Distributed Programming II

A.Y. 2016/17

Assignment n. 3 – part a)

All the material needed for this assignment is included in the *.zip* archive where you have found this file. Please extract the archive to an empty directory (that will be called *`[root]`*) where you will work.

This assignment is about the design of an XML-based RESTful API for remote access to the system considered in the previous assignments. The web service has to be designed according to the following specifications.

- The web service must allow its clients to:
    1. Read all the information available about NFFGs and policies.
    2. Add and remove whole NFFGs.
    3. Add, remove or update whole policies.
    4. Execute the verification of one or more reachability policies that are already available in the service (the client selects the policies to be verified).
    5. Execute the verification of one or more policies passed by the client, but without storing the policies in the service (in this case, the service returns the verification results of the policies that the client has sent).
- For point 1, the information to be provided is the same that was used for Assignment 1 (the same specifications apply).
- The operations that may alter the data about NFFGs and policies in the service must never violate the specifications given for Assignment 1 (e.g. an NFFG can be removed only after having removed all the policies that refer to it, but the service may also offer the possibility to automatically remove an NFFG along with all the policies that refer to it; similarly, a policy can be added only if it refers to an existing NFFG; for point 2, the addition of a NFFG with the same name as one of the previously added NFFGs must raise an error; etc.).
- For point 2, when a new NFFG is added, its update time is set to the current time.
- For point 3, it must be possible to add policies with or without verification results.
- For point 4, the execution of the verification changes the verification result of the policy (if there was one, it is overwritten by the new one and the verification time is set to the current time).

The output of this design phase has to be written into 2 files: a report, stored into a pdf file named `[root]/doc/nffgVerifier.pdf`, and an XML schema, stored into a file named `[root]/xsd/nffgVerifier.xsd`. If you want, you can split the schema into multiple files. In this case, `nffgVerifier.xsd` must be the main file which includes the other local files and all files must be stored in `[root]/xsd`.

The report must document the design of the RESTful API. It must include the following items:

1. Description of the conceptual structure of resources (identify the resources, with their hierarchical structure; use plural names for collections and singular names for resources that are not collections; specify the methods allowed for each resource).
2. Mapping of the resources to URLs (for simplicity, use URLs that are relative to the base URL of the service; for resources that belong to collections, use the notation {id}, where id

is the name of the field that identifies the resource in the collection)

3. Description of each possible operation. For each resource, and for each method allowed on the resource, specify the allowed query string parameters and/or the allowed request body, the possible status codes, and, for each possible status code, the response body. For each allowed query string parameter, specify name and XML schema type of the value (either a standard XML schema simple type or one defined by you in the `nffgVerifier.xsd` schema document). For request body and response body, specify the XML element, which must be defined at the global level in the `nffgVerifier.xsd` schema document. Schema elements and types in this description must be referenced by name. In practice, the `nffgVerifier.xsd` schema document must include a set of global named types and a set of global elements. As far as possible, you can reuse the definitions you already developed in the schema for Assignment 1. For each possible operation, give also a short description when not obvious.

The web service must be designed so as to be compliant with the REST principles and conventions, robust, easy to be used by clients, and enabling efficient execution of all the operations for which it has been designed. The documentation should be self-contained (it should include all that is necessary for a user to know to be able to consume the service, apart from what can be deduced from the service itself).

When designing the service, consider that it should be able to scale to hundreds of policies and NFFGs (with a total of many thousands of nodes).

Correctness verification

In order to be acceptable for examination, the produced XML schema document must be valid. This can be checked by means of the Eclipse schema validator.

Submission format

The produced documents will be submitted along with Part b) of Assignment 3, for which instructions will be given later on.