



Comunicação em Sistemas Multi-agentes Usando o Framework Jason



Introdução à Sistemas Multi-agentes

Prof.: Viviane Silva

Aluno: Carlos Eduardo Pantoja

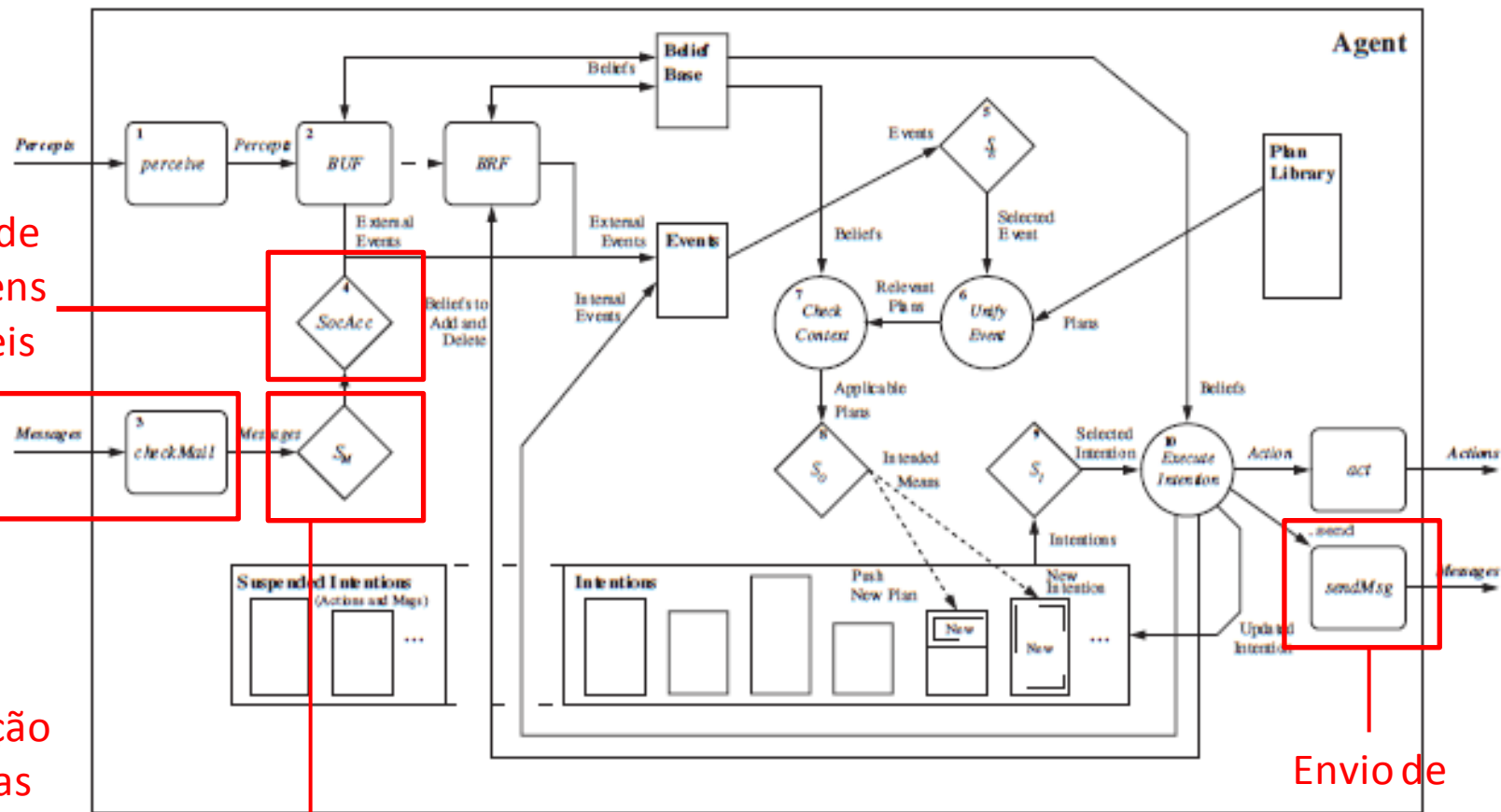
Sumário

1. Introdução
2. Background
3. Comunicação entre Agentes
4. Referências Bibliográficas

1. Introdução

No início de cada ciclo de raciocínio, o agente verifica mensagens que ele possa ter recebido de outros agentes

Baseada em *Speech Act* e *KQML*



Seleção de mensagens aceitáveis

Verificação de novas mensagens

Seleção de mensagens

Envio de mensagens

2. Background

- **Framework Jason**

O JASON é um framework baseado em AgentSpeak e Java que utiliza as principais características do PRS. Em JASON um agente é composto de **crenças, metas, planos e ações** e é programado utilizando o **AgentSpeak**.

Os agentes em JASON estão inseridos em um ambiente, que estende a classe Environment, onde as **percepções** e **reações a estímulos** do ambiente são programadas em Java (BORDINI et al., 2007).

a. Beliefs

Em Jason, um agente armazena as **informações** percebidas do ambiente; as informações internas; e informações de comunicação através de crenças.

As **crenças** são armazenadas em uma **Base de Crenças** (Belief Base).

As crenças são representadas como predicados da **lógica tradicional**. Os **predicados** representam propriedades particulares.

- **Tipos**

1. **Percepções do Ambiente (Percepts)**

Informações coletadas pelo agente que são relativas ao sensoramento constante do ambiente.

2. **Notas Mentais (Mental Notes)**

Informações adicionadas na base de crenças pelo próprio agente resultado de coisas que aconteceram no passado. Esse tipo de informação geralmente é adicionada pela execução de um plano.

3. **Comunicação**

Informações obtidas pelo agente através da comunicação com outros agentes.

- **Exemplos: Crenças Iniciais**

salario(5000).

missionStarted.

OBS.: Toda **crença inicial** em Jason deve
terminar com .

OBS.: Toda **crença** deve começar com letra
MINÚSCULA.

- **Exemplos: Strong Negation**

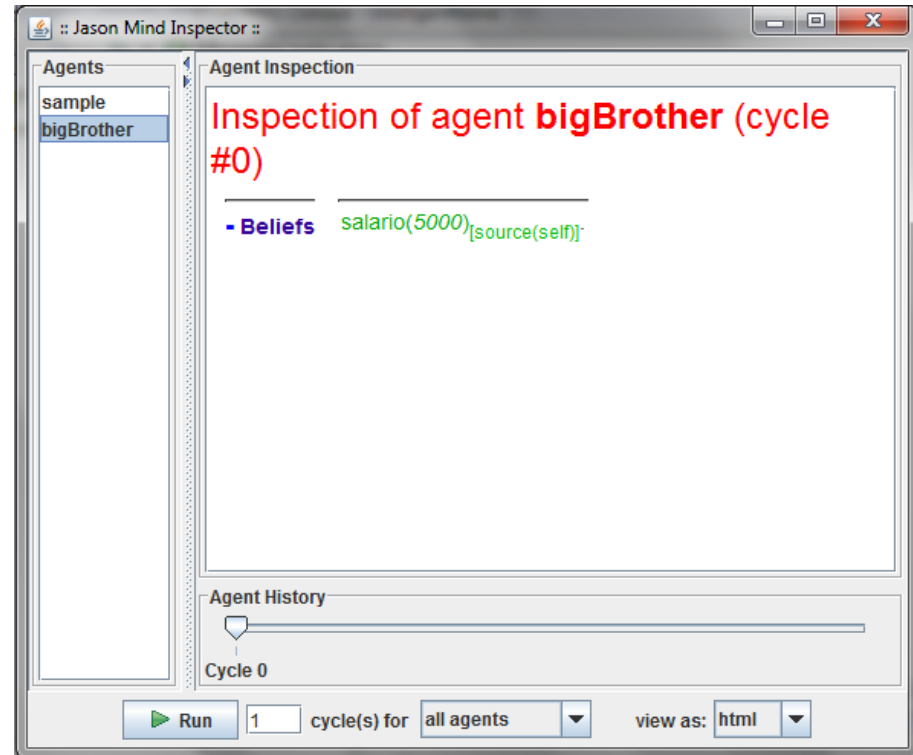
`~missionStarted.`

`~dia.`

OBS.: Toda **strong negation** em Jason
deve começar com **~**

- **Exemplos: Crenças Iniciais**

salario(5000).



b. Goals

Em Jason, os **goals** (objetivos) representam os estados do mundo em que o agente deseja atingir.

- **Tipos**

1. **Achievement Goals (!)**

É um objetivo para atingir determinado estado desejado pelo agente.

2. **Test Goals (?)**

É um objetivo que tem basicamente a finalidade de resgatar informações da base de crenças do agente.

- **Exemplos: Goals Iniciais**

!start.

!thinking.

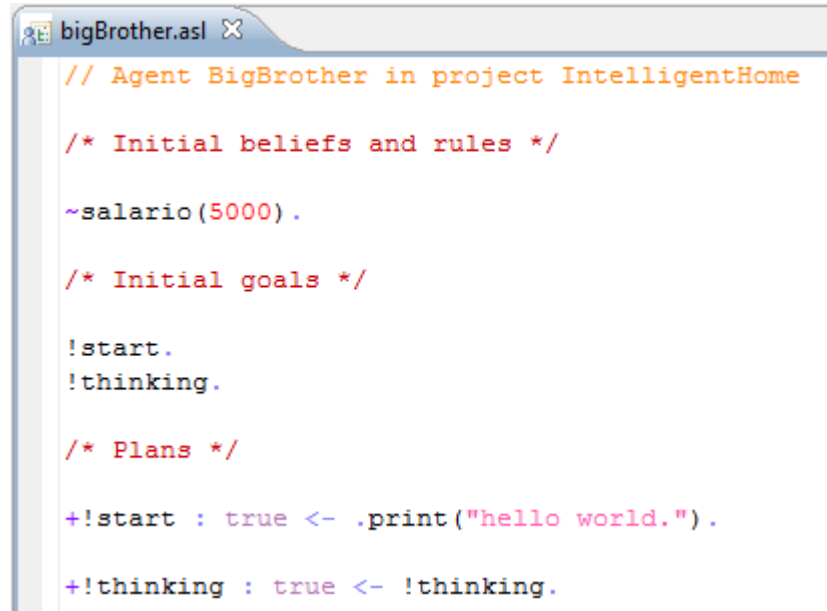
OBS.: Toda **goal inicial** em Jason deve ser um Achievement Goal; começar com **!**; e terminar com **.**

OBS.: Todo **goal** deve começar com letra **MINÚSCULA.**

- **Exemplos: Goals Iniciais**

!start.

!thinking.



```

bigBrother.asl
// Agent BigBrother in project IntelligentHome

/* Initial beliefs and rules */

~salario(5000) .

/* Initial goals */

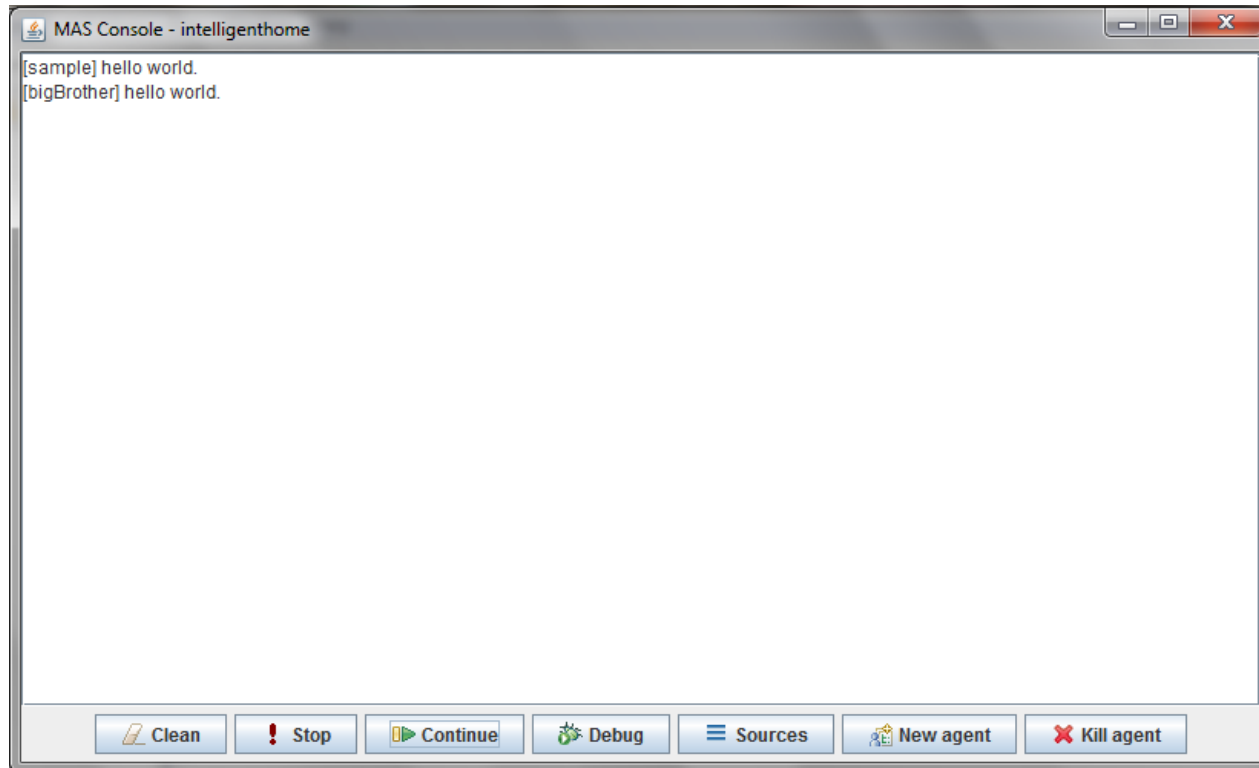
!start.
!thinking.

/* Plans */

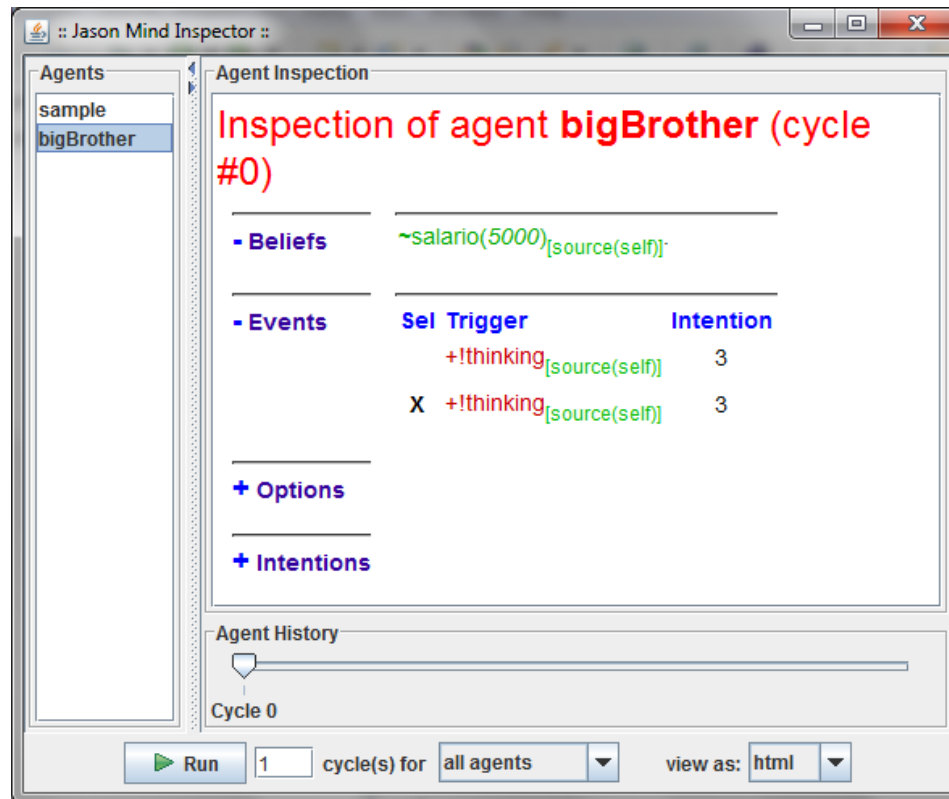
+!start : true <- .print("hello world.").

+!thinking : true <- !thinking.
  
```

- **Exemplos: Goals Iniciais**



- **Exemplos: Goals Iniciais**



c. Plans & Actions

Em Jason, um plano é composto por três partes:

Triggering_event : context <- body.

- **Descrição**

1. **Triggering Event**

Planos disponíveis para execução.

2. **Context**

Condição de ativação de determinado plano.

3. **Body**

Um conjunto de ações para determinado plano.

3. Comunicação Entre Agentes

- **Estrutura**

<sender; illocutionary forces; content>

- i. **Sender**

Uma proposição atômica representando o nome do agente que enviou a mensagem.

- ii. **Illocutionary Forces**

São as performativas que denotam as intenções do remetente.

- iii. **Content**

Conteúdo da mensagem enviada.

- **Estrutura no Jason**

.send(receiver, illocutionary forces, propositional content)

.broadcast(illocutionary forces, propositional content)

- i. **Receiver**

Uma proposição atômica em AgentSpeak representando o nome do agente que enviou a mensagem.

- ii. **Illocutionary Forces**

São as performativas que denotam as intenções do remetente.

- iii. **Propositional Content**

Um termo em AgentSpeak que varia de acordo com as forças ilocucionárias.

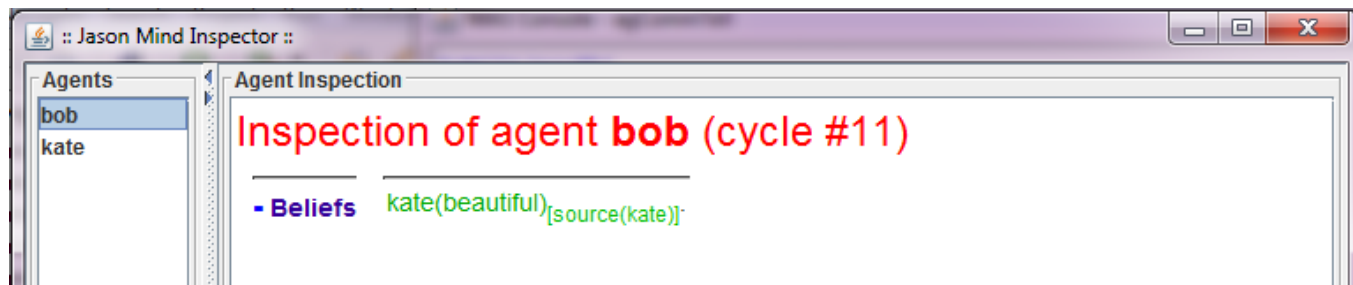
• Performativas Implementadas

1. tell

O agente remetente pretende que o receptor **acredite** que o conteúdo enviado é verdadeiro de acordo com as crenças do remetente.

Agente Kate

```
!talkTo.  
  
+!talkTo : true <-  
  .print("I'm beautiful.");  
  .send(bob, tell, kate(beautiful)).
```



- **Performativas Implementadas**

2. **untell**

O agente remetente pretende que o receptor **não acredite** que o conteúdo enviado é verdadeiro de acordo com as crenças do remetente.

Agente Kate

```
!talkTo.
+!talkTo : true <-
  .print("Hi Bob, I'm Beautiful!");
  .send(bob, tell, kate(beautiful)).

+~kate(beautiful) [source(bob)] <-
  .print("Sorry.");
  .send(bob, untell, kate(beautiful)).
```

Agente Bob

```
+kate(beautiful) <-
  +~kate(beautiful);
  .print("No, You Don't!");
  .send(kate, tell, ~kate(beautiful)).
```

- **Performativas Implementadas**

3. **achieve**

O agente remetente pede que o receptor **tente atingir um objetivo** de estado verdadeiro de acordo com conteúdo enviado.

Agente Kate

```
!talkTo.

+!talkTo : true <-
  .print("Please, turn on the lights.");
  .send(bob, achieve, turn(on)).
```

Agente Bob

```
+!turn(on) <-
  .print("Lights On.).
```



- **Performativas Implementadas**

- 4. **unachieve**

O agente remetente pede que o receptor **deixe de tentar atingir um objetivo** de estado verdadeiro de acordo com conteúdo enviado.

Agente Kate

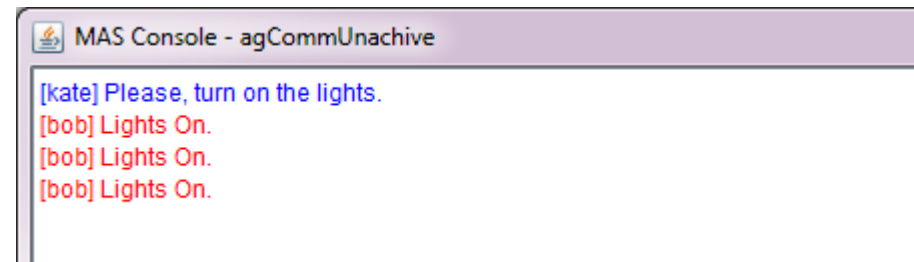
```
!talkTo.

+!talkTo : true <-
    .print("Please, turn on the lights.");
    .send(bob, achieve, turn(on)).

+light(on) <-
    .send(bob, unachieve, turn(on)).
```

Agente Bob

```
+!turn(on) <-
    .print("Lights On.");
    .send(kate, tell, light(On));
    !turn(on).
```



```
MAS Console - agCommUnachieve

[kate] Please, turn on the lights.
[ bob] Lights On.
[ bob] Lights On.
[ bob] Lights On.
```

- **Performativas Implementadas**

5. askOne

O agente remetente deseja saber se a resposta do receptor para determinada questão é verdadeira.

Agente Kate

```
!talkTo.

+!talkTo : true'<-
  .print("What's your name?");
  .send(bob, askOne, name(Name), Reply);
+Reply.
```

Agente Bob

```
name(bob).
```



- **Performativas Implementadas**

- 6. askAll

O agente remetente deseja saber todas as repostas do receptor sobre uma questão.

Agente Bob

```
weather(clean).
weather(sunny).
```

Agente Kate

```
!goToBeach.

+!goToBeach <-
    !talkTo;
    !analyze.

+!talkTo : true <-
    .print("What is the weather forecast?");
    .send(bob, askAll, weather(Name)).
```



• Performativas Implementadas

7. askHow

O agente remetente deseja saber todas implementações de planos do receptor para determinado plano.

Agente Kate

```
!talkTo.
!turn(on).

+!talkTo : true <-
    .print("Please, Can you teach me how to turn on the lights?");
    .wait(5000);
    .send(bob, askHow, "+!turn(on)").
```

Agente Bob

```
+!turn(on) <-
    .print("Lights On.").
```



- **Performativas Implementadas**

- 8. **tellHow**

O agente remetente informa ao agente receptor a implementação de um plano.

Agente Bob

```
!teach(kate).

+!teach(kate) <-
    .print("This is how we do it.");
    .send(kate, tellHow, "+!turn(on) <- .print(\"Lights On.\").");
    .wait(3000);
    .send(kate, achieve, turn(on)).

+!turn(on) <-
    .print("Lights On.");
```



- **Performativas Implementadas**

9. **untellHow**

O agente remetente solicita ao agente receptor a remoção da implementação de um plano da biblioteca de planos do receptor.

Agente Bob

```
!teach(kate).

+!teach(kate) <-
  .print("This is how we dance.");
  .wait(2000);
  .send(kate, tellHow, "@d +!dance : true <- .print(\"I'm dancing with myself!\");
                                     .wait(1000); !dance."
  );
  .wait(2000);
  .send(kate, untellHow, "@d").

@d
+!dance : true <-
  .print("I'm dancing with myself!");
  .wait(1000);
  !dance.
```

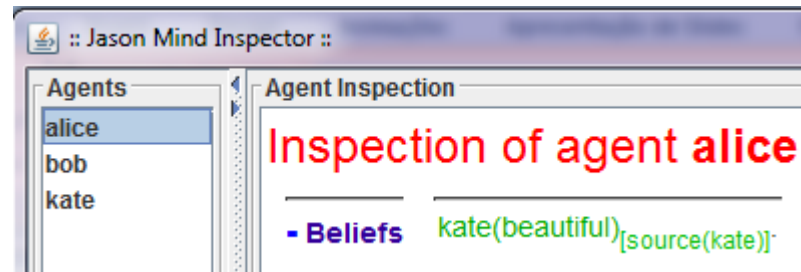
• Performativas Implementadas

10. broadcast

Permite o uso de todas as performativas vistas anteriormente. Contudo, não é preciso identificar o agente de destino, visto que ela será enviada a todos os agentes do SMA.

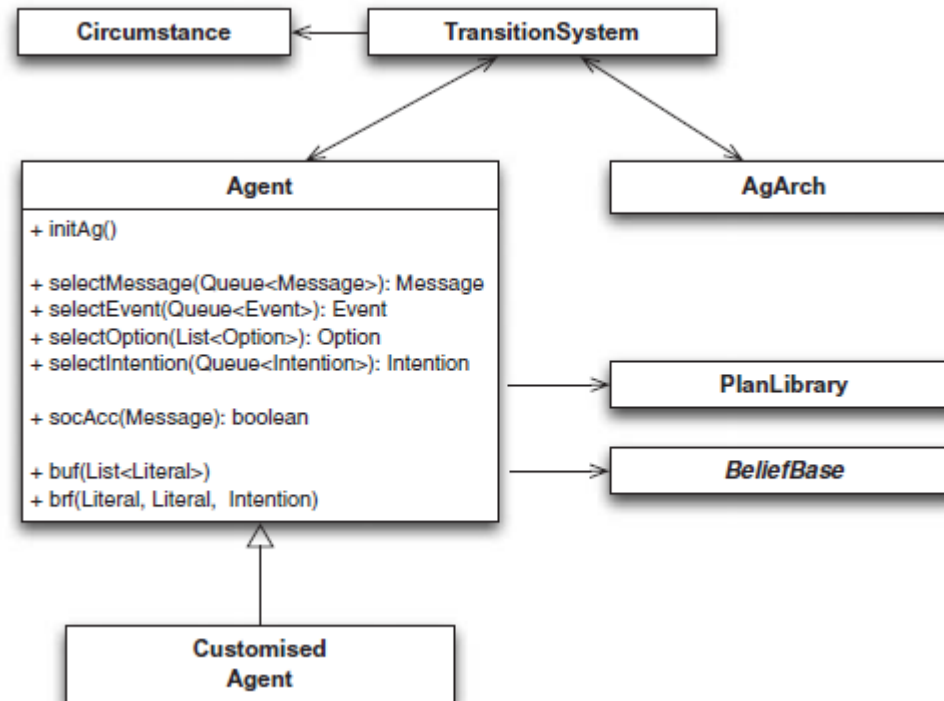
Agente Kate

```
!talkTo.  
  
+!talkTo : true <-  
  .print("I'm beautiful.");  
  .broadcast(tell, kate(beautiful)).
```



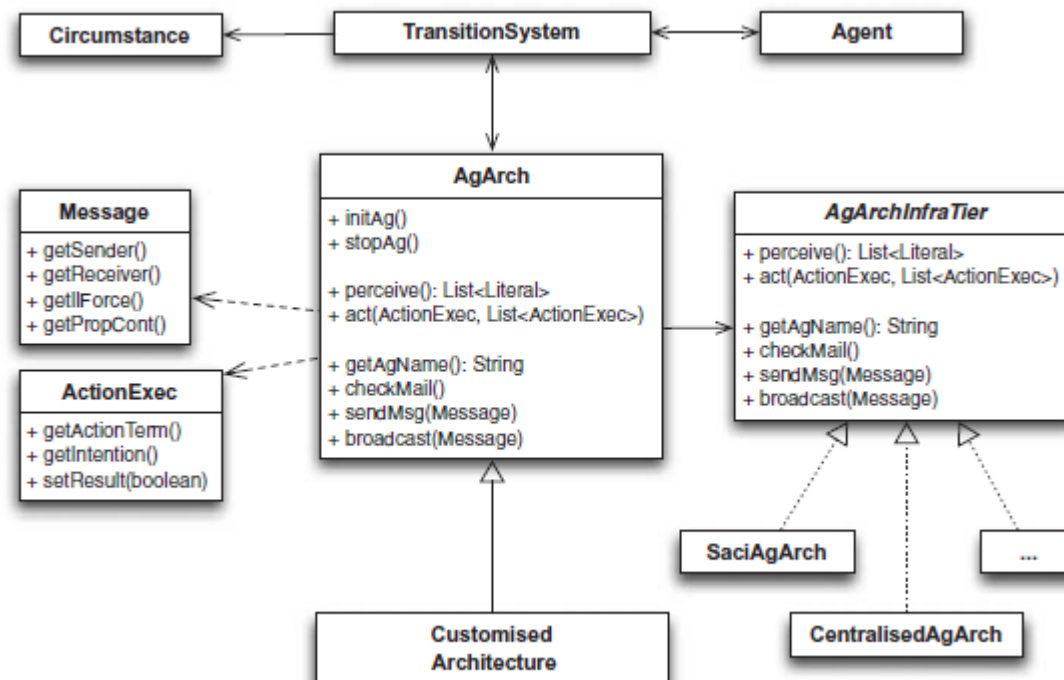
- **Por trás do Jason**

1. **Agente**



• Por trás do Jason

2. Arquitetura



4. Referências Bibliográficas

Bordini, R. H., Hubner, J. F., and Wooldridge, W. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley and Sons, London.

Boissier, O., Bordini, R. H., Hubner, J. F., Ricci, A., and Santi, A. (2012). JaCaMo project. <http://jacamo.sourceforge.net/>.