Object-Based Concurrent Programming and Distributed Artificial Intelligence

Les Gasser
Computational Organization Design Lab
Institute of Safety and Systems Management
USC, Los Angeles, CA 90089-0021
(213) 740-4046; gasser@usc.edu

Jean-Pierre Briot
LITP, Université Paris VI, 4 Place Jussieu
75252 Paris CEDEX 05, France
+33 (1) 44 27 40 55; briot@litp.ibp.fr

in "Distributed Artificial Intelligence: Theory and Praxis," edited by N.M. Avouris and L. Gasser, pages 81–107, Kluwer, 1992.

Abstract

This paper presents some aspects of our studies of the conceptual and implementation relationships between Object-Based Concurrent Programming (OBCP) and Distributed Artificial Intelligence (DAI). The basic argument of this paper is that to implement DAI systems we should find complementary relations between the theory of social organization chosen for DAI problem-solving and the theory of modeling and implementation used for system construction. To clarify the nature of this relationship, we draw on a number of socially-motivated DAI experiments. The underlying social theories used in these experiments were derived from symbolic interactionism, while modeling and implementation theories had roots in OBCP. Starting from these experiments, we continue on to discuss the relevance of OBCP as a basis to construct DAI systems. We then discuss how to enhance OBCP to better provide multi-layered OBCP/DAI architectures. While this research is preliminary, we expect it to help clarifying the foundations and the design of future DAI platforms.

1 Introduction

There has been significant discussion over the past several years about connections between research issues and practical techniques in the Object-Based Concurrent Programming (OBCP) and the DAI communities [Bond and Gasser 88a, Ferber and Carle 92, Gasser et al. 87, Gasser and Huhns 89, Tokoro and Ishikawa 84, Yonezawa 90]. Distributed AI has been concerned with representation, reasoning, and coordination in collections of computational "agents." Object-Oriented Concurrent Programming is based on the notion of active and autonomous objects which are computing concurrently and which interact by exchange of messages. OBCP is directed towards problems of running large and complex programs efficiently on multiprocessors. The concepts of autonomous objects and message passing would seem intuitively and practically to provide a good foundation to describe and implement coordinating agents.

This paper presents some aspects of our ongoing studies of the conceptual and implementation relationships between Object-Based Concurrent Programming (OBCP) and Distributed Artificial Intelligence (DAI). It discusses the basic issues perspectives of DAI research relative to OBCP, and treats several practical issues of implementing DAI systems. Our aim is to discover ways to improve the relationship between OBCP and DAI, in two ways. First, we would like to make OBCP more relevant to DAI problems, by incorporating into OBCP specific representation and computing structures driven by DAI needs. Second, we would like to improve the structure and precision of DAI implementation approaches by resting them on the implementation structures of OBCP which are better-understood both formally and practically.

The structure of this paper is as follows. First we introduce a picture of DAI as a set of problems of representation and control. These problems drive the need in DAI for implementation structures and engineering methodologies. Next, we conceive the problem of DAI implementation as the interaction of two complementary theories: a *theory of social organization* which describes how agents coordinate and evolve, and a *theory of modeling and implementation* which describes how to represent and compute evolving structures of agents and their communities. Our original experiments with socially-motivated DAI and our approaches to integrating these two theories are described, and we discuss how better modeling and implementation theories now emerging in OBCP can assist the integration process.

1.1 DAI: Representation and Control Problems

Distributed AI has focused on the problems of representing and reasoning about collective, coordinated activity among groups of large-grain agents. Each agent, and the community of agents as a whole, embodies a processing architecture that reflects either a descriptive theory of human problem-solving and cognition, or a normative high-level approach to problem-solving. That is, DAI is concerned with representing and modeling processes of problem-solving at both individual and social levels, using computational methods. DAI researchers are especially interested in modeling problem-solving in which *control decisions* (decisions about what reasoning or problem-solving actions to take next) and *representation decisions* (decisions about how to form, stabilize, and utilize significant concepts) are taken dynamically by a community of agents itself, rather than by programmers. To do this, DAI research has addressed five basic issues. These five problems are inherent to the design and implementation of any system of coordinated problem solvers, and thus provide in part a set of foundational questions for DAI implementation platforms. The problems include[1]:

- How to formulate, describe, decompose, and allocate problems and synthesize results among a group of intelligent agents.

[1] See also [Gasser 92] in this volume and [Bond and Gasser 88a, Gasser and Hill 90] for more detail.

- How to enable agents to communicate and interact: what communication languages or protocols to use, and what and when to communicate.
- How to insure that agents act coherently in making decisions or taking action, accommodating the nonlocal effects of local decisions and avoiding harmful interactions.
- How to enable individual agents to represent and reason about the actions, plans, and knowledge of other agents in order to coordinate with them; how to reason about the state of their coordinated process (e.g., initiation and termination).
- How to recognize and reconcile disparate viewpoints and conflicting intentions among a collection of agents trying to coordinate their actions.

Solving these problems in a fundamental way is quite complicated. Some of the basic DAI issues (e.g., some problems of disparate representations or of task and resource allocation) can be designed away in a carefully engineered DAI system. Still, most theoretical and experimental approaches to them give only a partial account of multi-agent collaboration, because they have not grappled with several other even more basic issues—issues such as practical bases of shared knowledge, semantics, and assumptions, reasoning under inconsistency or incompatible representations, etc. [Gasser 91]. Virtually all current approaches to these problems depend upon common interagent semantics with at most one or two meta (i.e., reflective or contextual) levels, correspondence theories of representation and belief, global measures of coherence, and the individual agent as the unit of analysis and interaction. Most DAI experiments and theories depend upon closed-system assumptions such as common communication protocols, a shared global means of assessing coherent behavior, some ultimate commensurability of knowledge, or some boundary to a system. These current theories are inadequate for supporting the integration of heterogeneous systems with possibly-incompatible internal semantics (e.g. Hewitt's "microtheories" [Hewitt 91]). Moreover, approaches based on common interagent semantics are incomplete from a theoretical standpoint [Gasser 91]. All of these issues have impacts on the structure and capabilities of implementation platforms and development methodologies for DAI.

1.2 DAI: Implementation Problems

Along with problem-solving frameworks for DAI, there is a set of practical issues that must be addressed to build and experiment with DAI systems and theories. We can summarize them as two more key issues:

- What are the appropriate implementation tools, languages, shells, and platforms for DAI research, experimentation, and development?
- What are appropriate methodologies, design techniques, and engineering practices for building DAI systems?

In attempts to integrate problem-solving and implementation approaches, numerous technology platforms for DAI have been built. These include testbeds such as the DVMT [Durfee et al. 87b] and MACE[Gasser et al. 87a]; integrative systems such as ABE [Erman et al. 88], and blackboard or distributed blackboard systems such as GBB [Corkill et al. 86], BB1 [Hayes-Roth 85], CAGE/POLIGON [Nii et al. 89], etc. In addition, OBCP languages and programming environments have been suggested as a promising implementation technology for DAI systems, and a number of OBCP languages have been expressly designed, in whole or in part, to accommodate the needs of knowledge-based systems and automated reasoning (e.g., ACTORS [Agha 86], ABCL/1 [Yonezawa 90], MERING-IV [Ferber and Briot 88, Ferber and Carle 92], ORIENT84/K [Tokoro and Ishikawa 84]). Still, few implementation platforms have been characterized in relation the social theory used in problem-solving. Especially useful would be general (programming-language and system-level) relationships that link social DAI theories with appropriate implementation structures. On the whole, from a language design and system operation perspective, none of these implementation approaches has yet provided adequate means for expressing solutions to the key problems faced by DAI theorists and system designers. In particular, none of the OBCP systems provides an adequate platform for the principled construction of DAI systems that takes into account the several basic issues of DAI stated above. To address these core problem-level issues, implementation platforms would best be driven by well-developed theories of agent interaction, organization, and aggregation. Steps have been taken in this direction, including:

- The Contract Net system which linked a market-like task sharing mechanism to a communication protocol language [Davis and Smith 83].
- Agent-oriented programming, with its implementation of structures of commitment and specialized interaction protocols based on speech acts [Shoham 90].
- Integrated goal-directed and data-directed control in the DVMT and BB1 systems [Hayes-Roth 85, Lesser 83] that link local and global problem-solving coherence goals to metalevel reasoning structures in an architecture.
- Acquaintance modeling for system integration and organization in the MACE system [Gasser et al. 87a] (see below).

However, the implementation support for addressing these issues has often been *ad-hoc*; there are few clear programming-language and agent-structure features that directly support theory-driven approaches to interaction or aggregation. While some researchers have developed interesting formal or substantive theories (e.g., [Cohen and Levesque 90, Durfee and Lesser 91, Rosenschein 82]) they either haven't been widely implemented, or representations for them have been created for the purposes of a particular domain-oriented system only—they are not programming languages. We believe the time has come to begin to think carefully about the general implementation structures that support DAI application and research at the programming language and system level, and it appears that we are not alone in this belief [Ferber and Carle 92, O'Hare and Wooldridge 92, Shoham 90, Shoham 91, Wittig 92].

1.3 Theories of Social Organization and of Modeling/Implementation

The principal motivation for the approach we have taken is less the need to model or solve particular sets of domain problems, and more our view that DAI tools and techniques should have as their basis two underlying theories. The first of these is an underlying *theory of social organization* which offers a set of socially-oriented conceptual models of the bases of aggregation, interaction, communication, coordination, etc. The second of these is a *theory of modeling and implementation* that can be used to bridge the gap between the socially-oriented conceptual model and a computational realization. In this paper we are concerned with the ways in which DAI theories of social organization that attempt to address the representation and control issues mentioned above (Section 1.1) drive requirements for DAI theories of modeling and implementation.

For us, the socially-oriented conceptual model derives from symbolic-interactionist sociology (e.g., [Mead 34, Blumer 69, Charon 79] and modern science studies (e.g. [Callon 80, Latour 87]). The basic ideas of symbolic interactionism are that:

- The primary units of analysis are *interactions*, not individuals.
- *Symbols* are both the means and content of important interactions. The significance and meaning of symbols are in turn established through interaction.
- *Individuals* are dynamic and evolving agents with many components, not stable, structured, unitary personalities.
- The many context-dependent, continuously-evolving *selves* of an individual are shaped via dynamic selection and interpretation (as symbols) of stimuli *in interaction*.
- Actions are guided (but not determined) by *perspectives*, which are learned through *communication*.
- *Society* comprises individuals in *processes of patterned interactions*, embedded in, and always (re-)creating, perspectives.

The relevant themes of science studies include the social construction of knowledge and of "facts," the dynamic, locally-adapted nature of knowledge, the idea of knowledge as patterns of action, rather than as physical law, etc.

A theory of modeling and implementation describes how to address each of these problems with specific description and programming mechanisms. OBCP provides us with the foundations for a theory of modeling and implementation.

1.4 OBCP as a Modeling and Implementation Foundation for DAI

A key issue in that differentiates DAI theorizing and the modeling and implementation theories in OBCP research worlds is great variance in the explicitness and completeness of the focal phenomena of interest to researchers. For example, in the OBCP arena, which is closer to programming research, there are several widely-shared and clearly delineated focal phenomena with relatively clear historical roots - the communicating entities, messages, etc. mentioned above. But these have much less substantive context or content than the focal phenomena in DAI. The theories of problem solving and social organization that they embody place the largest part of the burden of defining interaction, control, synchronization, etc., on the designer who modelizes the problem, and on the programmer who implements its solution. The focal phenomena in the DAI arena (e.g., modelizations of coordination and coherence activities—the theories of social organization) are much less clearly defined or shared among researchers, but those that exist have much greater substantive content than their counterparts in OBCP. This creates a difficult-to-bridge gap between the implementation and modelization *requirements* of DAI systems, and the implementation and modelization *facilities and techniques* provided by OBCP. The basic questions for integrating the two theories are 1) how to squeeze the DAI problem content into the OBCP modelization framework, and 2) how to accommodate the highly-structured OBCP representational machinery into relatively ad-hoc DAI implementation structures.

1.5 OBCP Concepts

Object-Based Concurrent Programming has emerged as one of the leading high-level programming paradigm for building distributed concurrent systems. The success and the further promises of OBCP come from its natural ability to describe computation as concurrent and communicative entities. The general nature of the concepts of *entity* and *message-based communication* lays ground to expand the versatility of OBCP. For example OBCP objects can model very fine entities such as neurons [Weitzenfeld and Arbib 90], or large components of expertise such as found in large-grained multi-agent systems (e.g., MACE [Gasser et al. 87a]).

The uniformity of communication mechanism plus the concept of encapsulation enable combination and cooperation of various levels and grains of agents. This helps constructing multi-grain intelligent architectures such as the architecture for brain simulation described in [Weitzenfeld and Arbib 90]. Consequently OBCP has already been recognized as a potentially natural and useful basis for building DAI systems.

Objects and message passing provide some good structural and computational foundations for describing DAI agents and their communication protocols. Objects are indeed agents, but they don't match the whole spectrum of DAI requirements. Current approaches to OBCP provide some support for DAI without really providing the needed bridge between DAI theory and implementation. To make this more clear, we will quickly summarize the main OBCP concepts.

OBCP consists in describing *communicating active entities*:

- An *entity*, called *object*, represents some expertise, which could be further described in terms of knowledge about other objects, called *acquaintances*, and skills, called *methods*, which are procedures associated to the object.

- *Communication* is the means of interaction between objects. This consists in *message passing*, that is an object may send and receive informations, called *messages*, to other objects. The basic protocol is for an object to send a request for some skill to one of its acquaintances. (Note that new acquaintances may be communicated through message passing, allowing reconfiguration of its knowledge). This will trigger the activation of one of the methods of the receiver, with the arguments conveyed in the message. Communication between two objects *may* include their *synchronization*, for instance if some results are needed to resume the computation.

- Each object is *active* in that it owns its computation power. Consequently objects may compute independently which brings *concurrency* between the objects.

These concepts form the core of OBCP paradigm. In order to structure and factorize the knowledge between objects, abstraction (*classes*) and sharing-mechanisms (such as *inheritance* between classes) may be added[2]. Various OBCP programming languages have been and are still being proposed. The differences between them come from different design decisions along main tradeoff issues and the different classes of applications and hardware architectures targeted.

We may include (sequential) OOP systems into the OBCP paradigm as a restricted case of OBCP. (OBCP objects could be considered in turn as "degenerate" agents). We believe that hybrid multi-granularity OBCP architectures, which mix OOP serial computation with OBCP concurrent computation, are the best appropriate for building DAI platforms, where the main concurrency is often restricted to concurrency between agents. The uniformity of a concurrent foundation such as actors is very appealing but makes it more difficult to implement and program on current hardware and software technology. In that respect we feel closer to hybrid OBCP platforms such as ABCL/1 [Yonezawa 90].

[2]These concepts are widely accepted and considered as key-concepts in (sequential) Object-Oriented Programming (OOP) languages community. But their incorporation into OBCP is currently a focus of debate in the OBCP research community, because concurrency and moreover distributed implementation make standard OOP sharing techniques partially inaccurate. The problems include including dealing with inheritance in a distributed environment, description versus execution consistency, dynamic distribution and load balancing, and composition of multiple concurrent objects [Briot and Yonezawa 87, Gasser et al. 87a]. Consequently we don't list them as key concepts here. Moreover, this is the reason why the term "Object-Based Concurrent Programming," which makes no reference to class and inheritance concepts, is often used in place of "Object-Oriented Concurrent Programming".

1.6 OBCP Investigations Have DAI Relevance

A number of OBCP investigations have included special focus on the articulation between OBCP and DAI technologies. These are our primary reference points.

The ACTOR computation model [Agha 86] is the first and one of the main OBCP formalisms. ORGs [Hewitt and Inman 91] are currently being proposed as an organizational layer to be implemented using actors. UBIK [deJong 90] is another model of organizations implemented on top of an Actor system (the PRACT/ACORE system [Manning 87]). [deJong 90b] discusses its relations with ACTORS.

ORIENT84/K [Tokoro and Ishikawa 84] provides an integration between OBCP aspects and logic-based aspects of a programming language. Knowledge Objects (KOs) are the result of combining a behavior part (variables and methods) and a knowledge part (facts and rules).

The MACE system [Gasser et al. 87a, Gasser et al. 87b] provides a testbed for DAI architectures which was one of the first systems to truly run in a distributed memory and processing implementation (on a multicomputer network comprising an iPSC Hypercube, a Sequent Symmetry, and Sun workstations). Parts of the MACE architecture are based on OBCP and MACE includes symbolic representations of other agents as well as aggregation concepts (*organizations*) [Gasser 89b, Gasser et al. 87a].

[Maruichi 90] discusses what concepts have to be added to OBCP in order to model cooperative autonomous agents, namely facilities for autonomy and organization. Note that the model of organization Maruichi proposes has more to do with a shared communication medium (to broadcast messages to a group) than a general model of organization.

Situated conversation theory is proposed in [Numaoka and Tokoro 90] as a model of interaction between agents, based on OBCP objects. Conversational protocols are based on speech act theory. Situated conversation theory is suggested as a way to help avoid deadlocks and for agents to mutually construct their own coordination protocol.

COMMAS [Bouron 91] is a communication model for DAI. It is based on OBCP augmented with principles based on speech-act theory and cooperative principles (general behaviors observed in natural language dialogues). Communication is exchanged between agents by asynchronous messages. The messages describe beliefs and goals including commitments and generate new beliefs and goals.

Agent-oriented programming [Shoham 90, Shoham 91] is another proposal for extending objects towards agents by adding concepts of mental state, intentional communication (also based on speech act theory), and rule-governed behavior, and thus by *restricting* the behavior of objects in accordance with a theory of interaction.

The MERING-IV [Ferber and Briot 88] language has been specially designed for DAI purposes. Higher-level concepts such as classes and slots are built up from basic actors. MERING-IV uses a multi-layered architecture [Ferber 88], with reflection being used as a glue and bridge between layers [Ferber and Carle 92].

The MAGES [Bouron et al. 90] multi-agent system has been built on an OBCP environment: an extension of Smalltalk called ACTALK [Briot 89, Lescaudron et al. 91]. MAGES extends ACTALK into a platform for simulating various DAI architectures. The architecture of MAGES is incremental, based on OBCP, and built-up step by step through higher-level classes. These classes extend the notion of instance variables into slots, base the behavior of agents on their inference cycle, and provide environments to model a shared perceptive space (usually topological) between several agents.

REACTALK, also based on ACTALK, provides a reflexive architecture of an OBCP system [Giroux and Senteni 91]. The behavior of an REACTALK actor is decomposed into an organization of other actors (that is as opposed to most of reflexive OBCP models, a meta-object is not a single object, but distributed among various perspectives/objects). This reflexive architecture is used for behavior simulation with a special focus on adaptation through introspection. High-level agents are modeled along Freudian description of self as an organization of three internal actors (duty, actual behavior, and will). These actors are themselves modeled as organizations of actors. In the REACTALK model, there is no gap between actor and agent, only recursive organizational levels.

2 Case Studies of Socially-Motivated DAI

By the early 1980s existing DAI research had shown that the concept of coordinated problem solving was feasible and interesting. During the same period, a number of critical studies had emerged that began to seriously investigate the micro- and mid-level social aspects of computing in organizations. Concurrently, technologies for computer-supported cooperative work began to appear. It was clear to some groups of researchers that an important set of related issues was beginning to appear. These concerned 1) the prescriptive and theoretical aspects of intelligent problem solving in aggregates (from DAI and concurrent computing) and 2) the descriptive, analytical and practical aspects of systems that involved people and machines working together with flexible, dynamic divisions of labor (from sociology, human-technology studies and computer-supported cooperative work) [Bendifallah et al. 88]. For several researchers who had worked across these disciplines, intellectual tools derived from symbolic interactionism proved useful in conceptualizing the problems of dynamic divisions of labor and of knowledge-level description of joint activity. In particular, at USC, over a number of years, we have pursued our study of DAI and of the tools and techniques for constructing DAI systems from a direction that stresses the relationship between DAI theory (especially theories of aggregation and organization) and DAI implementation. We began to think about how to build an implementation structure for DAI systems that would allow us to capture some notions of organization and interaction motivated by ideas from symbolic interactionism and science studies.

The collection of DAI case studies that we have pursued gives a framework of experience which we draw upon to think about the structure of modeling and implementation theories for DAI systems. ¿From the beginning, these implementations have been partly based on OBCP concepts. We discuss these experiences to help highlight the

Figure 1: An Approach to DAI

Figure 2: MACE Agents

relationships between OBCP-based modeling/implementation theories in general, and socially-oriented DAI theories.

2.1 Socially-Motivated DAI and the MACE testbed

Our socially-motivated approach to DAI is illustrated schematically in Figure 1. The key ideas on which we began to base DAI systems structure and implementation are the following. (Brief pointers to the particular projects and writings in which we explored these ideas in the context of DAI are noted after each item below.)

- The notion of *modeling other agents* as the key integrating mechanism for organizational and coordination processes. This idea was drawn from Mead's theory of the unification of self and society through the processes of "taking the role of the other" and reflexive self-identity [Mead 34]. It was reflected in the MACE system as the acquaintance structure that was a part of all agents. (See Figure 2.) (See [Gasser et al. 87a] and the OSD approach and experiments [Ishida et al. 92]).

- The idea that DAI systems should be multi-agent, possibly human-machine aggregates at all levels, which led to the approach of using the help of a community of *system agents* for constructing and interacting with a DAI system. (MACE [Gasser et al. 87a]; see Figure 3).

Figure 3: System Agents Interact with User Agents

Figure 4: Contract Net Modeled as a MACE Organization

- The idea that long-term problem-solving proceeded through a series of "frozen accidents" that became embedded, reified, and aggregated into stable structures of action and interaction (Studies of organizational integration (OI) [Gasser 86] and organizational knowledge (ORGKNOW) [Gasser et al. 89].)

- The idea of organizations as stable, overlapping, and nested patterns of action and knowledge—as patterns of *settled and unsettled questions*—rather than organizations as fixed structures of responsibility, communication, or control. For example, the MACE notion of organization was captured in the knowledge agents had about each other. The boundary of an organization was simply the boundary of knowledge about how and when to include other agents in particular problem-solving processes. The MACE representation of a single node of a contract net [Davis and Smith 83], for example was a four-agent aggregate in which each agent had highly restricted knowledge of with whom to interact (see Figure 4). Similar notions were used in the OSD experiments. (MACE, ORGKNOW and OSD [Gasser et al. 89, Ishida et al. 92].)

- The observation that social activity is inherently a multi-perspective process, and that disparities can occur at any level of description or context from many different points of view simultaneously. Hence, processes in practical DAI systems will be subject to observational and organizational dynamism and incongruity (OI, MACE, ORGKNOW, and OSD [Gasser 86, Gasser et al. 89, Gasser 91, Gasser 92].)

Figure 5: MACE System Structure

- The ideas that the loci of action and knowledge in social systems are dynamically aggregated units, that aggregation must be modeled across numerous levels and from numerous perspectives simultaneously, and that in effect, agents can be construed as emerging from interactions, rather than vice-versa (OSD, [Gasser 92, Ishida et al. 92].)

- The importance of both *prescriptive* (i.e., targeted for explicit organization design goals) and *exploratory* (or emergent) approaches to generating organizational form and structure (the Organizational Knowledge, OSD, and HITOP-A experiments [Gasser et al. 89, Ishida et al. 92, Majchrzak and Gasser 92]).

These basic ideas were the product of years of effort in many different streams of thought; they were drawn from and reflected in the work of such other sources as [Agha 86, Becker 86, Gerson 76, Hewitt 77, Mead 34, Latour 87, Strauss 78, Weick 79], and many others.

The MACE system was our first attempt to offer a general bridge between DAI theory of social organization and DAI theory of modeling and implementation. The implementation structure of the MACE system is shown in Figure 5. MACE was a partial OBCP system with the goal of both introducing true parallelism and allowing for controlled simulation of concurrent behavior. The system structure shown in Figure 5 incorporated multi-level concurrency (multiple processors, multiple operating system processes, multiple LISP processes, multiple MACE kernels, and multiple MACE agents). Synchronization was possible at any layer in the MACE design, the idea being that different concurrency layers would allow for different process granularity and carrying experimental synchronization behaviors, while maintaining some level of repeatability of experiments.

As an early DAI-OBCP bridge, the MACE experiments were valuable in defining some of the primary problems to be faced in later use of OBCP systems for DAI implementation. There were significant limitations to these experiments, however. In more recent work, researchers in OBCP and DAI have begun to focus on a set of common issues that reflect the theoretical and practical needs of each.

2.2 Limitations of Modeling and Implementation Approach

While provocative, these experiments revealed several limitations. First, the conceptions of objects, object interaction, and attribute sharing were too simple to be useful. We faced many of the problems of distributed objects (including the distributed inheritance, description/execution consistency and dynamic distribution and composition issues noted above). Second, our attempts at using acquaintance modeling to capture Mead's notion of role-taking were not reflexive or dynamic enough, and were not really well-integrated into the basic structure of objects. (Later we realized this as a problem directly linked to the capacity for reflection, now common approach in OBCP.) Third, the compositional mechanisms of MACE, while properly directed at knowledge-based and communication-based composition, (as versus structural composition), were not properly integrated across levels; there was a primitive and atomic level of MACE agent, which was actually quite complex. Finally, MACE was not well integrated as a development tool; it was quite difficult to actually build and maintain a system using MACE.

Other experiments such as ORGKNOW and OSD were attempts at conceptual modeling and performance enhancement, rather than at briding the social theory—modeling theory gap; they relied on ad-hoc implementation mechanisms as well.

3 DAI Theory and Issues Drive Implementation Needs

Design requirements of implementation platforms for DAI systems are driven by the needs for 1) distribution and interaction among problem-solvers, 2) an incremental development cycle, 3) advanced programming paradigms which help manage program and application complexity (which is very high in DAI applications), and 4) tools for analysis and visualization of concurrent program behavior. These requirements mean that implementation platforms typically require one or more of the following capabilities, along with standard features for advanced AI development environments. OBCP automatically provides support for a number of these. Some others require deeper thought about more basic conceptual issues, such as the nature of object identity and referencing, object composition and decomposition, and mutual object reflection.

Encapsulation and Local Control: Both to manage complexity in applications, and to provide a natural fit to the problems and domain requirements of DAI systems, object-based paradigms are common. Encapsulation of data and procedures (which Hewitt calls "arms-length relationships," [Hewitt 86] or "self-reliance" [Hewitt 91]) provides a natural basis for distribution and flexible task decomposition. It also creates particular problems of description and execution in distributed environments (e.g., inheritance) [Gasser et al. 87a, Gasser et al. 87b, Agha et al. 89]. OBCP automatically provides encapsulation of entities, and recent OBCP proposals have begun to address more flexible controlled sharing of knowledge [Ferber and Carle 92]. At a deeper level, encapsulation is directly related to the conceptual issues of of object boundaries, identity, and referencing, which become significant problems in the context of aggregated objects.

Message-Based Communication: Truly distributed DAI systems require message-based communications for interaction. This raises several problems for implementation platforms. Implementations which require consistency and completeness of data and knowledge across problem solvers—the so-called "Completely Accurate, Nearly Autonomous" (CA/NA) approach [Lesser 81] must cope with message delays and asynchrony, and therefore must incorporate mechanisms, such as synchronization locks, transaction managers, etc. Even shared-memory platforms such as blackboard systems with concurrent processing of knowledge sources require database consistency mechanisms under the CA/NA problem-solving assumption [Fennell 77]. Message communication schemes require underlying mail-delivery systems, which provide addressing tables, message forwarding, and run-time address resolution. In addition, they might support high-level standardized agent interaction protocols which implement particular communications or synchronization architectures in the form of fixed sets of messages [Gasser 89b, Shoham 91].

OBCP languages generally provide message-based communication, and are beginning to integrate higher-level protocol constructs and the ability to flexibly adapt protocols via reflection [Ferber and Carle 92, Watanabe and Yonezawa 88].

At a deeper level, message-based communication is related to encapsulation and to identity and referencing of objects, since messages must be directed to a receiver, and must be acted upon by an interpreter, which may itself be an aggregate (cf. [Giroux and Senteni 91]).

Heterogeneous Multi-Grain Objects: Agents or objects may exist at different levels and types of granularity in a DAI system, so an implementation platform must address the issues of mapping objects to processor-memory pairs for appropriate resource utilization and load balancing. Most existing development platforms either make no assumptions about objects of varying grain size, or assume that most objects will have fairly uniform and stable grain size, so that resource allocation is straightforward. For reasons of flexible system design and reconfiguration, run-time expansion/contraction of objects, and flexible object composition implementation platforms should be designed to explicitly consider the grain size of objects.

Beyond concerns of load-balancing, parallelism, and performance, agents in DAI systems must be conceptually modeled as collections of objects at several simultaneous levels of granularity, to reflect the natural processes of organization [Gasser 92, deJong 90].

Flexible implementation support for DAI systems must provide ways of integrating heterogeneous problem-solvers of different granularity. Along with providing a collection of knowledge representation and reasoning tools, a DAI system should provide a mechanism for integrating results from different problem-solving processes, and of configuring the interaction among problem-solving nodes. This is accomplished most easily with standard interface protocols, but the protocols must be designed to reflect the importance of the 5 basic DAI problems introduced above.

At a deeper level, heterogeneity and multi-granularity are also problems of referencing, activity, identity, and composition, since heterogeneity may arise from alternative compositions or aggregations of objects.

Language Support for Flexible Organization and Interaction Structures: To maximally exploit concurrency, as well as to provide for flexible reconfiguration of the interactions among objects in a DAI system, higher-level modules may be composed of lower-level objects which are themselves capable of local coordination. The approach to design may be either *reductionist/decompositional*—explaining and implementing high-level behaviors in terms of lower-level concurrency by constraining the behaviors of lower-level components—or *constructionist/compositional*—aggregating lower-level concurrent modules into emergent higher-level structures with emergent behaviors, which may not be directly derivable from the descriptions of the lower-level modules.

At a deeper level, flexible organization and interaction are also problems of referencing, activity, identity, and composition, since flexible organizations accommodate dynamic aggregation and thus encounter dynamic referencing, identity, and boundary issues.

Agent Modeling: Adaptive coordination, and especially coordination with limited communication, requires the ability to model the behavior and knowledge of other agents. Implementation platforms designed for DAI applications should support the ability to build models of the beliefs, plans, goals, etc. of other agents, as well as models of the state of the problem-solving process itself.

At a deeper level, agent modeling also involves problems of referencing, activity, identity, and composition, since aggregate agents may refer to models of other agents which are themselves dynamic aggregates. This of course raises issues of dynamic referencing, identity, and boundaries.

Reusable Shells: Practical DAI systems engineering will require the ability to reuse abstract descriptions of system components, and to reuse knowledgebases and coordination structures [Wittig 92]. OBCP languages already provide useful structures for concept and code reuse, and to the extent these can be elaborated into high-level integrative shells, the construction of DAI systems will be enhanced.

Testbed and Measurement Tools: Practical construction of DAI systems also demands high-level simulation, measurement, and control environments, and development tools to build and manage large-scale DAI systems. Development of DAI systems is time and labor-intensive process, made more difficult by indeterminacy and concurrency in the actions of agents. While some theoretical models are being advanced which minimize the effects of indeterminacy or allow programmers to reason explicitly about the degrees of uncertainty or knowledge limitations which can be tolerated [Agha 86, Mazer 91], these are presently in early stages of development, and have not been embedded in useful and widely available programming languages or frameworks. The need for controlled experimentation both for precision during development, and for performance tuning, argue for testbed-like implementation platforms, from which performance measurements and analyses can be drawn [Montgomery 90, Gasser et al. 87a, Lesser 83, Nii et al. 89].

The difficult problems of shell and testbed construction involve how to incorporate support for dynamic analysis of object identity, composition and aggregation, boundaries, concurrent object activity, etc.

4 OBCP and DAI Share Joint Issues

The implementation needs illustrated above are unified by the necessity to grapple with a core set of issues. Since OBCP provides many ways of thinking about a number of these issues, we find that OBCP and DAI do have a lot of background in common. As examples of convergence between OBCP and DAI we cite some joint issues which emerge at conceptual, design, and implementation levels, and which appear as components of the implementation needs cited above:

Composition/Decomposition Transformations: For purposes of load balancing, adaptive reorganization to fit new problem circumstances, (including adaptive clustering of processes, changes in data sources, etc.), and flexible control, it may be desirable to change the grain size of agents, and the control and knowledge relationships among agents, during the course of problem solving. This is only possible if the agents themselves are decomposable or composable. Thus, description languages must be available which allow for composition of agent-groups from individual agents, and treating agent groups as single agents at higher levels. Agha and Hewitt call their higher-level compositions of actors *actor configurations*. Moreover, it is important to be able to view the activities of individual problem solvers in a group at several levels of analysis. Just as social theorists conceptualize social action in numerous levels, such as individual, small group, organization, society, and so on, it may be useful to be able to analyze an intelligent distributed problem-solving process at varying levels in vitro (cf. [Parsons51, Gasser and Huhns 89]). This can only be done using tools and analytic techniques which treat and conceive the problem-solving agents at differing levels of granularity, using observational mechanisms of decomposition and aggregation. Much more research needs to be done to address these compositional, decompositional, and representational mechanisms, but several existing approaches can provide suggestive frameworks for implementation.

Compositional and decompositional transformations have been designed for OBCP systems and for agents in DAI systems. For example, [Shibayama 88] describes and studies the correctness of merging/splitting concurrent objects in ABCL/1. Ishida and Gasser have studied how to dynamically apply such transformations on agents in order to reorganize the macroarchitecture of a DAI system for better performance, and as a way of making conceptual links to a socially-motivated theory of multi-agent interaction [Ishida et al. 92].

The DAI testbed MACE [Gasser et al. 87a] included representation mechanisms for aggregates of agents called *organizations*, and suggestions have been advanced for decomposition composition mechanisms for pure actor systems [Agha 86, Hewitt and Inman 91].

Surrogates and Remote Referencing: A representation for higher-level aggregates in actor-based open information systems called ORG, introduced in [Hewitt and Inman 91], includes facilities to maintain boundaries between ORGs and to handle communications. Actors called *insiders* and *outsiders* are used as representatives when referencing actors in other ORGs. This mechanism of remote reference and reference interpretation during communication is the same mechanism used in distributed (OBCP) implementations for addressing and communicating with remote objects. (see for instance [Bennett 87]). One correspondence with the symbolic interactionist-based DAI social theory is that the general mechanism of arbitrary symbolization of actions and objects has been proposed as a basis for social aggregation and the development of "mind" [Charon 79]. Moreover, the formation of dynamic aggregates necessitates new views of the nature of *individuality*, since units in aggregates may be entering and leaving, and any unit may be participating in multiple aggregates simultaneously (with "multiple selves") [Gasser 92].

Meta-Level Architectures and Reflection: Meta-level (also called reflexive or reflective) architectures—those unified architectures in which some part of the architecture objectifies, reasons about, and influences some other part—are now advocated in both OBCP systems ([Maes 88], [Watanabe and Yonezawa 88], and [Ferber 89]) and in some aspects of DAI systems (e.g., [Hayes-Roth 85, Durfee et al. 87b, Gasser and Huhns 89]). Reflection is proposed as a unifying glue between OBCP and DAI levels in [Ferber and Carle 92, Gasser 91]. Ferber has suggested a combined mechanism of reflection and delegation for dynamic updating of surrogate references in actor-based DAI systems, as a technique to support dynamic reorganization. One approach uses reflection to decompose the behavior of an aggregate actor into organizations of actors and to use this composition/decomposition mechanism to buildup higher level adaptive agents [Giroux and Senteni 91]. The issue of modeling other agents has been proposed both as a foundation of DAI (and social) theory, and as a key component of working DAI systems [Bond and Gasser 88a, Gasser et al. 87a].

To understand how OBCP can provide structures that underly the DAI implementation needs, as well as these several issues of common concern, we divide our analysis of OBCP into several categories, including OBCP's treatment of communication and interaction, activity, state, organization, and identity.

5 Extending OBCP For Better Integration with DAI

5.1 OBCP is Insufficient to Match DAI Requirements

As can be seen in the previous section, OBCP can provides several bases to model interacting agents. However, because of a design and implementation focus at a general-purpose programming language level, OBCP deals too simplistically with the following points:

Communication/Interaction: In OBCP all communication is uniformly done through message passing. Any kind of message may be sent. This provides uniformity but does not give any explicit information about the *intention* of the communication, nor does it support a theory of communication and interaction linked to a theory of individual or joint problem-solving activities. In the general OBCP formulation, it is the programmer who determines the semantics of messages and the shape of protocols, not the community of agents.

Several OBCP languages provide distinction between some types of message passing. (See for instance ABCL/1's three types of message passing: asynchronous, synchronous and eager [Yonezawa 90].) But this classification focuses on data-flow and synchronization rather than on the purpose of communication or its place in a joint activity.

Activity: Activity of objects is performed in an inflexible and procedural way. Activity is usually based on implicit message acceptance. When accepting a message, the method to be performed is selected through the name of the message. In some systems pattern-matching may be a little less simplistic, in that patterns may be more general (e.g., Plasma [Hewitt 77]), the class of the arguments of the message may be relevant in the selection process (e.g., CLOS [Bobrow et al. 90]), and several methods may be selected and then combined (e.g., CLOS). However the description of the behavior of an object still remains very procedural, not declarative.

State: The state of an object is in general conceptually decomposed[3] into internal data and references to acquaintances, i.e., other objects it may communicate with. Some systems provide typing information, but there is usually no explicit description of the acquaintances. One needs to look into the description of the methods of the object to understand the *meaning* of the acquaintances.

Organization: Composition and inheritance are the basic mechanisms that OBCP provides to structure a collection of objects. But they are usually the only structuring mechanisms. When the number of objects becomes large, it becomes necessary to organize them in larger entities because individual referencing of objects

[3] This distinction does not exist at the language level in uniform systems (where everything is an object). However it proves to be useful to help delimiting the boundaries of conceptual objects.

tends to be less manageable. Such a grouping mechanism has been proposed for some specific needs such as debugging [Honda and Yonezawa 88]. *Configurations* [Agha 86], and furthermore *concurrent aggregates* [Chien and Dally 90], have also been proposed as constructs to abstract groups of objects. However these simple organization mechanisms do not take into account the various articulations between objects inside a same group and between groups (e.g., master-slave) to structure coordination between them to achieve a common task. Most conceptions of groups in OBCP employ a representative or object that serves as a surrogate for the group. DAI needs distributed representations of groups, so that groups may be viewed from different perspectives simultaneously.

5.2 Extending OBCP

Our proposals for ways in which to extend OBCP to provide a better support to address the DAI requirements are:

Identity: One key problem is the problem of object identity, which can be further analyzed as three issues: the need for distributed object representations, the need for dynamic object representations, and the need for situated object representations. Fixed boundaries for objects are too inflexible, and do not reflect either the theoretical positions emerging in the DAI world or the reality of multi-level aggregations of action and knowledge (e.g. multi-level groups or organizations). From these viewpoints, agents are dynamically defined by reference to their changing position in a community, and by reference to their position in a *frame of reference*—the "same" computational units (which may be individuals or groups, depending on levels of analysis) may participate in different "agents" simultaneously. Finally, when agents are aggregated into dynamic groups, the issue of *continuity of identity* emerges—what maintains the stable identity of an "agent" when it is composed of ever-changing object definitions and patterns of interaction? This can be an important issue; Hogg and Huberman have shown the potential for chaotic behavior among agents with particular decision making procedures, and has illustrated how chaos can be ameliorated by making agents better predictors of each others' behavior [Hogg and Huberman 90]. However, such prediction depends upon the continuous identity of agents.

Communication/Interaction: [Shoham 90], [Shoham 91], [Bouron 91], and [Numaoka and Tokoro 90] propose to constrain communications along the classification of speech act theory [Searle 69] by distinguishing the type of intention of the communication (inform, request, promise, etc.). As noted earlier, this relies on the assumption of common-shared communication semantics. A more general model could be based on the concept of *meta-communications* as a basis for setting up comprehensive communication between agents. For the future, we believe that a *social theory of interaction* is necessary. In such a social theory, the semantics of communications would be an emergent property of cycles of interpretation, symbolization, and response among many agents in a system, rather than in an *a priori* shared semantics. The notion of "object" in such a system would thus be usefully flexible (cf. **Identity** above), but stabilized by playing a role in a community.

Activity: Non-procedural models have been proposed in AI, such as rule-based systems, and data-driven actions (daemons and knowledge sources) to encompass the inflexibility and explicitness of procedural description of the behavior of an agent. ORIENT84/K [Tokoro and Ishikawa 84] provides a model of combination (and not unification) of structure/behavior and knowledge/inference. Instance variables are distinct from the knowledge base (facts and rules). Methods may trigger the knowledge engine and vice-versa. The knowledge base also supports data-driven daemons.

Another alternative is through the concept of cycle, by merging the cycle of knowledge inference with(in) the cycle of message acceptance, as in agent-oriented programming [Shoham 90, Shoham 91] or MAGES [Bouron et al. 90].

There is also the issue of at what *grain* to implement activity—*within an object* or *among objects*, how to make the grain flexible, and how to link the grain to the identity of objects. Grain and identity can be related through the notion of *flexible* bindings of action and knowledge. For example, Ishida and Gasser have constructed a dynamic problem-solving system based on production rules, in which the community of agents emerges from a fabric of knowledge dynamically, in response to changes in problem-solving (quality or time) demand [Ishida et al. 92]. The definitions of objects or agents in this system may be envisioned as being represented in the dynamic knowledge-knowledge and action-knowledge dependency relationships in the system.

More explicit control over the acceptance of messages may be gained to enhance the autonomy of the object [Maruichi 90], by extending the computation model (e.g. the waiting mode of ABCL/1 [Yonezawa 90]), or through reflection (e.g., by reifying the messages queue [Tomlinson and Singh 89]). Input may also be decoupled by introducing multiple message queues (e.g., in HYBRID [Nierstrasz 87]) drawing analogy with multiple-ports models of Concurrent Prolog and CCS.

¿From one point of view, message passing in OBCP could be expanded towards AI theories of *belief*, *inference*, and *mental state* (e.g, [Shoham 90, Shoham 91]). That is, from a DAI perspective, the interpretation of a message would be contingent on the belief state of the agent/object and on that agent's place in the joint problem-solving process. This of course complicates the problem of consistency in semantics, because the locus of meaning is found in the group and the circumstances, rather than in the specification of the object at programming time.

A second approach would be to begin to redefine the nature of knowledge and belief to rest in the *actions of the community* rather than in the mental states of agents - that is, to pose knowledge at a social level, grounded in (patterns of) interactions, not in states. Said another way, the patterns of agent interaction, or their organization can be a way of describing what they can and cannot do as a community [Gasser 91]. In a pragmatic sense this focus on patterns of interaction reflects a notion of *knowledge in action* rather than knowledge in a repository. OBCP has actually made some progress in this direction—see especially [Hewitt 77] which suggested a change in the ways we reason about control from a focus in instructions to a focus on patterns of messages.

State: The internal state of an object must be structured and connected with the inference ability. The structural and knowledge state may be kept distinct or merged. Orient84/K maintains the distinction, as discussed above. In case of merging, a way often used in object-based knowledge representation systems is to extend the notion of variable into the notion of *slot*. The value of the variable becomes one attribute among others, usually, default-value, domain, daemons, etc.

Another alternative may be found in agent-oriented programming [Shoham 90, Shoham 91]. The state of an agent is constrained (also its communication, as seen above) among several types called mental states: belief, commitment and capability.

The last issue has to do with representation of other agents. Simple referencing of acquaintances needs to be enriched to provide a representation of other agents (such as symbolic representation of other agents role and skills in MACE [Gasser et al. 87a]). Reflection, by modeling self-representations, seems a promising approach for agents to exchange representations.

Organization: Models of organizations are needed to structure coordination between agents, that is to describe *how* to compose/decompose and allocate tasks between agents and achieve coherent interaction. Models of organizations may come from the study of human organizations, administration (office semantics), organization theory, and from the formal design and implementation of stable coordination structures. [Hewitt and Inman 91] proposes concepts of organizations derived from the study of human organizations, as a structuring tool on top of objects. Explicit knowledge by the agents of their organizational role could allow them to manage themselves the reconfiguration of their organization [Ishida et al. 92].

6 Conclusions

This study is based on a survey of various OBCP and DAI platforms. We have begun to analyze the gap between OBCP and DAI more clearly by relating to a problem of integrating two types of theories: theories of social organization that describe how to represent and solve DAI problems, and theories of modeling and implementation that describe how to represent problem-solving theories in computational terms. We analyzed several proposed OBCP frameworks to find way to improve their abilities to model DAI social theories. We believe in progress of OBCP and DAI fields through mutual awareness of their respective domains and better coordination.

Some of the most promising avenues in OBCP research are the trends toward higher-level objects, object aggregation, and self representation mechanisms based on reflection. These promise to be useful techniques for implementing agent aggregates such as organizations, as well as for capturing the role-taking and remote-referencing needed in integrated social theories.

7 Acknowledgements

Much of this work took place under the generous support of the LAFORIA Laboratory at the Universite de Paris VI and the Centre de Sociologie de l'Innovation, Ecole Superieure des Mines de Paris. We especially thank Thierry Bouron, Michel Callon, Jacques Ferber, Bruno Latour, Jean-Francois Perrot, and Genevieve Teil. Several years' worth of discussions with Jacques on the ideas behind MACE were instrumental in formulating this paper. In addition, conversations with and comments from Michael Arbib, Nikos Avouris, Carl Hewitt, Marc van Liedekerke, Mario Tokoro, and Alfredo Weitzenfeld were helpful in developing and clarifying the ideas presented here.

References

[Agha 86] G. Agha, "*Actors: A Model of Concurrent Computation in Distributed Systems*," MIT Press, 1986.

[Agha et al. 89] G. Agha, P. Wegner, and A. Yonezawa (eds), "Proceedings of the OOPSLA'88 Workshop on Object-Based Concurrent Programming," *SIGPLAN Notices*, 24(4), April 1989.

[Agha 90] G. Agha, "Concurrent Object-Oriented Programming," *Communications of the ACM*, September, 1990.

[Becker 86] Howard S. Becker, *Doing Things Together*, Northwestern University Press, 1986.

[Bendifallah et al. 88] Salah Bendifallah, Francois Blanchard, Alberto Cambrosio, Joan Fujimura, Les Gasser, Elihu Gerson, Austin Henderson, Carl Hewitt, Walt Scacchi, Susan Leigh Star, Lucy Suchman, and Randy Trigg, "The Unnamable: A White Paper on Socio-Computational 'Systems.' " DAI Research Note 60, Computational Organization Design Lab, Institute for Safety and Systems Management, USC, April, 1988.

[Bennett 87] J.K. Bennett, "The Design and Implementation of Distributed Smalltalk," OOPSLA'87, *Special Issue of SIGPLAN Notices*, 22(12), pages 318–330, December 1987.

[Blumer 69] H. Blumer, *Symbolic Interactionism: Perspective and Method*, Prentice Hall, Englewood Cliffs, NJ, 1969.

[Bobrow et al. 90] D.G. Bobrow, L.G. DeMichiel, R.P. Gabriel, S.E. Keene, G. Kiczales, and D.A. Moon, "Common Lisp Object System," in *Common Lisp - the Language - Second Edition*, edited by G.L. Steele, Digital Press, pages 770–864, 1990.

[Bond and Gasser 88a] A.H. Bond and L. Gasser, "An Analysis of Problems and Research in Distributed Artificial Intelligence," in [Bond and Gasser 88b], 1988.

[Bond and Gasser 88b] A.H. Bond and L. Gasser (eds), "*Readings in Distributed Artificial Intelligence*," Morgan Kaufmann, 1988.

[Bouron 91] T. Bouron, "What Architecture for Communications Among Computational Agents?," *AAAI'91 Workshop on Cooperation Among Heterogeneous Intelligent Systems*, Anaheim, CA., 1991.

[Bouron et al. 90] T. Bouron, J. Ferber, and F. Samuel, "MAGES: a Multiagent Testbed for Heterogeneous Agents," *Proceedings of the 2nd European Workshop on Modelizing Autonomous Agents and Multi-Agent Worlds (MAAMAW'90)*, pages 221–239, Saint-Quentin en Yvelines, France, August 1990.

[Briot 89] J.-P. Briot, "Actalk: a Testbed for Classifying and Designing Actor Languages in the Smalltalk-80 Environment," *Proceedings of ECOOP'89*, Cambridge University Press, pages 109–129, July 1989.

[Briot and Yonezawa 87] J.-P. Briot and A. Yonezawa, "Inheritance Mechanisms in Object-Oriented Concurrent Languages," ECOOP'87, *LNCS*, No 276, pages 32–40, Springer-Verlag, 1987.

[Callon 80] M. Callon, "Struggles and Negotiations to Define What is Problematic and What is Not: The Sociologic of Translation," in K. Knorr, et al., eds., *The Social Process of Scientific Investigation,* Boston: Dordrecht, 1980.

[Casais 88] E. Casais, "An Object Oriented System Implementing KNOs," in *Proceedings of the ACM Conference on Office Information Systems*, pages 284–290, March 1988.

[Charon 79] Joel Charon, "*Symbolic Interactionism: An Interpretation, Evaluation, and Critique*," Prentice-Hall, Englewood-Cliffs, NJ, 1979.

[Chien and Dally 90] A.A. Chien and W.J. Dally, "Concurrent Aggregates," *Symposium on Principles and Practice of Parallel Programming*, March 1990.

[Cohen and Levesque 90] P.R. Cohen and H.J. Levesque, "Intention is Choice With Commitment," *Artificial Intelligence*, 1990.

[Corkill et al. 86] D.D. Corkill, K.Q. Gallagher and K.E. Murray, "GBB: A Generic Blackboard Development System," *AAAI'86*, pages 1008–1014, August 1986.

[Davis and Smith 83] R. Davis and R.G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving," *Artificial Intelligence*, 20(1), pages 63–109, 1983.

[Durfee et al. 87b] E.H. Durfee, V.R. Lesser, and D.D. Corkill, "Coherent Cooperation Among Communicating Problem Solvers," *IEEE Transactions on Computers*, C-36, pages 1275–1291, 1987.

[Durfee and Lesser 91] E.H. Durfee and V.R. Lesser, "Partial Global Planning," *IEEE Transactions on Systems, Man and Cybernetics*, 21(5), September/October, 1991.

[Erman et al. 88] L.D. Erman, J.S. Lark and F. Hayes-Roth, "ABE: An Environment for Engineering Intelligent Systems," *IEEE Transactions on Software Engineering*, 14(2), pages 1758–1770, December 1988.

[Fennell 77] Richard D. Fennell and Victor R. Lesser. "Parallelism in Artificial Intelligence Problem Solving: A Case Study of Hearsay-II," *IEEE Transactions on Computers*, C-26(2):98–111, 1977.

[Ferber 88] J. Ferber, "Des Objets aux Agents: Une Architecture Stratifiée," Research Report 88–18, Laforia, Université Paris VI, 1988.

[Ferber and Briot 88] J. Ferber and J.-P. Briot, "Design of a Concurrent Language for Distributed Artificial Intelligence," *International Conference on Fifth Generation Computer Systems (FGCS'88)*, Vol. 2, pages 755–762, Icot, Tokyo, Japan, November-December 1988.

[Ferber 89] J. Ferber, "Computational Reflection in Class-Based Object-Oriented Languages," OOPSLA'89, *Special Issue of Sigplan Notices*, 24(10), October 1989.

[Ferber and Carle 92] J. Ferber and P. Carle, "Actors and Agents as Reflective Concurrent Objects: a Mering-IV Perspective," *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6), November/December 1991.

[Gasser 86] L. Gasser, "The Integration of Computing and Routine Work," *ACM Transactions on Office Information Systems,*, 4(3), July, 1986.

[Gasser et al. 87a] L. Gasser, C. Braganza, and N. Herman, "MACE: A Flexible Testbed for Distributed AI Research," in M.N. Huhns, editor, *Distributed Artificial Intelligence*, Pitman - Morgan Kaufmann, pages 119–152, 1987.

[Gasser et al. 87b] L. Gasser, C. Braganza and N. Herman, "Implementing Distributed Artificial Intelligence Systems Using MACE," *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, pages 315–320, 1987.

[Gasser 89b] L. Gasser, "High-Level Objects in a Flexible Testbed for Distributed AI Research," in [Agha et al. 89], April 1989.

[Gasser and Huhns 89] L. Gasser and M.N. Huhns, "Themes in Distributed Artificial Intelligence Research," in L. Gasser and M.N. Huhns, eds., *Distributed Artificial Intelligence, Volume II*, Pitman Publishers, pages vii–xv, 1989.

[Gasser et al. 89] L. Gasser, N. Rouquette, R. Hill, and J. Lieb. "Representing and Using Organizational Knowledge in DAI Systems," in [Gasser and Huhns 89a], 1989.

[Gasser and Hill 90] L. Gasser and R.W. Hill, "Engineering Coordinated Problem Solvers," in *Annual Reviews of Computer Science*, Volume 4, Annual Reviews, Inc., Palo Alto CA, 1990.

[Gasser 91] L. Gasser, "Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics," in *Artificial Intelligence* Special Issue on Foundations of Artificial Intelligence, 1991.

[Gasser 92] L. Gasser, "Boundaries, Aggregation, and Identity: Plurality Issues for Multi-Agent Systems," in Y. Demazeau and E. Werner, eds. *Decentralized Artificial Intelligence 3*, Elsevier, 1992.

[Gerson 76] "On Quality of Life," *American Sociology Review* 41, 1976.

[Giroux and Senteni 91] S. Giroux and A. Senteni, "A Distributed Artificial Intelligence Approach to Behavioral Simulation," *Proceedings of 1991 European Simulation Multiconference*, Copenhagen, Denmark, June 1991.

[Hayes-Roth 85] B. Hayes-Roth, "A Blackboard Architecture for Control," *Artificial Intelligence*, Vol. 26, pages 251–321, 1985.

[Hewitt 77] C.E. Hewitt, "Viewing Control Structures as Patterns of Passing Messages," *Artificial Intelligence*, 8(3), pages 323–364, 1977.

[Hewitt 86] C.E. Hewitt, "Offices are Open Systems," *ACM Transactions on Office Information Systems*, 4(3), pages 271–287, 1986. Also in Huberman, B.A., *The Ecology of Computation*, Elsevier Science Publishers - North Holland, 1988.

[Hewitt and Inman 91] C. Hewitt and J. Inman, "DAI Betwixt and Between: From 'Intelligent Agents' to Open Systems Science," *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6), November/December 1991.

[Hewitt 91] C.E. Hewitt, "Open Information Systems Semantics for Distributed Artificial Intelligence," *Artificial Intelligence*, January 1991.

[Hogg and Huberman 90] T. Hogg and B.A. Huberman, "Controlling Chaos in Distributed Systems," Technical Report SSL–90–52, Dynamics of Computation Group, Xerox Palo Alto Research Center, Palo Alto, CA, 1990.

[Honda and Yonezawa 88] Y. Honda and A. Yonezawa, "Debugging Concurrent Systems Based on Object Groups," ECOOP'88, *LNCS*, No 322, Springer-Verlag, August 1988.

[Ishida et al. 92] T. Ishida, L. Gasser, and M. Yokoo, "Organization Self-Design of Distributed Production Systems," *IEEE Transactions on Data and Knowledge Engineering*, 4(2), pages 123–134, 1992.

[deJong 90] P. deJong, "Organizations of Cooperating Applications," draft, IBM Cambridge Scientific Center, 1990.

[deJong 90b] P. deJong, "Concurrent Organizational Objects," OOPSLA/ECOOP'90 Workshop on Object-Oriented Concurrent Programming, *OOPS Messenger*, 2(2), ACM Sigplan, April 1991.

[Kornfeld and Hewitt 81] W.A. Kornfeld and C.E. Hewitt, "The Scientific Community Metaphor," *IEEE Transactions on Systems, Man and Cybernetics*, 11(1), pages 24–33, January 1981.

[Latour 87] Bruno Latour, *Science in Action.* Harvard University Press, 1987.

[Lenat 75] D.B. Lenat, "BEINGS: Knowledge as Interacting Experts," *IJCAI'75*, pages 126–133, 1975.

[Lescaudron et al. 91] L. Lescaudron, J.-P. Briot and M. Bouabsa, "Prototyping Programming Environments for Object-Oriented Concurrent Languages: a Smalltalk-Based Experience," *Proceedings of the 5th Conference on the Technology of Object-Oriented Languages and Systems (Tools Usa'91)*, pages 449–462, Prentice-Hall, 1991.

[Lesser 81] Victor R. Lesser and Daniel D. Corkill. "Functionally Accurate, Cooperative Distributed Systems," *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(1):81–96, January 1981.

[Lesser 83] Victor R. Lesser and Daniel D. Corkill. "The Distributed Vehicle Monitoring Testbed: A Tool For Investigating Distributed Problem Solving Networks," *AI Magazine*, pages 15–33, Fall 1983.

[Maes 88] P. Maes, "Computational Reflection," *Knowledge Engineering Review*, 3(1) pages 1-19, March, 1988.

[Majchrzak and Gasser 92] A. Majchrzak and L. Gasser, "HITOP-A: A Tool To Facilitate Interdisciplinary Manufacturing System Design," *International Journal of Human Factors in Manufacturing*, 2(3), pages 255–276, 1992.

[Manning 87] C.R. Manning, "Acore: The Design of a Core Actor Language and its Compiler," *Revised Master Thesis*, EE and CS dept., MIT, May 1987.

[Maruichi 90] T. Maruichi, "Organizational Computation," *PhD Dissertation*, EE Dept., Keio University, September 1989.

[Mazer 91] Murray S. Mazer. "Reasoning About Knowledge to Understand Distributed AI Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, 21:6, pages 1333–1346, November/December, 1991.

[Mead 34] George H. Mead, *Mind, Self and Society*, University of Chicago Press, 1934.

[Montgomery 90] T.A. Montgomery and E.H. Durfee, "Using MICE to Study Dynamic Coordination," *Proc. Second Computer Society International Conference on Tools for Artificial Intelligence*, November, 1990.

[Nierstrasz 87] O. Nierstrasz, "Active Objects in Hybrid," OOPSLA'87, *Special Issue of Sigplan Notices*, 22(12), pages 243–253, December 1987.

[Nii et al. 89] H.P. Nii, N. Aiello and J. Rice, "Experiments on CAGE and POLIGON: Measuring the Performance of Parallel Blackboard Systems," in [Gasser and Huhns 89a], pages 319–384, 1989.

[Numaoka and Tokoro 90] C. Numaoka and M. Tokoro, "Conversation among Situated Agents," *Proceedings of the 10th International Workshop on DAI*, Chapter 13, MCC, Austin TX, October 1990.

[O'Hare and Wooldridge 92] G. O'Hare and M.L. Wooldridge, "A Software Engineering Perspective on Multi-Agent Systems Design: Experience in the Development of MADE," in N.M. Avouris and L. Gasser, *Distributed Artificial Intelligence: Theory and Praxis*, Kluwer Academic Publishers, 1992.

[Parsons51] Talcott Parsons and Edward A. Shils. *Toward a General Theory of Action.* Harvard University Press, Cambridge, MA, 1951.

[Rosenschein 82] J.S. Rosenschein, "Synchronization of Multi-Agent Plans," *AAAI'82*, pages 115–119, 1982.

[Searle 69] J.R. Searle, "Speech Acts: An Essay in the Philosophy of Language," Cambridge University Press, 1969.

[Shibayama 88] E. Shibayama, "How to Invent Distributed Implementation Schemes of an Object-Based Concurrent Language - a Transformational Approach -," OOPSLA'88, *Special Issue of Sigplan Notices*, 23(11), pages 297–305, October 1988.

[Shoham 90] Y. Shoham, "Agent Oriented Programming," Technical Report, Stan-Cs-90-1335, Stanford University, Palo Alto CA, 1990.

[Shoham 91] Y. Shoham, "Agent0: An Agent-Oriented Language and its Interpreter," *Proceedings of the National Conference on AI (AAAI-91)*, pages 704–709, 1991.

[Strauss 78] Anselm L. Strauss, *Negotiations*. San Francisco: Jossey Bass, 1978.

[Tokoro and Ishikawa 84] M. Tokoro and Y. Ishikawa. "An Object-Oriented Approach to Knowledge Systems," *Proceedings of the International Conference on Fifth Generation Computer Systems*, Icot, Tokyo, Japan, pages 623–631, 1984.

[Tomlinson and Singh 89] C. Tomlinson and V. Singh, "Inheritance and Synchronization with Enabled-Sets," OOPSLA'89, *Special Issue of Sigplan Notices*, 24(10), October 1989.

[Watanabe and Yonezawa 88] T. Watanabe and A. Yonezawa, "Reflection in an Object-Oriented Concurrent Language," *OOPSLA'88*, Special Issue of Sigplan Notices, 23(11), pages 306–315. November 1988.

[Weick 79] Karl E. Weick, *The Social Psychology of Organizing, 2nd Edition*, Addison Wesley, 1979.

[Weitzenfeld and Arbib 90] A. Weitzenfeld and M.A. Arbib, "A Concurrent Object-Oriented Framework for the Simulation of Neural Networks," OOPSLA/ECOOP'90 Workshop on Object-Oriented Concurrent Programming, *OOPS Messenger*, 2(2), pages 120–124, ACM Sigplan, April 1991.

[Wittig 92] Theis Wittig, ed., *ARCHON: An Architecture for Multi-Agent Systems* London, Ellis Horwood, 1992

[Yonezawa 90] A. Yonezawa, *ABCL: An Object-Oriented Concurrent System* MIT Press, 1990.

Contents