individual versus workflow to support group computing has the same requirement for reexamining the architectures, interunit communication, scalability, and performance issues.

There are other distributed system issues that we are addressing for the execution environment to be effective; we view this system as being a testbed for experimenting with our own distributed system technologies as well as ones designed by other research groups, e.g., distributed objects, IPC protocols, scheduling, and load balancing.

2.5.5. Workflow Studies

a) Incremental Environmental Relaxation

Several aspects of this research are concerned with the effective introduction of workflow technology into organizations. Excellent technology does not insure excellent technology acceptance and usage. This is especially true in a setting in which the technology is a collaborative resource with strong possible social and organizational impacts. Failures have frequently occurred for reasons such as the following:

- system lacks openness (no access to outside information and servers)
- system lacks organizational knowledge
- inability to handle dynamic organizational change
- inability to handle exceptions
- the social nature of work

The nature of workflow suggests that there need to be alternative step-by-step approaches to the introduction of technology into organizations. We are experimenting with an approach which we call *incremental environmental relaxation.* The basic idea is to incrementally evolve from workflow simulation to enactment by gradually relaxing constraints on the model, and replacing computerized simulation modules by people!

Stage 1: Creation and Design. During this stage, one or more workflow designers create simulations of activities and procedures which can effect the goals of their organization. These activities are linked together and assigned to roles which can be enacted by one or more simulated actors. It is quite useful to use participatory design methodologies at this and later stages [Bodker88].

Stage 2: Qualitative Simulation. After the preliminary workflow specification is complete, it can be evaluated via simulated execution. It is quite useful to have a user friendly graphical modeling and simulation system at this stage [Nutt81], so that designers can visually see bottlenecks and traffic patterns. What-if analysis can be easily performed by changing loads and configurations.

Stage 3: Quantitative Simulation and Analysis. After the qualitative simulation at stage two above provides initial feedback, it is frequently necessary to obtain more quantitative feedback by successive refinements of the model. Goal nodes (representing unstructured work) must be supplied with an approximation to their behavior at this stage. This approximation may simply cause simulated time to pass, or it may simulate planning behavior in a more complex, probabilistic manner. The analysis can range from simple estimates of throughput and turn-around times to sophisticated correctness and consistency analyses.

Stage 4: Distributed Simulation. A workflow system is inherently distributed. Many people may be working on many activities simultaneously. This type of system can be nicely modelled by distributing different parts of the simulation to different nodes of a computer network. If the parts of the workflow that would be enacted by a single actor are distributed to a single unique node, then added realism is obtained in the simulation. The scheduler, dispatcher, and other parts of the execution system can be utilized to coordinate and monitor the information exchanges.

Stage 5: Human - Computer Validation. Up to this stage we have been modeling the behavior of task execution performed by people as computerized simulation modules. Since it is distributed, we can pick any one actor, and replace the simulation at that node by an actual person performing the actual activity. We reap the benefits of having a controlled environment for that person. This powerful approach can be used for incremental testing of the system and response tolerances. It can also be used for training of new personnel, and for exploring alternative loads, and alternative user interfaces

Stage 6: Human - Human Validation. A very beneficial next step of this incremental approach is the ability to select a closely interacting group of people and replace all of their simulation modules by persons. This allows a level of validation that is rarely possible without running the risk of using the full actual system as a vulnerable testbed in daily usage. At this stage, we have supplied a controlled environment for a group of people to use the system in a simulated workflow mode. Whereas step 5 allowed validation of human - computer interaction mechanisms, step 6 allows validation of human - human interaction mechanisms. The sociology of working groups can be studied and worked out before the group is placed on-line in the real work of the organization. Notice that we still have a simulated, rather than real "in-situ" enactment.

Stage 7: Enactment. When all simulation modules are replaced by humans, we have a situation which accurately mimics the "in-situ" working environment of the organization. It is then a safer and easier step to switch a team of people from simulation to in-situ operation. This seven stage process is being investigated and applied within our research activity. Many other possibilities arise for this approach. Gaming and role playing within an organization become easy. The simulated corporation can be refined to allow organizational analysis and learning. [Bair92]

The investigators' Backtalk prototype [Nutt79] (mentioned in the Related Work) was an early predecessor of the environmental relaxation approach. Our investigations build upon this earlier experience. Backtalk depended upon the existence of Officetalk to provide each user's human-machine interface and a node-to-node interface based on electronic forms. Officetalk did not incorporate a workflow model; thus Backtalk was severely constrained in its capabilities.

Environmental relaxation is useful for many other phases within the life of a workflow system. Since organizations evolve rather than being crisply defined and then enacted, environmental relaxation can be used to facilitate this evolution. Such a facility allows one to precisely specify the load applied to any part of a workflow network; thus it can be used for regression testing and quality control as a system is modified. Another of its many usages is to study when and how it is appropriate for a person to delegate a previously manual task to an automated agent (computer.) This is an exciting aspect of our investigation; the approach, when properly applied, can be powerful but delicate. It must be applied and tested in a careful, unhurried, and sensitive manner.

A methodology such as the 7 stage approach described above should only be applied after one has a thorough knowledge of the subject organization and its procedures. It is not fruitful for the workflow designer (or design team) to attempt to draw ICN specifications of an organization which is not well understood. Thus within our research, the same team that is working upon the environmental relaxation approach to one or more organizations also does workflow studies in those same organizations. These studies capture goals, procedures, activities, repositories, etc., They also capture relevant aspects of the social environment and organizational structures that are important determinants of workflow acceptance.

This research also includes the introduction of our experimental workflow system prototype into several appropriate office applications. The total process from needs assessment to workflow study to workflow technology introduction via environmental relaxation to detailed analysis and evaluation of the system in execution is included. Novel areas of particular interest are the workflow model's ability to succinctly describe the "total" office process and organization being modelled, and the workflow system's ability to gracefully handle goals, exceptions, and dynamic change.

III. Bibliography

1. Ader, M., Lu, G., "The WooRKS Object Oriented Workflow System," OOPSLA92 Exhibition, booth 712-714, October 19-21, 1992. Developed as part of the ITHACA Research project within the ESPRIT Program.
2. Baecker, R., (ed.) *Readings in Groupware and Computer Supported Cooperative Work*, Morgan Kaufmann Publishers, January 1993.
3. Bair, J. and G. Stephen, "An Investigation of the Coordinator as an Example of Computer Supported Cooperative Work," Second Conference on Computer Supported Cooperative work, Portland, Oregon, September, 1988.
4. Bair, J. "A Layered Model of Organizations: Communication Processes annd Performance," Journal of Organizational Computing, (2)1, 1991, pp. 187-203.
5. Bair, J. and Culnan, M. "Human Communication Needs and Organizational Productivity: The Potential Impact of Office Automation," Journal of the American Society for Information Science, 34(3), July 1983, pp.215-221.
6. Bair, J. (Co-editor), "Office Automation Systems: Why Some Work and Others Fail," Stanford University Conference Proceedings, Stanford University , Center for Information Technology, 1981.
7. Bair, J. "Methods for Success with New Workflow Systems," GroupWare'92, edited by D. Coleman, Morgan Kaufmann Publishers, San Mateo, Ca., pp. 160-164.
8. Barber, G. "Supporting Organizational Problem Solving with a Workstation" ACM Transactions on Office Information Systems (1,1) 1983.
9. Beguelin, A., *Deterministic Parallel Programming in Phred*, Ph.D. thesis, University of Colorado, Department of Computer Science, Boulder, Colorado 80309-0430, 1990.
10. Beguelin, A., and G. Nutt, "Examples in Phred," Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing, Houston, Texas, March, 1991.
11. Beguelin, A., and G. Nutt, "Visual Parallel Programming and Determinacy," Department of Computer Science - University of Colorado, Boulder, 1991. Submitted for publication.
12. Bentley, R., et. al. "Ethnographically-Informed Systems Design for Air Traffic Control," Proceedings of ACM CSCW'92, Oct.31-Nov. 4, 1992, pp.123-129.
13. Bock G. "Workflow as Groupware: A Case for Group Language?" GroupWare'92, edited by D. Coleman, Morgan Kaufamann Publishers, San Mateo, Ca., pp.168-170.
14. Bodker, S., et. al., "Computer Support for Cooperative Design" Proceedings of the CSCW88. Portland, Oregon. September, 1988.
15. Bond, A. H. "A Computational Model for Organizations of Cooperating Intelligent Agents," Proceedings of ACM COIS'90, April, 1990, pp.21-30.
16. Bracchi, G. and Pernici, B. "The Design Requirements of Office Systems," ACM Transactions on Office Information Systems, 2, 2, April, 1984, pp. 151-170.
17. Bull Corporation, *FlowPath Functional Specification*, Bull S. A., Paris, France, September, 1992.
18. Chrysanthis, P. K., Stemple, D. and Ramamritham, K. " A Logically Distributed Approach for Structuring Office Systems," Proceedings of ACM COIS'90,April, 1990, pp.11-20.
19. *Computer Supported Cooperative Work (CSCW), An International Journal*, Kluwer Academic Publishers, Vol. 1, 1992.
20. Cook, C., "Office Streamlining Using the ICN Model and Methodology," Proceedings of the 1980 National Computer Conference. June, 1980.
21. Croft, W. B. and Lefkowitz, L. S. "Task Support in an Office System," ACM Trans. Office Information Systems 2, 3, July, 1984, pp. 197-212.
22. Croft, W. and Lefkowitz, L. "Using a Planner to Support Office Work," Proceedings of ACM COIS'88,March, 1988, pp.55-62.
23. Croft, W. and Lefkowitz, L. "Planning and Execution of Tasks in Cooperative Work Environments," in IEEE AI, 1989.
24. Davis, D. B. " Software That Makes Your Work Flow," Datamation, 37, 8, April, 1991, pp.75-58.
25. De Jong, P. "Structure and Action in Distributed Organizations," Proceedings of ACM COIS'90, April, 1990, pp. 1-10.
26. Demeure, I. M., *A Model, ParaDiGM, and a Software Tool, VISA, for the Representation, Design and Simulation of Parallel, Distributed Computations*. PhD thesis, University of Colorado, Boulder, August 1989.
27. Demeure, I. M. and G. J. Nutt, "Prototyping and Simulating Parallel, Distributed Computations with VISA," Department of Computer Science - University of Colorado, Boulder, 1990. Submitted for publication. (Also appears in Technical Report No. CU-CS-488-90).
28. Demeure, I. M. and G. J. Nutt, "The VISA Distributed Computation Modeling System," Department of Computer

Science - University of Colorado, Boulder, 1990. Fifth International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Turin, Italy, February, 1991.
29. Demeure, I. M., S. L. Smith, and G. J. Nutt, "Modeling Parallel, Distributed Computations Using ParaDiGM - A Case Study: The Adaptive Global Optimization Algorithm," Proceedings of the Fourth SIAM Conference on Parallel Processing for Scientific Computing, Chicago, IL, December 1989.
30. Dumas, P. *La Methode OSSAD*, Les Editions d'Organization, 1991.
31. Dyson, Esther,"Workflow," Release 1.0, EDventure Holdings, New York, September, 1992.
32. Ellis, C. A., "Information Control Nets: A Mathematical Model of Office Information Flow," Proceedings of the 1979 ACM Conference on Simulation, Measurement and Modeling of Computer Systems, August, 1979a, pp. 225-239.
33. Ellis, C. "OfficeTalk-P: An Office Information System Based Upon Migrating Processes" in Integrated Office Systems, Najah Naffah (ed), INRIA, France, 1979.
34. Ellis, C. A. and P. A. Morris, "The Information Control Nets Model" *Performance Evaluation Review*, Vol. 8, No. 3 (November, 1979c).
35. Ellis, C. A. and G. J. Nutt, "Office Information Systems and Computer Science," *ACM Computer Surveys*, Vol. 12, No. 1 (March, 1980), pp. 27-60.
36. Ellis, C. "OfficeTalk-D, An Experimental Office Information System" in Proceedings of the First ACM Conference on Office Information System, June 1982.
37. Ellis, C. "Formal and Informal Models of Office Activity" in Proceedings of the IFIP International Computer Congress, Paris, 1983.
38. Ellis, C. "On the Equivalence of Office Models" CTU Journal of Computing, September 1985
39. Ellis, C. , et.al. "Project Nick: Meeting Analysis and Augmentation" with Cook, P, et.al., in ACM Transactions on Office Information Systems 5,2, April 1987.
40. Ellis, C., "Artificial Intelligence in Office Information Systems" with P. Cook, B. Desai, C. Frasson, J. Mylopoulis, N. Naffah, presented at the Joint National Computer Conference, 1988.
41. Ellis, C., Gibbs, S. "Concurrency Control in Groupware Systems" in Proceedings of the ACM SIGMOD'89 Conference on the Management of Data, May 1989.
42. Ellis, C., Rein, G."The Nick Experiment Reinterpreted: Implications for Developers and Evaluators of Groupware" with G. Rein in Office Technology and People Journal, 5,1, September 1989.
43. Ellis, C., S.J. Gibbs, G. Rein "Design and Use of a Group Editor" in Engineering for Human Computer Interaction, G. Cockton, editor. North Holland, Amsterdam, 1990.
44. Ellis, C. A., S. J. Gibbs, and G. L. Rein, "Groupware: Some Issues and Experiences," Communications of the ACM, Vol. 34, No. 1 (January, 1991a), pp. 38-58.
45. Ellis, C. "rIBIS: A Real Time Group Hypertext System" with G. Rein, in the International Journal of Man Machine Systems, 34, 1991b.
46. Ellis, C. "Workflow Doesn't Work" presented to the French Informatics Society, July 1991c.
47. Ellis, C. "Organizational Analysis Using Information Control Nets" presented at Bull S.A. Kickoff Conference, April, 1992a.
48. Ellis, C., Nutt, G. J. "The Modeling and Analysis of Coordination Systems" Position paper for the CSCW-92 Tools and Technologies Workshop. October, 1992b.
49. Engelbart D. "Toward High-Performance Organizations: A Strategic Role for Groupware," Proceedings of the GroupWare'92 conference, edited by D. Coleman, Morgan Kaufmann Publishers, San Mateo, Ca., pp. 77-100.
50. Fikes, R. "Odyssey: A Knowledge Based Assistant," in Artificial Intelligence, vol.16, p331-361, 1981.
51. Fischer, G, et. al. "Supporting Collaborative Design with Integrated Knowledge Based Design Environments," NSF Research Grant, 1990 - 1993.
52. Freeman, E. and C. Lewis, "CU/USW Research Collaboration Mid-Project Report: 1991-1992," University of Colorado Technical Report No. CU-CS-607-92, August, 1992.
53. Flores, F., Graves, M., Hartfield, B. and Winograd, T. "Computer Systems and the Design of Organizational Interaction," ACM Trans. Office Information Systems, 6, 2, April, 1988, pp. 153-172.
54. Gasser, L. "The Integration of Computing and Routine Work," ACM Transactions on Office Information Systems, 4, 3, July,1986, pp. 205-225.
55. Grudin, J. "Why CSCW Applications Fail," Proceedings of the CSCW88 Conference, ACM, pp. 85 - 93.
56. Hackman, J.R.and Morris, C. G. "Group Process and Group Effectiveness: A Reappraisal," Group Processes, edited by L. Berkowtz et.al., Academic Press, 1978.
57. Harel, D., et. al., "STATEMATE: A Working Environment for the Development of Complex Systems," in IEEE

Transactions on Software Engineering (16,4) April 1990.
58. Higgins, C.A. and Safayeni, F.R. "A Critical Appraisal of Task Taxonomies as a Tool for studying Office Activites," ACM Transactions on Office Information Systems, 2, 4, October,1984, pp.331-339.
59. Hirschheim, R. A. *Office Automation: A Social and Organizational Perspective*, John Wiley and Sons, 1985.
60. Ho, C., Hong, Y. and Kuo, T. "A Society Model for Office Information Systems," ACM Transactions on Office Information Systems, 4, 4, April, 1986, pp. 104-131.
61. Holt, A., "Diplans: A New Language for the Study and Implementation of Coordination," in ACM Transactions on Office Information Systems (6,2), 1988.
62. Karbe, B., Ramsperger, N., "Concepts and Implementation of Migrating Office Processes," Verteilte Kunstliche Intelligenz und Kooperatives Arbeiten, 4. Internationaler GI-Kongress Wissensbasierte Systeme, Munchen, Germany, Oct. 1991, pp.136.
63. Karbe, B., Ramsperger, N. and Weiss, P. "Support of Cooperative Work by Electronic Circulation Folders," Proceedings of ACM COIS'90, April, 1990, pp. 109-117.
64. Kellner, M., "Software Process Modeling Support for Management Planning and Control," in Proceeding of the First International Conference on the Software Process, IEEE Computer Society, Oct. 1991, pp.8-28.
65. Kerzner, H. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. Van Nostrand-Reinhold Co., New York, 1984.
66. Konsynski, B. R., Bracker, L.C. and Bracker, Jr., W. E. "A Model for Specification of Office Communications," IEEE Transactions on Communications COM-30, 1, Jan, 1982, pp. 27-36.
67. Kreifelts, T. "Coordination of Distributed Work: From Office Procedures to Customizable Activities," Verteilte Kunstliche Intelligenz und Kooperatives Arbeiten, 4. Internationaler GI-Kongress Wisensbasierte Systeme, Munchen, Germany, Oct., 1991, pp. 148.
68. Kreifelts, T., Licht, U., Seuffert, P. and Woetzel, G. "DOMINO: A System for the Specification and Automation of Cooperative Office Processes," Proc. EUROMICRO'84, edited by Wilson and Myrhaug, 1984, pp. 3-41.
69. Kreifelts, T., Hinrichs, E., Klein, K-H., Seuffert, P. and Woetzel, G. "Experiences with the DOMINO Office Procedure System," Proc. ECSCW'91, Kluwer Dordrecht, 1991, pp. 117-130.
70. Kreifelts, T. and Woetzel, G. "Distribution and Exception Handling in an Office Procedure System," Office Systems: Methods and Tools, edited by Bracchi and Tsichritzis, 1987, pp.197-208.
71. Lai, K., Malone, Y., "Object Lens: A Spreadsheet for Cooperative Work," Proceedings of the CSCW88 Conference, September 1988.
72. Leung, Y. "Workflow Products Market Report" B.S.A. Technical Memorandum, Bull Corporation, 1992.
73. Li, Jianzhong. *AMS: A Declarative Formalism for Hierarchical Representation of Procedural Knowledge*, PhD Thesis, L'Ecole Nationale Superieure des Telecommunications, Paris, France, December, 1990.
74. Li, Shih-Gong, *Dynamic Change in Run Mode Systems*, PhD Thesis, Information Systems Department, University of Texas, 1991.
75. Lochovsky, F., Woo, C. and Williams, J. "A Micro-Organizational Model for Supporting knowledge Migration," Proceedings of ACM COIS'90, April, 1990, pp. 194-204.
76. Lochovsky F., Hogg, J., Weiser, S. and Mendelzon, A. "OTM: Specifying Office Tasks," Proceedings of ACM COIS'88, March, 1988, pp.46-54.
77. Luqi "A Graph Model for Software Evolution," IEEE Transactions on Software Engineering, 16, 8, August 1990.
78. Lutze, R. and Triumph-Adler "Customizing Cooperative Office Procedures by Planning," Proceedings of ACM COIS'88, March, 1988, pp.63-77.
79. Martial, F. "Activity Coordination via Multiagent and Distributed Planning," Verteilte Kunstliche Intelligenz und Kooperatives Arbeiten, 4. Internationaler GI-Kongress Wissensbasierte Systeme, Munchen,Germany, Oct. 1991, pp.90.
80. Malone, T., et. al., "Semistructured Messages are Surprisingly Useful for Computer Supported Coordination," ACM Transactions on Office Information Systems (5,2) April 1987.
81. McGrath, J. E. *Groups: Interaction and Performance*, Prentice Hall, Inc. NJ, 1984.
82. McWhirter, J. D., *DG: A Framework and System for the Specification and Instantiation of Graph Models*, Ph.D. Dissertation, in progress, October, 1992.
83. McWhirter, J. D. and G. J. Nutt, "A Characterization Framework for Visual Languages," *Proceedings of the 1992 IEEE Workshop on Visual Languages*, September, 1992, pp. 246-248.
84. Medina-Mora, P. et. al., "The Action Workflow Approach to Workflow Management Technology," Proceedings of ACM CSCW'92, Oct. 31-Nov. 4, 1992, pp. 281-288.
85. Medina-Mora, P. "Action Workflow TM Technology and Applications for Groupware," GroupWare'92, edited by

D.Coleman, Morgan Kaufmann Publishers, San Mateo, Ca., pp.165-167.
86. Newman, W. "Office Models and Office Systems Design," In Naffah, N. Ed. Integrated Office Systems, Burotics North-Holland,1980, pp.3-10.
87. Newman, William M., *Designing Integrated Systems for the Office Environment*, McGraw-Hill, New York, 1987.
88. Nutt, G. J., "An Experimental Distributed Modeling System," ACM Transactions on Office Information Systems, Vol. 1, No. 2 (April, 1983), pp. 117-142.
89. Nutt, G. J., "A Flexible, Distributed Simulation System," 10th International Conference on Application and Theory of Petri Nets, Bonn, West Germany, June 1989, pp. 210-226.
90. Nutt, G. J., "A Formal Model for Interactive Simulation Systems," Tech. Rep. CU-CS-410-88, Department of Computer Science - University of Colorado, Boulder, May 1989.
91. Nutt, G. J., "A Simulation System Architecture for Graph Models," In Advances in Petri Nets '90, G. Rozenburg, Ed. Springer Verlag, 1990.
92. Nutt, G. J., "Distributed Simulation Design Alternatives," Proceedings of the SCS Conference on Distributed Simulation, San Diego, CA, January 1990, pp. 51-55.
93. Nutt, G. J., *Open Systems*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1992.
94. Nutt, G. J., A. Beguelin, I. Demeure, S. Elliott, J. McWhirter, B. Sanders, "Olympus: An Interactive Simulation System," 1989 Winter Simulation Conference, Washington, D.C., December 1989, pp. 601-611.
95. Nutt, G. J. and C. A. Ellis, "Backtalk: An Office Environment Simulator," ICC '79 Conference Record, June, 1979, pp. 22.3.1-22.3.5.
96. Nutt, G. J. and P. A. Ricci, "Quinault: An Office Modeling System," IEEE Computer, Vol. 14, No. 5 (May, 1981), pp. 41-57.
97. Olfman, L. and Bostrom, R. "The Influence of Training on Use of End-User Software," Proceedings of ACM COIS'88, March, 1988, pp. 110-117.
98. Osterweil, L., "Automated Support for the Enactment of Rigorously Described Software Processes," Proceeding of the Third International Process Programming Workshop, 1988, pp122-125. IEEE Computer Society Press.
99. Palermo, A. and McCready, S. "Workflow Software: A Primer," GroupWare'92, edited by D. Coleman, Morgan Kaufmann Publishers, San Mateo, Ca., pp.155-159.
100. Panko, R., "35 Offices: Analyzing Needs in Individual Offices" Proceedings of the Second ACM-SIGOA Conference on Office Information Systems, June 1984.
101. Poltrock, S., Grudin, J., "Tutorial on Computer Supported Cooperative Work and Groupware," Presented at ACM SIGCHI Conference on Human Factors in Computing Systems, New Orleans, April 27, 1991.
102. Rasmussen, J., Duncan, K. and Leplat, J. (Ed.) *New Technology and Human Error*, John Wiley and Sons, 1987.
103. Rein, G., *Organization Design Viewed as a Group Proces Using Coordination Technology*, PhD Thesis Dissertation, Department of Information Systems, University of Texas at Austin. May 1992.
104. Rein, G., Singh, B., and Knutson, J., "The Grand Challenge: Building Evolutioary Technologies," Proceedings of the HICSS93 Conference, January 5-8, 1993.
105. Rodden, T. "The COMIC Project-COmputer-based Mechanisms of Interaction in Cooperative Work," ESPRIT Basic Research Project #6225, Technical Annexe, 27th, April, 1992.
106. Roos, L., and Stark, F. "Organizational Roles" in Handbook of Social Psychology, 2nd Edition, (pp488-567) editors Lindzey, G., and Aronson, E. 1968.
107. Sarin, K. S., Abbott, K. R. and McCarthy, D. R. "A Process Model and System for Supporting Collaborative Work," ACM COCS'91, pp. 213-224.
108. Sasso, W. C., Olson, J. R. and Merten, A. G. "The Practice of Office Analysis: Objectives, Obstacles, and Opportunities," IEEE Office Knowledge Engineering, 1,1, 1987, pp.11-24.
109. Sasson, P. "Cost Benefit Analysis of Information Systems: A Survey of Methodologies," Proceedings of ACM COIS'88, March, 1988, pp.126-133.
110. Shaw, M. E., Group Dynamics: the Psychology of Small Group Behavior, McGraw-Hill Book Company, NY, 1981.
111. Sirbu, M., Schoichet, S., Kunin, J. S., Hammer, M. and Sutherland, J. "OAM: An Office Analysis Methodology," Behaviour and Information Technology, 3,1, 1984, pp.25-39.
112. Strong, D. M. "Design and Evaluation of Information Handling Processes," Ph.D. Dissertation, Carnegie Mellon University, 1988.
113. Strong, D. M., Miller, S. M. "Exception Handling and Quality Control in Office Operations," Working Paper Number 89-16, Boston University, Boston, MA., September, 1989.
114. Suchman, L. A. "Office Procedure as Practical Action: Models of Work and System Design," ACM Transactions

on Office Information Systems, 1, 4, October, 1983, pp. 320-328.
115. Tueni, M., Li, J. and Fares, P. "AMS: A Knowledge-based Approach to Task Representation, Organization and Coordination," Proceedings of ACM COIS'88, March, 1988, pp.78-87.
116. University of Michigan, MCC, and Arthur Andersen Co. "Technology Support for Collaborative Workgroups". Grant proposal funded by the National Science Foundation, 1989-1991.
117. Von Martial, F. "A Conversation Model for resolving conflicts among Distributed Office Activities," Proceedings of ACM COIS'90, April, 1990, pp.99-108.
118. Winograd, T. "Groupware and the Emergence of Business Technology," GroupWare'92, edited by D. Coleman, Morgan Kaufmann Publishers, San Mateo, Ca., pp.69-72.
119. Winograd, T. and Flores, F. *Understanding Computers and Cognition: A New Foundation for Design,* Ablex Publishing Co., NY, 1986.
120. Woo, C. "SACT: A Tool for Automating Semi-Structured Organizational Communication," Proceedings of ACM COIS'90,April, 1990, pp.89-98.
121. Zigurs, I., Kozar, K., "An Exploratory Study of Roles in Computer Supported Groups" submitted to the Journal of Management Information Systems, 1992.
122. Zisman, M. D. *Representation, Specification, and Automation of Office Procedures,* Ph.D. dissertation, Wharton School, University of Penn., 1977.

IV. Biographical Sketches

4.1. Clarence A. Ellis
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, CO 80309-0430
(303)-492-7514

Professional Summary:

Dr. Clarence A. Ellis is Professor of Computer Science and Co-Director of the Collaboration Technology Research Group at the University of Colorado. At Colorado, he is a member of the Systems Software Lab, and the Institute for Cognitive Science. He is involved in research on groupware, coordination theory, and distributed systems. Dr. Ellis has worked as a researcher and developer at MCC, Xerox PARC, Bull Corp, Bell Telephone Labs, IBM, Los Alamos Scientific Labs, and Argonne National Lab. His academic experience includes teaching at Stanford University, MIT, University of Texas, Stevens Institute of Technology, and at Chiaotung University in China under an AFIPS overseas teaching fellowship.

Clarence (Skip) Ellis is on the editorial board of numerous journals, and has been an active instigator and leader of a number of computer associations and functions; he has been a member of the National Science Foundation Computer Science Advisory Board; of the NSF Computer Science Education Committee; and chairman of the ACM Special Interest Group on Office Information Systems (SIGOIS). His interests include coordination theory, object oriented systems, CSCW, office systems, databases, distributed systems, software engineering, systems design and modeling, and humane interfaces to computers.Skip Ellis has published over 100 technical papers and reports, lectured in more than a dozen countries, recently completed a book entitled "Distributed Cooperation in Integrated Information Systems," and was an invited speaker on object oriented systems at the most recent IFIP World Computer Conference.

Professional Positions:

University of Colorado, Boulder, Colorado
Position - Full Professor, Computer Science Department
Dates - January 1992 to present
Work areas -teaching, research, minority advising, graduate student supervision.
Teaching groupware, operating systems, computing for non-scientists, CSCW (in business school).
Co-Director of the Collaboration Technology Research Group.

Bull S.A.Corporation, Paris, France
Position - Chief designer and team leader, workflow product group
Dates - July 1991 to December 1991

Institute for the Future, Menlo Park, California
Position - Visiting Research Fellow
Dates - January 1991 to June 1991

Microelectronics and Computer Technology Corporation (MCC), Austin, Texas
Position - Research Scientist and Technical Director, Design Interface Group
Date - September 1985 to December 1990.
Work areas - Computer supported cooperative work (CSCW), object oriented systems, databases, software engineering, user interfaces. First major project was principal designer and implementor of the Gordion object oriented database for software engineering environment. Second project was (and is) technical research group leader for development of theory, prototypes, and empirical studies of groupware (developed electronic meeting rooms, and distributed collaboration systems and concepts.) Numerous minor projects simultaneously pursued.

University of Texas, Austin, Texas
Position - Adjunct Associate Professor, MSIS and Department of Computer Science
Dates - September 1988 to Feb. 1991
Work areas - teaching office information systems, and advising graduate students

Stanford University, Stanford, California
Position - Visiting Professor, Department of Computer Science
Dates - September 1984 to September 1985 (plus occasional teaching 1980-84)

Work areas - taught distributed systems, office systems; worked within database group; wrote book.

Xerox PARC (Palo Alto Research Center) Palo Alto, California

Position - Systems designer; computer researcher; group leader

Dates - September 1977 to 1985

Work areas - Office systems research and development, user interface management systems, smalltalk and object oriented systems, software environments, human computer interaction, software project management. Major project was working within smalltalk research group, creating novel user interface paradigms and exploring ideas and architecture of parallel smalltalk. Another major project was head of the office research group - created and pioneered the ICN analysis model - designed and implemented a series of Officetalk prototypes. Afterwards, participated as a designer and manager within the product division in the large software implementation project of the Xerox STAR Office Information System.

Massachusetts Institute of Technology (MIT), Cambridge, Massachusetts

Position - Assistant Professor of Computer Science

Dates - September 1975 to 1977

Education:

Bachelor's Degree - June 1964 (Math and Physics) Beloit College, Wisconsin

Master's Degree - June 1966 (Mathematics) University of Illinois, Urbana, Illinois

Ph.D. - June 1969 (Computer Science) University of Illinois, Urbana, Illinois

Relevant Publications:

"Groupware: The Research and Development Issues" with S. G. Gibbs, G. Rein, Journal of the ACM, 34,1, 1991.

"Formal and Informal Models of Office Activity" in Proceedings of the IFIP International Computer Congress, Paris, 1983.

"Information Control Nets: a Mathematical Model of Office Information Flow" in Proceedings of the Ninth Annual ACM Conference on Simulation, Measurement, and Modeling of Computer Systems, August 1979.

"OfficeTalk-D, An Experimental Office Information System" in Proceedings of the First ACM Conference on Office Information System, June 1982.

"The Architecture of Workflow Systems" with G. J. Nutt, University of Colorado Technical Report, Department of Computer Science, December 1992.

Other Significant Publications:

"Design of Office Information Systems" Book. Springer-Verlag, Heidelberg, Germany, 1987.

"Office Information Systems and Computer Science" with G. J. Nutt, in ACM Computing Surveys, 12,1, March 1980.

"The Nick Experiment Reinterpreted: Implications for Developers and Evaluators of Groupware" with G. Rein in Office Technology and People Journal, 5,1, September 1989.

"rIBIS: A Real Time Group Hypertext System" with G. Rein, in the International Journal of Man Machine Systems, 34, 1991.

"Performance Analysis of Two Concurrency Control Schemes for Design Environments" with Ege, A., Korth, H., and Yeh, S., in Information Sciences Journal, 49,1, 1989.

Colleagues:

Professor David Muller is Ellis' Thesis advisor.

4.2. Gary J. Nutt
Department of Computer Science
Campus Box 430
University of Colorado
Boulder, CO 80309-0430
(303)-492-7514

Professional Positions:
November, 1986 - present
Professor Department of Computer Science and
Department of Electrical and Computer Engineering
University of Colorado
Boulder, Colorado
Teaching and research in modeling, measurement, and simulation of distributed programs and systems. Director of the Colorado Open Systems Consortium. Co-Director of Collaboration Technology Research Group. Active industry and educational consultant.
March, 1984 - November, 1986
Vice President, Colorado Technical Office
Interactive Systems Corporation
Boulder, Colorado
Managed a cost center which included a staff of 25 programmers with an annual budget of approximately $2 million. Responsible for network and application software planning, development, and production; and UNIX system software development.
April, 1981 - March, 1984
Acting Director/Department Manager
NBI, Inc.
Boulder, Colorado
Managed an organization of 85 engineers with an annual budget over $4 million. Responsible for all hardware and software aspects for a product line composed of workstations, servers, and a LAN.
June, 1980 - April, 1981, Member of the Technical Staff, Bell Laboratories, Denver, CO.
July, 1978 - June, 1980, Member of the Research Staff, Xerox PARC, Palo Alto, CA.
August, 1972 - July, 1978, Associate Professor (tenured), Department of Computer Science, University of Colorado, Boulder, CO

Related Experiences:
Recent Grants
June, 1992 - August, 1993, Co-PI, Bull S. A., "FlowWorks Research and Development".
October, 1992-August, 1993: PI, NCR, "Generating Visual Frontends".
September, 1990 - August, 1995, Co-PI, NSF CISE II, "Effective Use of Parallel and Distributed Computing"
September, 1990 - August, 1993, Co-PI, US West, "CU-US West Research Partnership".
Selected Professional Committees
June, 1993: Program Committee, 14th International Conference on Application and Theory of Petri Nets, Chicago, Illinois.
March, 1993: Program Chairman, 1993 COSC OpenExpo, Denver, Colorado.
May, 1990: General Chairman, 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Boulder, Colorado.
September, 1980: Program Co-Chairman, 19th Annual Lake Arrowhead Workshop on Office Information Systems.

Education:
B.A., Mathematics, Boise State University, Boise, Idaho, 1967.
M.S., Computer Science, University of Washington, Seattle, Washington, 1970.
Ph.D., Computer Science, University of Washington, Seattle, Washington, 1972.

Relevant Publications:

"Office Information Systems and Computer Science," with C. A. Ellis, ACM Computing Surveys, Vol 12, No. 1 (March, 1980), pp. 27-60.

"An Experimental Distributed Modeling System," ACM Transactions on Office Information Systems, Vol. 1, No. 2 (April, 1983), pp. 117-142.

"A Flexible, Distributed Simulation System," Tenth International Conference on Application and Theory of Petri Nets, Bonn, West Germany, June, 1989, pp. 210-226.

"Olympus: An Interactive Simulation System," with A. Beguelin, I. Demeure, S. Elliott, J. McWhirter, and B. Sanders, Proceedings of the 1989 Winter Simulation Conference, Washington, D. C., December 4-6, 1989, pp. 601-611

Nutt, G. J., "A Simulation System Architecture for Graph Models," Advances in Petri Nets '90, G. Rozenburg, Ed. Springer Verlag, 1990.

Other Significant Publications:

Open Systems, Prentice Hall, Englewood Cliffs, NJ, 1992.

Centralized and Distributed Operating Systems, Prentice Hall, Englewood Cliffs, NJ, 1992.

"A Parallel Processor Operating System Comparison," EEE Transactions on Software Engineering, Vol SE-3, No. 6 (November, 1977), pp. 467-475.

"The VISA Distributed Computation Modeling System," with I. M. Demeure, Proceedings of the Fifth International Conference on Modeling Techniques and Tools for Computer Performance Evaluation, Turin, Italy, February, 1991.

"A Testbed for Studying Parallel Programs and Parallel Execution Architectures," with D. Grunwald, A.Sloane, D. Wagner, and B. Zorn, to appear in Proceedings of MASCOTS'93, San Diego, CA, January, 1993.

Colleagues:

Former Students

Isabelle M. Demeure, Ph.D, August, 1989 (advisor).
Adam L. Beguelin, Ph.D. August, 1990 (advisor).

Graduate and Postdoctoral Students Advised

During the last five years,the Co-PI has also advised the following PhD students: Mohammad Amin, Rick Blumenthal, Zulah Eckert, Stephen Elliott, Paul Maybee, Jeff McWhirter, Sung-Hee Nam, and Nicole Sloane. No postdoctoral students have been advised during the last five years.

Long Term Associates

Jerre D. Noe is the Nutt's thesis advisor.