УДК 519.7: 007 + 06

А.В. БЕЛЫХ, С.М. КОВАЛЕВ, О.В. ОЛЬХОВИК

ВИЗУАЛЬНО-ДЕКЛАРАТИВНЫЙ ЯЗЫК ДЛЯ ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

В работе представлен визуально-декларативный язык, N-Visual Language (NVL), предназначенный для проектирования программного обеспечения информационных систем: структуры базы данных и кода, реализующего бизнес-логику.
***Ключевые слова**: объектно-ориентированные базы данных, информационная система, визуальный язык, проектирование, класс, категория.*

Введение. Причины, побудившие нас к разработке NVL – это, с одной стороны, недостаточная выразительность для автоматической кодогенерации IDEF-нотаций, с другой – громоздкость стандарта UML, на что обращено внимание, например, в работе [1].

Теоретические основы NVL описаны нами в статье [2], декларативная состовляющая языка представлена в работе [3].

NVL предназначен для построения проектной диаграммы информационной системы (ИС), из которой автоматически генерируется структура базы данных и программный код для вычисления расчетных данных. Автоматическая генерация в таком случае позволяет:

- не разрабатывать проектные артефакты, описывающие поведение и алгоритмы для программных объектов, реализующих бизнес-логику;
- не писать вручную программный код для реализации бизнес-логики;
- автоматически поддерживать полное соответствие между проектными артефактами и программным кодом как в процессе разработки, так и во время эксплуатации системы.

NVL разработан так, чтобы для понимания проектной диаграммы и пользователи, и разработчики прикладывали минимальные усилия и тем самым могли проще находить общий язык. Для этого

- проект программного обеспечения информационной системы выполняется только одной проектной диаграммой;
- проектная диаграмма на уровне представления может разбиваться на любое количество субдиаграмм (подобно ER-диаграмме);
- в диаграмме допускаются идентификаторы, образованные с помощью национальных алфавитов;
- NVL прост и состоит всего из пяти визуальных элементов;
- визуальные элементы NVL аналогичны элементам из стандартов UML 2.2 [4] и IDEF1X [5] и поэтому интуитивно понятны не только разработчикам ИС, но и опытным пользователям.

При структурном подходе к проектированию ИС проектная диаграмма на NVL заменяет ER-диаграммы, диаграммы потоков данных (DFD) и диаграммы деятельности (IDEF3), описывающие программный код. Проектная диаграмма может связываться с моделью бизнес-процессов через спецификации входов и выходов на диаграммах IDEF0.

При объектно-ориентированном подходе к созданию программного обеспечения ИС проектная диаграмма может заменить диаграммы реализации UML для программного кода: классов (Class Diagram), кооперации (Collaboration Diagram) последовательности (Sequence Diagram), активности (Activity Diagram) и состояний (Statechart Diagram).

Практическое использование визуально-декларированного языка NVL. Прежде чем перейти к описанию языка, рассмотрим примеры его использования. Предметная область – организация выставок (упрощенный фрагмент). Компания на арендуемой территории проводит выставки, на которые в качестве экспонентов приглашаются юридические лица. Выставка характеризуется названием и сроками проведения. Выставка с одним названием может проводиться с периодичностью раз или два в год. За проведение выставки отвечает менеджер – сотрудник компании.

Экспонент подает заявку-договор на участие в выставке, где помимо собственных реквизитов указывает размер, расположение и тип арендуемого стенда (только одного). В зависимости от этих параметров рассчитывается сумма договора. Кроме того, в сумму договора входят обязательный взнос и аккредитация представителей экспонента. Для иностранных участников наценка составляет 25%. Для участников, оформивших заявку за девяносто дней до начала выставки, скидка 10%. Помимо суммы договора необходимо рассчитывать доходность выставок и количество заявок, с которыми работает каждый менеджер.

Проектная диаграмма для данного примера представлена на рис.1. Сумма договора-заявки рассчитывается в атрибуте «Общ_Сумма» класса «Заявка», который зависит от атрибутов «Сумма», «Наценка» и «Скидка». Атрибут «Сумма» определен в этом же классе и складывается из стоимости стенда, обязательного взноса и суммы аккредитации. Атрибут «Скидка» определен в самом классе как равный нулю, но перекрыт в категории «Ранняя_заявка». Поэтому для экземпляров класса «Заявка», попадающих в эту категорию по условию, определенному в ней, «Скидка» рассчитывается как 10%-ная от значения атрибута «Сумма».

В условии категории «Ранняя_заявка» используется функция IntDay, которая вычисляет разность между датами в днях. Эта функция не встроена в язык или в стандартный набор функций системы управления базами данных, но NVL позволяет использовать определенные программистами произвольные функции.

Атрибут «Наценка» определен в категориях класса «Заявка», которые образуют разбиение множества его экземпляров, поскольку связаны отношением исключения. Для экземпляров класса, попадающих в категорию «Зарубежная_заявка» (по условию), «Наценка» рассчитывается по заданной в этой категории формуле. Для остальных экземпляров, попадающих в категорию «Отечественная_заявка» (по исключению), наценка равна нулю, как это в ней определено.

Доход от выставки рассчитывается по суммам в договорах-заявках. Он определяется атрибутом «Доход» класса «Выставка» через ссылочные атрибуты «Стенды» и «Заявки», посредством которых можно получить доступ к атрибуту «Общ_Сумма» в свзязанных с каждой конкретной выставкой заявках. Эти ссылочные атрибуты неявно определены как обратные ссылки в классах «Выставка» и «Стенд» соответственно. Для образования обрат-

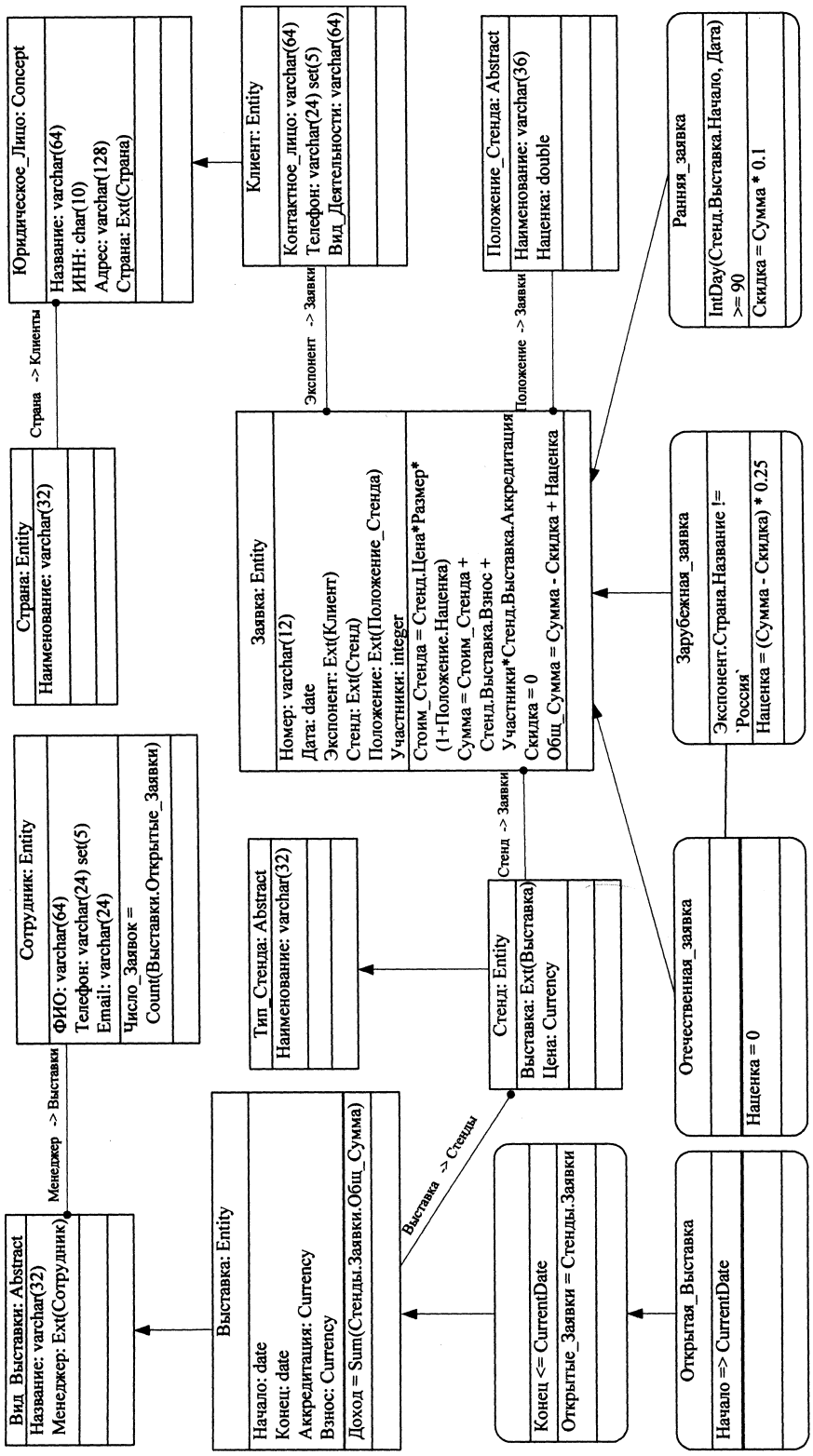

Рис.1. Проектная диаграмма «Организация выставок»

ных ссылок достаточно указать их имена в связи (после символа «->»). Функция SUM является стандартным агрегатом, суммирующим все полученные значения – в данном случае множества значений атрибута «Общ_Сумма» из связанных с выставкой заявок – для каждого экземпляра класса «Выставка».

Количество заявок, с которыми работает каждый из менеджеров, рассчитывается в атрибуте «Число_Заявок» класса «Сотрудник». Через ссылочный атрибут «Выставки» (неявно определенный как обратная ссылка) осуществляется доступ к атрибуту «Открытые_заявки». Стандартная агрегатная функция COUNT подсчитывает количество значений этого атрибута для каждого сотрудника.

Атрибут «Открытые_заявки» определен в категории класса «Выставка», который с классом «Сотрудник» непосредственно не связан. Тем не менее доступ к нему обеспечен благодаря механизму наследования, который позволяет не только в классе-потомке непосредственно обращаться к атрибутам класса-предка, но и в классе-предке также непосредственно обращаться к атрибутам классов-потомков. Например, если нужно получить список открытых на данный момент выставок (название, дата открытия, дата закрытия) на языке NDL [3], формулируется простой запрос:

```
Select Название, Начало, Конец
  from Выставка
    where CurrentDate between Начало and Конец;
```

Как видно, для атрибута «Название», хотя он и определен в другом классе, в запросе не надо делать никаких навигационных указаний – система, исходя из проектной диаграммы, итак поймет, какое значение и откуда нужно взять. Запрос можно сделать еще проще, если вспомнить, что в проектной диаграмме определена категория «Открытая_Выставка»:

```
Select Название, Начало, Конец from Открытая_Выставка;
```

В данном примере атрибут берется из класса-предка. В следующем примере продемонстрируем использование в запросах атрибутов классов-потомков. Как правило, осмысленные запросы требуют агрегации значений атрибутов, полученных из классов-потомков. Например, если надо узнать среднюю доходность по каждому виду выставок, можно выполнить запрос:

```
Select Название, AVG(Доход)
        from Вид_Выставки;
```

Но иногда и без агрегации обращение к атрибутам потомков может быть вполне информативным. Например, мы хотим получить список контрагентов, когда-либо принимавших участие в машиностроительных выставках (допустим, название таких выставок начинаеся со слова «Машиностроение»):

```
Select  distinct Стенды.Заявки.Экспонент.Название
   from Вид_Выставки
      where Название like 'Машиностроение%';
```

Кстати, эквивалентным по результату будет запрос:

```
Select  distinct Название from Клиент
  where exists (Заявки.Стенд.Выставка.Название
                  like 'Машиностроение%');
```

Очевидно, что и планы выполнения, и стилистика двух последних запросов совершенно различны, что дает разработчику вполне приличное пространство для маневра.

Последовательное описание NVL. Основной элемент проектной диаграммы – класс. Семантика класса описана в [2]. В NVL класс представлен прямоугольником, состоящим из четырех горизонтальных секций (рис. 2). В верхней секции записывается уникальное имя класса и через двоеточие – его тип: Concept, Abstract, Entity или State. Тип класса характеризует природу его экземпляров (абстракции, сущности, состояния). У класса типа Concept есть только один неявный экземпляр (понятие).

а) б)

Рис.2. Классы в NVL:
а – спецификация класса; б – пример класса

Во второй сверху секции описываются исходные атрибуты. Задаются их имена и типы данных (домены) или экстенсионалы для ссылок. В третьей секции определяются формулы расчетных атрибутов. По исходным атрибутам автоматически генерируется схема базы данных, по расчетным – программный код, реализующий бизнес-логику. Четвертая секция предназначена для спецификации методов. Описание атрибутов и методов выполняется согласно синтаксису декларативного языка NDL [3]. Имена атрибутов и методов уникальны внутри класса, а также внутри ветви наследования, за исключением случая явного перекрытия.

Категория – множество экземпляров класса, соответствующих некоторому логическому условию. Экземпляры класса задает пользователь. Множество экземпляров категории формируется автоматически из объектов, для которых условие категории в данный момент истинно. Категория изображается четырехугольником с закругленными углами, состоящим из четырех секций (рис.3). В первой секции записывается имя категории (может отсутствовать). Во второй секции задается логическое условие, определяющее категорию. Третья и четвертая секции такие же, как в классе.

а) б)

Рис.3. Категории в NVL:
а – спецификация категории; б – примеры категории

Между классами и категориями могут устанавливаться отношения трех типов: наследование, связь и исключение.

Наследование обозначается стрелкой, направленной от потомка к предку (рис.4). Формально семантика отношения определена нами в работе [2].

а) б)

Рис.4. Наследование в NVL:
а – спецификация наследования; б – пример наследования

Основные следствия установления наследования между классами и/или категориями следующие:

1. Отношение наследования является предпорядком.
2. Потомок наследует от предка методы и атрибуты со всеми ограничениями. При этом методы и атрибуты в потомке могут дополняться и перекрываться. Исходные атрибуты могут перекрываться только за счет сужения области значений. Условия в категориях также могут переопределяться только путем сужения области истинности в потомке.
3. Каждый экземпляр класса-потомка связан отношением наследования строго с одним экземпляром класса-предка. При этом значение каждого атрибута экземпляра-предка представляет собой объединение значений соответствующих атрибутов его потомков. Таким образом, наследование значений атрибутов происходит снизу вверх, а ограничений на значения – сверху вниз.
4. В классе (категории) непосредственно доступны все атрибуты, объявленные в нем самом, в его предках и в его потомках.
5. Атрибуты и методы Concept-класса неявно переопределяются (становятся терминальными в смысле [2]) в его непосредственных потомках.

С целью повышения качества проектирования на наследование введены ограничения, представленные в таблице.

Ограничения наследования

Объект	Может наследовать от:				
	Concept-класс	Abstract-класс	Entity-класс	State-класс	Категория
Concept-класс	+	-	-	-	-
Abstract-класс	+	+	-	-	-
Entity-класс	+	+	-	-	-
State-класс	-	-	+	+	-
Категория	-	+	+	+	+

Отношение связи соответствует связи один-ко-многим или многие-ко-многим в реляционных базах данных. Устанавливается только между классами. Связь изображается сплошной прямой или ломаной линией с кружком на конце у подчиненного класса (рис.5).

а) б)

Рис.5. Связи в NVL: а – спецификация; б – пример связи

Если прямая ссылка может иметь несколько значений (случай многие-ко-многим), кружок ставится с обоих концов линии. В этом случае выбор главного и подчиненного класса зависит только от предпочтений

проектировщика. Линия подписывается именами прямой и обратной ссылки, разделенными знаком «->». В подчиненном классе появляется явный ссылочный атрибут, через который происходит прямое связывание экземпляров подчиненного класса с экземплярами главного класса (внешний ключ). В главном классе при этом неявно образуется обратный атрибут-ссылка.

Отношение исключения связывает два объекта и означает, что их экстенсионалы не пересекаются. На проектной диаграмме оно устанавливается только между категориями. На практике его удобно использовать как предложение ELSE по отношению к условию, определяющему связанную категорию. Например, категория X определяется условием b и связана с категорией Y отношением исключения. Тогда категория Y определяется условием not b. Исключение изображается сплошной линией (рис.6).

а) б)

Рис.6. Исключение в NVL: а – спецификация; б – пример

В предложенном примере проектная диаграмма (см.рис.1) позволяет продемонстрировать почти все особенности и возможности NVL.

Рекурсивные вычисления разберем на примере другой упрощенной предметной области. Производственное предприятие выпускает изделия, состоящие из сборочных единиц. На изделия поступают заказы. Необходимо рассчитать подетальную плановую потребность (план) на каждую сборочную единицу и ее стоимость с учетом стоимости составных частей. Оба атрибута вычисляются рекурсивно. Проектная диаграмма представлена на рис.7.

Рис.7. Проектная диаграмма «Подетальное планирование»

Рекурсивные вычисления в общем случае выполняются, если:

1. Имеется связь между двумя классами в одной ветви наследования (в примере – это «Узел –> Части»).

2. Вычисление производится с помощью определения рекурсивного атрибута минимум двумя формулами, одна из которых рекурсивна. Примеры рекурсивных атрибутов: «Потребность» и «Стоимость».

3. Рекурсивная формула содержит конструкцию <ссылка на класс в ветви наследования>.<рекурсивный атрибут> или зависит от атрибута, формула которого имеет такую конструкцию. Пример первого случая – формула атрибута «Потребность» в классе «Часть», второго – формула атрибута «Стоимость» в категории «Узел».

4. Нерекурсивная формула должна быть определена в категории класса, в котором проводятся рекурсивные вычисления. В примере – это «Потребность» в категории «Изделие» и «Стоимость» в категории «Деталь».

Заключение. Разработана программная среда Information Systems Developer Studio (ISDS), которая позволяет строить проектные диаграммы на языке N-Visual Language и автоматически генерирует схему базы данных и программный код для расчетных данных. Сейчас проводится тестирование этой среды на практических примерах. Предварительно можно сказать, что процесс проектирования и разработки в ISDS проходит намного проще, легче и дешевле, однако необходимо решать проблему оптимизации генерируемого кода.

Библиографический список

1. Robert B. France, Sodipto Ghosh, Trung Dinh-Trong. Model-Driven Development Using UML 2.0: Promises and Pitfalls. //Computer. – 2006. – №2. – С. 59-66.

2. Белых А.В. N-Модель данных / А.В. Белых, О.В. Ольховик // Известия ЮФУ. Технические науки. Тематический выпуск "Интеллектуальные САПР". – 2009. – № 8.

3. Белых А.В. Декларативный язык для N-модели данных / А.В. Белых, С.М. Ковалев, О.В. Ольховик //Вестник РГУПС. – 2009. – №4.

4. OMG. Unified Modeling Language: Superstructure, version 2.2. URL: http://www.omg.org/cgi-bin/doc?formal/09-02-02.pdf (дата обращения: 10.09.2009).

5. Federal Information Processing Standards Publication 184. INTEGRATION DEFINITION FOR INFORMATION MODELING (IDEF1X). URL: http://www.idef.com/pdf/Idef1x.pdf (дата обращения: 10.09.2009).

Материал поступил в редакцию 18.05.09.

A.V. BELYKH, S.M. KOVALEV, O.V. OLHOVIK

VISUALLY-DECLARATIVE LANGUAGE FOR PROJECTION OF THE SOFTWARE OF INFORMATION SYSTEMS

Visually-declarative language, N-Visual Language (NVL), intended for projection of the software of information systems: database structure and a code realising business logic, is in-process introduced.

БЕЛЫХ Александр Валерьевич (р.1983), аспирант РГУПС, инженер сектора информатизации С-Кав. РДОП «Севкавэкспресс» сп ФПД - филиала ОАО «РЖД». Окончил Ростовский государственный университет путей сообщения (2005).
Область научных интересов: базы данных, информационные системы.
Имеет 4 публикации в этой области.

КОВАЛЕВ Сергей Михайлович (р. 1954), профессор кафедры «Автоматика и телемеханика на железнодорожном транспорте» РГУПС, доктор технических наук (2002). Окончил Таганрогский государственный радиотехнический университет (1976).
Область научных интересов: нечеткие системы, мягкие вычисления, искусственный интеллект и базы данных.
опубликовал 106 научных работ.

ОЛЬХОВИК Олег Владимирович (р. 1973), заведующий сектором внедрения информационных технологий в учебный процесс ИВЦ ДГТУ, кандидат технических наук (2000). Окончил Ростовский государственный университет путей сообщения (1995).
Область научных интересов: базы данных, информационные системы.
Имеет 12 публикаций.

white@donses.ru
kovalevsm@gmail.ru
olvick@spark-mail.ru