Beyond Code: An Introduction to Model-Driven Software Development (CISC 836, Fall 2021)

Assignment 4 (DSLs with Xtext)

Due: Tuesday, November 30

1. **Purpose of the Assignment:**
   To introduce you to the use of Xtext to
   - define the syntax of language and generate an editor for it
   - implement validation (i.e., well-formedness) constraints for the language
   - define the execution semantics of the language through the implementation of an interpreter
   - define the execution semantics of the language through the implementation of a code generator

2. **Getting the necessary software:**
   You will need the following software for this assignment:
   1. 'Eclipse IDE for Java and DSL Developers' (version 2020-12 R). The recommended way of obtaining this is to follow the instructions in Section 2 of the tutorial Creating a Domain Specific Language (DSL) with Xtext by Mooij and Hooman. The instructions use the Eclipse Installer which includes the required JRE. You can also download Eclipse as a package (https://www.eclipse.org/downloads/packages), but then you also need to manually install the JRE (https://adoptopenjdk.net/releases.html), if you don't have it already (Java 11 or greater is recommended).

3. **Getting started with Xtext:**
   The following documentation will help you get started:
   1. Work through Sections 3.1 to 3.7 in the tutorial Creating a Domain Specific Language (DSL) with Xtext. You can skip Sections 3.8 to 3.9 (material that is more advanced than what is needed for the course). Sections 4 and 5 contain some pointers for trouble shooting, hints for using Eclipse (e.g., exporting and importing projects), and version management, which you can also skip. Section 6 contains helpful background information on Xtext, Xtend, and parsing.
   2. If you want another intro to code generation with Xtext and Xtend, you can also view the video tutorial at javadude.com/articles/xtext.
   3. The reference documentation for Xtext (together with two more tutorials) can be found at www.eclipse.org/Xtext/documentation. A cheat sheet for the Xtext grammar langauge can be found here.
   4. This assignment does not require knowledge of Xtend. However, in case you are interested, documentation is at www.eclipse.org/xtend/documentation. An Xtend cheat sheet can be found here.

4. **Assignment: [50 points]**
   1. **Problem Description:**
      In this assignment, you will extend a domain-specific language called 'Urml', a much simplified, textual version of UML-RT.

      Just like UML-RT, Urml is meant for the description of embedded systems with soft real-time constraints as found, e.g., in the telecommunications industry. It supports structural modeling via 'capsules', i.e., components can communicate with each other only by sending and receiving

messages over 'connectors' and 'ports', i.e., boundary objects that are typed with 'protocols' which define the messages that are accepted by the port. Capsules can be contained in other capsules and every model has a root capsule. The behaviour of every capsule is described by means of a statemachine; a transition consists of a trigger, a (possibly empty) guard, and (possibly empty) action code. A message sent to a capsule will cause a transition to be taken and its action code to be executed, if the source state of the transition is currently active, the message matches the trigger, and the guard evaluates to 'true'. The action code is given in a simple imperative language with constructs of sending and receiving messages.

Urml was developed by Keith Yip and for more details on the language and its implementation, please see his MSc thesis.

2. **Preparation:**
    1. Import the Urml projects in the .zip file at UrmlWithoutChoose_CISC836_F21.zip. Instructions for how to import projects from a .zip file can be found in Section 5.2.2 of Creating a Domain Specific Language (DSL) with Xtext.
    2. Review Xtext grammar for Urml in `ca.queensu.cs.mase.urml/src/ca.queensu.c.mase/Urml.xtext`.
    3. Generate the language infrastructure.
    4. Review the generated Urml metamodel in `ca.queensu.cs.mase.urml/src-gen/ca.queensu.cs.mase/model/generated/Urml.ecore`.
    5. Review the Java code generated from the Urml metamodel in `ca.queensu.cs.mase.urml/src-gen/ca.queensu.cs.mase/urml`.
    6. Start the editor (i.e., another Eclipse instance running the Urml editor).
    7. In the editor, import Urml examples from the .zip file UrmlExamples_CISC836_F21.zip. To be able to run the Java code automatically generated from the Urml models, create a Java project in the editor and copy-and-paste the folders `model` and `urml` containing the generated code into the source folder of the Java project.
3. **Editor:**
    1. Explore the capabilities of the editor (syntax high-lighting, auto-completion with context assist, and validation).
    2. Review the validator code in package `ca.queensu.cs.mase.validation`.
4. **Interpreter:**
    1. Experiment with interpreter: right click the .urml file you want to execute, select 'Run As' and then one of three execution modes
        - '4 Urml Model (First transition)': when several transitions are enabled, the 'first' one found is picked
        - '5 Urml Model (Interactive)': when several transitions are enabled, the user is asked to pick
        - '6 Urml Model (Random transition)': when several transitions are enabled, one is picked at random
    2. Review interpreter in package `ca.queensu.cs.mase.interpreter.*`. Note that, e.g., whenever more than one transition is enabled, the interpreter (see class `TransitionSelector` in `ca.queensu.cs.mase.interpreter.filter`) prints 'NON-DETERMINISM' in the console.
5. **Code generator:**
    1. Review generated code: the code for all .urml files automatically generated in folders 'model' and 'urml' in folder 'src-gen'.
    2. Review code generator in package `ca.queensu.cs.mase.generator.*`.
6. **Tasks to be completed: [50 points]**

    Your task is to extend Urml and its interpreter and code generator with an 'choose' statement.
    1. [10 points] Add an `choose(x,e)` statement to the syntax of Urml where `x` is a local variable or an attribute and `e` is an integer expression.
    2. [10 points] Modify the interpreter to handle choose statements. An execution of `choose(x,e)` should first evaluate the expression `e`; if `e` evaluates to an integer `i` greater than `0`, then `x` is

randomly assigned an integer greater or equal to `0` and less than `i`; if `e` evaluates to an integer equal or less than `0`, `0` is assigned.
3. [10 points] Modify the code generator to handle `choose(x,e)` appropriately.
4. [15 points] Implement an Urml model to simulate the game 'Rock, Paper, Scissors'. Your implementation should contain two player capsules (`Player1`, `Player2`) and a referee capsule (`Referee`). The game starts when the referee sends a `go` message to each of the players, which starts the first round. In each round, the players pick one of 'rock', 'paper', or 'scissors' at random and send their pick to the referee. Game play ends after at most three rounds and the winner is determined using the 'best-of-three' rule, i.e., the player with the most winning hands after three rounds wins. The referee should output appropriate messages.
5. [5 points] Modify the system so that a game stops soon as possible (i.e., after possibly just two rounds).

5. **What to submit to OnQ:**
   - Export all Urml projects into a .zip file: In the Package Explorer, select `ca.queensu.cs.mase.urml`, `ca.queensu.cs.mase.urml.sdk`, `ca.queensu.cs.mase.urml.tests`, and `ca.queensu.cs.mase.urml.ui`, right-click, select 'Export...', select 'General' -> 'Archive File'. Use '`[firstName]_[lastName]_A4_CISC836_F21.zip`' as name where '`[firstName]`' and '`[lastName]`' are replaced by your first and last names, respectively. Upload this archive to OnQ.
   - In the Urml editor, export the project containing your Urml Rock-Paper-Scissors model into a .zip file. Use '`[firstName]_[lastName]_A4_CISC836_F21_RPSModel.zip`' as name where '`[firstName]`' and '`[lastName]`' are replaced by your first and last names, respectively. Upload this archive to OnQ.

Last modified: Wed Nov 10 2021 13:00:46