# Agent Architecture Modelling at the Knowledge Level

Carlos A. Iglesias
Departamento Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación, Ciudad Universitaria s/n, 28040 Madrid, Spain
`cif@gsi.dit.upm.es`

And

Andres Escobero
Sun Microsystems Ibérica S.A.
Torre Picasso - Planta 27, 28020 Madrid, Spain
`Andres.Escobero@Spain.Sun.COM`

## Abstract

The definition of an agent architecture at the knowledge level makes emphasis on the knowledge role played the data interchanged between the agent components and makes explicit this data interchange. This makes easi the reuse of these knowledge structures independently of the implementation.

This article defines a generic task model of an agent architecture and refines some of these tasks using inferen diagrams.

Finally, a operationalisation of this conceptual model using the rule-oriented language Jess [5] is shown.

**Keywords:** Agent Oriented Software Engineering, knowledge level, agent architecture, knowledge engineering

## 1. Introduction

This article deals with knowledge modelling of a generic agent architecture. The purpose of this work is to app a principled approach to the definition of a generic conceptual agent architecture.

This work is part of a general framework, the agent-oriented methodology *MAS-CommonKADS* [7], [9]. particular, this article deals with the *Expertise Model* of the methodology.

The remainder of the paper is organised as follows: Section 2 presents a short introduction to the Knowledge Moc of CommonKADS, that is used for defining a generic agent architecture at the knowledge level. Section 3 presen a task decomposition model of generic agent architecture. Sections 4, 5 and 6 presents a constructive approa to define inference structures for the previously presented task model, showing how this framework is general a extensible for defining agent architectures. Section 7 describes how this conceptual model can be operationalis using the rule oriented language Jess.

## 2. Introduction to Knowledge Modelling with CommonKADS

The CommonKADS knowledge model [16][1] is used for modelling the reasoning capabilities of the agents to car out their tasks and achieve their goals. Usually, several instances of the expertise model should be develope modelling inferences on the domain (i.e. how to identify a situation); modelling the reasoning of the agent (i problem solving methods to achieve a task, character of the agent, etc.) and modelling the inferences of t environment (how an agent can interpret the event it receives from other agents or from the world).

The knowledge model has three parts: *domain knowledge*, *inference knowledge* and *task knowledge*.
*Domain knowledge* represents the static domain-specific knowledge of the problem, modelled as a set of concep properties, expressions and relationships, similar to the object model of UML (Unified Modelling Language) [10]
*Inference knowledge* represents the basic inference steps that we want to make using the domain knowledge. It represented with *inference diagrams* where a functional decomposition is carried out. The basic predefined knowled functions are called *inferences* and are shown as ellipses. The inputs and outputs of these inferences are call *knowledge roles*, that can be static (not modified by the inferences) or dynamic, and are shown as squared boxes
*Task knowledge* represents how to achieve a goal, and the decomposition of this goal into sub-tasks, being t inferences the leaves of this decomposition. For defining this decomposition, tasks are described and problem solvi methods (PSMs) are defined. The PSMs define how to decompose a task into sub-tasks (or goals).

## 3. Agent Architecture Skeleton

The purpose of this analysis, based on Interrap conceptual agent model [15], is to define a framework for studyi systematically the components of a generic agent architecture and their relationships.

[1]Previously defined as *expertise model* [18], [12] in CommonKADS

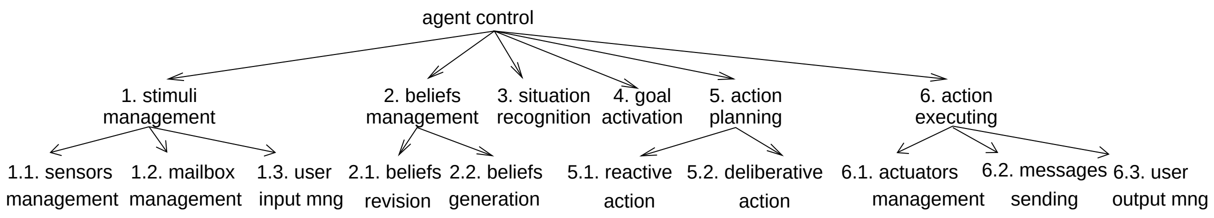

Fig. 1. Task model of a general agent architecture

The main tasks of this task analysis [3] is shown in figure 1 and are described below.
*Stimuli management* the agent receives the stimuli through its sensors, mailbox and user input.
*Beliefs management* deals with beliefs generation and updating taking as an input the perceived stimuli.
*Situation recognition* the agent extract structured situation from the unstructured agent beliefs. This situati
recognition allows the identification of the need to start an activity.
*Goal activation* determination of which goals are relevant to the identified situation.
*Action planning* determination of which actions must be carried out to fulfil the identified goals and the order
these goals.
*Action performing* this task consists of executing the planned action, and can be programmed.
The main domain concepts have been captured above help guiding the knowledge acquisition process. T
following domains can be identified:
*Own agent* reflexive knowledge about the agent itself. This knowledge allows the agent to reason about its abiliti
and its reasoning process.
*Rest of agents* The agent should know what agents it knows, their relevant characteristics and possible inferences
*Environment* In order to interpret the sensor data, the agent must know what are the possible objects of t
environment and their characteristics.
*Application* The agent must know the relevant concepts of the application domain.

## 4. Basic Reactive Agent Architecture at the Knowledge Level

In order to follow a constructive approach, we will start by studying a simple reactive agent that decides wh
action to do (task 6) depending on the observables (task 1), as shown in Figure 2. Once the inference structure h
been built, the domain model can be completed. In this case, the observables of the domain and the allowed actio
should be identified, and the rules to match these observables into actions.

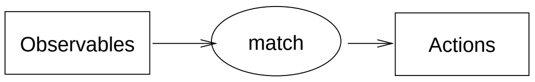

Fig. 2. Inference structure of a basic reactive agent

For example, for the Robocup domain [13], the observables from the environment are the *Ball*, the *Goals*, t
*Corner*, etc., and the actions are *Turn*, *Dash*, etc. The characterisation of these concepts makes up the age
ontology. In order to identify the *reactive situations*, the so-called *reactive cases* of the UER technique [8] can
used.
This inference structure can be easily extended for considering the transformation of *observables* into *beli*
(Figure 3) or considering a basic self-conscience of the agent (Figure 4).

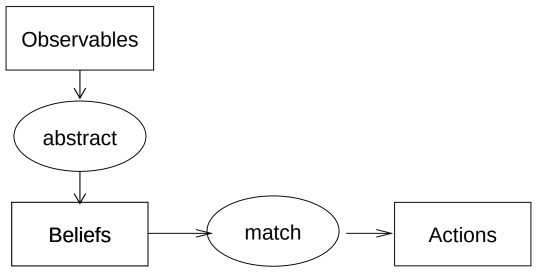

Fig. 3. Inference structure of a basic reactive agent with beliefs

The transformation of *observables* into *beliefs* can be trivial if we consider *symbolic sensors*, but can be furth
refined as a complex function with situation recognition as shown in Figure 5. This knowledge task of situati

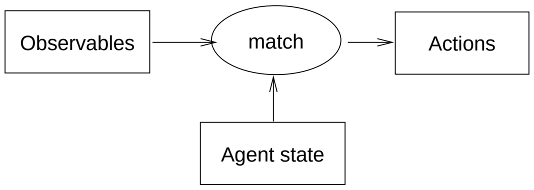

Fig. 4. Inference structure of a basic reactive agent with basic self-conscience

recognition has been modelled at the knowledge level in [2]. The *agent model* is a set of relevant agent properti
These agent properties depend on the application domain. For example, for the Robocup domain, some of the
relevant properties are the agent position and its stamina.

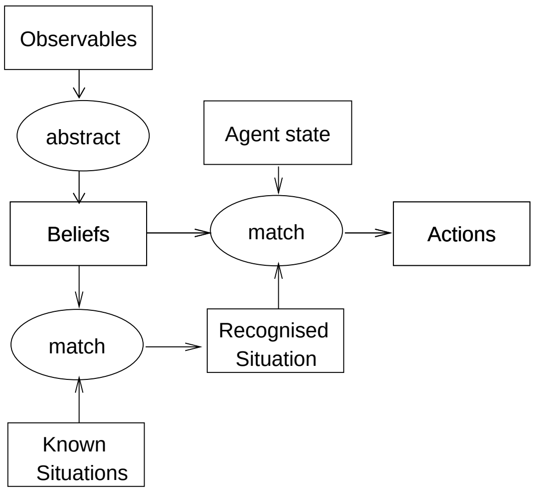

Fig. 5. Inference structure of a basic reactive agent with beliefs and situation recognition

## 5. Basic Deliberative Agent Architecture at the Knowledge Level

While the previous section dealt with reactive agents, this section will consider how the activation of goals a
the planification task of the generic task model (section 3).

### Basic BDI agent architecture

In order to illustrate the use of the knowledge model for specifying agent architectures at the knowledge lev
the inference process of the well-known BDI (Belief-Desire-Intention) architecture [19] is shown in Figure 6. In th
architecture, an example of how to perform the general tasks of *Beliefs Management* and *Goal Activation* (section
is shown.

A simpler example of goal activation without considering *desires* is shown in Figure 7. In this example, the age
communication abilities have been modelled considering the received messages from other agents. The beliefs of t
agent can be of the agent itself, its environment other agents or application domain concepts.

### Action Planning Task

The task *action planning* (section 3) can be considered a KADS basic inference [17] (see Figure 8).

This knowledge task has also been decomposed through Problem Solving Methods in [1]. A practical example
a simple planner [11] is shown in Figure 9.

Inside the JAEN project [14], taking as a generic model MAS-CommonKADS [7], the planning process is defin
through PSMs (Problem Solving Methods) as shown in Figure 10.

The *PSMs* define the way to decompose a *goal* into subtasks. Two general types of PSMs are defined: *autonomo*
*PSMs* and *cooperative PSMs*. While the resulting subtasks are executed by the agent itself using an autonomous PS
some of these subtasks can be carried out in cooperation with other agents using cooperative PSMs. For examp
given a goal such as *Finding the best price of an article*, depending on its state, an agent can use an autonomo
PSM such as *Go-to-all-the-shops-and-compare* or a cooperative PSM such as *Subcontract-task-to-other-agent*.

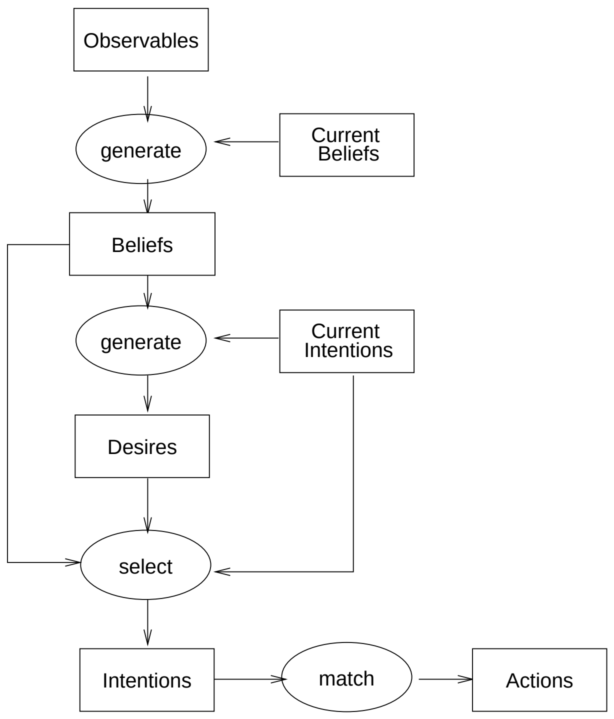

Fig. 6. Inference structure of a general bdi agent architecture

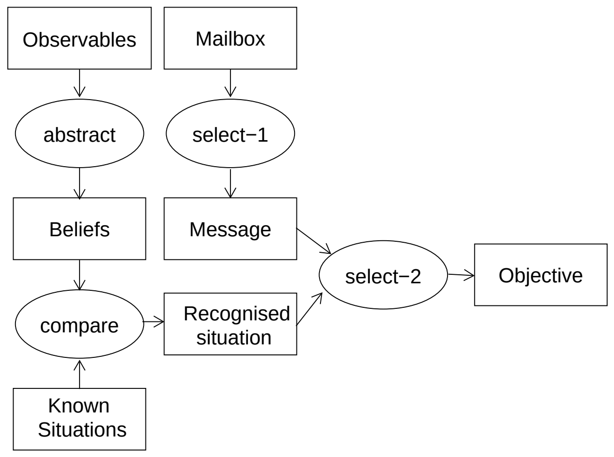

Fig. 7. Simple Inference structure for goal activation

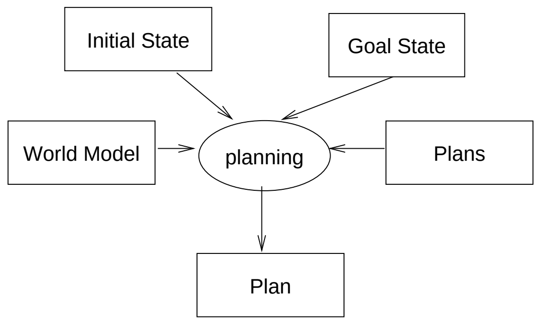

Fig. 8. Inference structure of a planning function [17]

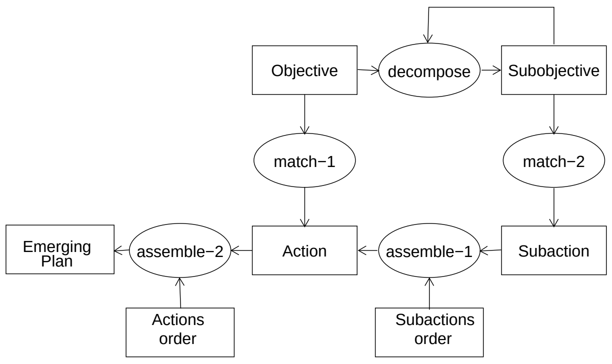

Fig. 9. Inference structure of a basic planner [11]

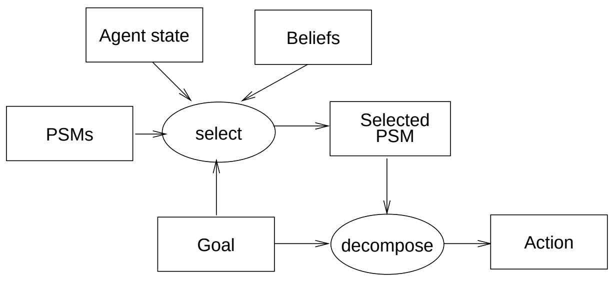

Fig. 10. Inference structure for action planning based on PSMs

## 6. Cooperative Agents

In the previous section, cooperation has been introduced through *cooperative PSMs*.

This section shows how two simple functions for handling the mailbox (task 1.2, sec. 3) can be defined at t
knowledge level.

When an agent wants to request some service from other agent, the agent should determine which protocol to u
from the known protocols, as shown in Figure 11.

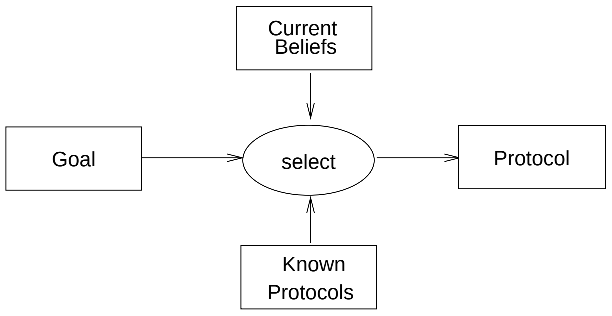

Fig. 11. Inference structure for selecting a protocol

The second function is how to decide if a service request is attended or not. As shown in Figure 12, it is need
to characterise the service request, the service policy and, as a result, a *commitment* is done.

## 7. Operationalisation of the architecture with Jess

The previously presented generic agent model has been operationalised using Jess as target language [14], [6].

As a simple example, initial knowledge about the known ontologies, protocols and knowledge representati
languages is shown in Figure 13.

Goals and PSMs can also be defined using Jess templates, and how a PSM decomposes a goal into subtas
or subgoals, as shown in Figure 14. Here, one goal (*FindArticle*)is defined and two available PSMs for this go
The PSM *AutonomousPSM* decomposes the goal into two tasks, *go-shops* and *compare*, while the cooperative PS
*CoopPSM* decomposes it into one task, *ask-help-broker*.

The final tasks which do not need further decomposition are operationalised as rules. As actions, communicati
acts of FIPA [4] can be used in a natural way, as shown in Figure 15, where an agent sends a request to anoth

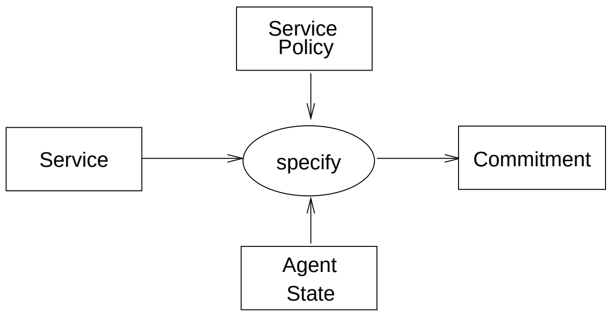

Fig. 12. Inference structure for attending a service request

```
(deffacts ExampleInializations
    (known-ontologies (ontologies
          (create$ fipa-agent-management
          fipa-acl DefaultOntology)))
    (known-protocols (protocols
          (create$ fipa-request)))
    (known-languages (languages
          (create$ JESS))
)
```

Fig. 13. Example of operationalisation of knowledge about agent capacities

agent (whose name is *BrokerAgent@shop.com*) asking the service *FindShops*.

## 8. Conclusions and Future Work

In the paper, we have tried to illustrate how an agent architecture can be defined at the knowledge level in easy way.

The knowledge description of agent architectures makes easier their acquisition and operationalisation. Know edge modelling determines the relationships between the different agent components and the required relationshi between them, i.e. temporal relationships between goals, required knowledge for selecting emergent goals, etc. T definition of agent components at the knowledge level makes easier their reuse.

In addition, the knowledge description of agent architectures makes explicit the role the domain concepts play the reasoning process and provides a good starting point for achieving a reflexive behaviour.

Finally, this theoretical work has been operationalised in an extension of the rule-based language Jess [5].

Future work will focus on extending the presented framework for providing a library of agent architecture co ponents defined at the knowledge level and operationalised. This work is complementary of our current work building agent-oriented CASE tools.

## Acknowledgements

This research is funded in part by the Spanish Government under the CICYT project JAEN TEL99-0925.

```
(goal (name FindArticle))
(PSM (name AutonomousPSM)(goal FindArticle)
     (tasks (create$ go-shops compare))
)
(PSM (name CoopPSM)(goal FindArticle)
     (tasks (create$ ask-help-broker))
)
```

Fig. 14. Example of operationalisation of PSMs and task decomposition

```
(defrule CollaborativeTask
  (task (name ask-help-broker)
        (goalid ?goalid)(input ?input))
=>
  (request :receiver BrokerAgent@shop.com
           :protocol fipa-request
           :language JESS
           :reply-with wanted
           :content "(service
                      (name FindShops)
                      (input " ?input "))"
           :goal-related ?goalid
   )
)
```

Fig. 15. Example of operationalisation of a Task

## 9. References

[1] V. R. Benjamins, Leliane Nunes de Barros, and Valente Andre. Constructing planners through probler solving methods. In B. Gaines and M. Musen, editors, *Proceedings of the 10th Banff Knowledge Acquisition f Knowledge-Based Systems Workshop*, volume 1, pages 14–1/20, Banff, Canada, November 1996. KAW.

[2] Dolores Cañamero. A Knowledge-Level Approach to Plan Recognition. In *Proceedings of the IJCAI'95 Worksh on Plan Recognition*, Montreal, Canada, August 1995.

[3] C. Duursma. Task model defintion and task analysis process. Technical Report Technical report KAD II/M5/VUB/TR/004/1.1b ESPRIT Project P5248, Free University Brussels, 1993.

[4] FIPA. Foundation for Intelligent Physical Agents. Agent Communication Language. FIPA Spec 2. Technic report, FIPA. Foundation for Intelligent Physical Agents, 1999.

[5] Ernest J. Friedman-Hill. *Jess, The Java Expert System Shell.* Distributed Computing Systems, Sandia Nation Laboratiories, Livermore, CA, version 4.1 edition, June 1998.

[6] Pablo Haya Coll. Diseño de Métodos de Coordinación entre Agentes dentro del Desarrollo de un Sisten Personal de Información. Master's thesis, E.T.S.I. de Telecomunicación. Universidad Politécnica de Madri September 1999.

[7] Carlos A. Iglesias. *Definition of a Methodology for the Development of Multi-Agent Systems.* PhD thes Departamento de Publicaciones, E.T.S.I. Telecomunicación, Universidad Politécnica de Madrid, February 199 In Spanish.

[8] Carlos A. Iglesias and Mercedes Garijo. UER Technique: Conceptualisation for Agent Oriented Developmer In Nagib Callaos and Michel Torres, editors, *Proceedings of the 3rd World Multiconference on Systemics, C bernetics and Informatics (SCI'99) and 5th International Conference on Information Systems Analysis a Synthesis (ISAS'99)*, volume 5, pages 535–540, Orlando (USA), August 1999.

[9] Carlos A. Iglesias, Mercedes Garijo, José C. González, and Juan R. Velasco. Analysis and design of multi gent systems using MAS-CommonKADS. In M. Wooldridge, M. Singh, and A. Rao, editors, INTELLIGEN AGENTS IV: Agent Theories, Architectures, and Languages, volume 1365, pages 313–329. Springer-Verla 1998. (A reduced version of this paper has been published in *AAAI'97 Workshop on Agent Theories, Archite tures and Languages.*

[10] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process.* Addisc Wesley: Reading, MA, 1999.

[11] J. K. Kingston, N. Shadbolt, and A. Tate. CommonKADS models for knowledge based planning. In *Proceedin of AAAI-96*, Portland, Oregon, August 1996. AAAI.

[12] John Kingston. Building a KBS for health and safety assessment. In *Applications and Innovations in Expe Systems IV, Proceedings of BCS Expert Systems '96*, pages 16–18, Cambridge, December 1996. SBES Public tions. Also published as technical report: AIAI-TR-202, Artificial Intelligence Applications Institute, Universi of Edinburgh.

[13] Álvaro Martínez Reol. Desarrollo de un sistema multiagente integrando jess y java para la robocup. Master thesis, E.T.S.I. de Telecomunicación. Universidad de Valladolid, May 2000.

[14] Juan Luis Mulas Platero. Desarrollo de una arquitectura de agente inteligente multiservicio con la plataforr javamast. Master's thesis, E.T.S.I. de Telecomunicación. Universidad de Valladolid, February 1999.

[15] Jörg P. Müller. *An Architecture for Dynamically Interacting Agents.* PhD thesis, Germain AI Research Cent
(DFKI GmbH), Saarbrücken, 1996.

[16] A. Th. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, N. R. Shadbolt, W. Van de Velde, a
B. J. Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology.* MIT Press, 19

[17] A. Valente. Planning models for the commonkads library. ESPRIT Project P5248 KADS-II KAD
II/M2.3/UvA/56/1.0, University of Amsterdam, 1993.

[18] B. J. Wielinga, W. van de Velde, A. Th. Schreiber, and H. Akkermans. Expertise model definition doc
ment. deliverable DM.2a, ESPRIT Project P-5248 /KADS-II/M2/UvA/026/1.1, University of Amsterda
Free University of Brussels and Netherlands Energy Research Centre ECN, May 1993.

[19] Michael Wooldridge. Intelligent agents. In Gerhard Weiss, editor, *Multiagent Systems. A Modern Approach
Distributed Artificial Intelligence.* MIT Press, 1999.