Agent Development Framework (ADF) Manual

RoboCup Rescue Simulation Team

Version 4.1, February 03, 2023

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 17
- Git
- Utilities like `wget`, `bash`, `xterm`, `tar`, `gzip`, etc.
  NOTE: If you are using Ubuntu, all of these utilities are present in the default software repositories.

2.2. Download

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

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

2.3. Directories

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

- `config/`: ADF and Agent Modules' configuration files
- `src/`: Sample agents' source codes
- `precomp_data`: results of a precomputation for each type of agents

2.4. Compiling

Execute the steps below to compile the ADF Sample Agent.

```
$ cd adf-sample-agent-java
```

```
$ ./gradlew clean 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 persistently.

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

```
$ cd rcrs-server
$ cd scripts
$ ./start-precompute.sh -m ../maps/test/maps -c ../maps/test/config
```

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 Sample Agent directory on another terminal window and run the agents executing the commands:

```
$ 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:27931)
[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, press *Control-C* and type `bash kill.sh` to stop the simulation server of running.

```
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 (see Section 3.1).

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

```
$ cd rcrs-server
$ cd scripts
$ 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 Sample Agent directory on another terminal window and run the agents using the commands:

```
$ 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 the ADF Sample Agent as the starting point.

4.1. Workflow for coding your agents

The steps necessary to code your own agents are:

- Implement the customized modules
- Change the `config/module.cfg` to point to the customized modules

4.2. Customize modules

ADF is a modular framework whose modules were define in the `adf-core-java` (https://github.com/roborescue/adf-core-java) repository together with a set of default implementations. To implement your own team of agents, you have to implement the modules' Java interfaces correspondent to the behavior you want to customize.

The default implementations of the modules' Java interfaces is available under the package `impl` in the `adf-core-java` repository. There you find default implementations for:

- `adf.impl.centralized`: source code of the *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.

- `adf.impl.extraction`: source code of the possible actions available to agents.
- `adf.impl.module`: source code of the algorithms, e.g., path planning, clustering, target detection, etc. representing the agents' behavior. The modules are split into
    - `adf.impl.module.algorithm`
    - `adf.impl.module.comm`
    - `adf.impl.module.complex`

To customize any of these modules, you can copy modules' file you want to customize to you team agents' repository and make changes to the implementation. Then you need to change the references to your modules by modifying `config/module.cfg` file (see below).

4.3. Modules' configuration file

The modules configuration file `config/module.cfg` indicates which class will 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 `DefaultTacticsAmbulanceTeam.Search` and `DefaultTacticsFireBrigade.Search` indicates that both modules refer to `sample_team.module.complex.SampleSearch`. An usage example is shown in Section 4.4.3.

Listing 1. *Part of a module configuration file*

```
## DefaultTacticsAmbulanceTeam
DefaultTacticsAmbulanceTeam.HumanDetector :
sample_team.module.complex.SampleHumanDetector
DefaultTacticsAmbulanceTeam.Search :
sample_team.module.complex.SampleSearch
DefaultTacticsAmbulanceTeam.ExtActionTransport :
adf.impl.extaction.DefaultExtActionTransport
DefaultTacticsAmbulanceTeam.ExtActionMove :
adf.impl.extaction.DefaultExtActionMove
DefaultTacticsAmbulanceTeam.CommandExecutorAmbulance :
adf.impl.centralized.DefaultCommandExecutorAmbulance
DefaultTacticsAmbulanceTeam.CommandExecutorScout :
adf.impl.centralized.DefaultCommandExecutorScout

## DefaultTacticsFireBrigade
DefaultTacticsFireBrigade.HumanDetector :
sample_team.module.complex.SampleHumanDetector
DefaultTacticsFireBrigade.Search :
sample_team.module.complex.SampleSearch
```

```
DefaultTacticsFireBrigade.ExtActionFireRescue :
adf.impl.extaction.DefaultExtActionFireRescue
DefaultTacticsFireBrigade.ExtActionMove :
adf.impl.extaction.DefaultExtActionMove
DefaultTacticsFireBrigade.CommandExecutorFire :
adf.impl.centralized.DefaultCommandExecutorFire
DefaultTacticsFireBrigade.CommandExecutorScout :
adf.impl.centralized.DefaultCommandExecutorScout
```

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

In this example, you will learn how to implement the A* Path Planning algorithm in a module and how to setup the ADF Sample Agent to use it instead of the Dijkstra Path Planning. Here we assume that you will apply the changes to the `adf-sample-agent-java` repository.

4.4.1. Copy the Dijkstra Path Planning file

First, you should copy the Dijkstra path planning (`src/main/java/adf/impl/module/algorithm/DijkstraPathPlanning.java`) from the `adf-core-java` repository to the `adf-sample-agent-java` repository (`src/main/java/sample_team/module/algorithm`).

```
$ cd adf-sample-agent-java
$ mkdir -p src/main/java/sample_team/module/algorithm
$ cp ../adf-core-
java/src/main/java/adf/impl/module/algorithm/DijkstraPathPlanning.j
ava
src/main/java/sample_team/module/algorithm/AStarPathPlanning.java
```

4.4.2. Edit the Dijkstra code

Listing 2 is the code of `DijkstraPathPlanning.java`, which implements the Dijkstra's algorithm. You should edit line 1 and 23th as well as replace the code in the method `calc()` starting on line 96. Remove the method `isGoal()` that is only used by the Dijkstra `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.

Listing 2. `DijkstraPathPlanning.java` *file*

```java
package adf.impl.module.algorithm; // Edit this line

import adf.core.agent.communication.MessageManager;
import adf.core.agent.develop.DevelopData;
import adf.core.agent.info.AgentInfo;
import adf.core.agent.info.ScenarioInfo;
import adf.core.agent.info.WorldInfo;
import adf.core.agent.module.ModuleManager;
import adf.core.agent.precompute.PrecomputeData;
import adf.core.component.module.algorithm.PathPlanning;
import java.util.Collection;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Set;
import rescuecore2.misc.collections.LazyMap;
import rescuecore2.standard.entities.Area;
import rescuecore2.worldmodel.Entity;
import rescuecore2.worldmodel.EntityID;

public class DijkstraPathPlanning extends PathPlanning { // Edit this line

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

  private EntityID from;
  private Collection<EntityID> targets;
  private List<EntityID> result;

  public DijkstraPathPlanning(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<>();
          }
```

```
44          };
45      for (Entity next : this.worldInfo) {
46        if (next instanceof Area) {
47          Collection<EntityID> areaNeighbours = ((Area) next
    ).getNeighbours();
48          neighbours.get(next.getID()).addAll(areaNeighbours);
49        }
50      }
51      this.graph = neighbours;
52    }
53
54    @Override
55    public List<EntityID> getResult() {
56      return this.result;
57    }
58
59    @Override
60    public PathPlanning setFrom(EntityID id) {
61      this.from = id;
62      return this;
63    }
64
65    @Override
66    public PathPlanning setDestination(Collection<EntityID>
    targets) {
67      this.targets = targets;
68      return this;
69    }
70
71    @Override
72    public PathPlanning updateInfo(MessageManager messageManager)
    {
73      super.updateInfo(messageManager);
74      return this;
75    }
76
77    @Override
78    public PathPlanning precompute(PrecomputeData precomputeData)
    {
79      super.precompute(precomputeData);
80      return this;
81    }
82
83    @Override
84    public PathPlanning resume(PrecomputeData precomputeData) {
85      super.resume(precomputeData);
86      return this;
87    }
```

```
88
89    @Override
90    public PathPlanning preparate() {
91      super.preparate();
92      return this;
93    }
94
95    @Override
96    public PathPlanning calc() {   // Replace the code in this
    method by the A* Path Planning algorithm
97      List<EntityID> open = new LinkedList<>();
98      Map<EntityID, EntityID> ancestors = new HashMap<>();
99      open.add(this.from);
100     EntityID next;
101     boolean found = false;
102     ancestors.put(this.from, this.from);
103     do {
104       next = open.remove(0);
105       if (isGoal(next, targets)) {
106         found = true;
107         break;
108       }
109       Collection<EntityID> neighbours = graph.get(next);
110       if (neighbours.isEmpty()) {
111         continue;
112       }
113       for (EntityID neighbour : neighbours) {
114         if (isGoal(neighbour, targets)) {
115           ancestors.put(neighbour, next);
116           next = neighbour;
117           found = true;
118           break;
119         } else {
120           if (!ancestors.containsKey(neighbour)) {
121             open.add(neighbour);
122             ancestors.put(neighbour, next);
123           }
124         }
125       }
126     } while (!found && !open.isEmpty());
127     if (!found) {
128       // No path
129       this.result = null;
130     }
131     // Walk back from goal to this.from
132     EntityID current = next;
133     LinkedList<EntityID> path = new LinkedList<>();
134     do {
```

```
135         path.add(0, current);
136         current = ancestors.get(current);
137         if (current == null) {
138           throw new RuntimeException(
139               "Found a node with no ancestor! Something is
    broken.");
140         }
141       } while (current != this.from);
142       this.result = path;
143       return this;
144     }
145
146     private boolean isGoal(EntityID e, Collection<EntityID> test)
    {
147       return test.contains(e);
148     }
149 }
```

Listing 3. `AStartPlanning.java` *file*

```
 1 package sample_team.module.algorithm; // Position of the file
 2
 3 import adf.core.agent.develop.DevelopData;
 4 import adf.core.agent.info.AgentInfo;
 5 import adf.core.agent.info.ScenarioInfo;
 6 import adf.core.agent.info.WorldInfo;
 7 import adf.core.agent.module.ModuleManager;
 8 import adf.core.agent.precompute.PrecomputeData;
 9 import adf.core.component.module.algorithm.PathPlanning;
10 import java.util.Collection;
11 import java.util.HashMap;
12 import java.util.HashSet;
13 import java.util.LinkedList;
14 import java.util.List;
15 import java.util.Map;
16 import java.util.Set;
17 import rescuecore2.misc.collections.LazyMap;
18 import rescuecore2.standard.entities.Area;
19 import rescuecore2.worldmodel.Entity;
20 import rescuecore2.worldmodel.EntityID;
21
22 public class AStarPathPlanning extends PathPlanning {
23
24   private Map<EntityID, Set<EntityID>> graph;
25
26   private EntityID from;
27   private Collection<EntityID> targets;
28   private List<EntityID> result;
```

```
29
30   public AStarPathPlanning(AgentInfo ai, WorldInfo wi,
     ScenarioInfo si, ModuleManager moduleManager, DevelopData
     developData) {
31     super(ai, wi, si, moduleManager, developData);
32     this.init();
33   }
34
35   ...
```

Listing 4. `calc()` *method*

```
1   @Override
2   public PathPlanning calc() {
3     List<EntityID> open = new LinkedList<>();
4     List<EntityID> close = new LinkedList<>();
5     Map<EntityID, Node> nodeMap = new HashMap<>();
6
7     open.add(this.from);
8     nodeMap.put(this.from, new Node(null, this.from));
9     close.clear();
10
11    while (true) {
12      if (open.size() < 0) {
13        this.result = null;
14        return this;
15      }
16
17      Node n = null;
18      for (EntityID id : open) {
19        Node node = nodeMap.get(id);
20
21        if (n == null) {
22          n = node;
23        } else if (node.estimate() < n.estimate()) {
24          n = node;
25        }
26      }
27
28      if (targets.contains(n.getID())) {
29        List<EntityID> path = new LinkedList<>();
30        while (n != null) {
31          path.add(0, n.getID());
32          n = nodeMap.get(n.getParent());
33        }
34
35        this.result = path;
36        return this;
```

```
37        }
38        open.remove(n.getID());
39        close.add(n.getID());
40
41        Collection<EntityID> neighbours = this.graph.get(n.
    getID());
42        for (EntityID neighbour : neighbours) {
43          Node m = new Node(n, neighbour);
44
45          if (!open.contains(neighbour) && !close.contains
    (neighbour)) {
46            open.add(m.getID());
47            nodeMap.put(neighbour, m);
48          } else if (open.contains(neighbour)
49              && m.estimate() < nodeMap.get(neighbour).estimate())
    {
50            nodeMap.put(neighbour, m);
51          } else if (!close.contains(neighbour)
52              && m.estimate() < nodeMap.get(neighbour).estimate())
    {
53            nodeMap.put(neighbour, m);
54          }
55        }
56      }
57    }
```

Listing 5. `Node` *class*

```
1  private class Node {
2      EntityID id;
3      EntityID parent;
4
5      double cost;
6      double heuristic;
7
8      public Node(Node from, EntityID id) {
9        this.id = id;
10
11       if (from == null) {
12         this.cost = 0;
13       } else {
14         this.parent = from.getID();
15         this.cost = from.getCost() + worldInfo.getDistance(from
    .getID(), id);
16       }
17
18       this.heuristic = worldInfo.getDistance(id,
19           targets.toArray(new EntityID[targets.size()])[0]);
```

```
20      }
21
22
23      public EntityID getID() {
24        return id;
25      }
26
27
28      public double getCost() {
29        return cost;
30      }
31
32
33      public double estimate() {
34        return cost + heuristic;
35      }
36
37
38      public EntityID getParent() {
39        return this.parent;
40      }
41    }
42  }
```

4.4.3. Edit the Modules' configuration file

After created the module code, you must edit the module configuration file `config/module.cfg` and replace the modules you would like to use your implementation. Listing 6 and Listing 7 show the part of the default `module.cfg` and the part of the edited `config/module.cfg` where the lines related to a path planning are changed. In this case, all `adf.impl.module.algorithm.DijkstraPathPlanning` are replaced with `sample_team.module.algorithm.AStarPathPlanning`.

Listing 6. *Default* `module.cfg`

```
## SampleSearch
SampleSearch.PathPlanning.Ambulance :
adf.impl.module.algorithm.DijkstraPathPlanning
SampleSearch.Clustering.Ambulance :
adf.impl.module.algorithm.KMeansClustering
SampleSearch.PathPlanning.Fire :
adf.impl.module.algorithm.DijkstraPathPlanning
SampleSearch.Clustering.Fire :
adf.impl.module.algorithm.KMeansClustering
SampleSearch.PathPlanning.Police :
adf.impl.module.algorithm.DijkstraPathPlanning
SampleSearch.Clustering.Police :
```

```
adf.impl.module.algorithm.KMeansClustering
```

Listing 7. *Edited* `module.cfg`

```
## SampleSearch
SampleSearch.PathPlanning.Ambulance :
sample_team.module.algorithm.AStarPathPlanning
SampleSearch.Clustering.Ambulance :
adf.impl.module.algorithm.KMeansClustering
SampleSearch.PathPlanning.Fire :
adf.impl.module.algorithm.AStarPathPlanning
SampleSearch.Clustering.Fire :
adf.impl.module.algorithm.KMeansClustering
SampleSearch.PathPlanning.Police :
adf.impl.module.algorithm.AStarPathPlanning
SampleSearch.Clustering.Police :
adf.impl.module.algorithm.KMeansClustering
```