5. A DSL for creating Q&A active games on Twitter

In this project, the goal is to design a DSL that facilitates the creation of Twitter bots for playing question/answer active games through the Twitter (http://twitter.com) social network. The project should provide: (a) an textual DSL to configure the games; (b) support for analyzing the game designs based on Petri nets; and (c) execution support for the games, based on Twitter APIs like Twitter4J (http://twitter4j.org/en/).

Requirements

The games we target will consist of tests that are to be solved across different spots within a city, within a given time limit. Each test can be configured with a place where the answer needs to be given, a time limit, the text of the question, a set of possible answers, a number of attempts, a set of hints (to be provided by the bot when the answer is wrong), and a number of points to be awarded if the answer is correct. In addition, the designer of the game needs to specify the next tests when the answer is correct, when the answer is wrong, and when no answer is provided within the time limits. Each game should have exactly one initial test and one or more final tests.

The bot should wait until a Twitter user follows his account. Then, it gives a welcome message (as a direct message) with playing instructions. Then, the bot should send information about the initial test, including the location, maximum time and allowed number of attempts. A test is successfully fulfilled by a user if: (i) the answer is correct; (ii) the position of the user when the tweet is sent is within the coordinates of the location, (iii) the tweet was within the time limit. Please note that the tweet text, timestamp and position can be extracted with Twitter4J.

In addition, to validate the design of the game, a transformation into (timed) Petri nets should be provided (where users are represented by tokens and tests by places). Using such Petri net, the game designer will be able to check reachability of final tests, calculate mean completion times of the game, etc.

Just as an example, a possible syntax for the DSL could be as follows:

```
location Sol “Puerta del Sol” [40.4167278, -3.7033387]  // we could also give a maximum deviation
location PlazaMayor “Plaza Mayor” [40.415363, -3.707398]

welcome: “Welcome to the game ‘El Madrid de los Austrias’ with quizzes about the history of Madrid”

initial Test t1 at Sol with duration 20 {
        Question: “Who is represented in the equestrian statue?”
        Answers: [“Carlos III”, “Charles the third”]
        Attempts: 2
        Hints:  “A king whose nickname was `The best Mayor of Madrid’”
        Points: 10
} on correct: t2, on fail: t3, on timeout: lose

Test t2 at PlazaMayor with duration 20 { // one attempt by default, and hence no hints needed
        Question: “When was the square renovated?”
        Answers: [“1961”, “61”]
        Points: 20
} on correct: t4, on fail: t5, on timeout: t2
```