MASS Java Annotations

Work performed, Next Steps

Matthew Sell
Winter 2019
Introduction

My work performed on MASS during Winter 2019 focused on annotation support:

1. Creation of initial annotations (interface classes)
2. Creation of an Event Dispatcher for queueing events
3. Integration of annotations into existing MASS library
4. Modification of QuickStart sample application to demonstrate annotation usage

The main goals of adding support for annotations into MASS are:

1. Simplify development of applications
2. Encourage an event-oriented approach which, for some simulation or data analysis applications, can be more intuitive for developers

Initial Annotations

The first three event annotations selected for implementation are:

- OnCreation
- OnArrival
- OnDeparture

All three annotations were created and simple implementations added to “SimpleTestAgent” to facilitate unit test creation.

Event Dispatcher

After discussions of the modifications necessary to add support for annotations to the existing MASS-Java library, it was decided that events of a single type (such as OnArrival) would need to be placed into a queue for batch execution. What I had envisioned as a relatively simple helper class to extract annotated methods from classes would actually need to be a queueing event dispatcher.

An interface, “EventDispatcher”, was created to set up the API for my first implementation and for future new implementations.

A new class, “SimpleEventDispatcher”, was created. This class contains helper methods for locating annotated methods as well as invoking (synchronously and asynchronously) and queueing these methods. Invoking the “invokeQueuedAsync” method (and providing the annotation) will asynchronously invoke all queued methods in parallel.

MASS-Java Annotation Integration

To perform a runtime test of annotated event method invocation, the constructor of AgentsBase was modified to add an Agent “OnCreation”-annotated method to the EventDispatcher queue as each Agent was created. After all Agents are instantiated, the queued OnCreation methods are invoked.
QuickStart Modifications

The “QuickStart” sample application was modified to use annotated methods in the “Nomad” Agent class. A List field was added to contain a history of Agent events; messages are added to the list as annotated event methods are invoked during the lifecycle of the Nomad.

Next Steps

- My initial implementation of method queues did not include a means for specifying a priority, which will be necessary to separate events for Places and Agents. For example, when an Agent is moved to a new Place, “OnArrival”-annotated methods for both the Place and Agent must be queued for invocation after all movements have been performed. When the EventDispatcher is instructed to invoke all OnArrival methods, it will do so asynchronously and it is quite probable that OnArrival methods for Places and Agents would be intermixed.
- Remaining “hooks” for queueing events in AgentsBase and PlacesBase need to be added. At this point the issue of queue priority needs to be solved first before more hooks are added.
- Add “hooks” for invocation of annotated event methods on remote nodes
- Determine which annotated events should be added next (such as those for Agent termination)