УДК 004.435

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ЯДРА СИСТЕМЫ МЕТАПРОГРАММИРОВАНИЯ

А. А. Седунов

Воронежский государственный университет

Поступила в редакцию 10.03.2011 г.

Аннотация. В данной работе представлена математическая модель структуры объектов, лежащей в основе разрабатываемой автором объектно-ориентированной инфраструктуры метапрограммирования TURBINE. Рассматриваются основные структуры данных, используемые для представления объектных моделей, отношения между объектами, а также основные разновидности и способы описания метаданных в объектных моделях.
Ключевые слова: метапрограммирование, ООП, теория, модель.
Annotation. The paper present mathematical model of object structure underlying object-oriented metaprogramming system TURBINE which is being developed by the author. It considers basic data structures which are used for presentation of object models, object relations and major cases/ways of description of metadata in object models.
Keywords: metaprogramming, OOP, theory, model.

ВВЕДЕНИЕ

На данный момент можно выделить несколько активно развивающихся подходов, направленных на преодоление ограничений, характерных для традиционной концепции объектно-ориентированного программирования, а также позволяющих улучшить модульность программных компонентов, разрабатываемых на основе методологии ООП. В контексте данной работы мы выделяем подход, основанный на декомпозиции классов, поскольку он может быть легко обобщен и приспособлен для модульного описания произвольных программных структур (помимо собственно классов). Метод основан на отказе от традиционного монолитного определения класса и замене его множественными определения отдельных аспектов. Например, в языке Diesel (одна из реализаций метода декомпозиции классов) выделяются следующие аспекты:

- класс как символическое имя;
- класс как субъект отношения генерализации;
- класс как контракт, или набор операций;
- класс как структура данных, или набор определений слотов;

При наличии подходящего набора языковых средств для определения подобных аспектов становится возможной привязка этих определений к конкретным модулям. Комбинируя модули, мы можем создавать контекстно-зависимые представления одного класса в зависимости от конкретного места, в котором употребляется его имя, например, изменяя набор методов, доступных его клиентам.

Далее мы представим математическую модель структуры объектов, лежащей в основе разрабатываемой автором объектно-ориентированной инфраструктуры метапрограммирования TURBINE и показано, каким образом метод декомпозиции позволяет достичь контекстно-зависимой интерпретации элементов программной структуры. Рассматриваются основные структуры данных, используемые для представления объектных моделей, отношения между объектами, а также основные разновидности и способы описания метаданных в объектных моделях.

ОСНОВНЫЕ ОБОЗНАЧЕНИЯ

Уточним ряд специфических обозначений, используемых в данной работе.

Для записи функций без присвоения им явного имени используется ограниченная форма лямбда-нотации вида , или , если множество S ясно из контекста. При этом $\mathrm{Dom}\,\lambda\left(x \in S\right)_{\varphi} E = S$.

Запись $Y \approx X$ означает, что функция является изоморфизмом, а множества X и Y равномощны. Если функция φ ясна из контекста,

будем сокращать запись до $Y \approx X$. Аналогичное обозначение будем использовать для элементов X и Y, т. е.

$$Y \overset{\varphi}{\approx} X, x \in X, y \in Y \Rightarrow y \overset{\varphi}{\approx} x \overset{def}{\leftrightarrow} y = \varphi(x).$$

Если конструктор φ не обозначен явно, вместо $\varphi(x)$ будем писать $\hat{}x$, а вместо $\varphi^{-1}(y)$ – соответственно $y\hat{}$.

Пусть $F : A \to S$, где S – набор множеств, тогда

$$\prod F \overset{def}{=} \{f \mid \mathrm{Dom}\, f = A \wedge \forall (x \in A)[f(x) \in F(x)]\}.$$

Если $\mathbf{I}$ – некоторое фиксированное счетное множество идентификаторов, то элементы произведения $\prod X$, где $X \in S^I$, S – набор множеств и $I \in \mathcal{P}\mathbf{I}$, назовем размеченными кортежами. Значение размеченного кортежа x в точке $i \in I$ обозначим через $x.i$. Размеченное декартово произведение при фиксированной мощности множества I будем записывать в виде $i_0 : X_0, \ldots, i_{n-1} : X_{n-1}$, где $X_k = X(i_k)$, а $\mathrm{Rng}\langle i_k \rangle_{k \in 0..n} = I$ (порядок идентификаторов i_k при этом роли не играет). Аналогичным образом для представления элементов такого произведения будем использовать обозначение $i_0 : x_0, \ldots, i_{n-1} : x_{n-1}$, где $x_k \in X_k$, или, если порядок идентификаторов фиксирован, $x_0, \ldots, x_{n-1}$.

Запись $Z = i_0 : X_0, \ldots, i_{n-1} : X_{n-1}, \ldots$ означает, что Z является объединением множеств вида $Z = \prod X$, причем $X(i_k) = X_k$ для всех $k \in 0..n$. Такую запись будем называть открытым произведением.

Нам также потребуются средства для представления размеченных объединений. Пусть $F : A \to S$, где S – набор множеств, тогда

$$\sum F \overset{def}{=} \left\{ [x, s] \middle| x \in A \wedge s \in F(x) \right\}.$$

Для элементов множества $\sum F$ введем операцию tag таким образом, что $[x, s] \in \sum F \Rightarrow \mathrm{tag}[x, s] \overset{def}{=} x$, а сами элементы будем обозначать $x(s)$: $[x, s] \in \sum F \Rightarrow x(s) \overset{def}{=} [x, s]$. Если множество $F(x)$ состоит из одного элемента, то запись $x(s)$ сократим до x.

Пусть $X \in S^I$, где S – набор множеств и $I \in \mathcal{P}\mathbf{I}$. Элементы множества $\sum X$ назовем размеченными вариантами (далее – варианты). При фиксированной мощности множества I объединение $\sum X$ будем записывать в виде $i_0(X_0) + \ldots + i_{n-1}(X_{n-1})$, где $X_k = X(i_k)$, а $\mathrm{Rng}\langle i_k \rangle_{k \in 0..n} = I$. Если среди множеств X_k есть одинаковые, то соответствующие члены группируются в виде $(i_{k_1} + \ldots + i_{k_p})(X_k)$. Скобки вокруг X_k могут быть опущены, если это не приводит к неоднозначности.

По аналогии с открытыми произведениями можно ввести понятие открытого объединения $i_0(X_0) + \ldots + i_{n-1}(X_{n-1}) + \ldots$ как объединения (обычного $\cup$) множеств вида $\sum X$, где $X_k = X(i_k)$ при любом $k \in 0..n$.

Пусть $U = i_0(X_0) + \ldots + i_{n-1}(X_{n-1})$ и $f_k : X_k \to Y$, тогда $f = i_0(f_0) \oplus \ldots \oplus i_{n-1}(f_{n-1})$ обозначает такую функцию f, что

$$\begin{cases} f : U \to Y \\ z = i_k(x) \in U \Rightarrow f(z) = f_k(x) \end{cases}$$

ОСНОВНЫЕ ПОНЯТИЯ

Рассмотрим формальное определение структуры объекта. Для этого вначале введем два фиксированных счетных множества $\mathbf{V}$ и $\mathbf{L}$, где $\mathbf{L}$ – счетное множество ссылок с выделенным элементом $\mathrm{null} \in \mathbf{L}$, а $\mathbf{V}$ – множество простых значений, имеющее вид открытого размеченного объединения

$$\mathbf{V} = \mathrm{int} : [int] + \mathrm{bool} : [bool] + \mathrm{void} : [void] + \ldots, \quad (1)$$

где $[int] \approx \mathbb{N}_0$ – используется для представления неотрицательных целых чисел, $[bool] \approx \mathbb{B}$ – для представления логических значений и $[void] = \{\mathbf{void}\}$.

Определим множества $[val]$, $[dic]$ и $[dat]$ следующей системой уравнений:

$$\begin{cases} [val] \approx \mathrm{pr}\,\mathbf{V} + \mathrm{memb}[dic] \\ [dic] = \mathbf{I}^{[dat]} \\ [dat] \overset{def}{=} \mathrm{val}[val] + \mathrm{ref}\,\mathbf{L} \end{cases} \quad (2)$$

Используя аппарат теории доменов, можно показать существование минимального набора $\langle [val], [dic], [dat] \rangle$, удовлетворяющего . Далее мы будем использовать множества из этого набора. Элементы множества $[val]$ назовем значениями (или статическими структурами), элементы, элементы $[dic]$ – словарями, а элементы $[dat]$ – элементами данных.

Константа используется в качестве значения в словаре и выражает тот факт, что соответствующий идентификатор отсутствует в структуре (при этом формально словарь остается функций, определенной на всем множестве $\mathbf{I}$).

Для произвольного множества Z через Z обозначим его расширение константой (при этом подразумевается, что $\notin Z$):

$$Z \overset{def}{=} Z \cup \{\}. \tag{3}$$

Для словаря введем функцию $\mathrm{Dom} : \left[dic\right] \to \mathcal{P}\mathbf{I}$, которая описывает множество идентификаторов x, отображаемых на элементы данных (т. е. полный прообраз множества $\left[dat\right]$):

$$\mathrm{Dom}\ d \overset{def}{=} d^{-1}\left[dat\right]. \tag{4}$$

Таким образом, $d\left(x\right) \neq$ при $x \in \mathrm{Dom}\ d$.

Множество кортежей, состоящих из элементов данных, обозначим $\left[seq\right]$:

$$\left[seq\right] \overset{def}{=} \left[dat\right]^{*}. \tag{5}$$

Словари и кортежи элементов данных объединим общим термином коллекции:

$$\left[col\right] \overset{def}{=} \mathrm{dic}\left[dic\right] + \mathrm{seq}\left[seq\right]. \tag{6}$$

Мы распространяем обозначение на произвольные коллекции. При этом в случае кортежей $\mathrm{Dom}\ d = \mathrm{Dom}\, d$. Кроме того, поскольку словари и кортежи имеют вид $X \to \left[dat\right]$, мы будем для упрощения записи интерпретировать элементы множества $\left[col\right]$ как функции вида $\mathrm{dic}\, d \oplus \mathrm{seq}\, s$.

Внутренняя структура объектов описывается следующим образом:

$$\left[obj\right] \overset{def}{\approx} \mathrm{null}\{\_\} + \\ + \mathrm{bd}\left(\mathrm{type} : \mathbf{T}, \mathrm{self} : \mathbf{L}, \mathrm{memb} : \left[dic\right]\right), \tag{7}$$

$$\left[list\right] \overset{def}{\approx} \left(\mathrm{self} : \mathbf{L}, \mathrm{itm} : \left[seq\right]\right). \tag{8}$$

Для элементов этих множеств введем наименования объект и список соответственно. Списки и объекты совместно будем называть динамическими структурами:

$$\left[dyn\right] \overset{def}{=} \mathrm{obj}\left[obj\right] + \mathrm{list}\left[list\right]. \tag{9}$$

Структурное различие между динамическими структурами и значениями состоит в наличии у первых компонента self, который содержит идентичность.

Множество $\mathbf{T}$ используются для классификации объектов. Мы будем считать его константой, содержащей конечное число элементов – “типов” (термин “тип” взят в кавычки для того, чтобы его можно было отличить от типов данных, которые рассматриваются позже). Сами “типы” будут конкретизироваться по ходу дальнейшего построения теории.

Элементы $\left[val\right]$ и $\left[dyn\right]$ в совокупности назовем структурами:

$$\left[struct\right] \overset{def}{=} \mathrm{val}\left[val\right] + \mathrm{dyn}\left[dyn\right]. \tag{10}$$

Для значений определим селектор $\mathrm{memb} : \left[val\right] \to \left[dic\right]$:

$$\begin{cases} \mathrm{memb}\left(v \middle| v \approx \mathrm{pr}\, z\right) \overset{def}{=} \mathbf{dic}_0, \\ \mathrm{memb}\left(v \middle| v \approx \mathrm{memb}\, z\right) \overset{def}{=} z, \end{cases} \tag{11}$$

где $\mathbf{dic}_0$ – это словарь, который отображает любой идентификатор на пустой кортеж:

$$\mathbf{dic}_0 \overset{def}{=} »\left(x \in \mathbf{I}\right). \tag{12}$$

Аналогично определим селекторы $\mathrm{memb} : \left[dyn\right] \to \left[dic\right]$ и $\mathrm{id} : \left[dyn\right] \to \mathbf{L}$ для динамических структур:

$$\mathrm{id}\left(\mathrm{obj}\left(z\right) \middle| z \approx \mathrm{null}\right) \overset{def}{=} \mathrm{null} \tag{13}$$

$$\mathrm{id}\left(\mathrm{obj}\, z\right) \overset{def}{=} (\mathrm{null}(\lambda\_.\mathrm{null}) \oplus \\ \oplus \mathrm{bd}(\lambda v.\left(v.\mathrm{self}\right)))z\hat{} \tag{14}$$

$$\mathrm{id}\left(\mathrm{list}\, z\right) \overset{def}{=} z\hat{}.\mathrm{self} \tag{15}$$

$$\mathrm{memb}\left(\mathrm{obj}\, z\right) \overset{def}{=} (\mathrm{null}(»\_.\mathbf{dic}_0) \oplus \\ \oplus \mathrm{bd}(»v.(v.\mathrm{memb})))z\hat{} \tag{16}$$

$$\mathrm{memb}\left(\mathrm{list}\, z\right) \overset{def}{=} \\ \overset{def}{=} \mathbf{dic}_0\left[\text{"}length\text{"} \leftarrow \left[int\right]\left(\left|z\hat{}.\mathrm{itm}\right|\right)\right]. \tag{17}$$

Операцию type будем использовать для обозначения “типа” объекта:

$$\mathrm{type}\left(\mathrm{obj}\, z\right) \overset{def}{=} (\mathrm{null}(\lambda\_.\mathrm{Null}) \oplus \\ \oplus \mathrm{bd}(\lambda v.(v.\mathrm{type})))z\hat{}, \tag{18}$$

где $\mathrm{Null} \in \mathbf{T}$ – специальная константа.

Таким образом, списки определяют специальный словарь, который связывает длину соответствующего кортежа элементов данных (представленную в виде значения $\left[int\right]$) с идентификатором “length”.

Операцию *memb* теперь можно поднять на уровень произвольной структуры, т. е. $\mathrm{memb} : \left[struct\right] \to \left[dic\right]$:

$$\mathrm{memb}\, z \overset{def}{=} \Big(\big(\mathrm{val} \oplus \mathrm{dyn}\big)\mathrm{memb}\Big) z\,. \qquad (19)$$

Отметим, что null используется как для обозначения специальной ссылки, так и идентификатора в размеченных объединениях, связанных с объектами.

Для каждой структуры s определим собственную коллекцию $\mathrm{Aol}\, s : \left[struct\right] \to \left[col\right]$ следующим образом:

$$\mathrm{Aol}\Big(\mathrm{val}\, z\Big) \overset{def}{=} \mathrm{memb}\, z \qquad (20)$$

$$\mathrm{Aol}\Big(\mathrm{dyn}\, \mathrm{obj}\, \_\Big) \overset{def}{=} \mathrm{memb}\, z \qquad (21)$$

$$\mathrm{Aol}\Big(\mathrm{dyn}\, \mathrm{list}\, v\Big) \overset{def}{=} v\hat{}\,.\mathrm{itm} \qquad (22)$$

Универсумом назовем размеченное объединение, включающее в себя элементы данных, структуры и коллекции:

$$\mathbf{U} \overset{def}{=} \mathrm{dat}\left[dat\right] + \mathrm{struct}\left[struct\right] + \mathrm{col}\left[col\right]. \qquad (23)$$

Для произвольного элемента универсума определим множество содержащихся в нем ссылок $\mathrm{Ref} : \mathbf{U} \to \mathcal{P}\mathbf{L}$:

$$\mathrm{Ref} \overset{def}{=} \varnothing \qquad (24)$$

$$\mathrm{Ref}\Big(\mathrm{dat}\, d\Big) \overset{def}{=} \overset{def}{=} \Big(\mathrm{ref}\Big(\text{»}x.\left\{x\right\}\Big) \oplus \mathrm{val}\Big(\text{»}x.\mathrm{Ref}\, x\Big)\Big) d \qquad (25)$$

$$\mathrm{Ref}\Big(\mathrm{struct}\, z\Big) \overset{def}{=} \mathrm{Ref}\, \mathrm{col}\, z \qquad (26)$$

$$\mathrm{Ref}\Big(\mathrm{col}\, d\Big) \overset{def}{=} \bigcup_{x \in \mathrm{Rng}\, d - \{\}} \mathrm{Ref}\, x \qquad (27)$$

Для обозначения принадлежности объекта $z \in \left[obj\right]$ определенному "типу" T используем отношение $z : T$, заданное следующими правилами:

$$\frac{\mathrm{type}\, z = T}{z : T} \qquad (28)$$

$$\frac{z : T, T <: T'}{z : T'} \qquad (29),$$

где $<:$ – отношение между "подтипами", выражающее их классификацию. Общие свойства отношения $<:$ включают предпорядок и минимальность Null:

$$T <: T \qquad (30)$$

$$\frac{T <: T', T' <: T''}{T <: T''} \qquad (31)$$

$$\mathrm{Null} <: T \qquad (32)$$

"Тип" называется абстрактным, если он не может быть значением выражения $\mathrm{type}\, z$ для какой-либо динамической структуры. Абстрактные типы используются, главным образом, для классификации в отношениях $:$ и $<:$.

Имя "типа" мы также будем использовать для обозначения множества всех принадлежащих ему объектов:

$$\Big\{e \in \left[obj\right] \Big| e : T\Big\} \qquad (33)$$

Для связи ссылок с объектами введем понятие контекста. Ядром контекста называется множество **CCore** функций вида $f : \mathbf{L} \to \left[dyn\right]$, удовлетворяющих условию $f\, \mathrm{null} = \mathbf{null}$, где

$$\mathbf{null} \overset{def}{=} \mathrm{obj}\Big(\hat{}\ \mathrm{null}\Big) \qquad (34)$$

Множеством контекстов $\mathbf{C}$ называется множество всех кортежей, содержащих ядро контекста с идентификатором core:

$$! \overset{def}{=} \Big\langle \mathrm{core} : \mathbf{CCore}, \ldots \Big\rangle \qquad (35)$$

Контекст может отображать указанную метку на константу, которая, как и в случае со словарями, выражает неопределенность метки в этом контексте.

Для обозначения простейшего контекста введем константу $\mathbf{C}_0$:

$$\mathbf{C}_0 \overset{def}{=} \Big\langle \mathrm{core} : \big(\text{»}\_.\big)\Big[\mathrm{null} \leftarrow \mathbf{null}\Big]\Big\rangle \qquad (36)$$

Обозначение Dom естественным образом расширяется на ядро контекста:

$$\mathrm{Dom}\ C.\mathrm{core} \overset{def}{=} \Big\{l \in \mathbf{L} \Big| C.\mathrm{core}\, l \neq \Big\} \qquad (37)$$

Если $C.\mathrm{core}\, l = z$ и $z \in \left[dyn\right]$, то l называется внешней идентичностью динамической структуры z. При этом $\mathrm{id}\, z$ называется внутренней идентичностью.

Динамическая структура z называется валидной в контексте C, (запись: $\square_C z$), если все содержащиеся в ней ссылки определены в этом контексте и внутренняя идентичность совпадает с внешней:

$$\square_C z \overset{def}{\Leftrightarrow} \begin{cases} \mathrm{Ref}\, z \subseteq \mathrm{Dom}\, C.\mathrm{core} \\ \mathrm{id}\, z \in \mathrm{Dom}\ C.\mathrm{core} \\ C.\mathrm{core}\Big(\mathrm{id}\, z\Big) = z \end{cases} \qquad (38)$$

Поднимем предикат валидности на уровень множества $\mathbf{U}$:

$$\square_C v \overset{def}{\Leftrightarrow} \begin{cases} \square_C z, v = \mathrm{struct}\left(\mathrm{dyn}\, z\right) \\ \mathrm{Ref}\, z \subseteq \mathrm{Dom}\, C.\mathrm{core} \end{cases} \quad (39)$$

Ядро контекста ! называется валидным (запись: $\square C.\mathrm{core}$), если множество $\mathrm{Dom}\ C.\mathrm{core}$ конечно и любая динамическая структура, содержащаяся в его множестве значений, валидна в C:

$$\square C.\mathrm{core} \overset{def}{\Leftrightarrow} \mathrm{Dom}\ C.\mathrm{core} \in \mathcal{F}\mathbf{L} \wedge \\ \wedge \forall \left(z \in \mathrm{Rng}\, C.\mathrm{core}\right)\left[\square_C z\right]. \quad (40)$$

Контекст называется валидным, если валидны все его компоненты и выполняется условие верификации:

$$\square C \overset{def}{\Leftrightarrow} \mathrm{ver}\, C \wedge \forall \left(z \in \mathrm{Rng}\, C\right)\left[\square_C z\right]. \quad (41)$$

Далее мы уточним определение верификации. При изложении настоящего параграфа мы будем считать условие верификации тождественно-истинным.

Понятие контекста позволяет нам распространить некоторые свойства динамических структур на соответствующие им ссылки. Пусть $\circ$ – некоторое бинарное отношение, левый операнд которого является динамической структурой. Введем обозначение $l\boxed{\circ}_C x$ для ссылки l, которой в контексте C соответствует такая структура y, что $y \circ x$:

$$l\boxed{\circ}_C x \overset{def}{\Leftrightarrow} l \in \mathrm{Dom}\ C.\mathrm{core} \wedge C.\mathrm{core}\left(l\right) \circ x\,. \quad (42)$$

Обозначение контекста будем опускать (и писать, например, $l\boxed{\in}x$), если это не приводит к неоднозначности.

Рассмотрим теперь операции доступа к компонентам структур. Для начала введем механизм адресации. Значение $x \in \mathbf{X}$, где $\mathbf{X} = \mathbf{I} \cup \mathbb{N}$, будем называть элементом адреса, а кортеж $X \in \mathbf{X}^*$ – адресом. На множестве адресов введем частичный порядок $\leq$ так, чтобы неравенство $X_1 \leq X_2$ имело место тогда и только тогда, когда X_1 является префиксом X_2:

$$X_1 \leq X_2 \overset{def}{\Leftrightarrow} \exists \left(X \in \mathbf{X}^*\right)\left[X_2 = X_1 X\right]. \quad (43)$$

$$X_1 < X_2 \overset{def}{\Leftrightarrow} X_1 \leq X_2 \wedge X_1 \neq X_2\,. \quad (44)$$

Там, где это не приводит к противоречиям, адрес, состоящий из одного элемента $\left\langle x \right\rangle$, будем обозначать x.

Пусть ! - контекст, X - адрес и $u \in \mathbf{U}$. Выражение вида $u\left[X\right]_C \in \left[dat\right]$ назовем извлечением. Определим для него следующие правила редукции:

$$\left[\_\right]_{\_} \rightarrow \quad (45)$$

$$\mathrm{dat}\, d\left[\left\langle\ \right\rangle\right]_{\_} \rightarrow d \quad (46)$$

$$\frac{u \in \mathbf{U}, \mathrm{tag}\, u \neq \mathrm{dat}}{u\left[\left\langle\ \right\rangle\right]_{\_} \rightarrow} \quad (47)$$

$$\mathrm{dat}\left(\mathrm{val}\, v\right)\left[X\right]_C \rightarrow \mathrm{struct}\left(\mathrm{val}\, v\right)\left[X\right]_C \quad (48)$$

$$\frac{l \notin \mathrm{Dom}\ C}{\mathrm{dat}\left(\mathrm{ref}\, l\right)\left[X\right]_C \rightarrow} \quad (49)$$

$$\frac{l \in \mathrm{Dom}\ C}{\mathrm{dat}\left(\mathrm{ref}\, l\right)\left[X\right]_C \rightarrow \mathrm{struct}\left(\mathrm{val}\, C.\mathrm{core}\, l\right)\left[X\right]_C} \quad (50)$$

$$\mathrm{struct}\, z\left[X\right]_C \rightarrow \mathrm{col}\left(\mathrm{col}\, z\right)\left[X\right]_C \quad (51)$$

$$\frac{u = \mathrm{col}\, z, x \notin \mathrm{Dom}\ z}{u\left[\left\langle x \right\rangle X\right]_C \rightarrow} \quad (52)$$

$$\frac{u = \mathrm{col}\, z, x \in \mathrm{Dom}\ z}{u\left[\left\langle x \right\rangle X\right]_C \rightarrow \mathrm{dat}\, z\left(x\right)\left[X\right]_C} \quad (53)$$

Редукцию, состоящую из произвольного (возможно, пустого) конечного числа шагов, обозначим символом $\overset{*}{\rightarrow}$:

$$\alpha \overset{*}{\rightarrow} \alpha \quad (54)$$

$$\frac{\alpha \rightarrow \alpha', \alpha' \overset{*}{\rightarrow} \alpha''}{\alpha \overset{*}{\rightarrow} \alpha''}, \quad (55)$$

где α обозначает произвольное извлечение.

Можно показать, что редукция данного вида всегда приводит к единственному элементу данных:

Теорема 1.

$$u \in \mathbf{U}, X \in \mathbf{X}^* \Rightarrow \exists! \left(d \in \left[\mathrm{dat}\right]\right) \times \\ \times \left[u\left[X\right]_C \overset{*}{\rightarrow} d\left[\left\langle\ \right\rangle\right]_C\right]. \quad (56)$$

Таким образом, извлечение $\_\left[\_\right]$ можно интерпретировать как функцию вида $\mathbf{C} \rightarrow \mathbf{U} \times \mathbf{X}^* \rightarrow \left[dat\right]$. Далее под $\mathrm{eval}\, u\left[X\right]_C$ будем понимать значение этой функции.

В некоторых случаях необходимо наложить на элементы, соответствующие префиксам адреса X, определенные ограничения. В этом

случае мы будем использовать запись $d = \mathrm{eval}\, u\left[X\right]_C$, где $\varphi : \left[dat\right] \to \mathrm{B}$ - предикат:

$$d \overset{\varphi}{=} u\left[X\right]_C \overset{def}{\Leftrightarrow}$$
$$\overset{def}{\Leftrightarrow} \begin{cases} d = \mathrm{eval}\, u\left[X\right]_C \\ \forall\left(\langle\ \rangle \le X' < X\right)\left[\varphi\left(\mathrm{eval}\, u\left[X'\right]_C\right)\right] \end{cases}. \quad (57)$$

Если $\varphi(d)$ имеет вид $d \in Y$, то запись $d \overset{\varphi}{=} \mathrm{eval}\, u\left[X\right]_C$ будем сокращать до $d \overset{Y}{=} \mathrm{eval}\, u\left[X\right]_C$.

Теорема 2.

Результат любого извлечения валиден при условии валидности контекста:

$$\Box C \Rightarrow \Box_C\left(\mathrm{eval}\, u\left[X\right]_C\right) \quad (58)$$

Таким образом, извлечение сохраняет свойство валидности при переходе от контекста к своему результату.

СУЩНОСТИ И МЕТАИНФОРМАЦИЯ

Перейдем теперь к рассмотрению более специфичной части ядра, предназначенной для описания метамоделей. В основе этой структуры лежат специальные объекты, называемые сущностями.

Для начала разделим все объекты на 2 категории. Объекты, имеющие "тип" Pure называются чистыми объектами. Объекты, "тип" которых отличен от Pure, называются сущностями. Абстрактный "тип" сущностей обозначается Ent.

Константа **null**, являясь объектом с "типом" Null, относится как к чистым объектам, так и к сущностям.

Выделяются 4 основных "подтипа" Ent: Attr (атрибуты), Link (связи), Binder (связывающие сущности) и Plent (простые сущности). Для этих "типов" вводятся соответствующие правила классификации:

$$\begin{cases} \mathrm{Attr} <: \mathrm{Ent} \\ \mathrm{Link} <: \mathrm{Ent} \\ \mathrm{Binder} <: \mathrm{Ent} \\ \mathrm{Plent} <: \mathrm{Ent} \end{cases} \quad (59)$$

Кроме того, вводится свойство замкнутости "типа" Ent, запрещающее вводить для него новые непосредственные подтипы.

$$z : \mathrm{Ent} \Rightarrow z : \mathrm{Attr} \vee z : \mathrm{Link} \vee \\ \vee z : \mathrm{Binder} \vee z : \mathrm{Plent} \quad (60)$$

Атрибуты и связи объединяются "типом" Wrapper (промежуточные сущности) со следующими свойствами:

$$\begin{cases} \mathrm{Attr} <: \mathrm{Wrapper} \\ \mathrm{Link} <: \mathrm{Wrapper} \\ \mathrm{Wrapper} <: \mathrm{Ent} \end{cases} \quad (61)$$

Правила классификации "типов" можно представить в виде графа, где вершинами являются сами типы, а наличие дуги соответствует отношению $<:$.

Отличие объектов-сущностей от чистых объектов состоит в том, что с ними ассоциируется метаинформация, позволяющая отслеживать их связи с другими объектами. Для хранения этой метаинформации мы расширим структуру контекста, добавив к ней ряд функций, заданных для ссылок на определенные виды сущностей. Таким образом, мы переопределяем множество $!$, введенное в предыдущем параграфе следующим образом:

$$\mathbf{C} \overset{def}{\leftarrow} ! \times \\ \times \begin{pmatrix} \mathrm{owner} : \mathbf{L} \to \mathbf{L}, \mathrm{target} : \mathbf{L} \to \mathbf{L}, \\ \mathrm{attrs} : \mathbf{L} \to \left(\mathbf{L}^*\right), \mathrm{deps} : \mathbf{L} \to \left(\mathbf{L}^*\right), \\ \mathrm{scope} : \mathbf{L} \to \mathbf{L}, \mathrm{bnds} : \mathbf{L} \to \left(\mathbf{L}^*\right), \ldots \end{pmatrix} \quad (62)$$

Для корректного использования переопределенного понятия нам потребуется также расширить константу $!_0$:

$$\mathbf{C}_0 \overset{def}{\leftarrow} !_0 \times \\ \times \left[\leftarrow \left\langle \begin{matrix} \mathrm{owner} : \text{»}x., \mathrm{target} : \text{»}x., \\ \mathrm{attrs} : \text{»}x., \mathrm{deps} : \text{»}x., \\ \mathrm{scope} : \text{»}x., \mathrm{bnds} : \text{»}x. \end{matrix} \right\rangle \right] \quad (63)$$

Кроме того, необходимо определить валидность для новых компонентов контекста.

Функция owner формирует граф композиции, в котором ссылка l предшествует ссылке l', если $C.\mathrm{owner}\, l = l'$. Для обозначения ссылок, которые могут быть связаны функцией owner, вводится предикат cntd и наименование узел. В рассматриваемой модели к таким ссылкам относятся ссылки на сущности и списки:

$$\mathrm{cntd}_C\, l \overset{def}{\Leftrightarrow} l\boxed{:}_C\, \mathrm{Ent} \vee \mathrm{tag}\left(C.\mathrm{core}\, l\right) = \mathrm{arr} \quad (64)$$

Для ссылок, не являющихся узлами, значением owner считается. Таким образом, полное определение валидности функции owner имеет вид:

$$f = C.\text{owner} \Rightarrow \Box_C f \overset{def}{\leftrightarrow}$$
$$\overset{def}{\leftrightarrow} \begin{cases} l = \text{null} \vee \neg \text{cntd}_C\, l \to f\left(l\right) = \\ \text{cntd}_C\, l \wedge e = f\left(l\right) \to \text{cntd}_C\, e \wedge \Box_{\hat{!}.\text{core}}\, e. \end{cases} \quad (65)$$

Функция target определена для ссылок "типа" Wrapper , которые предназначены для отслеживания узлов и формирует граф ассоциации. Для всех прочих ссылок значение target полагается равным:

$$f = C.\text{target} \Rightarrow \Box_C f \overset{def}{\leftrightarrow}$$
$$\overset{def}{\leftrightarrow} \begin{cases} l = \text{null} \vee \neg l[:]_C \text{ Wrapper} \to f\left(l\right) = \\ l[:]_C \text{ Attr} \to f\left(l\right)[:]_C \text{ Ent} \wedge \Box_C f\left(l\right) \\ l[:]_C \text{ Link} \to \text{cntd}\, f\left(l\right) \wedge \Box_C f\left(l\right). \end{cases} \quad (66)$$

При этом для ссылки "типа" Attr значение функции target может быть только ссылка на сущность.

Функция attrs используется для привязки набора атрибутов к ссылке на сущность. Для ссылок на другие структуры привязка атрибутов не разрешена, поэтому значением функции attrs для таких ссылок мы считаем:

$$f = C.\text{attrs} \Rightarrow \Box_C f \overset{def}{\leftrightarrow}$$
$$\overset{def}{\leftrightarrow} \begin{cases} \neg l[:]_C \text{ Ent} \to f\left(l\right) = \\ f\,\text{null} = \left\langle\ \right\rangle \\ L = f\left(l\right) \to \left|L\right| = \left|\text{Rng}\, L\right| \\ l[:]_C \text{ Ent} \wedge l' \in \text{Rng}\, f\left(l\right) \to \\ \to l'[:]_C \text{ Attr} \wedge \Box_C l' \end{cases} \quad (67)$$

Таким образом, аргумент функции attrs представляет собой ссылку на сущность, а возвращаемое значение – кортеж ссылок на атрибуты, связанных с этой сущностью (заметим, что второе условие в гарантирует отсутствие двух идентичных атрибутов в этом кортеже).

Функции scope и bnds связывают метаинформацию с сущностями "типа" Binder , назначение которых состоит в ограничении доступности ссылок на другие объекты. Функция scope отображает ссылку Binder на ссылку, которой соответствует некоторая сущность (она может относиться к любому "типу"), называемая областью видимости. Валидность функции scope определяется аналогично валидности target :

$$f = C.\text{scope} \Rightarrow \Box_C f \overset{def}{\leftrightarrow}$$
$$\overset{def}{\leftrightarrow} \begin{cases} l = \text{null} \vee \neg l[:]_C \text{ Binder} \to f\left(l\right) = \\ l[:]_C \text{ Binder} \to f\left(l\right)[:]_C \text{ Ent} \wedge \Box_C f\left(l\right) \end{cases} \quad (68)$$

Функция bnds используется для присоединения к сущности "типа" Binder набора связанных сущностей:

$$f = C.\text{bnds} \Rightarrow$$
$$\Box_C f \overset{def}{\leftrightarrow} \begin{cases} l = \text{null} \vee \neg l[:]_C \text{ Binder} \to \\ \to f\left(l\right) = L = f\left(l\right) \to \left|L\right| = \left|\text{Rng}\, L\right| \\ l[:]_C \text{ Binder} \wedge l' \in \text{Rng}\, f\left(l\right) \to \\ \to l'[:]_C \text{ Ent} \wedge \Box_C l' \end{cases} \quad (69)$$

Неформально смысл связывающей сущности b можно охарактеризовать следующим образом: ссылка на связанную сущность (т. е. из кортежа $\text{bnds}\, b$) может использоваться только в области видимости $\text{scope}\, b$, или в объекте, который является прямым или косвенным компонентом этой области видимости. Для уточнения этого свойства необходимо прежде всего формализовать отношения композиции и ассоциации.

Будем говорить, что узел l является компонентом узла l' (а l' соответственно контейнером l), если выполняется одно из следующих условий:

- l' – ссылка на список, и l – один из элементов этого списка;
- l' – ссылка на сущность, и l содержится в наборе атрибутов $C.\text{attrs}\, l'$;
- l' – ссылка на сущность, и l можно извлечь из l' при условии, что элементы, соответствующие собственным префиксам адреса, являются значениями;
- l' – ссылка на сущность "типа" Binder , и l либо совпадает с $C.\text{owner}\, l'$, либо содержится в наборе $C.\text{bnds}\, l'$.

Для формализации отношения композиции потребуется, кроме того, ввести расширенное множество адресов $\mathbf{X}$:

$$\begin{cases} \widehat{\mathbf{X}} \overset{def}{=} \mathbf{X} \cup \mathbf{X}', \\ \mathbf{X}' \overset{def}{=} \left\{\text{scope}\right\} \cup \text{attr}\, \mathbf{L} \cup \text{bnd}\, \mathbf{L}, \end{cases} \quad (70)$$

где $\mathbf{X} \cap \mathbf{X}' = \varnothing$.

Адрес вида $\text{attr}\, n$ обозначает атрибут, который является компонентом данной сущности и имеет идентичность n , адрес $\text{bnd}\, n$ – связанную

сущность с идентичностью n, scope – область видимости (последние два варианта применимы только в том случае, если данная сущность имеет "тип" Binder).

Таким образом, определение отношения композиции (обозначение $\prec_C$) можно представить в виде следующих правил вывода (в них подразумевается указанное выше условие $\mathrm{cntd}_C\, l \wedge \mathrm{cntd}_C\, l'$):

$$\frac{\mathrm{tag}\left(C.\mathrm{core}\, l'\right) = \mathrm{list}, l = \mathrm{eval}\left(l'\left[\left\langle n\right\rangle\right]_C\right)}{l \prec_C^{\langle n\rangle} l'} \quad (71)$$

$$\frac{l' :_C \mathrm{Ent}, l \in \mathrm{Rng}\left(C.\mathrm{attrs}\, l'\right)}{l \prec_C^{\langle \mathrm{attr}\, l\rangle} l'} \quad (72)$$

$$\frac{l' :_C \mathrm{Ent}, l \overset{[val]}{=} \mathrm{eval}\left(l'\left[X\right]_C\right), X \neq \left\langle\ \right\rangle}{l \prec_C^{X} l'} \quad (73)$$

$$\frac{b :_C \mathrm{Binder}, l = C.\mathrm{scope}\, b}{l \prec_C^{\langle \mathrm{scope}\rangle} b} \quad (74)$$

$$\frac{b :_C \mathrm{Binder}, l \in \mathrm{Rng}\left(C.\mathrm{bnds}\, b\right)}{l \prec_C^{\langle \mathrm{bnd}\, l\rangle} b} \quad (75)$$

Если конкретное значение адреса X не важно, запись $l \prec_C l'$ будем сокращать до $l \prec_! l'$:

$$l \prec_! l' \overset{def}{\Leftrightarrow} \exists\left(X \in \mathbf{X}^*\right)\left[l \prec_!^X l'\right] \quad (76)$$

Очевидно, что null не может быть контейнером для какой-либо ссылки.

Узел l называется корнем, если никакой другой узел не является его контейнером.

Расширим отношение $\prec_C$ на кортежи адресов (будем говорить о нем, как о косвенной композиции):

$$l \prec_C^{\langle\ \rangle *} l \quad (77)$$

$$\frac{l \prec_C^{X} l', l' \prec_C^{\mathrm{X}*} l''}{l \prec_C^{\langle X\rangle \mathrm{X}*} l''} \quad (78)$$

Отношение композиции в данном контексте должно быть согласовано с функцией owner этого же контекста. Сформулируем это ограничение в виде предиката $\mathrm{ver}_\prec C$, составленного из конъюнкции следующих утверждений (здесь подразумевается, что $\mathrm{cntd}_C\, l \wedge \mathrm{cntd}_C\, l'$):

$$\forall\left(l, l' \middle| l \prec_C l'\right)\left[C.\mathrm{owner}\, l = l'\right] \quad (79)$$

$$\forall\left(l, e \middle| e = C.\mathrm{owner}\, l \wedge e \neq \wedge e \neq \mathrm{null}\right) \times \times\left[l \prec_C e\right] \quad (80)$$

$$\neg l \prec_C^* l \quad (81)$$

Из этих свойств, в частности, следует, что ссылка l является корнем тогда и только тогда, когда $\mathrm{cntd}_C\, l$ и $C.\mathrm{owner}\, l = \mathrm{null}$.

Свойство исключает появление циклов в графе композиции при выполнении условия $\mathrm{ver}_\prec C$ (хотя в общем случае отношение косвенной композиции является рефлексивным – см.). В частности, ссылка на сущность в таком контексте не может быть своим собственным компонентом.

Путем к узлу l в контексте ! называется такая последовательность $\mathrm{Path}_C\, l \in \left(\mathbf{L}\right)^*$, что

$$\mathrm{Path}_C\, l = \left\{l_n\right\} \overset{def}{\Leftrightarrow} \begin{cases} l_0 = l \\ l_{n+1} = \begin{cases} C.\mathrm{owner}\, l, l_n \neq \\ , l_n = \end{cases} \end{cases} \quad (82)$$

Из определения непосредственно следует, что если $l_i, l_j \in \mathbf{L}$ и $i < j$, то первая ссылка является косвенным компонентом второй:

$$i < j \Rightarrow l_i \prec^* l_j \quad (83)$$

Заметим, что в валидном контексте в силу конечности $\mathrm{Dom}\ C.\mathrm{core}$ любой путь, начиная с некоторого номера, состоит только из. Действительно, если бы это было не так, то путь не содержал бы (иначе все последующие элементы также были бы равны), т. е. состоял бы из корректных ссылок. Пусть общее число ссылок в контексте $N = \left|\mathrm{Dom}\ C.\mathrm{core}\right|$, тогда по крайней мере l_N совпадает с одной из предшествующих меток – обозначим ее l_k. Но поскольку $l_k \prec^* l_N$, то равенство $l_k = l_N$ противоречит условию валидности контекста. Таким образом, любой путь, начиная с некоторого номера, состоит только из.

В силу сказанного мы можем заменить бесконечную последовательность кортежем, содержащим только элементы пути, отличные от null – такой кортеж мы также будем называть путем:

$$\begin{cases} \mathrm{path}_C : \mathbf{L} \to \mathbf{L}^* \\ \mathrm{path}_C\, l = \left\langle l_i \right\rangle_{i=0}^{n} \overset{def}{\Leftrightarrow} \mathrm{Path}_C\, l = \lambda n.\left[\leftarrow \left\langle l_i \right\rangle_{i=0}^{n}\right] \end{cases} \quad (84)$$

Обратным путем $\mathrm{htap}_C\, l$ называется кортеж, составленный из элементов пути, расположенных в обратном порядке.

$$\mathrm{htap}_C\, l = \left\langle l_{n-i-1} \right\rangle_{i=0}^{n} \overset{def}{\Leftrightarrow} \mathrm{path}_C\, l = \left\langle l_i \right\rangle_{i=0}^{n} \quad (85)$$

Перейдем теперь к отношению ассоциации. Назначение сущностей "типа" Wrapper состоит в хранении (посредством функции target расширенного контекста) узла, который не является компонентом данного контейнера (прямое хранение такой ссылки в списке, словаре сущности или соответствующей ей функции attrs при условии их валидности невозможно в силу ограничений, наложенных на отношение композиции). При этом сущности "типа" Wrapper, как и любые другие, могут иметь собственные компоненты.

Если узел l является контейнером для ссылки l', указывающей на сущность "типа" Wrapper и $C.\mathrm{target}\, l' = l''$, где l'' – узел, то l ассоциирован с (или использует) l'':

$$\frac{\mathrm{cntd}_C\, l, \mathrm{cntd}_C\, l'', l' \boxed{:}_C \mathrm{Wrapper}, l' \overset{X}{\prec}_C l, C.\mathrm{target}\, l' = l''}{l \overset{X}{\rightsquigarrow}_C l''}. \quad (86)$$

Заметим, что узел, доступная через ссылку на чистый объект, не рассматривается в качестве компонента сущности, содержащей эту ссылку (например, в своем словаре). В этом смысле чистые объекты для структуры, состоящей из узлов, играют роль "барьеров", разрывающих композиционные связи между ее элементами.

Ограничение на применение ассоциаций представим в виде предиката $\mathrm{obscured}(l, l')$, выражающего тот факт, что ссылка l не может использоваться ссылкой l'. Критерий валидности (обозначим его $\mathrm{ver}_{\rightsquigarrow} C$) при этом имеет вид:

$$\mathrm{ver}_{\rightsquigarrow} C \overset{def}{\Leftrightarrow} \neg\exists\left(l, l' \middle| l' \rightsquigarrow_{!} l\right) \mathrm{obscured}(l, l') \quad (87)$$

В рассматриваемой модели мы вводим единственное правило для предиката obscured. Это правило формализует рассмотренную выше семантику сущностей "типа" Binder.

$$\frac{b \boxed{:}_C \mathrm{Binder}, l \in \mathrm{Rng}(C.\mathrm{bnds}\, b), \neg(l' \prec_C^{*} C.\mathrm{scope}\, b)}{\mathrm{obscured}(l, l')}. \quad (88)$$

Помимо композиции и ассоциации мы вводим еще одно вспомогательное отношение зависимости представленное компонентом контекста deps. Ссылка на сущность e называется зависящей от ссылки на атрибут a в контексте C, если $a \in \mathrm{Rng}\, C.\mathrm{deps}\, e$. Валидность функции deps определяется следующим образом:

$$f = C.\mathrm{deps} \Rightarrow \Box_C f \overset{def}{\leftrightarrow}$$
$$\leftrightarrow \begin{cases} \neg l \boxed{:}_C \mathrm{Ent} \rightarrow f(l) = \\ f\,\mathrm{null} = \langle\ \rangle \\ L = f(l) \rightarrow |L| = |\mathrm{Rng}\, L| \\ l \boxed{:}_C \mathrm{Ent} \wedge l' \in \mathrm{Rng}\, f(l) \rightarrow \\ \rightarrow l' \boxed{:}_C \mathrm{Attr} \wedge \Box_C l' \wedge \neg\, \mathrm{obscured}(l', l) \end{cases} \quad (89)$$

Как и отношение ассоциативности, зависимость требует достижимости атрибута (выражаемой предикатом obscured).

Теперь мы можем конкретизировать предикат верификации контекста, определив его в виде:

$$\mathrm{ver}\, C \overset{def}{\Leftrightarrow} \mathrm{ver}_{\prec} C \wedge \mathrm{ver}_{\rightsquigarrow} C \quad (90)$$

Основное назначение атрибутов состоит в связывании с сущностями контекстно-зависимых наборов метаданных. Далее мы рассмотрим правила построения таких наборов, опираясь на введенные отношения композиции и ассоциации.

Пусть e – ссылка на сущность в контексте C. Локальным дескриптором e называется кортеж, образованный слиянием кортежей атрибутов, принадлежащих косвенным контейнерам e и самой сущности e, указывающих на e и взятых в порядке обратного пути e:

$$\mathrm{filter}\left(\mathrm{htap}_C\, e\right)\left(\text{»}l.l \boxed{\in}_C \mathrm{Ent} \setminus \mathrm{Null}\right) =$$
$$= \left\langle l_i \right\rangle_{i=1}^{n} \overset{def}{\Rightarrow} \mathrm{locdsc}_C\, e =$$
$$= \mathrm{concat}\left\langle \mathrm{filter}\left(C.\mathrm{attrs}\, l_i C.\mathrm{deps}\, l_i\right) \times \right.$$
$$\left. \times\left(\text{»}a.C.\mathrm{target}\, a = e\right)\right\rangle_{i=1}^{n} \quad (91)$$

Таким образом, локальный дескриптор описывает метаданные сущности, выводимые из ее расположения в композиционном графе контекста. Использование операции слияния означает, что один и тот же атрибут не встречается в дескрипторе более одного раза. Кроме того, поскольку порядок построения пути определяется обратным путем, атрибуты, введенные сущностями, находящимися ближе к e, имеют больший индекс в дескрипторе. Эти условия обобщают принятое во многих языках правило

приоритета определений, находящихся внутри некоторой конструкции по отношению к внешним определениям.

Гибкая модификация метаданных сущности достигается за счет возможности расширения локального дескриптора, атрибутами, доступными через композиционные связи узлов "типа" Wrapper , которые ссылаются на исходную сущность. Тем самым один и тот же объект может обладать различным набором метаданных в зависимости от того, какая ссылка используется для обращения к нему.

Для учета ассоциативных связей мы введем понятие полного дескриптора. В отличие от локального, полный дескриптор определен для сущностей-оберток, имеющих "тип" Wrapper (доступ к исходной сущности можно получить с помощью функции target), поскольку его структура зависит не только от самой сущности, но и от ее обертки.

$$\begin{gathered}\mathrm{fulldsc}_C\ w \overset{def}{=} \\ \overset{def}{=} \mathrm{concat}\left\langle \mathrm{locdsc}_C\ C.\mathrm{target}\ w, \mathrm{extdsc}_C\ w \right\rangle,\end{gathered} \tag{92}$$

где $\mathrm{extdsc}_C\ w$ – вспомогательное обозначение внешнего дескриптора:

$$\begin{gathered}\mathrm{filter}\left(\mathrm{htap}_C\ w\right)\left(\text{»}l.l \boxed{\in}_C \mathrm{Ent} \setminus \mathrm{Null}\right) = \\ = \left\langle l_i \right\rangle_{i=1}^{n} \overset{def}{\Rightarrow} \mathrm{extdsc}_C\ w = \\ = \mathrm{concat}\left\langle \mathrm{filter}\left(C.\mathrm{attrs}\, l_i C.\mathrm{deps}\, l_i\right) \times \right. \\ \left. \times \left(\text{»}a.C.\mathrm{target}\ a = e\right)\right\rangle_{i=1}^{n}\end{gathered} \tag{93}$$

Заметим, что поскольку обертки сами являются сущностями, для них определен также и собственный локальный дескриптор (однако он, вообще говоря, не имеет отношения к дескриптору сущности, на которую указывает обертка).

ЗАКЛЮЧЕНИЕ

Представленная модель объектной структуры может быть использована в качестве основы для реализации инфраструктуры метапрограммирования. Описанные в статье "типы" определяют набор структурных элементов, который представляется оптимальным с точки зрения основных шаблонов, возникающих при построении объектных моделей, в частности, отношений между объектами (композиция, ассоциация, зависимость). Кроме того, разработанный здесь механизм атрибутов, основанный на принципах контекстно-ориентированного программирования, позволяет обеспечить высокий потенциал расширяемости и модульности разрабатываемых моделей на уровне ядра соответствующей инфраструктуры.

В рамках проводимого автором исследования также разработаны теоретические основания вычислительных процессов, основанных на механизме объектных действий, поддерживающего модификацию структуры объектов и списков, манипуляции с метаданными контекста, а также обработку событий и, как следствие, отслеживание изменений в структуре моделей (с возможностью повторов и откатов). Однако подробное описание данного механизма выходит за рамки настоящей статьи.

СПИСОК ЛИТЕРАТУРЫ

1. *Clark A., Sammut P., Willans J.* Superlanguages. Developing Languages and Applications with XMF, 2008. http://itcentre.tvu.ac.uk/~clark/Papers/Superlanguages.pdf.

2. *Clark A., Sammut P., Willans J.* Applied Metamodeling. A Foundation for Language Driven Development (Second Edition), 2008. http://www.ceteva.com/home/Papers/Applied%20Metamodelling%20%28Second%20Edition%29.pdf.

3. *Gilad Bracha.* Newspeak Programming Language Draft Specification, Version 0.6, 2010. http://bracha.org/newspeak-spec.pdf.

4. *Chambers C.* The Diesel Language Specification and Rationale, 2006. http://www.cs.washington.edu/research/projects/cecil/www/Release/doc-diesel-lang/diesel-spec.pdf.

5. *Bruce K. B.* Foundations of Object-Oriented Languages. Types and Semantics. The MIT Press, 2002.

Седунов Алексей Александрович – ассистент кафедры программирования и информационных технологий Воронежского государственного университета, тел. 8-961-182-31-28, e-mail: alexey.sedunov@gmail.com

Sedunov A.A. – Assistant Department of Programming and Information Technologies, Voronezh State University, Tel. 8-961-182-31-28, e-mail: alexey.sedunov@gmail.com