Using process technology to support cooperative work: prospects and design issues

David Wastell and Phil White

Department of Computer Science, University of Manchester,
Oxford Road, Manchester, M13 9PL, U.K.

This chapter describes a form of technology known as PSS (Process Support System). PSS provides a set of techniques for modelling cooperative work and a flexible generic technology for building office information systems. The chapter introduces a number of related office technologies to set the scene for a description of PSS. The approach to process modelling in PSS is introduced using a medical case study, which is also used to give a flavour of the experience of process supported office work. Epistemological and methodological issues in process modelling are explored. Several lines of work are highlighted, including the development of a design methodology for PSS and language-action approaches to process modelling. A major implementation of PSS is described in outline. Issues arising from field studies are discussed, e.g. the question of the routineness of so-called routine work.

1. Introduction

CSCW is a diverse and eclectic field. The theme which unifies CSCW is the question of group coordination, how it is achieved as a social phenomenon and how it may be actively assisted by computer-based support. The nature of these social processes are variously discussed in many of this book's other chapters.

The issue of what is "true" CSCW and what is not is a contentious academic issue. Support for non-routine "professional" work such as collaborative writing would be widely accepted as a paradigm of CSCW (see, in particular, Sharples, Chapter 4; Gilbert, chapter 5; and Diaper, Chapter 6). Electronic mail, however, does not count for some as CSCW, because it is "not really tuned (or tunable) to the needs of the work group" (Greif, 1988). Technologies which support routine work would appear to fall into a particularly controversial category. Traditional office automation systems come under this heading. Data processing applications, in which computer systems which simply "drive" short-cycle repetitive clerical work (e.g. pay-roll systems): are they to be described as CSCW? Some would contemptuously say not. The authors' view is more moderate and less elitist. CSCW provides a fresh perspective, an antidote to the traditional view of systems which emphasises information processing and the internal design of the computer system. CSCW brings people and their tasks into focus. Any area of organisational work in which people use computers can be helpfully illuminated from a CSCW perspective, even pay roll! All work has a social dimension. CSCW brings it into the foreground.

This chapter describes a form of technology known as PSS (Process Support System) which developed out of an Alvey project involving STL, Manchester University and others (see below). PSS was originally known as IPSE 2.5 in its Alvey days (IPSE - Integrated Process Support Environment).

PSS provides both a powerful set of modelling tools for describing organisational work and a flexible generic technology for building systems to support cooperative work.The technology was originally oriented at supporting software production, but it has much wider applicability than this domain. PSS provides a technology ideally suited for co-ordinating and supporting routine work in offices. It should be noted, however, that PSS is a fundamentally different type of system to traditional office automation systems, which focus on automating a particular task or function. PSS focuses on the provision of co-ordination across functions, which may be performed by many different people and machines. In this chapter, we focus on the use of PSS as a means of supporting cooperative work in offices, i.e. as a technology for constructing office information systems (OISs). The chapter describes PSS, its origins, modelling conventions and the technology itself. Current developments and new research surrounding PSS are also discussed.

2. Office automation

The definitions of an Office, of Office-work and of Office Automation are themselves troublesome. The interested reader is referred to Hirschheim (1985) for an extended, if rather inconclusive, discussion. Hirschheim offers the following as a general purpose working definition: "Offices can be thought of as centres of organisational information handling and processing". The idea of rules and procedures is central to the popular view of office-work leading Olson and Lucas's (1982) to the following definition of office automation as "the use of integrated computer and communication systems to support administrative procedures in an office environment". The design of PSS reflects this view of office automation, although as we shall see the bureaucratic idea that office work is simple "procedure-following" is naive and misleading.

PSS is not the first system to attempt to provide task co-ordination across the office environment. In order to put PSS into context, we will briefly discuss some previous attempts in the OA/OIS field which are notable either for their general importance or their specific relevance to PSS.

Ellis and Nutt (1988) give details of several automated office information systems. Officetalk, for instance, was developed by Xerox PARC during 1976-77. Interestingly it grew out of a study of languages for expressing office procedures. Officetalk is intended to be used to aid in document management, preparation and communication. It is based on the idea of substituting electronic forms for the paper ones commonly used in offices. Thus the user has an electronic desktop with in-tray, out-tray, a set of blank forms, and a file for forms being worked on. Tools for tailoring blank forms are an important feature. An electronic mail system allows documents to move from one user to another.

Another system similar to Officetalk is XCP, developed by Sluizer and Cashman (1984) at DEC. They describe XCP as "an experimental coordinator tool which assists an office in implementing its procedures". Like Officetalk, XCP is very much a document-driven system. There are four basic concepts in XCP, those of person, role, actor and document. A person may perform several roles. Documents of pre-defined types may be created by certain roles, and they then move down pre-defined paths. Thus someone in role type A will perform certain actions relating to the document, and then pass the document either to a specific person performing role B, or back to the system to assign to someone in role type B. The system tracks the progress of the document, and keeps a history of it.

Ellis and Nutt draw a distinction between document-oriented models and process-oriented models of office automation systems. Documents are a way of organising information, and secondarily of organising activities. Both Officetalk and XCP are rather unsophisticated, largely document-oriented systems. Zisman's SCOOP (System for Computerization Of Office Processing) is based on Petri nets augmented by production rules as a means for modelling offices as asynchronous concurrent processes (see Ellis and Nutt for details). In contrast to Officetalk and XCP it is process-oriented. SCOOP is run by an *execution monitor,* driven by an internal representation of a set of augmented Petri nets. The firing of a Petri net transition causes an updating of the available productions. Processes may be either automatic or interactive. There is also the idea (a key feature of the design of PSS, as we shall see) of using external tools such as document generators and mail services.

Ellis and Nutt devote a major part of their review to describing a diagramming convention dubbed the Information Control Net (ICN) developed by themselves and others at Xerox PARC. An ICN is a diagrammatic method of representing office procedures as a set of activities, a set of repositories, and a set of functional mappings between these elements, backed by a formal definition. ICNs are intended to be "used both to describe offices to managers and to analyse the office". An ICN defines an office as a set of related procedures. Each procedure consists of a set of activities connected by precedence constraints.

Another interesting system is the Activity Model Environment (AME) of Smith, Hennessy, and Lunt (1991). They describe AME as "a framework for modelling organisational processes and a prototype execution environment". AME is an object-oriented tool to explore models of organisations. It has a conceptual framework formed of eight components: activities (goal-oriented cooperative tasks); people; roles (groups of logically related duties/responsibilities); workspaces ("conceptual work areas" containing resources for particular roles e.g an in-tray); messages ("persistent" objects such as documents, memos etc. which convey task relevant information); information units (atomic units of information, e.g. a name, account number etc.); rules (if/then productions defining and constraining the behaviour of roles, messages etc.); functions (atomic operations carried out by roles and messages as part of an activity).

Winograd has written several influential articles on CSCW in which he develops what he calls a Language/Action Perspective on the design of computer-based systems, as a counterpoise to the traditional information processing perspective (see Winograd, 1988 and Rodden, chapter 1 of this volume). It is interesting that similar ideas have recently become topical in the information systems field, apparently independently (see Lyytinen, 1987; Auramaki et al., 1988). Winograd makes use of the division in linguistics of language into syntax, semantics and pragmatics in order to lay the basis for his perspective. He is primarily concerned with pragmatics and draws considerably on Austin's speech act theory. The language/action perspective emphasises that "people act through language". This has two aspects. Firstly, that when people have a conversation they are *doing* something, often many things interleaved, such as: giving information, helping someone with their work, establishing personal ties etc. Secondly, that speech acts can only be seen as having meaning with regard to the social context in which they are performed, in that they fulfil obligations or set up new commitments.

The main focus of Winograd's work is on what he terms the "conversation for action" (CfA). He has taken the idea that most co-ordination in organisations occurs through conversations (either oral or *written*) which follow certain set patterns. CfAs are chiefly composed of two types of speech acts: directives (orders) and commissives (promises). The Co-ordinator (see Brooke, Chapter 2 and Sharples, Chapter 4) is a software tool for setting up and managing CfAs. It offers two main advantages over conventional media. First, clarity. Conversations in organisations are often mixed as to subject matter, and it may be unclear, in the case of requests, for instance, exactly what the request is for, its importance, its due date etc. The second important feature of computer-mediation is that it allows people, managers in particular, to "keep track of the action", i.e. to keep track of commitments, in particular to monitor their fulfilment. Auramaki et al (1988) advance a similar line of reasoning for their speech-act approach to modelling office activity.

3. PSS: Process Support System

3.1 Background and overview

As has been said, PSS was developed out of the IPSE 2.5 project - a part of the UK government's Alvey initiative. The IPSE 2.5 project involved a consortium of industry and academe: STL plc, Plessey, British Gas, Dowty Defence Systems and the University of Manchester. The original purpose of IPSE 2.5 was to provide the means by which the process of developing, maintaining, supporting and enhancing information systems could be made more efficient. Traditionally in software engineering, support environments have been considered as tools to assist the software engineer. The IPSE 2.5 project stood back from this traditional perspective and took a wider view of the problem. It was recognised that the key to better software development was not increasingly sophisticated tools, but better coordination amongst the many people who are necessarily involved in developing software, i.e. that support for software development as a complex sociotechnical process was required. Although the partners in IPSE 2.5 were primarily interested in supporting software development, there has always been awareness that the technology had a much wider scope. The technology produced, PSS, provides a generic platform for supporting cooperative work across many domains. Its application to general office-work is of particular interest.

At the heart of PSS is the concept of a Process Control Engine, or PCE. The PCE is a computer system which provides an "active work environment" for its users, i.e. a medium which actively supports and coordinates their activities. If a PCE is to provide active support for a process, then of course it must be supplied with comprehensive knowledge of the activities to be supported. An important part of the IPSE 2.5 project was thus the development of a Process Modelling Language (PML) which allows a PCE to be programmed, and *dynamically reprogrammed*, in order to support a given process. Using PML, a process (e.g. software production) is modelled as a network of asynchronous sub-processes which are loosely coupled and need from time-to-time to communicate with each other. These exchanges occur using an asynchronous rendezvous-like mechanism; the overall pattern of these exchanges reflects the cooperative structure of the modelled process.

The choice of terminology in PML reflects its work-oriented concern with support for human agents engaged in co-operative activities. Sub-processes are referred to as Roles and co-operative work is thus conceived of as a network of interacting Roles. Broadly speaking, developing a PML model involves breaking down cooperative work into Actions and Roles and defining the dependencies between Roles. Actions refer to primitive activities which may be carried out either by people or indeed automatically by software tools embedded in the support environment. A Role refers to a logically coherent collection of Actions related to the achievement of a defined goal, often (but not necessarily) to be carried out in a fixed temporal sequence. Holding a Role confers upon some agent the entitlement and obligation to perform certain tasks and the necessary resources for this purpose. The dependencies between Roles are referred to as Interactions. Such dependencies normally reflect the need of two Roles to exchange information, e.g. that person A needs to transmit such-and-such a document to person B etc.

The original implementation of PML was written in SMALLTALK (Goldberg, 1984) and the language has a distinctive object-oriented flavour. The main constructs are all conceptualised as objects (Roles, Actions etc.). It is important to distinguish between the class definition of an object (e.g. a Role-class, Action-class etc.) and particular instantiations of that class (Role-instances, Action-instances etc.).

3.2 Process modelling in PML: a hospital example

To illustrate the main features of process modelling in PSS and to give an impression of the experience of PSS supported work, a case study of medical office-work will be described. This example draws on a independent project, known as the Clinical Information Systems Project (CISP), which was set up in 1988 (for background see Wastell et al., 1987) to investigate computer support for hospital work. The field-work for the project was carried out in the Cardiology department of the Manchester Royal Infirmary (a large teaching hospital adjacent to the University) and the focus of the investigation was the medical office. The reader should picture the Cardiology office as a large untidy room in which five secretaries and two clerks, surrounded by piles of hospital records and using mechanical typewriters, endeavour valiantly to service the clinical work of the three consultants and their registrars. The project in its early days was a rather conventional exercise in systems analysis; the investigators did not initially have a CSCW outlook, nor was PSS of specific interest.

Medical office-work embraces a wide range of activities: typing, liaison, coordination, communication and record keeping, all of which serve the purpose of supporting, indeed enabling, the clinical work of the medical and paramedical staff. The early work in CISP followed traditional practices: documenting data flows, constructing data models etc. It soon became clear that the orthodox design and modelling techniques were limited in their perspective on office work. They conjure up an impersonal and mechanistic image of offices as information processing systems. What was needed was a more human view of office work reflecting its nature as cooperative social activity and doing justice to its social and technical complexity. Pava (1983) refers to the non-linear and concurrent character of office work. Office work consists of many overlapping tasks. Moreover, these tasks are not exact replicas of an idealised template; they comprise many steps which are not simply performed in a strict sequence. Modelling tools for describing office work must thus reflect its cooperative character, its parallelism and its indeterminacy. The analytical techniques of PSS appeared to provide such a richer language and the research effort in CISP turned to develop formal models of medical office work using PML, on which some working prototypes could be developed (Maresh and Wastell, 1990).

Process models for various areas of the work of the Cardiology office were developed. The example here draws on a large process model which was developed to express the administrative work supporting the organisation of the department's outpatients clinics. This process characteristically begins with a referral letter from a patient's general practitioner (GP) to the department which is assessed for urgency (by, for instance, a consultant's secretary) and an appointment is then made to see a doctor at an outpatients clinic. The process proceeds after the visit through a number of possible stages (e.g. laboratory investigations) culminating in a letter from the consultant to the GP reporting his expert assessment. A large number of people are involved in this process: patients, secretaries, clerks, records staff, porters, doctors, nurses etc. Frequently the process fails in various ways, e.g. notes cannot be traced, patients wait interminably, communication with GPs takes much longer than it ought. The work is complex and highly interdependent. There is, however, much that is routine and repetitive, and therefore formalisable and amenable to computer support, with the anticipation of tangible benefits.

3.3 A process model of medical office work

A simple diagramming convention called the Role Activity Diagram (RAD) has been adopted to help construct PML models and reason about processes. RADs are inadequate to express the full complexity embodied in a PML model, but are useful to give some idea of the modelling concepts and an overall view of the process under examination. Figure 1 shows an RAD for the whole outpatients process. The RAD notation is simple: individual actions are indicated by boxes; interactions by horizontal bars (transects). The clustering of actions into logically-related groups. i.e. Roles, is reflected in the columnar organisation of the RAD. Several roles have been defined in figure 1, including an overarching *Patient_Managing* Role which reflects the idea of some agent taking overall responsibility for orchestrating the whole progress of a single outpatient episode. Figure 1 serves to give a high level view of the full outpatients process but is rather cumbersome for didactic purposes. An abbreviated version of the outpatients process is shown in figure 2. This simplified version will be used for the remainder of the discussion.

Figure 2 shows three roles only, with overall management of the outpatients episode for each patient in the care of an *Outpatient-ing* role. The diagram should be understood as follows. In the real world, when a referral letter is received, this initiates a new patient episode; we may speak of a new instance of the outpatients process as being instantiated. Figure 2 simply defines this process to involve the instantiation of three conceptually distinct activities, three Roles. The first of these is *Outpatient-ing*. The first action in *Outpatient-ing* is to create a referral record for the new patient. Next, the urgency of the case must be assessed before an appointment can be made. The RAD in figure 2 introduces a second Role for this purpose (*Urgency_Assessing*). This Role is, as it were, triggered by *Outpatient-ing* which subsequently receives the results of the assessment via an interaction. The reason for splitting off a separate Role is to allow the assessment to be carried out by an agent different from the agent associated with *Outpatient-ing,* e.g the consultant rather than the secretary.

The relationship between agents and Roles needs to be clarified at this point. Roles are abstract collections of logically coherent activities. To carry out these activities, an agent is required. There is an important distinction between Actions (which are carried out automatically by the PSS) and UserActions which require a human agent. At any time a role-instance 'points at' a particular user who is thereby responsible for carrying out the

UserActions (if any) associated with the Role. Each PSS user is represented in the PSS by a piece of software known as a User Agent which manages the user interface for the user and mediates between the user and their set of role instances. Role-User assignments may be varied dynamically allowing considerable latitude in the division of labour, e.g. when an employee reports in sick, their work can be simply redistributed by changing Role assignments. The reader should also note that as new software becomes available, more and more of the process can be transferred to the PSS (i.e. UserActions become internal Actions). By such means, PSS (in principle) allows the man-machine boundary to be drawn and redrawn with considerable flexibility and without the need for radical intervention

UserActions, as we have said, require a human agent. For Assess_Urgency, this real participant might be the consultant to whom the patient is referred or it might be a junior doctor or even a secretary to whom the task has been delegated, via the appropriate Role assignment. Once urgency has been assessed, the RAD shows control (conceptually) to return to *Outpatient-ing* with the next step being to make an appointment. Patient details are then passed (via an interaction) to the third role, that of *Doctor-ing.* The patient subsequently attends the clinic; *Doctor-ing* carries out its two actions, culminating in the transmission of the doctor's expert opinion back to *Outpatient-ing* which in turn uses this information to produce a clinic letter to send to the GP. One further action is shown on figure 2, Edit Record. It is shown separately because it may be performed at any time, i.e. not at a fixed position in the *Outpatient-ing* sequence.

RADs are simple high-level pictorial tools for representing and reasoning about processes. The basic constructs of the RAD (Roles, Actions and Interactions) correspond directly to the principal constructs of PML and the RAD could be used as the basis of a graphical interface to PML. It should be noted that, whilst RADs provide a natural and convenient way of expressing temporal order, PML uses a much more general and versatile mechanism for handling the coordination of actions with roles. Actions in PML are simply embedded in production-rule like schemata: for each act, its preconditions are specified (e.g. the existence of an urgency assessment as the precondition for making an appointment); whenever the preconditions are satisfied, the action is said to be *enabled*, i.e. it may be performed. It is important to recognise that the enabling of an action does not force its performance, discretion remains with the user.

3.4 Work in a PSS: the electronic medical office

Process modelling with PSS is not simply a conceptual exercise. Process models can be "executed" to create an "active work environment" for a community of users. They are blueprints which can be used to create a working PSS supporting the process described in the model. At the heart of a working PSS, as we have said, is a process control engine, or PCE. The PCE is able to "bring to life" process descriptions couched in PML. Working PSSs can be built on a range of hardware/software platforms; a UNIX$^{(TM)}$ version of the PCE, for instance, is available as well as a version for ICL's VME mainframe operating system. PSS has been designed exclusively with WIMP user interfaces in mind; Sun workstations and IBM compatible PCs (Windows-3) are currently supported.

We can gain an impression of work in a PSS by considering the user's view of process-supported work. As part of CISP we have used the outpatients process model to create a prototype of "the medical office of the future" on a network of SUN workstations.To avoid gender bias, it is assumed in the following narrative that all the actors are female. Figure 3 shows an imaginary scenario derived from the outpatients model of figure 2 which illustrates how the simplified outpatients process would look to the medical secretary. Having switched on her Sun workstation and logged on to the PSS, several windows have been projected onto the workstation display. Work with PSS, we have said, is structured as actions within roles. Correspondingly, we have two generic types of window in figure 3: the Role Agenda and the Action Agenda. The Role Agenda simply lists those roles containing actions that are currently enabled (triggered). For each active role, its Action Agenda lists these enabled actions and, optionally, those actions that have been completed (indicated by a tick).

Thus in figure 3, we see a Role Agenda for the secretary which lists two currently active roles, both instances of *outpatient-ing*, one for a patient called Wastell, the other for Smith. Apparently the secretary has chosen to work on Wastell; the appropriate action agenda is shown in centre screen. Two actions are portrayed, i.e. the two actions whose preconditions are currently satisfied.The secretary has apparently chosen Edit_record; the appropriate icon has been clicked and a window has been projected onto the screen displaying the patient's details. As work within a role proceeds, new actions become relevant and old ones redundant. This progression will be reflected in the appearance and disappearance of icons from the Action Agenda.

The reader is now asked to imagine the workstation of the secretary's consultant, who is working in her own office down the corridor. Let us assume the secretary requires advice about the case's urgency, and she clicks Refer_for_Urgency. At the consultant's end (we assume the consultant is holding this role), this action will cause an Assess_Urgency icon to appear in the appropriate action agenda on her screen. To enter her opinion, the consultant clicks this icon. And so on and so forth.

3.5 PSS and the problem of change

The inevitable need to change the structure of the work environment was a central and distinctive precept of the philosophy of IPSE 2.5. It was recognised that change is an inherent feature of all organisations. Thus PSS was designed to be flexible and adaptable. One simple example is the ability to make dynamic changes to role assignments. If a doctor is absent, for example, her work can be transferred to another staff-member simply by re-directing her role assignments to another user agent. If an extra secretary starts work, a new user agent and new instances of the various roles associated with secretarial work (e.g roles for outpatienting, sending mail, typing papers etc.) are simply instantiated.

More generally, the need to support change led directly to the incorporation within PML of a number of basic mechanisms which allow active processes to be changed "on the fly": e.g. PML primitives have been developed to allow new roles to be defined and extant roles to change their behaviour. These PML facilities provide the basic mechanisms enabling the dynamic change of working PSSs. They allow, for instance, evolutionary changes to be made to established processes or entirely new processes to be introduced without stopping the "model"; they allow different staff-members or work-groups to modify their own processes to support preferred ways of working (i.e. different specialisations of a generic role-class). For further technical details of these mechanisms, the reader is referred to Warboys (1990).

Higher level constructs have also been developed to support management and change. PMMS (Process Model for Management Support), for instance, is a generic process model of management. Management is defined as the "indirect work" of coordinating people and technology to achieve organisational goals. PMMS is a process model of management based around the idea of hierarchical delegation in which projects are instantiated (recursively), objectives defined, people assigned and logistical and technical support identified. PMMS is dynamic and evolutionary: new projects can be defined as they arise, and the role-interaction topology changed, again from within the running "model", as the organisation develops. Again for more details the reader is referred to Warboys (1990).

4. Current directions: the IOPT project

PSS provides both a modelling technique for representing cooperative activity and a means of creating working CSCW systems which are characterised by a high degree of flexibility. The Alvey funded IPSE 2.5 project completed its work at the end of 1989. Further funding has now been obtained on a significant scale from the U.K. government (the Information Engineering Directorate's Advanced Technology Programme) to support further development of the technology and to promote its industrial exploitation.

The IOPT project (Introduction of Process Technology) is a major collaborative project involving STL (now ICL), Praxis, ISS and the University of Manchester. It has a number of important aims. The first is the development of the technology itself, specifically to develop the base technology and enrich the process modelling language with higher level constructs for describing collaborative work. A second line of work is the development of an underlying Open Systems architecture for PSS with improved tool and database interfacing capabilities. Whilst this chapter is not primarily concerned with the engineering aspects of PSS, these technical developments have important implications and are thus briefly outlined.

4.1 Tool integration

There are at least two ways of looking at PSS. The first perspective, which is the preoccupation of this chapter, is to see PSS as technology for supporting group work (the CSCW perspective). The second viewpoint focusses on PSS as medium for integrating a wide range of computer artifacts and systems into an homogenous framework (the systems integration perspective). Organisations have made great investments in applications software and many powerful individual office tools are available. Whilst PSS provides some generic facilities, its aim is not to replace specialised tools. Rather, PSS should be seen as a framework for bringing together a wide variety of tools in an integrated environment supporting people in their individual tasks as well as supporting group coordination. The job of the PSS is thus not to replace tools but to put to hand the right tool at the right time (e.g. the appointment diary, the secretary's preferred word processor, the manager's usual spreadsheet etc.). Mechanisms called tool servers are integral parts of PSS allowing the incorporation of external (alien) tools and applications.

The vision in IOPT is that a PSS will not simply cover a local work group but that the PSS will subsume the entire organisation. In other words, the user, seating himself before any workstation in the organisation, should be able to carry on his work, under the benign supervision of the PSS, from wherever it was left off previously. To continue our medical example, when the doctor logs off the workstation in the ward office and returns to her base, she logs on and resumes exactly where she left off. Perhaps the workstation on the ward is an IBM PC connected to the hospital X25 wide area network and her office system is a SUN workstation connected to a local departmental ethernet. That should be quite immaterial. With these exacting technical requirements in mind, there is a major effort in IOPT to develop an open systems platform, based on X/OPEN standards (X/OPEN, 1990), in which interfacing capabilities can be developed for a wide range of office tools and databases across a computing infrastructure potentially characterised by many heterogeneous elements (hardware and software).

4.2 Designing PSSs

Another important area addressed by IOPT, which is a particular concern of the authors, is the development of a systematic approach (a methodology) to the introduction and design of process support environments. The conventional paradigm of systems development depicts systems development proceeding in a more or less linear and prescribable fashion from an analysis of requirements, to a precise specification, through technical design to delivery and acceptance testing of a definitive product. This is the so-called life-cycle (or waterfall) model and is manifest in one form or another in the structured methods which are now the backbone of current design practice. Hirschheim et al (1987) have dubbed this paradigm the Technical Perspective. From this perspective, systems (e.g. Office Information Systems, Management Information Systems, etc.) are seen as machine-like artifacts and their development as a rational, deterministic problem-solving activity. The structured methods might be dubbed Procrustean in memory of the mythical brigand, slain by Theseus, who had the habit of fastening his victims to an iron bed; those who were too short, he stretched and those who were too long, were truncated.

The technical perspective has considerable difficulty in dealing with the all too common phenomenon of system failure (Hirschheim and Newman, 1988), failure being defined as a significant gap between the expectations of users (managers, end-users) and the actual performance of the system (Lyytinen, 1988). Hirschheim et al (1987) have developed an alternative model of system development, which they call the Social Action perspective. This perspective sees system development as a social process characterised by purposeful interaction between designers and users. The success or otherwise of the project reflects "the accumulated quality of these episodes"; consensus formation, rather than precise specification, is seen as the turning point of the design process. The attainment of genuine consensus requires "free and fair debate" and is hindered by differences in knowledge, interests and power between designers, users and managers.

In a similar vein, Lyytinen (1987) has argued that system design is essentially "language action", that reality is actively constituted during design by designers and users as a shared knowledge structure that makes sense of organisational life. Their involvement in design obliges users to make explicit the 'hidden' structure of organisational life and in a real sense design deepens the understanding of both designers and users. Lyytinen (1988) describes the design process as "pluralistic, ambiguous and conflict-laden" and argues that soft systems methodologies, which help to structure ill-defined problems, therefore have an important part to play. Similar views of design as a fundamentally social activity have been advanced by other theorists who have put forward metaphors for design based on hermeneutics rather than the orthodox engineering paradigm (see Wastell, 1988; Wastell and Cronin, 1988).

A number of soft systems methodologies (SSMs) have been developed which share the common goal of helping people to deal with complex problems. SSMs embrace both a dialectical epistemology (subject and investigator are acknowledged to participate actively in the synthesis of knowledge) and a perspectivist ontology (no single objective reality is assumed). In common, the SSMs insist that the role of the analyst is to seek out diversity and to construct a so-called “rich picture” of the problem situation accommodating as many worldviews as possible. One well known SSM is Checkland’s soft systems methodology (Checkland, 1981) which Lyytinen (1988) discusses in the ISD context.

Another interesting approach is Colin Eden’s Strategic Options Development and Analysis (SODA) methodology (Eden 1989). As well as stressing the analytical skills of the investigator, Eden pays considerable attention to the social dynamics of the SODA process and the interpersonal skills required by the investigator. SODA provides a very useful formal notation (the cognitive map) for expressing the logical structure of complex problems and indeed some mathematical techniques for processing these structures (e.g. looking for a common structure across a range of representations). Eden is careful to stress that SODA is a group activity whose aim is not to produce a definitive solution but to achieve sufficient agreement that groups “feel confident to take action”. The SODA process involves three main stages: first the analyst conducts individual interviews, producing cognitive maps that articulate each protagonist’s worldview. These maps are then combined (as far as possible) to produce a merged map. A group workshop is then convened, in which the merged map is taken as the basis for the group discussion.

In IOPT, a design methodology is being sought that is work-oriented, participative and open-ended, and which sees consensus formation as the critical success factor in design. The framework provided by SODA thus provides a very promising paradigm. PSS is not intended to be authoritarian and controlling (although we are aware of the dangerous potential of the technology); the aim is to liberalise work by enabling greater individual freedom and control. The nature of the technology only makes sense, and can only achieve its intended benefits, if it is put in the hands of the users themselves. Central to our methodological work is the development of software tools to allow users, using a high-level graphical formalism, to display, design and redesign their local processes to suit evolving personal and group requirements.

4.3 Language-action based process modelling

We have referred to a number of formalisms for representing office work, e.g. Petri nets, RADs, dataflow diagrams etc. Clearly some formalisms capture more semantic complexity than others. In this section, we describe some of David Wastell’s most recent work on modelling which has been strongly influenced by a new perspective in systems design known as the language-action (LA) perspective). In the CSCW field, the LA perspective is associated particularly with the name of Terry Winograd (Winograd, 1988) and in the Information Systems field, with Kalle Lyytinen (Lyytinen,1987; Auramaki, Lyytinen and Lehtinen, 1988).

The Language-Action perspective presupposes a pluralistic view of human organisations in which order is maintained between conflicting internal interests by contracts which “define how individuals join together and co-ordinate their efforts” (Ciborra, 1984). From this perspective, “information systems are needed to set up, control, and maintain the organisation’s contracts”. Information systems are thus regarded as mediating communicative acts (rather than modelling reality in the orthodox view) which form elements in conversations that reflect the day-to-day working of contracts. The task of the analyst is to reconstruct, with users, the structure and rules of these “organisational conversations”. Auramaki et al. (1988) describe a complete methodology for designing OISs based on speech act theory which involves a number of formal techniques for expressing organisational conversations. Inspired by this work, we have developed an embellishment of the RAD, called rather prosaically the SARAD (Speech Act based Role Activity Diagrams), which allows processes to be described in terms of instrumental acts (actual deeds, such as typing a letter) and speech acts (which typically involve one role requesting a service of another, or transmitting information). Detailed treatment of LA modelling and SARADs is outside the scope of this chapter. The interested reader is referred to Wastell (1991).

LA modelling certainly adds greater semantic richness to process modelling. In speech act terms, the outpatients process is remodelled as a network of instrumental acts (e.g. creating an appointment) intertwined with and coordinated by speech acts (e.g. the referral letter is seen as a directive speech act, requesting an expert opinion). By revealing the conversational dimension of office work, the LA perspective allows a range of powerful new intellectual tools to be brought to bear. Auramaki *et al.* argue that "dysfunctional discourses" are a major cause of inefficient office work. The quality of organisational conversations can be analysed along two dimensions: coherence (that the conversation unfolds "logically", e.g. that questions are followed by relevant answers etc.) and discourse completeness. By uttering speech acts, people make *commitments* which bind their future conduct. The idea of commitment is an important one; commitments are the "glue" of cooperative activity. A complete discourse is one in which all commitments (e.g. to give an expert opinion) have been in one way or another resolved. Setting up and monitoring commitments, it will be recalled, is the key purpose of OISs from the LA perspective. Auramaki *et al.* provide a number of useful graphical tools for analysing discourses for coherence and completeness (commitment networks, reachability trees etc.).

4.4 Field studies of process technology and cooperative work

Field work is another important ingredient of the IOPT project. An IOPT club has been inaugurated, whose role is to promote the dissemination of process technology and to coordinate a programme of case studies in organisations. The case studies are a vital part of IOPT. They provide the opportunity to explore a range of practical and theoretical issues relating to the development of the technology and its exploitation. Health care is one area of field-work which, as we have seen, has helped further the development of process modelling techniques, and which has provided a test bed for exploring the potential of process technology. Other studies of a similar nature are underway in a range of organisations who have expressed interest in the technology and a willingness to be "guinea pigs".

4.4.1 The Customer Services case study

The case studies are important as they provide the opportunity to evaluate PSS under real world conditions, complementing our laboratory based work. One major implementation of PSS is presently under way in the Customer Services (CS) division of a large manufacturing concern. At the time of writing, PSS is being introduced into one area of CS as a pilot experiment. A full longitudinal evaluation is under way, which we are not yet able to report on. It is, however, instructive to record that PSS has the satisfied CS's functional requirements. The most important of these were the need to introduce PSS incrementally (in particular to incorporate existing databases used by other groups), the need to provide embedded links to other tools (e.g. to an expert system assisting fault diagnosis), to provide support for flexible working and to allow local variations of working procedures. The

Customer Services pilot is an important one. If successful then implementation will proceed to cover the whole of the CS operation, which involves several hundred staff spread across the U.K. at twenty or so sites. The "process" itself is a complex and differentiated. It has discernible stages and involves distinct groups of actors (call registration, first line support, maintenance engineers, second line support) each with specialised skills and roles. The process is geographically distributed and indeed many of the work groups are spread across several sites.

4.4.2. Office automation and routine work

Finally, we will comment on one important general issue for CSCW and Office Automation (OA) emerging from the field work in IOPT, the question of the nature of routine work. PSS provides support for *Processes*. By Process we refer to some routine way of carrying out a desired activity: a definable, generalisable procedure involving people and materials designed to accomplish a well-defined objective. The concept of Process is clearly associated with the larger concept of bureaucratic organisation, i.e. the use of explicit rules and procedures to co-ordinate the internal operations of the organisation (Morgan, 1986). The idea that routine office work involves following formal procedures is at the heart of PSS and its OA precursors. But how routine is routine?

In an important paper, Gasser (1986) has addressed the question of "routine" work. From his field observations, Gasser concludes it is anything but routine. It is quite wrong to regard routine work as the slavish following of bureaucratic rules and procedures. Much of so-called routine work is skilled problem-solving (see also Wynn, 1979). Offices are open systems; "anomaly is ubiquitous"; often the correct procedure is not mechanically obvious, indeed often procedures must be adapted to suit the exigencies of the situation. For these reasons, no formal description can ever completely cover the contingent, ever-varying character of real work. The unanticipated always occurs, requiring resourcefulness and ingenuity. There are always short cuts. "Every real world system requires articulation to deal with unanticipated contingencies: articulation resolves these inconsistencies by packaging a compromise that gets the job done...the results of this work appear as workarounds or kludges...." (Gasser, 1986). Gasser takes the "kludge" and the "workaround" not as aberrational but as fundamental to office work, evidence of its skilled nature, of "articulation".

4.4.3 "Routine" work in Customer Services

Early attempts to introduce PSS in CS met with fierce opposition. The reasons were not hard to find. The system was too prescriptive. "Your system has been designed to be idiotproof, the trouble is we're not idiots...", as one user drolly put it. PSS's raison-d'etre is to support well-defined processes and indeed work in CS is governed by clear and detailed work instructions (WI). For example, the procedure for handling customer calls by the diagnosticians who provide first-line support is crudely as follows. Several diagnosticians work in a team and they take calls from a common stack which is ordered chronologically within priority level. The rule is to take the top call. Having taken (assigned a call), the WI prescribe the following call handling process: check details of call- ring customer - note contact made- discuss with customer - note synopsis of problem - route to field service if necessary - suspend or close call. The key performance target is to make contact within 40 mins.

Let us now look at what happens in practice by examining a typical hour in the life of one of the diagnosticians, DD.

16.00: 3 calls on stack. DD looks at top one. It is a "call back" for John [i.e. a customer has been asked to try something; CS to ring back later] scheduled for 15:45. DD decides to take it. Rings customer. Fault has been sorted out. Closes call.

16.05: Looks at stack again. Again takes top call. Another call back for John. Leaves it.

16.08: Looks at next call: software problem. Leaves [for Ted]- s/w not his forte.

16.15: DD checks queue again. Notes one of his call-backs is due. Assigns it to himself to "block anyone else taking it".

16.17: Leaves the call-back and takes a new hardware call which has appeared from an Important Client. A laser printer problem. Checks diagnostic database and rings customer. Asks him to check toner and other simple causes. No luck- routes to field service

16.30: Again checks stack. 3 new faults, all from one customer- VDU problems. Assigns all three to himself and rings customer. Brief discussion- probable software fault. Closes 2 of the faults [i.e. subsumes the trio to one problem], reclassifies as a software fault, adds narrative, and routes to second line support.

16.48: DD checks stack again. Takes his call-back[40 min deadline imminent] even though a "system down" has appeared [top priority]. Rings customer. Still having problems. Customer is exasperated- lots of problems recently. DD thinks Account Manager (AM) should be informed, but no mechanism. John suggests asking field service to alert AM. DD routes to field service with an appropriate message.

What do we see in the above scenario? The idea that DD simply follows procedures is patently absurd. Putting the procedures into action is clearly skilled work. At every turn judgement and articulation are on show; doing the job well also requires considerable informal knowledge and coordination. In one case, there is a genuine problem over what to do and a workaround is used. On at least four other occasions, procedure is "violated": e.g. when the software call is "handed off", when the "Important Client" (informal knowledge) is given priority, and when DD comes back to his "call-back" ahead of the "system down" call. In all these cases, DD, far from pleading guilty, would argue - probably vehemently - that he is working consistently with his interpretation of procedure, "in the spirit of the law", and in the best interests of efficiency. He knows, for instance, that Ted is better at software problems, and he knows that Ted knows he knows, so he knows he can safely leave the s/w call on the stack and that Ted will pick it up (informal coordination).

5. Conclusion

Oh, chestnut tree, great-rooted blossomer,
Are you the leaf, the blossom or the bole?
Oh body swayed to music. Oh brightening glance,
How can we know the dancer from the dance?
(W.B. Yeats, Among School Children)

PSS is an interesting CSCW technology with a number of notable features. The concept of the executable process model (i.e. process enactment), the emphasis on tool integration and support for change are significant and innovatory ideas which represent a significant step forward over its precursors especially in the OA field. PSS offers the promise of supporting cooperative work across many domains. However, its application in field studies has raised a number of profound questions about the nature of routine work in offices and the role of computer systems for supporting cooperative work.

With OA technologies, PSS shares a fundamental belief. This belief is important because it is ubiquitous in Computer Science. It is the belief in Formal Representation, more specifically that human activities (e.g. office work) can be formally described using some sort of graphical notation (RADs, Petri nets, dataflow diagrams) and that these diagrams are in some sense complete, they capture all that needs to be said; therefore they form a satisfactory and sufficient basis for designing computer systems. But what our field work suggests is that these descriptions fail to capture what is most essential about office work, namely its contingent, problem solving character. Gasser has observed that the designer's view of office-work is often naively mechanistic, indeed conventional design practices characteristically eschew detail (logicalisation) basing design on idealised abstractions of office work. These become reified in inflexible and obstructive office systems. Wynn (1979) makes a similar point from her ethnographical studies of office work.Our field-work strongly accords with these views.

What then is the relationship between the formal view and actual work practices? Understanding this relationship is clearly crucial. Formal procedures are normative: they define how work ought to be done. In some cases, formal rules may be rigidly enforced and slavishly followed but in general the relationship between procedure and practice is not a simple deterministic one: the application of rules requires interpretation and negotiation. Indeed, working out how to apply the rules, keeping them, as it were, in good repair is a key part of office work. In a real sense, people make the rules work, rather than the other way round (an inversion of Taylorism!). Suchman (1983) states "the procedural structure of organisational activities is the product of the orderly work of the office, rather than the reflection of some enduring structure that stands behind that work". Suchman may rather overstate matters in implying office procedures to be irrelevant. The procedure manual does provide a frame of reference for work. But this is not to say it determines what people do in mechanistic way; we all know that "working to rule" is not a byword for efficiency! In our view, the relation between rules and practice is dialectical: formal procedures shape but ultimately do not determine practice; practical experience in turn can lead to re-interpretation or reformulation of procedures as contradictions emerge.

Our field studies, such as the Customer Services case study, confirm Gasser's and Suchman's view of the skilled, problem-solving nature of apparently routine work and the dangers of being over-prescriptive. Although much of the work in CS is apparently prescribed in detailed work instructions, application of these rules requires considerable interpretation and discretion; sometimes judgements are wrong, requiring backtracking and the exploration of new approaches; often official procedures appear to be violated, but invariably in the interests of efficiency. To repeat our earlier saw, "People make the rules work".

The obvious implication of this is that co-ordination systems such as those based on PSS need to be extremely flexible. The recognition that offices are open systems motivated the progenitors of PSS to produce a technology on which it was possible to build flexible and adaptable co-ordination systems. PSS is not a simple tool like Co-Ordinator; it is a system designed to be extensively tailored to the specific situation in which it is used. It works by enacting a model of the processes occurring in that situation. The actual systems built using it can be rigid or flexible, depending on the models created by the designers of a particular system. There is thus a great need for a process design method and tools to aid the designers in the production of appropriate models, and as indicated above this is being addressed in the IOPT project.

References

Auramaki, E, Lehtinen E and Lyytinen, K. (1988). A speech-act based office modelling approach. *ACM Trans. Office Inf. Sys*., 6,126-152.

Checkland, P. (1981). *Systems thinking, systems practice.* Wiley: Chichester

Ciborra, C.U. (1984). Management information systems: from decision-making to exchange. In Bemelmans (Ed.), *Beyond productivity: ISD for organisational effectiveness.* North Holland: Amsterdam

Eden, C. (1989). Using cognitive mapping for strategic options development and analysis. In: Rosenhead, J. (Ed), *Rational Analysis for a problematic world,* Wiley: Chichester.

Ellis, C.A. and Nutt, G. (1988). Office Information Systems and Computer Science. In: Greif, I. (ed.), *Computer Supported Cooperative Work.* Morgan Kaufmann: California.

Goldberg, A. (1984). *Smalltalk-80: The Interactive Programming Environment*. Addison-Wesley: London.

Greif, I. (1988). *Computer Supported Cooperative Work*. Morgan Kaufmann: California.

Hirschheim, R. A.(1985). *Office automation: a social and organisational perspective*. Wiley: Chichester.

Hirschheim, R.A., Klein, H. and Newman, M. (1987). A social action perspective of information systems development. In: DeGross and Kriebel, (Eds) *Proc. of the 8th International Conference on Information Systems*. ACM: New York

Hirschheim R. and Newman, M. (1988) Information systems and user resistance: theory and practice. *Computer Journal,* 31, 398-408.

Lyytinen, K (1987) Two views of information modelling. *Information and Management,* 12, 9-19.

Lyytinen, K. (1988) Stakeholders, information system failures and soft systems methodology: an assessment. *J. Applied Systems Analysis*, 15, 61-81.

Maresh,J. and Wastell, D.G. (1990). Process modelling and CSCW: an application of IPSE technology to medical office work. In Diaper, D. et al (Eds.) *Proc. of INTERACT90*. North Holland: Amsterdam.

Morgan, G.(1986) *Images of Organization.* Sage: London

Olson, M and Lucas, H (1982). The impact of office automation on the organisation. Comm. of the ACM, 25.

Pava, C. (1983). *Managing new office technology: an organistaional strategy.* Fress Press: New York.

Sluizer, S.and P. Cashman. (1984). XCP: an Experimental Tool for Supporting Office Procedures. In: *Proceedings of the 1st International Conference on Office Automation.* IEEE: New York.

Smith,H., Hennessy,P. and Lunt, G. (1991). The Activity Model Environment: an Object-Oriented Framework for Describing Organisational Communication. In: Bowers, J. and Benford, S. (Eds), *Studies in Computer Supported Cooperative Work*. North Holland: Amsterdam

Suchman, L (1983). *Office procedures as practical action*. ACM Trans. Office Informations Systems, 1, 320-328.

Warboys, B (1990). The IPSE 2.5 project: a process model based architecture. In: Bennet, K. (Ed.), *Software Engineering environments: research and practice.* Wiley: New York.

Warboys, B. and Veasey, P. (1989). Twenty years with support environments. *ICL Technical Journal*, 6, 447-466.

Wastell, D.G.(1988) Phenomenology and participation: alternative principles for IS development. *Lecture Notes in Medical Informatics*, 35, 717-722.

Wastell, D.G. (1991). Process support technology, cooperative work and information system development. *Proceedings of the 11th International Conference on Information Systems,* ICIS, in press.

Wastell, D.G., Acheson, J., Cotter, L., Schady, W. and Lucas, S. (1987). Computing in clinical departments: implications for the design of hospital information systems. *Health Policy*, 8, 347-354.

Wastell, D.G. and Cronin, E. (1988). Soft systems methodologies in the design of information systems: insights from hermeneutics and Habermas's theory of communication. In: *Computers in Clinical Medicine,* British Medical Informatics Society: London.

Winograd, T. A Language/Action Perspective on the Design of Cooperative Work.In: Greif, I. (ed.), *Computer Supported Cooperative Work.* Morgan Kaufmann: California.

Wynn., E. (1979). *Office Conversation as an Information Medium.* Ph.D thesis, Univ. of Calif, Berkeley

X/Open, (1989). *X/Open portability guide.* Prentice-Hall: Englewood-Cliffs.

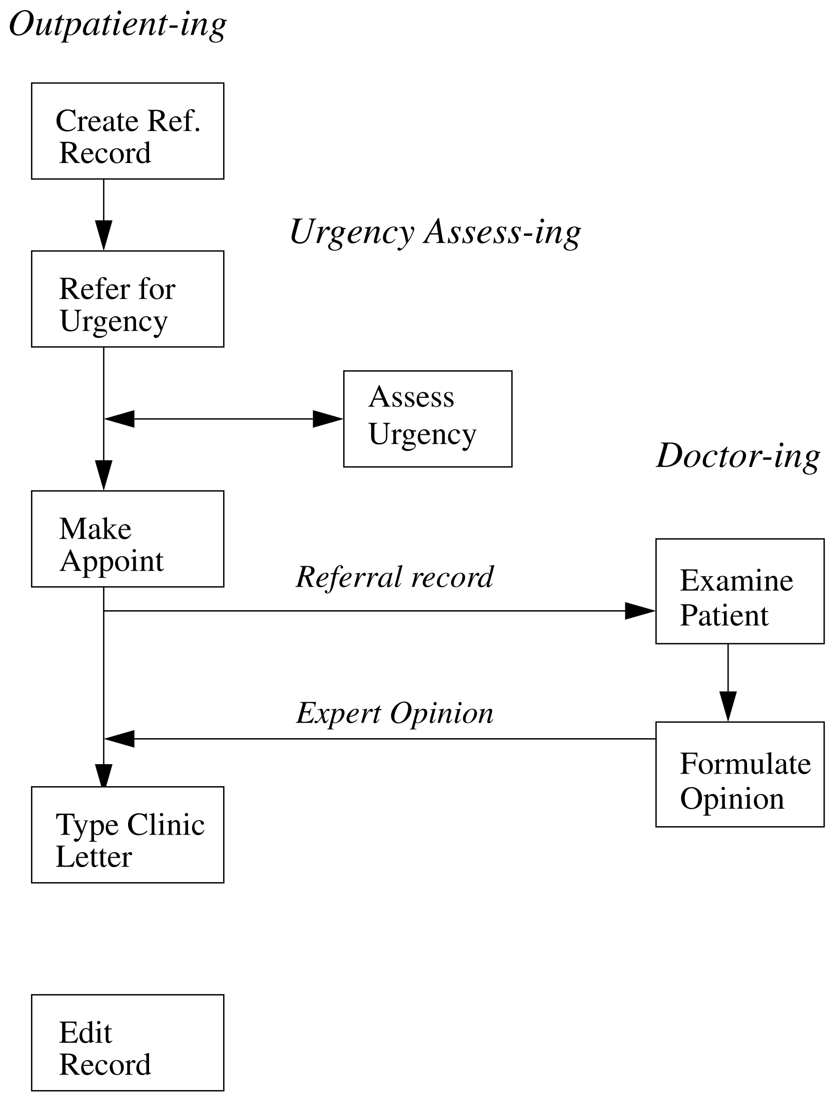

Figure 2. A simplified view of the outpatients process showing three roles.