

Xholon – an ABM tool based on “Monkey Topology”

Ken Webb
Primordion

SwarmFest 2007
DePaul University, Computer Science, Chicago
July 12-14, 2007

ken@primordion.com

Abstract

Xholon is an open source multi-paradigm modeling, transformation and simulation tool, in which applications are constructed using XML and Java. It supports the Unified Modeling Language (UML 2.1), systems biology modeling, other types of modeling, and many of the features found in existing agent-based modeling tools. It focuses especially on integration of various approaches. It can, for example, simulate 2D-grid and agent-based models created using UML tools, and can combine multiple agent-based grids in the same model with agent migration between the grids. It combines an optional limited NetLogo-like syntax for navigating and exploring a flat grid space, with the standardized XML-based XPath expression language used for navigating and exploring hierarchical tree spaces. Xholon is based on Object-Oriented and UML concepts of class, composite structure, ports and bindings to connect objects, and active objects (agents) that contain state machines and other behaviors. The central structuring mechanism, what gives Xholon the flexibility it needs to support multiple paradigms, is the tree. This presentation will focus on this hierarchical tree-ness (what could be called "monkey topology"), and how this is used in Xholon to extend the basic concepts of agent based modeling. The presentation will be largely practical in nature, and will include a mini-demo.

Main Points

- ✓ Xholon is based on tree structures rather than grids.
 - ✓ Grids are superimposed over nodes in a tree.
 - ✓ Agents and agent relationships can be defined independently of the type of grid/network.
- ✓ Xholon combines an XPath syntax for navigating trees, and an optional XLogo syntax for navigating grids.
- ✓ The focus on trees provides flexibility, without having to give up ABM concepts of grids.
 - ✓ Navigation between multiple grids.
 - ✓ Complex internal structure for agents.

Background

- ◆ ObjecTime, IBM Rational Rose RealTime commercial products.
- ◆ Jointly published several academic papers.
 - ◆ These models are inaccessible to other researchers.
 - ◆ And, I started using these tools in new ways.
- ◆ I started to create my own modeling tool, based on Unified Modeling Language version 2.x (UML 2.x), and built using Java and XML.

Xholon

- A research project and software development tool that executes models of systems.
 - Including event-driven applications
 - Multi paradigm
 - Systems can be of arbitrary size
 - Embedded systems, controllers; Agent-based, swarms, etc.
- Goal of Xholon
 - To be able to model and execute a broad range of event-driven and complex systems, using same basic constructs in all of them.

Core Concept - Trees

- ◆ Everything in Xholon is a node in a tree.
- ◆ Tree nodes can cross connect with each other, using UML ports and connectors.
 - ◆ Networks, graphs, grids are overlaid on top of the primary tree structure.
- ◆ Any node in the tree can be an active object or agent, and can navigate the tree to interact with any other object.
 - ◆ Can move to a new part of the tree, can create new nodes and subtrees, can move/delete other nodes, can act on other passive nodes, etc.

Examples – Why trees are important

- UML state machines as tree structures.
- Genetic programming for tree manipulation.
- XML as a standard for presenting trees in text.
- Biological systems as one system inside another.
- Other data structures, such as grids, can be overlaid on top of trees.
- Agents can readily navigate to any other node.

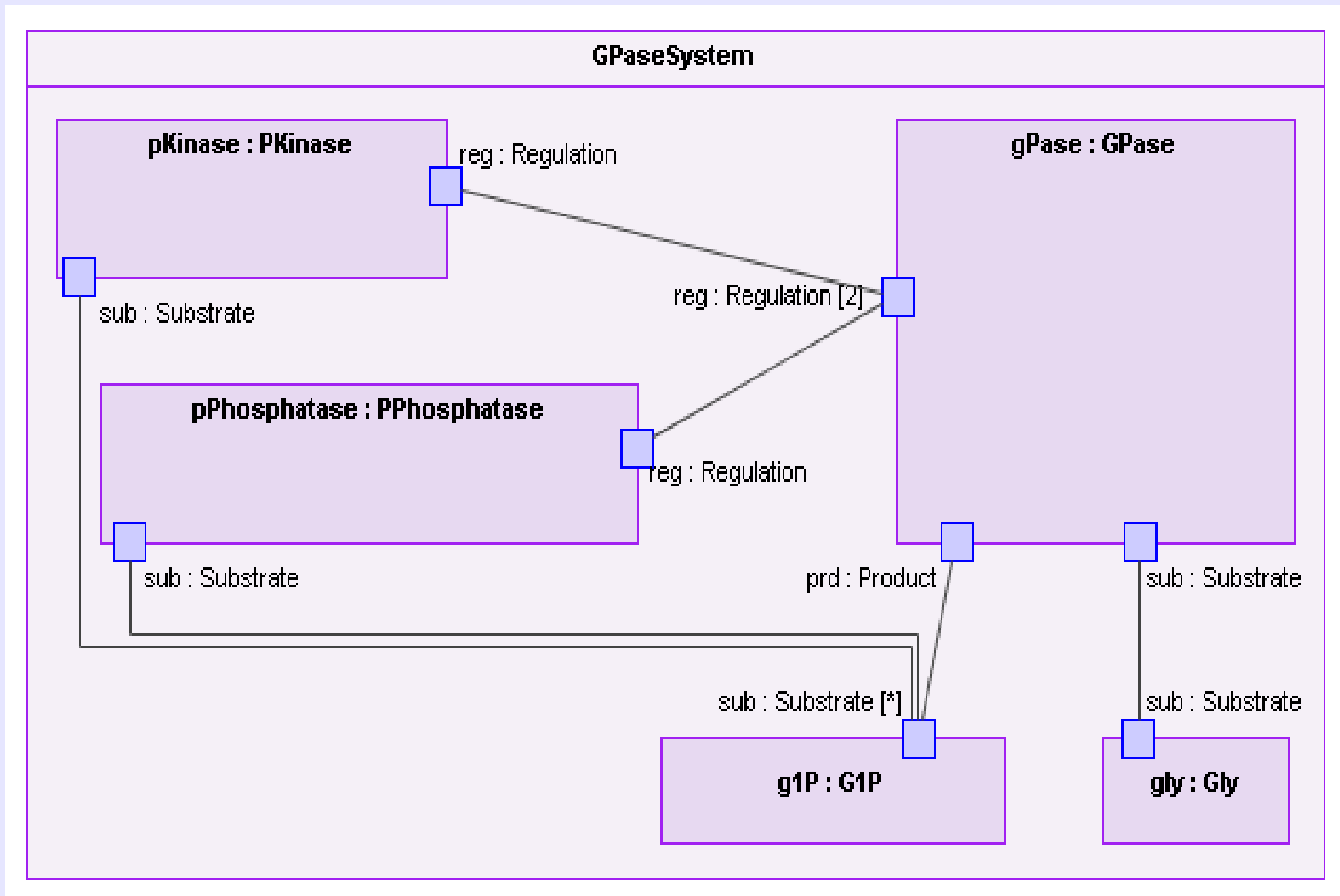
Xholon Modeling Constructs

- The basic Xholon modeling constructs are aligned with UML 2 constructs.
 - UML is a good starting point.
 - Topcased (open source), MagicDraw, Poseidon
- These constructs include - classes, composite structure, parts, ports, connectors, state machines.
- Active objects are agents, each with its own independent behavior.

Some Xholon ABM Features

- Moore, Von Neumann, Hex 2D neighborhoods.
 - Toroidal or non-wrapping grid.
- Line charts, histograms, view individual agents.
- Start, stop, pause/unpause, step, refresh.
- Colt probability distributions.
- Multiple grids in same model.
- Agents can be arbitrarily complex.
 - Can include state machines, neural networks, etc.
- Incorporates ECJ to evolve agent behaviors.
- Implements all 16 “stupid models”.

Composite Structure, Ports



This defines a tree structure, with lateral connections between nodes in the tree.

XPath

- x An expression language for navigating XML documents.
- x A W3C standard (<http://www.w3.org/TR/xpath>).
- x Models an XML document as a tree of nodes.
- x Xholon extensively uses a subset of XPath 1.0
 - x To allow nodes to locate other nodes that match certain conditions.
- x An expression can return a single node, or a node set.
- x An XPath expression is an ordered set of instructions that a monkey could follow to navigate from its current location (the context node) to any other part of the tree.

Common XPath Axes

x ancestor

x descendant

x following

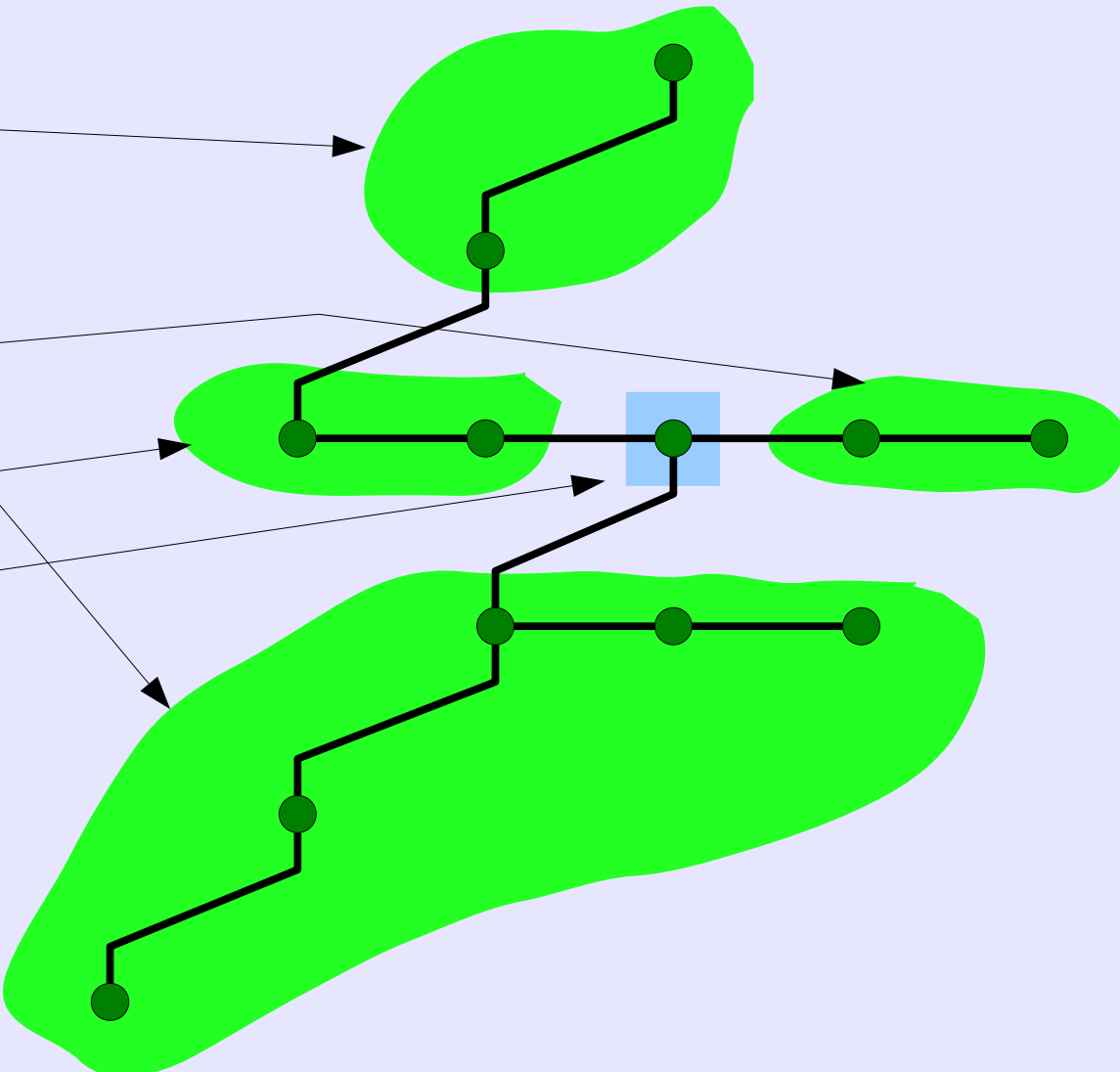
x preceding

x self

x child

x parent

x attribute



XPath expressions in Xholon

ancestor::TheSystem/PatchOwner

ancestor::GPaseSystem/Atp[@roleName='atp1']

ancestor::Grid/../../Statistics

ancestor::Elevator/ElevatorPanel/DoorControlButton[@roleName='bOpen']/attribute::port[2]/attribute::replication[1..*]

.[@uid='121219297']/ancestor::StateMachine/descendant::Transition[@uid='968519337']

NetLogo or StarLogo Syntax

- ◆ A sequence of logo commands is an ordered set of instructions that a turtle could use to navigate from its current location to any other part of the grid.
- ◆ One simple example:

forward 10

right 45

backward 5

left 90

XPath + XLogo

- ◆ Xholon combines the XPath and XLogo (NetLogo or StarLogo) syntaxes.
- ◆ Turtle Geometry
 - ◆ When an agent is moving within a grid, it can use the XLogo-like syntax.
- ◆ Monkey Topology
 - ◆ When an agent is moving through a tree, it can use the XPath-like syntax.

Standard combined syntax ?

- Could there be a combined standardized syntax for agent navigation through any type of space?
- XPath is already a standardized syntax for navigating tree structures.
- How exactly could the XLogo syntaxes fit into the XPath approach?
- Are there any efforts underway to harmonize the NetLogo and StarLogo syntaxes, and what similar syntaxes are other ABM tools using?
- Could there be a formal ABM syntax?

Demo of Xholon

MagicDraw UML Modeling Tool

MagicDraw UML 12.5 - Rcs_GP_FSM_Grid.mdzip [C:\Xholon\transform\MagicDraw\]

File Edit View Layout Diagrams Options Tools Analyze Teamwork Window Help

C:\Xh...raw\Rcs_GP_FSM_Grid.m...

RcsClasses RcsPortsSignalsInterface... GPSystem_CompositeStruct... GPase_Fsm PhosphorylaseKinase_Fsm

Containment

- Rcs_GP_FSM_Grid
 - AggregatorClasses
 - Configuration
 - GridClasses
 - RcsClasses
 - Relations
 - Aggregate
 - Application
 - Enzyme
 - G1P
 - Glc
 - Gly
 - GPase
 - GPaseSystem
 - GPaseSystemWithGrid

Common

- Note
- Text Box
- Anchor
- Containment
- Dependency
- Image Shape
- Separator
- Class Diagram
- Class
- Interface
- Package
- Generalization
- Association
- Aggregation
- Composition
- Interface Realization
- Usage
- Abstraction
- Instance
- Link
- Use Case Diagram
- Implementation Diagram
- Composite Structure Di...
- Information Flows
- Profiling Mechanism

Zoom Docume... Properties

Element Language properties

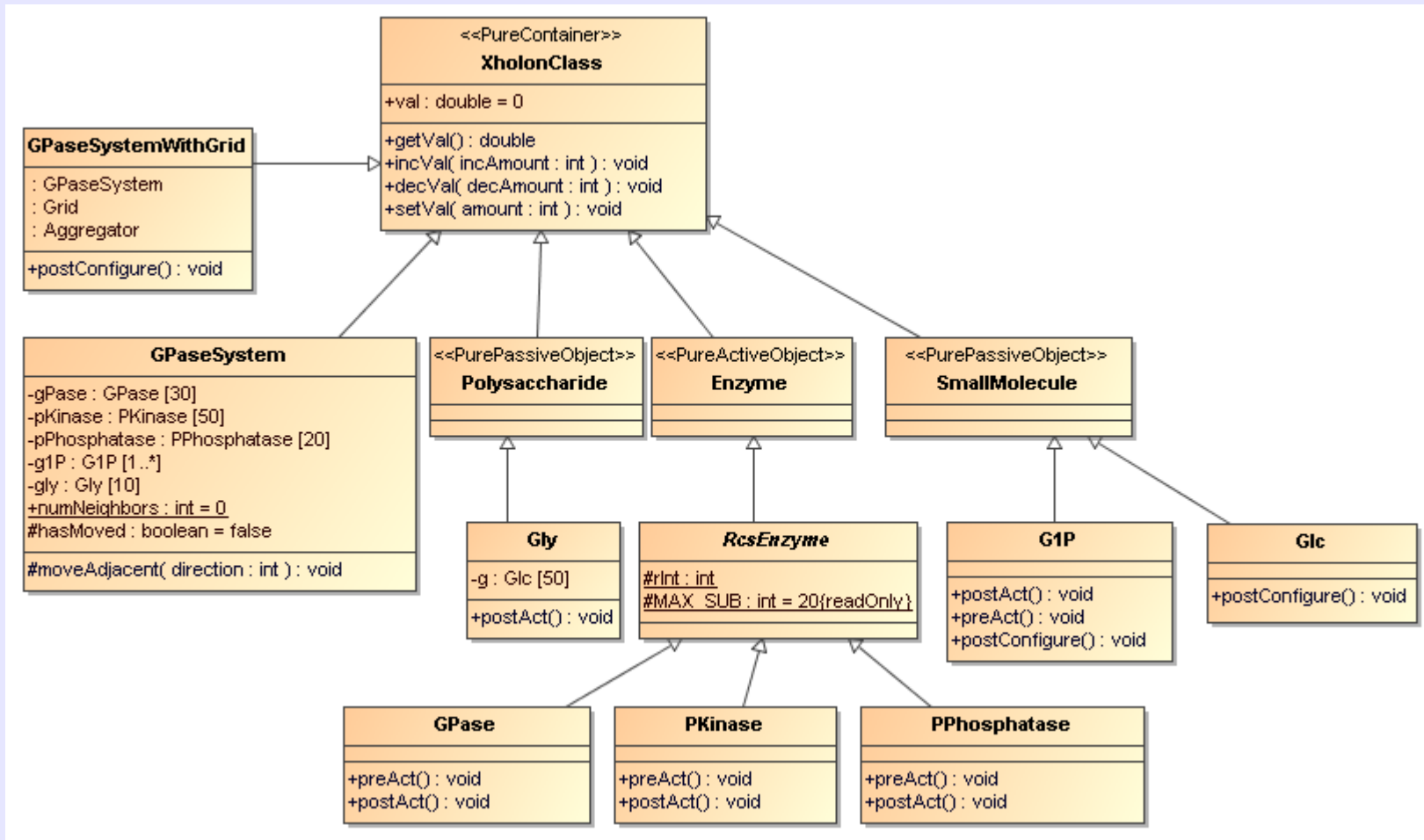
Class

Name	GPaseSystemWith...
Owner	RcsClasses
Applied Stereo...	
Base Classifier	XholonClass
Realized Interf...	
Visibility	public
Is Active	<input type="checkbox"/> false
Is Abstract	<input type="checkbox"/> false
To Do	

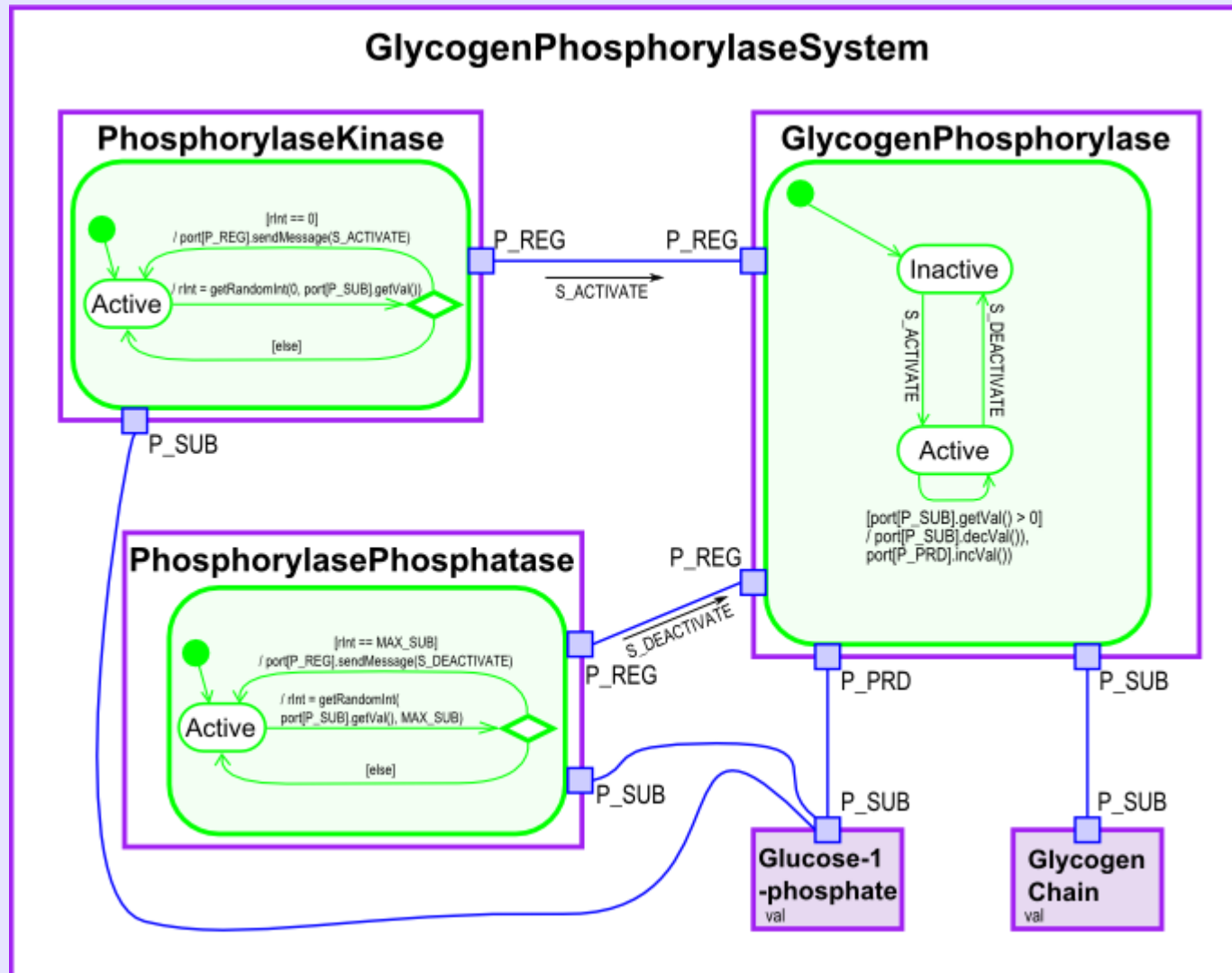
```

classDiagram
    class XholonClass {
        <<PureContainer>>
        +val : double = 0
        +getVal() : double
        +incVal(incAmount : int) : void
        +decVal(decAmount : int) : void
        +setVal(amount : int) : void
    }
    class GPaseSystemWithGrid {
        : GPaseSystem
        : Grid
        : Aggregator
        +postConfigure() : void
    }
    class GPaseSystem {
        -gPase : GPase [30]
        -pKinase : PKinase [50]
        -pPhosphatase : PPhosphatase [20]
        -g1P : G1P [1..*]
        -gly : Gly [10]
        +numNeighbors : int = 0
        #hasMoved : boolean = false
        #moveAdjacent(direction : int) : void
    }
    class Polysaccharide {
        <<PurePassiveObject>>
    }
    class Enzyme {
        <<PureActiveObject>>
    }
    class SmallMolecule {
        <<PurePassiveObject>>
    }
    class Gly {
        -g : Glc [50]
        +postAct() : void
    }
    class RcsEnzyme {
        #rInt : int
        #MAX_SUB : int = 20{readOnly}
        +postAct() : void
        +preAct() : void
        +postConfigure() : void
    }
    class GPase {
        +preAct() : void
        +postAct() : void
    }
    class PKinase {
        +preAct() : void
        +postAct() : void
    }
    class PPhosphatase {
        +preAct() : void
        +postAct() : void
    }
    XholonClass <|-- GPaseSystemWithGrid
    GPaseSystemWithGrid *-- GPaseSystem
    GPaseSystem <|-- Polysaccharide
    GPaseSystem <|-- Enzyme
    GPaseSystem <|-- SmallMolecule
    Polysaccharide <|-- Gly
    Enzyme <|-- RcsEnzyme
    RcsEnzyme <|-- GPase
    RcsEnzyme <|-- PKinase
    RcsEnzyme <|-- PPhosphatase
    
```

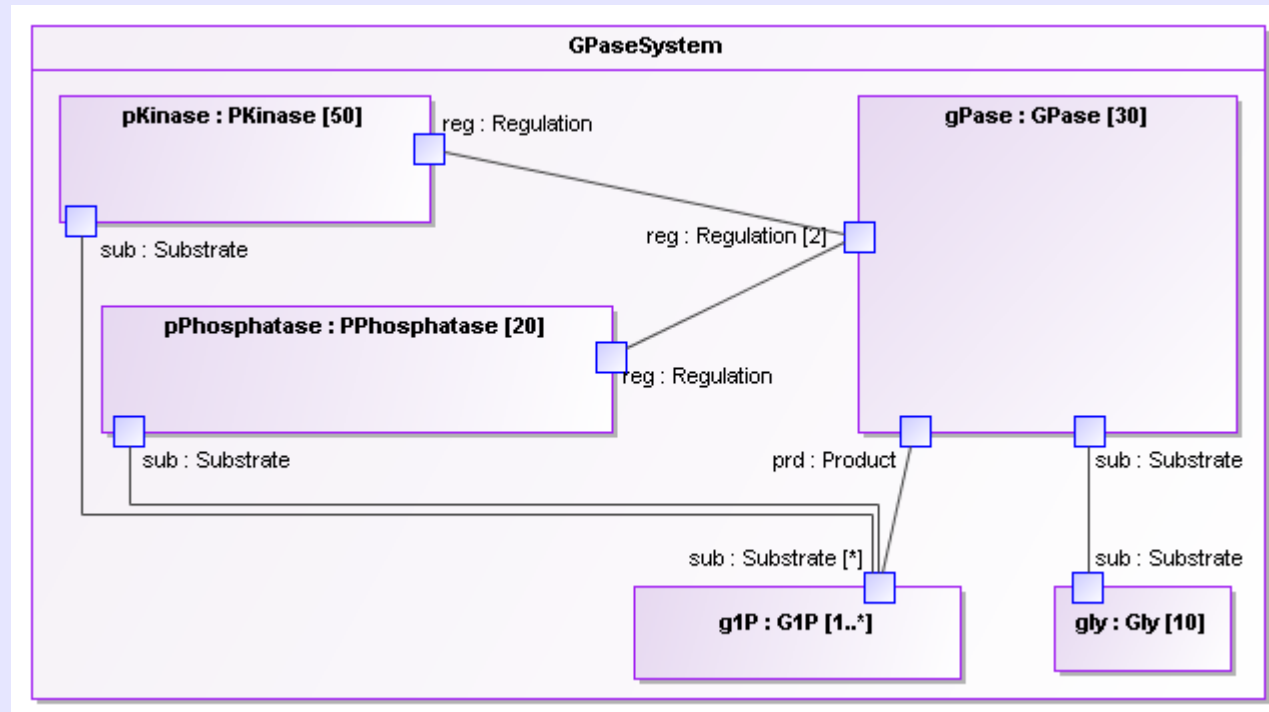
RCS – UML - Agent Classes



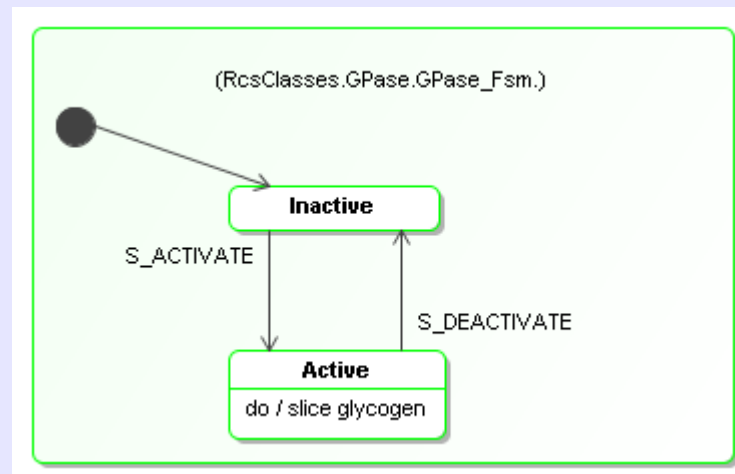
RCS – Agent Architecture



RCS – UML - Composite Structure

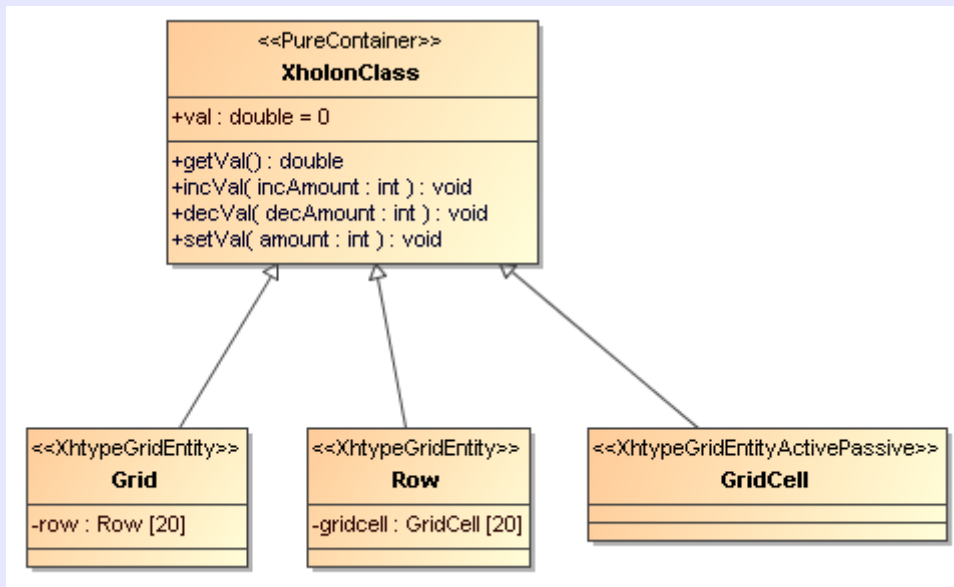


RCS – UML - State Machine

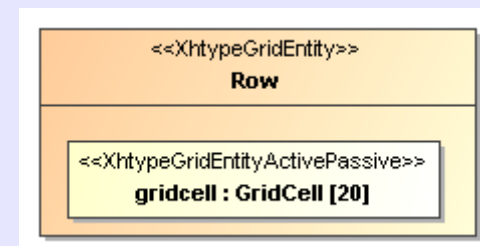
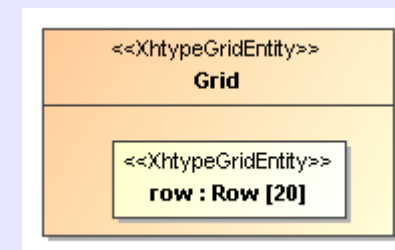
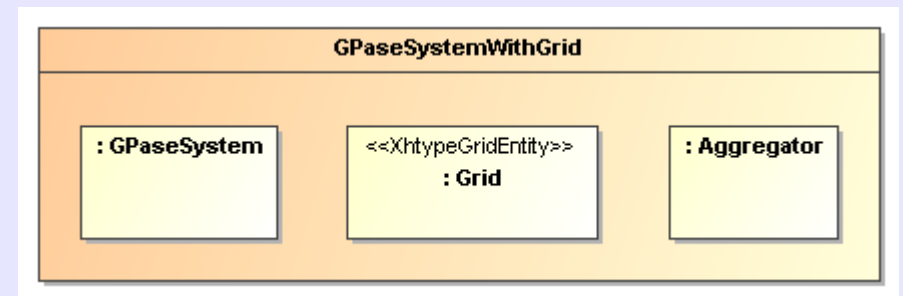


RCS – UML - Grid Classes

Class inheritance hierarchy



Composite Structure



Model Driven Software Development

The screenshot displays the Eclipse IDE interface. The main editor window shows an XML file named `Rcs_GP_FSM_Grid_CompositeStructureHierarchy.xml`. The XML content is as follows:

```
</Region>
</StateMachine>
</PKinase>
<PPhosphatase multiplicity="20" roleName="pPhosphatase">
  <StateMachine>
    <Region>
      <State uid="649">
        <Region>
          <Pseudocycle>
            <State>
              <Doc>
            </State>
          </Pseudocycle>
          <Transitions>
        </Region>
      </State>
    </Region>
  </StateMachine>
</PPhosphatase>
<G1P multiplicity="1" roleName="G1P">
<Gly multiplicity="10" roleName="Gly">
  <Glc multiplicity="50" roleName="Glc">
</Gly>
</GPaseSystem>
<Grid roleName="">
  <Row multiplicity="20" roleName="row">
    <GridCell multiplicity="20" roleName="gridcell"/>
  </Row>
</Grid>
<Aggregator roleName="">
  <Aggregator_Glc roleName=""/>
  <Aggregator_G1P roleName=""/>
</Aggregator>
</GPaseSystemWithGrid>
```

An overlaid terminal window titled "Xholon" shows the following output:

```
C:\Xholon\transform\MagicDraw>xmi2xh all Rcs_GP_FSM_Grid GPaseSystemWithGrid
Unzipping Rcs_GP_FSM_Grid.mdzip ...
Extracting: Rcs_GP_FSM_Grid.mdxml
creating InheritanceHierarchy XML file ...
creating ClassEnum Java file ...
creating CompositeStructureHierarchy XML file ...
creating ClassDetails XML file ...
creating Xh Java file ...
creating App Java file ...
creating Information XML file ...
creating _xhn XML file ...
C:\Xholon\transform\MagicDraw>
```


RCS – Execution in Xholon

The screenshot displays the Xholon GUI interface during the execution of RCS. The main window, titled "Xholon", shows a hierarchical tree structure of components. The root is "CompositeStructureHierarchy", which contains "gPaseSystemWithGrid_0". Under "gPaseSystemWithGrid_0", there is a sub-component ":gPaseSystem_1". Below this, there are three folders: "grid_1563", "row_1564", and "row_1585". Under "grid_1563", there are several "gridCell" components, including "gridCell_1607" through "gridCell_1622". The "gPase:gPase_58" component is highlighted in the tree.

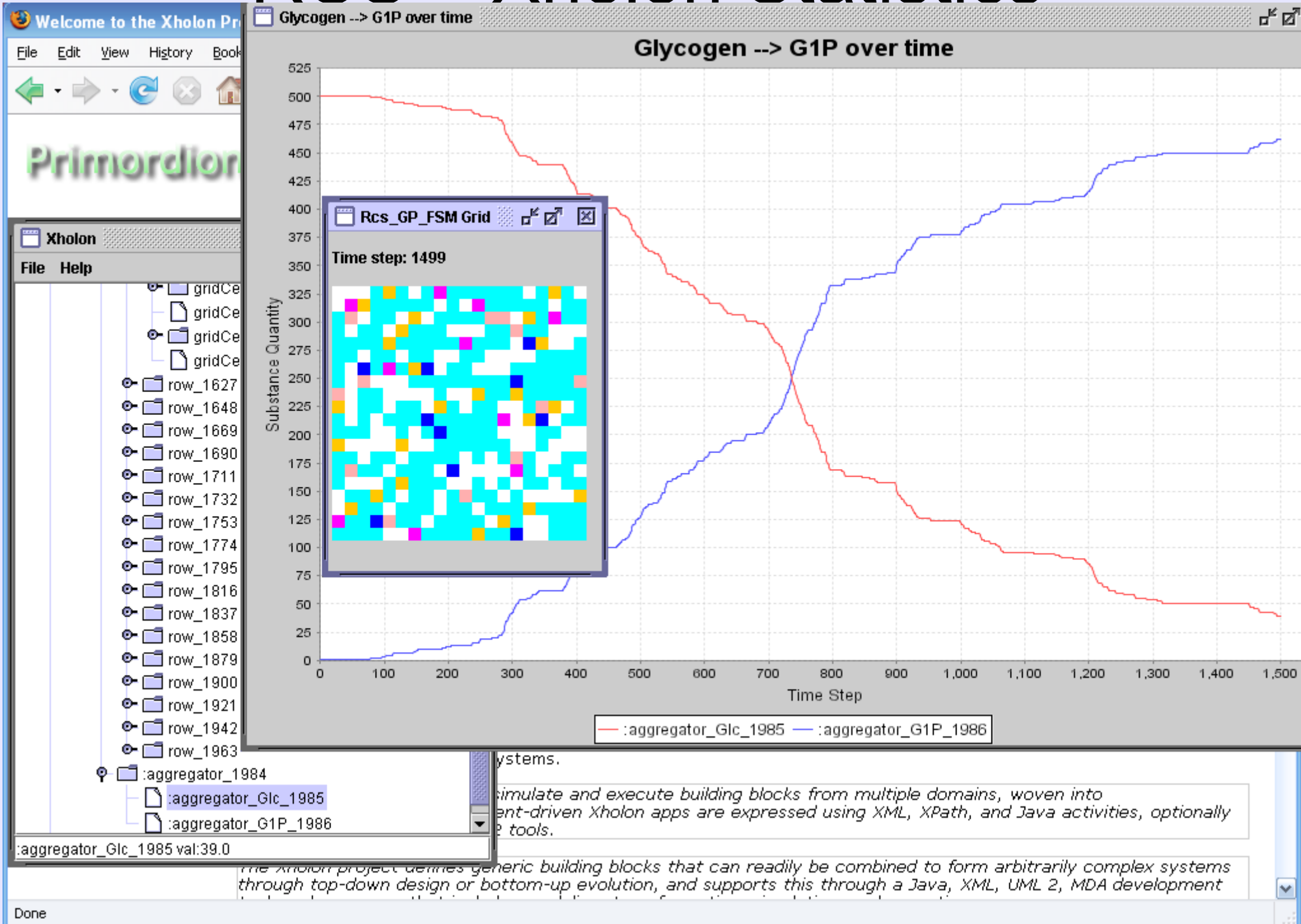
A smaller window titled "Rcs_GP_FSM Grid" is overlaid on the main window, showing a colorful grid pattern. The grid consists of various colored squares (yellow, pink, blue, cyan) arranged in a complex, non-linear pattern.

The status bar at the bottom of the Xholon window displays the text: "gPase:gPase_58 [port:gly:gly_1053 port:g1P:g1P_1052]".

The "xhnGUI" window in the background shows the following configuration parameters:

```
SizeMessageQueue : 20
InheritanceHierarchyFile : ./c
CompositeStructureHierarchyFil
ClassDetailsFile : ./config/xm
InformationFile : ./config/xmi
JavaClassName : org.primordion
JavaXhClassName : org.primordi
MaxPorts : 2
UseDataPlotter : JFreeChart
DataPlotterParams : Glycogen
UseGraphicalTreeViewer : false
GraphicalTreeViewerParams : ./
UseGraphicalNetworkViewer : fa
GraphicalNetworkViewerParams :
UseGridViewer :
GridPanelClassName :
GridViewerParams :
UseInteractions :
UseVrml : false
VrmlWriterClass :
VrmlParams : tr
UseTextTree : f
SaveSnapshots :
SnapshotParams :
```

RCS – Xholon Statistics



Stupid Model – 3 Grids

The image displays the Xholon IDE interface with a project structure on the left and three simulation windows on the right.

Project Structure (Left Panel):

- Application
 - Controller
 - View
 - gridViewer_30522
 - gridViewer_30523
 - gridViewer_30524
 - Model
 - CompositeStructureHierarchy
 - stupidModel_0
 - population_1
 - bugs_2
 - grid_4
 - population_10105
 - bugs_10106
 - grid_10207
 - population_20308
 - bugs_20309
 - grid_20410
 - row_20411
 - habitatCell_20412
 - habitatCell_20413
 - habitatCell_20414
 - habitatCell_20415
 - habitatCell_20416
 - habitatCell_20417
 - habitatCell_20418
 - habitatCell_20419
 - habitatCell_20420
 - habitatCell_20421
 - habitatCell_20422
 - habitatCell_20423
 - bug_20390
 - habitatCell_20424
 - habitatCell_20425

Simulation Windows (Right Panel):

- Xholon - SM5tg Population 1:** Time step: 214. Shows a large grid of white stars on a black background.
- Xholon - SM5tg Pop 2:** Time step: 214. Shows a smaller grid of orange dots on a black background.
- Xholon - SM5tg Pop 3:** Time step: 214. Shows a smaller grid of orange dots on a black background.

Text Output (Middle Panel):

```
le l/StupidModel  
upidModel/Stup  
dMode15tg/Stup  
lMode15tg/Stupi  
ls.StupidModel  
rials.StupidMod  
  
torials.Stupid  
/...5,Xholon -  
Row/...2,Xholo  
Row/...2,Xholo  
  
another one.  
  
le l/StupidModel  
upidModel/Stup  
dMode15tg/Stup  
lMode15tg/Stupi  
ls.StupidModel  
rials.StupidMod  
  
torials.Stupid  
/...5,Xholon -  
Row/...2,Xholo  
Row/...2,Xholo
```

Status Bar (Bottom Left): bug_20390 bugSize:79.94415020540632

Wolf, Sheep, Grass originally in NetLogo, also in Repast

The image displays a NetLogo environment running the Wolf Sheep Grass simulation. The interface is divided into several sections:

- Code Editor (Left):** Shows Java code for the simulation. The visible code includes:

```
168 /**
169  * Move one unit in a randomized heading
170  * (Sheep and Wolf).
171  */
172 protected void move()
173 {
174     //to move :: turtle procedure
175     // rt random-float 50 - random-float 50
176     // fd 1
177     //end
178     rt(Misc.getRandomDouble(0.
179     fd(1);
180 }
181
182 /**
183  * Eat grass
184  * (Sheep).
185  */
186 protected void eatGrass()
187 {
188     //to eat-grass :: sheep procedure
189     // :: sheep eat grass, turn the patch brown
190     // if pcolor = green [
191     //   set pcolor brown
192     //   set energy energy + sheep-gain-from-food :: sheep gain energy
193     // ]
194     //end
195     if (getPcolor() == ITurtlePatchColor.TPCOLOR
196         setPcolor(ITurtlePatchColor.TPCOLOR_BROWN);
197         ((IPatch)parent).aggregate(-0.25);
198         energy += sheepGainFromFood;
199     }
200 }
```
- Grid View (Center):** A 51x51 grid representing the world. Green patches represent grass, white patches represent sheep, and black patches represent wolves. The time step is 672.
- Control Panel (Right):** Contains a 'Wolf Sheep Grass Fluctuations' chart showing population over time (timesteps). The chart has three lines: a red line for 'Wolf:aggregator_2655', a blue line for 'Sheep:aggregator_2656', and a green line for 'Grass:aggregator_2657'. The y-axis is 'Population' (0 to 300) and the x-axis is 'Time (timesteps)' (0 to 700). Below the chart is a tree view showing the system hierarchy, including 'Application', 'Controller', 'View', and 'Model'.

NetLogo-like Syntax in Xholon

```
/**
 * Eat grass
 * (Sheep).
 */
protected void eatGrass()
{
    //to eat-grass ;; sheep procedure
    // ;; sheep eat grass, turn the patch brown
    // if pcolor = green [
    //   set pcolor brown
    //   set energy energy + sheep-gain-from-food ;; sheep gain energy by eating
    // ]
    //end
    if (getPcolor() == ITurtlePatchColor.TPCOLOR_GREEN) { // Xholon Java
        setPcolor(ITurtlePatchColor.TPCOLOR_BROWN);
        ((IPatch)parent).aggregate(-0.25);
        energy += sheepGainFromFood;
    }
}
```

NetLogo

Questions ?

<http://www.primordion.com/Xholon/>

Contact Ken: ken@primordion.com