Programming and Modelling
Robotic Arm Project Assignment Sheet

Weeks 5-6: Separating Controller from Environment

Submit your solutions via the assignment on Brightspace, Week 5-6. Please make sure to submit your solutions **by 16 March**, 23:59. In your submission, please include your Overture project of your robotic arm model, and a text file or PDF file called "ANSWERS" with your answers to each question and subquestion.

Exercises

(1) Following the *Sequential Design Model* stage that we discussed in lectures, make decisions about how aspects of the functionality in your environment class from your previous model should be moved into a new Controller class.

  (a) Based on these decisions, create UML class diagrams of this next version of your robotic arm system, following the approach we used in the lecture slides.
  Hint: at this modelling stage (Sequential Design Model) you should only have two classes, like we have done in the lectures. Furthermore, consider the topic of *different perspectives* carefully, and reflect these aspects in your next model design and UML class diagrams. Do not introduce too much complexity into your model in one step, keep the step in your model development clear, simple and easy to follow.

  (b) Briefly describe your new model, and your approach for separating functionality between environment and controller (3-4 sentences). Comment on whether the functionality has changed between your *System Boundary Definition* model and your *Sequential Design* model.

(2) Based on your new UML class diagrams in exercise (1) above, create a new robotic arm formal model project in Overture. In your World class, create an "echoState" operation that prints the relevant parts of the state to the console, in order to automatically generate scenario tables (as presented in the lecture slides).

(3) This exercise will focus on scenario simulations for your new model.

  (a) Adapt your four scenario simulations from the previous model (i.e. exercise (2) in your last robotic arm assignment) to this new model.

  (b) Create at least one new scenario simulation that demonstrates, and really highlights, the Controller's perspective of reality being "out of date".

  (c) Run these new scenario operations to create at least five simulation traces. For each scenario, generate the scenario table using World's "echoState" operation showing the value of state variables and time, for each scenario instruction.

(4) This exercise will focus on carefully introducing some Controller logic into your model. In the steam boiler case, basic controller logic was illustrated in a diagram that partitioned water quantity (q_e) into intervals of distinct controller behaviour (e.g. Figure 2 of the steam boiler case description).

  (a) Represent your intended controller behaviour as intervals that partition values for a given *system* variable. Sketch *at least two diagrams* where the horizontal axes in each sketch refer to different system variables.

Important: Do not include different controller modes - that is far too complicated for this modelling stage. Just focus on the same behaviours that you were considering in your previous *System Boundary Definition* model and *Sequential Design* model. Consider the steam boiler case discussed in the lecture, and notice that we did not introduce controller modes when we took these first steps for introducing basic controller logic.

(b) Give a brief explanation (2-4 sentences) of each sketch that you have made, including the intended controller behaviour, and why you decided on this behaviour (i.e. your rationale - explain how you arrived at the intervals and behaviour presented in your sketches).

Hints: Firstly, as a guide, consider the requirements that the system (with controller) must satisfy, just like we did with the steam boiler case. Make a list of variables that are relevant in terms of controller behaviour. For a given scenario, consider what variables you assume remain unchanged over the course of the simulation, and what variables change at each time step. Also consider constant values, as they may guide you towards some important behaviour cases. The variable on the horizontal axis does not necessarily need to be a variable in either the environment or controller class, and instead may be derived. Finally, consider landmark values (boundary cases) that are useful for defining intervals of numerical variables, e.g. less than -1, -1, 0, 1, greater than 1, etc.

Remark: This exercise involves creativity on your part in both (i) reflecting on what intended behaviour means in this case, and (ii) in how you describe and communicate this behaviour visually as a set of diagrams. Remember that the primary purpose of this exercise is for you (as software engineers) to deepen your understanding of the domain and system, and to then present a clear picture to readers on how the controller is intended to behave. There is no single, simple answer to these questions.

(5) In this exercise you will create the next model that includes controller logic.
  (a) Update your UML diagrams to include this new controller logic functionality.
  (b) Create a **new** Overture project for this next robotic arm model (just like we do with the steam boiler whenever we develop a new model).
  (c) Create at least two new scenario simulations that demonstrate different aspects of controller behaviour that you illustrated in exercise (4) above. Generate the scenario table. For each scenario, write a few sentences (1-3) that highlight the main point of the scenario, i.e. so that readers can clearly see what is being demonstrated in the rows of your scenario table.

(6) Your set of models is now growing. Create a "Robot Arm Model Versions" table, following the "Steam Boiler Model Versions" table approach presented in the lectures. Your table should have two columns: the model version and a brief summary description of the main point of each model. Fill in this table with your models so far; your table should now have three rows.