KARLSKRONA
RONNEBY

Research Report

1/94

Area: ***Computer Science***

"Societies of Computation: A Framework for Computing & Communication"

by

Rune Gustavsson and Staffan Hägg

ISSN 1103-1581
ISRN HKR-RES --94/l--SE

Högskolan i Karlskrona/Ronneby, **S-371 79 Karlskrona Tel: 0455 - 780 00 Fax: 0455 - 780 97**

SoC

Societies of Computation

Societies of Computation

A framework for Computing & Communication

ABSTRACT

Societies of Computation (SoC) is proposed as a frame work for design of real world complex systems involving computing and communication . The computational paradigm behind SoC is communities of rational agents. The framework has three components; Foundational research, Generic technologies and Applications.

Table of contents

1. Summary

One big challenge for the next decade is to build systems that can interact productively with each other, with humans, and with the physical world. The interaction between agents, human and artifacts, requires establishing, maintaining, and extending common grounds. We need systems that are open and can grow; that allocate resources fairly to get jobs done without requiring either centralized planning or excessive communication.

We propose a generic architecture, Societies *of Computation* (SoC), as a framework for developing and assessing issues concerning decentralized intelligent computing. SoC integrates advancements in distributed computing and design of knowledge intensive systems in the view of the advantages of system design at the knowledge level. Design at the knowledge level focus on implementation independent issues and has proven to simplify design and maintenance of complex systems.

In the SoC framework, common ground between human and computational systems can be aided by mediators that link the capabilities and natural properties of task agents, people, and the world we live in. With agents with multiple goals, coordination of actions must proceed through mutual commitments.

In working with intelligent agents, it is useful to explore what is happening in such fields as cognitive psychology, anthropology, economics, neural modeling and the whole of computer science.

In the first stage of developing Societies of Computation, we will, however, concentrate on building test beds for integrating and coordinating artificial agents. Specific issues concerning system-user interactions, i.e., HMI and CSCW issues, are investigated at later stages of the project.

Details of the current set of projects in the SoC framework are described elsewhere.

2. Introduction

Modern societies depend on large distributed systems in applications as diverse as telecommunications, enterprise integration, power distribution, transport, air traffic control, battle-field simulations, production systems, medical services, and citizen service networks. All these systems require careful management and control if they are to provide and acceptable level of service to the community. Similarly, complex systems such as railways, integrated traffic systems, power stations. steel manufacturing require sophisticated and flexible management and control. Despite significant automation in these areas, the resulting growth of these systems has meant that they have become highly complex and thus increasingly difficult for human operators to manage and control efficiently.

The automation of these management processes can be expected to improve productivity and safety, increase versatility, and reduce demands and stress on operational personnel. However, conventional computational systems are not well suited to this purpose.

Such systems are usually very inflexible and unresponsive to the skill level of the operators using them. The control and diagnostic procedures that are used cannot be

matched to the exigencies of the current situation, nor can they cope with reconfiguration or modification of the processes under control. Performance on tasks cannot be guided by useful advise from technicians and, when given a task that cannot be performed, no explanation is given as to the cause of failure. Moreover the software itself is difficult and expansive to develop, modify, and maintain.

Furthermore there is ample evidence that although some systems are very well technically designed, they are nevertheless not use due mainly to two factors. The systems do not fit into the organization for some reason and or the systems are not accepted by its users.

To achieve the benefits of automation, it is thus necessary to develop new methodologies as well as new techniques in order to design software systems to overcome these problems. Our proposed architecture of Societies of Computation is very well suited for enterprise design, or business re-engineering utilizing knowledge technology (KT). We will however in this context presuppose that the processes and their supporting knowledge have been identified and concentrate on technologies to develop software systems implementing and integrating processes with the following critical characteristics of the applications. In particular, the types of system management tasks mentioned above have the following features:

- the data an knowledge regarding a particular operation are distributed;
- the reasoning required to make a decision has to be carried out at more than one location;
- communication and control is required between geographically separated centers;
- the system has to respond in real time; and
- there is continuous change and that change affects the reasoning process.

In the project Societies of Computation we consider the design and application on systems based on methodologies adapted from the Esprit projects Kads-II, *CommonKads,* and COMIC. CommonKads defines new methodologies for Software Engineering (SE) with special emphasis on a risk driven process model and the use of reusable component in the development of the target systems. In COMIC we are at this point assessing different kinds of Mechanism of Interactions (MoI). These mechanisms are studied in COMIC as building blocks of systems of Computer Supported Cooperative Work (CSCW).

We furthermore build on results and prototypes developed at Australian Artificial Intelligence Institute, AAII, and at Stanford University. The prototypes for assessments include the *Procedural Reasoning System,* PRS, and *Agent Oriented Programming,* AOP. Appropriate high level models of communication between agents are of outmost importance in order to cope with the distributiveness and situativiness of our applications. The results and prototypes of the NSF- ARPA *Knowledge Sharing Effort* are, in these respects, valuable inputs to the SoC project.

SoC focus on the design and application of systems based on Artificial Intelligence (AI) technologies, utilizing state-of-the art methodologies, that are aimed at providing these kinds of capabilities. Such systems are called *distributed situated systems.* The SoC framework stresses and uses an agent oriented architecture for implementation of such systems.

3. Societies of Computation (SoC)

The development of distributed situated systems is still in its infancy. Some systems have been implemented and even tested on realistic applications such as air traffic control, production planning and scheduling, network management and concurrent engineering. In those examples focus has either been on local, stand alone, situated systems or communication and coordination issues. Very little research has so far been aiming at understanding the foundations and generic techniques that are applicable in domains as have been described above. The primary aims of the SoC research are to conduct foundational research into distributed situated systems; to develop generic technologies for collaborate problem solving, where groups of such systems cooperate with each other, share common goals and provide global solutions to the problems; and to demonstrate these technologies to selected applications such as management of power transportation systems.

In the following sections of this chapter we briefly review the research carried out in relevant areas of distributed and situated systems. We then shortly describe the concepts and ideas behind the framework of Societies of Computation.

3.1 Situated Reasoning Systems

One branch of AI that has received a great deal of attention and is now becoming commercially successful is the area of Knowledge Based Systems, KBS. Knowledge based systems are AI systems that encode domain-specific knowledge or expertise of a specialist in some field and utilize the knowledge to reason about specific situations.

Most of the early knowledge based systems tackled problems that were predominantly static in nature, i.e., they worked under static *environment* assumption. These systems were used to prove mathematical theorems, solve simple planning problems in an idealized domain, perform diagnostics, and synthesis problems such as design or rather configuration of equipment. Unfortunately, many if not most practical problems are not static; they are dynamic in nature.

Recently, considerable work has been undertaken on the development of technologies that overcome this limitation. Systems that relax the static environment assumption are called *situated reasoning systems*.

A situated reasoning system must be capable of receiving information from a variety of sources (or sensors) in an asynchronous fashion and, from the information and previously acquired knowledge, be able to assess the current situation. Having assessed the current situation and evaluated the overall goals, the system has to commit to a certain course of action, usually specified as a rule, plan, or procedure. While executing this course of action, the system has to continuously monitor the world and reassess the situation.

One of the most difficult - and least considered - problems facing designers of such systems is how to manage the execution of these rules and procedures under the stringent time constraints typical of many real world applications. Where the rules and procedures are relatively simple and independent of one another, execution can be effected in some straightforward manner, such as concurrently or on a first-come, first-served basis. However, when rules and procedures can take an arbitrary long time to

execute, and possibly involve other rules or procedures, it is essential to be able to reason about the *management* of these tasks. This may require reasoning about which tasks need to be performed to realize other tasks, the criticality or urgency of tasks (task utilities), the potential interactions among tasks, the order in which tasks should be performed, and which tasks need to be suspended or resumed given the current state of the system. Most importantly, all this must be done while continuing to attend to the process under control and reacting appropriately to changing circumstances.

A number of approaches have been proposed to address these issues. *Real-time knowledge based systems* are extensions of traditional knowledge based systems that provide facilities for temporal reasoning, focus of attention, continuous operation, truth maintenance, and interprocess communication. However, they usually use technologies that are not scalable and of other reasons are not suitable for real-world applications of the kinds we are interested in. In short they are build using adoptions of shells for static domains.

Blackboard architectures have been proposed as another alternative to implement situated systems. The main components of a blackboard architecture are the blackboard database and the knowledge sources. The former contains publicly accessible information and the latter represents the procedural knowledge that reads, uses, modifies, and writes into the blackboard database. The architecture also provides mechanisms for meta-level control in choosing the next knowledge source to execute. However, this meta-level control lacks flexibility and is not interuptable, which can pose a serious problem for providing real-time behavior.

The third approach to the design of situated systems is to base the architecture of the systems on the notion of a *rational agent.* In this approach, the system is viewed as consisting of agents. The agents have certain *mental attitudes* which in turn influence their decision making and determine their behavior. The simplest of these models for individual agents, called *BDI architectures,* are base on attitudes of *belief, desire* and *intention.* The first two attitudes represent, respectively, the information state and the motivation state of the agent. The last represents decisions the agent has made at a previous time, and is critical for achieving adequate or optimal performance when deliberation is subject to resource bounds.

The question for the system designer is to determine the role these, and other, attitudes play in governing the rational (effective) behavior for a system, required to act in a dynamic world, and subject to constraints on both time and information. In particular the reasoning and planning performed by the system must all be carried out in a continuously changing environment, requiring that the system appropriately balance the time taken thinking against the time needed for acting.

Generic Blackboard systems like BB1, RT and BDI-base systems like PRS and RAPS have developed in parallel with the foundational research into situated reasoning. These systems have been used in a number of applications. For example, BB1 has been used in medical monitoring. PRS has been used in real-time diagnosis for subsystems of the NASA Space Shuttle, in air traffic management, and telecommunications network management. Agent oriented architectures, using a somewhat different set of mental attitudes than the BDI-architectures, have also been tested at Stanford in application such as concurrent engineering and other design applications, PACT and SHADE.

3.2 Distributed and Decentralized Artificial Intelligence

Many early KBS assumed that a single system, besides the user, was solving the problem, i.e. they worked under the *single agent assumption.* However, for a majority of practical problems there is more than one expert or system involved in the decision-making process. Experts cooperate with each other to form joint plans and solve problems that any of them cannot solve individually. Systems that relax this single-agent assumption are called *distributed* or *decentralized* systems.

The collaborative solution of a global problem by a distributed group of systems or entities is referred to as Distributed Artificial Intelligence (DAI) if the entities are artifacts with some intelligence. The term Computer Supported Cooperative Work (CSCW) refer to systems that mainly are mediators between human problem solvers. Distributed systems as such is a well-known area in computer science.

On the other hand, a group of autonomous systems collaborating with each other to solve global problems is called Decentralized Artificial Intelligence (DzAI). In both DAI and DzAI, data and control may be logically and geographically distributed amongst the group. As a result, sharing of information and decisions is usually required to collaboratively solve a problem.

In DAI the global problem is given and the distributed entities have to be designed to solve it. The main issues in DAI are problem decomposition, problem assignment, and solution synthesis. In DzAI the autonomous systems already exists and the collaboration among them needs to be designed to solve global problems. The main issues in DzAI are problem identification, role assignment, and solution monitoring.

The boundary between DAI and DzAI starts disappearing when one considers autonomous systems that can create or request help from other systems. Issues related to communication, coordination, negotiation, and organizational structures are central to both DAI and DzAI.

3.3 Distributed Resource Management Systems

In Computing & Communication systems, the basic idea is to combine trends in respective areas in order to allow reuse of components as well as implementations of new applications utilizing for instance different kinds of electronic highways.

Many new applications involves several heterogeneous, interconnected subsystems. New technologies enables more functionality, such as the integration of applications possibly by remote access, using high speed communication links, and more user-friendly (and user-adaptive) human-machine interfaces. Subsequently, the need for more sophisticated communication and distributed resource management systems, DRMS, is well recognized and an area of much research and development. The purpose of such a DRMS is to bridge the heterogeneity of the interconnected systems, and to be adaptable to new situations, such as load changes, reconfigurations, and system failures.

Two approaches for addressing problems of distributed systems, the *communication* approach and the *distributed systems* approach, will be discussed briefly below.

In the communication approach (e.g. the IS0 OSI Reference Model), communication is handled in a layered fashion. The lower layers are dedicated to data communication layers. The higher layers are application integration layers. The data transportation layers can be useful for any low level DRMS. However the application integration layers fall short of the goals of a flexible high level DRMS due to at least the following reasons:

- The approach is based mainly on peer-to-peer communication mechanisms, thus leaving most of application integration to the system implementor.
- There is no generic mechanism for handling different application domains, this in turn leads to narrow definitions of application oriented protocols.
- There is no generic mechanism for handling system changes, this often leads to static systems and a lot of maintenance problems.

In the distributed systems approach (e.g. Mach-OSF), the general idea is to extend the standard operating system functions to network services, providing transparency of network resources to the users. The goal is definitely a flexible DRMS, but the task undertaken is very hard, and projects on distributed systems have shown that some of the difficulties are:

- Conflicting goals of transparency and efficiency.
- Accelerating needs of communication.
- Adaptability and generality is hard to describe and implement, which leads to small generic kernels and a lot of specialized code.

A main obstacle to many of the efforts to constructing a flexible DRMS (with either of the two approaches outlined above, is that conventional description and programming methods are too implementation dependent. We need a way of reasoning, in a principled way, about heterogeneity and change as such.

The SoC framework views a distributed systems as a collection of rational agents. This high level description allows reasoning about and addressing some problems of building a flexible DRMS. In SoC we have agents that;

- can reason about heterogeneity and change, which makes the society flexible to new situations,
- have a social behavior with learning capabilities, which enables one to minimize communication needs,
- can negotiate and make social commitments, giving flexibility to problem solving behavior,
- are inherently distributed, well fit for modular and robust implementations with distributed control,
- can easily be given real-time properties, which is essential for a DRMS.

Two application areas for DRMS's are given special attention in the SoC project

Distributed power distribution management. Power distribution companies are searching for flexible distribution management systems with distributed control and

more functionality for the end users. One effort in this area is the ESPRIT project ARCHON, which uses an agent oriented architecture.

Distributed communication network management. This application area has many parallels to the former one, demanding real-time planning with distributed control. Efforts in this area with agent oriented architectures are made at GTE Labs, Waltham, USA, and at the Australian Artificial Intelligence Institute.

3.4 Agent oriented interaction

Multiple autonomous systems communicating with each other require a common communication language and a well-defined protocol. Apart from the details of the expressive power and the pragmatics of communication acts, one should also take into account *what* information needs to be communicated, e.g. beliefs, goals, plans or intentions of agents; *when* a communication should take place, e.g. failure of a joint plan of action, failure of a goal, establishment of a team, contradiction between the local and global states; and *how* the communication should occur, e.g. broadcast or directed, synchronous or asynchronous, and whether the means for communication is unreliable or imperfect (i.e. noisy). While many of these issues have been studied in distributed processing and distributed systems, DAI addresses these issues within a more flexible and expressive system architecture.

When autonomous systems interact with each other they need to coordinate their actions by exploiting positive interactions and avoiding negative interactions. Furthermore, coordination of actions may become a necessity when multiple agents have to jointly do an action, which cannot be done independently by any one of them. Coordination requires reasoning about the other agent's mental state, communicating the information necessary for coordination, and then committing to a joint plan of action.

In the above process of solving a global problem through the coordinated activity of a group of agents, the mental states of all the agents will not be identical; sometimes they may be contradictory. After identifying their differences, the agents need to settle these and come to a mutual agreement regarding the joint activity they want to pursue and the means of pursuing this joint activity. For this purpose, game theory is often used to model negotiation under a number of different assumptions.

The distribution of the systems and their organizational structure plays an important role in all of the above joint activities. The organizational structure dictates the direction and content of information and decision flow within the distributed system. The organizational structure could be hierarchical, market like or hybrid. The structure may change dynamically based on the change in the environment.

Generic technologies such as Contract net protocols, Agent Oriented Programming and technologies developed within the ARPA-NSF Knowledge Sharing Initiative have been developed to address issues in communication, coordination, and negotiation, They have been applied to distributed diagnosis, distributed air traffic control, distributed network management, distributed vehicle monitoring, and concurrent engineering.

4 SoC - A Framework

We will describe the research and development performed using the SoC framework. The research and development involve activities performed in three parallel strands: *foundational research, generic technologies,* and *application domains.* In the following three sections we briefly describe the issues in the different strands.

4.1 Foundational Research

From assessments of experiences in the design of situated systems, we believe that the adoption of a BDI-architecture offers a better conceptual framework to address and solve issues related to situated reasoning, the other alternative approaches. In particular, this approach provides the philosophical and theoretical foundations as well as the flexibility to model different types of rational behavior. We thus start from the BDI-architecture and extend it to distributed situated systems.

Before one can build generic technologies for distributed situated systems based on the BDI-architecture, one needs to develop a theory of how such systems should behave and interact with the environment. First, we will investigate the logical foundations of BDI-architectures. In particular, we will assess current ideas of formalizations of the notations of belief, goal, and intention, and examine a number of properties related to their interactions. We will investigate the notion of commitment and model different types of commitment as constraints on how interactions have to be maintained from one time point to another. This logical theory will serve as the foundation for distributed situated systems.

Having described the logical theory of situated systems we will investigate two important processes of situated systems - *means-end reasoning* and *deliberation.* Means-end reasoning involves finding a sequence of actions that satisfies a certain end goal (or goals). Often, there is more than one such sequence. Deliberation addresses the problem of weighing alter-native sequences of action and choosing the best plan of action according to some well defined criteria, such as maximizing expected utility. Both these processes together lead to the selection of a best combination of actions that satisfies an end goal, which is then adopted as an intention.

The classical planning approach transforms the current state of the system to the desired goal state by sequencing a number of primitive actions or operators. This process result in a combinatorial explosion of states and is therefore unsuitable for means-end reasoning in situated systems that have stringent real-time constraints. We will follow the approach taken by researchers at AAII and introduce the notion of a *plan* as an abstract structure which embodies the means (or sub-goals that need to be satisfied and the order in which they need to be satisfied) to achieve a certain end goal. Plans capture the hierarchical and partial nature of means-end reasoning. Having plans as abstract structures need to be distinguished from intending a plan and executing a plan. The distinctions between these concepts will be formalized within the BDI-architecture.

Analyzing alternative decisions and choosing the best decision has been studies in a branch of mathematics and economics called *decision theory.* As a result, decision theory is a natural choice to investigate the process of deliberation within situated sys-

terns. However, a major problem in this investigation is the different representations and different computational paradigms used in decision theory and situated systems: the former uses decision threes and involve numeric computation, while the latter uses logic and involves symbolic computation. To address this problem there are suggestions to extend the symbolic model of the BDI-architectures to a numeric model. In particular, numeric quantities can be attached to symbolic entities: probabilities for beliefs, ***payoffs*** for goals, and utilities for intentions. This unified theory seems to facilitate symbolic and numeric computation essential for situated systems.

Next we will extend the single agent logical BDI model to a team of rational agents. A team is a collection of other agents or teams that share a common purpose and are willing to co-operate with each other for the common good of all its members. In addition to individual beliefs, goals, and intentions, a team of rational agents will also have

- mutual beliefs about the world and about each others actions
- joint goals that need to be achieved
- compiled joint plans or recipes, that specify the means of a joint goal or an external event
- committed joint plans or joint intentions, adopted as a response to a joint goal or an external event

We extend the notion of plans as abstract structures to encompass actions performed by other agents, and the coordination and the synchronization required for joint activity. We refer to these plans as joint plans and provide the syntax and semantics of these entities. We will also formalize joint mental attitudes such as mutual beliefs, joint goals, and joint intentions. Furthermore, we will investigate the role of social ***plans*** as a mean to control communication between agents. Ontologies and agentifications are means to allow communication between heterogeneous agents. Heterogeneous agents are typical components of an integrated enterprise.

4.2 Generic Technologies

The foundational research provides the logical and conceptual framework to investigate critical issues in the design of distributed situated systems. As a result, a number of generic technologies have been and will be developed to support the implementation of distributed situated systems. We will however develop and test generic technologies which can be assessed in the SoC framework.

A Society of Computation (SoC) is a knowledge level description of co-existence. The architectural components of Societies of Computation are Agents and Knowledge Media, KM. Agents are of three types; Personal Assistants PA's, Mediating Agents MA's and Artificial Agents AA's.

Knowledge Medias provides Mechanisms of Interactions in the COMIC sense. Examples are the Collaborative Desktop (CoDesk) and DIVE developed within the Multi-G framework. CoDesk provides a workplace utilizing video as a communication media and shared objects. DIVE is a MoI based on Virtual Realities.

Personal Assistants are the agents direct serving human members of a Society of Computation. Recent research in HMI, KBS and human centered design have identified agents such as PA's for dialogue management and intelligent help and guidance.

The following figure gives the architecture of SoC.

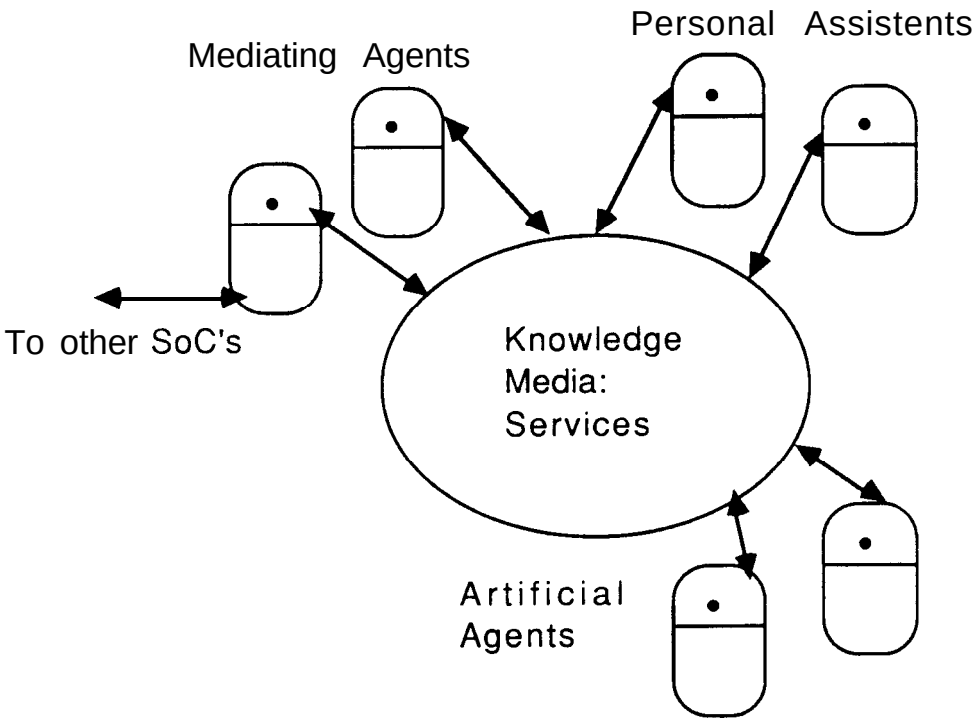

Figure 1. Architecture of Societies of Computation

Artificial Agents are in the spirit of Agent Oriented Programming and contemporary DAI or DzAI.

The communication patterns are derived from task distribution, organization structures and available MoI's. Those issues are partly addressed in the CommonKads methodology, the ARPA-NSF Knowledge Sharing Initiative, and in COMIC.

The Mediating Agents consistute the public and service sector of the SoC. They enables the cooperation of PA's and AA's, utilizing the MoI's of the Knowledge Media at hand. The Mediating Agents also serves as gateways to other Societies of Computation.

The agents, PA's, MA's and AA's have all a head-body architecture. Of special interest for development of SoC is the concept of social laws introduced in AOP. The ARCHON architecture for special purpose SoC, technical diagnosis, is also of interest to SoC.

4.3 Application Domains

4.3.1 Energy systems for the future

The energy market in Sweden moves towards a deregulation. This introduces a greater system complexity with several new players and market oriented activities. As a consequence, the need for control and management is increasing. New metering technology must be developed and systems built to handle this greater complexity. Large distributed systems utilizing technical communication and local computers will be applied in the future.

The open energy market have an impact on the distribution of energy. The electrical network can be compared to a transportation company delivering goods from different producers. In this new situation, automation of distribution, and intelligent support for management and control will be needed. Communication between geographically separated centers is one of the tools in efficient operation of the distribution of energy.

The new market is an area open for modern computing and communication. Sydkraft AR, producing and distributing electricity for the Swedish market, participates in the development of SoC by building test beds for the integration and co-ordination of artificial agents applied on energy systems.

4.3.2 Ubiquitous computing with focus on learning and informational retrieval, including media&arts

SICS is invited to contribute parts of the SoC frame work to the planned Center of Computing & Communication Systems at the Royal Institute of Technology (RIT).

The focus of the proposed center is on future telecommunication systems.

5 References

[Astor, Davidsson, Ekdahl, Gustavsson] *Anticipatory Planning,* LU-CS-TR:90-69, Dep of Computer Science, Lund University, 1990. Also in Advance Proceedings of the European Workshop on Planning, 1991.

[Astor, Davidsson, Ekdahl]: *Anticipatory Autonomous Agents,* LU-CS-TR:93-12, Dep. of Computer Science, Lund University, 1993. (in preparation).

[Balkenius, C.] Natural intelligence for autonomous agents, In B. Svensson (ed.) *Proceedings of the international workshop on mechatronical computer systems for perception and action,* Halmstad, 1993.

[Balkenius, C. & Pallbo, R.] Efficient implementation of neural networks in an object oriented environment. In *Neuro-Nimes '91,* Nimes, 733-736, EC2, 1991.

[Cypher, A. (ed.)] *Watch What I Do: Programming By Demonstration,* MIT Press, Cambridge, MA, 1993.

[Covo, Gersht, Kheradpir, Weihmayer] New Approaches to Resource Management in Integrated Service Backbone Long Haul Communication Networks, *Proceedings IEEE Network Operations and Management Symposium, NOMS '92,* Memphis, Tennessee, April 6-9, 1992.

[Davidsson] *Concept Acquisition by Autonomous Agents: Cognitive Modeling versus the Engineering Approach,* Lund University Cognitive Studies 12, 1992.

[diSessa, A. A.] Notes on the future of programming: Breaking the utility barrier. In D. A. Norman and S. W. Draper (ed.) User ***Centered System Design: New Perspectives on*** *Human-* ***Computer*** *Interaction,* Lawrence Eribaum Associates, Hillsdale, NJ, 1986.

[diSessa, A. A.] Local sciences: Viewing the design of human-computer systems as cognitive science. In J. M. Carroll (ed.) ***Designing Interaction.. Psychology at the Human-Computer Interface,*** Cambridge University Press, Cambridge, UK, 1991.

[Gasser, Huhns] ***Distributed Artificial Intelligence Vol II.*** Pitman, London, 1989.

(Gedenryd, H.] ***Active perception and the 'resonance metaphor'***, Lund University Cognitive Studies, no. 22, 1993.

[Gruber, Tenenbaum, Weber] Toward a Knowledge Medium for Collaborative Product Development, ***Proceedings of the Second International Conference on Artificial Intelligence in Design*** *1992*, Kluwer.

[McGuire, Koukka, Weber, Tenenbaum, Gruber, Olsen] ***SHADE: Technology for*** *Knowledge-* ***Bused Collaborative Engineering,*** Stanford Knowledge Systems Laboratory 1992.

[Gustavsson 93a] ***Mechanisms of Knowledge Sharing in COMIC. COMIC-SICS-3-1, 1993.***

[Gustavsson 93b] ***Mechanisms of Interactions and Interfaces at different levels. COMIC-SICS-3-2, 1993.***

[Gustavsson 93c] Societies of Computation - A Framework-. In Proceedings of AAAZ-93 ***Workshop on Al and Theories of Groups & Organizations: Conceptual & Empirical Research, 1993.***

[Gärdenfors, P. and Balkenius C.] Varför finns det inga riktiga robotar?, ***Framtider*** *1/93*, pp. 1 1-14, 1993.

[Jennings, Mamdanij Using Joint Responsibility to Coordinate Collaborative Problem Solving in Dynamic Environments. *Proc.* ***of*** *AAAI-92*, Morgan Kaufmann 1992.

[Jennings, Wittig] ARCHON Theory and Practice in Distributed Artificial Intelligence: ***Theory and Praxis*** (Eds Gasser and Avouris), Kluwer Academic Press, 1992.

[Kinney, Ljungberg, Rao, Tidhar, Werner and Sonnenberg] Planned Team Activity , in *Proc.* ***of the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World,*** MAAMAW'92, Rome.

[Kirch, Editor] Foundations of Artificial Intelligence. ***Artificial Intelligence Special*** *Issue,* ***Vo147 No 1-3,*** January 1991.

[Laurel] New Directions. Interface Agents: Metaphors with Character. ***The Art of Human Computer Interface Design.*** Addison-Wesley 1990.

[Malone] Computer support for organizations: Towards an organizational science. In Carroll (Ed.), ***Interfacing Thought: Cognitive Aspects of Human Computer Interactions.*** MIT Press, 1987.

[Marmolin, Sundblad] ***Sharing Knowledge in a distributed environment for collaboration, COMIC-SICS/KTH-4-1, 1993.***

[Myers, D. A.] Demonstrational interfaces: a step beyond direct manipulation, ***IEEE Computer, 61-73, 1992.***

[Neches, Fikes, Finin, Gruber, Patil, Senator and Swartout] Enabling Technology for Knowledge Sharing, *AI* ***Magazine,*** vol 12, no 3 Fall 1991.

[Pankoke-Babatz, Editor] ***Computer Based Group Communication. The AMIGO Activity Model,*** GMD & Ellis Horwood Ltd, 1989.

[Shoham] Agent oriented programming, an overwiev and summary of recent research, *Proc,* ***Fifth ACM Symposium on Principles of Database Systems,*** 1992.

[Simoudis, Editor] Workshop on: ***Cooperation Among Heterogeneous Intelligent Agents,*** AAAI 91.

[Tan, Weihmayer 1992] Integrating Agent-Oriented Programming and Planning for Cooperative Problem Solving, ***Proceedings AAAI Workshop on Cooperation among Heterogeneous*** *Intelligent* ***Systems,*** Tenth National Conference on Artificial Intelligence, San Jose, CA, July 12-17, 1992.

[Weihmeier, Brandau] Cooperative Distributed Problem Solving for Communication Network Management , ***Computer Communications, Vol. 13 No.9,*** pp. 547-557, November 1990.

[Weihmayer , Ghaznavi, Sheridan] A Distributed Architecture for Cooperative Management of Strategic Communication Networks, submitted to ***MILCOM '93,*** October 11-14, 1993, Boston, MA

[Weihmayer, Tan] Modeling Cooperative Agents for Customer Network Control using Planning and Agent-Oriented Programming , ***Proceedings GLOBECOM '92,*** Orlando, Florida, December 6-9, 1992.

[Widerhold] Mediators in the architecture of future information systems, ***IEEE Computer,*** March, 1991.

[Wielinga, Schreiber, Breuker] KADS: A modelling approach to knowledge Engineering. Knowledge Acquisition, 4(1), 1992, Special issue 'The KADS approach to knowledge engineering'.