Modélisation des objets en UML - Examen

You are requested to design the software of a manager that handles the FPGAs in a cloud data center, in UML. FPGAs are devices which are composed of set of hardware resources (e.g., logic elements, memory, etc.). An FPGA can execute a set of applications, in parallel, and then it can be reconfigured to execute another set of applications. Users interface the manager through an external GUI, which give the user access to a web site that is part of the system. Users want to execute their applications onto an FPGA, so they interacts with the GUI. Each user application is labelled with two parameters: (i) execution time, in units of time (from 1 to 300 ms), and (ii) the percentage of reconfigurable logic elements of the FPGA (minimum 10%, maximum 90%). When a user want to execute an application onto a cloud FPGA, it asks the FPGA manager through the GUI. The system is able to assign a portion of the FPGA to the application.

There are three phases:

1) *Collection round:* the FPGA manager collects the requests of the users that want to execute an application onto the FPGA.
2) *FPGA configuration*: this is the setup of the FPGA before the execution of the applications. It lasts exactly 50 units of time.
3) *Execution round:* the FPGA manager waits that the FPGA terminates to execute the assigned set of applications, then another *collection round* occurs. This phase lasts as the longest application of the set.

The *collection round* lasts for exactly 100 units of time. In this period, users can request the FPGA to execute their applications. The FPGA manager assigns the application to the FPGA in the order of arrival until the end of the 100 units of time. If the requests of the applications that require the FPGA area exceed the limits of the FPGA, the applications that exceeds the area are not selected to be executed, but they are placed in a buffer instead. Such buffer is a component that is external to the software you have to design and it can host the information of two applications maximum. The system interacts with the buffer through APIs. Such APIs can be used to put an application onto the buffer, or to retrieve the oldest application in the buffer or to any other operations that allow the system to see this external buffer as a *black box*.

At the beginning of the collection phase, priority will be given to the applications that are in the buffer, if any. All the other requests are rejected and the user is informed through the GUI of the web site. Any requests that arrive during the *execution round* or during the *FPGA configuration* is rejected as well and the web site is updated.

Output of applications are then shown in the website as well as when an application is placed onto the buffer.

Example of the collection - During the *collection round "k"*, three applications ask to be scheduled, in this order:

- Application X, 37 ms, 40% of FPGA
- Application Y, 140 ms, 70% of FPGA
- Application Z, 11 ms, 40% of FPGA
- Application W, 200 ms, 35% of FPGA
- Application P, 12 ms, 15% of FPGA
- Application N, 141 ms, 50% of FPGA

Then:

- Application X is accepted for the *execution round "k"*.
- Application Y is placed in the buffer (FPGA has not the 70% of resources after selecting Application X)
- Application Z is accepted for the *execution round "k"*.
- Application W is placed in the buffer
- Application P is accepted for the *execution round "k"*.
- Application N is refused (no FPGA spaces, buffer full)

In the collection round *"k+1"*, Application Y is selected to be executed in *execution round "k+1"* and it has priority over all the applications that arrive during the *collection round "k+1"*. Application W cannot be scheduled with Application Y (FPGA is not big enough to host both of them), so it will be part of the round *"k+2"* and it will have priority over everything else (i.e., a possible application sent in the buffer during the *collection round "k+1"* and all the applications that asks for scheduling in the *collection round "k+2"*).

Provide the following diagrams, with brief comments where necessary:

Assumptions: (3pt) (they can be provided throughout the report or in an ad-hoc section, as you prefer). Specify nature of assumptions (i.e., system or environment).

Requirements: (3pt)

Analysis: (6 pt)

- Choose one (or more, if you prefer, and we will consider the best among them) between:

  Use case diagram
  or
  Activity Diagram
  or
  2 SDs: one error scenario + one nominal trace.

Design:

- Block diagram: (3pt). In it, you can avoid to write name of signals and attributes. We will understand it from state machines. The name of the block is enough, if it is meaningful.
- The state machine of the system block(s) only (thus, no actors). (5pt).

Advices

1. Take meaningful **assumptions** to **simplify** the input problem. *Modelling* means abstraction and you must be able to abstract what it is essential for a first design (like you are delivering a prototype of the system).
2. Be aware about what is **inside** and what is **outside** your system, especially in use case diagram and in block diagram.
3. The quality is evaluated as well: avoid unclear work, too many typos, etc.

You can provide your diagram in English or French, but avoid mixing the languages and try to present a clean work.