Home Assignment – Critical Embedded Systems

SysML Modeling

You have to create a complete model for an automatic mower in Magic Draw. The assignment consists of **three phases** (requirement, structure and dynamic), each phase is related to a laboratory session that helps you to get used to certain diagram types. On each laboratory we will create a starting model from which everybody can start working on his/her solution.

Documents required: You have to create the **models in the Magic Draw framework**, and you also have to write a **summary** (in *pdf* format). The summary has to explain why you created your models as they are and what your **modeling decisions and assumptions** were. We are also evaluating the outline of the document, so please try to prepare a decent document!

Please check on the official home page if we have already received your choice of your pair. If not, please send the details to ahorvath@mit.bme.hu

The evaluation is the following. We can only accept solutions containing all three modeling phases and their proper documentation. In the final evaluation you have to be able to defend your models and decisions. In addition, you might be asked to modify your specification/model. Note that the defense will be organized separately for everybody and thus you will have to **defend** the complete model **on your own**.

Additional information

- It is strongly advised to **work on the assignment continuously** and start working on each phase right after its corresponding laboratory session is held.
- Final deadline is the 2nd of November 23:59 via upload to a ***designated website*** (will be announced later via news and also on the course's webpage). B**oth team members should send the solution on their own**! Compress your solution using zip.
- **Naming convention for the zip file to be sent**: *[Monogram]_[NeptunCode]*_CES2014.zip
  - *[Monogram]*: First characters of your name: e.g., Akos Horvath → AH
  - *[NeptunCode]*: Your neptun code
- Please submit the home work only once before the deadline! Double check that you have included all the required artifacts (complete model and its documentation).

1 Specification of the mower

Design an automatic mower, which can be used in the Cyber-physical agriculture system or simply operated by its owner.

The mower must be able to complete its mission based on pre-configured settings. The mission can be initiated by a start control message. Additionally, the mission can be aborted by a stop control message at any time.

The settings include the configuration of the work hours and the mission to be executed. The mission contains one or more mission phases. Every phase has a path which is a list of waypoints. The mower must follow these waypoints to complete the mission. Mowing can be enabled or disabled and mowing height can be set for each phase separately.

We must ensure the safe operation of the mower. It must avoid collisions and it must have overheat protection. To avoid collision the machine must stop when it senses an object in front of it. If the object moves out of the way or the system receives a start command, the execution of the mission has to continue. The overheat protection must interrupt the execution if the inner temperature rises above the predefined critical level (in our case 80 degrees Celsius) and resume the execution when it cools down below the critical level.

The system must notify the controller in case of events. These events consist of alerts when the mission is interrupted and a notification when the mission is completed or aborted.

Finally, it is important that the mower must be maintainable. The parts have to be replaceable and there must be an interface to maintain software, for example setting the clock.

Phase1: Requirement modeling and activity diagrams

Create the requirement models and activity diagrams for the robot mower.

1. Requirement model
    a. Create a new requirement diagram for functional requirements.
    b. Define the top level requirement, which represents the whole specification.
    c. Decompose the top level requirement to main functions.
    d. Refine the requirements based on the specification.
2. Use cases
    a. Define the use case diagrams to specify the various system interactions.
        i. Identify the actors.
        ii. Create the System boundary package.
        iii. Create system use cases and their relations to the system
        iv. Define information items (signals) to model information flow.
    b. Refine the defined use cases with the following elements:
        i. Use inclusion to include functionality of other use cases.
        ii. Use generalization.
        iii. Use extension to handle interruptions.

Design the Execute mission activity based on the requirements and the use cases.

1. Specify the normal execution. (without interruptions)
    a. Create the Execute mission activity and activity diagram.
    b. Add Start execution action. Create a new diagram that describes this action (receiving the start command, setting the first mission phase).
    c. Describe the process that satisfies the requirements. Control the engine and the blades according to the actual mission phase settings and move along the configured path (Move along waypoints action). Execute all the mission phases then stop the engine and send a mission completed message to the controller.
    d. Expand the process with the object flow.
2. Add the various interruptions and the handling operations to the diagram.
    a. Create an interruptible region around the execution.
    b. Add accept event actions inside the region which will interrupt the execution.
    c. Add an Interrupted execution action outside the region. Create a new diagram that describes this action. (e.g., turn off engines, send alert to controller, wait for events that resume execution) Don't forget the control/object flow: the execution must resume after the interrupted state.
    d. Define the stop command that aborts both the execution and the interrupted state.

3. Extra (for extra points) : design the following activities:
    a. Sensing environmental effects and sending messages to the execution activity.
    b. Checking work time intervals.
    c. Elaborate Move along path action.
        i. Calculate direction and distance from current location and next waypoint.
        ii. Control the engine according to calculated values.

Phase2: Structure modeling

1. Specify the system context.
    a. Create a "System context" block diagram.
    b. Identify system actors and associate them with the system.
    c. Add signals and add informational items to the associations that the system exchanges with the actors.
    d. Create a new block diagram: System context with ports. Identify system interaction points and add them to the system as ports. Associate the actors to the ports. Remember that more than one actor can use the same port!
2. Model system interfaces based on the system context.
    a. Create a "System interfaces" block diagram.
    b. Create interfaces that are required to sense the environment or other functions.
    c. Create interfaces that provide the system functions to actors.
    d. Create a "System ports" block diagram. Add the Mower block and make the ports visible. Assign the interfaces to the ports.
3. Model system structure.
    a. Create a "System structure" block diagram with the Mower block diagram.
    b. Use composition to add subsystems to the Mower and specify the system interfaces and requirements.
    c. Sensors are sending measured values (e.g. temperature data) at periodical intervals. Assume there is a Sensor adapter that converts it into logical data (e.g. overheated signal).
    d. The Direction and Location information can be sensed by a positioning system.
4. Model the subsystems. In the previous step we defined the system components. Now, it's time to model their connection.
    a. Create a "Subsystems" internal block diagram. The diagram context is the Mower block. Make the ports visible on the diagram frame.
    b. Include the subsystems to the diagram.
    c. Define ports to the subsystems to connect them.
    d. Define signals and create interfaces for the inter-subsystem communication.

Phase3: Behavior modeling

1. Model the states of the Temperature sensor. When the sensor is on it sends temperature data once per minute.
2. Model the states of the Sensor adapter. The adapter converts the sensor data into signals, which are received by the Mower controller. An overheated signal is sent if the temperature rises above 80 degrees and the temperature *OK* signal is sent when it falls below 70. A *proximity alert* signal is sent if an object gets too close to the mower (less than 1 meter).
3. Design the state based model of the Mower controller behavior.
    a. Create a new state machine diagram.

b. Add a ready state in which the system is by default.
c. Refine the state machine based on the commands that the system can receive. Use signals and the operations from the interfaces as triggers.
d. Elaborate the executing mission state.
    i. Use the Execute mission activity to design the states and transitions.
    ii. During execution the system also checks periodically the work time intervals at pre-defined checking intervals. This process creates signals that can affect the execution.
    iii. In case the execution is halted and later resumed, continue the execution at the state where it got interrupted.