УДК 004.4`2:004.94

МУЛЬТИАГЕНТНЫЕ МОДЕЛИ НА ОСНОВЕ НЕЧЕТКОЙ ЛОГИКИ ВЫСШЕГО ТИПА ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНОЙ СРЕДЫ

И.Н. Парасюк, С.В. Ершов

Институт кибернетики им. В.М. Глушкова НАН Украины,
03680, Киев-187, проспект Академика Глушкова, 40, тел.: 526 6422, e-mail: ivpar1@i.com.ua

Рассмотрены модели интеллектуальных агентов и мультиагентных систем на основе нечеткой логики высшего типа, позволяющие более информативно представить степень неопределенности системы нечетких правил при спецификации поведения таких агентов и систем. Описан моделе-ориентированный подход для порождения платформно-специфических моделей нечетких мультиагентных систем, функционирующих в перспективных высокопроизводительных средах, в том числе в кластерной системе СКИТ. Предложена архитектура мультиагентной системы для достижения задаваемых средствами нечеткой логики высшего типа таких аспектов поведения, как поддержание дистанции между агентами, согласование скоростей и, соответственно, обход препятствий.

Models of intelligent agents and multiagent systems based on fuzzy logic of higher type, that allows to represent more informatively an uncertainty degree of linguistic representation of fuzzy rules system in the specification of behavior of these agents are considered. A model-driven approach to generate platform-specific models of fuzzy multiagent systems that operate in advanced high-performance environments, including cluster systems SKIT, is described. Multi-agent system architecture to achieve such aspects of behavior, specified by means of higher type fuzzy logic, as maintaining a distance between the agents, speed coordination and, respectively, obstacle avoidance is proposed.

Мультиагентные системы представляют собой один из наиболее сложных видов программного обеспечения. Их работка усложнена отсутствием единого набора моделей, в рамках которых описывается функционирование агентов, а также отсутствием общепринятых "стандартных" платформ и языков реализации таких систем. Агентно-ориентированное моделирование процессов в наиболее сложных системах (социальных, экономических и других) требует одновременного выполнения до 10^{13} программ-агентов [1]. Кроме того, программирование интеллектуальных агентов для современных высокопроизводительных систем существенно усложняется необходимостью учитывать не только функциональные требования, основывающиеся на параллельных алгоритмах принятия решений агентами, но и специфические детали программного обеспечения технологической платформы параллельного выполнения и коммуникации в вычислительной среде (Message Passing Interface (MPI), Common Unified Data Architecture (CUDA) и другие).

С другой стороны, большое внимание уделяется методам представления знаний с помощью нечеткой логики высшего типа. Как известно [2, 3], средствами нечеткой логики можно построить методологии для вычислений со словами – основы представления знаний и моделирования рассуждений в интеллектуальных системах. Хорошо различимые границы “информационных гранул”, представляемых лингвистическими термами, могут быть описаны нечеткими функциями принадлежности. Это – основа теории нечетких множеств типа 1. Однако, когда границы плохо различимы, например, выражены некоторыми зонами, их можно выражать нечеткими множествами типа 2 и выше. У таких множеств отдельные значения принадлежности задаются функциями принадлежности, т. е. учитывается неопределенность определения принадлежности. Чаще всего, имеется ввиду лингвистическая неопределенность, связанная с различными оттенками смысла слов в моделях знаний, основанных на правилах [2]. Нечеткими множествами типа 2 можно характеризовать степени неопределенности, связанные с принятием решений в интеллектуальных системах, в том числе мультиагентных. Основные понятия, характеризующие нечеткую логику высшего типа, то есть функции и степени принадлежности второго типа, а также понятие следа неопределенности описаны [4, 5].

Уместно отметить, что создание агентов, использующих преимущества нечеткой логики такого типа, требует разработки архитектурных моделей нечетких агентов, моделей коллективного поведения группы агентов и методов генерации платформно-специфических моделей таких агентов для перспективных высокопроизводительных платформ. Это важно, например, при решении таких прикладных задач, как исследование эффективности процессов эвакуации людей в чрезвычайных ситуациях, поведения транспортных средств в условиях неопределенности (неполной информации) и ряда других процессов.

Цель работы – исследование и обоснование модели интеллектуальных агентов и мультиагентных систем с использованием нечеткой логики высшего типа, что позволит более информативно представить степень неопределенности лингвистического представления фактов и знаний (убеждений) при моделировании структуры и группового поведения интеллектуальных агентов в нечеткой среде [6–9].

Модель интеллектуального агента, основанная на нечеткой логике высшего типа

Основу функционирования интеллектуального агента, основанного на нечеткой логике типа 2, составляет система правил нечеткого вывода ЕСЛИ-ТО, но ее множества антецедентов или консеквентов этих правил – нечеткие множества (или нечеткие числа) типа 2. Основными компонентами архитектурной модели такого нечеткого агента являются следующие: фузификатор, система правил, средства нечеткого вывода и

процессор выходных значений. Процессор выходных значений состоит, соответственно из редуктора типа – создающего на выходе нечеткое множество типа 1 и дефузификатора – генерирующего соответствующего ему четкого значения.

Нечеткое множество типа 2 задается функцией принадлежности $\widetilde{A} = \left\{\left((x,u),\mu_{\widetilde{A}}(x,u)\right)\middle|\forall x \in X,\ \forall u \in J_x\right\}$, где $0 \le \mu_{\widetilde{A}}(x,u) \le 1$. При этом $J_x \in [0,1]$ представляет первичную степень принадлежности элемента *x*, а $\mu_{\widetilde{A}}(x,u)$ – нечеткое множество (типа 1) вторичной принадлежности.

Компонент "Фузификатор" отображает каждое входное значение $\mathbf{x} = (x_1,...,x_p) \in X_1 \times X_2 \times ... \times X_p$ в нечеткое множество $\widetilde{A}_x$ на *X*. Один из простейших методов фузификации – построение множества-синглетона. $\widetilde{A}_x$ представляет собой нечеткий синглетон 2-го типа, если $\mu_{\widetilde{A}_x}(x) = 1/1$ для всех $\mathbf{x} = \mathbf{x}'$ и $\mu_{\widetilde{A}_x}(x) = 1/0$ для всех остальных $\mathbf{x} \ne \mathbf{x}'$.

Таким образом, для агента с *p* входами $x_1 \in X_1$, ... , $x_p \in X_p$ и одним выходом $y \in Y$, если предположить, что всего имеется *M* правил, *i*-е правило агента можно представить следующим образом:

$$R^i: \quad \text{ЕСЛИ } x_1 = \widetilde{F}_1^i \text{ И } ... \text{ И } x_p = \widetilde{F}_p^i \quad \text{ТО } y = \widetilde{G}^i, \quad i = 1,...,M. \tag{1}$$

Основываясь на нечеткой логике типа 2, нечеткий выход комбинирует правила и дает отображение входных нечетких множеств в выходные нечетких множества (типа 2). Необходимо вычислить объединения $\sqcup$ и пересечения П, задаваемые как нечеткие отношения 2-го типа. Если $\widetilde{F}_1^i \times ... \times \widetilde{F}_p^i = \widetilde{A}^i$, правило (1) можно записать в виде

$$R^i: \quad \widetilde{F}_1^i \times ... \times \widetilde{F}_p^i \to \widetilde{G}^i \quad = \quad \widetilde{A}^i \to \widetilde{G}^i, \quad i = 1,...,M.$$

Правило R^i задает функцию принадлежности $\mu_{R^i}(x,y) = \mu_{R^i}(x_1,...,x_p,y)$, где $\mu_{R^i}(x,y)$ можно представить как

$$\mu_{R^i}(x,y) = \mu_{\widetilde{A}^i \to \widetilde{G}^i}(x,y) = \mu_{\widetilde{F}_1^i}(x_1) \prod ... \prod \mu_{\widetilde{F}_p^i}(x_p) \prod \mu_{\widetilde{G}^i}(y).$$

При использовании при разработке агента интервальных нечетких множеств типа-2 и умножения в качестве *t*-нормы, результат пересечения всех множеств антецедента $F^i(x) = \prod_{k=1}^{p} \mu_{\widetilde{F}_k^i}(x_i)$ дает в результате интервал срабатывания, представляемый парой значений:

$$F^i(x) = \left[\underline{f}^i, \overline{f}^i\right],$$

где $\underline{f}^i = \underline{\mu}_{\widetilde{F}_1^i}(x_1) * ... * \underline{\mu}_{\widetilde{F}_p^i}(x_p)$ – нижняя граница принадлежности, а $\overline{f}^i = \overline{\mu}_{\widetilde{F}_1^i}(x_1) * ... * \overline{\mu}_{\widetilde{F}_p^i}(x_p)$ – верхняя граница соответственно.

В общем случае, значение, поступающее на вход нечеткого правила R^i, задается нечетким множеством типа 2 со следующей функцией принадлежности:

$$\mu_{\widetilde{A}_x}(x) = \mu_{\widetilde{x}_1}(x_1) \prod ... \prod \mu_{\widetilde{x}_p}(x_p) = \prod_{k=1}^{p} \mu_{\widetilde{x}_k}(x_k),$$

где $\widetilde{x}_k$ (k =1,..., p) – обозначения нечетких множеств каждого из входов после фузификации. Каждое правило R^i задает результирующее нечеткое множество 2 типа такое, что

$$\mu_{\widetilde{B}^i}(y) = \mu_{\widetilde{A}_x \circ R^i}(y) = \coprod_{x \in X} \left[\mu_{\widetilde{A}_x}(x) \prod \mu_{R^i}(x,y)\right].$$

Данное выражение представляет собой соотношение между нечетким множеством 2-го типа, которое активирует одно правило вывода и нечетким множество 2-го типа, полученным в результате его применения.

Компонент "Редуктор типа" создает нечеткое множество типа 1, называемое центроидом, которое в дальнейшем преобразуется дефузификатором в четкое значение. Центроид нечеткого множества A определяется как

$$C_A = \frac{\sum_{k=1}^{R} x_k \mu_A(x_k)}{\sum_{k=1}^{R} \mu_A(x_k)}.$$

В непрерывном случае построение центроида нечеткого множества типа 2 оказывается сложной вычислительной задачей. Эта задача упрощается, если функции принадлежности второго порядка имеют интервальный характер. Для расчета центроида нечеткого множества, представляющего собой результат применения системы нечетких правил, вначале определяется центроид каждого из консеквентов нечетких правил

$$C_{\widetilde{G}^i} = \int_{\theta_1 \in J_{y1}} \dots \int_{\theta_N \in J_{yN}} 1 / \frac{\sum_{i=1}^{N} y_i \theta_i}{\sum_{i=1}^{N} \theta_i}. \quad (2)$$

Очевидно, что центроид (2) представляет собой нечеткое множество типа 1, т. е.

$$C_{\widetilde{G}^i} = \int_{x \in [x_l, x_p]} 1 / x \quad = [x_l, x_p].$$

Построение центроида $[\, y_l(x), y_r(x)\,]$ сводится к минимизации относительно уровней срабатывания f^l следующего значения:

$$\min \left[\sum_{i=1}^{M} y_l^i f^i \Big/ \sum_{i=1}^{M} f^i \right], \quad (3)$$

с учетом ограничений $f^i \in [\breve{f}^i, \hat{f}^i]$ и отысканию максимального значения

$$\max \left[\sum_{i=1}^{M} y_r^i f^i \Big/ \sum_{i=1}^{M} f^i \right], \quad (4)$$

где $f^i \in [\breve{f}^i, \hat{f}^i]$. Известен в работе [10] итерационный алгоритм для поиска вышеупомянутых значений $y_l(x), y_r(x)$.

Задача компонента “Дефузификатор” – на основе этих значений определить значение, являющееся результатом применения ко входным значениям $x = (x_1, \dots, x_p)$ системы нечетких правил (1): $y(x) = (y_l + y_r)/2$.

Обобщенная архитектурная модель интеллектуального агента на основе нечеткой логики показана на рис. 1. Каждый компонент такой архитектуры выполняет роли поставщика и потребителя значений переменных, представленных нечеткими множествами 2 типа, обычными нечеткими множествами (1-го типа) и четкими значениями на числовой шкале. Сочетание ролей внутри одного компонента однозначно определяет тип компонента, представленный в данной архитектуре. Так, например, сочетание ролей “Потребитель: нечеткое значение типа 2” и “Поставщик: нечеткое значение типа 1” указывает на компонент, называемый редуктором типа. Сочетание ролей “поставщик-потребитель” демонстрирует зависимости по данным приводит к невозможности параллельного выполнения указанных компонентов. Однако, существует возможность конвейерной обработки, при которой в каждом слое (ярусе) задействован один из компонентов, передающий полученное после обработки значение компоненту-потребителю в следующем ярусе. Такая многоярусная схема представляется целесообразной в том случае, если нечеткие агенты постоянно получают новые входные значения, но не чаще чем среднее время реализации нечеткого вывода или понижения типа средствами нечеткой логики типа 2. Заметим, что компонент нечеткого вывода также может быть представлен в виде двухъярусной схемы, в которой на первом ярусе параллельно определяются значения, являющиеся результатом действия каждого отдельного правила, а во втором ярусе осуществляется агрегирование (объединение) полученных значений, для чего применяется специально определенная операция суммы нечетких множеств типа 2, или понижение типа множества, задаваемая выражениями (3) и (4).

Модели мультиагентных систем для кластерной системы СКИТ

С увеличением количества значений (переменных), обрабатываемых одновременно нечетких агентом, система нечетких правил значительно усложняется. Основной причиной такого усложнения является число нечетких правил, которое экспоненциально зависит от количества входных значений и соответствующих им лингвистических термов (так называемая проблема экспоненциального роста [11]). В результате увеличения такого количества правил, вычисление выходных значений не только занимает больше времени, но затрудняется понимание самой системы правил, что особенно актуально для системы на основе нечеткой логики высшего порядка. Одним из подходов, применяемых с целью упрощения системы нечетких правил является построение эквивалентной иерархической нечеткой системы [11]. Такая система может иметь подчиненную структуру с точки зрения подсистем нечеткого вывода и связей между ними. В качестве таких подсистем могут использоваться нечеткие агенты, имеющие структуру на рис. 1. В этом случае каждая подсистема представляет собой отдельную базу правил, задаваемую выражением (1), тогда как каждое взаимодействие представлено промежуточной переменной связывающей пары соседних баз правил (рис. 2). Значение этой промежуточной переменной совпадает со значением выходной переменной для базы правил подчиненного агента и значением входной переменной агента-координатора. В простейшем случае каждый агент принимает значения не более 2 входных переменных и производит значение 1 промежуточной переменной на выходе, тогда координация является многоярусной с использованием не более двух входов на каждом ярусе.

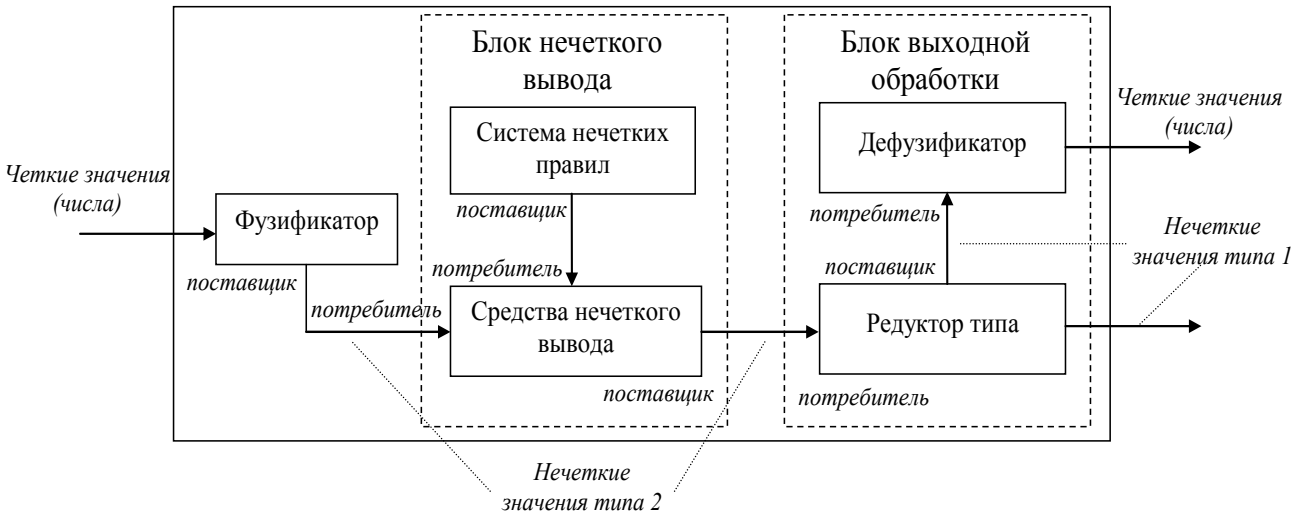

Рис. 1. Архитектура нечеткого агента на основе нечеткой логики высшего типа

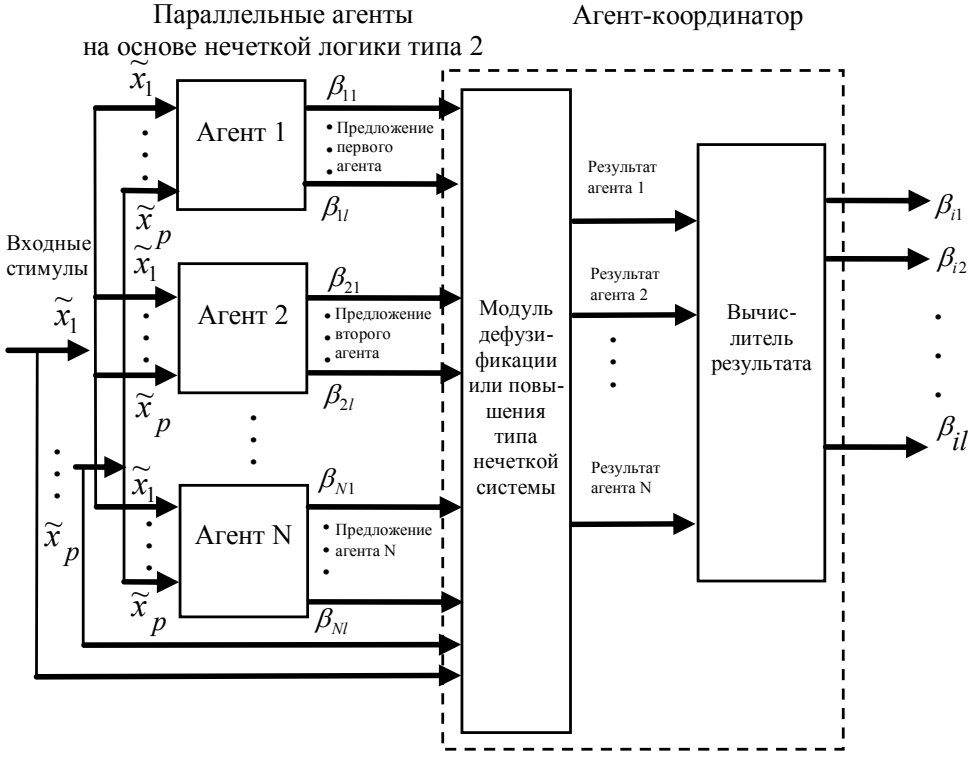

Рис. 2. Структура иерархической нечеткой мультиагентной системы для высокопроизводительной среды

- Иерархическая нечеткая мультиагентная система представляет собой подробное представление отдельного нечеткого агента с целью упрощения нечетких баз правил с явным учетом отдельных агентов и взаимодействия между ними. Кроме того, дополнительная эффективность достигается за счет возможности лучшего представления одновременного воздействия уменьшенного количества входов на отдельные (упрощенные) базы правил. Однако, при этом теряется точность из-за накопления ошибок в результате повторяющегося применения фузификации, нечеткого вывода, редукции типа и дефузификации в течение нескольких циклов работы такой системы. Для повышения точности редукция типа с дефузификацией и дальнейшей фузификацией промежуточных переменных могут отсутствовать, нечеткое значение, представляющее собой нечеткое множество типа 2 непосредственно передается между агентами.

Обобщением такой модели мультиагентных систем является ее представление как виртуальной сетки (grid). Каждый агент представлен вершиной сетки в то время как взаимодействие между агентами – соединениями между этими вершинами. Модель мультиагентной системы с $p \times q$ вершинами $\{N_{11} \ldots N_{p1}\}, \ldots, \{N_{1q} \ldots N_{pq}\}$, $p \times q$ входами вершин $\{x_{11} \ldots x_{\text{p1}}\}, \ldots, \{x_{1q} \ldots x_{pq}\}$ принимающими лингвистические значения из любых допустимых входных множеств, $p \times q$ выходами вершин $\{y_{11} \ldots y_{p1}\}, \ldots, \{y_{1q} \ldots y_{pq}\}$ принимающие лингвистические значения из любых допустимых множеств выходов, состоящая из p горизонтальных уровней и q вертикальных ярусов (слоев), может быть описана формулой (5).

$$\begin{array}{l} \qquad\qquad \text{Ярус 1} \ \ldots\ldots\ldots\ldots\ldots \ \text{Ярус } q \\ \text{Уровень 1 } N_{11}(x_{11}, y_{11}) \ldots\ldots\ldots\ldots N_{1q}(x_{1q}, y_{1q}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots . \\ \text{Уровень } p \ N_{p1}(x_{p1}, y_{p1}) \ldots\ldots\ldots\ldots N_{pq}(x_{pq}, y_{pq}) \end{array} \tag{5}$$

Структура сетки в формуле (5) определяет местоположение агентов, а также их входы и выходы. В этом случае, каждый вход и выход может быть как скаляром так вектором, представляющим набор нечетких значений типа 2. Уровни в этой структуре сетки представляют собой пространственные иерархии узлов в плане субординации в пространстве, тогда как ярусы задают временную зависимость с точки зрения последовательности выполнения.

Однако формула (5) не дает никакой информации о соединениях между вершинами в виртуальной сетке. Такая информация содержится в структуре соединений в формуле (6), согласно которой $p \times (q-1)$ соединений узлов $\{z_{11,12} \ldots z_{p1,p2}\}, \ldots, \{z_{1q\text{-}1,1q} \ldots z_{pq\text{-}1,pq}\}$ принимают лингвистические значения из допустимых множеств для соответствующих выходов и входов узла.

$$\begin{array}{l} \qquad\qquad \text{Ярус 1} \ \ldots\ldots\ldots\ldots\ldots\ldots \ \text{Ярус } q\text{-}1 \\ \text{Уровень 1 } z_{11,12}{=}y_{11}{=}x_{12} \ldots\ldots\ldots\ldots z_{1q\text{-}1,1q}{=}y_{1q\text{-}1}{=}\ \mathrm{x}_{1q} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots . \\ \text{Уровень } p \ z_{p1,p2}{=}y_{p1}{=}x_{p2} \ldots\ldots\ldots\ldots z_{pq\text{-}1,pq}{=}\ y_{pq\text{-}1}{=}\ x_{pq} \end{array} \tag{6}$$

Для единообразия, мультиагентные системы, описываемые выражениями (5) и (6) предполагают наличие узлов в каждой ячейке базовой структуры сетки. Однако, не все ячейки в модели должны быть заполнены агентами. Для упрощения, все соединения в вышеприведенной модели – между узлами одного уровня и в соседних ярусами. Тем не менее, некоторые соединения в мультиагентной системе могут быть между агентами на разных уровнях или агентами несоседних ярусов.

В качестве целевой платформы для выполнения мультиагентных систем используется кластерная система СКИТ-3 [12]. Для поддержки взаимодействия интеллектуальных агентов была создана специальная библиотека, которая включает ряд шаблонов, классов и методов для организации поведения агентов – циклического, одноразового и многоразового, а также для обмена разными типами асинхронных сообщений с использованием MPI в качестве носителя. Вариант библиотеки реализован для использования возможностей программно-апаратного стека CUDA. Библиотека дает возможность организовать модель системы как виртуальную сетку (grid) агентов, в которой агентам разрешается взаимодействовать тольки в пределах ограниченной дистанции (количество переходов по сетке). Из-за распределения глобального состояния между процессорами, часть состояния отдельных агентов может сохраняться у соседних агентов. При циклическом выполнении алгоритмов нечеткого вывода элементы этого состояния могут запрашиваться до начала выполнения следующей итерации. Специальный примитив используется для барьерной синхронизации выполнения итераций (этапов выполнения) нечетких агентов. Однако, такой подход не учитывает разнообразия типов памяти и время задержки передачи сообщений между агентами, которые используют такие типы памяти. Поэтому, для уменьшения затрат на коммуникацию нами предложен метод, основанный на виртуальной сетке агентов над иерархией типов памяти агентов.

Виртуальная сетка агентов разбивается на блоки агентов, закрепленные за отдельными элементами обработки (CPU, GPU). Каждый такой блок состоит из $L \times L$ агентов, окруженных E слоями, которые представляют агенты в соседних блоках. Благодаря этому, вычисления в пределах локального блока могут быть продолжены на E итераций, после чего становится необходима передача данных между блоками. Поскольку блок размера $L + 2E$ содержит всю информацию, необходимую для локальных вычислений, это вызывает уменьшение количества обменов в сетке. Затраты на коммуникацию после E итераций могут быть представлены по-разному: для CPU и GPU соответственно $C_{CPU} = c_w(L^2 - (L-2E)^2) + c_r((L+2R)^2 - L^2)$, $C_{GPU} = c_w(L^2) + c_r((L+2R)^2 - L^2)$, где c_w, c_r – затраты времени на чтение (запись) данных одного агента в среднем.

При данном подходе на самом нижнем уровне иерархии расположены агенты, выполняющиеся как потоки CUDA. Даже на этом уровне схема уменьшения затрат на коммуникацию достигается благодаря использованию блоков агентов $L + 2E$ в качестве блоков CUDA. Выше находятся блоки агентов на отдельных GPU, синхронизация между которыми осуществляется при последовательном выполнении ядер CUDA. На наивысшем уровне наибольшие блоки агентов распределяются между вершинами, коммуникация между которыми осуществляется на основе MPI. После E_{GPU} итераций, для блоков агентов с соответствующими рангами осуществляются неблокирующие вызовы MPI_Irecvs та MPI_Isends, которые обновляют состояния $(L+2R)^2 - L^2$ агентов, после чего их выполнение продолжается.

Моделе-ориентированный подход к разработке интеллектуальных агентов на основе нечеткой логики высшего типа

Основные процессы моделе-ориентированного программирования – это разработка (выявление и анализ) требований, проектирование агентов и реализация агентов мультиагентной системы. Значение отдельных моделей интеллектуальных агентов не одинаково для различных процессов. При этом на этапах разработки требований и проектирования создаются основные платформно-независимые модели: начальная модель нечеткого агента, начальная ролевая модель, уточненная ролевая модель и модель внутриагентного управления (МВУ). Платформно-зависимые модели включают в себя модели нечеткого агента, соответствующие всем деталям текстового представления. Естественно, что такие модели являются объектом процессов конструирования и тестирования. В настоящее время такие модели разработаны для таких платформ как FCL [13], JADE [14] и для разработанной нами библиотеки классов программирования нечетких агентов в среде суперкомпьютера СКИТ-3 [12].

Применяемые в процессах моделе-ориентированной разработки нечетких мультиагентных систем метамодели, модели и трансформации описаны на основе фреймворка Eclipse Modeling Framework [15], использующего для построения моделей метамодель высшего уровня Ecore. Средства описания метамоделей Ecore предназначены для поддержки времени выполнения, такой как уведомления об изменениях, средства для сохранения моделей со встроенными средствами сериализации на основе стандарта XMI (XML Metadata Interchange) [16], и эффективный интерфейс программиста для операций над объектами Ecore. Аналогичная технология для представления и обработки метамоделей – Meta-Object Facility (MOF) [17]. Тем не менее, метамодель EMF проще, чем метамодель MOF с точки зрения понятий, свойств и внутренней структуры.

Исходной при трансформационной разработке агентов может служить их платформно-независимая модель нечетких правил. Для представления системы нечетких правил разработана метамодель Ecore правил, содержащая все необходимые понятия для конструирования систем нечетких правил высшего типа. В качестве основных классов (понятий) выбраны FuzzyRule (нечеткое правило), FuzzyVariable (нечеткая переменная), FuzzyValue (нечеткое значение), Type1FuzzySet, Type2FuzzySet и LinguisticTerm (нечеткие множества типа 1 и 2, связанный с ними лингвистический терм). Каждое правило включает набор антецедентов и консеквентов, которым соответствует нечеткое значение, связанное в свою очередь с нечеткой переменной, описывающей набор допустимых лингвистических значений и их функций принадлежности. Различные способы выполнения нечетких правил типа 2 учитываются с помощью конкретных классов FuzzyRuleExecutor. Это позволяет расширять данную метамодель другими видами нечетких систем, не затрагивая при этом другие модели.

Однако, задавать исходную модель правил в текстовом виде или базовыми средствами дерева объектов Ecore было бы не наглядно и затрудняло бы ее восприятие. Для ввода и редактирования таких моделей создан специальный графический редактор со средствами навигации по нечетким правилам и композиции правил замены (удаление и добавление нечетких переменных, множеств, функций принадлежности, включение переменных в антецеденты и консеквенты правил и т. д.). Создание такого редактора производилось в рамках методологии предметно-ориентированного моделирования (domain-specific modeling), включающей систематическое использование визуальных (основанных на графах) предметно-ориентированных языков моделирования для представления различных аспектов системы. Абстрактный синтаксис такого языка для моделирования нечетких агентов задается метамоделью в системе EMF (формат Ecore). Для описания конкретного визуального синтаксиса диаграмм использовался фреймворк GMF (Graphical Modeling Framework) [18]. На основе такой модели порождается полный код программы-редактора диаграмм, работающий в среде Eclipse.

Определение правил трансформаций, отображающих модели, создаваемые с помощью сгенерированного редактора, осуществлялось с использованием языка Query/View/Transformation (QVT), разработанного под руководством OMG. Хотя стандартное определение QVT предназначено для отображений над элементами MOF- совместимых метамоделей, реализация, применяемая в Eclipse, основана на метамоделях Ecore. QVT определяет не один, а три предметно-ориентированных языка: реляционный (декларативный) Relations, ядро Core и язык операционных отображений OperationalMappings. Хотя известно множество других языков, родственных QVT по лежащим в их основе понятиям (например, ATL и Tefkat), в настоящее время именно QVT/OperationalMappings обеспечивает как наибольшую практическую проработанность, так и наибольшую степень совместимости с концепциями, предлагаемыми OMG в рамках моделе-ориентированной архитектуры.

В процессе разработки требований также определяется начальная ролевая модель (РМ) агентов. Предполагается что интеллектуальный агент обязывается выполнить одну или несколько ролей в течении своего жизненного цикла. Одной из особенностью разработки агентов является то, что виды деятельности агента связаны с ролью, а не с системой. Кроме того, после определения возможностей агентов и разложения их на простые деятельности, необходимо определить динамическую композицию из этих видов деятельности по каждой роли таким образом, что агент достигает поставленной цели.

В процессе проектирования генерируются дополнительные роли агента, уточняющие начальную ролевую модель. В общей ситуации, для каждой нечеткой переменной, входящей в начальную модель правил, порождаются две дополнительные роли в соответствии с шаблоном проектирования поставщик/потребитель [19]. Определенные ранее роли связываются с выбранными типами агентов, составляющими проектируемую мультиагентную систему и порождающими отдельные (типизированные) экземпляры агентов. При

необходимости размещения агентов на одной вершине для экономии коммуникационных ресурсов их типы могут быть совмещены, что приводит к суммированию (наложению) ролей, которые они должны выполнять. В процессе проектирования к уточненной ролевой модели применяются трансформации, позволяющие получить спецификации моделей внутриагентного управления – по одной для каждой сгенерированной роли. Для разработки таких трансформаций типа модель-модель (M2M) использован язык QVT/OperationalMappings.

Многие методологии разработки (как Tropos или INGENIAS [20, 21]) навязывают специфические ментальные модели агента. В отличие от этого, модель внутриагентного управления (МВУ) основана на нечетких схемах состояний и не делает никаких дополнительных предположений о таких аспектах как модель коммуникации, процесс вывода или ментальные состояния (убеждения-желания-намерения) агентов.

Работой таких агентов можно управлять с учетом степени достижимости их "размытых" состояний. Каждая модель МВУ уточняется определением условий и события приема/передачи сообщений, которые позволят переходы от одной системы нечетких правил к другой. МВУ задается нечеткой схемой переходов, представленной нечетким мультиграфом $G = (Q, T, \mu_G, \Sigma)$, где $Q = \{q_1, q_2, ..., q_n\}$ – множество состояний (вершин графа), где q_1 – начальное состояние; $T = \{t_1, t_2, ..., t_n\}$ – множество переходов (дуг) графа; $\mu_G : T \to [0,1]$ – функция принадлежности переходов данной схеме; $\Sigma : T \to A$ – функция разметки переходов событиями из некоторого конечного алфавита событий (сообщений) A.

Порождение кода мультиагентной системы состоит из двух трансформаций – выполнения ряда преобразований модель-модель с целью получения платформно-специфической модели нечетких агентов и запуска последующих трансформаций "модель-код" для получения текста программы целевого агента.

В настоящее время известно много платформ для поддержки функционирования мультиагентных систем – таких как Jason, 3APL, JACK и других [22]. Большинство этих платформ поддерживает только архитектуру Убеждение-Желание-Намерение (Belief-Desire-Intention, BDI), которая воплощает один из основных видов делиберативных ("разумных" агентов) и содержит явно представленную структуру данных, соответствующую этим трем указанным свойствам рассуждений, применяемых агентом при решении задач. Привлекательность BDI-подхода снижается, поскольку был предложен ряд других моделей (реактивные, мотивированные), позволяющих устранить проблемы BDI-агентов [7].

FCL – единственный стандартизированный предметно-ориентированный язык для спецификации систем нечеткого логического вывода [13]. Хотя его основное преимущество – наглядность и простота, использование его переносимых реализаций (в виде интерпретаторов) на языке C++ в составе GPU кластера не представляется возможным из-за ограничения памяти графических ускорителей и замедления работы. Поэтому порождается код на языке C++ (или на языке Java) для кластерной платформы, который содержит вызовы соответствующих операций над нечеткими множествами типа 2. В качестве целевых платформ выполнения сгенерированных мультиагентных систем используется кластерная система СКИТ-3 [12]. Поскольку, система СКИТ не содержит специальной платформы для поддержки взаимодействия агентов, была создана специальная библиотека на языке C++, включающая ряд шаблонов, классов и методов для организации поведения агентов – циклического, одноразового и многократного, а также для обмена различными типами асинхронних сообщений.

Трансформация МВУ в код агента для целевой платформы – автоматизированное преобразование "модель-текст", создающее класс агента и несколько классов *Behaviour* (Поведение) для каждой модели МВУ. Особенностью функционирования порождаемых агентов для СКИТ является то, что в отличие от JADE не используется язык ACL (агенты на кластере не взаимодействуют с другими открытыми платформами), вместо этого генерируется код, выполняющий маршалинг/демаршалинг основных типов сообщений. Также, связывание сервисов с агентами-поставщиками выполняется во время компиляции, а не динамически, что позволяет сократить накладные расходы на обмен сообщениями.

Для генерации текстовых файлов программ-агентов разработаны преобразования "модель-текст" на основе текстовых шаблонов платформы XPand, запускаемых в контексте компонентов EMFT Workflow. Преимущество Xpand то, что он является независимым от исходной модели. Это означает, что любая из программ-парсеров может быть использована для общих моделей программного обеспечения, таких как MOF или EMF. Такая трансформация генерирует методы поведения агента для получения и отправки сообщений и композитные поведения нечеткой схемы переходов, которые координируют выполнение простых поведений.

Поведение группы агентов, задаваемое с помощью нечеткой логики типа 2

Рассмотрим группу, состоящую из n агентов, поведение которой в *m*-мерном пространстве может быть задано следующими уравнениями движения:

$$p_i(t) = f\big(v_i(t), p_i(t-1)\big), \quad v_i(t) = g\big(u_i(t), v_i(t-1)\big), \tag{7}$$

где $p_i(t), v_i(t) \in \mathbb{R}^m$ – вектор положения и скорости агента i, соответственно, а $u_i(t) \in \mathbb{R}^m$ – рассматриваемый далее вектор воздействия на агент *i*. Предполагается, что все агенты полностью задействованы и могут взаимодействовать друг с другом. Состояния всех n агентов можно объединить в три вектора: $p = [p_1, ..., p_n] \in \mathbb{R}^{nm}$, $v = [v_1, ..., v_n] \in \mathbb{R}^{nm}$ и $u = [u_1, ..., u_n] \in \mathbb{R}^{nm}$.

Обозначим $\tilde{p}(t)$ и $\tilde{v}(t)$ требуемое положение и скорость для центра группы агентов. Для дальнейшего определения нужного положения и скорости ко всем агентам в группе применяется следующее выражение:

$$\tilde{u}_i = -c_1(p_i - \tilde{p}) - c_2(v_i - \tilde{v}), \tag{8}$$

где c_1 и c_2 – коэффициенты обратной связи, которые можно представить в виде n-мерных диагональных матриц. При использовании воздействия $\tilde{u}(t)$, все агенты принадлежащие к одной и той же группе следуют требуемой траектории. Центр группы может быть представлен средним значением всех состояний, т. е.

$$p_c = \frac{\sum_{i=1}^{n} p_i(t)}{n}, \; v_c = \frac{\sum_{i=1}^{n} v_i(t)}{n}, \tag{9}$$

Из (7) – (9) следует, что управляющее воздействие для центра группы имеет следующий вид:

$$\tilde{u}_c = -c_1(p_c - \tilde{p}) - c_2(v_c - \tilde{v}).$$

Тогда поведение центра группы может быть представлено как:

$$p_c(t) = f\big(v_c(t), p_c(t-1)\big), \;\; v_c(t) = g\big(\tilde{u}_c(t), v_c(t-1)\big).$$

Рейнольдсом [23] была предложена модель для характеризации коллективного поведения группы агентов, действующих по принципу “держаться вместе”. Такая модель состоит из трех следующих эвристических правил: правила сплочения (cohesion), определения степени разделения (separation) и согласования скоростей.

В данной работе разработаны не только три правила Рейнольдса, но и поведение для обхода препятствий посредством трех интервальных систем нечетких правил (СНП) 2-го типа для достижения поведения по принципу “держаться вместе” в среде с препятствиями. Два эвристических правила Рейнолдса, а именно правила сплочения и определения степени разделения можно представить в виде одной СНП для согласования интервалов. При этом для представления всех трех СНП (для согласования интервалов, согласования скоростей и, соответственно, обхода препятствий) используются трапециевидные интервальные нечеткие множества типа-2.

Для моделирования поведения по принципу “держаться вместе” разработаны три СНП, соответствующие каждому из типов составного поведения. Каждая СНП получает четкий вход и затем вырабатывает управляющие воздействия $u^1{}_i$, $u^2{}_i$ и $u^3{}_i$. Для определения перемещения каждого агента на следующем шаге в качестве u_i в (7) используется система нечетких правил второго яруса, принимающая на входе вектора управляющих воздействий, в том числе $\tilde{u}_i$, задаваемый выражением (8).

СНП для подержания интервала заставляет агентов собираться вокруг центра группы при сохранении постоянного расстояния между агентами, чтобы избежать столкновений с близлежащими агентами.

Для представления отклонения дистанции между агентами от необходимой разработано восемь трапециевидных нечетких множеств типа 2, след неопределенности каждого из которых задает неопределенность данных от датчиков: НБ (негативное большое), Н (негативное), НН (негативное небольшое), О (отсутствует), ПН (позитивное небольшое), П (позитивное), ПБ (позитивное большое) и ОБ (очень большое). Восемь нечетких правил для такого типа поведения можно представить следующим образом:

ЕСЛИ ($r_{ij}^2 - d^2$) = НБ ТО k_{ij} = Большая СО
ЕСЛИ ($r_{ij}^2 - d^2$) = Н ТО k_{ij} = Средняя СО
ЕСЛИ ($r_{ij}^2 - d^2$) = НН ТО k_{ij} = Небольшая СО
ЕСЛИ ($r_{ij}^2 - d^2$) = О ТО k_{ij} = Отсутствует
ЕСЛИ ($r_{ij}^2 - d^2$) = ПН ТО k_{ij} = Небольшая СП
ЕСЛИ ($r_{ij}^2 - d^2$) = П ТО k_{ij} = Средняя СП
ЕСЛИ ($r_{ij}^2 - d^2$) = ПБ ТО k_{ij} = Большая СП
ЕСЛИ ($r_{ij}^2 - d^2$) = ОБ ТО k_{ij} = Очень Большая СП

где r_{ij} – текущее расстояние между агентами i и j, а d – необходимое (безопасное) расстояние между двумя агентами. Функция воздействия на агент i, которая учитывает взаимодействия со всеми агентами в группе задается следующим образом:

$$u_i^1(t) = -\sum_{i}^{n} k_{ij}(p_i - p_j), \tag{10}$$

где $k_i^s = [k_{i1},...,k_{in}] \in \mathbb{R}^n$ – вектор, вырабатываемый системой нечетких правил. Функция воздействия применяется ко всем агентам в группе, то есть, если нечеткий вход $r_{ij}^2 - d^2$ принимает отрицательное значение, то на агенты действует сила отталкивания (СО); наоборот, если значение входа становится положительным, то

на агенты действует сила притяжения (СП). Таким образом, агенты в группе могут собираться вместе без столкновений друг с другом.

Для оценивания степени, в которой поведение агентов соответствует принципу поддержания интервала, введем следующую функцию для группы агентов:

$$P(p) = \sum_{i}^{n}\sum_{j=1}^{n}\psi(\left|p_i - p_j\right| - d)\,, \tag{11}$$

где $\psi(x) = x^2$, $\left|p_i - p_j\right|$ – модуль вектора разницы положения агентов i и j в пространстве. Чем меньше значение функции (11), тем стабильнее становится группа агентов.

СНП для согласования скоростей предназначена для выравнивания скорости каждого агента по отношению к остальным. Для определения выходного значение k_{ij}^{v} СНП использует только одно четкое входное значение $\left|v_i - v_j\right|$ модуля вектора разницы скоростей агентов i и j. Для этой СНП, входные и выходные функций принадлежности (след нечеткости которых показан на рис. 3) принимают значения Отсутствует, Средняя, Большая. Три простых нечетких правила представлены следующим образом:

ЕСЛИ $\left|v_i - v_j\right|$ = Отсутствует ТО k_{ij} = Отсутствует
ЕСЛИ $\left|v_i - v_j\right|$ = Средняя ТО k_{ij} = Средняя
ЕСЛИ $\left|v_i - v_j\right|$ = Большая ТО k_{ij} = Большая.

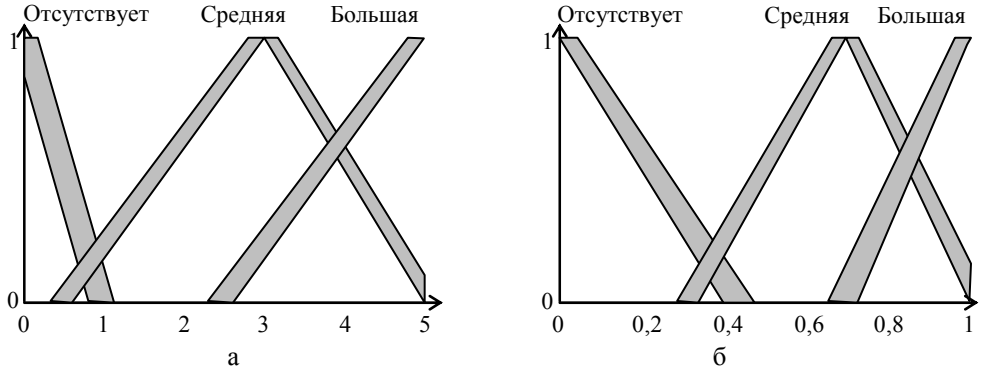

Рис. 3. Функции принадлежности для согласования скоростей: а – входные и б – выходные

Функция управления агента i для согласования скоростей следующая:

$$u_i^2(t) = -\sum_{i}^{n} k_{ij}^{v}(v_i - v_j)\,, \tag{12}$$

где $k_i^v = [k_{i1},...,k_{in}] \in \mathbb{R}^n$ – вектор, вырабатываемый системой нечетких правил. С помощью этой функции управления агент стремится выравнить свою скорость по отношению к другим агентам, имеющим с ним большую разницу скоростей.

Для измерения степени выравнивания скорости используется следующая функция отклонения скоростей:

$$V(v) = \sum_{i}^{n}\sum_{j=1}^{n}\psi(\left|v_i - v_j\right|)\,.$$

Если скорость $v_i(t)$ достигает средней скорости всех агентов в группе, направление их движения и модуль вектора скорости будет примерно одинаковы и функция отклонения скоростей $V(p)$ обращается в нуль.

Поведение обхода препятствий необходимо для безопасной навигации в среде с препятствиями. При разработке СНП 2-го типа предполагается, что препятствия являются неподвижными и каждый агент может определить расстояние от агента до препятствия, когда препятствие находится в пределах максимального определяемого расстояния $d_{\max}$. При этом предположении, диапазон значений датчика представлен тремя интервального нечеткими множествами типа 2: Отсутствует, Близко и Далеко. Консеквенты правил включают три нечетких множества: Отсутствует СО (сила отталкивания), Малая СО, Большая СО. Три простых нечетких правила представлены следующим образом:

ЕСЛИ d_i = Отсутствует ТО k_{ij} = Отсутствует СО

ЕСЛИ d_i = Близко ТО k_{ij} = Малая СО
ЕСЛИ d_i = Далеко ТО k_{ij} = Большая СО.

Функция управления *i*-го агента для обхода препятствий u_i^3 вычисляется аналогично (10) и (12). Если расстояние между агентом и препятствием становится меньше, то на агент воздействует большая сила отталкивания от препятствия. Таким образом, СНП типа-2 приводит к избеганию препятствий агентом.

При моделировании использовалась группа из 12 агентов, перемещающихся на плоскости (т. е. $m = 2$). Позиции и вектора скоростей всех агентов доступны любому агенту принадлежащему группе для согласования интервалов (10), согласования скоростей (12), и обхода препятствий. Позиции и скорости агентов изначально заданы в ограниченном диапазоне случайных чисел. Следующие параметры оставались неизменными при моделировании: $d = 0{,}3$ для интервалов, $d_{\max} = 1$ для обхода препятствий, $c_1, c_2 = 1$ для отслеживания заданной траектории. Такая траектория, которой должен следовать центр группы, представлена следующей прямой: $\tilde{p}(t) = [0{,}5t, 0]$.

Поведение группы агентов, задаваемое с помощью СНП типа-2 сравнивалось с аналогичным поведением агентов, порождаемых на основе обычных нечетких правил (типа-1). При использовании СНП 2-го типа агенты выполняют обход препятствий более плавно, чем при использовании обычных правил, с меньшим средним отклонением от центра группы. Что касается функции поддержания интервала (5), ее значения для СНП типа 2 в среднем меньше, чем для СНП 1-го типа. Кроме того, значение степени отклонения скоростей агентов, управляемых тремя СНП типа 2 меньше чем для соответствующих им обычных нечетких правил.

Выводы

Таким образом, описанные модели интеллектуальных агентов и мультиагентных систем на основе нечеткой логики высшего типа, позволяют более информативно представить степень неопределенности системы нечетких правил при спецификации поведения таких агентов и систем. Создан моделе-ориентированный подход для порождения платформно-специфических моделей нечетких мультиагентных систем, функционирующих в перспективных высокопроизводительных средах, в том числе кластерной системе СКИТ.

Предложена архитектура мультиагентной системы для достижения таких аспектов поведения, как поддержание дистанции между агентами, согласование скоростей и, соответственно, обход препятствий. Поведение группы агентов, задаваемое с помощью нечеткой логики высшего типа более информативно по отношению к критериям согласования дистанции и скоростей чем поведение, порождаемое на основе обычных нечетких правил.

1. *Macal C., North M.* Agent-based modeling and simulation for exascale computing. – SciDAC Review, Summer 2008. – P. 34–41.
2. *Zadeh L.A.* Fuzzy Logic = Computing With Words// IEEE Transactions on Fuzzy Systems. – 1996. – Vol. 4. – P. 103–111.
3. *Zadeh L. A*. The concept of a linguistic variable and its application to approximate reasoning // Information Sciences. – 1975. – Vol.8, N 8. – P. 199–249, P. 301–357.
4. *Mendel J.M., John R.I.* Type-2 sets made simple // IEEE Trans. on Fuzzy Systems. – 2002. – Vol.10, N 2. – P. 117–127.
5. *Karnik N.N., Mendel J.M., Liang Q*. Type-2 fuzzy logic system // IEEE Trans. on Fuzzy Systems. – 1999. – Vol.7, N 6. – P. 643–658.
6. *Парасюк И.Н., Ершов С.В*. Трансформационный подход к разработке интеллектуальных агентов на основе нечетких моделей // Проблеми програмування. – 2011. – № 2. – С. 62–78.
7. *Парасюк И.Н., Ершов С.В*. Нечеткие модели мультиагентных систем в распределенной среде // Проблеми програмування. – 2010. – № 2–3. – С. 330–339.
8. *Парасюк И.Н., Ершов С.В*. Моделе-ориентированная архитектура нечетких мультиагентных систем // Компьютерная математика. – 2010. – № 2. – С.139–149.
9. *Ершов С.В*. Принципы построения нечетких мультиагентных систем в распределенной среде // Там же. – 2009. – № 2. – С. 54–61.
10. *Mendel J.M.* Type-2 fuzzy sets and systems: an overview // IEEE Computational Intelligence Magazine. – 2007. – Vol. 2. – P. 20–29.
11. *Wang D., Zeng X., Keane J.* A survey of hierarchical fuzzy systems // International Journal of Computational Cognition. – 2006. – Vol. 4, N 1. – P. 18–29.
12. *Суперкомпьютеры* ИК НАН Украины. – http://icybcluster.org.ua.
13. *International* Electrotechnical Commission (IEC). Technical Committee No. 65: Industrial Process Measurement and Control. IEC 1131 – Programmable Controllers Part 7 – Fuzzy Control Programming. – http://www.fuzzytech.com/binaries/ieccd1.pdf
14. *Bellifemine F., Caire G., Greenwood D.* Developing Multi-Agent Systems with JADE. – Chichester: John Wiley & Sons Inc, 2007. – 303 p.
15. *Eclipse* Modeling Framework. – http://wiki.eclipse.org/EMF/
16. *MOF* 2.0/XMI Mapping. – http://www.omg.org/spec/XMI/2.1.1/
17. *OMG* MetaObject Facility. – http://www.omg.org/mof/
18. *Graphical* Modeling Framework. – http://www.eclipse.org/modeling/gmp/
19. *Application* Design Patterns: Producer/Consumer. – http://zone.ni.com/devzone/cda/tut/p/id/3023/
20. *Perini A., Susi A.* Automating model transformations in agent-oriented modeling // Agent-oriented software engineering VI. Lecture notes in computer science. – Springer, 2006. – P. 167–178.
21. *Pavon J., Gomez-Sanz J.* Agent-oriented software engineering with INGENIAS // Multi-agent system and applications III. Lecture notes in computer science. – Springer, 2003. – P. 8–38.
22. *Wooldridge M.J.* An Introduction to Multi-agent Systems. – Cambridge: MIT Press, 2002. – 366 p.
23. *Reynolds C.W*. Flocks, herds and schools: a distributed behavioural model // Comput. Graph. – 1987. – Vol.21, N 4. – P. 25–34.