State of the Art in Agent-Based Modeling and Simulation of Design Processes

Technical Report No: TR-PSI-2-2004
January 2004

Authors:
Vadim Ermolayev

Department of IT
Faculty of Mathematics
Zaporozhye State University
66 Zhukovskogo st. 69063 Zaporozhye
Ukraine

Abstract. The report surveys the research activities in the area of agent-enabled modeling and simulation of design processes. It covers known models, modeling approaches and agent-based implementations addressing or relevant to design process modeling and simulation. The report also overviews research projects which have produced some valuable results contributing to the state-of-the-art in the domain. It also provides the references to the leading research facilities in the area of agent-based design process modeling, references the relevant research papers and analyses the known approaches to model design activities in Semiconductor and Electronic Systems design. The results of the analysis of the state-of-the-art point to the fact that the renaissance in Agent-Enabled Engineering in early 90-s demonstrated by the constellation of the pioneering projects in the domain has unfortunately ended up with no industrial strength solution. The successors of the reviewed projects, though been declared still provided no evidence of substantial progress in the open sources. More recent academic research is centered on the development of the enabling solutions, methodologies and infrastructures for engineering design processes. The state-of-the-art in the field suggests that it might be not really feasible to simulate creative human activities by the available methodologies, e.g., automated problem solving. Though some pioneering publications appear in attempts to formalize designers' creativity, it might be more rational to focus on the optimizing of the collaborative work of a designers' team in a dynamic self-optimizing engineering design process.

Contents

List of Figures

List of Acronyms

ABCDE	Agent-Based Concurrent Design Environment
ACDS	Automated Configuration-Design Service
ACI	Advanced Collaboration Infrastructure
ACL	Agent Communication language
ADN	Agent-Based Decision Network
AI	Artificial Intelligence
AOSE	Agent Oriented Software Engineering
ARPA	Advanced Research Projects Agency
B-MAN	Business Mobile Agent Network
CAD/CAE	Computer Aided Design/ Computer Aided Engineering
CKM	Corporate Knowledge management
COMMA	COrporate Memory Management through Agents
CSCW	Computer Supported Collaborative Work
DARPA	Defense Advanced Research Projects Agency
DDPM	Decision-based Design Process Model
DDICSP	Distributed Dynamic Interval Constraint Satisfaction Problem
DEDP	Distributed Engineering Design Process
DIDE	Distributed Intelligent Design Environment
DMDO	Distributed Multi-disciplinary Design and Optimization
DS	Data Sharing
DSC	Design Space Colonization
E-COLLEG	Advanced Infrastructure for Pan-European Collaborative Engineering
EDA	Engineering Design Activity
EDM	Engineering Design Modeling
EDP	Engineering Design Process
FIPA	Foundation for Intelligent Physical Agents
GDS	Group Decision Support
IC	Integrated Circuit
IMPACT	Improving Manufacturing Productivity with Advanced Collaboration Technology
IP	Integrated Product
IPD	Integrated Product Design
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation language
LEAP	Lightweight Extensible Agent Platform
LKB	Local Knowledge Bases
MAS	Multi-Agent System
OBNM	Objective-Based Negotiation Model
OOSE	Object-Oriented Software Engineering
OSACA	Open System for Asynchronous Cognitive Agents

OWL	Web Ontology Language
PACT	Palo Alto Cooperative Testbed
PDA	Personal Digital Assistant
RAPPID	Responsible Agents for Product-Process Integrated Design
SDM	Shared Design Model
SES	Semiconductor and Electronic Systems
SHADE	Shared Dependency Engineering
SHARE	Methodology and Environment for Collaborative Product Development
SiFA	Single Function Agents
TRMS	Tool Registration and Management Services
XML	eXtended Markup Language

1 Introduction

"Design – a signature of human intelligence – was always a great challenge for artificial intelligence (AI) research" (cf. [VAN99]). Observations of how humans act in design inspired several fundamental ideas in AI, e.g., automated problem solving and reasoning [SIM69]. In return, AI research as the broad community has attacked the problems of design domain by attempting to engineer systems and infrastructures that are capable of supporting humans in accomplishing tasks that require intelligence.

The complete process of design has not been fully automated yet in a satisfactory way, though some attempts have been undertaken. These attempts have used agents (an engineering sub-area of AI) to create intelligent software infrastructures to support design processes performed by distributed teams and comprising contributions from various disciplines.

The report surveys the research activities in the outlined area covering:

- Known models, modeling approaches and agent-based implementations addressing or relevant to design process modeling and simulation
- Research projects which have produced some valuable results contributing to the state-of-the-art in the domain

The report also provides the references to the leading research facilities in the area of agent-based design process modeling, lists the most relevant research papers and analyses the known approaches which may be applicable to model design activities in Semiconductor and Electronic Systems (SES) design. The reminder of the text is structured as follows. Section 2 presents the opinion that SES design, according to the specificity of the domain, is the kind of an Integrated Product (IP) design, analyses the dimensions of complexity in IP Design (IPD). It then enumerates possible effective solution strategies, lists the most popular design process models and enumerates the features of software infrastructures which may facilitate to making design support effective and efficient. Section 3 argues that agent-based approach is applicable to modeling and simulation of engineering design processes by pointing to the fact that the characteristic features of agents and multi-agent systems are in line with the requirements mentioned in Section 2. Section 4 surveys some of the most significant pioneering research projects aimed to create the agent-based software systems to support design activities and rounds up with the lessons learned in these first attempts. Section 5 overviews more recent research activities focused on the search for the models and the enabling frameworks, methodologies, and technologies to address dynamics in engineering design processes. It lists the factors that bring dynamics to such processes, then, continues with the survey of the recent models of engineering design processes and rounds up with the overview of the projects providing more generic enabling technologies. Section 6 gives the conclusions.

2 SES Design is the kind of Integrated Product Design

A SES design may often be considered an IP comprising various blocks with different functions, physical properties and constraints, incorporating diverse technologies and design approaches. IC design is therefore frequently performed block by block by different design teams, which sometimes originate from different disciplines (e.g., analog IC design, digital IC design). That is why IC design may be referred to as a kind of an IPD. It is well known [SBN98] that the design of an IPD has often the following complications. It is multidisciplinary, performed by distributed teams, needs re-use of diverse components which rarely fit the idea for 100 percent, and therefore requires substantial effort for proper organization and co-ordination. All these complications are often underestimated leading to the losses in the productivity, design quality and, in some extreme cases, to project failures.

2.1 Dimensions of complexity in IPD

Many authors (e.g., [SBN98]) point to the following important dimensions of complexity in IPD.

2.1.1 Boundaries between Disciplines

In different disciplines knowledge is conceptualized and represented differently both from the point of view of the notation and the context. Indeed, different knowledge representation languages, different terms and different shared conceptualizations (ontologies) are used in different disciplines. The most substantial consequence of that is the lack of means for communicating to the outside world, which is very important for IPD. It is also difficult to collaborate, to resolve conflicts, etc.

2.1.2 Different Built-in Goals

In addition to different knowledge conceptualizations the representatives of different disciplines or different parts in a distributed design team may have different goals. As in different disciplines knowledge is accumulated and used independently, the local goals of autonomous participants are often in conflict with the global goals of the design. Ignoring the conflicts between local and global goals drives the design process to possible lifelocks or even deadlocks.

2.1.3 Design in ‘Big Chunks’

Disciplinary designs are processed in large segments. Examples are: an engine, a chassis, a body of a car; analog IC, digital IC. Big chunks make integration difficult because valuable information, for example, decisions that may lead to conflicts, is hidden from the rest of participants. There is also a large overhead in repeating large, discipline-based segments because of possible failures. However, the cycles are natural in the domain because of the iterative nature of design.

2.1.4 Counter-Intuitive Behavior in Design Teams

Complex Systems, like IPD teams, possess Counter-Intuitive Behavior. “It has become clear that complex systems are counter-intuitive, that is they give indications that suggest corrective action which will often be ineffective or even adverse in its results” (cf. [FOR69]). Reason: impossible to oversee all the details and the consequences. Systems (e.g., design teams for complex products like Boeing 777, Intel processor chip) performing designs are an example of complex systems with counter-intuitive behavior. Recipe: Proper corrective actions should emerge within the system.

2.2 Solution Strategies for IPD

The following solution strategies may effectively attack the mentioned complications in IPD [SBN98].

2.2.1 Small Design Methods

To be smoothly integrated the big chunks of design should be broken into small (atomic) pieces – design activities.

A design activity is a procedure or a body of orderly procedures for accomplishing a design task (e.g., design synthesis, design selection, and design evaluation). Breaking up the design into pieces corresponds to breaking design activities into smaller activities. Smaller design activities means:

- Fewer decisions are made in each activity
- Shorter time is spent in an activity
- Less information is produced as a result of executing that activity

Smaller design activities are simpler and consume less resources. However, breaking down the process in many activities at various degrees of granularity may be more complex from the point of view of the process control and coordination for different executives at different organizational levels.

2.2.2 Opportunistic Contribution

An opportunistic problem solving strategy facilitates integration of the contributions of different parties in the design process. An opportunistic approach in contrast to a predetermined order of contribution allows taking advantage of the diversity of different opinions and candidate solutions. In opportunistic approach every participant gets a fair chance to contribute to the goals of the design process so that all points-of-view are explored. A possible disadvantage of the opportunistic approach is that it brings less order and produces more difficulties to find out the preferred outcome.

2.2.3 Cooperation

A Cooperative Strategy provides mechanisms by which different participants adopt common goals (while rationally trying to reach their local goals). Implementation of the cooperative strategy in a distributed design process results in favoring the common goals of the design over local goals. As a result of such strategy different parties spend their diverse resources in the same direction, coherently. Cooperation

also means that the parties are aware of the other parties when posing their design constraints and proposing their solutions

Cooperation doesn't mean that the conflicts do not arise. It provides mechanisms for conflict resolution. A disadvantage of exploiting cooperative strategies is that it brings substantial computational overheads for communication and decision making among cooperating actors.

2.2.4 Least Commitment

Least commitment strategy stands for deferring the decisions that constrain future choices for as long as possible. A least commitment strategy thus reduces the number of conflicts, because, for example, it avoids committing to decisions that are made based on incomplete information. Otherwise, decisions may be made as soon as they can be, even if incomplete, arbitrary, or less trusted information is used. As a consequence, there might be more chance for conflicts to occur in the future, because such information may turn out to be invalid. It is however not clear how to assess or to measure a commitment, to ensure it to be as least as possible if the information is incomplete.

2.2.5 Concurrency in Design

Concurrent design is one of the main themes of the well-established Concurrent Engineering field. A Concurrent Strategy, in contrast to a Sequential Strategy, carries out some of the activities in parallel to each other. Concurrency in design gives freedom to all participants to contribute to the current state of the design in parallel. As a result, the design process speeds up, because the participants in the design do not have to wait in a line if they can make a contribution. Possible disadvantage of the Concurrent approach to design might be the substantial increase in the overhead for coordination and planning. Moreover, coordination and planning become more difficult because there are no ideal cases in design for which all the activities are truly independent and may be rightfully performed in parallel. It is known (see, e.g., [NL99]) that there might be various kinds of dependencies between activities. This implies that there exist only constrained possibilities to their concurrent performance.

2.3 Design Models

The most popular and widely used models for IPD are Axiomatic Design Model, Systematic Design Model, Decision-Based Design Model.

Axiomatic Design Model is based on the use of the following axioms:

– Independence axiom which suggests maintaining independence between functional requirements
– Information axiom which suggests minimizing the information content

Systematic Design Model is based on the following principles:

– Engineering design must be carefully planned and systematically executed
– A design method (activity) must integrate many different aspects of engineering

Decision-Based Design Model considers design process as the Cooperative Problem Solving Activity and suggests using problem solving methods to model

design activities. The projects which use decision-based design model and distributed problem solving technique in design are surveyed for example in [SB96].

These design models are good in theory, but are not really the methodologies because they don't help enough to the implementation in the real world settings.

2.4 What do we Need to Model DEDP, in Practice?

The analysis of the degrees of complexity and the solution strategies in IPD may suggest that the following frameworks, engineering methodologies and technologies are required to make the mentioned conceptual findings feasible:

- Dynamic Engineering Design Process Models – to ensure that the design process in the distributed settings will finally end up with the expected solution and will do it in an optimal way
- Interoperability solutions (common terminology + tool integration + open systems + dynamic environments)
- Dynamic Distributed Planning solutions (brokering, matchmaking, contracting, task decomposition, solution synthesis)
- Run-time Coordination methodologies (preserving the coherence of goals and decisions, managing the dynamically changing flow of interdependent activities)
- Dynamic Conflict Resolution techniques (equilibrium between individual rationality and group rationality, local goals vs common goals of a group, negotiation)
- Monitoring, credibility and quality assessment frameworks
- Run-time Fault Processing solutions

3 Agent-Based Approach in Design

Agent paradigm in software engineering is one of the powerful means to narrow the semantic gap between the conceptualizations we use to analyze and to model the phenomena of the real world and the resulting distributed software system.

Agent paradigm gain more and more popularity as the enabling approach in modeling dynamic distributed processes in many areas of creative activities, among which IPD holds its important place. Agents prove to be appropriate to IPD modeling and simulation because they possess, as a natural part of agency paradigm, very important features relevant to the implementation of the IPD solution strategies. The following section summarizes these features of an agent and a multi-agent system.

3.1 What an Agent is?

If compared to the objects in OOSE, which may be interpreted as the analogy of inanimate entities in the real world, agents generally represent animate objects, typically – human beings. Intelligent software agents are therefore used when the software needs to possess some 'human' features like the ability to perceive the environment and reactivity, apparent pro-active behaviour in succeeding a goal on

behalf of the human owner, ability to learn from their experience, social behaviour. One of the inherent intelligent features of agents is the ability to form social structures – teams, communities, coalitions, organizations. A **rational agent** as the member of a social structure needs to balance its **individual rationality** and **benevolence** in facilitating to the growth of the group utility. Agents often use **negotiation** mechanisms adopted from human encounters for that. An agent also needs to obey its social commitments and the conventions which regulate the group behaviour within the social structure. A team or an organization of agents that cooperate in a physically and, possibly, geographically distributed network form a software system called a **Multi-Agent System (MAS)**. An agent and a MAS are the main conceptual patterns of the Agent Oriented Software Engineering (**AOSE**).

From the engineering perspective, at the lower level of abstraction, the essential features of agents in MAS are their abilities to communicate with each other and to coordinate their activities. **Coordination** means achieving coherence in the group activities and thus providing that the solution of a problem or the accomplishment of a task is obtained with less effort, less resources consumed, and better quality. **Communication** stands for the ability to exchange the pieces of information within an encounter in a uniform way and using shared terminology. Communication among agents in an open system, which are typical in the majority of real world cases in e-business, enterprise application integration, etc., is a challenging interoperability task. The solutions are approached by standardizing the **communicative languages** (e.g., FIPA ACL) and developing formal machine-processable representations of the common terminology in the form of **ontologies**. Ontologies, formalized in ontology description languages (e.g., OWL) provide: a **conceptualization** – a formal model of the real world phenomena in a Domain; a **vocabulary** – a set of terms or symbols identifying concepts; an **axiomatization** – the rules and the constraints on concepts and their properties which capture characteristic aspects of the domain.

More details may be borrowed from, e.g. [EP02], [JEN00].

4 Pioneers in Agent-Enabled Design

There are several pioneering projects which used agent paradigm to facilitate design processes Agents in Design. The following ones are representative in the terms of the approaches used. All of them have been ended in pre-historic time – before 1996:

- SHADE: Shared Dependency Engineering [MKW93] + PACT: Palo Alto Cooperative Testbed [CEF93] (Stanford U., Lockheed, HP, Enterprise Integration Technologies, 1993)
- ACDS: Automated Configuration-Design Service (U Michigan, 1994)
- DIDE: Distributed Intelligent Design Environment (TU Compiegne, 1996)
- ABCDE: Agent-Based Concurrent Design Environment (U. Calgary, 1996)
- SHARE: A Methodology and Environment for Collaborative Product Development expanding further into: FirstLink, NextLink, ProcessLink (Stanford Centre for Design Research, EIT Inc., 1996, http://www-cdr.stanford.edu/SHARE/share.html)
- SiFA: Single Function Agents (AI in Design Group at WPI, 1996)

4.1 Shared Dependency Engineering (SHADE) and Palo Alto Cooperative Testbed (PACT)

SHADE, as reported in [MKW93], is just one but the basic initiative within a larger cooperative community looking at related issues of distributed CAD/CAE. SHADE is distinct from other approaches in its "…emphasis on a distributed approach to engineering knowledge rather than a centralized model or knowledge base. That is, not only does SHADE avoid the requirement of physically centralized knowledge, but the modeling vocabulary is distributed as well, focusing knowledge representation on specific knowledge-sharing needs." (cf. [MKW93]).

PACT [CEF93] is a demonstration of and a testbed for both the collaborative research effort and agent-based technology. SHADE and PACT, inspired by the work of Gensereth [GEN92], have given the push to the whole constellation of related initiatives. DARPA Knowledge Sharing Initiative [PFP92] is a community-wide effort to provide an adequate cross-domain semantic representation framework. The Lockheed project Knowledge Centered Design [KLS93] focused more closely on the problem of wrapping existing tools by specialized agents that were capable to communicate via the SHADE infrastructure. Another project at Lockheed, called Cosmos [MSO93], focused on providing support for negotiation and commitment reasoning within the SHADE infrastructure.

As it was reported in [CEF93] the main goal of the SHADE infrastructure and PACT project was to develop the approach to integrate existing multi-tool systems that are themselves design frameworks. The approach has been based on the following practically important constraint: individual engineering groups prefer to use their own tool suites and integration environments – there is significant investment in these self-contained systems. Provided the teams involved in design age constrained as above, the task of the project was to provide the framework for coordination and integration of such activities among distributed autonomous parts of design teams. The niche for the PACT results is thus the design projects that involve large segments of an enterprise or multiply enterprises and are intrinsically multi-disciplinary (please refer to Section 2.1).

PACT project has been focused to find solutions for:

- Cooperative development of interfaces
- Knowledge sharing among systems
- Computer-aided support for (human) negotiation and decision making

PACT is the multi-agent system comprising agents with different roles. The fist group are the agents which model different design groups using their own tool suites. These agents communicate with each other and use for that the common ontology which represent Shared Design Model (SDM). The opportunities provided by the SDM are reported as follows:

- A unified model is not needed. Instead tool models are encapsulated by SDM
- Though shared engineering language is needed for communication, it has to cover the SDM only

One of the outcomes of the project was the conclusion that it might be very hard to scale the software system which exploits SDM concentrated at the central node which inevitably becomes the single point of failure. Instead, PACT is built according to a

fully distributed scheme. PACT agents wrap respective design tools and have their local knowledge bases (LKB). These LKBs contain partial tool-specific knowledge in the terms of SDM.

Another group of PACT agents are facilitators which actually are the semantic bridges between the groups of wrapper-agents corresponding to different design teams and, therefore, having different local knowledge. The facilitators perform the following specific functions:

- Translate tool-specific knowledge into and out of standard knowledge exchange language (KIF)
- Provide a layer of reliable message passing (KQML)
- Rout outgoing messages to appropriate destination
- Initialize and monitor the execution

One of the first test cases for PACT evaluation was the running example of a robotic manipulator design – Fig. 1.

4.2 Distributed Intelligent Design Environment (DIDE)

The focus and the approach of DIDE project is very close to that of PACT and SHADE. As it was reported in [SB96] the approach of DIDE to large engineering projects is to decompose the overall task of designing a complex artifact into a set of different services. Such services are used by local teams which in turn communicate with other teams. The services are to be assigned both to human and software agents. Typically such teams would reside at different locations and be specialized in different aspects in the design of the product. The task of DIDE is to provide the intelligent environment to wrap the design tools and the design activities of different participants of the design process. Distributed Intelligent Design Environment is based on an architecture called OSACA [SB93] which stands for Open System for Asynchronous Cognitive Agents. The organization of DIDE is very close to PACT. The principal difference is that in DIDE all agents are “First Class”, i.e. truly autonomous, communicating directly, without facilitators.

4.3 Automated Configuration-Design Service (ACDS)

As reported in [DB94] ACDS project has focused on a particular types of design activities – a configuration design. It aimed to provide the solution of configuration-design problem that achieves the benefits of the concurrent engineering (CE) design paradigm. The essence of ACDS approach is that design concerns (manufacturability, testability, etc.) are applied to an evolving design throughout the design cycle. This approach attempts to identify conflicts early on, which avoids costly redesign and can lead to better products.

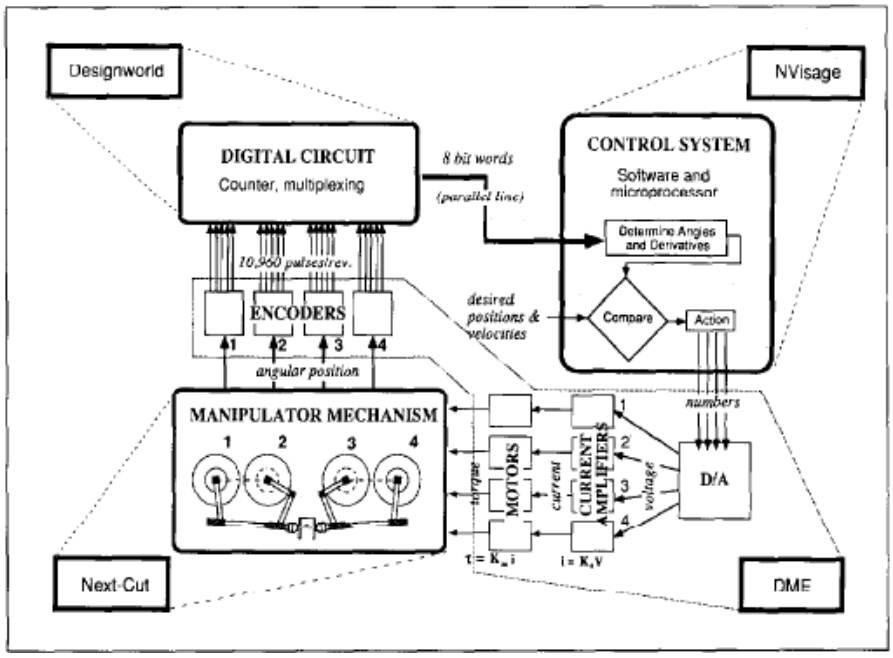

Fig. 1. Running example of a robotic manipulator design (adopted from the presentation by Greg Milette, CS525M, Spring 2002)

ACDS Framework is based on a distributed dynamic interval constraint satisfaction problem (DDICSP) model [DB94]. ACDS MAS comprises persistent catalog agents which map onto DDICSP variables and constraint agents which map onto DDICSP constraints. These agents:

– Use a set of operations and heuristics to navigate through the space of possible designs
– Rapidly eliminate sets of designs until a solution is found

ACDS is not a MAS in the traditional sense, but rather a collection of loosely-coupled, autonomous agents that organize communication among themselves based on design constraints. These agents represent part catalogs and design constraints, and consist of catalog agents, system agents, bid agents and constraint agents. ACDS agent is a computational process:

– With expertise about a limited portion of a design problem
– Capable of achieving specific goals
– Communicating with other agents by passing messages

Agents have the capability to direct other agents to perform operations within the context of the design representation and algorithm

To use ACDS, a designer needs to provide a high-level specification of the desired design and further on uses this specification to configure the ACDS network.

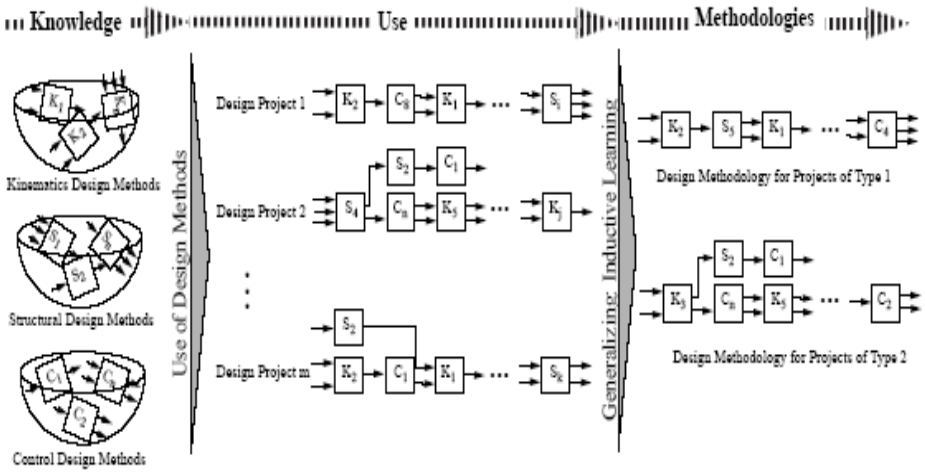

Fig. 2. SiFA: Discovering Methodologies for IPD (source [DB94]).

4.4 Single Function Agents (SiFA)

As reported in [SBN98]. The main emphasis of SiFA approach was to elaborate the agent-based methodology and knowledge repositories for modeling design processes as collaborative problem solving activities. Design is modeled as a cooperative multi-agent problem solving task where different agents possess different knowledge and evaluation criteria (single functions). The multi-agent paradigm intuitively captures the concept of deep, modular expertise that is at the heart of knowledge-based design. By implementing the opportunistic strategy in the multi-agent design system, design methods are dynamically selected from knowledge repository based on:

– The individual agents' view of the problem-solving situation
– Shared information about the capabilities of agents in the system

Therefore, the design methodology for the class of similar design processes emerges at run time. The process of design methodology discovery is shown on Fig. 2.

4.5 Agent-Based Concurrent Design Environment (ABCDE)

Agent Based Concurrent Design Environment is a multi-agent architecture for implementing concurrent engineering in manufacturing (see, e.g., [BN95]). A proof-of-concept system based on this architecture has design, manufacturability analysis, process planning, routing and scheduling as concurrent interacting activities. The system includes a feature-based design sub-system for prismatic components, implementing "intelligent features". A supervisory control interface manages shop-floor resources. Using a simulated environment of four production machines, the

system was tested with prismatic components being simultaneously designed. Manufacturability evaluation and shop floor planning were being carried out concurrently. Valid process plans, routing and scheduling were generated.

The above approach can be extended to concurrently include other product life cycle considerations at the design stage. The modularity and flexibility of this multi-agent approach can be seen to offer major advantages for implementing concurrent engineering. ABCDE has been developed with the strong emphasis to the manufacturing. Manufacturing domain has been further on investigated in MetaMorph and MetaMorph-II projects (http://imsg.enme.ucalgary.ca/), the successors of ABCDE.

4.6 SHARE: a Methodology and Environment for Collaborative Product Development

SHARE project has been broadly concerned with how information technology can help engineers develop products. Increasingly, product development in-working together over networks, supported by computation and information services. In anticipation of this future the project studied engineering teams operating in a prototype of such an environment. Specifically design teams were engaged who conceive, refine, and prototype systems for industrial sponsors

The experiences of SHARE lead to the acceptance of the view that team design is the process of reaching a “shared understanding” (cf. [TCL93]) of the domain, the requirements, the artifact, the design process and the commitments it entails. This understanding emerges and is incrementally refined in time as each group or team member develops their or his or her part of the design and diffuses the information among the others, which facilitates to their own progress. The process involves communication, negotiation, and community learning. SHARE focused on enabling these very activities which were not well supported by current CAD tools by the time of the project.

4.7 Pioneers: Lessons Learned

Reviewed projects which pioneered agent-enabled approaches to the development of the automated intelligent and distributed environments to support design processes have proved that agent paradigm is powerful enough to build feasible solutions. Two points should be stressed in this context. The first aspect to emphasize is that these projects aimed not to simulate the design activities, but rather concentrated on the provision of the enabling infrastructures, environments to support design processes. The second point is to notice that there is no evidence in the open sources that this heavy wave of effort in agent-enabled support to concurrent design and engineering produced any industrial-strength solutions by 2000. Possible reasons for this are:

– Design activities are very difficult to formalize – they are highly intuitive, creative, non-deterministic, …
– Agent technology was too immature by that time to address such a complicated issues, namely: powerful yet computationally efficient in resource bounded settings

formal frameworks for essential features were absent; AOSE methodologies were at the very beginning of their development; means for consensual knowledge representation were not sufficient

Since that the research community has concentrated on developing the enabling solutions both for formalizing the processes of design and for making AOSE more mature.

5 Enabling Solutions for Modeling Dynamic Engineering Design Processes

Major difficulties in modeling Engineering Design Processes (EDP), as found out by the projects reviewed in Section 3, are caused by the facts that EDPs, though well defined in many domains (e.g. in Semiconductor and Electronic Systems Design), are still too complex to be rigidly formalized and are highly dynamic in their nature. EDPs are dynamic according to a number of factors that make it impossible to plan or to define an EDP in all details before it actually starts:

- **Functional decomposition**. As the conceptual idea of a design may be decomposed into the functional blocks in different ways by different designers it is impossible to define the concurrent threads of this EDP in advance
- **Altering capabilities**. As the capabilities (the workload and the experience) of the designers change in time it is not well clear how to plan the optimal configuration of the flow performers with respect to the accomplishment time, the quality, and the cost of service.
- **Design Solution reuse**. As the designs are often re-used or adopted from the other designs the major technological design steps may vary out of this
- **Backtrack loops**. As by the result of the verification at any EDP step it might be necessary to backtrack to one of the previous steps, it is impossible to plan the number of such loops in advance
- **Tool choice**. As the characteristics of the different tools which may be used at a specific design step vary with respect to both productivity and the working experience a designer has in using these tools, it is hard to predict in advance which tool will be optimal for the step, and which one will be actually chosen by the designer. The choice of a tool may of course influence the resulting number of design iterations at this EDP step.

These factors point to the necessity to take the decisions on the configuration of an EDP “on the fly”, in line with its actual execution, each time an optimal path should be chosen from the set of possible alternatives. Agent-based approach may be appropriate to arrange such Dynamic EDPs (DEDP). The enabling agent-based solutions were under intensive investigation at least in the following projects. We place these research activities in two groups: the first one are the projects aiming to develop agent-based models and frameworks to support design activities; the second groups the some of the agent-related research, mainly in different aspects of organizational and environmental dynamics, which results may be further applicable to model DEDPs.

5.1 New(er) Models for DEDP

The following two projects are characteristic for the attempts to bring more dynamics to EDP models:

– RAPPID: Responsible Agents for Product-Process Integrated Design (Van Parunak, Altarum Inc., ARPA MADE funded project, 1999, http://www.erim.org/cec/rappid/rappid.htm)
– ADN – Agent-Based Decision Network (University of Southern California, IMPACT Lab, http://impact.usc.edu/projects.htm)

5.1.1 Responsible Agents for Product-Process Integrated Design (RAPPID)

As reported in [PWF99] the goal of the project was to develop DEDP Models based on the market mechanisms. RAPPID uses a marketplace to establish a price-per-unit for each characteristic of a design. Agents, representing design project stakeholders for each component, buy and sell units of these characteristics on a network-based market server. A component wrapper agent represents a part of the design, buys and sells characteristics in the market. These component wrappers may be organized in a hierarchy as the components of the design are themselves in the hierarchy. Agents may be controlled by a human user. A characteristic is understood as a definable attribute like weight or power. A characteristic wrapper agent maintains a marketplace for that item (refer to Fig. 3.).

A component (agent) that needs more latitude in a given characteristic (e.g., more weight) can purchase increments of that characteristic from another component (agent). However, it might need to sell another characteristic to raise resources for this purchase.

The market-based prices of characteristics:

– Reflect the relative scarcity of the various characteristics (that is, which ones constrain the design more closely)
– Rationalize communication among designers

In some cases, analytical models of the dependencies between characteristics are used to help designers estimate their relative costs. But even where such models are clumsy or nonexistent, prices set in the marketplace indirectly define the coupling among characteristics.

A specific Design Space in terms of RAPPID is the Cartesian product in the Cartesian space of design characteristics. The characteristics in the Design Space may be considered slack and constrained. The ratio of characteristic constraint is also defined by the market prices:

– Low prices mean slack characteristics
– High prices mean constrained characteristics

The approach of RAPPID is to find the proper equilibrium among the possible allocations of characteristics. A Design Space can be extended, shrank or even collapsed by buying up certain allocations of characteristics. This may give other agents more funds, more opportunities to purchase other characteristics instead. This may in turn cause the amount of certain characteristics to get fewer and converge on a price. However, Cartesian metaphor works well only if the characteristics are orthogonal. The main difficulty in formalizing such a Design Space is to determine the basic set of characteristics.

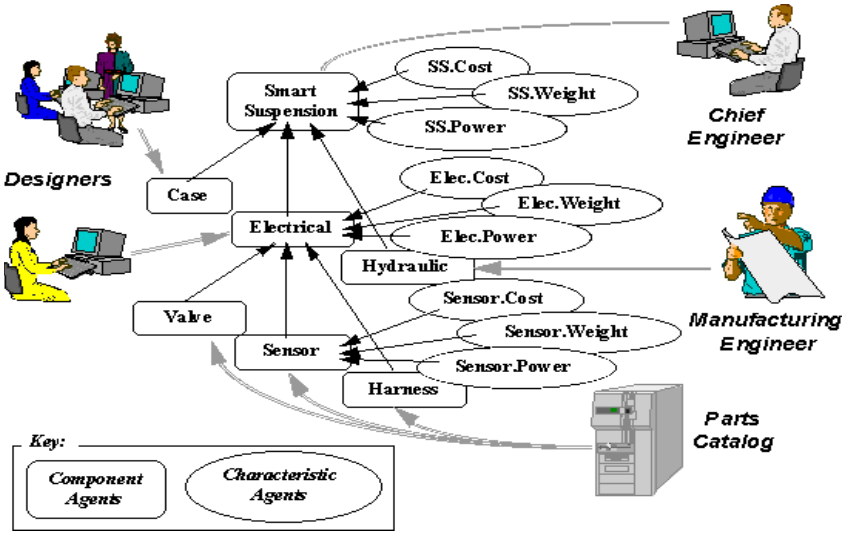

Fig. 3. An Example configuration of RAPPID marketplace (source – [PWF99]).

5.1.2 Agent-Based Decision Network (ADN)

ADN [DJ01] is the framework focusing on the collaboration aspect within DEDPs. The authors analyze the shortcomings of the existing frameworks for design process support which are prevalently based on the Data Sharing (DS) or Group Decision Support (GDS). They propose a new principle claim that collaborative design is not merely about data, but more about the processes of decision making that are carried out by multiple designers in specific organizational (functional and social) contexts and involves applications of specific design knowledge.

The ADN view is different from, e.g., GDS in that instead of focusing only on group meetings, the ADN thinking emphasizes the roles of individuals' decision processes and the links between those processes. This distinction is crucial for concurrent engineering because the key contents of concurrent engineering are individual designers' design processes and their coherent links. Group meeting is only a "snapshot" of the whole process and may not provide a complete understanding of a concurrent DEDP. The ADN view of concurrent engineering design puts emphasis on the two key actions each exercised at different levels:

– Decision-making by individual designers using decision-based design process model
– Coordination between designers on dependent activities

ADN focuses on making designers consider other team members' decisions when making their own and attempts to achieve coherent design decisions among designers by explicitly representing and enhancing individual design decision-making and negotiation processes. ADN is composed of:

– A decision-based design process model (DDPM) - captures individual designers' design processes

– An objective-based negotiation model (OBNM) - facilitates objective-based negotiation and tracks both dependencies generated and decisions made at each design stage for downstream negotiation support
– A number of intelligent agents, each associated with a human designer - generate and utilize the DDPM and OBNM information to support their designers

5.2 More Generic Enabling Solutions

The frameworks and the technologies obtained in many agent-related research projects, mainly in different aspects of organizational and environmental dynamics, may be as well beneficiary to better model DEDPs. Some examples of such projects are surveyed below:

– B-MAN: Business Mobile Agent Network (IST-2001-32285, http://www.b-man.org/)
– COMMA: Corporate Memory Management through Agents (IST, 2000, http://www.si.fr.atosorigin.com/sophia/comma/)
– E-COLLEG: Advanced Infrastructure for Pan-European Collaborative Engineering (IST-1999-11746, Jan. 2000 - Dec. 2003, http://www.ecolleg.org)
– MACRO: A tool to support Distributed Multi-disciplinary Design and Optimization (EPSRC Project (GR/L91245) June, 1998 – June, 2001, http://www.cranfield.ac.uk/coa/macro/)
– LEAP: Lightweight Extensible Agent Platform (IST-1999-10211, http://leap.crm-paris.com/infos.html)

Of course, lots of more successful research, technology transfer and development activities are on their course. Some references are, e.g., available from the Projects database of AgentLink network of Excellence (http://www.agentlink.org/resources/agentprojects-db.php).

5.2.1 Decentralized Workflow Engine

B-MAN project develops a software platform that aims at enabling the definition, enactment and management of cross-organizational business processes on top of Internet, combining:

– A decentralized agent-based workflow engine
– Secure and trusted contract-based business interactions

The features that make B-MAN different from the previous workflow solutions are:

– A fully decentralized workflow process control engine
– Explicit support for mobile computing

B-MAN platform provides **trustworthy** (as opposed to **merely secure**) cross-organizational process enactment.

5.2.2 Interoperability in CKM, CSCW

As reported in [GPR02] the emphasis of the COMMA project is the agent-based solutions for interoperability provision in distributed systems in Corporate Knowledge management (CKM) and Computer Supported Collaborative Work (CSCW). The

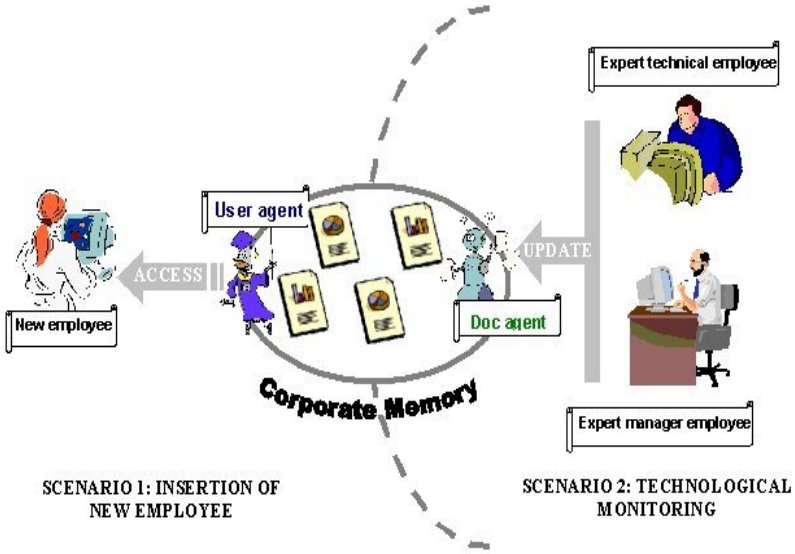

Fig. 4. Agent-enabled CMM in COMMA project (source – http://www.si.fr.atosorigin.com/sophia/comma/Documents/English/Final Paper.doc)

goal of the project is to implement and test a Corporate Memory management framework integrating several emerging technologies:

– Agent technology
– Knowledge modeling
– XML technology
– Information retrieval
– Machine learning techniques

The project intends to implement the system in the context of two scenarios:

– Enhance the insertion of new employees in the company
– Perform processes that detect, identify and interpret technology movements and interactions for matching technology evolutions with market opportunities to diffuse among employees innovative ideas related to technology monitoring activities

COMMA agents' interoperation within these two scenarios is conceptually sketched out in Fig. 4.

5.2.3 Advanced Infrastructures for Collaborative Design

The goal of the E-COLLEG project is to provide a new paradigm platform for distributed collaborative engineering through the definition and implementation of an advanced infrastructure for collaborative engineering – Advanced Collaboration Infrastructure (ACI) [KMP03]. The components of ACI are:

– Basic Collaborative Services
– Advanced Tool Registration and Management Services (TRMS)
– XML-based Integration Technologies
– Collaborative Extensions (wrappers) to Design Tools

E-COLLEG aims at the definition of the infrastructure and technology for such services developed from industrial applications in the field of embedded systems design. Moreover, the project studies common practices and metrics for Engineering Design Activities (EDA) workflow improvement.

In contrast to current industrial practice and available frameworks, this infrastructure and technology will consist of a set of interacting, location transparent services that can be dynamically configured and adapted to arbitrary tool configurations and location-independent design teams (changed consistency of the design team, transfer/delegation of tasks etc.) at run-time.

The project also develops a distributed simulation technology with collaborative verification extensions transparently linking geographically distributed designers into a concurrent verification session.

5.2.4 Networked CSCW Methodologies

MACRO project concept [MPD00] is based round the assumption that future design teams will become more distributed in nature as industry exploits the Internet and other integrated communication and data exchange systems. This concept is part of an attack on the problems associated with the total process of Distributed Multi-disciplinary Design and Optimization (DMDO). The concepts developed by the project rely on the creation of distributed self-building and self-organizing teams made up from members who are globally distributed. MACRO resulted in the implementation of the prototype software tool to support their approach to DMDO operating over the Internet. MACRO concept uses a kind of a distributed task model to support DMDO.

5.2.5 Agent Services to Support Mobile Enterprise Workforce

The LEAP project addresses the need for open infrastructures and services which support dynamic, mobile enterprises. In this context the technology developed by LEAP may be used as the enabling infrastructural solution for modeling and managing DEDPs. LEAP developed agent-based services supporting three requirements of a mobile enterprise workforce:

- Knowledge management (anticipating individual knowledge requirements),
- Decentralized work co-ordination (empowering individuals, coordinating and trading jobs)
- Travel management (planning and coordinating individual travel needs).

Central to these agent-based services is the need for a standardized Agent Platform. LEAP developed an agent platform that is: lightweight, executable on small devices such as PDAs and phones; extensible, in size and functionality; operating system agnostic; mobile team management application enabling, supporting wired and wireless communications and FIPA (http://www.fipa.org/) compliant.

6 Conclusions

The review of the extensive sources on the DEPD modeling and simulation has shown that AI research community has systematically attacked the problems of design

domain by attempting to engineer systems and infrastructures that are capable of supporting humans in accomplishing tasks that require intelligence since the early 90-ties. However, the complete process of design has not been fully automated yet in a satisfactory way, though some attempts have been undertaken. These attempts have used agents (an engineering sub-area of AI) to create intelligent software infrastructures to support design processes performed by distributed teams and comprising contributions from various disciplines.

The report surveyed the research activities in the outlined area covering:

- Known models, modeling approaches and agent-based implementations addressing or relevant to design process modeling and simulation
- Research projects which have produced some valuable results contributing to the state-of-the-art in the domain

The report also provided the references to the leading research facilities in the area of agent-based design process modeling, listed the most relevant research papers and analysed the known approaches which may be applicable to model design activities in SES design. The general impression resulting from the analysis of the state-of-the-art points to the fact that the renaissance in Agent-Enabled Engineering in early 90-s demonstrated by the constellation of the pioneering projects in the domain has unfortunately ended up with NO SILVER BULLET[1]. The successors of the mentioned projects, though been declared (e.g., Design Space Colonization (DSC, http://www-cdr.stanford.edu/DSC/), Engineering Design Modeling (EDM, http://impact.usc.edu/KICAD/projects.htm), have either gone to industry (classified) or provide no evidence of substantial progress in the open sources.

Academic research is centered on the development of the enabling solutions, methodologies and infrastructures for DEDP. The state-of-the-art in the field points to the fact that it might be not really feasible to simulate creative human activities by the available methodologies, e.g., for problem solving. Though some pioneering publications appear in attempts to formalize designers' creativity (e.g., John S. Gero et al., Key Centre of Design Computing and Cognition, University of Sydney NSW 2006, Australia [GS02]), it might be more rational to focus on the optimizing of the collaborative work of a designers' team. This is also a challenging problem because the solution must:

- Deal with the inherent DEDP dynamics and the non-determinism of its environment
- Provide mechanisms for Conflict Resolution, Backtracking, Negotiation, e.g., the formalism for negotiation in Engineering Design [SC99]
- Enable optimal Contracting in the Design Space
- Provide means for the monitoring of capabilities, credibility, quality of service exposed by freelance executives
- Facilitate to Teamwork Coordination and Planning in resource bounded settings

In order to approach such a solution it is necessary to accurately formulate the goals, to constrain the task – i.e. to shrink the Design Space, but with the least commitment possible. Then, feasible solutions will be produced with less effort.

[1] A SILVER BULLET is a methodology or a technology which expressivenes may enhance the productivity in the Domain by the order of magnitude.

References

[BN95] **Balasubramanian, S. and Norrie, D. H.**: A multi-agent intelligent design system integrating manufacturing and shop-floor control. In: Proc. First Int. Conf. on Multi-Agent Syst., San Francisco, pp. 3-9, 1995

[CEF93] **Cutkosky, M.R., Engelmore, R.S., Fikes, R.E., Genereseth, M.R., Gruber, T.R., Mark, W.S., Tenenbaum, J.M. and Weber, J.C.**: PACT: An Experiment in Integrating Concurrent Engineering Systems. *IEEE Computer* 26(1), p. 28-38, 1993

[DB94] **Darr, T. P., Birmingham, W. P.**: An Attribute-Space Representation and Algorithm for Concurrent Engineering. CSE-TR-221-94, The University of Michigan, Department of Electrical Engineering and Computer Science, Ann Arbor, Michigan 48109-2122, 1994

[DJ01] **Danesh, M. R. and Jin, Y.**: An Agent-Based Decision Network for Concurrent Engineering Design. CERA 9(1), 2001, pp 37-47

[EKK04] **Ermolayev, V., Keberle, N., Kononenko, O., Plaksin, S., Terziyan, V.**: Towards a framework for agent-enabled semantic web service composition. *Int. J. of Web Services Research*, 2004, to appear, http://eva.zsu.zp.ua/eva_personal/PS/JWSR-04-ZSU-UJF-Draft-Final.pdf

[EP02] **Ermolayev, V. A., Plaksin, S. L.**: Cooperation Layers in Agent-Enabled Business Process Management. *Problems of Programming* 1-2 (2002) 354-368

[FOR69] **Forrester, J. W.**: Urban Dynamics, MIT Press, 1969

[GEN92] **Genesereth, M.**: An Agent-Based Framework for Software Interoperability. In: Proc. of the DARPA Software Technology Conference, Meridian Corporation, Arlington, VA. 1992.

[GPR02] **Gandon, F., Poggi, A., Rimassa, G., Turci P.**: Multi-Agent Corporate Memory Management System. In: Engineering Agent Systems: Best of "From Agent Theory to Agent Implementation (AT2AI)-3", *Journal of Applied Artificial Intelligence*, Vol. 16, No 9-10. Oct. – Dec. 2002, Taylor & Francis, p. 699 – 720.

[GS02] **Gero, J.S. and Sosa, R.** (2002) Creative design situations. In: Eshaq, A., Khong, C., Neo, K., Neo, M. and Ahmad, S. (Eds), CAADRIA2002, Prentice Hall, New York, 2002, pp. 191-198. http://www.arch.usyd.edu.au/%7Ejohn/publications/2002/02SosaGeroCAADRIA.pdf

[JEN00] **Jennings, N. R.**: On Agent-Based Software Engineering. *Artificial Intelligence*, 117 (2) (2000) 277-296

[KLS93] **Kuokka, Livezey, Simoudis, and Hood**: Knowledge-Centered Design. Lockheed Artificial Intelligence Center, Technical Report, 1993.

[KMP03] **Kostienko, T., Mueller, W., Pawlak, A., Schattkowsky, T.**: Advanced Infrastructure for Collaborative Engineering in Electronic Design Automation. In: Proc. 10th ISPE Int. Conf. on Concurrent Engineering: Research and Applications, Madeira Island, Portugal, 26-30.07.2003.

[MKW93] **McGuire, J. G., Kuokka, D. R., Weber, J. C., Tenenbaum, J. M., Gruber, T. R., and Olsen G. R.**: SHADE: Technology for knowledge-based collaborative engineering. Concurrent Engineering: Research and Applications, 1(3), 1993.

[MPD00] **Morris, A.J., Payne, K.H., Deasley, P.J., Evans, S., Fielding, J.P., Guenov, M., Syamsudin, H., Thorne, J.**: MACRO - A Tool To Support Distributed MDO. In: Proc. AIAA/NASA/USAF/ISSMO Symposium on Multi-Disciplinary Analysis and Optimization, Long Beach, CA, Sept 2000. http://www.cranfield.ac.uk/coa/macro/aiaapa~1.pdf

[MSO93] **Mark, Schlossberg, Ogata, MacGregor, Kuokka, Hyde, and Livezey**: The Cosmos System for Distributed Design Negotiation Support. Lockheed Artificial Intelligence Center, Technical Report, 1993.

[NL99] **Nagendra Prasad, M. V., and Lesser, V. R.** (1999) Learning situation-specific coordination in cooperative multi-agent systems. *Autonomous Agents and Multi-Agent Systems*. 2(2), 1999, p. 173-207

[PFP92] **Patil, R.S., Fikes, R.E., Patel-Schneider, P.F., McKay, D., Finin, T., Gruber, T., and Neches, R**: The DARPA Knowledge Sharing Effort: Progress report. In: C. Rich, B. Nebel, and W. Swartout (eds), Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference, Cambridge, MA, Morgan Kaufmann. 1992.

[PWF99] **H.V.D. Parunak, A. Ward, M. Fleischer, and J. Sauter**: The RAPPID Project: Symbiosis between Industrial Requirements and MAS Research. *Autonomous Agents and Multi-Agent Systems* 2:2 (June 1999), 111-140

[SB93] **Scalabrin, E. and Barthes, J.-P**.: OSACA, une architecture ouverte d'agents cognitifs independants. In: Actes de la 2-eme Journee Nationale du PRC-IA sur les Systemes Multi-Agents, Montpellier, France, 1993

[SB96] **Shen W. & Barthes J.-P**. An Experimental Multi-Agent Environment for Engineering Design, *Int. J. of Cooperative Information Systems*, 5(2-3), pp 131-151, 1996

[SBN98] **Shakeri, C., Brown, D.C., Noori, N**.: Discovering Methodologies for Integrated Product Design. Proc. Artificial Intelligence and Manufacturing: Second Bi-annual AI & Mfg Workshop, Albuquerque, New Mexico, 1998.

[SC99] **Scott, M.**: Formalizing Negotiation in Engineering Design. PhD thesis, California Institute of Technology, Pasadena, CA, June 1999. http://citeseer.nj.nec.com/scott99formalizing.html

[SIM69] **Simon, H.**: The Sciences of the Artificial. MIT Press, Cambridge, 1969.

[TCL93] **Toye, G., Cutkosky, M.R., Leifer, L.J., Tenenbaum, J.M. and Hlicksman, J.**: SHARE: A Methodology and Environment for Collaborative Product Development. In: Post-Proceedings of the IEEE Infrastructure for Collaborative Enterprises. CDR-TR # 19930507, 1993, ftp://dart.stanford.edu/CDR/Publications/Reports/Share.ps

[VAN99] **Váncza, J.**: Artificial Intelligence Support in Design: A Survey. Keynote paper at the *1999 International CIRP Design Seminar*, Kluwer, 1999. http://www.sztaki.hu/~vancza/papers/AIsurvey1.pdf