

# RoboCup Rescue Agent Simulation League

VI Workshop-Escola de Sistemas de Agentes, seus Ambientes e aplicações



**Paulo Roberto Ferreira Jr.**

Centro de Desenvolvimento Tecnológico

Universidade Federal de Pelotas

[paulo.ferreira@ufpel.edu.br](mailto:paulo.ferreira@ufpel.edu.br)

**Luis Gustavo Nardin**

Laboratório de Técnicas Inteligentes

Universidade de São Paulo

[luis.nardin@usp.br](mailto:luis.nardin@usp.br)

# Roteiro

---

- ❑ **Introdução**
- ❑ **Simulador RoboCup Rescue**
- ❑ **Estrutura do Simulador**
- ❑ **Comportamento dos Agentes**
- ❑ **Comunicação dos Agentes**

# Roteiro

---

☐ **Introdução**

☐ **Simulador RoboCup Rescue**

Atividade Prática 1

☐ **Estrutura do Simulador**

☐ **Comportamento dos Agentes**

Atividade Prática 2

☐ **Comunicação dos Agentes**

Atividade Prática 3

# Introdução

## Desastres – Terremotos

Ano	Local	Magnitude	Mortos
2010	Haiti	7,0	316.000
1976	China	7,5	242.769
2004	Indonésia	9,1	227.898
1920	China	7,8	200.000
1923	Japão	7,9	142.800
1948	Turcomenistão	7,3	110.000
2008	China	7,9	87.587
2005	Paquistão	7,6	86.000
1908	Itália	7,2	72.000
1970	Peru	7,9	70.000

Os 10 terremotos mais mortais desde 1900



Porto Príncipe, 2010



China, 2008

# Introdução

---

- **Desastres** são situações de crise
  - Danos em larga escala
  - Recuperação limitada



# Introdução

- **Desastres** são situações de crise
  - Danos em larga escala
  - Recuperação limitada



## Limitações Complexas

- ✓ Informação **Incompleta**
- ✓ Decisões em **Tempo Real**
- ✓ **Grande** Quantidade de Agentes  
**Heterogêneos**

## Objetivos Complexos

- ✓ **Salvar** vidas
- ✓ **Prevenir** novos desastres
- ✓ **Coletar** dados
- ✓ **Planejar** a Curto/Longo prazo

# Introdução

---

Em 1999, Kitano et al. propuseram o simulador **RoboCup Rescue Simulation** para promover a pesquisa e o desenvolvimento de **políticas eficientes de resposta** a cenários de desastre

# Introdução

---

Em 1999, Kitano et al. propuseram o simulador **RoboCup Rescue Simulation** para promover a pesquisa e o desenvolvimento de **políticas eficientes de resposta** a cenários de desastre

## Objetivos

- Desenvolvimento de simuladores para reprodução fidedigna de cenários de desastre
- Desenvolvimento de técnicas de coordenação de tarefas entre múltiplos agentes e de tomada de decisão em tempo real para busca e resgate de vítimas



# Simulador RoboCup Rescue

---

- Introdução
- Características
- Arquitetura
- Dinâmica
- Compilação
- Execução
- Estrutura de Diretórios
- Configuração

# Simulador RoboCup Rescue

## Introdução

---

- RoboCup Rescue Simulation League
  - Agent Rescue
  - Virtual Robots
- Agent Rescue
  - Minimizar os danos causados aos civis e à infraestrutura em um determinado período de tempo



# Simulador RoboCup Rescue

## Características

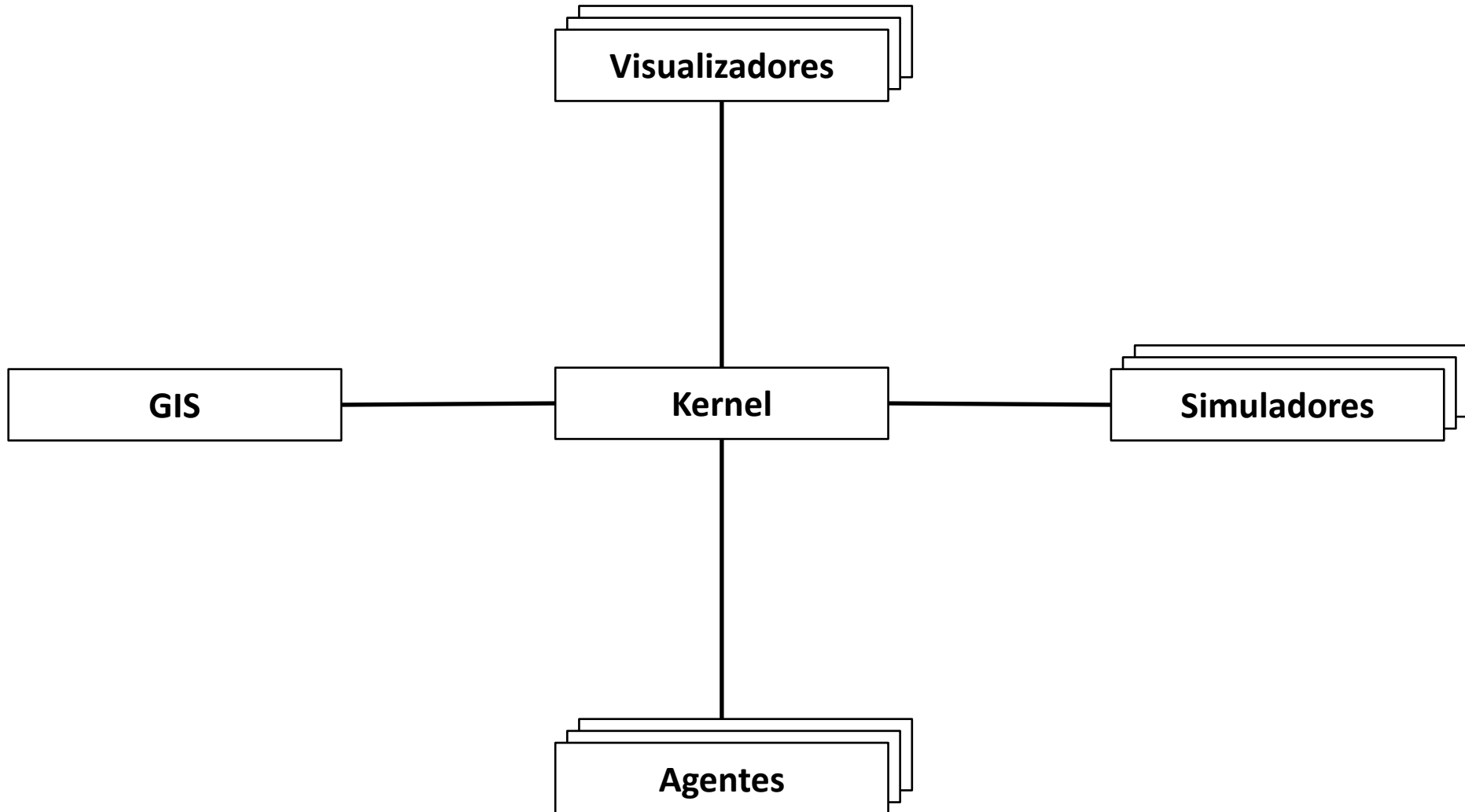
---

- Ambiente
  - Parcialmente observável
  - Dinâmico
  - Estocástico
- Comunicação
  - Restrita
  - Incerta

# Simulador RoboCup Rescue

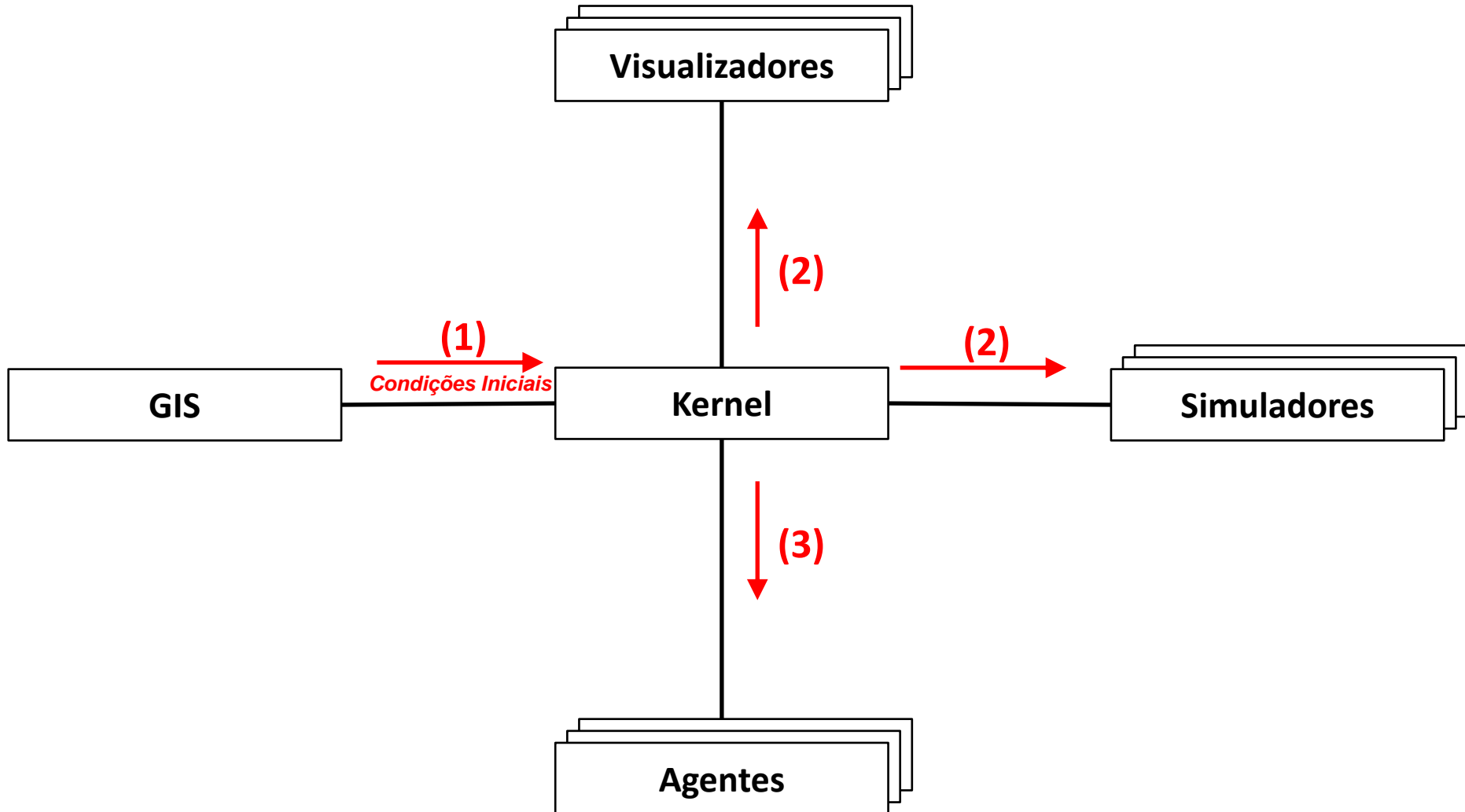
## Arquitectura

---



# Simulador RoboCup Rescue

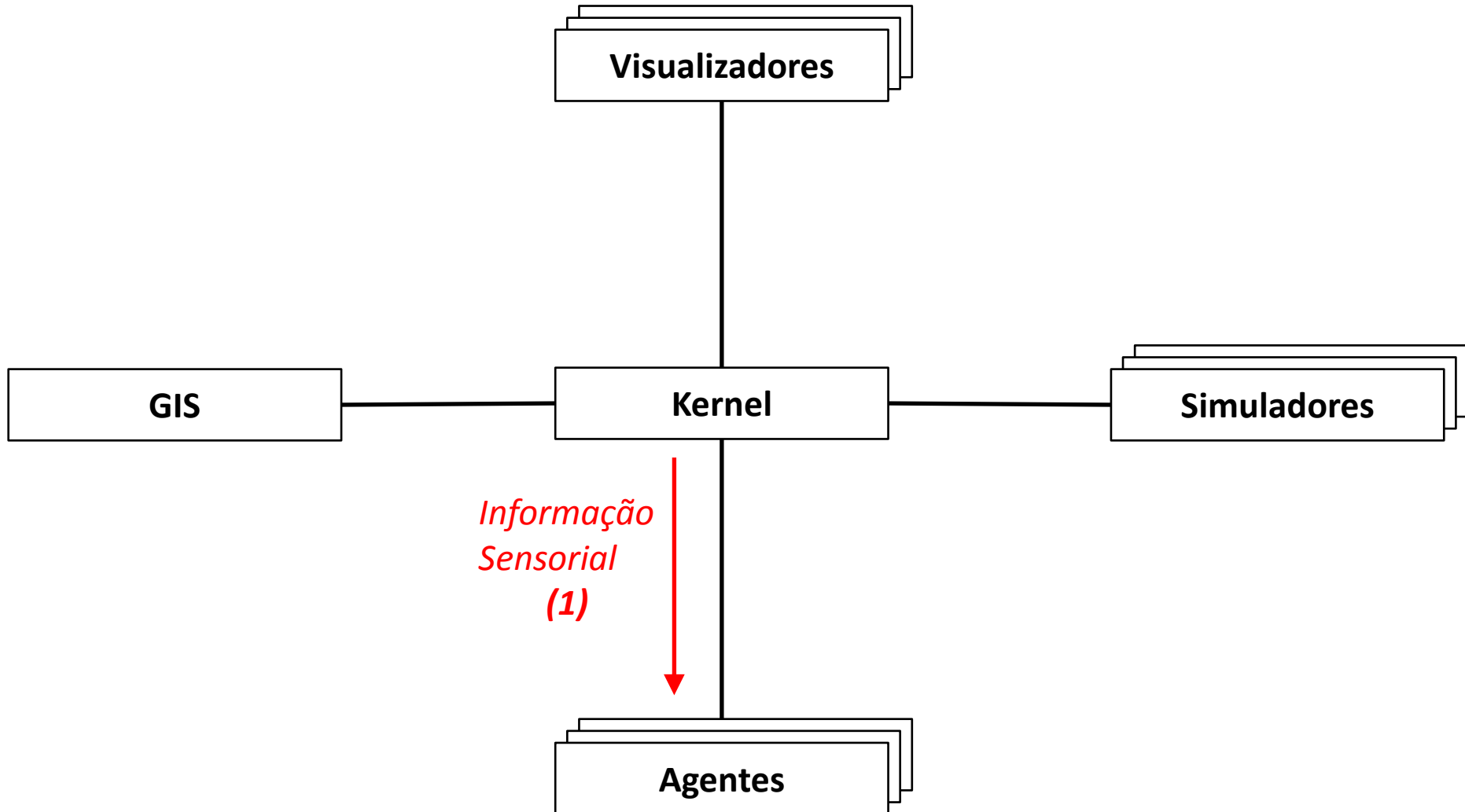
## Dinâmica – Inicialização



# Simulador RoboCup Rescue

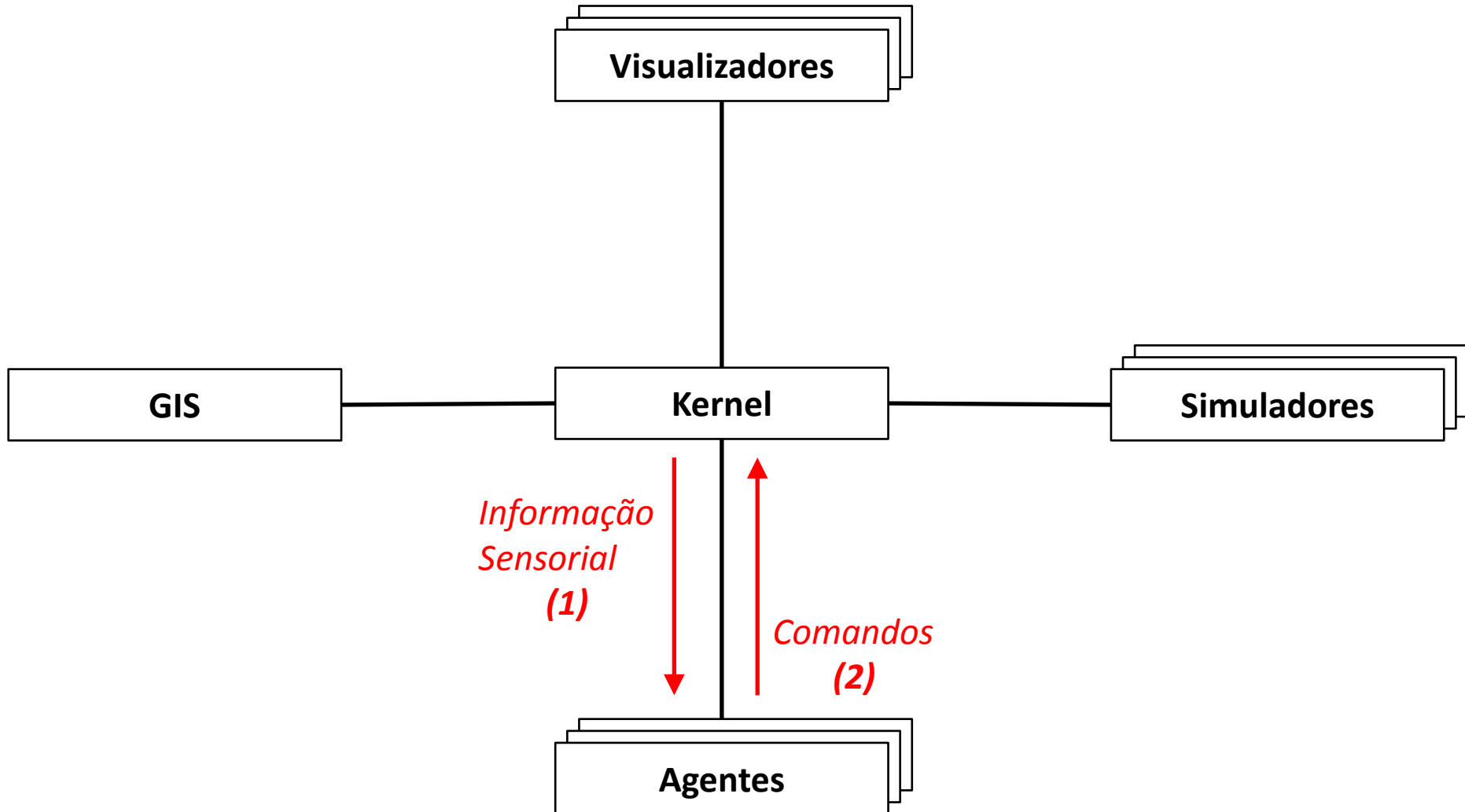
## Dinâmica – Simulação

---



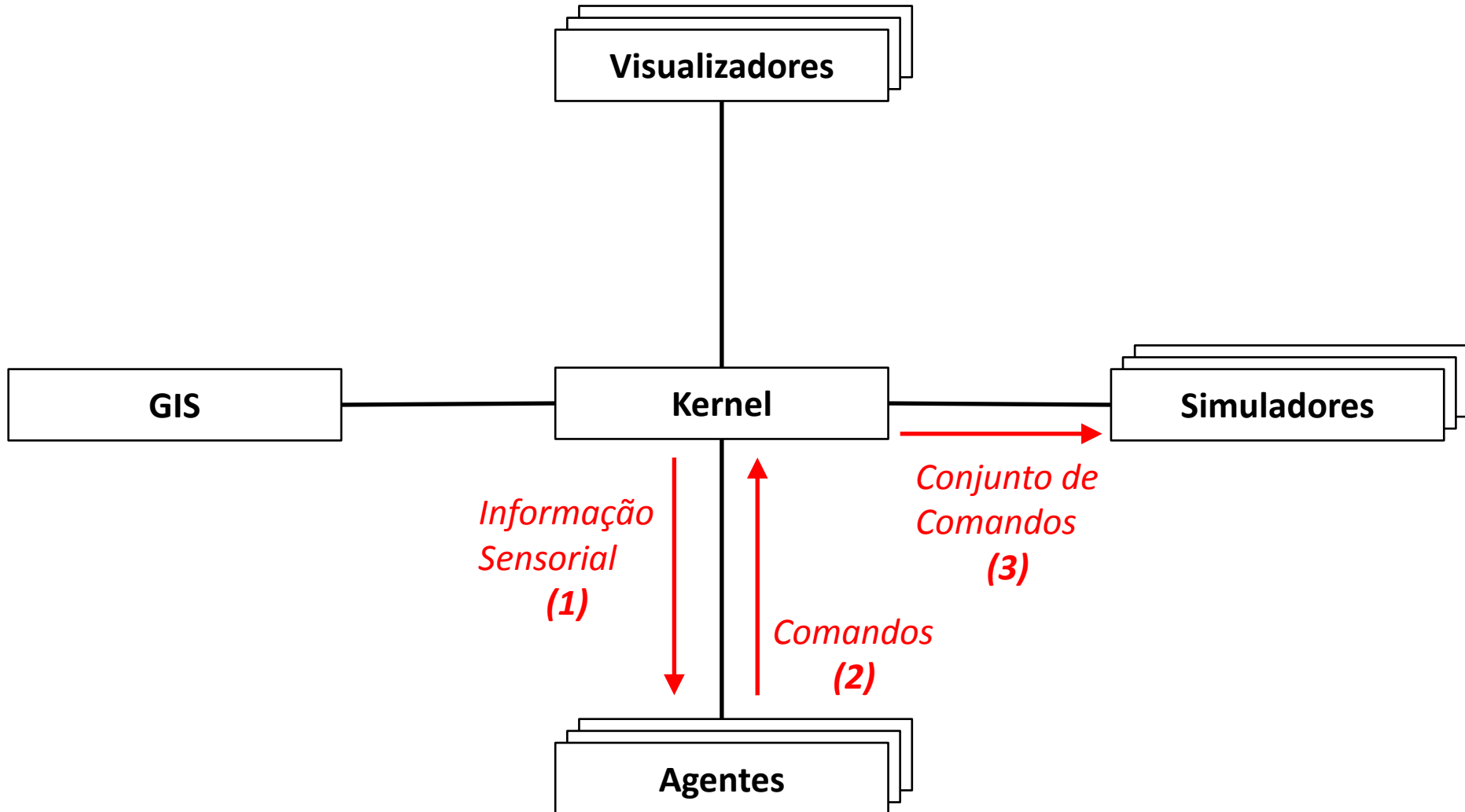
# Simulador RoboCup Rescue

## Dinâmica – Simulação



# Simulador RoboCup Rescue

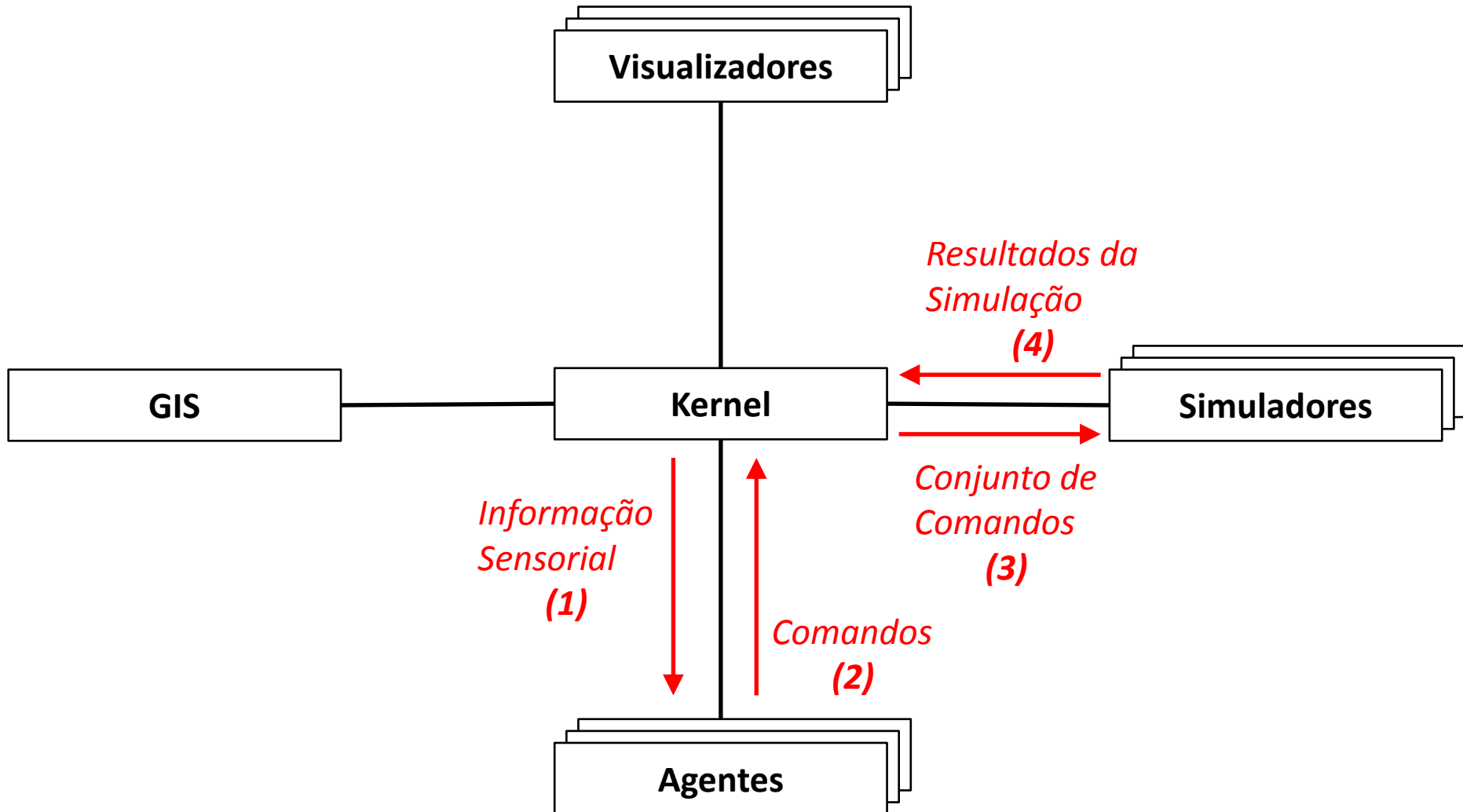
## Dinâmica – Simulação





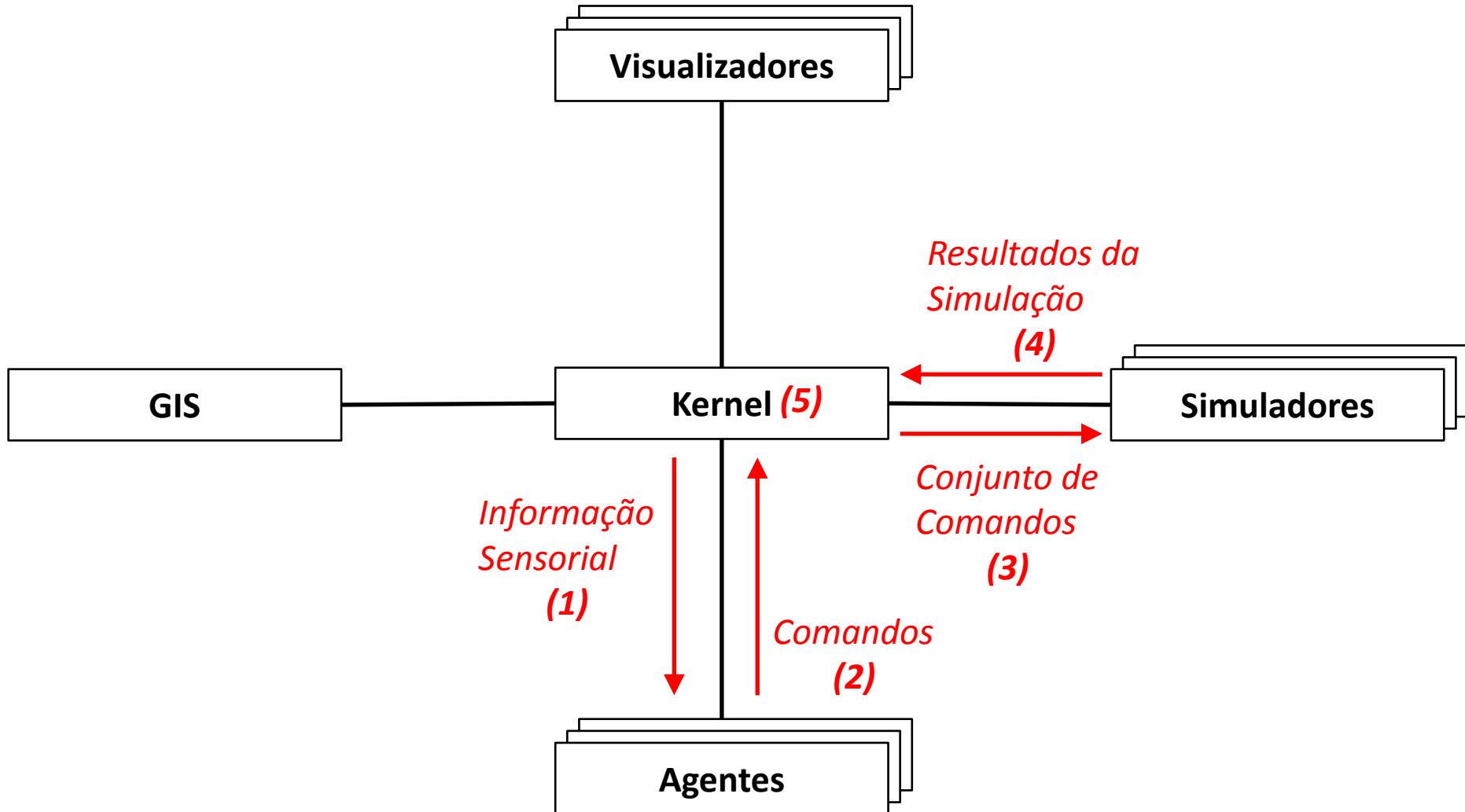
# Simulador RoboCup Rescue

## Dinâmica – Simulação



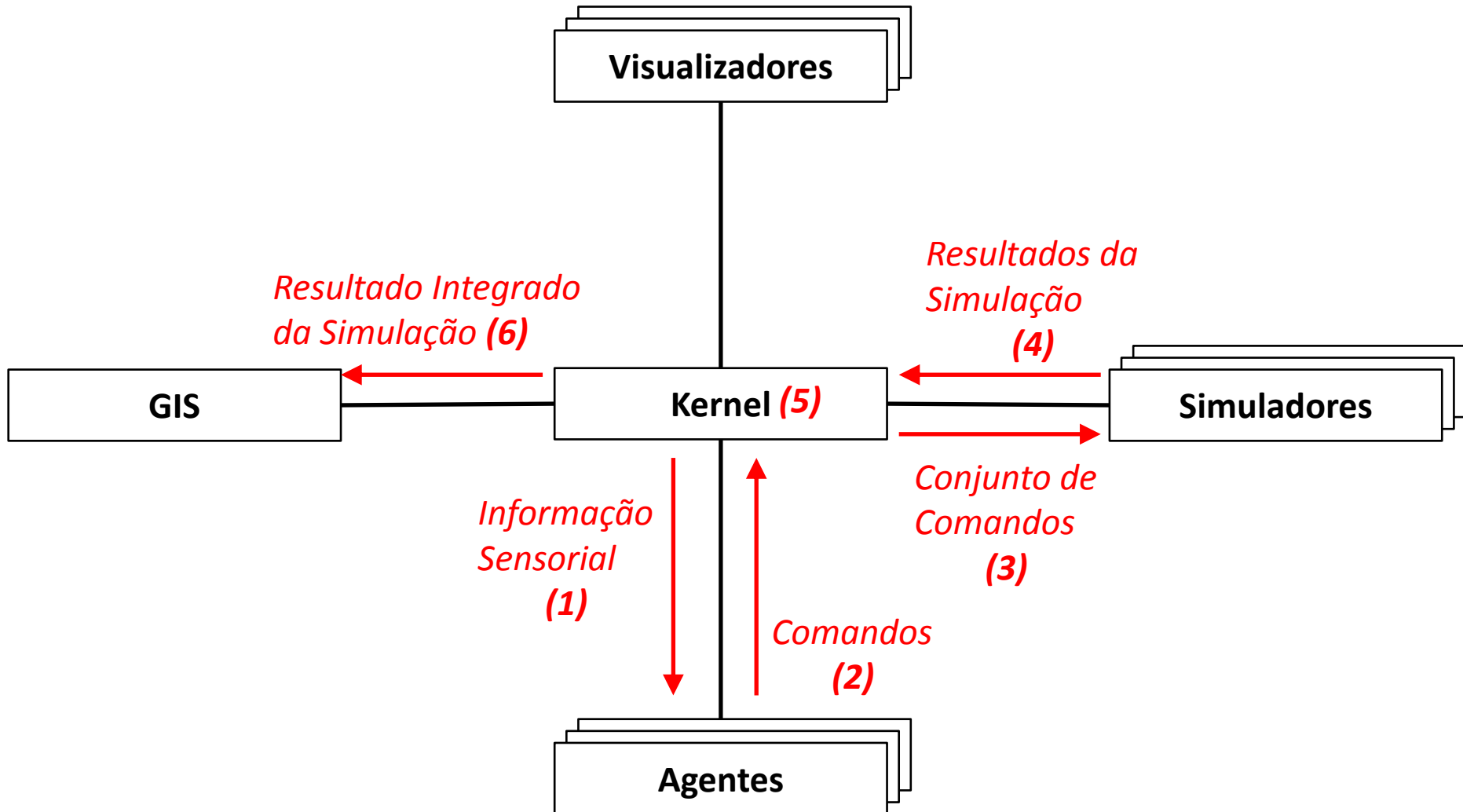
# Simulador RoboCup Rescue

## Dinâmica – Simulação



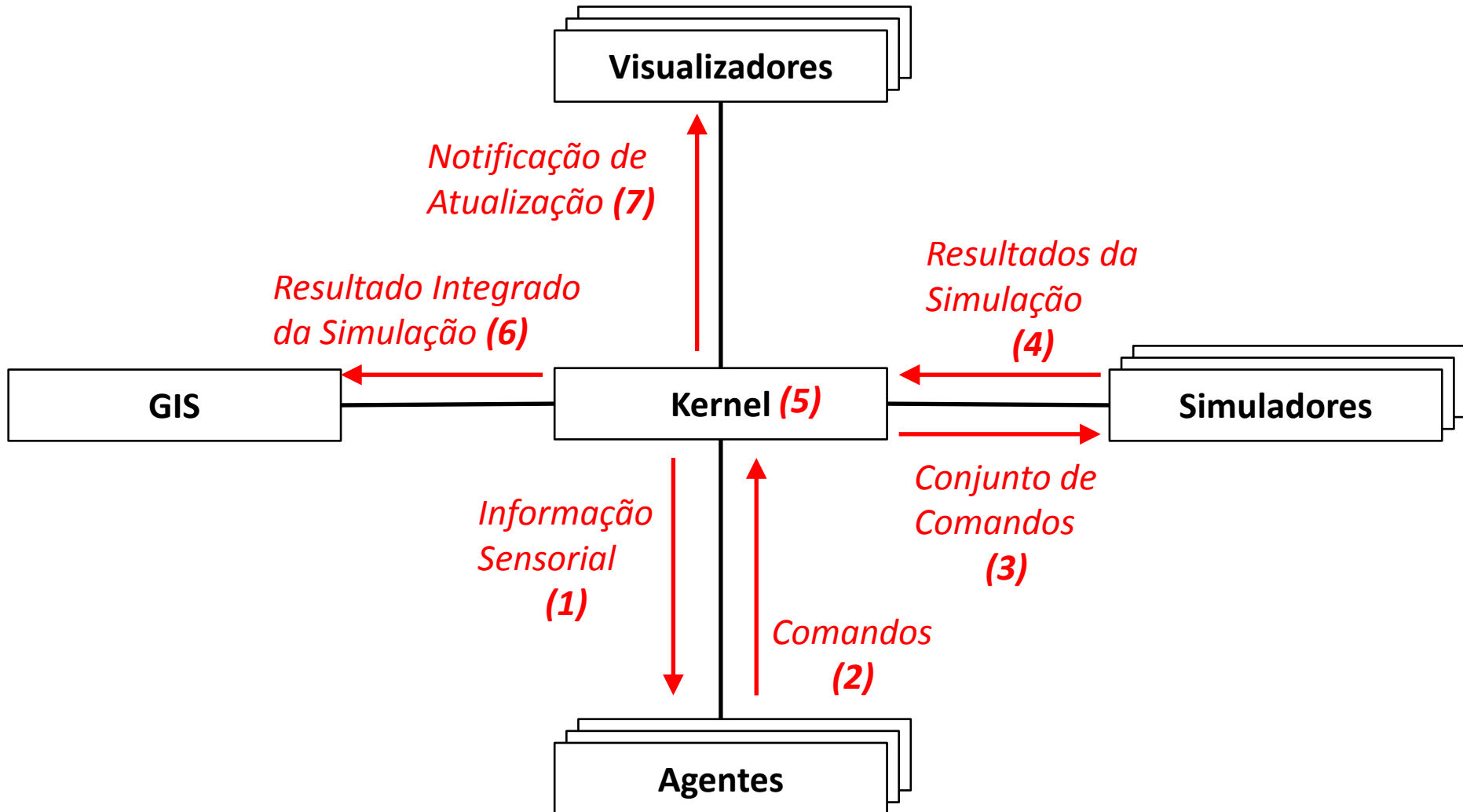
# Simulador RoboCup Rescue

## Dinâmica – Simulação



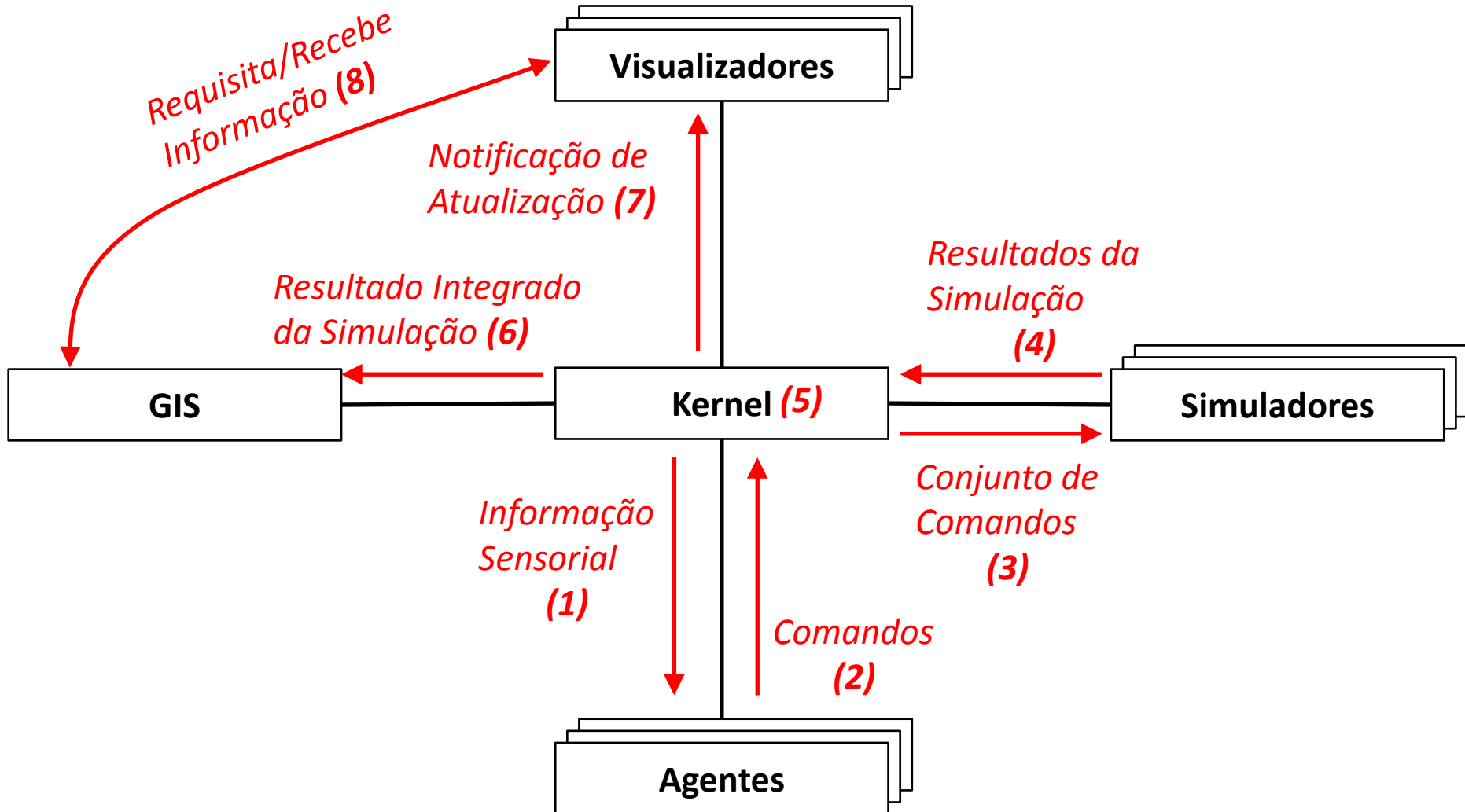
# Simulador RoboCup Rescue

## Dinâmica – Simulação



# Simulador RoboCup Rescue

## Dinâmica – Simulação



# Simulador RoboCup Rescue

## Compilação

---

- Download do RoboCup Rescue Simulador
  - <http://roborescue.sourceforge.net/>
    - Download -> Files -> 2011 -> rescue-1.0a-2011-src.tgz
- Software Necessário
  - Linux, Java JDK 1.6, ant 1.8+
- Compilar

```
$ tar -xvzf rescue-1.0a-2011-src.tgz
$ cd rescue-1.0a-2011
$ ant
```

# Simulador RoboCup Rescue

## Execução

---

- Executar o simulador

```
$ cd rescue-1.0a-2011/boot
```

onde `rescue-1.0a-2011` é o diretório do simulador

```
$ ./start.sh <opções>
```

<opções>

`-m <mapdir>` Define o diretório do mapa. Default:  
“../maps/gml/test”

`-l <logdir>` Define o diretório de log. Default: “logs”

`-s` Adicionar data e hora ao nome do diretório de log

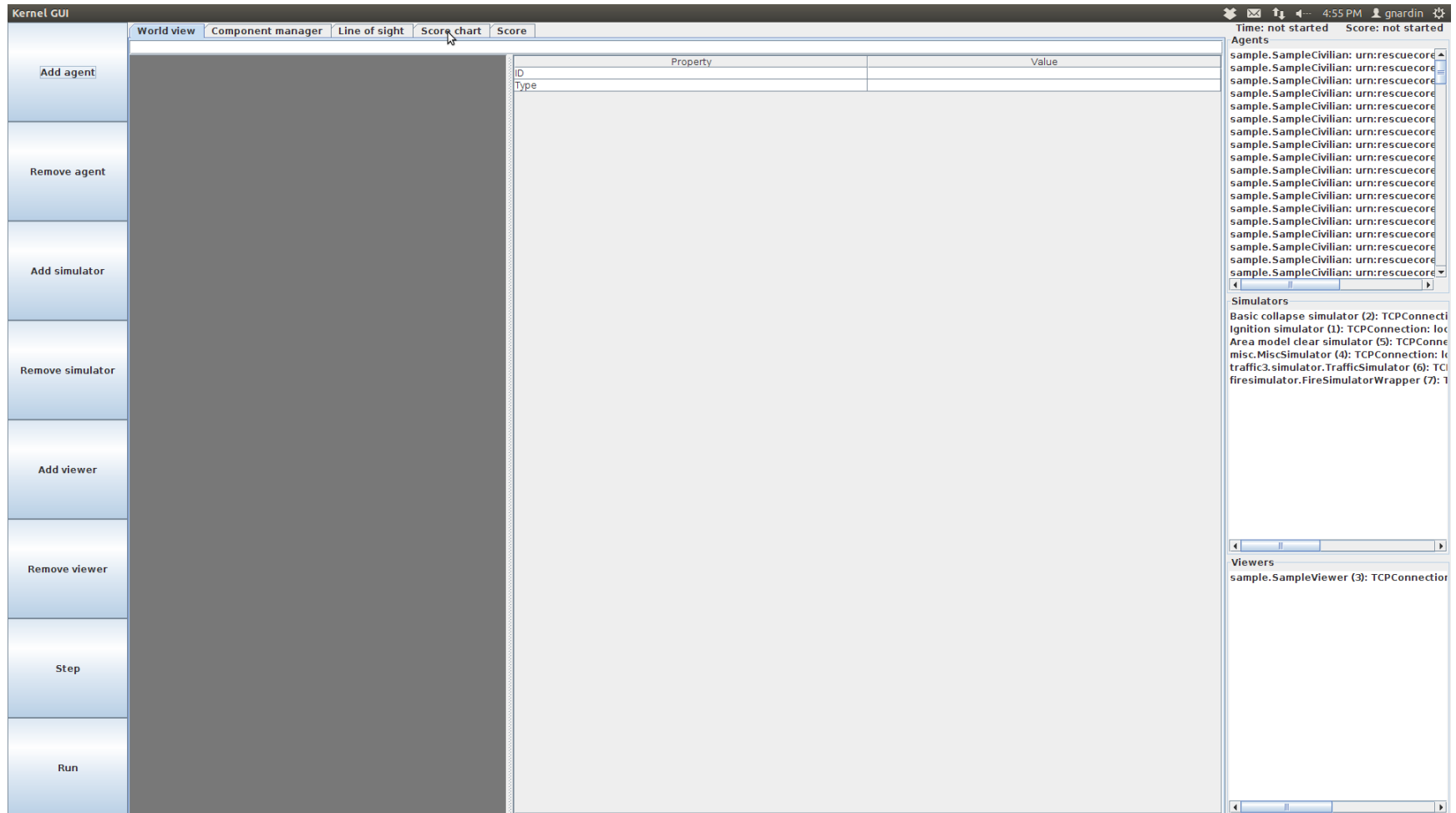
`-t <nometime>` Define nome do time. Default: ""

- Exemplo

```
$ ./start.sh -m ../maps/gml/paris
```

# Simulador RoboCup Rescue

## Execução





# Simulador RoboCup Rescue

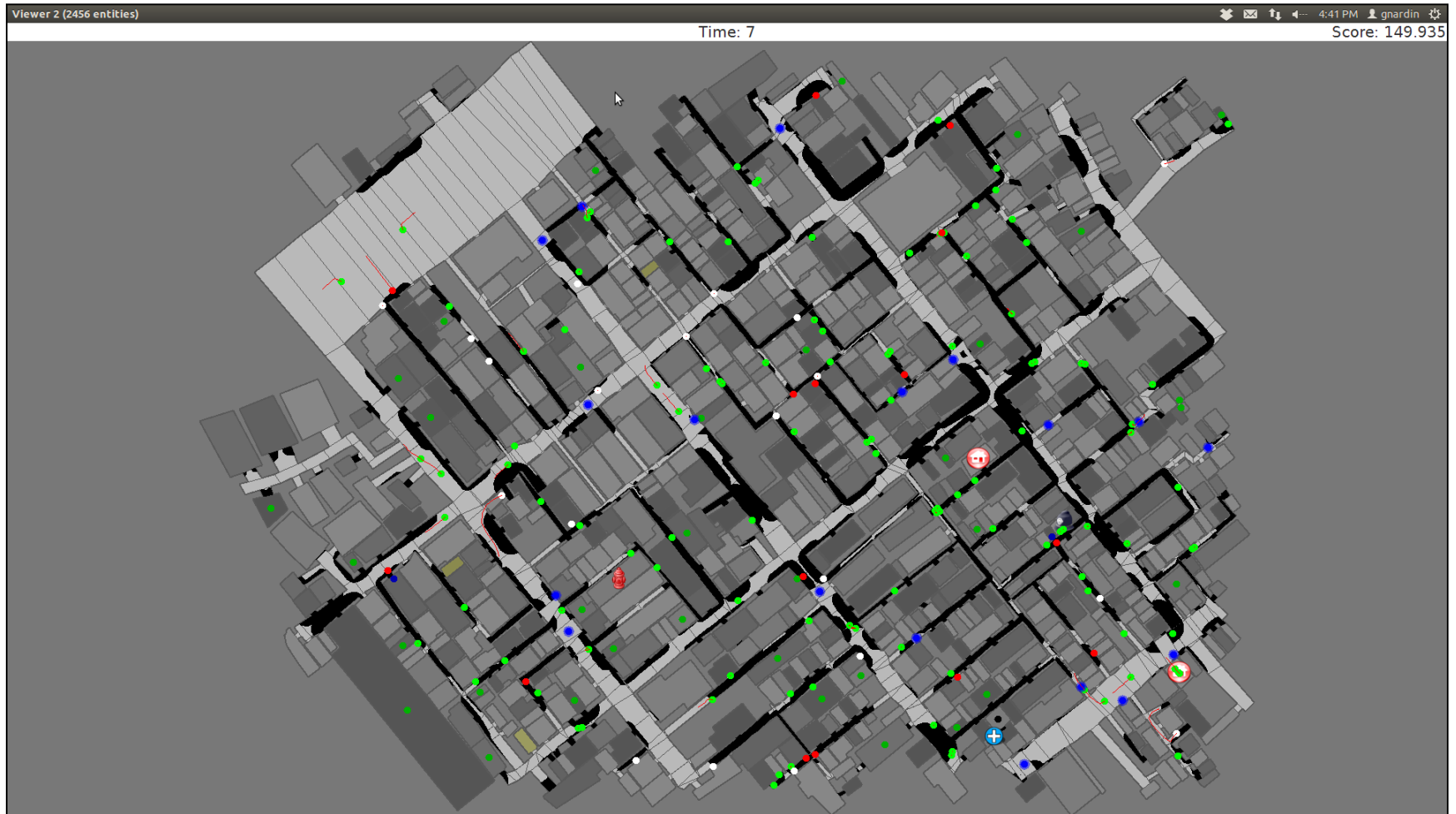
## Execução

---

- Executar os agentes, por exemplo
  - \$ cd rescue-1.0a-2011/boot
  - \$ ./sampleagent.sh
- Acionar o simulador
  - Selecionar o botão **Run**

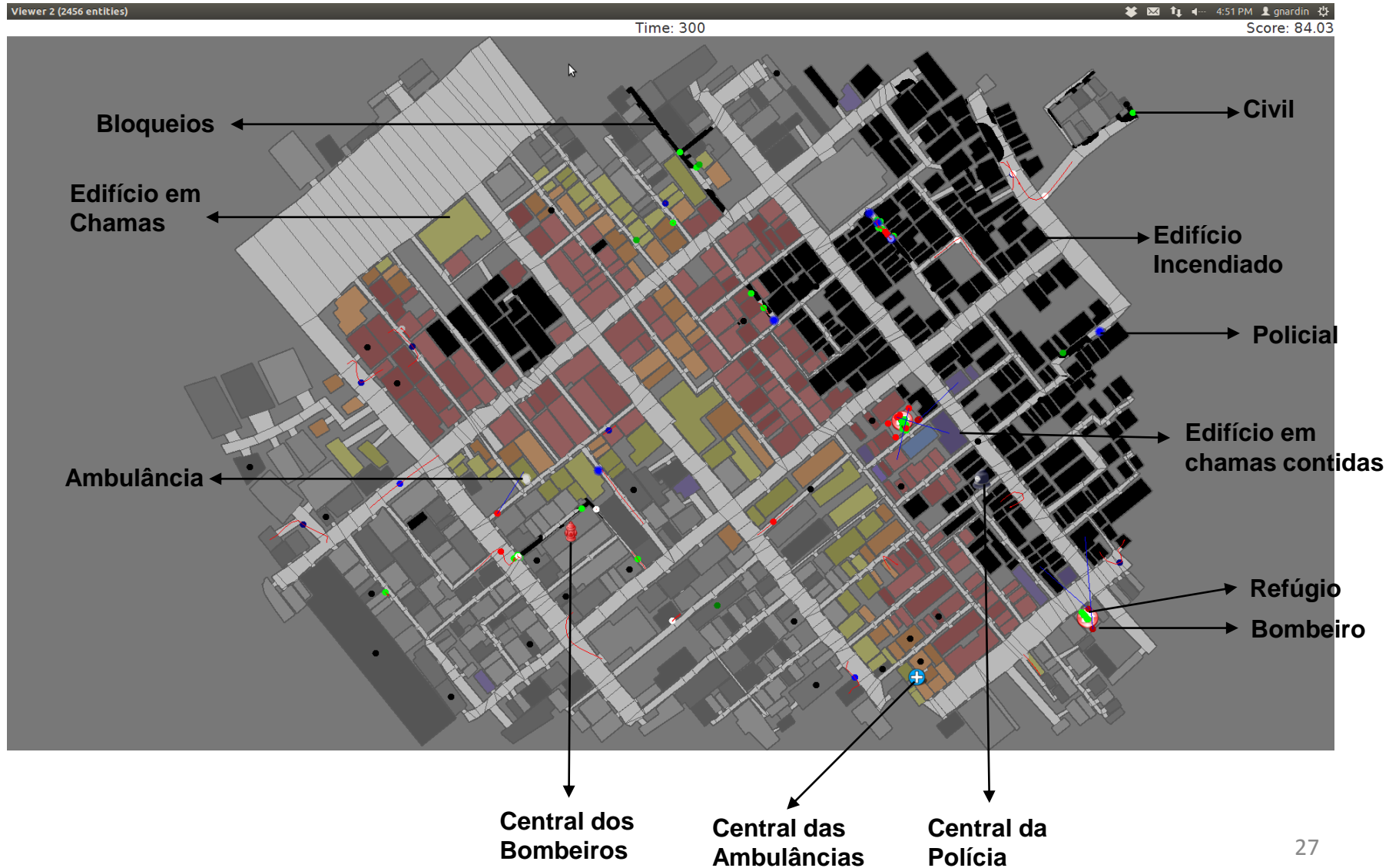
# Simulador RoboCup Rescue

## Execução



# Simulador RoboCup Rescue

## Execução



# Simulador RoboCup Rescue

## Execução

- Cálculo da Pontuação (Total Score)

$$V = \left( P + \frac{H}{Hint} \right) \times \sqrt{\frac{B}{Bmax}}$$

onde,

- **P** é o número de civis vivos
- **H** é a soma da saúde restante dos agentes
- **H<sub>int</sub>** é a soma da saúde de todos os agentes
- **B** é a soma da área de cada edifício \* fator
- **B<sub>max</sub>** é a soma da área de todos os edifícios

Se Fieryness = 0  
**fator = 1**

Se Fieryness = 1, 4 ou 5  
**fator = 0,66**

Se Fieryness = 2 ou 6  
**fator = 0,33**

Se Fieryness = 3, 7 ou 8  
**fator = 0**

# Simulador RoboCup Rescue

## Estrutura de Diretórios

---

<b>/boot</b>	scripts de execução do simulador
<b>/boot/config</b>	arquivos de configuração dos simuladores
<b>/boot/logs</b>	arquivos de log
<b>/build</b>	classes Java do simulador
<b>/docs</b>	documentação <i>javadoc</i> do simulador
<b>/jars</b>	arquivos JARs do simulador
<b>/lib</b>	bibliotecas usadas pelo simulador
<b>/maps</b>	mapas do simulador
<b>/modules</b>	arquivos fonte do simulador
<b>/oldsims</b>	arquivos fonte do simulador antigo

# Simulador RoboCup Rescue

## Configuração

---

- Os arquivos de configuração do simulador estão localizados no diretório **/boot/config**
- Alguns arquivos e parâmetros de configuração do simulador são
  - **common.cfg**
    - random.seed: 1
    - kernel.host: localhost
    - kernel.port: 7000
  - **kernel.cfg**
    - kernel.timesteps: 300
    - kernel.startup.connect-time: 300000

# Simulador RoboCup Rescue

## Configuração

---

- Alguns arquivos e parâmetros de configuração do simulador são (continuação)
  - **ignition.cfg**
    - ignition.random.lambda: 0.05
  - **perception.cfg**
    - perception.los.max\_distance: 30000
  - **resq-fire.cfg**
    - fire.tank.maximum: 7500
    - fire.tank.refill\_rate: 500
    - fire.extinguish.max\_distance: 50000

# Simulador RoboCup Rescue

## Configuração

---

- Alguns arquivos e parâmetros de configuração do simulador são (continuação)
  - **commsXXX.cfg**
    - comms.channels.count: 3
    - comms.channels.max.platoon: 2
    - comms.channels.max.centre: 2
  
    - comms.channels.0.type: **voice**
    - comms.channels.0.range: 30000
    - comms.channels.0.messages.size: 256
    - comms.channels.0.messages.max: 1
    - comms.channels.0.noise.input.dropout.use: yes
    - comms.channels.0.noise.input.dropout.p: 0.1



# Simulador RoboCup Rescue

## Configuração

---

- Alguns arquivos e parâmetros de configuração do simulador são (continuação)
  - **commsXXXX.cfg**
    - `comms.channels.1.type`: **radio**
    - `comms.channels.1.bandwidth`: 3000
    - `comms.channels.1.noise.input.failure.use`: yes
    - `comms.channels.1.noise.input.failure.p`: 0.2
    - `comms.channels.1.noise.input.dropout.use`: yes
    - `comms.channels.1.noise.input.dropout.p`: 0.2

# Atividade Prática 1

---

1. Executar o simulador usando as configurações padrão e o time de agentes *SampleAgent*
2. Executar o simulador usando o mapa **Berlin** que encontra-se no diretório **rescue-1.0a-2011/maps/gml/berlin**. Em seguida, executar o time de agentes *SampleAgent*
3. Alterar a quantidade de ciclos (timesteps) de simulação de **300** para **200**. Executar o simulador usando o mapa padrão e o time de agentes *SampleAgent*

# Estrutura do Simulador

---

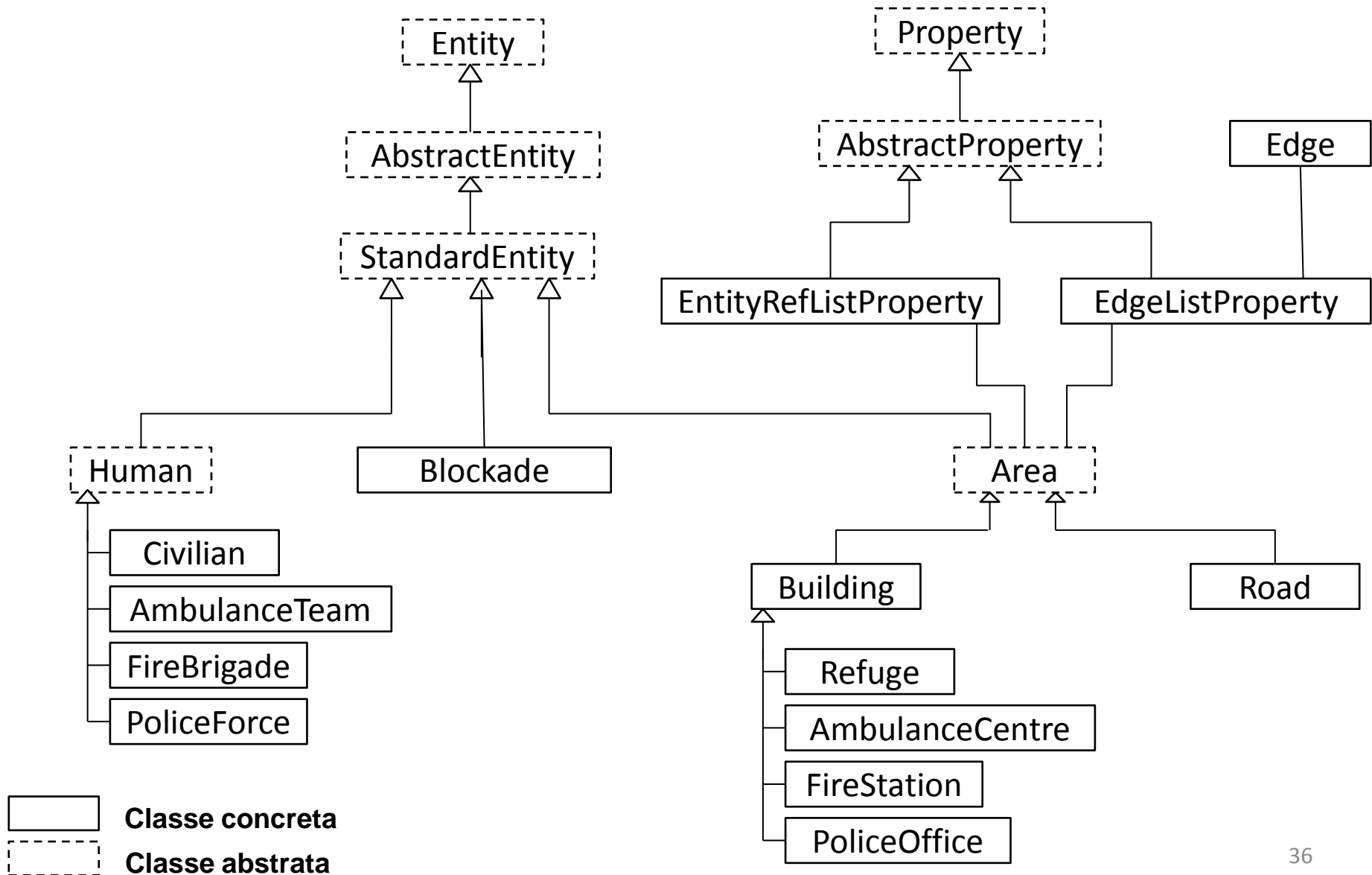
- ❑ Hierarquia de Classes

- ❑ Objetos

- World
- Building
- Road
- Blockade

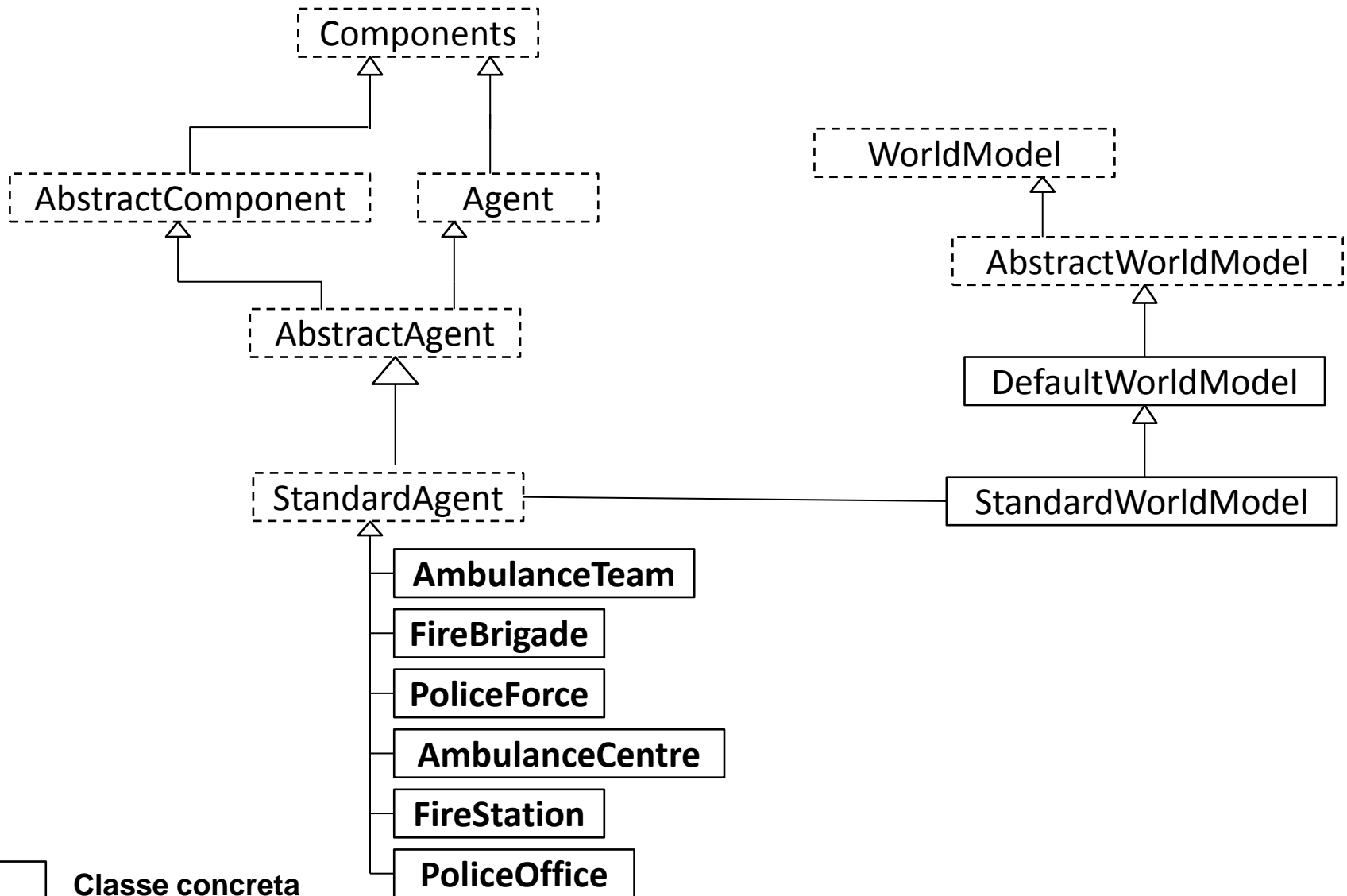
# Estrutura do Simulador

## Hierarquia de Classes



# Estrutura do Simulador

## Hierarquia de Classes

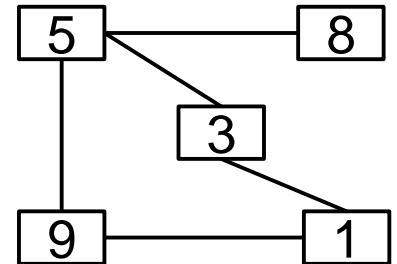


# Estrutura do Simulador

## Objetos – World

- **World**

- Representação do mundo simulado
- Composto por um conjunto de entidades, como Buildings, Roads e Humans
- As entidades **Road** e **Building** formam um grafo conexo
- É instância da classe **StandardWorldModel**
- Acessada por meio do atributo global **model**



- Métodos úteis da classe **StandardWorldModel**

`int getDistance(EntityID first, EntityID second)`

`int getDistance(StandardEntity first, StandardEntity second)`

Retorna a distância euclidiana entre as duas entidades

# Estrutura do Simulador

## Objetos – World

---

- Métodos úteis da classe **StandardWorldModel**

`Collection<StandardEntity> getEntitiesOfType(StandardEntityURN...urns)`

Retorna o conjunto de entidades de um ou mais determinado tipo

`Collection<StandardEntity> getObjectsInRange(EntityID entity, int range)`

`Collection<StandardEntity> getObjectsInRange(StandardEntity entity, int range)`

Retorna o conjunto de entidades dentro do raio *range*

`Collection<StandardEntity> getObjectsInRectangle(int x1, int y1, int x2, int y2)`

Retorna o conjunto de entidades circunscritas pelas coordenadas (x1, y1) (x2, y2)

# Estrutura do Simulador

## Objetos – Building

---

- **Building**
  - Representa os edifícios
  - Estende a classe **Area**
  - É instância da classe **Building**
  
- **Propriedades**

Propriedade	Descrição
ID	Número de identificação do edifício
Brokenness	Indica o quanto a construção está danificada
Fieryness	Indica a intensidade do incêndio
Temperature	Indica a temperatura do edifício
TotalArea	Área total da construção incluindo todos os andares
Blockades	Lista de bloqueios existentes na área do edifício



# Estrutura do Simulador

## Objetos – Building

---

- Métodos úteis da classe **Building**

`List<EntityID> getNeighbours()`

Retorna a lista dos objetos vizinhos a esse edifício

`int getBrokenness()`

Retorna o quanto o edifício está danificado

`int getFieryness()`

Retorna o valor da intensidade do incêndio conforme tabela

0 → UNBURNT	5 → MINOR_DAMAGE
1 → HEATING	6 → MODERATE_DAMAGE
2 → BURNING	7 → SEVERE_DAMAGE
3 → INFERNO	8 → BURNT_OUT
4 → WATER_DAMAGE	

# Estrutura do Simulador

## Objetos – Building

---

- Métodos úteis da classe **Building**

`int getTemperature()`

Retorna o valor da temperatura do edifício

`int getTotalArea()`

Retorna a área total do edifício que inclui a área de todos os andares do edifício

`int getBuildingCode()`

Retorna o código do tipo de material da construção

Código	Tipo	Taxa de Transmissão
0	Madeira	1,8
1	Aço	1,8
2	Concreto	1,0

# Estrutura do Simulador

## Objetos – Building

---

- Métodos úteis da classe **Building**

`List<EntityID> getBlockades()`

Retorna a lista de bloqueios na área do edifício

# Estrutura do Simulador

## Objetos – Road

---

- **Road**
  - Representa as vias de navegação
  - Estende a classe **Area**
  - É instância da classe **Road**
- **Propriedades**

Propriedade	Descrição
ID	Número de identificação da via
Blockades	Lista de bloqueios existentes na área da via

# Estrutura do Simulador

## Objetos – Road

---

- Métodos úteis da classe **Road**

`List<EntityID> getBlockades()`

Retorna a lista de bloqueios na área da via

`List<EntityID> getNeighbours()`

Retorna a lista dos objetos vizinhos a essa via

# Estrutura do Simulador

## Objetos – Blockade

---

- **Blockade**

- Representa os bloqueios
- Composto por um conjunto de propriedades
- É instância da classe **Blockade**

- **Propriedades**

Propriedade	Descrição
ID	Número de identificação do bloqueio
Position	Entidade sobre a qual o bloqueio está posicionado
RepairCost	Custo de reparo do bloqueio

# Estrutura do Simulador

## Objetos – Blockade

---

- Métodos úteis da classe **Blockade**

`EntityID getPosition()`

Retorna o EntityID da entidade sobre a qual o bloqueio está posicionado

`int getRepairCost()`

Retorna o valor de reparo do bloqueio, quanto maior o valor, mais tempo será necessário para removê-lo

# Comportamento dos Agentes

---



- Introdução
- Capacidade
- Police Force
- Ambulance Team
- Fire Brigade



# Comportamento dos Agentes

## Introdução

---

- Existem dois tipos de agentes
  - Civis (Civilian   )
  - Agentes de Resgate

### Platoon (Móveis)

-  Ambulance Team
-  Fire Brigade
-  Police Force

### Center (Fixos)

- Ambulance Center
- Fire Station
- Police Office

# Comportamento dos Agentes

## Introdução

---

- Estrutura mínima da classe que implementa o agente

```
public class [NOME_CLASSE_AGENTE] extends StandardAgent<[StandardEntity]>{

    @Override
    protected EnumSet<StandardEntityURN> getRequestedEntityURNsEnum() {
        return EnumSet.of(StandardEntityURN.[StandardEntityURN]);
    }

    @Override
    protected void postConnect() {
    }

    @Override
    protected void think(int time, ChangeSet changed, Collection<Command>
heard) {
    }
}
```

# Comportamento dos Agentes

## Introdução

---

- **StandardEntityURN**

- CIVILIAN
- AMBULANCE\_TEAM
- AMBULANCE\_CENTRE
- FIRE\_BRIGADE
- FIRE\_STATION
- POLICE\_FORCE
- POLICE\_OFFICE

- **StandardEntity**

- Civilian
- AmbulanceTeam
- AmbulanceCentre
- FireBrigade
- FireStation
- PoliceForce
- PoliceOffice

# Comportamento dos Agentes

## Introdução

---

- `protected EnumSet<StandardEntityURN> getRequestedEntityURNsEnum()`

Método que retorna o tipo da entidade implementada por essa classe

- `protected void postConnect()`

Método executado uma única vez após a conexão do agente ao *kernel* da plataforma de simulação e antes do início da simulação.

Utilizado para realizar o pré-processamento das informações recebidas do simulador antes do início da simulação

Todos os agentes têm que finalizar a execução do método **postConnect** após um tempo limite (padrão 5 minutos).

# Comportamento dos Agentes

## Introdução

---

- `protected void think(int time, ChangeSet changed, Collection<Command> heard)`

Método que implementa o funcionamento do agente e é chamado pelo *kernel* a cada ciclo de simulação.

Esse método tem um tempo limite a ser executado (padrão 30 segundos).

# Comportamento dos Agentes

## Introdução

---

- Acessando os valores dos parâmetros dos arquivos de configuração

```
this.config.getIntValue([key])
```

onde, **[key]** é o nome do parâmetro especificado em algum arquivo de configuração

- Exemplo

```
this.config.getIntValue("perception.los.max_distance")
```

```
this.config.getIntValue("fire.extinguish.max_distance")
```

# Comportamento dos Agentes

## Introdução

---

- Exemplo de agente **AmbulanceTeam**

```
public class ExemploAT extends StandardAgent<AmbulanceTeam>{

    @Override
    protected EnumSet<StandardEntityURN> getRequestedEntityURNsEnum() {
        return EnumSet.of(StandardEntityURN.AMBULANCE_TEAM);
    }

    @Override
    protected void postConnect() {
        int max_distance =
            this.config.getIntValue("perception.los.max_distance");
        ...
    }

    @Override
    protected void think(int time, ChangeSet changed, Collection<Command>
heard) {
        ...
    }
}
```

# Comportamento dos Agentes

## Introdução

- Todos os agentes estendem a classe **StandardAgent** e realizam o controle de uma subclasse da classe **StandardEntity** (especificamente da classe **Human**)
- Propriedades

Propriedade	Descrição
ID	Número de identificação do agente
X	Coordenada X do agente no mapa
Y	Coordenada Y do agente no mapa
Buriedness	Indica o quanto o agente está soterrado
HP	Indica o quanto o agente tem de saudável
Damage	Taxa de diminuição do valor do HP
Position	Entidade sobre a qual o agente está posicionado



# Comportamento dos Agentes

## Introdução

---

- Métodos úteis da classe **Human**

`int getBuriedness()`

Retorna o valor de quanto o agente está soterrado

`int getHP()`

Retorna o quanto o agente está saudável, quanto maior melhor. Quando esse valor chega a 0 (zero), o agente morre

`int getDamage()`

Retorna a taxa de diminuição do valor do HP do agente

`EntityID getPosition()`

Retorna o EntityID da entidade sobre a qual o agente está posicionado

# Comportamento dos Agentes

## Introdução

---

- Métodos úteis da classe **Human**

`Pair<Integer,Integer> getLocation(WorldModel<? extends StandardEntity>  
world)`

Retorna a localização X e Y do agente no mapa

# Comportamento dos Agentes

## Capacidades

---

Tipo	Capacidade
Civilian	Sentir, ouvir, dizer, mover
Ambulance Team	Sentir, ouvir, dizer, mover, comunicar-se via rádio, resgatar, carregar, descarregar
Fire Brigade	Sentir, ouvir, dizer, mover, comunicar-se via rádio, extinguir, encher o tanque
Police Force	Sentir, ouvir, dizer, mover, comunicar-se via rádio, limpar
Ambulance Centre	Ouvir, comunicar-se via rádio
Fire Station	Ouvir, comunicar-se via rádio
Police Office	Ouvir, comunicar-se via rádio

# Comportamento dos Agentes

## Capacidades

---

- **Sentir**

Essa capacidade possibilita ao agente perceber o ambiente delimitado por seu raio de visão. Essas percepções são recebidas pelo agente por meio do parâmetro **changed** (classe **ChangeSet**) do método **think**.

A chave **perception.los.max\_distance** que define o raio de visão dos agentes é especificada no arquivo **perception.cfg**

### Método da classe **ChangeSet**

– **Set<EntityID> getChangedEntities()**

Retorna o conjunto de identificação de todas as entidades que sofreram alguma alteração desde o último ciclo

# Comportamento dos Agentes

## Capacidades

---

- **Ouvir**

Essa capacidade possibilita ao agente receber mensagens de outros agentes por meio de comunicação. As mensagens são recebidas pelo agente como um conjunto pelo parâmetro **heard** (classe **Command**) do método **think**.

Maiores detalhes sobre essas capacidades serão apresentados na seção de *Comunicação dos Agentes*

# Comportamento dos Agentes

## Capacidades

---

- **Dizer (Say)**

Essa capacidade possibilita ao agente transmitir uma mensagem de voz de curta distância.

- **Comunicar-se via radio (Speak)**

Essa capacidade possibilita ao agente transmitir uma mensagem via rádio de longa distância.

Maiores detalhes sobre essas capacidades serão apresentados na seção de *Comunicação dos Agentes*

# Comportamento dos Agentes

## Capacidades

---

- **Mover**

Essa capacidade possibilita fazer com que o agente se desloque no ambiente.

### Método/Comando

**void sendMove(int time, List<EntityID> path)**

Comando usado para movimentar o agente por uma sequência de entidades interconectadas no grafo

**void sendMove(int time, List<EntityID> path, int destX, int destY)**

Comando usado para movimentar o agente por uma sequência de entidades interconectadas no grafo ou até as coordenadas X e Y do mapa

# Comportamento dos Agentes

## Police Force

---

- **Limpar**

Essa capacidade possibilita ao policial limpar um determinado bloqueio.

### Método/Comando

`void sendClear(int time, EntityID target)`

Comando usado para o agente limpar um bloqueio (**target**) especificado



# Comportamento dos Agentes

## Fire Brigade

---

- **Extinguir**

Essa capacidade possibilita ao bombeiro jogar água em um edifício.

### Método/Comando

`void sendExtinguish(int time, EntityID target, int power)`

Comando usado para o agente jogar uma quantidade específica de água (**power**) no edifício (**target**)

# Comportamento dos Agentes

## Fire Brigade

---

- **Encher o tanque**

Essa capacidade possibilita ao bombeiro ficar parado sobre o refúgio enquanto enche seu tanque de água.

### Método/Comando

**void sendRest(int time)**

Comando usado para o agente aguardar parado sobre o refúgio enquanto enche seu tanque de água

# Comportamento dos Agentes

## Ambulance Team

---

- **Resgatar**

Essa capacidade possibilita à ambulância desenterrar um outro agente soterrado.

### Método/Comando

**void sendRescue(int time, EntityID target)**

Comando usado para o agente desenterrar um outro agente soterrado (**target**)

# Comportamento dos Agentes

## Ambulance Team

---

- **Carregar**

Essa capacidade possibilita à ambulância carregar um agente para transportá-lo a outra localidade.

### Método/Comando

**void sendLoad(int time, EntityID target)**

Comando usado para o agente carregar um outro agente  
(**target**)

# Comportamento dos Agentes

## Ambulance Team

---

- **Descarregar**

Essa capacidade possibilita a ambulância descarregar o agente que está carregando.

### Método/Comando

**void sendUnload(int time)**

Comando usado para o agente descarregar o agente que está carregando

# Atividade Prática 2

---

1. Executar o simulador usando as configurações padrão e o time de agentes que se encontra no projeto **libraryAgentSample**, classe

*jp.ac.nagoyau.is.ss.kishii.ob.launch.LaunchSampleAgents*

2. Alterar o comportamento do agente **FireBrigade** no projeto **libraryAgentSample** para que ele abandone uma determinada tarefa se ele ficar bloqueado em uma posição por mais de  $n$  ciclos, sendo  $n$  uma constante.

*jp.ac.nagoyau.is.ss.kishii.ob.sample.SampleFireBrigade*

# Comunicação dos Agentes

---

- ❑ Estrutura de Comunicação
- ❑ Métodos de Comunicação
- ❑ Biblioteca RCRSCS
  - Dinâmica
  - Uso
  - Mensagens

# Comunicação dos Agentes

## Estrutura de Comunicação

---

- Existem dois tipos de comunicação
  - **Voz**
    - Possui limite de distância para sua recepção por outros agentes
    - Canal único de comunicação (id 0)
  - **Rádio**
    - Não possui limite de distância para comunicação
    - Necessita da adesão a um canal de comunicação para seu uso
    - Há um limite máximo de canais aos quais pode se aderir



# Comunicação dos Agentes

## Métodos de Comunicação

---

- **Ouvir**

Essa capacidade possibilita ao agente receber mensagens de outros agentes por meio de comunicação. As mensagens são recebidas pelo agente como um conjunto pelo parâmetro **heard** (classe **Command**) do método **think**.

# Comunicação dos Agentes

## Métodos de Comunicação

---

- **Dizer (Say)**

Essa capacidade possibilita ao agente transmitir uma mensagem de voz de curta distância.

A distância padrão de transmissão desse tipo de comunicação é 30 metros.

### Método/Comando

```
void sendSay(int time, byte[] data)
```

Comando usado para transmitir uma mensagem de voz no ambiente.

# Comunicação dos Agentes

## Métodos de Comunicação

---

- **Comunicar-se via radio (Speak)**

Essa capacidade possibilita ao agente transmitir uma mensagem via rádio em qualquer canal de comunicação. Porém, para que seja possível a recepção de mensagens é necessário que o agente tenha se inscrito ao canal.

### Método/Comando

**void sendSubscribe(int time, int... channels)**

Comando usado para o agente se inscrever a um ou mais canais de comunicação.

**void sendSpeak(int time, int channel, byte[] data)**

Comando usado para transmitir uma mensagem de rádio para os outros agentes.

# Biblioteca RCRSCS

---

- Perceba que **NÃO** há uma definição do formato das mensagens, elas são definidas como **byte[]**
- É necessário que cada equipe defina o formato das mensagens trocadas entre seus agentes.
- Nessa Oficina será estudada a biblioteca **RCRSCS** da *Universidade de Nagoya* disponível em <http://es.sourceforge.jp/projects/rcrscs/>

# Biblioteca RCRSCS

---

- O controle dos agentes pode ser
  - Distribuída
  - Centralizada
  - Híbrida
- Essa biblioteca utiliza controle híbrido dos agentes
  - Possibilita o controle dos agentes pela sua central
  - Permite que o agente decidir usar ou não a tarefa recebida como sua próxima ação

# Biblioteca RCRSCS

## Dinâmica

---

1. Os **agentes** percebem o ambiente e transmitem as informações para outros agentes de resgate (**centrais e agentes**)
2. As **centrais**, uma de cada tipo, analisam a situação de acordo com a informação recebida
3. As **centrais** enviam uma mensagem do tipo *TaskMessage* para os **agentes**
4. As **centrais** integram as diferentes informações e retransmitem para os **agentes**
5. Os agentes agem de acordo com a mensagem *TaskMessage* recebida

# Biblioteca RCRSCS

## Uso

---

- Estender a classe **AbstractCSAgent** ao invés da classe **StandardAgent**
- Necessário sobrescrever os métodos
  - `protected EnumSet<StandardEntityURN> getRequestedEntityURNEnum()`
  - `protected void thinking(int time, ChangeSet Changed, Collection<Commnad> heard)`

# Biblioteca RCRSCS

## Uso

---

- No método **think** é necessário
  - **setMessageChannel(int channel)**  
Definir o canal a ser utilizado para comunicação
  - **addMessage(RCRSCSMessage message)**  
Adiciona uma mensagem na lista de mensagens a serem transmitidas no ciclo de simulação



# Biblioteca RCRSCS

## Mensagens

---

- **Information Message**
  - Contém informações obtidas do ambiente e não inclui informações estáticas para reduzir o tamanho das mensagens
- **Task Message**
  - Ordena a ação de um agente indiretamente, mas possibilita que o agente escolha agir de acordo ou não com a ação ordenada
- **Report Message**
  - Relata o resultado de cada estratégia

# Biblioteca RCRSCS

## Mensagens

- Information Message

Information Message	Elementos
Building Message	Fieryness, Brokenness
Blockage Message	Road ID, Barycentric Coordinate*, Repair Cost
Victim Message	Area ID, HP, Buriedness, Damage, Position Coordinate*
Position Message	Agent ID, Position Coordinate
Transfer Message	Agent ID, IDs de algumas Area
Unpassable Message	Agent ID, fromAreaID, toAreaID
FireBrigade Message	Agent ID, HP, Buriedness, Damage, Water Quantity, Area ID
PoliceForce Message	Agent ID, HP, Buriedness, Damage, Area ID
AmbulanceTeam Message	Agent ID, HP, Buriedness, Damage, Area ID

\* Elemento opcional

# Biblioteca RCRSCS

## Mensagens

---

- **Task Message**

Task Message	Elementos
Clear Route Message	PoliceForce ID, ID da Area inicial, ID da Area final
Rescue Area Message	AmbulanceTeam ID, Lista de ID Rescue
Extinguish Area Message	FireBrigade ID, Lista de ID Extinguish

- **Report Message**

Report Message	Elementos
Done Message	ID Agente
Exception Message	ID Agente

# Atividade Prática 3

---

1. Executar o simulador usando as configurações padrão e o time de agentes que se encontra no projeto **libraryAgentSample**, classe

*jp.ac.nagoyau.is.ss.kishii.ob.launch.LaunchOBAgents*

2. Alterar o comportamento da classe **AbstractOBSampleAgente** no projeto **libraryAgentSample** para que ele só transmitam as informações dos objetos que realmente mudaram desde último ciclo

*jp.ac.nagoyau.is.ss.kishii.ob.sample.AbstractOBSampleAgent*

# Referências

---

- Morimoto, T. *How to Develop a RoboCupRescue Agent*. <http://robomec.cs.kobe-u.ac.jp/robocup-rescue>
- RCRSCS. *Libraries for Communication – ReadMe*. <http://es.sourceforge.jp/projects/rcrscs/>