Hot Topics

LNAI 3446

Toru Ishida
Les Gasser
Hideyuki Nakashima (Eds.)

Massively Multi-Agent Systems I

**First International Workshop, MMAS 2004
Kyoto, Japan, December 2004
Revised Selected and Invited Papers**

Springer

Toru Ishida Les Gasser
Hideyuki Nakashima (Eds.)

Massively Multi-Agent Systems I

First International Workshop, MMAS 2004
Kyoto, Japan, December 10 – 11, 2004
Revised Selected and Invited Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Toru Ishida
Kyoto University
Department of Social Informatics
Yoshida-Honmachi, Kyoto, 606-8501, Japan
E-mail: ishida@i.kyoto-u.ac.jp

Les Gasser
University of Illinois at Urbana-Champaign
Graduate School of Library and Information Science
501 East Daniel St., Champaign, IL 61820, USA
E-mail: gasser@uiuc.edu

Hideyuki Nakashima
Future University - Hakodate
116-2 Kameda-Nakanocho, Hakodate, Hokkaido, 041-8655, Japan
E-mail: h.nakashima@fun.ac.jp

Library of Congress Control Number: 2005927899

CR Subject Classification (1998): I.2.11, I.2, C.2.4, H.4, H.5.3

ISSN	0302-9743
ISBN-10	3-540-26974-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-26974-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11512073 06/3142 5 4 3 2 1 0

Lecture Notes in Artificial Intelligence 3446

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Preface

We are now in the era of ubiquitous computing and networking: millions of electronic devices with computing facilities in the public space are connected with each other in ad hoc ways, but are required to behave coherently. Massively multiagent systems (MMAS) can be a major design paradigm or an implementation method for ubiquitous computing and ambient intelligence. As the infrastructure of massively multiagent systems, technologies such as grid computing together with semantic annotation can be combined with agent technologies. A new system design approach, society-centered design, may be realized by embedding participatory technologies in human society. Applications include large-scale navigation, scientific or social simulations, e-homes, e-offices, e-cities, and e-science.

The 1st International Workshop on Massively Multiagent Systems (MMAS 2004), was held from December 10 to 11 in Kyoto, Japan. The workshop consisted of 12 invited talks, 3 chair talks, 20 oral and poster presentations, and excursions to world heritage sites in Kyoto. Participation in the workshop was by invitation only, and was limited to around 50 professionals who have made significant contributions to the topics of the meeting. Attendees were from many countries including Algeria, Australia, China, France, Korea, Luxembourg, the US, and Japan. This volume includes 25 of the papers presented at the workshop. The papers cover the area of massively multiagent technology, teams and organization, ubiquitous computing and ambient intelligence; all are related to massively multiagent systems in the public space.

At the end of the workshop, we had discussions on why MMAS should be the focus of attention rather than just MAS. Massively multiagent systems create applications for society as a whole; this raises the possibility of having a new structure in our social life via mass-support rather than individual-support. "Massive" means "beyond resource limitation": the number of agents exceeds local computer resources, or the situations are too complex to design/program given human cognitive resource limits. The discussion will be continued at the next workshop, which will be held in 2006.

March 2005

Toru Ishida
Les Gasser
Hideyuki Nakashima

Organization

Workshop Chairs

General Chair

Koiti Hasida	Information Technology Research Institute, AIST, Japan

Program Co-chairs

Toru Ishida	Department of Social Informatics, Kyoto University, Japan
Les Gasser	University of Illinois at Urbana-Champaign, USA
Hideyuki Nakashima	Future University — Hakodate, Japan

Organization Co-chairs

Hideyuki Nakanishi	Department of Social Informatics, Kyoto University, Japan
Itsuki Noda	Information Technology Research Institute, AIST, Japan

Program Committee

Robert Axtell	The Brookings Institution
Francois Bousquet	IRRI, CIRAD
Dan Corkill	University of Massachusetts
Alexis Drogoul	Laboratoire d'Informatique de Paris 6
Satoru Fujita	Internet System Research Laboratories, NEC Corporation
Nick Gibbins	University of Southampton
Hiroshi Ishiguro	Osaka University
Nadeem Jamali	University of Saskatchewan
WooYoung Kim	Univ. of Illinois at Urbana-Champaign
David Kinny	Agentis Software
Yasuhiko Kitamura	Kwansei Gakuin University
Satoshi Kurihara	NTT Network Innovation Labs.
Koichi Kurumatani	Information Technology Research Institute, AIST
Kazuhiro Kuwabara	ATR Intelligent Robotics and Communication Laboratories
Victor Lesser	University of Massachusetts
Jiming Liu	Hong Kong Baptist University
Ryusuke Masuoka	Fujitsu Laboratories of America, Inc.
Azuma Ohuchi	Hokkaido University
Ei-ichi Osawa	Future University – Hakodate
Akihiko Ohsuga	Toshiba Corporation
Van Parunak	ALTARUM
Jeffrey S. Rosenschein	Hebrew University
Larry Rudolph	MIT Laboratory for Computer Science
Norman M. Sadeh	School of Computer Science, Carnegie Mellon University
Ichiro Satoh	National Institute of Informatics
Paul Scerri	Robotics Institute, Carnegie Mellon University

Toshiharu Sugawara	NTT Communication Science Laboratory
Satoshi Tadokoro	Kobe University
Millind Tambe	University of Southern California
Walt Truszkowski	NASA Goddard Space Flight Center
Gaku Yamamoto	IBM Research, Tokyo Research Lab.
Makoto Yokoo	Kyushu University

Sponsored by

National Institute of Advanced Industrial Science and Technology (AIST)
Center of Excellence on Knowledge Society, Kyoto University
Future University — Hakodate

Supported by

Japan Science and Technology Agency (JST)
IEICE Special Interest Group for Artificial Intelligence and Knowledge Processing (SIG-AI)
IPSJ Special Interest Group for Ubiquitous Computing Systems
JSSST Special Interest Group for Multi-agent and Cooperative Computation
Support Center for Advanced Telecommunications Technology Research
The Obayashi Foundation
Microsoft Corporation
IBM Japan, Ltd.

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 3559: P. Auer, R. Meir (Eds.), Learning Theory. XI, 692 pages. 2005.

Vol. 3533: M. Ali, F. Esposito (Eds.), Innovations in Applied Artificial Intelligence. XX, 858 pages. 2005.

Vol. 3528: P.S. Szczepaniak, J. Kacprzyk, A. Niewiadomski (Eds.), Advances in Web Intelligence. XVII, 513 pages. 2005.

Vol. 3518: T.B. Ho, D. Cheung, H. Liu (Eds.), Advances in Knowledge Discovery and Data Mining. XXI, 864 pages. 2005.

Vol. 3508: P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low, M. Winikoff (Eds.), Agent-Oriented Information Systems II. X, 227 pages. 2005.

Vol. 3505: V. Gorodetsky, J. Liu, V. A. Skormin (Eds.), Autonomous Intelligent Systems: Agents and Data Mining. XIII, 303 pages. 2005.

Vol. 3501: B. Kégl, G. Lapalme (Eds.), Advances in Artificial Intelligence. XV, 458 pages. 2005.

Vol. 3492: P. Blache, E. Stabler, J. Busquets, R. Moot (Eds.), Logical Aspects of Computational Linguistics. X, 363 pages. 2005.

Vol. 3488: M.-S. Hacid, N.V. Murray, Z.W. Raś, S. Tsumoto (Eds.), Foundations of Intelligent Systems. XIII, 700 pages. 2005.

Vol. 3476: J. Leite, A. Omicini, P. Torroni, P. Yolum (Eds.), Declarative Agent Languages and Technologies II. XII, 289 pages. 2005.

Vol. 3464: S.A. Brueckner, G.D.M. Serugendo, A. Karageorgos, R. Nagpal (Eds.), Engineering Self-Organising Systems. XIII, 299 pages. 2005.

Vol. 3452: F. Baader, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XI, 562 pages. 2005.

Vol. 3446: T. Ishida, L. Gasser, H. Nakashima (Eds.), Massively Multi-Agent Systems I. XI, 349 pages. 2005.

Vol. 3438: H. Christiansen, P.R. Skadhauge, J. Villadsen (Eds.), Constraint Solving and Language Processing. VIII, 205 pages. 2005.

Vol. 3430: S. Tsumoto, T. Yamaguchi, M. Numao, H. Motoda (Eds.), Active Mining. XII, 349 pages. 2005.

Vol. 3419: B. Faltings, A. Petcu, F. Fages, F. Rossi (Eds.), Constraint Satisfaction and Constraint Logic Programming. X, 217 pages. 2005.

Vol. 3416: M. Böhlen, J. Gamper, W. Polasek, M.A. Wimmer (Eds.), E-Government: Towards Electronic Democracy. XIII, 311 pages. 2005.

Vol. 3415: P. Davidsson, B. Logan, K. Takadama (Eds.), Multi-Agent and Multi-Agent-Based Simulation. X, 265 pages. 2005.

Vol. 3403: B. Ganter, R. Godin (Eds.), Formal Concept Analysis. XI, 419 pages. 2005.

Vol. 3398: D.-K. Baik (Ed.), Systems Modeling and Simulation: Theory and Applications. XIV, 733 pages. 2005.

Vol. 3397: T.G. Kim (Ed.), Artificial Intelligence and Simulation. XV, 711 pages. 2005.

Vol. 3396: R.M. van Eijk, M.-P. Huget, F. Dignum (Eds.), Agent Communication. X, 261 pages. 2005.

Vol. 3394: D. Kudenko, D. Kazakov, E. Alonso (Eds.), Adaptive Agents and Multi-Agent Systems II. VIII, 313 pages. 2005.

Vol. 3392: D. Seipel, M. Hanus, U. Geske, O. Bartenstein (Eds.), Applications of Declarative Programming and Knowledge Management. X, 309 pages. 2005.

Vol. 3374: D. Weyns, H.V.D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems. X, 279 pages. 2005.

Vol. 3371: M.W. Barley, N. Kasabov (Eds.), Intelligent Agents and Multi-Agent Systems. X, 329 pages. 2005.

Vol. 3369: V.R. Benjamins, P. Casanovas, J. Breuker, A. Gangemi (Eds.), Law and the Semantic Web. XII, 249 pages. 2005.

Vol. 3366: I. Rahwan, P. Moraitis, C. Reed (Eds.), Argumentation in Multi-Agent Systems. XII, 263 pages. 2005.

Vol. 3359: G. Grieser, Y. Tanaka (Eds.), Intuitive Human Interfaces for Organizing and Accessing Intellectual Assets. XIV, 257 pages. 2005.

Vol. 3346: R.H. Bordini, M. Dastani, J. Dix, A.E.F. Seghrouchni (Eds.), Programming Multi-Agent Systems. XIV, 249 pages. 2005.

Vol. 3345: Y. Cai (Ed.), Ambient Intelligence for Scientific Discovery. XII, 311 pages. 2005.

Vol. 3343: C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, T. Barkowsky (Eds.), Spatial Cognition IV. XIII, 519 pages. 2005.

Vol. 3339: G.I. Webb, X. Yu (Eds.), AI 2004: Advances in Artificial Intelligence. XXII, 1272 pages. 2004.

Vol. 3336: D. Karagiannis, U. Reimer (Eds.), Practical Aspects of Knowledge Management. X, 523 pages. 2004.

Vol. 3327: Y. Shi, W. Xu, Z. Chen (Eds.), Data Mining and Knowledge Management. XIII, 263 pages. 2005.

Vol. 3315: C. Lemaître, C.A. Reyes, J.A. González (Eds.), Advances in Artificial Intelligence – IBERAMIA 2004. XX, 987 pages. 2004.

Vol. 3303: J.A. López, E. Benfenati, W. Dubitzky (Eds.), Knowledge Exploration in Life Science Informatics. X, 249 pages. 2004.

Vol. 3301: G. Kern-Isberner, W. Rödder, F. Kulmann (Eds.), Conditionals, Information, and Inference. XII, 219 pages. 2005.

Vol. 3276: D. Nardi, M. Riedmiller, C. Sammut, J. Santos-Victor (Eds.), RoboCup 2004: Robot Soccer World Cup VIII. XVIII, 678 pages. 2005.

Vol. 3275: P. Perner (Ed.), Advances in Data Mining. VIII, 173 pages. 2004.

Vol. 3265: R.E. Frederking, K.B. Taylor (Eds.), Machine Translation: From Real Users to Research. XI, 392 pages. 2004.

Vol. 3264: G. Paliouras, Y. Sakakibara (Eds.), Grammatical Inference: Algorithms and Applications. XI, 291 pages. 2004.

Vol. 3259: J. Dix, J. Leite (Eds.), Computational Logic in Multi-Agent Systems. XII, 251 pages. 2004.

Vol. 3257: E. Motta, N.R. Shadbolt, A. Stutt, N. Gibbins (Eds.), Engineering Knowledge in the Age of the Semantic Web. XVII, 517 pages. 2004.

Vol. 3249: B. Buchberger, J.A. Campbell (Eds.), Artificial Intelligence and Symbolic Computation. X, 285 pages. 2004.

Vol. 3248: K.-Y. Su, J. Tsujii, J.-H. Lee, O.Y. Kwong (Eds.), Natural Language Processing – IJCNLP 2004. XVIII, 817 pages. 2005.

Vol. 3245: E. Suzuki, S. Arikawa (Eds.), Discovery Science. XIV, 430 pages. 2004.

Vol. 3244: S. Ben-David, J. Case, A. Maruoka (Eds.), Algorithmic Learning Theory. XIV, 505 pages. 2004.

Vol. 3238: S. Biundo, T. Frühwirth, G. Palm (Eds.), KI 2004: Advances in Artificial Intelligence. XI, 467 pages. 2004.

Vol. 3230: J.L. Vicedo, P. Martínez-Barco, R. Muñoz, M. Saiz Noeda (Eds.), Advances in Natural Language Processing. XII, 488 pages. 2004.

Vol. 3229: J.J. Alferes, J. Leite (Eds.), Logics in Artificial Intelligence. XIV, 744 pages. 2004.

Vol. 3228: M.G. Hinchey, J.L. Rash, W.F. Truszkowski, C.A. Rouff (Eds.), Formal Approaches to Agent-Based Systems. VIII, 290 pages. 2004.

Vol. 3215: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part III. LVII, 906 pages. 2004.

Vol. 3214: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part II. LVIII, 1302 pages. 2004.

Vol. 3213: M.G.. Negoita, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part I. LVIII, 1280 pages. 2004.

Vol. 3209: B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou, G. Stumme (Eds.), Web Mining: From Web to Semantic Web. IX, 201 pages. 2004.

Vol. 3206: P. Sojka, I. Kopecek, K. Pala (Eds.), Text, Speech and Dialogue. XIII, 667 pages. 2004.

Vol. 3202: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Knowledge Discovery in Databases: PKDD 2004. XIX, 560 pages. 2004.

Vol. 3201: J.-F. Boulicaut, F. Esposito, F. Giannotti, D. Pedreschi (Eds.), Machine Learning: ECML 2004. XVIII, 580 pages. 2004.

Vol. 3194: R. Camacho, R. King, A. Srinivasan (Eds.), Inductive Logic Programming. XI, 361 pages. 2004.

Vol. 3192: C. Bussler, D. Fensel (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 522 pages. 2004.

Vol. 3191: M. Klusch, S. Ossowski, V. Kashyap, R. Unland (Eds.), Cooperative Information Agents VIII. XI, 303 pages. 2004.

Vol. 3187: G. Lindemann, J. Denzinger, I.J. Timm, R. Unland (Eds.), Multiagent System Technologies. XIII, 341 pages. 2004.

Vol. 3176: O. Bousquet, U. von Luxburg, G. Rätsch (Eds.), Advanced Lectures on Machine Learning. IX, 241 pages. 2004.

Vol. 3171: A.L.C. Bazzan, S. Labidi (Eds.), Advances in Artificial Intelligence – SBIA 2004. XVII, 548 pages. 2004.

Vol. 3159: U. Visser, Intelligent Information Integration for the Semantic Web. XIV, 150 pages. 2004.

Vol. 3157: C. Zhang, H. W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence. XX, 1023 pages. 2004.

Vol. 3155: P. Funk, P.A. González Calero (Eds.), Advances in Case-Based Reasoning. XIII, 822 pages. 2004.

Vol. 3139: F. Iida, R. Pfeifer, L. Steels, Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence. IX, 331 pages. 2004.

Vol. 3131: V. Torra, Y. Narukawa (Eds.), Modeling Decisions for Artificial Intelligence. XI, 327 pages. 2004.

Vol. 3127: K.E. Wolff, H.D. Pfeiffer, H.S. Delugach (Eds.), Conceptual Structures at Work. XI, 403 pages. 2004.

Vol. 3123: A. Belz, R. Evans, P. Piwek (Eds.), Natural Language Generation. X, 219 pages. 2004.

Vol. 3120: J. Shawe-Taylor, Y. Singer (Eds.), Learning Theory. X, 648 pages. 2004.

Vol. 3097: D. Basin, M. Rusinowitch (Eds.), Automated Reasoning. XII, 493 pages. 2004.

Vol. 3071: A. Omicini, P. Petta, J. Pitt (Eds.), Engineering Societies in the Agents World. XIII, 409 pages. 2004.

Vol. 3070: L. Rutkowski, J. Siekmann, R. Tadeusiewicz, L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004. XXV, 1208 pages. 2004.

Vol. 3068: E. André, L. Dybkjær, W. Minker, P. Heisterkamp (Eds.), Affective Dialogue Systems. XII, 324 pages. 2004.

Vol. 3067: M. Dastani, J. Dix, A. El Fallah-Seghrouchni (Eds.), Programming Multi-Agent Systems. X, 221 pages. 2004.

Vol. 3066: S. Tsumoto, R. S lowiński, J. Komorowski, J.W. Grzyma la-Busse (Eds.), Rough Sets and Current Trends in Computing. XX, 853 pages. 2004.

Vol. 3065: A. Lomuscio, D. Nute (Eds.), Deontic Logic in Computer Science. X, 275 pages. 2004.

Vol. 3060: A.Y. Tawfik, S.D. Goodwin (Eds.), Advances in Artificial Intelligence. XIII, 582 pages. 2004.

¥462.56元

Table of Contents

Massively Multi-agent Technology

Team and Organization

Ubiquitous Computing and Ambient Intelligence

Massively Multi-agent Systems in Public Space

Agent Server Technology for Managing Millions of Agents

Gaku Yamamoto

IBM Research, Tokyo Research Laboratory,
1623-14, Shimo-tsuruma, Yamato-shi,
Kanagawa-ken 242, Japan
+81-462-73-4639
yamamoto@jp.ibm.com

Abstract. In this paper, we describe technologies for an agent server capable of hosting millions of agents. The agent server needs a thread management mechanism, a memory management mechanism, and a recovery management mechanism. We have developed a framework and agent execution environment named Caribbean. First, we describe the programming model of Caribbean. Following the description, we explain technologies for managing millions of agents. Some application scenarios of real commercial systems using the technology are also introduced. We describe what we learned from the development of the real applications.

1 Introduction

We used a multiagent programming model for a real commercial system in 1998. The application is a service that provides information on airline tickets to consumers through the Internet. In this service, consumers have their own computer agents. Consumers input their query conditions on flights. In the system, travel agencies also have their own computer agents providing airline ticket information. A travel agency's agent has the agency's own sales policies. When a consumer inputs his or her query conditions, the consumer's computer agent sends a query condition message to the travel agency's computer agents. A travel agency's agent replies with a message containing airline ticket recommendations in accordance with its sales policies. The consumer's agent shows the airline ticket recommendations from the travel agency's agents using a Web browser. The agent also retains the input query conditions and the airline ticket information, and can show the information anytime when the consumer uses the agent again. The lifetime of a consumer agent is one week.

Through the development of the system, we found that the multiagent programming model is flexible, and it is easy to design and develop a system. However, we had to solve the serious problem that there was no multiagent platform that managed a very large number of agents. Since each consumer has an agent in the system and each agent lives in the system for a week in the commercial system, the number of consumer agents may be in the tens of thousands. Therefore, we developed our first multiagent platform that manages tens of thousands of agents on top of the

T. Ishida, L. Gasser, and H. Nakashima (Eds.): MMAS 2004, LNAI 3446, pp. 1 – 12, 2005.

Aglets framework, a mobile agent framework [1]. In 2000, we redesigned the platform and developed a new multiagent platform named Caribbean, which can manage hundreds of thousands of agents on a single platform [2-9]. In 2003, we added a server clustering function to Caribbean so that millions of agents can be managed in one system.

In this paper, we describe the technologies for a multiagent platform capable of hosting millions of agents. We call the multiagent platform that manages many agents the "agent server" in this paper. In Section 2, we describe the programming model of Caribbean. Section 3 describes the runtime structure of a Caribbean agent server. A server clustering mechanism of Caribbean is introduced in this section. Several application scenarios will be introduced in Section 4. The lessons learned from developing real applications will be described in Section 5. Conclusion is written in Section 6.

2 Caribbean Programming Model and Framework

2.1 Programming Model

In the Caribbean programming model, an agent is responsible for given roles and performs its tasks to meet its design objectives. In the typical application scenario shown in a later section, an agent is a proxy of a user. All agents are created in a server and stay in the server for a long time. Agents in Caribbean are reactive agents that execute jobs by receiving messages or events. This means that the agent does not own a thread. Occasionally, an agent requires a special service like a timer service. Such a service is provided by a "Service Object." A service object can own threads. An agent communicates with other agents by using asynchronous peer-to-peer messaging. Messages sent by an agent are put into the message queue of the destination agent. A Caribbean agent server creates a message queue for each agent. An agent server delivers a message stored in a queue to the destination agent at an appropriate time. When a message is delivered to an agent, a callback method of the agent's methods will be invoked. In that method, the agent performs its job and may send messages to other agents.

An agent server provides agents with fundamental functions such as agent creation, agent removal, and messaging. An agent server must host a large number of agents. If each agent owns a thread, the agent server will be overloaded. If too many agents are in memory, an agent server will also fail because of the resulting memory shortage. Therefore, an agent server must manage the activities of agents to control thread assignment and memory usage.

An agent server does not provide intelligent agent capabilities. If an application requires intelligent agent capabilities, the capabilities can be added as agent logic.

2.2 Framework

Figure 1 shows an overview of the Caribbean framework.

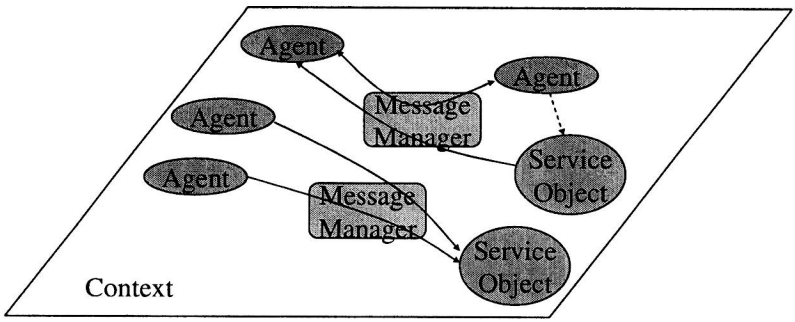

Fig. 1. Caribbean Framework

ObjectBase. The base class of agents is the "ObjectBase" class. An agent is an instance of a class extended from this class. An agent is identified by a unique identifier whose class is "OID." An agent can belong to an agent group. An application can define an agent group in a system configuration file. In the configuration file, properties for an agent group can be defined. An agent can obtain the properties from "Context." An agent must not deliver to other agents an object reference to itself. Instead, it delivers its own identifier. Instead of method calls, an agent sends other agents messages, which are instances of either the "Message" class or a class extended from the "Message" class. An agent program is based on a message-driven programming model. When an agent receives a message, a callback method "handleMessage" of the agent will be calledbacked. The delivered message is handled as an argument of the method. The agent executes the task corresponding to the message within a short period. An agent excepting an instance of the ServiceObject class described later must not own any threads. Because of memory limitations, an agent might be moved into storage by an agent server and loaded from storage back into memory. The methods that an agent must implement are as follows;

```
public void onCreation(Object arg)
```

arg: an argument of Context#create()

```
public void onDisposing()
```

This method is called by an agent server just before this agent is disposed.

```
public void onActivation()
```

This method is called by an agent server just after this agent is loaded into memory.

```
public void onDeactivating()
```

This method is called by an agent server just before this agent is stored into storage.

```
public boolean handleMessage(OID sender, Message msg,
MessageManager msgman)
```

sender: an agent identifier of a sender agent
msg: a message

msgman: an object reference to the message manager which the sender agent used to send the message.
This method is called to handle the message.

The following methods are provided by the "ObjectBase" class.

```
public Context getContext()
```

Get an object reference to a context.

```
public String getGroup()
```

Get the name of the group to which this agent belongs.

```
public OID getOID()
```

Get an agent identifier.

```
public Properties getProperties(OID oid)
```

oid: an agent identifier
Get properties of a group to which the agent specified by an agent identifier belongs

Context. Agents invoke use the agent server's functions provided through the "Context" interface. The interface provides methods for creating agents, disposing of agents, getting lists of agents in an agent server, etc. "Context" provides the following methods:

```
public OID create(String classname, Object arg)
```

classname: a class name of a created agent
arg: an argument passed to the onCreation method of ObjectBase.
Create an agent. Return the identifier of the created agent.

```
public OID create(String classname, String agentgroup,
Object arg)
```

classname: a class name of a created agent
agentgroup: an agent group name
arg: an argument passed to the onCreation method of ObjectBase.
Create an agent. The created agent belongs to the agent group. Return the identifier of the created agent.

```
public void dispose(OID oid)
```

oid: An agent identifier
Dispose of an agent.

```
public OID[] getAllOIDs()
```

Get an array of the identifiers of all agents in this agent server.

```
public OID[] getAllOIDs(String agentgroup)
```

group: an agent group

Get an array of the identifiers of all agents belong to the agent group. Return an array of agent identifiers.

```
public String getGroup(OID oid)
```

oid: an agent identifier
Get the name of the agent group to which an agent belongs

```
public MessageManager getMessageManager(String name)
```

name: the name of the messaging manager
Get an object reference of a messaging manager.

```
public SimpleMessageManager getSimpleMessageManager()
```

Get an object reference to the default messaging manager.

```
public ServiceObjectBase lookupService(String name)
```

name: a name of a service object
Get an object reference to the service object.

MessageManager. A MessageManager is an object that provides messaging functions to agents. It provides an asynchronous messaging function and a multicast messaging function to agents as fundamental functions. "MessageManager" class is an abstract class. The Caribbean package provides "SimpleMessageManager" class as a default message manager. An application can define an application-dependent MessageManager and can plug it into an agent server. For example, an application may need an anonymous messaging function that distributes a message to an appropriate agent in accordance with a message name. An application-dependent MessageManager is defined in a system configuration file. An agent server registers it into a message manager repository at system startup time. An agent obtains an object reference to a MessageManager from Context, and sends messages using those messaging functions. Context manages multiple MessageManager objects that are identified by names. The methods provided by "SimpleMessageManager" class are as follows;

```
public void post(OID[] oids, Message msg)
```

oids: an array of identifiers of destination agents
msg: a message
Multicast a message to destination agents.

```
public void post(OID oid, Message msg)
```

oid: an identifier of a destination agent
msg: a message
Post a message to a destination agent. This method does not wait until the destination agent handles the message.

ServiceObjectBase. A "Service Object" is the object that provides agents with services. A service object is an instance of a class extended from "ServiceObjectBase" class, a subclass of "ObjectBase" class. A service object has all the functions of agents. It can send messages to other agents. It also can receive