St. Petersburg State University

Faculty of Mathematics and Mechanics
Chair of Software Engineering

# Everyday Intelligent Advisor (EIA):

# Expert system based on Knowledge.Net

Graduate paper by a student of group 461

*Sogomonyan Marina Mamikonovna*

Scientific advisor
(Professor of computer science)

/ Signature /_____/Safonov Vladimir Olegovich

Reviewer
(Post-graduate student)

/ Signature /_____/ Ermakov Gennady

« Admitted to proof»
(Head of the chair, Dr. of Phys. and Math. Sci.)

/Signature/_____/Terekhov Andrey Nikolayevich

St.Petersburg, 2010

Санкт-Петербургский Государственный Университет

Математико-Механический Факультет
Кафедра системного программирования

# Everyday Intelligent Advisor (EIA):

# экспертная система на базе Knowledge.Net

Дипломная работа студентки 461 группы

*Согомонян Марины Мамиконовны*

Научный руководитель
(профессор, доктор технических наук)

/подпись/_____/ Сафонов Владимир Олегович

Рецензент
(аспирант)

/подпись/_____/Геннадий Александрович Ермаков

«Допустить к защите»
(Заведующий кафедрой, доктор физико-математических наук)

/подпись/_____/ Терехов Андрей Николаевич

Санкт-Петербург
2010г

# Оглавление

# Введение

Начиная с середины XX столетия и до сих пор, одним из самых значимых направлений развития науки и техники является искусственный интеллект. Задача этого направления заключается в обеспечении разумных рассуждений и действий с помощью вычислительных систем и других искусственных устройств.

Отдельная область искусственного интеллекта - инженерия знаний – занимается разработкой экспертных систем, развитием средств представления и обработки знаний на компьютере.

**Экспертная система** - система искусственного интеллекта, содержащая знания об определенной слабо структурированной и трудно формализуемой узкой предметной области и способная предлагать пользователю разумные решения и их объяснения. Экспертные системы играют роль эксперта и решают задачи диагностики, прогнозирования или планирования.

Рассмотрим примеры наиболее известных экспертных систем, с которых началось создание и развитие этого типа программных средств:

- **MYCIN** - это экспертная система, разработанная в начале 1970х годов в Стэндфордском университете. Система предназначена для диагностики и лечения медицинских инфекций. Исходя из представленных пациентом симптомов, система ставит диагноз и рекомендует курс соответствующего медикаментозного лечения. [3]
- **PROSPECTOR** - экспертная система, применяемая при поиске месторождений полезных ископаемых. Система делает выводы на основе геологических анализов. [4]
- **DENDRAL** - это старейшая экспертная система в мире. Экспертная система автоматизировала процесс определения химической структуры вещества и определяла строение органических молекул по химиче-

ским формулам и спектрографическим данным о химических связях в молекулах. [5]

Мы видим, что экспертные системы имеют дело с предметами реального мира и выполняют операции, которые обычно требуют наличия значительного опыта, накопленного человеком. Обычно для создания сложной экспертной системы, требуются знания и практические навыки даже нескольких специалистов. Во-первых, для этого нужен эксперт - человек, умеющий находить  решения проблем в конкретной области. Во-вторых, нужен инженер знаний - человек, который знает каким образом построить экспертную систему, как структурировать и организовать знания эксперта. Ну и наконец, для построения экспертной системы необходим программист, который будет разрабатывать саму программу, используя специальные инструментальные средства или создавая собственные.

Раньше на проектирование и создание одной экспертной системы потребовалось бы около 20 человек-лет. В настоящее время имеется ряд инструментальных средств, значительно ускоряющих разработку.

**Инструментальное средство разработки экспертных систем** – это язык программирования, используемый разработчиком для построения экспертной системы.

Сегодня для создания экспертных систем существуют различные программные средства. Одно из них — система управления знаниями Knowledge.Net, разработанная лабораторией профессора В.О.Сафонова (СПбГУ). Система представляет собой удобный инструмент для построения экспертных систем и их последующего редактирования. [2]

Целью данной дипломной работы является разработка экспертной системы Everyday IIntelligent Advisor (Советчик в повседневной деятельности) на базе Knowledge.Net. Система помогает пользователю в решении типичных каждодневных задач, таких как выбор оптимального маршрута, поиск подходящего магазина или ресторана.

Также, поскольку система управления знаниями Knowledge.Net на данный момент не имеет широкого распространения, советчик призван стать полнофункциональным демонстрационным примером возможностей системы.

## Постановка задачи

Итак, если обобщить задачи, которые стояли перед нами в ходе дипломной работы, получим следующее:

1. Изучить систему управления знаниями Knowledge.Net
2. Сравнить Knowledge.Net с другими подобными системами
3. Реализовать экспертную систему Everyday Intelligent Advisor в качестве апробации системы Knowledge.Net

# Экспертные системы (ЭС)

## Основные понятия экспертных систем

Экспертные системы бывают двух типов: статические и динамические. Статические экспертные системы не изменяются с течением времени, а динамические изменяются, приспосабливаясь к изменениям внешнего мира. Например, динамические экспертные системы могут использовать показания каких-либо датчиков или заимствовать дополнительную информацию из сети Internet.

Поскольку данная дипломная работа посвящена созданию экспертной системы статического типа, то на них мы и остановимся подробнее. Рассмотрим структуру статической экспертной системы (рис.1)


**Рисунок 1. Структура экспертной системы.**

Структура экспертной системы содержит следующие объекты:

- база знаний
- рабочая память
- механизм логического вывода
- подсистема диалога
- подсистема приобретения и пополнения знаний
- подсистема объяснения

Теперь остановимся на каждом из этих понятий подробнее.

**База знаний** — это одно из ключевых понятий в области экспертных систем, означающее совокупность знаний (фактов и правил) о предметной области, обычно на некотором языке, близком к естественному. Под правилами в базе знаний понимаются правила вывода, описывающие отношения между фактами. Факты же имеют классическое значение и представляют собой знания в форме утверждений.

Помимо правил и фактов в базу знаний может входить некоторая процедурная часть, выполняющая расчетные, оптимизационные или другие нужные алгоритмы.

**Рабочая память** предназначена для временного хранения исходных данных и промежуточных фактов (полученных от пользователя) решаемой в текущий момент задачи. Как правило, рабочая память размещается в оперативной памяти ЭВМ.

**Механизм логического вывода** предназначен для получения новых фактов на основе сопоставления исходных данных из рабочей памяти и знаний из базы знаний. Механизм логического вывода во всей структуре экспертной системы занимает наиболее важное место, т.к. именно он моделирует ход рассуждений эксперта.

**Подсистема диалога** отвечает за организацию интерфейса для общения с пользователем в процессе решения задачи и получения результата. Именно этот элемент экспертной системы в дальнейшем будет называться пользовательским интерфейсом.

**Подсистема приобретения и пополнения знаний** автоматизирует процесс наполнения экспертной системы знаниями, осуществляемый пользователем-экспертом, и адаптации базы знаний системы к условиям ее функционирования. Адаптация экспертной системы к изменениям в предметной области реализуется путем замены правил или фактов в базе знаний.

**Подсистема объяснения** позволяет пользователю понять, каким образом система пришла к итоговому решению задачи или почему решение не было найдено. Это повышает доверие пользователя к полученному результату и упрощает разработчику тестирование программы. Возможность объяснять свои действия является одним из самых важных свойств экспертной системы, потому что еще больше приближает ее работу к работе эксперта в данной области.

## Классификация экспертных систем по решаемой задаче

Чтобы лучше понимать принципы работы экспертных систем, рассмотрим классы задач, которые можно решать с их помощью:

- **Интерпретация**

  Согласование полученных извне данных и получение исходя из них ответа на общий вопрос.
- **Диагностирование**

  Проведение различных диагностик (например, определение неисправностей в каком-либо механизме).
- **Мониторинг**

  Непрерывное отслеживание данных и сообщение о выходе тех или иных параметров за допустимые пределы. При решении задач мони-

торинга с помощью экспертной системы, существует риск того, что система будет подавать ложные сигналы, или наоборот, пропускать плохие ситуации. Это связано с тем, что границы допустимого обычно размыты.

- **Проектирование**

  Задача проектирования заключается в том, чтобы подготовить необходимый пакет документов для создания объекта, с заранее заданными
  свойствами.

- **Прогнозирование**

  По полученному набору данных система должна предугадать дальнейшее развитие событий.

- **Обучение**

  Система обучения наблюдает за процессом изучения какой-либо дисциплины пользователем-учеником и, принимая во внимание его ошибки, подсказывает правильные решения и дает советы.

- **Планирование**

  Экспертная система составляет план действия для объекта, выполняющего определенные функции.

Таким образом, мы видим, что экспертные системы могут решать довольно широкий круг задач. Это тем более оправдывает их использование в различных областях.

## Достоинства экспертных систем

Конечно не всегда машина, обладающая искусственным интеллектом, может заменить человека-эксперта в своей области. Но в ряде случаев, применение экспертной системы обладает заметными преимуществами перед использованием знаний эксперта. Итак, преимущества экспертных систем:

- **Постоянство**

Человеку свойственно забывать. Поэтому любой перерыв в профессиональной деятельности эксперта может отразиться на полноте его знаний. В случае с экспертной системой, это исключено, программа может годами хранить помещенные в нее знания в их первоначальном виде.

- **Решение трудоемких задач**

Экспертная система удерживает в памяти многочисленные факты, правила и условия задачи. Это позволяет оперативно выполнять действия с данными и быстро приходить к какому-либо выводу.

- **Простота передачи**

Передача знаний от одного человека другому – длительный, трудоемкий и часто недешевый процесс. Гораздо проще передать программу, которая будет заменять эксперта своими действиями.

- **Стабильность результатов**

Экспертные системы устойчивы к «помехам». Человек же легко поддается влиянию внешних факторов, которые непосредственно не связаны с решаемой задачей. Эксперт-человек может принимать разные решения в одинаковых ситуациях из-за эмоциональных факторов.

- **Способность к нечетким рассуждениям**

Экспертные системы способны воспринимать нечеткие данные и проводить по ним рассуждения. К таким системам относятся системы планирования и прогнозирования, опирающиеся на неполную информацию.

- **Стоимость**

Высококвалифицированные специалисты обходятся очень дорого. Экспертные системы же хоть и отличаются дороговизной разработки, но не требуют больших затрат при эксплуатации.

Вместе с тем разработка экспертной системы не позволяет полностью отказаться от эксперта-человека. Хотя экспертная система хорошо справляется со своей работой, тем не менее, в определённых областях человеческая

компетенция явно превосходит искусственную. Однако и в этих случаях экспертная система может позволить отказаться от услуг высококвалифицированного эксперта, оставив эксперта средней квалификации, используя при этом экспертную систему для усиления и расширения его профессиональных возможностей.

## Средства реализации ЭС

Есть несколько категорий инструментальных средств реализации экспертных систем. Все они имеют свои плюсы и минусы. Для того чтобы определить, какой метод лучше подходит для разработки нами экспертной системы, рассмотрим каждую категорию подробнее.

### Оболочки экспертных систем

Оболочки экспертных систем представляют собой «пустые» экспертные системы. Как правило, это хорошо зарекомендовавшие себя на практике системы, из которых удалены все специфичные для области непосредственного применения компоненты. Примером такой оболочки может служить система EMYCIN (Empty MYCIN - пустой MYCIN) [6], созданная на основе прошедшей длительную обработку системы MYCIN. В EMYCIN сохранен интерпретатор и все базовые структуры данных – таблицы знаний и связанные с ними механизм индексации. Оболочка дополнена специальным языком, улучшающим читабельность программ, и средствами поддержки библиотеки типовых случаев и заключений, выполненных по ним экспертной системой.

Преимущество оболочек экспертных систем перед другими средствами реализации заключается в том, что создать систему с их помощью может сам эксперт проблемной области, не прибегая к помощи программистов при этом. Достаточно будет просто заполнить базу знаний. Однако данный метод создания экспертных систем становится слишком трудоемким, если предметная область плохо укладывается в модель, используемую в готовой оболочке.

### Языки программирования высокого уровня

Инструментальные средства этой категории избавляют разработчика от необходимости углубляться в детали реализации системы – способы эффективного распределения памяти, низкоуровневые процедуры доступа и манипулирования данными. Одним из наиболее известных представителей таких языков является OPS5 [7]. Этот язык хорош тем, что прост в изучении и предоставляет программисту гораздо более широкие возможности, чем типичные специализированные оболочки. Однако большинство подобных языков так и не было доведено до уровня коммерческого продукта и представляет собой скорее инструмент для исследователей.

### Среда программирования, поддерживающая несколько парадигм

Средства этой категории включают несколько программных модулей, что позволяет пользователю комбинировать в процессе разработки экспертной системы разные стили программирования и получать в итоге более полноценный продукт. Среди первых проектов такого рода была исследовательская программа LOOP, которая допускала использование двух типов представления знаний: базирующегося на системе правил и объектно-ориентированного. На основе этой архитектуры во второй половине 1980-х годов было разработано несколько коммерческих программных продуктов, из которых наибольшую известность получили KEE, KnowledgeCraft и ART. Эти программы предоставляют в распоряжение квалифицированного пользователя множество опций и для последующих разработок, таких как КАРРА и CLIPS [10], и стали своего рода стандартом. Однако освоить эти языки программистам далеко не так просто, как языки, отнесенные к предыдущей категории.

### Дополнительные модули

Средства этой категории представляют собой автономные программные модули, предназначенные для выполнения специфических задач в рамках выбранной архитектуры системы решения проблем.

## Выбор средства реализации

Многие знания, которые необходимые для решения трудных практических задач, носят гибридный характер. С одной стороны, для создания полноценной экспертной системы требуются концептуальные (определение концепций предметной области и связей между ними), фактуальные (факты и их связи между собой) и эвристические знания (правила, с помощью которых происходит процесс рассуждения). С другой стороны, необходимы процедурные знания, такие как алгоритмы.

Становится очевидным, что наиболее универсальным и удобным инструментальным средством для разработки экспертной системы является среда программирования, поддерживающая несколько парадигм. Поэтому для решения задачи в рамках дипломной работы была выбрана система Knowledge.Net, разрабатываемая лабораторией профессора В.О.Сафонова (СПбГУ) с 1980-1990х годов. [8]

# Система Knowledge.NET

До 1980 года не существовало средств реализации экспертных систем, которые содержали бы удобные и эффективные средства традиционного программирования. В то же время, ни один инструмент из области традиционного программирования не позволял адекватно представлять и обрабатывать знания.

По этой причине, в Санкт-Петербургском Государственном Университете, в лаборатории проф. В.О. Сафонова (до 2001 г. – лаборатории технологии программирования и экспертных систем, с 2001 г. – лаборатории Java-технологии НИИ математики и механики), начиная с 1980-х – 1990х гг., разрабатываются методы интеграции инженерии знаний и инженерии программ.

Одной из основных разработок лаборатории является система представления знаний Knowledge.Net, на базе платформы Microsoft.Net. Данная система предназначена для разработки и использования библиотек (баз) знаний из разнообразных предметных областей, которые могут проектироваться и реализовываться непосредственно на языке Knowledge.Net, а также могут автоматически извлекаться из сети Internet.

Система управления знаниями Knowledge.NET реализована как расширение (add-in) одной из наиболее современных интегрированных сред программирования - Microsoft Visual Studio.NET 2005. Эта архитектурная черта Knowledge.NET позволяет применять при разработке баз знаний широкий спектр возможностей проектирования, разработки и отладки программ, предоставляемых средой Microsoft Visual Studio.NET.

Методы интеграции инженерии знаний и инженерии программ в системе Knowledge.Net реализованы в языке C# Expert, который является расширением языка C# средствами представления фреймовых и продукционных знаний.

## Архитектура системы Knowledge.Net

Как уже было сказано, архитектура системы базируется на платформе Microsoft.Net, которая и оказала огромное влияние на устройство системы управлений знаниями.

Система Knowledge.NET состоит из следующих основных программных компонент:

- конвертора с языка Knowledge.NET в базовый язык C# платформы Microsoft.NET (последующая компиляция программ на C#, полученных в результате конвертирования, обеспечивается штатными средствами интегрированной среды Visual Studio.NET);

- редактора и визуализатора знаний Knowledge Editor, обеспечивающего визуализацию, проектирование, реализацию и модификацию в интерактивном режиме исходных текстов баз знаний на языке Knowledge.NET;

- системы Knowledge Prospector извлечения знаний в формате Knowledge.NET из текстов на естественном языке (для извлечения знаний из Интернета);

В системе также предусмотрено конвертирование полученных, сформулированных на языке Knowledge.Net знаний в общеупотребительный формат представления знаний – KIF (Knowledge Interchange Format). [9]

## Язык Knowledge.Net

Языком Knowledge.Net является C# Expert [1]. Поскольку базовым языком C# Expert выбран объектно-ориентированный язык C#, то при создании экспертной системы можно использовать не только средства управления знаниями, но и использовать объектно-ориентированный подход для реализации дополнительного функционала. При этом можно использовать все существующие библиотеки, разработанные под платформу .NET. Таким

образом, исходная программа может содержать как код специфичный для экспертных систем, так и простой код на языке C#.

В качестве прототипа языка C# Expert был выбран язык Турбо Эксперт,
разработанный проф. Сафоновым В.О. Также как в Турбо Эксперт, в C# Expert есть несколько способов представления знаний:

- фреймы (frames)

  Фреймы – это структурные элементы, которые служат для представления в базе знаний различных объектов и описания их иерархии. Фрейм может содержать слоты и атрибуты.
- слоты (instance slot, own slot)

  Слоты задают свойства фреймов. Слоты могут выражаться простыми и сложными типами, а также делегатами.
- правила

  Наборы правил (Rule frames) осуществляют логический вывод. В C# Expert наборы правил рассматриваются как разновидность фрейма.

Основные возможности языка представления знаний C# Expert:

- возможность взаимодействия с крупными моделями алгоритмического знания – программами и компонентами, написанными для платформы .NET
- возможность представления нечетких знаний
- возможность описания и использования наборов правил
- возможность интеграции с графическим интерфейсом

## Основные идеи Everyday Intelligent Advisor

В качестве апробации системы управления знаниями, мною была создана экспертная система Everyday Intelligent Advisor (EIA) – советчик в повседневной деятельности. Советчик призван помогать пользователю в решении задач, которые являются нетрудными, но отнимают много времени, если не планировать их осуществление заранее.

К задачам такого рода относятся выбор подходящего магазина, ресторана или маршрута для прогулки. Для того чтобы определить, какое именно заведение того или иного типа необходимо посетить, часто требуется помощь другого человека или доступ к сети Internet и свободное время в запасе. Именно поэтому было решено создать подобную экспертную систему.

Everyday Intelligent Advisor решает следующие задачи:

1. выбор магазина
2. выбор ресторана
3. выбор маршрута для прогулки
4. выбор фильмов для просмотра

При решении данных задач, система EIA старается учесть все пожелания пользователя и помочь ему максимально быстро. Для каждого предлагаемого решения EIA имеет обоснование, предназначенное для любознательного пользователя. Также система дает полезные советы, которые могут пригодиться в различных ситуациях.

На данный момент, программа может предлагать решения только жителям и гостям города Санкт-Петербург. Это обусловлено тем, что база знаний охватывает только объекты этого города. Не исключено, что в будущем, база знаний системы EIA будет расширена.

# Реализация EIA

## Методика построения системы EIA

При разработке экспертной системы EIA использовалась модель быстрого прототипирования (RAD). Это означает, что сначала был создан прототип экспертной системы, который мог решать узкий круг задач, а затем, уже после его отладки – сама экспертная система.

Существует определенная методика разработки экспертных систем. В данной дипломной работе она также использовалась. Рассмотрим основные этапы проектирования экспертной системы, которые были пройдены при разработке Everyday Intelligent Advisor:

- идентификация
- концептуализация
- формализация
- выполнение
- тестирование
- опытная эксплуатация

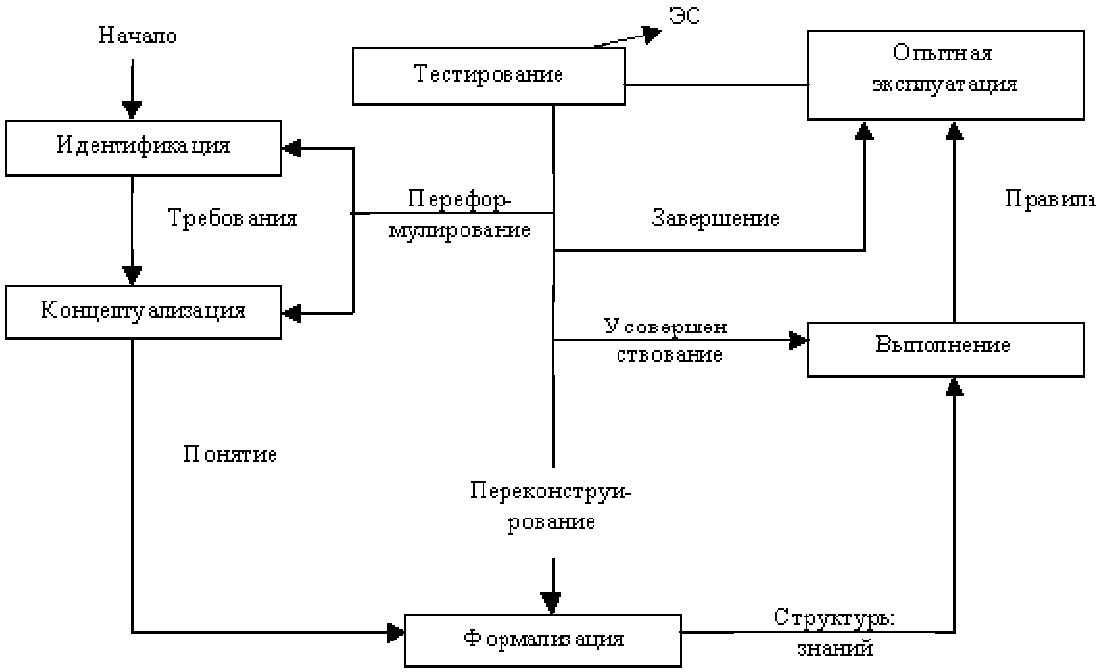


**Рисунок 2. Методика разработки ЭС.**

## Этап идентификации

Для того чтобы начать работу над проектом, необходимо прежде всего осознать какие цели стоят перед нами и каким образом их достигнуть. Идентификация задачи заключается в составлении ее неформального описания. При этом выделяются подзадачи, ключевые понятия и их входные и выходные данные поставленной перед нами задачи. Также на этом этапе должен быть определен предположительный способ решения задачи.

Этап идентификации экспертной системы EIA мы уже описали в разделе «Основные понятия EIA», поэтому перейдем сразу к следующему шагу разработки экспертной системы – этапу концептуализации.

## Этап концептуализации

На данном этапе проводится содержательный анализ проблемной области. Для каждой подзадачи выявляются используемые понятия, взаимосвязи между ними, определяются методы решения задач и состав используемых знаний.

При реализации системы EIA, для построения  модели предметной области, включающей основные понятия, их взаимосвязи с семантические отношения, использовался структурный подход.

На данном этапе, оказалась очень полезной система создания карт (map) comapping [11]. Она позволяет легко строить иерархию объектов, а также наглядно прослеживать пути от корня (начала программы) к листу полученного дерева (выводу). Каждый путь в дереве представляет собой цепочку рассуждений, которая и приводит нас к тому или иному решению.

На рисунке Зизображена модель экспертной системы Everyday Intelligent Advisor в свернутом состоянии. Так она выглядит в системе comapping. Полную схему можно найти в приложении 1 (см. EIA Model.pdf).

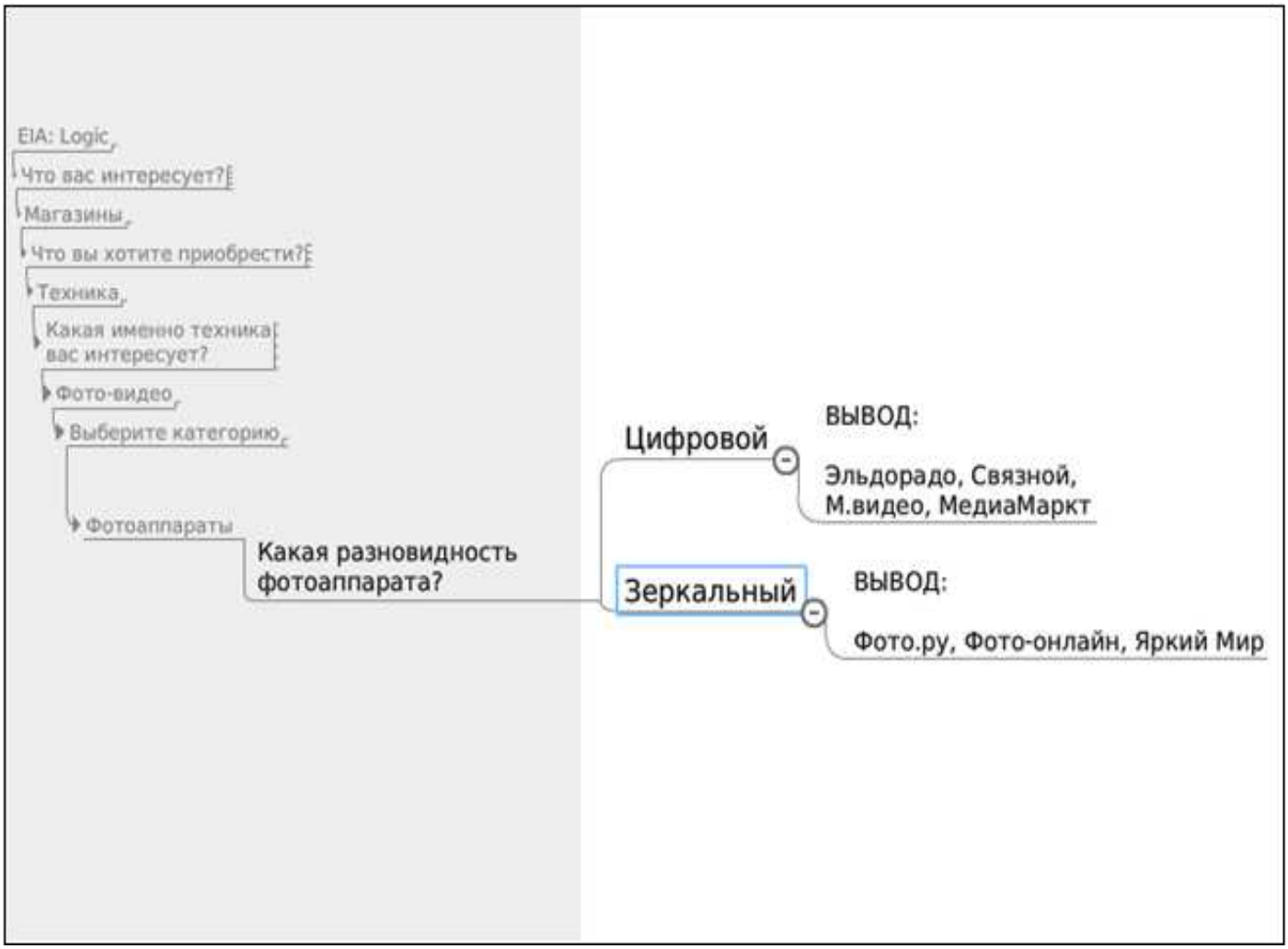


**Рисунок 3. Модель ЭС EIA в системе go.comapping.com.**

Количество переходов в цепочке может служить мерой "смыслового расстояния" между двумя понятиями. Психологами проводились многочисленные опыты, которые подтверждают гипотезу о том, что для двух любых слов (понятий) существует ассоциативная цепочка, состоящая не более чем из семи слов. На самом деле, в нашей модели максимальная длина пути от одного понятия до другого не превышает 7 уровней.

Выделенные понятия предметной области и установленные между ними взаимосвязи служат основанием для дальнейшего построения экспертной системы, применяя уже более формальные методы.

## Этап формализации

На данном этапе определяются способы представления декларативных и процедурных знаний, это представление осуществляется и в результате формируется описание решения задачи ЭС на формальном языке.

В системе Everyday Intelligent Advisor, реализованной на языке C# Expert, в качестве способа представления знаний выступают фреймы. Рассмотрим структуру фрейма в программе.

Исходный код на C# и код языка Knowledge.Net разделяются ключевым словом ‘#frames’.

**Листинг 1.**

```
// C# Expert specific code

#frames
frame class Restaurant
{
     public enum Cuisine
    {
      russian = 1,
      italian,
      french,
      japan,
      undefined
    };
    own_slots
          Restaurant.Cuisine CuisineType =
Restaurant.Cuisine.undefined;

     public enum AverageBill
    {
      less_than_400 = 1,
      400_1000,
      more than_1000,
      undefined
    };
    own_slots
          Restaurant. AverageBill Price =
Restaurant.AverageBill.undefined;
```

```
frame ruleset GetPlace (ref GetPlace. Result result)
{
     own_slots
          context Restaurant;
          goal result;
          rule
          {
               if (CuisineType == Restaurant. Cuisine. Russian
               && Price == Restaurant.AverageBill.less_than_400)
               then
               {
                     result = "Teremok, Chainaya lozhka";
               }
               comment "We advise you to visit Teremok or
     Chainaya lozhka because these places are among the best
     moderate restaurants with russian cuisine. You can find
     it’s addresses below";
               }R01;
```

В языке C# Expet фреймы бывают трех видов. Здесь представлен фрейм-класс `Restaurant`, который используется для описания классов объектов в подзадаче «выбор ресторана». От него наследуются фреймы-экземпляры `Cuisine` и `AverageBill`. Данные фреймы описывают конкретные экземпляры класса. Фрейм – экземпляр должен быть унаследован от фрейма – класса, множественное наследование не допускается.

Фрейм-экземпляр состоит из структурных единиц двух типов: атрибуты и слоты. В качестве атрибутов может выступать любой код на языке C#, но в данном случае это перечисления. Значения атрибутов не сохраняются во внешней памяти.

Слоты же описывают свойства фреймов. В данном случае `own_slots` задает значение фрейма по умолчанию(`undefined`). Слоты сохраняются во внешней памяти между запусками экспертной системы.

Теперь рассмотрим структуру правил, которые будут участвовать в процессе логического вывода и в формировании конечного ответа в каждой

из подзадач EIA. Правило в C# Expert также является разновидностью фрейма. Также как фрейм-класс и фрейм-экземпляр, правила имеют атрибуты и слоты. В примере кода программы EIA типами слота являются правила, составляющие набор, целевая переменная и контекст фреймов, в котором исполняется данный набор правил.

Правило задает условие (`if`), при истинности которого, выполняется (`then`) код на языке C#. В примере, взятом из EIA, условием правила является « пользователь выбрал ресторан с русской кухней». Если условие истино, то пользователь получит совет пойти обедать в соответствующие заведение. Надо отметить, что отображение комментариев (`comment "We advise you.."`) обеспечивается машиной логического вывода, поставляемой вместе с библиотекой «C# ExpertLibrary».

Рассмотренная здесь часть кода экспертной системы Everyday Intelligent Advisor демонстрирует основные разновидности объектов и то как они могут быть связаны.

## Этап выполнения

Цель этого этапа — создание прототипа экспертной системы, решающего требуемые задачи. Разработка прототипа состоит в программировании его компонентов и наполнении базы знаний.

В ходе реализации, программа Everyday Intelligent Advisor прошла следующие состояния:

- прототип, имеющий простую структуру и содержащий несколько десятков правил
- расширенная версия прототипа, но работающая неустойчиво
- итоговый вариант программы

Описывая алгоритм работы экспертной системы, мы можем разделить итоговую программу на три части.

- Часть 1

Пользователь выбирает интересующий его подраздел программы. После этого ему задаются вопросы, и система EIA предлагает выбрать один и вариантов ответа. Вопросы задаются нелинейно, каждый следующий вопрос зависит от ответа на предыдущий. Именно на этапе реализации этой части программы EIA, происходило активное обращение к уже составленной модели системы (карты), рассмотренной выше.

- Часть 2

По правилам, хранящимся в базе знаний экспертной системы, Советчик осуществляет логический вывод решения выбранной пользователем подзадачи.

- Часть 3

Исходя из принятого в ходе рассуждений решения, экспертная система дает пользователю советы.

# Заключение

В ходе данной дипломной работы были выполнены следующие задачи:

1. Изучена система управления знаниями Knowledge.Net;
2. Реализована экспертная система Everyday Intelligent Advisor (Советчик в повседневной деятельности) в качестве апробации системы Knowledge.Net.

Советчик содержит базу знаний, включающую более 100 правил. Система является легко расширяемой, поэтому в дальнейшем возможно увеличение числа правил и добавление нового функционала в существующую программу.

Впоследствии необходимо выявлять потребности пользователей при помощи различных тестирований и расширять базу знаний Советчика в соответствии с ними. Также, поскольку на данный момент не существует широко распространенных аналогов экспертной системы Everyday Intelligent Advisor, рассматривается перспектива разработки коммерческой версии данного программного продукта, например в качестве веб-сервиса с клиентами ориентированными на мобильные платформы.

# Список литературы


[1] Новиков А.В. «Разработка языка представления знаний C# Expert», 2004 г

[2] Сафонов В.О., Новиков А.В., Сигалин М.В., Смоляков М.Л., Черепанов Д.Г. Интеграция методов инженерии знаний и инженерии программ: система управления знаниями Knowledge.NET, 2005г

[3] Экспертная система MYCIN

http://ru.wikipedia.org/wiki/MYCIN

[4] Экспертная система PROSPECTOR

http://penguin.photon.ru/doc/ai.shtml

[5] Экспертная система Dendral

http://en.wikipedia.org/wiki/Dendral

[6] Система EMYCIN

http://sapr.mgsu.ru/biblio/ex-syst/Glava10/Index7.htm

[7] Язык OPS5

http://inf.susu.ac.ru/~pollak/expert/eclipsed/RuleLanguages.htm

[8] Система управления знаниями Knowledge.Net

http://www.knowledge-net.ru/

[9] Knowledge Interchange Format (KIF)
http://logic.stanford.edu/kif/kif.html

[10] Clips Tutorial

http://www.iweb.tntech.edu/bhuguenard/ds6530/ClipsTutorial/CLIPS%20tutorial%201.htm

[11] go.comapping.com

http://go.comapping.com/comapping.html