CSS497 Fall 2005

Redesigning and Enhancing the UWAgent Execution Engine

Status Report: Fri 10/14/2005

Summary

Dates: Sat 10/08/2005 – Fri 10/14/2005

This work period:

- Finished modifying UWAgent.java as specified in ~/orange/temp_AgentUtil/UWAgent.java.
- Modified the files that use UWAgent, to accommodate the changes: UWAgentMailbox, UWInject, UWPlace, HopTest*, SpawnTest*, MessageTest, CascadeTest.
- Ran test agents to verify that they still work.

Next work period:

- Implement inter-agent communication over a gateway.

Code Changes

Some comments on the code changes specified in ~/orange/temp_AgentUtil/UWAgent.java:

- One of the suggested modifications for UWAgent.java was to set several member variables in a constructor rather than through public methods. This is problematic: UWInject.engine() uses the newInstance() method to instantiate a UWAgent derived class and pass arguments to its constructor. It then casts the instantiated object to the base class type, and initializes several variables using public methods. If these initializations were moved into a UWAgent base class constructor, the derived class constructor could not also be called in UWInject.engine(). A base class constructor can be called using super() inside a derived class constructor. But we don't have any control over what goes in the constructor of a user agent that is derived from UWAgent. So we will need to leave agentId, agentName, clientUserName, timeStamp, and maxChildren as public properties of UWAgent.
- Without the agentWithinDomain() method and the agentAccessList Vector, there is no longer any concept of a "valid sender" (or an invalid message log) in enqueueMessage().
- An entry at the top of the list suggests renaming the boolean variable endFlag to noBlockedOnReceive. Later (in restartThread()), the suggested name is noMessageWaiting. I chose noMessageWaiting.
- Here are the new and modified spawnChild methods. I removed the spawnChildSubMethod method, but created two private spawnChildInternal methods because of the requirement of passing in a specific child ID to create:

```
// No specified child ID
public UWAgent spawnChild(String agentName, String[] agentArgs,
          String host, String[] classNames) {

     return spawnChild(agentName, agentArgs, host, classNames, -1);
}
```

```
/**
 * Method spawnChild allows a UWAgent to spawn a child
 * process of any class, inheriting appropriate identifier information.
 *
 * If other classes are needed to spawn the agent, then this overload is used.
 * The class names are given in the String[] classNames argument.
 */
public UWAgent spawnChild(String agentName, String[] agentArgs,
                String host, String[] classNames, int specifiedChildId) {

        UWUtility.LogEnter("agentName = " + agentName);

        if (classNames != null) {
                for (int i = 0; i < classNames.length; i++) {
                        if (!agentClassNames.contains(classNames[i])) {
                                agentClassNames.add(classNames[i]);
                        }
                }
        }
        UWUtility.LogExit();
        return spawnChildInternal(agentName, agentArgs, host, specifiedChildId);
}

/**
 * If specifiedChildId is in the range of my child agent identifiers and the
 * corresponding child agent does not actually exist, create a new
 * agent and give it the specifiedChildId.
 */
private UWAgent spawnChildInternal( String agentName, String[] agentArgs,
                String host, String[] classNames, int specifiedChildId ) {

        // Root agent is limited to one fewer children than other agents
        int maxChildrenAdj = maxChildren;
        int parent = getParentId(agentId);
        if (parent == -1)
                maxChildrenAdj--; // this is the root agent

        // Check child agent id range
        if ( (specifiedChildId < (agentId * maxChildrenAdj)) ||
             (specifiedChildId > (agentId * maxChildrenAdj + maxChildrenAdj)) ) {

                return null;
    }

        // Check if child agent already exists
        if (childIds.contains(new Integer(specifiedChildId)))
                return null;

        // Return new agent
        return spawnChild(agentName, agentArgs, host, classNames, specifiedChildId);
}

// No specified child ID
public UWAgent spawnChild(String agentName, String[] agentArgs,
        String host) {

        return spawnChildInternal(agentName, agentArgs, host, -1);
}

/**
 * Method spawnChild allows a UWAgent to spawn a child
 * process of any class, inheriting appropriate identifier information.
 */
private UWAgent spawnChildInternal(String agentName, String[] agentArgs,
                String host, int specifiedChildId) {

        UWUtility.LogEnter("agentName = " + agentName);

        if (!agentClassNames.contains(agentName)) {
                agentClassNames.add(agentName);
        }

        Object agentObj = makeObjectFromClass(agentName, agentArgs);

        if (agentObj == null) {
                UWUtility.Log("Cannot create an object for agent " + agentName);
                return null;
        }

        // Root agent is limited to one fewer children than other agents
        int maxChildrenAdj = maxChildren;
        int parent = getParentId(agentId);
        if (parent == -1)
                maxChildrenAdj--; // this is the root agent

        // Verify that we haven't exceeded child limit
        if (childIds.size() >= maxChildrenAdj) {
                UWUtility.Log("Cannot spawn more than " + maxChildrenAdj + " children.");
                return null;
        }

        ((UWAgent) agentObj).setNextFunc("init");
        ((UWAgent) agentObj).setClientName(clientUserName);
        ((UWAgent) agentObj).setMaxChildren(Integer.toString(maxChildren));
        ((UWAgent) agentObj).setName(agentName);
        ((UWAgent) agentObj).setIp(getIp());
        ((UWAgent) agentObj).setTimeStamp(getTimeStamp());

        // Set or calculate child ID
        int childId = -1;
        if (specifiedChildId == -1) {
                childId = agentId * maxChildrenAdj + nextChildId++;
                if (agentId == 0) {
                        childId++;
                }
        } else {
                childId = specifiedChildId;
        }

        ((UWAgent) agentObj).setAgentId(childId);
        childIds.add(new Integer(childId));

        try {
                if (host.equals("localhost")) {
                        registerAgentIp(childId, InetAddress.getLocalHost());
                } else {
                        registerAgentIp(childId, InetAddress.getByName(host));
                }
                ((UWAgent) agentObj).setParentAgentIp(InetAddress.getLocalHost());
        } catch (java.net.UnknownHostException e) {
                UWUtility.Log(e.toString());
                UWUtility.Log("Cause: " + e.getCause());
        }

        injectAgent((UWAgent) agentObj, host);

        UWUtility.LogExit();
        return (UWAgent) agentObj;
}
```