Thesis Progress Report 4

Aytar Akdemir
150170115

December 15, 2021

1 Naessens' Formalization Methodology

Formalization is a collection of specifications that will define a system. While implementing a system, the rules of formalization will be used. Defining the formal specifications, comes after the requirements analysis and before the implementation. For the most abstract formalism, A Flow Graph is utilised.

In [Cite naessens], Naessens divides conceptual design to three phases:

- Order constraints and multiplicity constraints
- Authentication and authorization constraints
- Control measures

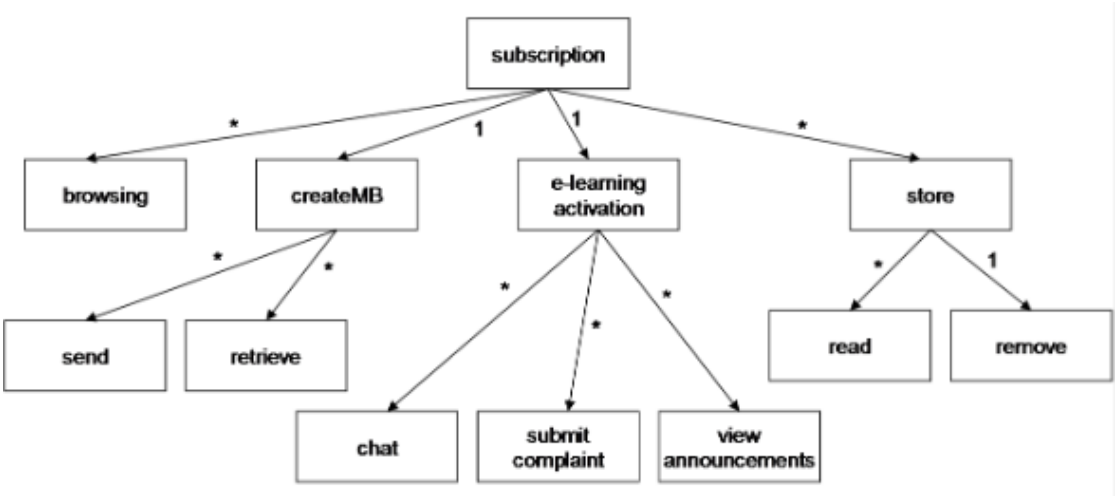
Flow chart are used for order constraints and Petri nets are used for the authorization constraints. A Linkability Graph is used for modelling control measures.

In the article, they transform the Petri Net into a Flow Graph and the Linkability Graph into a Petri Net.

1.1 Order and Multiplicity Constraints

Order and Multiplicity Constraints are basically the rules on the order of the actions that can be performed and the number of times those actions can be performed. For example a user cannot remove a file before creating that file first. As an example of the multiplicity constraint, file creation can be performed multiple times denoted by "*"; file deletion can only be performed once on a file, which is denoted by "1".

An example of a flow graph is given below.

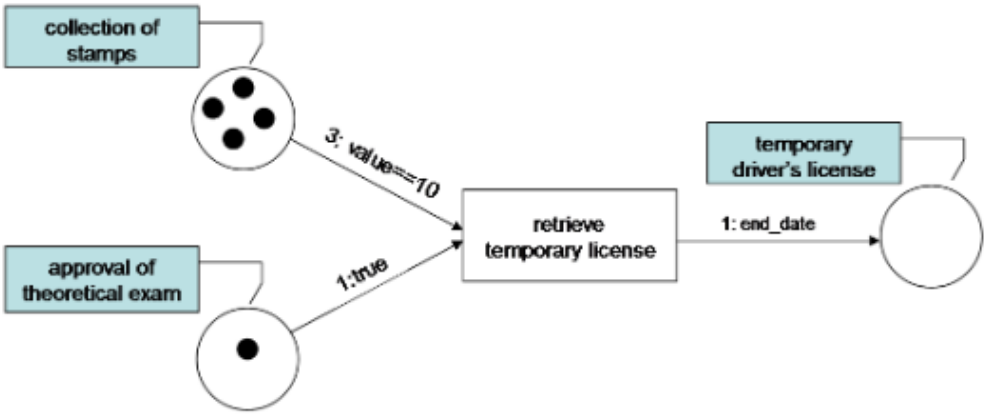


1.2 Petri Net Representation

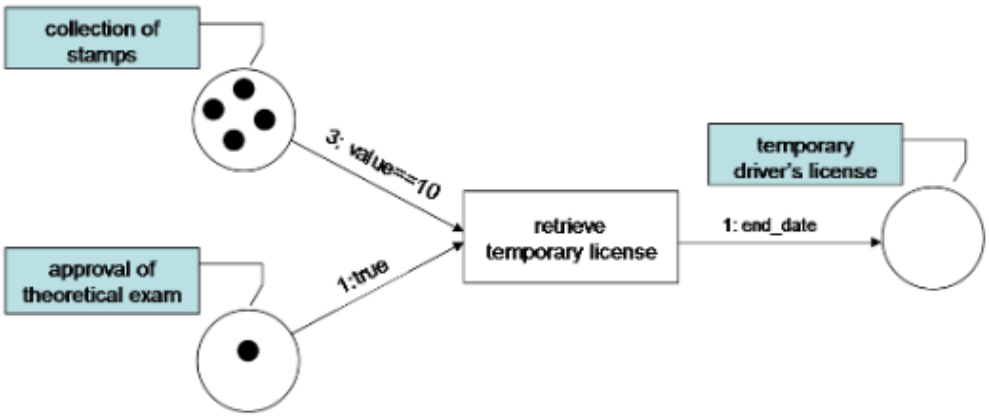
In Petri Net, each right is represented with a token. Tokens are stored in Places and each Action has Places before and after it. If there is at least one token in the Places that point to the Action, Action can be performed and the next Place is marked with a token. Inputs to an Action might have a natural number and a condition property:

- Condition can further constrain if a token is accepted by the Action.
- The natural number is the minimum number of tokens accepted.

If the output has a natural number, the specified amount of tokens will be placed to the next Place. The condition on the output is the additional constraints that are not tied to the inputs.



If the place after the action doesn't need to have a token, i.e. just prove it has the tokens to obtain a right, the token can be returned to the place before the action. After the transition, the state is unchanged even though the action has been performed.



The tokens attributes might change after the action. After the action, a mutator changes the value of the token. It is possible to have more than one transition for different conditions.