Ingeniería de Software Orientada a Agentes

Jason en el CIIA

Dr. Alejandro Guerra-Hernández

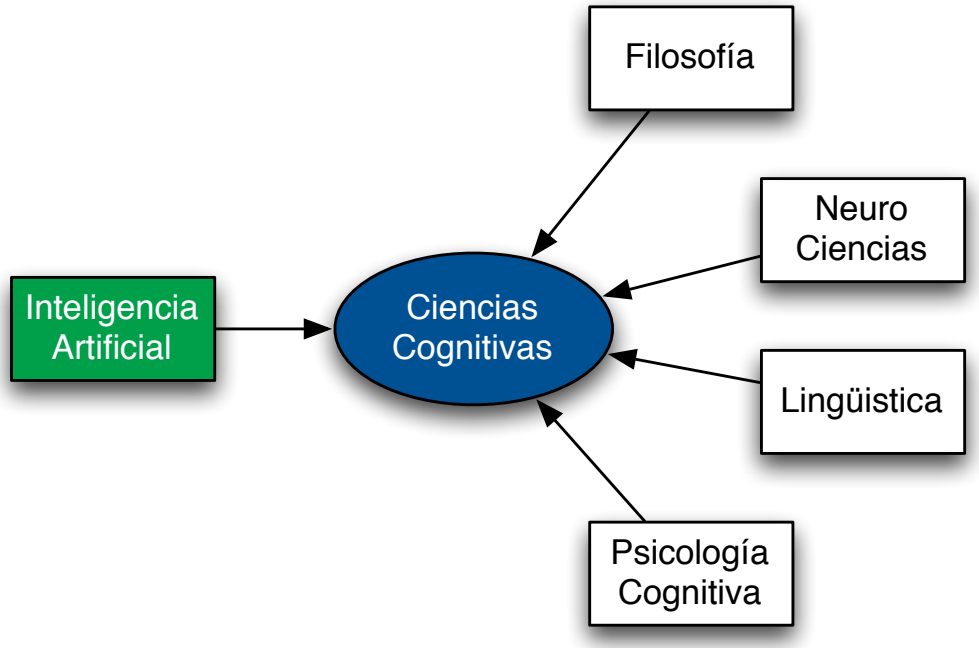
Universidad Veracruzana
Centro de Investigación en Inteligencia Artificial
CA Investigación y Aplicaciones de la IA
Sebastián Camacho No. 5, Xalapa, Ver., México 91000

mailto:aguerra@uv.mx
http://www.uv.mx/personal/aguerra

Escuela de Verano en Ingeniería de Software 2019

La IA como Ciencia e Ingeniería

- Las entidades inteligentes y su comportamiento.
- ... pero su meta no tiene que ver únicamente con la comprensión de tales entidades, sino con su construcción.



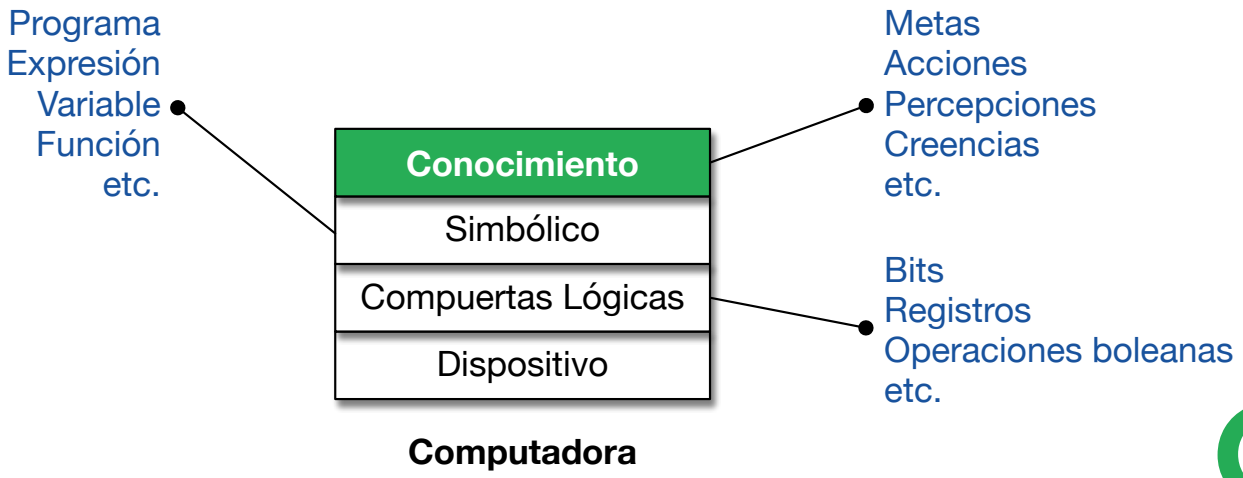
Adaptado de Varela [16].

Inteligencia como computación

- Hipótesis: Un Sistema Simbólico Físico tiene los medios necesarios y suficientes para un comportamiento inteligente general –Newell y Simon [12].

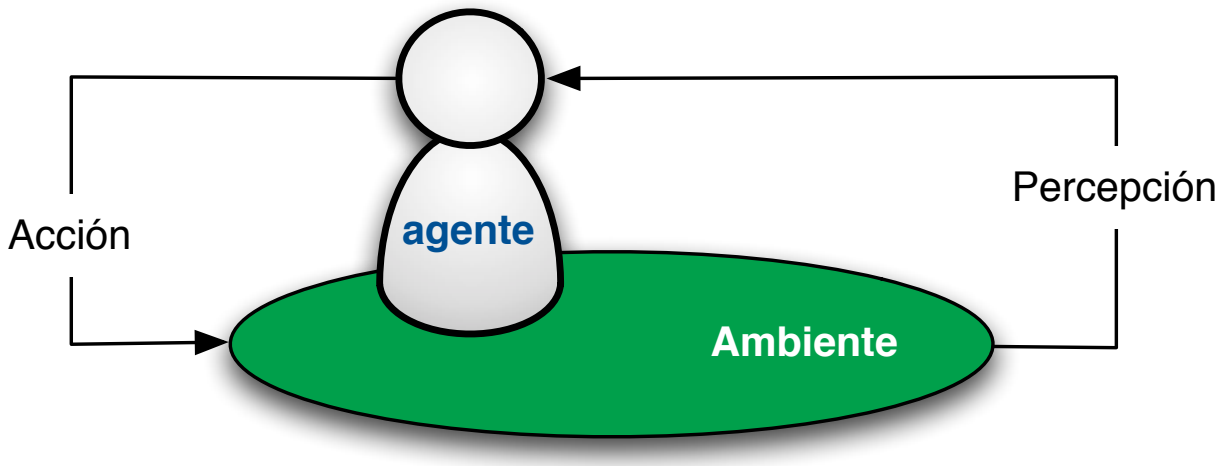
$$\text{Símbolos} + \text{Estructuras Simbólicas} + \text{Procesos}$$

- Descripción: El nivel del conocimiento –Newell [11]



Agentes

- Un agente es un sistema computacional capaz de actuar de manera autónoma para satisfacer sus objetivos y metas, mientras se encuentra situado persistentemente en su medio ambiente.



- Principio de racionalidad: Si un agente sabe que una de sus acciones conduce a satisfacer una de sus metas, elegirá esa acción.

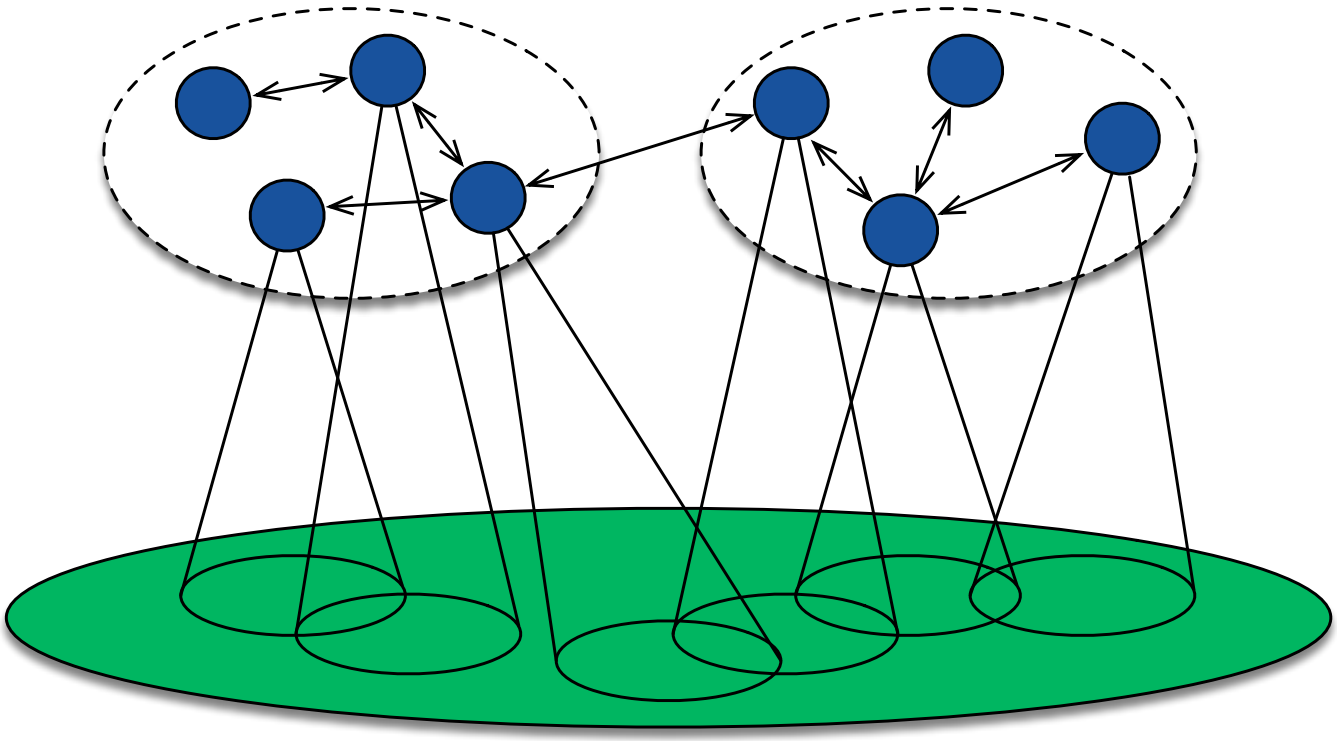
Tipos de Agentes

- Selección de Acción: ¿Cómo hacer lo correcto?

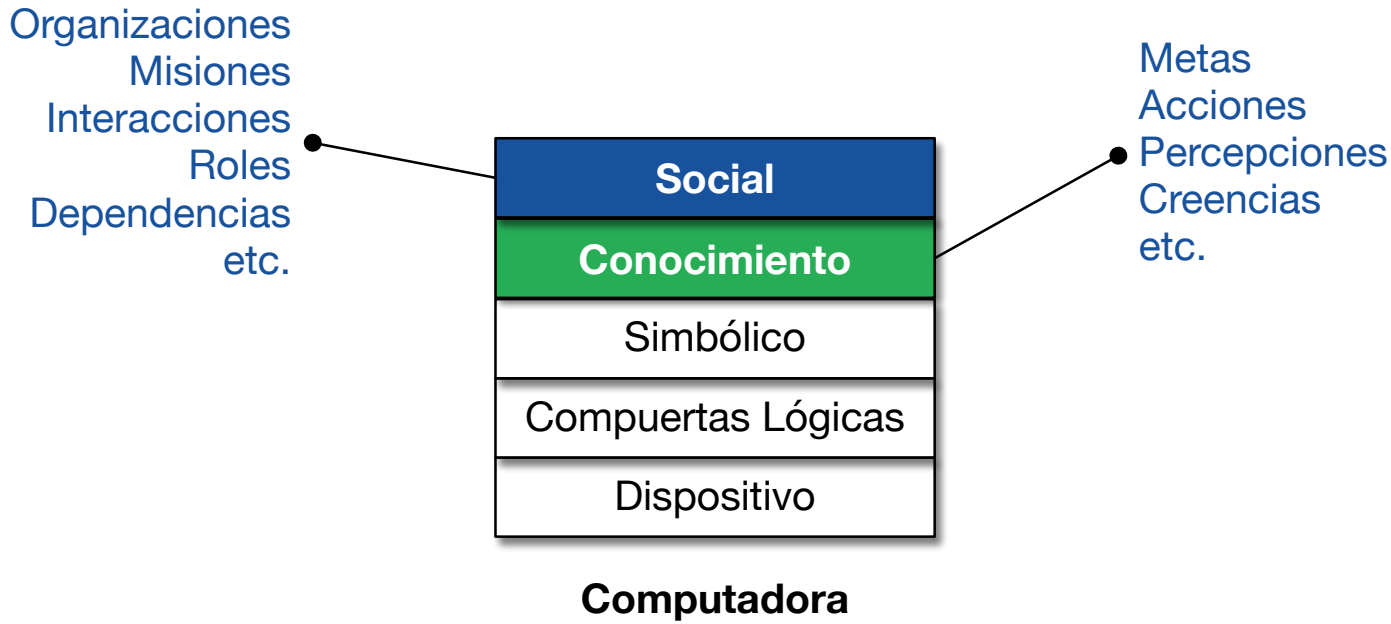
Arquitecturas		Lenguajes de Programación	
Subsumsión	Maquinas Touring	GOLOG	PRS
Pengi	InteRRap	conGOLOG	dMARS
Automatas situados	3T	MetateM	Jam
Redes de comportamiento	Stanley	concurrent MetateM	3APL
etc.			**Jason**

Puramente reactivos — Híbridos — Deliberativos — Intencionales

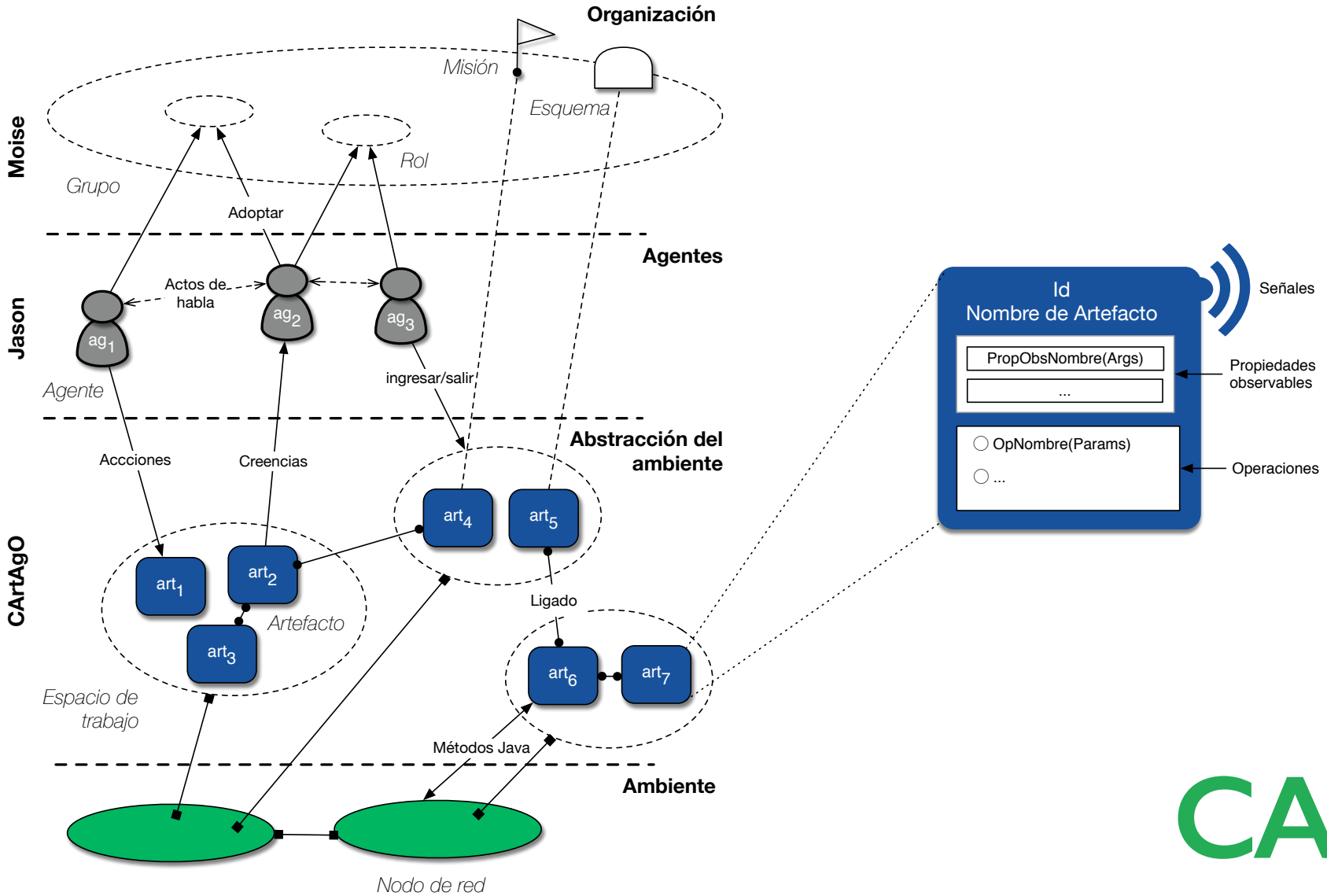
Interacciones



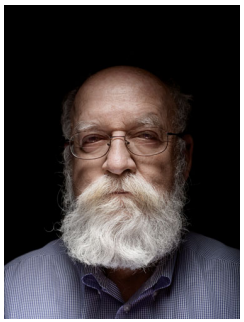
Nivel Social de la Computación



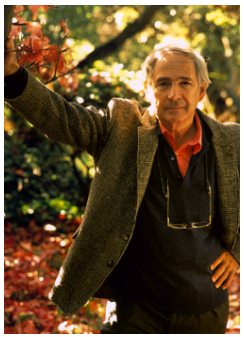
Plataforma JaCaMo



Fundamentos filosóficos BDI



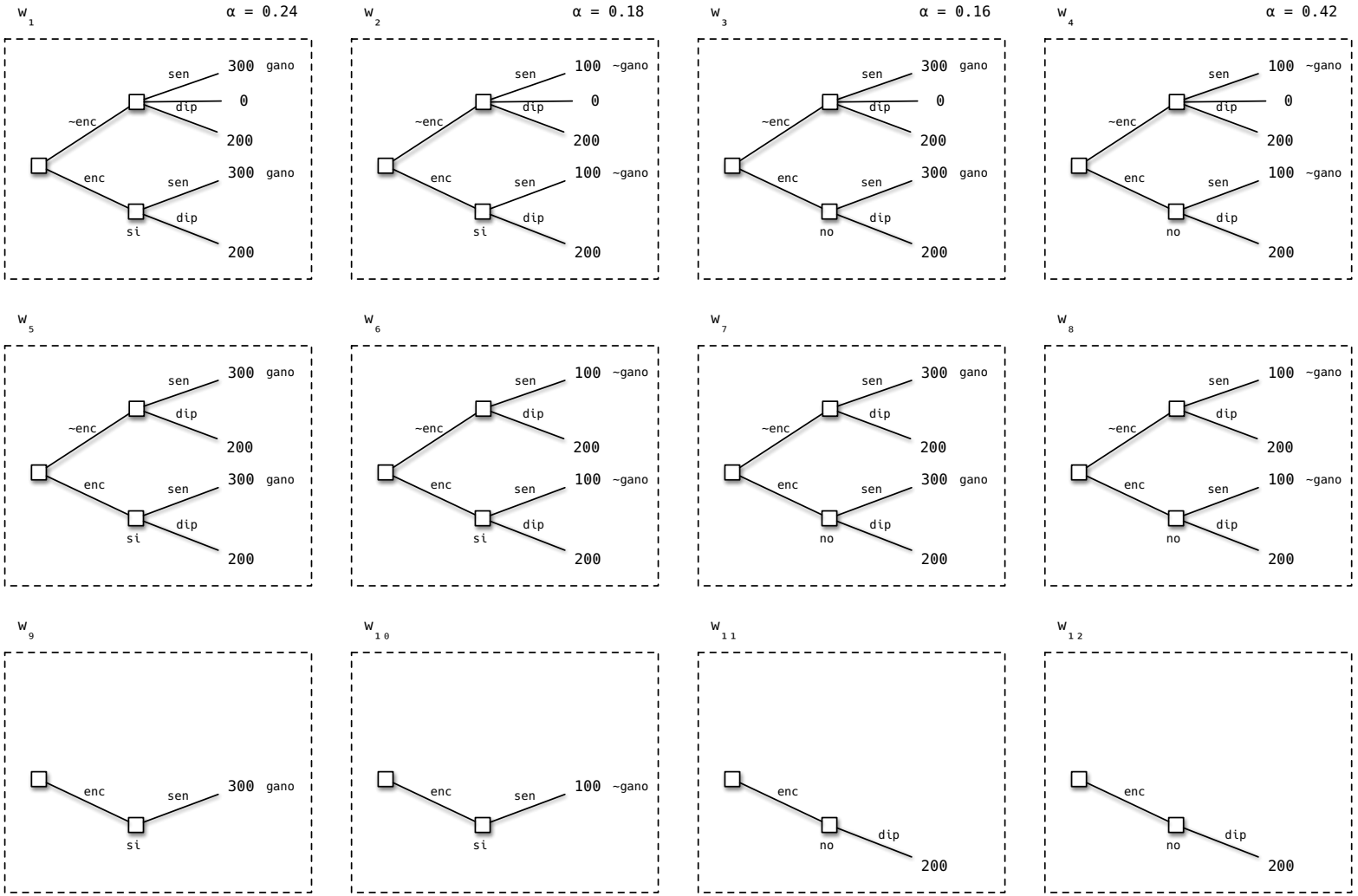
- **Daniel Dennett.** La postura intencional. Adscribir creencias, deseos (metas), intenciones a los demás, para interpretar su comportamiento.



- **John Searle.** Comunicación basada en actos de habla. Informar, ordenar, preguntar, instruir, son mensajes intencionales.

- **Michael Bratman.** Razonamiento práctico. Las intenciones son planes parciales, jerárquicos y dinámicos que se adoptan intencionalmente.

Decisiones BDI



Lógicas BDI_{CTL}

- Se trata de una lógica multimodal, ver Hughes y Cresswell [6].
- Un componente temporal (CTL) para razonar hacerca de lo que sucedera en el siguiente estado, eventualmente, en toda corrida, etc.
- Un componente ontológico para razonar acerca de las creencias (B), axiomas $KD45$
- Un componente para los deseos (D), axiomas KD.
- Un componente para las intenciones (I), axiomas KD.
- Un componente para eventos que permite razonar acerca de lo que está sucediendo o ha sucedido, con éxito o fracaso.

Principios de racionalidad

- **Anand Rao.** Modelos formales (lógicas BDI_{CTL}, $AgentSpeak(L)$) para el diseño, implementación y verificación de agentes BDI.

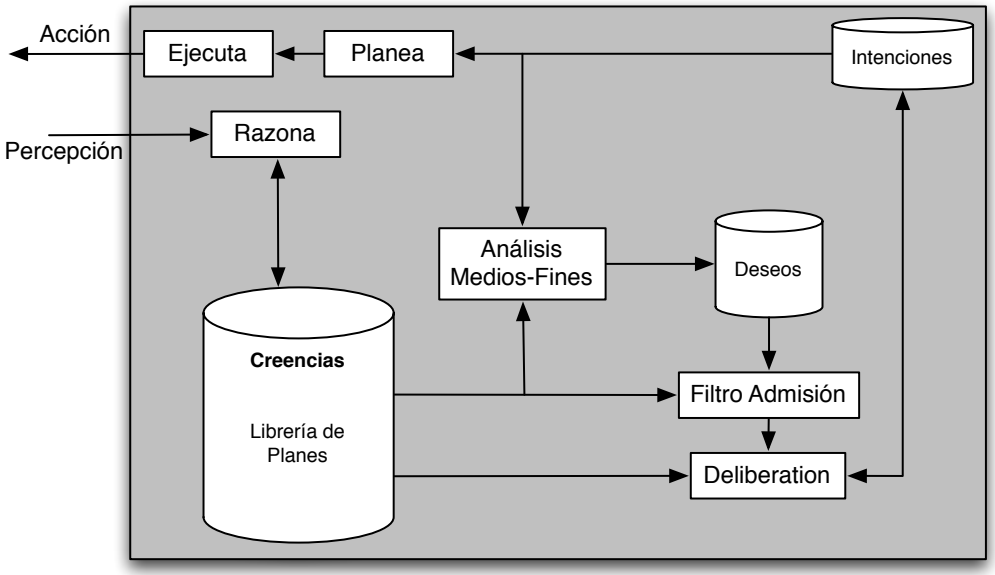
Posposición finita: $\mathsf{INT}(\phi) \rightarrow \mathsf{A}\Diamond(\neg\mathsf{INT}(\phi))$.

Compromiso racional: $\mathsf{INT}(\mathsf{A}\Diamond\phi) \rightarrow \mathsf{A}(\mathsf{INT}(\mathsf{A}\Diamond\phi)\ \mathsf{U}\ (\mathsf{BEL}(\phi) \vee \mathsf{BEL}(\mathsf{E}\Diamond\phi)))$.

Problema: ¿Cómo implementamos estas especificaciones?

Arquitecturas BDI

- Utilizar un lenguaje de especificación más computacional, p. ej., Z [7] como d'Inverno y Luck [3]:



- Problema: Una brecha muy grande con los fundamentos previamente formalizados.

Lenguajes de Programación Orientados a Agentes

- **Yoav Shoham.** Agent0 un lenguaje de programación con operadores BDI. Basado en una semántica operacional.

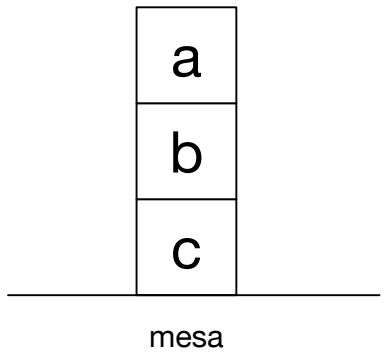
- Modelos ejecutables de agencia.
- Ejemplo. Jason [2], la implementación en Java de *AgentSpeak*(*L*): Creencias, Metas, Intenciones, Planes, Acciones, Eventos, Actos de Habla, etc.

Sintaxis

ag	$::=$	$bs \quad ps \quad gs$	
bs	$::=$	$b_1, \ldots, b_n$	$(n \geq 0)$
b	$::=$	$lit \mid rule$	$ground(lit)$
lit	$::=$	$at \mid \sim at$	
at	$::=$	$p(t_1, \ldots, t_n)$	$(p \in Pred, n \geq 0)$
	$\mid$	$p(t_1, \ldots, t_n)[s_1, \ldots, s_m]$	$(p \in Pred, n \geq 0, m > 0)$
s	$::=$	$\texttt{percept} \mid \texttt{self} \mid id$	
$rule$	$::=$	lit :- fbf	
fbf	$::=$	$lit \mid \neg fbf \mid fbf \wedge fbf \mid fbf \vee fbf$	
t	$::=$	$X \mid f \mid f(t_1, \ldots, t_n)$	$(X \in Var, f \in Func, n \geq 1)$
ps	$::=$	$p_1, \ldots, p_n$	$(n \geq 1)$
p	$::=$	$te : ct \leftarrow h$	
te	$::=$	$+lit \mid -lit \mid +g \mid -g$	
ct	$::=$	$fbf \mid \top$	
h	$::=$	$h_1; \top \mid \top$	
h_1	$::=$	$a \mid u \mid g \mid h_1; h_1$	
a	$::=$	$ac(t_1, \ldots, t_n)$	$(ac \in Actn, n \geq 0)$
u	$::=$	$+b \mid -lit$	
g	$::=$	$!lit \mid ?lit$	
gs	$::=$	$g_1, \ldots, g_n$	$(n \geq 0)$

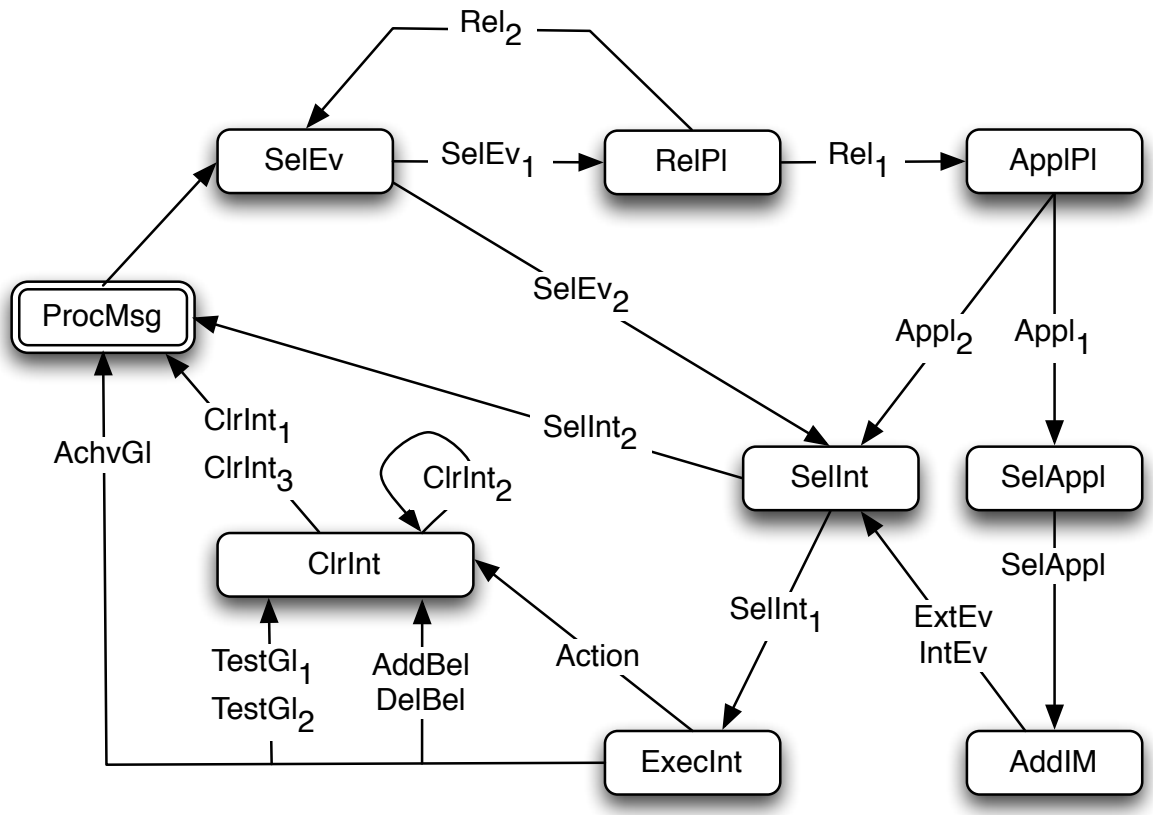
Ejemplo de Programa

```
1  /* Creencias*/
2
3  en(a,b).
4  en(b,c).
5  en(c,mesa).
6
7  libre(mesa).
8  libre(X) :- not(en(_,X)).
9
10 /* Planes */
11
12 +!poner(X,Y) : libre(X) & libre(Y) <-
13   mover(X,Y);
14   .print("Poner se llevó a cabo con éxito.");
15   .send(experimenter,tell,experiment(done)).
16
17 -!poner(X,Y) : true <-
18   .print("Poner falló.");
19   .send(experimenter,tell,experiment(done)).
```



Semántica Operacional

- Basada en un autómata $\gamma = \langle ag, C, M, T, s \rangle$



- Problema: ¿Y los operadores temporales/BDI?

Solución

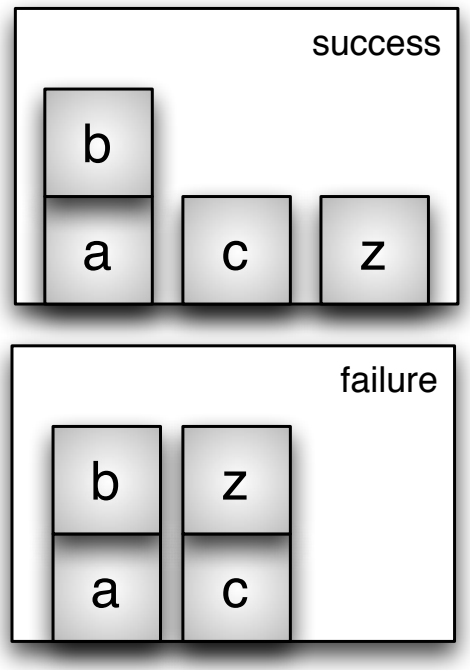
- Guerra-Hernández, Castro-Manzano y El-Fallah-Seghrouchni [5] definen CTL sobre esta estructura de Kripke.
- Los operadores BDI se definen como sigue:

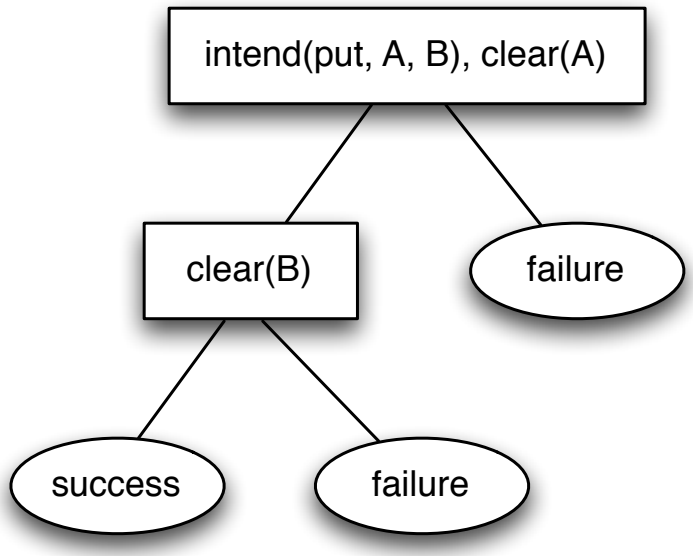
$$\begin{aligned}
\mathrm{BEL}_{\langle ag,C\rangle}(\phi) &\equiv ag_{bs} \models \phi \\
\mathrm{INT}_{\langle ag,C\rangle}(\phi) &\equiv \phi \in \bigcup_{i \in C_I} agls(i) \vee \bigcup_{\langle te,i\rangle \in C_E} agls(i) \\
\mathrm{DES}_{\langle ag,C\rangle}(\phi) &\equiv \langle +!\phi, i\rangle \in C_E \vee \mathrm{INT}(\phi)
\end{aligned}$$

- Ej. Los agentes Jason satisfacen posposición finita, no satisfacen compromiso ciego y son parcialmente racionales en su reconsideración.
- Problema: Demostración manual... model checking.

Aprendizaje Intencional

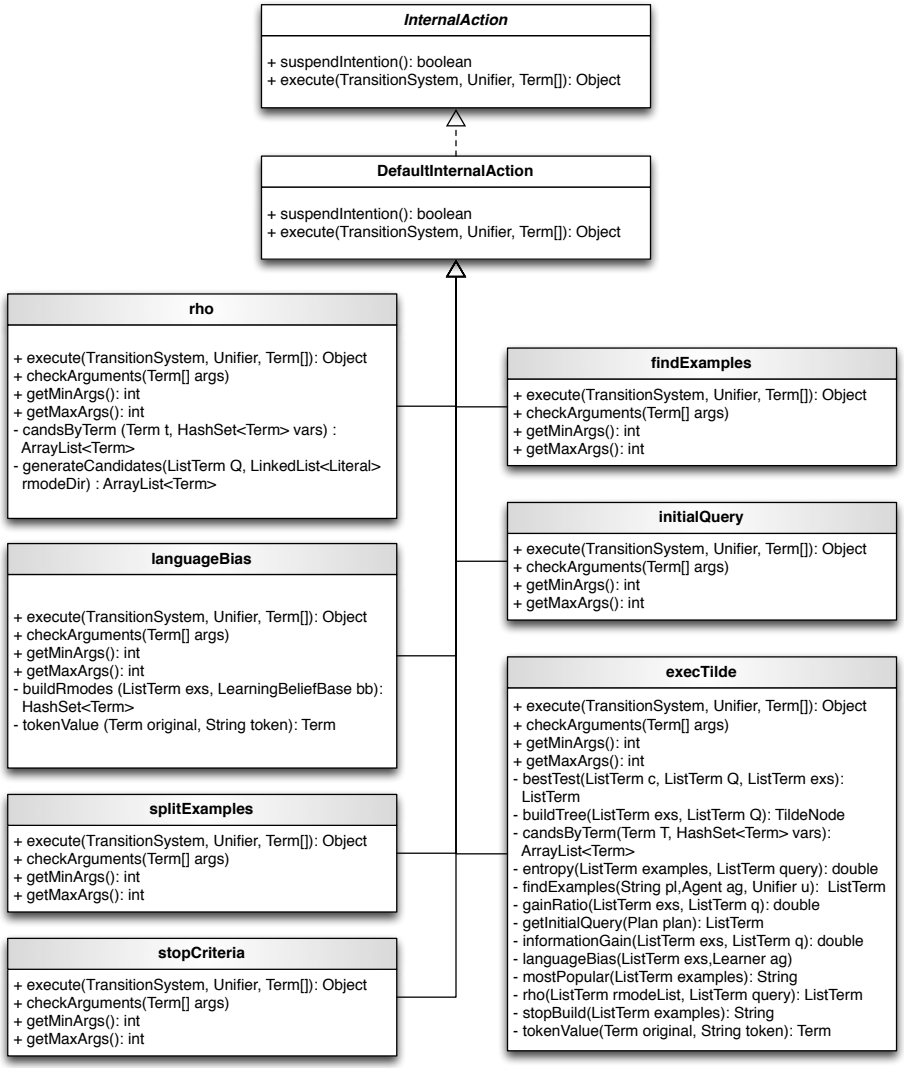
- Basado en Tilde [1] (ID3 en primer orden):



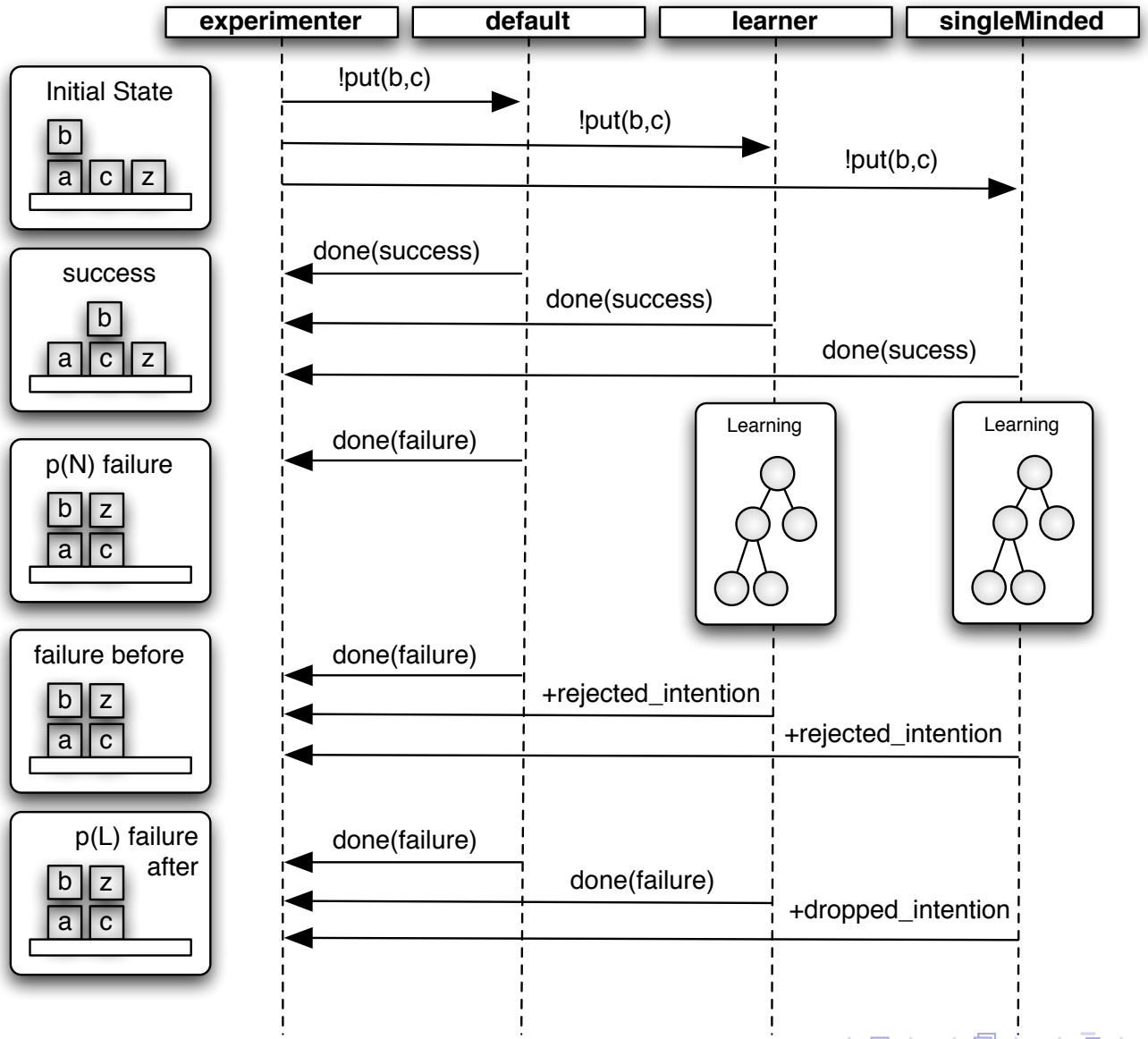


- A partir de la experiencia, aprender que solo puedo poner A en B si ambos están libres.

¡Especifiación UML!

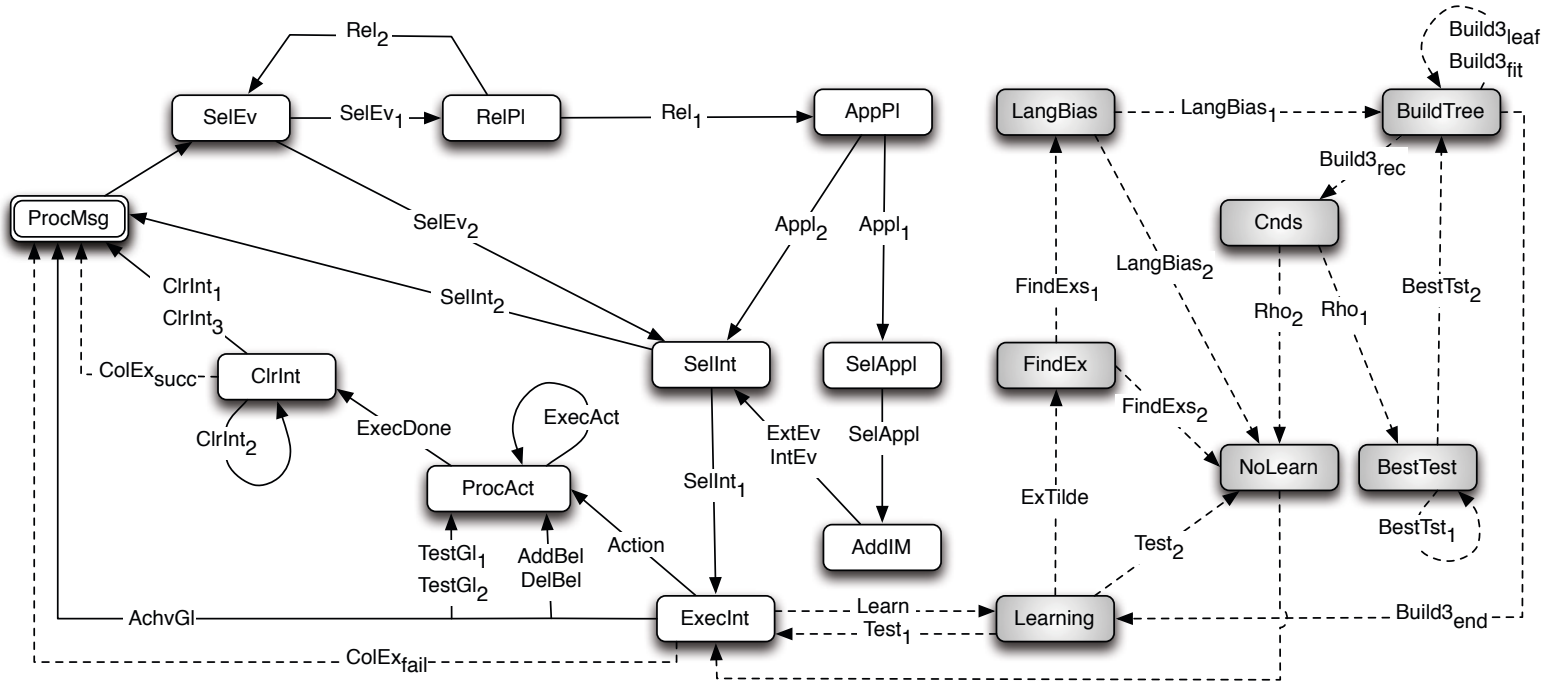


Efecto


experimenter
default
learner
singleMinded
Initial State
b
a c z
!put(b,c)
!put(b,c)
!put(b,c)
success
b
a c z
done(success)
done(success)
done(sucess)
p(N) failure
b z
a c
done(failure)
Learning
Learning
failure before
b z
a c
done(failure)
+rejected_intention
+rejected_intention
p(L) failure after
b z
a c
done(failure)
done(failure)
+dropped_intention

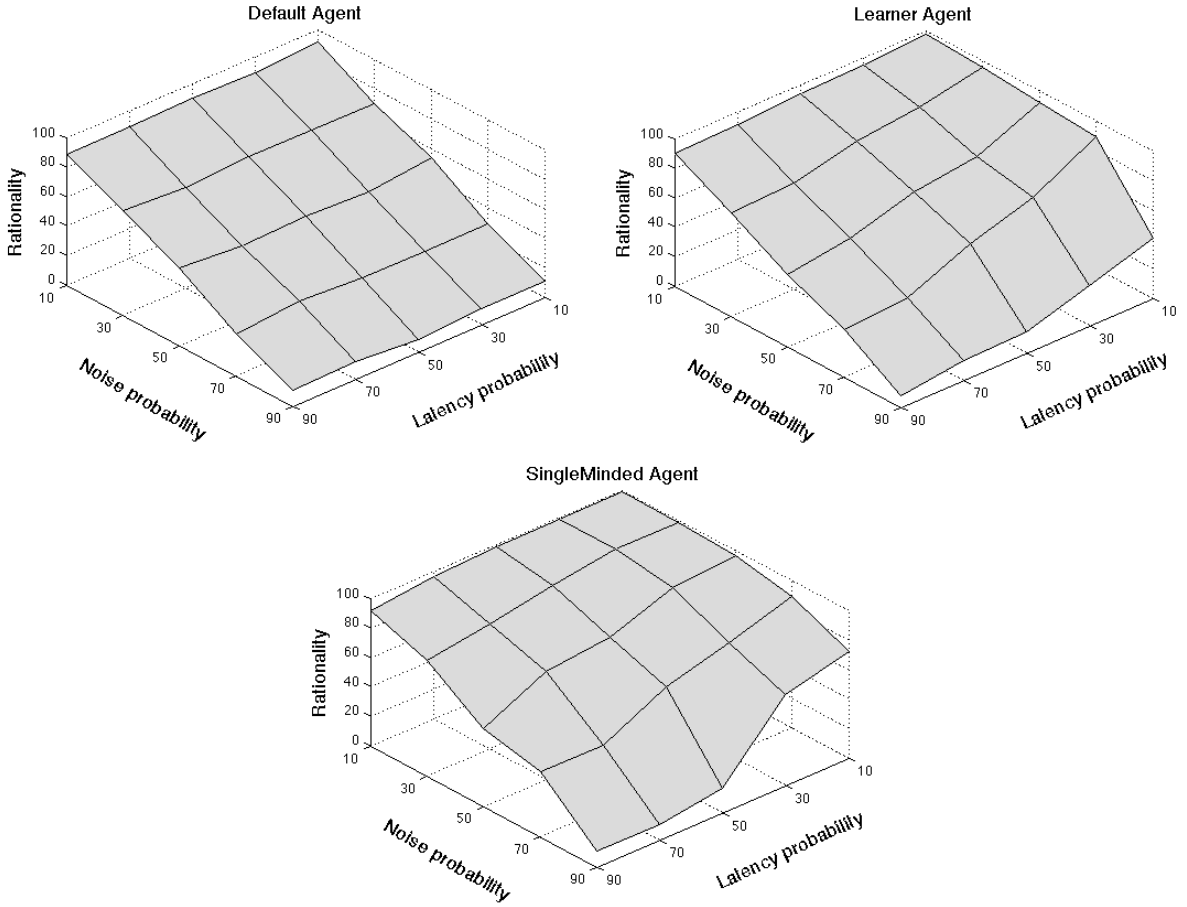
Jildt: Una arquitectura de agentes aprendices

- González-Alarcón, Grimaldo y Guerra-Hernández [4]:



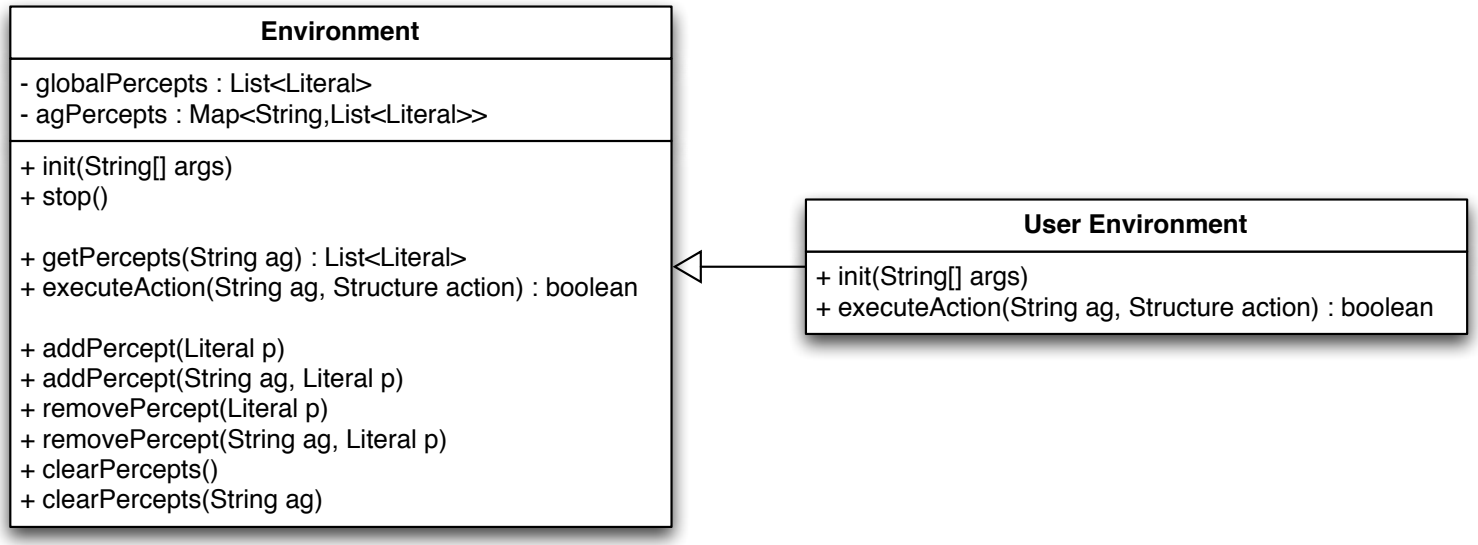
- Disponible en: `http://jildt.sourceforge.net`
- Problema: OOP + AOP muy complicado.

Resultados


Default Agent
Rationality
Noise probability
Latency probability
Learner Agent
Rationality
Noise probability
Latency probability
SingleMinded Agent
Rationality
Noise probability
Latency probability

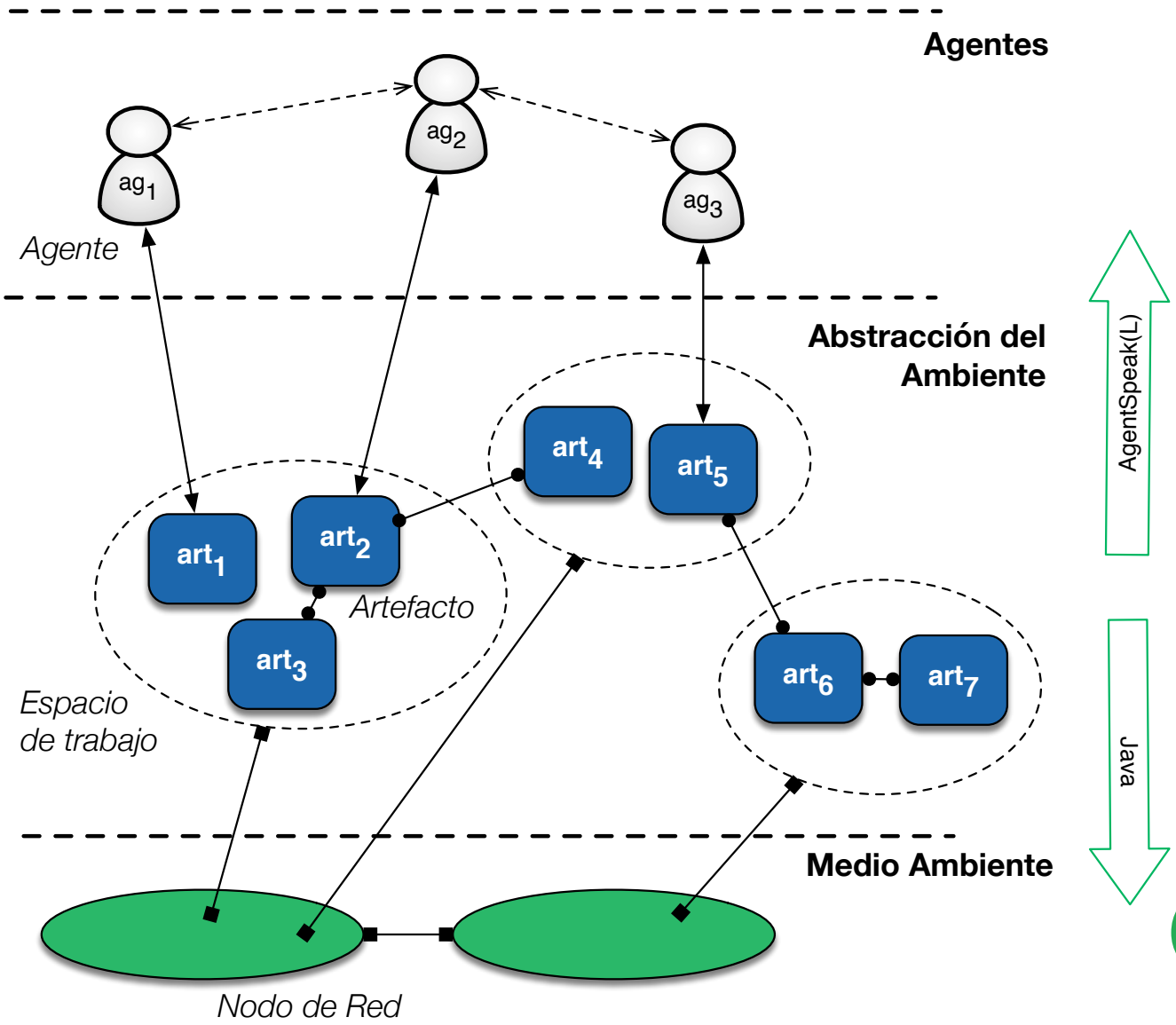
Solución tradicional

- Jason provee una clase Environment para simular ambientes implementados en Java.

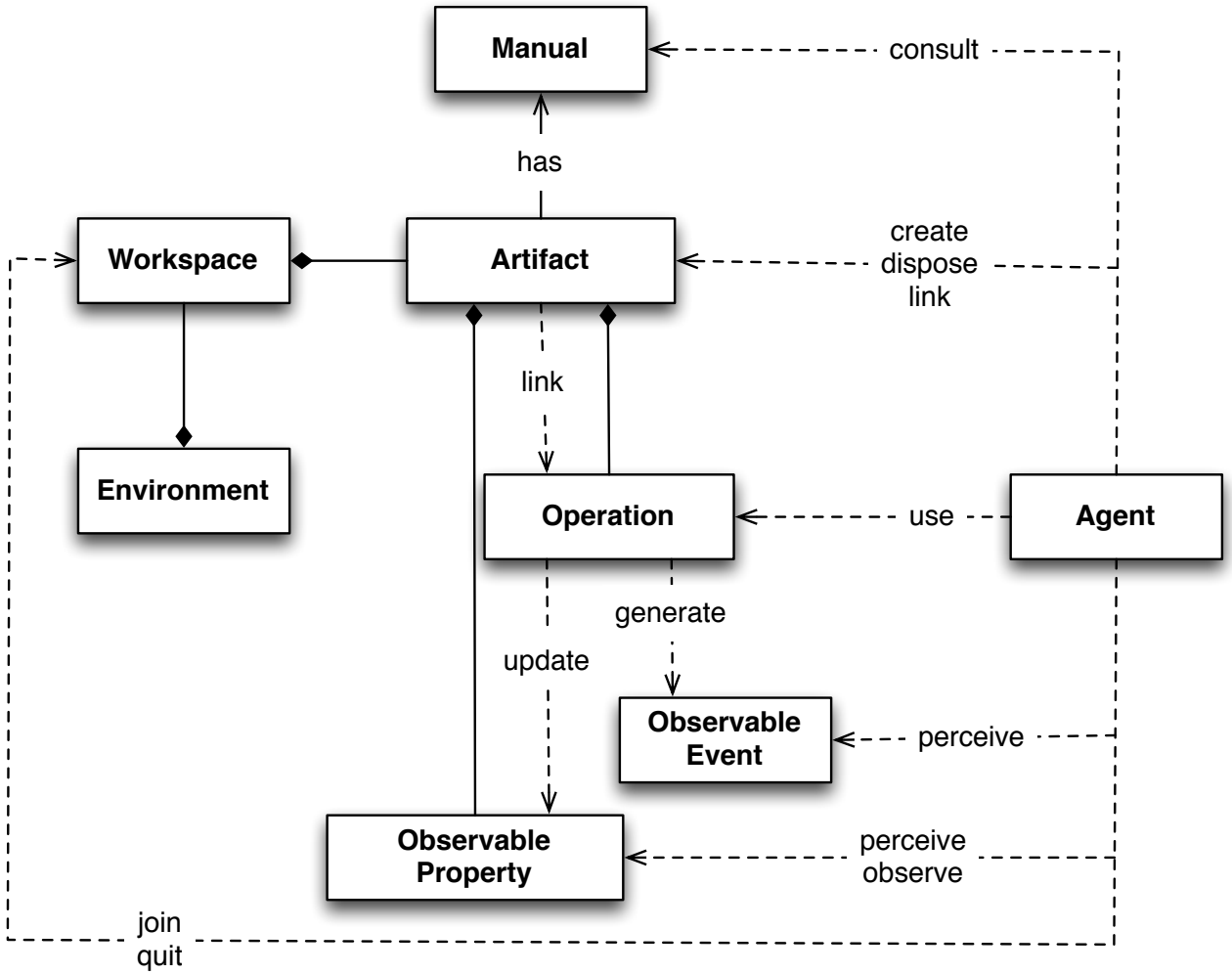


- Se comporta como un controlador de MVC.
- Problema: Centralizado.

La abstracción del ambiente como interfaz



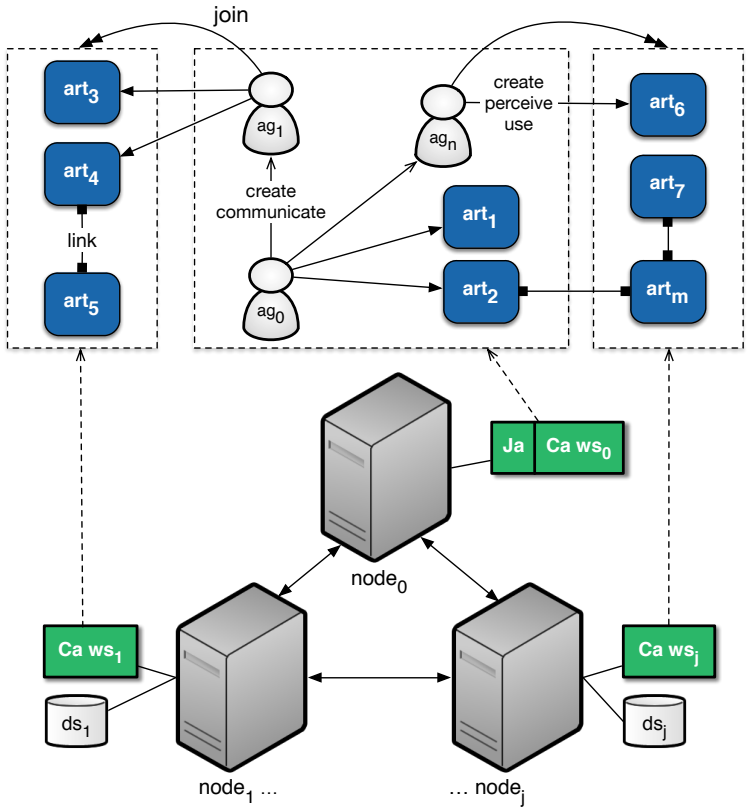
Meta-modelo de Agentes y Artefactos



Adaptado de Ricci, Piunti y Viroli [14]

JaCa-DDM

- Limón y col. [10] desarrollaron una herramienta para diseñar, implementar y evaluar estrategias de minería de datos distribuida:



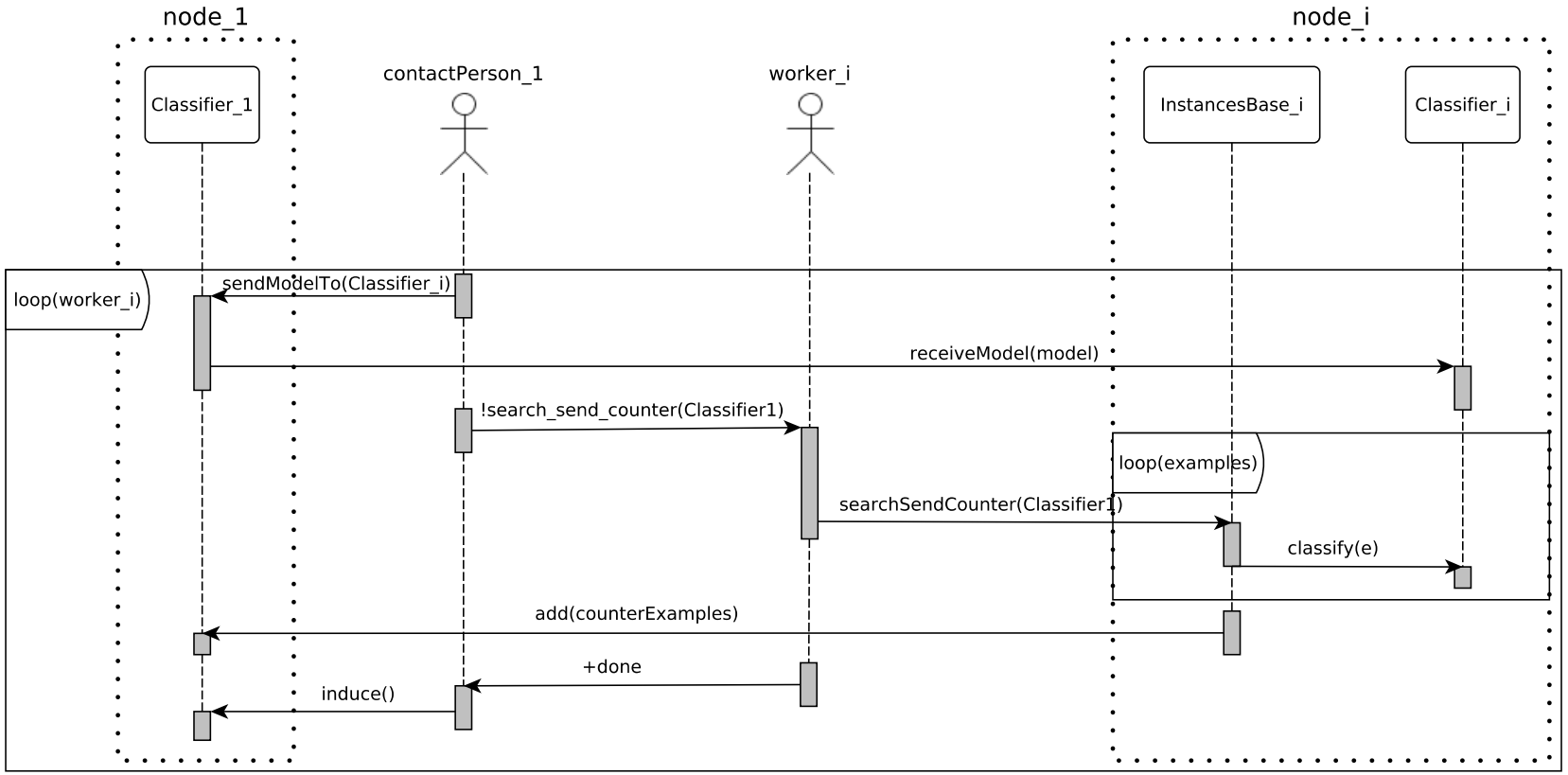
- Disponible en `https://github.com/xl666/jaca-ddm`

Basado en Weka/MOA

- Representación uniforme de ejemplos, atributos, modelos, etc.
- Herramientas de evaluación de modelos.
- Soporte: Libro, documentación, comunidad activa, etc.
- Diversos algoritmos, código abierto, para:

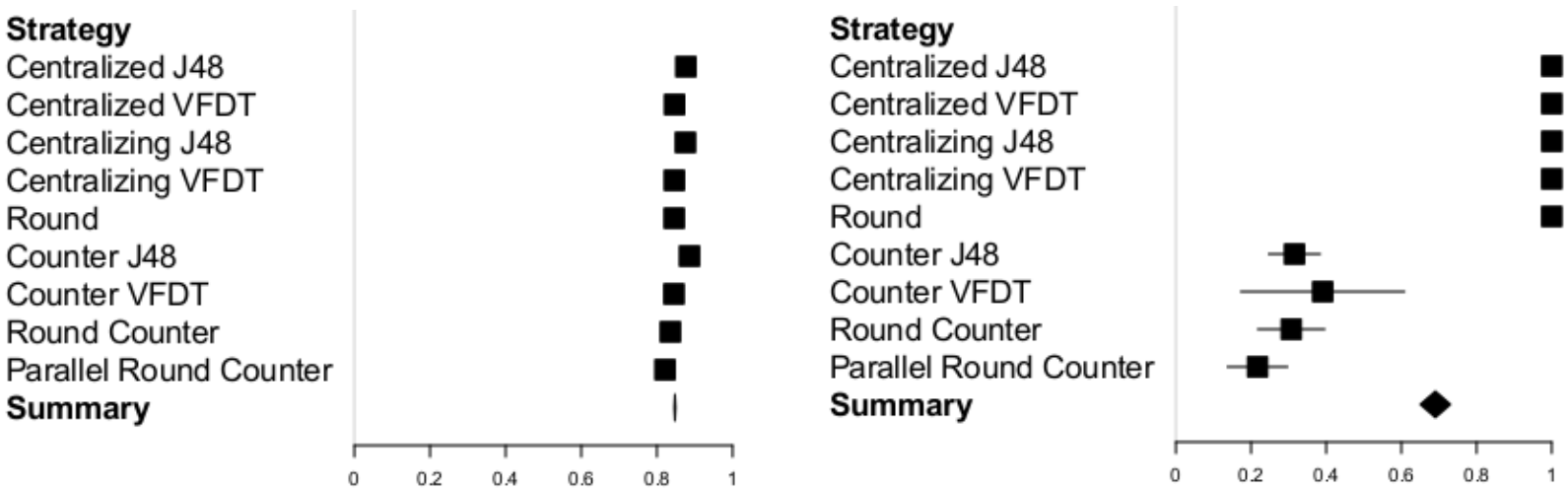
Aprendizaje	Pre-Procesamiento	Meta-Aprendizaje
Árboles de decisión	Selección de atributos	Bagging
Reglas	Discretización	Boosting
Modelos lineales	Proyección	Combinación de modelos
Modelos basados en casos	Muestreo	Regresión aditiva
Predicción numérica	Calibración	Stacking
Modelos bayesianos	etc.	etc.
Redes neuronales		
Clustering		
Flujos		

Estrategia basada en Contra Ejemplos



Precisión contra Tamaño del Conjunto de Entrenamiento

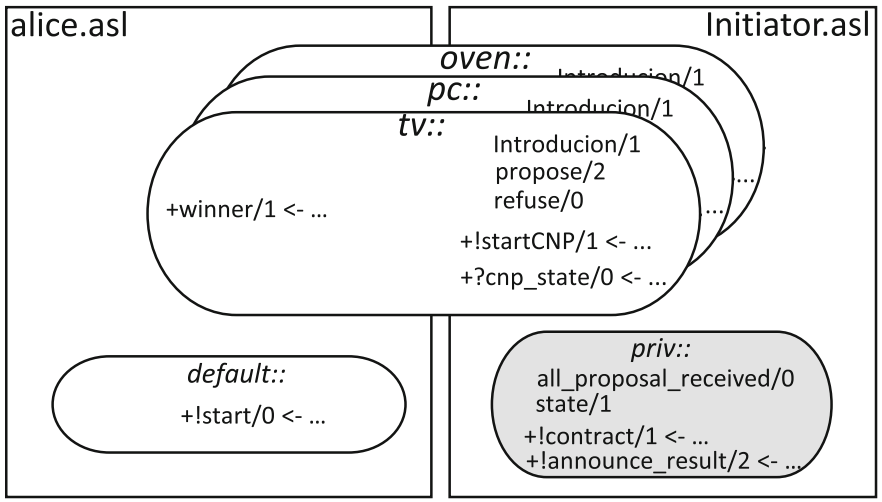
- Probando sobre 18 conjuntos de entrenamiento:



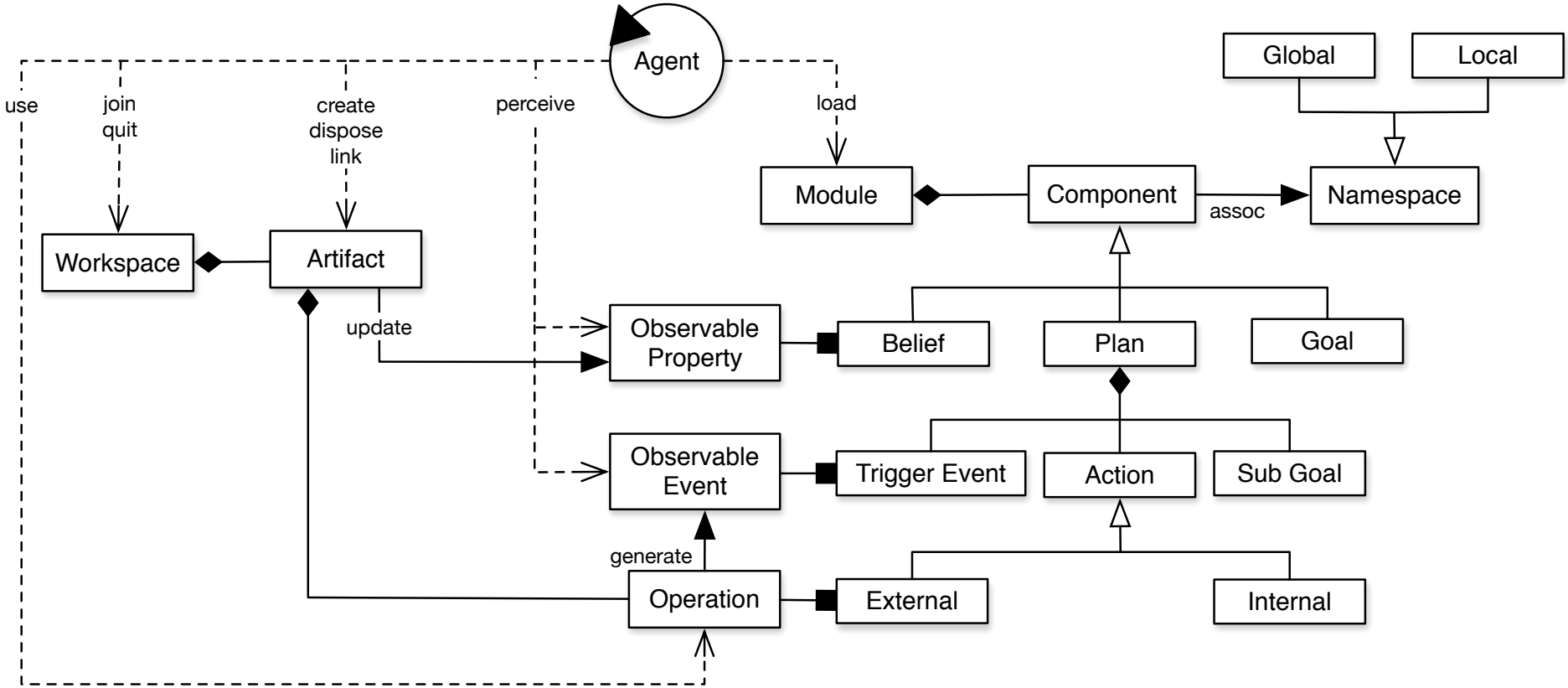
- Limón y col. [9] proponen mejoras en rendimiento usando GPUs.

Ideas

- Ortiz-Hernández y col. [13]: Las estrategias para contender con la complejidad, como descomposición, abstracción y jerarquización, toman los agentes como módulos base.
- Problema: Los agentes son en si mismos muy complejos.
- Ej. Un módulo para aprender, otros para las tareas de cada agente.
- Solución: Sintáctica basada en espacios de nombres.



Módulos JaCa



Ventaja/Problema

- Una gran diversidad:

Table 7.1 AOSE methodologies

Methodology name	Domain of origin	Methodology name	Domain of origin
AAII [2, 3]	AI-KE	MAS-CommonKADS [4]	SE + AI-KE
ADELFE [5]	SE	MASSIVE [6]	SE
ADEM [7]	SE	MESSAGE [8]	SE
ADEPT [9–11]	AI-KE	Nemo [12]	SE + AI-KE
AO [13]	SE	ODAC [14]	SE
AOR [15]	SE	OPEN for MAS [16]	SE
Cassiopeia [17]	SE	PASSI [18]	SE + AI-KE
CoMoMas [19]	SE + AI-KE	Prometheus [20]	SE + AI-KE
DESIRE [21–23]	AI-KE	Roadmap [24]	SE + AI-KE
FAF [25–29]	SE	SADDE [30]	SE
GAIA [31–33]	SE	SODA [34]	SE
INGENIAS [35]	SE	Styx [36]	SE
MASD [37]	SE + AI-KE	Tropos [38]	SE + AI-KE
MaSE [39]	SE		

SE Software Engineering, *AI-KE* Artificial Intelligence and Knowledge Engineering

Adaptado de Shehory y Sturm [15], cap. 7.

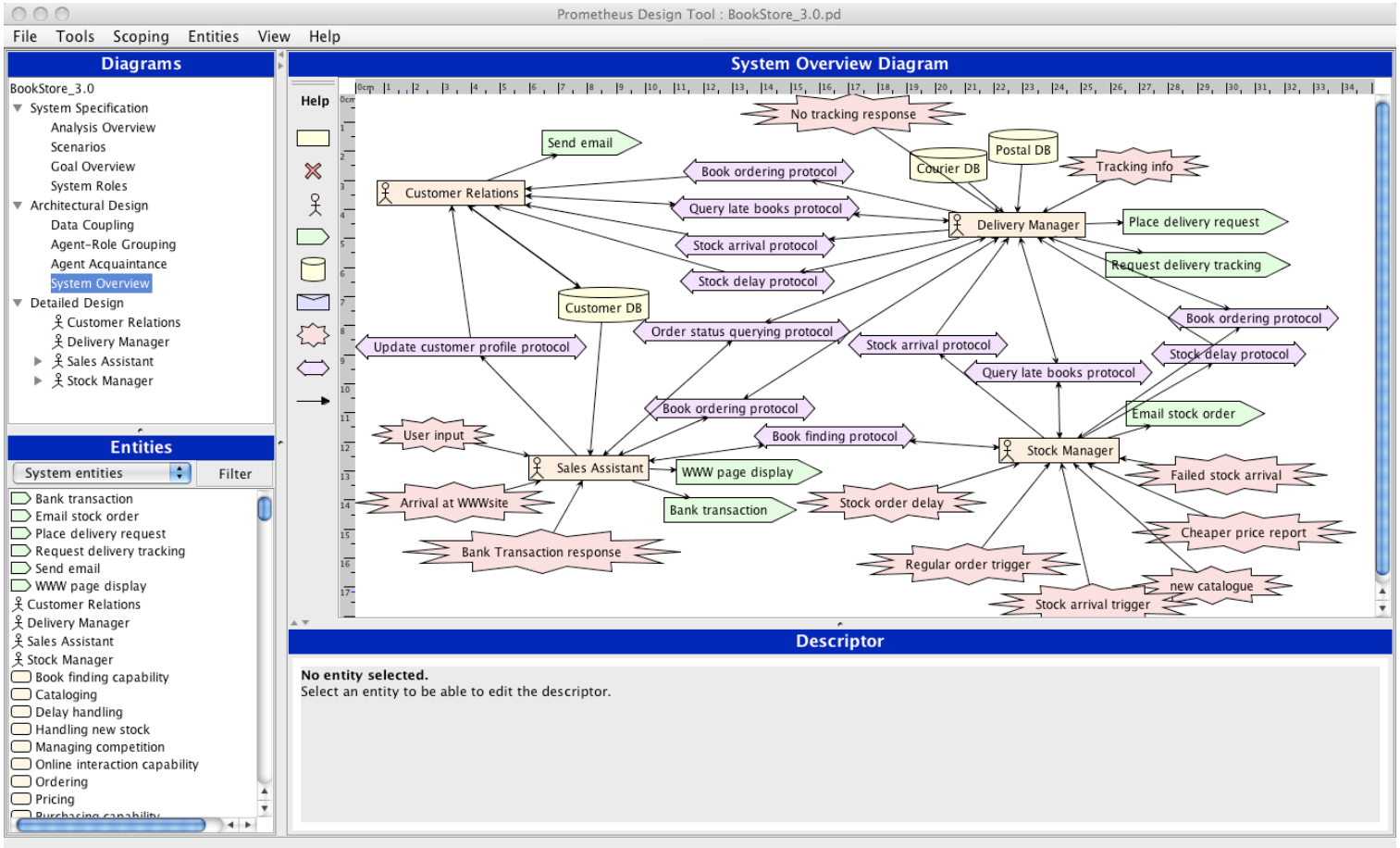
Fases y Modelos

- Constituida por tres fases y diversos modelos:

	Dynamic Models	*Structural Overview Models*	*Entity Descriptors*
System Specification	Scenarios	Goals	Functionalities actions & percepts
Architectural Design	(interaction diagrams) Interaction Protocols	(coupling diagram) (agent acquaintance) System Overview	Agents Messages
Detailed Design	Process Diagrams	Agent Overview Capability Overview	Capabilities Plans, Data, Events

Adaptado de Winikoff y Padgham [17].

Herramienta PDT

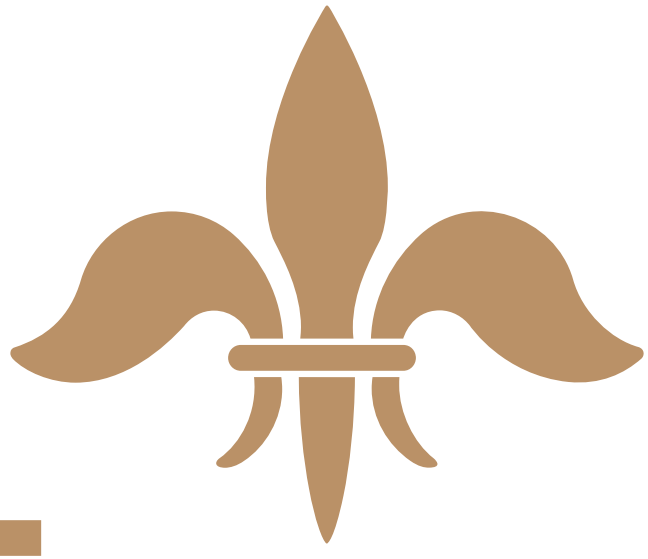


Problema. No se actualiza al ritmo de JaCaMo/Eclipse.

Ideas

- SMA que dan soporte a la ingeniería del software, p. ej., Limón y col. [8] para transacciones distribuidas en micro-servicios. Idea: Agentes para la planeación, coordinación, etc., de proyectos. Agentes mismos como servicios, p. ej., JaCa-DDM como servicio web.
- Uso de computación en la nube. Ideas: Módulos como repositorios en la nube. Infraestructura en la nube.
- SMA para el Internet de las Cosas. Idea: JaCaMo tiene una correspondencia natural.

Por otros 25, gracias



25 Años *1994-2019*

Inteligencia Artificial en Xalapa

CAi

Referencias I

H Blockeel y L De Raedt. "Top-down induction of first-order logical decision trees". En: *Artificial Intelligence* 101.1–2 (1998), págs. 285-297.

RH Bordini, JF Hübner y M Wooldridge. *Programming Multi-Agent Systems in Agent-Speak using Jason*. John Wiley & Sons Ltd, 2007.

M d'Inverno y M Luck. *Understanding Agent Systems*. Second. Berlin Heidelberg New York: Springer, 2004.

CA González-Alarcón, F Grimaldo y A Guerra-Hernández. "Jason Intentional Learning: An Operational Semantics". En: *Progress in Artificial Intelligence. 16th Portuguese Conference on Artificial Intelligence, EPIA 2013, Angra do Heroísmo, Azores, Portugal, September 9-12, 2013.* Ed. por L Correia, LP Reis y J Cascalho. Vol. 8154. Lecture Notes in Artificial Intelligence. Berlin Heidelberg: Springer Verlag, 2013, págs. 432-443.

A Guerra-Hernández, JM Castro-Manzano y A El-Fallah-Seghrouchni. "CTL AgentSpeak(L): a Specification Language for Agent Programs". En: *Journal of Algorithms* 64 (2009), págs. 31-40.

G Hughes y M Cresswell. *A New Introduction to Modal Logic*. 1998.ª ed. London, England: Routledge, 1998.

Referencias II

D Lightfoot. *Formal Specification Using Z*. Macmillan Computer Science Series. London, UK: The Macmillan Press LTD, 1991.

X Limón y col. "SagaMAS: A Software Framework for Distributed Transactions in the Microservice Architecture". En: *2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT)*. 2018, págs. 50-58. URL: doi.ieeecomputersociety.org/10.1109/CONISOFT.2018.8645853.

X Limón y col. "A Windowing strategy for Distributed Data Mining optimized through GPUs". En: *Pattern Recognition Letters* 93.Suplement C (2017), págs. 23-30.

X Limón y col. "Modeling and implementing distributed data mining strategies in JaCa-DDM". En: *Knowledge and Information Systems* 60.1 (2019), págs. 99-143.

A Newell. "The Knowledge Level". En: *AI Magazine* 2 (1981), págs. 1-20.

A Newell y HA Simon. "Computer Science As Empirical Inquiry: Symbols and Search". En: *Commun. ACM* 19.3 (1976), págs. 113-126. ISSN: 0001-0782. URL: http://doi.acm.org/10.1145/360018.360022.

Referencias III

G Ortiz-Hernández y col. "A Namespace Approach for Modularity in BDI Programming Languages". En: *Engineering Multi-Agent Systems, 4th International Workshop, EMAS 2016. Singapore, Singapore, May 9–10. Revised, Selected, and Invited Papers*. Ed. por M Baldoni y col. Vol. 10093. Lecture Notes in Artificial Intelligence. Berlin Heidelberg: Springer Verlag, 2016, págs. 117-135.

A Ricci, M Piunti y M Viroli. "Environment programming in multi-agent systems: an artifact-based perspective". En: *Autonomous Agents and Multi-Agent Systems* 23.2 (2011), págs. 158-192.

O Shehory y A Sturm. *Agent-Oriented Software Engineering: Reflections on Architechtures, Methodologies, Languages, and Frameworks*. Berlin Heidelberg: Springer-Verlag, 2014.

F Varela. *Invitation aux Science Cognitives*. Paris, France.: Editions du Seuil, 1989.

M Winikoff y L Padgham. "Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering handbook". En: ed. por MPG Federico Bergenti y F Zambonelli. Kluwer Publishing, 2004. Cap. The Prometheus Methodology.