МАТЕМАТИЧЕСКИЕ МОДЕЛИ
В ИНФОРМАТИКЕ
(Вычислительные системы)
2002 год Выпуск 169

УДК 519.682+519.685+519.688:519.767.6

ГЕНЕРАЦИЯ СТАТИСТИЧЕСКИХ UML–МОДЕЛЕЙ ПО ТЕКСТАМ ЕСТЕСТВЕННОГО ЯЗЫКА

А.В. Гончарук

Введение

Первый этап разработки любой сложной программной системы — фаза анализа. На данном этапе происходит формализация предметной области, с которой работает система, и функциональных требований к системе.

Работы на этапе анализа осуществляется на основе требований заказчика и неформального (естественно–языкового) описания предметной области.

В настоящей работе описывается подход, позволяющий полуавтоматически создавать UML–диаграммы классов (статическую модель системы) по тексту на естественном языке, а также пополнять готовую модель на основе лингвистической информации об именах классов. Также описывается программная реализация данного подхода. В качестве источника лексических знаний используется электронная лексическая база данных WordNet, генерация UML производится при помощи библиотеки NS UML, интерактивная часть построена на базе системы моделирования ArgoUML, и генератор, и интерактивная часть реализации построены на технологии Java.

Изложенный в настоящей статье подход позволяет частично автоматизировать деятельность аналитика путем генерации UML–спецификации предметной области по естественно–языковой спецификации. Сгенерированная модель содержит классы и отношения агрегации (часть–целое) и наследования (общее–частное) между ними.

В первую очередь, следует отметить, что на настоящий момент не существует промышленных вариантов систем, которые порождают UML–модели по текстам естественного языка (программным спецификациям).

Система Color-X. Color–X является аббревиатурой для COnceptual Linguistically based Object oriented Representation Language for Information and Communication Systems (Ics сокращается как X). Система описывается в работе [3]. В серии работ, посвященных Color-X, исследуется, как лингвистические знания могут быть использованы в построении информационных систем и систем коммуникаций. В Color-X используется CPL (Conceptual Prototyping Language), который был предложен Дигнумом (Dignum, [4]). В качестве языка графического представления используется подмножество языка OMT (Object Modeling Technique), который был предложен Рамбо (Rumbaughs, 1991).

В качестве источника лингвистических знаний используется WordNet. Лексикон используется в процессе моделирования в двух случаях:

1) для проверки корректности модели, после того, как разработка модели завершена;

2) для интерактивного использования лексикона в процессе моделирования системы.

В первом случае по данной модели генерируются предложения на естественном языке, которые затем анализируются дизайнером и сравниваются с исходной спецификацией. Во втором случае система автоматически предлагает аналитику необходимые отношения и классы. Исходная статическая модель генерируется автоматически. Разработчики Color–X делали упор на первом пункте использования.

Реализация Color–X была не полной и производилась на уровне прототипирования. Также основной целью Color-X было построение лингвистически корректных моделей и генерация исполняемого кода или текста на естественном языке по модели. Разработчики Color-X сосредотачивали внимание на разработке языка, который бы позволял согласовывать модели и естественно–языковые представления.

Система LOLITA. В работе [17] дается описание системы LOLITA, которая специализировалась на создании больших объектно–ориентированных систем с использованием лингвистических знаний. В построении системы использовался подход концептуальных графов (Concept Graphs), анализ текста производился на морфологическом и синтаксическом уровне. Семантический уровень был представлен слабо. Генерация модели происходила в подмножество языка OMT. По полученной модели производилась генерация естественного языка для проверки корректности модели.

В работах [15, 16] описываются два подхода к созданию так называемых Entity–Relationship диаграмм (ER–диаграмм), которые обычно используются для представления структур баз данных.

В [15] рассматривается генерация расширенной ER (Extended ER или EER)–модели по текстам естественного языка. Основу подхода составляет применение эвристик и правил, которые преобразуют структуры, полученные после синтаксической фазы в обработке текста. По полученным в результате преобразований структурам происходит построение ER–модели.

В работе [16] рассматривается подход к генерации структур баз данных по текстам естественного языка и синтезу текста по данной структуре базы данных для проверки корректности построенной модели. В работе использовались концептуальные графы и так называемая case–грамматика, которая была предложена Филлмором (Fillmore).

2. Язык UML

История UML. В последнее время язык UML (Unified Modeling Language) стал индустриальным стандартом в области проектирования программных систем и моделирования бизнеса предприятий. UML является графическим языком для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению. При помощи UML можно разработать детальный план создаваемой системы, отражающий не только ее концептуальные элементы, такие как системные функции и бизнес–процессы, но и конкретные особенности реализации, в том числе

классы, написанные на специальных языках программирования, системы баз данных и программные компоненты многократного использования.

Объектно–ориентированные языки моделирования появились в период с середины 70–х до конца 80–х годов, когда исследователи были вынуждены начать разработку альтернативных подходов к анализу и проектированию. В период с 1989 по 1994 число различных объектно–ориентированных методов возросло с десяти до более чем пятидесяти. Пользователи затруднялись сделать выбор в пользу одного из методов и языков моделирования. В результате в 1994 году началась работа над UML, в котором создатели основных на тот момент методов решили объединить преимущества различных подходов. Основой для UML явились языки Booch, созданный Грейди Бучем, OOSE (Object–Oriented Software Engineering), созданный Айваром Джекобсоном, и OMT (Object Modeling Technique), автором которого является Джеймс Рамбо. В 1997 году консорциум OMG (Object Management Group) утвердил стандарт UML 1.0. Текущая версия UML — UML 1.3 [1]; UML 1.4 находится в разработке, доступна предварительная версия спецификации языка.

OMG разработал XML (eXtensible Markup Language) стандарт на представление UML моделей — XMI (XML Metadata Interchange), [2] который будет использоваться в дальнейшей работе.

Основные понятия UML. Типичные объекты, входящие в статическую UML–модель — это классы и отношения между ними. Классы — это самые важные строительные блоки в любой объектно–ориентированной системе. Они представляют собой описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой. Класс реализует один или несколько интерфейсов.

Классы различным образом взаимодействуют между собой. Существует три основных вида отношений:

• **зависимости**, которые описывают существующие между классами отношения использования;

---

[1] Смотри http://www.omg.org/technology/uml/index.htm

[2] Смотри http://www.omg.org/technology/xml/index.htm

• **обобщения или генерализации**, связывающие обобщенные классы со специализированными;

• **ассоциации**, представляющие структурные отношения между объектами (частым случаем ассоциации является агрегация).

Каждый из этих типов отношений позволяет по–разному комбинировать абстракции. Самой важной частью статической (или структурной) UML–модели являются диаграммы классов. На рис.1 приведен пример UML–диаграммы классов.

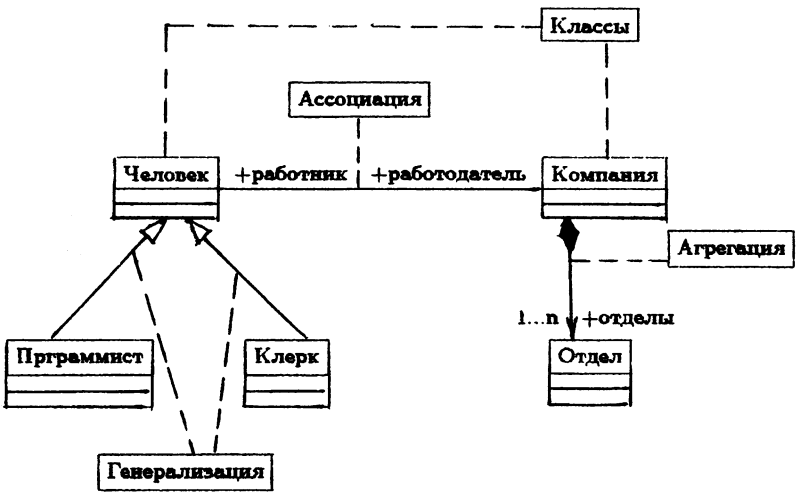

Рис. 1

Как видно из приведенной на рис. 1 диаграммы, она описывает следующие отношения и классы: **Человек** работает на **Компанию**, **Компания** содержит один или несколько **Отделов**. Существуют такие виды работников, как **Программист** и **Клерк**.

Помимо диаграмм классов в статическую UML–модель входят диаграммы объектов, диаграммы компонентов, диаграммы

развертывания. В динамическую модель входят диаграммы состояний и деятельности, use–case диаграммы (диаграммы вариантов использования системы, прецедентов), диаграммы последовательностей и кооперации.

Средства UML моделирования.

Rational Rose наиболее распространенный UML–редактор от основного создателя UML фирмы Rational. Основной недостаток — неполная поддержка UML 1.3, а также отсутствие экспорта модели в стандартное представление (XMI)[3].

Object Domain — UML–редактор наиболее полно поддерживающий UML 1.3[4].

ArgoUML — свободно распространяемый UML–редактор с открытым кодом (open source). Основывается на библиотеке NSUML. Недостатки — продукт недоработан, нестабилен, отсутствуют возможности расширения функциональности при помощи plugin–интерфейсов. Проект ArgoUML зародился в Калифорнийском университете. Создатели ArgoUML [5] ставили перед собой следующие задачи:

- построение когнитивного средства моделирования, (в настоящей версии ArgoUML содержит механизм подсказок, который облегчает работу дизайнеру);
- поддержка UML 1.3;
- поддержка XMI.

Poseidon for UML[6] — UML–редактор, являющийся переработанной версией ArgoUML. Поддерживаются plugins; Poseidon for UML более стабилен по сравнению с ArgoUML.

Для доработки был выбран ArgoUML, так как он единственный среди вышеперечисленных редакторов, распространяется с открытым кодом и базируется на библиотеке NS UML, которая также распространяется свободно.

---

[3]Смотри http://www.rational.com/products/rose/

[4]Смотри http://www.objectdomain.com

[5]Смотри http://argouml.tigris.org

[6]Смотри http://www.gentleware.com/products/index.php3

Библиотека NS UML[7].

Библиотека NS UML была разработана в компании Novosoft Inc. Она предоставляет следующие сервисы: полная реализация физической метамодели UML 1.3, простые интерфейсы, событийные механизмы, поддержка undo/redo, рефлексивный API (Application Programmer Interface), сохранение и загрузка XMI.

OMG использует четырехуровневую архитектуру метаданных (см. рис. 2).

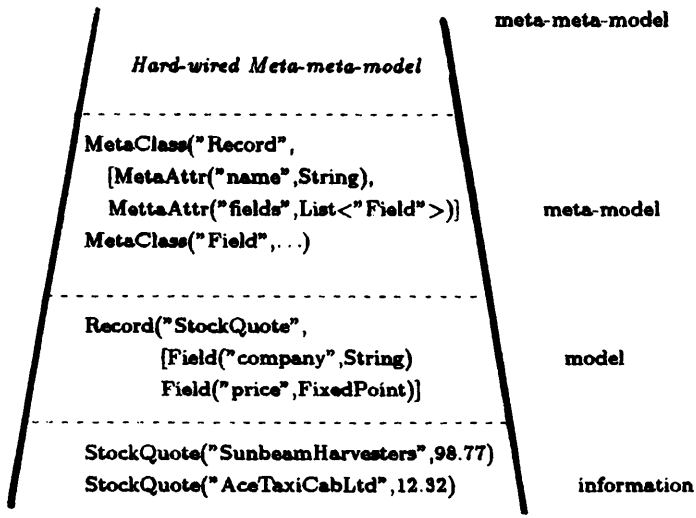

Рис. 2

Уровень информации — это уровень конкретных объектов, уровень модели соответствует уровню классов, метамодели — уровню метаклассов (экземпляром метакласса является класс), на уровне метаметамодели содержатся метаметаклассы. Библиотека NS UML оперирует на уровне метамодели.

Преимущества от использования библиотеки NS UML:

1) библиотека распространяется с открытым кодом;

2) библиотека разработана с применением технологии Java.

[7]Исходный код библиотеки можно найти по адресу http://nsuml.sourceforge.net

4. Электронная лексическая база WordNet

Электронная лексическая база WordNet[6] была разработана в лаборатории когнитивных наук Принстонского университета. WordNet — это система лексических ссылок, которая базируется на современных психолингвистических теориях. Основу базы данных составляют множества синонимов (synonym set, synset, в дальнейшем синсет), каждое из которых определяет словарный смысл. Существительные, прилагательные, глаголы объединяются в синсеты. Синсеты связываются различными отношениями.

Основная идея, которой руководствовались создатели WordNet — организация концептуального, но не алфавитного, поиска по лексической базе. В настоящее время WordNet содержит 95600 различных словарных форм (51500 слов и 44100 словосочетаний), которые организованы в 70100 словарных значений. Главным отличием WordNet от остальных словарей является то, что

- WordNet разделяет слова на пять категорий: существительные, прилагательные, глаголы, наречия, функциональные слова;
- WordNet упорядочивает лексическую информацию по смыслам, а не по словарным формам.

WordNet базируется на двух словарях — словаре синонимов Родаля (Rodale's The Synonym Finder, 1978) и толковом словаре Роже (Roget's International Thesaurus, 1977). В настоящее время базы данных WordNet разработаны для английского, немецкого, голландского, итальянского языков. К сожалению, версия WordNet для русского языка пока не разработана.

Основополагающим отношением в WordNet является лексическая матрица (табл.1; $\{F_1, ..., F_n\}$ — множество словоформ, $\{M_1, ..., M_m\}$ — множество смыслов).

В данной матрице F_1 и F_2 — синонимы; F_2 обладает свойством полисемии. Отношение между словарными формами и смыслами является отношением много–много. Так, некоторые словарные формы имеют несколько смыслов и некоторые

[6] Смотри http://cogsci.princeton.edu/ wn

Таблица 1

Лексическая матрица WordNet

Значения слов	Словарные формы				
	F_1	F_2	F_3	...	F_n
M_1	$E_{1,1}$	$E_{1,2}$			
M_2		$E_{2,2}$			
M_3			$E_{3,3}$		
...				...	
M_m					$E_{m,n}$

смыслы могут быть выражены несколькими словарными формами. Лексическая матрица может также представлять отношение между словами и синсетами.

Синсеты в WordNet связаны семантическими и лексическими отношениями. Если существует отношение R между значением $\{x, x', \ldots\}$ и $\{y, y', \ldots\}$, то существует отношение R' между $\{y, y', \ldots\}$ и $\{x, x', \ldots\}$. В дальнейшем, если существует отношение R между синсетами $\{x, x', \ldots\}$ и $\{y, y', \ldots\}$, то существует отношение R между конкретными словарными формами x и y. В WordNet присутствуют следующие отношения.

Синонимия. Как уже было сказано, синонимия — основное отношение в WordNet. По одному из определений (которое обычно приписывается Лейбницу) два выражения являются синонимами, если при замене одного выражения на другое в предложении, истинность предложения не меняется. По этому определению, "настоящих" синонимов достаточно мало. Ослабленное определение синонимии ставит синоним в зависимость от контекста: два выражения являются синонимами в лингвистическом контексте C, если замена выражения на его синоним не меняет истинности в контексте. Из этого определения следует, что слова, которые относятся к различным частям речи не могут быть синонимами. Последнее определение является неудовлетворительным, если речь идет о синонимии прилагательных (дискретного разделения на истину и ложь в данном случае может не хватить, например, в случае, когда прилагательные соответствуют какому-то свойству различной силы — *хороший — отличный — великолепный*).

Антонимия. Другим известным отношением является антонимия, которое на удивление трудно определить. Антонимом слова x является что–то не x, но это применимо не во всех случаях. Например, антоним к слову *богатый* — *бедный*, но не всегда кто не *богат* является *бедным*. Этот феномен связан, в первую очередь, со свойствами отрицания в естественном языке.

Антонимия является лексическим, но не семантическим отношением. Связываются словарные формы, а не множества синонимов. Например, значения {вставать, подниматься} и {падать, опускаться} являются противоположными, но антонимами являются только подниматься/опускаться и вставать/падать. Следует отметить, что антонимия широко используется в подразделах наречий и прилагательных WordNet.

Гипонимия (hyponymy). В отличие от антонимии и синонимии, гипонимия и гипернимия являются семантическими отношениями между значениями слов, например, {клен} гипоним {дерево}, а {дерево} гипоним {растение}. Гипернимия – это отношение, обратное гипонимии. Смысл, представляемый множеством синонимов $\{x, x', \ldots\}$ является гипонимом для синсета $\{y, y', \ldots\}$, если носитель языка может сказать, что x является подвидом y.

Гипонимия — это транзитивное и антисимметричное отношение. И так как в большинстве случаев для каждого смысла имеется только один родитель, отношение порождает иерархичную структуру. Гипоним наследует все свойства родительского смысла и добавляет нечто новое. В этом смысле гипонимия является прямым аналогом генерализации (отношения общее–частное) в UML. Это отношение будет использоваться для построения UML–модели. Следует отметить, что WordNet содержит верхний уровень абстракции для каждой из частей речи. Например, для существительных в английской версии WordNet самыми абстрактными являются смыслы, приведенные в табл.2.

Данное отношение является важным особенно для представления существительных.

Меронимия (Meronymy). Синонимия, антонимия и гипонимия широко используются людьми в речи. Другим важным

Таблица 2

Абстрактные смыслы для существительных в WordNet

{act, action, activity}	{natural object}	{event, happening}
{animal, fauna}	{natural phenomenon}	{feeling, emotion}
{artifact}	{person, human being}	{food}
{attribute, property}	{plant, flora}	{group, collection}
{body, corpus}	{possession}	{location, place}
{cognition, knowledge}	{process}	{motive}
{communication}	{quantity, ammount}	{relation}
{shape}	{state, condition}	{substance}
{time}		

семантическим отношением является меронимия или отношение "часть–целое". Смысл, представленный синсетом $\{x, x', \ldots\}$ является меронимом смысла $\{y, y', \ldots\}$, если носитель языка может сказать, что *x является частью y*. Отношение меронимии может быть использовано, для построения другого вида иерархии — иерархии частей, например, *у собаки есть лапы и хвост, лапы состоят из костей и мышц, мышцы из клеток* и т.д.

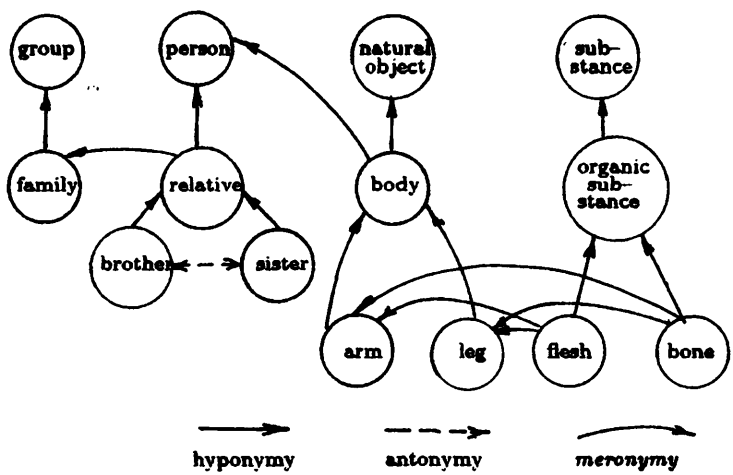

Рис. 3

Морфологические отношения. Важный класс отношений – это морфологический отношения между словами. В начале разработки WordNet не планировалось добавлять морфологические отношения в систему, но затем эти отношения были добавлены, так как возможна такая ситуация: пользователь ищет в базе данных слово *деревья*, было бы удобно, если бы база данных распознавала, что *деревья* — это множественное число от *дерево*.

Общие замечания. Следует отметить, что перечисленные выше отношения в полной мере относятся к представлению существительных, представления глаголов, прилагательных, наречий не рассматриваются, так как они не используются в построении генератора.

Пример отношений для существительных в WordNet приведен на рис. 3.

4. Создание UML–модели по текстам естественного языка

Общие принципы. Процесс генерации состоит из двух этапов:

1) полуавтоматический этап формирования входных данных для алгоритма генерации;

2) построение необходимых отношений по входным данным и генерация UML–модели.

Входными данными для алгоритма являются пары (слово, смысл в WordNet), которые формирует пользователь системы на первом этапе. Анализируются только существительные, которые входят в исходный текст на естественном языке. Это ограничение делается в связи с постулатом, широко используемом в объектно–ориентированном анализе и проектировании — существительные, который встречаются в спецификации программной системы — это претенденты на классы [5,6].

В процессе создания модели WordNet используется в качестве источника лингвистических знаний. Используются следующие встроенные отношения WordNet:

- меронимия,
- гипернимия.

Использование WordNet позволяет значительно упростить алгоритм построения модели, *так как по семантике отношения меронимии и гипернимии полностью соответствуют отношениям генерализации и агрегации в* UML. Таким образом, приведенный далее алгоритм является достаточно прямым. Также, WordNet используется для выделения существительных из входного текста.

Формирование входных данных для алгоритма генерации. На первом этапе из входного текста выделяется множество существительных (используется WordNet) и для каждого существительного выделяется множество смыслов WordNet. Далее пользователь выбирает слова и соответствующие смыслы, которые будут затем переданы на вход алгоритма генерации.

Алгоритм генерации. Как уже было отмечено выше, входными данными для алгоритма генерации является набор пар (*слово, номер смысла WordNet*). По данному набору необходимо построить иерархию наследования классов и для каждого из полученных классов построить классы–агрегаты.

В результате работы алгоритма строится дерево наследования, в котором вершинами являются пары (*слово, смысл WordNet*). Пара B является потомком пары A, если смысл пары B является наследником смысла пары A. Каждая из пар соответствует некоторому UML–классу, именем которого является слово, входящее в соответствующую пару. Словарный компонент пары может быть пустым. В этом случае именем класса будет являться первый из синонимов, входящий в смысловую компоненту пары.

Шаг 1. Создается выделенная вершина для корня дерева.

Шаг 2.1 Для каждой пары строилась последовательность родительских пар, т.е., для каждой пары A строится последовательность пар $A_1, \ldots, A_n$, где A_n — один из корневых смыслов существительных в WordNet, и смысл пары A_{i+1} является родительским смыслом для смысла пары A_i (смыслы A_i и A_{i+1} находятся в отношении гипернимии WordNet), $A_1 = A$. Эта последовательность строится следующим образом: для каждой входной пары A строится родительский смысл A_1, для пары A_1 строится

родительский смысл A_2, и т.д. до тех пор, пока не будет достигнут один из корневых смыслов существительных WordNet.

Шаг 2.2 Каждая полученная последовательность пар (*слово, смысл*) добавляется в результирующее дерево. Вершина A_n — потомок корня дерева, A_{n-1} — потомок A_n, и т.д. Следует заметить, что если на каком–то из этапов добавления вершины в дерево добавляемая вершина уже существует в дереве, то производится переход на следующий шаг иерархии.

В построенном дереве связи между вершинами соответствуют UML–отношению генерализации.

Шаг 3. Для каждой из листовых вершин дерева и вершин, у которых значение первого элемента пары (слово) !=∅, добавляется список агрегатов (выбираются смыслы WordNet, которые находятся с данным смыслом в отношении меронимии). Для этого вводится понятие агрегатной дуги в дереве, которая связывает смысл с его агрегатами. Таким образом, производится добавление новых листьев в дерево.

Шаг 4. По полученному дереву производится обход и создаются необходимые UML–классы (используется NS UML API). Для каждого узла дерева создается соответствующий класс (имя класса полагается равным полю "слово" в вершинной паре или первому из синонимов в соответствующем смысле если поле "слово" пусто) и необходимые отношения. Создается отношение генерализации между классами, если существует дуга обобщения между двумя соответствующими смыслами в дереве. Отношение агрегации создается в том случае, если вершины дерева связаны агрегатной дугой.

Шаг 5. Производится сохранение UML–модели средствами NS UML.

Полученный алгоритм является прямым, так как используемые отношения между смыслами WordNet эквивалентны отношениям генерализации и агрегации UML.

Примеры.

Заметим, что все примеры, которые будут приводиться далее, будут на английском языке, так как версии WordNet для русского языка в настоящий момент не существует. После запуска процедуры генерации на вход к генератору поступают пары *(слово, смысл)*, по которым и производится генерация. Для каждого смысла выделялась последовательность смыслов, которые являются родителями данного смысла (использовалась гипернимия WordNet). Например, для слова dog (собака) в смысле "собака как животное" получается следующая последовательность родительских смыслов

(dog, canine, carnivore, mammal, ...)

или

(собака, животное из семейства псовых, плотоядное животное, млекопитающее, ...).

После построения таких цепей для каждого из входящих в спецификацию существительных, все цепи сливались в единое дерево наследования. Например, если существует еще одна цепь для cat (кошка), в смысле "кошка как животное"

(cat, feline, carnivore,...)

или

(кошка, животное из семейства кошачьих, плотоядное животное,...),

то одинаковые смыслы объединялись в одну вершину дерева. В данном примере, ветвление начинается на вершине (carnivore - плотоядное животное).

Далее, для каждого листового смысла выделялись его агрегаты (использовалась меронимия WordNet). Например, для слова car (машина) в смысле "автомобиль", выделялись следующие части:

(accelerator, air bag, auto accessory, automobile engine, automobile horn,ldots)

или

(акселератор, подушка безопасности, авто аксессуары, двигатель автомобиля,...)

Следует отметить, что в описанных процедурах неявно использовалась синонимия. Например, в случае присутствия в спецификации слов *felid* и *feline* (*два синонима для "животное из семейства кошачьих"*), им бы соответствовал один и тот же смысл (набор синонимов), и дерево было бы построено корректно (*feline* и *felid* представляли бы одну вершину в дереве).

Затем производилось добавление агрегатов для каждого листового смысла и смысла с выделенным именем. Например, для слова собака (*dog*) выделялись следующие агрегаты: *лапа* (*paw*), *хвост*(*tail*), *шерсть* (*hair*), и т.д.

По данному дереву при помощи библиотеки NS UML создавалась модель и сохранялась в XMI-формате, который затем может быть открыт в ArgoUML. Далее приводится пример текста и диаграммы, которые были получены в результате генерации модели по тексту:

The presented text should have the following structure:

Text should be divided into chapters; chapters should be divided into paragraphs,

And each paragraph consists of a number of words.

По нему были получены диаграмма наследования (рис. 4) и диаграмма агрегации (рис. 5).

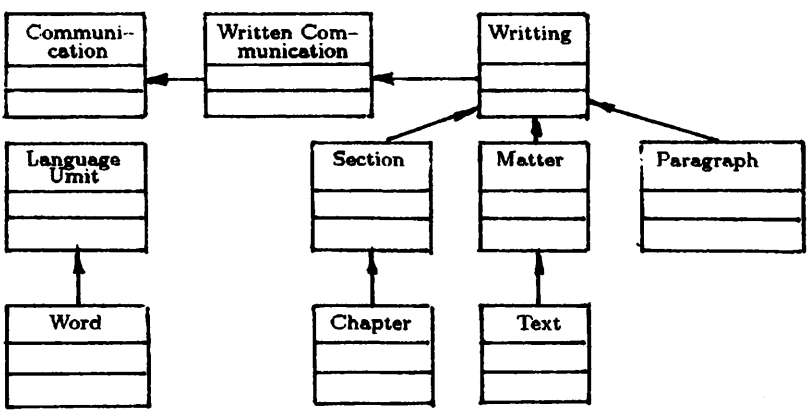

Рис. 4

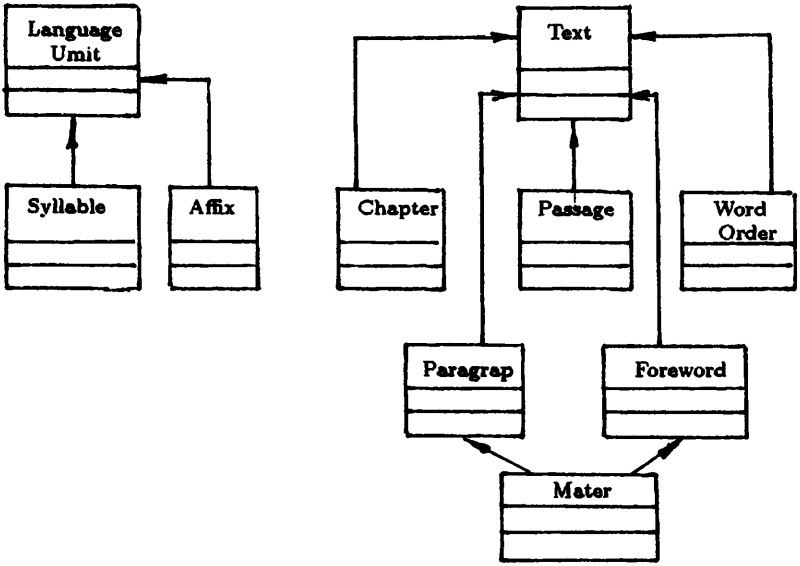

Рис. 5

Следует отметить, что с диаграмм были удалены лишние связи (генерализации и агрегации). В целом, объем удаленных отношений не превышает 30%. Кроме того, формат XMI не предоставляет возможности представлять UML–диаграммы, возможно лишь сохранение данных (классов и связей между ними). Диаграмма же — это лишь графическое представление данных. Таким образом, полуавтоматизм подхода проявляется на двух этапах создания модели:

- выбор необходимых для генерации существительных;
- помещение конкретных классов на диаграммы и удаление ненужных отношений.

Интерактивное пополнение модели. После завершения генерации и удаления ненужных отношений и сущностей, аналитик может дорабатывать модель, добавлять необходимые классы и связи. На этом этапе аналитику может быть полезна такая функциональность средства моделирования, как полуавтомати-

ческое добавление новых классов и отношений между ними, которое будет использовать лингвистическую информацию.

В рамках настоящей работы были сделаны дополнения в системы ArgoUML, которые позволяют пользователю в интерактивном режиме добавлять классы и отношения, руководствуясь данными о конкретном слове (имени класса), которое содержится в WordNet. Данная функциональность может быть полезна при проектировании сложных систем, в которые входят множества классов и важно учесть все возможные отношения между ними. Например, если в модели существует класс *Mammal (Млекопитающее)*, то используя данные, содержащиеся в WordNet возможно автоматическое добавление всех суперклассов или подклассов для *Mammal*. Или по классу *Car (Машина, Автомобиль)*, автоматически добавить все части автомобиля в модель. Разработка необходимых дополнений была также выполнена на языке Java, использовались библиотеки JWordNet и NS UML.

Например, следующие классы-агрегаты могут быть получены при помощи ArgoUML с плагином естественно–языковой генерации (см. рис. 6)

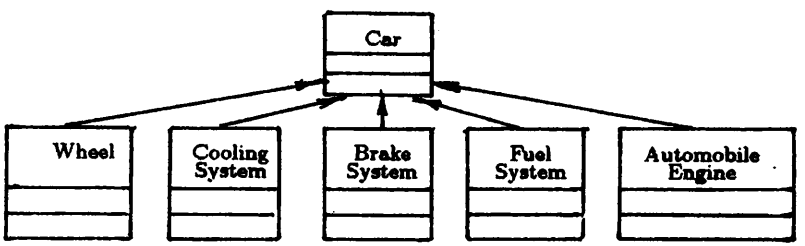

Рис. 6

Заключение

В результате работы были выявлены следующие проблемы, связанные со структурой WordNet:

• WordNet не содержит информации, специфичной конкретных узких предметных областей;

• нет явного разделения контекстов;

• некоторая информация является избыточной, например, слову *машина* (*car*) соответствуют около ста различных прямых и наследованных агрегатов.

В связи с этим, необходима разработка структуры лексической базы данных, которая бы соответствовала классу решаемых задач, а именно, классу задач генерации UML–моделей по текстам естественного языка.

В дальнейшей работе планируется:

• совершенствовать существующий подход к генерации статической модели, например, порождать операции и атрибуты классов, порождать отношения ассоциации между классами;

• разработать структуру лексической базы данных, которая бы соответствовала классу решаемых задач;

• генерация динамической модели. В первую очередь, интересна задача порождения use–case спецификаций по текстам естественного языка, так как разработка use–case спецификации системы (спецификация функциональных требований к системе) является одним из наиболее важных этапов в разработке программной системы. На этом этапе может быть использована теоретическая база, которая была разработана И.А. Мельчуком и Ю.Д. Апресяном и разработчиками системы автоматического перевода ЭТАП–3, в частности, модель "смысл ↔ текст";

• разработать plugin для Object Domain.

Все перечисленные подзадачи можно объединить в одну общую задачу — построение системы "рабочего места аналитика программного проекта".

Литература

1. MILLER G.A., BECKWITH R., FELLBAUM Ch., GROSS D., MILLER K. — Introduction to WordNet: An On–line Lexical Database. — 1993.
ftp://ftp.cogsci.princeton.edu/pub/wordnet/5papers.pdf

2. MILLER G.A. Nouns in WordNet: a lexical inheritance system //International Journal of Lexicography. —1990. — Vol. 3, № 4. — P. 245–264.

3. BURG J.F.M., Van der RIET V.R. Color–X: Object Modeling profits from Linguistics //Proceedings of the KB&KS'95, the

Second International Conference on Building and Sharing of Very Large-Scale Knowledge Bases, Enschede, The Netherlands. — 1995. ftp://ftp.cs.vu.nl:/pub/lics/kbks95.ps.gz

4. DIGNUMF P.M. A Language for Modeling Knowledge Bases. Based on Linguistics, Founded in Logics: PhD thesis. — Vrije Universiteit. — Amsterdam. — 1989.

5. БУЧ Д., РАМБО Д., ДЖЕКОБСОН А. Язык UML. Руководство пользователя. — М., 2000.

6. БУЧ Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд. — М., 1999.

7. MILLER G.A. WordNet: a lexical database for English //Communications of the ACM. — 1995. — Vol. 38, № 11, November. — P. 39–41.

8. TJOA A. Min, BERGER L. Transformation of Requirement Specifications Expressed in Natural Language into ERR model //Lecture Notes in Computer Science. — 1993. — Vol. 823. Entity-Relationship Approach ER'93.

9. METAIS E., MEUNIER J.-N., LEVREAU G. Database Schema design: A Perspective from NL technique to Validation and View Generation //Lecture Notes in Computer Science. — 1993. — Vol. 823. Entity-Relationship Approach ER'93.

10. MICH L. A Linguistic Approach to the Development of OO Systems using NL System LOLITA //Lecture Notes in Computer Science. — 1994. — Vol. 858. Object-Oriented Methodologies and Systems.

Поступила в редакцию
23 мая 2002 года