CARLETON UNIVERSITY
Department of Systems and Computer Engineering

SYSC 5104. METHODOLOGIES FOR DISCRETE EVENT MODELLING AND SIMULATION

Assignment 1

The goal of this assignment is to show your understanding on modeling discrete-event models using the DEVS formalism, and to use the CD++ and Cadmium simulation tools to execute discrete-event simulations. We will define a real system that can be represented using DEVS, build a model of the system of choice, and run simulations in order to analyze different conditions on the system. The system of your choice can be any natural or artificial system. There are two options for this assignment: the students can adapt an existing CD++ model to Cadmium, or write a complete model of their choice from scratch (20% bonus marks for those students writing their own).

Important dates:
Feb 10 – Assignment 1 model choice (or conceptual model document), midnight. The chosen model (or conceptual model document) must be delivered as an attachment via email.

Feb 24 – **Assignment 1 Report delivery, midnight.** The document must be delivered as an attachment via email, and the model (as well as the document) should be made available on a GitHub repository.

Part I:

The first stage is to select a model of choice from the list of models in the course, or to identify a real system of choice that can be described as a Discrete-Event Dynamic System model to developed from scratch.

For students working individually, the model of choice should have 2 to 4 levels (two levels means: the top model has a number of submodels, and at least one of them is a coupled model. A third level means that this model should also be decomposed into submodels. The model should have at least **three** atomic submodels (and at least 2 of them must be different). The remaining submodels can be atomic or coupled. Students working in teams (**at most, three per team; see conditions in the Appendix**) must define more complex models. Teams should be identified at this stage.

If the students decide to build their own model from scratch, they must provide a one-page report including a conceptual description. This document should include:
- a description of the problem to be solved,
- a brief sketch of the model structure, and,
- for each component, a brief description of the behavior of the component.

Part II

The original model in CD++ should be adapted, tested and executed using Cadmium. If you are writing a new model, the conceptual model defined in Section 1 will be used as a requirement document for the final work. You should:

- Organize your models as atomic/coupled, defining the structure and coupling scheme discussed in Part I. This could result in a redefinition of the structure proposed for the conceptual model document of Part I (which will be included in the final report).
- Write a DEVS formal specification for each of the coupled models.
- Define each of the atomic models using the DEVS Graph Modeler (https://devssim.carleton.ca/DEVS-Graph/
- Based on the DEVS Graph, write a DEVS formal specification for each of the atomic models, including a description for the transition functions (using pseudocode, state-based representations, DEVS Graphs or others).
- Propose an experimentation strategy for each one of the models. Document the experiments to be executed (if you reuse the CD++ models, you should add new experiments to the ones proposed in the original document).

You need to conduct "model experimentation" in which you show how your proposed test is combined with the formal model, and the results obtained (example: if you have an atomic model "Queue" as the one we discussed in class, you should consider two different tests, show how the formal model reacts to those tests, and explain the results; all in paper: no programming yet). For students adapting CD++ models: if the model found in the original CD++ document is wrong, fix it and discuss the modifications. At least two of the model experiments should be presented (in paper) and the combined results of the execution with your formal model discussed. The experimentation strategies should consider incremental expansion of the tests, starting at the level of an atomic model, and hierarchically expanding the test base in the model hierarchy. This will be the experimental framework for each of the models involved.

Part III

Build each of the Atomic Models in Part II using Cadmium. Build individual experiments for each of them following the test cases proposed in Part II, trying to ensure correctness of the atomic models. Once the atomic models have been tested individually, build coupled models, repeating the testing strategies proposed in the specification document. Build the top model of your application, and conduct integration testing.

Run different simulation examples, changing the original experimental framework, and showing the reaction of the model to different inputs than those defined in the specifications. Analyze the input/output trajectories of the model. Write a report showing these execution results and analyzing the behavior of the model according to the specifications. Document any changes done to the original specifications derived from model execution analysis. Include printouts of the atomic model execution in your report (and analyze the results observed).

Marking Scheme

Part I: The goal of this activity is to evaluate the capacity in identifying real systems that can be specified as DEVS models. The students writing their own models from scratch will receive an extra 20% as a bonus (meaning you can obtain 120% for Assignment 1). The deliverable is the one-page report associated to this stage of the assignment.

Part II: This specification document is worth 35% of the final mark. It will include the model specification used in the third stage.

Part III: This part of the assignment is worth 65% of the final mark.

Deliverables (mandatory)

I) DEVSmodelsForm

You should fill out this form, and upload it to your GitHub repository. It includes a short word document. The idea is to provide meaningful explanation about the model/ports.

II) Final Report

Your should prepare only one final report, which will be organized as a mix of the documents you created in Parts I-III, all combined in a single document. This document will include:

- Part I
- Part II
- Part III, which should explain the execution results of your model, any variations you have done to the original specification, and the new results obtained by varying the original experimental framework.

Remember that he documents for the 3 parts should be combined in a single document to be delivered, included in the same repository with the simulation software and documentation.

III) Simulation software

Besides the report, you should include the software developed. You should include:

- Source code for each of the atomic and coupled models
- **Scripts** to execute each of the models using different experimental frameworks
- Any '**read.me**' files explaining the detailed behavior for each one of the scripts provided.
- Your **makefiles**
- **Test** files containing input values, if needed
- Log files showing the execution results
- Your final **report** document (in .doc or .rtf format)
- The **DEVSmodelsForm** describing the model and giving author information

All of this information should be included in a GitHub repository. Your repository will be cloned; include instructions to clone and run the model if needed.

Use meaningful and self-descriptive names for every file included. Consider using names that represent the models you built (DO NOT use "Assign1.doc", "assign1.hpp"; "MyNameAssign1.cpp", etc… Use a name that represents the content of the model).

Failure in following these instructions will result in returning and resubmission of the assignment. Check the "alternatebitprotocol" model in Cadmium's GitHub repo to see what is expected. Use exactly the same structure you found there when creating your own repository.

The files submitted should run without any problems using the simulation tool. The repo will be cloned and your assignment will be analyzed as follows (recommendation: **do the same yourselves**):

. The repo will be cloned in an empty project folder
. It will be recompiled. The files should compile cleanly, generating the proper simulator in the end.
. The examples will be run using the scripts you provided.
. The test cases provided will be checked.
. After these basic steps are carried out, different changes will be included, in order to find possible errors. Therefore, be thorough when preparing test cases.

In order to ensure proper execution, try to execute these same steps by yourselves, before delivering the software. If any of the steps cannot be carried out, but can be fixed by the instructor, you will lose marks.

APPENDIX

Students working in teams (**at most, three per team**) must define more complex models using the following constraints:

- Two-member groups: the model should have three or four levels, and at least four atomic models (at least three of them different).

- Three-member groups: the model should have three to five levels, and at least five atomic models (at least three of them different).

CHECKLIST

Read carefully the requirements specified before, and to everything there explained; **if something is missing, the assignment will be returned unread**.

Tick each one of these items, and verify you comply with them.

___ You have combined the 3 parts of the report in ONE report. Each section has a meaningful name (not "Part I,II,III", but a descriptive title explaining the content of the section). The report has a meaningful name.

___ You have included the report document in your repository

___ You have included the necessary files:
- ____ Source code for each of the atomic and coupled models
- ____ Scripts to execute each of the models using different experimental frameworks
- ____ 'read.me' files explaining the detailed behavior for each one of the scripts provided.
- ____ makefiles
- ____ any files containing input values, if needed
- ____ logs showing the execution results
- ____ Your final report document (.doc/.rtf/.xml files)

___ You have put meaningful and self-descriptive names to each of the files

___ You have created a new project, and tried to recompile and run each of the tests, and you were successful in each of them.

___ You have checked the "alternatebitprotocol" example on our Github to see an example of what is required.