# AgentSpeak for UAV Development: Towards a Technology Bridge

**Marcelo T. Hama**[1],
**Rafael H. Bordini**[1], **Rodrigo S. Allgayer**[2]

[1] Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)

[2]Departamento de Engenharia Elétrica
Universidade Federal do Rio Grande do Sul (UFRGS)

***Abstract.*** *We present the UAVAS framework (Unmanned Aerial Vehicles AgentSpeak), a research related to UAV intelligence/behavior code development in an agent oriented paradigm. The UAVAS framework model is derived from the AgentSpeak runtime implemented in Jason IDE and hardware components from the Mikrokopter project. The research intention is to customize an UAV prototype to support AgentSpeak agents (extended agents from the Jason Architecture), which can reason about courses at action.*

## 1. Introduction

Although the Unmanned aerial vehicle (UAV's) existence cames from some long date, it's in the last decades that great advances has come in scene. Some of these advances are in the field of artificial intelligence and complex behavior development. In this approach the multi agent systems are taking key role in some researches around the world, with case studies in path planning, cohesive/coordinated jobs, free-flights management, and others. To support all of these scientific and technologic developments, the researchers are in demand of specific IDE's, platforms and tools. According to our findings, there's some lack for highly abstract languages, to develop complex AI and behaviors. In specific cases, a good language/framework can be found, like JACK[1], which is a mature agent-oriented framework, but the problematic fact is that it's a commercial tool and cannot be used without bureaucracies/fees. In the academic area, we haven't found any mature open-source framework or language to support the UAV's study with highly abstract programming languages. Guided by this premisse, we are taking efforts in developing a framework system, implemented in Java, and that uses a set of protocoled actions to serve as technology bridge between the UAVs and AgentSpeak agent-oriented language. This protocol is to be used in AgentSpeak codification to call the internal methods of the UAVs through the encoding of actions to methods and signals to percepts. AgentSpeak is an agent oriented language with open-source implementation and academic source [Rao 1996] (thus, with plenty documentations). The experiments are in on-going phase and we are researching technologic compatibilities, specific hardware intrinsic from Mikrokopter[2] UAV's project, speech act semantics and agent oriented theory application. The remainder of this paper is divided in 5 more sections. Section 2 does a brief recapitulation of the UAV's recent researches. Section 3 exposes the field's problems and the proposed approach. Section 4

[1]JACK Intelligent Agents or JACK, is a framework in Java for multi-agent system development.
[2]The Mikrokopter project is an open-source project, with firmware available to public access.

does a brief introduction to Mikrokopter project and UAV technology. Section 5 presents our current architecture model approach. Finally in section 6, we present some conclusions and current work.

## 2. Recent Research

In this last decade, the research in avionic's algorithm took important space in academic view. A recent categoy type of UAV's research is the free flights, where avionics are capable to make decisions about their own paths, in a dynamic way. In [Pechoucek and Sislak 2009], the simulation of free-flights control/manager is a cited issue with analisys of flight control, path planning. The flight plans are made by each avionic by himself, with a contextual intelligence acquired by IA techniques. The system's behavior are studied and collision avoidances algorithms are implemented inside each UAV node, in a specific framework described in [Sislak et al. 2008]. Also, for this same issue, there's an approach in which deconfliction for collision avoidance is treated as an negotiation between the UAV's [Pechoucek et al. 2006]. Others researchers [Ahmadzadeh et al. 2006] prefer a more formalistic view in multi-coordinated path-planning, with mathematical programming development in an attempt to optimize the logistic efficiency. To enable the tests, a testbed research laboratory described in [Michael et al. 2010] has been equiped with motion capture sensors, allowing the treatment of UAV's moves to be controlled by high advanced image-processing software. Using these same system of sensors, there's an approach where the scenario is a 3D city [Keller and Kumar 2008], in which the major objectives are the structures coverage. The multi-agent behavior context is also a very common citation in UAV's research. The analisys of coordination between coalitions [DeJong 2005] and large-scale dynamics [Glinton et al. 2011] are research efforts to establishes a more predictable system in chaotic systems with large numbers of agents. Path planning is another issue greatly mentioned, involving much of mathematical graph theory [Jun and D'Andrea 2002] and heuristics, both for unpredicted environments as for defined ones [Marsh 2005]. One specific research applied to path planning is the issue of visual-field optimization and efficiency [Kim and Crassidis 2010] with main concepts in trigonometry calculus. Yet, others research fields that can be mentioned are autonomy, risks analysis, see-and-avoid systems or networked coordination, being done in a good number of laboratories scattered around the world, of which we can mention the Grasp Lab[3] (Pennsylvania), Santos Lab[4] (Brazil) and Centaurus Technology[5] (Malaysia). As we can see, the UAV artificial intelligence and system behavioral approach applied to UAV are both becoming popular. The special concept that all of these researches have in common is the very specific AI for UAV's. In this study type, there's so much barriers and difficulties. Mainly, it's in the last decades that common people could acquire some of UAV prototypes due to more accessible prices (sometimes, not so) and additionally, the technologies, softwares and informations became more accessible. The UAV's aren't a trivial research area, and there's so much obstacles that researchers can find. We consider the UAV research a recent study field and a hard-to-research area. Some reasons we can give to this difficulty in research are put as follow.

[3] http://www.grasp.upenn.edu/

[4] http://www.santoslab.com.br/index.htm

[5] http://www.centaurustechnology.com/

- Recent Technologies: As viewed early, UAV is an recent technology, wich makes somewhat hard to find documentation, open-source projects or well developed frameworks.
- Costly Hardware: In most cases, boards, softwares and eletronics are hard-to-find and are very specific. Generally, an single UAV prototype shows an overwhelming cost (even that his cost had been dropped in this last decades) and very few researchers can buy it.
- Top-Level Research Field: The UAV's are constructed with a multi-technological knowledge, which involves aeronautical engineering, computer sciences, eletronics, image processings and others. This great number of skills means that it's hard to do significant progress with a little group of researchers or without direct governmental help.
- Specific Areas For Tests: We can't test UAV in open-civil areas due to the dangers that the prototype can offer in some cases. Closed laboratories or open uninhabited areas are generally the best choices to test these devices. Also, for some countries, the researcher needs a permission to use the UAV.
- Software complexity: to program the behaviors of the devices, it's necessary to have a background in specific low-level programming languages, such Micro-C or Assembly. The code that defines the behavioral characteristics of UAV's aren't simple in such languages.

Yet, others issues like airworthiness, certifications, complex control architectures, energy efficient, fail-safe systems, payloads care, navigation systems, security, smart sensors, system integration and regulations aren't trivial problems to solve. A great number of pre-requisites must be achieved to enable the full development of the UAV's technology.

### 2.1. Agent Oriented Paradigm

A topic that's becoming common in the citations of UAV's area is the Agent Oriented Paradigm applied to UAV's. Multi-Agent System (MAS) is a sub-field [Shoham 1993] area of Distributed Artificial Intelligence that focuses on study and research of software that maintains states and autonomies in a universe populated by them. In a MAS, each agent is known as one of its active entities, where a set of these entities form a multi-agent society. In this context, an agent can be viewd as an UAV and so much interesting experiments can be done with this paradigm. Usually, each agent has a set of behavioral capabilities, a set of goals and an autonomy (intelligence) needed to use these behavioral capacities. Decisions on what action to take are determined taking into consideration the changes in the environment and the desire to achieve goals.

## 3. Lack of Tools and Abstraction Gap

In the research that we have done, we haven't found any open-source agent-oriented language that implement frameworks to UAV contexts, with academic perspective and open source tools. We can state that the coding/programming for UAV's development is very complex in low-level languages. In view of these points, our focus with this research is to create an open-source tool to study UAV's, with specific environment and IDE's. AgentSpeak comes in hand because of it's modern agent paradign, consistent semantics and development suport. Also, we can mention that it has a free (open-source) implementation called Jason [Bordini et al. 2007]. Jason is an open-source interpreter for AgentSpeak language developed under the Java technology that is being implemented, extended and studied by academic community in the last decades. Through an integrated API and an IDE with a deliberative multi-agent runtime, multi-agent systems can be implemented, opening up the possibility to research various topics of our daily, including avionics and UAV's. Some of the existing works that uses the Jason/AgentSpeak are the research of virtual environemtns [Ricci et al. 2009] and team simulations[6]. AgentSpeak is an agent oriented language to program multi-agent-systems and complex agent behaviors. A brief description of this language is given below.

### 3.1. AgentSpeak: An Top-Level Language

AgentSpeak is a very popular agent-oriented programming language implemented in this last decade [Bordini and Hubner 2006] as a tool to fulfil the gap between theoretic and practical work in agent's research. An AgentSpeak agent is defined [Alechina et al. 2006] by a set of beliefs that gives the initial state of the agent's belief base, which is a set of logical atomic formulas (first-order), and a set of plans that are its library plans. An AgentSpeak plan has a *head* that consists of a triggering event (specifying the events in which this plan is relevant), and a conjunction of belief literals representing a context. The conjunction of literals in the context must be a logical consequence of current beliefs that the agent performs in the case that the plan is considered appropriate, given the time that the triggering event happens (only applicable plans can be chosen for execution). The plan also has a body, which is a sequence of basic actions or sub-goals that the agent must perform to accomplish the plan. Basic actions represent atomic operations that the agent can do to alter the environment in accordingly to his desires. These actions are also written as atomic formulas, but using a set of stock symbols instead of symbols of predicate. AgentSpeak distinguishes two types of goals: *achievement goals* and *test goals*. Achievement goals are formed by an atomic formulas prefixed with the operator {**!**}, while the test objectives are prefixed with the operator {**?**}. Plans (goals) are triggered by the operators {**+**} (principal plans) or {**-**} (contingency plans).

[6]http://www.multiagentcontest.org/

### 3.2. AgentSpeak Applied to UAV's Development

Thinking about the problems for UAV's researches appointed earlier, our focus is in direction of Multi-Agent Systems concepts applied to UAV's, with main area in direction of methodological development. Ou major objective is to create a system module for UAV behavior programming powered with an Agent-Oriented Language, which could enable more expressiveness in programming and coding with a top-level abstraction. In this field, some works and attempts are in on-going research. Examples that can be cited are the Automated WingMan [Wallis et al. 2002], where a framework in JACK agents has been programmed in a model to work in UAV environmental paradigm, applying the BDI model to such scope. A more generic methodological approach for UAV behavioral design is in collaborations controlled by hierarchical rules [Dargar et al. 2002] to employ the notions of task send/receive, path plannings and work partitioning. One can find the question: why to use AgentSpeak? AgentSpeak is a well developed agent oriented language, with formal operational semantics and a free-open source IDE (Jason). There are other interesting languages like Jack, 3APL or Brahms but, in our point of view, none of these languages match some of AgentSpeak's carachteristics appointed above, such as open-source implementations and good documentations. Below we show the code, a codification in direct low-level C from the Mikrokopter project (described in next section). This snippet adds a new waypoint[7] to be achieved by the prototype.

```
u8 WPList_Append(Waypoint_t* pwp) {
        if(WPNumber < WPLISTLEN) {
                memcpy(&WPList[WPNumber], pwp, sizeof(Waypoint_t));
                WPNumber++;
                NaviData.WaypointNumber = WPNumber;
                return TRUE;
        }
        else return FALSE;
}
```

C source code for adding a new waypoint. Font: `http://gallery.mikrokopter.de/`

In view of the great gap between behavioral coding and C/Assembly coding, our objective is to minimize the abstraction gap and create an IDE for UAV's behavioral development. Methodologically speaking, we propose a way to do write code in AgentSpeak to help the development of UAV systems, assisted by the moderm agent oriented paradigm.

## 4. MikroKopter Project

The MikroKopter is a technological UAV project that deploys a set of propelled prototypes, with differente number of propellers and configurations. In all these configurations, there's three board modules in common, each of these doing a specific task. These boards are:

**Flight-Ctrl:** The *Flight-Ctrl* takes care of the system and environment measurements such as, angular velocity of the axes, acceleration, atmospheric pressure, evaluations, battery voltage, processing and computing angular position and electronic speed controllers. This board is distributed is versions v1.0 (green board) and v1.3 (red board). This board works with a Atmel ATMEGA644 20MHz (generally).

[7] In the Mikrokopter project, the **waypoint** is the definition to a geographical point where the UAV must pass.

**Navi-Ctrl:** This board extends the functionalities of the *Flight-Ctrl* with GPS system and coordinations managements. The features **GoHome** and **HoldPosition** is available with this hardware.

**Brushless-Ctrl:** This board can be seen as an update since it diminishes interface errors, interference risks and empowers the efficiency and performances.

Each board works with a signal protocol which is used by the system to "decode" and manage the programmed instructions. In Figure 1 a four-propelled mikrocopter is shown.

**Figure 1. Mikrocopter configured in a four propellor set. Font:** `http://gallery.mikrokopter.de/main.php/v/Nachbau/z.JPG.html`

## 5. The Proposed Framework

Working with the Mikrokopter technology, our aim is to create a agent runtime inside each prototype, which will serve as platform to the system agents. The system agents will extend the architecture of AgentSpeak/Jason agents, with implementation of a UAV based communication protocol. This protocol is intended to offer signal decodification features intrinsic of the mikrokopter platform extended with speech-acts, environment sensitive functions and percept/act codification. The protocol purpose is to have functionalities of a technologic bridge between AgentSpeak/Jason and an UAV. To run the system, an additional hardware is intended to be inserted in the prototype with the specific task of manage the system framework, exchanging signals with other hardwares via USB and/or serial port. In Figure 2 we show the overall concept of the system model, and in Figure 3 the focused concept of the framework. The body-level represents the UAV structures as it comes, composed by the Flight, Navi and brushless boards. This body will be in listener mode with the other additional hardware. To support the runtime, an operational system with Java interpreter libs (for the Jason agents) is intended to be chosen. Taking a close look in the agent runtime engine, it encapsulates functionalities to encode/decode signals to actions/percepts. The agent platform then manage these decoded signals and convert then to a given protocol action. The current protocol version is described as follow (in informal semantics):

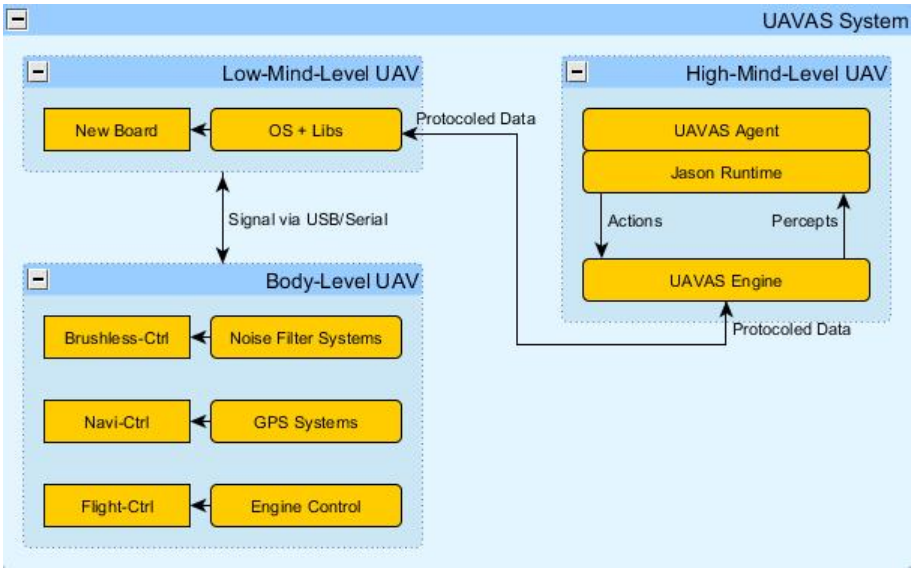


**Figure 2. System concept**

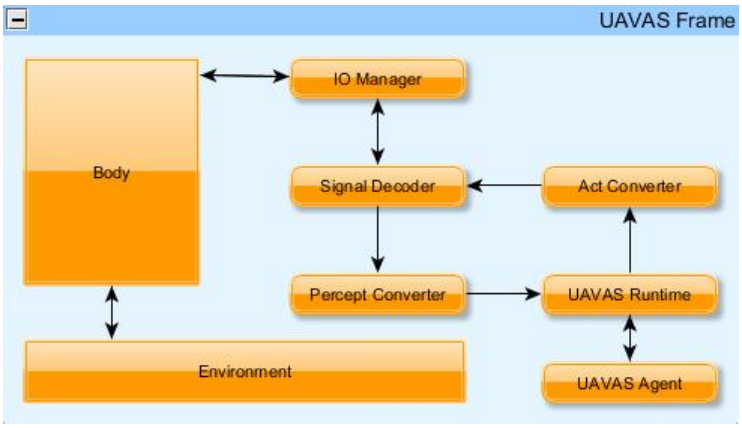


**Figure 3. Framework internal concept**

### 5.1. Bridge Protocol

**Logistic::MoveTo(Point)** → This act add an objective to go to the given location, in the shortest possible path.

**Logistic::DoPatrol(List¡Point¿)** → When this act is performed, for each point in the given list and begining in the index 0, the agent executes the MoveTo act, assuming as argument the indexed points in the list. When the final index is achieved, the index is restarted to 0.

**Logistic::GoHome()** → Go to the pre-defined home point location, using the firmware method.

**Logistic::HoldPosition(Point)** → Holds the position in the given point.

**Illocutionary::Request(ToAgent,literal)** → Sends a request to other agent, where the passed literal is assumed to be false, and the other agent is requested to try to make the literal true.

**Illocutionary::Refuse(ToAgent)** → Sends to other agent a ”deny operation”, related to some received request act.

**Illocutionary::Ask(ToAgent,literal(context))** → This act sends a "query" to the other agent through a literal formulae.

**Illocutionary::Reply(ToAgent,{TRUE|FALSE|UNKNOWN|literal(context)})** → This act sends a reply to a given Asked/Request act. The reply value can be "yes, no, don´t know, or a query", encoded in "TRUE, FALSE, UNKNOWN, *literal(context)*".

**Illocutionary::Inform(ToAgent,Literal)** → Sends to other agent the given literal. The semantic is similar to *replay*, except that *inform* doesn't need a *request*/*ask* to be done before.

**Sense::SenseClimatics()** → Send a query to Flight-Ctrl and retrieve information (in form of percepts) about wind speed, temperatures, accelerations, angles, and similar issues.

**Sense::SenseFuel()** → Send a query to Flight-Ctrl and retrieve information (in form of percepts) about battery voltage.

**Sense::SensePosition()** → Send a query to Navi-Ctrl and retrieve information (in form of percepts) about GPS current position.

**Internal::AddWayPoint(Point)** → Add a waypoint localization, using the firmware method.

**Internal::SetHome(Point)** → Set the given point as the home position.

**Internal::Abort()** → Aborts all intentions with priority equals or lower than 2.

**Internal::Drop()** → Aborts the last intention with priority equals or lower than 2.

These acts aren´t just procedures called by the UAVAS framework but agents intrinsic methods that are called only by UAVAS agents. There are three main groups of acts: logistics, illocutionaries and senses. The logistic acts purposes are to change the GPS positional localization in some way. Illocutionary acts are methods that make possible agents to send messages to each other. Sensing acts are methods to query the environment or agent conditions. If the agent doesn't have any intention, it assumes as default the HoldPosition(LastPassedPoint) intention. The *priorityLevel* cited earlier (last 2 actions) is an integer value, between 1 and 3, that defines the priority of the intention, where 3 value is for the most important, and 1 value to the lesser. Intentions that are in the same level (with exception for the priority-3 ) are concurrent by nature. Only one intention of priority 3 can exist in a given momment (priority-3 are singleton intentions), and if other priority-3 is added, then it´s denied/discarded/ignored by the UAVAS agent. Yet, we didn't define the *priorityLevel* to be assigned to each act, and assume that this protocol is a preliminar version that can be updated in a future moment.

## 6. Future and Current Work

Currently, we are working in three main directions:

- A UAV communication protocol to suport AgentSpeak code. This protocol is a set of instructions described in KQML[8] structures to enable AgentSpeak agents to known specific received tasks.
- UAV's Simulator. To test the agents behaviors developed in AgentSpeak, we are working in a base simulator implementation where the environment is a virtual airspace, and the possible commands are the protocol.
- Android agent platform. The main goal here is to develop a well-suited base where the agents can run and receive/send messages. Some of the concerns are the implementation of a Wi-Fi or BlueTooth mechanism to develop the communication protocol, and the USB/Serial data send/receive to the UAV prototype.

At this moment, we are trying to make this initial framework run into a Beagle Board[9] hardware fixed in the UAV prototype, and above it the Jason runtime with the AgentSpeak agents. The main platform chosed to implement the first version of the proposed framework is the Google Mobile Platform, *Android* because of it's compatibility with the Beagle Board, Java interpretation and low energy consumption. In Figure 4 the proposed framework architecture at glance. The selection of *Android* Platform was made taking in


**Figure 4. Framework architecture at glance**

view our objective to use open-source tools. Although AgentSpeak/Jason is a free IDE, currently there's no plugin or implementaion that make possible to work with others platforms such J2ME (a more suitable platform). For now, the Android platform are helping us in view of it's technologic compatibility, light-weight runtime and open sources. Also, for this first version, we are not taking in view the energy/battery consumption rates. For this first moment, due to lack of time, these research has been left to an future moment. Future work in this research could come from various directions. Some possible directions are in the implementation of plugins for Jason to work with this technologies, the development of a IDE for UAV specific AgentSpeak codes in a consistent way for this scope, efforts in this proposed framework to be technologically more fashionable, suitable and consistent and J2ME implementations.

[8]The Knowledge Query and Manipulation Language, or KQML, is a language and protocol for communication among software agents and knowledge-based systems.

[9]The Beagle Board is a low-power, low-cost single-board computer produced by Texas Instruments in association with Digi-Key.

## References


[Ahmadzadeh et al. 2006] Ahmadzadeh, A., Keller, J., Jadbabaie, A., and Kumar, V. (2006). Multi-uav cooperative surveillance with spatio-temporal specifications. *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 5293–5298.

[Alechina et al. 2006] Alechina, N., Bordini, R. H., Hubner, J. F., Jago, M., and Logan, B. (2006). Automating belief revision for agentspeak. *International Workshop on Declarative Agent Languages and Technologies*, page 16.

[Bordini and Hubner 2006] Bordini, R. and Hubner, J. (2006). Bdi agent programming in agentspeak using jason. *Computational Logic in Multi-Agent Systems*, pages 143 – 164.

[Bordini et al. 2007] Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak Using Jason*. John Wiley and Sons, Ltd.

[Dargar et al. 2002] Dargar, A., Christensen, G., Kamel, A., and Nygard, K. E. (2002). An agent-based framework for uav collaboration. *Electronic Theses and Dissertations*, pages 54 – 59.

[DeJong 2005] DeJong, P. (2005). Coalition formation in multi-agent uav systems. *Electronic Theses and Dissertations*, page 113.

[Glinton et al. 2011] Glinton, R., Scerri, P., and Sycara, K. (2011). An investigation of the vulnerabilities of scale invariant dynamics in large teams. *Autonomous Agents And MultiAgent Systems*, page 8.

[Jun and D'Andrea 2002] Jun, M. and D'Andrea, R. (2002). Path planning for unmanned aerial vehicles in uncertain and adversarial environments. *Cooperative Control: Models, Applications and Algorithms*, pages 95–111.

[Keller and Kumar 2008] Keller, J. and Kumar, V. (2008). Ieee/rsj international conference on intelligent robots and systems. *Robotics and Automation Magazine, IEEE*, pages 2750–2757.

[Kim and Crassidis 2010] Kim, J. and Crassidis, J. L. (2010). Uav path planning for maximum visibility of ground targets in an urban area. *International Conference on Information Fusion*, page 7.

[Marsh 2005] Marsh, L. (2005). Multi-agent uav path planning. *International Congress on Modelling and Simulation Modelling and Simulation Society*, pages 2188–2194.

[Michael et al. 2010] Michael, N., Mellinger, D., Lindsey, Q., and Kumar, V. (2010). The grasp multiple micro uav testbed. *Robotics and Automation Magazine, IEEE*, pages 56–65.

[Pechoucek and Sislak 2009] Pechoucek, M. and Sislak, D. (2009). Agent-based approach to free-flight planning, control, and simulation. *IEEE Intelligent Systems*, pages 14–17.

[Pechoucek et al. 2006] Pechoucek, M., Sislak, D., Pavlícek, D., and Uller, M. (2006). Autonomous agents for air-traffic deconfliction. *International Conference on Autonomous Agents*, pages 1498–1505.

[Rao 1996] Rao, A. S. (1996). Agentspeak(l): Bdi agents speak out in a logical computable language. *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 42 – 55.

[Ricci et al. 2009] Ricci, A., Piunti, M., Viroli, M., and Omicini, A. (2009). Environment programming in cartago. *Multi-Agent Programming*, page 259.

[Shoham 1993] Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, pages 51 – 92.

[Sislak et al. 2008] Sislak, D., Pechoucek, M., Volf, P., Pavlicek, D., Samek, J., Marik, V., and Losiewicz, P. (2008). Agentfly: Towards multi-agent technology in free flight air traffic control. *Whitestein Series in Software Agent Technologies and Autonomic Computing*, pages 73–96.

[Wallis et al. 2002] Wallis, P., Rönnquist, R., Jarvis, D., and Lucas, A. (2002). The automated wingman - using jack intelligent agents for unmanned autonomous. *Aerospace Conference Proceedings*, pages 2615 – 2622.