Introduction to Rescue Simulation

2022 WORKSHOP ON AI AND SIMULATION FOR NATURAL DISASTER MANAGEMENT

JULY 17, 2022

Agenda

- RoboCup
- Natural Disaster Overview
- Rescue Agent Simulator
- Scenario Editor
- Simulator Structure
- Agent Behavior
- Agent Communication

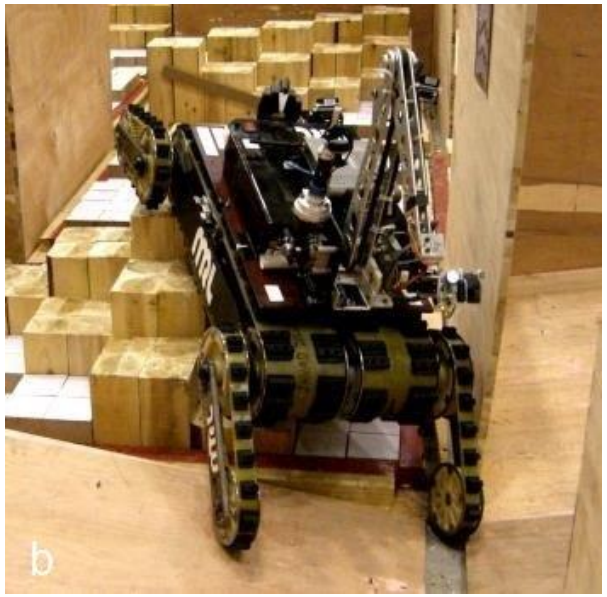
RoboCup

What is RoboCup?

- An international effort to foster **Artificial Intelligence and Robotics** research by providing **standard problems** where wide range of technologies can be integrated and examined
- Managed by **RoboCup Federation** since 1996
- Involve more than 40 countries and more than 3000 participants
- Working Groups dedicated to explore the application of AI and Robotics in different socially significant domains

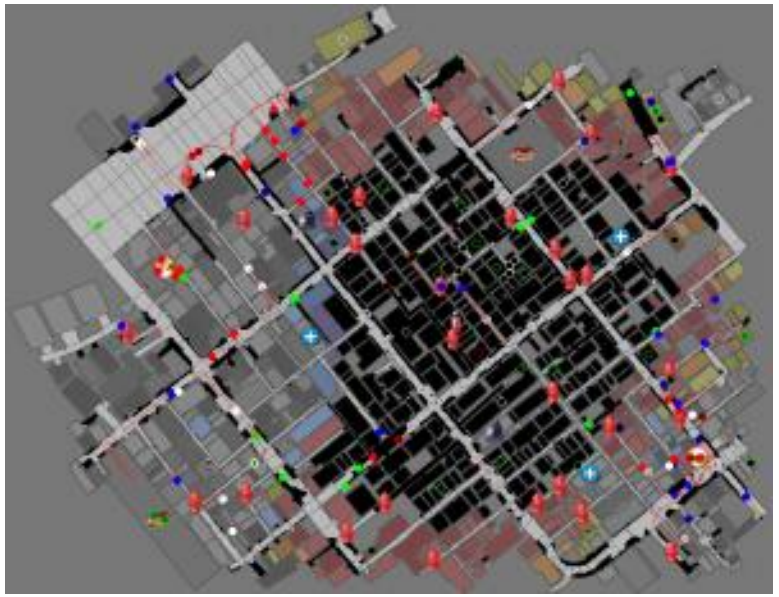
Rescue Simulation Working Group

- Created in 1999 focused on **promoting research and development** in the socially significant domain of **natural disaster**
- Use of AI and Robotics to support **planning** and **decision-making** of rescue teams in a post-disaster urban setting



Robot

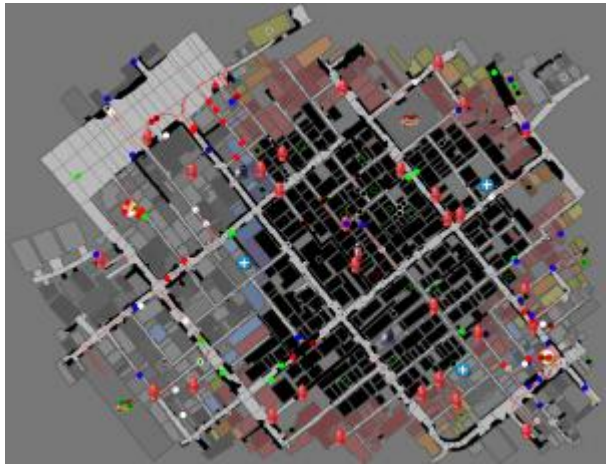
Virtual Robot

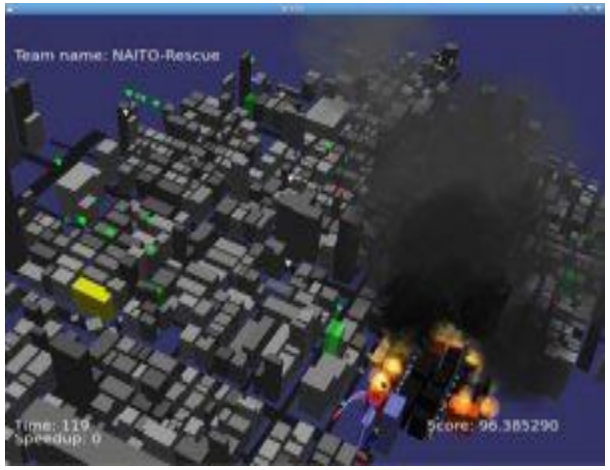


Agent Simulation

Rescue Agent Simulation

- **Develop a simulator** able to represent natural disaster scenarios to support disaster response planning
- **Evaluate response plans** elaborated by policy-makers to act in real natural disaster scenarios
- **Organize competitions** to stimulate the exchange of ideas and experience between researchers and practitioners





Natural Disaster Overview

Natural Disaster

- Natural disasters are major adverse events
- Natural disasters impact the infrastructure and environment of the affected areas causing
  - loss of shelter
  - food shortage
  - spread of infectious diseases
- Effective monitoring for immediate post disaster response help reduce
  - economic losses
  - fatalities

Source: National Geographic

Source: TipTopTens

Number of natural disasters in China in 2018
22

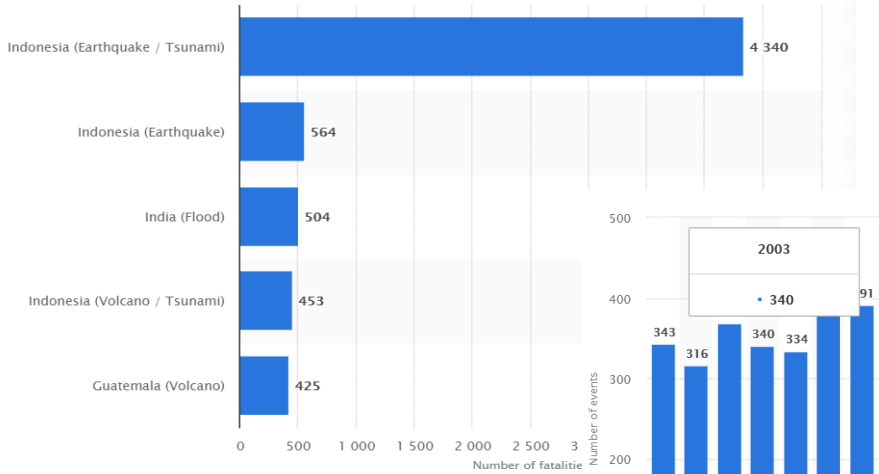
Share of fatalities from natural disasters in Asia in 2018
79.8%

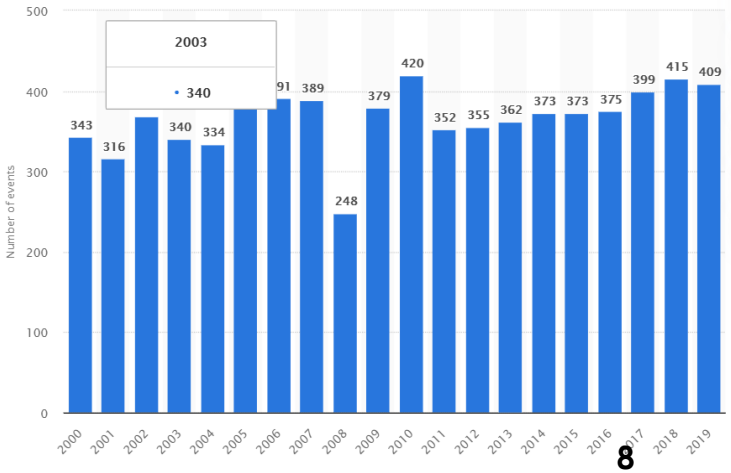
Share of natural disaster cost in the Americas
53 %

Type of natural disaster with most victims in 2018 - floods
34.2 million people

Most frequent type of natural disaster in 2018 - Flood
127

Cost of damages of storms in 2018
70.8bn USD





Natural Disaster

Year	Country	Magnitude	Death toll
2010	Haiti	7.0	316,000
1976	China	7.5	242,769
2004	Indonesia	9.1	227,898
1920	China	7.8	200,000
1923	Japan	7.9	142,800
1948	Turkmenistan	7.3	110,000
2008	China	7.9	87,587
2005	Pakistan	7.6	86,000
1908	Italy	7.2	72,000
1970	Peru	7.9	70,000

10 deadliest earthquakes since 1900

Source: United States Geological Survey (USGS)

[Deadliest Earthquakes on Record]

9

Disaster Management

Response Phase

Objectives

- **Save** lives
- **Prevent** new disasters
- **Collect** data
- Short-/Long-term **planning**

Response Phase

Objectives

- **Save** lives
- **Prevent** new disasters
- **Collect** data
- Short-/Long-term **planning**

Limitations

- **Limited** resources
- **Incomplete** information
- **Real-time** decision-making
- **Large number** of people involved
- **Heterogeneity**

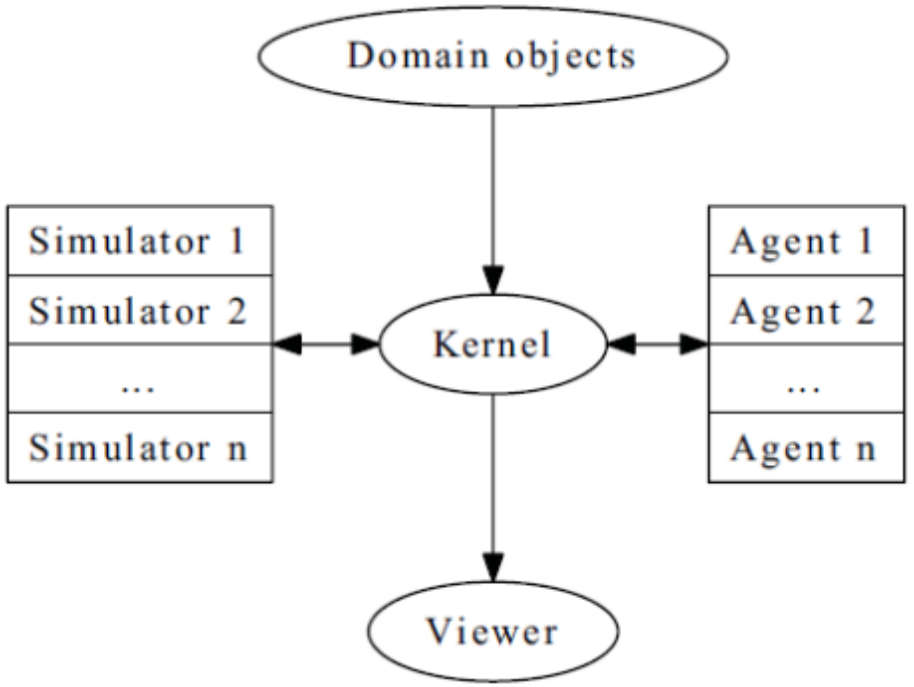
Rescue Agent Simulator

OVERVIEW

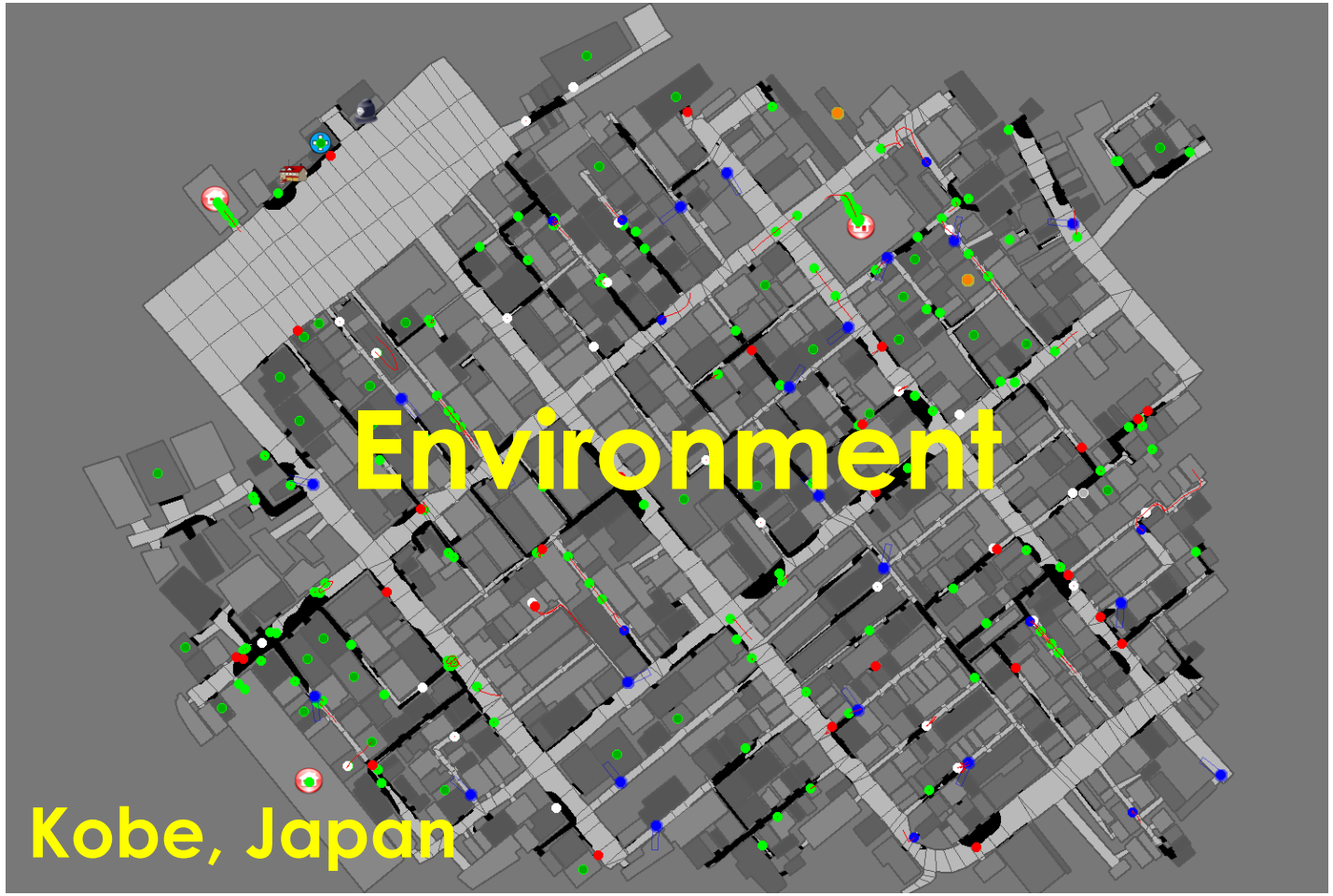
Rescue Agent Simulation Platform

- A computer simulation platform that can **represent natural disaster scenarios**
- Large multiagent simulation system
- Composed of multiple components
- Kernel coordinates all components

Simulator	Description
Traffic	Agents' movement
Collapse	Buildings' structural damage and blockade creation
Clear	Manage blockade removal
Ignition	Ignite fires randomly
Fire	Fire spread and extinction
Miscellaneous	Human damage and buriedness



Rescue Agent Simulation Platform



Rescue Agent Simulation Platform



Ambulance Team



Fire Brigade



Police Force



Civilians

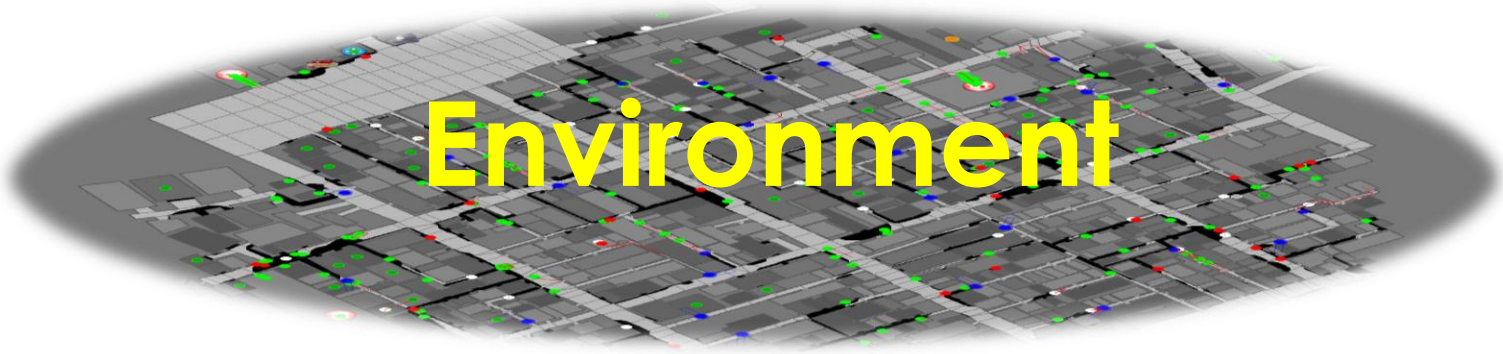

Ambulance Centre


Fire Station

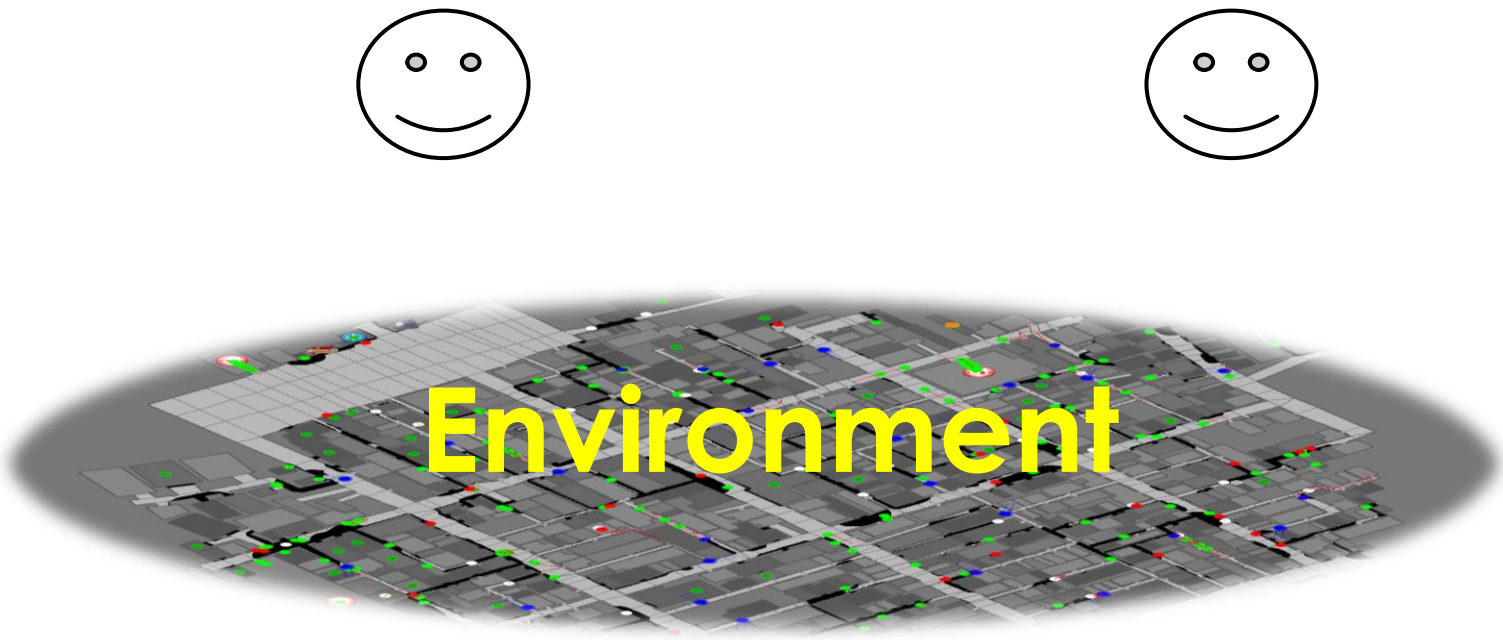

Police Office


Refuge / Shelter

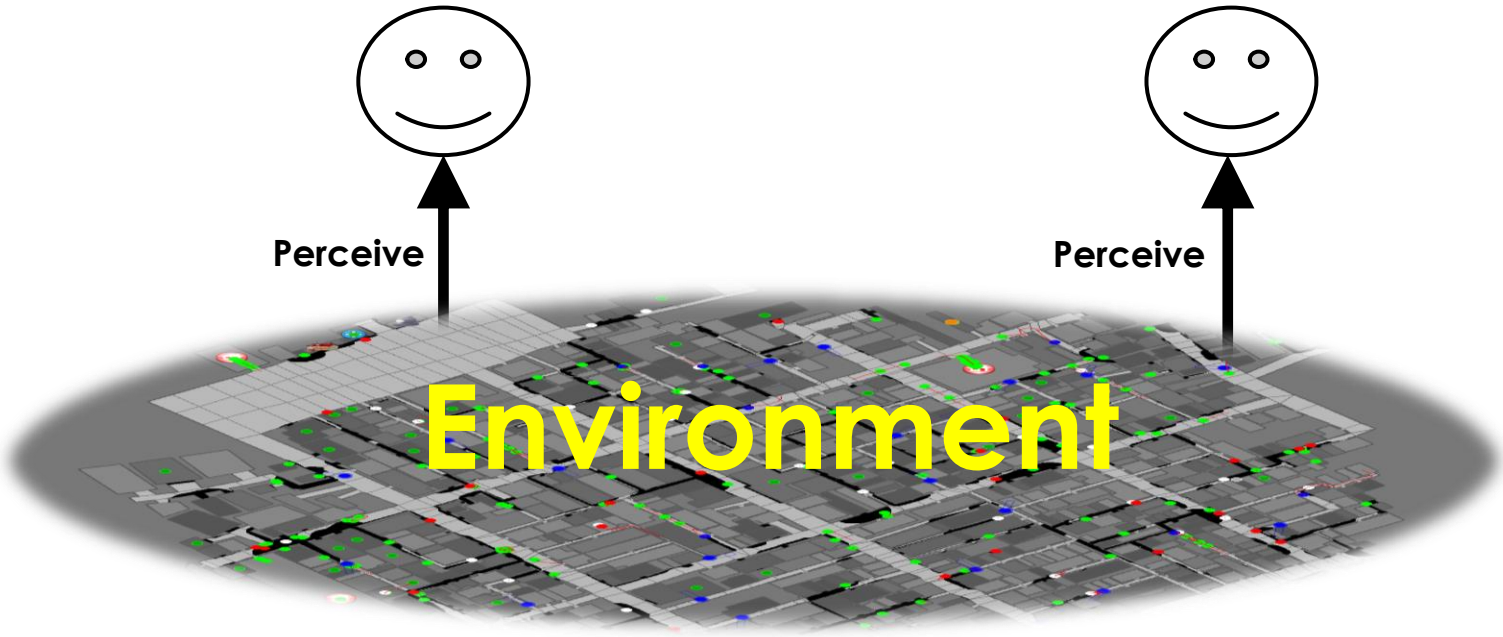
Rescue Agent Simulation Platform



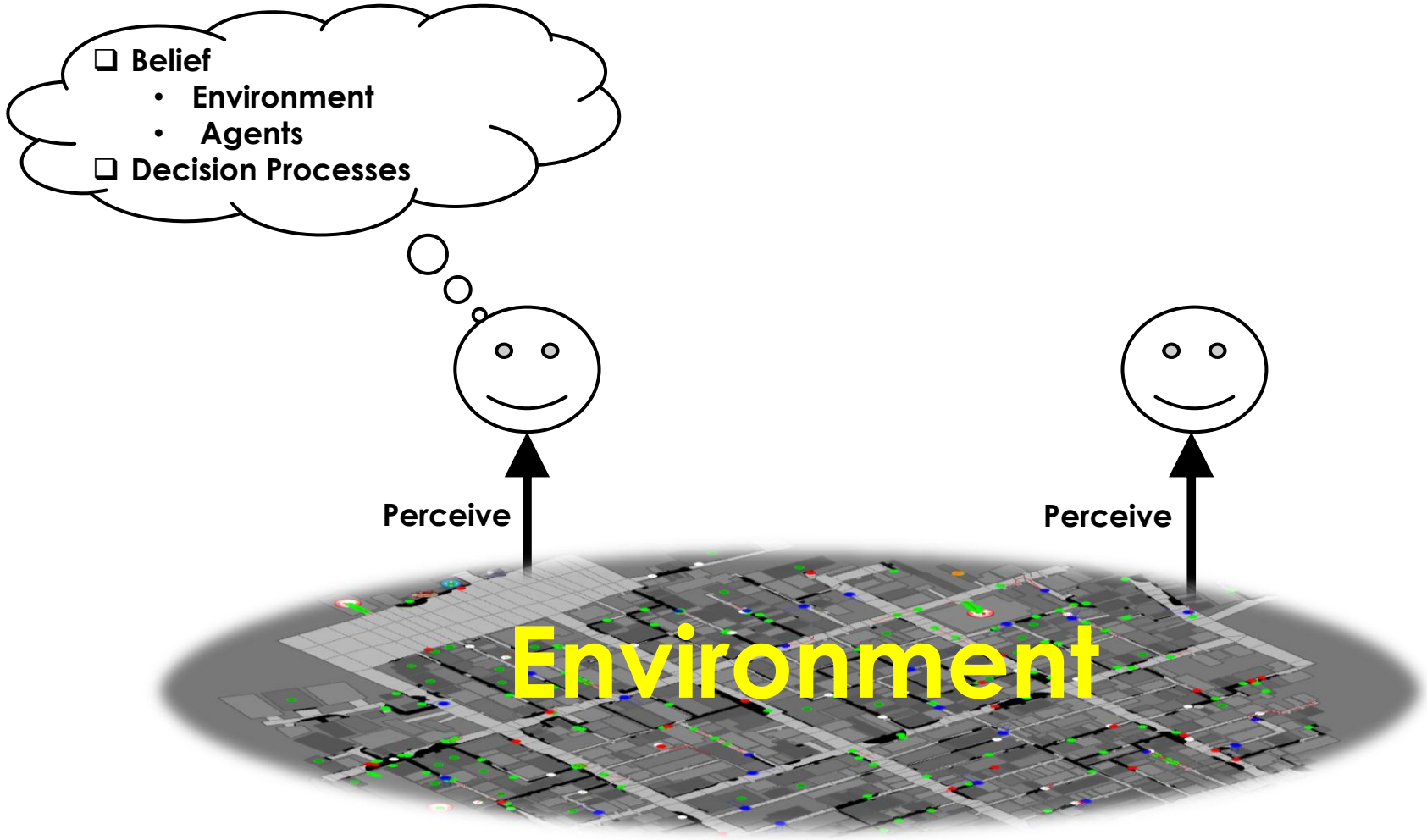
Rescue Agent Simulation Platform



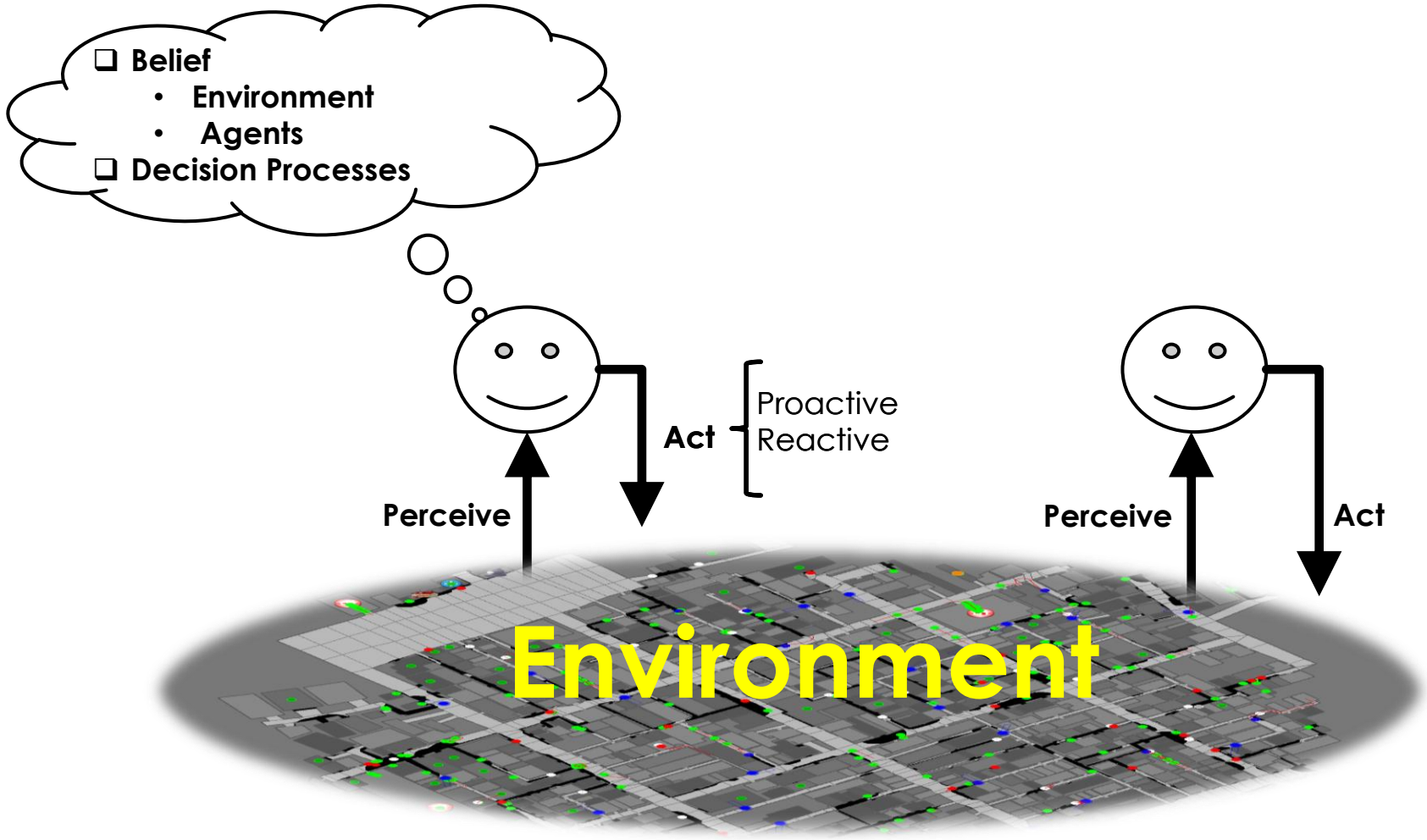
Rescue Agent Simulation Platform



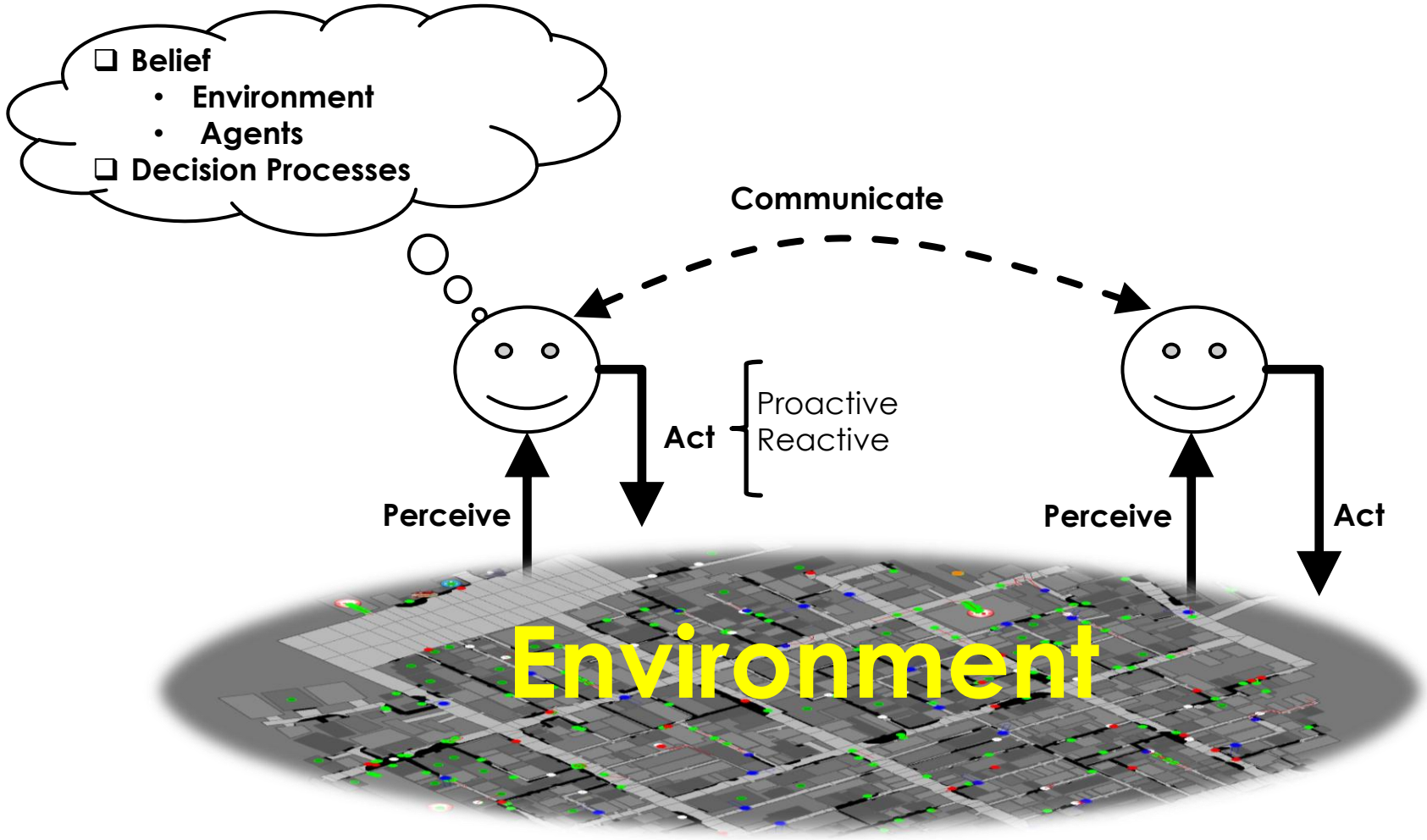
Rescue Agent Simulation Platform



Rescue Agent Simulation Platform



Rescue Agent Simulation Platform



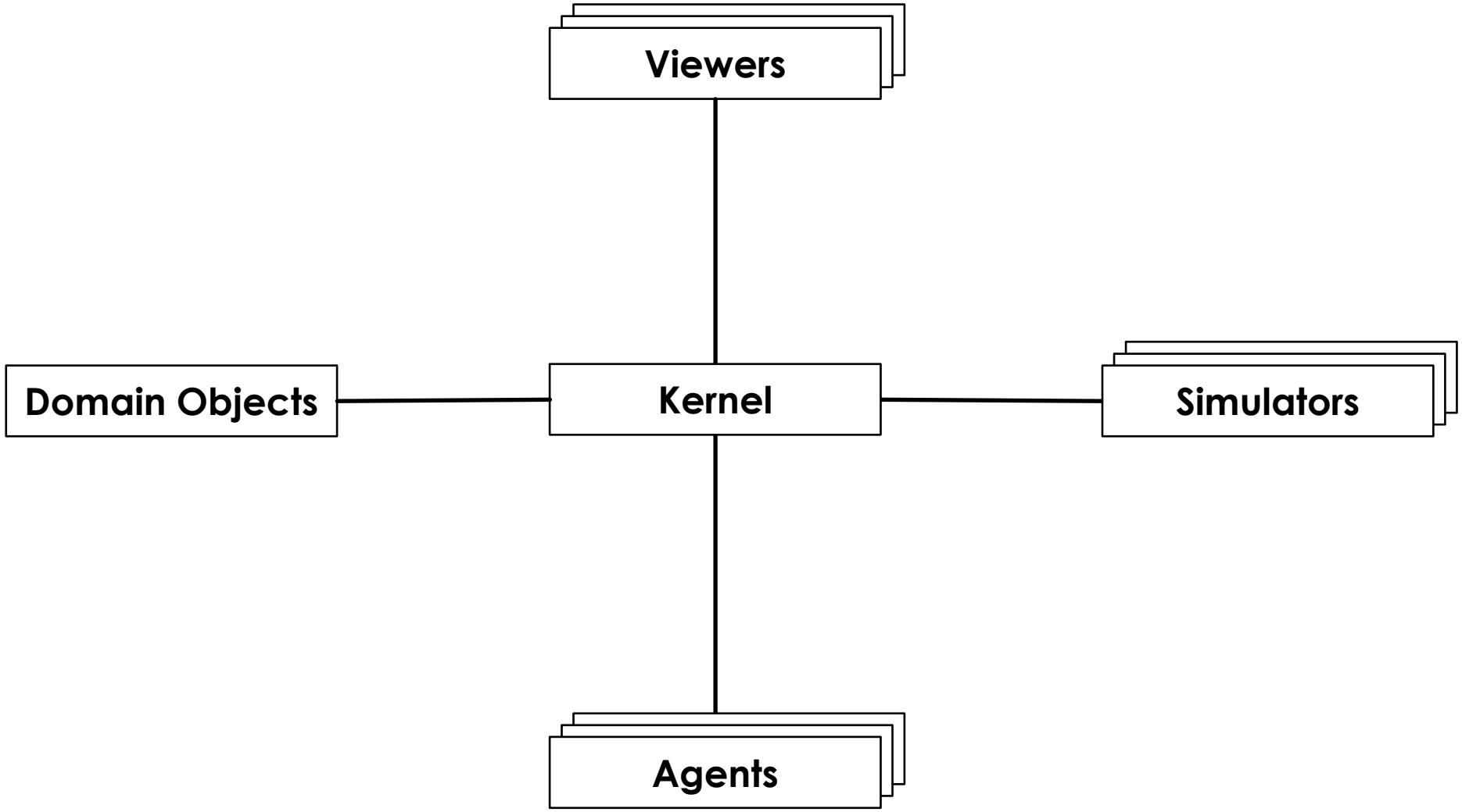
Environment

- **Partially Observable**
  - Agents have a limited range of perception
  - They have complete knowledge of the map
- **Dynamic**
  - The state of the disaster changes over time, e.g., the civilian loses health over time
- **Stochastic**
  - The initial condition and the evolution of the disaster is randomly defined

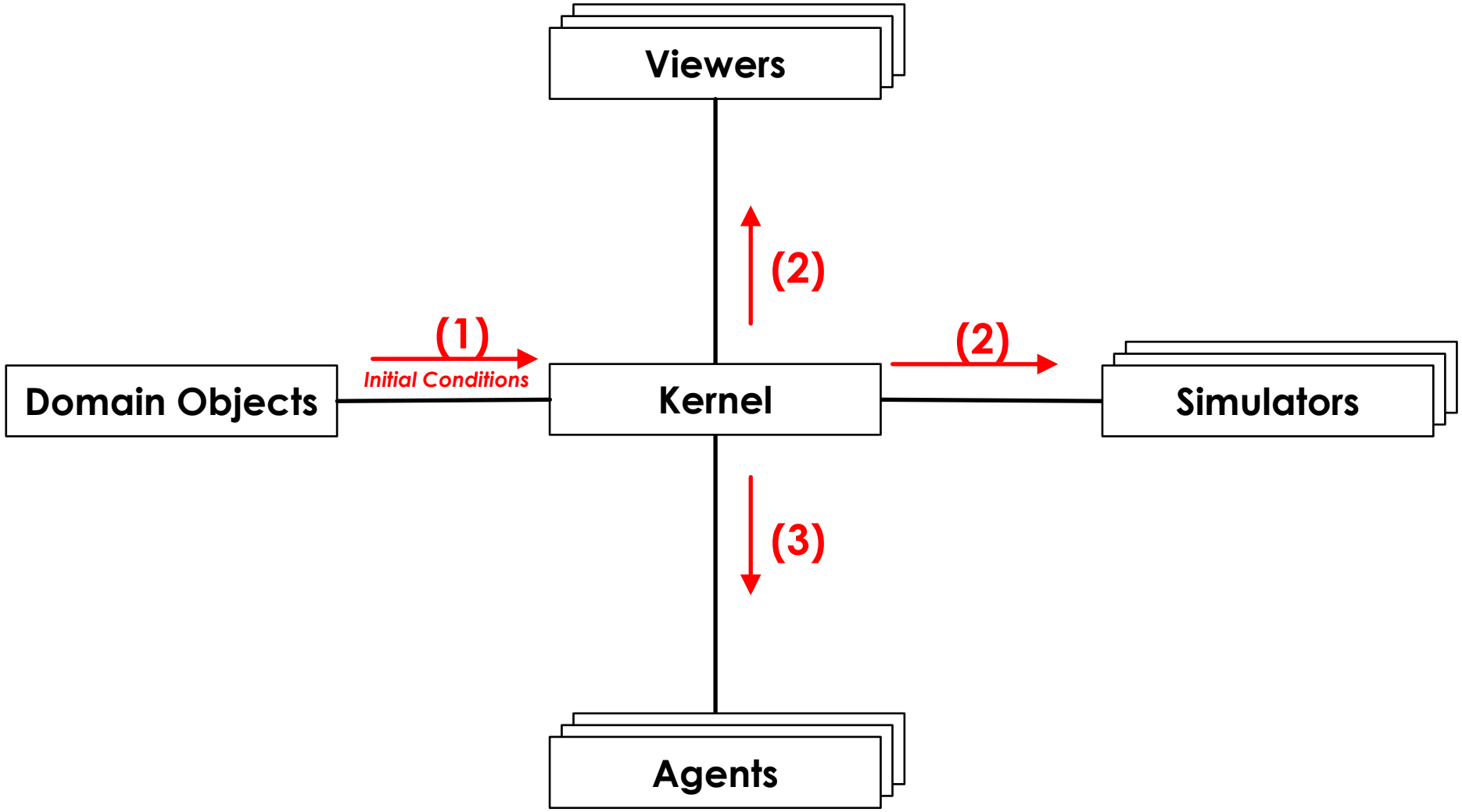
Communication

- **Voice vs Radio**
  - Agents can communicate directly with other agents in a short-range distance
  - Agents communicate broadcast radio to subscribed agents
- **Restricted**
  - The communication channels have a limited bandwidth to communicate
- **Unreliable**
  - Messages may be dropped

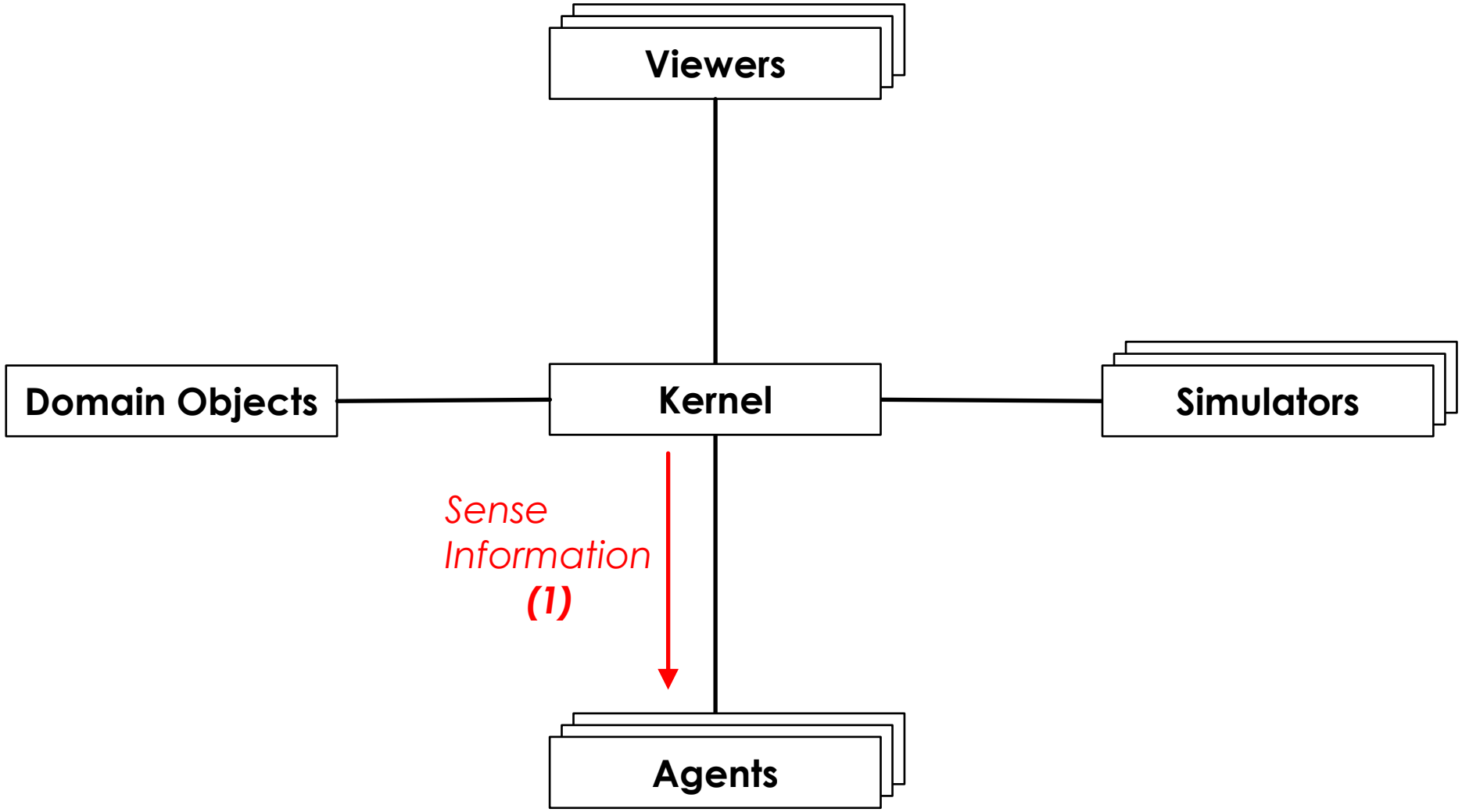
Architecture



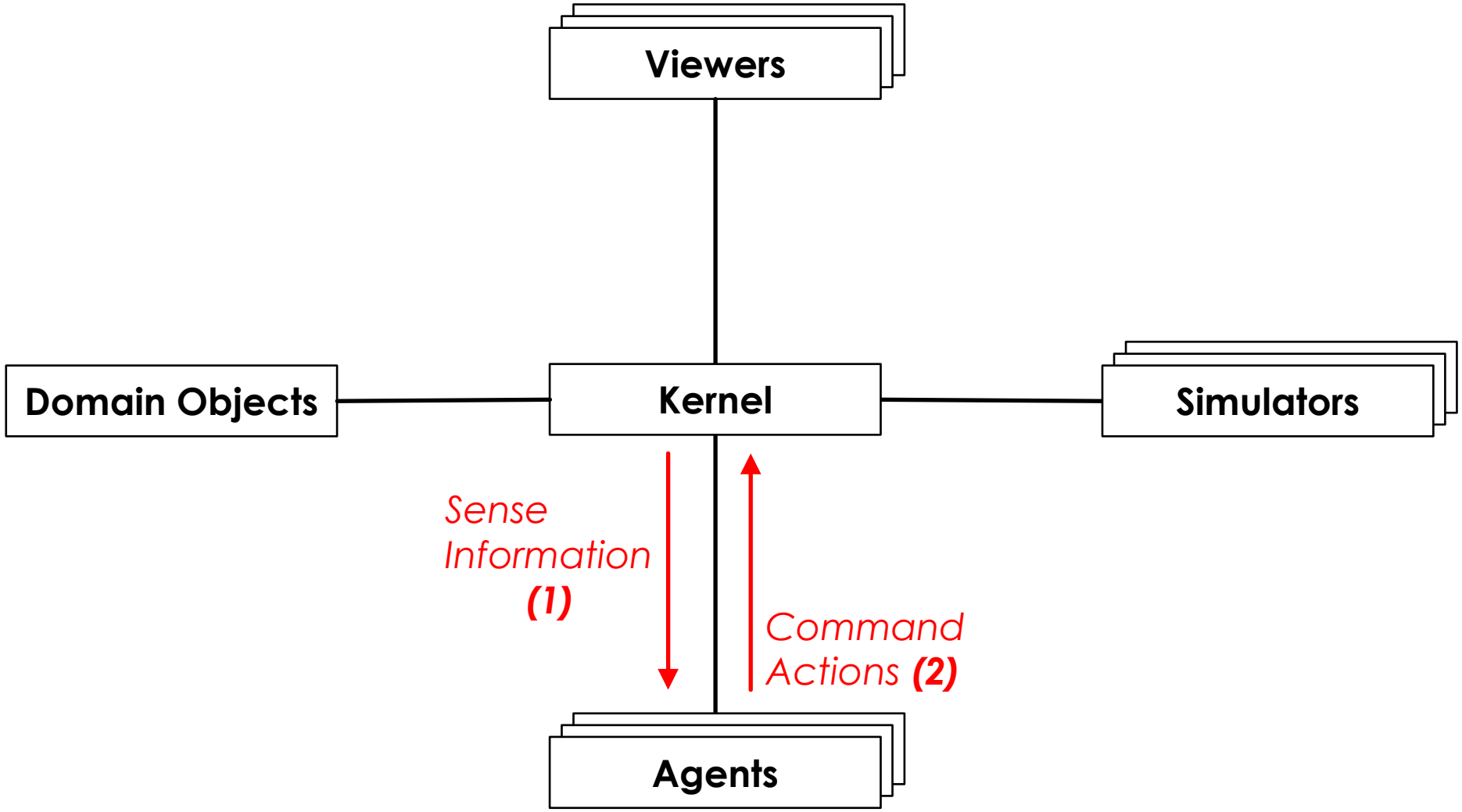
Initialization



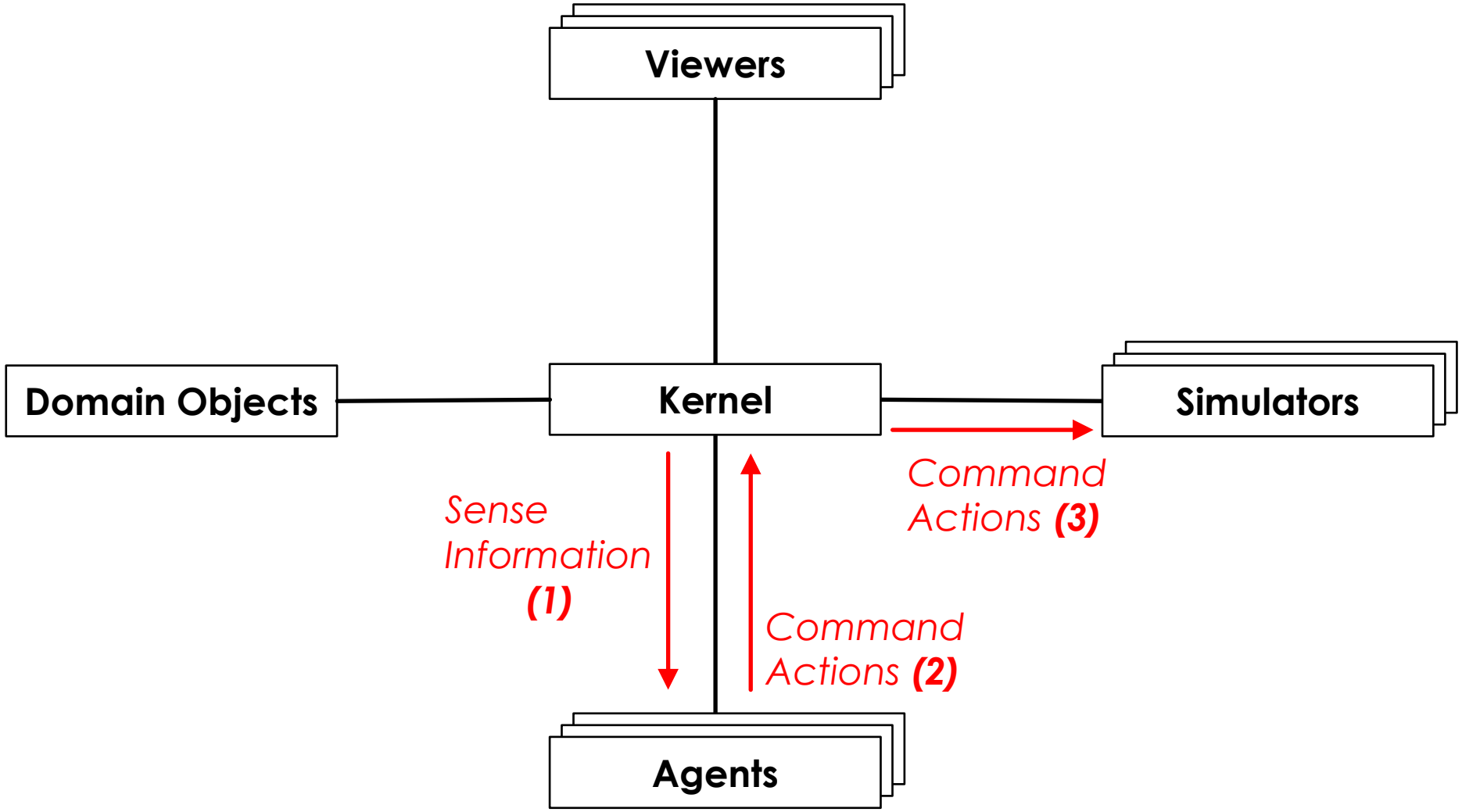
Simulation Dynamics



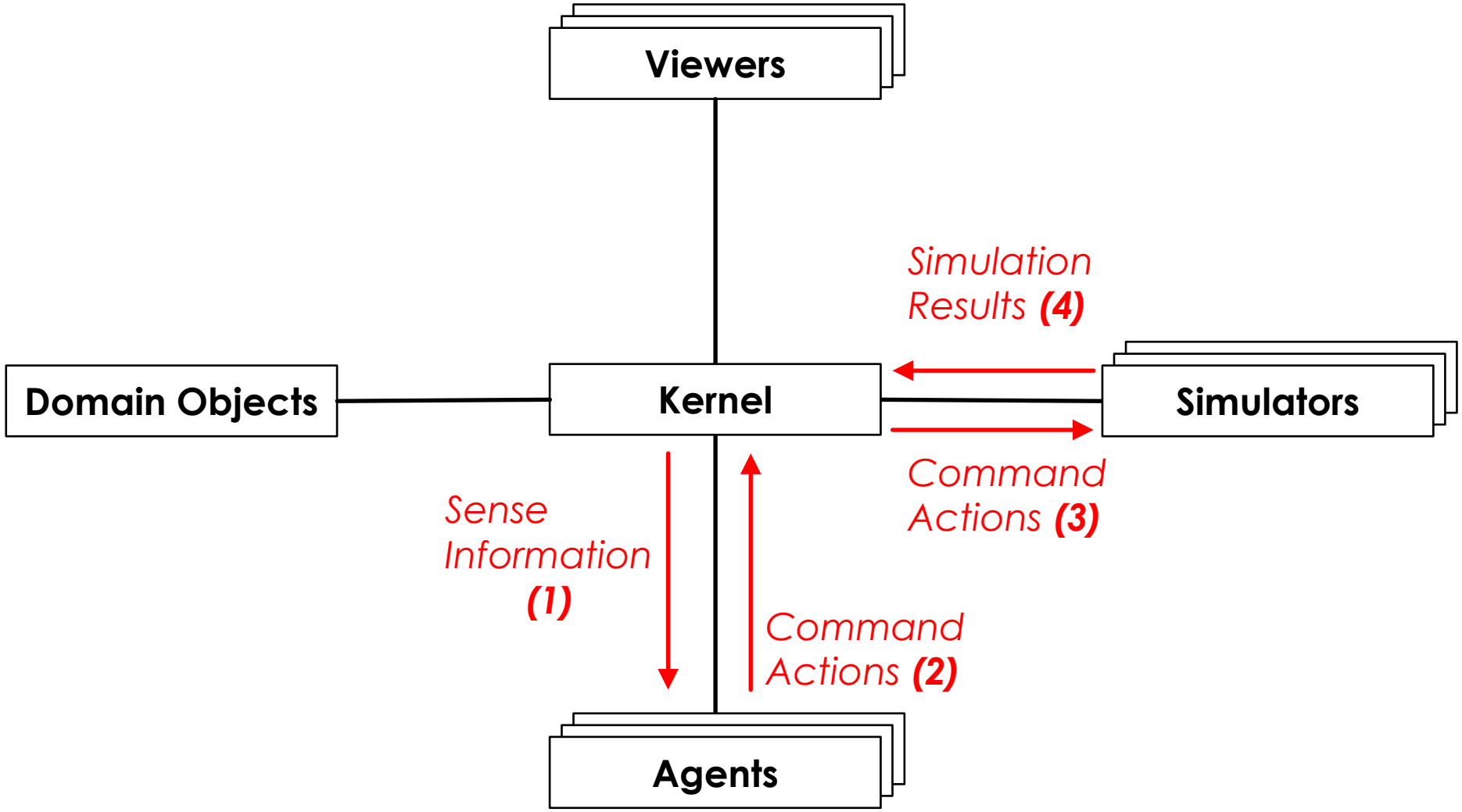
Simulation Dynamics



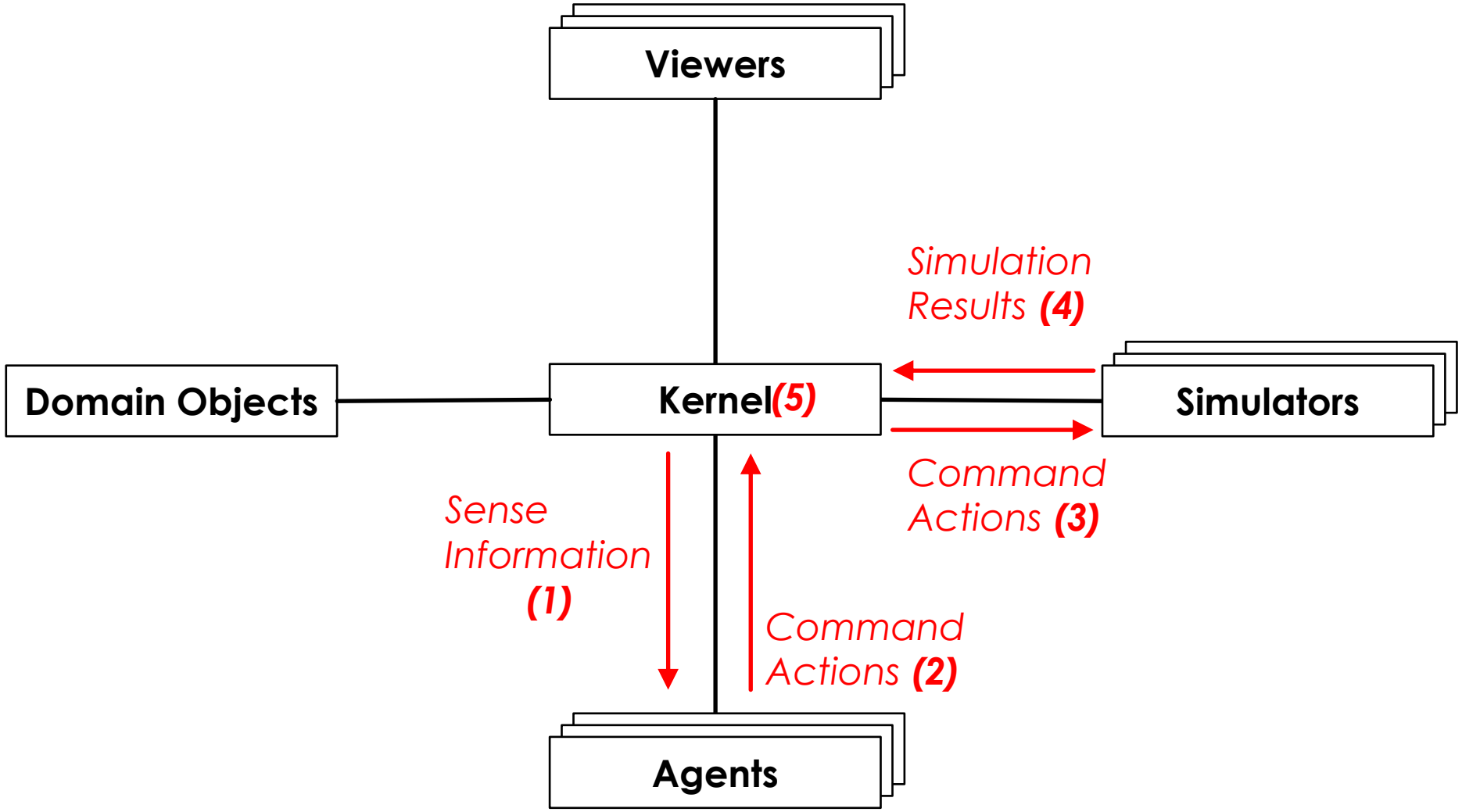
Simulation Dynamics



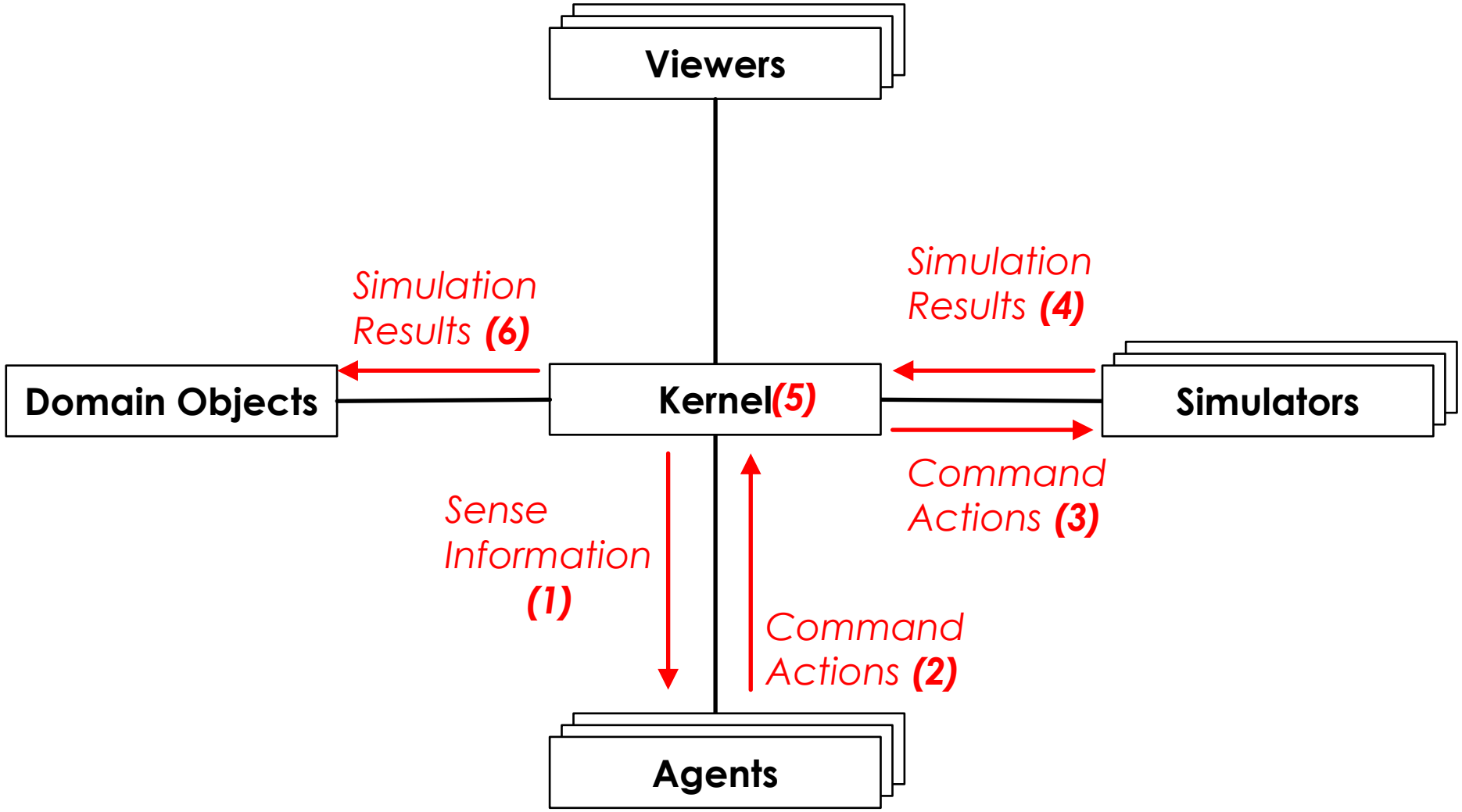
Simulation Dynamics



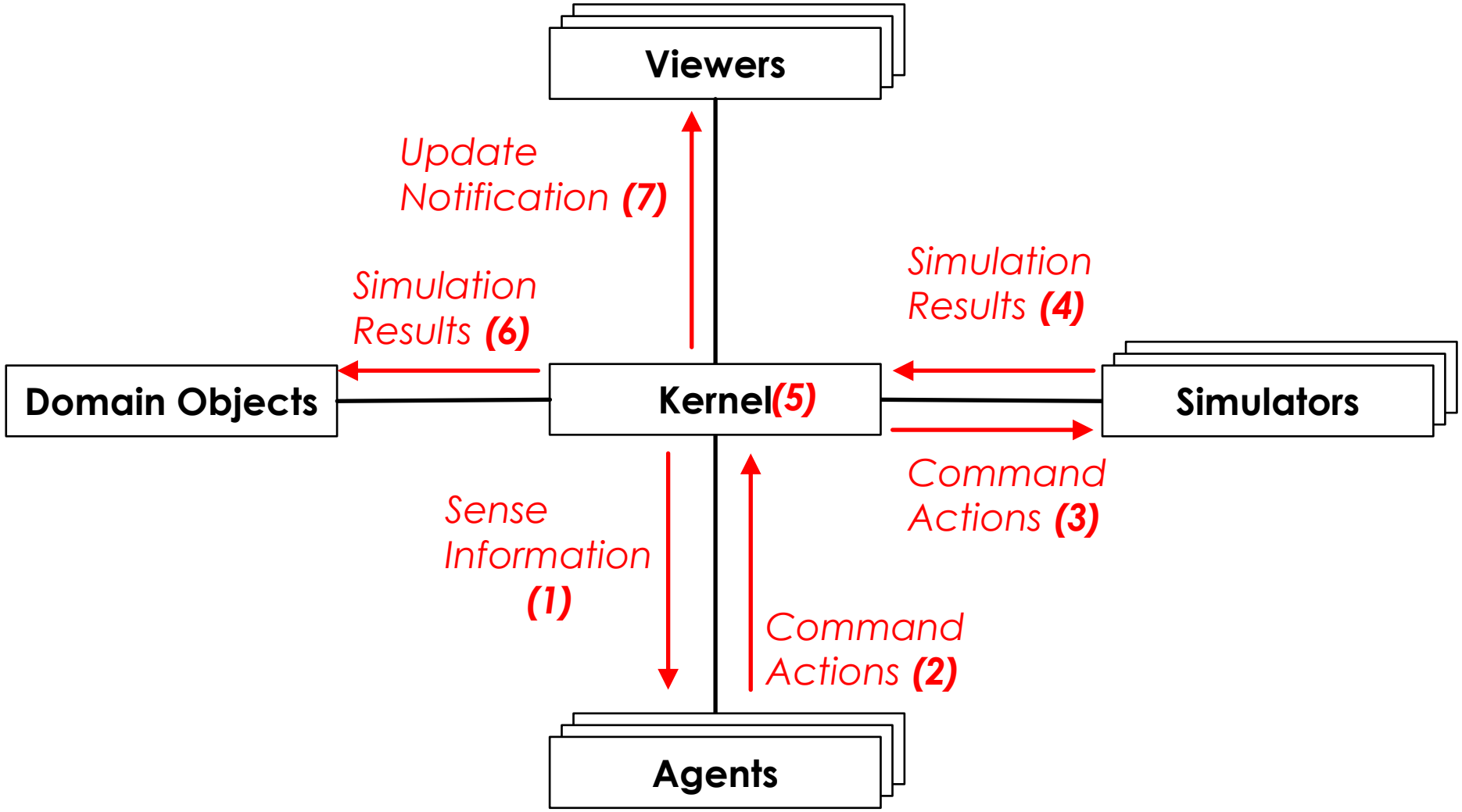
Simulation Dynamics



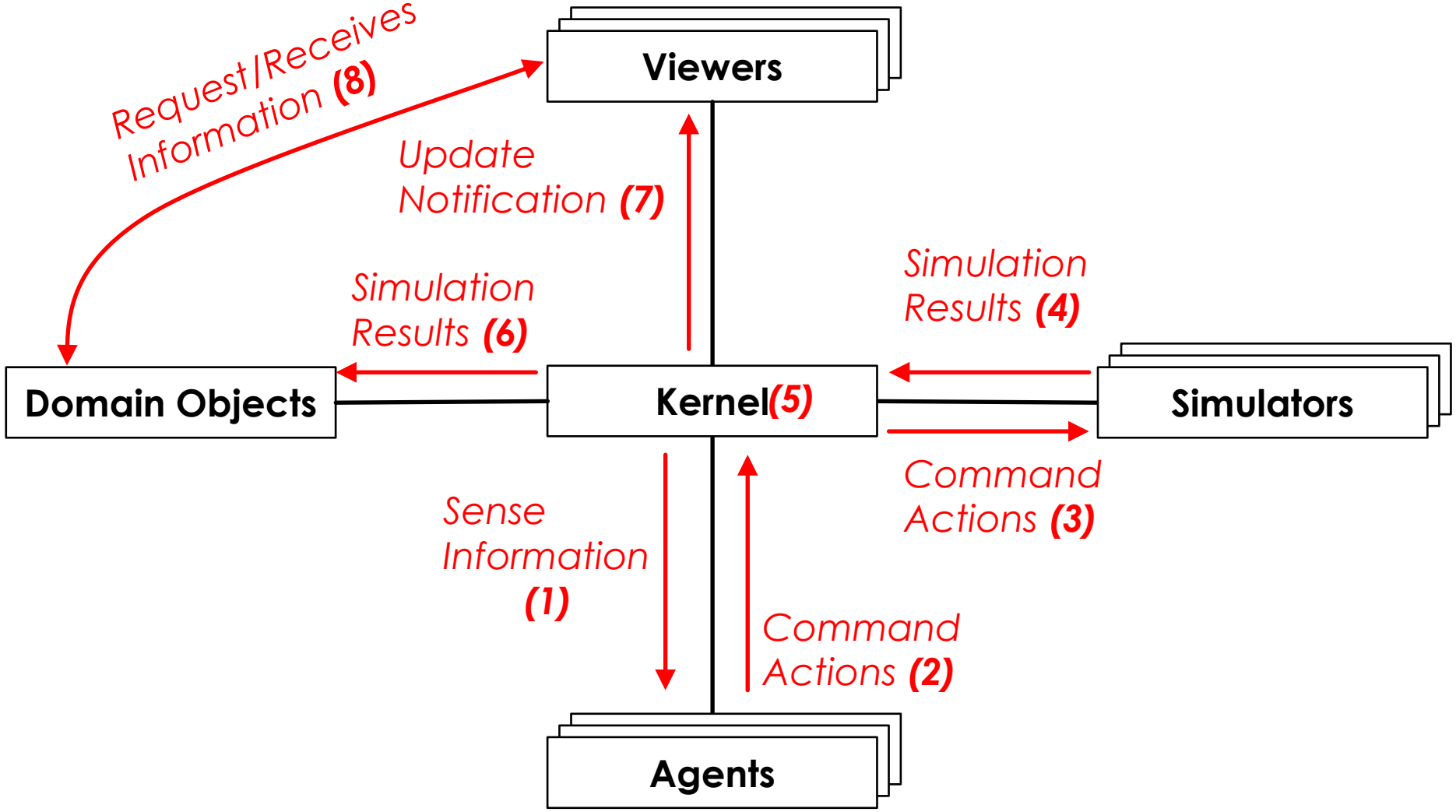
Simulation Dynamics



Simulation Dynamics



Simulation Dynamics



Rescue Agent Simulator

INSTALLATION AND EXECUTION

Rescue Agent Simulator Installation

- Software – Prerequisites
  - Git
  - OpenJDK Java 17
  - Gradle
- Download

```
$ git clone https://github.com/roborescue/rcrs-server.git
```

- Compile

```
$ cd rcrs-server
$ ./gradlew completeBuild
```

Sample Agent Installation

- Software – Prerequisites
  - Git
  - OpenJDK Java 17
  - Gradle
- Download

```
$ git clone https://github.com/roborescue/rcrs-sample-agent-java.git
```

- Compile

```
$ cd rcrs-sample-agent-java
$ ./gradlew clean build
```

Rescue Agent Simulator Execution

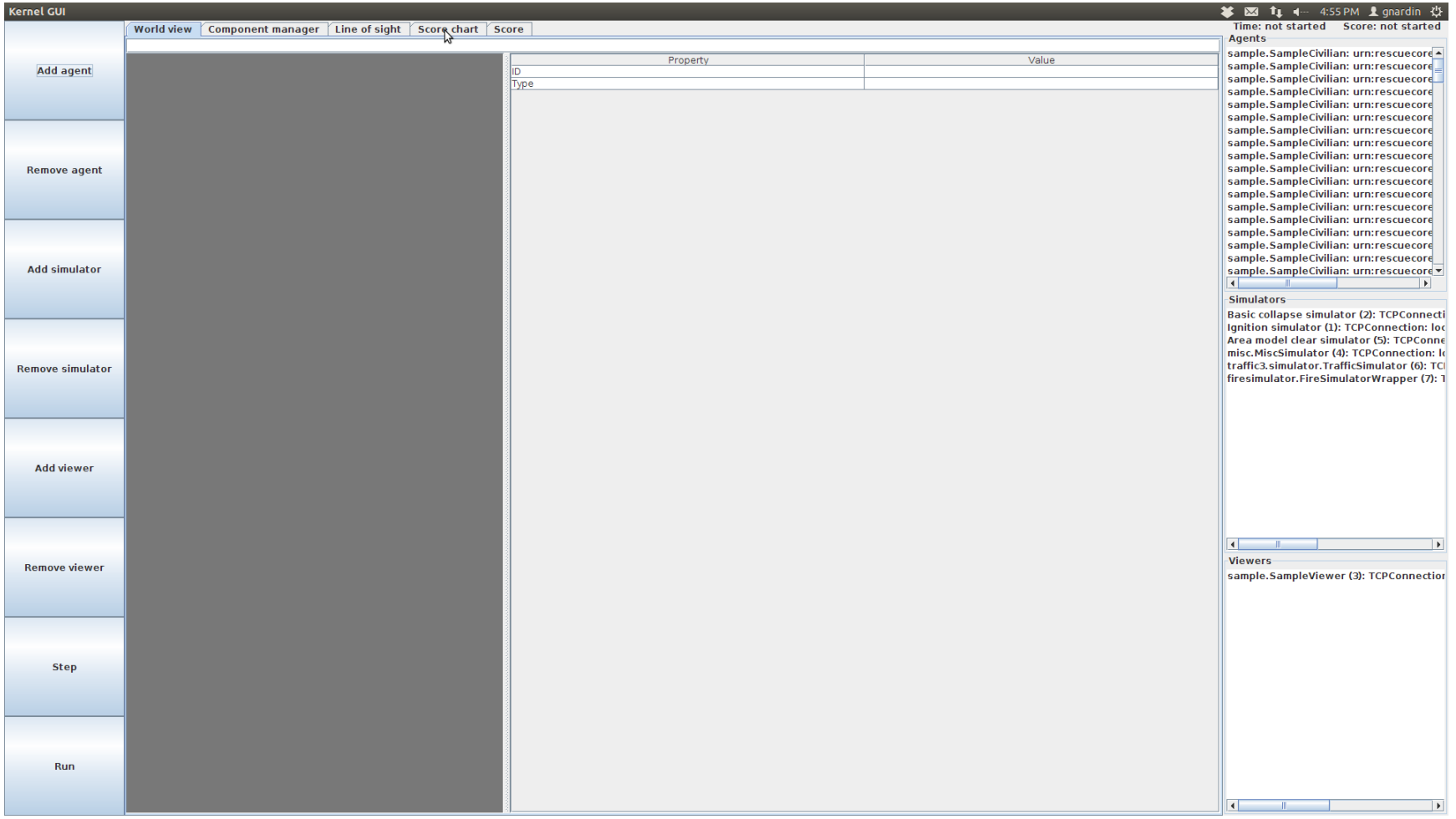
start.sh [options]

-m <scenariodir>	Scenario directory
-l <logdir>	Log directory
-s	Add date and time in the log directory
-t <teamname>	Name of the team

- **Example**

```
$ cd rcrs-server/scripts
$ ./start.sh –m ../maps/kobe/map –c ../maps/kobe/config
```

Control Panel

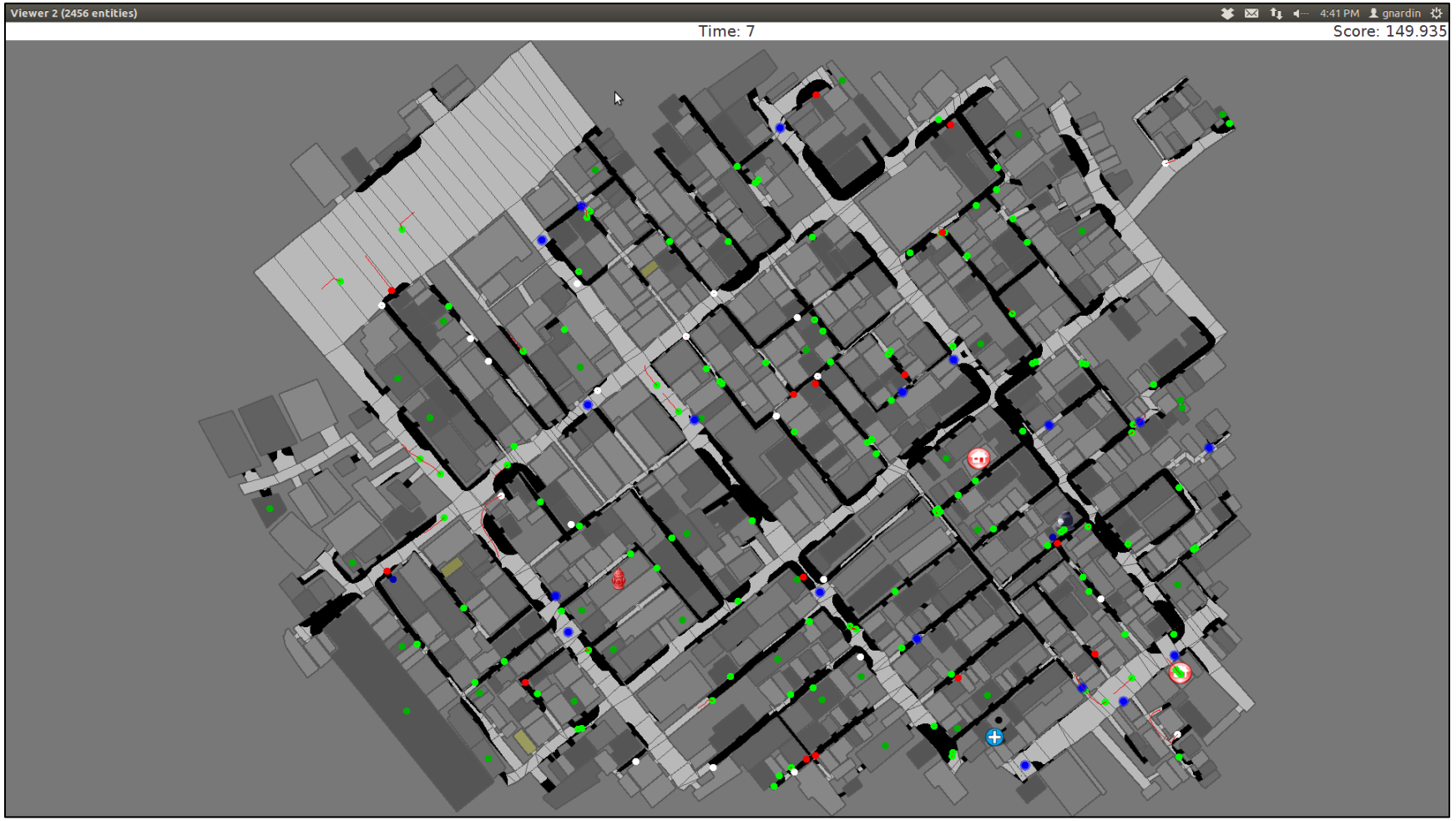


Sample Agent Execution

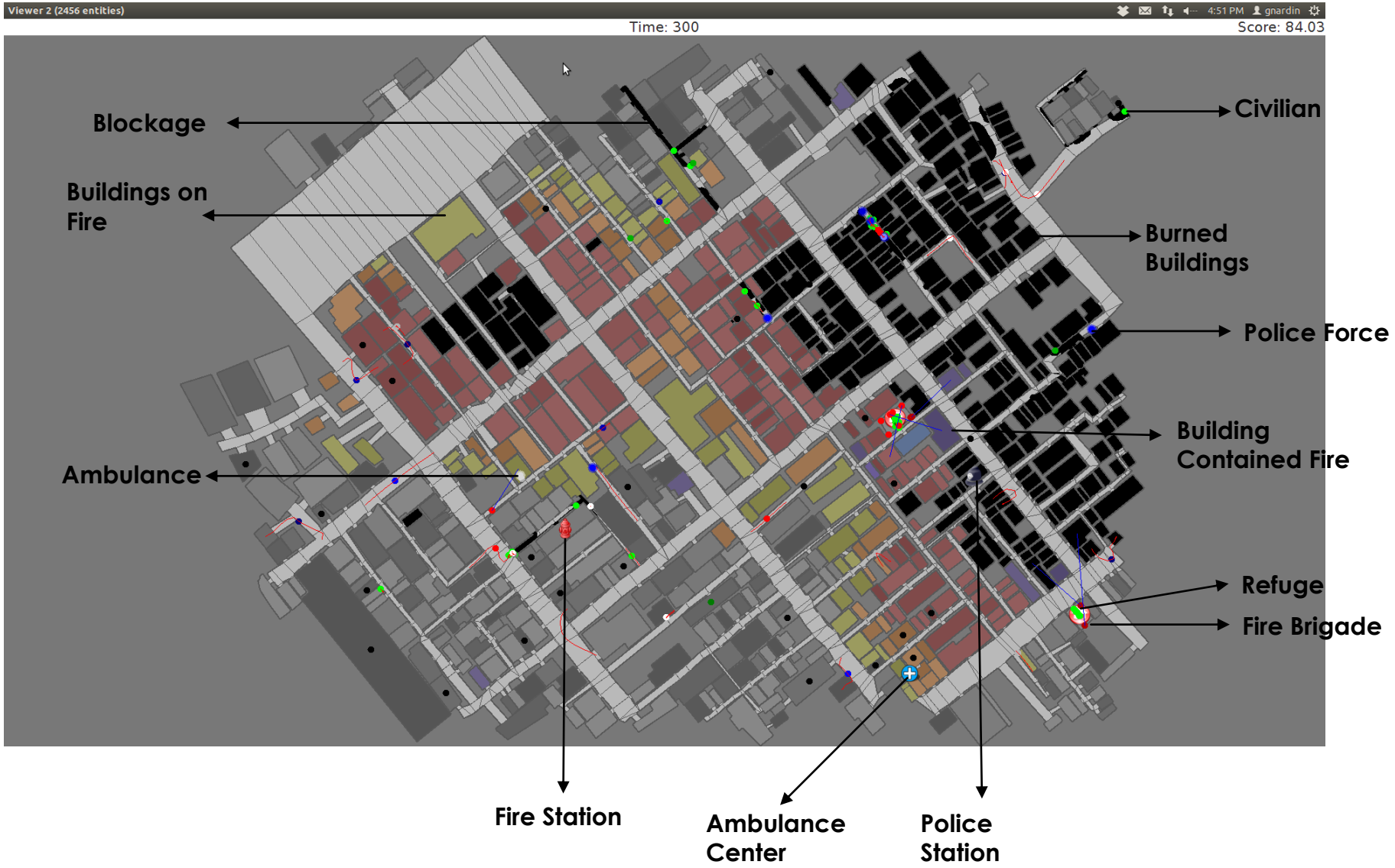
- Run using Gradle

```
$ cd rcrs-sample-agent-java
$ ./gradlew launch
```

Simulation Run



Simulation Run



Default Score

- Score Calculation

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

onde,

- **P** is the number of civilians alive
- **H** is the sum of the health index of the remaining civilians
- **H_{int}** is the sum of the health index of all civilians
- **B** is the sum of the area of each building * factor
- **B_{max}** is the sum of the area of all buildings

If Fieryness = 0
factor = 1
If Fieryness = 1, 4 or 5
factor = 0,66
If Fieryness = 2 or 6
factor = 0,33
If Fieryness = 3, 7 or 8
factor = 0

Folder Structure

/build	Java classes
/docs	simulator documentation
/jars	simulator JAR files (generated after compilation)
/lib	libraries used by the simulator
/log	scenario execution logs
/maps	scenarios
/modules	simulator' source-code

Hands-on

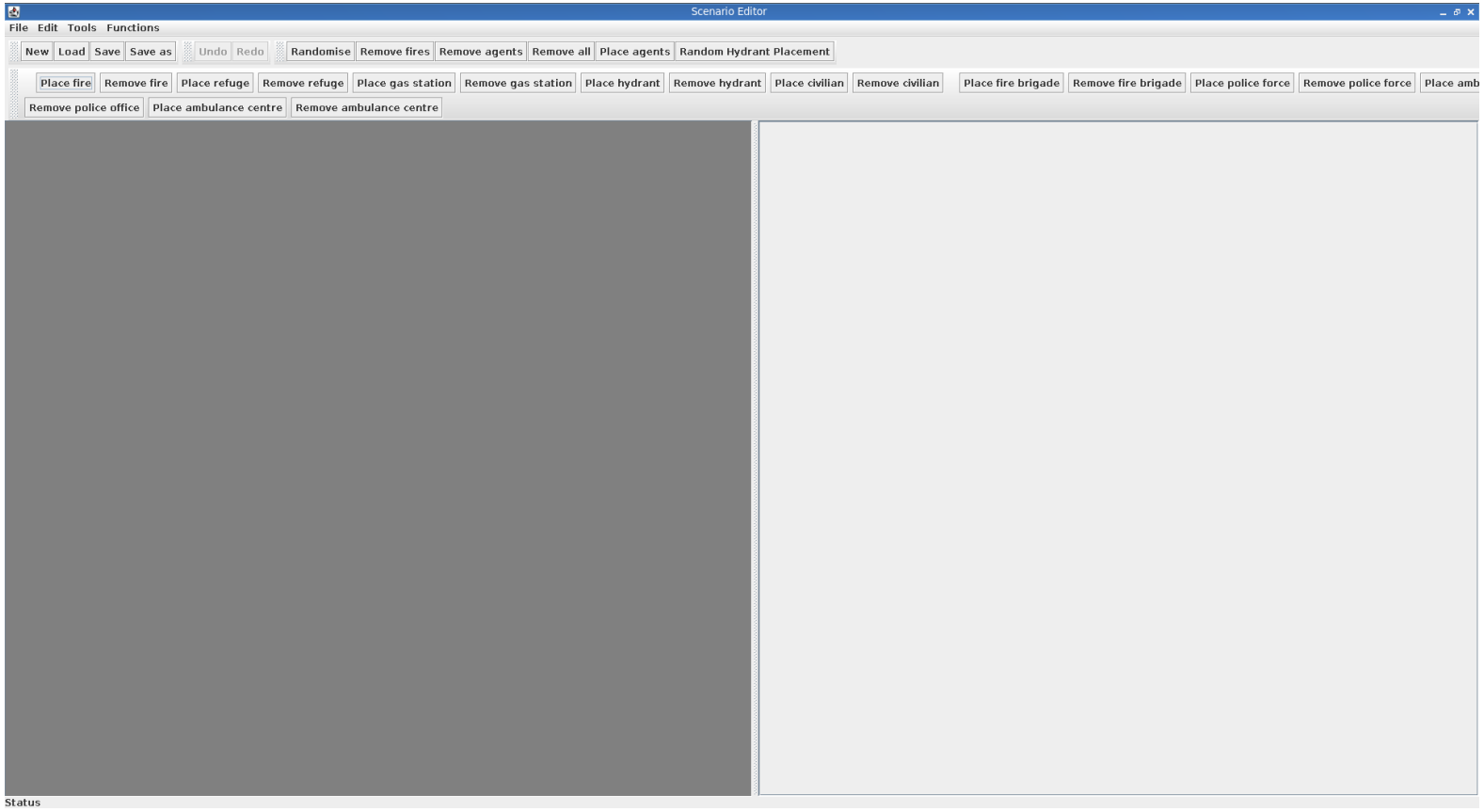
1. Install the Rescue simulator and SampleAgent
2. Execute the Rescue simulator using the **Berlin** scenario (folder ../maps/berlin)
3. Execute the SampleAgent team
4. Alter
    I. number of cycles (timesteps) from 250 to 200
    II. no radio communication
5. Execute the Rescue simulator and SampleAgent team in the **Berlin** scenario

Scenario Editor

Scenario Editor Execution

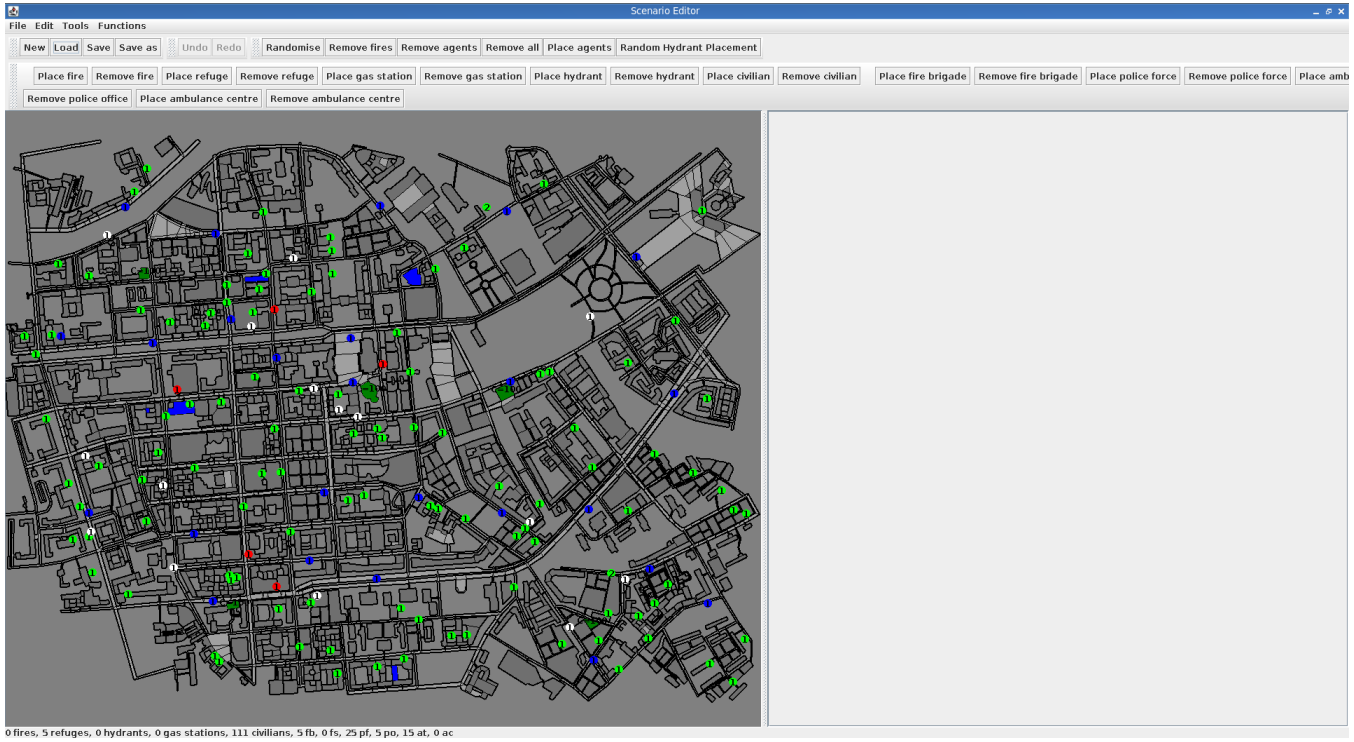
- Run using Gradle

```
$ cd rcrs-server
$ ./gradlew scenarioEditor
```



Scenario Editor Execution

- Select Load
  - Choose the **maps/berlin/map** folder
- Select **Remove all** button
- Select **Place agents** to add agents
- Select **Place Refuge** to add refuges
- Select **Place fire station**, **Place ambulance centre** and **Place police station** to add Fire Station, Ambulance Centre and Police Station respectively



Scenario Configuration

- Configuration files are placed in the config folder under the scenario folder. Example: **maps/berlin/config**
- Relevant configuration files and parameters are
  - **common.cfg**
    - random.seed: 1
    - kernel.host: localhost
    - kernel.port: 27931
  - **kernel.cfg**
    - kernel.timesteps: 300
    - kernel.startup.connect-time: 180000

Scenario Configuration

- Continue
  - **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

Scenario Configuration

- Continue
  - **clear.cfg**
    - clear.repair.rate: 20
    - clear.repair.distance: 20000
    - clear.repair.rad: 2000
  - **collapse.cfg**
    - collapse.create-road-blackages: true
    - collapse.floor.height: 5
    - collapse.wall-extent.min: 0.5
    - collapse.wall-extent.max: 1.0

Scenario Configuration

- Continue
  - **commsXXXX.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

Scenario Configuration

- Continue
  - **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

Hands-on

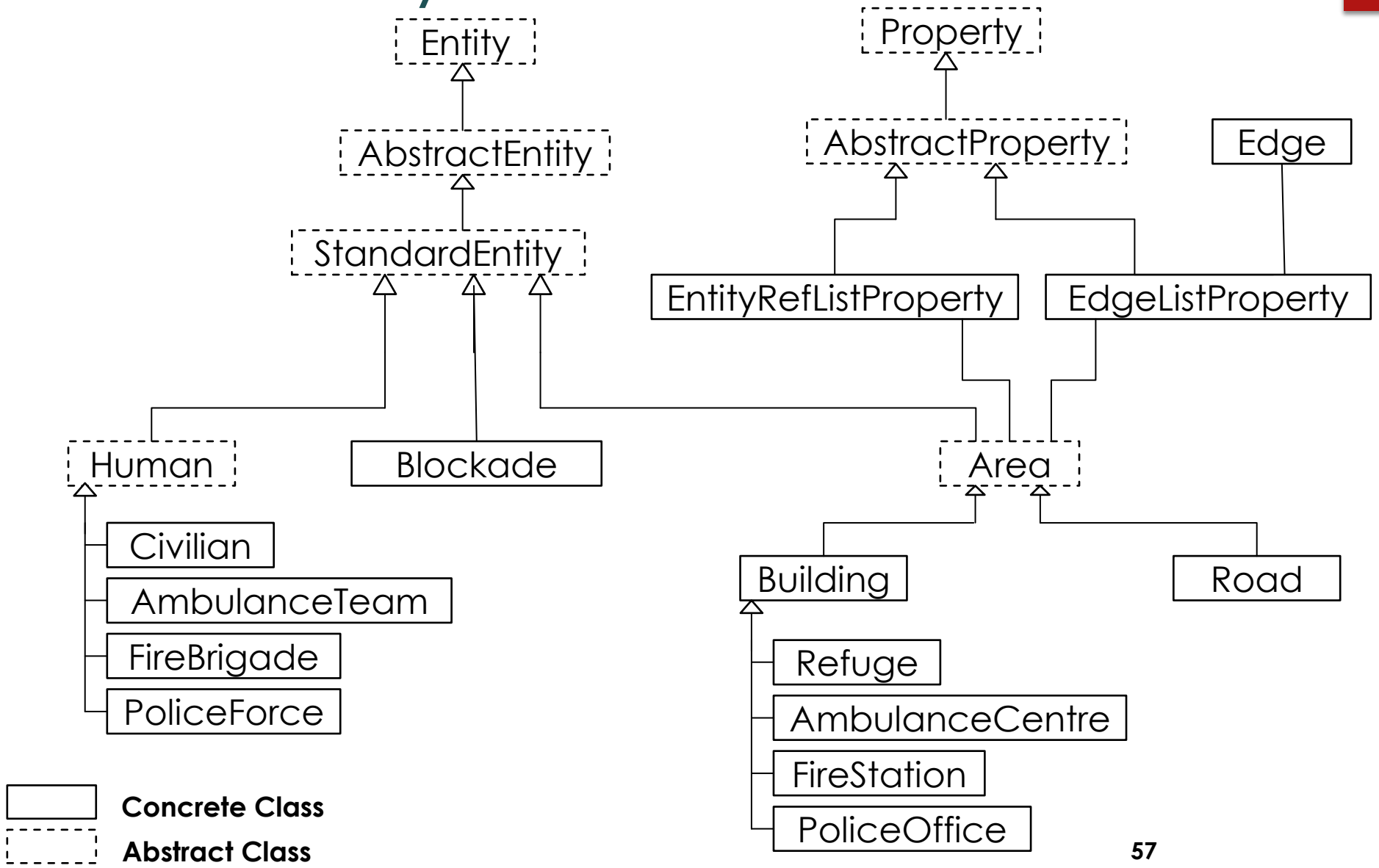
1. Copy the Berlin scenario to a folder with another name
2. Using the Scenario Editor clean the all elements in the map and add
   - 100 civilians
   - 10 PoliceForces, FireBrigades, AmbulanceTeams
   - 1 PoliceStation, FireStation, AmbulanceCentre
   - 4 Refuges with capacity of 3
3. Alter in the scenario configuration
   I. number of cycles (timesteps) from 250 to 200
   II. no radio communication
4. Execute the Rescue simulator and SampleAgent team in the created scenario

Simulator Structure

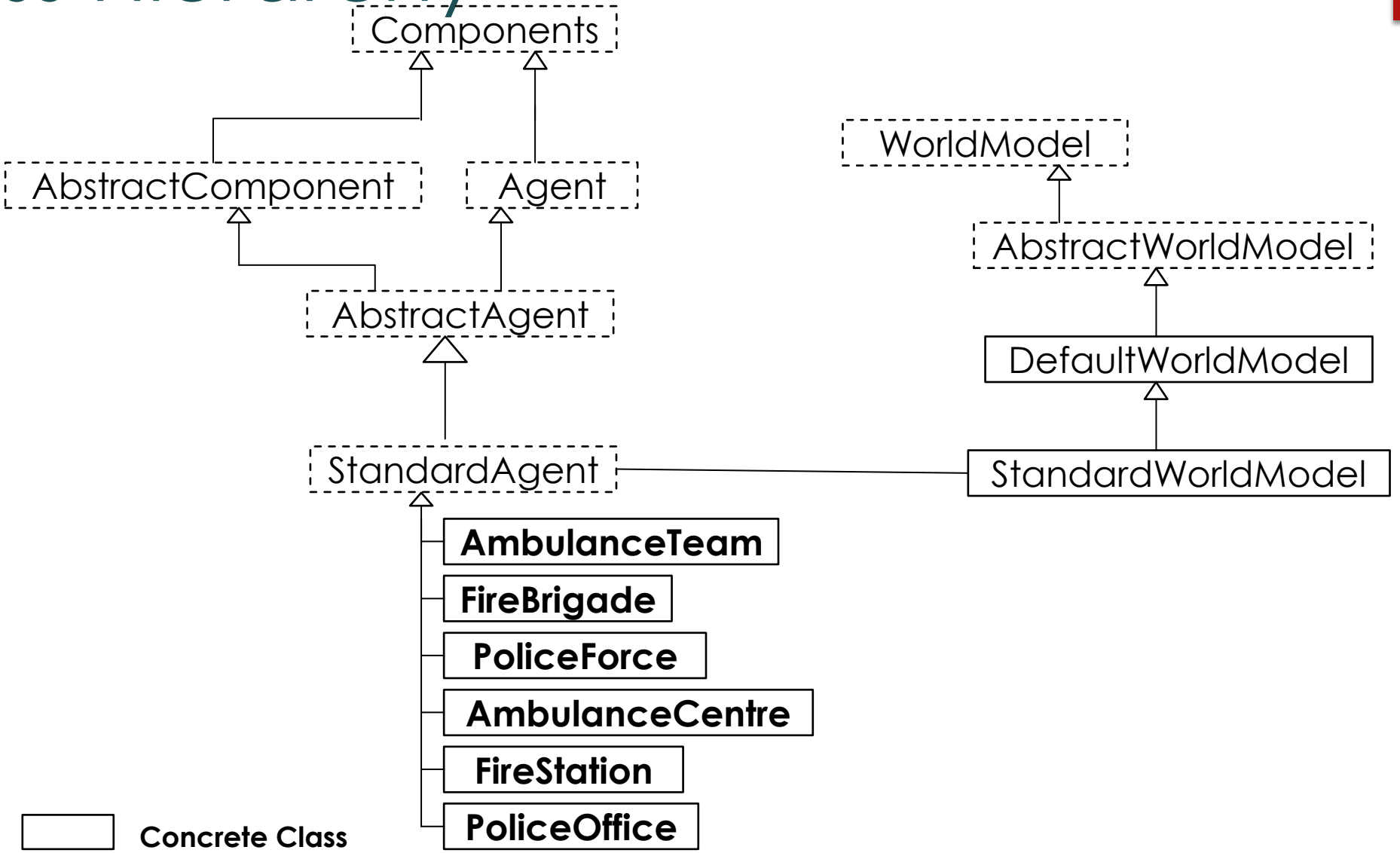
Simulator Structure

- **Class Hierarchy**
- **Objects**
  - **World**
  - **Building**
  - **Road**
  - **Blockade**

Class Hierarchy

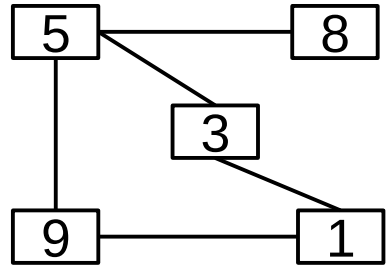


Class Hierarchy



Object - World

- Represent the simulation environment
- Composed of a set of entities, like Buildings, Roads and Humans
- The **Road** and **Building** entities form a connected graph
- Instance of the **StandardWorldModel** class
- Accessed through the global attribute **model**



- Useful methods in the **StandardWorldModel** class

  int getDistance(EntityID first, EntityID second)

  int getDistance(StandardEntity first, StandardEntity second)

  Return the Euclidean distance between two entities

Object - World

- Useful methods in the **StandardWorldModel** class

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

  Return the set of entities of one or more types

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

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

  Return the set of entities inside a range

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

  Return the set of entities inside the coordinates (x1, y1) (x2, y2)

Object - Building

- Represent buildings
- Extend the **Area** class
- Instance of **Building** class

Property	Description
ID	Unique identification number of the building
Brokenness	Amount of damage inflicted in the building
Fieryness	Fire intensity
Temperature	Temperature
TotalArea	Total area including all floors
Blockades	List of blockades in the building area

Object - Building

- Useful methods in the **Building** class

  List<EntityID> getNeighbours()

  Return list of neighbor entities

  int getBrokenness()

  Return the damage suffered by the building

  int getFieryness()

  Return fire intensity

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

Object - Building

- Useful methods in the **Building** class

  int getTemperature()

  Return temperature

  int getTotalArea()

  Return total area

  int getBuildingCode()

  Return type of building material

Code	Type	Transmission Rate
0	Wood	1,8
1	Steel	1,8
2	Concrete	1,0

Object - Building

- Useful methods in the **Building** class

  List<EntityID> getBlockades()

  Return the list of blockades in the building area

Object - Road

- Represent the road ways
- Extend the **Area** class
- Instance of **Road** class

Property	Description
ID	Unique identification number of the road
Blockades	List of blockades in the road area

Object - Road

- Useful methods in the **Road** class

  List<EntityID> getBlockades()

  Return the list of blockades in the road area

  List<EntityID> getNeighbours()

  Return the list of neighbor entities

Object - Blockade

- Represent blockades
- Instance of the **Blockade** class

Property	Description
ID	Unique identication number of the blockade
Position	Entity where the blockade is located
RepairCost	Cost to clear (repair) the blockade

Object - Blockade

- Useful methods in the **Blockade** class

  EntityID getPosition()

  Return the EntityID of the entity where the blockade is located

  int getRepairCost()

  Return the repair cost of the blockade

Agent Behavior

Agent Types

- Civilian 

- Rescue Agents

Platoon

- Ambulance Team
- Fire Brigade
- Police Force

Center


Ambulance Center


Fire Station


Police Office

Minimal Class Structure

```
public class [NAME_CLASS_AGENT] 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){
    }
}
```

Standard Entity

- **StandardEntityURN**
  - CIVILIAN
  - AMBULANCE_TEAM
  - AMBULANCE_CENTRE
  - FIRE_BRIGADE
  - FIRE_STATION
  - POLICE_FORCE
  - POLICE_OFFICE

Agent Methods

- **protected EnumSet<StandardEntityURN> getRequestedEntityURNsEnum()**

  Return the agent type implemented

- **protected void postConnect()**

  Method executed once the agent connects to the simulator kernel before the simulation begins. Used to perform information pre-processing.

  The agents have a time limit to finish executing the postConnect method (default: 5 minutes)

Agent Methods

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

  Implement the agent behavior. Called every simulation cycle.

  Limited time to execute (default 30 seconds)

Accessing Configuration Parameters

this.config.getIntValue([key])

where, **[key]** is the name of the parameter in the configuration file

- Example

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

Example 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){
      ...
    }
}
```

StandardAgent

- Agents extend the **StandardAgent**

Property	Description
ID	Unique identification number of the agent
X	Agent X coordinate in the map
Y	Agent Y coordinate in the map
Buriedness	How much the agent is buried
HP	Health index
Damage	Health index rate of decrease
Position	Entity where the agent is located

StandardAgent

- Useful methods in the **StandardAgent** class

  int getBuriedness()

  Return the amount the agent is buried

  int getHP()

  Return the agent's health index. Zero means the agent is dead.

  int getDamage()

  Retunr the health index rate of decrease

StandardAgent

- Useful methods in the **StandardAgent** class

  EntityID getPosition()

  Return the EntityID of the entity where the agent is located

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

  Return the X and Y coordinate of the agent in the map

Capabilities

Type	Capability
Civilian	sense, hear, say, move
Ambulance Team	sense, hear, say, move, communicate via radio, rescue, load, unoad
Fire Brigade	sense, hear, say, move, communicate via radio, rescue, extinguish, rest
Police Force	sense, hear, say, move, communicate via radio, clear
Ambulance Centre	hear, communicate via radio
Fire Station	hear, communicate via radio
Police Office	hear, communicate via radio

Capabilities

- **Sense**

  Enable the agent to sense the environment inside a certain range. The perceptions are received through the **changed** parameter (**ChangeSet** class) in the **think** method.

  The key **perception.los.max_distance** in the **perception.cfg** file defines the perception range.

  ChangeSet class method

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

  Return the set of entities that changed in the last simulation cycle inside the range of perception.

Capabilities

- **Hear**

  Enable the agent to receive message from other agents via radio communication. The messages are received through the **heard** parameter (**Command** class) in the **think** method.

Further details in Agent Communication

Capabilities

- **Say**

  Enable the agent to send a short-distance voice message.

- **Speak**

  Enable the agent to broadcast a long-distance message.

Further details in Agent Communication

Capabilities

- **Move**

  Enable the agent to move in the environment.

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

  Command to move the agent through a sequence of entities

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

  Command to move the agent through a sequence entities or a specific X and Y coordinate in the map

Capabilities

- **Clean**

  Enable the agent to remove a blockade.

  void sendClear(int time, EntityID target)

  Command to clean a specific blockade (**target**)

  void sendClear(int time, int X, int Y)

  Command to clean a specific X and Y coordinate

Capabilities

- **Extinguish**

  Enable the agent to throw water in a building.

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

  Command used to throw a specific quatity of water (**power**) in the building (**target**)

Capabilities

- **Rest**

  Enable the agent to rest on top of a refuge to fill up its water tank.

  void sendRest(int time)

  Command used to indicate that the agent wants to fill up its water tank when applied positioned on a refuge

Capabilities

- **Rescue**

  Enable the agent to unbury a buried agent.

  void sendRescue(int time, EntityID target)

  Command to unbury a buried agent (**target**)

Capabilities

- **Load**

  Enable to load an agent for transport.

  void sendLoad(int time, EntityID target)

  Command to load an unburied agent (**target**)

Capabilities

- **Unload**

  Enableto unload a transported agent.

  void sendUnload(int time)

  Command to unload an agent

Hands-on

1. Change the FireBrigade behavior (sampleagent.FireBrigade) to prioritize the rescue of Civilians
   - Look at the **getTargets()** method

2. Change the AmbulanceTeam behavior (sampleagent.AmbulanceTeam) to prioritize the rescue of agents with greater HP
   - Look at the **getTargets()** method
   - Currently the priority is based on distance

Agent Communication

Types of Communication

- **Voice**
  - Limited communication distance
  - Single communication channel (id 0)

- **Radio Communication**
  - No communication distance
  - Require subscription to the communication channel
  - Limited number of channels an agent can subscribe

Capabilities

- **Hear**

  Enable the agent to receive message from other agents via radio communication. The messages are received through the **heard** parameter (**Command** class) in the **think** method.

Capabilities

- **Say**

  Enable the agent to send a short-distance voice message. The deafult distance is 30 meters.

  void sendSay(int time, byte[] data)

  Command used to send a voice message to the environment.

Capabilities

- **Speak**

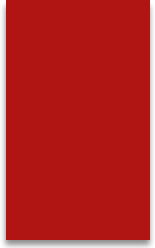
  Enable the agent to broadcast a long-distance message.

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

  Command to subscribe to a communication channel.

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

  Command to send a message in a communication channel.



Thank You!!