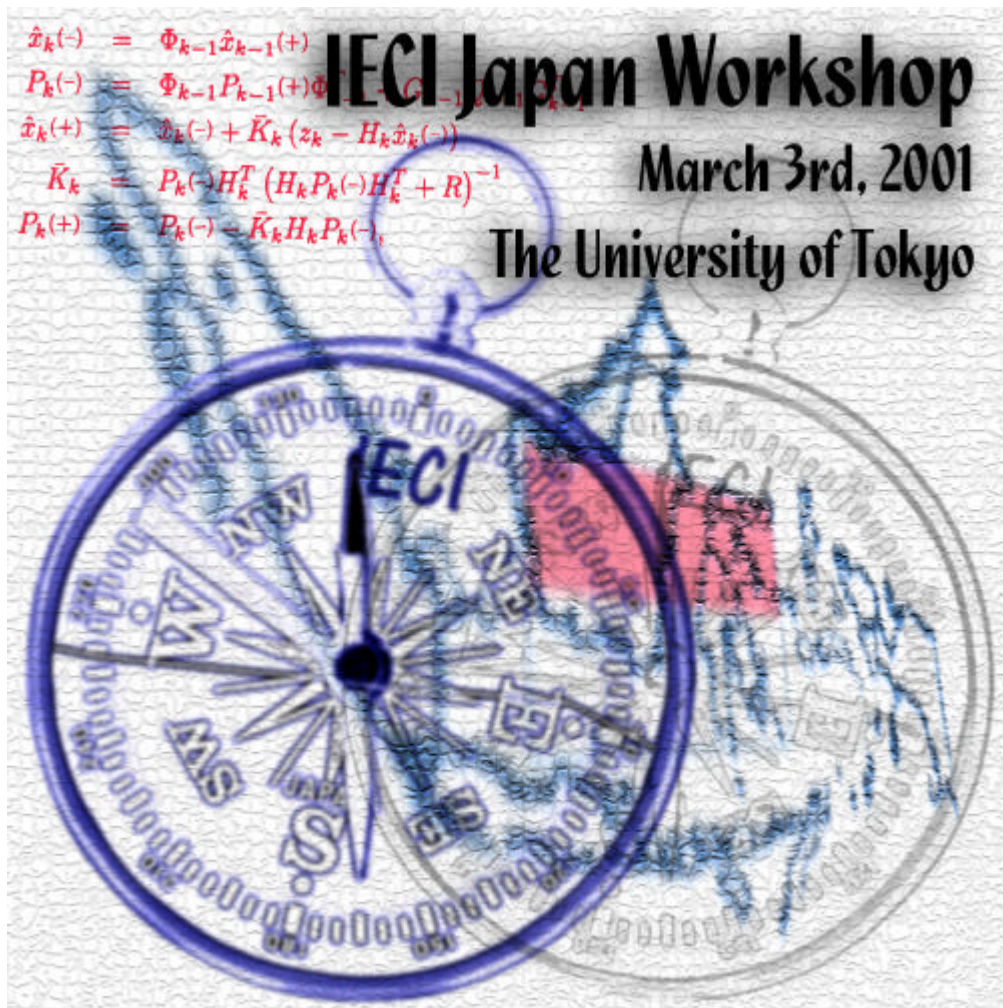


Proceedings of the IECI Japan Workshop 2001

IJW-2000



Supported by

Indonesian Society on Electrical, Electronics, Communication and Information (IECI)

Indonesian Students Association (PPI)

Institute for Science and Technology Studies (ISTECS)

Organized by

Indonesian Society on Electrical, Electronics, Communication and Information
(IECI) Japan

In Cooperation With

The University of Tokyo

Design Issues on Multi-Agent System Simulation Framework: Implications from Agent Design Strategy and Architecture

Ewin Mardhana and Behrouz H. Far

Department of Information and Computer Sciences
Graduate School of Science and Engineering, Saitama University, Japan
ewin@cit.ics.saitama-u.ac.jp

Abstract:

Despite the rapid growth of research on multi-agent system, there is still a few realization concerning a common simulation framework and standardization on it. This paper includes a survey on the process-flow of a design of multi-agent system simulation framework and extracts some key points that characterize the framework's modeling phase. A multi-agent system design methodology could serve as the basic approach for the design of such a framework. Multitude of multi-agent-based framework share some concepts while differ on some others. An interrelationship view between architecture aspects is drawn to look closer on what components are to be modeled and built when implementing a multi-agent system simulation framework.

Keywords: multi-agent system simulation framework, distributed simulation system design, multi-agent system.

Abstrak :

Walaupun riset tentang multi-agent system telah berkembang dengan pesat, hanya ada sedikit realisasi tentang *framework* simulasi yang dapat dipakai bersama dan standardisasinya. Paper ini memaparkan hasil survey tentang alur proses sebuah desain *framework* simulasi sistem *multi-agent* dan menarik poin-poin kunci yang menjadi ciri-ciri fase *modeling* sebuah *framework*. Suatu metodologi desain multi-agent system dapat menjadi pendekatan dasar untuk mendesain *framework* tersebut. Ada konsep-konsep yang sama dipakai oleh beberapa *framework* berbasis *multi-agent*, disamping perbedaan dalam hal lainnya. Suatu wawasan keterhubungan antara aspek-aspek arsitektur digambarkan untuk melihat lebih dekat komponen-komponen apa yang harus dimodelisasi dan dibangun pada saat mewujudkan sebuah *framework* simulasi sistem *multi-agent*.

Kata Kunci : *framework* simulasi sistem *multi-agent*, desain sistem simulasi terdistribusi, sistem *multi-agent*.

1. INTRODUCTION

Multi-agent system (MAS) has experienced rapid growth since 1990's and is predicted as an important breakthrough of a near future in software engineering as stated in [Singh et al., 1997] relative to DAI-based technology, adaptive complex systems and dynamical knowledge representation.

Agent concepts open possibility of software entities to travel across environments. Each of them interacts with other agents, software or human, while maintaining information about its mental states and code. This interaction varies from a knowledge sharing (where each agent

passes information relating to its beliefs, desires and intentions to its peers) to complex negotiation (where each agent tries to influence others in order to optimize its goal achievement). With information obtained or inferred from its interaction results, agent can decide a course of action in a system-wide perspective.

1.1. Notions of Agency and MAS

The definition of MAS here refers to a community of agents, which partially or completely, comply with some notions of agency. In [Wooldridge and Jennings, 1995], it is listed some notions of agency, a weak one, a stronger one and optional ones. A weak notion of agency

congregates characteristics such as

- *autonomy* (agent's self-control of its actions and internal states),
- *social ability* (agents interact with other agents),
- *reactivity* (agents can perceive their environment and adapt to changes) and
- *pro-activeness* (agents can take initiative to exhibit goal-directed behaviors).

A stronger notion of agency includes

- *mentalistic* notion [Shoham, 1993] (agents are characterized by their knowledge, belief, intention and obligation behaviors) and
- *emotional* notions [Bates, 1994].

Some optional notions are also added like

- *mobility* (ability to move around electronic network),
- *veracity* assumption (agents will not knowingly communicate false information),
- *benevolence* assumption (agents will always try to achieve their mission) and
- *rationality* assumption (agents will act towards their goals and not inversely).

Those notions above are concepts applied to human; similarly agent concepts are born to imitate various aspects of human behaviors.

A special attention could be given to the mentalistic behaviors, which inspired many researches concerning agent intelligence. The development of related concepts has been marked by works such as theory of intention [Cohen and Levesque, 1990], belief-desire-intention (BDI) agent model [Rao and Georgeff, 1995], and belief-desire-joint-intention [Jennings, 1993]. From this viewpoint, formal specifications have been proposed to represent agent's following aspects

- the *beliefs* that represent the information they have about their environment, which may be incomplete or incorrect
- the goals that agents will try to achieve.
- the actions that agents perform and their effects
- the ongoing interaction that agents have: how agents interact with each other and their environment over time.

1.2. MAS framework

A MAS framework is a set of software and/or hardware provided for problem solving using MAS architecture. A MAS simulation framework is one

of it, where a simulated environment replaces some part representing the world.

A MAS simulation framework can be used

- as a preview of an application,
- as a prediction tool for an observable phenomenon.

For the former, when the result is considered good enough some part of simulator should be replaced by real-world interface. For the latter, the result serves as an input for subsequent decision-making process. In this case, sometimes people don't need to convert the simulation into real world problem, simply because it is not useful. For example, a simulation of hunting progress in a region to predict the survivability of animals, where each animal and hunter is represented by a situated agent.

Despite MAS' rapid growth due to its central position among the extensions of popular concepts in computer science, lot of efforts still have to be contributed in term of standardizations and common framework, for application or simulation.

In the next parts of this paper, we try to extract from a survey some elements that seem important for anyone who want to build a MAS framework. However, here we try to focus on the aspects of a software-engineering process-flow for creating a simulation framework, without going too far onto the questions of application deployment over computer systems.

Firstly, in the second part, we will try to identify current trends on adopted agent development methodologies, in other words the MAS modeling strategy, which will be simulated in the framework.

Secondly, we propose a classified overview about MAS simulation architecture available, to help understanding the needs of the framework implementation.

Thirdly, based on that, we try to talk about different components revealed as consequences of MAS architecture. We try to formulate some shared concepts of simulation framework design task and to summarize some popular exceptions on those concepts.

Finally, some conclusions are made about the whole process-flow and what are components or concepts still to be developed.

2. METHODOLOGIES FOR MULTI-AGENT SYSTEM DESIGN

In a perspective of behaviors and performance analysis of the simulated agent models, they must be based on a well-defined design methodology, or a hybrid one. This forms a fundamental support for the domain-field of the simulator framework.

We can formulate MAS methodology by extending existing methods in object-oriented or knowledge engineering methodologies [Iglesias et al., 1999]. However, most apparently, the mainstream tends to build multi-agent systems based on previous research or programming experiences on software engineering, machine learning, organization and so on. In fact, many potential approaches exist due to the central position that agent occupies among some software engineering aspects like distributed software design, knowledge acquisition, finite state representation, automata and so on.

A semi-formal approach is somewhat handy solution to this uneasy analysis and design task. Without frustrating too much to formal specification and by using disparate software engineering concepts, people just can simulate agent behaviors correctly.

2.1. Object-oriented approaches for MAS design

The approach of extending object-oriented methodologies presents advantages [Iglesias et al., 1999] because of the popularity of various object-oriented design methodologies such as Object Modeling Technique (OMT) [Rumbaugh et al., 1991], Object-Oriented Software Engineering (OOSE) [Jacobson et al., 1992], Object-Oriented Design (OOD) [Booch et al., 1991] and Unified Modeling Language (UML) [Rumbaugh et al., 1998].

As stated in [Shoham, 1993], agents can be considered as *active objects*, objects with a mental state. Effectively, reusable concepts from object-oriented methodologies are

- object structures and
- structural relationships.

The extension would consist on

- remodeling method invocation and its parameterization in object-oriented method to more complex message-communicating protocol allowing implementation of

complex interaction model in agent-oriented method

- enhancing object internal structure to enable the representation of agent mental states as mentioned previously in the notions of agency.

An example of object-oriented approach can be found in [Kendall et al., 1995], which use case-driven approach in object-oriented software engineering to design the dataflow and workflow management as an agent-based system for enterprise integration.

2.2. Other approaches

The approach from Knowledge Engineering perspective starts from the fact that agents may reuse existing techniques for knowledge acquisition process. Nevertheless, extensions have to be made in order to reflect the character of distributivity and social ability. This task is much thoughtful than starting from object-oriented approaches.

Some formal approaches are also used like Concurrent MetateM language [Fischer, 1997]. Some of the most intensive works in this direction is done by MAS logical modeling [Wooldridge, 1992] and Computationally Grounded Theory of Agency [Wooldridge, 2000]. This direct overpass from agent system formal specification to agent software framework remains a heavy development task. In this case, the simulator must implement well-based formalisms, which will enable agent models, equations or logics to become computationally runnable. Other methods start from functional programming to define formalism as a multi-agent system framework.

Besides general works such as ARCHON and AgentBuilder, there are also many frameworks built from experiences in integrated manufacturing [Shen and Norrie, 1998], enterprise integration [Kendall et al., 1995], soccer game simulation [Noda, 1995], etc.

2.3. Organizational aspects

Organizations are first class entities in agent systems; explicit structures and flexible mechanisms are central to the agent paradigm [Jennings and Wooldridge, 2000]. There are many concepts for representing MAS organizational abstraction, as many as there are different structures available in human society, because MAS can

naturally be viewed as a computational organization. Complex system hierarchies, conventions and constraints are main keywords related at the same time to the concepts of organization and agency. Agent canonical view is drawn in Figure 1.

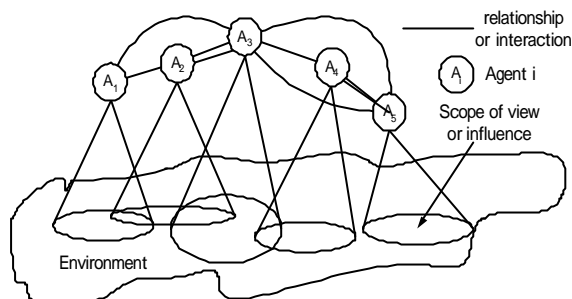


Figure 1: Canonical view of multi-agent system.

The main focus of an organizational abstraction in MAS [Zambonelli et al., 2000] is the definition of

- Organizational rules: organization constraints that allow or restrict exploitation of agent skills
- Organizational structures: topology of agent patterns and control regime of the organization
- Organizational patterns

[Wooldridge et al., 1999] propose a division of organization concepts into two categories: abstract and concrete. The abstract one, used for analysis phase, is centered around the notion of role model associated to a set of permissions, responsibilities and a number of protocols. Whereas the concrete one, used for design phase, is composed on agent model, service model and acquaintance model. This concept of role is considered more flexible and more representative of the real society than the concept of agent. In fact, a role is analogous to an agent type.

As an organization unit can be more or less loosely linked each other, there are also two main classes of multiple agent system:

- Distributed problem-solving systems, where agents cooperate to achieve a common global goal.
- Open systems, where agents not necessarily share a common goal, and can dynamically leave and enter the system.

Those organizational aspects influence the choice of the MAS architecture, which we describe

in the next part of this paper.

2.4. Case study: Cassiopeia

Cassiopeia is an agent-oriented, role-based method for the design of MAS [Drogoul and Collinot, 1998] [Drogoul and Zucker, 1998]. The organizational approach used is somewhat close to object-oriented methods. It relies on three important notions:

- (1) independence from the implementation techniques;
- (2) definition of an agent as a set of three different levels of roles;
- (3) specification of a methodological process that reconciles both the bottom-up and the top-down approaches to the problem of organization.

Several basic concepts are the foundation of Cassiopeia, namely those of role, agent, dependency and group. The main idea is an agent abstraction as a set of roles, among which we distinguish three levels (Figure 2):

- Individual roles, which are the different behaviors that the agents are individually able to perform, regardless of the policy they will choose to perform them with.
- Relational roles, that is how they choose to interact with one another (by enabling/disabling individual roles), with respect to the mutual dependencies of their individual roles.
- Organizational roles, or how the agents can manage their interactions to become or stay organized (by enabling/disabling some relational roles)

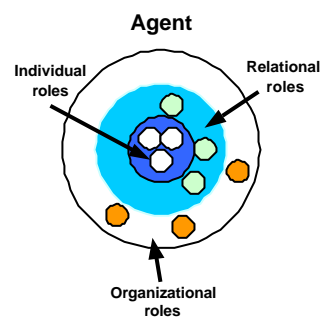


Figure 2: An abstract view of an agent with its three level of roles

The method proceeds from the definition of the collective task to the design of the MAS along five steps, depicted as layers that reconcile both the

local and global views of an MAS:

- (1) individual role layer (the bottom layer),
- (2) dependencies layer,
- (3) relational roles layer,
- (4) groups layer, and
- (5) organizational roles layer (the top layer).


Developers can take a top-down or bottom-up approach, but the usual way is to begin by the individual role layer and to end by the organizational role layer. It uses also the layered technique both for analysis and learning.

3. MAS SIMULATION FRAMEWORK ARCHITECTURE

To implement simulation framework of MAS, several type of architecture can be adopted. When all agents are separately operating in different programming context, let's call it distributed simulation architecture. However, even such a MAS simulation could also be realized in one programming context using simple inter-object messaging, let's call it centralized simulation architecture (Figure 4).

Indeed, more the simulation has a distributed character, more the transfer-process to real application is easy. It is sufficient to replace program entities simulating the world state evolution to real world interfaces.

Table 1: Different kind of architecture adopted by available MAS simulation framework.

Distributivity	Type	Key entities
centralized  distributed	Centralized architecture	Classes, objects
	Client/Server	Master Slave
	Blackboard	Planner Scheduler
	Autonomous middle agents	Assistant Facilitator Mediator Broker Matchmaker Interface
	Autonomous agents	Human Peer Cluster

A completely centralized architecture is simply the simulation architecture of classical simulation software. A client/server architecture is constituted by one or more server and one or more clients. A blackboard architecture is the transition architecture between client/server and autonomous agents.

The more the architecture is distributed, the more each agent has to manage itself all resources it needs, and to be capable to find out solutions of its problems either by itself or by consulting other agents. It implies that a framework for that case is constituted by autonomous agent libraries. To make the interactions understandable, the agents have to agree on some **conventions** and **protocols**. Anyway, to make agents distributed is the basic philosophy of agency concepts, and that will make MAS simulation models easier to implement.

Nevertheless, when the simulation codes are not intended to be converted directly to real machines, the choice of architecture is more flexible. This is for example the case of CORMAS [CIRAD, 2000], which use objects in Smalltalk to implement agents concepts based on cellular automata.

We can add a remark that, in some practice, the agent system may adopt a mixed architecture. It may exist and useful for situation-specific strategy. For example, a MAS can use facilitator agent for only one situation otherwise it adopt completely distributed architecture. One solution to not making mixed architecture like this is to adapt some agent's program in order to be able to change role.

3.1. Case study: Open Agent Architecture

Open Agent Architecture (OAA) [Martin et al., 1999] [Martin et al., 1998] has been developed through the use of *facilitator agents* as its pinpoint. All agents that are not facilitators are referred to as *client agents*. The tasks of facilitators are to coordinate agent communications and cooperative problem-solving. The used notion of facilitation is similar to that proposed in [Genesereth and Singh, 1993], where a facilitator :

- Maintains a knowledge base that records the capabilities of a collection of agents
 - Uses that knowledge to assist requesters and providers of services in making contact.
- However, in [Martin et al., 1999], it was stated

that the OAA facilitation is stronger in four respects :

- It encompasses a very general notion of transparent delegation
- It is distinguished by its handling of compound goals
- It can employ strategies and advice given by the requesting agent
- It handles the distribution of both data update requests for installation of triggers.

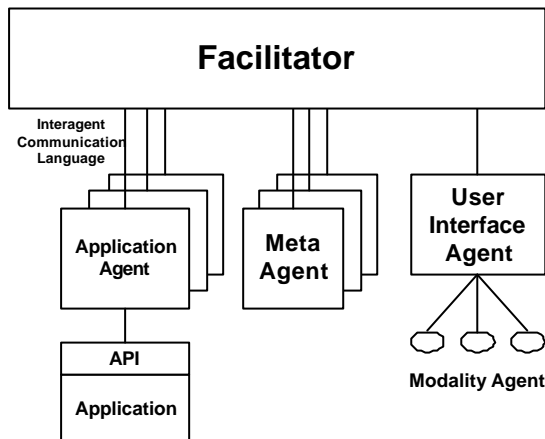


Figure 3: OAA System Structure

Finally, OOA provide autonomous monitoring with four types of triggers :

- Communication triggers
- Data triggers
- Task triggers
- Time triggers

4. FRAMEWORK COMPONENTS AND THEIR IMPLEMENTATION

An agent-oriented methodology focuses on how the structure of agents will be built. To enable this in a simulation framework, we have to build some features (components) that will allow agent designers to concentrate in their agent design-process without being irritated by the complexity of low-level programming.

To begin, we draw simplified schema of MAS sequences and list the needs for an implementable simulator. Then we talk about available strategies for those questions.

4.1. MAS sequences and related framework components

The architecture of the whole system counts for

the agent world where data, dynamics, scheduling and interaction are represented. The sequences are described like below:

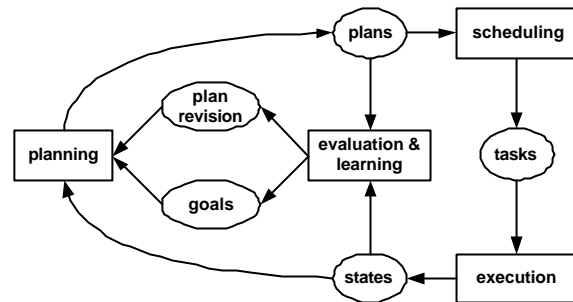


Figure 4: Simplified MAS simulator sequences

As a problem solver, MAS activities [Barber et al., 1999] includes

- **operating** within an **organization** of agents whether that organization is specified at design time or during run-time,
- **generating plans** under that organization structure,
- **allocating tasks** to proper agents,
- **integrating** agent individual plans and schedules, and
- **executing** the plans to solve the problem.

The above activities represent the needs for implementation of MAS simulation, no matter what kind of MAS design method or architecture are adopted. However, the difficulties vary depending on our choice on those previously presented approaches.

As a complement, we can add also some optional functionalities:

- analysis and learning platform (to facilitate agents' performance improvement).
- resource manager in case the agents have to deal with complex or voluminous resource for their knowledge learning.

From above, we can list related components of a MAS simulation framework:

- **communication protocol**: infrastructure and rules for agent communication has to be well-defined
- **communication synchronizer**: agent needs correct information from their environment
- **resource manager/interface**: resources are limited and conflicts have to be avoided.
- **plan generator**: plan are generated based on agent knowledge and skills
- **task allocator**: tasks are given to selected

potential agents

- **task scheduler:** agents have priorities on task and concurrent tasks have to be resolved
- **execution**
- **efficiency-driven conventions:** some efficiency strategies are needed to minimize the implementation charges

Reusability is the keyword that characterizes the components of a MAS simulator framework. It concerns at the same the problem of modeling and programming environment, execution environment and MAS improvement.

4.2. Implementation of organizational structure

As a consequence of organizational concepts pointed out earlier, we can define two kinds of organizational structure for the simulator framework:

- **Structure of agents**
Agents are constrained directly in organizational hierarchy.
- **Structure of agent roles**
Roles, not agents, are constrained in organizational hierarchy.

It seems that the second strategy has more flexibility than the first, and is closer to modern concepts of managements. Indeed, the best is to implement both strategies in a common view, where the first is only a specialization of the second. This is comparable to the concepts of *team hierarchy* or *group hierarchy* where big teams/groups are usually subdivided into smaller ones to facilitate the distribution of management tasks.

4.3. Communication language, protocol and synchronization

Some agent communication languages (ACL) are designed to fulfill the requirements of agency theories. We could cite the most popular:

- The revised Knowledge Query and Manipulation Language (KQML) [Finin and Labrou, 1997]
- FIPA Agent Communication Language (FIPA-ACL)

Otherwise, Java and C++ are also frequently used because of the appropriate use of object-oriented languages for agent communication. The DCOM technology from

Microsoft is also growing as one new among other standards.

Many agent communication protocols are based on Speech-Act theory. According to this concept, an agent receiving a message is not necessarily forced to follow it. Instead, the agent has the freedom to firstly try to understand the message and to decide what will be its reactions. As described previously, this fact is comparable to some kind of extension of inter-object communication methods in object-oriented concepts.

The questions about communication synchronization rise following the requirement that agents need valid information at a precise time. Especially, when the communication flow is very dense that it could cause interferences, the need for synchronization becomes vital.

4.4. Planning and scheduling

Plans are generated following the objective of the simulation. For that, the framework must be able to manage elementary plans (which is the simplest form of a plan) and reason based on objectives to define composed plans. Strategies for plan generation are:

- Centralized plan generation by a module in the framework
- Distributed plan generation by some planner agents
- Distributed plan generation by each agent

The centralized plan generation is reasonable only when there are no or few needs for plan revision, for example in manufacturing plant control. When the simulated system is in its stable state, there is practically no need to change the plans. When small problems arise, agents (or human operators) only need to report its specifications and let the planner module revise the plan.

When there are large or intensive needs to use adaptive plans, the second or the third strategy could be used. For systems with very high dynamics like battlefield simulation, it is better that each agent has capabilities to generate and to revise plans. Eventually, this task can be hierarchically distributed and load-balanced based on MAS organizational structure.

The second strategy could be considered as an intermediate solution between the two others,

where number of planner agents can be tuned following the planning task loads.

Common strategies for task allocation and coordination includes:

- ContractNet protocol [Smith, 1980] and its extensions [Sandholm and Lesser, 1995] [D'Inverno and Luck, 1996],
- Matchmaking and Brokering [Decker et al., 1996],
- Coalition [Shehory et al., 1998].

The task scheduling is crucial when the time is critical, therefore scheduling with many hard constraints has to be handled seriously.

As a good example in planning and scheduling phase management, TÆMS [Decker, 1995b] is developed based on Generalized Partial Global Planning (GPGP) framework [Decker, 1995a], which extended the concept of Partial Global Planning (PGP) introduced in [Durfee and Lesser, 1991] as a framework for distributed planning. The PGP framework has been initially used to manage planning in Distributed Vehicule Manipulation Testbed (DVMT). Beside that, GPGP and TÆMS has been applied for simulation of hospital scheduling where there are many deadline constraints. It uses a planning architecture independent from agent planning themselves, for managing problems of real-time execution.

GPGP extends also planning techniques of PGP significantly in some aspects :

- Heterogeneous agents: marking scheduling commitments as hard, negotiable, and soft; separation of coordination mechanisms and local scheduling mechanisms
- Dynamic agents: ability to communicate tasks at several levels of details; restructuration of node plan to hold ranges of solving methods
- Real-time agents: extension of scheduling context to better resolve hard deadlines.

4.5. Framework scalability for specific needs

There are at least 3 orientations of agent simulation frameworks existing actually: application-oriented, planning oriented, ontology-oriented.

Before implementing the simulator, we must well-distinguish in the design what to be integrated in the (simulation) framework and what to be distributed in agent programs.

Every simulator should implements only quantitatively measured state variables. It implies that the whole agent system's abstract model including goals, beliefs, desires and intentions has to be represented with measured variables. It constitutes the heaviest task of the design phase.

Because of origins of some existing simulator frameworks come from extension of an application-oriented simulation, it could be a good idea to extend those frameworks when there is needs to do research in related topics. For this, the designers have to identify the reusable modules and place them in a perspective of tailored strategies of design and implementation.

5. CONCLUSIONS AND FUTURE DIRECTIONS

This paper has provided a detailed survey and analysis of main strategies in design process-flow of MAS simulation framework. The needs for implementation of such a framework can be tailored by the selected approach in MAS design and architecture. We have tried to push the survey to satisfy the distributive behaviors of MAS.

The MAS design method is influenced by the availability of modeling and building tools. Thus, it influences the choice of programming tool and language for the framework implementation.

The MAS architecture is influenced by the simulation task load and it influences the framework's flexibility as well as its conversion to real application.

Designers can choose among different strategies exposed to scale the design process-flow and put additional customizations for efficiency.

It is obvious that many questions remains on this analysis, which could lead to further studies, for example:

- How to measure the design process task-load for a system implementation with minimized efforts and satisfied results?
- How to formulate the integrality of different steps in one modeling tools, to bridge to facilitate the system implementation?
- How to integrate realizations in machine learning in such directions? What adaptations are necessary?

6. REFERENCES

- [Barber et al., 1999] Barber, K. S., Liu, T. H., and Han, D. C. "Agent-Oriented Design." In F. J. Garrison and M. Boman, editors, *Multi-Agent System Engineering. Lecture Notes in Artificial Intelligence*, Springer, Berlin, 28-40, 1999.
- [Bates, 1994] J. Bates. "The role of emotion in believable agents." *Communications of the ACM*, 37(7):122-125, 1994.
- [Booch et al., 1991] Grady Booch. "Object-Oriented Design with Applications." Benjamin/Cummings, Redwood City, CA, 1991.
- [CIRAD, 2000] CIRAD, Department of Territories, Environment and People Land and Resources Programme. "CORMAS, Common Pool Resources and Multi-Agent Systems, User's Guide", August 2000. Available at <http://www.cirad.fr/presentation/programmes/espace/cormas/static/CormasUG.pdf>
- [Cohen and Levesque, 1990] Philip R. Cohen and Hector J. Levesque. "Intention is choice with commitment." *Artificial Intelligence*, 42(3): 213-261, 1990.
- [Decker, 1995a] Keith S. Decker. "Environment Centered Analysis and Design of Coordination Mechanisms." PhD Dissertation, University of Massachusetts Amherst, available as *UMass CMPSCI-TR-95-69*. May 1995.
- [Decker, 1995b] Keith S. Decker. "TÆMS: A framework for analysis and design of coordination mechanisms." In G. O'Hare and N. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, chapter 16. Wiley Inter-Science, 1995.
- [Decker et al., 1996] Keith Decker, Mike Williamson, and Katia Sycara. "Matchmaking and Brokering." In *Proceedings of the Second International Conference in Multi-Agent Systems (ICMAS'96)*, Kyoto, Japan, December 1996.
- [D'Inverno and Luck, 1996] M. D'Inverno and M. Luck. "Formalising the contract net as a goal-directed system." In *Proceedings of 7th MAAMAW*, pages 72-85, Eindhoven, Netherlands, 1996. LNAI Series, 1038.
- [Drogoul and Collinot, 1998] Alexis Drogoul and Anne Collinot. "Applying an Agent-Oriented Methodology to the Design of Artificial Organisations : a Case Study in Robotic Soccer." *Autonomous Agents and Multi-Agent Systems*, 1(1), 1998.
- [Drogoul and Zucker, 1998] Alexis Drogoul and J.-D. Zucker. "Methodological issues for designing multi-agent systems with machine learning techniques: Capitalizing experiences from the robocup challenge." *Technical Report LIP6 1998/041*, Laboratoire d'Informatique de Paris 6, October 1998
- [Durfee and Lesser, 1991] Edmund H. Durfee and Victor L. Lesser. "Partial global planning: A coordination framework for distributed hypothesis formation." *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5): 1167-1183, September 1991.
- [Finin and Labrou, 1997] Tim Finin, Yannis Labrou, and J. Mayfield. "KQML as an Agent Communication Language," in J. M. Bradshaw, editor, *Software Agents*, MIT Press, pp. 291-316, 1997.
- [Fischer, 1997] Michael Fisher. "An alternative approach to concurrent theorem proving." In J. Geller, H. Kitano, and C. B. Suttner, editors, *Parallel Processing in Artificial Intelligence 3* pages 209-230. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1997.
- [Genesereth and Singh, 1993] Genesereth, M. R. and Singh, N. "A Knowledge Sharing Approach to Software Interoperation," Logic Group, Computer Science Department, Stanford University, 1993.
- [Grosz and Kraus, 1993] Barbara Grosz and S. Kraus. "Collaborative plans for group activities." In *Proceedings of 13th International Joint Conference on Artificial Intelligence*, 367-373. 1993. Cambridge University Press. 220-246.
- [Iglesias et al., 1999] Carlos A. Iglesias, Mercedes Garijo and José C. González. "A Survey of Agent-Oriented Methodologies." In *Intelligent Agents V - Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Heidelberg, 1999.
- [Jacobson et al., 1992] Ivar Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object-Oriented Software Engineering. A Use Case Driven Approach*. ACM Press, 1992.
- [Jennings, 1993] Nicholas R. Jennings. "Specification and Implementation of a Belief-Desire-Joint-Intention Architecture For

- Collaborative Problem Solving.” *International Journal of Intelligent and Cooperative Information Systems*, 2 (3): 289-318, 1993
- [Jennings and Wooldridge, 2000] Jennings, N., and Wooldridge, M. "Agent-Oriented Software Engineering". In J. Bradshaw, editor, *Handbook of Agent Technology*, AAAI/MIT Press, 2000.
- [Jennings et al., 1992] Nicholas R. Jennings, E. H. Mamdani, I. Laresgoiti, J. Perez and J. Corera "Grate: A General Framework for Cooperative Problem Solving." In *IEE-BCS Journal of Intelligent Systems Engineering*, 1 (2):102-114, 1992.
- [Kendall et al., 1995] Elizabeth A. Kendall, Margaret T. Malkoun and Chong Jiang, "A Methodology for Developing Agent Based Systems for Enterprise Integration." *IFIP Working Conference of TC5 Special Interest Group on Architectures for Enterprise Integration*, Queensland, Australia, November 1995.
- [Martin et al., 1998] David L Martin, Adam J. Cheyer and Douglas B. Moran. "Building Distributed Software Systems with the Open Agent Architecture." In *Proceedings of the Practical Application of Intelligent Agents and Multi-Agent Systems*. London, UK. 355-376. 1998.
- [Martin et al., 1999] David L Martin, Adam J. Cheyer and Douglas B. Moran. "The Open Agent Architecture: A framework for building distributed software systems." *Applied Artificial Intelligence: An International Journal. Volume 13*, Number 1-2. January-March 1999.
- [Noda, 1995] Itsuki Noda. "Soccer server : a simulator of robocup." In *Proceedings of AI Symposium '95*, pages 29-34. Japanese Society for Artificial Intelligence, December 1995.
- [Nwana, 1996] Hyacinth S. Nwana. "Software Agents: An Overview", *The Knowledge Engineering Review* 11 (3), 1996.
- [Rao and Georgeff, 1995] Anand S. Rao and Michael Georgeff. "BDI-Agents, from Theory to Practice." In *Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312-319, 1996.
- [Rumbaugh et al., 1991] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy and William Lorensen. "*Object-Oriented Modeling and Design*." Prentice-Hall, 1991.
- [Rumbaugh et al., 1998] James Rumbaugh, Ivar Jacobson, and Grady Booch. "*The Unified Modeling Language Reference Manual*." Addison-Wesley, 1998.
- [Sandholm and Lesser, 1995] Tuomas Sandholm and Victor Lesser. "Issues in automated negotiation and electronic commerce: Extending the contract net framework." In *Proceedings of 1st International Conference on Multiagent Systems*, pages 328-335, San Francisco, 1995.
- [Shehory et al., 1998] Onn Shehory, Katia Sycara, and Somesh Jha. "Multi-agent coordination through coalition formation." In Munindar P. Singh, Anand S. Rao, and Michael J. Wooldridge, editors, *Proceedings of ATAL'97*. Springer Verlag, Heidelberg, Germany, 1998.
- [Shen and Norrie, 1998] W. Shen and D. H. Norrie. "Agent-Based Approaches for Intelligent Manufacturing: A State-of-the-Art Survey." In *Proceedings of DAI'98, Fourth Australian Workshop on Distributed Intelligence*, Brisbane, Australia, 13th July 1998.
- [Shoham, 1993] Yoav Shoham. "Agent-oriented programming." *Artificial Intelligence*, 60(1):51-92, March 1993.
- [Singh et al., 1997] Munindar P. Singh, Daniel G. Bobrow, Michael N. Huhns, Margaret King, Hiroaki Kitano and Ray Reiter. "The Next Big Thing : Position Statements." Launched at a panel on International Joint Conference of Artificial Intelligence, 1997.
- [Smith, 1980] Reid G. Smith. "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver." *IEEE Transactions on Computers*, C-29(12): 1104-1113. 1980.
- [Tambe, 1997] Milind Tambe. "Agent Architectures for Flexible, Practical Teamwork." In *AAAI-97*, 22-28, 1997.
- [Veloso, 1997] Manuela M. Veloso. "Learning in Planning". Tutorial in ICML/COLT, 1997.
- [Vincent et al., 1998] Régis Vincent, Bryan Horling, Tom Wagner and Victor Lesser "Survivability Simulator for Multi-Agent Adaptive Coordination". In *Proceedings of International Conference on Web-Based Modeling and Simulation*, 30(1):114-119, 1998.
- [Wooldridge et al., 1999] M. Wooldridge, N. R. Jennings and D. Kinny. "A Methodology for Agent-Oriented Analysis and Design." *Proceedings of the Agents'99*, pp. 69, Seattle,

USA, May 1999.

- [Wooldridge and Jennings, 1995] Michael J. Wooldridge, and Nicholas R. Jennings. 1995. "Intelligent agents: Theory and practice." *Knowledge Engineering Review* 10(2).
- [Wooldridge, 1992] Mike Wooldridge. "The Logical Modelling of Computational Multi-Agent Systems." PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992.
- [Wooldridge, 1994] Michael Wooldridge. "This is MYWORLD: The logic of an Agent-Oriented DAI Testbed." In *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*, pages 160-178, Amsterdam, -The Netherlands, 1994. Also appears as *Lecture Notes in Computer Science (LNAI) 890*. Springer Verlag, Heidelberg, 1995.
- [Wooldridge, 2000] Michael Wooldridge. "Computationally grounded theories of agency." In *Proceedings of ICMAS-2000, International Conference of Multi-Agent Systems*. 2000.
- [Zambonelli et al., 2000] Franco Zambonelli, Nicholas R. Jennings, and Michael J. Wooldridge. "Organisational abstractions for the analysis and design of multi-agent systems." In Paolo Ciancarini and Michael J. Wooldridge, editors, *1st International Workshop Agent-Oriented Software Engineering (AOSE 2000)*, Limerick (Ireland), June 10 2000. University of Limerick, pages 127-141, 2000.



Behrouz Homayoun Far, received BSc. and MSc. degrees in Electronic Engineering in 1983 and 1986, respectively, from Tehran University, Iran. He has received his Ph.D. degree from Chiba University - Japan, in 1990. He is currently an Associate Professor at the Department of Information and Computer Sciences, Saitama University - Japan. The research fields of his interest includes qualitative reasoning, automatic programming and distributed AI. Dr. Far is a member of the ACM, IEEE Computer Society, Japanese Society for Artificial Intelligence, IEICE and Information Processing Society of Japan.

7. BIOGRAPHY of AUTHORS



Ewin Mardhana, received Maîtrise (Bachelor Degree) in Informatics in 1995, from University of Nancy, France. Since then he has been involved in research activities in Industrial Control Research Group, Directorate for Information Technology and Electronics, BPPT, Jakarta. He is now a MEng candidate at Department of Information and Computer Sciences, Saitama University, Japan. His research fields of interests includes Multi Agent Systems, Modeling and Simulation, Fuzzy Systems, Control Systems and Software Engineering. He is a member of the Indonesian Society of Control (MASDALI) and Indonesian Society on Electrical, Electronics, Communication and Information (IECI).

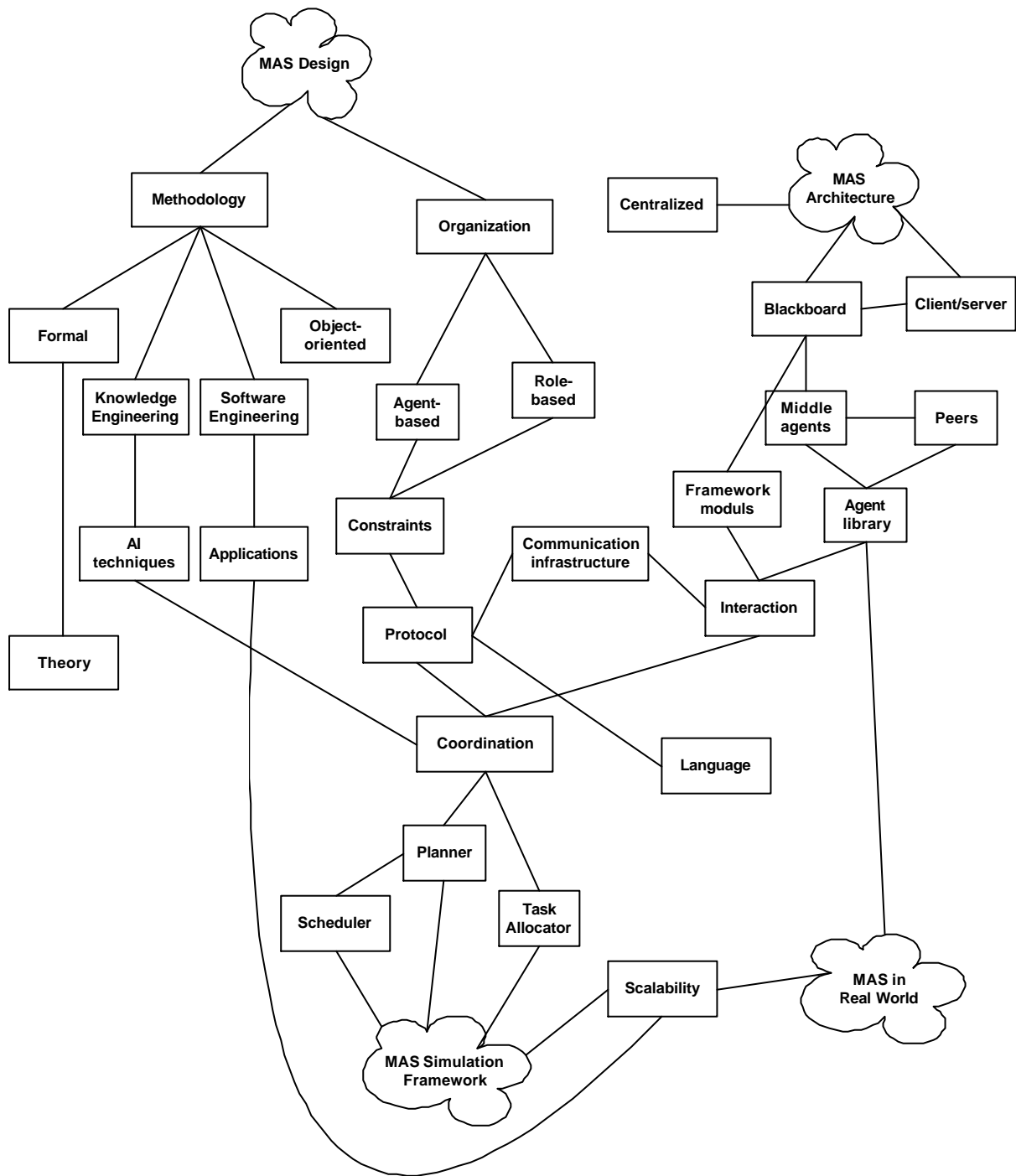


Figure 5: Interrelationship between MAS design, architecture, and simulation framework