Sven A. Brueckner
Salima Hassas
Márk Jelasity
Daniel Yamins (Eds.)

LNAI 4335

Engineering Self-Organising Systems

4th International Workshop, ESOA 2006
Hakodate, Japan, May 2006
Revised and Invited Papers

Springer

Sven A. Brueckner Salima Hassas
Márk Jelasity Daniel Yamins (Eds.)

Engineering Self-Organising Systems

4th International Workshop, ESOA 2006
Hakodate, Japan, May 9, 2006
Revised and Invited Papers

Springer

Series Editors

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

Volume Editors

Sven A. Brueckner
NewVectors, LLC, Emerging Markets Group (EMG), Ann Arbor, USA
E-mail: Sven.Brueckner@newvectors.net

Salima Hassas
CExAS Team - LIRIS, Université Claude Bernard Lyon 1, Villeurbanne, France
E-mail: hassas@liris.cnrs.fr

Márk Jelasity
University of Szeged, Institute of Informatics, Hungary
E-mail: jelasity@inf.u-szeged.hu

Daniel Yamins
Harvard University, Div. of Engineering and Applied Sciences, Cambridge, USA
E-mail: yamins@fas.harvard.edu

Library of Congress Control Number: 2006940399

CR Subject Classification (1998): I.2.11, C.2.4, C.2, D.2.12, D.1.3, H.3, H.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-69867-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-69867-8 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11976783 06/3142 5 4 3 2 1 0

Lecture Notes in Artificial Intelligence 4335

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Preface

The Fourth International Workshop on Engineering Self-Organizing Applications (ESOA) was held on May 9, 2006 in conjunction with the 2006 Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2006), in Hakodate, Japan. The present post-proceedings volume contains revised versions of the seven papers presented at the workshop, and six additional invited papers. Continuing the tradition of previous editions, this book discusses a broad variety of topics in an effort to allow room for new ideas and discussion, and eventually a better understanding of the important directions and techniques of our field.

In "Hybrid Multi-Agent Systems: Integrating Swarming and BDI Agents"—an article based on an invited talk at the workshop by Van Parunak—Parunak et al. address an important question facing the ESOA community: how should self-organizing swarm-like agent approaches relate to the techniques of the multi-agent community at large? ESOA techniques primarily rely on simple *reactive* agents, whose intelligence emerges at the group level via carefully designed interaction rules. These simple agents might have some *internal state* that allows them to remember the history of their interactions at some (low) level of detail, but generally the complexity in such systems arises from the dynamics. In contrast, the mainstream multi-agent systems community uses intelligent agents, which apply sophisticated algorithms to build up internal models of their environments and complex protocols to communicate about their models. This general approach, of which the BDI frameworks are an example, warrant a more cognitive analogy than the typical ESOA ideas. Parunak et al.'s work shows how the two approaches could profitably interact.

Two of the articles advance novel "design concepts"—that is, architectures that are optimized for achieving decentralized behavior from the algorithm design perspective. In "An Analysis and Design Concept for Self-Organization in Holonic Multi-Agent Systems," Rodriguez et al. describe the concept of a *holon* as a particular kind of multi-agent hierarchy, and apply it to design adaptive systems. Tom De Wolf and Tom Holvoet fold the standard motifs of emergent multi-agent systems into the programming techniques of standard computer science. By making gradients, a technique of spatial distributed systems, and market-mechanisms, a technique of non-spatial distributed systems, into standardized design patterns, they provide the beginnings of a framework for systematic design of self-organizing systems.

In "Measuring Stigmergy: The Case of Foraging Ants," Gulyas et al. begin to explore well-defined system-level observables that can quantitatively capture the qualitative sense of *emergence* in multi-agent systems. In the specific case of ant agents foraging in a 2D spatial environment for a conserved food resource, they define an entropy-like measure of disorder on the ant positions and food positions, and observe the dynamics off these measures. Although preliminary, this paper raises the important question of whether there are underlying statistical mechanics-like principles that apply to emergent multi-agent systems. Answering this question will in the long run provide an important part of the underlying theory of emergent distributed systems.

In "Dynamic Decentralized Any-Time Hierarchical Clustering," Parunak et al. introduce a technique for maintaining the hierarchical clustering of dynamically changing, streaming data using strictly local computations. The algorithm is inspired by ant nest-building.

Mamei and Zambonelli's work on "Programming Modular Robots with the TOTA Middleware" and Shabtay et al.'s paper on "Behaviosites: A Novel Paradigm for Affecting Distributed Behavior" shared the common theme of developing frameworks for the simple manipulation and control of distributed systems. The TOTA middleware, as developed in the past few years by Mamei and Zambonelli, is an efficient and elegant language through which general distributed behaviors can be designed and propagated through multi-agent systems. A *Tuple on the Air* is a data structure that contains a behavioral program, together with rules for its propagation and maintenance. If a single robot in a system is infected with the appropriate TOTA, its behavior can effectively propagate and emergently control the functioning of the system as a whole. Here, Mamei and Zambonelli apply the TOTA approach to programming a variety of motion routines (walk, crawl, roll, etc.) in snake-like robots.

While the Mamei and Zambonelli work is inherently spatial, referring as it does to the geometric motions of physical robots, the Shabtay et al. work is about behavioral programming in non-spatial systems. Their behaviosites are pieces of code that infect and multiply within a community of functioning agents, manipulating the responses of the agents so as to change and potentially improve their behavior. They apply their idea to the El Farol bar problem. Although the two papers are applied to quite different problems, their common idea of standardized code fragments that affect global behavior as they infect local agents is striking.

In "An Adaptive Self-Organizing Protocol for Surveillance and Routing in Sensor Networks," Jorge Simao exploits the diffusion of information around a sensor network to design a decentralized routing protocol that is efficient both in identifying an emergency situation as well as in using energy at each sensor. By using the correlations between sources of information and event types, the algorithm propagates information along a gradient in much the way the De Wolf and Holvoet design patterns describe.

In "Towards the Control of Emergence by the Coordination of Decentralized Agent Activity for the Resource Sharing Problem," Armetta et al. propose an agent communication model, called a negotiation network, in which (situated) agents and contracts are assigned to each other, and stigmergic coordination procedures, where agents can dynamically evaluate and select contracts. The performance of the resulting system, CESNA, is comparable to centralized optimization techniques.

A number of contributions apply or target evolutionary techniques, a main source of inspiration for achieving self-organization. Eiben et al. optimize evolutionary algorithms on the fly in "Reinforcement Learning for Online Control of Evolutionary Algorithms." They use a control loop that involves a reinforcement learning component to capture the abstract structure of the ongoing optimization, that is, the way performance depends on parameters.

In "Greedy Cheating Liars and the Fools Who Believe Them", Arteconi et al. apply an important idea originating from evolutionary computing: tags. In the protocol they present, the tag-based evolutionary system can successfully resist certain types of malicious attacks. All network nodes make local decisions to implement selection, replication and fitness evaluation, and still, through the application of tags, it becomes possible to implicitly reward *groups* that work together in a cooperative way.

In Nowostawski and Purvis' work on "Evolution and Hypercomputing in Global Distributed Evolvable Virtual Machines Environment," the authors exhibit a blend between evolving genetic algorithms and distributed spatial structures. They develop agents on a discrete grid that can cooperate with (or parasitize) each other to evolve solutions to linear polynomial computations. In doing so, they are able to observe the diffusion of knowledge and know-how through the multi-agent system, providing a clear and effective demonstration of the abstract principles of collective intelligence and learning.

In "A Decentralized Car Traffic Control System Simulation Using Local Message Propagation Optimized with a Genetic Algorithm," Kelly and Di Marzo Serugendo describe a decentralized approach to control traffic in an urban environment. The control system is based on emergent phenomena that can be tuned using a few simple parameters. In the article these parameters are set using a genetic algorithm that utilizes a simulation of the control system to evaluate candidate settings.

Finally, we need to mention that this edition is the last in the ESOA series as the workshop will merge into the *International Conference on Self-Adaptation and Self-Organization (SASO)*: a federated conference series covering our field starting in 2007. This signals the growth of interest in engineering self-organization. We believe that this last volume represents interesting contributions in this direction that the readers will find inspiring and useful in their research.

November 2006

Sven Brueckner
Salima Hassas
Márk Jelasity
Daniel Yamins

Organization

Organizing Committee

Sven Brueckner	NewVectors LLC, USA
Salima Hassas	University of Lyon, France
Márk Jelasity	University of Szeged and MTA, Hungary
Daniel Yamins	Harvard University, USA

Program Committee

Sergio Camorlinga	University of Manitoba, Canada
Vincent Cicirello	Drexel University, USA
Giovanna Di Marzo Serugendo	Birkbeck College, London, UK
Maria Gini	University of Minnesota, USA
Marie-Pierre Gleizes	IRIT Toulouse, France
Manfred Hauswirth	Swiss Federal Institute of Technology, Switzerland
Christophe S. Jelger	University of Basel, Switzerland
Anthony Karageorgos	University of Thessaly, Greece
Manolis Koubarakis	Technical University of Crete, Greece
Marco Mamei	University of Modena and Reggio Emilia, Italy
Paul Marrow	BT, UK
Philippe Massonet	CETIC, Belgium
Alberto Montresor	University of Bologna, Italy
Andrea Omicini	University of Bologna, Italy
H. Van Dyke Parunak	NewVectors LLC, USA
Martin Purvis	University of Otago, New Zealand
Mikhail Smirnov	Fraunhofer Fokus, Berlin, Germany

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 4371: K. Inoue, K. Satoh, F. Toni (Eds.), Computational Logic in Multi-Agent Systems. X, 315 pages. 2007.

Vol. 4369: M. Umeda, A. Wolf, O. Bartenstein, U. Geske, D. Seipel, O. Takata (Eds.), Declarative Programming for Knowledge Management. X, 229 pages. 2006.

Vol. 4342: H. de Swart, E. Orłowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments II. X, 373 pages. 2006.

Vol. 4335: S.A. Brueckner, S. Hassas, M. Jelasity, D. Yamins (Eds.), Engineering Self-Organising Systems. XII, 212 pages. 2007.

Vol. 4334: B. Beckert, R. Hähnle, P.H. Schmitt (Eds.), Verification of Object-Oriented Software. XXIX, 658 pages. 2007.

Vol. 4333: U. Reimer, D. Karagiannis (Eds.), Practical Aspects of Knowledge Management. XII, 338 pages. 2006.

Vol. 4327: M. Baldoni, U. Endriss (Eds.), Declarative Agent Languages and Technologies IV. VIII, 257 pages. 2006.

Vol. 4304: A. Sattar, B.-H. Kang (Eds.), AI 2006: Advances in Artificial Intelligence. XXVII, 1303 pages. 2006.

Vol. 4303: A. Hoffmann, B.-H. Kang, D. Richards, S. Tsumoto (Eds.), Advances in Knowledge Acquisition and Management. XI, 259 pages. 2006.

Vol. 4293: A. Gelbukh, C.A. Reyes-Garcia (Eds.), MICAI 2006: Advances in Artificial Intelligence. XXVIII, 1232 pages. 2006.

Vol. 4289: M. Ackermann, B. Berendt, M. Grobelnik, A. Hotho, D. Mladenič, G. Semeraro, M. Spiliopoulou, G. Stumme, V. Svátek, M. van Someren (Eds.), Semantics, Web and Mining. X, 197 pages. 2006.

Vol. 4285: Y. Matsumoto, R.W. Sproat, K.-F. Wong, M. Zhang (Eds.), Computer Processing of Oriental Languages. XVII, 544 pages. 2006.

Vol. 4274: Q. Huo, B. Ma, E.-S. Chng, H. Li (Eds.), Chinese Spoken Language Processing. XXIV, 805 pages. 2006.

Vol. 4265: L. Todorovski, N. Lavrač, K.P. Jantke (Eds.), Discovery Science. XIV, 384 pages. 2006.

Vol. 4264: J.L. Balcázar, P.M. Long, F. Stephan (Eds.), Algorithmic Learning Theory. XIII, 393 pages. 2006.

Vol. 4259: S. Greco, Y. Hata, S. Hirano, M. Inuiguchi, S. Miyamoto, H.S. Nguyen, R. Słowiński (Eds.), Rough Sets and Current Trends in Computing. XXII, 951 pages. 2006.

Vol. 4253: B. Gabrys, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part III. XXXII, 1301 pages. 2006.

Vol. 4252: B. Gabrys, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part II. XXXIII, 1335 pages. 2006.

Vol. 4251: B. Gabrys, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part I. LXVI, 1297 pages. 2006.

Vol. 4248: S. Staab, V. Svátek (Eds.), Managing Knowledge in a World of Networks. XIV, 400 pages. 2006.

Vol. 4246: M. Hermann, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XIII, 588 pages. 2006.

Vol. 4223: L. Wang, L. Jiao, G. Shi, X. Li, J. Liu (Eds.), Fuzzy Systems and Knowledge Discovery. XXVIII, 1335 pages. 2006.

Vol. 4213: J. Fürnkranz, T. Scheffer, M. Spiliopoulou (Eds.), Knowledge Discovery in Databases: PKDD 2006. XXII, 660 pages. 2006.

Vol. 4212: J. Fürnkranz, T. Scheffer, M. Spiliopoulou (Eds.), Machine Learning: ECML 2006. XXIII, 851 pages. 2006.

Vol. 4211: P. Vogt, Y. Sugita, E. Tuci, C.L. Nehaniv (Eds.), Symbol Grounding and Beyond. VIII, 237 pages. 2006.

Vol. 4203: F. Esposito, Z.W. Raś, D. Malerba, G. Semeraro (Eds.), Foundations of Intelligent Systems. XVIII, 767 pages. 2006.

Vol. 4201: Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, E. Tomita (Eds.), Grammatical Inference: Algorithms and Applications. XII, 359 pages. 2006.

Vol. 4200: I.F.C. Smith (Ed.), Intelligent Computing in Engineering and Architecture. XIII, 692 pages. 2006.

Vol. 4198: O. Nasraoui, O. Zaïane, M. Spiliopoulou, B. Mobasher, B. Masand, P.S. Yu (Eds.), Advances in Web Mining and Web Usage Analysis. IX, 177 pages. 2006.

Vol. 4196: K. Fischer, I.J. Timm, E. André, N. Zhong (Eds.), Multiagent System Technologies. X, 185 pages. 2006.

Vol. 4188: P. Sojka, I. Kopeček, K. Pala (Eds.), Text, Speech and Dialogue. XV, 721 pages. 2006.

Vol. 4183: J. Euzenat, J. Domingue (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 291 pages. 2006.

Vol. 4180: M. Kohlhase, OMDoc – An Open Markup Format for Mathematical Documents [version 1.2]. XIX, 428 pages. 2006.

Vol. 4177: R. Marín, E. Onaindía, A. Bugarín, J. Santos (Eds.), Current Topics in Artificial Intelligence. XV, 482 pages. 2006.

Vol. 4160: M. Fisher, W. van der Hoek, B. Konev, A. Lisitsa (Eds.), Logics in Artificial Intelligence. XII, 516 pages. 2006.

Vol. 4155: O. Stock, M. Schaerf (Eds.), Reasoning, Action and Interaction in AI Theories and Systems. XVIII, 343 pages. 2006.

Vol. 4149: M. Klusch, M. Rovatsos, T.R. Payne (Eds.), Cooperative Information Agents X. XII, 477 pages. 2006.

Vol. 4140: J.S. Sichman, H. Coelho, S.O. Rezende (Eds.), Advances in Artificial Intelligence - IBERAMIA-SBIA 2006. XXIII, 635 pages. 2006.

Vol. 4139: T. Salakoski, F. Ginter, S. Pyysalo, T. Pahikkala (Eds.), Advances in Natural Language Processing. XVI, 771 pages. 2006.

Vol. 4133: J. Gratch, M. Young, R. Aylett, D. Ballin, P. Olivier (Eds.), Intelligent Virtual Agents. XIV, 472 pages. 2006.

Vol. 4130: U. Furbach, N. Shankar (Eds.), Automated Reasoning. XV, 680 pages. 2006.

Vol. 4120: J. Calmet, T. Ida, D. Wang (Eds.), Artificial Intelligence and Symbolic Computation. XIII, 269 pages. 2006.

Vol. 4118: Z. Despotovic, S. Joseph, C. Sartori (Eds.), Agents and Peer-to-Peer Computing. XIV, 173 pages. 2006.

Vol. 4114: D.-S. Huang, K. Li, G.W. Irwin (Eds.), Computational Intelligence, Part II. XXVII, 1337 pages. 2006.

Vol. 4108: J.M. Borwein, W.M. Farmer (Eds.), Mathematical Knowledge Management. VIII, 295 pages. 2006.

Vol. 4106: T.R. Roth-Berghofer, M.H. Göker, H.A. Güvenir (Eds.), Advances in Case-Based Reasoning. XIV, 566 pages. 2006.

Vol. 4099: Q. Yang, G. Webb (Eds.), PRICAI 2006: Trends in Artificial Intelligence. XXVIII, 1263 pages. 2006.

Vol. 4095: S. Nolfi, G. Baldassarre, R. Calabretta, J.C.T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, D. Parisi (Eds.), From Animals to Animats 9. XV, 869 pages. 2006.

Vol. 4093: X. Li, O.R. Zaïane, Z. Li (Eds.), Advanced Data Mining and Applications. XXI, 1110 pages. 2006.

Vol. 4092: J. Lang, F. Lin, J. Wang (Eds.), Knowledge Science, Engineering and Management. XV, 664 pages. 2006.

Vol. 4088: Z.-Z. Shi, R. Sadananda (Eds.), Agent Computing and Multi-Agent Systems. XVII, 827 pages. 2006.

Vol. 4087: F. Schwenker, S. Marinai (Eds.), Artificial Neural Networks in Pattern Recognition. IX, 299 pages. 2006.

Vol. 4068: H. Schärfe, P. Hitzler, P. Øhrstrøm (Eds.), Conceptual Structures: Inspiration and Application. XI, 455 pages. 2006.

Vol. 4065: P. Perner (Ed.), Advances in Data Mining. XI, 592 pages. 2006.

Vol. 4062: G.-Y. Wang, J.F. Peters, A. Skowron, Y. Yao (Eds.), Rough Sets and Knowledge Technology. XX, 810 pages. 2006.

Vol. 4049: S. Parsons, N. Maudet, P. Moraitis, I. Rahwan (Eds.), Argumentation in Multi-Agent Systems. XIV, 313 pages. 2006.

Vol. 4048: L. Goble, J.-J.C.. Meyer (Eds.), Deontic Logic and Artificial Normative Systems. X, 273 pages. 2006.

Vol. 4045: D. Barker-Plummer, R. Cox, N. Swoboda (Eds.), Diagrammatic Representation and Inference. XII, 301 pages. 2006.

Vol. 4031: M. Ali, R. Dapoigny (Eds.), Advances in Applied Artificial Intelligence. XXIII, 1353 pages. 2006.

Vol. 4029: L. Rutkowski, R. Tadeusiewicz, L.A. Zadeh, J.M. Zurada (Eds.), Artificial Intelligence and Soft Computing – ICAISC 2006. XXI, 1235 pages. 2006.

Vol. 4027: H.L. Larsen, G. Pasi, D. Ortiz-Arroyo, T. Andreasen, H. Christiansen (Eds.), Flexible Query Answering Systems. XVIII, 714 pages. 2006.

Vol. 4021: E. André, L. Dybkjær, W. Minker, H. Neumann, M. Weber (Eds.), Perception and Interactive Technologies. XI, 217 pages. 2006.

Vol. 4020: A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi (Eds.), RoboCup 2005: Robot Soccer World Cup IX. XVII, 727 pages. 2006.

Vol. 4013: L. Lamontagne, M. Marchand (Eds.), Advances in Artificial Intelligence. XIII, 564 pages. 2006.

Vol. 4012: T. Washio, A. Sakurai, K. Nakajima, H. Takeda, S. Tojo, M. Yokoo (Eds.), New Frontiers in Artificial Intelligence. XIII, 484 pages. 2006.

Vol. 4008: J.C. Augusto, C.D. Nugent (Eds.), Designing Smart Homes. XI, 183 pages. 2006.

Vol. 4005: G. Lugosi, H.U. Simon (Eds.), Learning Theory. XI, 656 pages. 2006.

Vol. 4002: A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.), Finite-State Methods and Natural Language Processing. XIV, 312 pages. 2006.

Vol. 3978: B. Hnich, M. Carlsson, F. Fages, F. Rossi (Eds.), Recent Advances in Constraints. VIII, 179 pages. 2006.

Vol. 3963: O. Dikenelli, M.-P. Gleizes, A. Ricci (Eds.), Engineering Societies in the Agents World VI. XII, 303 pages. 2006.

Vol. 3960: R. Vieira, P. Quaresma, M.d.G.V. Nunes, N.J. Mamede, C. Oliveira, M.C. Dias (Eds.), Computational Processing of the Portuguese Language. XII, 274 pages. 2006.

Vol. 3955: G. Antoniou, G. Potamias, C. Spyropoulos, D. Plexousakis (Eds.), Advances in Artificial Intelligence. XVII, 611 pages. 2006.

Vol. 3949: F.A. Savacı (Ed.), Artificial Intelligence and Neural Networks. IX, 227 pages. 2006.

Vol. 3946: T.R. Roth-Berghofer, S. Schulz, D.B. Leake (Eds.), Modeling and Retrieval of Context. XI, 149 pages. 2006.

Vol. 3944: J. Quiñonero-Candela, I. Dagan, B. Magnini, F. d'Alché-Buc (Eds.), Machine Learning Challenges. XIII, 462 pages. 2006.

¥452.00元

Table of Contents

Overall Design and Fundations

Algorithms and Techniques

Applications

Self-organization and Evolutionary Computing

Hybrid Multi-agent Systems: Integrating Swarming and BDI Agents

H. Van Dyke Parunak[1], Paul Nielsen[2], Sven Brueckner[1], and Rafael Alonso[3]

[1] NewVectors LLC, 3520 Green Court Suite 250, Ann Arbor, MI 48105
{van.parunak, sven.brueckner}@newvectors.net
[2] 28095 Hawberry Rd, Farmington Hills, MI, 48331
paul_eric_nielsen@yahoo.com
[3] SET Corporation, 1005 North Glebe Road, 4th Floor, Arlington, VA 22201
ralonso@setcorp.com

Abstract. The individual agents that interact in a multi-agent system typically exist along a continuum ranging from heavyweight cognitive agents (often of the "BDI" type) to lightweight agents with limited individual processing (digital ants). Most systems use agents from a single position along this spectrum. We have successfully implemented several systems in which agents of very different degrees of internal sophistication interact with one another. Based on this experience, we identify several different ways in which agents of different kinds can be integrated in a single system, and offer observations and lessons from our experiences.

1 Introduction

It has been said that to a small boy with a hammer, every problem looks like a nail. Technologists often seek to press every problem into the mold of their favorite mechanism. In the domain of multi-agent systems, a wide range of agent models have been developed. Some are highly sophisticated cognitive agents that aspire to individual human level intelligence, while other emulate insect-level cognition and exhibit intelligence only at the level of collective behavior.

For several years, we have been exploring different ways of combining heterogeneous models of cognition in a single system. Our experiences show that this approach is not only possible, but that it yields benefits that would be difficult to obtain within a homogeneous framework.

Just as there is no single best agent model, there is no single best way to combine different models. We exhibit a number of different architectures and discuss the situations in which they can be most profitably applied.

Section 2 describes the range of agent models that we hybridize in our work. Section 3 surveys and illustrates the different modes of integration that we have explored. Section 4 offers discussion and conclusion.

2 Alternative Agent Models

Software agents exist across a range of complexities. At some risk of oversimplification, we describe two extremes, and then illustrate some points in the middle. For expository

S. Brueckner et al. (Eds.): ESOA 2006, LNAI 4335, pp. 1 – 14, 2007.

purposes, we call the two extremes "heavyweight" and "lightweight" agents, but these titles are more mnemonic than definitive. Table 1 summarizes the differences between these two extremes.

2.1 Heavyweight Agents

Heavyweight agents are based on the cognitively-inspired AI programs developed in the heyday of artificial intelligence. Each such program then, and each individual agent now, aspires to human-level intelligence. In this domain, an intelligent agent system consists of a system composed of individually intelligent agents. Well-known researchers in the heavyweight tradition include Durfee [7], Jennings [16], Laird [19], Lesser [21], Sycara [34], and Wooldridge [37].

Heavyweight agents have inherited classical AI's emphasis on symbolic representations manipulated with some form of formal logic. The symbols are intended to represent cognitive constructs that are meaningful to people, such as beliefs, desires, and intentions (thus the common rubric "BDI agent" [13, 30]). These constructs, and the logical entailments among them, are elicited by a process of knowledge engineering. This process seeks to capture human intuitions about the appropriate partitioning of a problem domain and self-reflective models of how people reason about the domain.

Because heavyweight agents are built around human-inspired cognitive constructs, they facilitate communication with their users. If an agent has a concept of "danger" that corresponds to a human's concept, there is a good chance that when the agent tells the human that a situation is dangerous, the human will understand.

This benefit comes at a cost. The process of knowledge engineering is intensive and time-consuming. In addition, logical computation is subject to a number of limitations. For example, logical computations are often

Table 1. Two Extreme Types of Software Agents

Class of Agent	Heavyweight	Lightweight
Origins	Artificial Intelligence/Cognitive Science	Artificial Life
Locus of intelligence	Within a single agent	In the interactions among agents
Internal representations and processing	Symbolic	Numeric: polynomials, neural networks, matrix manipulations
Concepts represented	Explicit beliefs, desires, intentions/goals, plans	Sensor states, actuator levels
Development approach	Knowledge engineering	Optimization
Strengths	Intelligible to humans	Computationally efficient Degrades gracefully
Weaknesses	Computationally intractable for large problems Brittle	Difficult to understand

- Intractable, their computational complexity increasing exponentially or worse in the size of the problem, so that problems of realistic size cannot be executed fast enough for the answer to be useful [10],
- Undecidable, so that some questions expressible in the logic simply cannot be answered [11], or
- Brittle, with performance that degrades rapidly (either in accuracy or speed) as one nears the limits of the domain.

2.2 Lightweight Agents

At the other extreme, lightweight agents draw their inspiration from computerized work in ethology, the study of animal behavior. Biologists often construct computer models of animals in order to study their interactions with one another. In many cases (such as ants), no one imagines that the individual agent has anything like human-level intelligence, but the society as a whole can exhibit impressive behavior that might be described as intelligent. In this context, an intelligent agent system is a system of agents that is collectively intelligent. Well-known researchers in this tradition include Bonabeau [3], Brueckner [4], Ferber [8], Ilachinski [15] and Parunak [22].

Lightweight agents do not rely on cognitively meaningful internal representations, for two reasons. First, biologists tend to resist anthropomorphizing the mental behavior of ants and termites. Second, even if it were appropriate to describe their mental operations in the same terms that emerge from human introspection, we would have no way to interrogate the organism about these constructs. What is accessible to the biologist is the entity's environment and its observed actions, so the representation tends to focus on sensory inputs and signals sent to actuators. These are customarily described in analog terms, leading to widespread use of numerical reasoning, usually as some form of matrix algebra (a framework that includes weighted polynomials and neural networks).

Programming such an agent is a matter of identifying the appropriate numerical parameters and setting their values. Knowledge engineering is of little use with a digital insect. Instead, one uses optimization methods such as evolutionary computation to explore the parameter space. We can compare the observed behavior of the agent either with the observed behavior of the domain entity (in a modeling application) or with the desired behavior (in a control application), and use the difference between the two as an objective function [32].

Because their internal processes are essentially numerical, lightweight agents are usually more computationally efficient than heavyweight agents, avoiding issues of tractability and decidability. Their representations extrapolate naturally, avoiding the challenge of brittleness. But they can be difficult for users to understand, for two reasons.

1. The mapping from internal numerical parameters to cognitively meaningful constructs may not be direct. An agent's behavior may be dominated by the fact that the weight between two nodes in a neural network is 0.375, but that knowledge is of little use to a human seeking to validate the agent.

2. Lightweight agents often yield useful behavior, not as individuals, but as a collective. The dynamic of emergence, by which global behavior arises from individual behaviors, is often counter-intuitive [31].

2.3 Intermediate Agents

The two categories of "heavyweight" and "lightweight" agents as described above are extreme cases, and a number of intermediate architectures have been used.

Scripted agents use a state machine to shift from one cognitively meaningful state to another, based on external stimuli. Thus they avoid some of the computational complexity issues associated with richer computational models such as theorem proving.

One mechanism for scripted agents is the Task Frame [5]. Task Frames are used in military simulations such as OneSAF, JSAF, and ModSAF to decompose tasks, organize information, and sequence through the steps of a process. Finite state machines are used to sequence through the task states, however the code within these states is unrestricted.

Sometimes scripted transitions are combined with lighter-weight mechanisms. MANA [20] is a combat model whose agents make decisions based on matrix multiplications, along the line of EINSTein [15]. However, the personality vector that weights the effect of environmental stimuli can be changed discontinuously by certain distinguished events, allowing the agent to change among different behavior patterns depending on environmental stimuli.

Bayes networks [29] combine symbolic and numeric processing, in a manner similar to iconic neural networks. Each node corresponds to a concept or proposition that is meaningful to a human, in a manner consistent with symbolic representations, but the links among the nodes represent conditional probabilities, and processing consists of numeric computations of the propagation of evidence through the network.

Bayes networks have proven most useful at interpretation of activity from observations. For example, seeing a person with wet hair enter the office could either imply that it is raining or they have just taken a shower. However, if we observe several people with wet hair the belief that it is raining would increase.

These intermediate agent architectures combine in a single agent mechanisms from different points in the spectrum *in a single agent.*

3 Integration Modes

In this section, we discuss why it is difficult to integrate agents with different cognitive levels in a single system, and then exhibit a number of different approaches that we have explored. Our list of examples is open-ended, and we invite other researchers to expand it on the basis of their experience.

3.1 Why Is Integration Difficult?

The hybrid systems that we discuss here differ from the "intermediate agents" discussed in Section 2.3. Those examples combined mechanisms from different points in the spectrum *in a single agent*. Here, we explore patterns for combining *distinct agents* that differ in their cognitive mechanisms.

There are three challenges in developing a hybrid system: issues internal to individual agents, issues relating an individual agent to its external environment, and issues dealing with the overall structure of the system.

Internal Issues: Any agent, however simple or complex, is responsible to perceive its environment and take some action based on that perception. Each agent in the system must have the capacity to solve the problem with which it is tasked.

External Issues: The widespread use of heavyweight agents leads naturally to agent interactions that draw on the cognitive constructs that the individual agents are presumed to support. This assumption is the basis of messaging standards such as KQML/KIF and FIPA ACL. Lightweight agents interact through their sensors and actuators rather than through messages with explicit cognitive content. Thus communication between the two types of agents requires special attention.

System Issues: When we provide our small boy with a screwdriver and a wrench in addition to his hammer, we have made his life much more complicated. Now he has to decide which tool to use in which situation. When we permit the use of multiple levels of agent cognition in a single system, we need to think about which kind of agent to use where, and why.

3.2 Swarm as Subroutine

Sometimes a community of lightweight agents can perform a specialized task in support of a heavyweight agent. We used this approach in an experimental extension of TacAir-Soar [17]. The basic TacAir-Soar system is a classic heavyweight architecture based on the Soar architecture for general AI. Geospatial reasoning such as path planning is cumbersome in such an architecture [9], but straightforward for a swarm of lightweight agents. Biological ants use a simple pheromone mechanisms to generate minimal spanning trees that connect their nests with food sources [12], and these mechanisms have been applied successfully in robotic path planning to approach targets while avoiding threats [33].

We merged these two classes of agents by having the Soar agent invoke a swarm of lightweight agents to plan paths. Fig. 1 shows the structure of the implemented system. Communication between the agents is at the cognitive level required by the pilot agent. A wrapper around the path planning swarm handles the translation. In a typical dialog, the pilot reports its current location and its destination, and requests a route. The wrapper instantiates a nest of agents at the current location, a food source at the destination, and turns the swarming agents

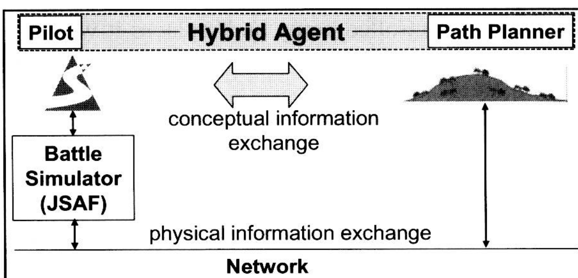

Fig. 1. Swarming path planner as subroutine to Soar AI pilot