# Cooperative Information Systems: A Manifesto[*]

**Giorgio De Michelis[1], Eric Dubois[2], Matthias Jarke[3], Florian Matthes[4], John Mylopoulos[5], Mike Papazoglou[6], Klaus Pohl[7], Joachim Schmidt[8], Carson Woo[9], Eric Yu[10]**

## Abstract

*Information systems technology, computer-supported cooperative work practice, and organizational modeling and planning theories have evolved with only accidental contact to each other.* ***Cooperative information systems*** *is a relatively young research area which tries to systematically investigate the synergies between these research fields, driven by the observation that* ***change management*** *is the central issue facing all three areas today and that all three fields have indeed developed rather similar strategies to cope with change.*

*In this paper, we therefore propose a framework which views cooperative information systems as composed from three interrelated facets, viz. the system facet, the group collaboration facet, and the organizational facet. We present an overview of these facets, emphasizing strategies they have developed over the past few years to accommodate change. We also discuss the propagation of change across the facets, and sketch a basic software architecture intended to support the rapid construction and evolution of cooperative information systems on top of existing organizational and technical legacy.*

## 1. Introduction

*Cooperative Information Systems* is a relatively young research area whose birth in the early 1990's has been marked by the launching of an international journal, an on-going conference, an international foundation, dedicated special issues in international journals, numerous meetings and workshops held over the past five years. Given all this activity, we believe that it is time to take a closer look at the area, its premises, primary research challenges and prospects. Also, to better delimit its boundaries, separating topics it covers from ones it does not, thereby offering an identity relative to other more established areas in Computer Science.

Taking past calls-for-papers for the CoopIS conference (e.g., [CoopIS94]) as starting point in identifying Cooperative Information Systems, we find them being described primarily as "next generation information systems":

---

[*] Appeared in: *Cooperative Information Systems: Trends & Directions*, Mike P. Papazoglou and Gunter Schlageter (eds), Aacdemic-Press, 1997.

[1] **Address:** Dipartimento di Scienze dell' Informazione, Universita degli Studi di Milano, Via Comelico 39, 20135 Milano,Italy; **email:** gdemich.hermes@hermes.mc.dsi.unimi.it

[2] **Address:** Institut d' Informatique, Facultes Universitaires de Namur, 21, rue Grandgagnage, B-5000 Namur, Belgium; **e-mail:** du@info.fundp.ac.be

[3] **Address:** RWTH Aachen, Informatik V, Ahornstrasse 55, 52072 Aachen, Germany; **email:** jarke@informatik.rwth-aachen.de

[4] **Address:** Fachbereich Informatik, Universitaet Hamburg, Vogt-Koelln Strasse 30, D-22527 Hamburg 54, Germany; **e-mail:** matthes@dbis1.informatik.uni-hamburg.de

[5] **Address:** Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Canada M5S 3H5; **e-mail:** jm@cs.toronto.edu

[6] **Address:** Tilburg University, INFOLAB, Warandelaan 2, 5037 AB Tilburg, The Netherlands; **email:** M.P.Papazoglou@kub.nl

[7] **Address:** RWTH Aachen, Informatik V, Ahornstrasse 55, 52072 Aachen, Germany; **email:** pohl@informatik.rwth-aachen.de

[8] **Address:** Fachbereich Informatik, Universitaet Hamburg, Vogt-Koelln Strasse 30, D-22527 Hamburg 54, Germany; **e-mail:** J_Schmidt@dbis1.informatik.uni-hamburg.de

[9] **Address:** Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, Canada V6T 1Z2; **e-mail:** carson.woo@ubc.ca

[10] **Address:** Faculty of Information Studies, University of Toronto, Toronto, Canada M5S 1A5; **e-mail:** eric@fis.utoronto.ca

"...The paradigm for the next generation of information systems will involve large numbers of information systems distributed over large, complex computer/communication networks. This paradigm ranges from the vast and visionary Electronic Superhighway, to the large and complex billing system of a telephone company, an even to the small patient information system in a one-doctor office. Such information systems will manage or have access to large amounts of information and computing services. They will support individual or collaborative human work. Computation will be conducted concurrently over the network by software systems that range from conventional to advanced application systems including expert systems, and multiagent planning systems. Information and services will be available in many forms through legacy and new information repositories that support a host of information services. Communication among component systems will be done in a centralized or distributed fashion, using communication protocols that range from conventional ones to those based on distributed AI. We call such next generation information systems *cooperative information systems*...." [CoopIS94]

Described in such terms, Cooperative Information Systems poses a series of technological challenges which arise primarily because of technological advances (in telecommunications, hardware, software etc.). In contrast, the CSCW area focuses on the opportunities offered by such technological advances to broaden team collaboration across boundaries of place and time, and on the question how this will change human inter-actions [Roseman96]. Yet others argue that research should be driven primarily from a shift in (business) organizational structures away from traditional functional forms towards goal- and customer-oriented processes [Scott-Morton91, Keen91, Hammer93, Hamel94]. However, across all these perspectives, it seems that continuous *change* is the one constant theme in all of them. Information technology -- used by collaborating groups or formal business organizations -- is faced with the task of either learning to cope with change or risk early retirement.

Each of the mentioned areas has addressed only a limited facet of what is needed to understand and realize cooperative information systems. This paper articulates a vision produced by researchers participating in a collaborative effort between European and Canadian universities, to create an integrated framework for research on cooperative information systems which addresses the mutual impact of continuous change in the system-oriented, group collaboration, and organizational facets of cooperative information systems.

Section 2 of the paper links the notion of cooperation to that of organizational change. In Section 3, we offer a framework to view and understand cooperation and change in cooperative information systems from three different facets. Sections 4 to 6 describe research challenges arise from each of the facets and survey promising research directions and on-going projects working towards meeting these challenges. Section 7 ties all three facets together and outlines the impact of change from one facet to the others. The section also discusses research challenges facing the management of change and some possible solutions for them. Given the materials we presented in Sections 3 to 7, we propose a generic architecture for cooperative information systems in Section 8. The final section summarizes our vision of cooperative information systems.

## 2. Cooperation and Change

In distributed systems and cooperative computing, the word "cooperation" has a neutral meaning. *Cooperation* presupposes agents that have goals and can act upon them. Moreover, *cooperation among agents* entails that these agents have some common goals and act towards their fulfillment. More generally, an agent is *cooperative* if she/he/it tends to share goals with other agents in its environment and acts towards the fulfillment of these common goals.

What does "cooperation" have to do with information systems? Information systems can be thought of as collections of human or computerized agents which can carry out actions such as printing a report or requesting information. Moreover, the functional and non-functional requirements of an information system, intended to describe the “purpose” of the system, can be treated as its “goals”.

When is then an information system cooperative? Consistent with our earlier definition, an information system is cooperative if it shares goals with other agents in its environment, such as other information systems, human agents and the organization itself, and contributes positively towards the fulfillment of these common goals. Cooperation with other information systems presupposes the ability to exchange information and to make a system's own functionality available to other systems. These features are often referred to as *interoperability* in the literature and should be treated as prerequisites to cooperation.

However, cooperation in organizational systems and business processes is more complex than sharing and interoperability. Management and Organizational studies, as well as the studies carried on in the Computer Supported Cooperative Work (CSCW) field, focus on cooperation as the basic means through which groups and organizations, while performing for their clients, continuously redesign themselves and their business processes, modify their boundaries, tune their objectives, and open themselves to new possibilities [Argyris78, Argyris96, Nonaka95]. In this framework, cooperation is not a neutral characteristics of human and automated agents. Rather, it focuses on the basic *inter-actions* between the human members of a group and/or an organization. In this sense, cooperation offers a linguistic category for characterizing the process through which they perform together, and together change the organizational structure to which they belong.

The growing complexity of contemporary societies does not allow us to reduce this process to a matter of sharing information, rules, and goals. This is because information, rules, and goals are created and modified in the same process. Different methods have been proposed in the managerial literature to deal with this complexity (e.g., business process reengineering, organizational learning, empowerment, knowledge workers and professionals, and network organizations). Despite their relevant differences, all of these methods agree on the necessity of enhancing performance effectiveness, learning capability, and communication competence of individuals and groups.

Supporting cooperation, therefore, requires the system to be capable of reflecting both the changes that are decided for its performances and the continuously ongoing changes of the practices of its members. The problem then is not how to build information systems which share goals with their organizational environment, human users, and other existing systems at *some time point.* Rather, the problem is how to build information systems which *continue* to share goals with their organizational environment, human users, and other existing systems as they all *evolve*. **In short, it is *continuous* organizational and technological *change* that makes cooperative information systems a challenge.**

## 3. A Framework for Research in Cooperative Information Systems

This concept of on-going cooperation and evolution leads to a new vision for information systems. In particular, the paradigm entails that the significance of a cooperative information system lies not in its tangible ingredients such as its hardware or software legacy systems. Rather, it lies in the system’s ability to contribute to some goals in a larger social, organizational, and technical context. An information system, under the new paradigm, is defined more by its connections and relationships with the outside world than by its internal technological make-up, or even its stand-alone functionalities and capabilities. The identity of an information system can persist over time even as its underlying hardware and software components are continually reconfigured or replaced to adapt to the needs of a changing external environment. A cooperative information system is not a collection of databases, applications, and interfaces. Rather, it is an architectural framework which maintains consistency among a variety of computer-based systems, user groups, and organizational objectives as they all evolve over time.

We, therefore, envision that research problems and issues in cooperative information systems arise from three areas of concern -- systems, group collaboration, and organization -- and the interactions among them

(Figure 1). We call these the three *facets* of cooperative information systems. There has been considerable research focusing on addressing issues in each of these facets. However, in order to provide viable cooperative information system solutions, all three facets must be taken into account and dealt with in a coherent way.

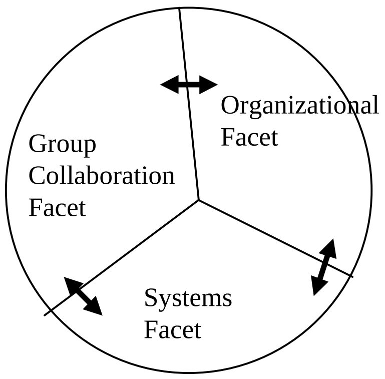

**Figure 1:** The Three Facets of Cooperative Information Systems

The *systems facet* includes various types of existing information, workflow and other computer-based systems, developed in terms of conventional technologies such as programming languages, DBMS and workflow systems which are executing on conventional, distributed hardware and software. Typically, these systems were developed in a bottom-up fashion by individuals, groups or even the organization itself and were intended to serve local and often less-than-permanent needs. For cooperation to function within the systems facet, one needs to deal with issues of the heterogeneity of, and thus the incompatibility among systems. Much work has been done under the rubric of system integration, addressing interoperation concerns like data transfer, semantic and control integration. This is to ensure that all systems within an organization (and sometimes across different organizations) can share data and use each others' functionality, independently of the platform on which they were developed or the one on which they are executing, also independently of their original purpose or origin. The problem that this facet poses to the group collaboration facet is that once built, systems tend to become inflexible, and are not easily adapted to rapidly changing work situations. The systems facet also poses problems to the organizational facet in that traditional system development efforts are costly, involve long lead times, and are hard to manage.

The *group collaboration* facet is concerned with how people working on a common business process or other project -- such as co-authoring a report -- can coordinate their activities, can deal with contingencies, and can change their practices through discussions and learning, as though they were working physically together. The nature and style of work may vary and are often not predictable beforehand. Much progress has been made to address problems in this facet in terms of groupware, organizational computing, and other tools. The problem posed to the systems facet is that group collaboration requires a high degree of flexibility and malleability in the systems that support the work. The open-ended, fluid nature of group collaboration is also in constant tension with the more stable, formal, and pre-planned nature of organizational structuring.

The *organizational* facet is concerned with managing work from a formal organizational perspective, regardless of by whom (the group collaboration facet) or with what technology (the systems facet). It addresses global organizational concerns, including organizational objectives and business goals, policies, regulations, and resulting workflow or project plans. This facet has been addressed using models of organizational processes and entities (as in traditional systems analysis), and more recently, models of business rules and policies, and goals and interdependencies among organizational agents. The organizational facet is where system requirements and specifications typically originate. However, subsequent refinements usually take into account group collaboration issues and the constraints and capabilities of the supporting technologies.

This framework is useful for identifying the origins and impacts of change, and thus provides a way to characterize research problems and issues in cooperative information systems. Change can originate in any of the three facets. From the systems facet, a change entails the introduction of new systems that need to be integrated into an existing cooperative information system architecture. Such changes need to be propagated within the systems facet (how can the given system interoperate with other systems within the cooperative information system architecture?); with the group collaboration facet (how can it contribute to human collaborative processes?); and with the organizational facet (how can the new system enhance the achievement of organizational objectives?). Likewise, changes may originate from the group collaboration facet (new business processes, changes in team membership) or from the organizational facet (new objectives, different prioritization of existing objectives). For each of these, the framework suggests a set of questions that need to be answered, such as “How should a change of type X be propagated from one facet to another?” Just as importantly, each facet has its own laws of inertia: for the systems facet, it is legacy systems and the lack of interoperability; for the people facet, it is people wanting to continue doing things the way they were before; for the organizational facet, it is unwillingness to risk change. To deal effectively with change, one has to establish the change vision in a given technical, social, and organizational context [Jarke93], or, as [Ehn88] put it, to achieve transcendence while recognizing tradition. We shall return to this question after looking at each individual facet.

## 4. The Systems Facet

While other facets are concerned primarily with the "why", "what" and "who" questions behind cooperative work, the main activities of the systems facet is to specify *how* information is to be processed making best use of existing systems. As will become clear in the sequel, constant organizational change at other facets leads to a strong demand for system solutions where the answers to the "where" and "when" of information processing are delayed as long as possible (leading to *persistence* and *mobility* requirements).

The typical answer to the "how" question provided by traditional Information Systems and Databases research has been to provide *generic systems* (DBMS, fourth-generation languages, repositories, etc.) and *expressive data models* (relational, deductive, object-oriented, etc.) to improve the productivity of the information system construction process [Atkinson90, Stonebraker90]. Information system construction is often perceived as a software development process centered around a growing, centralized corporate database to meet the requirements of a specific business task (such as billing, inventory control, or accounting).

The vision of cooperative information systems expands this traditional notion of an information system along two important dimensions: Firstly, it emphasizes the role such an information system takes within a broader context (intra- or cross-organization cooperation) by focusing on its system interfaces to other remote information systems in heterogeneous, distributed networks (*coordination* and *interoperation*). Secondly, it emphasizes the role such an information system plays in the entire lifecycle of an organization by focusing on the need for continuous change to accommodate changing organizational demands but also drastic changes in the underlying system technologies (*change management* from a systems perspective).

As a consequence of this broader view, which is driven by recent technological advances such as ubiquitous computing and communication, the system solutions that emerge are based on the decentralized paradigm. Technologies for building repositories of network-linked objects, and mechanisms for partitioning applications and architectures are emerging to provide network interfaces upon which to construct large-scale distributed applications out of pre-existing, heterogeneous and autonomous components. These consist typically of advanced application building blocks and widely accessible information services, available to applications developers and users. Such architectures exhibit an increased autonomy of change and lead to subsystems with multiple lateral interfaces supporting peer-to-peer cooperation which extend beyond simple (database-centric) client/server configurations.

This shift in system architectures does not imply that the classical database system requirements and modeling requirements can simply be abandoned. Consequently, *system support* for longevity of data, bulk data management and data consistency and integrity continue to be of high relevance for cooperative information systems. However, system cooperation now imposes a series of new requirements concerning the modeling and representation of higher level functions which include partitioning, placement, and migration of applications into distributed components of communications, computation, and storage and the allocation of system resources to their execution. Moreover, better mechanisms are needed for the support of long-term business processes that rely on coordination of semi-autonomous legacy systems and applications, system evolution and change management. Some of these issues will be highlighted in the following. In particular, we concentrate on two highly inter-related system dimensions: high-level interoperation and coordination.

### 4.1 Improved Interoperation

In this section, we identify some of the main technical opportunities, relating to cooperative information systems, which can lead to improved system interoperation and graceful system evolution.

#### 4.1.1 Abstraction From the Underlying System Platforms

This abstraction can be achieved by utilizing problem-oriented, high-level languages and models available uniformly on multiple platforms. Software in standardized, high-level languages (like SQL or ODBC) has the potential to absorb changes in the computing environment (hardware, operating system, system libraries) and to narrow the semantic gap between organizational models and implementation models. Recently, one can also perceive an increased interest in *run-time portability* which goes beyond standard compile-time application portability by enabling active applications (including their bindings to data objects and external resources like windows) to migrate between multiple computers (e.g., connected via the
Internet). Interpreted high-level script languages like Tcl, Java, Obliq, Tycoon or Telescript are particularly well-suited to implement such migratory applications [Cardelli94], mobile code [Java95], migrating and persistent threads [Matthes94] or network agents [Wayner94, White94].

#### 4.1.2 Exploitation of Generic Servers

As they are identified, common services -- such as persistent data storage, graphical interaction with human users, authentication, network communication, transaction monitoring and workflow management -- are usually factored out from individual applications, to be realized using generic off-the-shelf tools such as relational databases, GUI toolkits, RPC services, transaction monitors or workflow engines. Such middleware components [Bernstein93] not only simplify and speed up changes of existing applications but also lift the level of abstraction available for the interoperation between independently developed systems. For example, relational schemata, RPC service descriptions and high-level GUI event specifications often provide a valuable starting point for the interoperation between information systems.

However, when these generic server abstractions are developed over time, the architectural challenge arises: how to make the different abstractions fit together. For example, while Visual Basic nicely integrates programming language and user interface concerns, the interaction with databases via ODBC involves a major paradigm shift which makes information systems implementation in this setting difficult. Integrated database programming languages [Schmidt77] or, more recently, object-oriented databases, solve some of these mismatches, but at the expense of openness and sharing with other kinds of systems. Recent advances in higher-order programming, exemplified in the research arena by Tycoon and in commercial database programming by novel APIs, as well as light-weight interface standards, such as HTML, seem to offer some promise in finally resolving these decade-old issues [Berners-Lee96].

#### 4.1.3 Compartmentalized Applications

Contrary to the architectural rationale of centralized information systems -- for example, economies of scale, high level of system consistency, full data integration, division of labour between users, and IT department -- the rationale of cooperative information systems is to favor small, light-weight, *modular* components and applications that are linked directly to individual organizational needs, goals, and structures. As a consequence of this approach, the cost and time required for the initial construction and the long-term maintenance of these applications and of the gateways between these applications can be attributed much more directly to specific business objectives and business processes with positive consequences for project management and business process re-engineering.

It is interesting to note that this *accountability* argument is partially in conflict with the vision of grand unifying system frameworks (*switchboard architectures*). For instance, distributed object systems such as DSOM of IBM [Lau94], DOM of GTE [Manola92], CORBA of OMG [OMG91] and OSF DCE/DME [OSF93] on one hand provide ideal component-ware frameworks, however, they also require a heavy investment to set up and to maintain a common, corporate-wide system infrastructure.

### 4.2 Liberated Coordination

A severe limitation of traditional information systems is their rigid control model. Essentially, users of an information system are limited to a fixed set of hard-wired transactions or transaction sequences and there is little, if any, support for human intervention or "intelligent" exception handling. To overcome these limitations, the following themes are particularly relevant for Cooperative Information Systems.

#### 4.2.1 Factoring Out Control From Individual Information Systems

Obviously, atomic transactions executed in isolation against a centralized database are not an appropriate mechanism for the coordination of long-term, cooperative human work. As a consequence, there has been significant database research to define and implement richer *synchronization* and *recovery* models (nested transactions, sagas, etc.). There is also a growing interest from database researchers in *coordination* models developed in the area of CSCW [Alonso96] and in organizational modeling (business process modeling). Similar to the idea of TP monitors that factor out details of multi-database transaction processing from individual information systems, workflow management systems and active database systems promise to factor out application control logic from individual applications into high-level workflow definitions or declarative rule specifications [Hsu93, Georgakopoulos95].

This general theme of eliminating control from applications can also be found in recent developments on the desktop application market. For example, typical standalone Microsoft products (Word, Excel, Access) are currently being enriched with *scripting capabilities* with the long-term vision of having a common scripting language (VisualBasic) to coordinate and control integrated application systems that are composed of several of these products [Microsoft94]. In light of these developments, it should be mentioned that the shipping of scripts (code mobility) or autonomous agents (thread mobility) may lead to interesting new modes of cooperation and coordination in future distributed information systems [Mathiske95].

#### 4.2.2 Integration of Human and System Communication

Advanced e-mail software and novel WWW applications, as well as commercial applications based on groupware platforms like Lotus Notes and Novell Groupwise demonstrate the *synergy* that can be obtained from the integration of desktop and *communication* tools (like e-mail readers, group calendar managers, telephone and fax systems, bulletin boards, video conferencing, hypertext browsers) with *database* functionality (information retrieval and boolean query functionality, replication, scripting support). More generally, cooperative information systems will have to learn from past research and development in CSCW to make best use of both, human and system resources ("agents") to achieve a given business goal. Ideally, such integrated systems support a smooth *transition* between *ad-hoc cooperative work* of humans (e.g., for problem solving and exception handling) and standardized, *automated interaction* between autonomous information systems (e.g., via EDI messages or workflow management software).

### 4.3 Maintenance of Links Between Implementation and Model

Database systems provide explicit information about the database schema itself via *metadata* -- schema representations at several levels of abstraction, statistics, access rights, stored procedures, query plans, etc. Such information is of particular interest for change management and interoperation tools that have maintain links to other information system components and to higher-level data models (schema translators, re-engineering tools, gateway generators, IDL stub generators, etc.).

Similarly, cooperative information systems have to maintain *explicit links* between individual information system components (applications, agents, transactions, schemata, workflows, etc.) and related model components (business objects, business objectives, design decisions etc.) of the organizational facet [Jarke88] (see Section 6). These links are not only crucial to propagate organizational changes rapidly to the supporting information systems but also to provide a feedback on the organization's performance in terms of the business objectives. For example, it becomes possible to monitor the time and (human) resources consumed by standard business processes that span multiple information systems or to track the probability of exceptions (for instance, because of human interventions) that occur during a certain process step.

Ultimately, cooperative information systems should be able to inspect and modify their own behaviour in the course of their long-term operation in an organization. Such *reflective* systems can be built either by using behavioural reflection or linguistic reflection. With *behavioural reflection,* a system can alter its own behaviour by manipulating its evaluator. One way to achieve this is for an interpreted language to allow access to the internal structures of the interpreter itself at run-time. This form of reflection can be found in variants of Lisp, Prolog or special-purpose research prototypes. In linguistic reflective systems, systems can change themselves directly, rather than the mechanisms of changing a system's run-time behaviour supported by behavioural reflection. A reflective system may, for example, alter the data structure that represents the system itself, alter the compiled code that is being executed or generate new data structures to be interpreted or new code to be executed. These options offer different tradeoffs between flexibility, execution efficiency, and assurance of system consistency which have been studied, for example, in the context of database programming languages [Stemple91].

## 4.4 Information Agents

To achieve cooperative information processing, we must overcome the barriers to building and deploying mission-critical information systems from reusable software components. This can be achieved by assembling information services on demand from a montage of networked legacy applications and information sources. However, it requires locating, re-using, combining, processing and organizing already existing chunks of information and application software in such a way that does not impact detrimentally on an organization's operation. In the previous we have briefly outlined some of the requirements, tools and technical opportunities regarding this issue. A promising approach to this problem is to provide access to a large number of information sources by organizing them into a collection of *information agents* [Papazoglou92, Knoblock94]. The goal of each agent is to provide information and expertise on a specific topic by drawing on information from already existing information sources and other such information agents. Information agent capabilities typically include interprocess communication mechanisms and services, such as naming, translation, information discovery, syntactic/semantic-reconciliation, partial integration, distributed query processing and transaction management and a variety of other services that can be shared between diverse incompatible information systems.

In the following we will briefly discuss the key components of such information agents.

### 4.4.1 Agent Organization

Each information agent is another information source, however, it draws on already existing information repositories and applications based on organizational and business model components and provides an abstraction of these in the form of a *wrapper*. A wrapper is the appropriate interface code (including translation facilities) which allows an existing information source to be transformed into an information agent while allowing it to conform to the conventions of an organization. These wrapper characteristics are largely influenced by the requirements engineering phase of the organizational facet. The wrapper specifies services that can be invoked on data objects(originating from various sources) by completely hiding implementation details. Thus it provides the basis for building new network-based applications by mapping services into a collection of host information systems and applications. The advantage of this approach is that it promotes conceptual simplicity and language transparency. This technological advancement can be achieved, for example, by harnessing the emerging distributed object management technology and by appropriately compartmentalizing existing software and applications.

### 4.4.2 Information Brokering

One of the most important functions of an information agent is to find the appropriate information sources and support services (i.e., wrapper enabled services) in order to fulfill an organizational requirement or a user's needs. To achieve this purpose an agent must have a model of its own domain of expertise *the agent domain model* and a model of the other agents that can provide relevant *information the agent awareness model.* These two models constitute the *knowledge model* of an agent and are used to determine how to process an information request.

Currently there is widespread interest in using *ontologies* as a basis for modeling an agent's domain model [Wiederhold94, Milliner95, Kashyap97]. An ontology can be defined as a linguistic representation of a conceptualization of some domain of knowledge. This ontology consists of abstract descriptions of classes of objects in the domain, relationships between these classes, terminology descriptions and other domain specific information and establishes a common vocabulary for interacting with an information agent and its underlying information sources. The information sources in the network describe both their contents and their relationship in accordance with an ontology [Milliner95]. Hence an ontology can be viewed as some form of a knowledge representation scheme.

The domain model does not need to contain a complete description of the other agents capabilities, but rather only those portions which may directly relevant when handling a request that can not be serviced locally. Each agent is specialized to single area of expertise and provides access to the available information sources in that domain. The objects and relationships in an ontology do not correspond to the objects described in a particular information source, but rather provides a semantic description of that domain for interaction. Each information source may relate to many information agents. This results in the formations of clusters (or groups) of information sources around domains of expertise handled by their respective information agents. This approach provides conceptual simplicity, enhances scalability and makes interactions in a large collection of information sources become tractable.

Both domain and awareness models may be expressed in some concept representation language such as Loom [MacGregor90], or KL-ONE [Brachman85] and be formalized by using concept lattices [Wille92] (a formal method based on a set-theoretical model for concepts and conceptual hierarchies). An example of the use of concept oriented languages for building ontologies can be found in [Kashyap97].

### 4.4.3 Agent Communication Languages and Protocols

In order to perform their tasks effectively, information agents depend heavily on expressive communication with other agents, not only to perform requests, but also to propagate their information capabilities. Moreover, since an information agent is an autonomous entity, it must negotiatiate with other agents to gain access to other sources and capabilities. The process of negotiation can be stateful and may consist of a “conversation sequence”, where multiple messages are exchanged according to some prescribed protocol.

To enable the expressive communication and negotiation required, a number of research efforts have concentrated on knowledge sharing techniques [Patil92]. To organize communications between agents a language that contains brokering performatives can be particularly useful. For example, the Knowledge Query and Manipulation Language (KQML) [Finin95] can be used to to allow information agents to assert interests in information services, advertise their own services, and explicitly delegate tasks or requests for assistance from other agents. KQML's brokering performatives provide the basic message types that can be combined to implement a variety of agent communication protocols. This type of language can be used for both communication and negotiation purposes and provides the basis for developing a variety of inter-agent communications protocols that enable information agents to collectively cooperate in sharing information.

### 4.4.4 Self-Representation Abilities

One of the most challenging problems to be addressed by cooperative information systems is the development of a methodology that opens up the general process of constructing and managing objects based on dispersed and pre-existing networked information sources. Activities such as object integration, message forking to multiple object subcomponents, scheduling, locking and transaction management are until now totally hidden and performed in an ad hoc manner depending on the application demands. What is needed is the ability to work with abstractions that express naturally and directly system aspects and then combine these into a final meaningful implementation.

Such ideas can benefit tremendously from techniques found in reflection and metaobject protocols. The concepts behind metaobject protocols may be usefully transferred to cooperating systems. Core cooperative tasks, carried out in an ad-hoc manner up to now, can be performed using metalevel facilities. These can be used to separate implementation from domain representation concerns and to reveal the former in a modifiable and natural way [Edmond97]. Therefore, information agents may provide a general coordination and policy framework for application construction and management, one that allows particular policies to be developed and applied in a controlled manner. This results in self-describing, dynamic and reconfigurable agents that facilitate the composition (specification and implementation) of large-scale distributed applications, by drawing upon (and possibly specializing) the functionality of already existing information sources.

### 4.4.5 Application Development

The goal of information agents is to create a collaborative client/server business object environment. The basis for application interoperability can be realized across applications by means of cooperating business and other data objects. Business objects provide a natural way for describing application-independent business concepts such as customers, orders, billing, invoices, payment and so on. Rather than relying on a monolithic application cooperative information systems applications will be consist of suite of cooperating business objects [Brodie97]. These should be able to communicate with each other at a semantic level and encapsulate the storage, metadata, concurrency and business rules associated with the specific business entity they describe. Such business rules and objectives are related to the manner that organizational objectives are modeled and are covered in the organizational facet. These objects can then be linked together via scripts written in a variety of scripting languages which are needed to describe and execute arbitrary tasks. This makes it possible to design new kind of collaborative applications that can be controlled through scripts and semantic events. Semantic events provide a form of messaging that component objects can use to dynamically request services and information from each other.

Scripting is a critical element for allowing developers to automate the interaction of several business object applications to contribute to the accomplishment of a particular common task. This can be perceived as a form of automated workflow, a core aspect of the next generation of business applications. Scripting technology is essential for agents, workflows, and cooperative (long-lived) transactions. Scripting languages provide added flexibility as they use late binding mechanisms and can, thus, attach scripts dynamically to component (business) objects. Accordingly, they provide the means for creating ad-hoc (dynamic) collaborations between components which is the very essense of cooperative systems.

## 5. The Group Collaboration Facet

In order to effectively support group collaboration, we must take the following two main tasks into consideration. Firstly, analysis of work practice [DeMichelis95, DeMichelis96] shows that we must ensure that a group of people, working together on a common business process or on another project can communicate with each other through the medium they want, coordinate their activities, get the timely information they need wherever it is, and deal with any contingencies that may occur, independent of time and space distances. Secondly, we must be able to locate the same group of people in the (organizational) context of the process to which they participate, so that the information characterizing it is transparent and/or visible to them whenever they need it, and their work space is shaped by it, disregarding any other information about any other process which is not relevant at that moment.

In order to explain the above more clearly, the group collaboration facet describes cooperative information systems within a spatial metaphor. In traditional cooperation, people are either together or not together in space and time. Cooperative information systems, on the other hand, shape the (virtual) space where people cooperate on a work process, in a much more flexible way. Several spatial metaphors, including document replication (Lotus Notes), electronic circulation folders [Karbe91, Prinz96], team rooms [Roseman96], or work spaces [Appelt96] have been used in this context.

Since the environment where users do their work and cooperate with each other can be considered as the virtual space inhabited by the users, a cooperative information system can, in fact, be considered as an electronic extension of the physical space inhabited by the users. The physical space of a group is, in general, a distributed space. Inhabitants moving in it can either be in the same place or in different places. If they are in different places, they can communicate either synchronously or asynchronously.

Within the group collaboration context, users can be in the same or different virtual spaces. If they are in the same space, they are sharing the work space. If they are doing things outside the collaboration environment or in a different part of it, then they are in a no sharing work space situation. However, organizational workers might go through different modes of cooperation when performing an organizational task. Cooperative information systems should, therefore, also support the movements between shared work spaces and distinct (i.e., not shared) work spaces, as well as between synchronous and asynchronous communication.

Research surveyed in this section originates in Computer Supported Cooperative Work (CSCW), an interdisciplinary research area based on the idea that complex work processes are generally cooperative processes, where people distributed in time and space interact through electronic media. CSCW systems (called groupware and workgroup computing systems in existing products) are systems supporting cooperation between human beings.

## 5.1 The Main Features of the Group Collaboration Facet

The group collaboration facet of a Cooperative Information System is a means to extend the physical space in which a group of people cooperate, to transform it into a share work space. The most interesting and challenging situation for the use of a cooperative information system is the one where the group is not co-located in a physical place.

There are two types of share work spaces -- synchronous and asynchronous. A synchronous work space provides the necessary mechanisms for agents at a distance to interact as if they were face-to-face. Thus, at any point in time, all agents sharing the work space can view and update the same work space concurrently. For example, a video conferencing system creates a synchronous work space for its users where they can see each other and they can point to the same object on the screen. An asynchronous work space, on the other hand, is a work space such that at any point in time, only one agent has access to the work space and only this agent can view and modify what it contains. For example, the channel through which a file folder flows from one person to another is an asynchronous work space.

It is important to consider how the two different share work spaces support their users, in different ways, the awareness of their working context.

### 5.1.1 Synchronous Sharing

The facilitation of synchronous sharing has raised a number of research challenges, of which we will only discuss two.

Firstly, there are interesting cognitive and behavioral considerations when designing a system for a group of distributed users [Mantei91]. It is impossible to provide an identical face-to-face environment for them because there are too many happenings in a meeting. For example, in the face-to-face meeting everyone may notice that while Joe hands a piece of paper to Chris, Ken's feet are shaking. From this happening everyone may infer that Ken is nervous. Which of these happenings are absolutely necessary for the purpose of the meeting? For example, is it necessary in a distributed environment to let everyone in the meeting know that Joe is sending a note to Chris? If showing body movements (including shaking) is important, then how are we going to support it?

Secondly, there are many technical and behavioral issues related to the development of user interface for real-time sharing [Greenberg94, Munson94]. For example, how can two persons avoid modifying the same sentence of a document they are co-editing at the same time? Or, how are results to be merged, if users concurrently modified different parts of a document.

In both cases, it is evident that synchronous sharing offers a very effective means to support the mutual awareness of the actors of what they are doing, while different strategies for handling information sharing are embedded in the different solutions offered to address the first research challenge mentioned above.

### 5.1.2 Asynchronous Sharing

Asynchronous sharing means that people are in different locations and their interactions are not in real-time. This type of interaction has been with us for a long time. For example, a government officer writes a letter to a taxpayer, notifying her that she did not pay enough income tax. The taxpayer writes back with additional evidence on why she does not need to pay more taxes. The government officer then sends her an additional form to be filled. The taxpayer fills the form and sends it back. Since these two persons are using regular postal mail (i.e., an asynchronous communication medium), each subsequent letter sent by the taxpayer needs to identify the subject matter and the government officer needs to keep a file of the case in order for him to follow up with it.

It should be clear from the above example that knowing the correct context (e.g., missing income tax payment) in asynchronous interactions is very important. If we use computers to support this type of interaction, contextual information will form a major component of the share work space, and we need a representation for it. Similar to any other knowledge representation problem, such representation needs to be simple but still captures the essence of each context. A common approach in providing semantics for interactions is the language/action perspective [Winograd86, Flores88], which is based on the speech act theory originally developed in Linguistics. The fundamental idea behind this theory is that for each possible action, there exists a verb to describe it, and these verbs belong to a small number of speech act categories each of which defines a different pragmatical relation between the conversing persons – using the technical terms in Linguistics, each of these speech act categories has a clear and distinct illocutionary point. Most computer-based systems using this approach require users to identify a speech act category for each message they are composing. Many authors demonstrate that it is impossible to be sure that different persons can agree the specification and interpretation of the speech acts they communicate. Suchman, in particular, attributed these difficulties to the unnaturalness of constraining human interactions [Suchman94].

A different approach has been used in the Conversation Handler of the Milano system [DeMichelis94]. It allows users to bring forth activity templates on which they agree, without forcing them to reduce their utterances to well defined speech acts. We believe that the language/action approach is promising but needs to accommodate additional contextual information to be usable. In the Milano System [Agostini95], for example, each message is linked both to the previous messages of the conversation to which it belongs and to the results of the activities negotiated within it. Moreover, it includes all the relevant documents to identify its context. An alternative solution proposed by Janson and Woo [Janson96] is to use a double loop learning to aid the specification and interpretation (e.g., if things are unclear, use a sequence of speech acts exchanged previously to specify or to interpret the correct meaning).

### 5.1.3 Managing the Switches

In an environment where organizational goals change frequently and the environments to which persons can interact are open-ended, it is possible that an originally isolated actor now needs to collaborate with some other persons – to share with them a work space. It is also possible that two persons communicating asynchronously now need to meet face-to-face (i.e., communicating synchronously).  In other words, the mode of communication can change. Reder and Schwab [Reder90] show that these switches between communication media are very frequent.

Electronic work spaces can assist these switches if there is a common standard for creating and using them, or if they share with all the other work spaces the appropriate contextual information. These issues have been studied at a coarse grain of cooperation in information systems for cooperative design (e.g., [Jarke92]), but the current challenge is to provide light-weight awareness solutions for continuous cooperation [Gutwin96, Palfreyman96].

The common standard is like a protocol that separates exchanging information from its processing. It can also be viewed as a translator between two incompatible software agents. For example, if agent X needs to share something with agent Y, then the following translation happens:

agent X's representation → work space representation → agent Y's representation

This will facilitate previously developed applications (e.g., legacy systems) or agents not conforming to the standard (e.g., e-mail systems) to be able to share. What will be a sufficient standard for the work space is still a research question. In OASIS [Martens97], for example, the work space consists of a set of forms where a form can either provide information or state an unaccomplished goal.

When the persons involved in a cooperative process continuously switch from one type of share work space to another, it is necessary that, in all of these work spaces, they can access the information characterizing the past history of the cooperative process. This is the contextual information mentioned above. In Milano [Agostini97], for example, this information is recorded in a partial order list linking together the e-mail conversation, their associated documents, and the activites performed within a common work process. Similar solutions have also been developed for traceability of cooperative software processes involving multiple heterogeneous tools [Pohl96].

In the following, we list four different types of switchings and their corresponding problems to solve or supports needed by the computer:

- from distinct work space to share work space (both synchronous and asynchronous): need to determine what information to make available within the work space for sharing and then translate, if necessary, from the individual representations to the work space representation.
- from share work space (both synchronous and asynchronous) to distinct work space: need to determine who gets what information.
- from asynchronous sharing to synchronous sharing: the task-at-hand here is very similar to that of setting up a meeting.
- from synchronous sharing to asynchronous sharing: the task here is akin to the continuation of work after a meeting.

It is important to note that the switching effort between different types of work spaces, from the user’s point of view, should be as effective as the continuous modification of the same work space.

## 5.2 Linkages Between Group Collaboration and Organizational Model

[Suchman84] points out that people tend to act in a situated manner and employ organizations and plans just as tools which they may or may not use. Thus, there is a continuous interplay between the work practice of the group collaboration facet and the organizational facet. The continuously changing work space where a group cooperates, is populated by the objects supporting the various communication media as well as the objects making accessible the context of the ongoing collaboration. Both types of objects are instances of corresponding objects in the organizational model, where they are stored with their specific access rights. Cooperation is performed, therefore, instantiating some objects from the organizational model. Conversely, to address problems, group collaborators may modify the objects in the organizational model, either at the level of the data stored in them, or at the level of the objects themselves, whenever an activity results into a modification of a role definition, of a procedure definition, etc. (Figure 2).

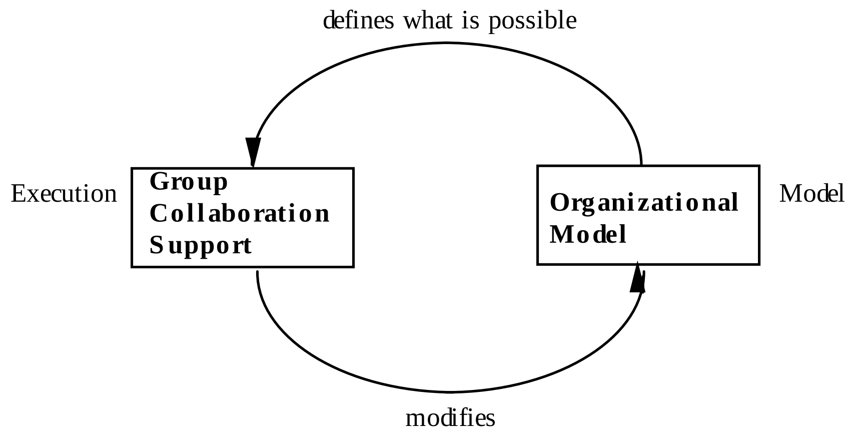

**Figure 2:** Linkages between group collaboration support and organizational model.

Using the spatial metaphor, cooperative information systems emerge as complex systems where legacy systems are artifacts in the virtual work space of their users, while the cooperation support templates are objects stored in the distributed architecture. While the user work space is characterized by the history of cooperation within which users are active at any given moment, the legacy systems are characterized by the information they have accumulated and the communication protocols they support. Cooperative Information Systems have the potential -- and the ambition -- to build systems which neither reduce human work spaces to information systems nor information systems to human work spaces.

The group collaboration facet is subject to radical changes with the emergence and the diffusion of the Internet as the basis for the creation of computer support for cooperative work. The first experiences of Intranet applications as well as the first research projects in this field (such as the European CoopWWW project offering a basic cooperation support infrastructure on the Web [Appelt96]) offer some hints on the type of evolution we may foresee.

## 6. The Organizational Facet

While the group collaboration facet can be best understood from a spatial point of view, the organizational facet can be understood from a linguistic perspective -- its objects characterize the lexicon (or, ontology) through which the members of an organization conceive and speak of their work (goals, roles, structures, tools, resources, procedures, ...).

The notion of cooperation in the delivery of information services within an organization cannot be fully addressed unless the goals and desires of organizational agents *and those of the organization itself* are taken into account. Agents cooperate when they share goals and work together to fulfill those goals. Computer systems are cooperative to the extent that they contribute to the goals of human and organizational agents.

In traditional systems development, the identification of organizational objectives and the determination of system features and characteristics to meet those goals is usually carried out under the title of "requirements analysis". It is well recognized that defining what software system is to be built before designing and implementing it, is critical to the success of the software system. Another area that has addressed the problem of capturing and accommodating organizational objectives and goals is that of Enterprise Integration, where research has focused on the integration of production and administrative processes, often in the context of a manufacturing organization.

As we move towards a vision of cooperative information systems, the task of requirements engineering remains crucial, although a number of important adjustments and extensions will be needed. Firstly, both forward and reverse engineering will be required. In traditional software systems development, as well as in Enterprise Integration, an entire system/enterprise is typically designed from scratch. Under the cooperative information systems paradigm, cooperative information system development does not necessarily proceed uniformly from requirements to design to implementation. There will already be implemented system components which are integrated to some degree (within the Systems Facet) for some purpose (Organizational Facet). It is, therefore, important to support both forward and reverse requirements engineering, i.e., from an existing situation to the set of objectives which account for it, and from a new set of objectives to a new cooperative information system.

Secondly, requirements models for a cooperative information system under the new paradigm will potentially cover a much broader scope, since a cooperative information system typically brings together many systems (which may be existing or to be developed). Explicit models that describe what a system is supposed to do and the organizational rationales behind them become even more important since they are important knowledge assets in their own right for the organization, regardless of how they are implemented technologically. As the system evolves (whether to take advantage of technological advances, or to address changing organizational needs), this facet provides, on an ongoing basis, the linkages needed to ensure that the technical systems are meeting organizational goals.

Thirdly, requirements models will have an active role to play throughout the life of a system. Under the traditional paradigm of information system development, requirements definition is usually seen as a phase or step through which one has to go through in order to arrive at the final product -- the operational system. Requirements typically appear in a static, printed form, often serving as a contractual document among different stakeholders (users, customers, developers, etc.). Once a system is operational, the requirements do not have an active role to play. Under the emerging paradigm, due to ongoing evolution, organizational objectives and systems requirements must be "kept alive" and remain a part of the running system (e.g., in the form of a knowledge base or repository). Otherwise, one cannot ensure that the system(s) are cooperative on an continuing basis. System specifications and requirements are also needed to provide context for interpreting system behaviour.

Within this section we briefly review three areas which can potentially contribute to the Organizational facet: Enterprise Integration, Early Requirements, and Systems Requirements.

## 6.1 Enterprise Integration

Enterprise integration is a research area in its own right, focusing on methods, tools and concepts for introducing technology within an organization to improve the dissemination of information, the coordination of decisions and the management of actions; for a recent and thorough account of the field, see [Vernadat96]. Generally speaking, enterprise integration is accomplished through a conceptual model of an enterprise which accounts at varying levels of detail and from a variety of perspectives the inner working of the enterprise. In addition, different modeling frameworks come with a variety of tools for creating, editting, and analyzing an enterprise model.

Enterprise modeling is generally tackled by offering notations for modeling a variety of organizational information, including organizational structure, business processes and activities, people, resources, tools and expertise, (existing) information system structure and functionality, information sources, their format and their contents. An *enterprise reference model* provides a data dictionary (ontology) of concepts that are common across different organizations, such as products, materials, personnel, orders, departments and the like.

Many reference models have been proposed in the literature and there is on-going work to create industry-wide standards in Europe, North America, and the Far East [Bernus96]. Some of the references models mentioned most frequently are:

- **CAM-I** -- US-based non-profit group of industrial organizations, striving to establish manufacturing software and modeling standards [Berliner88];
- **CIM-OSA** -- reference model developed within the AMICE ESPRIT project, used widely in Europe [Vernadat84];

- **ARIS** -- developed by August-Wilhelm Scheer at the University of Saarland, can be seen as the conceptual counterpart to the highly successful SAP standard software architecture [Scheer94];
- **PERA -- The Purdue Reference Model** -- developed at the University of Purdue by Theodore Williams, focuses on manufacturing [Williams92].

The CIM-OSA reference model and methodology (CIM-OSA stands for Computer-Integrated Manufacturing -- Open System Architecture) provides a three-level architecture to describe a manufacturing enterprise in terms of requirements, design, and implementation. In addition, CIM-OSA offers a reference architecture, from which particular architectures may be derived for a specific manufacturing process. This architecture was designed so as to support and encourage modularity, abstraction and open endedness. Currently, these efforts are being followed up by the GERAM attempt to unify the various existing enterprise models, whereas many details are being worked out in numerous domain-specific modeling effort (e.g., STEP in the production industry, PI-STEP in the process industries, and many others). In some ways, these architectures are similar to the one proposed here, but seem to have a top-down, monolithic flavour which is contrary to the spirit of this manifesto's view of cooperative information systems.

## 6.2 Modelling Organizational Objectives

In order to relate system functions and behaviours to organizational objectives (and thus to understand the "whys" behind system requirements), a cooperative information system must have suitable representations or models of organizations, and mechanisms for supporting reasoning about them [Bubenko80, Wieringa96].

### 6.2.1 Representing and Using Knowledge about Organizational Objectives

Traditionally, organizational objectives are treated only during the earliest stages of system development (e.g., during "systems planning"). Although these objectives are used to determine system requirements, there are seldom systematic links from the requirements (or other system models) back to organizational or business objectives. Under the cooperative information systems paradigm, these links (both forward and reverse) will be crucial for ensuring that technical systems continue to meet organizational objectives as they all evolve.

One important obstacle to bringing business objectives into an overall information system architecture is the lack of formal representations for this kind of knowledge. The difficulty in drawing conclusions from such informal bodies of recorded knowledge substantially weakens the cost-benefits of maintaining the knowledge. To understand cooperative systems environments, we need to manage and draw conclusions from large amounts of knowledge covering the business rationales that cut across many systems and business domains.

Recent research in requirements engineering has developed techniques that can potentially contribute to addressing these issues. The explicit representation of goals and their use in deriving requirements (e.g., [Feather87, Dardenne93, Chung93]) is an important technique that can be applied at the level of business objectives. Business reasoning, however, frequently require more flexible forms of reasoning beyond those supported by traditional techniques (e.g., classical problem-solving techniques in AI). Variations of goal-based reasoning (e.g., [Lee92]), with additional support such as qualitative reasoning and the concept of satisficing have been developed (e.g., [Mylopoulos92, Chung93]).

### 6.2.2 Dealing With the Organizational Dimension of Cooperative Information Systems

Adopting a cooperative information systems perspective places special demands on the knowledge representation framework for modelling business objectives and rationales for information systems. Cooperative systems are not merely distributed systems --- they are also organizational. They consist of "agents" who relate to each other as a social organization. They share responsibilities, have commitments to each other, and may have common or different values and beliefs. While they may cooperate at one level, they can also have competing or conflicting interests at another.

Research on organizational issues of computing (e.g., [Kling82, Gasser91]) has contributed significantly to the understanding of the embedding of computer information systems in organizations. Frameworks that emphasize organization modelling have started to emerge. For example, the Action Workflow model of [Medina-Mora92] highlights relationships among customers and performers. The framework of [Bubenko93] uses a multi-model approach to link enterprise objectives to system requirements. The *i** framework [Yu95] introduces the concept of intentional dependency for modelling strategic relationships among organizational actors (the Strategic Dependency model). Means-ends reasoning is used to help capture rationales and support reasoning about business objectives and alternative solutions (the Strategic Rationale model).

## 6.3 Modelling Systems Requirements

Systems requirements describe the observable external behaviour of a system (the "what") rather than internal details about its implementation (the "how"). In the context of cooperative systems where there are multiple heterogeneous agents (software/hardware, humans, devices, etc), it should make clear the responsibilities associated with each individual agent in relation with the "whats" (i.e., what each of them will guarantee).

### 6.3.1 Representing Systems Requirements for Cooperative Information Systems

At the systems requirements level, we need to determine the system features that will make the achievement of the organizational goals possible. Typically, at that level, these system features will be characterized in terms of the "environment" of the system (i.e., the portion of the real-world whose current behaviour is unsatisfactory in some way). Systems requirements will be expressed by describing how the system is connected to the environment in such a way that the behaviour of the environment will be satisfactory. For example, requirements inherent to a Library system will be expressed in terms of data and information (e.g., borrowed books, books on shelves, and reservations) manipulated by different actors (e.g., the librarian and the users) belonging to the environment.

Systems requirements should not be confused with the specification of the system itself. The specification of the system belongs to the system facet (i.e., it describes the system by adopting an internal perspective). At this level, a conceptual representation of the system is usually produced in terms of various object classes: active controllers, domain entities (mirroring the problem domain objects), and interface objects. This conceptual representation can be expressed by using semi-formal methods (e.g., OMT, Coad-Yourdon, Fusion, Objectory) or by using more formal methods (e.g., RML [Greenspan86], GIST [Feather87], Oblog [Sernadas91], and LCM [Wieringa94]).

Requirements languages that can be used at the systems requirements level, on the other hand, should offer the following two facilities:

- The language should offer mechanisms (action/state perception, action/state information) to distinguish the boundary between the system and its environment, as well as the responsibilities associated with each of them. This possibility of making clear the distinction is advocated by several authors. For example, Zave and Jackson recommend to make a distinction between "indicative requirements" (statements about the environment) and "optative" requirements (statements about the system) [Jackson95]. An adequate language should, therefore, provide (i) an ontology including an appropriate concept of *agent*, (ii) mechanisms for expressing different actions and responsibilities associated with their control, and (iii) communication mechanisms for describing time-varying information and perception.

- The language should support a *declarative* style of specification which supports a natural mapping (i.e., without having to introduce any overspecification) of stakeholders' informal statements in terms of their formal counterpart. This is due to the possibility offered by the language to model requirements adopting a *God's eye view perspective*, i.e., to express requirements by considering the whole admissible life (the sequence of states of the system) rather than adopting the usual constructive (operational) style where the value of a state at a given moment is computed from the subsequence of past states. For example, we want to express easily a statement like "the borrowing of a book should be followed by its return within the three next weeks". This *naturalness* property is essential for the purpose of traceability, i.e., to keep a close relationship between the informal requirements and their formal reformulation.

In terms of semi-formal notations, we can say that many existing OORA (Object-Oriented Requirements Analysis) methods are trying to incorporate these aspects in their suggested methodology. Describing the role of the system from the external users' perspectives is typically the goal of the *use case* originally introduced in Objectory [Jacobson92] and also incorporated in the new Unified Modelling Language UML. A similar attempt exists in the ROOM [Selic94] where *scenarios* are used for capturing the environment behaviour in presence of the system.

In terms of formal notations, recent proposals are the agent-oriented extension of GIST [Feather87] and the ALBERT language [DuBois97]. Besides the existence of precise rules of interpretation, it is expected that the mathematical/logical foundations underlying these languages will permit the development of a new generation of tools supporting:

- the verification of requirements produced by the analyst such as the detection of inconsistencies (e.g., leading to deadlocks) and incompletenesses (e.g., failure to preclude undesirable system behaviours), as well as the possibility of proving global desired goals at the system requirements level;
- the validation of the requirements against customer expectations by, for example, reformulating the formal requirements for easier interpretation (through natural language generation, or translation to some existing, less formal, notations), or by producing animations (or simulations) to allow customers to explore different possible behaviours of the system, and to check against use cases and scenarios.

### 6.3.2 An Expanded Systems Requirements Process

The cooperative information systems paradigm implies the need to expand the systems requirements process, since systems requirements must now be linked extensively to organizational objectives on one side (e.g., [Yu95b]), and to system designs and implementations on the other (e.g., [Chung95]). Requirements activities may proceed in forward and reverse engineering directions, and may have different scopes at different facets (business unit boundaries need not coincide with system boundaries, and vice versa). These activities may include:

- identifying the "why", i.e., business objectives and rationales for the new information system (e.g., the improvement of the management of the borrowings in the library in order to make it more attractive);
- identifying the agents in the environment (e.g., the librarian and the users), together with some assumptions and domain knowledge related to their behavior [Jackson95] (e.g., users are issuing books requests to the librarian and, when available, the librarian removes these books from the shelves);
- mapping business goals to system goals, i.e., goals related to the desired global behavior of the system [Dubois89, Dardenne93] (e.g., to prevent the user from keeping more than three books at once);
- "operationalizing" these system goals by reducing them into constraints that agents can be responsible for through their actions [Feather87, vanLamsweerde95] (e.g., the system will raise an alarm when the maximum number of loans allowed is exceeded and the librarian will not give the requested book to the user);
- deciding if the identified agents and their associated responsibilities will correspond to:
  - well-described tasks (or procedures). This will be usually the case for the software part of the system. The description of these tasks is the objective of the system specification activity (which is part of the *Systems facet*)
  - loosely defined tasks. The objective here is to provide high-level guidance and leave the decision of execution details at run-time. This is often the case when human agents are involved in the task performance (which is part of the *Group Collaboration facet*).

## 7. Change Management

The distinguishing feature of a cooperative information system is that it must ensure *continued* cooperation between the system, the group collaboration, and the organization facet. This implies that we must identify where changes come from in each facet, how they propagate, and what problems are involved in this.

Based on this analysis, facilities for change management can be discussed which are used to ensure that changes introduced at any facet of a cooperative information system are propagated to all other facets. The importance of change management has been recognized by architectures such as DAIDA [Jarke92a], CIM-OSA [AMICE89] or ARIS [Scheer94] through the inter-linking of requirements, design, and implementation levels. It is perhaps characteristic of the difficulties of cooperative information systems evolution that commercial environments have not really defined the links in depth, even though some research prototypes, such as the DAIDA and ITHACA [Constantopoulos95] environments, have experimented with a richer set of dependency types.

When developing a framework for change management, we follow the famous paper by [Ackoff67] to distinguish between *active* change management (planned change), *reactive* change management (unplanned change), and *pro-active* change management (making a system fit for easy reactive change).

A basic framework for continuous change management is shown in figure 3. The approach shown therein for *active* change management (planned changes) is borrowed from the so-called Carroll cycle, a method which has proven useful in the design of user interfaces [Carroll91]. In each such cycle, the current reality is reverse-engineered into models to uncover its underlying rationale. Then, a change is defined on the models, and implemented into a new reality, integrating the legacy of the existing context.

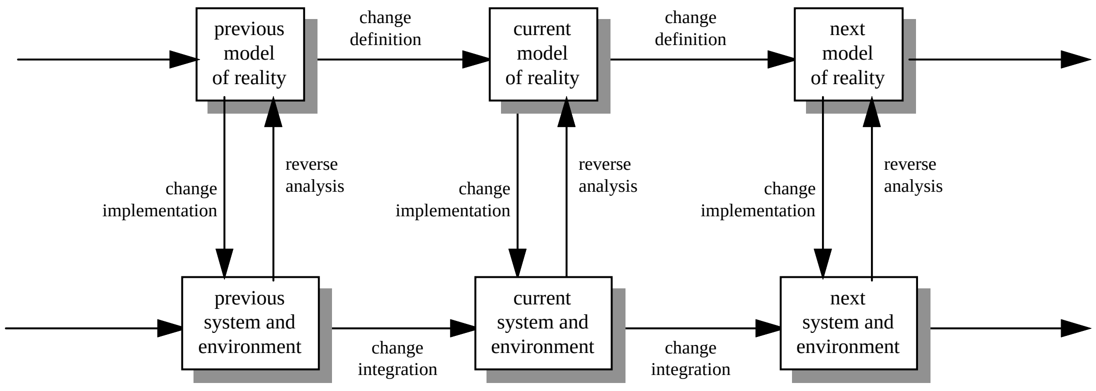

**Figure 3:** The change process for cooperative information systems

Besides these planned changes, there can also be unplanned changes where reality is changed directly, without resorting to any models. Change management then becomes *reactive*, to reverse-analyse such changes into models, thus analyzing their consequences and identifying possible follow-up changes. But purely reactive change management will not work. [Ackoff67] has pointed out that information systems should be designed to cope with such unplanned changes, i.e., measures should be taken to make them *pro-active*, not just reactive. This has, in fact, been the goal in each facet discussed in sections 4 through 6, but additional measures for pro-activeness must be taken at the level of change management across facets.

In section 7.1, we discuss active, reactive, and pro-active changes in more detail. The continuous tension between the envisioned change and the perseverance and autonomous activity of existing context has been identified as the main driver for requirements engineering which was therefore defined as “the process of establishing a change vision in the technical, cognitive, and social context” [Jarke93]. Reverse engineering as well as re-implementation are significantly facilitated if they can reuse experiences gained in previous cycles. This is why repository technology as a means of not only linking heterogeneous environments, but also as a computerized organizational memory, is playing a central role in change management (section 7.2). However, a cooperative information system will also need active components to manage the mediation between group collaboration, system change, and organizational change. Section 7.3 offers a sketch of such a “coach” for change.

## 7.1 Perspectives on Change

Let us briefly retrace each facet, in order to identify typical sources and patterns of change in cooperative information systems.

The group collaboration facet focuses on *practice* -- people, their culture, work practices, and interactions. This facet appears to be the most frequent initiator of change. Sources of change may include cultural changes (e.g., a local software development team is augmented by some people in India), different levels of knowledge (e.g., a system like Lotus Notes is no longer used in a closed organization, it is on the internet), or user interests. Within this facet, change is enabled by the spatial metaphor and the CSCW attempts to make collaboration spaces more flexible.

The organizational facet focuses on *language* – models of organizational structure, goals and policies, processes and workflows, even of information technology. Typically, changes in this facet are initiated by managers (from the group collaboration facet, but with defined organizational roles) or external consultants (with less relationships to the group collaboration facet). They involve policy changes, changes in environmental conditions (e.g., laws), implementation of meta-strategies such as business process re-engineering or total quality management, or the execution of organizational mergers. In this facet, the understanding of change is facilitated by extending the suite of formalisms available to support the full path from organizational goals and dependencies to implementation models.

The systems facet focuses on *technology* – components and coordination tools, mobility, persistence, and computational resources. Reasons for change from this facet include technical innovation (e.g., email and WWW) or degradation (our old mainframe finally seems to die!), but they may also include various kinds of maintenance all the way to re-integration and re-architecturing of the whole system. Componentization, liberated coordination, and reflection are identified as main facilitators of change in this facet.

We now turn to change propagation patterns across the facets. It is obvious that changes in one facet might affect all other facets. In this light, it may be surprising that most existing research has focused on only two facets, as indicated in figure 4.

- Formal organization <-> system integration ensuring correct implementation of business model changes: This is the classical approach in organizational information systems. A typical example for this kind of change management is the SAP approach which maps organizational reference models to technical reference models, albeit only in the limited sense of parameter variation. The need for supporting this type of change was also recognized by database vendors (Oracle CASE). This kind of change support typically neglects the humans in the organization, i.e., does not (or to an insufficient degree) consider the group collaboration facets.
- group collaboration <-> system integration: This is the interface between CSCW and Computer Science research. Typical examples are OLE, Tcl, visual programming languages; These approaches try to empower the users to adapt their computerized environment according to their individual needs without taking the organization models into account. Thus, changes can be made which are in conflict with, for example, workflow models.

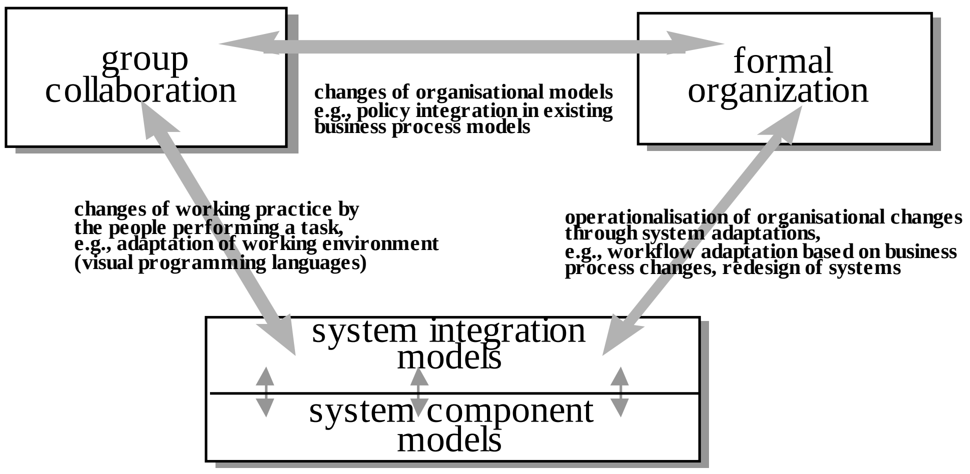
group collaboration
formal organization
changes of organisational models e.g., policy integration in existing business process models
changes of working practice by the people performing a task, e.g., adaptation of working environment (visual programming languages)
operationalisation of organisational changes through system adaptations, e.g., workflow adaptation based on business process changes, redesign of systems
system integration models
system component models

**Figure 4:** Three isolated change perspectives

- group collaboration <-> organization facet: This has been studied for many years by researchers in organizational behavior. The fact that all the relevant people in an organization must be considered when changing the organization models is recently more and more recognized, e.g., the involvement of all kinds of stakeholders during the requirements engineering task has gained more attention and is seen as very important. Traceability is seen as an important prerequisite to enable such change support.

To identify more comprehensive change scenarios, we focus on the two extreme cases of active and reactive change management. Both start from the group collaboration facet, but one from people with manager roles, the other from "normal" group members.

### 7.1.1 Active Change Management

Active changes can have three main goals:

- better system supporting the same "workflow or business process"
- adaptation of system according to particular people performing the process
- change of current working practice.

Figure 5 depicts a typical active change cycle. Usually, managers supported by consultants initiate the change in the organization models, often by changing some kind of business goal or policy. This change results in the adaptation of existing or the definition of new business process models as sketched in section 6. Subsequently, the changes in the business process models are propagated to the systems facet, e.g., via workflow models. Such changes can

- lead to the embedding of new or the change of existing system components
- lead to the change in the system integration models
  - use of different components for performing the tasks (does not effect the overall working procedure too much)
  - define alternative components which can be chosen by the people performing a task (enables freedoms for the individual to choose a component for a certain tasks which fits best)
  - application of new collaboration approaches within the workflow (try to achieve the business goal without changing the workflow)
  - new workflow definitions -> ordering of tasks, new task steps, additional collaborations, other work distributions, etc.

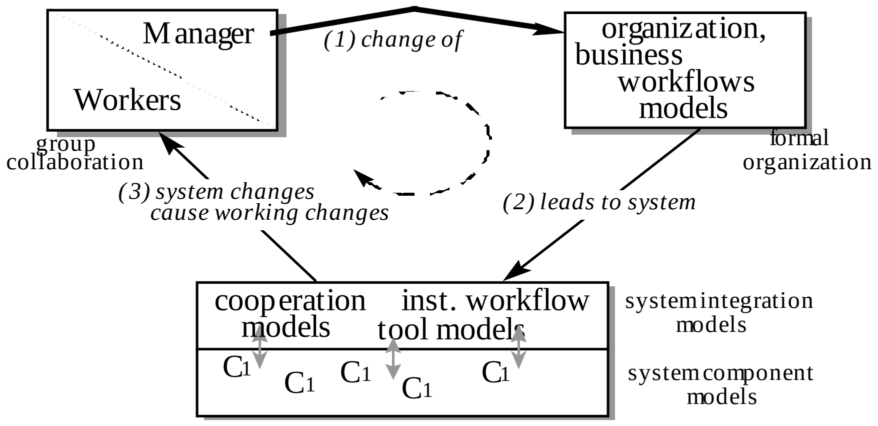
Manager
Workers
(1) change of
organization, business workflows models
group collaboration
formal organization
(3) system changes cause working changes
(2) leads to system
cooperation models
inst. workflow tool models
system integration models
C1
system component models

**Figure 5:** Active change

Indirectly, the changes of the cooperative information systems affect the people performing the tasks, i.e., changes of the cooperative information systems leads to new working practice. Of course, there are also active changes which do not change the system at all but are intended to influence directly the work practice.

When looking for formal models or computerized support for such active changes, relevant research can be found in design engineering research, project planning and scheduling, software engineering, and more recently workflow management. For example, [Madhavji92] categorized software process changes according to the scope they affect, and co-edited a special issue of IEEE Transactions on Software Engineering on process change [Madhavji93]; very similar studies have also been conducted for dynamic change management in workflow systems by [Agostini94, Ellis95]. In the area of design as well as software engineering, the use of reason maintenance systems for effective re-planning has been extensively studied [Dhar88, Petrie93, Chung95].

### 7.1.2 Reactive Change Management

Reactive changes are typically initiated by the people performing the tasks in the organization. They can again result in changes of components or changes of coordination in the systems facet:

- use of other components for performing a predefined task or process step, i.e., such changes do not effect the organizational facet
- introduction of new components by the user. This change should at least be considered at the systems facet and may also lead to changes in the organizational facet, e.g., a change of the workflow model
- 1: many replacements: a component is replaced by a set of components by which the task defined in the business process or workflow model can be better achieved
- many : 1 replacements: a set of predefined components is replaced by a more powerful component which has the same functionality
- changes of collaboration models; adapting collaboration models to user specific needs or using a different collaboration technique as the one predefined
- changes of instantiated workflow models: performing predefined tasks in a different order or performing additional tasks, or other tasks -> may lead to unexpected results which may not conform with the goals defined in the organizational facet

There are also changes of work practice which do not impact the systems facet. A possibly unfortunate consequence of this is that they cannot be recognized as long as they lead to the expected results of the business process or workflow (i.e., no support is needed).

A typical change process initiated by reactive changes is depicted in Figure 6. In contrast to current practice, reactive changes should be considered in the organizational facet, i.e., whenever an reactive change effects an organizational model, this change should be recognized by the people responsible for maintaining the organizational models. The reflection of reactive changes, in the sense of Total Quality Management [Feigenbaum91], have been recognized as an invaluable source for process improvements (e.g., in the Experience Factory approach studied in Software Engineering [Oivo92]).

Change support is needed for steps (2) and (3) of Figure 6. In particular, the need for short-term notification and long-term traceability has been recognized especially in design-oriented applications. Notification and more general awareness features are offered by configuration management systems (e.g., [Jarke92]) and are one of the main foci of CSCW research at the moment (cf. section 5). The importance of traceability is emphasized by empirical studies [Gotel94, Ramesh93] as well as mandated by several engineering standards, such as DOD-2167a in software engineering.

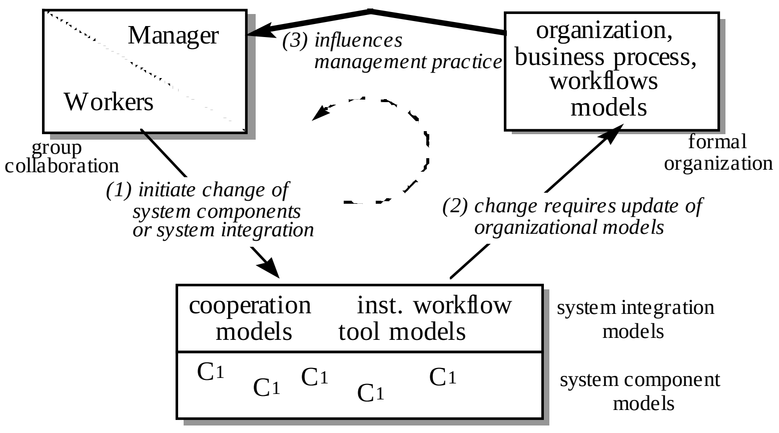
Manager
Workers
(3) influences management practice
organization, business process, workflows models
group collaboration
formal organization
(1) initiate change of system components or system integration
(2) change requires update of organizational models
cooperation models
inst. workflow tool models
system integration models
C1
C1
C1
C1
C1
system component models

**Figure 6:** Reactive change

### 7.1.3 Pro-active Change Management

To avoid getting overloaded with reactive changes, we must place more emphasis on pro-active containment strategies [Ackoff67]. Several of these strategies -- componentization and control liberation in system integration, generalized work space metaphors in CSCW, and more abstract organizational models avoiding unnecessary over-commitment -- introduce user freedoms, without sacrificing the possibility to record aberrations from expected behavior for process improvement.

Another group of pro-active change management strategies is related to the reuse of experiences in earlier change processes. Together with the richer models of early phases in requirements engineering and enterprise modeling, reason maintenance techniques which have been intensively studied in design and software engineering since the late 1970's [Stallman77, Doyle79, Dhar88, Petrie93], have emerged as a central technology for dealing with this problem. Repositories, to be discussed in the next subsection, are the main framework for this kind of pro-active change management.

A final group of pro-active change management strategies which have received less attention so far, are pro-active workflow modeling or project planning techniques.

One important aspect is the choice of process models; for example, there is some evidence that artifact-centered process models (e.g., based on the electronic circulation folder metaphor [Prinz96]) or compositional, situation-oriented process models (e.g., [Rolland94]) are more amenable to unforeseen exceptions and process changes than those enforcing an elaborate, strict control structure, such as the initially proposed Petri Nets (e.g., Domino [Kreifelts86], FUNSOFT [Gruhn91], SPADE [Bandinelli93]), or CSCW Process Grammars (e.g., [Glance96]).

A second aspect is the organization of work within a given process model. For example, [Srikanth89] present a decomposition theory for work assignment in project plans that is optimized for containing unplanned changes within as small a subgroup of the project team as possible. Such strategies -- about which, in general, relatively little is known so far -- enable a rather stable decomposition of responsibilities as well as defined change escalation strategies in case changes with larger impact do occur.

## 7.2 Cooperative information systems support for change

Support for change must obviously involve all facets of a cooperative information system: there must be a group collaboration on an intended change, the change must be supported and recorded by organizational change models, and last not least, there must be technological support for change. In this subsection, we focus on the latter.

Technical support for change management has already been partially discussed, in particular, section 4 presented ways to make system integration more flexible and thus amenable to change. In addition, certain kinds of software agents are being experimented with, which specifically support change in a group setting:

- *Awareness agents* notify managers of relevant adhoc changes and support the adaptation of the organization models; notify workers of organizational changes, explain the changes, guide them in adapting their working practice to the changes.
- *Change integration agents* provide functionalities such as impact analysis, traceability, notification of effected people, and cost analysis.
- *Simulation agents* test changes and their impact before implementing. Since changes are modeled and the system facet is enable to adapt the cooperative information systems behavior according to these models, such tests are easier than with conventional approaches [Oberweis94, Peters96].

The conceptual platform of models and tools, on which such agents provide their service, is typically defined in meta models and managed in centralized or distributed *repositories*.

Change management demands a global (though specialized) view of all facets and components of a cooperative information system. This change view has to be patched together from several partial models, put together in terms of different notations, at different times and by different people. A standard way of accomplishing this patching together is through metamodeling. Already mentioned by Abrial as early as 1974 [Abrial74], metamodeling has been researched as the prime approach for dealing with the problem of tailoring and linking models since the mid-1980’s [Kotteman84]. In the early ’90s, a series of workshops and conferences on method engineering has emerged which focus on this problem.

Metamodeling is the activity of creating descriptions of modeling formalisms and possible links between modeling perspectives. The creation of metamodels requires a language in which they are specified, which can in turn be seen as a metametamodel.

Many metamodeling techniques are quite informal and just rely on extensible graphical notations; the aforementioned ARIS system or the RDD-100 traceability environment [Alford90] are good examples. While both of these rely on extended entity-relationship formalisms, object-oriented approaches augment structural descriptions of objects with behavioral ones.

Metamodeling has progressed to the stage where we are beginning to see the first reports on commercial experiences for metamodeling environments where modeling tools are automatically created or linked via logic-based metalevel specifications. For example, the MetaEdit environment of the University of Jyväskylä [Smolander91, Kelly96] allows the automatic generation of browser/editor interfaces for formal conceptual models from graphical specifications, using a metametamodel called GOPRR which offers Graph, Object, Property, Relationship, and Role as basic meta-linguistic concepts and a pre-defined set of constraints.

Based on ideas from semantic networks in knowledge representation, the Telos language [Mylopoulos90] allows method engineers to tailor such metametamodels to specific problem domains. Languages like Telos can be used to rapidly create integrated problem-specific analysis and design environments for CIS. This has been demonstrated in a number of applications of the ConceptBase system developed at RWTH Aachen [Jarke95]. These meta meta models address, for example, the computer-assisted analysis of the relationships between multiple company departments [Peters95], business analysis perspectives [Nissen96], software process modeling, and traceability approaches [Ramesh92, Jarke94, Pohl96]. They also facilitate forward and reverse engineering between system specification and implementation [Jeusfeld95], and can be linked to metamodels of dynamic simulation models which analyse and predict the behavior of cooperative information system [Peters96].

Using the principles of metamodeling, *repositories* or *data warehouses* have been proposed as databases which describe other information sources, in order to help either in their usage or in their evolution. A good overview of the questions involved in implementing repositories is given in [Bernstein94]. Not surprisingly, technical and ontological extensibility, multiple perspectives and granularities, and a rich set of referential relationships are among the most important features mentioned.

In the NATURE project, a process-oriented repository-centered architecture for change management has been proposed [Jarke94b, Pohl96]. Its main feature is that it views the trace of an ongoing work process under multiple plans and traceability models, thus enabling the evaluation and control of processes from multiple organizational perspectives as well as the active support by change tools such as planners, decision support systems, and invocation of existing automated components. In the next section, we sketch the vision of a change coach that could operate on such process repositories in order to facilitate the mediation of change between the three facets.

## 7.3 Coach: An Envisioned Multi-Facet Cooperation Support Agent

In this section, we provide an example of a multi-facet cooperation support agent call “coach”. As explained in Section 5.2, the organizational model defines what is possible in group collaboration, while part of the activities in group collaboration can be to modify the organizational model. The coach is, therefore, in charge of facilitating the information exchange between group collaboration support and organizational model as well as managing the propagation of changes between the two, possibly including the systems facet if the changes require, for example, re-integration or re-architecturing of the underlying system. It is an active, autonomous intelligent agent in that it can self-trigger when needed and its behaviour is guided by its own knowledge.

In the following, we present an example scenario to help us explain our vision of the coach. First, we present some background information about the example. Employees of an organization used to follow a particular procedure in order to obtain travel reimbursements. This procedure is specified, with other relevant definitions, in the organizational model. Its main steps are: before leaving — prepare an estimate, submit it for approval both to the project leader and to the head of the department, and if responses are positive, send the signed form to the administration office; after coming back from the trip — calculate the actual expenses (enclosing proof for all of them), submit the documentation for approval both to the project leader and to the department head, and, if responses are positive, send all documents to the administration office for the reimbursement. In other words, the procedure need a double signature from the person in charge of the project funds (i.e., the project leader) and the person responsible for the administration of the department (i.e., the department head) both before and after the trip.

The department head, upon receiving several complains from her employees, initiates an investigation of the efficiency of the procedure. Instead of just getting some historical records or a statistical report, the coach is able to have a meaningful valuation of the relationships between the planned and the actual behaviour. This is because the coach is capable of associating the model (i.e., the procedure definition as specified in the organizational model) with the enacted workflow (i.e., all instances of the procedure executed as part of the group collaboration). This example demonstrates situations where the coach is not representing any particular facet, but gathering, analyzing, and integrating multi-facet information.

Although not surprising, the outcome of the analysis does show that the activities to obtain the signatures, both before and after the trip, are the bottleneck of the procedure in that they consume the most amount of time. The coach also points out that if the actual trip expenses is lower than estimated, the project leader's decision of not approving never happens. Finally, the department head knows that she is not interested in the details, since she does not have direct responsibility on the project funds, but just wants to have an overall control. Therefore, she decides to modify the procedure definition, canceling the signature of the project leader after the trip in case of expenses below the estimate, and leaving the approval of just the department head at the end of the procedure.

Automatically, the new procedure substitutes the old one in the organizational model. Once again, it is the duty of the coach to maintain the coherence between the model and the enacted workflow. In this case, it is to propagate the changes from the organizational model to the appropriate facet(s). In doing so, it changes the remaining activities of all *switchable* instances currently in execution without having to restart them from the beginning.

The ability to dynamically change all running instances of a procedure needs further explanation. The change mainly involves three steps: (1) collect all running instances, (2) assess which of them are switchable, and (3) change those that are switchable to the new workflow definition without causing errors (e.g., without skipping activities to be performed or performing an activity twice). Due to the lack of certain domain knowledge, it is not always possible to automatically determine which running instances can be switched (e.g., adding new activities to a procedure) [Agostini94, Ellis95]. Under those circumstances, the coach helps the department head to define switchability policy.

One can argue that the aforementioned three steps can be carried out by the workflow management system itself and, therefore, having a coach is unnecessary. However, procedures are part of the organizational model and, therefore, if we do not want to create a hierarchical link between the different facets, we need an autonomous agent to perform them. The idea of the coach, and the multi-facet cooperation support agents in general, is a way to manage change without complicating the underlying systems in performing their regular activities (i.e., the workflow management system manages the execution of workflows, while the coach manages changes).

Note that most of coach's features are not new. The novelty is rather in recognizing the need, in a distributed and complex environment, for someone to be in charge of tasks that cover several units/applications/sites.

## 8. Towards a Generic Architecture for Cooperative Information Systems

In the preceding sections, we have outlined a framework for identifying and characterizing research problems and issues in cooperative information systems. This clearer picture also enables us to envision possible solutions. In this section, we propose a generic architecture which puts forth a vision of how cooperation and on-going change can be supported, while addressing all three facets and their interactions.

The purpose of this generic architecture (or architectural model) is to help focus and coordinate our research efforts towards solutions that are implementable within the expected time frame, while allowing for considerable freedom within each subarea of research. The architecture identifies the major architectural layers and component types, what kinds of functions they perform, what information they maintain, and what kinds of interdependencies they manage in order to deal with various classes of changes and modes of cooperation. The generic architecture leaves open many issues to be addressed during specific architectural design (e.g., the interface mechanisms among particular architectural components).

We start with the premise that existing practices with computer-based technologies, such as compilers, DBMSs, and workflow products will continue to exist, get used and evolve on their own for some time to come, partly because of existing investments in legacy systems and human training that can't be simply written off, partly because of general laws of inertia. The task at hand for us working in this area is not to replace this practice, but rather to augment it. Accordingly, our conception of cooperative information systems is one of providing a value-added layer on top of legacy and other existing systems, thereby ensuring that they continue to fulfill users' information needs and remain consistent with organizational objectives.

The generic architecture thus consists of two layers (Figure 7):

- the base layer -- includes information systems which directly provide information services to users, and
- the cooperation support layer -- a value-added layer which provides facilities for dealing with all three facets, their interactions, and for supporting cooperation and change.

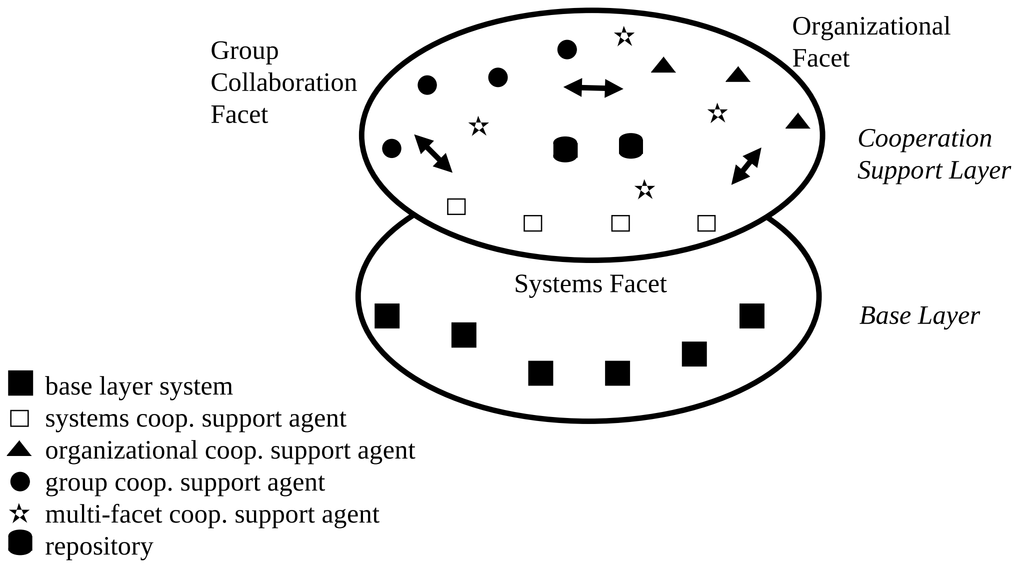
Group Collaboration Facet
Organizational Facet
Cooperation Support Layer
Systems Facet
Base Layer
■ base layer system
□ systems coop. support agent
▲ organizational coop. support agent
● group coop. support agent
☆ multi-facet coop. support agent
repository

**Figure 7:** A Generic Architecture for Cooperative Information Systems

As shown in the figure, the cooperation support layer consists of several types of cooperation support agents (or components). The systems, group, and organizational agents provide interfaces to their respective facet and interact with each other. The fourth type of agents, the multi-facet cooperation support agent, does not represent any single facet. They play a control and/or coordination role among the aforementioned three types of agents (Section 7.3).

Repositories in the cooperation support layer are used to maintain the various types of models and meta-models, their interdependencies, and their relative changes over time (as discussed in Section 7.2).

Finally, the architecture also maintains a set proximity relations. A proximity relation is defined by the interdependencies among the three facets, involving objectives, people, and systems. These may include data and/or control dependencies, visibility relationships (import/export properties), and change interdependencies (what kinds of changes should be propagated in what ways). For example, there is a proximity relation between the “increase profitability” goal (in the organizational facet), the structure of a new team of workers (in the group collaboration facet), and a new system which helps them do their work the “new way” (in the systems facet). This dependence is important to record and keep track of because if anything changes, it will tell us what else is potentially affected. Proximity relations are modelled in the repository and can be used by the four types of cooperation support agents when performing their duties.

This approach of providing a value-added layer for cooperation support is complementary to approaches which attempt to build cooperation and flexibility for change into base layer components (e.g., recent componentized system approaches that use standardized middleware for interoperability and reusability). Our belief is that even with such approaches, there will still be need for value-added components which deal specifically with cooperation and change.

In addition to dealing with cooperation and change, we believe that this architecture is also useful in providing a framework for delivering information services within an organization, in support of business processes, and in aligning them with organizational objectives. Such delivery is based on the execution of legacy information systems and workflows which disseminate, retrieve and distribute information. The systems cooperation support agents compose systems transactions into scripts (or, long transactions). Each of these scripts is executing in a distributed fashion and uses the services of several legacy systems. The systems cooperation support agents also deal with issues related to integration and interoperation when executing these scripts. Such scripts are, in turn, executing in support of human collaborative work defined in terms of business processes through the group cooperation support agents. Finally, such work contributes to objectives at the organizational facet.

The generic architecture entails a research agenda that is being pursued by the research teams represented by the authors. We are working towards cooperative information systems solutions that can be deployed in the time frame of five to ten years from now. A number of pieces of the technology already exist as research prototypes and are being further developed in relation to the generic architecture. The architecture assumes that computing environments in the targeted time frame will be highly interconnected, involving heterogeneous systems and networks. High-capacity multimedia network-based computing will be commonplace, making use of local and global networks, which may be wired, mobile, or fibre-based. There will be standardization efforts in many areas, but uniformity of standards will unlikely be achieved in most areas. There will be high expectations for performance, reliability, and information quality. Socio-economic issues will remain crucial to the success of new technologies, systems, and services.

## 9. Conclusions

We have argued that the basic issue that needs to be addressed by Cooperative Information Systems as a research area is *organizational and technological change*. We have then reviewed some of the issues which Cooperative Information Systems bring forth and some of the approaches being tried to address these issues. We have also proposed a generic, coarse-grain architecture for systems which is capable of relating legacy systems to human collaborative activity and organizational goals, for purposes of driving information services and guiding change.

The proposed architecture is only a starting point for the research in Cooperative Information Systems and we consider it as a basis for a multidisciplinary discussion on the subject. It is our opinion that Cooperative Information Systems is a field opening new relevant research challenges to various computer science sub-disciplines and calling far breaking the existing boundaries between them. Some explanations for the above claim are given below.

1. Cooperative Information Systems, unlike Information Systems and Databases, need to address not only technological issues, such as "How do I build a system that does X?", but also human and social ones ("How does a group of people accomplish task Y?") as well as organizational ones ("How can we meet objective Z?"). Within this framework, it seems to us that neither the research in information systems can conquer the CSCW field as if CSCW systems were mere innovative interfaces for distributed Information Systems, nor can CSCW scholars and practitioners avoid to address the typical issues of information systems research as if the latter were mere information sources which users can access during their work. On the contrary, Cooperative Information Systems emerge as newly conceived systems coupling the human process orientation of CSCW systems with the distributed systems orientation of information systems. Cooperative information systems are based on an

architecture where the process oriented domain of human activity and the distributed systems of information processing are two autonomous cooperating modules of one and the same system. Building such systems will require technological as well as social and organizational know-how.

2. The new and growing complexity of the architecture we have proposed for Cooperative Information Systems is based on active and passive components. There is, therefore, the need for a growing number of active components (of intelligent agents) managing the relations between and among the different facets and, in particular, between the group collaboration support module and the organizational model. It is our conviction that the creation of these intelligent agents provides new research issues such as knowledge contextualization for the Distributed Artificial Intelligence field.

3. Cooperative Information Systems are a new type of systems, oriented to support not only information processing but also change. The design and engineering of Cooperative Information Systems does not appear well supported by the typical techniques and methods emerging from Software Engineering: it requires on the contrary new methods and tools supporting user participation, continuous changes, and on line performance evaluation.

## Acknowledgments

The authors gratefully acknowledge financial support for this collaborative project from the European Commission and Ontario's Information Technology Research Centre. A particular thank is due to Alessandra Agostini who contributed to the development of the Coach concept. The following people read an earlier version of this paper and provided numerous constructive suggestions on improving the paper during a three-day workshop held in Vancouver, Canada on August 5-7, 1996. The authors wish to thank Mike Bauer, Alex Borgida, Philippe Du Bois, Tamer Ozsu, Remo Pareschi, Peter Peters, Bala Ramesh, Jacob Slonim, and Francois Vernadat.

## References

[Abrial74] Abrial, J.-R., "Data Semantics", in J.W.Klimbie, K.L. Koffeman (eds.): *Data Management Systems*, North-Holland, 1974.

[Ackoff67] Ackoff, R.L. "Management misinformation systems", *Management Science 14,* 4 (1967), 147-156.

[Agostini94] Agostini, A., De Michelis, G., and Petruni, K., "Keeping Workflow Models as Simple as Possible", *Proceedings of the Workshop on Computer-Supported Cooperative Work, Petri-Nets and related formalisms*, within the 15th International Conference on Application and Theory of Petri Nets (ATPN '94), 1994, 11-23. (Available on request from CTL, University of Milano, email: gdemich@dsi.unimi.it).

[Agostini95] Agostini, A., De Michelis, G., Grasso, M., "The Milano System", in "A Computational Model of Organizational Context". *COMIC Deliverable 1.3, 1995,* 163-192 (Available on request from the Computing Department, University of Lancaster, Lancaster LA1 4YR, UK, e-mail: tom@comp.lancs.ac.uk).

[Agostini97] Agostini, A., De Michelis, G., Grasso, M., Prinz, W. and A. Syri, "Contexts, Work Processes, Work Spaces", *Computer Supported Cooperative Work: An International Journal,* Kluwer Academic Publishers, The Netherlands, 1997, to appear.

[Alford90] Alford, M.W., "Software Requirements Engineering Methodology (SREM) at the Age of Eleven -- Requirements-Driven Design". In P.A.Ng, R.T.Yeh (eds.): *Modern Software Engineering,* Van Nostrand Reinold, 1990.

[Alonso96] Alonso, G. “The role of database technology in workflow management systems”, Proceedings of CoopIS’96 (Brussels, Belgium, 1996), IEEE Computer Society Press, 176-179.

[AMICE89] ESPRIT Project AMICE, *Open Systems Architecture for CIM,* Springer 1989.

[Appelt96] Appelt, W., “CoopWWW - interoperable tools for cooperation support using the World-Wide Web”. *Proc. 5th ERCIM/W4G Workshop ‘CSCW and the Web’,* Arbeitspapiere der GMD 984, GMD, Sankt Augustin, Germany, 1996, 95-99.

[Argyris78] Argyris, M. and Schon, D., *Organizational Learning,* Addison Wesley, Reading, 1978.

[Argyris96] Argyris, M. and Schon, D., *Organizational Learning II,* Addison Wesley, Reading, 1996.

[Atkinson90] Atkinson, M., Bancilhon, F., De Witt, D., Dittrich, K., Maier, D., and Zdonik. S., “The Object-Oriented Database System Manifesto”, In *Deductive and Object-oriented Databases*. Elsevier Science Publishers, Amsterdam, Netherlands, 1990.

[Bandinelli93] Bandinelli, S.C., Fuggetta, A., and Ghezzi, C. “Software process model evolution in the SPADE environment”, *IEEE Trans. Software Eng. 19,* 12 (1993), 1128-1144.

[Berliner88] Berliner, C. and Brimson, J., *Cost Management for Today’s Advanced Manufacturing: The CAM-I Conceptual Design,* Free Press, 1988.

[Berners-Lee96] Berners-Lee, T. “WWW: past, present, and future”, *IEEE Computer 29,* 10 (1996), 69-77.

[Bernstein93] Bernstein, P., “Middleware: An Architecture for Distributed System Services”, CRL 93/6, Digital Equipment Corporation, Cambridge Research Lab, March 1993.

[Bernstein94] Bernstein, P.A. and Dayal, U., “An Overview of Repository Technology”. *Proceedings 20th Intl. Conf. Very Large Data Bases* (Santiago de Chile, 1994), 705-713.

[Bernus96] Bernus, P. and Nemes, L. (eds.) *Modelling and Methodologis for Enterprise Integration,* Chapman and Hill ,1996.

[Brachman85] Brachman, R.J. and Schmolze, J.G., "An Overview of the KL-ONE Knowledge Representation System", *Cognitive Science 9,* 2 (1985), 171-216.

[Brodie97] Brodie, M., "The Emperor's Clothes are Object-Oriented and Distributed", in this book.

[Bubenko80] Bubenko, J. A. “Information Modeling in the Context of System Development,” Proceedings of Information Processing ’80, S.H. Lavington (ed), North-Holland, 1980, 395-411.

[Bubenko93] Bubenko, J., “Extending the Scope of Information Modeling”, *Proceedings of the Fourth International Workshop on the Deductive Approach to Information Systems and Databases,* (Lloret-Costa Brava, Catalonia, Sept. 20-22, 1993), 73-98.

[Cardelli94] L. Cardelli. “Obliq: A Language with Distributed Scope”, Technical report, Digital Equipment Corporation, Systems Research Center, Palo Alto, California, June 1994.

[Carroll91] Carroll, J.M., Kellogg, W.A., Rosson, M.B., “The Task-Artifact Cycle”, in J.M. Carroll (ed.): *Designing Interaction. Psychology at the Human-Computer Interface.* Cambridge, 1991.

[Chung93] Chung, K. L., *Representing and Using Non-Functional Requirements for Information System Development: A Process-Oriented Approach*, Ph.D. thesis, University of Toronto, 1993.

[Chung95] Chung, L., Nixon, B. and Yu, E., “Using Non-Functional Requirements to Systematically Support Change,” *IEEE International Symposium on Requirements Engineering* (RE'95), IEEE Computer Society Press, 1995, 132-139.

[Constantopoulos95] Constantopoulos, P., Jarke, M., Mylopoulos, J., and Vassiliou, Y., “The Software Information Base: A Server for Reuse”, *VLDB-Journal 4*, 1 (1995), 1-43.

[CoopIS94] Second International Conference on Cooperative Information Systems (CoopIS-94), Toronto, Canada, May 17-20, 1994.

[Dardenne93] Dardenne, A., Lamsweerde, A.V., Fickas, S., “Goal-directed requirements acquisition”. *Science of Computer Programming 20,* 1 (1993), 3-50.

[DeMichelis94] De Michelis, G. and Grasso, M., “Situating Conversations within the Language/Action Perspective: The Milano Conversation Model”, *Proceedings of the Conference on CSCW* (Chapel Hill, North Carolina, October 22-26, 1994), ACM Press, 89-100.

[DeMichelis95] De Michelis, G., “Work Processes, Organizational Structures and Cooperation Supports: Managing Complexity”, *Proceedings of the Fifth IFAC Symposium on Automated Systems Based on Human Skills - Joint Design of Technology and Organization*, Pergamon Elsevier International, New York, 1995.

[DeMichelis96] De Michelis, G., “Computer-based systems between people and social complexity”, *Proceedings of the Toshiba Chair Symposium on Human Oriented Information Technology and Complex Systems,* Keio University, Tokio, 1996.

[Dhar88] Dhar, V. and Jarke, M. “Dependency-directed reasoning and learning in systems maintenance support”, *IEEE Trans. Software Eng. 14,* 2 (1988), 211-227.

[Doyle79] Doyle, J. “A truth maintenance system”, *Artificial Intelligence 12,* 3 (1979), 231-272.

[Dubois89] Dubois, E., “A Logic of Action for Supporting Goal-Oriented Elaborations of Requirements,” *Proceedings of the Fifth International Workshop on Software Specification and Design* (IWSSD'89), CS Press, 1989, 160-168.

[DuBois97] Du Bois, P., Dubois, E. and Zeippen, J-M. “On the Use of a Formal RE Language: the Generalized Railroad Crossing Problem”, *IEEE Intl Symposium on Requirements Engineering* (Annapolis, USA, January 1997).

[Edmond97] Emond, D. and Papazoglou, M.P., "Reflection is the Essense of Cooperation", in this book.

[Ehn88] Ehn, P., *Work-oriented Design of Computer Artifacts*. 2nd ed. Stockholm. Arbetslivscentrum, 1988.

[Ellis95] Ellis, C. A., Keddara, K., and Rozenberg, G., “Dynamic Change within Workflow Systems”, *Proceedings of the Conference on Organizational Computing Systems* (Milpitas, CA, August 13-16, 1995), ACM Press, 10-21.

[Feather87] Feather, M., “Language Support for the Specification and Development of Composite Systems,” *ACM Transactions on Programming Languages and Systems 9,* 2 (April 1987), 198-234.

[Feigenbaum91] Feigenbaum, A. *Total Quality Control*, McGraw-Hill, 1991, [Finin95] Finin, T., Lambrou, Y., and Mayfeld, J., "KQML as an Agent Communication Lamguage", in *Software Agents*, J. Bradshaw (ed), AAAI/MIT Press, Menlo-Park, Ca., 1995.

[Flores88] Flores, F., Graves, M., Hartfield, B. and Winograd, T., “Computer Systemns and the Design of Organizational Interaction”, *ACM Transactions on Office Information Systems 6,* 2 (April 1988), ACM Press, New York, 153-172.

[Gasser91] Gasser, L., “Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics”, *Artificial Intelligence 47* (January 1991), 107-138.

[Georgakopoulos95] D. Georgakopoulos, M. Hornick, and Sheth, A. “An Overview of Workflow Management: From Process Modeling to Infrastructure for Automation”. *Journal on Distributed and Parallel Database Systems, 3,* 2 (April 1995), 119-153.

[Glance96] Glance, N.S., Pagani, D.S., and Pareschi, R. “Generalized process structure grammars (GPSG) for flexible representations of work”. *Proceedings of CSCW’96* (Cambridge, Mass., 1996), ACM Press, 180-189.

[Gotel94] Gotel, O., and Finkelstein, A., “An Analysis of the Requirements Traceability Problem,” *Proceedings IEEE International Conference on Requirements Engineering,* Computer Science Press, April 1994.

[Greenberg94] Greenberg S. and Marwood, D., “Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface”, *Proceedings of the Conference on CSCW* (Chapel Hill, North Carolina, October 22-26, 1994), ACM Press, New York, 207-217.

[Greenspan86] Greenspan, S., Borgida A., and Mylopoulos, J., “A Requirements Modeling Language and its Logic”, *Information Systems 11,* 1 (1986), 9-23.

[Gruhn91] Gruhn, V. *Validation and Verification of Software Process Models*, Ph.D. Thesis, Univ. Dortmund, Germany, 1991.

[Gutwin96] Gutwin, C., Roseman, M., Greenberg, S. “A usability study of awareness widgets in a shared workspace groupware system”. *Proceedings of CSCW’96* (Cambridge, Mass., November 16-20, 1996).

[Hamel94] Hamel, G. and Prahalad, C., *Competing for the Future,* Harvard Business School Press, Boston, Mass., 1994.

[Hammer93] Hammer, M. and Champy, J., *Reengineering the Corporation: A Manifesto for Business Revolution,* HarperBusiness, 1993.

[Hsu93] Hsu M., “Special Issue on Workflow and Extended Transaction Systems”, *Bulletin of the Technical Committee on Data Engineering 16,* 2 (June 1993), IEEE Computer Society.

[Jacobson92] Jacobson, I., et al. *Object-Oriented Software Engineering: A Use Case Driven Approach,* Prentice-Hall, 1992.

[Jackson95] Jackson M. and Zave, P., “Deriving Specifications from Requirements,” *Proceedings of the Seventeenth International Conference on Software Engineering,* ACM Press, 1995, 15-24.

[Janson96] Janson, M. and Woo, C.C., “A Speech Act Lexicon: An Alternative Use of Speech Act Theory In Information Systems”, *Information Systems Journal 6,* 3 (July 1996), United Kingdom: Blackwell Science.

[Jarke88] Jarke, M., Rose, T., "Managing Knowledge About Information System Evolution", *Proceedings ACM-SIGMOD Conf.*, Chicago, 1988

[Jarke92] Jarke, M., Maltzahn, C.G.v., and Rose, T., “Sharing processes: team support in design repositories”, *Intl. J. Intelligent and Cooperative Information Systems 1*, 1 (1992), 145-167.

[Jarke92a] Jarke, M., Mylopoulos, J., Schmidt, J.W., and Vassiliou, Y., “DAIDA: An Environment for Evolving Information Systems”, *ACM Transactions on Information Systems 10*, 1 (January 1992), 1-50.

[Jarke93] Jarke, M. and Pohl, M., “Establishing Visions in Context: Towards a Model of Requirements Processes”, *Proc. 14th Intl. Conf. Information Systems* (Orlando, Fl, December 5-8, 1993), 23-34.

[Jarke94] Jarke, M., Pohl, K., Rolland, C., and Schmitt, J.R., “Experience-Based Method Evaluation and Improvement: A Process Modeling Approach”, *Proc. IFIP WG 8.1 Conf. CRIS*, Maastricht, North Holland, 1994

[Jarke94b] Jarke, M., Pohl, K., Dömges, R., Jacobs, S., and Nissen, H.W., “Requirements Information Management: The NATURE Approach”, *Engineering of Information Systems 2*, 6, 1994.

[Jarke95] Jarke, M., Gallersdörfer, R., Jeusfeld, M.A., Staudt, M., and Eherer, S., “ConceptBase - A Deductve Object Base for Meta Data Management”, *Journal of Intelligent Informations Systems 4*, 2 (1995), 167-192

[Java95] *Java: The Inside Story.*, http://www.sun.com/sunworldonline/swol-07-1995/swol-07-java.html, 1995.

[Jeusfeld95] Jeusfeld, M.A. and Johnen, U. “An Executable Meta Model for Re-Engineering of Database Schemas”, *International Journal of Cooperative Information Systems 4*, 2&3 (June & September 1995) -- Special Issue on ER’94, Singapore: World Scientific, 237-258.

[Karbe91] Karbe, B., Ramsberger, N., “Concepts and implementation of migrating office processes”. In Brauer/Hernandez (eds.): *Verteilte Kuenstliche Intelligenz und Kooperatives Arbeiten*, Springer, 1991.

[Kashyap97] Kashyap, V. and Sheth, A., "Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies", in this book.

[Keen91] Keen, P., *Shaping the Future: Business Design Through Information Technology*, Harvard Business School Press, Boston, Mass., 1991.

[Kelly96] Kelly, S., Lyytinen, K., and Rossi, M. “MetaEdit+: a fully configurable multi-user and multi-tool CASE and CAME environment”, *Proceedings of CAiSE’96* (Heraklion, Greece, 1996), 1-21.

[Kling82] Kling, R. and Scacchi, M., “The Web of Computing: Computer Technology as Social Organization”, *Advances in Computing 21* (1982), 1-90.

[Knoblock94] Knoblock, C.A., Arens, Y., Hsu, C.N., "Cooperating Agents for Information retrieval", Proceedings on the Second International Conference on Cooperative Information Systems (Toronto, Canada, May 17-20, 1994), 122-133.

[Kotteman84] Kotteman, J. and Konsynski, B., “Dynamic Metasystems for Information Systems Development”, *Proc. 5th Intl. Conf. Information Systems*, Tucson, Az, 1984

[Kreifelts86] Kreifelts, T., Woetzel, G., “Distribution and error handling in an office procedure system”. *Proceedings of IFIP WG8.4 Working Conference on Methods and Tools for Office Systems* (Pisa, Italy, October 22-24, 1986), 197-208.

[vanLamsweerde95] van Lamsweerde, A., Darimont R. and Massonet, P., “Goal-Directed Elaboration of Requirements for a Meeting Scheduler,” *Proceedings of IEEE International Symposium on Requirements Engineering* (RE’95), IEEE Computer Society Press, 1995, 194-203.

[Lau94] C. Lau. *Object-Oriented Programming using SOM and DSOM*. Van Nostrand Reinhold, Thomson Publishing Company, New York, 1994.

[Lee92] Lee, J., *A Decision Rationale Management System: Capturing, Reusing, and Managing the Reasons for Decisions*, Ph.D. thesis, MIT, 1992.

[MacGregor90] McGregor, R., "The Evolving Technology of Classification-Based knowledge Representation Systems", in *Principles of Semantic Networks,* J. Sowa (ed), Morgan-Kaufmann, 1990.

[Madhavji92] Madhavji, N. “Environment evolution: the Prism model of change”, *IEEE Transactions on Softwar Engineering 18,* 5 (1992), 380-392.

[Madhavji93] Madhavji, N. and Penedo, M.H. (eds.). Special Section on the Evolution of Software Processes. *IEEE Trans. Software Eng. 19,* 12, 1993.

[Manola92] Manola, Heiler, F., Georgakopoulos, D., Hornick, M. and Brodie, M., “Distributed Object Management”, *International Journal of Intelligent and Cooperative Information Systems 1,* 1 (March 1992), Singapore: World Scientific, 5-42.

[Mantei91] Mantei, M., Baecker, R., Sellen, A., Buxton, W., Milligan, T. and Wellman, B., “Experiences in the Use of a Media Space”, *Proceedings of CHI* (New Orleans, April 28 - May 2, 1991), ACM Press, New York, 127-138.

[Martens97] Martens, C. and Woo, C.C. “OASIS: An Integrative Toolkit for Developing Autonomous Applications in Decentralized Environments,” *Journal of Organizational Computing and Electronic Commerce 7,* 3 (1997), New Jersey: Ablex Publishing Co.

[Mathiske95] Mathiske, B., Matthes, F. and Schmidt, J.W. “Scaling Database Languages to Higher-Order Distributed Programming”. *Proceedings of the Fifth International Workshop on Database Programming Languages* (Gubbio, Italy, September 1995), Springer-Verlag.

[Matthes94] Matthes F. and Schmidt, J.W. “Persistent Threads”, *Proceedings of the Twentieth International Conference on Very Large Data Bases* (Santiago, Chile, September 1994), 403-414.

[Medina-Mora92] Medina-Mora, R., Winograd, T., Flores, R. and Flores, F., “The Action Workflow Approach to Workflow Management Technology”, *Proceedings of Conference on Computer Supported Cooperative Work* (Toronto, Canada, October 31 - November 4, 1992), ACM Press, 281-288.

[Microsoft94] Microsoft Corporation. *Microsoft Office Developer's Kit,* 1994.

[Milliner95] Milliner, S., Bouguettaya, A., and Papazoglou, M.P., "A Scalable Architecture for Autonomous Heterogeneous Database Interactions", *21st VLDB Conference* (Zurich, Sept. 1995).

[Munson94] J.P. Munson and Dewan, P., “A Flexible Object Merging Framework”. *Proceedings of the Conference on CSCW* (Chapel Hill, North Carolina, October 22-26, 1994), ACM Press, New York, 231-242.

[Mylopoulos90] Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. “Telos: Representing Knowledge about Information Systems”, *ACM Transactions on Information Systems 8,* 4 (October 1990), 325-362

[Mylopoulos92] Mylopoulos, J., Chung, L. and Nixon, B., “Representing and Using Non-Functional Requirements: A Process-Oriented Approach”, *IEEE Transactions on Software Engineering 18,* 6 (June 1992).

[Nissen96] Nissen, H.W., Jeusfeld, M.A., Jarke, M., Zemanek, G.V., and Huber, H. “Managing multiple requirements perspectives with meta models", *IEEE Software, Special Issue on Requirements Engineering*, March 1996.

[Nonaka95] Nonaka, I. and Takeuchi, H. *The Knowledge-Creating Company*. Cambridge University Press, 1995.

[Oberweis94] Oberweis, A., Scherrer, G., and Stucky, W. “Income/Star: methodology and tools for the development of distributed information systems”, *Information Systems 19,* 8 (1994), 643-660.

[Oivo92] Oivo, M. and Basili, V.R. “Represneting software enginering models: the TAME goal-oriented approach”, *IEEE Trans. Software Eng. 18,* 10 (1992), 886-898.

[OMG91] Object Management Group. *The Common Object Request Broker: Architecture and Specification*. Document 91.12.1, Rev. 1.1, OMG, December 1991.

[OSF93] Open Software Foundation. *OSF DCE Application Development Guide*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[Palfreyman96] Palfreyman, K., Rodden, T., “A protocol for user awareness on the World Wide Web”. *Proceedings of CSCW’96* (Cambridge, Mass., November 16-20, 1996).

[Papazoglou92] Papazoglou, M.P., Laufmann, S., and Sellis, T., "An Organizational Framework for Cooperating Intelligent Information Systems*", International Journal of Intelligent and Cooperative Information Systems 1,* 1 (1992), 169-202.

[Patil92] Patil, R., Fikes, R., Patel-Schneider, P., MacKay, D., Finnin, T., Gruber, T., and Neches, R., "The DARPA KNowledge Sharing Effort*", Proceedings 3rd International Conference on Principles of knowledge Representation and Reasoning,* Morgan-Kaufman, 1992.

[Peters95] Peters, P. Szczuko, P., Jeusfeld, M., and Jarke, M., “Business Process Oriented Information Management: Conceptual Model at Work”, *Proceedings of Conference on Organizational Computing Systems* (Milpitas, Calf., August 13-16, 1995), ACM Press, 216-224.

[Peters96] Peters, P. and Jarke, M. “Simulating the impact of information flows on networked organizations”, *Proc. 17th Intl. Conf. Information Systems* (Cleveland, Ohio, December 15-18, 1996).

[Petrie93] Petrie, C. “The Redux server”, *Proc. First Intl. Conf. on Intelligent and Cooperative Information Systems* (Rotterdam, Netherlands, May 12-14, 1993), IEEE Computer Society Press, 134-143.

[Pohl96] Pohl, K. *Process Centered Requirements Engineering*, John Wiley Research Science Press, 1996.

[Prinz96] Prinz, W. “Support for workflows in a ministerial environment”, *Proceedings of CSCW’96* (Cambridge, Mass., November 16-20, 1996), ACM Press, 199-208.

[Ramesh92] Ramesh, B. and Dhar, V. “Supporting systems development by capturing deliberations during requirements engineering”. *IEEE Trans. Software Eng. 18,* 6 (1992), 498-510.

[Ramesh93] Ramesh, B., Edwards, M., “Issues in the development of a requirements traceability model,” *Proc. Intl. Symp. Requirements Engineering* (San Diego, Ca, 1993).

[Reder90] Reder, S. and Schwab, R.G. “The Temporal Structure of Cooperative Activity”. *Proceedings of the Conference on Computer-Supported Cooperative Work* (Los Angeles, CA, October 7-10, 1990), ACM Press, 303-316.

[Rolland95] Rolland, C., Souveyet, C., and Moreno, M. “An approach for defining ways-of-working”, *Information Systems 20*, 3 (1995).

[Roseman96] Roseman, M. and Greenberg, S. “TeamRooms: network places for collaboration”, *Proceedings of CSCW’96* (Cambridge, Mass., November 16-20, 1996), ACM Press, 325-333.

[Scheer94] Scheer, A-W., *Enterprise-Wide Data Modeling*, Springer-Verlag, 1994 (4th edition).

[Schmidt77] Schmidt, J.W., “Some high-level language constructs for data of type relation.” *ACM Transactions on Database Systems 2*, 3 (September 1977), ACM Press, 247-261.

[Scott-Morton90] Scott Morton, M., ed., *The Corporation of the 1990s: Information Technology and Organizational Transformation*, Oxford University Press, 1991.

[Selic94] Selic B., et al., *Real-Time Object-Oriented Modeling*, Wiley, 1994.

[Sernadas91] A. Sernadas, C. Sernadas, P. Gouveia, P. Resende and Gouveia, J., “OBLOG - Object-Oriented Logic: An Informal Introduction”, *Technical Report*, INESC, Lisbon, 1991.

[Smolander91] Smolander, K., Lyytinen, K., Tahvanainen, V.-P., and Martiin, P., “MetaEdit -- a Flexible Graphical Environment for Methodology Modeling”, *Proceedings of CAiSE’91* (Trondheim, Norway, 1991).

[Srikanth89] Srikanth, R. and Jarke, M. “The design of knowledge-based systems for managing ill-structured software projects”, *Decision Support Systems 5,* 4 (1989), 425-448.

[Stallman77] Stallman, R.M. and Sussman, G.J. “Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis”, *Artificial Intelligence 9,* 2 (1977), 135-196.

[Stemple91] Stemple, D., Morrison, R. and Atkinson M. “Type-Safe Linguistic Reflection”, *Database Programming Languages: Bulk Types and Persistent Data*, Morgan Kaufmann Publishers, 1991, 357-362.

[Stonebraker90] Stonebraker, M., Rowe, L., Lindsay, B., Gray, J., Carey, M., Brodie, M., and Bernstein, P., “Third-Generation Data Base System Manifesto”, *ACM SIGMOD Record 19,* 3 (September 1990), ACM Press, 31-44.

[Suchman84] Suchman, L. *Plans and Situated Actiond*, Cambridge University Press, 1984.

[Suchman94] Suchman, L., “Do Categories Have Politics? The Language/Action Perspective Reconsidered”, *Computer Supported Cooperative Work: An Intenational Journal 2,* 3 (April 1994), The Netherlands: Kluwer Academic Publishers, 177-190.

[Vernadat84] Vernadat, F., “Computer-Integrated Manufacturing: On the Database Aspect”, Proceedings of CAD/CAM and Robotics Conference, Toronto, 1984.

[Vernadat96] Vernadat, F., *Enterprise Modeling and Integration*, Chapman and Hall, 1996.

[Wayner94] Wayner, P. “Agents Away”, *BYTE* (May 1994), 113-118.

[White94] White, J.E.,*Telescript Technology: The Foundation for the Electronic Marketplace*. White paper, General Magic Inc., Mountain View, California, USA, 1994.

[Wiederhold94] Wiederhold, G., "Interoperation, Mediation and Ontologies", *Proceedings of International Workshop on Heterogeneous Cooperativer Knowledge-Bases* (Tokyo, Dec. 1994).

[Wieringa94] Wieringa, R., "LCM and MCN: Specification of a Control System Using Dynamic Logic and Process Algebra," *in Case Study Production Cell - A Comparative Study of Formal Software Development*, C. Lewerentz and T. Lindner (eds), LNCS, 1994.

[Wieringa96] Wieringa, R., *Requirements Engineering: Frameworks for Understanding*, Wiley, 1996.

[Wille92] Wille, R., "Concept Lattices and Conceptual Knowledge Systems" *in Semantic Networks in Artificial Inteligence*, F. Lehmann (ed), 1992, Pergamon Press, 493-515.

[Williams92] Williams, T., "The Purdue Enterprise Reference Architecture", Purdue Laboratory for Applied Industrial Control, Purdue University, 1992.

[Winograd86] Winograd, T. and Flores, F. *Understanding Computers and Cognition; A New Foundation for Design*, Ablex 1986.

[Yu95] Yu, E., *Modelling Strategic Relationships for Process Reengineering*, Ph.D. thesis, University of Toronto, 1995.

[Yu95b] Yu, E., Du Bois, Ph., Dubois E. and Mylopoulos, J., "From Organization Models to System Requirements - A `Cooperating Agents' Approach", *Proceedings of the Third International Conference on Cooperative Information Systems* (Vienna, May 9-12, 1995), University of Toronto Press, 194-204.