КОМПЬЮТЕРНАЯ МАТЕМАТИКА

УДК 681.3.06

С.В. ЕРШОВ

ТРАНСФОРМАЦИОННЫЙ ПОДХОД К РАЗРАБОТКЕ АДАПТИВНЫХ ИНТЕЛЛЕКТУАЛЬНЫХ АГЕНТОВ НА ОСНОВЕ НЕЧЕТКИХ СХЕМ ПЕРЕХОДОВ

Рассматривается трансформационный подход к разработке интеллектуальных агентов на основе моделей нечетких схем переходов. Описаны основные преобразования, позволяющие сгенерировать нечеткие платформнонезависимые модели внутриагентного управления. Разработана эволюционная стратегия, позволяющая настроить значения принадлежности нечеткой схемы переходов. Построены критерии для оценки степени соответствия таких схем обучающим примерам.

Наблюдаемый в последнее время рост интереса к возможностям мультиагентных интеллектуальных систем в контексте разработки программного обеспечения ставит задачу создания методов и средств моделе-ориентированной трансформационной разработки (Model-Driven Engineering, MDE) таких систем. Хотя многие известные методологии разработки мультиагентных систем, такие как Tropos [1] и INGENIAS [2], предлагают методы и инструменты для автоматизации модели трансформаций. Они предназначены для использования только на некоторых этапах разработки и не могут рассматриваться как систематические.

Отдельным агентам таких распределенных систем открыт доступ только к их собственным состояниям, а не к глобальному состоянию всей системы, о котором агент может делать только предположения. К тому же любая среда передачи асинхронных сообщений между агентами не является абсолютно надежной и приводит к запаздыванию сообщений, что еще больше увеличивает степень неопределенности схем состояний агента. Поэтому работой таких агентов можно управлять только с учетом степени достижимости таких «размытых» состояний, для чего обоснованным представляется использование нечеткой математики.

Цель настоящей статьи – исследование и обоснование трансформационного подхода к разработке интеллектуальных агентов с использованием нечетких схем переходов, и

разработка эволюционной стратегии обучения нечетких схем переходов на примерах, что позволяет придать создаваемым с помощью такого подхода агентам свойство адаптивности. Перечисленные проблемы тематически вписываются в дальнейшее развитие исследований в направлении становления моделе-ориентированных архитектур программных систем в нечетком представлении [3–5].

Моделе-ориентированная разработка мультиагентных систем в значительной степени зависит от последовательной трансформации их моделей [6]. При этом наиболее важными являются способы преобразований моделей:

• Преобразование модель-модель (M2M). Этот вид преобразования используется для преобразования одного типа графической модели (схемы, диаграммы) к другому типу графической модели. Преобразование M2M основано на исходной и целевой метамоделях и определяет преобразования элементов исходной модели в элементы целевой модели.

• Преобразования модель-текст (M2T). Такие преобразования используются для трансформации визуального представления в код программы; синтаксис целевого языка определяется вместе с метамоделью графической модели.

Данные трансформации разработаны на основе фреймворка Eclipse Modeling Framework [9], использующего для построения моделей метамодель Ecore. Она определяет, что модель состоит из экземпляров типа EClass, которые могут иметь атрибуты (экземпляры типа EAttribute) или ссылки на другие экземпляры EClass (через тип EReference). Наконец, атрибуты могут быть различными экземплярами EDataType (например, целыми числами, строками, действительными числами и т. д.).

Аналогичная технология для представления и обработки метамоделей – Meta-Object Facility (MOF) [10]. Тем не менее, метамодель EMF проще, чем метамодель MOF с точки зрения понятий, свойств и внутренней структуры. Таким образом, отображение понятий EMF в понятия MOF может быть проведено относительно просто.

Исходной при трансформационной разработке агентов может служить их ролевая модель (РМ). Предполагается что интеллектуальный агент обязывается выполнить одну или несколько ролей в течение своего жизненного цикла. Одной из особенностей 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. Метамодель ролевой модели определяет понятие *Роль*, которое ссылается на следующие понятия:

• деятельность, которая включает два атрибута, имя (название) и функциональность (описание того, что эта деятельность делает);

• возможность, относящаяся к направлениям деятельности, на которые она ссылается, достигающим высокоуровневой цели;

• протокол, которому приписывается имя.

Трансформация ролевой модели (РМ) в выходную модель – автоматизированная задача, которая создает несколько начальных моделей МВУ на основе ранее созданной РМ, по одной для каждой роли. Для разработки трансформации модель-модель (M2M) использован язык Query/View/Transformation (QVT).

Многие методологии разработки (Tropos или INGENIAS [1, 2]) 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 ментальное состояние (например, убеждение-желание-намерение) агентов.

Метамодель МВУ (рис. 1) определяет понятие *Модель*, которая включает нечеткие состояния, переходы и значения. Имена модели должны удовлетворять современному формату пространства имен пакетов Java или C#. Нечеткие состояния содержат следующие атрибуты: имя узла, тип узла, соответствующий типу состояния в схеме состояний, как правило, один из И, ИЛИ, НАЧАЛО, КОНЕЦ, метка узла, и деятельность, связанная с узлом.

Состояния и переходы также ссылаются на нечеткие значения, которые в свою очередь включают атрибут «значение» в диапазоне [0, 1]. Следующим понятием, определенным в этой метамодели, является «Переход», который, кроме ссылки на нечеткое значение, включает также следующие атрибуты:

• имя – обычно в форме <метка исходного узла> <метка целевого узла>

• выражение перехода (ВП), через которое определяется основная управляющая информацию в МВУ. Это выражение содержит условия и события, которые делают переход возможным. Кроме того, может быть использовано получение или передача сообщений между агентами (в случае взаимодействующих агентов);

• источник (исходное состояние), цель (целевое состояние).

Таким образом, каждая модель МВУ уточняется, определением условий и события приема / передачи сообщений, которые позволят переходы от одной деятельности (задачи) к другой.

Модель 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), 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*.

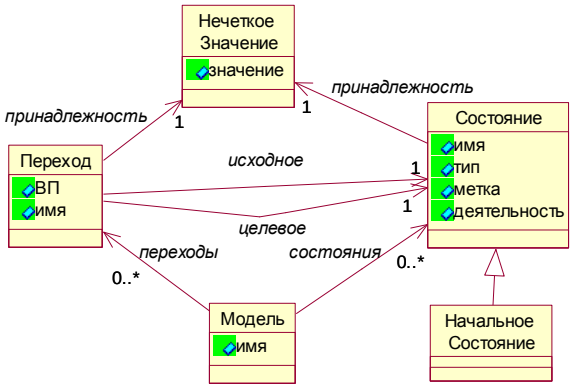


РИС. 1. Метамодель внутриагентного управления

Способ представления принадлежностей переходов фактически определяет структура нечеткой схемы переходов. Пусть такая схема включает *n* состояний, степень принадлежности ее переходов может быть представлена в виде трехмерной матрицы

$$P=\begin{bmatrix} p_{11}, p_{12},..., p_{1n} \\ p_{21}, p_{22},..., p_{2n} \\ ..... \\ p_{n1}, p_{n2},..., p_{nn} \end{bmatrix}, \qquad (1)$$

где $p_{ij} = [\mu_{ij}^1, \mu_{ij}^2, ..., \mu_{ij}^m]$ – вектор степени принадлежности переходов между вершинами *i* и *j*, размеченных соответственно символами a^k, причем $\mu_{ij}^k \in [0,1]$ и символы в разметке не повторяются. Ненулевые значения μ_{ij} соответствуют переходам состояний $q_j = \delta(q_i, a_{ij}^k, \mu_{ij}^k)$, в то время как $\mu_{ij} = 0$ означает, что между двумя состояниями перехода нет. Правильная схема переходов должна удовлетворять следующему условию: в одном и том же векторе символы событий отличаются друг от друга ($a^k \neq a^l$ для всех $k \neq l$) и принадлежат конечному алфавиту событий *A*.

Определим нечеткое (размытое) состояние как вектор пар

$$Q = [< q_1, \mu(q_1) >, < q_2, \mu(q_2) >, ..., < q_n, \mu(q_n) >], \qquad (2)$$

второй компонент которых $\mu(q_i)$ – степень принадлежности агента данному состоянию. Реакцией (выполнением) схемы переходов в ответ на наступление события a^k является новое состояние

$$Q_{Q\otimes P} = [< q_1, \mu_{Q\otimes P}(q_1) >, < q_2, \mu_{Q\otimes P}(q_2) >, ..., < q_n, \mu_{Q\otimes P}(q_n) >],$$

такое, что

$$\mu(q_i) = \max\{\mu(q_j) * \mu_{ji}^k\} \ (\forall < q_j, \mu(q_j) > \in Q).$$

Последовательное выполнение (поведение), задаваемое парой событий a^k, a^l и схемой нечетких переходов, соответствует max-* композиции нечетких отношений $P \otimes P$:

$$\mu_{ij} = \max\{\mu_{ir}^k * \mu_{rj}^l\} \ (\forall \mu_{ir}^k \in p_{ir}, \mu_{rj}^l \in p_{rj}, \ r = 1,...n).$$

Деятельности, задаваемые схемой с *n* состояниями, непосредственно представлены вектором

$$D=[d_1, d_2, \ldots, dn],$$

где d_j является деятельностью, выполняемой в *j*-м состоянии. Как правило, выполняется только деятельность, степень принадлежности состояния которой наибольшая среди всех состояний.

Таким образом, любая нечеткая схема переходов с *n* состояниями может быть представлена матрицей принадлежности *n*n* (1) и вектором состояния (2).

Одна из интересных возможностей настройки нечеткой схемы переходов – возможность ее обучения на основе представленных примеров и специально разработанной эволюционной стратегии.

В настоящее время известно несколько эволюционных стратегий (ЭС). Два наиболее широко используемых подхода к ЭС обозначаются как $(\mu + \lambda)$ и (μ, λ) [7]. Первый выбирает μ лучших особей (закодированных решений задачи) как среди μ родителей, так и λ потомков в качестве родителей следующего поколения, а второй выбирает лучшие μ особи только из λ потомков. Считается, что стратегия (μ, λ) превосходит $(\mu + \lambda)$, поскольку (μ, λ) имеет большую вероятность достижения глобального оптимума.

Разработанный алгоритм ЭС задан с использованием следующих обозначений:

$$(\mu, \lambda) = (I, \mu, \lambda, m, s, \sigma, f),$$

где I – вектор действительных чисел, представляющих особи в популяции; μ и λ – количество родителей и потомков соответственно; σ является параметром для управления шагом; m представляет собой оператор мутации, являющийся основным оператором в механизме ЭС. При этом изменяются (мутируют) не только векторы, задающие особей, но и параметр σ, определяющий размер шага. Параметр *s* означает метод выбора (в данном случае родители будут выбраны только из λ потомков); f – целевая функция, значение которой должно быть сведено к минимуму.

Параметр управления размером шага эволюции принадлежности изменяется пропорционально расстоянию между пригодностью f_c, порожденной лучшим текущим представителем в популяции и его ожидаемым значением f_e:

$$\delta = \frac{f_e - f_c}{f_c} * \tau,$$

где τ – константа.

Эволюция матрицы переходов осуществляется следующим образом. Случайным образом генерируется целое число: $h = [(\delta * \mathrm{rand}(0,1)]$, генерация повторяется, пока оно не станет таким, что $0 \le h \le n$; это значение представляет количество строк, которые должны быть изменены. При этом rand(0,1) – случайное значение, генерируемое в интервале [0, 1], [x] – округление полученного значения x до ближайшего целого. Параметр δ управляет размером шага мутации матрицы принадлежности: если текущая пригодность находится далеко от ожидаемой, δ является большим и, следовательно, больше шаг мутации матрицы.

Чтобы избежать слишком ранней остановки ЭС, положим $h = [\delta * \mathrm{rand}(0,1)] + 1$, когда $e1 < f_c < e2$. Наконец, когда $f_c < e1$, начинается процесс тонкой настройки:

- случайным образом генерируется целочисленный вектор $B = (b_{1j})$, $j = 1,\ldots h$, где b_{1j} $(1 \le b_{1j} \le n)$ представляют собой индекс строки для последующей мутации на j-м шаге;
- аналогично генерируется вектор $C = (c_{j1})$, $j = 1,\ldots h$, j-й элемент $c_{j1}(1 \le d_{j1} \le n)$ указывает, что вектор в колонке c_{j1} и строке b_{1j} коммутирует с первым ненулевым вектором в той же строке;
- соответствующие элементы матрицы принадлежности заменяются в соответствии с индексами, полученными выше.

Приведем пример для иллюстрации оператора мутации: пусть $P = (p_{ij})$, n=5, h=2, B=[2 4], $C = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$, тогда мутация включает два этапа: p_{23} обменивается с первым ненулевым элементом в строке 2 и p_{42} коммутирует с первым ненулевым вектором в строке 4.

Вектор начального состояния эволюционирует следующим образом:

$$\mu'(q_i) = \mu(q_i)(+-)\eta * \mathrm{rand}(0,1), \cdots, \; i = 1, 2, \ldots, n, \qquad (3)$$

причем знак в формуле (3) выбирается случайным образом, а η – параметр управления размером шага. Если измененное значение принадлежности вне ожидаемого диапазона, повторяется следующий процесс, пока не станет выполняться $0 \le \mu'(q_i) \le 1$: если $\mu'(q_i) > 1$, то $\mu'(q_i) = \mu(q_i) - \eta * \mathrm{rand}(0,1)$, если $\mu'(q_i) < 0$, то $\mu'(q_i) = \mu(q_i) + \eta * \mathrm{rand}(0,1)$.

Пригодность нечетких схем переходов оценивается по двум критериям пригодности, которые оценивают: согласованность выполнения нечеткой схемы переходов с обучающими примерами (f_E) и степень обобщения сгенерированной схемы (f_g):

$$f_E = \sum_{i=1}^{N} (T_i - O_i)^2,$$

где T_i – желаемое значение принадлежности *i*-го примера; O_i – фактическое значение, когда последовательность событий *i*-го примера используется для выполнения сгенерированной схемы переходов; N – количество обучающих примеров.

$$f_g = K/V,$$

где K – число строк в проверочных примерах, которое схемы переходов могут правильно распознать; V – количество тестовых примеров.

Приспособленность порожденных схем переходов оценивается следующей целевой функцией:

$$f = \alpha f_E - \beta f_g,$$

где α – пропорциональный коэффициент согласованности; β – коэффициент способности обобщения в функции пригодности.

Наглядной иллюстрацией данного подхода служит разработка модели МВУ агента-координатора для осуществления протокола «контрактной сети» [8], используемого при разработке мультиагентных систем для закупки / продажи материальных ценностей. Нечеткая схема переходов, служащая основой для создания МВУ, содержит состояния, три из которых ($q1$, $q2$ и $q3$) соответствуют незавершенному статусу взаимодействия агентов, остальные – различным итогам завершенного взаимодействия. Использованы следующие сообщения: *cfp* – поступление заявки, *refuse* – отказ от обслуживания заявки, *propose* – ответное предложение, уточняющее способ удовлетворения заявки, *cancel* – отмена обслуживания, *reject* – отклонение предложенной заявки, *accept* – принятие предложения, *failure* – невозможность выполнения принятой заявки, *inform* – заключение соглашения.

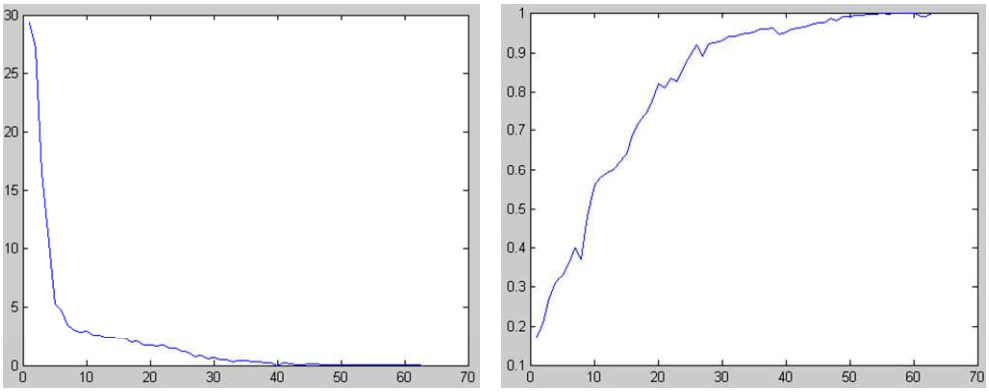
Для обучения нечеткой схемы переходов использовалось множество из 80 примеров, причем первые 40 примеров использовались для оценки согласованности, а остальные – для получения степени обобщения. Примеры поведения агента отсортированы в зависимости от их длины. Любой пример состоит из пары $(P_i, \mu(q_i))$, где P_i – последовательность, образованная символами алфавита событий A, а $\mu(q_i)$ – степень принадлежности результирующего состояния q_i нечеткому состоянию, получаемому после выполнения схемы переходов.

Цель обучения – используя ЭС найти полную нечеткую схему переходов, наиболее адекватно соответствующую приведенным примерам. При этом использовались следующие параметры ЭС: количество родителей μ=4, каждый из

родителей генерирует 7 потомков; константа τ имеет значение 0,2; управляющие параметры $e1 = 0{,}5$; $e2 = 5{,}3$; размер шага управления мутации выхода $h = 0{,}08$; соразмерные коэффициенты $\alpha = 1$, $\beta = 0{,}3$; ожидаемая пригодность установлена в $1{,}7 \times 10^{-5}$. На рис. 2, а показана эволюция приспособленности для каждого из 63 поколений. В поколении 63 выполнение порожденной схемы переходов почти полностью соответствует всем обучающим примерам и тестовым проверкам. На рис. 2, б показана эволюция степени обобщения. Результирующая схема переходов после обучения показана на рис. 3.

Независимые от платформы модели МВУ должны быть преобразованы в зависимую от платформы модель и исполняемый код. Трансформация МВУ в код агента для целевой платформы – автоматизированное преобразование «модель-текст», создающее класс агента на языке C++ (или на языке Java) и несколько классов Behaviour для каждой модели МВУ. В качестве целевых платформ выполнения сгенерированных мультиагентных систем используются кластерная система СКИТ-3 и платформа JADE.

Для разработки трансформаций в платформнозависимый код был использован язык Xpand, предлагаемый платформой Eclipse. Преимущество Xpand – это то, что он является независимым от исходной модели, т. е. 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.



а б

РИС. 2. Эволюция значения приспособленности – а;
эволюция степени обобщения как фактора приспособленности – б

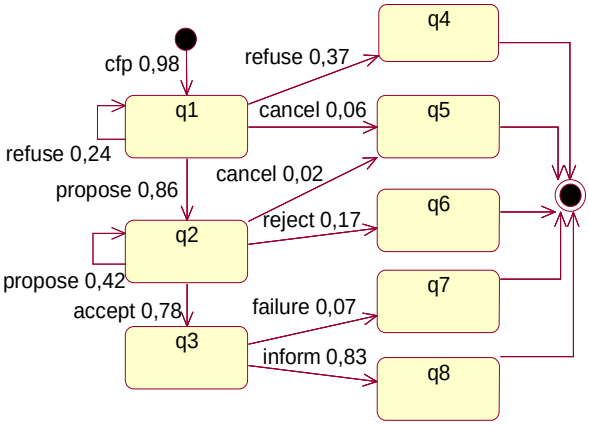


РИС. 3. Нечеткая схема переходов, полученная на основе эволюционной стратегии

Выводы. Таким образом, предложен трансформационный подход к разработке интеллектуальных агентов на основе моделей нечетких схем переходов. В рамках моделе-ориентированной архитектуры разработаны основные преобразования, позволяющие сгенерировать нечеткие платформно-независимые модели внутриагентного управления. Создана эволюционная стратегия, позволяющая настроить значения принадлежности нечеткой схемы переходов, и построены критерии оценки степени соответствия таких схем обучающим примерам.

Предложенный трансформационный подход охватывает все основные фазы разработки мультиагентных систем (от требований до реализации), переход одной фазы в другую осуществляется с помощью преобразований моделей, основу которых составляют нечеткие схемы. В результате модели каждой из фаз обогащают информацией, постепенно приводящей к реализации адаптивных интеллектуальных агентов.

С.В. Єршов

ТРАНСФОРМАЦІЙНИЙ ПІДХІД ДО РОЗРОБКИ АДАПТИВНИХ
ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ НА ОСНОВІ НЕЧІТКИХ СХЕМ ПЕРЕХОДІВ

Розглядається трансформаційний підхід до розробки інтелектуальних агентів на основі моделей нечітких схем переходів. Описано основні перетворення, що дозволяють згенерувати нечіткі платформно-незалежні моделі внутрішньоагентного управління. Розроблена еволюційна стратегія, що дозволяє настроїти значення належності нечіткої схеми переходів. Побудовано критерії для оцінки ступеня відповідності таких схем навчальним прикладам.

S.V. Yershov

TRANSFORMATIONAL APPROACH TO DEVELOPING ADAPTIVE INTELLIGENT AGENTS BASED ON FUZZY TRANSITION SCHEMES

Transformational approach to developing intelligent agents models based on fuzzy transition schemes is considered. Basic transformations that allow to generate fuzzy platform-independent models for intra-agent control are described. An evolutionary strategy to adapt the membership values of fuzzy transition scheme is developed. Criteria for evaluation of degree of adequacy of such schemes to learning examples is constructed.

1. *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.
2. *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.
3. *Парасюк И.Н.*, *Ершов С.В*. Нечеткие модели мультиагентных систем в распределенной среде // Проблеми програмування. – 2010. – № 2–3. – С. 330–339.
4. *Парасюк И.Н.*, *Ершов С.В*. Моделе-ориентированная архитектура нечетких мультиагентных систем // Компьютерная математика. – 2010. – № 2. – С. 139–149.
5. *Ершов С.В*. Принципы построения нечетких мультиагентных систем в распределенной среде // Там же. – 2009. – № 2. – С. 54–61.
6. *Sendall S.*, *Kozaczynski W.* Model transformation: the heart and soul of model-driven software development // IEEE Software. – 2003. – **20**, N 5. – P. 42–45.
7. *Beyer H.-G.*, *Schwefel H.-P*. Evolution Strategies: A Comprehensive Introduction // J. Natural Computing. – 2002. – **1**, N 1. – P. 3–52.
8. *Wooldridge M.J.* An Introduction to Multiagent Systems. – Cambridge: MIT Press, 2002. – 366 p.
9. *Eclipse* Modeling Framework. – http://wiki.eclipse.org/EMF/
10. *OMG* MetaObject Facility. – http://www.omg.org/mof/

Получено 10.11.2010

Об авторе:

Ершов Сергей Владимирович,
кандидат физико-математических наук, старший научный сотрудник
Института кибернетики имени В.М. Глушкова НАН Украины.
sershv@ukr.net