



Research Repository UCD

Title	Agent-based coordination for the sensor web
Authors(s)	Tynan, Richard, O'Hare, G. M. P. (Greg M. P.), O'Grady, Michael J.
Publication date	2010-03-26
Publication information	Tynan, Richard, G. M. P. (Greg M. P.) O'Hare, and Michael J. O'Grady. "Agent-Based Coordination for the Sensor Web." ACM, March 26, 2010. https://doi.org/10.1145/1774088.1774512 .
Conference details	Presented at the 25th annual ACM Symposium on Applied Computing (SAC'10), Sierre, Switzerland, March 22-26 2010
Publisher	ACM
Item record/more information	http://hdl.handle.net/10197/1923
Publisher's statement	ACM, 2010. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution.
Publisher's version (DOI)	10.1145/1774088.1774512

Downloaded 2025-12-04 22:51:00

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)



© Some rights reserved. For more information

Agent-Based Coordination for the Sensor Web

Conor Muldoon
CLARITY: The Centre for
Sensor Web Technologies,
School of Computer Science
and Informatics, University
College Dublin,
Belfield, D4, Ireland
conor.muldoon@ucd.ie

Richard Tynan
CLARITY: The Centre for
Sensor Web Technologies,
School of Computer Science
and Informatics, University
College Dublin,
Belfield, D4, Ireland
richard.tynan@ucd.ie

Gregory M. P. O'Hare
CLARITY: The Centre for
Sensor Web Technologies,
School of Computer Science
and Informatics, University
College Dublin,
Belfield, D4, Ireland
gregory.ohare@ucd.ie

Michael J. O'Grady
CLARITY: The Centre for
Sensor Web Technologies,
School of Computer Science
and Informatics, University
College Dublin,
Belfield, D4, Ireland
michael.j.ograde@ucd.ie

ABSTRACT

This paper addresses the problem of coordination within the Sensor Web, where the Sensor Web is defined as an amorphous network of spatially distributed nodes that sense various phenomena in the environment, that are battery powered, and that communicate and coordinate wirelessly. The approach described advocates the use of a multi-agent system, and specifically the use of multi-agent distributed constraint optimisation algorithms. Developing software for low powered sensing devices introduces several problems to be addressed; the most obvious being the limited computational resources available. In this paper we discuss an implementation of ADOPT, a pre-existing algorithm for distributed constraint optimisation, and describe how it has been integrated with a reflective agent platform developed for resource constrained devices, namely Agent Factory Micro Edition (AFME). The usefulness of this work is illustrated through the canonical multi-agent coordination problem, namely graph colouring.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent systems

General Terms

Algorithms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'10 March 22-26, 2010, Sierre, Switzerland.

Copyright 2010 ACM 978-1-60558-638-0/10/03 ...\$10.00.

Keywords

Multi-Agent Systems, Distributed Constraint Optimisation, Wireless Sensor Networks

1. INTRODUCTION

Wireless Sensor Networks (WSNs) have received considerable research interest of late; within a WSN, the goal is to enable resource constrained battery powered devices to autonomously monitor phenomena, such as light, temperature, and sound, and to transmit the sensed data wirelessly to some repository or base station. This paper is concerned with the Sensor Web, whereby collaboration and coordination are key facets. Although all WSNs are collaborative in a loose sense, how to achieve collaborative behaviour optimally and ensure resources are managed effectively remains an open question and is one of the main objectives of the Sensor Web.

In this paper, we advocate the use of ADOPT [8, 9], a pre-existing algorithm for distributed constraint optimisation, to address coordination problems. With distributed constraint optimisation algorithms, the goal is to answer the following question: "How do a set of agents optimise over a set of constraints such that a solution is found with some degree of global quality?". The constraint optimisation problem in general is known to be NP hard. Thus, it is necessary to use approximate approaches for large problem instances. ADOPT provides one such approach, however, the current reference implementation [7] has been developed as a simulator and for standard Java. The implementation discussed in this article has been developed for Java Micro Edition (JME) and has been tested on Sun SPOT motes. One of the main advantages of ADOPT is that it provides a theoretical or analytical bounded error on the approximation. This bounded error can be increased or decreased by varying a threshold value. It provides a principled approach in the trade-off between solution accuracy and resource consumption.

The work described in this paper, builds upon prior re-

search on the development of agent based technologies for the Sensor Web, namely AFME (Agent Factory Micro Edition) [10, 11]. AFME is a reflective agent platform and is loosely based on the BDI model of agency [12]. The BDI model, however, is not really sufficient to truly model collaborative behaviour in a practical sense. It is primarily concerned with the development of agents from an individual perspective. That is, when the developer is writing agent code, the code is written on a per agent basis. Although collaborative behaviour ultimately comes down to the decisions made by individuals, lest we contravene the notions of autonomy and rationality, from a practical perspective, it will often be better to encode common goals in some form of group setting or formalism. Distributed constraint optimisation provides one such approach. In this paper, we discuss how ADOPT is incorporated into the AFME infrastructure as an agent platform service.

This work is concerned with the development of Java-based agents. The ongoing improvement in WSN node technology has lead to the emergence of Java Micro Edition (JME) enabled devices such as the iMote2 and Sun SPOT running the SQUAWK JVM. Such developments pave the way for the porting of existing Java based agent environments to the field of WSNs. The system discussed here is based on the CLDC subset of the JME specifications.

2. AGENT FACTORY MICRO EDITION

In the paper, we advocate the use of computationally reflective or intelligent agents. Computational reflection is one of the most important areas of artificial intelligence; it is a technique that enables a system to maintain meta-information about itself and use this information to change its behaviour or, in other words, to adapt. In the agent community parlance, this meta-information is commonly referred to as an agent's belief set or an agent's model or the world. The behaviour of intelligent agents is typically represented using declarative antecedent-consequence rules, somewhat similar to those of rule-based expert systems.

Intelligent agents symbolically model their environment and manipulate these symbols in order to act. To model intentional agency, an abstraction level is chosen for the symbols such that they represent mental attitudes. Agent oriented programming is a paradigm for directly programming the behaviour of agents using languages whose semantics capture a theory of rational agency. Agents often have a set of beliefs, desires, and intentions and an interpreter that determines how the agents should achieve their goals within the environment.

Agents typically make the closed world assumption and maintain a simple cache of information believed to be true. With the closed world assumption, that which the agent does not have a belief about is taken to be false, or in other words, its negation is taken to be true.

We shall now discuss a minimised footprint embedded reflective agent platform, namely Agent Factory Micro Edition (AFME) [8, 9]. AFME is based on Agent Factory [3], a pre-existing agent platform for desktop environments. AFME expresses an agent's internal state through the mentalistic notions of belief and commitment. In AFME, as with many other intelligent agent platforms, system functionality is delivered through a combination of imperative and declarative code. The imperative functionality is in the form of a set of perceptrors and actuators. Perceptrors generate be-

liefs about the agent's state and its environment. Actuators enable agents to affect their environment; they provide the imperative functionality for primitive or atomic actions. Rules that define the conditions under which commitments are adopted are used to encode an agent's behaviour in an agent programming language. In short, perceptrors generate meta-information (beliefs) about the system state and the environment. Using this information and its internal declarative rule set, the agent will decide on the plans or actions that need to be performed.

Perceptrors, actuators, and rules within AFME are taken to be nondeterministic. By this, we mean that the developer should not program agents on the basis that there is a temporal ordering among perceptrors, among actuators, or among rules in terms of their execution time. In this way, the various components are viewed as being independent or atomic. For instance, it is possible to use an actuator developed for one application, within a different application, without including other actuators that it depends upon to change system state or perform some action before it executes. If such a temporal ordering is required, it is encoded explicitly within an explicit plan.

AFME agents follow a sense-deliberate-act cycle. The agents are executed at periodic intervals. Four functions are performed when an agent is executed. First, the perceptrors are fired and beliefs are updated. Second, the agent's desires are identified through the use of resolution based reasoning. Third, the agent's intentions are identified. Typically, the agent's desires will be its intentions, but in certain circumstances a knapsack procedure will be invoked. Fourth, depending on the nature of the commitments adopted, various actuators are fired.

3. DISTRIBUTED CONSTRAINT OPTIMISATION

With the BDI model of agency, reasoning and resource management is primarily viewed from the point of view of the individual. In this section, the focus is on collaborative behaviour concerning a team of agents. It should be noted, however, that in any collaborative activity, the behaviour and performance of the team ultimately comes down to the decisions made by the individual team members. Nonetheless, in situations where agents do decide to collaborate, it is essential that we have practical algorithms to facilitate collaborative decision making and enable multiple agents to coordinate their behaviour. Distributed constraint optimisation provides one such approach.

A Distributed Constraint Optimization Problem (DCOP) is a constraint optimization problem that is solved in a distributed manner by a group of collaborating agents. The agents share a common goal of choosing values for a set of variables such that the cost of a set of constraints over the variables is either minimized or maximized. A DCOP is defined as a tuple $\langle A, V, \mathcal{D}, f, \alpha, \sigma \rangle$, where:

A is a set of agents;

V is a set of variables, $\{v_1, v_2, \dots, v_{|V|}\}$;

\mathcal{D} is a set of domains, $\{D_1, D_2, \dots, D_{|V|}\}$, where each $D \in \mathcal{D}$ is a finite set containing the values to which its associated variable are assigned;

f is a function $f : \bigcup_{S \in \mathcal{P}(V)} \prod_{v_i \in S} (\{v_i\} \times D_i) \rightarrow \mathbb{N} \cup \{\infty\}$ that maps variable assignments to costs;

α is a function $\alpha : V \rightarrow A$ that maps variables to agents.

$\alpha(v_i) \mapsto a_j$ implies that agent a_j assigns the value of variable v_i ;

σ is an operator that aggregates the individual f costs for the variable assignments. This is accomplished as follows:
 $\sigma(f) \mapsto \sum_{s \in \bigcup_{S \in \mathcal{P}(V)} \prod_{v_i \in S} (\{v_i\} \times D_i)} f(s)$.

The purpose of a DCOP algorithm is to enable each agent to assign values to their variables in order to either minimize or maximize $\sigma(f)$ for a given assignment. In Section 4, we discuss the graph colouring problem, which can be solved using a DCOP algorithm.

3.1 ADOPT Algorithm

In this section, we provide a high level and slightly simplified overview of the ADOPT algorithm, which is used to solve DCOPs. This will be sufficient for our purposes. An in depth discussion of the technical details of the algorithm goes beyond the scope of this article, but the interested reader is directed towards the preexisting literature [9] and the original ADOPT paper [8].

Initially in the ADOPT algorithm there is a preprocessing step. Within this step the constraint graph is converted into a constraint tree. The tree is constructed in such a manner that there are constraints only between a vertex and its ancestors or descendants.

As the algorithm is executing, every vertex of the constraint graph maintains (1) its current value, which is chosen from its domain, and (2) the values of its connected ancestors in the constraint tree, which are referred to as its current context. These values represent a partial solution of the DCOP. The vertices maintain for each value, the lower bounds on the cost of the solution that is consistent with the value and its current context. The lower bounds are initialized with the summation of the costs of the constraints between the connected ancestors. It is possible to determine these costs as the current context is known.

The vertices maintain an upper bound on the cost of the solution that is consistent with their current context. The upper bound is initialised to infinity when the algorithm begins to operate. The lower bound of the current value of a vertex is referred to as its current lower bound. The smallest lower bound of all values is referred to as the best lower bound. The value associated with the best lower bound is referred to as the best value. When the algorithm begins to operate, the initial current value chosen by a vertex is the best value. The vertices also maintain a threshold value, which is initialised to zero. ADOPT maintains the following invariant in relation to the threshold: If it is lower than the best lower bound, it is increased to the best lower bound. If it is larger than the upper bound, it is reduced to the upper bound.

As the algorithm executes, if the current lower bound of a vertex is greater than the threshold, the current value is changed to the best value. Otherwise, the vertex keeps its current value. If the current value is changed, the vertex's connected descendants in the constraint tree are informed of its new value. The descendants perform similar computations; enabling the vertex to decrease its upper bound and increase its current lower bound. When the threshold of the root vertex of the tree is equal to its upper bound, the algorithm terminates.

As mentioned earlier if the current lower bound of the vertex is greater than the threshold, it changes its current value to its best value. There are two possible scenarios

when this occurs:

1. If there are values whose lower bounds are less than the threshold, the best value is taken on and is kept until the lower bound of that value increases above the threshold. This process is repeated until all lower bounds are greater than or equal to the threshold. If this is the case, then the algorithm has reached the second possible scenario. It should be noted that during this first scenario, each value is only taken on once, provided the ancestors do not switch values. The value is kept so long as the lower bound of the value is less than the threshold, even if a different value has a smaller lower bound. This effectively represents a depth-first search.
2. If all lower bounds are greater than or equal to the threshold, the vertex increases the threshold to the best lower bound and then takes on its best value until the lower bound of that value increases. The procedure is then repeated. It should be noted that within this second scenario, the algorithm cannot return to the first scenario, provided the vertex's ancestors do not switch values. In the second scenario, the algorithm always chooses the best value first; this procedure therefore represents a best-first search strategy.

3.2 CLDC Implementation

The current reference implementation of ADOPT [7] has been developed for simulation and within standard Java. In this section, an implementation of ADOPT that has been designed for the Constrained Limited Device Configuration (CLDC) subset of Java Micro Edition (JME) is discussed. CLDC is an extremely limited version of Java. It supports a very small subset of the standard Java classes along with some additional classes (such as those that form the Generic Connection Framework). CLDC is the most widely used version of Java on mobile phones and WSN motes, such as the Sun SPOT and the Sentilla.

There are two reasons why the CLDC implementation of ADOPT was developed. (1) The current reference implementation has been designed for simulation. In this article, we consider the use of the algorithm for a real Sun SPOT WSN node deployment. (2) The current reference implementation has been designed for standard Java rather than CLDC. It therefore could not be used for the vast majority of mobile phones and Java-based WSN motes.

With the development of the CLDC implementation of ADOPT, a number of custom classes were created. The reason for this is that the reference implementation of ADOPT has dependencies on standard Java classes that are not available in CLDC. For instance, it uses the generic linked list class of Java. Creating customised classes provides a means to reduce the footprint and improve the maintainability of the software in that they need only meet the exact requirements of the problem to be addressed rather than provide a generic solution that can be used in a number of different cases. For example, methods return a specific class type rather than a generic object. This removes the need for casting.

In addition to re-implementing the algorithm such that it was compliant with CLDC, it was necessary to create classes that enable the nodes to communicate with each other over the radio channel. This functionality was implemented us-

ing the Sun SPOTs radio gram protocol, which facilitates datagram-based packet exchange.

The design of CLDC implementation of ADOPT, in terms of the object-oriented components of the system, differs significantly from the original implementation. The design has been strongly influence by the ‘Law of Demeter’ (LoD) [6] or the principle of least knowledge. This specifies the coding guideline ‘only talk to your immediate friends, not to strangers’. It requires that a method *M* of an object *O* only invokes the methods of the following objects: *O* itself, the parameters of *M*, and objects created or instantiated within *M*, and *O*’s direct component objects. Developing code that conforms to the law tends improve the maintainability of the software and reduce the footprint by minimising code duplication [13, 2].

The CLDC implementation of ADOPT has been designed to be capable of operating in conjunction with AFME. As mentioned earlier, AFME is a reflective agent platform that has been designed for use with resource constrained devices. The idea is that AFME agents would use ADOPT to facilitate collaborative behaviour in situations where a particular problem has been formalised as a DCOP¹. In such cases, AFME agents use ADOPT as a discrete collaborative action that is performed at a procedural level². AFME agents use ADOPT by incorporating it as a service on the local platform. Although the ADOPT implementation has been designed to be compatible with AFME, it is quite possible to use ADOPT independently. This is useful in situations when there are not enough resources to operate both AFME and ADOPT. For instance, when using very low specification devices.

4. GRAPH COLOURING

ADOPT, and DCOP algorithms in general, are designed to optimise multiple constraints in a distributed manner and can be used to solve the most important multi-agent coordination problem, namely graph colouring. Graph colouring is concerned with the assignment of labels (numbers), which are referred to as colours, to elements of a graph subject to certain constraints. The graph colouring problem is computationally hard. In its simplest form, it is a way of assigning numbers to the nodes of a network such that no two adjacent nodes are assigned the same number; this is commonly referred to as vertex colouring. Similarly, an edge colouring assigns a colour or number to each edge so that no two adjacent edges share the same value.

Graph colouring has several real world applications, such as scheduling and task allocation. Consider the situation in which there are a number of atomic tasks to be performed and each task must be assigned to a specific time slot. Each task takes one unit of time to complete and the tasks can be performed in any order. There are conflicts amongst tasks in that some of the tasks cannot be performed at the same time (this will occur if two or more tasks require unilateral access to a common resource at the same). This is an example of a graph colouring problem where there is a vertex for each task and an edge for conflicting tasks. Algorithms, such

¹In this type of scenario, it is assumed that the DCOP is only part of the problem in that if it were the entire problem there would be no need for the AFME infrastructure.

²ADOPT is implemented at an imperative rather than declarative level.

as ADOPT, enable graph colouring to be performed in a distributed manner.

Consider the situation in which there are a number of moving targets within a WSN environment. The nodes are heterogeneous in the sense that they have differing sensing capabilities along with differing coverage and transmission ranges or areas. In this example, each target represents a sensing source, for instance, some targets will emit light, others temperature, etc. Not all of the nodes will be capable of detecting all of the sources because (1) a source will be out of range or (2) it will not have the appropriate sensing capability on board, for instance, some nodes will have temperature sensors but not acoustic sensors. In this scenario, the nodes periodically check their sensors to see if a target is in range. This effectively determines the colours that the node can take on. For example, a node could potentially be assigned to monitor temperature or sound. In this example, we assume that there is a certain amount of overlap within the coverage areas of the sensors. If two nodes have overlapping coverage areas and can detect the same phenomenon, there will be an edge between the two nodes. Once the graph has been constructed, the nodes are coloured using the ADOPT algorithm. Subsequently, no two adjacent nodes will be assigned the same sensing task and the mobile targets will be reasonably well covered.

Some of the advantages of using ADOPT, along with details of experiments performed using the CLDC implementation, are discussed in Section 5.1.

5. RELATED WORK AND DISCUSSION

The application of agent technologies within WSNs has been increasing over the last number of years. In particular, the underlying agent support frameworks for WSNs. The Mobile Agent Platform for Sun SPOTs (MAPS) is a Java-based framework for wireless sensor networks based on Sun SPOT technology [1]. MAPS uses components that interact through events. Each component offers a set of services to mobile agents that are modeled as multi-plane state machines driven by event condition action rules. These services include message transmission, agent creation, agent cloning, agent migration, timer handling, along with support to access sensor node resources. Agilla [4] is a middleware platform for deploying mobile agents; essentially mobile code. The agent architecture is tailored toward the computational constraints typical of WSN nodes. It allows for multiple agents to exist on a single sensing node and provides methods for the reliable movement of agents between nodes. Sensing platforms provide context to an agent through tuples (a set of predefined descriptors about the node) in a tuple-space. The tuple-space also serves as the communication forum between agents on a node. Mate [5] allows WSN programs to be written in TinyScript, a scripting language that is compiled into executable bytecodes for an application-specific virtual machine. Allowing the virtual machine to be application-specific means that the programs for it can be clear and concise and thus less prone to failure, but this approach reduces compatibility of agents across different platforms. The bytecodes are less like mobile agents, but rather are like intentional viruses. Once a single instance of a bytecode program is introduced to the network, it automatically spreads by controlled flooding until all nodes of the network have a copy of the program.

The work discussed in this paper differs from other agent-

based approaches in a number of ways. This paper detailed the development of the CLDC implementation of the ADOPT algorithm for coordination and constraint optimisation and its integration with AFME, a preexisting platform for resource constrained devices. Distributed constraint optimisation is an NP hard problem. As discussed in Section 4, we use ADOPT to address the graph colouring problem. ADOPT is the first ever distributed, asynchronous, optimal algorithm for DCOP. The algorithm only requires polynomial space at each agent. One of the primary advantages of using ADOPT is that it contains an in built bounded error approximation mechanism. As the algorithm operates, the upper and lower bounds converge towards a solution. The algorithm need not continue operating until the threshold of the root vertex of the tree is equal to its upper bound, but when it is within a specific range. This provides a principled approach in the tradeoff between solution quality and resource usage.

5.1 Experimentation

In order to test the CLDC implementation of ADOPT, we replicated experiments developed for the reference implementation using the data set from [14]. The experiments were performed using both the reference implementation and the CLDC implementation³. As expected, the CLDC implementation provided the same results. From a practical perspective, however, the footprint of the CLDC version will be lower.

When considering the footprint of the software, we must also consider the footprint of other components it requires to execute. The current reference implementation of ADOPT was developed for standard Java. The footprint of CLDC is considerably less than standard Java. This comes at a cost in terms of flexibility, however. For instance, the JVM of CLDC does not facilitate the dynamic loading of foreign objects. The reason for this is that within CLDC code must be preverified. This improves the performance of the JVM. The ADOPT algorithm does not require this functionality, thus this in no way inhibits its execution.

6. CONCLUSION

We discussed how to facilitate coordination amongst intelligent agents for the Sensor Web using a combination of AFME and a new implementation of ADOPT for the JME CLDC environment. The ADOPT algorithm is useful for solving distributed constraint optimisation problems. With this approach, at various stages through execution AFME reflective agents identify situations whereby coordination is necessary. They subsequently use ADOPT to assign nodes to different sensing modalities. Constraint optimisation is general is known to be NP Hard. ADOPT provides a principled bounded error approach that enables the tradeoffs between solution quality and resource consumption to be managed effectively. The current reference implementation of ADOPT is intended for simulation and has been designed for standard Java. The implementation discussed in this article has been developed for CLDC and has been tested on Sun SPOT nodes.

7. ACKNOWLEDGMENTS

³These experiments were conducted on a desktop machine.

This material is based on works supported by the Science Foundation Ireland under Grant No. 07/CE/I1147. Conor Muldoon would like to acknowledge the support of The Irish Research Council for Science, Engineering and Technology (IRCSET) and the People Marie Curie Actions.

8. REFERENCES

- [1] F. Aiello, R. Gravina, A. Guerrieri, and G. Fortino. MAPS: A Mobile Agent Platform for WSNs based on Java Sun Spots. *Proceedings of the third international workshop on Agent Technology for Sensor Networks (ATSN)*, 2009.
- [2] G. Booch. *Object-oriented Analysis and Design, 2nd edition*. Addison Wesley, 1994.
- [3] R. W. Collier, G. M. P. O Hare, T. Lowen, and C. Rooney. Beyond prototyping in the factory of the agents. *3rd Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'03), Lecture Notes in Computer Science (LNCS)*, 2691, 2003.
- [4] C.-L. Fok, G.-C. Roman, and C. Lu. Mobile agent middleware for sensor networks: An application case study. In *Proc. of the 4th Int. Conf. on Information Processing in Sensor Networks (IPSN'05)*, pages 382–387. IEEE, April 2005.
- [5] P. Levis and D. Culler. Maté : a virtual machine for tiny networked sensors. *ASPLOS*, October 2002.
- [6] K. Lieberherr, I. Holland, and A. Riel. Object-oriented programming: An objective sense of style. in *Object Oriented Programming Systems, Languages and Applications Conference, in special issue of SIGPLAN notices*, pages 323–334, 1988.
- [7] P. Modi. USC DCOP repository <http://teamcore.usc.edu/dcop> (2/11/2009).
- [8] P. Modi, W. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2003.
- [9] P. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2006.
- [10] C. Muldoon, G. M. P. O Hare, and J. F. Bradley. Towards Reflective Mobile Agents for Resource Constrained Mobile Devices. In *AAMAS 07: Proceedings of the Sixth International Joint conference on Autonomous Agents and Multiagent Systems*, Honolulu, Hawai'i, May 14-18 2007. ACM.
- [11] C. Muldoon, G. M. P. O Hare, R. W. Collier, and M. J. O Grady. *Multi-Agent Programming: Languages, Platforms and Applications*, chapter Towards Pervasive Intelligence: Reflections on the Evolution of the Agent Factory Framework, pages 187–210. Springer-Verlag Publishers, 2009.
- [12] A. S. Rao and M. P. Georgeff. BDI Agents: from theory to practice. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 312–319, June 1995.
- [13] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object Oriented Modeling and Design*. Prentice Hall, 1991.
- [14] Z. Yin. USC DCOP repository <http://teamcore.usc.edu/dcop> (2/11/2009).