Кожен агент працює з своїм екземпляром мережі, яка знаходиться у пам’яті агента. Структура даних мережі у всіх агентів однакова. Різними є тільки процедури обробки даних.

Усі агенти передають один-одному фрагменти мережі (або всю мережу) у однаковому форматі. Формат опису мережі для обміну наведено нижче:

<Nodes > - описує вузли мережі

<Linkgroups> - описує шари мережі

<Edges> - описує зв’язки між вузлами та приналежність вузлів і зв’язків деякому шару

<Receptorlist> - описує перелік рецепторів для кожного концептора.

<ControlVertex> - описує контрольних вузлів для кожного класу обєктів

Значення guid генерується за допомогою генератора випадкових чисел у Windows та для всіх обєктів крім самого графу береться тільки остання група цифр, наприклад: "278CBBF8BCA5".

Приклад мережі:

<?xml version = "1.0" encoding = "**Windows-1251**"?>

<Graph nodesize="10" vspacing="3" hspacing="20" padding="4" showarrows="true" guid="E388B302-8923-2827-EDFC-278CB9AA095D" searchurl="">

<datagroups>

</datagroups>

<Nodes>

<Node guid="278CBBE71A83" nodeName="Class\_1" nclass="Class\_1" shape="square" color="80" xPos="" yPos="" font="Verdana" fontsize="10"/>

<Node guid="" nodeName="OB1" nclass="" shape="square" color="13421772" xPos="" yPos="" font="Verdana" fontsize="10"/>

<Node guid="" nodeName="OB2" nclass="" shape="square" color="13421772" xPos="" yPos="" font="Verdana" fontsize="10"/>

<Node guid="" nodeName="A1" nclass="" shape="square" color="13421772" xPos="" yPos="" font="Verdana" fontsize="10"/>

<Node guid="" nodeName="A2" nclass="" shape="square" color="13421772" xPos="" yPos="" font="Verdana" fontsize="10"/>

</Nodes>

<Linkgroups>

<Group name="Входить до" color="10066329"/>

<Group name="Class\_1" color="10066335"/>

</Linkgroups>

<Edges>

<Edge guid="" edgeName="link\_1" node1="A1" node2=" OB1" group="Входить до" istwoway="false"/>

<Edge guid="" edgeName="link\_2" node1="A2" node2=" OB2" group="Входить до" istwoway="false"/>

<Edge guid="" edgeName="link\_3" node1="A1" node2=" OB2" group="Входить до" istwoway="false"/>

<Edge guid="" edgeName="link\_4" node1="OB1" node2="Class\_1" group="Class\_1" istwoway="false"/>

<Edge guid="" edgeName="link\_5" node1="OB2" node2="Class\_1" group="Class\_1" istwoway="false"/>

</Edges>

<Receptorlist>

<Conceptor node1="OB1" node2="A1"/>

<Conceptor node1="OB2" node2="A1"/>

<Conceptor node1="OB2" node2="A2"/>

</Receptorlist>

<ControlVertex>

<Class name ="Class\_1">

<Node nodeName="OB1"/>

<Node nodeName="A1"/>

</Class>

</ControlVertex>

</Graph>

Опис класів носить характер рекомендацій і потрібен для пояснення роботи алгоритму.

**Node** — базовий клас для вузлів пірамідальної мережі.

Для кожного вузла **Node** задаються наступні властивості:

**name: String;**

**node\_index: int –** розраховується за допомогою **setОbjectIndex** з **name**

**receptorSet: set of Node\_index. (**список рецепторів у вигляді множини set. Формується на основі прямої вказівки в вихідних даних. При виконанні процедур побудови мережі не змінюється. Для новостворених вузлів заповнюється 1 раз створенні вузла. Для рецепторів не потрібен**)**

**receptorList: list of Node\*.** (Формується аналогічно Receptor\_set. Потрібен для визначення порядку на множині рецепторів та реалізації різних варіантів алгоритму побудови мережі. Для рецепторів не потрібен)

**conceptorList: list of Node\*.** (Формується виключно для рецепторів. Містить перелік об’єктів з якими зв’язаний рецептор).

Функції **getNode(Name, set)** та **getNode(Node\_index, set)** повинні повертати вказівникна **Node** за допомогою пошуку за **node\_index** у множині **set**.

**downNodeSet: set of node\_index.** Множина Node з субмножини поточного вузла, з якими поточний вузол безпосередньо з’єднано.

**upperNodeSet: set of node\_index.** Множина Node з супермножини поточного вузла, з якими поточний вузол безпосередньо з’єднано.

**nodeType:int (0 – Receptor;1 – Conceptor; 2 – Class)** Під словом Class розуміється вузол, який відповідає імені шару мережі. Тут є дублювання. Шар використовується для завдання приналежності до нього концепторів та рецепторів, а також як окремий вузол до якого належить визначена множина концепторів мережі.

**nodeLevel: int (**спочатку дорівнює кількості рецепторівв множині Receptor\_list. Після закінчення зчитування мережі з вихідних даних, якщо якийсь з рецепторів став концептором, то потрібно до Receptorlist.Count додати кількість рецепторів нового концептора. Receptorlist та Receptorset при цьому не змінюються)

**nodeLayer: set of Nodelayer**

**ОbjectIndex** — шаблонний клас, буде виконувати індексацію об’єктів , тобто кожному об’єкту ставитиме у відповідність число і навпаки. Має set**ОbjectIndex(string name)** У майбутньому це дозволить зекономити операції порівняння для складних типів (текстових рядків, об’єктів, масивів та їх комбінацій). Для довільних типів це можна зробити або за допомогою шаблонних класів або наслідування.

**NodeLayer** – клас для шарів пірамідальної мережі.

клас **NodeLayer** має властивості:

**name:** String;

**layerIndex:** Int;

**layerNodeSet**: set of node\_index– множина вузлів мережі, які належать до визначеного шара, як концепторів так і рецепторів. Може для цього підійде тип list.

**Net —** пірамідальна мережа.

У мережі повинні бути наступні властивості:

**nodeSet** – загальна множина всіх (крім вузлів-класів) вузлів мережіt

**classNodeSet** – множина вузлів-класів мережі

**layerSet** – множина вузлів-шарів мережі

**LewelNodeList**(i) – множина вузлів за рівнями(визначається на основі кількості рецепторів вузла)

Функція **getNodeName(node\_index: Int): name:string.**

Операції при побудові мережі з XML файлу:

1) створити об’єкт класу пірамідальна мережа **– Net;**

2) додати до мережі об’єкти класу Node з вихідних даних.

3) після зчитування даних запускаються процедури обробки інформації.