УДК 681.3

# ПОСТРОЕНИЕ ШАБЛОНОВ КОДА ПО ТЕКСТАМ СПЕЦИФИКАЦИЙ

***И. С. Лебедев,***
*канд. техн. наук, старший научный сотрудник*
*Научно-исследовательский институт точной механики*

*Приведен подход к созданию шаблонов кода на основе анализа естественно-языковых структур текстов спецификаций. Рассматривается алгоритм преобразования естественно-языковых конструкций.*

***Ключевые слова*** *— структуры естественного языка, генерация кода, вычисление структур шаблонов, кода.*

Применение моделей естественного языка (ЕЯ) является одной из сложных проблем, стоящих на пути создания и использования комплексов программ обработки текстовой информации в интеллектуальных информационных системах.

Анализ естественно-языковых конструкций вызывает необходимость построения их модели, которая основывается на ряде предположений, оказывающихся основополагающими для ее применения. Любая создаваемая модель ЕЯ изначально содержит некоторые ограничения, так как описывается определенной совокупностью свойств. Одновременно с этим при повышении требований к точности модели резко возрастает объем учитываемых языковых факторов и затрат на получение адекватной структуры лингвистического описания. О выполнении этого условия говорится во многих работах как прикладного, так и теоретического характера Г. С. Цейтина, В. Ф. Хорошевского, Н. Н. Леонтьевой, П. А. Толпегина, В. А. Тузова и др.

В настоящее время для формализации естественного русского языка предлагаются решения, базирующиеся на словарях компаний АОТ, RCO [1, 2] и др. Однако в большинстве случаев полная информация о составе, структуре и организации этих словарей отсутствует ввиду их коммерческого применения, что затрудняет их использование в исследовательских работах. С другой стороны, существуют открытые модели, например семантическая модель ЕЯ профессора В. А. Тузова [3]. Данная модель была выбрана в силу ее проработанности в рамках научно-исследовательской работы «Разработка концептуальных основ проектирования информационных систем» [4], где одним из вопросов стоит анализ возможности автоматически обрабатывать предложения спецификаций на программное обеспечение на основе продукционных правил, например:

*До исполнения Теста камеры Приложение должно предоставить пользователю описание этого Теста.*

*Во время исполнения Теста камеры Приложение должно выдать пользователю подсказку, чтобы он «поймал» изображение, а затем отобразить на экране то, что видит камера через видоискатель.*

*У пользователя должна быть возможность подтвердить или отвергнуть результат Теста камеры в ответ на вопрос от Теста.*

Целью таких спецификаций является помощь программисту в кодировании алгоритмов, в устранении неоднозначностей в понимании отдельных параметров [5]. На основе предложений строятся и уточняются схемы, по которым пишется код:

<Precond: if Тест на камеру не исполнен>
<User:: Param: Экран > <- <App:: Param: описание тестов>
<Action: Представить> <Param: Описание этого теста>
<Postcond: Экран = «описание этого теста»>

Предметная область исследуемых текстов спецификаций позволяет наложить на модель ЕЯ определенные ограничения, связанные, в частности, с уменьшением иерархии классов базовых понятий, что дает возможность избежать обработки значений слов, не используемых при описании программного обеспечения. По текстам спецификаций размером около 1,5 МБ, полученным от нескольких компаний-разработчиков программного обеспечения, на основе семантико-синтаксического словаря В. А. Тузова построен словарь, содержащий около 12 тыс. слов. Благо-

даря ограничениям предметной области удалось в нем упростить описание информации, что является существенным плюсом при пополнении и модификации. В силу этих причин построение структур предложения может основываться на системе приоритетов [6], что позволяет избежать экспоненциального роста количества переборов возможных связей слов от количества слов предложения.

Объекты текстовой информации в полученной адаптированной модели ЕЯ описываются существительными. Эта часть речи представлена как элемент, обладающий множеством характеристик, влияющих на связи с другими словоформами:

$$S^{\{p1, p2, p3, p4\}}$$

где $p1$ — падеж; $p2$ — число; $p3$ — род; $p4$ — одушевленность.

Связи существительного также можно представить в виде предиката:

$$S(Z1{:}\text{Род}, Z2{:}\text{Дат}, Z3{:}\text{Вин}, Z4{:}\text{Тв}, Z5{:}\text{Пред}).$$

В тексте существительные могут быть связаны между собой и составлять единый информационный объект [1, 7].

Каждое существительное обладает определенным набором признаков, которые позволяют идентифицировать его в тексте. Одной из частей речи, характеризующей существительное, является прилагательное. В отличие от существительного прилагательное характеризуется только множеством $\{p1, p2, p3\}$:

$$\text{Pril}^{\{p1, p2, p3\}}.$$

В адаптированной модели прилагательное зависит от характеристик существительного и уточняет обозначаемое им понятие. Его можно рассматривать как функцию над существительным:

$$\text{Pril}(S^{\{p1, p2, p3, p4\}}).$$

Числительные характеризуют количественно объекты предложения. Их синтаксическая роль состоит в уточнении или замене аргумента, подставляемого в функцию. В зависимости от вида эта часть речи может выступать в качестве самостоятельных объектов предложения или давать количественные характеристики существительных:

$$\text{Chisl}^{\{p1, p2, p3\}}.$$

Основой большинства предложений является глагол. Число аргументов глагольной функции конечно. Многие из них в значительной степени определяются семантикой других частей речи:

$$G(x1, \ldots, xn).$$

Аргументы $x1, \ldots, xn$ могут быть различными частями речи. В формальном языке глагол идентифицирует собой действие. Формальное определение этого понятия имеет вид [1]

$$f(x1, x2, \ldots, xn) = \{F; f1, f2, \ldots, fm\},$$

где $f$ — символ (идентификатор) действия; $x1, x2, \ldots, xn$ — аргументы, являющиеся именами объектов и действий; $F$ — последовательность суперпозиций базисных функций или уже описанных функций, выражающая сущность действия; $fi$ — признаки этого действия [1].

В формализованных конструкциях ЕЯ роль глагольной функции заключается в связке аргументов. Общий шаблон глагольной функции можно представить в следующем виде:

$$G(Z1{:}\text{Им}, Z2{:}\text{Род}, Z3{:}\text{Дат}, Z4{:}\text{Вин}, Z5{:}\text{Тв}, Z6{:}\text{Пред}).$$

В любом глагольном предикате обязательно должно быть место для потенциального подлежащего — существительного или объекта на базе существительного в именительном падеже. В приведенном выше шаблоне — это аргумент $Z1$. Аргументы $Z2$–$Z6$ представляют собой объекты существительных предложения в различных падежах. Использование шаблонного описания глагола как основной функции предложения позволяет организовывать связи между словами внутри предложения.

Наречие — неизменяемая часть речи, которая в формализованном языке является функцией над глаголом или другими частями речи:

$$\text{Nar}(G(x1, \ldots, xn)).$$

Аналогично прилагательным, большинство классов наречий также можно рассматривать как функции над глаголами. Поскольку любое действие (глагольная функция) характеризуется определенным набором признаков, то наречие соответствующего типа при взаимодействии с глаголом конкретизирует один из них.

Предлог — служебная часть речи, выражающая отношения между именем существительным, местоимением, числительным и словами других частей речи, а также между существительными. Предлоги, как и все служебные слова, не могут употребляться самостоятельно, они всегда относятся к какому-нибудь существительному (или слову, употребляемому в функции существительного). Они обслуживают управление как вид подчинительной связи, будучи связаны с управляющим компонентом словосочетания:

$$\text{Predl}(S^{\{p1, p2, p3, p4\}}).$$

Роль точки (.) однозначна: она всегда является признаком окончания конструкции. Аналогично точке на конец конструкции указывают восклицательный (!) и вопросительный (?) знаки. Двое-

точие (:) служит признаком того, что в конструкции участвуют несколько одинаковых аргументов, по типу соответствующих первому, стоящему до двоеточия. Более специфична роль запятой (,). С одной стороны, она служит признаком наличия в конструкции множества аргументов одного типа. С другой — уточняет и разделяет сложные аргументы. Вместо запятой может употребляться точка с запятой (;), играющая аналогичную роль в формализованном простом предложении.

Система приоритетов основывается на порядке взаимодействия конструкций, который в упрощенном виде состоит из последовательности шагов, приведенных ниже [6].

1. Присоединение подчиненных прилагательных к существительным. На этом шаге основная информация берется из морфологического описателя словоформы. При первом просмотре предложения слева направо ищутся ближайшие, согласующиеся по падежу, роду и числу, прилагательные и существительные. Так как прилагательное может находиться справа от существительного, то необходим аналогичный второй просмотр справа налево, во время которого осуществляется попытка присоединения оставшихся прилагательных, не вошедших в конструкцию.

2. Присоединение предлогов к конструкциям существительных и прилагательных. Особенностью шага является то, что предлог всегда находится слева от конструкции существительного. Основная информация для реализации свертки — это описатель предлога и морфологический описатель конструкции существительного. Информация по предлогу содержит падеж и семантический класс присоединяемого существительного, а также семантико-грамматический тип (например, *где?, куда?*), вырабатываемый при этом соединении.

3. Присоединение конструкций существительных к другим объектам проводится на основании анализа синтаксических и морфологических характеристик в предикатах левой и правой конструкций слева направо. Вне зависимости от описаний объекты существительных в родительном падеже присоединяются к конструкциям, стоящим слева.

4. Все созданные конструкции вставляются в предикат глагольной функции.

5. Наречия и собранные конструкции, не вошедшие в предикат глагола, приписываются к нему со своим семантико-грамматическим типом.

Следует отметить, что русский язык является довольно регулярным, и исключения из правил составляют не более 10 %.

Причастные, деепричастные обороты, подчиненные предложения, начинающиеся со слова *который*, отделяются перед анализом. Над ними отдельно выполняются описанные выше шаги, а затем полученные конструкции присоединяются к основному предложению.

Составные конструкции типа *если ... то, ни ... ни*, вложенные предложения, начинающиеся с вопросительных союзов *что, где, когда, какой, который* и т. п., союзами *чтобы, хотя* и т. п., анализируются абсолютно так же, как соответствующие конструкции в языках программирования. Например, при анализе конструкций типа *если ... то* слово *если* управляет сборкой сначала первой части предложения, потом присоединяет ее в качестве своего первого аргумента, после чего повторяет то же самое со второй частью предложения, сохраняя в качестве разделителя союз *то* [1].

Полученный анализатор позволяет адекватно вычислять около 85 % структур конструкций.

Обработка информации таким анализатором осуществляется на трех уровнях: морфологическом, синтаксическом, семантическом. Причем для текстов спецификаций семантический уровень представлен 18 классами, необходимыми для вычисления семантико-грамматического типа (СГТ). Ниже приведен вид получаемой структуры.

*До исполнения Теста камеры Приложение должно предоставить пользователю описание этого Теста.*

| *N слова* | *Разбор* | *СГТ* |
|---|---|---|
| *007* | *должно предоставить* | [*Глагол*] |
| *002* | *До исполнения* | [*доКогда*] |
| *003* | *Теста* | [*Чего*] |
| *004* | *камеры* | [*наЧто*] |
| *005* | *Приложение* | [*Что*] |
| *008* | *пользователю* | [*Кому*] |
| *009* | *описание* | [*Что*] |
| *011* | *Теста* | [*Чего*] |
| *010* | *этого* | [*Какого*] |

Анализируя полученные структуры на базе спецификаций, описывающих продукционные правила, можно выделить следующие составляющие [4]: *условие — объект — действие.*

Объект спецификации — это подлежащие предложения спецификации. Внутри структуры спецификации объектом является любое существительное.

Действие — это глагол со своей группой. Группа глагола определяется структурой графа предложения.

Условие выделяется на основе слов-идентификаторов и семантико-грамматических связей (*если, то, когда* и т. д.).

Используя систему описаний можно автоматически преобразовать предложения спецификаций в следующий вид:

<УСЛОВИЕ: До исполнения Теста на камеру >
<ДЕЙСТВИЕ: должно предоставить <<Приложение>>

```
пользователю описание Теста этого
    <ОБЪЕКТ: <<Приложение>> >
```

Рассмотрим выделенный объект предложения спецификации. *<<Приложение>>* обладает своими свойствами и методами. С одной стороны, свойства и методы можно рассматривать как модульные процедуры, для которых имеются стандартные наборы параметров в различных инструментальных средах программирования (например, VC++, BC++ Builder). С другой стороны, возможно построение этого объекта в ходе «чтения» спецификаций. Причем в этом случае можно выделить взаимосвязанные параметры. Например, анализ объекта <<Приложение>> в перечисленных спецификациях позволяет получить следующее описание:

```
<ОБЪЕКТ:  << Приложение >>
ПАРАМЕТР-УСЛОВИЕ1: До исполнения Теста камеры;
ПАРАМЕТР-УСЛОВИЕ2: Во время исполнения Теста камеры;
< ДЕЙСТВИЕ1: должно предоставить пользователю описание
Теста этого>
< ДЕЙСТВИЕ2: должно выдать пользователю  подсказку > >
```

Обозначив действия как процедуры, а параметры — в виде булевых переменных, можно автоматически генерировать псевдообъект:

```
class App
{ bool par1;//ПАРАМЕТР-УСЛОВИЕ1:До исполнения Теста камеры
 bool par2;//ПАРАМЕТР-УСЛОВИЕ2:Во время исполнения Теста
камеры;
obj action1();//<ДЕЙСТВИЕ1: должно предоставить
                // пользователю описание Теста этого>
obj action2();//<ДЕЙСТВИЕ2: должно выдать пользователю
//подсказку>
  };
```

Объекты, заданные условиями, определяют внешние связи между объектами спецификаций. Состояние параметров, идентифицирующих условия, могут быть использованы для выбора действия спецификации:

```
IF(Тест.выполняется == true)
{
par2=true;
App_.action2(par2);
}
```

Предложение спецификации в виде продукционного правила задает некоторую функцию со своими входными и выходными параметрами в виде объектов и их свойств и условием выполнения этой функции.

Если упорядочить последовательность спецификаций и считать, что каждое предложение описывает одну логически законченную функцию, то возможен вариант автоматически создаваемого кода каркаса приложения:

```
App_.initialization();
While(App_.param_n== true)
 Switch(App_.condition)
 {
  case 1: App_.action1(par1); break;
  case 2: App_.action2(par2); break;
...
  case N: App_.actionN(parN); break;
  default :
 }
```

В приведенном примере вначале выполняется блок инициализации приложения. Затем приложение входит в цикл анализа текущего состояния, где в зависимости от текущих условий выполняются функции.

Детализацию анализа предложений спецификаций можно продолжать и дальше, идентифицируя объекты с помощью словарей линейки синонимичных понятий, которым ставится в соответствие нужный кодовый шаблон.

В заключение необходимо отметить, что в статье приведен подход к автоматическому преобразованию текстовой информации спецификаций, разработанный в рамках НИР [4]. Однако даже на таких примерах можно убедиться, что с помощью автоматического анализа текстовой информации реально не только вычислять объекты и их связи, но и создавать правила обработки, основанные на использовании системы морфологических, синтаксических и семантических характеристик естественно-языковых конструкций. Рассмотренные примеры преобразования спецификаций в шаблоны кода требуют ограничений на представление естественно-языковых предложений, но показывают возможность организации управляющих правил для преобразования получаемых структур текстовой информации.

## Литература


1. **Сокирко А. В.** Первичный семантический анализ // АОТ. Технологии. Первичный семантический анализ. http://www.aot.ru/docs/seman.html
2. **Плешко В. В., Ермаков А. Е., Голенков В. П.** RCO на РОМИП 2004 // Тр. Второго российского семинара РОМИП'2004. Пущино, 1 октября 2004 г. / НИИ химии СПбГУ. СПб., 2004. С. 43–61.
3. **Тузов В. А.** Компьютерная семантика русского языка. — СПб.: Изд-во СПбГУ, 2004. — 400 с.
4. **Разработка** концептуальных основ проектирования информационных систем: отчет о НИР (промежуточ.) / СПбГУ; рук. Г. А. Ботвин. — СПб., 2008. — 211 с.
5. **Котляров В. П., Юсупов Ю. В., Швецов В. В.** Метод предсказания количества регрессионных тестов с учетом изменений исходного кода // XXXII неделя науки СПбГПУ: Материалы межвуз. науч.-техн. конф., 24–29 ноября 2003 г. Ч. V. СПб., 2004. С. 19–20.
6. **Лебедев И. С.** Построение семантически связанных информационных объектов текста // Прикладная информатика. 2007. № 5 (11). С. 83–89.
7. **Лебедев И. С.** Способ формализации связей в конструкциях текста при создании естественно-языковых интерфейсов // Информационно-управляющие системы. 2007. № 3. С. 23–26.