Метод объектно-ориентированного представления многоуровневых семантических моделей

С. Э. Грегер[*], *С. В. Поршнев*[**]

[*]*Нижнетагильский технологический институт (филиал) ГОУ ВПО «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина»*
622013 Свердловская обл., Нижний Тагил, Красногвардейская, 59,

[**]*ГОУ ВПО «Уральский федеральный университет имени первого Президента России Б. Н. Ельцина», Екатеринбург.*
620000, г. Екатеринбург, ул. Ленина, 51

e-mail: segreger@gmail.com s.v.porshnev@urfu.ru

Аннотация. Рассматриваются особенности описания семантики и разработки программной реализации компонента для представления элементов многоуровневых семантических моделей при использовании объектно-ориентированного подхода. Обсуждается применение понятий power type, clabject и метода многоуровневого объектно-ориентированного отображения при многоуровневом семантическом моделировании.
Ключевые слова: многоуровневых семантические модели, объектно-ориентированные отображения, объектно-ориентированные системы.

1. Введение

При разработке моделей предметной области, обладающей сложной структурой, состоящей из нескольких логических уровней, традиционно руководствуются стандартом MOF (Mega-Object Facility) и используют языки моделирования, основанные на метамоделях и поддерживающие ограниченное количество уровней моделирования. Известные технологии метамоделирования, например EMF, предоставляют возможность использования двух уровней классификации: метамодель, предоставляющую классы, и модель, содержащую объекты классов метамодели. При этом для изменения доступен, как правило, только уровень модели. В то же время мультиуровневый подход к проектированию не ограничивает число уровней, однако при наличии нескольких уровней требуется уточнение характера соотношений между классами и объектами, а также способ объектно-ориентированного представления, не зависящий от числа уровней. В этой связи рассмотрение вопросов, связанных с семантической интероперабельностью многомерных моделей и их объектно-ориентированного представления, является актуальным.

2. Существующие подходы к решению проблемы

В [1] была предложена методология разработки программного обеспечения, для практического использования которой была создана метамодель ISO24744, содержащая понятия *power type* и *clabject*. Цель разработки данной методологии состояла в обеспечении возможности построения множества классификационных уровней, объединенных в мультиуровневую модель. Основная проблема, решенная в обсуждаемой методологии, состояла в построении унифицированного и адаптируемого метода классификации элементов предметной области. (Здесь под унификацией понимается наличие возможности отображения всех моделей всех уровней моделирования через ограниченный набор концепций и символов.) Адаптируемость обеспечивается тем, что все концепты моделей каждого из уровней представляются в виде редактируемых данных, которые могут быть изменены в процессе взаимодействия. Изменение элемента на любом уровне немедленно отражается на всех связанных с ним элементах. В методологии введены три уровня моделирования – метамодели, метода и проекта. Структурная схема данной многоуровневой модели представлена на рис. 1.

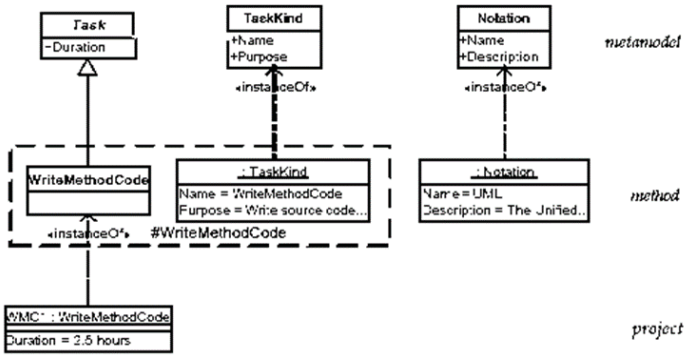

Рисунок 1. Многоуровневая модель

Из рис. 1 видно, что между элементами *power type Task/*Task* и *clabject #WriteMethodCode* установлено отношение *instanceOf* и, те самым, определен класс *Task/*Task*, специализирующий класс «паттерн *power type*», который на уровне проекта представлен объектом WMC1. Важно, что атрибуты объекта зеркально не отражают атрибуты clabject.

Проводя исследование особенностей процесса метамоделирования, авторы отмечают, что, следуя концепции «строгого метамоделирования», в большинстве ме-

тодологий классы метамодели используются для построения классов уровня методов путем их специализации, проводимой через установление отношения реализации (instanceOf). Классы уровня метода реализуются через объекты уровня проекта. Это создает противоречие: методология представляется коллекцией классов, а проект — коллекцией объектов. В то время как способы «волшебного» превращения классов в объекты в методологиях не определены, так как методология позволяет только осуществлять контроль над преобразованием, проводимым на границе между уровнем метода и уровнем проекта.

Классы уровня метамодели рассматриваются здесь как «сильные типы» (*power type*) для классов слоя метода. Каждому объекту *power type* класса ставится в соответствие группа объектов класса слоя метода, наследующего *power type* класс. При этом классы уровня метода становятся *partitioned* и определяют разделение на группы объекты в слое проекта. Таким образом, в то время как *power types* задают разделение объектов в слое метода, *partitioned*-типы задают разбиение объектов в слое проекта. Авторы назвали такое представление двойным представлением. Строгое наследование частично подменяется прототипированием — следованием некоторому образцу, при котором наличие связи между объектами *power type* классов на уровне метамодели, задающих классификацию, определяет необходимость установления связи такого же типа между классами уровня метода, классифицированными этими объектами. Каждому классу и его объекту уровня проекта ставится *partitioned* класс уровня метода и объект *power type* класса уровня метамодели. Такая композиция и получила название *clabject*. Моделирование в понятиях *clabject* обеспечивает возможность определения в метамодели характеристик элементов и правил их организации как уровня метода, так и уровня проекта, и их передачу на необходимые уровни вниз через специализацию и классификацию, вверх — через создание иерархий.

Atkinson и Kühne [2] ввели понятие *potency* и метод *deep instantiation* порождения объектов между уровнями иерархии специализации, основанный на понятии возможностей (*features*). При этом «традиционный» метод порождения объектов из классов, присущий объектно-ориентированному подходу, рассматривается как частный случай более общего механизма, названного «*deep instantiation*», при котором производится специализация между *clabjects*, а не между классами и объектами. При специализации *clabject A* на основе некоторого *clabject B* каждой возможности (*feature*) элемента, представленной его атрибутами и связями, приписывается числовой атрибут *potency*, значение которого определяет способ трансформации возможности элемента. Для каждой возможности элемента A со значением *potency* > 0 в элементе B определяется возможность со значением *potency*, уменьшенным на 1. При нулевом значении *potency* возможность атрибута элемента A представля-

ется в элементе B как атрибут простого типа, а ассоциация элемента A представляется в элементе B ссылкой на элемент с *potency* = 0. На всех уровнях моделирования элементы уровней характеризуются атрибутами двух типов — с *potency* > *0* и с *potency* = 0. Проектный уровень представлен элементами, с атрибутами у которых *potency* = 0. Введение понятия "*potency"* позволило избавиться от отношения *power type* и определить способ образования объектов проектного уровня из классов уровня метода с учетом спецификаций, задаваемых уровнем метамодели. Таким образом было показано существование унифицированного концепта (*clabject*), позволяющего проводить многоуровневое моделирование для различных предметных областей. Для обеспечения многоуровневого моделирования была предложена архитектура ортогональной классификации OCA (Orthogonal Classification Archtectute), основанной на применении принципов строгого метамоделирования и deep *instantiation*. Между уровнями в многоуровневой модели устанавливаются отношения классификации, когда класс более высокого уровня рассматривается как классификатор классов нижних уровней [3].

3. Постановка задачи

С точки зрения многоуровневого семантического моделирования существуют семантические сети, связывающие классы *C* и индивидуалы этих классов *O* посредством связей *R*. В качестве атомарного элемента обычно используют так называемый RDF-триплет (RDF — Resource Description Framework) — набор информационных сущностей «объект-предикат-субъект». Предметная область, описанная таким способом, обладает компьютерной семантикой, т. е. имеется возможность устанавливать и обрабатывать смысловые отношения между понятиями с помощью программных алгоритмов.

Семантическое или онтологическое моделирование рассматривается как способ использования гетерогенных информационных ресурсов с целью организации их в рамках единого информационного пространства и как способ организации взаимодействия разнородных информационных и инструментальных систем.

Решение проблемы семантической интероперабельности видится в создании универсальных, кросс-платформенных информационных структур, использующих семантические метаописания данных, аннотирующих как собственные, так и внешние разнородные информационные массивы. Процесс описания предметной области сопровождается привязкой к ней спецификаций, выраженной в концептах некоторой онтологии, согласованием понимания предметных областей взаимодействующих ресурсов и построения их взаимодействия, основываясь на согласованной семантике предметной области. Унификация информационных моделей является необходимым шагом при работе с неоднородными онтологическими спецификаци-

ями. Модели приводятся к некоему каноническому однородному представлению, в котором будут производиться все дальнейшие манипуляции спецификациями. Модели онтологий называются исходными, а каноническая модель — целевой. Задача унификации множества исходных информационных моделей становится актуальной при необходимости масштабирования по количеству неоднородных моделей. При унификации моделей должно быть построено отображение исходных моделей в целевую. При отображении моделей производится поиск близких конструкций моделей и их выражение друг через друга [4].

Существует необходимость в декомпозиции семантической модели на атомарные более крупные элементы — триплеты. Частично эта проблема была решена в языке OWL введением понятий онтологических классов и онтологических связей в версии языка OWL 2, позволяющей наследовать онтологические классы от индивидуалов онтологических классов. Стало возможным использование конструкций, в которых онтологический класс имеет фреймы, связанные как с другими онтологическими классами, так и с индивидуалами онтологических классов. Структурно такие конструкции подобны *clabjects*. Но новые возможности онтологического моделирования практически не поддерживают существующие инструменты онтологического моделирования. С другой стороны, при разработке инструментария для семантического моделирования в основном используются объектно-ориентированные методологии и технологии. При этом обычно создается некоторая семантическая модель в рамках некоторой онтологии, примитивы и результаты моделирования представлены в виде RDF-триплетов, хранимых в реляционной базе данных. В соответствии с традиционной технологией объектно-реляционного отображения каждому онтологическому классу, представленному набором триплетов, ставится в соответствие объектно-ориентированный класс [5], что связано с высокой трудоемкостью разработки, все еще плохо поддающейся автоматизации. Для повышения эффективности необходимо, чтобы каждой семантической конструкции когнитивного уровня рассуждения был сопоставлен набор унифицированных объектов, размещенный в памяти соответствующего инструмента, автоматически создаваемого в соответствии с моделью, также представленной в виде семантической сети. В этом случае появляются следующие возможности:

- проектирование компьютерных систем можно осуществлять на основе унифицированных логико-семантических моделей, рассматривая разработку логико-семантической модели проектируемой системы как первый этап ее проектирования;
- обеспечить модульную (компонентную, крупноблочную) разработку логико-семантических моделей компьютерных систем на основе биб-

лиотек совместимых типовых многократно используемых компонентов (онтологий, логических операций и т. д.);

- рассматривать различные варианты технической реализации компьютерных систем как различные способы интерпретации унифицированных логико-семантических моделей компьютерных систем, что дает возможность разрабатывать такие интерпретаторы независимо от проектирования конкретных систем и включать эти интерпретаторы в состав среды проектирования;
- обеспечить полную совместимость средств проектирования с проектируемыми системами — среда проектирования строится как интеллектуальная система на основе унифицированных логико-семантических моделей;
- включить в состав среды проектирования компьютерных систем комплекс интеллектуальных help-систем, ориентированных на повышение квалификации разработчиков;
- ориентироваться на методику поэтапного эволюционного проектирования компьютерных систем на основе быстрого создания прототипов [6].

4. Компоненты хранения объектной семантической сети

Для реализации подхода, описанного в предыдущем разделе, необходимо принять решение о объектной модели унифицированного элемента, используемого для представления крупных элементов семантической сети как в оперативной, так и в долговременной памяти программного инструмента. Для построения систем проектирования, реализующих данный подход, необходимы программные компоненты и сервисы, обеспечивающие создание, хранение и управление унифицированными семантическими сетями, представленными в виде сетей унифицированных элементов.

В [7] на основе объектно-ориентированной модели семантической сети [8] нами был предложен набор компонентов *CADT*={*Ontology, OntoClass, DatProperty, ObjectProp, ClsDataProperty, ClsObjectProperty, OntoIndividual, IndDataProperty, IndObjectProperty*}. Диаграмма классов набора представлена на рис. 2. Компоненты предназначены для представления семантической сети в объектно-ориентированной среде и хранения представления в объектно-ориентированной базе данных [9]. Было показано, что набор компонентов обеспечивает отображение высказываний языка абстрактной семантической сети в сеть объектов и позволяет создавать и сопровождать базы знаний через интерфейс пользователя веб-приложения. В настоящее время этот набор компонентов изменен.

Семантическая сеть *G*={{*Ci*}, {*Rj*}} рассматривалась нами как множество узлов C и связей R, причем узел имеет тип из множества типов узлов *TC*={*Onto, OCls,OntoInd*}, а связь — тип из множества типов связей *TR*={*DatProp, ObjProp, ClsDatProp, ClsObjProp,IndDatProp, IndObjProp*}. В настоящий момент этот набор компонентов подвергся изменению с целью введения возможности взаимного семантического аннотирования элементов семантической сети. Эта возможность позволяет строить многоуровневую семантическую сеть, представляя ее в памяти как многоуровневую объектную модель, в соответствии с архитектурой ОСА.

Определим семантику узлов и связей:

- *Onto* — множество онтологических модулей *O*;
- *DP* — множество элементов типа DatProp, определяющих сорта описания данных;
- *P* — множество элементов типа *ObjProp*, определяющих сорта связей;
- *OCls* — множество элементов типа *OntoCls*;
- *OI* — множество элементов типа *OntoInd*;
- *O=<title,id ,OC, DP, OP,OI>* — онтологический модуль, представляющий сеть более низкого уровня, *title* — наименование узла, *id* — идентификатор узла;
- *OntoCls=<title,id,subClassOf,hasKindOf,rootclass,ClsDP,ClsOP>* — элемент сети, представляющий класс онтологии, *subClassOf* — множество классов, являющихся родительскими, *hasKindOf* — множество классов и объектов, аннотирующих класс семантическим описанием, *rootclass* — атрибут, указывающий на уровень в иерархии классов в модуле;
- *DatProp=<title, id, range:XMLSchemeDatetype>* — тип связи, определяющий характер представления элемента сети простым типом данных, где *range* — множество классов, наследующих класс *XMLSchemeDatetype*, представляющий семейство простых типов, определяет верхний уровень в иерархии элементов типа *ClsDP*;
- *ObjProp=<title,id,range:NotXMLSchemeDatetype>* — тип связи, определяющий характер связи элемента сети с элементами, не принадлежащими множеству простых типов данных, определяет верхний уровень в иерархии элементов типа *ClsOP*;
- *ClsDP=<title,id,parent,dataProperty:DProp,range>* — элемент, представляющий множество значений атрибута класса *parent, dataProperty,* указывает на элемент в частной иерархии элементов *ClsDP*, *range* — ограничение, дополнительно накладываемое на множество ограничений на тип простых значений, определяемое иерархией *dataProperty*;

– *ClsOP=<title,id,parent,SubPropertyOf,range>* — элемент, представляющий множество допустимых связей атрибута с именем *title* класса *parent*, *SubPropertyOf* указывает на элемент в частной иерархии элементов *ClsOProp*, *range* — ограничение, дополнительно накладываемое на множество ограничений, на тип связываемых элементов, определяемое иерархией *SubPropertyOf*;
– *OntoInd=<title,id,sourceClass,IndDP,IndOP>* — экземпляр класса онтологии, где *sourceClass*— родительский класс для экземпляра;
– *IndDP=<title,id,parent,dataProperty,range>* — элемент, представляющий множество значений атрибута класса *parent*, *dataProperty* указывает на элемент в частной иерархии элементов *ClsDP*, *range* — ограничение, дополнительно накладываемое на множество ограничений, на тип простых значений, определяемое иерархией *dataProperty*;
– *IndOP=<title,id,parent,SubPropertyOf,range>* — элемент, представляющий множество допустимых связей атрибута с именем *title* класса *parent*, *SubPropertyOf* указывает на элемент в частной иерархии элементов *ClsOProp*, *range* — ограничение, дополнительно накладываемое на множество ограничений, на тип связываемых элементов, определяемое иерархией *SubPropertyOf*.

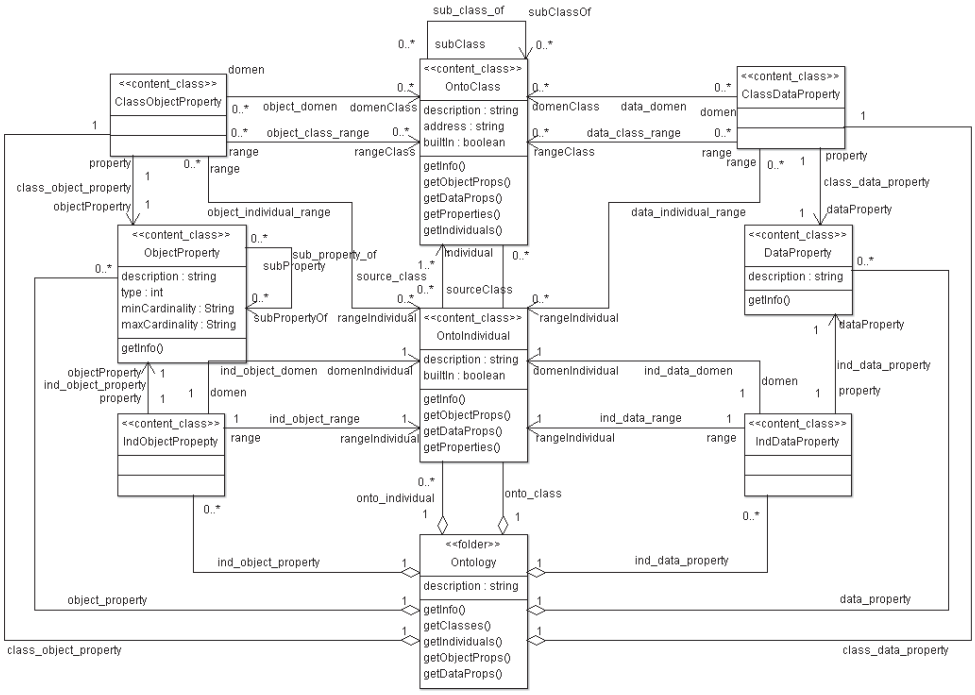

Рисунок 2. Диаграмма классов набора компонентов

Связи классов различных уровней моделирования с использованием разработанного отображения и классов для компонента, отображающего элемент, представляющий *clabject*, представлены на рис. 3. Для упрощения изображения классы, представляющие связи, изображены линиями. В архитектуре OCA каждому атрибуту и связи приписывалось значение числового атрибута *potency*. При этом максимальное значение *potency* имели атрибуты и связи самого верхнего уровня моделирования.

5. Алгоритм многоуровневого моделирования

При создании модели в разработанном нами редакторе производится последовательное создание экземпляров компонентов. В соответствие с предложенным подходом отношение онтологической классификации *power type* реализуется атрибутом *hasKindOf* класса *OntoClass*. Экземпляр класса *OntoClass* является контейнером для набора экземпляров типа *ClassObjectProperty*, каждый из которых хранит информацию о связываемых элементах модели. В связи с тем, что во многих случаях число уровней моделирования не известно, начальное значение *potency* не определено. Для преодоления этого недостатка мы ввели иерархию классификации между связями, определяемую атрибутом *SubPropertyOf*.

Рассмотрим последовательность $[C_{i1}, C_{i2}, \ldots C_{in}]$ классов, наследующих друг друга и расположенных различных уровнях моделирования, т. е $C_{ik}.SubClassOf \rightarrow C_{ik-1}$.

Для оптимизации программы мы ввели дополнительный атрибут *hasMetaClass*, определяющий наследование между классами, находящимися в разных онтологических модулях, т. е. атрибуты *SubClassOf* и *hasMetaClass* семантически эквивалентны, но не могут быть использованы одновременно. Классу C_{ik} сопоставлено множество $\{R_{ij}\}_k$ связей, представленных элементами типа *ClassObjectProperty* и *ClassDataProperty*. При этом при наследовании классов устанавливается отношение $R_{ijk}.SubPropertyOf \rightarrow R_{ijk-1}$, а каждая связь определяет ограничения для области определения своих значений, ограничение может быть рассчитано как $R_{ijk}(R_{ijk-1}(..R_{ij1}(Rij1.range)))$.

Ограничения связи может быть указано в атрибутах связи *R* или определено в классе или индивидуале класса классификатора, определяемого отношением *hasKindOf*.

При выборе элемента модели, определяющего связь, в памяти системы ищется объект типа *ClassObjectProperty* и строится иерархия объектов, связанных отношением *SubPropertyOf*. Характер связей определяется соответствующей моделью ограничений связей *M_Constrain*, включающей в себя классы ограничений *C_Constrain*. Класс ограничений *C_Constrain* определяется для каждой связи узла *f*. Связь *f* и класс ограничений связаны отношением *SubPropertyOf*. Модель ограниче-

ний строится как объединение классов ограничений текущей связи и всех классов ограничений связей, входящих в иерархию связей, также определяемую отношением *SubPropertyOf*. Ограничения нижнего уровня перекрывают ограничения верхнего уровня. Правила согласования ограничений в настоящее время определены в коде редактора. Предполагается в дальнейшем правила интерпретации ограничений представить в виде соответствующей модели и редактировать их подобно другим моделям.

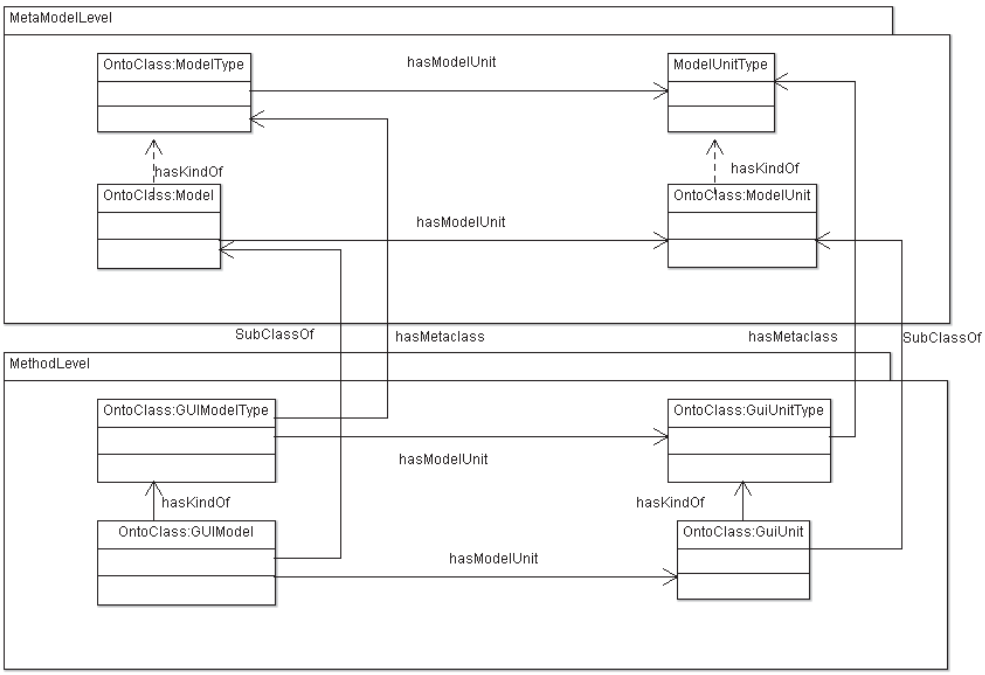

Рисунок 3. Связь классов различных уровней моделирования

Задание типа связи для верхнего уровня иерархии связей определяет поведение всех элементов иерархии. Так, в онтологии системы управления знаниями O_{KW}, включающей понятия **Метатип**, **Тип**, **Класс**, **Подкласс**, **Экземпляр** определим тип связи **Порождает**, модель ограничения которой определяет способ создания класса понятия на основе некоторого класса, определяющего метаинформацию понятия. Определим набор классов связей, расширяющих ее: ***метатип порождает тип***, ***класс порождает подкласс***, ***класс порождает экземпляр***. Ограничения, накладываемые на связи, определяют правила создания, расширения и интерпретации класса на основе описания класса метаинформации. Окно разработанного нами редак-

тора концептуальных моделей в режиме редактирования класса *GUI Model*, наследующего класс *GUI Model*, представлено на рис. 4.

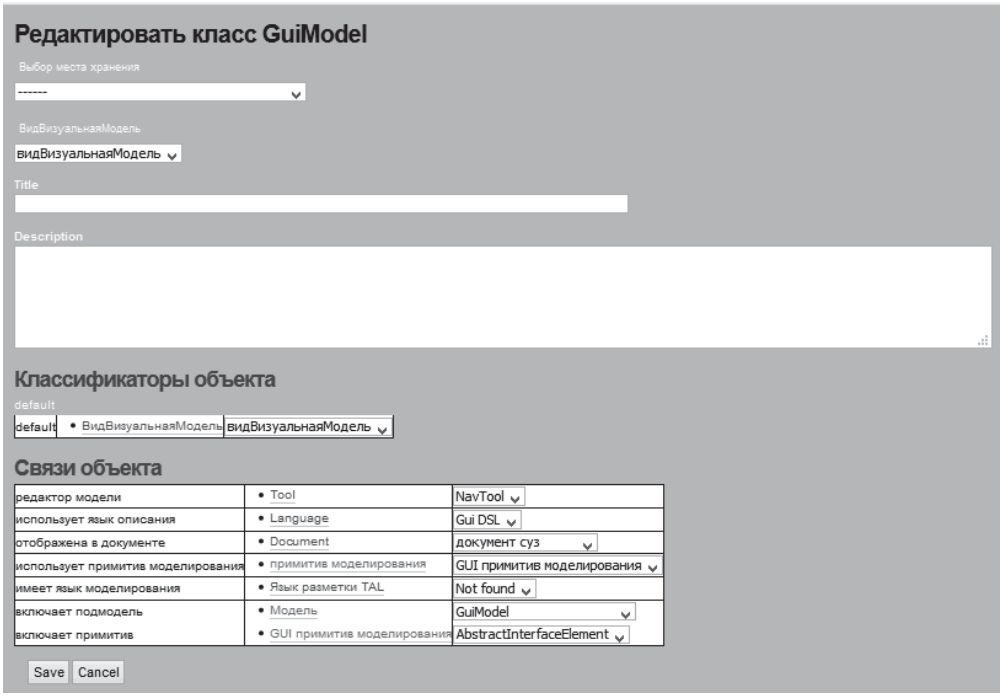

Рисунок 4. Окно редатора многоуровневой модели

При построении модели соответствующий редактор анализирует класс *Model* и создает копии всех принадлежащих ему элементов типа *ClassObjectProperty* и *ClassDataProperty*. Атрибуты *title* прототипа копируются в соответствующий атрибут *title* создаваемого элемента. В атрибут *SubPropertyOf* нового элемента записывается ссылка на прототип, добавляя новый элемент в иерархию связей. Тип элементов для области значений атрибута *range* получается из соответствующего атрибута прототипа, затем строится список наследующих его классов. Пользователю предоставляется возможность определить ограничение на тип связываемого элемента, выбрав его из списка.

6. Заключение

Предложенный способ представления позволяет на разных уровнях моделирования проводить фильтрацию типов элементов, которые могут быть использованы при разработке модели выбранного типа. Созданная нами система многоуровневого семантического моделирования использует разработанные компоненты.

Для выбранного типа модели в редакторе модели на основе информации о связях определяются типы примитивов моделирования и другие компоненты моделирования, например язык описания модели и тип редактора для управления моделью. При построении модели редактор анализирует класс *Model* и создает копии всех принадлежащих ему элементов типа *ClassObjectProperty* и *ClassDataProperty*. Атрибуты *title* прототипа копируются в соответствующий атрибут title создаваемого элемента. В атрибут *SubPropertyOf* нового элемента записывается ссылка на прототип, добавляя новый элемент в иерархию связей. Тип элементов для области значений атрибута *range* получается из соответствующего атрибута прототипа, затем строится список наследующих его классов. Пользователю предоставляется возможность определить ограничение на тип связываемого элемента, выбрав его из списка.

Литература

[1] *Henderson-Sellers B., Gonzalez-Perez C.* The rationale of powertype-based metamodelling to underpin software development methodologies // Proc. 2nd Asia-Pacific conference on Conceptual modeling. Vol. 43. — Australian Computer Society, Inc., 2005. P. 7–16.

[2] *Atkinson C., Kühne T.* The essence of multilevel metamodeling // «UML»2001:Modeling Languages, Concepts and Tools / Eds. Gogolla M., Kobryn C. Vol. 2185. — Springer-Verlag, Berlin, 2001. P. 19–33.

[3] *Atkinson C., Gerbig R., Tunjic C.* Towards multi-level aware model transformations // 5th International Conference on Theory and Practice of Model Transformations. — Springer Berlin Heidelberg, 2012. P. 208–223.

[4] *Скворцов Н. А.* Вопросы согласования неоднородных онтологических моделей и онтологических контекстов // *Онтологическое моделирование*: сб. тр. — М. : ИПИ РАН, 2008 С. 149–166.

[5] *Beck H., Pinto H. S.* Overview of approach, methodologies, standards, and tools for ontologies // The Agricultural Ontology Service. — UN FAO, Rome, 2002.

[6] *Голенков В. В., Гулякина Н. А.* Принципы построения массовой семантической технологии компонентного проектирования интеллектуальных систем // Материалы междунар. науч.-техн. конф. «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS-2011). — Минск, 2011. С. 21–58.

[7] *Грегер С. Э., Сковородин Е. Ю.* Построение онтологического портала с использованием объектной базы // Объектные системы – 2010: Материалы I Международной научно-практической конференции. — Ростов н/Д, 2010. С. 74–78.

[8] *Загорулько Ю. А.* Подход к построению интеллектуальных информационных систем на основе семантических сетей // Международная научно-техническая конференция «Открытые семантические технологии проектирования интеллектуальных систем» (Open

Semantic Technologies for Intelligent Systems) — OSTIS-2011. — Минск : Белорусский государственный университет информатики и радиоэлектроники, 2011.

[9] *Jean S., Ait-Ameur Y., Pierra G.* An object-oriented based algebra for ontologies and their instances // East European Conference on Advances in Databases and Information Systems. – Springer Berlin Heidelberg, 2007. P. 141–156. (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.82.68).

Авторы:

Сергей Эдуардович Грегер — старший преподаватель кафедры информационных технологий; Нижнетагильский технологический институт (фил.), Уральский федеральный университет имени первого Президента России Б. Н. Ельцина, Нижний Тагил

Сергей Владимирович Поршнев — доктор технических наук, профессор, заведующий кафедрой радиоэлектроники информационных систем; Уральский федеральный университет имени первого Президента России Б. Н. Ельцина, Екатеринбург

Method for Transformation of Multi-Level Semantic Models to Object-Oriented System

S. E. Greger, S. V. Porshnev***

** Nizhny Tagil Technologycal Institute*

ul. Krasnogvardejskaja, 59, Nizhny Tagil, Russia, 622013

Ural Federal University,
Екатеринбург. Lenina ave., 51, Ekaterinburg, 620000

e-mail: segreger@gmail.com s.v.porshnev@urfu.ru

Abstract. The article discusses the development of semantics and program implementation of the component to represent elements of the multi-level semantic models in an object-oriented system. The application of the concepts of power type, clabject and multilevel method of object-oriented display with multi-level semantic modeling.
Keywords: multi-level semantic model, object-oriented mapping, object-oriented systems.

References

[1] *Henderson-Sellers B., Gonzalez-Perez C.* (2005) The rationale of powertype-based metamodelling to underpin software development methodologies. In Proc. 2nd Asia-Pacific conference on Conceptual modelling, vol. 43, pp. 7–16

[2] *Atkinson C., Kühne T.* (2001) The essence of multilevel metamodeling. In International Conference on the Unified Modeling Language, pp. 19–33

[3] *Atkinson C., Gerbig R., Tunjic C.* (2012) Towards multi-level aware model transformations. In International Conference on Theory and Practice of Model Transformations, pp. 208–223

[4] *Skvorcov N. A.* (2008) Voprosy soglasovanija neodnorodnyh ontologicheskih modelej i ontologicheskih kontekstov. In book Ontologicheskoe modelirovanie. Moscow, pp. 149–166 [In Rus]

[5] *Beck H., Pinto H. S.* (2002) Overview of approach, methodologies, standards, and tools for ontologies. The Agricultural Ontology Service. UN FAO, Rome

[6] *Golenkov V. V., Gulykina N. A.* (2011) Principy postroenija massovoj semanticheskoj tehnologii komponentnogo proektirovanija intellektual'nyh system. In Conf. Open Semantic Technologies for Intelligent Systems (OSTIS-2011). Minsk, pp. 21–58 [In Rus]

[7] *Greger S. E., Skovorodin E. Y.* (2010) Postroenie ontologicheskogo portala s ispol'zovaniem obektnoj bazy. In Conf. Obektnye sistemy – 2010. Rostov-na-Donu, pp. 74–78. [In Rus]

[8] *Zagorulko Y. A.* (2011) Podhod k postroeniju intellektual'nyh informacionnyh sistem na osnove semanticheskih setej. In Conf. OSTIS-2011. Minsk [In Rus]

[9] *Jean S., Ait-Ameur Y., Pierra G.* (2007) An object-oriented based algebra for ontologies and their instances. In /East European Conference on Advances in Databases and Information Systems, pp. 141–156