ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕЕНОЕ НАУЧНОЕ УЧРЕЖДЕНИЕ

«ИНСТИТУТ ИНФОРМАТИЗАЦИИ ОБРАЗОВАНИЯ»

РОССИЙСКОЙ АКАДЕМИИ ОБРАЗОВАНИЯ

ВАГРАМЕНКО Я.А., ЯЛАМОВ Г.Ю., ФАНЫШЕВ Р.Г.

НАУЧНАЯ СТАТЬЯ

«БАЗА ЗНАНИЙ В ИНФОРМАЦИОННОЙ СИСТЕМЕ ДЛЯ САМООБУЧЕНИЯ»
(ДЛЯ ПРЕПОДАВАТЕЛЕЙ И СПЕЦИАЛИСТОВ СИСТЕМЫ УПРАВЛЕНИЯ ОБРАЗОВАНИЕМ), 1 П.Л.

Тема исследования: «Интеллектуализация информационных систем формирования и представления распределенного контента образовательного назначения, ориентированного на самообучение»

Задача исследования: Обосновать и разработать научно-методические подходы к формированию базы знаний в информационных системах, ориентированных на самообучение

Москва – 2014

Современные достижения в области информационных технологий позволили за короткий промежуток времени скопить в хранилищах данных различных организаций большие объемы информации, которая содержит скрытую информацию в виде знаний, поэтому задача аналитической обработки больших объемов информации становится весьма актуальной. Центральное место в процессах интеллектуального анализа естественно – языковой информации занимают следующие технологии: Data Mining, Text Mining и Semantic Web Ontology (язык OWL – Ontology Web Language), задачей которых является получение ранее неизвестных либо не выявленных знаний и закономерностей фактов в больших хранилищах данных. Источниками исходной информации для аналитической обработки могут являться базы знаний разных типов. Например, огромнейшая база знаний Internet, хранилища данных различных организаций. Значительная часть информации в этих источниках представлена в виде естественных текстов, процесс аналитической обработки которых требует создания принципиально новых моделей, методик и систем интеллектуального анализа информации [1].

Сложность задачи построения систем интеллектуальных информационных систем обусловлена тем, что она связана с аналитической обработкой естественных текстов. Тем не мене необходимо учитывать тот факт, что нет необходимости проводить извлечение всех закономерностей из естественных текстов для формирования информационно–аналитических структурных компонентов и элементов учебного процесса вузов в силу специфики познавательной деятельности обучаемых. Это позволяет сделать вывод о том, что нет смысла проектировать модель естественного текста на основе глубокого семантического анализа текста. Поэтому, в первую очередь при интеллектуальном анализе текстовой информации в базах знаний экспертной системы учебного назначения необходимо создать унифицированный формат мета-знаний, позволяющий обеспечивать необходимый уровень информативности и спецификации знаниевых единиц, адаптированных к международным стандартам в области технологий информационных систем учебного назначения [2].

В настоящее время при решении вышеперечисленных проблем, на наш взгляд, целесообразно использовать технологии Semantic Web. Это связано с тем, что данная технология сопоставляет любую хранимую информацию с точным смыслом, связанным с этой информацией. Этот смысл, идентифицируется однозначно, даже в случае совпадения лексических единиц (фраз, слов, словосочетаний) встреченных в различных контекстах. Фактически это означает, что любая информация связывается с некоторым

неотделимым от нее контекстом. Здесь можно использовать два подхода извлечения необходимых фактов из естественно–языковых текстов [3, с. 67]:

1) Извлечение всей лингвистической информации (синтаксиса, анафорических связей), а затем на ее основе извлекаются факты.

2) Поиск ключевых слов, которые наращиваются в тексте в соответствии с лингвистическими правилами с целью формирования цепочки, полностью описывающей факт.

Результат применения обоих подходов сводит текстовой анализ к формированию готовых словарей, способных к автоматическому самообновлению [4, 5].

Использование веб-онтологий в рамках учебных процессов дает возможность спецификации основных компонентов учебных дисциплин – лекций, практикумов, лабораторных работ, используемых учебно-методических материалов. Станет возможным организация эффективного распределенного доступа к учебным ресурсам, путем создания единой базы знаний, сочетающей в себе множество учебных дисциплин. Данная база будет фактически распределенной по сети Интернет. Это позволит сделать ее независимой от интерпретации конкретного учебного процесса. Обучающие системы в таком случае будут выполнять роль интеллектуальных агентов, производящих выборки из баз знаний в зависимости от контекста обучения. Здесь возможно построение агентов для автоматического пополнения или редакции такой базы знаний.

В соответствии с принципами Semantic Web, процесс создания электронных документов можно разделить на две части:

- подпроцесс создания веб-онтологии документа, которая содержит ряд терминов и понятийных структуры;
- подпроцесс визуализации содержимого онтологии, т.е. получение содержимого онтологии в некотором виде и формате [6].

Таким образом, в Веб-онтологии определяется смысл используемых понятий, характерных для конкретной дисциплины, т.е. специфицируется объекты предметной области, а с помощью языков трансформаций и форматирования – XSLT и XSL-Fo получается визуальное представление содержимого онтологии в необходимом формате, например HTML, DOC и т.п.

Язык трансформаций XSLT позволяет выполнять трансформации структурированных документов, написанных на XML-подобных языках,

например, OWL. Результатом трансформаций является некоторый набор данных, форматирование которого можно осуществить с помощью XSLT-Fo.

Язык форматирования XSL-Fo позволяет с большой точностью задавать макет и другую стилевую информацию, относящуюся к содержимому документов. Учитывая все вышеперечисленные принципы и возможности, был разработан способ стандартизации элементов образовательного содержимого учебных материалов, так называемого каркаса, для организации электронных материалов учебных курсов с возможностью их последующего вывода как на экран, так и на печать.

Данный способ представляет собой шаблон, описывающий структуру электронных материалов учебного курса. Таким образом, была создана онтология, которая специфицирует структуру и понятия характерные для большинства создаваемых учебных курсов.

Мы понимает под предметной областью всю терминологию, используемую при организации учебного курса: тема, лекция, практическое занятие, лабораторная работа, контрольные вопросы, примеры, списки дополнительной литературы, а также все более мелкие компоненты составляющие каждый из объектов.

Учитывая то, что большая часть самообучения студентов происходит в Интернете или подобных сетевых базах знаний (Intranet и корпоративно-вычислительные системы вузов), была использована технология формирования базы знаний экспертной системы учебного назначения на основе разработанной Веб-онтологии «Учебная дисциплина». Веб-онтология «Учебная дисциплина», состоящая из компонентов учебных дисциплин и материалов, в основу которой легли основные принципы, используемые для структуризации лекционного материалы, практических, лабораторных, курсовых и дипломных заданий. В соответствии с этими принципами была сформирована структура Веб-онтологии «Учебная дисциплина». Было реализовано описание данной онтологии с использованием технологии Semantic Web на языке C# на основе механизма вывода Эйлера. Для того, чтобы разобраться с подходами, которые входят в состав специализированной библиотеки SemWeb на C#, была использована онтология, моделирующая простой конечный автомат [5].

Объединение данной онтологии с онтологией конкретной дисциплины, т.е. конкретными данными, соответствующими учебной дисциплине, была получена полноценная информационная база, с которой можно проводить различные действия. Например, применив к данной онтологии инструкции визуализации XSL и XSL-FO, возможно получить на выходе различные представления учебных материалов.

Таким образом, разработав онтологию «Учебная дисциплина» и заполнив ее конкретными данными, соответствующими учебной дисциплине, было получено эффективное средство, для поддерживающее интеграцию существующих баз знаний в базу знаний экспертной информационной системы для самообучения [5].

Большинство существующих экспертных систем используют в качестве базовых языков Prolog и Lisp. Такой подход мотивируется удобством использования данных языков в задачах искусственного интеллекта. На наш взгляд, такой выбор имеет ряд недостатков. Во-первых, нестандартность семантики этих языков требует специальной подготовки инженеров знаний. Например, на языке Prolog, выполнение программы это не последовательное исполнение команд, а вывод некоторой переменной исходя из начальных значений и правил, а в языке Lisp используется обратная запись для представления арифметических выражений. Соответственно, в этом случае, инженеры знаний должны иметь специальную подготовку для создания ЭС на базовых языках такого рода. Другой недостаток данного подхода, это плохая приспособленность этих языков к задачам разработки графического интерфейса. Удобный, интуитивно понятный графический интерфейс – это важная составляющая качественной экспертной системы, т.к. ЭС, прежде всего, ориентированы на обычных пользователей, без навыков программирования. Язык представления знаний C# Expert основан на языке C#. Такое решение имеет ряд преимуществ. Прежде всего, при разработке ЭС на C# Expert'е может быть применен объектно-ориентированный подход привычный большинству программистов (инженеров знаний), более того, C# популярный язык, на котором легко могут работать также специалисты использующие C++ или Java. Все это делает возможным использование C# Expert широким кругом специалистов (инженеров знаний). C# – это язык платформы .NET. Таким образом, еще одно значительное преимущество выбранного подхода, это возможность разрабатывать ЭС под .NET. На сегодняшний день, .NET – это наиболее современная и перспективная платформа для коммерческих приложений.

Существует множество уже готовых решений под .NET. Следует также отметить, что .NET обеспечивает хорошую межъязыковую совместимость и широкие возможности использования программных компонент (Assembly, COM, DLL).

Основная сложность при создании экспертных систем это представление знаний экспертов в базе знаний наиболее подходящим образом для решения задач в заданной области. Для обеспечения такой возможности, базовый язык

ЭС должен иметь четкий, хорошо структурированный способ представления данных и знаний.

К примеру, хороший способ представления знаний обеспечивает ЭС GURU, разработанная фирмой Micro Data Base Systems, США. Эта система ориентирована на разработку ЭС в области деловых расчетов. К полезным возможностям GURU следует отнести возможность описания массивов как элементов данных, поддержка работы с таблицами и базами данных. Более того, в отличие от многих других ЭС, GURU предоставляет интегрированный подход к обработке данных, позволяя совместное использование наборов правил (продукций) с таблицами и реляционными базами данных. В ЭС GURU реализована гибкая подсистема логического вывода, позволяющая осуществлять как прямой, так и обратный вывод на одних и тех же наборах правил, имеющая способ разрешения конфликтов правил (conflict resolution) с помощью указания их приоритетов и порядка выполнения. Также GURU позволяет работать с нечеткими знаниями и использовать нечеткий логический вывод на основе коэффициентов уверенности, поддерживается возможность использования различных формул для вычисления коэффициента уверенности. Тем не менее, в системе GURU содержится и ряд ограничений, к примеру, отсутствуют средства представления сложных структурированных объектов и понятий сложной структуры данных, не предусмотрена возможность описания процедурных знаний. По-видимому, эти ограничения объясняются спецификой области применения: деловые расчеты, а также, возможно, устарелостью системы в целом.

В качестве, примера другой удачной экспертной системы, можно упомянуть ЭС KEE (Knowledge Engineering Environment). Это фреймовая экспертная система, где основным элементом данных базы знаний является юнит (фрейм). Юниты состоят из слотов, а слоты в свою очередь могут содержать данные простых типов (число, строка и т.п.), таблицы, графику, указатели на другие юниты или процедурные знания, написанные на языке Lisp. В системе KEE также реализован механизм наследования, который позволяет организовывать юниты в иерархические структуры, обеспечивая логически связанное представление информации в базе знаний. Безусловно, фреймовая структура данных, реализованная в системе KEE, обеспечивает более широкие возможности представления данных, чем структуры данных ЭС GURU. Основным недостатком ЭС KEE является использование языка Lisp в качестве базового языка системы, и как следствие сложная семантика базового языка с достаточно нетрадиционной формой записи для большинства инженеров знаний [7].

При генерации результирующего кода слоты фрейма представляются в виде объектов класса Slot и содержаться в коллекции CSharpExpertAbstract.slots. Общая схема работы конвертора показана на рисунке 1.

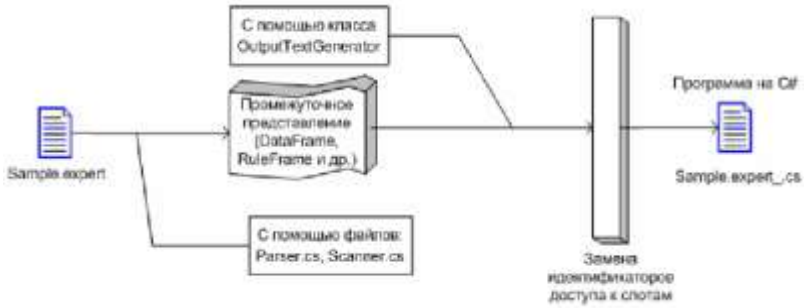

Рис. 1. Общая схема компиляции проекта на языке C# Expert

Точка входа в C# Expert конвертор находиться в методе ExpComp.Main(string[] arg); Метод получает имя исходного файла и запускает метод Parser.Parse() который осуществляет синтаксический анализ и генерацию промежуточного кода. После этого вызывается метод OutputTextGenerator.generateOutputProgram (StreamWriter s) который проводит семантический анализ промежуточного кода, замену идентификаторов доступа к слотам и генерацию результирующей программы на C#.

Ниже опишем подход, предложенный в рамках проекта KnowledgeProspector.Net для извлечения знаний из текста. Кратко можно разбить работу алгоритма на три части:

- морфологический анализ текста и построение по его результатам набора сущностей;
- семантический анализ наборов сущностей и построение по его результатам графа знаний;
- анализ графа.

Первый этап работы алгоритма - это перевод текста с естественного языка в набор сущностей. Данный этап осуществляется при помощи разного вида словарей:

MRD-словаря. Русский словарь содержит около 170 тысяч лемм русского языка. Английский словарь содержит более 100 тысяч. Словарь позволяет производить морфологический анализ слов, узнавать грамматические характеристики слова.

Xml-словари. Словарь, который содержит дополнительные конструкции для перевода слов естественного языка в сущности. Например, для слова «подмножество» можно задать свойство, которое позволит при последующем анализе сущностей рассуждать о отношении между сущностями, связанными этим словом. То есть, появится возможность из предложения «завод – это подмножество зданий», сделать вывод: «завод» – подмножество «зданий».

Имеется возможность создания произвольного словаря. Для этого необходимо реализовать интерфейс IDictionary и подключить словарь к экземпляру класса DictionaryManager.

Второй этап работы заключается в анализе набора сущностей с помощью шаблонов. Результат работы второго этапа – построение графа знаний. Вершинами графа являются сущности, связанные между собой отношениями.

Последующий этап заключается в анализе графа знаний. В задачи данного этапа входит:

- объединение различных свойств в единые классы. Например, свойства «большой», «огромный», «маленький» имеет смысл объединить в один тип.
- очистка графа от ненужных связей. Например, если у родительского класса есть некоторое свойство, то у его наследников его можно не указывать.

В качестве основы языка представления знаний Knowledge.NET возможно использовать фреймовые языки Knowledge.NET и Turbo Expert [8]. В отличие от обычных фреймовых языков, Knowledge.NET расширен языковыми конструкциями для представления онтологических знаний. Семантика онтологии Knowledge .NET схожа с концепцией языка представления знаний OWL, разработанного W3C консорциумом Также следует отметить, что в Knowledge .NET используется терминология, принятая в онтологических системах (Concept-Property-Individual), вместо терминологии фреймовых систем (Frame-Slot-Instance) [8].

Для структурирования информационных ресурсов в базе знаний информационной системы для самообучения наиболее эффективным представляется подход инкапсулирования на основе механизма классов объектно-ориентированного программирования. Ниже опишем основные типы представлений знаний с точки зрения онтологического подхода к формированию элементов базы знаний и опишем варианты их использования в экспертной системе информационной поддержки самостоятельной работы студентов.

Для представления множества экземпляров одного типа в онтологической модели используется понятие концепта (concept). Аналогично, в объектно-ориентированной модели объекты описываются с помощью классов.

В качестве примера рассмотрим онтологию Электронно-образовательный ресурс (Knowledge resource). В данной онтологии мы можем предположить, что все электронно–образовательные ресурсы являются экземплярами некоторого концепта Электронный библиотечный каталог (Biblio Catalog). При описании сложных онтологий, обычно концепты организованы в виде иерархической структуры (концепт-подконцепт). Иерархическую структуру концептов иногда называют таксономией. В вершине дерева концептов находится концепт Thing. Другими словами, все концепты унаследованы от концепта Thing. Также вводится понятие концепта Nothing. Концепт Nothing является подконцептом любого концепта (т.е. унаследован от любого концепта).

Концепт может быть конкретизацией (унаследован от) одного или более других концептов. Если концепт A унаследован от концептов B и C, в программе на Knowledge .NET это записывается с помощью одной из следующих конструкций:

A is_subconcept_of B,C;

Самая короткая запись, может быть использована, если определяемая сущность (в данном случае концепт) задается с помощью только одного параметра (is_subconcept_of)

{ A is_subconcept_of B,C;}

Данная запись обычно используется, когда сущность определяется несколькими параметрами. В приведенном примере такая запись несколько избыточна.

```
A
{
    is_subconcept_of B;
is_subconcept_of C;
}
```

Если параметр (в данном примере is_subconcept_of) в качестве значения получает несколько сущностей (в данном примере: B и C), тогда данный параметр можно продублировать для каждой полученной сущности.

Дублирование параметра особенно удобно, если одна из сущностей является анонимным концептом.

Концепт может быть определен с помощью перечисления всех его экземпляров. Такие концепты называются Набором. Если перечисляемый в наборе экземпляр не определен в системе, Knowledge .NET создает экземпляр с данным именем на основе концепта Thing.

enumerated concept CONCEPT_IDEN is_one _of INDIVIDUAL1, INDIVIDUAL2, …;

CONCEPT_IDEN – идентификатор концепта

INDIVIDUAL1, INDIVIDUAL2 – идентификаторы экземпляров

Свойство представляет связь между двумя сущностями (во фреймовых системах свойства называются слотами). Ниже опишем следующие основные типы свойств:

- Простые свойства. Простое свойство связывает экземпляр со значением некоторого простого типа данных. Например, со строкой (string) или целым числом (int);
- Объектные свойства. Объектное свойство связывает между собой два экземпляра;
- Аннотации. Аннотации также относятся к свойствам. С помощью свойства Аннотация пользователь может добавить описание (метаданные) к концептам, экземплярам, объектным и простым свойствам.

Для каждого свойства должен быть определен домен (domain) и область значений(range).

Отметим, что домен и область значений может содержать более одного идентификатора. В этом случае идентификаторы перечисляются через запятую.

В используемой программной платформе Knowledge .NET свойства могут быть реализованы для задания ограничений. В онтологии каждому ограничению удовлетворяет ноль или более экземпляров. Таким образом, можно рассматривать ограничения как концепты. Поддерживаются три основных типа ограничений:

- ограничения по квантору;
- ограничения по мощности;
- ограничения по значению.

Ограничения по квантору состоят из трех компонентов:

- Свойство. Любое простое или объектное свойство, определенное в онтологии;
- Квантор. Допускается квантор существования (∃, some_values_from) и квантор всеобщности (∀, all_values_from);
- Наполнитель. Концепт или простой тип, возможно использование анонимного концепта, т.е. концепта определенного прямо в данном месте исходного кода[9].

Важным типом свойств являются Аннотации. С помощью свойства Аннотация пользователь может добавить описание (метаданные) к концептам, экземплярам, объектным и простым свойствам, правилам. Аннотация может содержать следующие поля:

- Информация о Версии (version_info). Описание версии сущности (концепта, экземпляра, свойства);
- Название (label) Развернутое название сущности на естественном языке;
- Комментарий (comment);
- Автор (created_by);
- Дата Создания (creation_date).

Все поля являются строками в формате Юникод. Ниже приведен синтаксис определения аннотации.

annotation

{

[*version_info "****SOME_TEXT****";]***

[*label "****SOME_TEXT****";]***

[*comment "****SOME_TEXT****";]***

[*created_by "****SOME_TEXT****";]***

[*creation_date "****SOME_TEXT****";]***

}

SOME_TEXT – *любой текст в формате Юникод;*

Для того, чтобы пользователи базы знаний имели возможность без труда находить в ней фрагменты и записи любой сложности и уровней детализации (верхний уровень – уровень пользователя – студента, нижний уровень – уровень инженера знаний - преподавателя) реализован механизм языка запросов.

Язык запросов позволяет выбирать из онтологии экземпляры по некоторому заданному критерию (запросу).

Следует отметить, что сами по себе запросы не являются онтологическими знаниями. Язык запросов скорее является инструментом, который позволяет осуществлять экспертную деятельность на заданной онтологии.

Язык запросов поддерживается с помощью класса QueryEngine определенного в библиотеке классов Knowledge.NET.

Ниже рассмотрим синтаксис запроса, запуск запроса и формат результата выполнения запроса.

Запрос позволяет инженеру знаний получить коллекцию экземпляров онтологии, которые удовлетворяют определенному критерию. Все запросы можно разделить на три основных типа:

- запросы, критерий которых основан на некоторой коллекции концептов и значениях их свойств;
- запросы, критерий которых основан только на значениях свойств без указания концептов;
- запросы, критерий которых содержит только перечисление концептов без указания свойств и их значений;

С точки зрения синтаксиса, запрос представляет собой текстовую строку (*string*) в следующем формате:

individuals of concepts ***CONCEPT1, CONCEPT2, CONCEPT3…***

where ***EXP1***

of concepts **CONCEPT1, CONCEPT2, CONCEPT3…** — задает множество концептов из которых будут выбираться экземпляры. Данная часть запроса не является обязательной (например, она не определяется в запросах второго типа).

Некоторые из концептов могут быть *Набором Экземпляров,* который определен с помощью подзапроса. Например:

individuals of concepts ***CONCEPT1, CONCEPT2,***

individuals of ***CONCEPT3, CONCEPT4…***

where ***EXP2***

where ***EXP1***

Where **EXP1 —** задает ограничения на свойства выбираемых концептов (критерий). Описание синтаксиса выражения **EXP1** смотрите ниже. Данная часть запроса не является обязательной (например, она не определяется в запросах третьего типа). Тем не менее, одна из вышеописанных частей запроса: "*of concepts*" или "*where*" должна быть обязательно задана в запросе.

Выполнение запросов происходит с помощью класса QueryEngine определенного в библиотеке классов Knowledge.NET. Для выполнения запроса, используется статический метод:

ICollection<Individual> QueryEngine.execute(string query)

Данный метод в качестве параметра получает запрос и возвращает коллекцию экземпляров, полученную в результате выполнения запроса. В случае если запрос содержит ошибку, метод execute() генерирует исключение QueryValidationException.

Так же для реализации полного функционала машины логического вывода необходимо описать правила информационной системы для самообучения.

Набор правил задает совокупность продукций используемых для логического вывода. Основные компоненты набора правил:

- локальные переменные объявленные в наборе правил в C# нотации;
- целевая переменная;
- правила составляющие набор.

Контекстом набора правил является онтология, в которой объявлен данный набор.

Набор правил имеет следующую структуру:

ruleset ***IDEN***

{ ***C#_CODE***

goal ***V;***
[rule ***RULE_IDEN***
{

if ***(B1)***
then ***S2;***
[else ***S3;]***

*[priority **INT_VALUE**;]*

}]+

}

Опишем составляющие правил логического вывода.

- IDEN — идентификатор набора правил
- C#_CODE — некоторый код на C#: локальные переменные и константы используемые в правилах вывода. Одна из этих переменных может быть помечена как целевая (goal) переменная. Целевая переменная может быть переопределена при вызове метода consult().
- RULE_IDEN — имя правила;
- B1 — условие правила, представляется в виде выражения принимающего значения истинна (true) или ложь (false); S2 — заключение (код C# исполняемый в случае истинности условия);
- S3 — код C# исполняемый в случае если выражение B1 ложно;
- priority INT_VALUE — позволяет определить приоритет правила, данный параметр может использоваться машиной логического вывода для разрешения конфликтов (в случае если на каком-либо шаге вывода несколько правил имеют истинное значение условия).

Библиотека классов Knowledge. NET предоставляет доступ к наборам правил через специальные программные элементы и интерфейсы работы с базой знаний что позволило реализовать свою собственную машину вывода. На рисунке 2 показана диаграмма классов с базовым набором компонентов ядра экспертной системы [10].

Также библиотека содержит простую машину логического вывода реализованную в виде класса ProductionSystem. Данный класс предоставляет следующие статические свойства и методы:

- public static bool showAnnotations – данное свойство задает, будут ли показываться аннотации определенные в правилах в ходе консультации или нет;
- public static void consult (string ruleset);
- public static void consult (string ruleset, string goal);
- public static void consult (string ruleset, string goal, ChainingMehod method);

Данные методы начинают консультацию посредствам вызова заданных наборов правил.

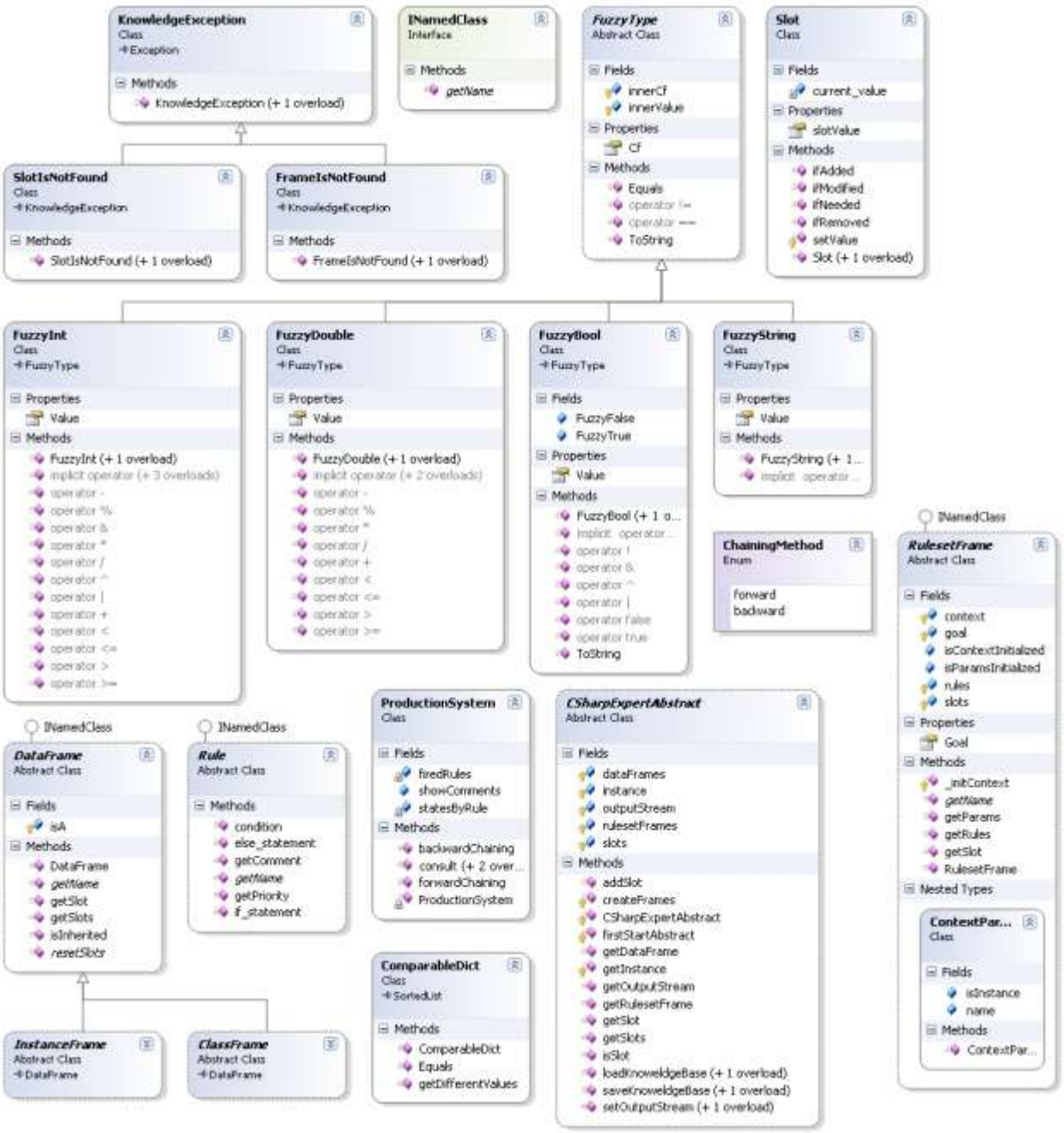

Рис. 2. Диаграмма классов с базовым набором компонентов ядра информационной системы для самообучения

Для использования машины логического вывода выделим основные параметры функций:

- ruleset – имя набора правил, используя который будет осуществляться консультация;
- goal – некоторая локальная переменная из контекста набора правил. Данный параметр не обязателен, если цель не определена, выводиться цель, определенная в наборе правил;

- method – стратегия логического вывода, может быть прямой или обратной (По умолчанию используется прямой вывод).

Метод разрешения конфликтов, используемый машиной вывода, заключается в выборе правила, имя которого будет первым исходя из лексикографического порядка.

Таким образом, можно реализовать следующий список понятий базы знаний и машины логического вывода информационной системы для самообучения:

- Фрейм-класс – ClassFrame;
- Фрейм-экземпляр – InstanceFrame;
- Набор правил – RulesetFrame;
- Слот – Slot;
- Правило – Rule;
- Концепт – Concept;
- Экземпляр – Individual;
- Свойство – Property

В правой части описанной выше записи описаны имплементации основных понятий в библиотеке Knowledge.Core.

Таким образом, для решения задач информационной системы для самообучения, может быть решена задача оптимизации специальной библиотеки – расширение (Add-in) Knowledge.NET.

Рассмотрим на уровне программирования основных компонентов информационной системы для самообучения вопросы оптимизации базовых программных элементов языка C#. При конвертации программы из Knowledge.NET в C#, метакомпилятор создает класс Knowledge, унаследованный от класса KnowledgeAbstract. Разработчик экспертной системы, в конечном счете, оперирует набором статических атрибутов и методов, которые предоставляют интерфейс для доступа к фреймам, концептам и наборам правил. Данный подход позволил сохранять и загружать данные в/из базы данных любого общепринятого стандарта (ODBC, OLE DB и т.д.).

Для реализации алгоритмов логического вывода фрагментов образовательного контента необходимо представить его в виде набора деревьев, имеющих перекрёстные ссылки, что отражало не только иерархичность структуры обучающего материала, но и различного рода ссылки, создающие первичные, вторичные и другие структуры учебного

материала, отражающие взаимосвязи различных учебных целей, задач компетенций и управляющих воздействий.

В зависимости от типа модели самообучающегося и его индивидуальных подходов к обучению (в общем виде подходы могут быть индуктивный, дедуктивный и гибридным) предлагается использовать три вектора обучения (быстрый, нормальный и медленный). На рисунке ниже (рисунок 3) изображены линейные стратегии обучения, соответствующие процессам освоения образовательной единицы в соответствии с эталонной моделью учебной программы дисциплины.

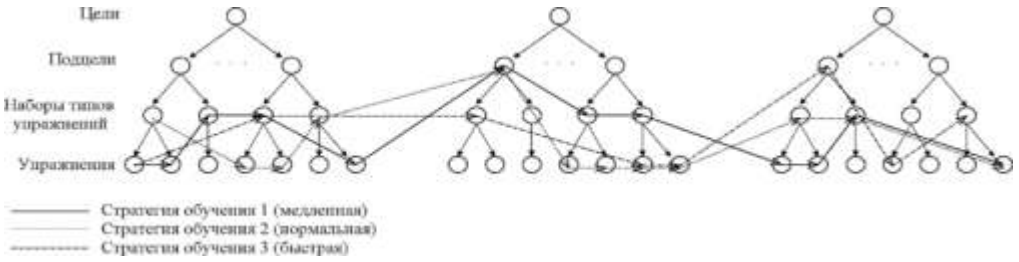

Рис. 3. Линейные стратегии обучения

В процессе формирования алгоритмов логического вывода информационной системы для самообучения необходимо реализовывать возможности выбора стратегий обучения, в качестве анализа способностей к той или иной стратегии система должна предложить вариант повторно упражнения того же типа. В случае допущения самообучающимся ошибок локального характера, необходимо вернуть пользователя к ранее пройденному материалу. Используя подготовленную преподавателем (инженером базы знаний экспертной системы) систему оценочных шкал уровня освоения предметной области, поэтапно проконсультировать пользователя по способам их ликвидирования. В случае большого количества разнородных ошибок или изменения качества ошибок система должна отследить траекторию прохождения узлов графа самообучаемым, реализующего эталонные алгоритмы обучения, зафиксировать весовые коэффициенты и ссылки на фрагменты перекрестного (смежного) контента, для последующей генерации нового алгоритма поэтапного логического вывода и представления образовательного контента.

Для реализации интерактивного самообучения учащегося в режиме on-line необходимо выбрать соответствующий алгоритм и программную реализацию формирования правил логического вывода хранящегося в рабочей области образовательного контента в момент консультации с

экспертной системой. Правила экспертной системы – это продукции вида: «Если _условие_, то_действие».

Механизм логического вывода необходимо реализовать при помощи отдельного модуля «РЕШАТЕЛЯ» информационной системы для самообучения, который должен поддерживать следующие возможности:

- поддержка прямой и обратной (дедуктивной, индуктивной) стратегии вывода на основе «просмотра» узлов графа учебной дисциплины;
- поиск решения «в глубину» (просмотр фрагментов базы знаний формальных XML - структур электронно-образовательных ресурсов);
- поиск и разрешение конфликтов правил на основе интервьюирования преподавателя и верификации имеющихся правил;
- сохранение состояний и типов фрагментов и типов метазнаний (на основе модели Dublin Core) и правил (активное, неактивное) в рабочей памяти экспертной системы.

Процедуры «РЕШАТЕЛЯ» информационной системы для самообучения можно представить с помощью системы общепринятых процессов в виде:

$$I = < V, S, K, W >, \quad (2.2)$$

Где V – процесс выбора, осуществляющий выбор из P и R подмножества активных продукций и подмножества активных данных; S – процесс сопоставления, определяющий множество пар: правило p_i данные$\{d_j\}$, где $p_i \in P_v$ $\{d_j\} \subset R_v$, причем каждое p_i применимо к элементам множества $\{d_j\}$; K – процесс разрешения конфликтов (или процесс планирования), определяющий, какой из идентификаторов будет выполняться; W – процесс, осуществляющий выполнение выбранного идентифицированного правила (то есть выполнение действий, указанных в правой части правила). Результатом выполнения является модификация данных в R или операция ввода/вывода. Механизм вывода должен реализовываться не основе семантических и синтаксических методах выборки фрагментов данных из базы знаний. Данный подход позволяет без труда интегрировать подобные правила (метоправила) непосредственно в «РЕШАТЕЛЬ», так как они совершенно не зависят от рассматриваемой предметной области и способствуют декомпозиции сложных фрагментов метаинформации на подзадачи и методы их использования для генерирования исходного образовательного контента.

Ниже рассмотрим алгоритм реализации начальной выборки. Данный подход производиться на основе имеющихся списка целей, что позволяет сократить предметную область выборки и логического вывода предварительной информации и способствует более детальному понимания

рекомендаций экспертной системы пользователю. В случае если набор целей существует – используется режим обратного вывода (от более детальных целей), в случае, если список целей отсутствует – используется метод прямого ввода (например, текста запроса).

В процессе тестирования данных алгоритмов было выявлено, что число конфликтов прямо пропорционально степени неточности искомой информации. Для разрешения данных противоречий и конфликтов правил в базе знаний экспертной системы используется нотация метаописания правил (метаправила). Ниже приведем их формальное описание:

$$
\begin{aligned}
&\textit{если} P(r_i) > P(r_2) \textit{то} K(r_i) > K(r_2) \textit{иначе} \\
&\textit{если} N(r_i) < N(r_2) \textit{то} K(r_i) > K(r_2) \textit{иначе} \\
&\textit{если} D(r_i) > D(r_2) \textit{то} K(r_1) > K(r_2) \textit{иначе} K(r_1) <= K(r_2) \\
&r_1, r_2 \in P, N,
\end{aligned}
$$

Где N – количество не идентифицированных атрибутов в правилах, D – количество не идентифицированных атрибутов в действии правил, R – набор всех активных правил.

Таким образом, для начальной выборки необходимо выполнить расчет приоритетов правил, который выше для правил, имеющих меньшее число не идентифицированных атрибутов, большее число не идентифицированных атрибутов в действии и высокую вероятность появления. Формирование подобной выборки зависит от того, какая стратегия вывода используется в данном рабочем цикле.

Естественно, для корректного логического вывода необходимо применять алгоритмы сопоставления правил начальной выборки и метаправил в рабочей области активного цикла. Данный алгоритм осуществлен в виде рекурсивной функции, описанной ниже:

$$F(R, A, O, K), \quad (2.4)$$

Где R – набор активных правил,A – список текущих целевых атрибутов (параметры запроса), O – список идентифицированных атрибутов, K – список идентифицированных правил. Ниже рассмотрим алгоритм работы рекурсивной функции сопоставления правил:

1. правила начальной выборки заново сопоставляются с набором атрибутов в рабочей памяти – R в порядке приоритета;
2. в процессе выполнения одной копии функции F производиться ее повторный запуск для внесения вновь внесенных изменений в базу правил.

Таким образом, алгоритм разрешения конфликтов в правилах основан на процедуре расчета приоритета, который выше для тех, которые имеют большее количество идентифицированных экспертом (преподавателем) атрибутов правил.

Ниже приведем общий вид правил адаптивного логического вывода информационной системы для самообучения:

Если {(событие1, тип кривой 1, интервал 1)и

(событие2, тип кривой 2, интервал 2)и…}

то {действие}

Троек (событие, тип кривой, интервал) может быть от 1 до 10.

Каждому событию (в протоколе активности пользователя) в процессе обучения ставиться в соответствие кривая определенного типа, заданная на некотором интервале.

Примеры событий (либо учебно-тематические заданий - УТЗ):

- частота пользования словарем;
- ошибки при выполнении упражнений;
- превышение временных рамок, отведенных на выполнение УТЗ;
- время обдумывания (ожидания, либо задержки) ответа на вопрос.

В процессе пользования объектами учебного назначения пользователь за счет протоколирования накапливает собственный стиль и траекторию изучения их фрагментов. К примеру, изучая раздел или тему курса лекций, пользователь периодически обращается к глоссарию понятий, фактов и терминов, в данном случае целесообразно во всем интервале времени собирать статистику пользования данным типом контента для её дальнейшего анализа и выработки решений экспертной системы. В данном случае речь идет об аналоговых величин и отношениях. Некоторые виды кривых пользования глоссария учебного объекта (Рисунок 3) (словаря понятий – фактов дисциплины), интерпретирующих динамику событий обучаемого, которые можно использовать в качестве эталонных моделей траекторий освоения учебного материала.

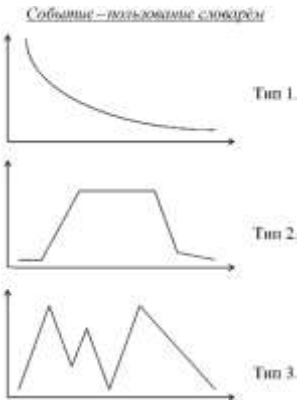

Рис. 3. Примеры типов кривых ожидания событий

На кривых выше по вертикали отложена частота использования словаря, по горизонтали время (время урока, время изучения темы и т.д.)

Главной целью разработки алгоритмов обучения для рассматриваемой информационной системы была задача проектирования такой системы логических выводов, которая моделировала бы всех участников образовательного процесса (учителя, ученика, учебный материал) и организовывала оптимальное их взаимодействие.

Здесь заложены алгоритмы формирования моделей обучаемого и преподавателя, введён определённым образом организованный учебный материал (формальная XML-структура) с элементами мультимедиа. На этой основе имитируется процесс реального обучения с учётом таких характерных его особенностей, как взаимная интеграция процессов верификаций моделей обучаемого, преподавателя и учебного курса, способности ученика, оптимальность стратегии дозировки знаний и упражнений учителем, скорость запоминания и забывания знаний учеником, продолжительность и устойчивость его активного состояния и т. п.

Самым важным моментом реализации алгоритма обучения является функция объяснения экспертной системы, которая должна быть основана на интеграции отдельно взятых графах (траектории обучения) и дерева решений.

Синтез индивидуальных сервисов и интеграция информационной системы для самообучения с информационной средой вуза, имеющей специфичный информационный потенциал, позволит предоставить студенту адекватный набор информационных ресурсов, необходимых для его учебной и самостоятельной деятельности. Другими словами, построение архитектуры экспертной системы, которая соответствует потребностям студентов в выделении соответствующих информационных ресурсов из внешних и внутренних источников информации, позволит предоставить современные возможности для их самообучения.

Литература

1. Библиотечно-библиографическая классификация [Электронный ресурс] / Российская Государственная Библиотека. – Электрон. дан. – Режим доступа: http://www.rsl.ru/index.php?f=30, свободный. – Загл. с экрана.

2. Будапештская Инициатива «Открытый Доступ» [Электронный ресурс] / Институт Открытое общество. – Электрон. статья. – Режим доступа: http://www.soros.org/openaccess/ru/index.shtml, свободный. – Загл. с экрана.

3. Интернет-портал по грид-технологиям [Электронный ресурс]. – Электрон. дан. – Режим доступа: http://www.gridclub.ru, свободный. – Загл. с экрана.

4. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления : ГОСТ 7.82-2001. – Введ. 2001-05-22. – М., 2001. – (Система стандартов по информации, библиотечному и издательскому делу). – http://law.edu.ru/norm/norm.asp?normID=1167282.

5. Ваграменко Я.А., Фанышев Р.Г. Технология интеллектуального анализа текстовой информации в базах знаний образовательной экспертной системы // Педагогическая информатика. – 2011. - № 1. – С. 57-62.

6. Гадасин В. А., Конявский В. А. От документа – к электронному документу. Системные основы [Электронный ресурс] / Документ.RU. – Электрон. статья. – Режим доступа: http://document.ru/readingroom/edoc/1/index.asp?id=0, свободный. – Загл. с экрана.

7. Гребнев А. Н. AtLeap – Java каркас с открытым исходным кодом для Web-приложений // Вестник ИжГТУ : период. науч.-теор. журн. – Ижевск : Изд-во ИжГТУ, 2006. – №1. – 116 с. – С. 64-68.

8. Электронные издания. Основные виды и выходные сведения : ГОСТ 7.83-2001. – Введ. 2001-11-02. – М., 2001. – (Система стандартов по информации, библиотечному и издательскому делу). – http://law.edu.ru/norm/norm.asp?normID=1167295.

9. Сафонов В.О. Экспертные системы – интеллектуальные помощники специалистов. – СПб.: «Знание», 1992.

10. Сафонов В.О. и др. Язык представления знаний Турбо-Эксперт. – Кибернетика, 1991, N 5.

Российский портал информатизации образования содержит: законодательные и нормативные правовые акты государственного регулирования информатизации образования, федеральные и региональные программы информатизации сферы образования, понятийный аппарат информатизации образования, библиографию по проблемам информатизации образования, по учебникам дисциплин цикла Информатика, научно-популярные, документальные видео материалы и фильмы, периодические издания по информатизации образования и многое другое.