МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

СОГЛАСОВАНО
Генеральный директор
ЗАО «АйТи»

_____ Бакиев О.Р.
"19" декабря 2011 г.

УТВЕРЖДАЮ
Ректор НИУ ИТМО

_____ Васильев В.Н.
"19" декабря 2011 г.

МНОГОПРОФИЛЬНАЯ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА СОЗДАНИЯ И УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СРЕДОЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ CLAVIRE

ПРОГРАММНЫЙ КОМПОНЕНТ ДИАЛОГА ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ CLAVIRE/ITREE

ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.80066-06 13 18-ЛУ

Представители
Организации-разработчика

Руководитель разработки,
профессор НИУ ИТМО

_____ Бухановский А.В.
"19" декабря 2011 г.

Ответственный исполнитель,
с.н.с. НИУ ИТМО

_____ Луценко А.Е.
"19" декабря 2011 г.

Нормоконтролер
ведущий инженер НИУ ИТМО

_____ Позднякова Л.Г.
"19" декабря 2011 г.

2011

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УТВЕРЖДЕН
RU.СНАБ.80066-06 13 18-ЛУ

МНОГОПРОФИЛЬНАЯ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА СОЗДАНИЯ И УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СРЕДОЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ CLAVIRE

ПРОГРАММНЫЙ КОМПОНЕНТ ДИАЛОГА ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ CLAVIRE/ITREE

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 18

ЛИСТОВ 25

2011

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

АННОТАЦИЯ

Документ содержит описание программного компонента диалога поддержки принятия решений CLAVIRE/iTree RU.СНАБ.80066-06 01 18 многопрофильной инструментально-технологической платформы создания и управления распределенной средой облачных вычислений CLAVIRE, разрабатываемой в рамках проекта «Создание распределенной вычислительной среды на базе облачной архитектуры для построения и эксплуатации высокопроизводительных композитных приложений» (Договор № 21057 от 15 июля 2010 г., шифр 2010-218-01-209) в рамках реализации постановления Правительства РФ № 218 «О мерах государственной поддержки развития кооперации российских высших учебных заведений и организаций, реализующих комплексные проекты по созданию высокотехнологичного производства». Программный компонент предназначен для решения следующих задач: а) поддержка принятия решений пользователем МИТП CLAVIRE в процессе построения композитных приложений с использованием базы экспертных знаний, доступных посредством компонента CLAVIRE/iKnow; б) обеспечение методической поддержки пользователей (в рамках предметно-ориентированных центров компетенции) за счет предоставления систематизированного доступа к знаниям, хранящимся в базе знаний.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Программный компонент CLAVIRE/iTree RU.СНАБ.80066-06 01 18 предназначен для поддержки принятия решений пользователя МИТП при формировании вычислительных задач и описании условий их выполнения. Настоящая версия компонента обеспечивает анализа структуры программных решений, используемых алгоритмов, а также структуры и форм представления данных, реализуемых в составе МИТП.

Программный компонент разработан на языке C# с использованием технологии .NET Framework, в его состав входят:

1) Модуль логического вывода и построения дерева принятия решений с использованием базы знаний, доступных посредством компонента CLAVIRE/iKnow. Основной задачей данного модуля является построение дерева принятия решений на основе имеющегося набора активных фактов, сформированных в ходе взаимодействия с пользователем. Доступ к модулю осуществляется посредством WCF-сервиса, реализующего базовый функционал построения дерева принятия решений с использованием базы знаний.
2) Модуль представления дерева принятия решений. Данный модуль реализован в форме web-приложения с применением технологии Silverlight. Приложение представляет собой клиентскую часть компонента и работает с использованием базовых функциональных возможностей модуля логического вывода и построения дерева принятия решений.

Компонент может функционировать:

1) на вычислительной системе под управлением ОС Linux (с ядром 2.6.22 и выше) с установленной средой Mono Framework с поддержкой библиотек .NET 3.5 и выше (рекомендуется версия Mono Framework 2.8 и выше). Для корректного функционирования необходимо наличие установленного web-сервера с поддержкой технологии WCF-Services (рекомендуется использование web-сервера XSP, поставляющегося в составе среды Mono Framework);
2) на вычислительной системе под управлением ОС Windows (версии XP и выше) с установленной средой .NET Framework версии 3.5 и выше. Для корректного функционирования необходимо наличие установленного web-сервера с поддержкой технологии WCF-Services (рекомендуется использование Microsoft IIS версии 7.0 и выше).

Программный компонент CLAVIRE/iTree использует компонент CLAVIRE/iKnow для доступа к базе знаний о вычислительных сервисах, доступных в составе МИТП, и способах их применения для решения задач предметной области.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1. Область применения

Программный компонент CLAVIRE/iTree предназначен для решения следующих задач.

- Поддержка пользователя в процессе решения задач избранной предметной области за счет: а) предоставления структурированной информации о параметрах, наиболее значимых для решения поставленной задачи; б) оценки доступных пользователю вариантов решения поставленной задачи; в) формирования скриптового представления на языке EasyFlow для доступных пользователю решений.
- Методическое изучение структуры базы знаний с использованием web-интерфейса системы (в рамках предметно-ориентированных центров компетенции) с возможностью сравнительного анализа методов решения по различным критериям качества (точность, скорость и т.п.).

2.2. Основные технологические функции

Программный компонент CLAVIRE/iTree представляет собой интерактивное web-приложение, интерфейс которого построен на базе концепции дерева принятия решений, которое строится с использованием иерархии базы знаний для выбранной области компьютерного моделирования (см. Описание программы для компонента CLAVIRE/iKnow). При этом пользователь в процессе взаимодействия с интерфейсом приложения осуществляет формирование базы активных фактов, проход по дереву решений, сравнение доступных вариантов решения задачи в соответствии с предложенной системой критериев качества и выбор подходящего варианта решения.

Основными технологическими функциями, реализуемыми программным компонентом CLAVIRE/iTree, являются:

1) обеспечение анализа предоставленных пользователем данных для выбора наилучшего варианта решения. В том числе должны учитываться: а) заданные пользователем параметры решения задачи, б) выбранные варианты значений

параметров из словарей базы знаний, в) загруженные пользователем в хранилище файлы данных, г) величины, требуемые пользователю в качестве выходных;

2) предоставление пользователю интерфейса, отражающего структуру дерева принятия решений и возможные пути прохода по этому дереву с формированием базы активных фактов;
3) предоставление пользователю возможных вариантов решения задачи в форме скрипта на языке EasyFlow, готового к исполнению ядром МИТП. При этом варианты решения должны быть снабжены оценками в соответствии с набором критериев качества, доступных в составе базы знаний.

2.3. Ограничения на применение

Для корректной работы механизма оценки альтернатив решения компонента CLAVIRE/iTree необходимо наличие в базе знаний (доступной посредством компонента CLAVIRE/iKnow) информации о совокупности доступных сервисов в системе, реализуемых ими методах, входных и выходных данных, способах оценки качества методов. При этом качество работы системы определяется, с одной стороны, совокупностью правил и закономерностей по оценке и выбору методов, формализованных в составе базы знаний, а с другой – сохраненными в составе базы знаний коэффициентами оценки, полученными по экспериментальным данным.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Общие принципы логического вывода

Логический вывод на базе онтологической структуры, доступной посредством соответствующего сервиса, обеспечивает возможность построения цепочек заданий (WF) с привлечением экспертных знаний различных уровней абстракции. При этом базовая процедура логического вывода может быть описана следующим образом.

1) Производится формализация требований пользователя к решаемой задаче: требования пользователя формируют базовый набор активных фактов, записанный в терминологии используемой базы знаний. Факты, предоставляемые пользователем, можно разделить на две допускающие пересечение группы: *определяющие* – набор фактов, на основании которого формируется базовая цепочка решения (достаточный набор для однозначного определения решаемой задачи); *дополнительные* – набор

требований и ограничений, более точно определяющих специфику решаемой задачи. Базовый набор активных фактов должен характеризоваться *полнотой* (позволять однозначно идентифицировать решаемую задачу) и *непротиворечивостью* (множества определяющих и дополнительных фактов не должны содержать взаимоисключающих утверждений). В случае неполноты базового набора активных фактов возможно применение дополнительных процедур по устранению неполноты фактов: активный диалог с пользователем (на основании набора вопросов, доступных системе); попытка устранения неполноты знаний на основе знаний с низкой достоверностью; построение активных фактов на основании опыта системы. Противоречивость активных фактов снимается на основании качественной оценки правил, выделения внутренне непротиворечивых альтернативных подмножеств активных фактов. Базовый набор активных фактов расширяется на основании правил логического вывода, доступных системе. Полученный таким образом набор активных фактов (или несколько альтернативных наборов) используется в последующих процедурах.

2) *Фильтрация*. На основании построенного набора активных фактов производится оценка элементов онтологии в пространстве критериев качества. На основании произведенной оценки производится первоначальная фильтрация онтологии, отсеивающая элементы, недопустимые в контексте решаемой проблемы. Отфильтрованная таким образом онтология представляет собой ориентированный граф, в котором каждой вершине или ребру сопоставлено значение в пространстве критериев качества. В простейшем варианте этот шаг позволяет выбрать допустимые в данной ситуации элементы онтологической структуры.

3) *Определение доступных методов*. В составе построенного графа могут быть выделены связанные подграфы, каждый из которых состоит из элементов онтологии, определяющих один из доступных методов предметной области. При этом доступность метод определяется: а) реализацией этого метода хотя бы одним из сервисов, входящих в состав программного комплекса; б) соответствием метода сформированному набору активных фактов (определяется на основании процедуры фильтрации (см. шаг 2)). В общем случае каждый метод описывается подграфом фиксированной структуры, включающим описание входных и выходных параметров, дополнительных ограничений, реализации метода и ассоциированных с ним вспомогательных процедур и т.п., а также расширения базовой структуры за счет отношений предметной области.

4) *Композиция*. На основании построенного набора доступных методов должен быть построен абстрактный сценарий решения поставленной задачи. Структурная связь методов определяется: а) комбинацией типовых схем взаимодействия (последовательный запуск, цикл, параллельный перебор, ветвление и пр.); б) соответствием зависимостей по данным (входные данные для каждого из реализуемых методов должны быть либо предоставлены пользователем, либо получены в ходе выполнения предыдущих методов). Возможны различные варианты построения абстрактного сценария на основании формализованного запроса и набора доступных методов. Простейшим вариантом можно назвать последовательный запуск сервисов, приводящий в итоге к получению заданной величины. Процесс формирования абстрактного WF происходит под управлением интеллектуальной системы, действующей на основании активных фактов: ИС определяет возможные формы абстрактного WF, допустимые методы для использования в его составе и подходящую процедуру перебора. В итоге каждый из построенных сценариев представляет собой допустимый вариант решения поставленной задачи (абстрактный WF).
5) *Оценка*. Каждый из построенных WF должен быть подвергнут автоматическому анализу, результатом которого является интегральная оценка, позволяющая: а) провести ранжирование альтернативных вариантов решения для автоматического выбора лучшего из них (по совокупности критериев) или предоставления выбора пользователю; б) отсеять заведомо некачественные варианты решения. При этом входными данными для построения интегральной оценки служат оценки элементов онтологии, входящих в состав подграфов, описывающих отдельные методы, используемые в решении (см. шаг 3). Функция интегральной оценки должна позволять учитывать не только суммарное качество отдельных методов, но и характеристики абстрактного WF как единой системы.

В общем случае на каждом этапе могут порождаться дополнительные альтернативы решения. Данная процедура проводится итеративно на каждом из концептуальных уровней базы знаний. Таким образом, строится дерево решений, структура которого определяется базовой иерархией концептов (см. Описание программы для компонента CLAVIRE/iKnow). На заключительном этапе производится оценка каждого из листьев полученного дерева таким образом, чтобы можно было либо провести автоматический выбор оптимального (с точки зрения качества) решения, либо предоставить пользователю осмысленные и обоснованные результаты оценки с

предложением осуществить самостоятельный выбор (например, установив какой-либо из критериев качества как приоритетный).

Рассматривая процедуру логического вывода, следует отметить, что ключевую роль в ней играет система критериев качества формируемых решений. В состав простейшей системы критериев качества, используемой в рамках текущего исследования, можно включить три базовых параметра.

1) *Точность предлагаемого решения*, определяемая с применением экспертных знаний об: а) точности используемых методов, б) качестве реализации этих методов в составе вычислительных пакетов.
2) *Время работы*, определяемое *a priori* на основании моделей производительности. При этом возможна оценка как на уровне вычислительных пакетов (относительная оценка), так и на уровне сервисов (абсолютная оценка). Относительная оценка позволяет сравнить варианты реализации каких-либо методов между собой, а абсолютная – оценить время исполнения задания в требуемых условиях.
3) *Надежность решения* основывается на истории запусков вычислительных сервисов, входящих в состав решения (в первую очередь, принимается во внимание доля отказов сервиса).

3.2. Структура компонента

В состав компонента входят следующие элементы:

- WCF-сервис логического вывода, включающий в себя все необходимые библиотеки (в т.ч. библиотеку доступа к базе знаний компонента CLAVIRE/iKnow). Более подробно структура и интерфейс сервиса рассмотрены в разделе 3.3;
- Silverlight-приложение, готовое для интеграции в состав web-решений (в том числе вызова из интерфейса компонента взаимодействия с пользователем CLAVIRE/Ginger. Более подробно структура и интерфейс сервиса рассмотрены в разделе 3.4;
- хранилище деревьев принятия решений, сформированных в рамках пользовательских сессий. На данном этапе реализовано в виде файлового хранилища (выделенной папки), в котором сохраняются данные о структуре дерева принятия решений в форме XML-файлов (листинг. 3.1).

```
<?xml version="1.0"?>
<EntityTreeDescription xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <RootNode>
    <NodeId>2ef0b884-1300-423a-ba72-35b2d8850783</NodeId>
    <Parametrised>false</Parametrised>
    <GroupOnlyNode>true</GroupOnlyNode>
    <ChildNodes>
      <EntityNodeDescription>
        <NodeId>0f11aa25-760d-4409-98f5-315438693fa8</NodeId>
        <Parametrised>false</Parametrised>
        <GroupOnlyNode>false</GroupOnlyNode>
        <State>New</State>
        <Entity>
          <Id>4</Id>
          <Name>Задача оптимизации геометрии</Name>
          <Description>Задача оптимизации геометрии представляет собой приближенное
решение уравнение Шредингера для анализа функции энергии молекулы в зависимости от
положения координат атомов.</Description>
          <Uri>GeometryOptimization</Uri>
        </Entity>
        <FactCollection>
          <Fact>
            <IsRequest>false</IsRequest>
            <LayerEntityId>14</LayerEntityId>
            <KeyId>5</KeyId>
            <Value>14</Value>
            <RefId>Квантовая химия</RefId>
            <FClass>LayerEntitySelected</FClass>
            <DisplayName>Предметная область</DisplayName>
          </Fact>
          <Fact>
            <IsRequest>true</IsRequest>
            <LayerEntityId>4</LayerEntityId>
            <KeyId>3</KeyId>
            <RefId>1</RefId>
            <FClass>FileUploaded</FClass>
            <DisplayName>Анализируемая система</DisplayName>
          </Fact>
        </FactCollection>
        <Script />
      </EntityNodeDescription>
    </ChildNodes>
    <State>Expanded</State>
    <Entity>
      <Id>14</Id>
      <Name>Квантовая химия</Name>
      <Uri>QChem</Uri>
    </Entity>
    <FactCollection />
    <Script />
  </RootNode>
</EntityTreeDescription>
```

Рисунок 3.1 – Пример описания дерева принятия решений в формате XML

3.3. Модуль логического вывода и построения дерева принятия решений

Модуль логического вывода реализован в виде WCF-сервиса, задачей которого является построение дерева принятия решений в соответствии с информацией, получаемой в ходе взаимодействия с пользователем. Таким образом, ключевым объектом работы данного модуля являются дерево принятия решений и его элементы. На рис. 3.2 приведена структура основных классов данного модуля. Структура классов более детально описана в разделе 3.4. Наиболее важен в составе данной структуры класс *EntityNodeDescription*, описывающий элемент дерева, ассоциированный с определенным

уровнем в иерархии базы знаний. Информация о соответствующем элементе базы знаний хранится в форме экземпляра класса *LayerEntityDescription*. Класс *EntityTreeDescription* описывает дерево целиком и предоставляет доступ к базе знаний (с использованием возможностей компонента CLAVIRE/iKnow – см. класс *EscienceModelContainer*).

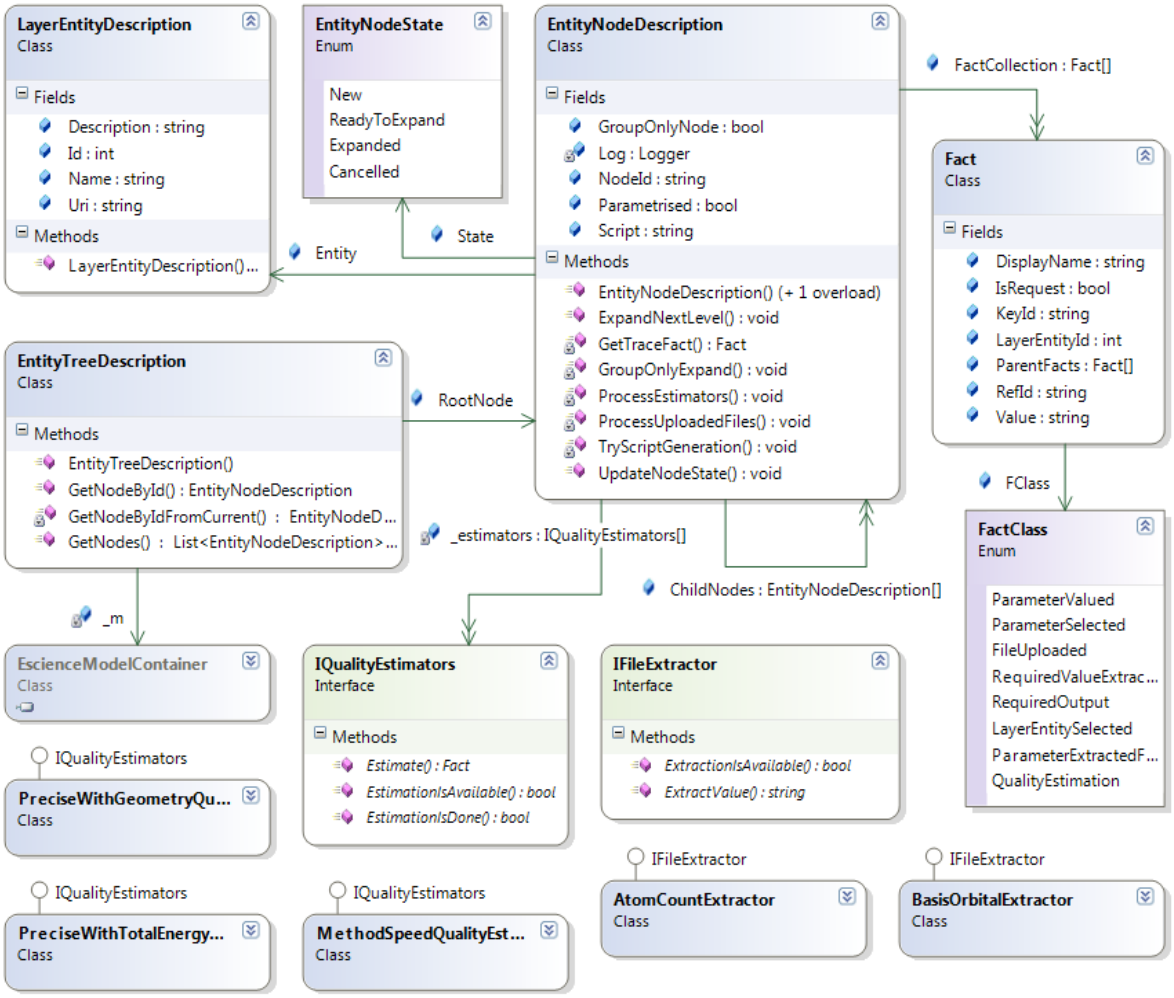

Рисунок 3.2 – Основные классы модуля логического вывода и построения дерева принятия решений

Отдельно следует упомянуть интерфейсы и реализующие их классы, в общем случае составляющие императивную часть базы знаний.

- *IQualityEstimator*. Интерфейс, определяющий структуру классов, описывающих способы определения качества предлагаемых решений. Помимо непосредственно оценки качества решения по текущему набору активных фактов интерфейс определяет возможность проверки допустимости оценки и поиск уже вынесенной оценки в составе набора фактов. В текущей версии базы знаний реализованы три класса, обеспечивающие оценку качества решений для области «Квантовая химия».

- *IFileExtractor*. Интерфейс, определяющий классы, описывающие способ извлечения информации из файлов. При этом для извлечения информации из файлов может быть использован текущий набор активных фактов. В текущей версии базы знаний реализованы два класса, обеспечивающие анализ файлов формата *.xyz для области «Квантовая химия».

Алгоритм построения дерева принятия решений строится на основе следующих принципов.

1) Для каждого уровня иерархии, представленного в базе знаний, формируется два уровня элементов дерева принятия решений: общее описание элемента данного уровня (общий подуровень) и варианты наборов параметров для данного элемента (параметризованный подуровень). Исключение составляют уровни, обеспечивающие группировку сущностей (*GroupOnlyNode*). Такие уровни не разделяются на подуровни.
2) Для перехода от общего подуровня к параметризованному осуществляется поиск возможных вариантов параметров в соответствии с: а) заданными на общем уровне пользовательскими параметрами, б) набором активных фактов, накопленных на предыдущих уровнях.
3) Для перехода от параметризованного подуровня к общему следующего уровня осуществляется подбор допустимых значений следующего подуровня в соответствии со связями, заданными в базе знаний.
4) Ключевыми элементами, на основании которых осуществляется принятие решений о подборе дочерних элементов, являются активные факты (класс *Fact* на рис. 3.2). Рассмотрим более подробно структуру этого класса.
    a. *DisplayName*. Заголовок, с которым данный факт представляется пользователю.
    b. *IsRequest*. Параметр, определяющий, является ли факт запросом значения.
    c. *KeyId*. Идентификатор ключевой сущности, ассоциированной с данным фактом (параметр, файл, значение и т.п.).
    d. *LayerEntityId*. Идентификатор сущности, входящей в базовую иерархию, для которой данный факт был определен.
    e. *ParentFacts*. Массив фактов, использованных при формировании данного.
    f. *RefId*. Идентификатор вспомогательной сущности, ассоциированной с данным фактом.
    g. *Value*. Значение, ассоциированное с данным фактом.

h. *FClass*. Класс факта. В текущей версии компонента присутствуют следующие классы фактов:
   i. *ParameterValued*. Установлено значение для параметра.
   ii. *ParameterSelected*. Выбрано значение для параметра.
   iii. *FileUploaded*. Загружен файл.
   iv. *RequiredValueExtraction*. Требуется получение значения путем анализа загруженного файла.
   v. *RequiredOutput*. Указанная величина требуется в качестве выходной.
   vi. *LayerEntitySelected*. Выбран элемент базовой иерархии (определяет путь прохода по дереву).
   vii. *ParameterExtractedFromFile*. Параметр получен путем анализа загруженного файла.
   viii. *QualityEstimation*. Произведена оценка качества с использованием класса, реализующего интерфейс *IQualityEstimator*.

5) Для осуществления процедур фильтрации и композиции на этом шаге используются правила, хранящиеся в базе знаний и определяющие допустимость каждого из элементов следующего уровня («сторожевые» условия). Эти правила используют набор фактов, накопленный при прохождении предыдущих уровней дерева. При этом универсальность такого подхода позволяет использовать как информацию, введенную пользователем в явном виде (заданные значения, выбранные на разных уровнях опции и т.п.), так и результаты логического и структурного анализа введенных данных. Правила формализованы в базе знаний в виде строк, описывающих λ-выражения на языке C#, представляющие собой предикаты, по текущему набору фактов определяющие применимость тех или иных элементов дерева принятия решений.
6) Построение скрипта готового решения осуществляется на параметризованном подуровне уровня «Пакет» базовой иерархии.

Важнейшие методы, определяющие интерфейс реализованного сервиса, приведены в табл. 3.1. Наибольшей значимостью обладает метод ExpandNode, в рамках которого реализована основная логика алгоритма логического вывода и построения дерева принятия решений.

Таблица 3.1

Интерфейс WCF-сервиса компонента CLAVIRE/iTree

Метод	Входные данные	Выходные данные	Описание
CreateEntityTreeDescription	treeId – глобальный идентификатор дерева	Описание дерева	Создать базовое дерево для задачи
ExpandNode	treeId – глобальный идентификатор дерева, nodeId – идентификатор узла, factUpdate – набор фактов для обновления	Описание обновленного дерева	Обработать элемент дерева и построить элементы-потомки
GetTreeById	treeId – глобальный идентификатор дерева	Описание дерева	Получить описание дерева по идентификатору
GetEasyFlowForNode	treeId – глобальный идентификатор дерева, nodeId - идентификатор узла	Скрипт для заданного узла	Получить скрипт для листового элемента дерева
UpdateNodeState	treeId – глобальный идентификатор дерева, nodeId – идентификатор узла, newState – новый статус	Описание обновленного дерева	Обновить статус узла

3.4. Основные классы модуля логического вывода и построения дерева принятия решений

Ниже перечислены наиболее важные классы модуля логического вывода и построения дерева принятия решений с указанием базовых операций, реализованных в них.

3.4.1. Класс EntityNodeDescription

Описание узла дерева принятия решений.

Свойства:

- NodeId (тип: string) – идентификатор узла;
- Parametrised (тип: bool) – узел с параметрами (второй уровень);
- GroupOnlyNode (тип: bool) – узел для группировки элементов;
- ChildNodes (тип: EntityNodeDescription[]) – дочерние узлы ;
- State (тип: EntityNodeState) – состояние узла;
- Entity (тип: LayerEntityDescription) – описание узла;
- FactCollection (тип: Fact[]) – набор активных фактов;
- Script (тип: string) – сгенерированный скрипт.

Методы:

- ExpandNextLevel – создание узла нового уровня (параметризация)
    - Входной параметр addFactCollection (тип: IEnumerable<Fact>) – дополнительные факты.
- UpdateNodeState – обработка запроса на обновление состояния
    - Входной параметр newState (тип: EntityNodeState) – новое состояние.

3.4.2. Класс EntityTreeDescription

Описание дерева целиком и предоставление доступа к базе знаний.

Свойства:

- RootNode (тип: EntityNodeDescription) – корневой элемент дерева;
- DefaultStorage (тип: IFileStorage) – хранилище данных по умолчанию.

Методы:

- GetNodeById – поиск узла по идентификатору
    - Входной параметр nodeId (тип: string) – идентификатор узла;
    - Возвращаемое значение (тип: EntityNodeDescription) – найденный узел.
- GetNodes – получение списка всех элементов дерева, начиная с заданного
    - Входной параметр (опциональный) rootNode (тип: EntityNodeDescription) – корневой элемент. По умолчанию – корневой элемент всего дерева.
    - Возвращаемое значение (тип: List<EntityNodeDescription>) – список элементов дерева.

3.4.3. Класс Fact

Описание отдельного факта в наборе активных фактов.

Свойства:

- IsRequest (тип: bool) – запрашивается от пользователя;
- LayerEntityId (тип: int) – идентификатор сущности, для которой получен факт;
- ParentFacts (тип: Fact[]) – факты, использованные для вывода;
- KeyId (тип: string) – идентификатор-ключ;
- Value (тип: string) – значение ;
- RefId (тип: string) – ссылочный ключ. Пример: тип файла ;
- FClass (тип: FactClass) – класс факта;
- DisplayName (nbg^ string) – jnj,hf;ftvjt bvz/

3.4.4. Интерфейс IFileExtractor

Интерфейс, описывающий извлечение данных из файла/

Методы:

- ExtractionIsAvailable – проверка доступности извлечения
  - Входной параметр fileStorageId (тип: string) – идентификатор файла в хранилище;
  - Входной параметр factCollection (тип: List<Fact>) – текущая коллекция активных фактов;
  - Возвращаемое значение (тип: bool) – извлечение данных доступно.
- ExtractValue –извлечь данные
  - Входной параметр fileStorageId (тип: string) – идентификатор файла в хранилище;
  - Входной параметр factCollection (тип: List< Fact >) – текущая коллекция активных фактов;
  - Возвращаемое значение (тип: string) – извлеченное значение.

3.4.5. Интерфейс IFileStorage

Интерфейс хранилища данных.

Методы:

- DownloadFile – выгрузить файл из хранилища
  - Входной параметр dataId (тип: string) – строковый идентификатор;
  - Входной параметр targetStream (тип: Stream) – выходной поток для выгрузки;
  - Возвращаемое значение (тип: long) – размер выгруженного файла.
- UploadFile – загрузить файл в хранилище
  - Входной параметр dataId (тип: string) – строковый идентификатор;
  - Входной параметр sourceStream (тип: Stream) – входной поток для загрузки;
  - Возвращаемое значение (тип: long) – размер загруженного файла.

3.4.6. Интерфейс IQualityEstimators

Интерфейс, описывающий процедуру оценки качества.

- Estimate – произвести оценку
  - Входной параметр factCollection (тип: List<Fact>) – текущий набор фактов;
  - Возвращаемое значение (тип: Fact) – факт с оценкой.
- EstimationIsAvailable – проверка доступности оценки

- Входной параметр factCollection (тип: List<Fact>) – текущий набор фактов;
- Возвращаемое значение (тип: bool) – оценка доступна.

- EstimationIsDone – проверка, выполнена ли оценка
  - Входной параметр factCollection (тип: List<Fact>) – текущий набор фактов;
  - Возвращаемое значение (тип: bool) – оценка выполнена.

3.4.7. Класс LayerEntityDescription

Описание элемента базы знаний.

Свойства:

- Id (тип: int) – идентификатор записи в таблице БД;
- Name (тип: string) – строковое наименование элемента;
- Description (тип: string) – описание элемента;
- Uri (тип: string) – уникальный в пределах БЗ строковый идентификатор;
- LayerOrderId (тип: int) – переменная для определения порядка представления слоев элементов в дереве принятия решений.

3.5. Модуль представления дерева принятия решений

Модуль представления дерева принятия решений построен в форме Silverlight-приложения, основным рабочим элементом которого является визуальный компонент TreeView (см. рис. 3.3а), включающий в себя формы, представляющие информацию об экземплярах класса *EntityTreeDescription* (см. раздел 3.3).

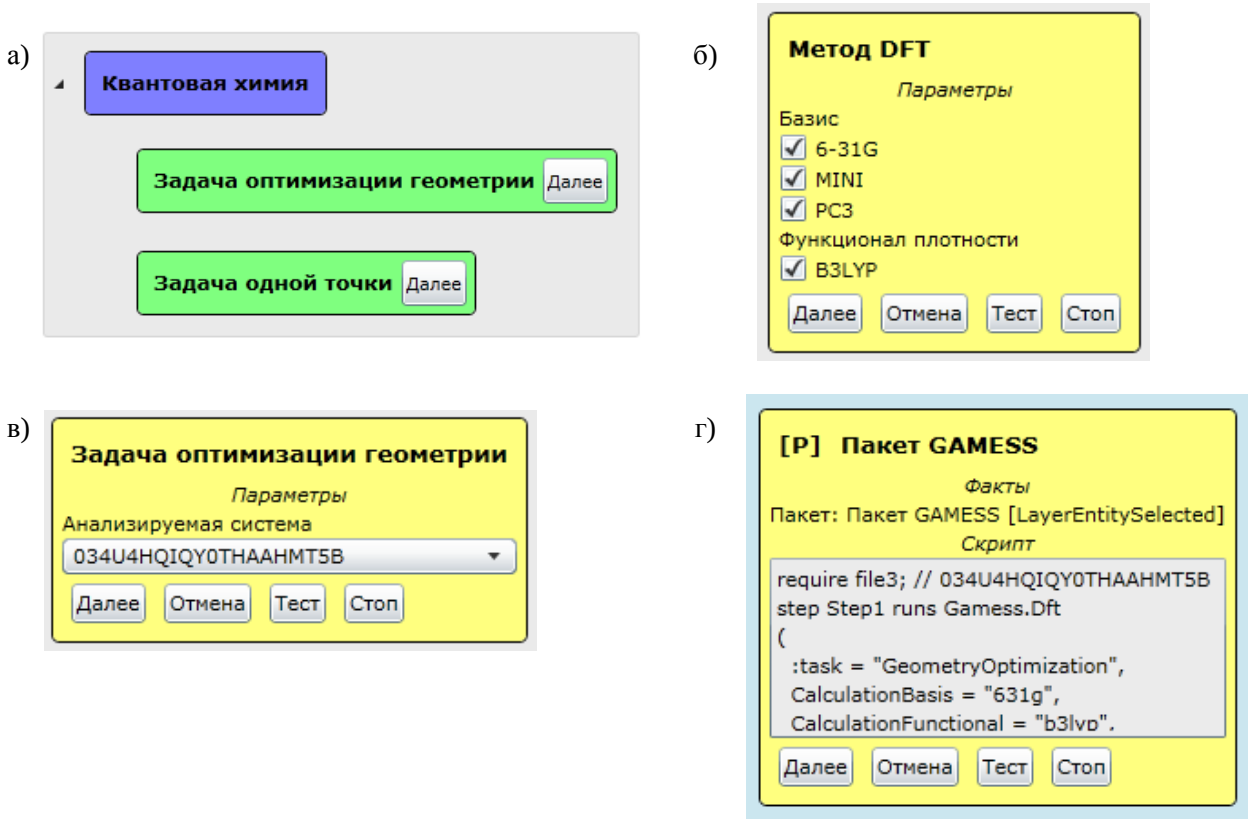

Рисунок 3.3 – Элементы представления дерева принятия решений. а) Организация иерархической структуры; б) выбор допустимых значений для построения параметризованного подуровня; в) выбор файлов из хранилища данных по строковому идентификатору; г) построение скрипта EasyFlow для выбранного решения

При этом формы включают в себя элементы управления, позволяющие пользователю задавать или выбирать значения параметров (см. рис. 3.3б,в), которые далее могут быть преобразованы в соответствующие активные факты. Для осуществления навигации по дереву пользователь должен задать значения (если таковые имеются) в интересующей его форме и нажать кнопку «Далее». Наиболее значимым результатом работы пользователя с системой является получение варианта решения задачи в форме скрипта на языке EasyFlow (см. рис. 3.3г).

Следует отметить, что в качестве условных обозначений состояния узлов дерева в данной версии системы используются цвета фона узлов и пометка «[P]», маркирующая узлы параметризованного подуровня дерева (см. рис. 3.4).

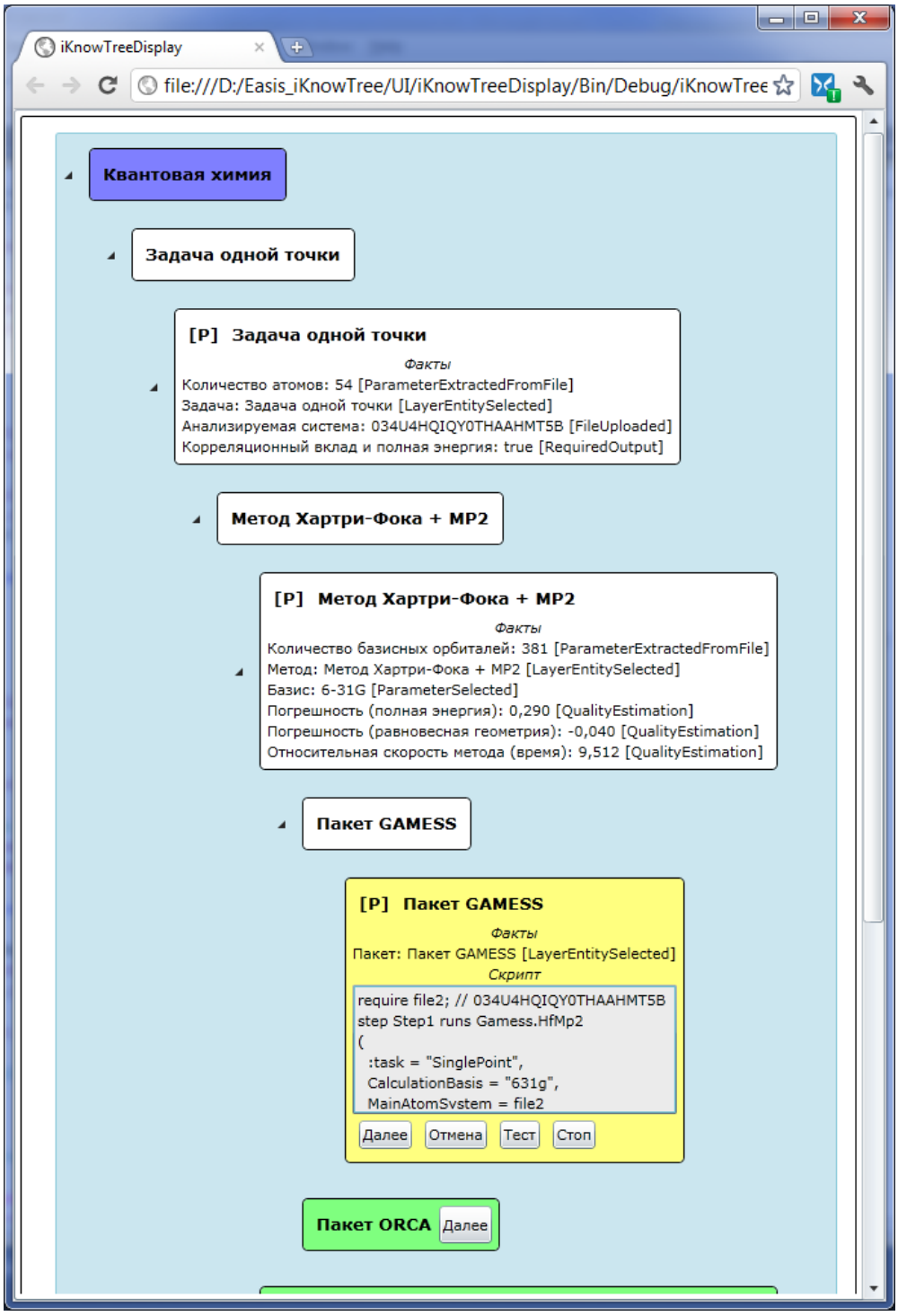

Рисунок 3.4 – Пример интерфейса модуля представления дерева принятия решений

Цвета элементов дерева имеют следующую интерпретацию.

1) *Синий.* Группирующий элемент иерархии (GroupOnlyNode).
2) *Зеленый.* Новый элемент иерархии, для которого анализ еще не проводился. Для активации элемента следует нажать кнопку «Далее» справа от заголовка элемента.
3) *Желтый.* Активный элемент, в котором проанализированы текущие факты и сформированы запросы к пользователю. Для обработки этого элемента необходимо

ответить на запросы системы с использованием соответствующих элементов управления и нажать кнопку «Далее».

4) *Белый.* Элемент пройден, проанализированные факты для этого элемента зафиксированы, построены дочерние элементы.
5) *Красный.* Элемент отключен пользователем (дальнейший проход по дереву отменен).
6) *Серый.* Элемент недоступен для работы.

В процессе построения дерева на пройденных элементах отображаются полученные на соответствующем этапе факты, в том числе отображаются результаты анализа файлов в хранилище данных (см., например, факты «Количество атомов» и «Количество базисных орбиталей» на рис. 3.4) и результаты оценки качества решений (см., например, факты «Погрешность (полная энергия)», «Погрешность (равновесная геометрия)», «Относительная скорость метода (время)» на рис. 3.4). Анализ выведенных фактов позволяет получить более детализированную информацию о введенных данных, а также проводить сравнительную оценку качества предлагаемых решений.

3.6. Основные классы модуля представления дерева принятия решений

3.6.1. Класс EntityTreeViewItem

Основной класс, используемый для организации диалога с пользователем, описывает визуальное представление узла дерева принятия решений.

Свойства:

- MainTreeId (тип: string) – идентификатор отображаемого дерева при работе с сервисом.
- EsClient (тип: EntityStorageServiceClient) – клиент доступа к сервису построения дерева принятия решений.
- AvailableFiles (тип: Dictionary< string, string >) – словарь доступных файлов. Пары (имя: идентификатор в хранилище).
- ForwardButtonsVisibility (тип: Visibility) – отображение кнопки «Далее».
- EntityNode (тип: EntityNodeDescription) – описание элемента дерева принятия решений в БЗ.

Методы:

- Конструктор
  - Входной (опциональный параметр) nodeDescription (тип: EntityNodeDescription) – стартовый элемент базы знаний.

- BlockNode – блокировка текущего узла
  - Входной параметр blockMessage (тип: string) – сообщение о блокировке.

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Для корректной работы компонента инструментальной среды используемые ЭВМ должны обладать характеристиками, обеспечивающими стабильную работу ОС и web-сервера в процессе организации последним доступа к web-сервисам и WCF-сервисам. Для запуска и работы основных модулей требуется x86-совместимый компьютер с тактовой частотой процессора не ниже 500 МГц и оперативной памятью не менее 256 МБ, на котором должна быть установлена платформа .NET (версии 3.5 и выше) либо платформа Mono (версии 2.6 и выше). Для взаимодействия с другими компонентами системы требуется наличие выхода в Интернет или локальную сеть (если web-сервисы других подсистем доступны из локальной сети) с соответствующей поддержкой со стороны оборудования.

5. ВЫЗОВ И ЗАГРУЗКА

Установка и настройка работы *модуля логического вывода и построения дерева принятия решений* осуществляются на базе сервера IIS, XSP или Apache (с установленным модулем Mod_Mono) стандартным образом в соответствии с документацией на указанные продукты (включая документацию платформы Mono для модуля Mod_Mono). Для корректной работы web-сервиса необходимо наличие следующих файлов, входящих в дистрибутив компонента:

1) конфигурационный файл Web.config;
2) файл с определением интерфейса сервиса Easis.iKnow.EntytyAccess.WCF.EntityStorageService.svc;
3) директория bin с исполняемыми файлами сервиса, включающими следующие сборки:
   - iKnowEntityDialog.dll – основные классы модуля;
   - iKnowEntityStorage.dll – библиотека доступа к базе знаний (компонент CLAVIRE/iKnow);
   - iKnowEntityStorage.WCF.dll – реализация WCF-сервиса;
   - NLog.dll – библиотека журналирования;

4) директория App_Data, содержащая файл escience_new.sdf с базой знаний, используемой посредством компонента CLAVIRE/iKnow.

Кроме того, должна быть создана директория (по умолчанию «c:\TreeStorage\»), используемая в качестве хранилища сессий (для каждой из сессий сохраняются дерево логического вывода и наборы соответствующих фактов). Пользователь, от имени которого осуществляется исполнение сервиса, должен иметь права на чтение и запись в данной директории.

Загрузка сервисов осуществляется стандартным для выбранного web-сервера способом.

Установка, настройка и вызов *модуля представления дерева принятия решений* осуществляются стандартным для выбранного web-сервера способом. При вызове приложения ему необходимо передать в качестве входного параметра массив идентификаторов файлов, доступных текущему пользователю в хранилище. Для этого задается параметр инициализации приложения с именем UserFiles, значением которого является строка, содержащая идентификаторы файлов, разделенные символом «|». Например, встраивание приложения в html-страницу может выглядеть следующим образом (рис. 5.1).

```
<asp:Silverlight
  ....
  InitParameters="UserFiles=O34U4HQIQY0THAAHMT5B|O34U4HQIQY0THAAHMT49" />
```

Рисунок 5.1 – Передача списка доступных файлов компоненту

6. ВХОДНЫЕ ДАННЫЕ

Входные пользовательские данные в интерактивном режиме задаются пользователем в процессе взаимодействия с экранными формами узлов дерева принятия решений. В частности, пользователь осуществляет выбор из списка доступных файлов в хранилище; задает строковые значения параметров рассматриваемой задачи; выбирает требуемые выходные величины; ограничивает возможные варианты перебора значений по словарям величин.

7. ВЫХОДНЫЕ ДАННЫЕ

Наиболее значимые выходные данные программного компонента – скрипты для решения задачи, описанной пользователем. Полученные в результате интерактивной работы с пользователем скрипты в текущей версии компонента доступны: а) для копирования в буфер обмена в соответствующих экранных формах компонента (узлы параметризованного подуровня уровня «Пакет» базовой иерархии); б) при вызове соответствующего метода WCF-сервиса, предоставляющего доступ к функциональности модуля логического вывода и построения дерева принятия решений.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

API	Application programming interface (интерфейс программирования приложений)
DFT	Метод функционала плотности
WCF	Windows Communication Foundation (программный фреймворк)
WF	Workflow (поток заданий)
XML	eXtensible Markup Language (расширяемый язык разметки)
БД	База данных
БЗ	База знаний
ИС	Информационная система
МИТП	Многопрофильная инструментально-технологическая платформа
ОС	Операционная система
ПО	Программное обеспечение
СУБД	Система управления базами данных
ЭВМ	Электронно-вычислительная машина

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
	изменённых	заменённых	новых	Аннулированных					