RoboCup Rescue Simulator and Agent Development Framework Manual

October 14, 2018

Contents

1 Introduction

The purpose of this manual is to facilitate the understanding of the first contact with the RoboCup Rescue Simulation server and to help people interested in participating in RoboCup Rescue Agent Simulation competitions.

2 Installation

This manual assumes the simulator and 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[1] or Fedora[2].

[1] `https://www.ubuntu.com/`
[2] `https://getfedora.org`

2.1 Software Requirements

- Java OpenJDK 8+[3]
- Git
- Gradle
- Utilities like *wget*, *bash*, *xterm*, *tar*, *gzip*, etc.
  If you are using Ubuntu, all of these software are present in the default software repositories.

2.2 Installing RoboCup Rescue Simulation (RCRS) Server

1. Clone the simulation server from `https://github.com/roborescue/rcrs-server`.
2. Change to the directory "rcrs-server".
   (a) If you use macOS, patch the file "boot/functions.sh" like

   "`sed -i -e "/readlink/s/^/#/" boot/functions.sh`".
3. Compile the simulator using the commands `gradle clean` and `gradle completeBuild`.
4. Check the message at the end of the installation. If the installation is successfully completed, you get the message "BUILD SUCCESSFUL"; otherwise you get "BUILD FAILED".

If you are using Ubuntu, the installation proceeds according to the commands below:

Installation on Ubuntu

```
$ git clone https://github.com/roborescue/rcrs-server.git
$ cd rcrs-server
($ sed -i -e "/readlink/s/^/#/" boot/functions.sh # Only required for macOS)
$ gradle clean
$ gradle completeBuild
```

The following message will be appeared if the installation is successfully completed.

Install Completion

```
BUILD SUCCESSFUL in 2s
1 actionable task:  1 executed
```

2.3 Compiling the Agent Development Framework (ADF) Sample Agents

Download the sample agents with ADF by cloning the `https://github.com/roborescue/rcrs-adf-sample.git` repository. Then, you move to `rcrs-adf-sample` directory and compile the sample agents using the script `compile.sh`.

If you are using Ubuntu, you can get and compile the ADF with the following commands:

Download ADF on Ubuntu

```
$ git clone https://github.com/roborescue/rcrs-adf-sample.git
$ cd rcrs-adf-sample
$ ./compile.sh
```

3 Running the ADF Sample Agents on RCRS Server

3.1 Running without Precomputation

To run the sample agents, you must open two terminal windows. One is used to run the simulation server (i.e., the simulator) and the other is used to run the agents.

[3] `https://openjdk.java.net/`

3.1.1 Running the Simulation Server

Use one terminal window and move to the boot directory inside the simulator's folder (`rcrs-server`). Then, type `bash start-comprun.sh`. The sequence of commands are:

Running Simulation Server

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

When the simulation server runs correctly, the window in Fig. 1 will appear.

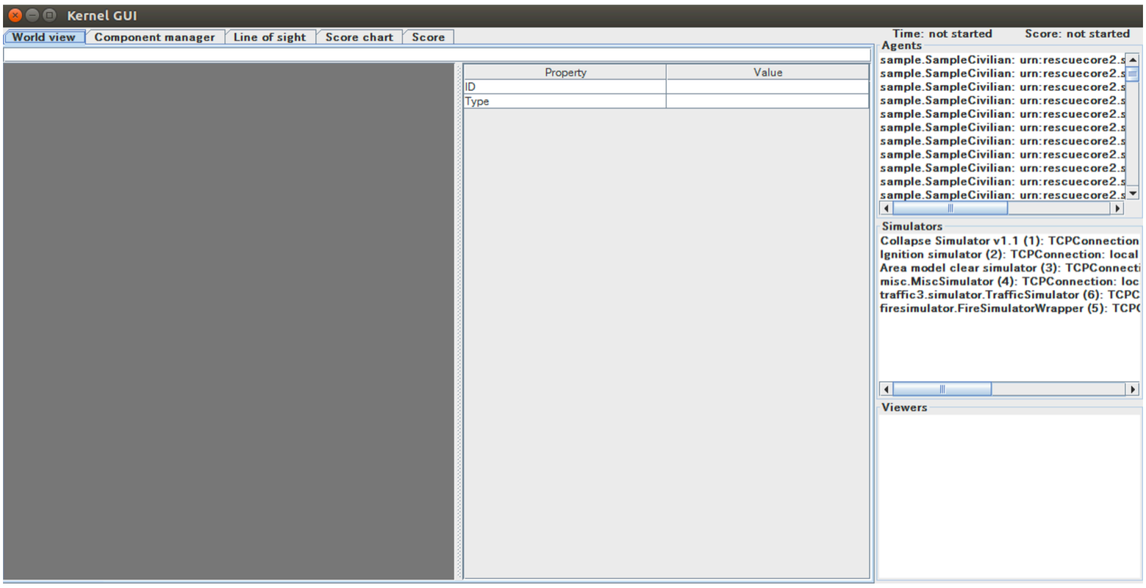

Figure 1: Running the Simulation Server

3.1.2 Running the ADF Sample Agents

After running the simulation server, move to `rcrs-adf-sample` directory on the other terminal window and run the agents by the following commands:

Running Sample Agents

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

If the agents can connect with the simulator, the state of the agents and a city are shown on the left-hand side in the window shown in Fig. 1. Then, the simulation is started as shown in Fig. 2.

3.2 Running with Precomputation

Agents can examine a simulation scenario by performing a precomputation of it before starting the simulation. The length of the precomputation is predefined to 2 minutes in the competition. The precomputation needs two terminal windows likewise.

3.2.1 Running the Simulation Server for Precomputation

Use one terminal window and move to the boot directory inside the simulator's folder (`rcrs-server`). Then, type `bash start-precompute.sh`. These commands are:

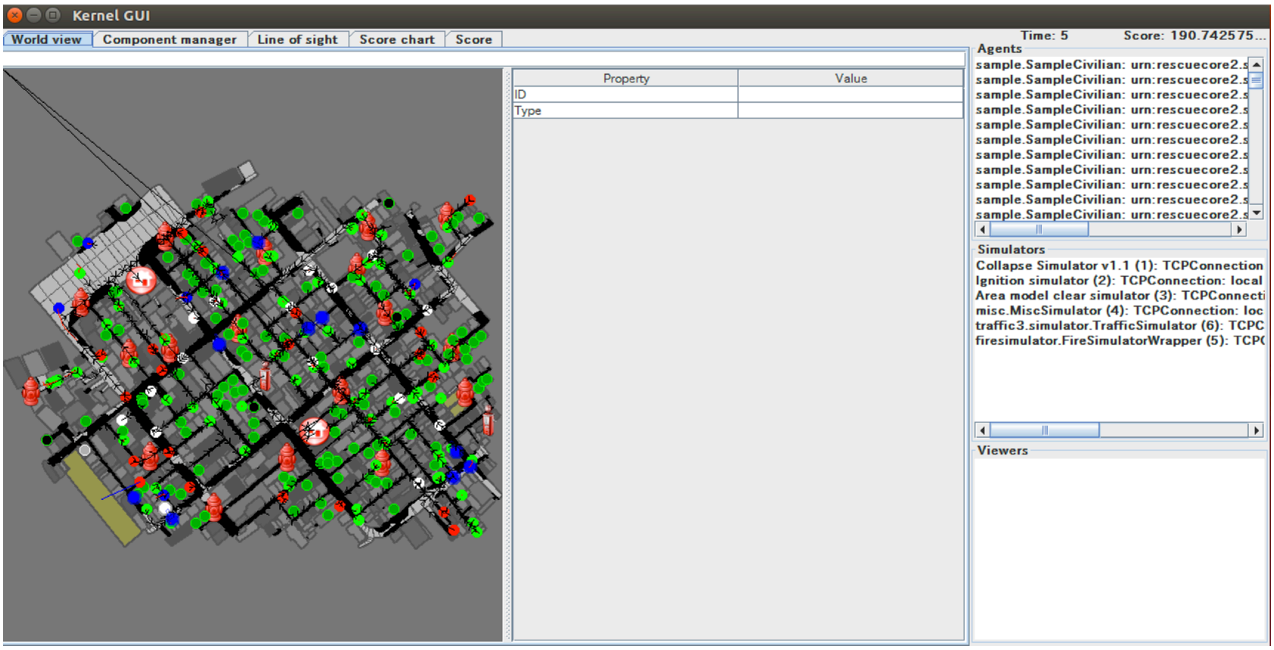

Figure 2: Starting the Simulation

Running Simulation Server

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

3.2.2 Running the ADF Sample Agents for Precomputation

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

Running Sample Agents

```
$ sh 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)
```

3.2.3 Running the Simulation

When the precomputation is completed, push `Control-C` and type `sh kill.sh` to stop the simulation server of running. Then, type `bash start-comprun.sh` to start the simulation server again to run the simulation scenario. The commands are:

Running Simulation Server

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

3.2.4 Running the ADF Sample Agents

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

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

4 Simulation Server Options

4.1 Directories

Important directories of the simulation server are:

- boot/: scripts to run the simulation server
  - boot/config/: configuration files of the simulation server.
  - boot/logs/: log files.
- build/: the simulation server's Java classes.
- jars/: the simulation server's JAR files.
- lib/: libraries used by the simulation server.
- maps/: maps that can be ran in the simulation server.
- modules/: the simulation server's source code.
- oldsims/: source code of some of the simulation server's older versions.

4.2 Parameters

The following parameters can be used to run the simulation server:

- -m MAPDIR or --map MAPDIR, where MAPDIR is the path to the directory containing the map you want to run (default is ../maps/gml/Kobe2013/map).
- -c CONFIGDIR or --config CONFIGDIR, where CONFIGDIR is the directory containing the configuration associated with a map (default is ./config).
- -l LOGDIR or --log LOGDIR, where LOGDIR is the directory where the log files will be stored (default is ./logs).

These parameters can be used at running a precomputaion and a simulation. You must use the same parameters regarding MAPDIR and CONFIGDIR to run a simulation server with or without precomputation. An example of how to run the simulation server using these parameters is:

Running Simulation Server with Options

```
$ bash start-precompute.sh -m ../maps/gml/berlin -l logs2
(After completing agents' precomputation)
Control-C
$ sh kill.sh
$ bash start-comprun.sh -m ../maps/gml/berlin -l logs2
```

5 How to Create Your Own Agents with ADF

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

5.1 Important Directories

Important directories of ADF (`rcrs-adf-sample`) 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.

5.2 Files to Create 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.

5.3 Work Flow of 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.

5.4 Modules Configuration File

The modules configuration file `src/adf/config/module.cfg` indicates which codes would be used as agents' module. Fig. 3 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 Fig. 3 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 5.5.3.

```
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
⋮
```

Figure 3: Modules Configuration File

5.5 Example of Implementing A* Algorithm for Path Planning Algorithm

5.5.1 Copy A Sample Code

First of all, 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
```

5.5.2 Editing the Sample Code

Listing 1 is the code of `SamplePathPlanning.java`, which has Dijkstra's algorithm. You should edit 1st line, 18th line and 27th line (these lines are indicated with red comments). You would implement your own code in the method `calc()`, and remove the method `isGoal()` that is only used by `calc()`. Listing 2 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 3 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 4. It must be put in the file `AStarPathPlanning.java`.

Listing 1: SamplePathPlanning.java

```
1  package adf.sample.module.algorithm; // Edit this line
2
3  import adf.agent.communication.MessageManager;
4  import adf.agent.develop.DevelopData;
5  import adf.agent.info.AgentInfo;
6  import adf.agent.info.ScenarioInfo;
7  import adf.agent.info.WorldInfo;
8  import adf.agent.module.ModuleManager;
9  import adf.agent.precompute.PrecomputeData;
10 import adf.component.module.algorithm.PathPlanning;
11 import rescuecore2.misc.collections.LazyMap;
12 import rescuecore2.standard.entities.Area;
13 import rescuecore2.worldmodel.Entity;
14 import rescuecore2.worldmodel.EntityID;
15
16 import java.util.*;
17
18 public class SamplePathPlanning extends PathPlanning { // Edit this line
19
20     private Map<EntityID, Set<EntityID>> graph;
```

```java

    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 algrithm 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;
```

```java
                }
                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 2: Part of AStarPlanning.java

```java
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 3: calc()

```java
    @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;
                }
```

```java
            }
            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 4: Node Class

```java
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;
    }
}
```

5.5.3 Editing the Module Configuration File

You must edit the module configuration file `src/adf/config/module.cfg` related to a path planning to use your code. The Figs. 4 and 5 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.

```
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
```

Figure 4: Default 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
```

Figure 5: Edited module.cfg