Agent Development Framework (ADF) Manual

RoboCup Rescue Simulation Team
Version 2.3, September 27, 2021

Table of Contents

1. Purpose

The manual instructs how to install and execute the RoboCup Rescue Simulation Agent Development Framework (ADF) Sample Agents, and how to implement a new team of agents using the ADF Sample Agents.

2. Installation

This manual assumes the agents will run in a Linux machine even though it is possible to run them in Microsoft Windows or Apple macOS. We recommend to use Linux because it is open-source and most of the distributions have a good support from the users' community. If you have never used Linux before and intend to, we recommend starting with a user-friendly distribution, such as Ubuntu or Fedora.

2.1. Software Requirements

- Java OpenJDK 11+
- Git
- Gradle

2.2. Download

You can download the sample agents with ADF by cloning the `https://github.com/roborescue/rcrs-adf-sample` repository. Clone this repository using the command

```
git clone https://github.com/roborescue/rcrs-adf-sample.git
```

2.3. Directories

The `rcrs-adf-sample` contains multiple directories. The important directories are:

- `config/`: configuration file of agents
- `src/`: agents' source codes
- `precomp_data`: results of a precomputation for each type of agents
- `build/`: agents' Java classes
- `library/`: libraries used by agents

2.4. Compiling

Execute the steps below to compile the ADF Sample Agent.

1. Change to the directory `rcrs-adf-sample`
2. Compile the project using the commands

```
$ ./gradlew clean
$ ./gradlew build
```

2.5. Example

In Ubuntu, the installation proceeds according to the following commands.

Installation on Ubuntu

```
$ git clone https://github.com/roborescue/rcrs-adf-sample.git
$ cd rcrs-adf-sample
$ ./gradlew clean
$ ./gradlew build
```

3. Running

There are two modes of execution of the simulation server and ADF Sample Agent: **Precomputation** and **Normal**.

3.1. Precomputation Mode

In the precomputation mode, the simulator connects one agent of each type and allows them to write the computation results.

The sequence of commands to run the simulation server in precomputation mode are:

Running simulation server in precomputation mode

```
$ cd rcrs-server
$ cd boot
$ bash start-precompute.sh
```

See RoboCup Rescue Simulator Manual for further information on how to compile and run the RoboCup Rescue Simulator server.

After running the simulation server for the precomputation, move to the ADF directory on another terminal window and run the agents executing the commands:

Running Sample Agents in precomputation mode

```
$ bash launch.sh -t 1,0,1,0,1,0 -h localhost -pre 1 & APID=$! ; sleep 120 ; kill
$APID
[START] Connect to server (host:localhost, port:7000)
[INFO] Connected - adf.agent.platoon.PlatoonFire@756ec19c (PRECOMPUTATION)
[INFO] Connected - adf.agent.platoon.PlatoonPolice@366bbbe (PRECOMPUTATION)
[INFO] Connected - adf.agent.platoon.PlatoonAmbulance@2a453513 (PRECOMPUTATION)
********************
[FINISH] Connect PoliceForce (success:1)
[FINISH] Connect AmbulanceTeam (success:1)
[FINISH] Connect FireBrigade (success:1)
[FINISH] Done connecting to server (3 agents)
```

Once the precomputation is completed, push *Control-C* and type `sh kill.sh` to stop the simulation server of running.

Stop running simulation server in precomputation mode

```
Control-C
$ bash kill.sh
```

3.2. Normal Mode

In the normal mode, the simulator connects all agents defined in the scenario and allows them to use the precomputation output.

The sequence of commands to run the simulation server in normal mode are:

Running simulation server in normal mode

```
$ cd rcrs-server
$ cd boot
$ bash start-comprun.sh
```

See RoboCup Rescue Simulator Manual for further information on how to compile and run the RoboCup Rescue Simulator server.

After running the simulation server, move to the ADF directory on the other terminal window and run the agents using the commands:

Running Sample Agents in normal mode

```
$ bash launch.sh -all
[FINISH] Done connecting to server (3 agents)
```

4. Develop your own agents using ADF

This section explain how to implement your agents using ADF samples.

4.1. Files to develop your agents

You can develop your own agents codes using only the files in the directories:

`src/adf/sample/centralized`

source codes for *central agents*. This is the type of agents whose only interaction with the world is through radio communication. There are three types of central agents: **Ambulance Centers**, **Fire Stations** and **Police Office**, and they are represented as buildings in the simulation server.

`src/adf/sample/extraction`

codes of combining actions described in the directory below.

`src/adf/sample/module`

concrete codes of algorithms, e.g., path planning, clustering, target detection, etc. The directory contains two directories:

- `src/adf/sample/module/algorithm`

- `src/adf/sample/module/complex`

NOTE You **must not** make any changes of files in `src/adf/sample/tactics`. This is the restriction for our current competition rule.

You should fundamentally copy the sample codes, not edit them. The reason is that the sample codes would be used if ADF could not find your own codes. You can easily change reference to your modules by modifying `src/adf/config/module.cfg`. The usage of the file is described below.

4.2. Workflow for coding your agents

The steps necessary to code your own agents are:

1. Copy sample codes related to agents which you want to create,
2. Edit the copied files.
3. Edit `src/adf/config/module.cfg` according to the edited files.
4. Compile and run.

4.3. Modules' configuration files

The modules configuration file `src/adf/config/module.cfg` indicates which codes would be used as agents' module. Listing 1 shows part of the modules configuration file. The left-hand side of the colon indicates the module name, the right-hand side is the class name. In most cases, modules of which targets' problems are the same should refer to an identical class for all agent types. The example in Listing 1 is in `TacticsAmbulanceTeam.Search` and `TacticsFireBrigade.Search` indicates that both modules refer to `adf.sample.module.complex.SampleSearch`. An usage example is shown in Section 4.4.3.

Listing 1. Module's configuration file

```
TacticsAmbulanceTeam.HumanDetector :
adf.sample.module.complex.SampleHumanDetector
TacticsAmbulanceTeam.Search : adf.sample.module.complex.SampleSearch

TacticsAmbulanceTeam.ActionTransport : adf.sample.extaction.ActionTransport
TacticsAmbulanceTeam.ActionExtMove : adf.sample.extaction.ActionExtMove

TacticsAmbulanceTeam.CommandExecutorAmbulance :
adf.sample.centralized.CommandExecutorAmbulance
TacticsAmbulanceTeam.CommandExecutorScout :
adf.sample.centralized.CommandExecutorScout

TacticsFireBrigade.BuildingDetector :
adf.sample.module.complex.SampleBuildingDetector
TacticsFireBrigade.Search : adf.sample.module.complex.SampleSearch

TacticsFireBrigade.ActionFireFighting : adf.sample.extaction.ActionFireFighting
TacticsFireBrigade.ActionExtMove : adf.sample.extaction.ActionExtMove
```

4.4. Example of implementing A* algorithm for Path Planning algorithm

4.4.1. Copy the ADF Sample code

First, you should copy the sample code for path planning which is `SamplePathPlanning.java`. The example is described below. Note that the second command is split into two lines because of space limitations, but it should be entered as a single line.

Copy the Sample Path Planning

```
$ mkdir -p src/myteam/module/algorithm
$ cp src/adf/sample/module/algorithm/SamplePathPlanning.java
src/myteam/module/algorithm/AStarPathPlanning.java
```

4.4.2. Edit the ADF Sample code

Listing 2 is the code of `SamplePathPlanning.java`, which has Dijkstra's algorithm. You should edit 1st line, 18th line and 27th line. You would implement your own code in the method `calc()`, and remove the method `isGoal()` that is only used by `calc()`. Listing 3 shows the results of editing these lines.

You must implement the method `calc()` to get its calculation result by the method `getResult()`. The type of `getResult()` returning is `List<EntityID>`.

Listing 4 indicates the contents of the method `calc()`. In addition, you should write the new private class `Node` which is used by the method `calc()`. The code is shown in Listing 5. It must be put in the file `AStarPathPlanning.java`.

Listing 2. `SamplePathPlanning.java` *file*

```
package adf.sample.module.algorithm; // Edit this line

import adf.agent.communication.MessageManager;
import adf.agent.develop.DevelopData;
import adf.agent.info.AgentInfo;
import adf.agent.info.ScenarioInfo;
import adf.agent.info.WorldInfo;
import adf.agent.module.ModuleManager;
import adf.agent.precompute.PrecomputeData;
import adf.component.module.algorithm.PathPlanning;
import rescuecore2.misc.collections.LazyMap;
import rescuecore2.standard.entities.Area;
import rescuecore2.worldmodel.Entity;
import rescuecore2.worldmodel.EntityID;

import java.util.*;

public class SamplePathPlanning extends PathPlanning { // Edit this line

  private Map<EntityID, Set<EntityID>> graph;

  private EntityID from;
  private Collection<EntityID> targets;
  private List<EntityID> result;
  // Edit the following line
  public SamplePathPlanning(AgentInfo ai, WorldInfo wi, ScenarioInfo si,
```

```
ModuleManager moduleManager, DevelopData developData) {
    super(ai, wi, si, moduleManager, developData);
    this.init();
  }

  private void init() {
    Map<EntityID, Set<EntityID>> neighbours = new LazyMap<EntityID,
Set<EntityID>>() {
      @Override
      public Set<EntityID> createValue() {
          return new HashSet<>();
      }
    };
    for (Entity next : this.worldInfo) {
      if (next instanceof Area) {
        Collection<EntityID> areaNeighbours = ((Area) next).getNeighbours();
        neighbours.get(next.getID()).addAll(areaNeighbours);
      }
    }
    this.graph = neighbours;
  }

  @Override
  public List<EntityID> getResult() {
    return this.result;
  }

  @Override
  public PathPlanning setFrom(EntityID id) {
    this.from = id;
    return this;
  }

  @Override
  public PathPlanning setDestination(Collection<EntityID> targets) {
    this.targets = targets;
    return this;
  }

  @Override
  public PathPlanning updateInfo(MessageManager messageManager) {
    super.updateInfo(messageManager);
    return this;
  }

  @Override
  public PathPlanning precompute(PrecomputeData precomputeData) {
    super.precompute(precomputeData);
    return this;
  }

  @Override
  public PathPlanning resume(PrecomputeData precomputeData) {
    super.resume(precomputeData);
    return this;
  }

  @Override
  public PathPlanning preparate() {
    super.preparate();
    return this;
  }
```

```
  @Override
  public PathPlanning calc() { // Renew this method (implement your algorithm
here)
    List<EntityID> open = new LinkedList<>();
    Map<EntityID, EntityID> ancestors = new HashMap<>();
    open.add(this.from);
    EntityID next;
    boolean found = false;
    ancestors.put(this.from, this.from);
    do {
      next = open.remove(0);
      if (isGoal(next, targets)) {
        found = true;
        break;
      }
      Collection<EntityID> neighbours = graph.get(next);
      if (neighbours.isEmpty()) {
        continue;
      }
      for (EntityID neighbour : neighbours) {
        if (isGoal(neighbour, targets)) {
          ancestors.put(neighbour, next);
          next = neighbour;
          found = true;
          break;
        }
        else {
          if (!ancestors.containsKey(neighbour)) {
            open.add(neighbour);
            ancestors.put(neighbour, next);
          }
        }
      }
    } while (!found && !open.isEmpty());
    if (!found) {
      // No path
      this.result = null;
    }
    // Walk back from goal to this.from
    EntityID current = next;
    LinkedList<EntityID> path = new LinkedList<>();
    do {
      path.add(0, current);
      current = ancestors.get(current);
      if (current == null) {
        throw new RuntimeException("Found a node with no ancestor! Something is
broken.");
      }
    } while (current != this.from);
    this.result = path;
    return this;
  }
  // Remove the method (it is only used by calc()).
  private boolean isGoal(EntityID e, Collection<EntityID> test) {
    return test.contains(e);
  }
}
```

Listing 3. `AStartPlanning.java` *file*

```
package myteam.module.algorithm; // Position of the file

import adf.agent.communication.MessageManager;
import adf.agent.develop.DevelopData;
import adf.agent.info.AgentInfo;
import adf.agent.info.ScenarioInfo;
import adf.agent.info.WorldInfo;
import adf.agent.module.ModuleManager;
import adf.agent.precompute.PrecomputeData;
import adf.component.module.algorithm.PathPlanning;
import rescuecore2.misc.collections.LazyMap;
import rescuecore2.standard.entities.Area;
import rescuecore2.worldmodel.Entity;
import rescuecore2.worldmodel.EntityID;

import java.util.*;

public class AStarPathPlanning extends PathPlanning { // Same as the file name

  private Map<EntityID, Set<EntityID>> graph;

  private EntityID from;
  private Collection<EntityID> targets;
  private List<EntityID> result;
  // Same as the file name
  public AStarPathPlanning(AgentInfo ai, WorldInfo wi, ScenarioInfo si,
ModuleManager moduleManager, DevelopData developData) {
    super(ai, wi, si, moduleManager, developData);
    this.init();
  }
```

Listing 4. `calc()` *method*

```
  @Override
  public PathPlanning calc() {
    List<EntityID> open = new LinkedList<>();
    List<EntityID> close = new LinkedList<>();
    Map<EntityID, Node> nodeMap = new HashMap<>();
    open.add(this.from);
    nodeMap.put(this.from, new Node(null, this.from));
    close.clear();

    while (true) {
      if (open.size() < 0) {
        this.result = null;
        return this;
      }
      Node n = null;
      for (EntityID id : open) {
        Node node = nodeMap.get(id);
        if (n == null) {
          n = node;
        } else if (node.estimate() < n.estimate()) {
          n = node;
        }
      }
      if (targets.contains(n.getID())) {
        List<EntityID> path = new LinkedList<>();
        while (n != null) {
          path.add(0, n.getID());
          n = nodeMap.get(n.getParent());
        }
        this.result = path;
        return this;
      }
      open.remove(n.getID());
      close.add(n.getID());

      Collection<EntityID> neighbours = this.graph.get(n.getID());
      for (EntityID neighbour : neighbours) {
        Node m = new Node(n, neighbour);
        if (!open.contains(neighbour) && !close.contains(neighbour)) {
          open.add(m.getID());
          nodeMap.put(neighbour, m);
        }
        else if (open.contains(neighbour) && m.estimate() <
nodeMap.get(neighbour).estimate()) {
          nodeMap.put(neighbour, m);
        }
        else if (!close.contains(neighbour) && m.estimate() <
nodeMap.get(neighbour).estimate()) {
          nodeMap.put(neighbour, m);
        }
      }
    }
  }
```

Listing 5. `Node` *class*

```
private class Node {
  EntityID id;
  EntityID parent;

  double cost;
  double heuristic;

  public Node(Node from, EntityID id) {
    this.id = id;

    if (from == null) {
      this.cost = 0;
    } else {
      this.parent = from.getID();
      this.cost = from.getCost() + worldInfo.getDistance(from.getID(), id);
    }

    this.heuristic = worldInfo.getDistance(id, targets.toArray(new
EntityID[targets.size()])[0]);
  }

  public EntityID getID() {
    return id;
  }

  public double getCost() {
    return cost;
  }

  public double estimate() {
    return cost + heuristic;
  }

  public EntityID getParent() {
    return this.parent;
  }
}
```

4.4.3. Edit the Modules' configuration file

You must edit the module configuration file `src/adf/config/module.cfg` related to a path planning to use your code. Listing 6 and Listing 7 show the part of the default `module.cfg` and the part of the edited `module.cfg` where the lines related to a path planning are changed. In this case, all `adf.sample.module.algorithm.SamplePathPlanning` in the file are replaced with `myteam.module.algorithm.AStarPathPlanning`. If you would like to use the code in some modules, you can indicate that the only modules refer to it.

Listing 6. Default `module.cfg`

```
SampleRoadDetector.PathPlanning : adf.sample.module.algorithm.SamplePathPlanning
SampleSearch.PathPlanning.Ambulance :
adf.sample.module.algorithm.SamplePathPlanning
SampleSearch.PathPlanning.Fire : adf.sample.module.algorithm.SamplePathPlanning
SampleSearch.PathPlanning.Police :
adf.sample.module.algorithm.SamplePathPlanning
ActionExtClear.PathPlanning : adf.sample.module.algorithm.SamplePathPlanning
ActionExtMove.PathPlanning : adf.sample.module.algorithm.SamplePathPlanning
ActionFireFighting.PathPlanning : adf.sample.module.algorithm.SamplePathPlanning
ActionTransport.PathPlanning : adf.sample.module.algorithm.SamplePathPlanning
CommandExecutorAmbulance.PathPlanning :
adf.sample.module.algorithm.SamplePathPlanning
CommandExecutorFire.PathPlanning :
adf.sample.module.algorithm.SamplePathPlanning
CommandExecutorPolice.PathPlanning :
adf.sample.module.algorithm.SamplePathPlanning
CommandExecutorScout.PathPlanning :
adf.sample.module.algorithm.SamplePathPlanning
CommandExecutorScoutPolice.PathPlanning :
adf.sample.module.algorithm.SamplePathPlanning
```

Listing 7. Edited `module.cfg`

```
SampleRoadDetector.PathPlanning : myteam.module.algorithm.AStarPathPlanning
SampleSearch.PathPlanning.Ambulance : myteam.module.algorithm.AStarPathPlanning
SampleSearch.PathPlanning.Fire : myteam.module.algorithm.AStarPathPlanning
SampleSearch.PathPlanning.Police : myteam.module.algorithm.AStarPathPlanning
ActionExtClear.PathPlanning : myteam.module.algorithm.AStarPathPlanning
ActionExtMove.PathPlanning : myteam.module.algorithm.AStarPathPlanning
ActionFireFighting.PathPlanning : myteam.module.algorithm.AStarPathPlanning
ActionTransport.PathPlanning : myteam.module.algorithm.AStarPathPlanning
CommandExecutorAmbulance.PathPlanning :
myteam.module.algorithm.AStarPathPlanning
CommandExecutorFire.PathPlanning : myteam.module.algorithm.AStarPathPlanning
CommandExecutorPolice.PathPlanning : myteam.module.algorithm.AStarPathPlanning
CommandExecutorScout.PathPlanning : myteam.module.algorithm.AStarPathPlanning
CommandExecutorScoutPolice.PathPlanning :
myteam.module.algorithm.AStarPathPlanning
```