Norges Teknisk-Naturvitenskapelige Universitet

TPK4186 - Advanced Tools for Performance Engineering Spring 2020

Assignment 2: A New Control System

Prepared by Antoine Rauzy

Contents

1 Introduction **1**
1.1 Presentation of the Problem 1
1.2 Requirements 3

2 Tasks **3**
2.1 Data Structure for Projects 4
2.2 XML Interface 4
2.3 Checker 4
2.4 Monte-Carlo Simulation 5
2.5 Classification 5
2.6 Regression 6

1 Introduction

1.1 Presentation of the Problem

You are working of an oil and gas company and in charge of evaluating the risk in a project consisting in installing an innovative new control system for a subsea plant. The main risk is that the project takes too long to be completed, including because the new system has to be certified by the regulation authority. Consequently, you are in charge of assessing this risk. To do so, you shall perform Monte-Carlo simulations. Moreover, as your management is willing to go for artificial intelligence, machine learning and all these fancy things. You are thus in charge of assessing the possibilities opened by the use of machine learning techniques to raise alerts when a project is seriously drifting from the planned schedule.

The project consists of a number of tasks with some precedence constraints, as shown Figure 1. A XML description of the model is given in the joined file "`ControlSystemProject.xml`".

The model described on the figure extends classical PERT diagrams and is remotely inspired from Business Process Modeling and Notation (BPMN) [1]. It is made of the following elements:

- Gates "Start", "MidProject" and "End", represented by circles;
- Tasks like "SpecifySystem", represented by plain rectangles;
- Precedence constraints, represented by arrows;
- Lanes like "MethodologyGroup", represented by dashed rectangle;

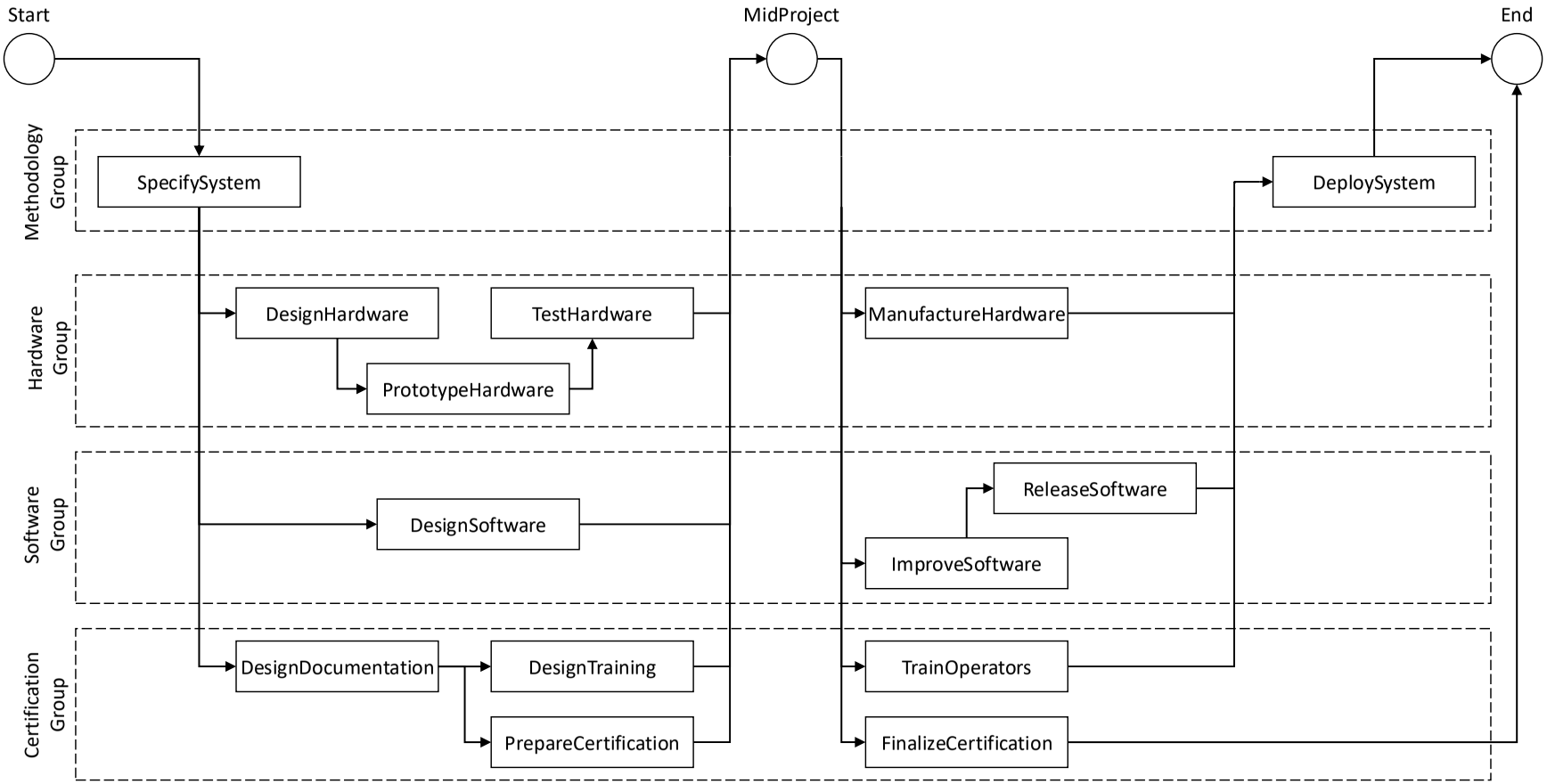

Figure 1: The diagram representing the "Control System" project

– Finally the project itself.

All elements but precedence constraints have a name.

Gates are used to described important steps of the project. In our case study, there are three important steps, the beginning and the end of the project as well as a mid-term review. Machine learning techniques will be applied using this intermediate gate.

Lanes tells which services of your company are in charge of what.

The durations of tasks are assumed to obey triangular distributions. Each task is thus given with a minimum and a maximum duration. Its expected duration, the mode of the distribution, is calculated as follows.

$$\texttt{expected-duration} = \texttt{minimum-duration} + (\texttt{maximum-duration} - \texttt{minimum-duration}) \times \texttt{workload} \quad (1)$$

where `workload` is a factor common to all tasks of a lane. This factor, which is one of the source of uncertainty, is assumed to uniformly distributed in $[0, 1]$.

1.2 Requirements

The objective of this assignment is not only, nor even primarily, to assess the risk in control system project, but rather to show your Pythonic skills.

Here follows a number of requirements.

1. You must provide your program together with a small document explaining how it is organized, what it is doing (which functionalities are implemented) and reporting experiments you have performed with it. You may also provide additional files such as those you use to test your program and to perform experiments.

   The report must be written as a HTML page.

2. Assuming your name is Jack Sparrow, all of the above files must be included in a zip archive named:

   `TPK4186 - 2020 - Assignment2 - Sparrow Jack.zip`

   The deliverable of the assignment is this zip archive.
3. The assignment must be made invidually.
4. You must program in an object-oriented style, preferably splitting your program into modules.
5. Recall that the quality of a program can be judge along three criteria: its completness, its correctness and its maintainability:
   – A program is complete if it provides all functionalities demanded by the client. Some functionalities are however more important than other. You must first concentrate on the main functionalities, then develop the "nice-to-have" ones.
   – A program is correct if it is bug free. To ensure that your program is correct, you must test it extensively. Design tesst before writing the first line of code. There is no such a thing than a program or a functionality that works "most of the time". Either it works, or not. If you are not able to make a functionality work, do not deliver it.
   – A program is maintainable if it is well presented, if the identifiers are significant, and so on. But before all, a program is maintainable if it well organized and as modular as possible. Separate the concerns.

2 Tasks

The assignment consists in the following tasks.

1. Design a data structure, i.e. a set of classes, to encode models as the one represented in Figure 1.
2. Design a XML interface for the models, i.e. a parser and a printer able to read and write models from and in XML files.
3. Design a small tool to verify that a model read in a file is well designed.
4. Design a stochastic simulator for the model so to study the duration of a project.
5. Apply classification techniques to try to predict whether the duration of a project will exceed a given limit, given the dates of completion of all task preceeding the mid-term gate.
6. Apply regression techniques to try to predict the duration of a project, given the dates of completion of all task preceeding the mid-term gate.

The remaining of this section gives you hints to perform these tasks.

2.1 Data Structure for Projects

The classes used to store projects should be gathered in a separated module.

There are five types of model elements:

1. gates,
2. tasks,
3. precedence constraints,
4. lanes, and
5. project themselves.

A good idea is therefore to design five different classes, one per type of elements, and to make these classes derive from a common base class.

One can note however that gates and tasks share a number of attributes:

- A list of predecessors;
- A list of successors;
- A start date;
- A completion date.

Consequently, it is a good idea to make the corresponding classes derive from a common class representing "nodes" (itself derived from the base class representing project elements).

In the same way, lanes and projects are both containers for project elements. It is thus a good idea to make them derive from a common class representing "containers". The creation of elements in containers must obey the "factory" pattern.

When a precedence constraint is created, it is a good idea to update immediately the list of successors of its source node and the list of predecessors of its target node.

2.2 XML Interface

The XML interface will be very similar to those we have implemented so far. Note that it may be easier to generalize the parser and the printer so to make it possible for lanes to contain lanes.

The XML interface must be implemented in a separated module.

2.3 Checker

A model is well designed if the following conditions hold.

- There is one and only one node with no predecessor, and this node is a gate.
- There is one and only one node with no successor, and this node is a gate.
- These is no loop in precedence constraints.

Verify the third condition requires computing strongly connected components. We shall assume that models verify it. It is therefore not necessary to check it.

The design of the checker requires to be able to collect all nodes (gates and tasks) of a project. You can implement a recursive function to do so. This function takes typically a list of nodes as parameter.

The checker should be implemented in a separated module, possibly with the simulator described hereafter.

2.4 Monte-Carlo Simulation

The simulator should implement to base functions:

- A function to draw at pseudo-random the durations of tasks. This function is typically recursive as it requires to draw at pseudo-random the workload factor of lanes.
- A function to calculate the start and completion dates of each node of the project.

Regarding the second function, it is possible to make it of linear complexity in the size of the project. However, you do not need to go that deep. The simplest algorithm to calculate start and completion dates can be stated as follows.

Step 1 Initialize start and completion dates of all nodes at -1 (to represent the fact that they have not been calculated yet).

Step 2 While it remains a node whose the start and completion dates have not been calculated, pick a up a node such that the start and completion dates of all of its predecessors have been calculated. Then,

- Set the start date of the node as the maximum of the completion dates of its predecessors (and to 0 if it has no predecessor).
- Set the completion date of the node as the sum of its start date and its duration.
- Iterate step 2.

It is then easy to implement a function that performs a Monte-Carlo simulation.

You have to make the following statistics on the outcome of your simulation (a numpy array of project durations).

- Mean duration;
- Standard deviation of the duration;
- Minimum and maximum durations;
- Quantiles 0.5 (median) and 0.9;
- Histogram (using `matplotlib`).

2.5 Classification

The idea is to use completion dates of all nodes preceeding a given gate as attributes for the classification and the duration of the project to calculate the label.

This requires implementing a function that extracts the list all nodes preceeding a given node. This function can be implemented recursively as follows.

- If the current node is already in the list, then there is nothing to do.
- Otherwise, we add the node to the list and call the function recursively on all its predecessors.

This requires also implementing a function that modifies the Monte-Carlo simulation in order to generate a sample. This function should take three parameters: the project, the name of a gate and a maximum duration. All trials of the simulation whose duration are less than the maximum duration receive the label 0, the other receive the label 1.

It is strongly demanded:

1. To use different classification algorithms (at least 3).
2. To study different maximum dates (at least 3).

Assess the performance of the diffeent algorithms.

2.6 Regression

Same requirements as for classification: use at least three different regression algorithms.
Assess the performance of the diffeent algorithms.

References

[1] Stephen White and Derek Miers. *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies Inc., Lighthouse Point, FL, USA, 2008.