

DIT945/ TDA593 Model Driven Software Development, 7.5 higher education credits - Model Driven Software Development, 7,5 högskolepoäng

Project Description

The goal of this project is to model a robotic system with various UML-based modeling techniques, and implement a simple demonstrator of such system. The description below gives a general idea of the robotic system to model.

Robot rovers are special kind of robots that are specifically designed for autonomously moving within an environment, sensing information, and actuating accordingly.

The system that we want to model is a platform called ROVU that allows non-technical operators to monitor the status of the environment in which rovers perform their missions. In ROVU each rover has proximity sensors for sensing the presence of obstacles or other rovers, a GPS sensor for moving along geographical positions, a camera mounted on the upper side of the rover itself, wheels for moving, and a networking device for communicating with either other rovers or a central station.

The system to be modeled is made of robots that have movement capabilities. Robots should perform missions that may change over time. Missions concerns sequences of points the robot should reach. A mission is achieved when all the points are visited by the robot to which the mission is achieved. Missions can be achieved using different strategies that may change over time. Strategies refer to the order in which the points within a mission are visited. Robots can perform missions in various and different environments. Environments are closed spaces (limited by Boundaries). An environment is partitioned into areas, and areas can be nested one within another. Areas can be logical areas or physical areas; logical and physical areas may coincide or not. Two examples of environments in which the robots can be deployed are a Zoo and a University building.

- *A zoo is composed by wooden and iron cages, which can be nested one within another. For example, the Lion iron cage can be nested within a wooden cage containing dangerous animals. Logical areas*

represent for example WiFi zone that can be entered or left by the robots.

- *A University building is made by rooms nested one within the other. For example, the Office 007 can be nested the Department area 001. Logical areas represent for example (i) a WiFi zone that can be entered or left by the robots, or (ii) an eating area, which is not clearly delimited by physical boundaries.*

Non-technical operators should have an interface that shows where robots are located within the environment. Several different interfaces can be provided. For example, an interface can be a graphical interface that shows, for each logical area, the robots that are within the area. Another interface can be designed for blind people; this interface should read the name of the robots within the different areas. Additional interfaces may also be added (e.g., a command line interfaces). Note that these interfaces are passive, i.e. they just show the state of the environment. Interfaces should also show reward points.

Reward points are computed every 20 seconds as the robots explore the environment. The system uses one of two procedures (A or B) to compute reward points. Procedure A is based on physical areas, procedure B is based on logical areas. The procedure A based on physical areas gives one point for each robot contained into an office. Two points are given if the robot is in the teaching room. The procedure B gives one point for each robot in the Wifi zone and two points for each robot in the eating area. At the beginning the system uses procedure A. The procedure might change over time according to the following rules:

- *If the current procedure is procedure A and there is at least a robot inside a logical area, then the procedure should change to procedure B.*
- *If the current procedure is procedure B and there is at least a robot inside a physical area, then the procedure should change to procedure A.*

The ROVU platform is especially designed to maintain a desired degree of resilience of the system, i.e., to ensure that the system will persistently deliver its services in a trustworthy way even when facing changes and unforeseen situations. More specifically, the system can react to the

following two events:

- *Fault on a rover - in this case the rover should consider the type of fault (e.g., a fault in one of the wheels or a fault in the camera, etc.); in the worst case the rover should simply immediately stop and turn itself off.*
- *Mission change - the framework will support also changes on the mission made at run-time, for instance the operator may push a "Stop-everything" button for immediately stopping the whole mission due to some severe circumstances.*