Arquitetura baseada em sistemas multiagentes para cooperação entre robôs em jogos de perseguição-evasão

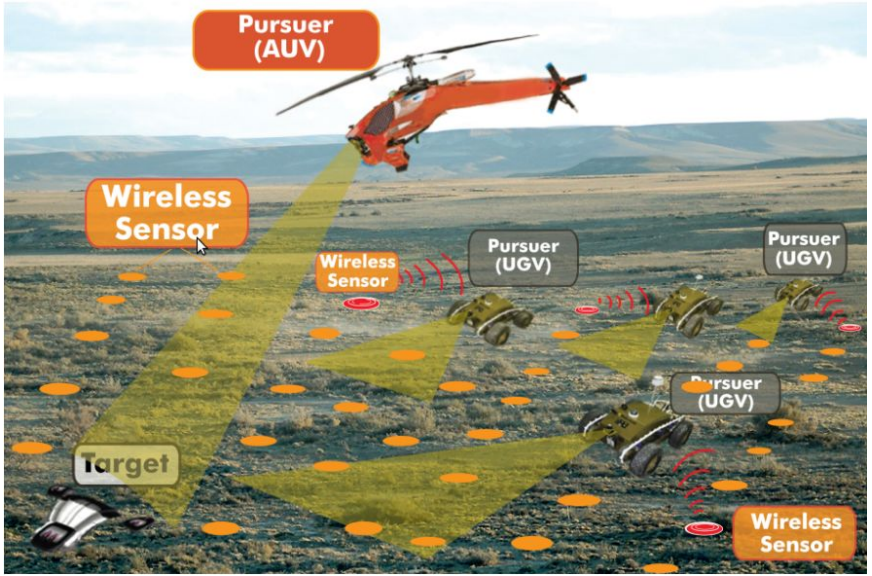
Eduardo Rehbein de Souza

1. Introdução

1. Apresentação da problemática
2. Solução proposta no artigo base
3. Avaliação da solução proposta

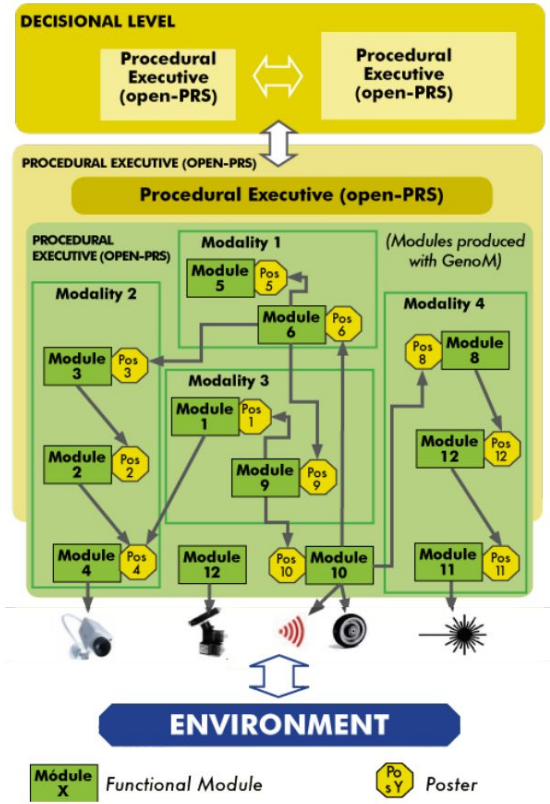
2. Jogos de perseguição-evasão

1. O que são?
2. Quais as aplicações?
3. Estágios
   - 3.1. Exploração
   - 3.2. Cobertura
   - 3.3. Perseguição



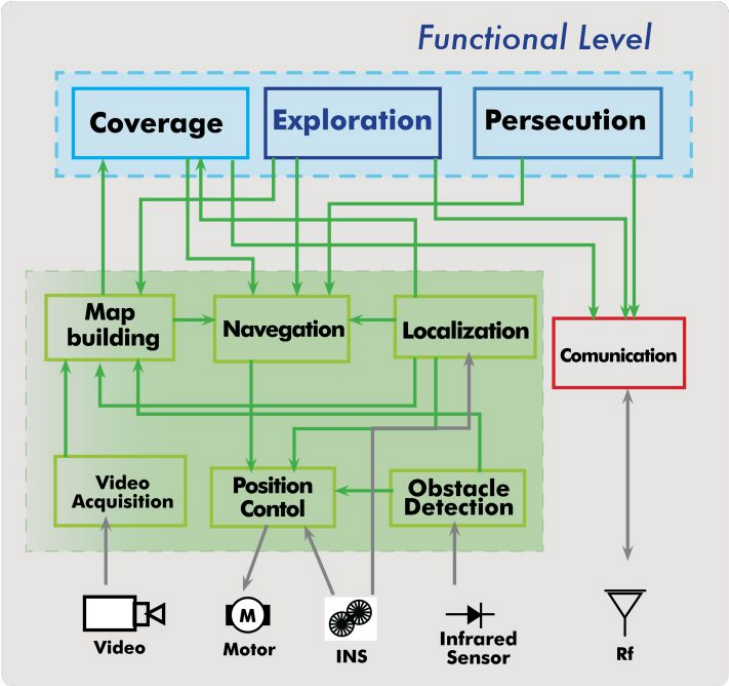
3. Arquitetura proposta para o software dos robôs

1. Arquitetura macro híbrida (LAAS)
   1.1. Nível de decisão
   1.2. Nível funcional
      1.2.1. Módulos
      1.2.2. Anúncios



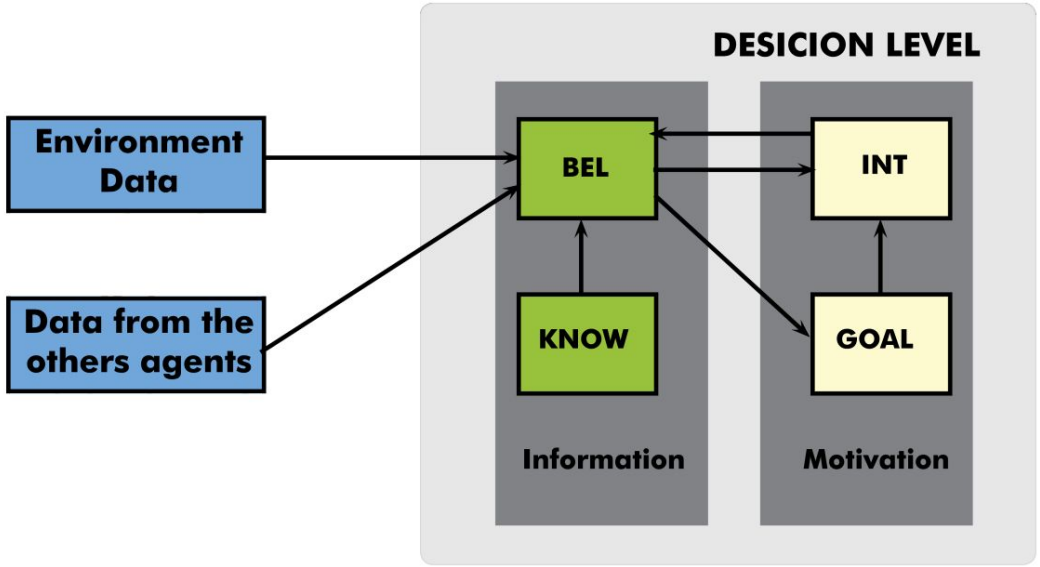
3. Arquitetura proposta para o software dos robôs

2. Nível funcional proposto



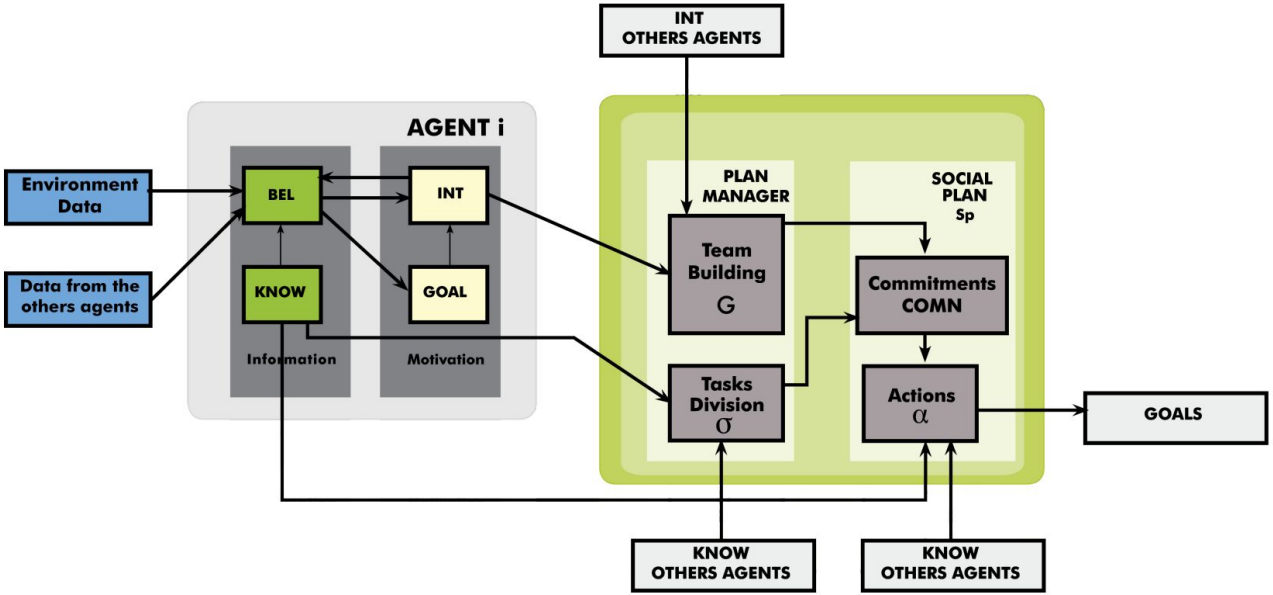
3. Arquitetura proposta para o software dos robôs

3. Nível de decisão proposto



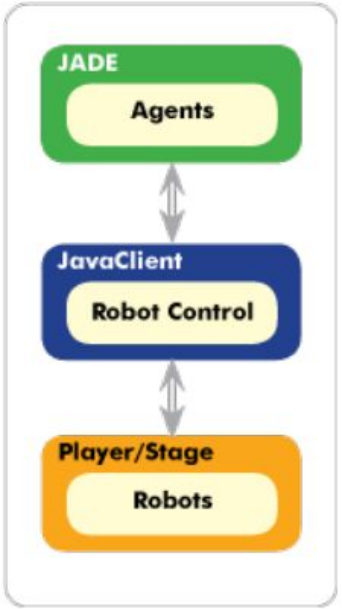
3. Arquitetura proposta para o software dos robôs

4. Arquitetura de cooperação proposta



4. Implementação da arquitetura proposta

1. Programação orientada a agentes



4. Implementação da arquitetura proposta

2. Páginas amarelas

Code 1. Intentions

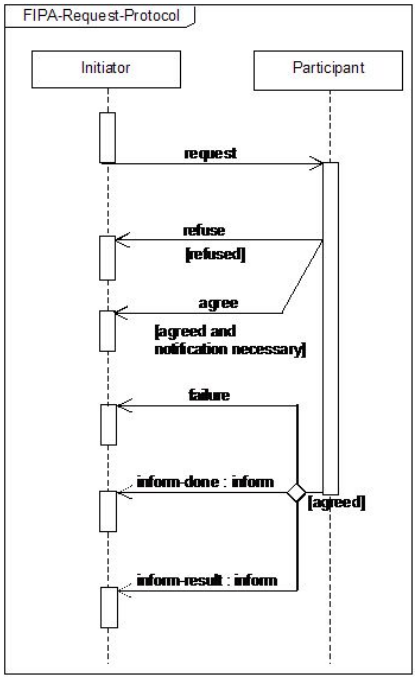
```
protected  ACLMessage prepareResultNotification(↩
    ACLMessage request, ACLMessage response) throws ↩
    FailureException{
       if (request.getContent().equals("atitudes")){
       //inicia paginas amarelas
     //Inten  es

    sd = new ServiceDescription();
    sd.setType("Intencao");
    sd.setName("Mapear");
    dfd.addServices(sd);

    sd = new ServiceDescription();
    sd.setType("Intencao");
    sd.setName("Perseguir");
    dfd.addServices(sd);

    sd = new ServiceDescription();
    sd.setType("Intencao");
    sd.setName("L der da equipe");
    dfd.addServices(sd);
```

Code 2. knowledge

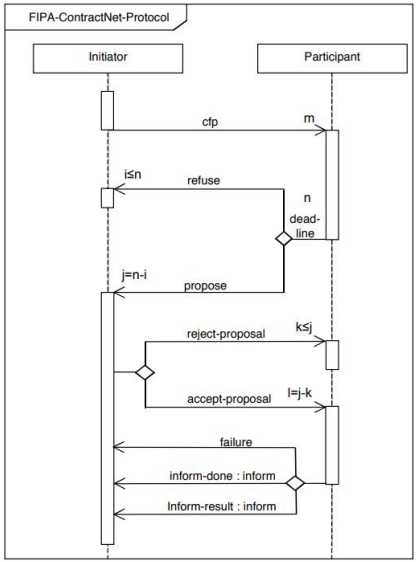
```
protected  ACLMessage prepareResultNotification(↩
    ACLMessage request, ACLMessage response) throws ↩
    FailureException{
       if (request.getContent().equals("atitudes")){
       //inicia paginas amarelas

       dfd.setName (getAID( )) ;// informa AID do ↩
            agente
       //criar inten  es e conhecimentos
       ServiceDescription sd = new ServiceDescription()↩
           ;
       sd.setType("Conhecimentos");
       sd.setName( "Mapeamento" );
       dfd.addServices(sd);

       //otros conhecimentos do agente
    sd = new ServiceDescription();
    sd.setType("Conechimentos");
    sd.setName("persegui  o");
    dfd.addServices(sd);

    sd = new ServiceDescription();
    sd.setType("Conechimentos");
    sd.setName("Lider da equipe");
    dfd.addServices(sd);
```

4. Implementação da arquitetura proposta

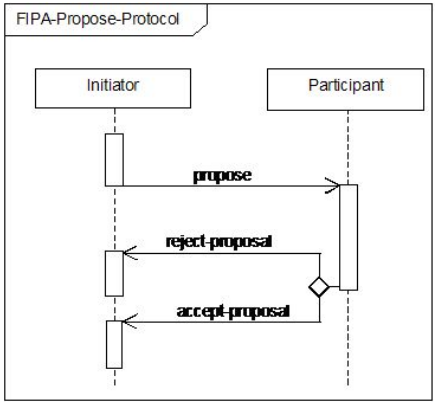
3. Principal Chief
   - 3.1. Requisição de atualização das PAs
   - 3.2. Construção de times
   - 3.3. Definição de líderes



3.1



3.2



3.3

4. Implementação da arquitetura proposta

4. Comunicação durante a execução da missão

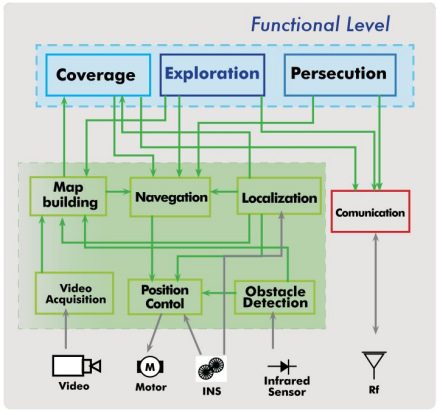
Parameter	Category of Parameters
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

5. Avaliação da arquitetura e implementação propostas

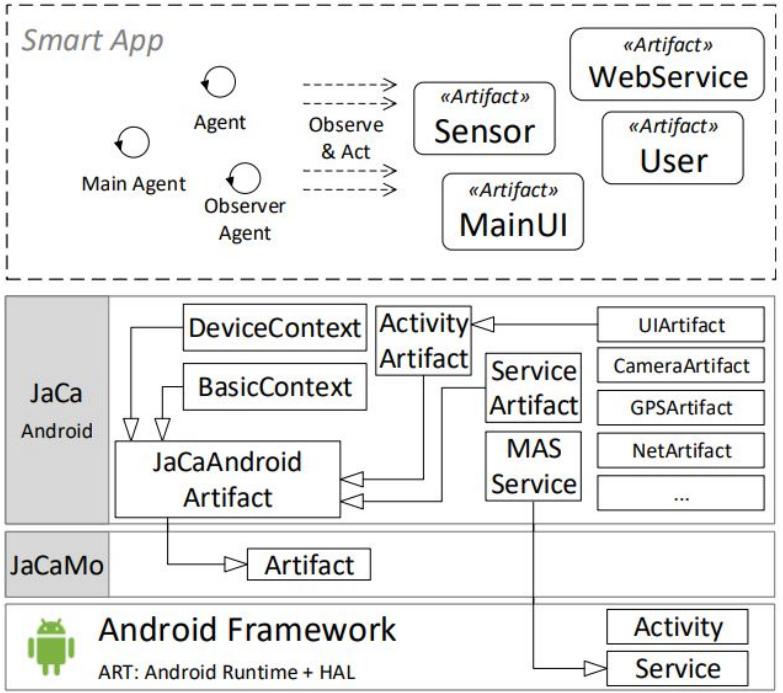
1. Estrutura macro (LAAS)
2. AOP vs MAOP
   - 2.1. Ambiente
   - 2.2. Organização
3. Páginas amarelas
4. Protocolos de comunicação

5. Avaliação da arquitetura e implementação propostas

2. AOP vs MAOP
   2.1. Ambiente
      2.1.1. Nível funcional
      2.1.2. Exemplo JaCa-Android



2.1.1

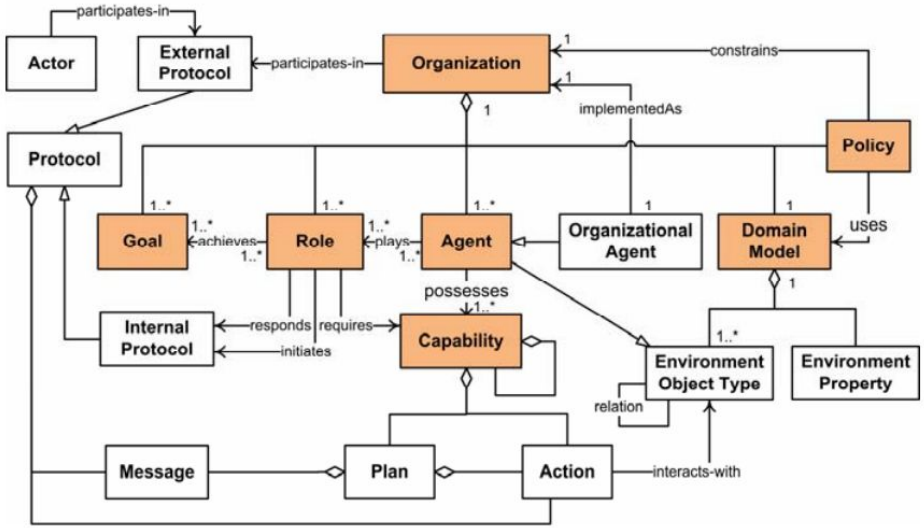


2.1.2

5. Avaliação da arquitetura e implementação propostas

2. AOP vs MAOP
   2.2. Organização
      2.2.1. Hierarquia forte

Exemplo: O-MaSE



6. Referências bibliográficas

1. E. Peñazola, U. Moreno: Architecture for cooperation between robots based on multi-agent system for pursuit evasion games
2. R. Woodman et al: Safety Control Architecture for Personal Robots: Behavioural Suppression with Deliberative Control
3. A. Croatti, A. Ricci: Programming Agent-based Mobile Apps: The JaCa-Android Framework
4. S. DeLoach, J. García-Ojeda: O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems
5. S. Rodriguez et al: User and System Stories: An Agile Approach for Managing Requirements in AOSE
6. B. Bauer et al: Agent UML: A Formalism for Specifying Multiagent Software Systems
7. A. Dardenne et al: Goal-directed requirements acquisition
8. J. Hübner: Um Modelo de Reorganização de Sistemas Multiagentes