ПРОГРАММНО-ТЕХНИЧЕСКИЕ КОМПЛЕКСЫ

И.Н. ПАРАСЮК, С.В. ЕРШОВ

УДК 681.3:517.11

ТРАНСФОРМАЦИОННЫЙ ПОДХОД К РАЗРАБОТКЕ ПРОГРАММНЫХ АРХИТЕКТУР НА ОСНОВЕ НЕЧЕТКИХ ГРАФОВЫХ МОДЕЛЕЙ

Ключевые слова: *трансформационный подход, нечеткие графы, трансформации графов, архитектура, управляемая моделями, теория категорий.*

ВВЕДЕНИЕ

Первые работы, оказавшие влияние на становление и развитие отдельных аспектов подхода к разработке программных систем на основе трансформации их формальных информационных моделей, появились в 70-х годах прошлого века, в частности, в [1] представлены формальные механизмы композиционного программирования, в [2] проектирование программных систем путем целенаправленной трансформации сложных структур данных обеспечивает метод формализованных технических заданий, в [3] для большей технологичности и гибкости операций проектирования, обеспечения адекватности сервисных и прикладных функций предложена двухуровневая моделе-ориентированная архитектура программных систем, использование которой регламентирует метод Сигма-Дельта, в [4] графовая модель положена в основу представления информационных и логических связей программных систем.

Особую актуальность данное направление приобретает в связи с появлением спецификаций архитектуры программных систем, управляемой моделями, — MDA (Model Driven Architecture) [5, 6], которая предоставляет ряд преимуществ по сравнению с существующими методиками: упрощение разработки многоплатформных систем, простота смены технологической платформы, повышение скорости разработки и качества программ.

В основе MDA лежат понятия платформно-независимой и платформно-зависимой моделей — ПНМ (PIM, platform-independent) и ПЗМ (PSM, platform-specific model). ПНМ определены на более высоком уровне абстракции, чем ПЗМ. Независимая от платформы модель — представление системы в независимом от платформы виде [5, 6], в то время как платформно-зависимая определена как вид системы относительно определенной платформы. Использование этих понятий позволяет исключить технологические и технические детали, несущественные для фундаментальной функциональности системы (или ее части).

Разделение платформно-зависимой и платформно-независимой моделей обеспечивает ряд преимуществ по сравнению с традиционным подходом. В частности, облегчается перенос программного обеспечения (ПО) на другую платформу и его модификация, так как при этом можно использовать прежнюю платформно-независимую модель и разрабатывать заново только платформно-зависимую.

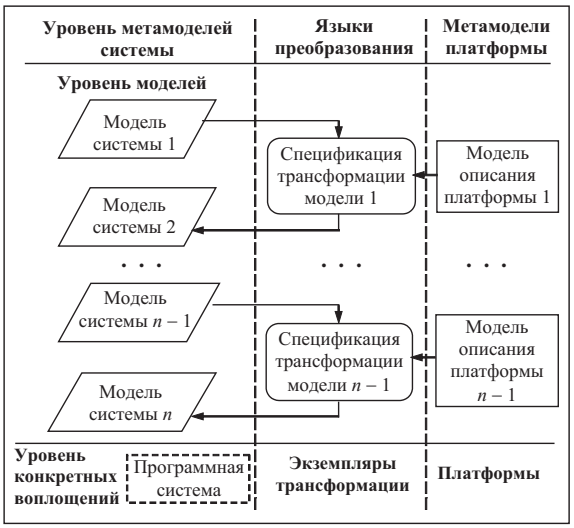


Рис. 1. Схема трансформации моделей

Отличительное свойство моделе-ориентированного подхода состоит в том, что с его помощью можно ускорить разработку ПО, несмотря на то, что нужно создать две модели вместо одной. Это достигается за счет автоматизированной генерации ПЗМ по ПНМ. Процесс перехода к ПЗМ, основанной на конкретной технологической платформе, может быть в значительной степени формализован с помощью трансформаций моделей (рис. 1). При этом модели трансформации используют знания о возможностях платформ, содержащиеся в моделях платформ — исходной и целевой (с большим индексом).

Основная операция, которая применяется над моделями в MDA, — трансформация моделей. Это отображение одного набора моделей на другой согласно спецификации трансформации, в которой отображение, заданное на уровне метамоделей платформ, определяет соответствия между элементами в начальных и целевых моделях. Процесс трансформации принимает ПНМ как вход и генерирует ПЗМ в качестве выходных данных.

Если удастся задать подобное отображение таким образом, чтобы оно выполнялось автоматически, то не потребуется создавать ПЗМ вручную и процесс разработки существенно ускорится, поскольку значительное количество элементов, специфичных для технологии реализации, будет вноситься в модель автоматически.

Программная архитектура, составляющая основу моделе-ориентированного подхода, — выражение важнейших знаний о программной системе, которые позволяют использовать на практике наиболее эффективный способ проектирования системы с учетом определенных ограничений [7]. Особый вид программной архитектуры возникает при разработке программных систем на основе сервисно-ориентированной архитектуры (SOA) — распределенной сетевой архитектуры, в которой четко отделяются обеспечиваемые сервисы от объектов, потребляющих эти сервисы.

Подход MDA не зависит от инструментов и языка моделирования. Для описания моделей программных архитектур при разработке ПО с применением моделе-ориентированного подхода будет использоваться язык моделирования UML [7].

Существует ряд подходов, пригодных для трансформации моделей: реляционный, прямое манипулирование, трансформации графов [8] и др. Так как и ПНМ, и ПЗМ — модели, представленные на языке UML, переход между ними, по сути, является трансформацией UML-модели по заданной спецификации трансформации (содержащей формальное описание преобразования UML-модели общего вида к конкретной платформе реализации). В частности, можно представить UML-модель в виде графа и использовать математический аппарат трансформации графов [8].

Поскольку информация относительно элементов моделей ПО и их отношений в реальных системах часто неопределенная или неоднозначная, возникает необходимость применения моделей на основе нечетких графов. Нечеткие графовые модели могут использоваться непосредственно при описании программной архитектуры, составляющей основу трансформационного похода, принятии решений относительно выбора трансформации на основе нечеткого представления характеристик архитектур ПО.

Цель данной статьи — разработка теоретических принципов, методов и средств создания программных систем на основе моделе-ориентированного трансформационного подхода с использованием нечетких графовых моделей программных архитектур, обоснование концепции нечетких «пространств» для спецификации и оценивания программных архитектур, а также описание возможностей инструментальных средств трансформации моделей ПО, представленных нечеткими графами.

НЕЧЕТКИЕ ПРОСТРАНСТВА СПЕЦИФИКАЦИИ И ОЦЕНИВАНИЯ АРХИТЕКТУР

Основное назначение современных средств спецификации и оценивания программных архитектур — учет их адаптивности и эволюции во время разработки и сопровождения. Существующие методы трансформации в MDA не обеспечивают в полной мере идентификации альтернативных преобразований, которые возникают во время эволюции, и их сравнения на основе определенных качественных характеристик результирующих моделей. Учет основных источников нечеткости информации (несовместимость, неточность, неопределенность, неуверенность, неоднозначность) в процессе трансформации моделей приводит к необходимости построения нечеткого пространства моделирования и разработке подхода к спецификации и оцениванию архитектур на его основе [9, 10].

MDA определяет только одно измерение классификации для моделей, основанное на различии уровней абстракции ПНМ и ПЗМ. Очевидно, что оценивание функциональных аспектов (управление параллелизмом, безопасность, сетевая распределенность и обработка ошибок) проводится на основе большего набора критериев, каждый из которых задает отдельное измерение.

Пространство моделирования — многомерное пространство над рядом независимых измерений, каждое из которых формирует отдельное множество координат, а каждая точка в таком пространстве представляет трансформацию, применяемую к модели, являющейся экземпляром целевой метамодели. Каждый элемент в начальной модели определяет одно измерение в пространстве моделирования. Элемент модели, который определяет измерение, — экземпляр конструкции из начальной метамодели. Для этой конструкции определен набор элементов из целевой метамодели, который используется для формирования множества координат измерения. Конструкция из начальной метамодели может отображаться на каждую конструкцию в этом наборе. Точки в пространстве моделирования интерпретированы как альтернативные преобразования начальной модели. Для каждой начальной модели целевая модель — точка над измерениями, соответствующими начальной модели, которая представлена как кортеж с компонентами для каждого измерения и координатой в нем:

$$(d_1.cd_1, d_2.cd_2, \ldots, d_n.cd_n).$$

Здесь d_i — имя измерения, cd_i — координата точки в этом измерении.

Рассмотрим измерение для элемента $a1$ в исходной модели. Пусть элемент $a1$ — экземпляр конструкции $A1$ исходной метамодели. Конструкция $A1$ может быть отображена на элементы $B1, B2$ или $B3$ в целевой метамодели, что определяет множество координат $\{B1, B2, B3\}$ для измерения $a1$. Поскольку $a2$ также экземпляр $A1$, такое же множество координат задано для измерения $a2$. Допустим, элемент $a3$ исходной метамодели отображен на одну из конструкций $B2, B3$ или $B4$, тогда множество координат для измерения $a3$ — $\{B2, B3, B4\}$. Описанное пространство моделирования показано на рис. 2.

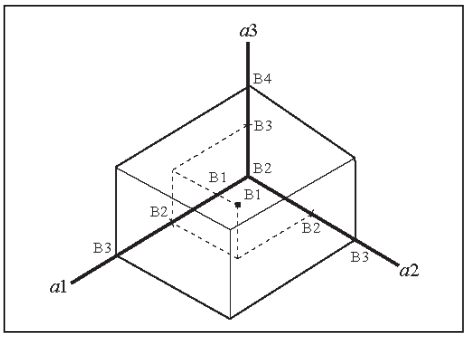


Рис. 2. Пример «пространства» моделирования для исходной модели

В процессе моделе-ориентированной разработки на основе пространства моделирования выбираются альтернативные трансформации для представленной модели. Разные требования относительно результирующих целевых моделей приводят к функционально эквивалентным реализациям, которые отличаются атрибутами качества [11].

Очевидно, что средства представления моделей в нечетком пространстве моделирования позволяют более адекватно представлять и оценивать программные архитектуры, чем двузначная характеристика отдельных показателей функционирования.

Нечеткий модуль может быть представлен в нечетком пространстве целевой модели как $[T,(\mu_T,V_M),(P_1,V_1),(P_2,V_2),\ldots,(P_n,V_n)]$, где значение функции принадлежности V_M зависит от отдельных трансформаций, которые генерируют указанный модуль на основе начальной модели. Свойства $P_1,P_2,\ldots,P_n$ задают характеристики модуля по каждому отдельному измерению. Например, значение функции принадлежности модуля зависит от значений принадлежности свойств «Расширяемость» и «Автономность». Нечеткие лингвистические термы $V_1,V_2,\ldots,V_n$, такие как «слабая», «незначительная», «средняя», «существенная», «значительная», могут использоваться для уточнения координат модуля в пространстве моделирования.

В случае учета нескольких показателей используются нечеткие продукции, которые на основе значений показателей $V_1,V_2,\ldots,V_n$ выдают нечеткую оценку V_M, которую можно интерпретировать как интегральный показатель применимости трансформации в нечетком пространстве.

Один из общепринятых подходов к моделированию объектно-ориентированных структур, описывающих программную архитектуру, — использование языка UML для моделирования таких структур на основе графов [9].

Построение и теоретическое обоснование алгоритмов и вычисляемых объектов теоретико-категорными средствами — одно из современных направлений computer science [12]. Поэтому для формализации нечеткости основных понятий, которые входят в состав нечетких графов [8, 13, 14], использован категорный подход как раздел математики, изучающий наиболее общие свойства отношений между математическими объектами, независимые от внутренней структуры объектов.

Любой нечеткий граф построен на нечетких точках типа ЕСЛИ $x_1 = A_1$ И $\ldots x_n = A_n$, ТО $y = B$. Он может быть представлен как суперпозиция m нечетких точек соотношением

$$f^* = \sum_{j=1}^{m} (A_j \times B_j).$$

Рассмотрим нечеткий граф, построенный на нечетких точках. В случае $n=1$ получаем бинарный нечеткий граф $\langle A,\alpha\rangle$, т.е. пару, состоящую из нечеткого множества A, элементы которой являются объектами категории **Fuzz**([0,1]), и нечеткого отношения α: $A \to A$. При этом множество истинности — интервал [0,1], а множество носителя — вершины графа.

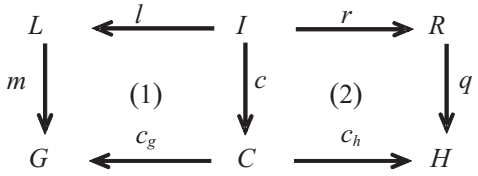
Морфизмом нечетких графов $f: A \to B$ называется пара $f = \langle f_E : E_A \to E_B,\ f_N : V_A \to V_B \rangle$ морфизмов (определенных над дугами E и вершинами N нечеткого графа соответственно), таких, что $f_N \circ s_A = s_B \circ f_E \wedge f_N \circ t_A = t_B \circ f_E$. Нечеткие графы вместе с множеством их морфизмов и определенной покомпонентно композицией морфизмов $g \circ f = \langle g_E \circ f_E,\ g_N \circ f_N \rangle$ образуют категорию **FGraph**.

Оценивание программных архитектур на основе нечетких графов требует представления модели, в которой вершины — набор значений нечетких атрибутов. С этой целью понятие нечеткого графа расширено до (L,U)-нечеткого графа, в котором элементы множеств истинности, соответствующих меткам вершин и дуг, являются элементами произвольно выбранной решетки, а не только интервала [0,1]. Пусть L — полная решетка значений истинности, U (называемое пространством элементарных утверждений — атрибутов) — произвольно выбранное множество. Тогда (L,U)-нечеткий граф — направленный граф, в котором каждая дуга e помечена значением из множества L, каждая вершина n — элементами L-значимого множества (U_n,ζ_n), где $U_n \subseteq U$ — множество элементарных утверждений, заданных в вершине n, а $\zeta_n : U_n \to L$ — функция, назначающая значение из множества L каждому элементу U_n.

Нечеткие модели в трансформационных методах используются: 1) для получения наиболее «качественной» реализации путем оценки возможных преобразований в нечетком пространстве моделирования; 2) для учета изменения требований пользователя, степень неопределенности которых может варьироваться, что требует автоматической генерации и выбора наиболее подходящего способа реализации (декомпозиции) системы при наличии соответствующих преобразований. Необходимыми дополнительными составляющими данного подхода являются последующая дефуззификация нечеткой ПЗМ, которая была сгенерирована, а также сохранение нечеткого пространства моделирования.

Трансформация нечетких графов включает два понятия. Одно из них — правило замены, которое устанавливает соответствие между вершинами и может быть формализировано как частичная функция на множестве вершин, другое — «совпадение», при котором могут осуществляться соответствующие правила замены. Совпадение должно выделять определенные подграфы в графах, подлежащих трансформации.

Продукция над нечеткими графами $p = (I \xrightarrow{l} L;\ I \xrightarrow{r} R)$ — множество, состоящее из нечетких графов L, I, R и двух морфизмов: $I \xrightarrow{l} L$ и $I \xrightarrow{r} R$. Непосредственное преобразование $G \Rightarrow H$ на основе p и m (или $G \overset{p,m}{\Rightarrow} H$) из нечеткого графа G в нечеткий граф H задано диаграммами (1) и (2) в категории нечетких графов **FGraph**:



В процессе применения продукции при нечетком совпадении между $L \cup I$ и элементами начального графа G удаляются элементы L, добавляются элементы R, элементы I остаются неизменными. Таким образом, каждая продукция задает точку (L, R) в нечетком пространстве моделирования, где L — обозначение измерения, а R — нечеткая координата в этом измерении.

Для формализации систем трансформаций нечетких графов (СТНГ) [13, 14], которые являются обобщением последовательных систем трансформаций графов (графовых грамматик) и учитывают основные виды нечеткости, возникающие как при построении базовых категорий нечетких объектов, так и при описании трансформаций нечетких графов, порождаемых нечеткими множествами, используется теоретико-категорный подход.

Множество всех нечетких графов, построенных на множестве меток Σ, обозначим G_Σ. СТНГ — это система $FGG = (S, P, \Delta, L, \varphi)$, где $S \in G_\Sigma$ — начальный нечеткий граф системы FGG, P — конечное множество правил трансформации графов (продукций), $\Delta \in \Sigma$ — множество терминальных символов, L — множество весов (например, дистрибутивные решетки с 0 и 1), φ: $P \to L$, $\varphi(p)$ — степень принадлежности выведения продукции p. Порождаемый системой FGG язык включает все нечеткие графы $G \in G_\Sigma$, помеченные на множестве меток Δ, которые выводятся из начального графа S с помощью применения продукций из множества P, т.е. $L(FGG) = \left\{ G \in G_\Sigma \,\middle|\, S \overset{*}{\underset{P}{\Rightarrow}} G \right\}$.

Пусть заданы система трансформаций FGG и нечеткие графы $G_u, G_v \in G_\Sigma$, где G_Σ — множество объектов категории **FGraph**. При этом G_u непосредственно по-

рождает G_v со степенью $\min(\mu(g(p)),\varphi(p))$, если найдется такое нечеткое совпадение $g(p)$: $G_u \to G_{lp}$, которое выделяет в графе G_u подграф, соответствующий левой части продукции со степенью $\mu(g(p))$: $G_u \overset{\mu(g(p))*\varphi(p)}{\underset{p}{\to}} G_v$.

Предусматривается, что трансформации над отдельными компонентами выполняются одновременно. Параллельное выполнение распределенных действий может быть выражено параллельной продукцией на нечетких графах. Продукция нечетких графов $\hat{p}_1 + \hat{p}_2 = (\hat{I}_1 + \hat{I}_2 \xrightarrow{\hat{l}_1+\hat{l}_2} \hat{L}_1 + \hat{L}_2;\ \hat{I}_1 + \hat{I}_2 \xrightarrow{\hat{r}_1+\hat{r}_2} \hat{R}_1 + \hat{R}_2)$ называется параллельной продукцией, которая состоит из $\hat{p}_1$ и $\hat{p}_2$, где $\hat{p}_1 = (\hat{I}_1 \xrightarrow{\hat{l}_1} \hat{L}_1; \hat{I}_1 \xrightarrow{\hat{r}_1} \hat{R}_1)$ и $\hat{p}_2 = (\hat{I}_2 \xrightarrow{\hat{l}_2} \hat{L}_2; \hat{I}_2 \xrightarrow{\hat{r}_2} \hat{R}_2)$ — отдельные продукции нечетких графов, $\hat{L}_1 + \hat{L}_2$, $\hat{I}_1 + \hat{I}_2$ и $\hat{R}_1 + \hat{R}_2$ — копродукты графов, $\hat{l}_1 + \hat{l}_2$, $\hat{r}_1 + \hat{r}_2$ — порожденные морфизмы.

Пусть P^+ — наименьшее расширение множества P, включающего все параллельные продукции $\hat{p}_1 + \hat{p}_2$ для $\hat{p}_1, \hat{p}_2 \in P^+$. Множество всех возможных спецификаций архитектуры, которая задается системой *FGG*, представлено классом всех возможных преобразований, начинающихся с S и использующих множество P^+ продукций графов, т.е. $S \overset{P^+}{\Rightarrow}{}^{*}_{di} G$.

Каждая продукция строится на основе трех нечетких графов, которые представляют собой антецедент, результат и входной граф 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

$$V(N_1)\&V(N_2)\&\ldots\&V(N_N)\&V(E_1)\&V(E_2)\&\ldots\&V(E_M),$$

где $\{N_1, N_2, \ldots, N_N\}$ — множество вершин нечеткого графа, $\{E_1, E_2, \ldots, E_M\}$ — множество ребер нечеткого графа, $V(e)$ — нечеткое множество соответствующей вершины или ребра нечеткого графа. Обозначим принадлежность вершины и ребра графу-антецеденту и графу-консеквенту соответственно верхним индексом A и C.

Таким образом, простую продукцию, задающую соответствующее преобразование, можно записать в виде

ЕСЛИ

$$V'(N_1^A) = V(N_1^A)\&\ldots\&V'(N_N^A) = V(N_N^A)\&V'(E_1^A) = V(E_1^A)\&\ldots\&V'(E_M^A) = V(E_M^A)$$

ТО

$$V'(N_1^C) = V(N_1^C)\&\ldots\&V'(N_K^C) = V(N_K^C)\&V'(E_1^C) = V(E_1^C)\&\ldots\&V'(E_L^C) = V(E_L^C),$$

где $V(N_i^A)$, $V(E_j^A)$, $V'(N_i^A)$, $V'(E_j^A)$, $V(N_k^C)$, $V(E_l^C)$, $V'(N_k^C)$, $V'(E_l^C)$, $i = \overline{1,N}$, $j = \overline{1,M}$, $k = \overline{1,K}$, $l = \overline{1,L}$, — дискретные нечеткие множества или нечеткие числа, причем $V(N_i^A)$, $V(E_j^A)$, $V(N_k^C)$, $V(E_l^C)$ — указанные в продукции нечеткие множества, $V'(N_i^A)$, $V'(E_j^A)$ — значения истинности вершин и ребер нечеткого графа, к которому применяется продукция, обычно отличающиеся от значений, указанных в продукции; $V'(N_k^C)$, $V'(E_l^C)$ — исправленные значения истин-

ности нечетких множеств графа, получаемого в результате применения продукции.

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

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

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 уменьшаются. Адекватность применения определенного типа вывода зависит от задачи и может отличаться для различных продукций, соответствующих определенным этапам решения.

При монотонном выводе существующие значения истинности вершин и ребер графа не могут уменьшаться при наличии дополнительных свидетельств. Пусть NC — множество вершин, EC — множество ребер графа-консеквента. Формула монотонного вывода для значений истинности вершин и ребер, которые добавляются или остаются в соответствии с указанной продукцией, имеет вид

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

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

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

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

Нисходящий монотонный вывод эффективен, когда значения истинности $\mu'(V', x)$ представляет верхний предел возможного значения

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

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

Распределенные системы трансформаций нечетких графов являются определенным обобщением СТНГ, что позволяет описывать допустимые преобразования сетевых структур распределенных компонентов (модулей). Для этого категория нечетких графов обобщена на категорию распределенных нечетких графов (FD-графов). Ее объектами являются графы компонентов, каждый из которых, в свою очередь, содержит нечеткий граф [14].

Изменение формы сети компонентов (модулей) на протяжении шага преобразования дает возможность моделировать динамические сетевые структуры компонентов, которые задают сервисно-ориентированную архитектуру. При этом в одной продукции СТНГ можно задавать изменение структуры графа одновременно в нескольких разных сетевых компонентах. Например, параллельно создаются порты коммуникации как компонента-клиента, так и компонента, обеспечивающего сервис.

На уровне ПЗМ рассматриваются аспекты поведения системы, связанные с ее функциональностью, причем используется только вычислительная инфраструктура, основанная на нечетких компонентах, портах и соединителях распределенных систем и их типах соответственно. Уровень сервисно-ориентированной архитектуры моделируется как система трансформаций нечетких распределенных графов и представляет специфические механизмы для публикации и поиска сервисов. Аспекты обеспечения сервиса, безопасности и мобильности представляются более специфическими уровнями, которые формируют иерархию уточнения вплоть до моделей специфических платформ.

Обе модели представляют разные уровни абстракции платформы, поэтому корректное соотношение между ними определено как соответствующее понятие конкре-

тизации архитектуры. Для каждого шага трансформации $s = (G \Rightarrow P)$ в ПНМ, которая задается системой преобразований $\wp^{\text{нез}}$, последовательность применения продукций $s^{SOA} = (G^{SOA} \Rightarrow^{*} P^{SOA})$ системы $\wp^{SOA}$ сервисно-ориентированной модели представляет собой корректную конкретизацию шага s, если G^{SOA}, P^{SOA} — структурная конкретизация нечетких графов G и P соответственно.

К характеристикам сервисно-ориентированной архитектуры (атрибутам [15]) относят такие, как: совместимость (интероперабельность), надежность, пригодность сервисов, уровень безопасности, производительность, масштабируемость, расширяемость, адаптивность и т.п. Трансформации сервисно-ориентированной архитектуры с учетом отдельной характеристики могут быть заданы соответствующими продукциями СТНГ.

Выбор решения среди нескольких альтернативных трансформаций осуществляется в нечетком пространстве моделирования с учетом многочисленных параметров оценки (атрибутов). Весы и оценки каждого параметра задаются нечеткими числами или словами, поэтому реализован метод нечеткого многоатрибутного принятия решений.

Для каждой продукции СТНГ $p = (I \xrightarrow{l} L;\ I \xrightarrow{r} R)$, которую можно применить к модели системы, определим описания, дающие вербальную оценку атрибутов трансформаций и соответствующие им лингвистические переменные. Пусть P — множество альтернативных правил трансформации, A — множество атрибутов альтернативных трансформаций, произведение $P \times A$ — область определения переменных. Тогда переменные, значения которых находятся во множестве D, назовем лингвистическими переменными $L(p, a)$, указывающими оценку атрибута a альтернативной трансформации p.

Элементы нечеткого графа $(R - I)$, добавляемые в результате трансформации, определяют точки (L, R), которые оцениваются в нечетком пространстве моделирования. Свяжем с продукцией атрибут a. Тогда оценку по атрибуту a продукции СТНГ задано как T-норму на всех элементах $(R - I)$:

$$T_a(R - I) = \mu_a(r_1) \overset{T}{*} \mu_a(r_2) \overset{T}{*} \ldots \overset{T}{*} \mu_a(r_n),$$

где $R - I = r_1 \cup r_2 \cup \ldots \cup r_n$.

При принятии решений относительно трансформаций прежде всего осуществляется оценка атрибутов и свойств моделей, связанных с многоатрибутными решениями. В результате путем лингвистического сопоставления результата операции $T(R - I)$ со словарем D описаний лингвистических переменных получаем вербальное представление характеристики определенной трансформации, например «средняя», «существенная», «значительная» для атрибута адаптивности.

На основе оценки атрибутов осуществляется оценка альтернативных трансформаций, т.е. процесс получения лингвистической оценки Z функции F по лингвистическому представлению оценки альтернативных трансформаций и их атрибутов (значениям лингвистических переменных $L_1, L_2, \ldots, L_k$) можно записать в виде $Z = F(L_1, L_2, \ldots, L_k)$. В процессе принятия решений выбирается трансформация, дефуззификация функции принадлежности которой дает наибольшее значение.

ИНСТРУМЕНТАРИЙ НЕЧЕТКОГО ТРАНСФОРМАЦИОННОГО ПОДХОДА

Инструментальные средства поддержки СТНГ содержат библиотеку прикладных классов и набор примеров преобразований, разработанных в среде программирования Java Development Kit. Библиотека включает основные классы, необходимые для создания нечетких множеств, нечетких отношений, нечетких графов и продукций, основные методы (операции), необходимые для композиционного создания объектов указанных классов и выполнения правил транформации с последующим выведением

полученных нечетких значений и их дефуззификацией. Разработанные средства основаны на теоретико-категорном подходе [13, 14], в процессе их организации выделен ряд уровней (стратифицированных моделей) абстракции для упрощения построения графовых моделей более высокого уровня, которые реализуются в виде объектов соответствующих категорий и морфизмов более низкого уровня.

Для строгой формализации свойств СТНГ использованы определения абстрактных классов категорий, объектов категорий, их морфизмов и основополагающая категория множеств. Категории нечетких множеств содержат классы для вычислений на основе дискретных нечетких множеств, т.е. их категории, объекты и морфизмы. Каждый объект указанной категории строится на основе пары объектов четких множеств, называемых множеством носителя и множеством истинности. Нечеткие графы и их соответствующие морфизмы используются в дальнейшем для построения нечетких продукций.

Согласно предложенному в [14] подходу класс `Production` правил трансформации нечетких графов (продукций) позволяет создавать каждую такую продукцию на основе трех нечетких графов — L, I и R:

```
Production rel = new Production("rel",L,I,R).
```

Здесь L — граф-антецедент в левой части продукции, R — граф-результат в правой части, I — промежуточный граф, позволяющий определить место присоединения добавляемых вершин $(R-I)$ по отношению к удаляемым вершинам $(L-I)$.

Результирующий граф может быть определен при выполнении продукции с использованием связанного с ним объекта-исполнителя класса `FuzzyRuleExecutor`. Однако если бы входной граф полностью не соответствовал графу-антецеденту G, результирующий граф после применения продукции содержал бы пустые нечеткие значения вершин и ребер. Поэтому продукция выполняется только тогда, когда метод `testRuleMatching` возвращает значение «истина»:

```
if (prod.testProductionMatching(G))
{
     FuzzyGraph fgraph = prod.execute();
     ... }
```

Такое решение позволяет сэкономить время, необходимое для выполнения продукции, когда она не изменяет результирующего графа. Это особенно актуально для продукций, которые имеют сложные нечеткие графы-антецеденты.

Каждая продукция СТНГ связана с объектом класса `FuzzyRuleExecutor`. С его помощью метод `execute` осуществляет выбор способа выполнения продукции и выдает результат в виде нечеткого графа. В классе `FuzzyRuleExecutor` реализованы три способа нисходящего нечеткого вывода, соответствующие оператору вывода Min Мамдани [16, 17], произведению Ларсена и правилу, определенному Цукамото [18].

На уровне конструирования СТНГ предусмотрены методы для добавления, модификации и управления поиском нечетких графовых продукций, а также переключения общей стратегии нечеткого вывода (монотонного, немонотонного, нисходящего монотонного). В частности, реализован метод `findBestMatching` поиска продукции с наибольшим значением совпадения по отношению к входному нечеткому графу:

```
Production p = fgg.findBestMatching(G0);
if (p!=null) {
     res_graph = p.execute(G0);}
```

ПРИМЕР

Рассмотрим применение данного подхода к построению фрагмента ПНМ и ПЗМ с использованием нечетких графов для задачи управления парковкой автомобиля [19], которая очень актуальна в настоящее время.

Положение грузовика определяется значением трех переменных: x, y — координатами по горизонтальной и вертикальной осям, ϕ — углом, под которым грузовик находится относительно оси y. Грузовик движется с постоянной скоростью, а в качестве управляемой выходной переменной выступает изменение угла его поворота $\Delta\phi$. Область допустимых значений переменной x: [0,+100] разбита на пять областей, которым соответствуют треугольные функции принадлежности $\{M_2, M_1, S, D_1, D_2\}$. Аналогично области значений переменной ϕ: [–90,+270] и $\Delta\phi$ разбиты на семь и пять частично перекрывающихся областей, для которых сгенерированы соответствующие функции принадлежности. Таким образом, в точке пересечения смежных функций $\mu(\phi) = 0,5$, $\mu(\Delta\phi) = 0,5$. Уравнения для $x(t+1)$, $y(t+1)$ и $\phi(t+1)$, описывающие динамику перемещения грузовика, приведены в [16].

Построим ПНМ на основе системы трансформаций нечетких графов. Сформулируем отдельные шаги процедуры для решения подобных задач нечеткого управления.

1. Построить модель входного нечеткого графа.

2. Фуззифицировать входной граф (присвоить его вершинам и ребрам нечеткие значения).

3. Применить входной граф ко всем подходящим продукциям в системе, выполняя продукции по одной и обеспечивая глобальное накопление изменений в результирующем нечетком графе.

4. Дефуззифицировать результирующий нечеткий граф (создать четкие числа из нечетких значений отдельных ребер и вершин).

5. Повторять шаги 1–4, пока не достигнута цель управления.

Преимущество данного подхода в том, что использование СТНГ позволяет не задавать явно алгоритм рассмотренной ранее процедуры, заменив его декларативным описанием правил вывода. В табл. 1 приведена система трансформаций нечетких графов, состоящая из 40 продукций. При этом разным продукциям соответствуют различные типы вывода: монотонный, нисходящий монотонный или немонотонный. Для упрощения выполнения процедурных действий, таких как фуззификация или дефуззификация значения, связанного с вершиной графа, добавлены действия по умолчанию, выполняемые при удалении или добавлении вершины нечеткого графа.

Первые 35 продукций предназначены для управления направлением движения грузовика при различных сочетаниях нечетких чисел Xt и Phi. Чтобы каждая продукция выполнилась не более одного раза при каждой коррекции направления, применимость продукций задается нечетким множеством Control, носителем которого являются номера продукций, а степень принадлежности данной продукции указывает на возможность его применения. При выполнении продукции соответствующее ей значение принадлежности обнуляется. Продукции 36 и 37 служат для накопления суммы коррекций DeltaPhi, образовавшихся в результате запуска предыдущих продукций. Продукция 38 срабатывает при необходимости дефуззификации переменной SumDeltaPhi, а продукция 39 — для последующей фуззификации переменных Xt, Yt, Phi. Наконец, продукция 40 задает условие, при котором система трансформаций заканчивает работу. Данное условие соответствует состоянию, при котором автомобиль находится у портала под прямым углом, поэтому порождается объект «Трап».

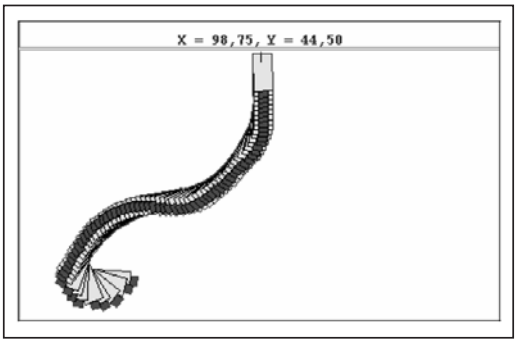
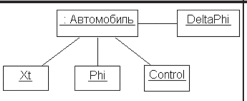
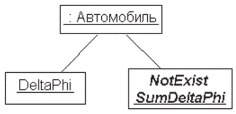
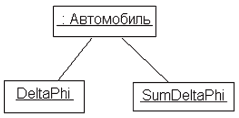
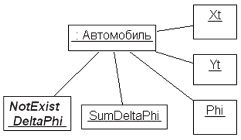
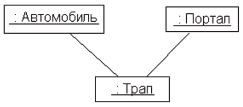


Рис. 3. Траектория движения грузовика, управляемого системой трансформации нечетких графов

На рис. 3 показана одна из возможных траекторий движения грузовика, получаемая в результате применения продукций, приведенных в табл. 1. Начальный нечеткий граф содержит вершины «Автомобиль» и «Необходима фуззификация», представляющие собой нечеткие множества — синглетоны. Таким образом, первой всегда выполняется продукция фуззификации 39, после чего выбираются все применимые продукции в диапазоне 1–35.

Таблица 1. Система трансформации нечетких графов для задачи парковки автомобиля

Номер продукции	Граф в левой части продукции	Граф в правой части продукции	Тип вывода	Действия добавления/ удаления
1–35			Нисходящий монотонный	Отсутствуют
36			Немонотонный	~DeltaPhi { SumDeltaPhi= DeltaPhi; }
37			Немонотонный	~DeltaPhi { SumDeltaPhi= max(DeltaPhi, SumDeltaPhi); }
38			Монотонный	~SumDeltaPhi { c_DeltaPhi = Defuzzify(this); }
39			Немонотонный	Xt { Fuzzify(c_Xt); } Yt { Fuzzify(c_Yt); } Phi { Fuzzify(c_Phi); }
40			Монотонный	Отсутствуют

Данная ПНМ преобразуется в ПЗМ, основанную на языке и метамодели, принятых для ПО реального времени автомобильных систем [19]. В том числе предусмотрено использование AML (Automotive Modeling Language) — языка моделирования автомобильных систем, опирающегося на ряд специфических понятий, таких как функциональные кластеры, порты и обмен сигналами. При появлении новых платформ и их метамоделей, в том числе сервис-ориентированных, определяются дополнительные правила трансформации, позволяющие получить нечеткую оценку эффективности создаваемых моделей в нечетком пространстве моделирования для целевой платформы.

ЗАКЛЮЧЕНИЕ

Рассмотренный в работе подход позволяет управлять процессом эволюционного изменения моделей программной архитектуры на основе принятия решений в нечетком пространстве моделирования относительно атрибутов функционирования целевой платформно-зависимой архитектуры. В процессе изменения требований к решению, которое предоставляется модулями целевой платформы, процесс трансформации может быть направлен продукциями, заданными системой трансформаций нечетких графов. Очевидно, что, поскольку трансформации

и метамодели также специфицируются нечеткими графами, их эволюция может быть задана в виде систем последовательных преобразований.

Процедура принятия решений относительно выбора трансформации на основе нечеткого представления характеристик программных архитектур протестирована на серии архитектур-образцов. Инструментальные средства поддержки нечеткого трансформационного подхода содержат методы композиционного создания нечетких графов и правил трансформации (продукций) с последующим выведением полученных нечетких значений и их дефуззификацией. Использование нечетких графовых моделей позволяет свести процедуру получения ПЗМ к декларативной спецификации правил и стратегии нечеткого вывода.

СПИСОК ЛИТЕРАТУРЫ

1. Редько В.Н. Основания композиционного программирования // Программирование. — 1979. — № 3. — С. 3–13.
2. Глушков В.М., Капитонова Ю.В., Летичевский А.А. О применении метода формализованных технических заданий к проектированию программ обработки структур данных // Там же. — 1978. — № 6. — С. 31–43.
3. Парасюк И.Н., Сергиенко И.В. Пакеты программ анализа данных: технология разработки. — М.: Финансы и статистика, 1988. — 159 с.
4. Криштопа И.В., Непомнящий Б.Д., Перевозчикова О.Л., Ющенко Е.Л. ДИСУППП — диалоговая система управления специализированными пакетами прикладных программ // Кибернетика. — 1980. — № 2. — С. 70–76.
5. Miller J., Mukerji J. MDA Guide Version 1.0. OMG Document: omg/2003-05-01, 2003. — http://www.omg.org/mda/mda sub files/MDA sub Guide sub Version1-0.pdf.
6. Kleppe A., Warmer J., Bast W. MDA Explained. The model driven architecture: practice and promise. — New York: Addison-Wesley Professional, 2003. — 192 p.
7. Басс Л., Клементс П., Кацман Р. Архитектура программного обеспечения на практике. — СПб.: Питер, 2005. — 576 с.
8. Сергієнко І.В., Парасюк І.М., Єршов С.В. Нечіткий трансформаційний підхід до розробки програмних систем // Пробл. программирования. — 2004. — № 2–3. — С. 122–132.
9. Парасюк І.М., Єршов С.В. Методи аналізу програмних архітектур, представлених нечіткими графовими моделями // Там же. — 2006. — № 1–2. — С. 101–110.
10. Ершов С.В. К проблеме формализации объектно-ориентированных методов разработки программного обеспечения на основе нечеткой логики // Компьютерная математика. — 2003. — № 2. — С. 62–77.
11. Основы инженерии качества программных систем / Ф.И. Андон, Г.И. Коваль, Т.М. Коротун, В.Ю. Суслов / Под ред. И.В. Сергиенко. — К.: Академпериодика, 2002. — 504 с.
12. Сергиенко И.В., Парасюк И.Н., Провотар А.И. О применении категорных методов в computer science // Кибернетика и системный анализ. — 1995. — № 1. — С. 146–154.
13. Парасюк И.Н, Ершов С.В. Категорный подход к построению нечетких графовых грамматик // Там же. — 2006. — № 4. — С. 80–96.
14. Парасюк И.Н, Ершов С.В. О трансформации нечетких графов, задаваемых *FD*-грамматиками // Там же. — 2007. — № 2. — С. 129–147.
15. O'Brien L., Bass L., Merson P. Quality attributes and service-oriented architectures: Techn. Note. — CMU/SEI-2005-TN-014. — 2005. — 29 p.
16. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. — М.: Горячая линия–Телеком, 2004. — 452 с.
17. Mamdani E.H. Application of fuzzy logic to approximate reasoning using linguistic systems // Fuzzy Sets and Systems. — 1977. — **26.** — P. 1182–1191.
18. Carlsson C., Fullér R. Optimization with linguistic variables: TUCS Techn. Rep. — N 157. — Jan. 1998. — 16 p.
19. Fraichard T., Garnier P. Fuzzy control to drive car-like vehicles // Robotics and Autonomous Systems. — 2001. — N 34(1). — P. 1–22.

Поступила 04.12.2007