UML for Embedded Systems

Exam FALL 2017
Rover Software

Ludovic Apvrille

ludovic.apvrille@telecom-paristech.fr

http://soc.eurecom.fr/UMLEmb/

During an exam, you are not supposed to talk with someone else, by any means (including mobile phones, chat, etc.). Access to Internet is restricted to the website of the UMLEmb course only. You may consult your own UML/SysML models made in the scope of the labs, but not other models. Electronic devices are not allowed at all, apart from your computer ;-).

A grade is provided for each question. 1 bonus point is awarded for writing quality.

1 Objective

Your objective is to model the **software of a Rover robot**.
You have exactly 3 hours to model this system, and answer various questions: the time is very short. This means that **you have to take modeling assumptions**. **Keep your diagrams simple and readable**, in particular the analysis diagrams.

Your grade takes into account your report and your models. At the end of the exam, **reports** (in pdf format) and **models** (in TTool format) **must be sent to me by email**. Also, **the report must be printed** and **given to Alexia Cepero right after the end of the exam session**. The report should contain explanations concerning your models, as well as relevant screen captures of models (e.g., interesting simulation traces, formal verification results).

2 System specification

2.1 Description

2.1.1 Overall description

Autonomous vehicles and robots are already used to assist rescuers during emergency situations. The robot — also called "Rover" — is a small autonomous vehicle able to search for buried victims in rubble. The Rover has 6 wheels, with one electric engine per wheel, and distance sensors — sampling frequency ranging from 1Hz to 100Hz – located at the front, the rear, the top and on each side. These sensors are useful to identify obstacles and to autonomously navigate. Once deployed in operation, the rover makes a cartography of the area by scanning a 30m radius circle around its starting location. This explored area is limited by its battery capacity that the software should monitor in order for the Rover to be able to come back to its departure location. The maximum speed of the Rover is 3km/h.

The Rover adapts its data acquisition to the current situation. Indeed, when no obstacles are detected around it, the Rover decreases the sampling frequency of distance sensors thus assuming that no obstacles can suddenly appear in its immediate surroundings. Whenever

an obstacle is detected in its “safety bubble” — its safety bubble is a circle of 1m radius around its center — then the Rover decreases its speed and increases its sampling frequency. Since the distance computation depends on temperature and humidity conditions, the Rover has one temperature and one humidity sensor. It is required that an obstacle in the safety bubble is detected in less than 1 second. Thus, the sampling rate is computed according to this requirement.

The Rover can be monitored and driven by rescue teams using a Wifi hotspot settled by the Rover. The software of the Rover also includes a website that allows users to monitor its speed, its battery, and the percentage of explored area — a map representing the explored area is also displayed in the website —. Anomalies are also listed in the website. Last but not least, "an emergency button makes it possible to request that the robot return to its original position.

3 Assignments

I. Assumptions

1. Your assumptions should be clear. Do list them in the report: that list might evolve according to the models you will make afterwards. Make a clear separation between environment and modeling assumptions. [2 points]

II. Requirements

1. Create a requirement diagram. [3 points]

III. Analysis

1. Make a use case diagram. [3 points]
2. Continue the analysis in the form you want: activity diagrams, nominal scenario, error scenarios, . . . : you are free to use the diagrams you want. Of course, the idea here is to show important points of the specification. [3 points]

IV. Design and validation

1. Make a block diagram. Put the emphasis on which blocks are used to model the system being designed, and which ones are used either to model the environment, or to prove properties (observers). [2 points]
2. Draw state machines, and provide a nominal simulation trace, as well as an error trace. [3 points]

3. Prove that whenever a search is performed, either a result is provided, or an error is returned to the user. Also, from requirements, pick up a property of your choice, and prove whether it is satisfied (or not!). And obviously, explain how you have modeled those properties [3 points]

Good luck, have fun!