УДК 004.942:519.768

В.И. ПАВЛОВСКИЙ, А.Л. ЗИНЧЕНКО

Черниговский государственный технологический университет, Украина

ЛИНГВИСТИЧЕСКОЕ МОДЕЛИРОВАНИЕ КАК СРЕДСТВО ПОВЫШЕНИЯ НАДЕЖНОСТИ РАБОТЫ СО СЛАБОТИПИЗИРОВАННЫМИ ИЕРАРХИЧЕСКИМИ ДАННЫМИ

Рассмотрены основные способы моделирования структур данных. Отмечено, что при сложившемся подходе при переходе от одной модели к другой происходит потеря семантических связей между сущностями предметной области. Проведен аналитический обзор основных типов отношений. Отмечены преимущества и недостатки современных средств моделирования. Рассмотрена задача лингвистического моделирования слаботипизированных иерархических данных и способы верификации таких моделей. Определены требования к программному средству лингвистического моделирования. Приведена архитектура программного средства лингвистического моделирования слаботипизированных иерархических данных.

Ключевые слова: *лингвистическое моделирование, слабая типизация, иерархические данные, отношения, семантические связи, релевантность, верификация, модель.*

Введение

Эволюция современных информационных систем приводит к интенсивному росту интернет и корпоративных приложений, расширению круга пользователей. Все эти системы хранят огромные объемы информации. Данные хранятся на протяжении многих лет, активно используются и изменяются, что, повышает требования по контролю непротиворечивости наполнения и манипулирования контентом в новых программных приложениях (ПП). При этом важно подчеркнуть, что:

– современные базы (БД) или хранилища данных (ХД) содержат сведения о большом количестве сущностей моделируемых предметных областей (ПО) и имеют сложную схему данных;

– проектирование и модификацию ХД осуществляет сравнительно узкий круг высококвалифицированных аналитиков, архитекторов и администраторов БД. Они являются носителями информации о семантических связях между данными, которые в явном виде не представлены в БД из-за ограниченности используемых моделей данных. Этот недостаток стараются компенсировать за счет текстовых или схематических описаний, но гарантировать, что они являются полными, актуальными и однозначно понимаемыми всеми заинтересованными лицами на сегодня не приходится;

– разработчиками ПП является более широкий круг программистов, которые зачастую не имеют подготовки на уровне аналитиков и администраторов БД, но на уровне программного кода и запросов к БД должны обеспечить полноту и непротиворечивость семантических связей между сущностями моделируемой ПО;

– тексты запросов из ПП к БД остаются простыми знаковыми конструкциями до момента связи ПП с БД, что не позволяет выполнить проверку релевантности запросов к ХД до начала этапа выполнения программы.

Таким образом, в процессе разработки ПП возникают две проблемы:

– обеспечение полноты и непротиворечивости семантических связей между сущностями программы и запросов к БД и сущностями моделируемой ПО;

– обеспечение возможности проверки семантической релевантности запросов к ХД уже на этапе проектирования программы.

Именно этим и определяется живучесть и длительность жизненного цикла программных приложений.

1. Проблемы живучести и длительности жизненного цикла программных приложений

Широкая индустриализация программного обеспечения получила свое бурное развитие с появлением объектного подхода к созданию ПП – объектных ПП. Это существенно расширило круг разработчиков, и предоставило возможность представлять сущности ПО в виде инкапсулированных объектов.

К сожалению и в объектных ПП ответственность за поддержание полноты и непротиворечивость семантических связей между сущностями ПП и сущностями ПО, а так же обеспечение релевантности запросов к ХД на этапе проектирования приложения по-прежнему полностью лежит на программисте. Следовательно, для обеспечения живучести и длительности жизненного цикла объектных ПП, а также сокращения затрат труда на реализацию программистом большого количества программной логики актуальной является задача создания декларативных средств и программных инструментов, обеспечивающих:

1. Поддержание полноты и непротиворечивость семантических связей между сущностями ПП и ПО;
2. Возможность проверки релевантности запросов к ХД на этапе проектирования приложения.

Для хранения и обработки данных в современных ПП используются преимущественно реляционные и объектные СУБД, а также контент-репозитории (КР) разновидность объектных СУБД.

Таким образом, для решения поставленных выше задач структура данных БД должна:

– точно отражать структуру и семантику ПО;
– быть приспособленной для внесения изменений в семантику БД, связанных с расширением ПО.

После создания схемы данных модели выполняется ее реализация в виде полной спецификации с помощью программы на языке программирования (например, SQL/DDL) и создание электронного ХД, которое будет обеспечивать доступ к данным со стороны ПП и персистентность данных.

На сегодня широкое распространение получили системы визуального моделирования данных. При всех своих достоинствах они не предоставляют возможности отобразить (сохранить, выразить) все отношения и связи (семантику) моделируемой ПО. Причина в том, что отношения между объектами реальных систем богаче, чем набор отношений, которые можно описать, используя распространенные средства моделирования. К сожалению, на сегодня не существует ни одной формальной системы описания данных, которая позволяет полностью описать реальную ПО на уровне схемы данных. В результате, как правило, на этапе перехода от описания ПО к соответствующей модели данных происходит потеря ассоциативных связей между сущностями. Эти связи в последующем необходимо восполнять на программном уровне и в тексте запроса к БД.

Указанные недостатки приводят к тому, что программисту необходимо самостоятельно реализовать большое количество программной логики, что требует хорошего понимания семантики ПО и организации БД. Например, при использовании реляционных БД теряются такие связи ПО как род-вид (класс-подкласс), целое-часть и т.п., а в процессе нормализации реляционной БД теряются такие связи как транзитивная зависимость и т.п.

Причина указанных недостатков в том, что современные средства моделирования данных ориентированы на представление "предметных" (проблемных) связей между данными, например связь "поставка товара", которая выполняется для отдельных сочетаний экземпляров сущностей ПО, а не ассоциативных типа род-вид, которые существуют между отдельными понятиями-сущностями ПО. Таким образом, возникает вопрос о расширении средств описания ассоциаций между данными. Наличие таких средств, позволяет дополнить существующие модели данных отношениями, которые можно описывать самостоятельно, а затем проводить верификацию этих отношений в модели. Если пользователь предоставленными средствами сможет описать ассоциации между сущностями ПО, то можно будет сгенерировать средства контроля этих ассоциаций. Для решения указанной задачи на инструменты моделирования данных необходимо взглянуть с точки зрения полноты описания типов отношений между данными. Тип отношения – это данные (метаданные, метаинформация) о семантических связях между данными. Рассматривая тип отношения как специфические данные, можно достичь расширяемости множества отношений.

Среди всего множества иерархических, ассоциативных, поведенческих отношений, а также ограничений целостности и непротиворечивости, которые теряются при использовании современных средств моделирования данных можно, например, выделить следующие типы отношений [6]:

1. Иерархические (род – вид, главный – детальный и др.);
2. Ассоциативные (часть – целое, использование и др.);
3. Поведенческие (причина – следствие, описание жизненного цикла и др.);
4. Ограничение целостности и непротиворечивости данных (must, maybe и др.).

Используя, например, UML 2.0 можно построить модель с описанием всех перечисленных видов отношений, но нет возможности контроля соблюдения этих отношений в объектных ПП и реляционных БД, построенных на основании этой модели.

2. Система моделирования слаботипизированных иерархических данных

Программное средство моделирования данных должно обеспечивать:

– базироваться на уже существующих технологиях UML (Unified Modeling Language), MOF (Meta Object Facility) и XMI (XML Metadata Interchange);

– возможность описания и визуального представления семантических связей между сущностями предметной области;

– возможность расширения типов семантических связей между сущностями;

– средства контроля полноты и непротиворечивости семантических связей между сущностями ПП и сущностями моделируемой ПО;

– возможность проверки семантической релевантности запросов к ХД на этапе проектирования программы;

Одним из возможных решений компенсации потерь семантических связей ПО при ее представлении в БД является сопровождение БД специальным метафайлом, в котором на некотором языке, на сегодня наиболее удобным представляется язык XML, описываются связи, потерянные в результате моделирования ПО [4]. Этот метафайл является основой для генерации поведенческих классов, классов связей, типов ассоциаций и данных, а так же системы вспомогательных классов объектной ПП, которые восстанавливают потерянные связи между объектами программы, полученными в результате объектно-реляционного отображения БД, и обеспечивают контроль доступа и управления данными на уровне запросов к БД.

При моделировании ПО особо следует остановиться на таком понятии как слабая типизация, суть которой состоит в том, что в реальных системах один и тот же объект может находиться в разных иерархиях объектов и при этом выступать в разных отношениях, определяемых его местоположением в соответствующей иерархии. В результате один и тот же объект может характеризоваться несколькими ролями в своих ассоциациях. Например, некоторая личность может быть сотрудником университета, преподавателем, сотрудником научной лаборатории, членом ученого совета, тренером команды шахматистов, участником университетской команды по баскетболу и т.д. С другой стороны преподаватель находится в отношениях с читаемыми дисциплинами, с учебными потоками, с кафедрой и т.д.

Таким образом, один и тот же объект предметной области – сотрудник университета может выступать в реальной жизни в нескольких ролях – лектор и тренер, т.е. может существовать объект-лектор Иванов И.И., который, одновременно является тренером по шахматам – объект-тренер Иванов И.И., и наоборот.

Средствами UML можно описать данную иерархию, используя множественное наследование [1]. Вследствие этого роли объектов выносятся как отдельные классы, в результате чего в объектной модели появляются разные сущности, описывающие один и тот же объект реальной ПО, что приводит к неоднозначностям [6]. С целью устранения неоднозначностей в описании объектов в современных средствах объектного программирования (Java, C# и т.д.) от поддержки множественного наследования классов отказались. Однако для некоторого класса задач это требует специальных решений и большого количества программной логики. Так, например, в приведенном выше примере, для создания объекта, который одновременно является и лектором и шахматистом необходимо дополнительно создать два интерфейса и ILector и IChessPlayer и класс-фабрику, генерирующий эти объекты. С другой стороны, для получения спортивного разряда по шахматам объекта-лектора Иванова И.И., необходимо создать сложный запрос к БД со склейкой двух или более таблиц, в зависимости от структуры спроектированной БД.

Для устранения данного недостатка предлагается ввести новый тип отношение между классами – same. Диаграмма классов для данного примера приведена на рис. 1.

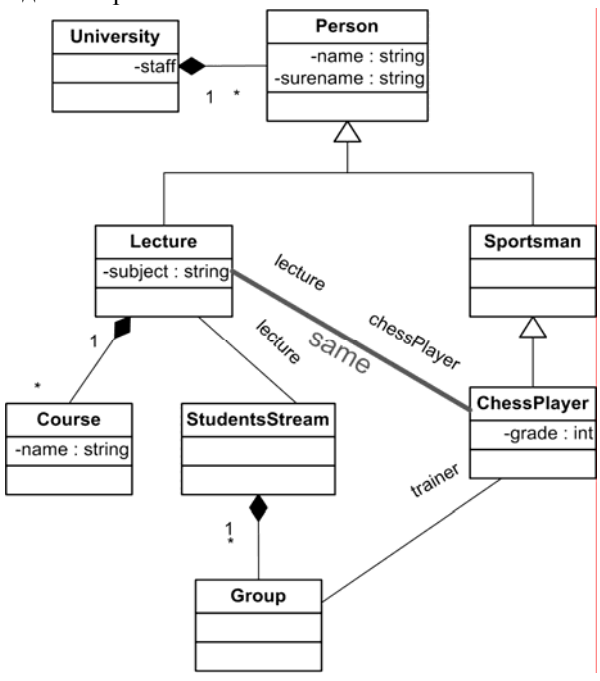

Рис. 1. Диаграмма классов ПО

На основании полученной модели будет сгенерирован XML файл с метаданными. Фрагмент этого файла приведен на рис. 2.

По XML описанию модели будут сгенерированы классы сущностей ПО [3]. В иерархию наследования дополнительно вводится базовый класс для всех сущностей модели – BasicObject, который явно не присутствует в диаграмме классов ПО.

Кроме того, в каждый класс, описывающий сущность ПО, добавляются свойства соответствующие отношению same. Фрагмент диаграммы классов приведен на рис. 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<lmi:Metamodel xmlns:lmi="http://lmi.stu.cn.ua/lmi"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <lmi:relations>
  <lmi:association id="1001">
   <lmi:name>same</lmi:name>
   <lmi:visability>Public</lmi:visability>
   <lmi:feature id="2001">
    <lmi:name>ua.cn.stu.lmi.sample.Lecture</lmi:name>
    <lmi:visability>Public</lmi:visability>
    <lmi:isNavigable>true</lmi:isNavigable>
    <lmi:aggrigation>None</lmi:aggrigation>
    <lmi:multiplicity>
     <lmi:upper>1</lmi:upper>
     <lmi:lower>1</lmi:lower>
    </lmi:multiplicity>
    <lmi:ordering>UnOrdered</lmi:ordering>
   </lmi:feature>
   <lmi:feature id="2002">
    <lmi:name> ua.cn.stu.lmi.sample.ChessPlayer</lmi:name>
    <lmi:visability>Public</lmi:visability>
    <lmi:isNavigable>true</lmi:isNavigable>
    <lmi:aggrigation>None</lmi:aggrigation>
    <lmi:multiplicity>
     <lmi:upper>1</lmi:upper>
     <lmi:lower>1</lmi:lower>
    </lmi:multiplicity>
    <lmi:ordering>UnOrdered</lmi:ordering>
   </lmi:feature>
  </lmi:association>
  <lmi:generalisation id="1002">
   <lmi:name>general</lmi:name>
   <lmi:visability>Public</lmi:visability>
   <lmi:child id="2003" xsi:type="lmi:ModelElement">
    <lmi:name>ua.cn.stu.lmi.sample.Person</lmi:name>
    <lmi:visability>Public</lmi:visability>
   </lmi:child>
   <lmi:parent id="2004" xsi:type="lmi:ModelElement">
    <lmi:name> ua.cn.stu.lmi.sample.Lecture</lmi:name>
    <lmi:visability>Public</lmi:visability>
   </lmi:parent>
  </lmi:generalisation>
 </lmi:relations>
 <lmi:elements>
  <lmi:element id="3001">
   <lmi:name> ua.cn.stu.lmi.sample.Lecture</lmi:name>
   <lmi:visability>Public</lmi:visability>
   <lmi:isAbstract>false</lmi:isAbstract>
  </lmi:element>
  <lmi:element id="3002">
    <lmi:name> ua.cn.stu.lmi.sample.ChessPlayer</lmi:name>
    <lmi:visability>Public</lmi:visability>
    <lmi:isAbstract>false</lmi:isAbstract>
  </lmi:element>
 </lmi:elements>
</lmi:Metamodel>
```

Рис. 2. Фрагмент XML файла с метаданными

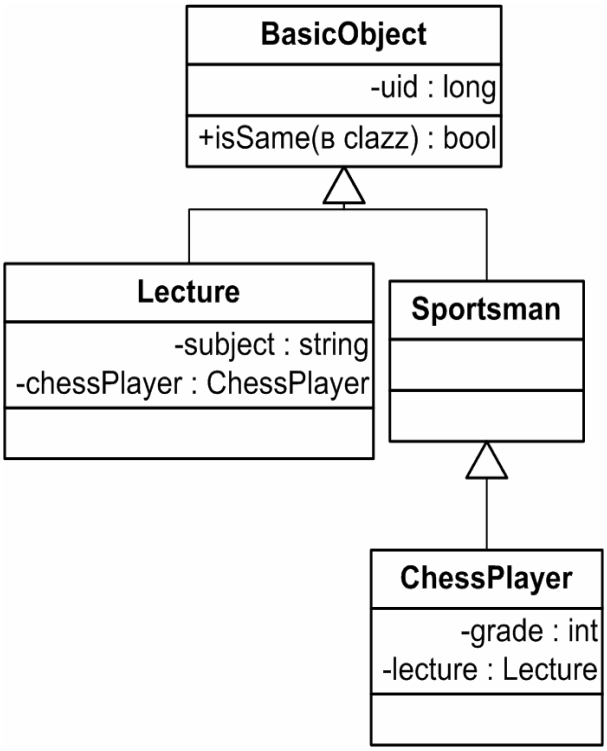

Рис. 3. Фрагмент диаграммы сгенерированных классов сущностей ПО

Следующий фрагмент кода (рис. 4) предназначен для получения этой информации.

```
/*Извлекаем из хранилища */
Lecture ivanov =LectureDAO.getByName("Иванов И.И.");

/*проверяем на наличие отношения*/
if (ivanov.isSame(edu.stu.lmt.sample.ChessPlayer))
   int grade = ivanov.getChessPlayer().getGrade();
```

Рис. 4. Фрагмент кода, получения разряда по шахматам лектора Иванова И.И.

3. Описание связи объектов с хранилищем данных

На основании модели будут сгенерированы классы сущностей ПО.

При использовании технологии JPA (Java Persistence API) для организации персистенции необходимо сгенерировать соответствующие аннотированные entity классы. Пример такого entity класса Lecture, представлен на рис. 5. Аналогично представляется класс ChessPlayer.

Для программной реализации отношения same в класс Lecture необходимо добавить transient свойство chessPlayer и добавить аннотацию, описывающую запрос к БД для извлечения данных об объекте шахматист. Фрагмент кода представлен на рис. 6.

```
@Entity
@Table(catalog = "sample", name = "lectures")
public class Lecture extends BasicObject{
   public Lecture() {   }
   //Другие конструкторы с параметрами
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "id", nullable = false, length = 10)
   public Integer getId() {  return id;  }
   public void setId(Integer id) {  super.id = id;  }
   private Person person;
   @OneToOne
   @JoinColumn(name="personid",referencedColumnName=”id”)
   public Person get Person() {  return person;  }
   public void setPerson(Person person){  this.person = person;  }
   private Collection<Course> courses;
   @OneToMany(mappedBy = "lecture", fetch =
FetchType.EAGER, cascade = CascadeType.ALL)
   public Collection<Course> getCourses() {  return courses;  }
   public void setCourses(Collection<Courses> courses) {
      this. courses = courses;  }
}
```

Рис. 5. Entity Class Lecture

Архитектура программного средства, которое осуществляет генерацию файла с метаинформацией, аннотированных классов сущностей и дополнительных классов представлена на рис. 7.

```
@NamedQueries({@NamedQuery(name="Lecture.getChessPlaye
r",
 query= "SELECT c FROM ChessPlayer c, Person p WHERE
p.name=:name AND p.surname=:surname AND
p.id=c.personid")})
   private ChessPlayer chessPlayer;

   @Transient
   public ChessPlayer getChessPlayer () {
      return chessPlayer;
   }
}
```

Рис. 6. Дополнительное свойство и аннотация для класса Lecture

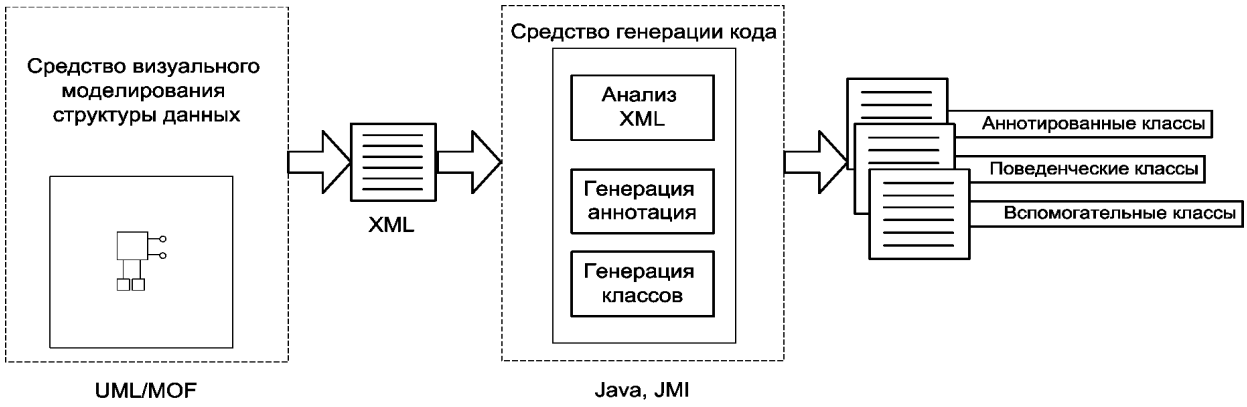

Рис. 7. Архитектура программного средства моделирования

Заключение

В статье предложен подход к решению задачи расширения средств описания ассоциаций между данными, которые отсутствуют в современных моделях данных, и получения на их основе соответствующих классов сущностей ПО, поведенческих и вспомогательных классов.

Наличие таких средств, позволяет дополнить существующие модели данных отношениями, которые можно описывать самостоятельно, а затем проводить верификацию этих отношений в модели.

Если пользователь предоставленными средствами опишет ассоциации между сущностями ПО, то по этому описанию будут сгенерировать средства контроля введенных ассоциаций, что позволит повысить надежность разрабатываемых программных систем.

В частности в статье показан процесс генерации соответствующих классов для поддержки семантической связи между объектами типа same.

Литература

1. UML – Unified Modeling Language, UML 2.0 Infrastructure Specification [Электронный ресурс]ю – 2005. – 405 р. – Режим доступа к ресурсу: http://www.omg.org/spec/UML/2.

2. MOF – Meta Object Facility, MOF 2.0 Specification [Электронный ресурс]. – 2004. – 345 р. – Режим доступа к ресурсу: http://www.omg.org/spec/MOF/2.0.

3. XMI – XML Metadata Interchange, XMI 2.0 Specification [Электронный ресурс]. – 2005. – 440 р. – Режим доступа к ресурсу: http://www.omg.org /spec/XMI/2.0.

4. CWM 1.1 Specification [Электронный ресурс]. – 2003. – 210 р. – Режим доступа к ресурсу: http://www.omg.org/spec/CWM/1.1.

5. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++ / Г. Буч. – М.: Бином, 1998. – 560 с.

6. Чен П. Модель "сущность-связь" – шаг к единому представлению о данных / П. Чен // СУБД. – 2007. – № 3. – С. 137 – 158.

Поступила в редакцию 2.02.2009

Рецензент: д-р техн. наук, проф., заведующий кафедрой специализированных компьютерных систем Тарасенко В.П., Национальный технический университет Украины «КПИ», Киев.

ЛІНГВІСТИЧНЕ МОДЕЛЮВАННЯ ЯК ЗАСІБ ПІДВИЩЕННЯ НАДІЙНОСТІ РОБОТИ ЗІ СЛАБОТИПІЗОВАНИМИ ІЄРАРХІЧНИМИ ДАНИМИ

В.І. Павловський, А.Л. Зінченко

Розглянуто основні способи моделювання структур даних. Відзначено, що при сучасному підході під час переходу від однієї моделі до іншої відбувається втрата семантичних зв'язків між сутностями предметної області. Наведено аналітичний огляд основних типів відношень. Відзначено переваги та недоліки сучасних засобів моделювання. Розглянута задача лінгвістичного моделювання слаботипізованих ієрархічних даних та засоби верифікації таких моделей. Визначені вимоги до програмного засобу лінгвістичного моделювання. Наведена архітектура програмного засобу лінгвістичного моделювання слаботипізованих ієрархічних даних.

Ключові слова: лінгвістичне моделювання, слабка типізація, ієрархічні дані, відношення, семантичні зв'язки, релевантність, верифікація, модель.

RELIABILITY ENHANCING OF WEAK TYPED HIERARCHICAL DATA PROCESSING BY LINGUISTIC MODELING

V.I. Pavlovsky, A.L. Zinchenko

Basic modeling means of data structures was examined. Marked that traditional approach causes looses of semantic links between objects of modeling in process of transition from one model to another. Analytical review of relations basic types' is done. Drawbacks and advantages of modern modeling means are marked. Task of linguistic modeling of weak typed hierarchical data and verification methods of such models was evaluated. Requirements are defined for linguistic modeling software. Software architecture has been proposed for linguistic modeling of weak typed hierarchical data.

Key words: linguistic modeling, weak typification, hierarchical data, relations, semantic links, relevant, verification, model.

Павловский Владимир Ильич – канд. техн. наук, доцент, заведующий кафедрой информационных и компьютерных систем Черниговского государственного технологического университета, Чернигов, Украина, e-mail: Pavlovsky@stu.cn.ua.

Зинченко Андрей Леонидович – старший преподаватель кафедры информационных и компьютерных систем Черниговского государственного технологического университета, Чернигов, Украина, e-mail: a.zinchenko@stu.cn.ua.