



На правах рукописи

ЛЕБЕДЕВ Андрей Сергеевич

**РАЗРАБОТКА И ИССЛЕДОВАНИЕ СИСТЕМЫ
КОНЦЕПТУАЛЬНОГО ПРОГРАММИРОВАНИЯ С
ИСПОЛЬЗОВАНИЕМ ЛИНГВИСТИЧЕСКОГО
ПРОЦЕССОРА**

05.13.11 — Математическое и программное обеспечение вычислительных
машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Москва — 2011

Работа выполнена в Московском государственном институте электроники и математики (техническом университете).

Научный руководитель

Путилов Георгий Петрович

доктор технических наук,
профессор

Официальные оппоненты

Никольский Сергей Николаевич

доктор технических наук,
доцент

Кантор Илья Александрович

кандидат технических наук

Ведущая организация

**Институт проблем информатики
Российской академии наук (ИПИ РАН)**

Защита состоится 29 марта 2011 года в 16:00 на заседании диссертационного совета Д 212.133.01 Московского государственного института электроники и математики (технического университета) по адресу: 109028, г. Москва, Большой Трехсвятительский пер., д. 3.

С диссертацией можно ознакомиться в библиотеке Московского государственного института электроники и математики.

Автореферат разослан «28» февраля 2011 года.

Ученый секретарь диссертационного совета

кандидат технических наук,

доцент



С.Е. Бузников

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность проблемы.

Разработка современного программного обеспечения (ПО) носит, как правило, циклический характер, обусловленный потребностью заказчика в постоянном улучшении программного продукта, добавлении новой функциональности, а также изменением внутренней логики бизнес-процессов. Практика разработки коммерческого ПО показывает, что характер поддержки и доработки такого ПО после внедрения не затрагивает архитектуры системы и носит преимущественно проблемно-ориентированный характер. Современные тенденции решения данной задачи заключаются в разработке так называемых *малых языков* или *языков предметной области* (domain specific language - DSL). Между тем, необходимо отметить сходство концепции "малых языков" с идеями *концептуального программирования* - подходом, изложенным Э.Х. Тыгуу еще в 80-х годах XX-го века. Данный подход в силу ряда причин не получил к настоящему моменту развития и широкого применения. Этому, по всей видимости, способствовал уровень развития вычислительной техники и компьютерной лингвистики того исторического периода. Однако в настоящее время идеи концептуального программирования вновь набирают все большую популярность, в том числе и за рубежом (например, язык программирования XL). К этим идеям относятся следующие аспекты:

- программирование в терминах предметной области решаемых задач,
- использование ЭВМ на этапе постановки задач.

Близость подхода, применяемого в DSL и концептуальном программировании, заключается также в тенденции приблизить язык программирования (или, говоря в терминах концептуального программирования, *язык описания задачи*) к *естественному языку*. Это позволяет, в конечном счете, не только максимально облегчить труд программиста, но и привлечь к такому "программированию" бизнес-аналитиков, незнакомых с обычными языками программирования и средствами разработки.

Таким образом, актуальность данной работы обусловлена двумя факторами. Во-первых, использование естественного языка в качестве языка программирования предметной области позволяет существенно упростить поддержку уже внедренного ПО и повысить ее эффективность. Во-вторых, ввиду сложности разработки предметно-ориентированных

языков существует необходимость создания и исследования системы, позволяющей упростить процесс конструирования, поддержки и встраивания таких *естественных языков предметной области* в существующие программные продукты.

Цели и задачи настоящего исследования. Целью работы является разработка системы концептуального программирования с использованием лингвистического процессора, назначение которой - существенно упростить задачу создания и встраивания поддержки естественных предметно-ориентированных языков в программное обеспечение.

Для достижения поставленной цели в ходе настоящего исследования было найдено решение следующих задач:

- анализ моделей автоматического понимания текстов (АПТ), языков и средств описания онтологий;
- выбор и синтез подходов к автоматической обработке текстов (АОТ), выбор модели формализованного представления "смысла" текста;
- разработка концепции естественно-языкового спецификатора объектно-ориентированного языка программирования.
- разработка программного комплекса, включающего в себя лингвистический процессор и средства его настройки на предметную область; компоненты построения концептуального графа, кодогенерации, а также инструментарий естественно-языковой спецификации классов объектно-ориентированного языка программирования.

Методы исследования основаны на использовании технологии объектно-ориентированного программирования, технологии построения интерпретаторов языков программирования, принципов модели концептуальной зависимости и модели "мягкого понимания текстов", аппарата теории грамматик, теории графов и теории множеств.

Научная новизна

1. Предложен способ описания онтологии с помощью объектно-ориентированного языка программирования; разработаны и

реализованы соответствующие программные средства естественно-языкового спецификатора, что является дальнейшим развитием концептоцентрического подхода.

2. Предложен подход к представлению "смысла" текста в виде динамической объектной системы, состоящей из экземпляров классов языка Java.
3. Показана возможность обратного перехода от предложенной динамической объектной системы к концептуальному графу путем распознавания в объектной системе новых отношений и концептов, и переход от концептуального графа к естественно-языковому тексту.
4. Разработан визуальный редактор расширенных сетей переходов, имеющий встроенный морфологический анализатор, что позволяет производить поиск по имеющимся сетям и строить новые на основе вводимой фразы-шаблона.
5. В результате проведенной работы создана система концептуального программирования Nalaps, имеющая возможность модульной интеграции со средой разработки программ на языке Java - IntelliJ IDEA, что позволяет использовать все современные средства интенционального программирования. Показана практическая возможность разработки естественных языков предметной области в системе Nalaps.
6. В качестве примера использования системы рассмотрен аспект автоматической генерации программных тестов для верификации исправленных ошибок в программном продукте на основе предоставляемых записей из систем отслеживания ошибок, а также система поддержки естественно-языковой настройки бизнес-правил в банковском ПО.

Основные положения, выносимые на защиту:

1. Подход к представлению онтологической базы знаний, построенный на использовании классов объектно-ориентированного языка программирования и карт настройки классов на предметную область, описываемых на языке XML. Предложенный подход позволяет эффективным образом сочетать описания понятий предметной области с правилами и алгоритмами их обработки, выполняющимися при автоматической обработке текста.
2. Способ представления "смысла" исходного естественного-языкового запроса (текста) в виде динамической объектной системы, а также возможность обратного перехода от данного представления к концептуальному графу. Предложенный подход позволяет в реальном времени отслеживать состояние объектов автоматизации и строить естественно-языковые диалоги с системой.
3. Техническая реализация предложенных подходов и методов в программном комплексе Nalaps - системе концептуального программирования, которая предназначена для разработки базы знаний предметной области, программирования лингвистического процессора и непосредственной автоматической обработки текста, что позволяет встраивать в существующие программные продукты поддержку естественных языков предметной области.

Теоретическая ценность работы заключается в создании методических основ для построения базы знаний предметной области на основе классов объектно-ориентированных языков программирования. Описана математическая модель предложенного подхода.

Практическая ценность

1. Разработанная типовая структура программного обеспечения для организации связей классов языка Java с предметной областью позволяет при необходимости создавать по предложенной методике аналогичные системы для других объектно-ориентированных языков программирования (C++, C#.net и др.).
2. Создан ряд учебно-игровых и практически применимых примеров использования системы Nalaps. Среди них:

- учебно-игровая среда управления трехмерными объектами путем подачи естественно-языковых текстовых команд;
 - среда построения генеалогической базы данных и родословного древа путем текстового описания родственных связей между персонами;
 - система автоматической генерации программных тестов на основании входящих естественно-языковых описаний ошибок и доработок в программном обеспечении.
3. Для эффективного использования системы программистами она может быть встроена в качестве программного модуля в среду разработки IntelliJ IDEA, что позволяет полностью интегрировать процесс создания базы знаний предметной области в процесс написания программных средств для обработки этих знаний.
4. В качестве важных областей применения системы следует выделить следующие:
- создание естественных языков предметной области для внутренних нужд разработки;
 - создание естественно-языковых интерфейсов общения пользователя с программными средствами;
 - автоматическая обработка потоков текстовой информации на предмет регистрации определенных информационных сигналов (событий) и построение автоматической реакции на них;
 - учебно-игровой аспект: обучение объектно-ориентированному программированию.
5. Метод автоматической обработки текстов, разработанный в ходе диссертационных исследований, был использован также и в других научно-исследовательских работах, в частности, в системе семантического поиска.
6. Создан инструмент программирования расширенных сетей переходов - визуальный редактор со встроенным морфологическим анализатором и внутренним скриптовым языком, использующимся для более гибкого описания семантических знаний.

Публикации и апробации работы. По материалам диссертационной работы опубликовано 1 статья, 8 тезисов на Всероссийских и международных конференциях, получено 3 свидетельства о государственной регистрации программ для ЭВМ, в которых были использованы методы, разработанные в рамках диссертационной работы. Результаты настоящего исследования были представлены на следующих научных конференциях и семинарах:

1. Научно-техническая конференция студентов, аспирантов и молодых специалистов МИЭМ в 2008 г.;
2. Научно-техническая конференция студентов, аспирантов и молодых специалистов МИЭМ в 2009 г.;
3. Международная конференция «Диалог 2009» (Бекасово, 27-31 мая 2009 г.);
4. Пятая азиатская международная школа-семинар “Проблемы оптимизации сложных систем” ИВМиМГ СОРАН в 2009 г.;
5. Международная научная конференция Megaling 21-26 сентября 2009, Украина, Киев;
6. Научно-техническая конференция студентов, аспирантов и молодых специалистов МИЭМ в 2010 г.;
7. Международная конференция «Диалог 2010»;
8. 11th Annual Conference for Software Testing Professionals "Discover the Tester in You"(July 6, 2010, University of Latvia, Riga).

Результаты диссертационной работы вошли также в следующие отчеты о НИР:

1. "Научно-методическое обеспечение процессов анализа диссертационных дел и авторефератов соискателей ученой степени на основе использования информационно-коммуникационных технологий", № гос. регистрации 0120.0 810155, инв. номер 02200900612, 2008 г.;
2. "Разработка теоретических положений решения задачи семантического анализа результатов выполненных НИР и ОКР

на примере ФЦП "Исследование и разработка по приоритетным направлениям развития научно-технологического комплекса на 2007-2012 г.", № гос. регистрации 0120.0 808037, инв. номер 02200900878, 2008 г.

Структура диссертации и объем работы. Диссертация изложена на 173 страницах машинописного текста и состоит из введения, обзора литературы, обсуждения результатов, экспериментальной части, выводов и списка литературы, включающего 73 источника. Диссертация иллюстрирована 44 рисунками и содержит 4 таблицы.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обоснована актуальность работы, сформулированы цель и задачи диссертационного исследования, новизна и практическое значение полученных результатов, области и методы исследования, а также положения, выносимые на защиту, обоснованность и достоверность, реализация и апробация работы. Кратко изложено содержание диссертации по разделам.

В первой главе проводится обзор существующих средств морфологического анализа: парсера Mystem, компонента морфологического анализа системы ДИАЛИНГ, инструментария ABBYY Retrieval & Morphology Engine. Дано также описание методов и средств синтаксического анализа, таких как: система LinkParser, компонентов синтаксического анализа систем ЭТАП-3 и ДИАЛИНГ, аппарата расширенных сетей переходов. Проводится обзор подходов к семантическому анализу: систем ПОЛИТЕКСТ и ДИАЛИНГ. Рассматриваются существующие модели автоматического понимания текстов, обосновывается выбор модели мягкого "понимания" текста для поставленных задач. Проводится обзор таких близких подходов в программировании, как литературное программирование, языко-ориентированное программирование и концептуальное программирование, а также существующих программных реализаций упомянутых парадигм. Дается обзор моделей и построенных на их основе программных средств извлечения знаний из текста (процессора Semantix и системы

”Аналитический курьер”). В конце главы формулируются выводы. На основе проведенного обзора существующих методов, моделей и подходов, определяется и обосновывается выбор модели концептуальной зависимости в качестве семантической модели языка и концептоцентрического подхода для конструирования базы знаний.

Во второй главе дается общее описание архитектуры программного комплекса Nalaps. Предложенная структура состоит из следующих

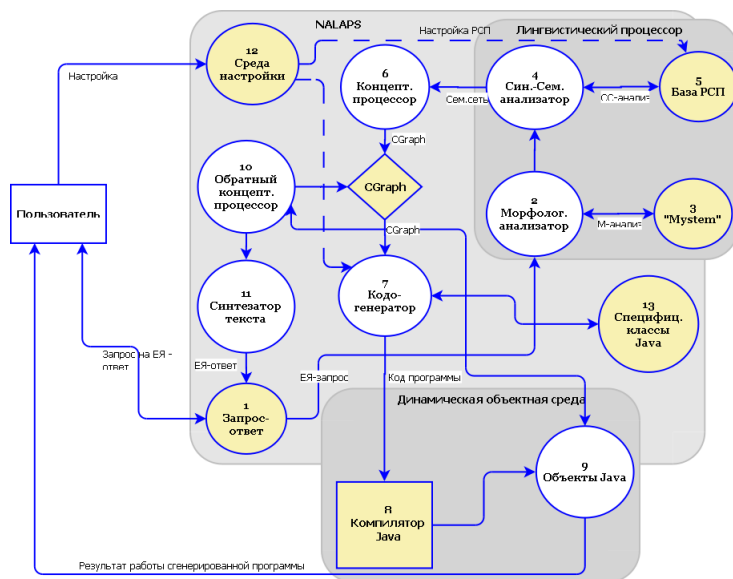


Рис. 1: Предложенная архитектура системы Nalaps

основных компонентов (рис. 1):

- лингвистический процессор, используемый для получения синтактико-семантического представления поданного на вход естественно-языкового текста;
- концептуальный процессор, формирующий концептуальный граф на основе синтактико-семантического представления и базы концептуальных знаний;
- генератор кода программы на языке Java, синтезирующий текст программы, путем интерпретации концептуального графа через динамическое создание объектов и вызов их методов; данный

компонент отвечает также за создание динамической объектной системы, представляющей "смысл" исходного естественно-языкового запроса;

- обратный концептуальный процессор, отслеживающий состояние связей и новых объектов в динамической объектной системе и реализующий возможность отражения измененных и новых связей и объектов в концептуальном графе (обратный переход).

Дается общее описание целей и задач каждого компонента системы. В главе также описывается выбранная модель лингвистического процессора. Лингвистический процессор состоит из морфологического и синтактико-семантических анализаторов. В роли морфологического анализатора выступает бесплатный морфологический парсер Mystem компании Yandex, для взаимодействия с которым в рамках диссертационного исследования реализована программная оболочка на языке Java. Синтактико-семантический анализатор реализован на основе формализма расширенных сетей переходов (ATN). Для повышения эффективности и облегчения процесса программирования такого лингвистического процессора в ходе исследований разработан визуальный редактор расширенных сетей переходов. В его описание включены характеристики и функциональные возможности редактора, а именно:

- графический интерфейс пользователя для редактирования графовидной структуры сетей;
- встроенный морфологический анализатор для автоматизированного создания скелетов сетей по введенной шаблонной фразе;
- вспомогательные интуитивные подсказки в интерфейсе пользователя;
- поиск по графам с учетом рекурсивного включения;
- внутренний простейший скриптовый язык для маркировки различных семантических единиц.

Раскрывается ряд усовершенствований и эвристических подходов, примененных при проектировании настоящего ATN-анализатора, среди которых:

- использование внутреннего простейшего скриптового языка, где реализована возможность использования глобальных переменных, хранящих заданные или динамически присвоенные значения семантических единиц;
- некоторые эвристические подходы выбора путей обхода дерева расширенной сети переходов при мультивариантном морфологическом анализе лексем.

В конце второй главы приводятся сведения об апробации данной модели лингвистического процессора в других научно-исследовательских работах. Подчеркнута модульная архитектура компонентов системы Nalaps, что дает возможность в случае необходимости произвести элементарную замену существующего лингвистического процессора на иной, построенный с использованием более совершенных и подходящих механизмов.

В третьей главе предлагается такая структура базы концептуальных знаний, где в качестве концепта предлагается рассматривать описание класса объектно-ориентированного языка программирования, а настройку на предметную область и конкретный естественный язык производить с помощью конфигурационных файлов на языке XML. Обоснованием предложенного подхода является близость понятий класса из объектно-ориентированной модели языка программирования и лингвистического понятия концепта. С позиции концептуального подхода взаимосвязь между понятиями внеязыковой действительности выражается в классах объектно-ориентированных языков программирования. Данная взаимосвязь определяется в виде наличия типизированных полей, методов, типизированных по параметрам и возвращаемым значениям. Связь с естественно-языковыми терминами, описывающими концептуальные знания, представляется на языке XML. Работа концептуального процессора описывается с приведением примеров его работы. Основополагающие моменты, такие как автоматический переход от формализованного представления текста к его программной интерпретации на языке Java, оформлены в виде математической модели в основу которой положен аппарат теории множеств и теории графов.

I. Работа морфологического анализатора.

1. Пусть $S = x_1x_2x_3x_4x_5...x_n$ - входная строка, где x_i - символы алфавита $A = \{ _ A..Za..zA..Яa..я0..9.,?! \}$.

2. Функция $St(S) = \{st_i | i = \overline{1..N}\}$ разбивает строку S на предложения st_i (членение по символам "!.?").
3. Функция $Wd(st) = \{w_i | i = \overline{1..M}\}$ разбивает предложение st на слова и знаки пунктуации (членение по символам "_,").
4. Каждому слову w отображение Mr ставит в соответствие фреймовую структуру f_m :
например, для словоформы "Петра":

Атрибут (p_i)	Множество значений (V_i)
Неопределенная форма	Петр
Падеж	род., вин.
Число	ед.
...	...

$$f_m = \{p_i, V_i = \{v_{ij} | j = \overline{1..N_{v_i}}\} | i = \overline{1..L_d}\}$$

Множество всех возможных фреймовых структур:

$$F_m = \{f_{m_k} | k = \overline{1..M}\}$$

II. Работа синтактико-семантического анализатора

Определение 1. Расширенная сеть переходов - это связный ориентированный граф: $atn = (Nd, T, label_{Nd}, label_T)$,
где Nd - упорядоченный набор вершин,
 T - упорядоченный набор ребер графа.

- Отображение $label_{Nd}(nd \in Nd) = \{relation_i(nd_k, nd_l) | i = \overline{1..N}\}$ задает метки вершин графа.
- Отображение $label_T(t \in T) = \{f_{m_i} | i = \overline{1..N}\}$ или $tr \in V_n$, где V_n - словарь нетерминалов, задает метки ребер графа.

Определение 2. Машина сети переходов: $M_{ATN} = (V_T, V_N, S, M, L)$,
где:

- V_T - словарь терминалов, совпадает с F_m ,
- V_N - словарь нетерминалов,
- S - начальное состояние,
- M - набор конечных автоматов со входным словарем $V = V_T \cap V_N$,

- I - функция индексирования, назначающая каждому $v_{N_i} \in V_T$ уникальный автомат из M .

Результат работы M_{ATN} - путь обхода сетей:

$$M_{ATN} \circ Mr \circ Wd \circ St(S) = \{atn_{k_1}.nd_{l_1}, atn_{k_1}.nd_{l_2}, \dots, atn_{k_n}.nd_{l_m}\},$$

где $atn_{k_i}.nd_{l_j}$ - конкретные вершины (nd_{l_j}) конкретных сетей (atn_{k_i}).

Синтактико-семантическое представление SSP - это ориентированный граф, с вершинами $atn_{k_i}.nd_{l_j}$ и ребрами, инцидентность которых вычисляется путем анализа меток: $label_{Nd}(nd \in Nd) = \{relation_i(nd_k, nd_l) | i = \overline{1..N}\}$.

Каждому узлу $atn.nd$ графа ставится в соответствие f_m , соответствующая терминалу (слову) из входного потока.

Каждому ребру, где $relation$ описывается через терминал, ставится в соответствие f_m , соответствующая этому терминалу (слову) из входного потока.

III. Проблема интерпретации формализованного представления ЕЯ-текста на языке программирования.

В общем случае задача состоит в построении проекции Π из CG_{JAVA} в CG_{NL} , такой, что, $CG_{NL} \leq CG_{JAVA}$ (рис. 2).

Пусть CG_{Ob} - подграф CG_{NL} для отдельного объекта ЕЯ. Он задается путем выделения узла самого объекта, а также всех узлов и ребер, с которыми связан данный узел. CG_{Class} - концептуальный граф отдельного класса Java.

Тогда:

$$CG_{NL} = CG_{Ob_1} + CG_{Ob_2} + \dots + CG_{Ob_N} = \Pi(CG_{JAVA}) = \Pi(CG_{Class_{K_1}} + CG_{Class_{K_2}} + \dots + CG_{Class_{K_N}}) = \Pi(CG_{Class_{K_1}}) + \Pi(CG_{Class_{K_2}}) + \dots + \Pi(CG_{Class_{K_N}}).$$

Таким образом, с каждым Ob необходимо сопоставить специфицированный класс $Class_{Java}$, каждое отношение отношения типа "свойство" интерпретировать как поля класса, а остальные - как методы класса.

Ont(nat) = (Объект, свойство, отношение)

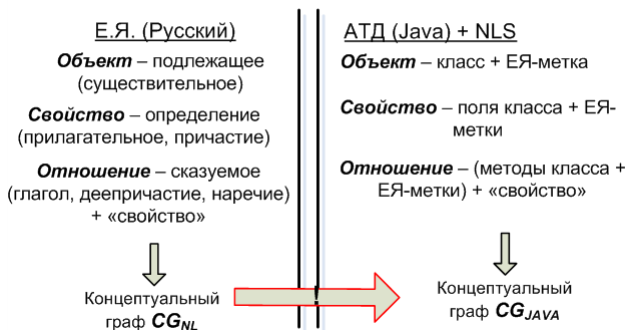


Рис. 2: Проблема интерпретации формализованного представления ЕЯ-текста на языке программирования

Определение 3. Концептуальный граф CG задается как ориентированный граф:

$CG = (C, U, label)$, где:

- C - множество концептов, $C \neq \emptyset$, задающее вершины графа,
- U - множество ребер, задающее отношение между концептами.
- Каждая вершина c имеет метку, определенную отображением $label(c) = \{f_{m_{c_i}} | i = \overline{1..N_c}\}$, которое сопоставляет с вершиной конкретную лексему и ее фреймовую структуру.
- Каждое ребро u имеет метку:
 $label(u) = \{\text{"свойство"}\} \cup \{f_{m_{u_i}} | i = \overline{1..N_u}\}.$

CG_{NL} задается с помощью SSP , полученного на выходе из синтактико-семантического анализатора.

IV. Связь с концептуальным графом Дж. Совы.

Рассмотренная структура не является концептуальным графом Дж. Совы. Отличие заключается в способе представления отношений между концептами (см. рис. 2). На рис.2 представлено схематичное представление падежного фрейма Ч. Фильмора в представлении концептуального графа Дж. Совы и семантической сети Риченса. Отношение r в определенном в данной работе концептуальном графе рассматривается в смысле отношений в семантической сети Риченса.

Однако существует такое отображение B_r , которое позволяет осуществить

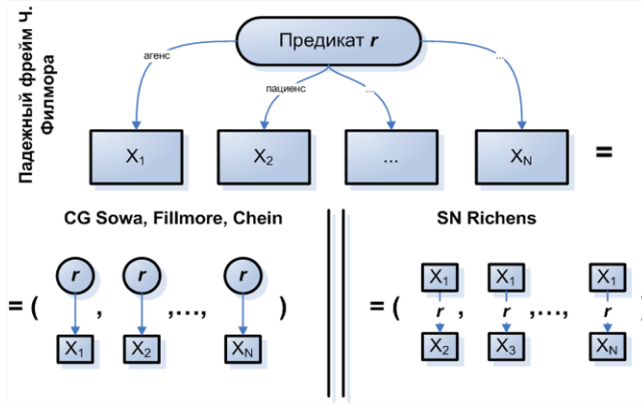


Рис. 3: Представление предиката

переход от любого концептуального графа Дж. Совы к концептуальному графу рассмотренного выше вида:

$$\exists B_r(CG_{sowa}) = CG$$

и соответственно обратный переход:

$$B_r^{-1}(CG) = CG_{sowa}$$

В рассмотрение вводится следующее обозначение. Пусть r - отношение в семантической сети Риченса, тогда $CG_1(r) = x_1$, и $CG_2(r) = x_k$.

Отображение B_r может быть построено следующим образом:

$$B_r = \begin{cases} 1. r : \{r_1, r_2, \dots, r_n\}; r_i = (r, x_i) \\ 2. \exists r_a : r_a = \text{"агенса"} \\ 3. \forall r_i \in r \setminus \{r_a\} \exists r'_i = (x_a, x_i) \end{cases}$$

На первом (1) шаге описывается отношение (предикат) r в концептуальном графе Совы CG_{sowa} , он состоит из набора семантических ролей r_i , каждая из которых является фактически дугой графа и связывает узел r и узел x_i (объект заполненной валентности). На втором (2) шаге находится среди таких r_i семантическая роль *агенса*, определяющая подлежащее (активный участник ситуации) и обозначается как r_a . На третьем (3) шаге для каждого r_i исключая r_a конструируется отношение r'_i нового концептуального графа, являющееся фактически дугой, связывающей агента действия (x_a) и соответствующего r_i участника ситуации.

Обратный переход к концептуальному графу Дж. Совы описывается отображением B_r^{-1} :

$$B_r^{-1} = \begin{cases} 1. r_i^i = (x_a, x_i) \\ 2. \forall r_i' \exists r_i = (r, x_i) \\ 3. r_a = (r, x_a) \end{cases}$$

На первом (1) шаге рассматриваются отношения данного концептуального графа r_i' , являющиеся ребрами графа, соединяющими агента действия (x_a) с другими объектами ситуации (x_i). На втором (2) шаге для каждого r_i' конструируется r_i - ребро графа События, соединяющее узел r - отношение, и x_i - участника называемой предикатом ситуации. На третьем (3) шаге восстанавливается семантическая роль *агенса* как ребро от узла r к узлу x_a .

Для концептуального графа Дж. События определены следующие операции. Концептуальный граф H может быть выведен из концептуального графа G путем:

- копирования;
- удаления (любого концепта, при сохранении связности графа и отсутствии "висячих" связей);
- разделения (удаления любой связи, при сохранении связности графа);
- обобщения;
- уточнения;
- соединения.

Поскольку построено биективное отображение из концептуального графа Дж. События в предложенную модель конц. графа, то данные операции применимы и для новой модели. Таким образом, операции *удаления*, *разделения* и *соединения* делают возможным разбиение $CG_{NL} = CG_{Ob_1} + CG_{Ob_2} + \dots + CG_{Ob_N}$. При этом концептуальный граф отдельного объекта (Ob) определяется как:

$$CG_{Ob} = (\{c_{Ob}\} + \{c_i | \exists u_k = (c_{Ob}, c_i)\}, \{u_j | u_j = (c_{Ob}, c_i)\})$$

V. Концептуальный граф класса Java

ЕЯ-специфицированный класс языка Java может быть представлен в виде: $class_{nl_s} = (name, Fl, Mt, label)$, где:

$name$ - имя класса, Fl - упорядоченный набор его полей (атрибутов), Mt - упорядоченный набор методов. $label$ - отображение, задающее

естественно-языковую разметку. Совокупность таких классов образует набор терминологических аксиом T_{Cjava} :

$\forall fl \in Fl : fl = (name_{fl}, cl_{fl} \in T_{Cjava})$, где:

cl_{fl} задает тип поля, а $name_{fl}$ - его имя;

$\forall mt \in Mt : mt = (name_{mt}, \{\emptyset\} \cup \{arg_{mt} \in T_{Cjava}\})$, где:

arg_{mt} - тип аргумента метода.

Определение 4. концептуальный граф CG_{java} класса $class_{nls}$ описывается как:

$CG_{java} = (C_J, U_J, label)$, где:

C_J - упорядоченный набор специфицированных классов, являющихся вершинами графа:

$C_J = \{class_{nls}\} + \{arg_{mt} | mt \in class_{nls}.Mt\} + \{cl_{fl} | fl \in class_{nls}.Fl\}$;

U_J - упорядоченный набор ребер графа:

$U_J = \{"свойство"\} + \{name_{mt} | mt \in class_{nls}.Mt\}$

$\forall u \in U_J : u = (class_{nls}, arg_{mt_i})$ или $u = (class_{nls}, cl_{fl_i})$;

Отображение $label$ задает естественно-языковую разметку (маркеры) для каждой вершины ребра графа в соответствии с разметкой класса:

$label(u) = \{"свойство"\} + \{f_{m_{u_i}} | i = \overline{1..N_u}\}$.

VI. Сравнение фреймовых структур f_m .

Функция σ задает способ сравнения элементов на множестве Fm :

$$f_{m_i} < f_{m_j} \Leftrightarrow \sigma(f_{m_i}, f_{m_j}) < 0,$$

$$f_{m_i} = f_{m_j} \Leftrightarrow \sigma(f_{m_i}, f_{m_j}) = |f_{m_i}|,$$

$$f_{m_i} <> f_{m_j} \Leftrightarrow \sigma(f_{m_i}, f_{m_j}) = 0,$$

где последнее утверждение задает условие *несравнимости* элементов f_m .

Такая функция σ может быть построена следующим образом:

$\sigma(f_{m_i}, f_{m_j}) = \sum_k \chi(p_{i_k}, p_{j_k}) \bullet \prod_k ||\chi(p_{i_k}, p_{j_k})||$, где функция χ задается как:

$$\chi(p_i, p_j) = \begin{cases} 0, V_i \cap V_j = \emptyset \\ 1, V_i \supseteq V_j \\ -1, V_i \subset V_j \\ \frac{1}{2}, |V_i| \geq |V_j|, V_i \cap V_j \neq \emptyset \\ -\frac{1}{2}, |V_i| < |V_j|, V_i \cap V_j \neq \emptyset \end{cases}$$

VII. Построение проекции из CG_{java} в CG_{NL} .

Проекция Π из $CG_{java} = (C_J, U_J, label_J)$ в $CG_{NL} = (C_N, U_N, label_N)$ - это упорядоченная пара $\Pi = (f, g)$ отображений f из U_J в U_N , и g из C_J в C_N , таких, что выполняются следующие условия:

1. $\forall u \in U_J, f(u) \in U_N$, и если $c = CG_{java_i}(u)$, то $g(c) = CG_{NL_i}(f(u))$
2. $\forall u \in U_J, label_J(u) \geq label_N(f(u)) \Leftrightarrow \sigma(f_{m_u}, f_{m_f(u)}) > 0$
3. $\forall c \in C_J, label_J(c) \geq label_N(g(c)) \Leftrightarrow \sigma(f_{m_c}, f_{m_g(c)}) > 0$

В главе также описывается работа генератора кода программ на языке Java на основе полученной концептуальной модели. Генерация кода программ разбивается на два этапа.

Первый этап заключается в генерации объектов для каждого из полученных концептов. Имена объектов генерируются путем транслитерации неопределенных форм лексем, использованных в тексте при их описании.

Второй этап кодогенерации заключается в вызове методов, соответствующим семантическим ролям, которые распознаны в тексте и интерпретируются как имена и характеристики действий (глаголы, наречия). Кроме того, вызываются установщики тех полей класса, которые распознаны как характеристики объекта и для которых найдены соответствующие описания в тексте; указанные связи обозначаются такими семантическими отношениями, как *квалификатив*, *локатив*, *темпоратив*.

В четвертой главе предложенная структура, реализованная в виде программного комплекса Nalaps, описывается с точки зрения технических особенностей реализации. Дается описание пользовательских интерфейсов системы (за исключением визуального редактора расширенных сетей переходов). Раскрываются подходы и методики, примененные в ключевых инженерных решениях, такие как:

- использование стандартной библиотеки языка Java - Reflection, позволяющей программным образом получать описание классов языка Java, динамически генерировать объекты и вызывать их методы;
- создание оболочки для вызова внешнего морфологического анализатора *mystem*, являющегося заменяемым модулем системы;

- создание плагина для среды разработки IntelliJ IDEA, включая описание т.н. целевых концептов (intention), парсера естественного-языка и пр.

В пятой главе рассматривается полученный в результате реализации предложенных методик программный комплекс с точки зрения его апробации в опытных и промышленных масштабах. Дается описание созданных проектов с использованием системы Nalaps, среди которых системы с естественно-языковым текстовым интерфейсом:

- учебно-игровой пример управления объектами трехмерной комнаты;
- среда управления базой генеалогических данных, автоматического построения родословного древа;
- система генерации автоматизированных тестов на основании текстовых описаний ошибок.

В заключении подводятся итоги проделанной работы. Перечисляются основные результаты диссертации и следующие из них выводы.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. Усовершенствован метод описания расширенных сетей переходов. Создан простейший скриптовый язык для гибкой разметки семантических единиц и пометки ложных путей обхода.
2. Разработан визуальный редактор расширенных сетей переходов, имеющий встроенный морфологический анализатор, что позволяет производить поиск по базе РСП и строить их "скелеты" по введенному шаблону.
3. Предложен способ описания единиц базы концептуальных знаний путем связки классов объектно-ориентированного языка программирования с предметной областью с помощью лексематических меток, записываемых на языке XML.
4. Построена математическая модель предложенного метода получения концептуального графа и связи его с концептами предложенной модели базы знаний.

5. Разработан программный комплекс Nalaps, позволяющий создавать и использовать предложенную модель базы концептуальных знаний. Система Nalaps позволяет разработчикам создавать предметные базы знаний параллельно основной разработке приложений, а затем встраивать в разрабатываемые программные средства естественно-языковые интерфейсы. Созданный комплекс Nalaps может применяться в учебных целях, для демонстрации принципов объектно-ориентированного программирования на естественно-языковых примерах. Система Nalaps может применяться в игровой индустрии для эмуляции естественно-языкового интерфейса пользователя. Возможна обработка потоков текстовых данных и настройка системы на регистрацию определенных информационных сигналов с последующей их произвольной обработкой.
6. Найдено применение системы для генерации автоматических тестов для проверки работоспособности программных комплексов на основании поступающих описаний ошибок в текстовом виде.
7. Подход генерации концептуального графа, разработанный в рамках диссертационной работы, применен автором и в других научно-исследовательских работах, в частности в модуле поиска авторефератов системы InfoNIOKR (Свидетельство о государственной регистрации программы для ЭВМ № 2009613349).
8. Система Nalaps и ее основные компоненты зарегистрирована в Роспатенте в сентябре 2010 года:
 - (а) "Nalaps": Свидетельство о государственной регистрации программы для ЭВМ №2010616060.
 - (б) "Визуальный редактор расширенных сетей переходов": Свидетельство о государственной регистрации программы для ЭВМ №2010616061.

Основные результаты диссертации изложены в следующих публикациях:

1. Путилов Г. П., Лебедев А. С. Визуальный редактор расширенных сетей переходов // Качество. Инновации. Образование. 2009. №5. С. 42-49.

2. Лебедев А. С. Система концептуального программирования с использованием лингвистического процессора // Научно-техническая конференция студентов, аспирантов и молодых специалистов МИЭМ. Тезисы докладов. М.: МИЭМ, 2008. С. 154-156.
3. Лебедев А. С. Механизм семантического поиска авторефератов // Научно-техническая конференция студентов, аспирантов и молодых специалистов МИЭМ. Тезисы докладов. М.: МИЭМ, 2009. С. 101-102.
4. Лебедев А. С. Редактор расширенных сетей переходов с графическим интерфейсом пользователя // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог 2009» (Бекасово, 27-31 мая 2009 г.). Вып. 8 (15).– М.: РГГУ, 2009. С. 284-290.
5. Лебедев А. С. Естественно-языковое программирование как средство извлечения знаний // Труды ИВМиМГ, Информатика, 9, Новосибирск, 2009. С. 64-71.
6. Путилов Г. П., Лебедев А. С. Система естественно-языкового концептуального программирования Nalaps // Материалы международной научной конференции Megaling 21-26 сентября 2009, Украина, Киев, С. 65-66.
7. Лебедев А. С. Автоматическая генерация тестов на основании текстового описания ошибки с помощью системы Nalaps// Научно-техническая конференция студентов, аспирантов и молодых специалистов МИЭМ. Тезисы докладов. М.: МИЭМ, 2010. С. 200-201.
8. Путилов Г. П., Лебедев А. С. Создание автоматических тестов по текстовым пользовательским сообщениям средствами системы Nalaps // Материалы международной конференции «Диалог 2010». - 2010 [Электронный ресурс]. URL: <http://www.dialog-21.ru/dialog2010/materials/html/Lebedev.htm> (дата обращения: 01.02.2011).
9. Lebedev A. Automatic test generation using the Nalaps system - natural-language processing tool // 11th Annual Conference for Software Testing Professionals "Discover the Tester in You July 6, 2010, University of Latvia, Riga.

Подписано в печать: 27.02.11

Объем: 1.5 усл.п.л.

Тираж: 120 экз. Заказ №771

Отпечатано в типографии "Реглет"

119526, г. Москва, пр-т Вернадского, 39

(495) 363-78-90; www.reglet.ru