**Free University of Bozen-Bolzano – Faculty of Computer Science**
**Data and Process Modelling – A.Y. 2015/2016**
**Exam – 01/07/2016** ***Solutions***

This is a closed book exam: the only resources allowed are blank paper, pens, and your head. Explain your reasoning. Write clearly, in the sense of logic, language and legibility. The clarity of your explanations affects your grade. Write your name and ID on every solution sheet. Good luck!

# 1 Data Modelling

## 1.1 Design

L. B. "Jeff" Jefferies (from the *Rear Window* movie by Alfred Hitchcock) is in a wheelchair due to a racetrack accident. He observes carefully what happens in his neighbors through the open windows of his apartment. To keep track of all his observations, he decides to create a dedicated information system.

To decide how to represent facts and observations, he starts from some sample notes he took on paper:

1. "Lars Thorwald lives in building nr.3 on the third floor in the second apartment."
2. "From 30/06/2016 at 8:00 AM to 30/06/2016 at 9:00 AM, an animal was digging in the garden of building nr.3."
3. "From 29/06/2016 at 11:00 PM to 30/06/2016 at 1:00 AM, Tom Doyle was visiting Jeff in his own apartment, the first one at the 4th floor of building nr.1."
4. "On 28/06/2016 at 11:30 PM, a scream occurred".

He decides to organize such notes using four key concepts: living beings, locations, actions, and events.

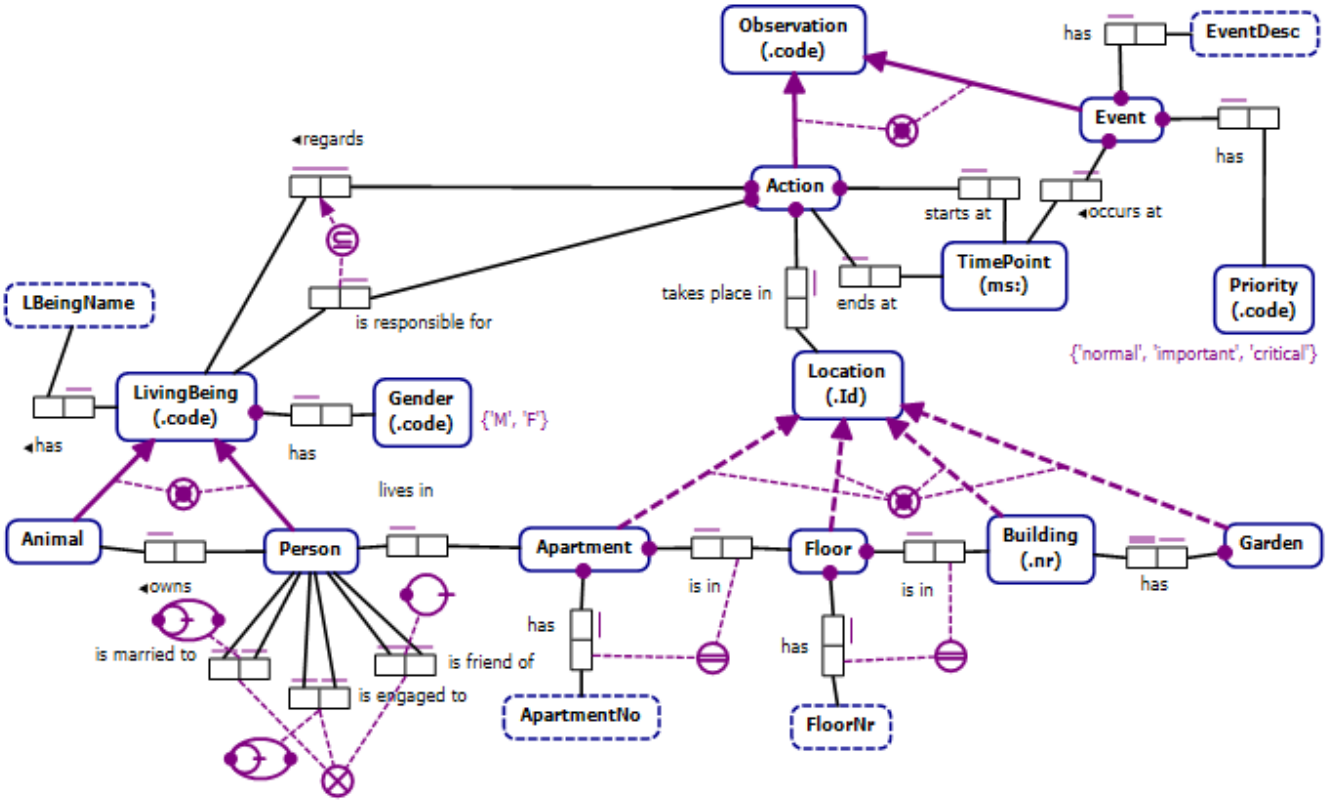
A living being is identified using a code, classified as a male or female, and optionally described with a name. Among living beings, Jeff decides to distinguish animals from persons. Each animal may be owned by a person, and each person may be married to or engaged to another person, and may be friend of (possibly many) persons. Such relations are mutually exclusive.

Locations identify spatial places where living beings do something. Jeff is interested in mapping buildings, floors, apartments, and gardens. Each building is uniquely identified with a number, and consists of several floors. Each floor may contain multiple apartments, each identified using a number that is unique within that floor. Every apartment may host persons who live in that apartment. Additionally, every building may have a garden. On the other hand, each garden belongs to one and only one building, which also provides the identification for that garden.

Observations refer to events and actions. An example of event is (4) above. Examples of actions are (2) and (3) above. To avoid ambiguities, events and actions are identified using an internal code. Events punctually occur at a given moment in time, are associated to a textual description, and are classified into "normal" events, "important" events, and "critical" events. Actions instead span over a definite time window, characterized by an initial timestamp, and possibly by a final timestamp. Actions take place in a location. In addition, they have a main subject consisting of a living being who is responsible for the execution of the action, and regard one or more living beings that are involved in the action (including, obviously, the main subject of the action).

**Problem 1.1** [6 points] Design an ORM conceptual schema that represents the fact types, object types, constraints and derivation rules related to the domain described above. Consider the textual description and the provided examples. Remember to specify all required constraints, documenting the assumptions made when the source documentation is not explicit.

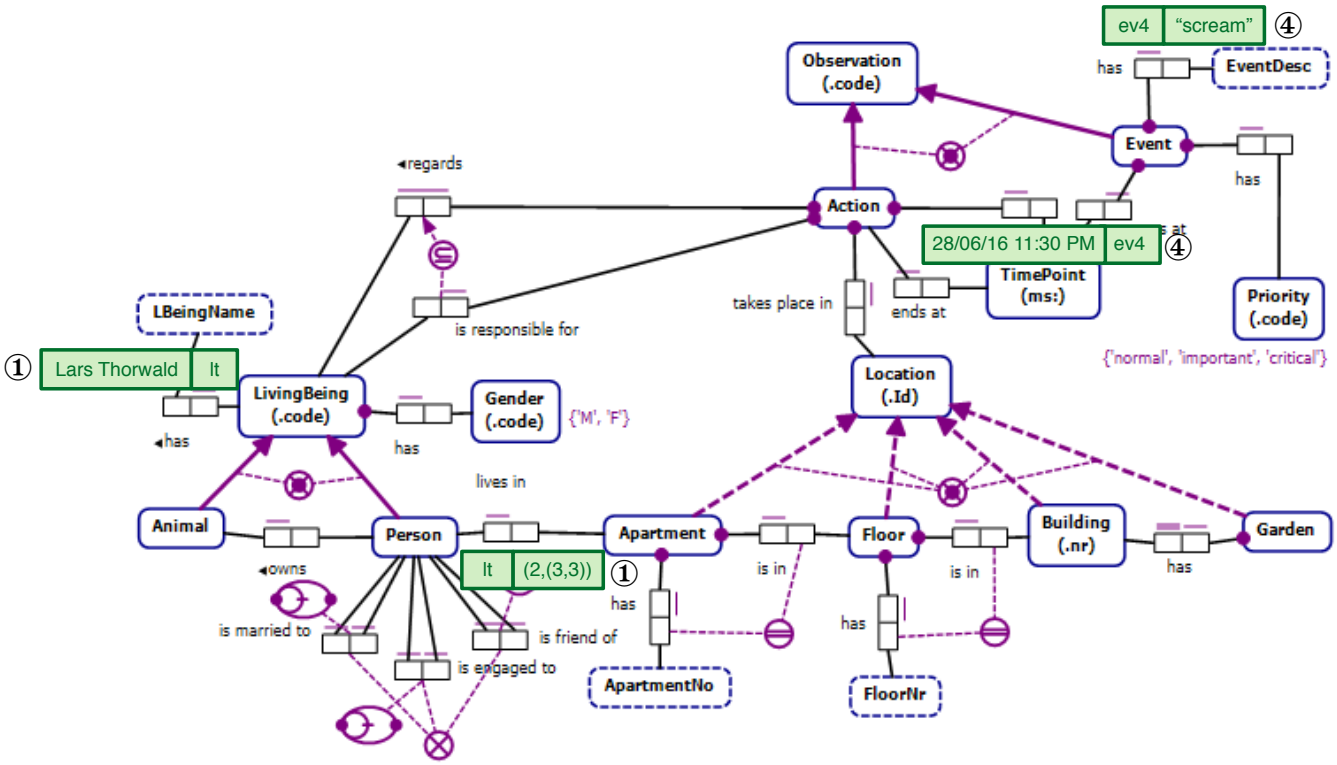
*Solution:*




N.B.: to be able to fully model the sample facts, we need to add a description also to Actions. This could be easily done by reformulating the fact type `Event has EventDesc` as the more general `Observation has ObsDesc`.

**Problem 1.2** [2 points] Enrich the ORM schema with fact tables that show how the 4 sample facts listed above may be suitably represented within the modeled diagram.

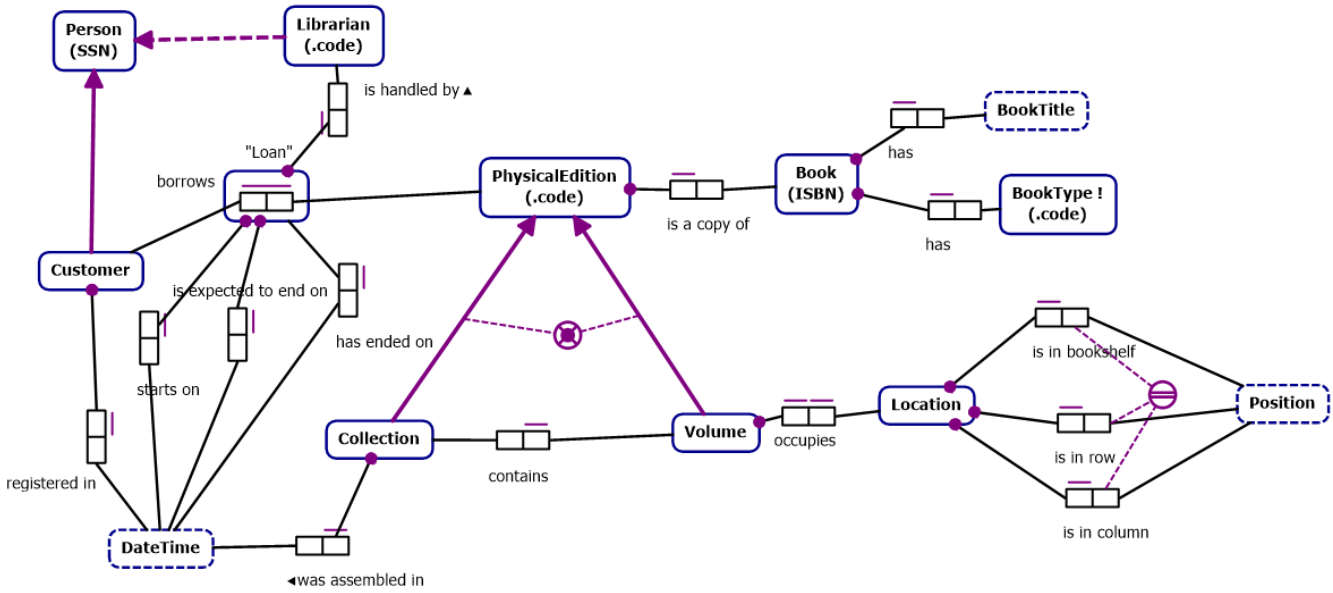
*Solution:* Facts 1 and 4 are shown, limiting to those fact types that are central for capturing them (there are obviously more facts indirectly involved). Obviously, the identification codes are not explicitly listed inside the facts, and have therefore to be invented. The time point for fact 4 is represented using the more readable string representation, not in milliseconds (as it should be, according to the schema).




As for facts 2 and 3, it is necessary to introduce a surrogate `id` for the locations, then connecting this `id` to the corresponding identification schemes of the subtypes.

## 1.2 Relational Mapping

Consider the following ORM schema, modelling a fragment of an information system used to keep track of books and loans in a library.

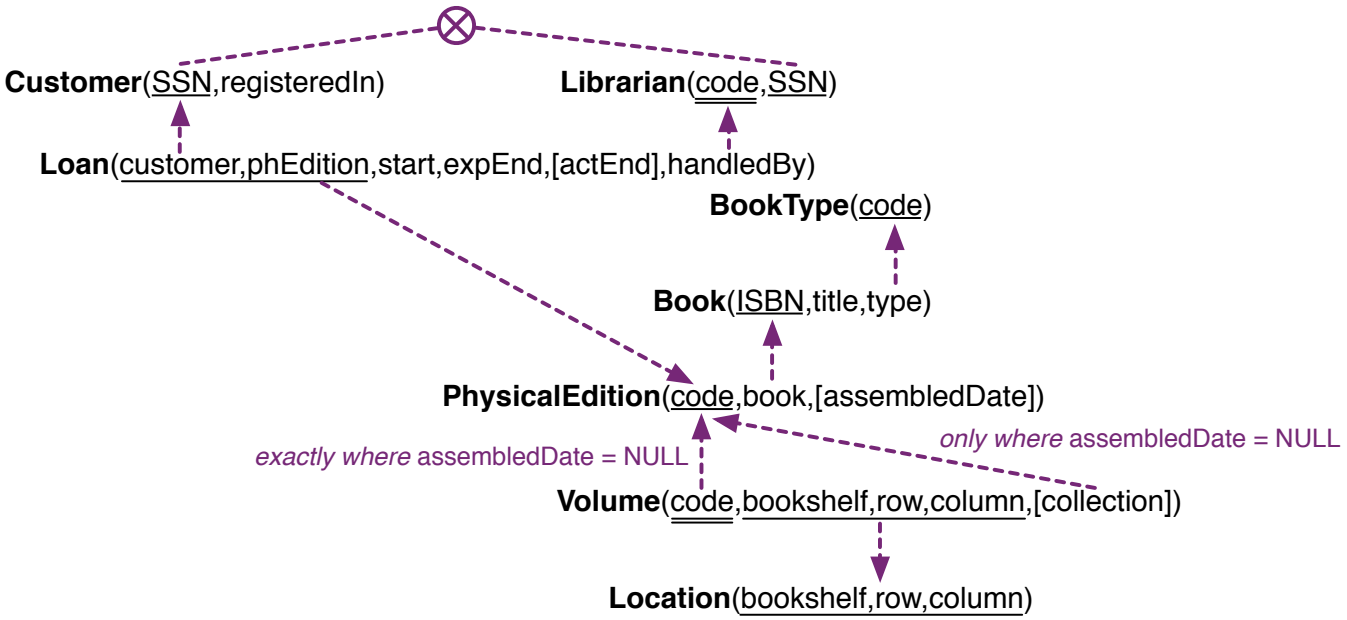



**Problem 1.3** [5 points] Build a relational schema corresponding to the (possibly restructured) ORM schema, following the `Rmap` procedure and:

- *absorbing* `Collection` into its supertype;
- keeping `Volume` *separate*;
- applying the *partition* strategy for the `Person` hierarchy.

For each relation schema that you produce, highlight the primary key, alternative keys, mandatory and optional attributes. Depict relevant constraints, including foreign keys.

*Solution:*



# 2 Process Modelling

## 2.1 Design

Consider the process for helping customers of a library to find books of interest and manage the corresponding loans.

The process starts when a request is received by a librarian from a customer, with all the necessary details. The process consists of two main phases (to be modeled as sub-processes):

1. seeking a book that matches the customer request. If no book is found, the process terminates, whereas if a book is found, the second phase is executed.
2. loan creation and book delivery.

If all the two phases are correctly executed, the process terminates by communicating to the customer the details of the created loan.

The first phase is executed as follows. First, the librarian finds a book that matches the customer request. If no book is found, the subprocess terminates. If a book is found, the librarian checks whether the book is available. If not, the librarian tries to find another matching book. If so, the book is proposed to to the customer. The librarian then waits for an answer from the customer. If the customer accepts the book, then the librarian flags the book (so as to mark that it is not available anymore) and the subprocess terminates. If the customer refuses the book, then the librarian restarts the seeking phase, trying to find another matching book.

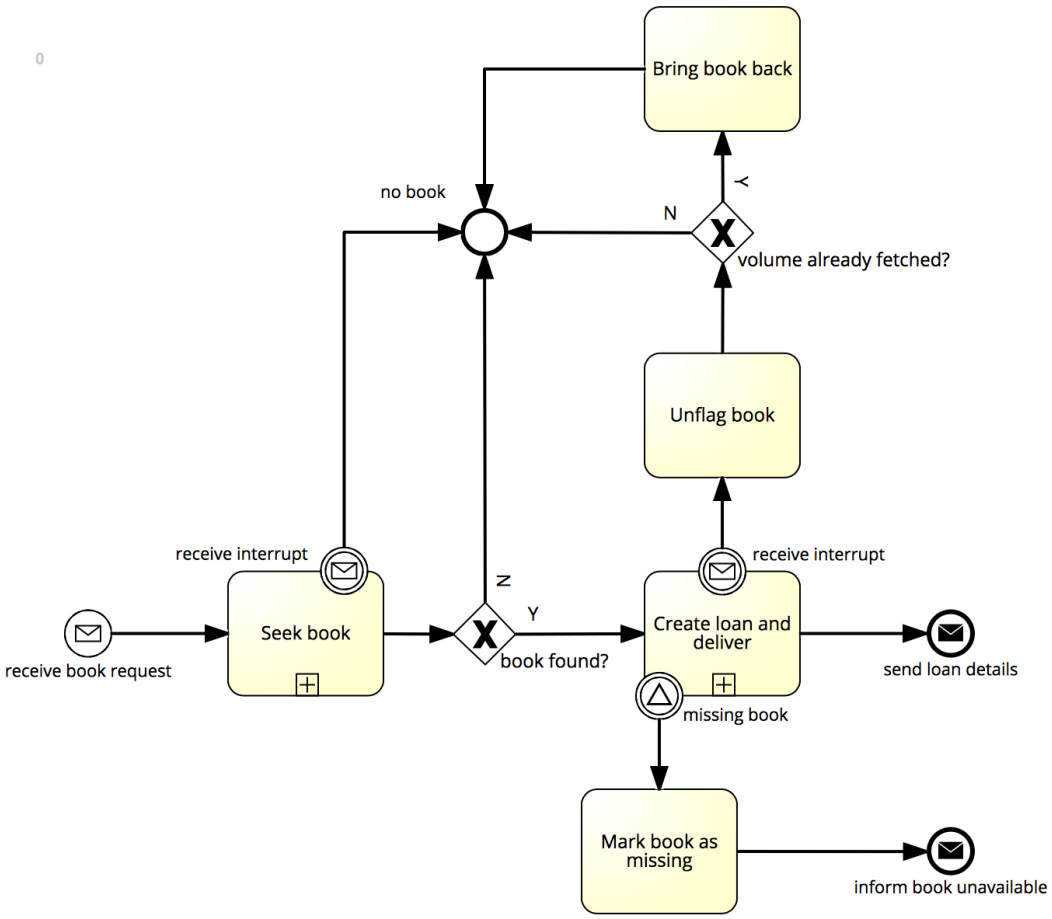
The second phase is executed as follows. First of all, the librarian physically goes fetching the volume corresponding to the book in the library (we assume that the book corresponds to a single physical volume). If the volume is not found, the librarian signals that the book is missing (this is considered to be an error, since the book should be available). If the volume is instead found, the librarian first inserts the book in a bag, then creates a corresponding loan. The librarian then delivers the book to the customer, and the subprocess terminates.

If a "missing book" signal is triggered during the second phase, such a phase immediately terminates, and the librarian reacts by marking that the book is missing, and then terminates the process by informing the customer that the book is not available.

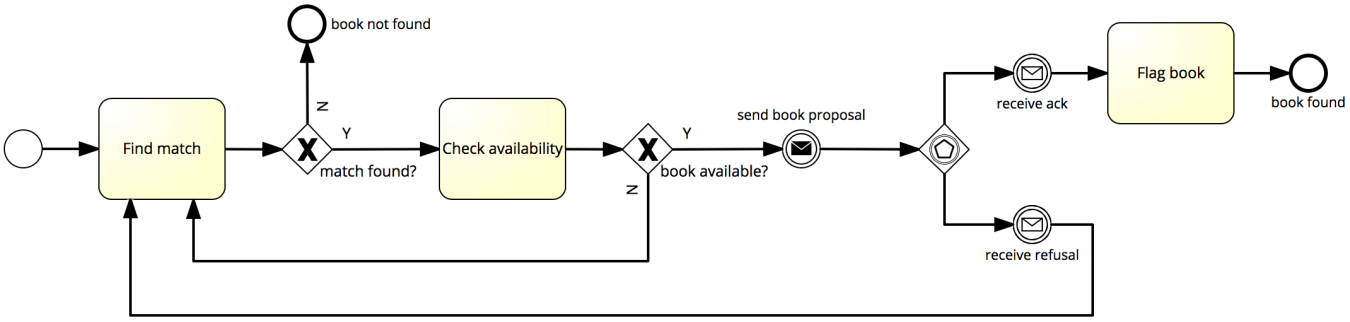
The customer may decide to interrupt the process at any moment. If the interrupt arises during the seeking phase, the librarian reacts by simply terminating the process. If instead the interrupt is received during the second phase, the librarian reacts by unflagging the book. Furthermore, if the corresponding volume has been already fetched, the librarian brings it back to its location in the library.

**Problem 2.1** [7 points] Model the internal process of the library from the librarian point of view, making use of a main process and of subprocesses that capture the two main phases.

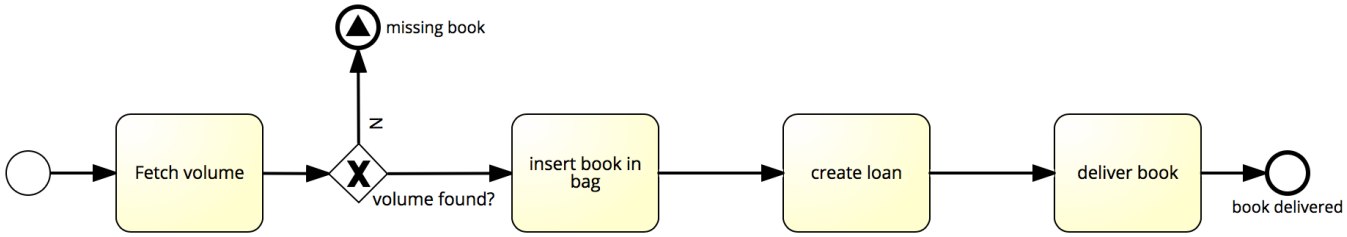
*Solution:* Main process:




`Seek book` subprocess:

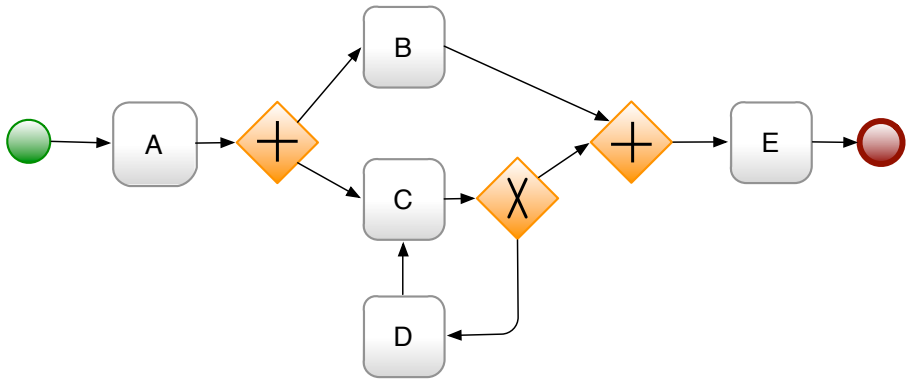



`Create loan and deliver` subprocess:



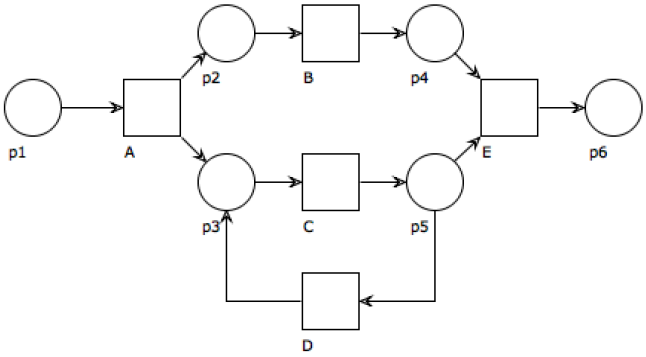
## 2.2 Analysis

Consider the following BPMN process:

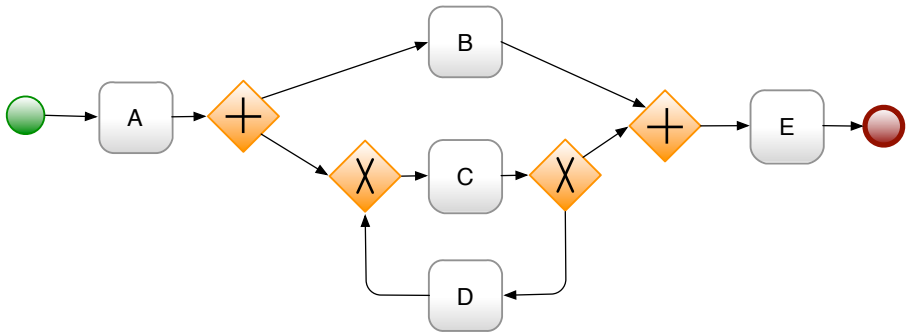



**Problem 2.2** [2 points] Convert the BPMN process into a corresponding workflow net.

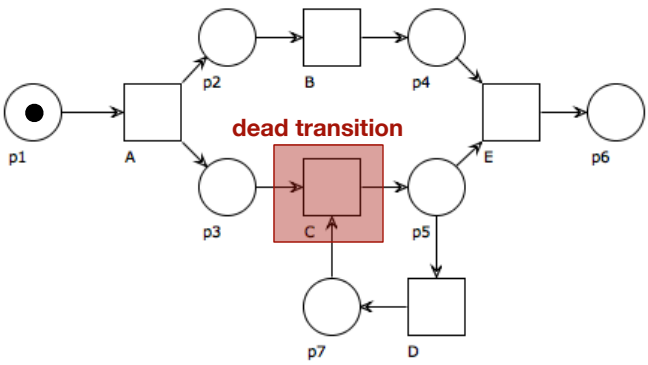
*Solution:*




**Common error made by students.** The BPMN process above is completely equivalent to the following one:



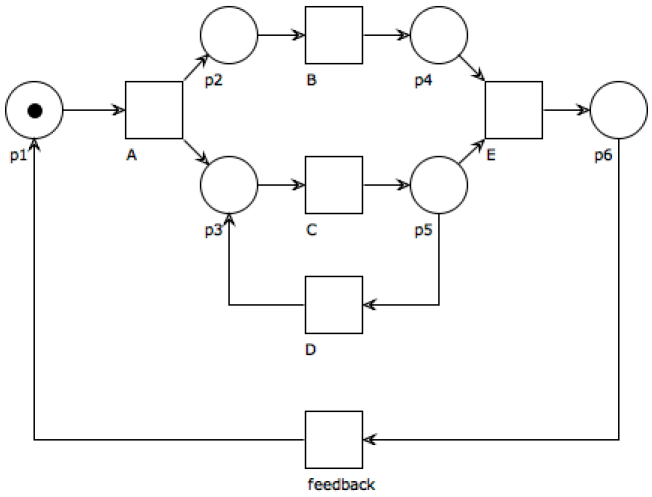

Hence, the following workflow net **DOES NOT** faithfully represent the BPMN process:



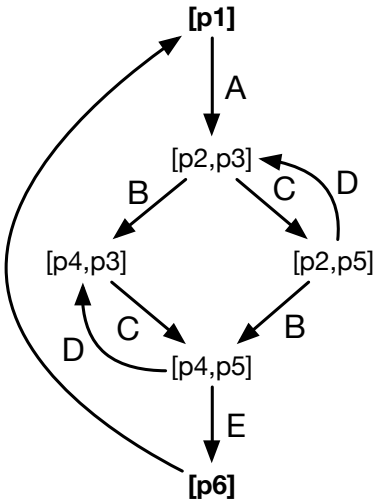
In fact, in this workflow net task `C` (and, consequently, `D` and `E`) can never be executed when starting from the usual initial marking that assigns a single token to the input place.

**Problem 2.3** [3 points] Short-circuit the obtained workflow net, and argue about its soundness by constructing and analyzing its coverability graph.

*Solution:* The marked net to be considered is:




Its coverability graph corresponds to:




We know that the original net is *sound* if and only if the corresponding marked, short-circuited net is *live* and *bounded*.

**Boundedness**. The short-circuited marked net is clearly bounded, because there is no $\omega$-marking. In fact the coverability and reachability graph coincide.

**Liveness**. The short-circuited marked net is live, since each of its transitions is live. In fact, for every transition of the net and every marking of the coverability graph, it is possible to reach another marking in which that transition can be fired.

This shows that the original process is *sound*. Notice that the presence of a loop does not undermine soundness, as soundness prescribes the "possibility to finish a case", not that this ending point is for sure achieved within a pre-defined number of steps.