May 15, 2017

Generating Basic Robotics Programs: Requirements

Purpose and Scope:

Purpose:

⇒ User can generate basic code for robotics programs.

Scope:

⇒ User can generate basic elements that we spoke about in Robotics 235.
⇒ Code will be generated in Java.

Actors and Stakeholders:

Actors:

⇒ Users
⇒ Application

Stakeholders:

⇒ Dr. Ferrer – Wants this program to work for future Robotics students
⇒ Creator – Wants to capture Dr. Ferrer's vision and help future students with little knowledge of Java to be able to write programs efficiently and effectively for the course.
⇒ User – The user should be able to write code efficiently and should have a template for writing Java code for robotics.

---

1. Use Case: Launch application –

Primary Actor: User

Precondition: Computer on and working

Trigger: User selects application

Minimal Guarantee: Computer will not crash

Success Guarantee: App will successfully launch & be ready for use

Main Success Scenario:

1. User launches app
2. App prepared for user

Extensions:

1a. App fails to launch

1a1. Error message displayed to user

1a2. Use case terminated

2a. App incorrectly loaded

2a1. Error message displayed to user

2a2. App restart

2. Use Case: Add Condition and Mode Enums–

Primary Actor: User

Precondition: Computer functioning properly, app launched correctly

Trigger: User indicates that they would like to add Conditions and Modes

Minimal Guarantee: notification of successful addition

Success Guarantee: App will successfully add Conditions and Modes to current project

Main Success Scenario:

1. User selects 'Conditions and Modes'
2. User will enter conditions and modes desired
3. App will create enums for conditions and modes containing desired conditions and modes

Extensions:

2a. User does not enter any conditions or modes

2a1. Empty enum set is created

3. Use Case: Add Flagger–

Primary Actor: User

Precondition: Computer functioning properly, app launched correctly

Trigger: User indicates that they would like to add a flagger

Minimal Guarantee: notification of successful addition

Success Guarantee: App will successfully add a flagger of desired type to code output.

Main Success Scenario:

1. User selects flagger type
   a. If type is sensor, they will identify whether touch or ultrasonic
2. User will identify sensor port, motor, or button if selected flagger is Sensor, Motor, or Button
3. App will add code for desired sensors

Extensions:

2a. User does not identify port, motor, or button

2a1. Code is generated in a comment (to avoid overwhelming errors) and description is in place of port, motor, or button

4. Use Case: Add Transition Tables-

Primary Actor: User

Precondition: Computer functioning properly, app launched correctly

Trigger: User indicates that they would like to add a transition table

Minimal Guarantee: notification of successful addition

Success Guarantee: App will successfully add a transition table to code output.

Main Success Scenario:

1. User indicates that they would like to add a transition table
2. Add modes and conditions
3. App will add code for table

Extensions:

2a. User does not identify modes and conditions

2a1. Table will be empty

FUTURE THOUGHTS

1. Use Case: Add Methods: Q-Learning–
Primary Actor: User
Precondition: Computer functioning properly, app launched correctly
Trigger: User indicates that they would like to use Q-Learning
Minimal Guarantee: notification of addition
Success Guarantee: App will successfully add elements of Q-Learning to the code.
Main Success Scenario:
1. User indicates that they would like to use Q-Learning
2. App will add QController and StateClassifier to the generated code

2. Use Case: Add Methods: PID –
Primary Actor: User
Precondition: Computer functioning properly, app launched correctly
Trigger: User indicates that they would like to use PID
Minimal Guarantee: notification of addition
Success Guarantee: App will successfully add elements of PID to the code.
Main Success Scenario:
1. User indicates that they would like to use PID
2. App will add PIDCalculator and appropriate mode selector to generator code

3. Use Case: Add Methods: Fuzzy Logic –
Primary Actor: User
Precondition: Computer functioning properly, app launched correctly
Trigger: User indicates that they would like to use Fuzzy Logic
Minimal Guarantee: notification of addition
Success Guarantee: App will successfully add elements of Fuzzy Logic to the code.
Main Success Scenario:
1. User indicates that they would like to use Fuzzy Logic
2. App will add FuzzyRuleBase, Defuzzifier, and appropriate mode selector to generator code

4. Use Case: Add Methods: Cluster Training –
Primary Actor: User
Precondition: Computer functioning properly, app launched correctly
Trigger: User indicates that they would like to use Cluster Training
Minimal Guarantee: notification of addition
Success Guarantee: App will successfully add elements of Cluster Training to the code.
Main Success Scenario:
1. User indicates that they would like to use Cluster Training
2. App will add Move enum, training file, retraining file, and TrainedController to project (need to look deeper into this)

5. Use Case: Add Methods: Landmarks –
Primary Actor: User
Precondition: Computer functioning properly, app launched correctly
Trigger: User indicates that they would like to use Landmarking
Minimal Guarantee: notification of addition
Success Guarantee: App will successfully add elements of Landmarking to the code.
Main Success Scenario:
1. User indicates that they would like to use Landmarking
2. App will add landmark flagger to main method, map trainer, and map viewer to project.