Deliverables for Iteration 4 – State Machines (6%)

The deliverable for Iteration 4 aims to specify two key features of the Quoridor game using state machines and implement them with the help of automated code generation as provided by Umple. Using a set of core interfaces provided by us for the related Controller methods as well as the guards, you need to (a) provide (one or more) state machines to handle movements in a specific direction (up, down, left, right) for Move Pawn and Jump Pawn features, (b) generate code from the state machine(s) using Umple, (c) implement the methods in the Controller interface and helper methods in the statemachine class, (d) provide the related Gherkin step mappings so that acceptance tests would pass, and (e) integrate the behavior in the View (GUI) of your Quoridor application.

The deliverable is due on **Sunday, November 17, 2019, at 23:30**.

See the Project Overview document for a general description of the ***Quoridor*** application, an overview of all deliverables, technical constraints, and general rules regarding project reports, submission of source code, and member contributions. Note that you are responsible for this deliverable as a team.

1 Statemachine Specification(s)

You need to implement state-based behavior for a game of two ***Quoridor*** features to support ***pawn moves*** and ***pawn jumps***. Specify the state machine(s) in **a separate Umple file** provided to you in *PawnStatemachine.ump* in a separate branch (iteration-4). All relevant triggers, guards and actions in your statemachine (on transitions as well as in states) need to be defined in (public or private) methods. A state machine itself may only show which methods are called for guards and actions.

You are required to use the pre-declared private methods in your statemachine "as-is" (i.e. you still need to provide their implementation). Moreover, you are allowed to define extra private methods if you need. You need to use the automated code generation support of Umple to derive a fully functional implementation of the two features from the state machine.

The source code automatically generated from the statemachine specification (as well as from the domain model) has to be committed to the GitHub repository using the predefined naming pattern.

2 Implementation of Controller and View

You need to define two methods in the Controller interface corresponding to the two required features that interact with the statemachine(s) by triggering relevant events. In addition, you need to extend the View (i.e. your graphical user interface) to initiate the call to the corresponding methods of the Controller interface. All methods implemented manually have to be documented using JavaDoc.

By the end of this Deliverable, one could play an entire game with your Quoridor application (i.e. all pawn moves and wall moves should be supported), but you are not yet required to support the remaining 12 features of Phase 2 (e.g. the path to the target area is allowed to be blocked at this stage).

Each team is required to use the **common Umple domain model** provided to you. The Umple model can be extended, but existing definitions in the Umple model cannot be changed! You are required to use the code generated by Umple.

The whole team is also responsible for the **Quoridor** application including its uniform look and feel across all features and the completeness of the application. Your application will be assessed as a whole.

3 Mapping of Gherkin Scenarios

The mapping of each step in the two Gherkin scenarios (Move Pawn and Jump Pawn) need to be assigned to individual team members documented in the JavaDoc header (*@author*) of each step mapping code. Nevertheless, the mapping of Gherkin steps will be graded for the entire team this time (with shared responsibility). As part of this deliverable:

- Connect the When clause to trigger a step in your statemachine.
- Implement all Given, Then and And clauses. These steps shall use the Umple instance model (Quoridor object) or an appropriate query method (similarly to Iterations 2 and 3).
- Implement all GUI related steps by acquiring data from your View.
- Ensure that your project successfully compiles (no compile errors).
- Ensure that the execution of Gherkin steps can be successfully initiated by Cucumber (using the Gradle build file provided to you in the initial content of the source code base).
- Ensure that all Gherkin scenarios pass successfully as acceptance tests.

4 Project Report

You need to document the behavior of your state machine on the project wiki of your team. This should include a UML Statechart diagram for each of your statemachine and the documentation of any extra methods you introduced for guard and actions.

Submission

Your team is required to follow the General Rules explained in the Project Overview document.

Marking Scheme

Deliverables for Iteration 4 of Project	***Marks***
Quoridor project (team mark)	90
Statechart specification in Umple	25/90
Statechart implementation (incl. Move Pawn and Jump Pawn features in Controller)	25/90
Implementation of Gherkin step mappings	15/90
Integration with GUI	10/90
Documentation in JavaDoc (for all manually implemented methods)	10/90
Uniform look and formatting of application	05/90
Statechart Project Report (team mark)	10
Description of extra methods for triggers, guards and actions	05/10
UML Statechart diagram	05/10
Total Marks:	100
The total mark may be adjusted based on the actual contributions of a team member to the deliverables.	