Challenges and directions for EI: a distributed AI perspective

Mark Klein

The Applied Research Laboratory

*Pennsylvania State University, P.O. Box 30, State College, PA
16804-0030 USA. +1 (814) 863-5381 (Voice), +1 (814) 865-
7097 (Fax). klein@quark.arl.psu.edu*

Abstract

Integrating an enterprise can be viewed as an exercise in coordinating a diverse and highly distributed set of inter-dependent activities, many of them involving cooperative synthesis of artifacts like marketing plans, schedules, products, manufacturing procedures and so on. Several distinct kinds of "coordination technology" have evolved to support effective coordination in cooperative work, including process management, conflict management and rationale capture tools. This paper reviews some of the major weaknesses with current coordination technology and suggests several technical directions for addressing these weaknesses. These directions include developing semi-structured process representations that explicitly capture cooperative work inter-dependencies, exploiting advanced product and software design technologies for process design, and integrating coordination technologies to synergistically combine their strengths and avoid their individual weaknesses.

Keywords

coordination rationale conflict collaborative process model design exceptions enterprise integration

1 WHY WE NEED COORDINATION TECHNOLOGY

Increasingly, work in most enterprises, especially larger ones, is being achieved by large-scale processes distributed across time, participants and functional perspectives. The design of a commercial jet, for example, requires the integrated contribution of thousands of individuals spread over several continents and a span of decades. Effective coordination is critical to the success of such cooperative processes since the distributed activities are typically highly interdependent e.g. due to shared resources, input-output relationships and so on. The sheer complexity of these inter-dependencies, however, has come to overwhelm traditional manual organizational schemes and paper-based coordination techniques, resulting in huge unnecessary rework costs, slowed schedules and reduced product quality. As a result, while individual productivity may be high, failures of existing coordination support practices and technologies have severe impacts on the bottom line.

But what actually do we mean by coordination? Support for coordination can be viewed as being divided into three layers, each built on top of the ones beneath it:

Figure 1. Layers of Support for Cooperative Work.

- **Communication**: allowing participants in the decision process to share information (this involves networking infrastructures)
- **Collaboration**: allowing participants to collaboratively update some shared set of decisions (this involves support for tele-conferencing etc.)
- **Coordination**: ensuring the collaborative actions of the individuals working on a shared set of decisions are coordinated to achieve the desired result efficiently

This paper focuses on the topmost layer, with the understanding the significant challenges exist in providing mature effective technology for the supporting layers. In the conclusion to this paper the requirements that integrated coordination technology presents for the underlying collaboration and communication layers is discussed.

Coordination is only required, of course, when distributed activities are inter-dependent. Dependencies can be distributed in three ways (Figure 2), which has led to the emergence of three distinct types of coordination support technology:

Figure 2. Types of Distribution in Cooperative Work.

- **Distribution across participants** requires support for the sequencing of tasks and flow of information among participants; this is addressed by process management (i.e. process modelling, workflow, simulation, planning/scheduling) technologies.
- **Distribution across perspectives** arises because the participants involved work on different interacting aspects of the decisions and/or have different often-incompatible goals; consistency among their actions can be maintained with the support of conflict management (i.e. constraint or conflict management) technologies.
- **Distribution across time** arises because the nature and rationale for decisions made at an earlier stage often need to be available later on, for example to support product design changes or retrieve solutions with similar requirements; this is addressed by memory management (i.e. rationale capture, organizational memory) technologies.

Each of these technologies face different challenges that have limited their effectiveness in different ways. This paper will review the state of the art of existing coordination technologies, identify their current limitations and identify some promising directions being pursued by the author and others to meet these limitations.

2 CURRENT COORDINATION TECHNOLOGY

For each class of coordination technology we will consider the coordination problem addressed, the current state of the art as well as key open technical challenges.

2.1 Process management

The distribution of cooperative work across multiple participants requires that we be able to control the flow of tasks and information among them, i.e. be able to define, optimize and actually enact effective cooperative work *processes*.

Figure 3. The Lifecycle for Processes.

We can identify the full range of process management functions by considering the process "lifecycle" (Figure 3: ovals represents functions, while "dog-eared" rectangles represent data sets). Process models once defined (or "modelled") must be stored (in some kind of process model library) using a representational formalism. They can then be analyzed to derive summary attributes (e.g. the cost or quality), simulated (e.g. to provide training or an alternate source of summary attributes), or used to guide actual process enactment. All of these can produce metrics reflecting the performance of the process models that can then be used to help improve them. Pre-existing process models can be re-used as examples or building blocks to define new process models that address similar challenges.

We review the technologies for addressing each of these functions in the paragraphs below.

Process Definition: A wide variety of commercial tools, ranging from simple flowcharters to high-end integrated process modelling and simulation tools, have emerged to support process definition. They typically allow one to graphically edit a visual representation of the tasks involved in a process, as well as the resources needed to perform these tasks and the (often conditional) sequencing among them. The tools differ mainly in the underlying representation they use (discussed in the next section) and in the services (e.g. simulation) that are included in the process definition tool.

Current process definition tools have several important limitations:

- **Scalability:** They can be overwhelmed by the sheer scale of cooperative work processes. Even a moderate business process can include hundreds or thousands of tasks. While the use of hierarchical process decomposition can help, it is easy to lose track of where one is, whether all important tasks have been captured, and so on. This problem is greatly exacerbated when we try to include explicit provision for different possible process "exceptions" (i.e. deviations from the "preferred" work process such as mistakes, requirements or resource changes, decision conflicts etc; see below for more detail).
- **Collaborative Definition:** While cooperative work processes typically involve many stakeholders whose inputs should be incorporated to produce an effective model, process definition tools only support single-user editing, producing a significant bottleneck.
- **Formal Quality Standards:** Process definition tools offer little support for ensuring that the processes they capture are of high quality. While many tools allow us to use simulation to assess a processes' effectiveness post hoc, they do not for example include formally defined rules that ensure that processes meet quality standards *as they are being defined*.
- **Process Integration:** The growing utilization of "virtual" organizations (i.e. organizations formed rapidly and opportunistically from teams of smaller organizations to meet a focused need) implies the need to support standard ways of interfacing the business processes used by the team members. Current technology does not yet provide such interfaces. Current EDI transaction sets, for example, support mainly such "arms-length" interactions as bidding and electronic funds transfer.
- **Rationale:** Process definition tools typically do not capture the *rationale* for a process having a particular structure. This can make it difficult for people to understand the current process definition, or to modify it intelligently to adapt it to their own, perhaps differing requirements.
- **Integrating Process and Organizational Design:** Current tools do not integrate process and organizational design. Most process modellers capture little, if anything, about the organization for which the process is designed. No process modellers support co-design of the business process and the organizations that will enact them.

Process Model Library: There are two major classes of process representations: IPO (input-process-output) and speech-act based. IPO representations describe a process as a series of tasks that take several inputs (representing both control and data) and (typically using resources such as people or machinery) produce one or more outputs that are then routed to other tasks (Figure 4):

Figure 4. IPO Process Models Capture Tasks, I/O and Resource Needs.

Examples of IPO representations include IDEF [Bravoco (1985)], CIMOSA [Kosanke (1993)], Petri Nets [Murata (1989)] and Finite State Automata [Peluso (1994)].

Speech-act representations [Winograd (1986), Flores (1992)] take the alternative approach of modelling cooperative work as being built up of prototypical loops of requesting, accepting, completing and accepting delivery of tasks (Figure 5: each loop represents a task, with the customer on the left and the performer on the right):

Figure 5. Speech Act Models Capture Processes as Task Loops.

This representation encourages a useful mental discipline when defining processes in that one is always forced to identify the customer and supplier for every deliverable as well as explicitly consider such steps as customer acceptance of the final deliverable. This does come at the cost however of producing models that can be more complex than IPO representations.

Both IPO and speech act representations as currently conceived face significant limitations:

- **Multiple-Use Models:** Ideally, a process definition tool should be able to capture a single process model that can be used for any process management function (i.e. analysis, simulation, enactment or reuse) - if not we may be forced to re-enter the same process several times. The different uses of process models, however, place different constraints on the nature of the process representation. Process analysis and simulation tools need only task resource need and sequencing information, while enactment tools typically require much more information (such as hooks to legacy systems and electronic form definitions). The representational needs can be incompatible. Some forms of project management analysis tools (e.g. critical path and resource need analysis) require that the process description include no conditional branches, but this is inadequate for simulation and enactment uses.
- **Rigidity/Laxness:** Current process representations make it difficult to balance the need to prescribe a consistent process against the importance of allowing people to use their own judgement and initiative. One can either prescribe in detail the alternative ways a task can be performed (using a sub-process model), or else leave it entirely up to the performer. A model can thus end up either excessively rigid or excessively lax (the temptation is often towards excessive rigidity because it is "safer"); both can be counter-productive. Ideally a process representation should allow us to capture constraints (e.g. use no more than amount X of resource Y) and other guidelines without forcing us to prescribe the detailed procedure, but this is not supported by current representations.
- **Fragility:** Models defined using current process representations can be "fragile" in the sense that they risk becoming irrelevant, misleading or even harmful when unforeseen changes occur in the realities that shaped the original process (Figure 6):

Figure 6. The Factors That Shape Processes

Collaborative processes arise when a given task is too large to be performed effectively by a single individual, and thus must be decomposed (often recursively) into subtasks that taken together achieve the top-level goal. These tasks have interdependencies that may require some serialization in task execution. Some tasks for example may require as input the outputs produced by other tasks. Tasks that both use some limited organizational resource (e.g. a piece of equipment) may be serialized to avoid contention for that resource. Tasks can be merged if it turns out that one task, perhaps expanded slightly in scope, can achieve the requirements for two or more tasks more efficiently. Finally, tasks will be assigned to individuals or groups based on how skills and responsibilities are distributed throughout the organization. Since there are often many different alternatives available for how we do any of these things, processes are shaped by (implicit or explicit) meta-level control strategies that determine which options will be selected. An example of a meta-level strategy for task assignment is "support cross-training by routing tasks to people who need experience with that kind of task" or "maximize throughput by routing tasks to acknowledged experts". Many organizational policies are expressed as these kind of meta-level strategies.

A given process model thus represents a frozen picture of what the "ideal" approach was for a set of constraints that applied at some time in the past. It has been widely recognized, however, that formal process models, to be effective, must also capture what should be done when constraints change, i.e. when an "exception" occurs [Suchman (1983), Grudin (1994), Mi (1991), Karbe (1990)]. We can consider an exception to be any departure from a process that perfectly utilizes the current organizational resources, satisfies all extant policies and encounters no problems during process execution. Exceptions can thus include any significant change in resources, organizational structure, company policy, task requirements or task priority. They can also include incorrectly performed tasks, missed due dates, resource contentions between two or more distinct processes, unexpected opportunities to merge or eliminate tasks, conflicts between actions taken in different process steps and so on. Process models typically include conditional branches to deal with common anticipated exceptions. If unanticipated exceptions occur, however, which they do frequently, we are faced with "patching" the process model without any

computer support to ensure that the changes address the new situation without violating other unchanged constraints. Including explicit branches for exceptions also greatly complicates process models and can obscure the outlines of the preferred process.

- **Poor Support for Ad Hoc Interactions:** Conventional process models are ill-suited to capture highly interleaved and ad-hoc interactions such as those typical of multi-functional design teams. Such interactions are difficult to formalize because strong and unpredictable dependencies among the participants' tasks arise frequently.

Analysis: Process analysis tools provide a way of assessing important process attributes without having to simulate or actually enact them. Examples of such analysis include critical path analysis, resource needs forecasting, concurrency opportunity detection [Eppinger (1993)] and "controllability" assessment [Peluso (1994)]. A key weakness of current process analysis technology, as noted above, is that it typically places severe limits (e.g. no conditionals) on the expressiveness of the process representations it can work with.

Simulation: Simulation tools represent a highly robust and effective technology. One must take great care, however, to use realistic process models, to design the "experiments" (simulation runs) to adequately address the key questions, and to correctly map the masses of low-level statistics typically produced by simulation tools into high-level intuitions about how to improve the process.

Actual Enactment A wide range of process enactment technologies have become available. These technologies can be placed on a continuum according to the extent to which they impose limits on how the process can be enacted (Figure 7):

Figure 7. Types of Process Management Systems

Ad-hoc systems, typically based on shared databases or electronic mail, depend on process participants to decide to whom a work package should go next but ensure that the work package is routed correctly once this decision is made. They also typically provide some degree of tracking. Structured conversation systems such as those based on the contract-net [Smith (1980)] or speech act approaches provide a system-imposed protocol that structures how tasks can be distributed. These typically allow one to track entire "conversations" (series of task assignment negotiations and progress updates) as a unit, but do not constrain what tasks are created or to whom they are assigned. Workflow systems interpret a detailed process model of either the IPO or speech-act type. Participants in a given process simply perform their individual tasks and rely on the workflow system to understand the process and ensure it is followed and tracked correctly.

Many of the limitations faced by current process enactment systems are inherited from the process representations they use. Enactment tools can be, for example, excessively

rigid and fragile because, as we noted above, traditional process models typically are as well. Enactment tools in general use idiosyncratic process representations and can not import/enact models defined by process definition tools. They tend as a result to include organizational models more rudimentary than those captured by many process definition tools, which limits the expressiveness of their languages for describing how tasks should be assigned to agents in the organization. Finally, current process enactment tools, with the exception of small number of high-end systems, do not incorporate the "self-awareness" (e.g. late task detection) and "mutability" (i.e. ability to modify a process while it is executing) features needed to support effective exception handling.

Process Improvement: To date little technology has been developed to provide substantive guidance for re-designing processes to better meet user needs. This is still largely an area where human experience reigns supreme. This kind of expertise is relatively rare and can be difficult to access, however.

Reuse: While it is clearly possible to collect "flat" libraries of process models that can act as examples for the design of new processes, current technology provides limited support for this function. The lack of a widely accepted process representation standard, of course, is one major problem. Other gaps include representational schemes that allow one to quickly find processes relevant to one's problem, as well as tools to help ensure that one does not unknowingly violate any process requirements when customizing a template to fit one's particular needs.

Overall Issues: One problem that cuts across all currently available process management technologies is their lack of integration. Process models defined by a process definition tool, for example, are typically not executable by a process enactment tool. Process enactment tools also typically utilize their own rudimentary organizational models (i.e. model of the resources in an organization) rather than exploit the more expressive organizational models captured by many process definition tools.

2.2 Conflict management

Cooperative work is typically distributed across multiple functional perspectives addressing inter-related aspects of a single set of decisions. Maintaining consistency (i.e. avoiding conflicts) among these distributed activities becomes a significant challenge with major potential impacts on cost, quality and timeliness.

Current conflict management approaches are almost entirely manual, relying on coordination memos, cross-functional team meetings and the like. Such approaches have become increasingly costly and ineffective, however, as the sheer scale of cooperative activities have grown. Conflict management technology is emerging to address these issues by providing computer support for detecting [Stefik (1981), Mark (1994)] and resolving [Sussman (1980), Fox (1984), Descotte (1985), Brown (1985), Marcus (1987), Goldstein (1975), Hewitt (1986), Wilensky (1983), Lander (1988)] conflicts. This technology relies in general on the fact that there are relatively few abstract classes of conflict and associated general strategies for resolving them. Conflict resolution can then be achieved by heuristically classifying the conflict [Clancey (1984)] and then instantiating an associated strategy to generate a suggested conflict resolution approach [Klein (1991)].

Determining the cross-perspective impact of decisions require representation standards capable of representing decision semantics throughout the life cycle. Existing standards do not in most domains provide adequate coverage or expressiveness yet. Another key challenge is improving the scalability of the decision impact assessment process;

untrammeled dependency propagation can quickly become computationally intractable and overwhelm agents with floods of information on the impact of multitudes of changes made elsewhere. Abstract or qualitative representations of decisions and dependencies are needed to allow meaningful if approximate assessments of decision impacts at significantly reduced computational cost. We also need to be able to specify context-sensitive policies concerning what kind of dependency impact detection should be done when. Finally, better support needs to be provided for computer-supported conflict resolution. The first step in resolving a conflict is typically to understand how it occurred and why. Current conflict management technology records what other decisions impinged on this one but doesn't keep track of the intent or history (e.g. rejected options) behind the decisions. We need to be able to access a rich representation of the rationale behind conflicting decisions. In addition, the resolution of a conflict is typically reached through a multi-step process (e.g. negotiation over resource assignments) involving the people who produced the conflicting decisions. These processes need to be integrated into a general and robust process management approach.

2.3 Memory management

The distribution of cooperative work across time implies the need to remember the reasoning underlying decisions made throughout the life cycle. When a set of decisions is made the typical output is made up of documents describing the final result of a long series of deliberations and tradeoffs. The underlying history, intent and logical support (i.e. the rationale) for the decisions captured therein, however, is usually lost, or is represented at best as a scattered collection of paper documents, project and personal notebook entries as well as individual's recollections. This rationale information can be very difficult to access by humans and is represented such that computers can provide little support for managing and utilizing it.

Many decision rationale capture approaches have been developed (e.g. [Lee (1991)], [Yakemovic (1990)], [McCall (1987)], [MacLean (1991)], [Fischer (1991)]). Shortcomings with existing approaches include limited expressiveness and therefore limited computational usefulness due to lack of integration with decision capture technologies. They have for example the potential for inconsistency between the rationale and decision descriptions, spotty capture of rationale and the tendency to waste time on issues that later prove unimportant [Fischer (1991)]. Existing rationale language technology has been used mainly to capture the pros and cons of alternative solutions for a given problem, but not intent, the relationships between decisions or their history, all of which, of course, can be extremely important. Rationale capture can be burdensome, especially given that the person who benefits is generally different than the person required to describe it. This is exacerbated by the fact that it is often unclear where and in how much detail rationale should be described, since the person who enters the rationale is unlikely to know how it will be used or even by whom. We should be able to describe preferred rationale capture processes to delimit what and how detailed rationale should be captured. If possible, these preferred processes should provide default rationale templates that can be quickly customized to describe the rationale for a given decision.

3 PROMISING DIRECTIONS

3.1 Integrate coordination technologies

Many of the weaknesses of current coordination technologies derive from the fact that they evolved separately and are not as a result functionally integrated. Truly effective coordination, however, requires an integrated approach wherein dependencies across all the dimensions of cooperative work distribution are modelled and managed in a single computational framework [Klein (1995)]. The figure below summarizes some of the ways in which this integration can synergistically combine the strengths and avoid the weaknesses of the component coordination technologies (Figure 8):

Figure 8. Synergism Among iDCSS Component Technologies.

Rationale capture technology provides argumentation and intent information for process (and product) decisions. This can be used to support process redesign (e.g. allowing us to avoid process changes that violate important goals). Rationale capture derives increased expressiveness from the notions of process execution history and prescriptive constraints. We can also use process templates to delimit what and how much rationale information needs to be captured for given decisions (e.g. by specifying for which sub-goals we need to justify the way the goal was achieved), helping to reduce the potential rationale capture burden. Process management technology benefits from the exception handling capabilities made possible by generalizing conflict management technology. The beginnings of an approach to integrating coordination technologies is described in [Klein (1995)].

3.2 Beyond traditional process representations

As we have seen, process management plays a pivotal role in the area of coordination technology. Many of the problems with current process management tools stem from two properties of traditional process representations:

- Process models are "compiled" responses to highly fluid work interdependencies. Changing a process model to account for new contingencies is akin to changing a computer program by patching the executable - a difficult and error-prone undertaking.

- Process models can be too highly prescriptive. In many cases it is appropriate to give process performers significant discretion in the steps they perform subject to general constraints such as total allowable resource usage. Current representations are designed to capture only task decomposition/sequencing constraints.

The fundamental problem is that while process templates represent a good way of capturing some kinds of procedures they are the wrong level of abstraction to fully capture coordination expertise. Rather than tweaking process models we should be identifying the kinds of underlying work inter-dependencies that exist in our work environments and uncovering/inventing the coordination mechanisms (including for example traditional process models, speech act approaches, asynchronous messaging or multi-functional teams) that work for these kinds of dependency structures [Malone (1991)]. Computer systems can then help us opportunistically identify and apply the appropriate coordination mechanism as needed. When the underlying work dependencies change, "exception handling" tools [Klein (1995)] can detect this and help us decide whether to incrementally modify the current coordination mechanism (e.g. by "patching" a process model) or select an entirely different one. In a design context, for example, unexpectedly severe impacts of design innovations on manufacturability could motivate us to replace a serial design process with multi-functional design teams for a particular product. For this to work, of course, we need to address the issues of how we can represent and maintain explicit models of organizational goals and constraints as well as work inter-dependencies.

We also need to invent a wider range of approaches for specifying coordination, including some that are "semi-structured" (i.e. only partially prescriptive). One promising approach is to use "proscriptive" rather than "prescriptive" constraints. Such constraints describe the limits on the desired result (e.g. don't conflict with decision Z) and the process used to achieve it (e.g. don't use more than amount X of resource Y) rather than how the decision should be made.

We can summarize the differences between traditional process models and the flexible coordination approach proposed here as follows:

	Traditional Process Models	Flexible Coordination Approach
Representational depth	"Compiled" process descriptions only.	Explicit representation of organizational goals, constraints and dependencies, along with the coordination mechanisms used.
When coordination mechanisms are selected	Processes are fully defined ahead of time.	Coordination mechanisms can be opportunistically selected given current coordination requirements.
Prescriptiveness of coordination mechanism	Highly detailed prescriptive models.	Semi-structured (possibly proscriptive) models.

Adaptability of coordination mechanism to exceptions	Processes are adhered to as rigidly and consistently as possible. Exception handling if any is pre-defined.	Coordination mechanisms are sensitive to exceptions and adapt themselves appropriately.

Note that this is not suggesting that traditional process models be abandoned, but rather that they be incorporated into a broader more flexible approach. In some cases (e.g. most kinds of manufacturing) it is important to maintain consistent high-level processes to achieve consistent product quality. It should be noted, however, that even highly formalized work environments can be surprisingly exception prone.

3.3 Exploit product & software design technologies

Many of the challenges faced by current process management technology have been addressed by research on physical artifact and software system design. Tools for requirements specification, rationale capture, cooperative design, case-based design and exception handling have been developed in these contexts, and are likely to provide important insights into how analogous problems can be dealt with in the process world. While some initial work has been performed, e.g. in applying case-based [Malone (1991)] and collaborative design [Dean (Winter 1994-95)] approaches to process definition, this direction is worthy of more investigation.

4 REFERENCES

- Flores et al. (1992) The Action Workflow Approach to Workflow Management Technology, in *Proceedings of CSCW '92*.
- Bravoco, R.R. and Yadav, S.B. (1985) Requirements Definition Architecture - An Overview. *Computers in Industry*, 6, 237-251.
- Brown, D.C. (1985) *Failure Handling In A Design Expert System*. Butterworth and Co., November.
- Clancey, W.J. (1984) Classification Problem Solving. *AAAI*, 49-55.
- Dean, D.L.D., Lee, J.D.L., Orwig, R.E.O., and Vogel, D.R.V. (Winter 1994-95) Technological Support for Group Process Modelling. *Journal of Information Management Systems*, 11 (3) .
- Descotte, Y. and Latombe, J.C. (1985) Making Compromises Among Antagonist Constraints In A Planner. *Artificial Intelligence*, 27, 183-217.
- Eppinger, S.D., Whitney, D.E.W., Smith, R.P.S., and Gebala, D.A.G. (1993) A Model-Based Method for Organizing Tasks in Product Development. Technical Report. Working Paper 3569-93-MS, May.
- Fischer, G., Lemke, A.C., McCall, R., and Morch, A.I. (1991) Making Argumentation Serve Design. *Journal of Human Computer Interaction*, 6 (3-4) , 393-419.
- Fox, M.S. and Smith, S.F. (1984) ISIS - A Knowledge-Based System For Factory Scheduling. *Expert Systems*.
- Goldstein, I. (1975) Bargaining Between Goals, in *IJCAI* 2, pp. 175-180.
- Grudin, J. (1994) Groupware and Social Dynamics: Eight Challenges for Developers. *Communications of the ACM*, 37 (1) , 93-105.

- Hewitt, C. (1986) Offices Are Open Systems. *ACM Transactions on Office Information Systems*, 4 (3), 271-287.
- Karbe, B.H. and Ramsberger, N.G. (1990) Influence of Exception Handling on the Support of Cooperative Office Work, in *Multi-User Interfaces and Applications*, Gibbs, S. and Verrijin-Stuart, A.A., Eds. Elsevier Science Publishers, 355-370.
- Klein, M. (1991) Supporting Conflict Resolution in Cooperative Design Systems. *IEEE Systems Man and Cybernetics*, 21 (6) .
- Klein, M. (1995) iDCSS: Integrating Workflow, Conflict and Rationale-Based Concurrent Engineering Coordination Technologies. *Concurrent Engineering Research and Applications*, 3 (1).
- Klein, M. (1995) Conflict Management as Part of an Integrated Exception Handling Approach 1995.. *AI in Engineering Design Analysis and Manufacturing (AI EDAM)*, 9, 259-267.
- Kosanke, K. (1993) *CIMOSA: Open System Architecture for CIM*. Springer Verlag.
- Lander, S. and Lesser, V.R. (1988) Negotiation To Resolve Conflicts Among Design Experts. Technical Report. August.
- Lee, J. and Lai, K.Y. (1991) What's In Design Rationale?. *Human-Computer Interaction*, 6 (3-4), 251-280.
- MacLean, A., Young, R., Bellotti, V., and Moran, T. (1991) Questions, Options and Criteria: Elements of a Design Rationale for User Interfaces. *Journal of Human Computer Interaction: Special Issue on Design Rationale*, 6 (3-4), 201-250.
- Malone, T. and Crowston, K. (1991) Towards an Interdisciplinary Theory of Coordination. Technical Report. Technical report 120.
- Marcus, S., Stout, J., and McDermott, J. (1987) VT: An Expert Elevator Designer. *Artificial Intelligence Magazine*, 8 (4) , 39-58.
- Mark, W. and Dukes-Schlossberg, J. (1994) Cosmos: A System for Supporting Engineering Negotiation. *Concurrent Engineering Research and Applications: Special Issue on Conflict Management in Concurrent Engineering*, II (3), 173-182.
- McCall, R. (1987) PHIBIS: Procedurally Heirarchical Issue-Based Information Systems, in *Proceedings of the Conference on Planning and Design in Architecture*. ASME, Boston MA.
- Mi, P.W. and Scacchi, W. (1991) Modelling Articulation Work in Software Engineering Processes, in *Proceedings of the First International Conference on the Software Process*, IEEE Computer Society Press, 188-201.
- Murata, T. (1989) Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77 (4), 541-580.
- Peluso, E., Goldstine, J., Phoha, S., Sircar, S., Yukish, M., Licari, J., and Mayk, I. (1994) Hierarchical Supervision for the Command and Control of Interacting Automata, in *Proceedings of the Symposium on Command and Control Research*. June 21-23 .
- Smith, R.G. (1980) The Contract Net Protocol: High-Level Communication And Control In A Distributed Problem Solver. *IEEE Transactions on Computers*, C-29 (12), 1104-1113.
- Stefik, M.J. (1981) Planning With Constraints. *Artificial Intelligence*, 16 (2), 111-170.
- Suchman, L.A. (1983) Office Procedures as Practical Action: Models of Work and System Design. *ACM Transactions on Office Information Systems*, 1 (4), 320-328.
- Sussman, G.J. and Steele, G.L. (1980) Constraints - A Language For Expressing Almost-Hierachical Descriptions. *Artificial Intelligence*, 14, 1-40.
- Wilensky, R. (1983) *Planning And Understanding*. Addison-Wesley,
- Winograd, T. (1986) A Language/Action Perspective on the Design of Cooperative Work, in *Proceedings of CSCW '86*.
- Yakemovic, K.C.B. and Conklin, E.J. (1990) Report on a Development Project Use of an Issue-Based Information System, in *CSCW 90 Proceedings*. 105-118.