УДК 681.3.06

И.Н. ПАРАСЮК, С.В. ЕРШОВ

# МОДЕЛЕ-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА НЕЧЕТКИХ МУЛЬТИАГЕНТНЫХ СИСТЕМ

*Рассматриваются нечеткие архитектурные модели мультиагентных систем для распределенной среды, обладающие свойствами реактивности и мотивированности. Предложены средства описания поведения нечетких агентов на основе трансформаций нечетких графов, способы их координации и адаптации с учетом нечетких значений полезности. Описаны основные метамодели и трансформации нечетких мультиагентных систем на основе моделе-ориентированной архитектуры.*



**Введение.** Программные модели гуманистических систем – систем, в основу которых положен принцип гранулирования информации, а процессы создания которых отображают сознание человека [1], являются важным объектом изучения теории нечетких систем. В настоящее время они широко используются при проектировании средств компьютеризации в различных сферах человеческой деятельности. К таким современным и перспективным программным моделям относятся и нечеткие мультиагентные системы, для успешного становления и развития которых нужен высокий уровень абстрактности целевых платформ разработки, создание конструктивных архитектурных моделей и средств порождения соответствующих целевых агентов.

Цель настоящей работы – формализация архитектуры мультиагентных систем на основе нечетких моделей и разработка соответствующих процедур, метамоделей и порождающих их трансформаций для моделе-ориентированной разработки с использованием подхода MDA. Перечисленные проблемы тематически вписываются в дальнейшее развитие исследований в направлении становления моделе-ориентованных архитектур программных систем в нечетком представлении [2–4].

**Архитектура интеллектуальных агентов: общие положения.** Под термином интеллектуальный агент понимаются программы, получающие информацию из окружающей среды и выполняющие над ней соответствующие операции, при этом их поведение

рационально [5]. Архитектуры таких агентов делятся на четыре общих класса: дедуктивные агенты, реактивные агенты (агенты, основанные на поведении), делиберативные («разумные») агенты и агенты, основанные на побуждениях [6–10].

При дедуктивном подходе агент манипулирует символьным представлением среды, используя операции логического вывода [6–8]. Агенты играют роль программ автоматического доказательства теорем, непосредственно выполняя логические спецификации. Основные недостатки данного подхода: проблема трансдукции, необходимость обеспечения непротиворечивых, надежных и «четких» знаний, невозможность манипулировать описанием динамической и стохастической среды, чрезвычайная иерархичность, необходимость построения полного агента для проведения испытаний, что не позволяет проводить итерационные пошаговые процессы разработки.

Парадигма Убеждение-Желание-Намерение (Belief-Desire-Intention) воплощает один из основных видов делиберативных («разумных» агентов) и содержит явно представленную структуру данных, соответствует этим трем указанным свойствам рассуждений, применяемых агентом при решении задач [6–8].

В агенте BDI [6, 7], среда $E$ начинается в специфическом состоянии конечного множества дискретных состояний: $E = \{e, e', ...\}$. Агент $Ag$ выбирает специфическое действие из доступного множества возможных действий $A$ на основании состояния среды и предыдущих выполненных действий: $A = \{\alpha, \alpha', ...\}$. Запуск $R$ – последовательность чередующихся состояний и действий. Среда запускается в определенном состоянии и агент выбирает действие, чтобы выполняться при этом состоянии. В результате этого действия, среда может перейти в ряд возможных состояний:

$$R = e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} \ ... \ \xrightarrow{\alpha_{u-1}} e_u .$$

Функция $Ag$ отображает запуски на действия, агент выбирает следующее действие, основываясь на доступной истории запуска системы: $Ag : R \to A$. Агент пробует найти и выполнить лучший план $plan\ (\pi)$, который является серией действий, выполнимых в определенном порядке: $\pi = \{\alpha_1, \alpha_2, ..., \alpha_n\}$.

BDI агент выполняет следующие функции:

1. Функция пересмотра убеждений агента (т. е. *brf*) модифицирует текущие убеждения агента на основании множества всех убеждений *Bel* и текущих результатов перцепции: $brf : \wp(Bel) \times Per \to \wp(Bel)$.

2. Желание, которое агент стремится достичь, считывается из стека *намерений*. Этот стек содержит все *желания* (т.е. *цели*), которые агент обязался выполнить. Агент отыскивает *планы*, указывающие желание, извлеченное из стека, в качестве постусловий. Из этих планов, только те, которые удовлетворяют своему предусловию (на основе его текущих убеждений), становятся возможными *вариантами* или *желаниями* для агента.

Делиберативный агент начинает *размышлять* используя два функциональных компонента: функция *выбора* $options: \wp(Bel) \times \wp(Int) \to \wp(Des)$, которая для текущего множества убеждений и намерений *Int* агента и производит множество возможных желаний *Des*. Для того чтобы выбрать между конкурирующими вариантами, агент использует функцию *фильтрации* $filter: \wp(Bel) \times \times \wp(Des) \times \wp(Int) \to \wp(Bel)$. В процессе рассуждений дальнейшие желания помещаются в стек намерений, что в свою очередь запускает поиск большего количества планов, чтобы достичь указанной цели, и так далее. Процесс заканчивается отдельными действиями, которые могут быть непосредственно выполнены. Если специфический план достижения цели терпит неудачу, то агент может выбрать другой план достижения желания из всех возможных планов.

Так как размер структуры намерений становится очень большим, ограничения, накладываемые на вычислительные ресурсы для планирования, уточнения и оценки этих намерений, вызывают «познавательную перегрузку» агента BDI. К сожалению, ранее упомянутые механизмы фильтрации, не помогают значительно сократить «познавательную перегрузку» поскольку, со временем генерируется ряд действительных альтернатив, которые все совместимы с текущими намерениями агента.

Другим возможным решением является использование реактивных (поведенческих) агентов [6–9]. В таких агентах не рассматриваются планирующие действия, основанные на сложных внутренних представлениях окружающей среды из-за их неотъемлемых ошибок их четкого представления и ассоциированных с их обработкой временных затрат. Реактивный агент $Ag_r$ производит ответ из множества возможных ответов $R = \{r_1, r_2, ...\}$, на основе текущих стимулов $S$, что может быть представлено следующим отображением: $Ag_r: \beta(S) \to R$. Каждый индивидуальный стимул или результат перцепции $s_i$ (где $s_i \in S$) – кортеж, состоящий из специфического типа или перцепционного класса $p$ и значения силы $\lambda$: $s_i = (p, \lambda)$. Агент $Ag_r$ выбирает текущий ответ $r$) из множества возможных доступных ответов $R$ каждый раз когда $\lambda$ больше, чем порог $\tau$. Функция поведения $\beta$, ответственная за отображение входных стимулов на множество последовательных действий, может быть дискретной или непрерывной. Реактивные агенты могут быть разработаны с использованием итеративного процесса с использованием кооперативных или конкурентных методов композиции отдельно задаваемых функций поведения $\beta_i$ (где $\beta_i \in \beta$). Недостатком таких методов композиции является их сложность. Так, в архитектуре реактивного агента, если задано $n$ уровней поведения и каждый уровень способен предложить $m$ возможных действий, то это означает, что необходимо задать $m^n$ их взаимодействий. Кроме того, стремясь упростить функцию поведения, такие агенты не учитывают возможного состояния самого агента и дальнейшей мотивации.

В качестве возможной альтернативы можно рассматривать мотивированные агенты [10] как подход, совместимый с идеями делиберативных и поведенческих агентов. Различные побуждения (мотивации) – ключевые определяющие факторы, с помощью которых агент может производить разнообразное поведение, имеющее высшую степень выгоды (или полезности) как для агента, так и для мультиагентной системы в целом.

При таком подходе мотив $m$ – это отображение текущих убеждений *bel* агента о состоянии его среды во множество активных побуждений $m : \wp(bel) \to \wp(m)$. Такой агент будет иметь две функции: порождение цели и активация цели, обе из которых предназначены для генерации активных целей в ответ на обнаруженные изменения в его текущих убеждениях. Порожденные цели добавляются ко множеству активных целей агентов: $gen : : \wp(m) \times \wp(bel) \to \wp(mg)$. Активация (вызов) цели происходит, когда интенсивность побуждения связанного с целью превышает определенный *th*-порог: $act : \wp(mg) \times \wp(bel) \times \wp(th) \to \wp(goal)$.

Привлекательность этого подхода по сравнению с реактивными агентами, в том, что пользовательские ожидания относительно поведения агента отображаются в профиль побуждений, который может затем быть сопоставлен с реальным поведением агента и усовершенствован. Кроме того, в отличие от делиберативного метода, такой тип агентов не использует сложную систему активации целей и с ней фильтры или триггеры.

**Нечеткие архитектурные модели мультиагентных систем.** Формальные системы трансформаций нечетких графов (СТНГ) могут использоваться при проектировании нечетких агентов с двумя целями: 1) как обобщение алгоритмов нечеткого логического вывода, в частности, таких, как системы Мамдани и Ларсена над нечеткими графами; 2) с целью спецификации возможных преобразований между моделями, задающими разные уровни абстракции нечетких агентов в подходе MDA (так называемая «вертикальные» трансформации или преобразования «уточнения»).

Каждая продукция СТНГ строится на основе нечетких графов [2, 3], которые представляют собой антецедент и граф-результат продукции. Такая продукция может быть записана следующим образом:

ЕСЛИ нечеткий граф-антецедент ТО нечеткий граф-результат.

Возможность указанного представления продукций СТНГ основана на том факте, что нечеткий граф-антецедент и нечеткий граф-результат (консеквент) могут быть записаны в виде

$$\tilde{N}_1 \& \tilde{N}_2 \ldots \tilde{N}_N \ \& \ \tilde{E}_1 \ \& \ \tilde{E}_2 \ \& \ \ldots \tilde{E}_M ,$$

где $\{N_1, N_2, \ldots, N_N\}$ – множество вершин нечеткого графа; $\{E_1, E_2, \ldots, E_M\}$ – множество ребер нечеткого графа; $\tilde{e}$ – нечеткое множество соответствующей вершины или ребра нечеткого графа. Обозначим принадлежность вершины и ребра графу-антецеденту и графу-консеквента соответственно верхним индексом *A* и *C*.

Таким образом, простая продукция, задающая соответствующее преобразование, может быть записана в виде

$$\text{ЕСЛИ } \tilde{N}_1'^A = \tilde{N}_1^A \,\&\, \ldots \,\&\, \tilde{N}_N'^A = \tilde{N}_N^A \,\&\, \tilde{E}_1'^A = \tilde{E}_1^A \,\&\, \ldots \,\&\, \tilde{E}_M'^A = \tilde{E}_M^A,$$

$$\text{ТО } \tilde{N}_1'^C = \tilde{N}_1^C \,\&\, \ldots \,\&\, \tilde{N}_K'^C = \tilde{N}_K^C \,\&\, \tilde{E}_1'^C = \tilde{E}_1^C \,\&\, \ldots \,\&\, \tilde{E}_L'^C = \tilde{E}_L^C,$$

где $\tilde{N}_i^A$, $\tilde{E}_j^A$, $\tilde{N}_i'^A$, $\tilde{E}_j'^A$, $\tilde{N}_k^C$, $\tilde{E}_l^C$, $\tilde{N}_k'^C$, $\tilde{E}_l'^C$ – дискретные нечеткие множества или нечеткие числа, причем $\tilde{N}_i^A$, $\tilde{E}_j^A$, $\tilde{N}_k^C$, $\tilde{E}_l^C$ – указанные в продукции нечеткие множества, $\tilde{\tilde{N}}_i'^A$, $\tilde{\tilde{E}}_j'^A$ – значения истинности вершин и ребер нечеткого графа к которому применяется продукция, обычно отличающиеся от значений указанных в продукции; $\tilde{\tilde{N}}_k'^C$, $\tilde{\tilde{E}}_l'^C$ – исправленные значения истинности нечетких множеств графа получаемого в результате применения продукции.

Пусть $P$ – значение истинности антецедента продукции, $NA$ – множество вершин, а $EA$ – множество ребер графа-антецедента. Тогда

$$P = \min \begin{cases} \min_{n \in NA}(\max(\min_{\forall x}(\mu'(n,x), \mu'(n,x))), \\ \min_{e \in EA}(\max(\min_{\forall x}(\mu'(e,x), \mu'(n,x))). \end{cases}$$

Реализовано три типа вывода над нечеткими графами: монотонный, при котором последовательные значения истинности вершин и ребер могут только возрастать; немонотонный, при котором последовательные значения истинности могут как расти, так и уменьшаться; нисходящий монотонный, при котором последовательные значения истинности только уменьшатся. Адекватность применения определенного типа вывода зависит от задачи и может отличаться для различных продукций, соответствующим определенным этапам решения.

При монотонном выводе существующие значения истинности вершин и ребер графа не могут быть уменьшены при наличии дополнительных свидетельств. Пусть $NC$ – множество вершин, а $EC$ – множество ребер графа-консеквента. Формула монотонного вывода для значений истинности вершин и ребер, которые добавляются или остаются в соответствии с указанной продукцией:

$$\forall o \in NC \cup EC : \mu'(o,x) = S(P, \mu(o,x)),$$

где функция $S$ – так называемая $S$-норма, например $S(P, \mu(o,x)) = \max(P, \mu(o,x))$.

При немонотонном выводе допускаем, что новые результаты, обеспечиваемые запуском указанной продукцией, надежнее, чем любые существующие свидетельства:

$$\forall o \in NC \cup EC : \mu'(o,x) = P \ .$$

Нисходящий монотонный вывод полезен, когда значения истинности $\mu'(o,x)$ представляет верхний предел возможного значения:

$$\forall o \in NC \cup EC : \mu'(o,x) = T(P, \mu(o,x)),$$

где функция $T$ – любая $T$-норма, например $T(P, \mu(o,x)) = \max(P, \mu(o,x))$, что соответствует оператору вывода Мамдани.

Архитектура нечеткого агента, основанного на мотивации, показана на рис. 1. В данной реализации используются системы СТНГ для того, чтобы непосредственно отображать стимулы *X* на ответы *R*. Выбор этого способа был основан на простоте наличия единственной системы нечетких правил, обеспечивающей функцию отображения $\beta$, так как этот выбор не требует механизма координации. Отсутствие координатора поведения внутри самого агента упрощает его реализацию и делает более очевидным эффект влияния нечетких значений побуждения на результирующее поведение агента.

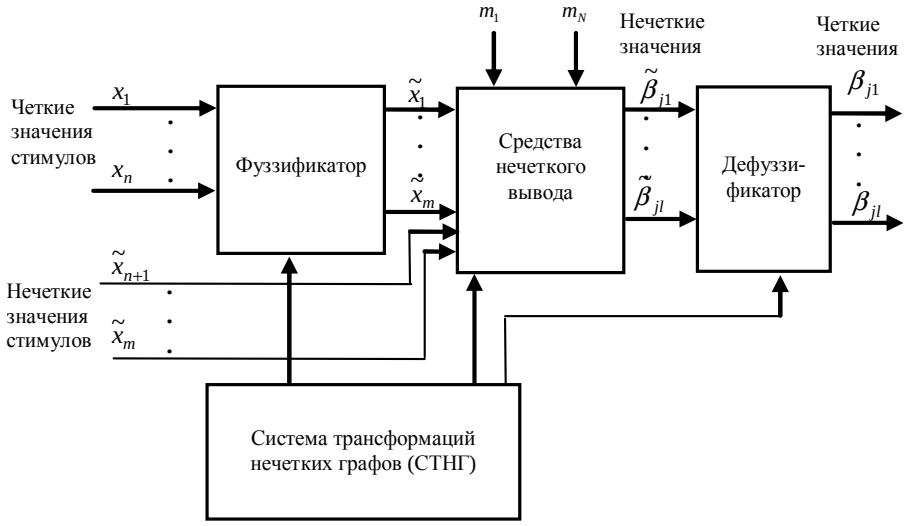


РИС. 1. Структура нечеткого мотивированного агента

В случае четких значений входных стимулов $x = \{x_1,...,x_n\}$ фуззификатор приводит их к соответствующим нечетким значениям. Блок нечеткого вывода осуществляет сопоставление нечетких правил СТНГ не только с нечеткими стимулами $\tilde{x} = \{\tilde{x}_1,...,\tilde{x}_n\}$, но и с нечеткими значениями мотивации $\tilde{m} = \{\tilde{m}_1,...,\tilde{m}_N\}$. Для этого задана проекция нечетких правил нечеткого графа предметной области (ПО) – нечеткое множество побуждений, т. е., отображение, определяющее, какие из вершин графа ПО могут рассматриваться в качестве мотивов (побуждений). Правила СТНГ, в левой части которых находятся нечеткие значения мотивов, рассматриваются как строго мотивированные. При использовании нисходящего нечеткого вывода высота функции принадлежности мотивации в правой части правила может только уменьшаться, что приведет к остановке выполнения СТНГ через несколько итераций ввиду отсутствия необходимых побуждений (мера нечеткого совпадения меньше необходимого порога). Дефуззификатор *j*-го агента отображает множество вершин нечеткого графа ПО, соответствующих нечетким значениям поведения во множество одновременных действий $\beta_j = \{\beta_{j1},...,\beta_{jl}\}$.

**Модели координации и адаптации с учетом нечетких значений полезности.** На рис. 2 показана архитектура нечеткой мультиагентной системы, состоящей из $N$ мотивированных агентов. Все агенты получают одинаковые стимулы $\tilde{x} = \{\tilde{x}_1,...,\tilde{x}_m\}$. Каждый агент осуществляет применение правил СТНГ и вырабатывает решение относительно приемлемого поведения $\beta_j = \{\beta_{j1},...,\beta_{jl}\}$. Выработка окончательного решения осуществляется координатором, задачей которого также является настройка нечетких правил. С этой целью модуль оценки предложений агентов на основе $\beta_1$, $\beta_2$, ..., $\beta_N$, набора соответствующих этим предложениям входных символов и набора мотиваций осуществляет настройку правил СТНГ для каждого агента и передает полученный результат модулю вычисления результата. В результате сопоставления с использованием индексов нечеткого ранжирования вырабатывается окончательное решение мультиагентной системы, означающее «наилучшее» предложенное поведение $\beta_i = \{\beta_{i1},...,\beta_{il}\}$.

В нашей реализации координатора мультиагентной системы начальные установки побуждения используются для того, чтобы на основе системы нечетких правил Такаги – Сугено определить нечеткую пригодность поведения агента в различных средах (рис. 3). Множество побуждений $M$ нечеткого агента для закупки (продажи) товаров и услуг включает: отсутствие у агента необходимого количества товаров (услуг) ($m_1$), избыток товаров ($m_2$), большой срок хранения товаров или пользования услугой ($m_3$) и возможность закупки / продажи у минимального количества поставщиков/потребителей ($m_4$). Эти побуждения используются как входные установки (нечеткие числа) до запуска каждого эксперимента.

Для возможности такой настройки пользователь выбирает значения побуждений ($m_1$, ...,$m_N$), которые должны быть использованы в нечетком вычислении полезности (т. е. побуждения, которые помогают выбрать лучшего агента в популяции) – одну из допустимых сред обучения агента.

Модуль оценивания предложений агентов осуществляет следующие шаги:

– на основе генетического алгоритм генерирует популяцию различных мотивированных агентов;

– каждый агент популяции в свою очередь вырабатывает и выполняет последовательность (набор) действий $\beta_j$ в среде;

– на основе предложенного поведения каждого агента координатор производит набор $N$ значений полезности;

– результирующая нечеткая полезность $F$ вычисляется на основе побуждений $m_1$, ...,$m_N$ и отдельных значений полезности $f_1,...f_N$.

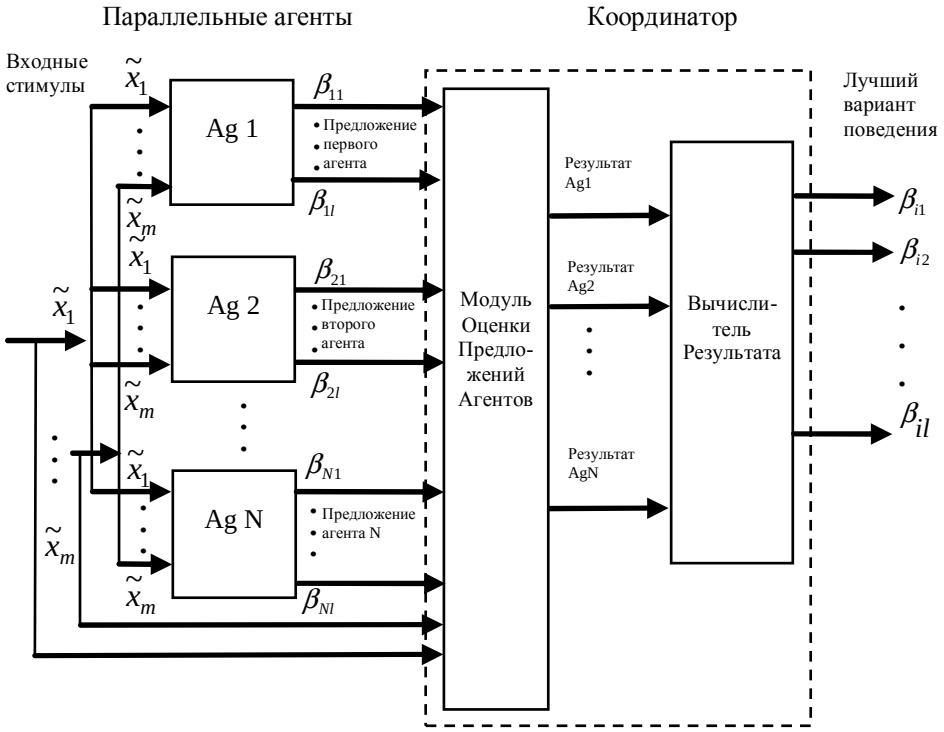


РИС. 2. Структура нечеткой мультиагентной системы с координатором

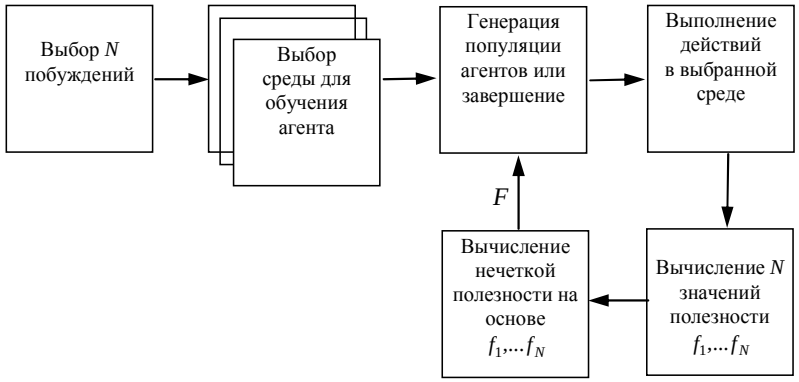


РИС. 3. Схема настройки нечетких правил на основе побуждений и функции полезности

Оценивание заканчивается, когда достигнуто число максимальных повторений генетического алгоритма. Лучший агент в популяции сохраняется как заключительный вариант агента. Для определения нечеткого значения полезности для специфического поведения, используется не требующая дефуззификации система Такаги – Сугено. Результатом каждого правила являются четкие значения, суммируемые как взвешенная средняя величина [11]. При определении

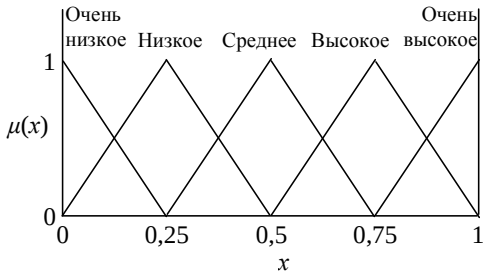


РИС. 4. Нечеткие функции принадлежности

значения полезности для каждого из четырех побуждений торгового агента использованы треугольные функции принадлежности (рис. 4). Таким образом, использованы четыре нечетких переменных с пятью функциями принадлежности каждая ($5^4$ = 625 различных нечетких правила). Псевдокод алгоритма расчета значений нечеткой полезности показан на рис. 5.

*Входные значения*
N : количество мотиваций;
M : количество функций принадлежности каждой из мотиваций;
X[N] : массив заданных мотиваций;
Y[N] : массив значений функций полезности;
C[N] : массив коэффициентов;
μ[N][M] : матрица функций принадлежности для мотиваций;
*Переменные*
w[n] : оцениваемый вес каждого нечеткого правила; f[n] : определяемая полезность;
n, m0, m1, . . . , mN : целые значения;
*Выходные значения*
F : нечеткое значение полезности;
НАЧАЛО
n := 1;
ДЛЯ $m_1, m_2, \ldots, m_N$ ОТ 1 ШАГ 1 ДО *M*
  НАЧАЛО
w[n] := min{μ[1][m1], μ[2][m2], . . . , μ[N][mN]};
  f[n] := $\sum_{i=1}^{N} X[i]Y[i]C[m_i]$;
  n := n + 1;
  КОНЕЦ;
F := ( $\sum_{i=1}^{M^N} w[i]f[i]$ )/( $\sum_{i=1}^{M^N} w[i]$ );
КОНЕЦ

РИС. 5. Алгоритм вычисления нечеткого значения полезности

Для торгового агента определены следующие переменные и соответствующие им критерии: надлежащее окончание действий в состоянии, при котором значение ресурсов (имеющихся товаров или услуг) не меньше установленного для агента уровня ($f_1$), не больше установленного для агента уровня ($f_2$), необходимость предпринимать дальнейшие операции из-за ограничения срока хранения или расходования товаров или ресурсов ($f_3$), количество поставщиков / потребителей, с которыми были проведены операции ($f_4$). Значения для этих критериев нормализованы (содержат нечеткие числа) и вычислены после того, как агент завершает действия.

В процессе обучения, управляемая среда выбрана, и генетический алгоритм случайным образом генерирует начальную популяцию агентов. После этого, каждый агент выполняет свое задание (покупка и продажа) и вырабатывается набор значений полезности, соответствующих выполненному заданию. Наконец, на основе значений побуждения (мотивации) и полученных значений полезности по отдельным критериям, координатор вычисляет значение нечеткой полезности.

**Основные метамодели и трансформации.** В основу моделе-ориентированной разработки нечетких мультиагентных систем положены их метамодели, разработанные с использованием формата 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 XMI , и эффективный интерфейс программиста для операций над объектами Ecore. При этом компонент Ecore Tools обеспечивает среду для операций над моделями, в том числе проверку правильности и создание генераторов.

Основные трансформации моделей описаны с использованием СТНГ в качестве платформно-независимой модели агента, для чего создан специальный графический редактор со средствами навигации по нечетким графам и композиции правил замены (удаление и добавление вершин, выбор начальной и целевой метамоделей и т. д.). Порождение мультиагентной системы состоит из двух этапов: 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.

Для представления СТНГ разработана метамодель Ecore нечетких правил, содержащая все необходимые понятия для конструирования нечетких правил замены. В качестве основных классов (понятий) выбраны FGraph(нечеткий граф), InModel (входная модель), OutModel (выходная модель), FNode и FEdge (нечеткие вершины и ребра). Каждой вершине или ребру соответствует нечеткое значение, связанное в свою очередь с нечеткой переменной, описывающей набор допустимых лингвистических значений и их функций принадлежности. С помощью специальной трансформации СТНГ превращается в модель нечетких правил, называемых также «нечеткими системами», находящихся ближе по уровню к целевой платформе. Различные способы выполнения нечетких правил учитываются с помощью конкретных классов FuzzyRuleExecutor, например класс LarsenRuleExecutor соответствует системе Ларсена. Это позволяет расширять данную метамодель другими видами нечетких систем, описанными в литературе, не затрагивая при этом уровень СТНГ. Фрагмент метамодели, относящийся к представлению нечетких правил, представлен на рис. 6. Модель, соответствующая метамодели нечетких правил, показана на рис. 7.

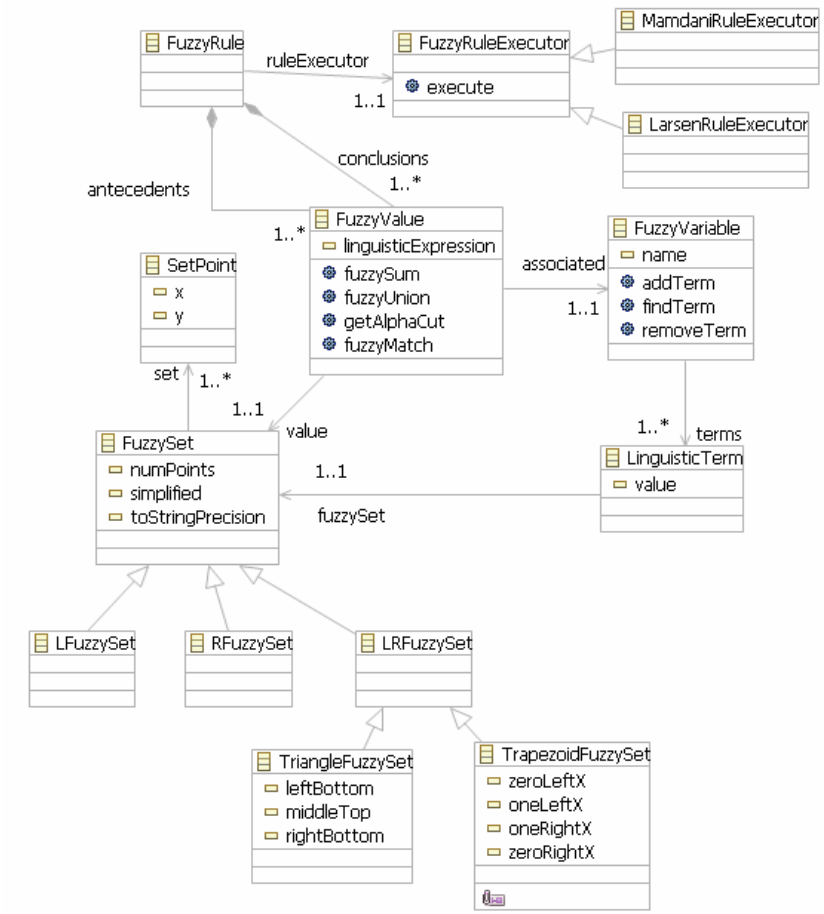


РИС. 6. Фрагмент метамодели для представления нечетких правил

Однако средства представленной выше метамодели находятся достаточно далеко от средств программирования мультиагентной целевой платформы, такой как JADE или ABLE. Поэтому для интеграции нечетких правил в среду выполнения указанных платформ используется преобразование, отображающее нечеткие правила в модель на основе JEM (Java EMF Model), позволяющей представить основные конструкции классов языка Java и платформы J2EE.

Для генерации текстовых файлов программы-агента разработаны преобразования “модель-текст” на основе текстовых шаблонов платформы XPand, запускаемых в контексте компонентов EMFT Workflow. По сравнению с непосредственной генерацией кода на основе СТНГ использование промежуточных моделей значительно упрощает создание текстовых шаблонов.

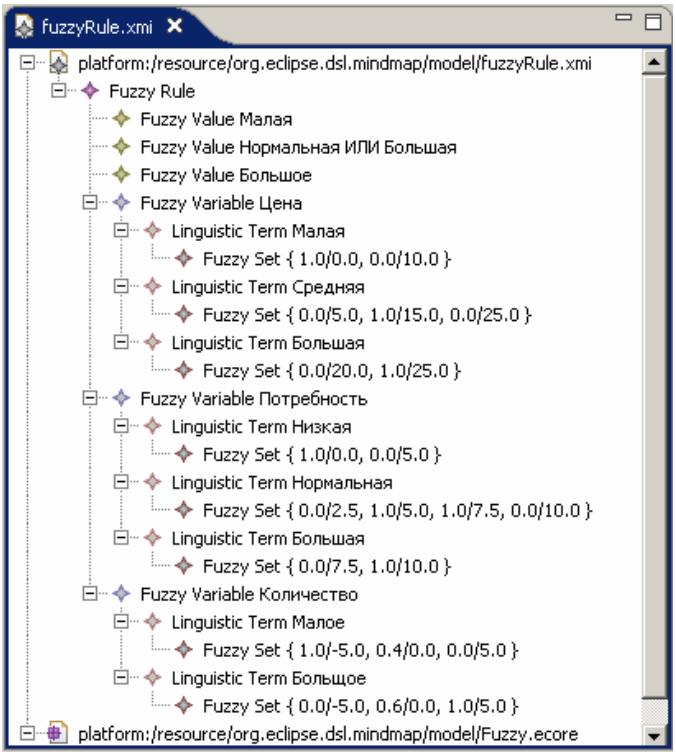


РИС. 7. Модель в формате Ecore, представляющая нечеткое правило

**Выводы**. Таким образом, предложены нечеткие архитектурные модели мультиагентных систем, предназначенных для решения задач в распределенной среде. Такие модели обладают свойствами как реактивных, так и мотивированных делиберативных агентов, что позволяет использовать преимущества обоих подходов. Формализованы средства описания поведения нечетких агентов на основе трансформаций нечетких графов, способы координации нечеткой мультиагентной системы и ее адаптации с учетом нечетких значений полезности. Разработаны метамодели и основные трансформации нечетких мульти-агентных систем на основе использования MDA (Model Driven Architecture, Архитектуры, управляемой моделями).

*І.М. Парасюк, С.В. Єршов*

МОДЕЛЕ-ОРІЄНТОВАНА АРХІТЕКТУРА НЕЧІТКИХ МУЛЬТИАГЕНТНИХ СИСТЕМ

Розглядаються нечіткі архітектурні моделі мультиагентних систем для розподіленого середовища, що мають властивості реактивності і мотивованості. Запропоновані засоби опису поведінки нечітких агентів на основі трансформацій нечітких графів, способи їх координації і адаптації з урахуванням нечітких значень корисності. Розроблені основні метамоделі та трансформації нечітких мультиагентних систем на основі моделе-орієнтованої архітектури.

*I.N. Parasyuk, S.V. Yershov*

MODEL-ORIENTED ARCHITECTURE OF FUZZY MULTIAGENT SYSTEMS

Fuzzy architectural models of multiagent systems for distributed environment that posses properties of reactivity and motivation are investigated. Means of description of fuzzy agents behavior on the basis of transformations of fuzzy graphs, methods of their coordination and adaptation taking into account fuzzy values of utility are proposed. Basic metamodels and transformations of the fuzzy multiagent systems on the basis of model-oriented architecture are developed.

1. *Заде Л.А.* Роль мягких вычислений и нечеткой логики в понимании, конструировании и развитии информационных / интеллектуальных систем // Новости Искусственного Интеллекта. – 2001. – № 2–3. – С. 7 – 11.
2. *Парасюк І.М., Єршов С.В.* Методи аналізу програмних архітектур, представлених нечіткими графовими моделями // Проблеми програмуваннння. – 2006. – № 1–2. – С. 101–110.
3. *Ершов С.В.* Нечеткие графы функциональных зависимостей как основа метамоделирования программных систем // Компьютерная математика. – 2005. – № 3. – С. 139–149.
4. *Ершов С.В.* К проблеме формализации объектно-ориентированных методов разработки программного обеспечения на основе нечеткой логики // Там же. – 2003. – № 2. – С. 62–77.
5. *Рассел С., Норвиг П.* Искусственный интеллект. Современный подход. М.: Вильямс, 2007. – 1408 с.
6. *Wooldridge M.J.* An Introduction to Multiagent Systems. – Cambridge: MIT Press, 2002. – 366 p.
7. *Bussmann S., Jennings N., Jennings N.R., Wooldridge M.J.* Multiagent systems for manufacturing control: a design methodology. – 2004. – 288 p.
8. *Luck M.M., Ashri R., D'Inverno M.* Agent-based software development, 2004. – 208 p.
9. *Anumba C.J., Ugwu O. O., Ren Z.* Agents and multi-agent systems in construction, 2005. – 329 p.
10. *Plekhanova V.* Intelligent agent software engineering. – 2003. – 240 p.
11. *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы. – М.: Горячая линия, Телеком, 2004. – 452 с.

Получено 19.02.2010

***Об авторах:***

*Парасюк Иван Николаевич,*
член-корреспондент НАН Украины, доктор технических наук,
заведующий отделом Института кибернетики имени В.М. Глушкова НАН Украины,
sershv@i.com.ua

*Ершов Сергей Владимирович,*
кандидат физико-математических наук, старший научный сотрудник
Института кибернетики имени В.М. Глушкова НАН Украины.
sershv@i.com.ua