А. И. Зобнин,
А. В. Сахарова

Универсальная система разметки текста ObjectATE

1. Предпосылки создания системы

На данный момент отсутствуют многофункциональные средства создания лингвистических текстовых корпусов, позволяющие заниматься лингвистической разметкой корпуса начиная с того уровня (морфологического, поверхностно-синтаксического, семантического и т. п.), который выбирает разметчик и по тем параметрам, которые он задает сам. Однако именно такое средство необходимо для создания лингвистически размеченного корпуса древних письменных памятников. Поскольку лексика и грамматика древних памятников не изучены в полном объеме, а сами тексты не свободны от разного рода ошибок и темных мест, их грамматическая разметка должна быть ручной. В то же время было бы хорошо, если бы применяемая для этой процедуры информационная система позволяла частично автоматизировать разметку.

Создаваемая система обработки текста ObjectATE (Object-oriented ancient text editor) призвана решить эти проблемы. Она разрабатывается и используется в Отделе лингвистического источниковедения Институте русского языка им. В. В. Виноградова РАН с 2006 г. [Зобнин, Маркелова 2006, 2008; Сахарова 2008; Пичхадзе 2005] и пришла на смену предыдущей системе ATE, с помощью которой велась ручная и полуавтоматическая разметка морфологии в древнерусских текстах — переводных памятниках и летописях (переводная антология «Пчела», Киевская летопись по Ипатьевскому списку, Новгородская первая летопись и др.).

Новая система создавалась для осуществления прежде всего ручной синтаксической разметки этих текстов. Однако рутинную часть работы в ней можно будет впоследствии автоматизировать с учетом имеющейся морфологической разметки и формулируемых пользователем правил (морфологических и формально-синтаксических). При этом система призвана быть максимально гибкой и многофункциональной, позволяющей создателю корпуса строить в принципе любые единицы лингвистического анализа по своим собственным (а не только по тем или иным общепринятым) моделям представления данных.

Решение этой задачи было предложено на основе объектно-ориентированного подхода, широко применяемого в программировании. На разработку программы оказала большое влияние информационно-аналитическая система «Манускрипт» [*http://manuscripts.ru*]. Уже в процессе создания ObjectATE авторы познакомились с такими системами обработки текста, как Emdros [*http://emdros.org*] и GATE [*http://gate.ac.uk*]. Эти системы сложно (или даже невозможно) приспособить к решению поставленной задачи, поэтому сомнений в необходимости создания собственной разработки не возникало. Однако знакомство с идеями, заложенными в этих системах, оказалось очень полезным.

2. Функциональные возможности

Система ObjectATE разрабатывается как программное средство для создания, хранения и обработки текстов, проанализированных на любом лингвистическом уровне. Она позволяет заниматься в ручном режиме морфологической разметкой предварительно уже раз-

деленного на словоформы текста, т. е. присваивать словоформам значения морфологических категорий (полей словоформ); при этом пользователь может сам создавать или редактировать список этих категорий и их значений. Лемматизацию, автоподстановку морфологических параметров, создание словников и указателей предполагается внедрить в эту систему в будущем, а в настоящий момент (осень 2008 г.) для этих целей в Отделе лингвистического источниковедения применяется другое программное средство — Редактор древнерусских текстов ATE. Из него в систему синтаксической разметки тексты переносятся уже лемматизированными и морфологически размеченными, хотя при необходимости эту разметку можно дополнять и редактировать.

Однако даже при отсутствии морфологической информации (т. е. если текст только разделен на словоформы) система ObjectATE обеспечивает возможность ручной синтаксической разметки текста, т. е. создания в базе данных новых объектов — единиц синтаксического анализа. Чтобы разбирать текст, предположим, по зависимостям, пользователь должен создать список необходимых ему типов синтаксических связей и начать связывать друг с другом пары словоформ: вершинную и подчиненную ей (т. е. «склеивать» из этой пары словоформ новый объект). После того как пользователь сформулирует, какой именно узел у каждой связи является вершинным, для размеченных предложений становится возможным построение ориентированного дерева зависимостей. Система обеспечивает и возможность создания вспомогательных для синтаксического анализа узлов, функционирующих как аналоги словоформ: например нулевых подлежащих личных глаголов или фантомных эллиптических нулей с указанием на опущенную словоформу.

Но, как известно, синтаксические правила применяются к словосочетаниям и группам слов, а не к отдельным словоформам. Система ATE позволяет заниматься синтаксическим анализом предложений и по группам. Самый примитивный способ такого анализа — просто выделять группу слов, вешая на нее ту или иную метку: скажем, найдя в тексте дательные самостоятельные, выделить все слова, входящие в конструкцию, и создать из них соответствующий объект. При необходимости все члены такой группы можно выде-

лить одинаковым образом, как равноправные объекты, или же одну или несколько словоформ такой группы можно выделить не так, как все остальные.

Однако система предоставляет возможность не просто отмечать в тексте определенные отрезки, но и заниматься полноценным синтаксическим анализом по группам, образующим друг с другом иерархическую структуру. Это значит, что пользователь может создавать в базе синтаксические объекты из других уже существующих синтаксических объектов, которые только в частном случае представляют собой словоформы. Для этого, создав класс синтаксических объектов (в терминологии грамматики составляющих — фразовую категорию), пользователь должен определить, чем он может быть представлен (например, сформулировать, что в качестве сказуемого предложения может выступать одна словоформа, восстановленный ноль, аналитическая конструкция и т. п.). Для того чтобы описать это явление, в системе предусмотрено применение механизма надстроек (задаваемых пользователем множеств словоформ и синтаксических объектов, обладающих определенными свойствами и, следовательно, могущих в силу этих свойств играть определенную синтаксическую роль). Например, надстройка «Глагол-связка» включает в себя как словоформу (глагол *быти* в личной форме), так и синтаксический объект под названием «Аналитическая личная форма» *(еси былъ, былъ бы)*. Создав такую надстройку «Глагол-связка», мы должны оговорить, что синтаксическая группа «Глагол-связка» должна образовываться только из объектов, входящих в эту надстройку.

Если морфологическая информация о словоформах для разбираемого текста уже имеется, система может использоваться для упрощения и частичной автоматизации синтаксической разметки. Для этого также применяется механизм надстроек: он позволяет задавать условия на морфологические свойства словоформы, при которых она может играть определенную синтаксическую роль. Предположим, можно создать надстройку «Сказуемое», в которую будут входить только все личные глаголы, и надстройку «Подлежащее», куда попадут все субстантивы в именительном падеже. Вхождение в надстройку «Подлежащее» окажется в данном случае не достаточным, а только необходимым условием создания связи «Под-

лежащее–Сказуемое», так как, как известно, имя в именительном падеже может играть и другую синтаксическую роль.

Система позволит задавать морфологические условия и на вхождения целой группы во множество объектов (так, вводя группу «Словосочетание с числом» во множество потенциальных подлежащих, можно оговорить, что вершинное числительное в группе должно стоять в именительном падеже), для чего предполагается выводить параметры самой синтаксической группы из параметров входящих в нее словоформ. Можно задавать не только условия вхождения словоформы или синтаксического объекта в надстройку, но и ограничения самого синтаксического объекта — условия сочетания свойств словоформ, в него входящих (предположим, определить согласование подлежащего и сказуемого по лицу: если лицо сказуемого — первое, то его подлежащее — либо ноль, либо местоимение первого лица). Также при необходимости можно оговаривать порядок словоформ относительно друг друга.

Создавая надстройки, т. е. описывая условия на вхождение словоформ и синтаксических объектов в определенное множество, мы можем использовать это множество не обязательно в качестве класса синтаксических объектов, но и просто с целью создания запроса по сформулированным условиям (например, можно создать надстройку «Составное сказуемое с нулевой связкой» и сразу же запросить список всех подобных объектов). Все это означает, что благодаря наличию морфологической разметки и составлению простейших синтаксических правил языка (основных правил согласования и управления, связанных с морфологическим обликом словоформ, а также с порядком слов) существенно снижается вероятность ошибки при синтаксической разметке. Синтаксический объект не создастся, если его части не входят в соответствующие надстройки (скажем, существительное в косвенном падеже не будет трактоваться системой как подлежащее) или если ограничения самого объекта этого не позволяют (если, например, имя стоит в именительном падеже, но отличается от глагола по лицу, из них не получится создать объект «Подлежащее–сказуемое»). Информация о простейших синтаксических правилах языка, которой располагает пользователь, позволяет ему сделать синтаксический анализ полуавтоматическим, используя конструктор объектов, создающий

по сформулированным правилам несколько синтаксических объектов сразу.

В разрабатываемой системе можно также осуществлять ручную разметку текста на более глубоких языковых уровнях, вводя специальные метки (коммуникативный статус, семантическая роль и т.п.) для синтаксических объектов или отрезков предложений.

Наконец, для переводных текстов система ObjectATE предусматривает наличие простейших средств описания соответствий между оригиналом и переводом. При необходимости в систему может помещаться второй текст (оригинал) в виде списка словоформ, что предоставляет пользователю возможность создавать особые объекты анализа (фрагменты перевода), устанавливая соответствия между словоформами перевода и словоформами оригинала.

3. Объектная модель данных

Такую гибкость и такой широкий набор функций система имеет потому, что, как уже было сказано, она решена на основе объектно-ориентированного подхода, широко применяемого в программировании. Этот подход тесно связан с понятием отнологии в информатике.

Весь размеченный документ представляется как набор объектов. Процесс разметки состоит в создании и модификации объектов.

В начале работы пользователь задает метаданные, т. е. данные о структуре будущих объектов. Метаданные состоят из шаблонов и надстроек над ними. Шаблон можно понимать как абстрактный тип данных, определяющий вид объекта. Например, в стандартных текстах, с которыми работает система разметки, предполагаются такие шаблоны, как «Страница», «Строка», «Словоформа». Напротив, конкретные страница, строка или словоформа в тексте — это объекты соответствующих шаблонов. Всякий шаблон имеет уникальное имя.

Каждому шаблону приписан определенный набор полей и ограничений. С помощью полей одни объекты в документе могут быть связаны с другими. Так, строка текста относится к какой-то странице, слова расположены в определенных строках, а всякая словоформа обладает определенной частью речи. Поля шаблона — это набор типов признаков, которые могут быть у объекта этого шабло-

на. Соответственно, каждому полю шаблона приписано имя, а также указано, какие объекты могут выступать в качестве значения этого поля у объектов данного шаблона. Так, пользователь может определить шаблон «Главные члены предложения» с полями «Подлежащее» и «Сказуемое». На поля такого шаблона могут быть наложены естественные ограничения. Эти ограничения относятся и к типу данных значений полей (ясно, что подлежащее не может быть «Страницей», «Строкой» или «Частью речи») и на сами значения полей и их подполей (например, если подлежащее — это отдельная словоформа, имеющая падеж, то этот падеж должен быть именительным). Ограничения последнего вида можно накладывать на весь шаблон в целом. Такие ограничения записываются в виде логических условий на поля (и их подполя с любым уровнем вложенности). Истинность этих ограничений зависит от потенциального набора значений полей. Предполагается, что для всякого объекта данного шаблона эти ограничения должны превращаться в тождественно истинные выражения.

Шаблоны могут выстраиваться в иерархии наследования. Эта возможность оказывается очень удобной при описании метаданных. Шаблон-наследник приобретает все свойства (поля и ограничения) шаблона-предка, добавляя к ним, возможно, свой набор полей и ограничений. Шаблон-предок может быть объявлен абстрактным. Это значит, что он используется только как общий предок для других шаблонов-наследников, а создавать объекты такого шаблона нельзя. Например, если пользователь хочет наделить все объекты синтаксической разметки полем «Комментарий», он может определить это поле у общего абстрактного шаблона «Синтаксический объект» и вывести из этого шаблона другие шаблоны.

В системе реализован механизм множественного наследования, позволяющий включать один и тот же шаблон в различные иерархии. При этом от идеи условного наследования было решено отказаться. Вместо этого используется механизм надстроек.

Надстройка отдаленно напоминает абстрактный шаблон. Она строится над уже существующими шаблонами или надстройками, которые называются кандидатами на вхождение в эту надстройку. Каждому кандидату надстройки может быть приписано условие на его вхождение в надстройку. Как и ограничение шаблона, это

условие представляет собой логическое выражение, зависящее от конкретного объекта, его полей, подполей и т. д. Можно индуктивно определить понятие реализации объектом надстройки или шаблона. Всякий объект *O* реализует свой собственный шаблон и все шаблоны-предки этого шаблона. Далее, пусть *K*—кандидат надстройки *H* и объект *O* реализует *K*. Тогда считается, что *O* реализует надстройку *H*, если для объекта *O* выполнено условие на вхождение кандидата *K* в *H*.

Надстройки появились в модели по крайней мере по двум причинам. Во-первых, механизм надстроек позволяет детально задать условия на поля шаблонов, а во-вторых, надстройки позволяют описывать простые запросы к данным. Рассмотрим эти возможности подробнее. Ранее полю шаблона строго сопоставлялся его тип—другой шаблон. Считалось, что только объекты этого другого шаблона могут являться значениями полей. Это вызывало определенные трудности, прежде всего с «нулевыми» синтаксическими объектами. Нужно было сделать так, чтобы синтаксические нули наравне со словоформами могли быть полями других синтаксических объектов. Однако в случае, когда такие поля выражены словоформами, должно было выполняться дополнительное условие. В нынешней модели типом поля шаблона может быть или шаблон, или надстройка. Соответственно, объект может быть значением такого поля, если он реализует его тип. Такой подход позволяет более гибким образом описать модель разметки. При этом в программе имеется возможность проверить, реализует ли данный объект указанную надстройку, вывести список надстроек, реализуемых данным объектом, а также вывести все объекты, реализующие данную надстройку. Сами эти объекты могут иметь разные шаблоны; их объединяет лишь то, что при выполнении условий вхождения мы относим их к данной надстройке. Поэтому надстройки удобно рассматривать как описания простых запросов к данным, т. е. таких запросов, которые возвращают отдельный список объектов.

Надстройка, как уже было сказано, задает достаточные условия для отнесения объекта к некоторой категории. В системе предусмотрен простой механизм, позволяющий показать, что для данного объекта данная надстройка задает и необходимые условия.

Всякий объект имеет обязательную текстовую компоненту «Содержание». Содержание объекта может либо задаваться пользователем, либо вычисляться по определенным правилам через содержания полей. Объекты имеют также два поля для сортировки и сравнения: дескрипторы начала и конца объекта. Считается, что все объекты данного фиксированного шаблона можно естественным образом упорядочить по их дескрипторам. Если дескрипторы начала и конца различаются, то объект считается «протяженным». Так, естественный порядок имеется на страницах, строках и словах текста. Удобно считать слово «атомарным» объектом, а дескрипторы начала и конца строки приравнивать к дескрипторам первого и последнего слова в строке. Аналогично, дескрипторы начала и конца страницы приравниваются соответственно к дескрипторам первой и последней строки в этой странице. Правила назначения дескрипторов новым объектам можно задавать при описании метаданных. Дескрипторы позволяют строить запросы и ограничения на порядок слов (например, найти все связи «Субстантив–атрибут», в которых субстантив находится раньше атрибута).

Поля шаблонов могут быть трех видов: обычные поля, коллекции и диапазоны (архитектурно предусмотрен четвёртый вид — коллекция диапазонов, но он пока не реализован, так как на данный момент не востребован). Поле-коллекция отличается от обычного поля тем, что предполагает сразу несколько различных значений. Диапазон — это «связная» коллекция, т. е. множество объектов, идущих подряд в смысле упорядочения по дескрипторам. Для диапазона достаточно задать начальный и конечный объект. Типичный пример диапазонов — строки в странице или какие-либо естественные связные большие фрагменты текста (например, блоки, части, прямая речь и т. д.).

Поля шаблонов делятся на обязательные и опциональные. Обязательное поле заполняется при создании объекта (например, при синтаксической разметке). Для опциональных полей предлагается список возможных вариантов заполнения. Данный список формируется на основе ограничений шаблона и уже заполненных полей. Например, в «Киевской летописи» шаблон «Словоформа» имеет необязательные поля «Часть речи» и «Разряд», а также ограничение

```
IF ([Часть речи]= 'числительное',
[Разряд] IN {'количественное', 'порядковое', 'собирательное'})
```

Таким образом, если часть речи для данной словоформы определена как числительное, список допустимых вариантов для поля «разряд» будет состоять из трех указанных значений.

Если все кандидаты надстройки имеют общие поля, то при записи условия на поле типа этой надстройки такие поля можно использовать в выражениях. Кроме того, надстройки могут иметь свои поля. Собственные поля надстройки всегда являются опциональными. Объект приобретает такое поле только в том случае, если он реализует надстройку. С помощью подобного механизма удобно описывать морфологическую разметку. Именно так была организована морфологическая аннотация в базе данных «Новгородская первая летопись». В этой модели, например, словоформа имела только поле «Часть речи», а другие морфологические поля появлялись у нее лишь в том случае, если она реализовывала какие-либо надстройки. Так, поле «Падеж» возникало только у словоформ, реализующих надстройку «Имя», и т. д.

Условия и ограничения в метаданных задаются на специальном языке, который интерпретируется программой. Пользователь может создавать их как с помощью конструктора ограничений, так и записывать вручную. Язык содержит основные логические операторы AND, OR, NOT, операторы равенства (=), неравенства (<>), принадлежности (IN) и непринадлежности множеству (NotIN). В выражениях могут участвовать поля и их подполя с любым уровнем вложенности. Имена подполей задаются в квадратных скобках и разделяются точкой. Поле-коллекция всегда рассматривается как множество; кроме того, множество может описываться в явном виде — перечислением входящих в него объектов, которые записываются в фигурных скобках. По умолчанию сравнение объектов производится по их содержанию. Вот пример ограничения на шаблон «Связь с согласованным атрибутом»:

```
([Атрибут].[Часть речи] IN {'прилагательное', 'причастие'})
OR  (([Атрибут].[Часть речи] = 'местоимение')
AND ([Атрибут].[Лицо] NotIN {'1-е', '2-е', '3-е'})
AND ([Атрибут].[Лексема] NotIN {'и'}))
OR  ([Атрибут].[Часть речи] = 'числительное').
```

(Записанное здесь условие на лицо атрибута просто означает, что это лицо отсутствует.)

Перечислим еще некоторые важные операторы этого языка.

1. Оператор проверки реализации IS. Он позволяет проверить, реализует ли данное поле объекта указанную надстройку или шаблон, ср., например, «[Атрибут] IS Словоформа». Также в синтаксис языка ограничений добавлено ключевое слово Me, обозначающее сам проверяемый объект. В условиях на вхождение в надстройку удобно писать выражения вроде «Me IS Субстантив».

2. Условный оператор IF. С его помощью можно корректно обращаться к полям объектов, которые, вообще говоря, не являются общими. Вместе с оператором IS он частично заменяет механизм надстроек, обеспечивая большую гибкость. Пусть, например, поле «Подлежащее» может быть выражено как словоформой, так и нулем. Пусть шаблоны «Словоформа» и «Ноль» безусловно входят в некоторую надстройку. У шаблона «Ноль» нет поля «Падеж»; к падежу можно обратиться только у «Словоформы». Поэтому условие на подлежащее можно записать так:

```
IF ([Подлежащее] IS Словоформа, [Подлежащее].
[Падеж] = 'именительный').
```

3. Операторы сравнения <= и >= позволяют сравнивать объекты по их дескрипторам сортировки и, в частности, строить запросы на порядок слов.

3. Интерфейс программы

Большое внимание постоянно уделяется интерфейсу программы и повышению удобства работы с ней. Последние изменения связаны с новой панелью свойств, новым подходом к выделению и подсветке объектов, настраиваемой структурой программных окон, панелью для сортировки объектов, панелью шаблонов. Был усовершенствован диспетчер шаблонов и надстроек. Кроме того, реализованы новые функциональные возможности, предусмотренные измененной объектной моделью (проверка реализации объектом надстройки, отображение всех объектов заданной надстройки или шаблона и т. д.).

Ключевым понятием интерфейса программы является панель объектов. Панели объектов бывают разных видов; их основная за-

дача—отображать специальным образом определенные объекты. На данный момент предусмотрены следующие виды панелей объектов:

- панель навигации (содержащая простой список, например, страниц или годов в летописи с возможностью поиска);
- панель основного текста (центральное окно программы; содержит текст в формате RTF, построенный из объектов типа «страница», «строка», «словоформа» и т. д.);
- панель-список (содержит перечень объектов с указанием их шаблона);
- панель с возможностью сортировки (помимо функций панели-списка она позволяет изменять взаимное расположение объектов, то есть переупорядочивать их). В первую очередь эта панель была создана для указания порядка слов в греческом тексте;
- панель-дерево (для отображения иерархической информации, такой как словоуказатель, схема синтаксических связей, геометрическая иерархия текста и т. д.);
- панель свойств (содержит информацию о свойствах всех выделенных объектов).

Взаимосвязи между панелями, а также порядок действий по их наполнению описываеются в отдельном xml-файле. Это тоже своего рода «метаданные», относящиеся к интерфейсу.

Пользователь может выделять в панелях группы объектов. Каждая группа имеет свой цвет (всего бывает четыре группы). Разделением выделенных объектов на группы удобно пользоваться при создании новых объектов: в этом случае каждая группа выделенных объектов соответствует отдельным наборам обязательных полей. Список выделенных объектов теперь совмещен с панелью свойств. Возможности панели свойств по отображению форматированного текста существенно расширены.

Кроме того, во внешнем xml-файле описаны правила подсветки других объектов при выделении. Подсветка—это дополнительное программное цветовое выделение отдельных объектов в панелях. Например, можно указать правило, по которому при выделении синтаксического объекта будут подсвечиваться все входящие в него словоформы. Удобно также пользоваться правилом подсветки всех словоформ, имеющих тот или иной морфологический признак.

Это позволяет «на месте» наглядно видеть результаты простейших запросов. При выделении нескольких объектов программа подсвечивает по заданным правилам их общие поля.

Фрагменты окон работающей программы приведены на рис. 1, 2 и 3.

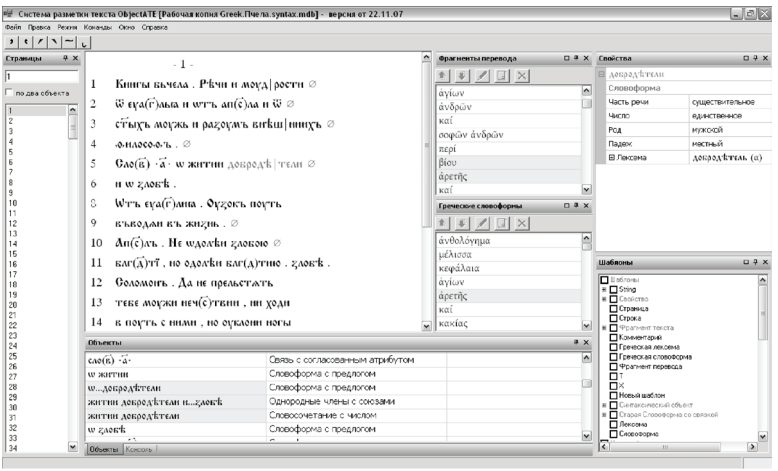

Рис. 1. Панели программы. Выделенные объекты и подсветка объектов

В системе создается механизм работы с фрагментами текста, связанными синтаксическими связями. На уровне метаданных задаются правила перехода от шаблонов к их полям и обратно. Эти правила позволят строить деревья синтаксических зависимостей, автоматически вычислять границы предложений (или иных фрагментов текста) по указанному объекту-представителю (главному слову) и т. д. Так, если размечены бинарные связи между членами предложения и указана вершина (сказуемое, причастие в причастном обороте и т. д.), то по этим связям можно, начиная с вершины, вычислить все объекты, входящие в это предложение, и отобразить их в виде дерева. Такой подход позволяет единообразно описывать правила конструирования как синтаксических деревьев, так и словоуказателей.

Текущая версия системы реализована на платформе Microsoft .NET Framework с использованием реляционных баз данных Microsoft Access и Microsoft SQL Server.

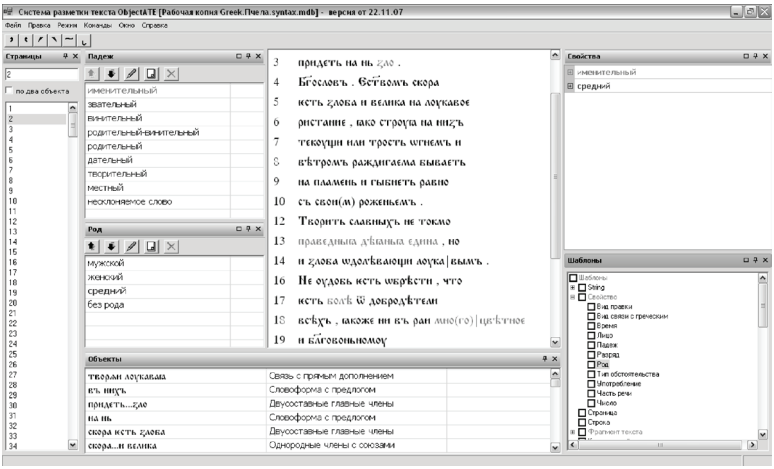

Рис. 2. Выполнение простейших запросов с помощью подсветки (запрос на словоформы среднего рода в именительном падеже)

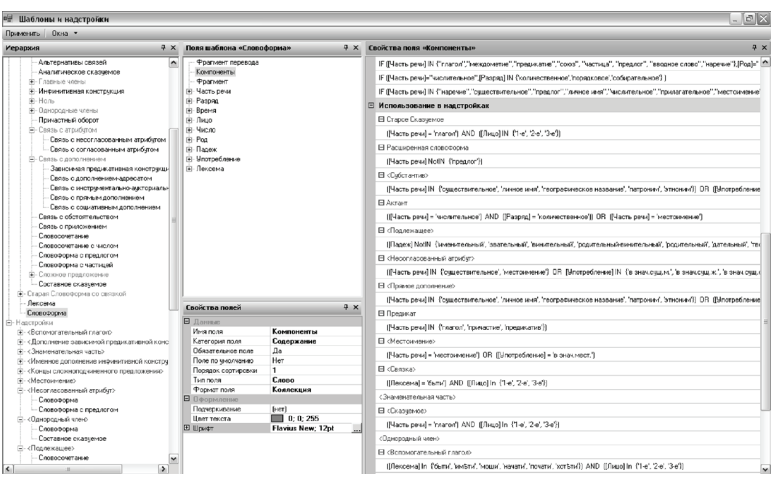

Рис. 3. Диспетчер шаблонов и надстроек. Показаны свойства шаблона «Словоформа»

Зобнин, Маркелова 2006—Зобнин А. И., Маркелова А. В. Универсальная система разметки текста ATE-2 // Современные информационные технологии и письменное наследие: от древних рукописей к электронным текстам: Материалы международной научной конференции (Ижевск, 13–17 июля 2006 г.). Ижевск, 2006. С. 51-55.

Зобнин, Маркелова 2008—Зобнин А. И., Маркелова А. В. Универсальная система разметки текста ObjectATE // Современные информационные технологии и письменное наследие: от древних текстов к электронным библиотекам: Материалы международной научной конференции (Казань, 26–29 августа 2008 г.). Казань, 2008. С. 114–117.

«Манускрипт»—Информационно-аналитическая система. *http://manuscripts.ru*.

Пичхадзе 2005—Пичхадзе А. А. Корпус древнерусских переводов XI–XII веков и изучение переводной письменности Древней Руси // Национальный корпус русского языка: 2003–2005. Результаты и перспективы. М., 2005. С. 251–262.

Сахарова 2008—Сахарова А. В. Возможности применения универсальной системы синтаксической разметки текста ObjectATE // Современные информационные технологии и письменное наследие: от древних текстов к электронным библиотекам: Материалы международной научной конференции (Казань, 26–29 августа 2008 г.). Казань, 2008. С. 247–249.

Emdros—The database engine for analyzed or annotated text. *http://emdros.org*.

GATE—General Architecture for Text Engineering. *http://gate.ac.uk*.