GMoDS-based Runtime Agent Role Interpreter

Presentation II

Kyle Hill
Agenda

• Action Items
• Project Plan
• Architecture Design
• Formal Specification
• Technical Inspection Checklist
• Test Plan
• Limitations and Future Work
• Architecture Prototype Demo
• Questions and Comments
Action Items

• Added additional information in project plan for code size estimate by component

<table>
<thead>
<tr>
<th>Component</th>
<th>SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launcher</td>
<td>50</td>
</tr>
<tr>
<td>Agents</td>
<td>700</td>
</tr>
<tr>
<td>Capabilities</td>
<td>1000</td>
</tr>
<tr>
<td>Mapping and Navigation</td>
<td>1600</td>
</tr>
<tr>
<td>Goals</td>
<td>100</td>
</tr>
<tr>
<td>Roles</td>
<td>300</td>
</tr>
<tr>
<td>Role Interpreter</td>
<td>1000</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>4750</strong></td>
</tr>
</tbody>
</table>
### Project Plan – Productivity

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time (hours)</th>
<th>Quantity</th>
<th>Productivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Writing</td>
<td>89</td>
<td>11 documents</td>
<td>0.12 documents/hour</td>
</tr>
<tr>
<td>Development</td>
<td>77</td>
<td>~4038 SLOC (85% of 4750)</td>
<td>52.4 SLOC/hour</td>
</tr>
<tr>
<td>Research</td>
<td>22</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>188</strong></td>
<td><strong>N/A</strong></td>
<td><strong>N/A</strong></td>
</tr>
</tbody>
</table>
Project Plan – Phase 3 Deliverables

• Final Source Code
• Component Design
• User Manual
• Testing and Assessment Evaluation
• References
• Technical Inspection Letters
• Project Evaluation
## Project Plan – Cost Estimate

<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Cost (hours)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Source Code</td>
<td>30</td>
<td>1500 SLOC / 50 SLOC/hr</td>
</tr>
<tr>
<td>Component Design</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>User Manual</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Testing and Assessment Evaluation</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>References</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Technical Inspection Letters</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Project Evaluation</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>84</strong></td>
<td></td>
</tr>
</tbody>
</table>
Architecture Design - Overview

• 3 Main Components
  – Role Interpreter
  – Demonstration Agent Architecture
  – OMACS and GMoDS interface Code

• Concentrate on Role Interpreter
  – Agent Architecture shown previously
  – Interface code is mostly defined by OMACS and GMoDS
Architecture Design - Components
Architecture Design - Class Diagram
Architecture Design – Role Interpreter

```
RoleLevelGoalModel
- goalTree : GoalTree
+ RoleLevelGoalModel(File, String)
+ event(InstanceGoal, SpecificationEvent) : InstanceTreeChanges
+ getEventsToFire(SpecificationGoal, Object) : Set<SpecificationEvent>
+ getNextInstanceGoal() : InstanceGoal<InstanceParameters>
+ hasActiveInstanceGoals() : boolean
+ reset(InstanceParameters)

RoleInterpreter
- rlgm : RoleLevelGoalModel
+ RoleInterpreter(RoleLevelGoalModel)
+ execute(ExecutionComponent, InstanceGoal)
+ isDone() : boolean
+ isPreemptible(ExecutionComponent) : boolean

GoalCapabilityMap
- map : Map<String, CapabilityMapEntry>
+ addMapping(String, Method, Capability)
~ invoke(String, InstanceParameters) : Object

CapabilityMapEntry
- capability : Capability
- method : Method
+ CapabilityMapEntry(Method, Capability)
+ getCapability() : Capability
+ getMethod()
+ operation()
```
Architecture Design – Role Interpreter

- **Role Interpreter**
  - Acts as Role ExecutionPlan interface
  - Links GoalCapabilityMap and RoleLevelGoalModel

- **GoalCapabilityMap**
  - Maps Capability Methods to InstanceGoal names and parameters
  - Uses Java Reflection API

- **RoleLevelGoalModel**
  - Façade for GMoDS GoalTree (instance and specification trees)
context RoleLevelGoalModel::getEventsToFire(g : ParameterizedSpecificationGoal, r : Boolean) : Set(SpecificationEvent)
-- The given specification goal must exist within the specification tree and be
-- unique
pre GoalInTree:
    goalTree.specTree.goals->select(id = g.id)->size() = 1
-- If the return value is not boolean, then all goal model specified events are
-- returned
post NoBoolReturnsAll:
    r.isUndefined() implies goalTree.specTree.events = result
-- If the return value is true, then all the "true" prefixed events are returned.
-- If no events are prefixed with "true", then all unconditional events are returned.
-- Otherwise, if the return value is false, then all the "false" prefixed events are returned.
-- If no events are prefixed with "false", then all unconditional events are returned.
post CorrectEventsReturned:
    let allEvents    : Set(SpecificationEvent) = goalTree.specTree.events in
    let trueEvents   : Set(SpecificationEvent) = allEvents->select(id.toLower().substring(1, 4) = 'true') in
    let falseEvents  : Set(SpecificationEvent) = allEvents->select(id.toLower().substring(1, 5) = 'false') in
    let unconEvents  : Set(SpecificationEvent) = allEvents - trueEvents - falseEvents in
    if r then
        if trueEvents->isEmpty() then
            result = unconEvents
        else
            result = trueEvents
        endif
    else
        if falseEvents->isEmpty() then
            result = unconEvents
        else
            result = falseEvents
        endif
    endif

context RoleLevelGoalModel::getNextInstanceGoal() : InstanceGoal
-- The returned instance goal is either null, or it is in the set of active
-- goals and it is a leaf goal
post NullOrActiveLeaf:
    result.isUndefined() or
        (goalTree.instTree.activeGoals->includes(result) and
         goalTree.specTree.goals->select(id = result.id and isLeaf)->notEmpty())
GoalCapabilityMap Constraints

countext GoalCapabilityMap::addMapping(s : String, e : CapabilityMapEntry)
-- The given mapping contains no null values
pre NoNulls:
    not s.isDefined() and
    not e.isDefined() and
    not e.method.isDefined() and
    not e.capability.isDefined()

-- The entry has been added to the mapping
post EntryAdded:
    entries = entries@pre->including(e) and
    e.id = s

countext GoalCapabilityMap::invoke(g : ParameterizedSpecificationGoal, p : InstanceParameters)
-- The given goal is not undefined
pre NoNulls:
    not g.isDefined()

-- A mapping is already present in the GCM for this goal
pre MappingExists:
    entries->exists(id = g.id)

-- The specification parameters and instance parameters are null, or
-- There exists an instance parameter for each specification parameter of the goal
pre FormalParamsMatch:
    let specificationParams = g.param.params in
    let instanceParams = p.params in
    ((g.param.isDefined() or specificationParams->isEmpty()) and (p.isDefined() or instanceParams->isEmpty())) or
    specificationParams->forall(fp : SpecificationParameter | instanceParams->exists(ap : InstanceParameter | fp.key = ap.key))
Technical Inspection Checklist

- The symbols used in class diagrams conform to UML 2.0.
- The symbols used in sequence diagrams conform to UML 2.0.
- The symbols used in component diagrams conform to UML 2.0.
- The Component Interface Specification in Section 2.2 clearly defines the major public operations on the system.
- The Analysis Class Diagram in Figure 2 clearly describes the high-level relationships between model elements.
- The High-Level Class Diagram in Figure 3 correctly shows how this system interfaces with the OMACS and GMoDS frameworks.
- The provided descriptions for high-level components are clear and accurate.
- The provided component diagrams are clear and accurate.
- The provided component descriptions are clear and accurate.
- The description of the Role Execution Sequence is accurate and correct.
- The provided USE/OCL model classes, attributes, and relations are consistent with the high-level class model in Figure 3.
- The RoleLevelGoalModel constraints are correctly specified and understandable.
- The GoalCapabilityMap constraints are correctly specified and understandable.
Test Plan

• Functional Black box testing
• Focus on Role Interpreter through testing Demonstration Agent Architecture
• Each part of Role Interpreter must work for Agent Architecture to Work
• 5 Scored WumpiWorld Scenarios, each Testing the same 5 roles
# Test Plan

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Passing Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>HunterKiller.xml</td>
<td>Single HunterKiller agent</td>
<td>Score of at least 19000</td>
</tr>
<tr>
<td>HunterKillerTeam.xml</td>
<td>HunterKiller agent team</td>
<td>Score of at least 23000</td>
</tr>
<tr>
<td>GoldDigger.xml</td>
<td>Single GoldDigger agent</td>
<td>Score of at least -2000</td>
</tr>
<tr>
<td>GoldDiggerHunterKillerTeam.xml</td>
<td>GoldDigger and HunterKiller team</td>
<td>Score of at least 22000</td>
</tr>
<tr>
<td>WumpiWorld.xml</td>
<td>3 HunterKillers, 2 Gold Diggers</td>
<td>Score of at least 25000</td>
</tr>
</tbody>
</table>
Limitations and Future Work

• Agent-Role Assignments hardcoded, No overall Organization to make assignments
  – Simplifies agent architecture for demonstration, No technical limitation

• Can’t handle capability methods that take more than one parameter of the same type
  – Limitation due to the way goal parameters are stored in GMoDS
Limitations and Future Work

• No way to send more than 1 new, previously-unknown parameter
  – What would it come from? We can do 1 using method return value
  – No Type Safety for GMoDS parameters

• Goal-Capability Mapping defined by Capability classes.

• OMACS Interface Capability could use more definition
Architecture Prototype Demo

• Built on previous prototype
• Full WumpiWorld scenario
• All agent actions defined through RoleLevelGoalModels
Questions and Comments