TI-AREM Exercise 4. – Dynamic change of Algorithms for the RealTime loop using the Strategy Pattern and using a Real Time abstraction

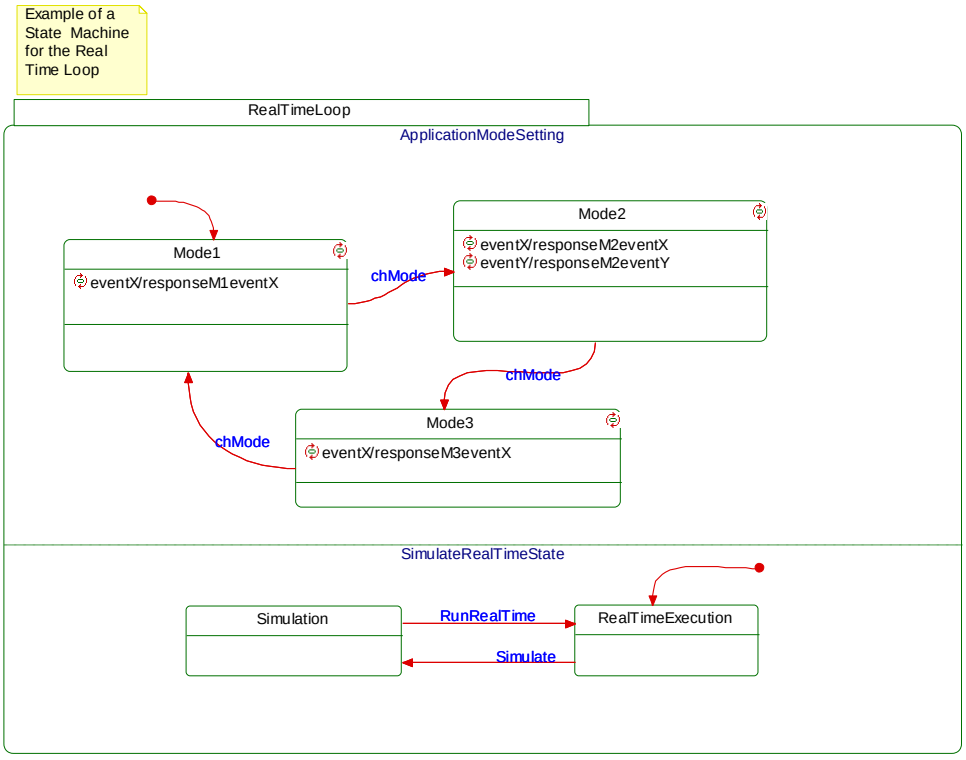
Description:
The state machine from exercise 1. has a RealTimeLoop state with 3 mode states – each of these modes corresponds to executing of a specific algorithm for the real time computation.
In exercise 3. the state machine was enhanced with and AND-state describing a simulation mode with simulated input and output and a RealTimeExecution state where both input and output is with the real HW-drivers. Introduce first concurrency with two threads and after this introduce the GoF Strategy pattern in connection with the real time thread.

Goals:
When you have completed this exercise, you will
- have experience using an RTOS abstraction
- have experience with implementing the GoF Strategy Pattern and dynamic configuration of this based on a state transition in the state machine

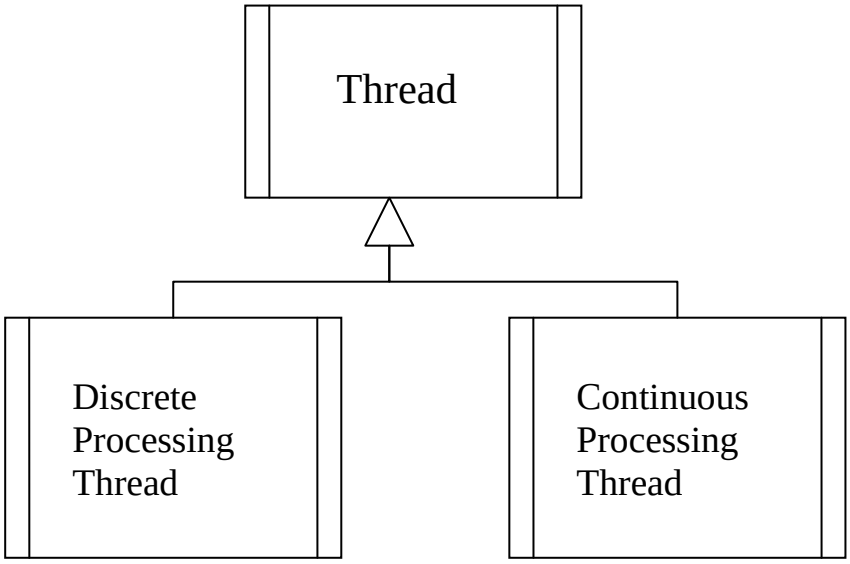
Estimated Time: 2 weeks

Exercise 4:

State machine from Exercise 3:



The real time processing shall be performed in its own thread and the discrete part (the state machine) shall be performed in another thread.



The Thread class is an OS-abstraction to be placed in the Abstract OS package introduced in exercise 3.

4.1. Implement the OS-abstraction thread class on a windows PC. Se the description of the OS-Abstraction class for windows.

4.2 Design and implement the DiscreteProcessing Thread encapsulating the State Machine from exercise 1-3.

4.3. Design and implement the ContinuousProcessing Thread implementing the functionality to be executed in the RealTime Loop state of the state machine. This thread should have the following functionality (specified in pseudo code). The solution shall use the GoFStrategyPattern.

```
run()      // run method in Continuous Processing Thread
{
    while (the state machine is in the RealTimeLoop)
    { // step 1-3 should all be performed with the same set of
      // strategies (either simulated or real)
      1. Read input from an input driver via an input strategy with two
         possibilities either from a simulated input-driver (depending of the
         SimulateRealTime State) or the real hw-input driver.
      2. Execute algorithm via an algorithm strategy (with three strategies one
         for each mode).
      3. Output result of the algorithm via an output strategy with two
         possibilities either from a simulated output-driver (depending of the
         SimulateRealTime State) or the real hw-output driver.
     }
}
```

Skeleton code for the input and output drivers can be placed in the Abstract HW package.

4.4.1. Design the communication between the two threads using either a POSA2 Monitor class or the BPD Guarded Gall Pattern. Both solutions shall use a Mutex-class in connection with a Guard class form the Scoped Locking Idiom.

4.4.2. Design and implement the Mutex and Guard classes and place them in the Abstract OS layer. NB! Implementation of the Mutex-class should take care of the priority inversion problem by using the right synchronization mechanism in the actual operating system API.

4.4.3 Add the necessary actions (methods) to the state chart and implement the necessary classes and operations in the design from step 4.4.1

4.5 Test your solution with the state machine from exercise 1, the command pattern from exercise 2 and the and-state from exercise 3.

For this purpose a simple UI-thread with a user command interface can be used, where each command triggers one of the events defined in the command-pattern from exercise 2.

4.6. Finalizing exercises 3+4 and test your current solution on the PC.

Hand-in no. 2:
The second hand-in includes your solution to exercise 3+4.

The hand-in should consist of a short journal report (see template), which includes your design of the solution (class diagrams, state diagrams and sequence diagrams) with explanation of the design. The journal shall be handed in as a separate pdf file via Blackboard.

In addition to this the hand-in includes a Visual Studio project with your code, packed as a zip file and uploaded via Blackboard.

Deadline: See the Course Schedule.