МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

СОГЛАСОВАНО
Генеральный директор
ЗАО «АйТи»

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

УТВЕРЖДАЮ
Ректор НИУ ИТМО

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

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

ПРОГРАММНЫЙ КОМПОНЕНТ ХРАНЕНИЯ ЗНАНИЙ CLAVIRE/IKNOW

ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.80066-06 13 17-ЛУ

Представители
Организации-разработчика

Руководитель разработки,
профессор НИУ ИТМО

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

Ответственный исполнитель,
с.н.с. НИУ ИТМО

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

Нормоконтролер
ведущий инженер НИУ ИТМО

_____ Позднякова Л.Г.
"28" 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 17-ЛУ

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

ПРОГРАММНЫЙ КОМПОНЕНТ ХРАНЕНИЯ ЗНАНИЙ CLAVIRE/IKNOW

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 17

ЛИСТОВ 23

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

2011

АННОТАЦИЯ

Документ содержит описание программного компонента хранения знаний CLAVIRE/iKnow RU.СНАБ.80066-06 01 17 многопрофильной инструментально-технологической платформы создания и управления распределенной средой облачных вычислений CLAVIRE, разрабатываемой в рамках проекта «Создание высокотехнологичного производства комплексных решений в области предметно-ориентированных облачных вычислений для нужд науки, промышленности, бизнеса и социальной сферы» (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) хранения экспертных знаний в форме, доступной для обращения со стороны других компонентов МИТП CLAVIRE; б) поддержки процесса диалога с пользователем за счет предоставления доступа к базе знаний, организованного в рамках процедуры определения доступных решений стоящей перед пользователем задачи; в) предварительной обработки знаний, в процессе оценки доступности и качества решений, представленных в базе знаний, при решении конкретной задачи.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Программный компонент хранения знаний CLAVIRE/iKnow RU.СНАБ.80066-06 01 17 предназначен для хранения, систематизации экспертных знаний о базовых методах компьютерного моделирования в заданной предметной области, а также обеспечения поддержки принятия решений пользователем в процессе конструирования, настройки и использования распределенных высокопроизводительных композитных приложений в заданной предметной области. Являясь одним из ключевых сервисов в рамках концепции iPSE [1], данный сервис обеспечивает комплексный доступ к моделям представления знаний в области компьютерного моделирования.

Разработанная и включенная в состав макета версия компонента реализует комплексный интерфейс доступа к данным, обеспечивающий работу напрямую с онтологическими структурами, а также доступ к базе данных, построенной на основе онтологической структуры, что обеспечивает ускоренную обработку запросов к хранилищу знаний.

Программный компонент CLAVIRE/iKnow разработан на языке C# с использованием технологии .NET. Компонент использует программное средство Pellet [2] (версии 2.2.2 и выше) для выполнения запросов к онтологической структуре, хранящей базу экспертных знаний. Реализация модуля доступа к онтологической структуре с применением API средства Pellet была выполнена на языке Java. Для описания базы экспертных знаний использован язык OWL [3]. Для хранения базы данных, построенной на основе онтологической структуры, была использована СУБД Microsoft SQL Server Compact [4]. Для организации доступа к базе данных была использована технология ADO.NET Entity Framework [5]. 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 2.0 и выше (рекомендуется версия Mono Framework 2.6 или выше). Для корректного функционирования необходимо наличие установленного web-сервера с поддержкой технологии ASP .NET WebServices (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 ASP .NET WebServices (рекомендуется использование Microsoft IIS версии 5.1 и выше).

Программный компонент CLAVIRE/iKnow используется программным компонентом CLAVIRE/iTree для построения интерфейса советующей системы поддержки пользователя в выборе методов решения задач предметной области с применением доступных в рамках МИТП сервисов.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1. Область применения

Программный компонент CLAVIRE/iKnow предназначен для решения следующих задач:

1) поддержка процесса построения диалога системы с пользователем при определении доступных способов решения поставленной задачи предметной области, включая поиск и применение соответствующих вычислительных сервисов в заданной предметной области;
2) оценка доступных способов решения поставленной задачи, их фильтрация и ранжирование в соответствии с заданными критериями качества (временем работы, точностью решения), в целях автоматизации описания задач в системе на формальном языке предметной области, с последующей трансляцией в исполнимое композитное приложение, описываемое в форме потока заданий (WF).

2.2. Основные технологические функции

Программный компонент CLAVIRE/iKnow представляет собой высокоуровневый интерфейс доступа к экспертным знаниям (в форме соответствующей базы знаний) в различных областях компьютерного моделирования. Основными технологическими функциями, реализуемыми программным компонентом CLAVIRE/iKnow посредством его взаимодействия с программным компонентом диалога поддержки принятия решений CLAVIRE/iTree, являются:

- организация доступа к формализованным и систематизированным экспертным знаниям в процессе построения диалога с пользователем в рамках выделенной сессии;
- обеспечение возможности расширения онтологической структуры за счет внедрения пользовательских знаний (в режиме прямой работы с онтологической структурой);

- ускоренный доступ к экспертным знаниям, представленным в форме базы данных, сформированной на основе имеющейся онтологической структуры.

2.3. Ограничения на применение

Для эффективного применения компонента CLAVIRE/iKnow в составе ядра МИТП необходимо наличие данных о совокупности доступных сервисов в системе, реализуемых ими методов, 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 построения базы знаний

С точки зрения методологии представления знаний о компьютерном моделировании можно выделить несколько базовых концептов, имеющих ключевое значение для полноты формируемого описания.

1) Объект моделирования. Этот концепт является ключевым объектом компьютерного моделирования, нацеленного на определение некоторых характеристик данного объекта за счет использования одной или нескольких моделей, ассоциированных с ним. При этом может рассматриваться комплексный объект моделирования, включающий в себя несколько объектов. В этом случае модели могут быть ассоциированы как с отдельными объектами в рамках композитного приложения, так и с несколькими, описывая их взаимодействие. Объект моделирования описывается набором характеристик, которые условно можно разделить на две группы: определяющие – фиксирующие объект моделирования и отличающие его от других объектов данного класса; переменные –описывающие состояние объекта моделирования и чаще всего являющиеся входными или выходными данными для процесса моделирования.
2) Модель – объект, служащий для получения знаний об объекте моделирования. Зачастую представляет собой некоторую математическую или логическую структуру, отражающую избранный набор характеристик исследуемого объекта. Для

работы с моделью могут быть применены различные методы моделирования, обеспечивающие, например, возможность получения по одним характеристикам модели других.

3) Метод – императивное описание процедуры работы с моделью с заданной целью: получением характеристик, проверкой гипотезы и т.п. Именно методы чаще всего используются в качестве базового объекта для реализации в составе вычислительных пакетов и представляют наибольший интерес для пользователей этих пакетов. Очевидно, что может существовать несколько реализаций одного и того же метода.

4) Реализация метода – алгоритм, реализующий некоторый метод и включенный в состав доступного пользователю вычислительного пакета.

5) Вычислительный сервис – вычислительный пакет, установленный на выделенном ресурсе, доступный в режиме удаленного использования.

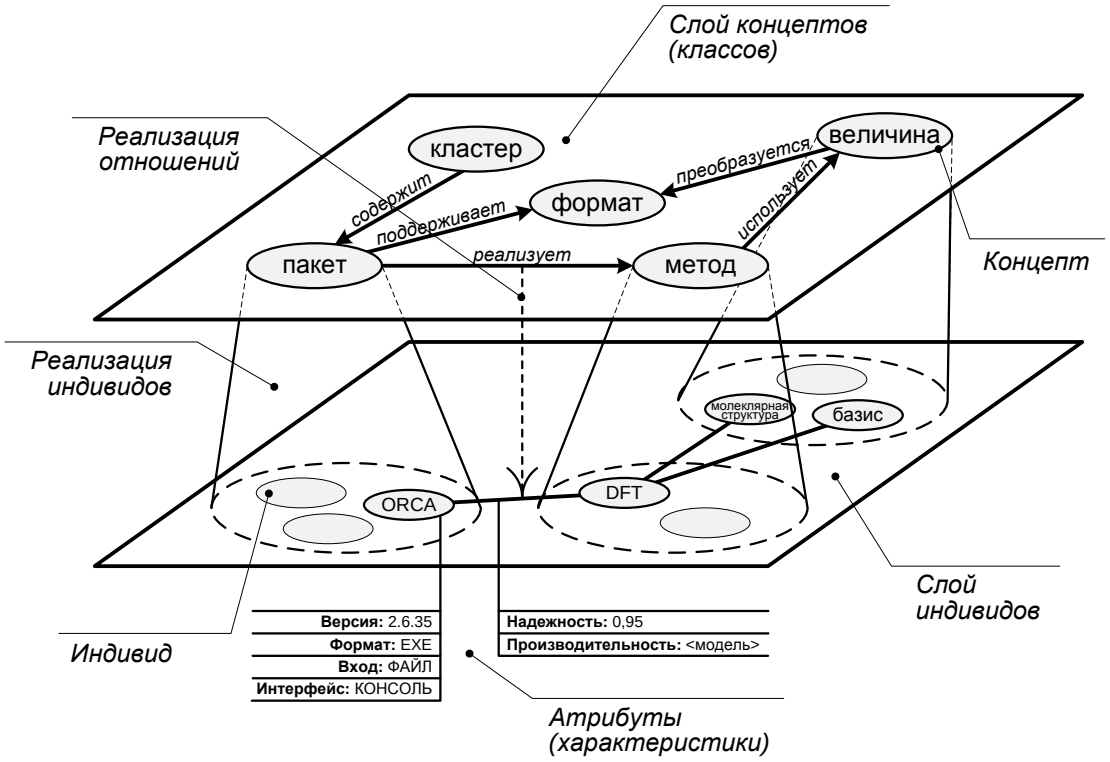

Рисунок 3.1 – Типовая структура онтологии представления исходных знаний в базе знаний компонента CLAVIRE/iKnow

На рис. 3.1 в качестве иллюстрации реализованного подхода приведена частичная структура предлагаемой онтологии, применимой для описания совокупности экспертных знаний, используемых в процессе организации высокопроизводительных вычислений. В составе онтологии можно выделить два основных слоя: описание концептов (классов) и индивидов, реализующих концепты. При этом индивиды могут быть связаны

отношениями, определенными на уровне концептов. Кроме того, допустимы отношения между отдельными концептами (например, отношение генерализации). В простейшем случае множество отношений может быть ограничено двумерными отношениями. Тем не менее этот вариант зачастую приводит к необоснованному усложнению структуры множества отношений за счет декомпозиции семантически связанных многомерных отношений на совокупность двумерных отношений. Еще одним элементом онтологии являются атрибуты (характеристики) индивидов, детализирующие их описание. Кроме того, одним из возможных расширений является ассоциация характеристик не только с индивидами (как реализациями классов), но и со связями между ними (как реализации классов допустимых связей). Такое расширение применимо при введении операций второго порядка (например, анализ «неидеальности» различного рода для связей, интерпретируемых как экспертные знания).

В качестве примера на рис. 3.1 приведен ряд индивидов и связей между ними, относящихся к предметной области квантовой химии (соответствующей задачам моделирования атомно-молекулярных наноразмерных структур и комплексов). Интерпретировать приведенную совокупность индивидов и концептов можно следующим образом: пакет ORCA [6] реализует метод функционала плотности (DFT), использующий в качестве параметров такие объекты, как собственно молекулярную структуру и базис, по которому раскладывается решение. При этом характеристики связи между методом и пакетом позволяют: а) оценить качество реализации (в данном случае в соответствии с критерием надежности); б) получить доступ к модели производительности, соответствующей данной реализации, для последующего использования в процессе моделирования.

Формально слой классов онтологии определяется как граф $O = \langle C, R \rangle$, где C – множество классов, R – множество абстрактных отношений, связывающих классы. Аналогично слой индивидов онтологии определяется как граф $\tilde{O} = \langle \tilde{C}, \tilde{R} \rangle$, где $\tilde{C}$ – множество индивидов, а $\tilde{R}$ – множество отношений между индивидами. При этом для каждого элемента слоя индивидов определены:

а) отношение генерализации $gn^{(C)} : \tilde{C} \to C$, $gn^{(R)} : \tilde{R} \to R$, определяющее связь индивидов и связей между ними с соответствующими классами и связями классов;

б) «сторожевые» условия (guard condition), определяющие применимость элементов в данных условиях $gc^{(C)}(F) : \tilde{C} \to \{0,1\}$, $gc^{(R)}(F) : \tilde{R} \to \{0,1\}$, где F – множество активных фактов, определенных для текущей задачи;

в) функция критериальной оценки $k^{(C)}(F):\left\{\tilde{c}\in\tilde{C}\mid gc^{(C)}(\tilde{c})=1\right\}\to\Psi^{(C)}$, $k^{(R)}(F):\left\{\tilde{r}\in\tilde{R}\mid gc^{(R)}(\tilde{r})=1\right\}\to\Psi^{(R)}$, где $\Psi^{(C)}$ и $\Psi^{(R)}$ – соответственно пространство критериев оценки индивидов и отношений между ними.

Следует отметить, что в общем случае сторожевые условия могут рассматриваться как один из критериев оценки. Тем не менее в данном документе они выделены в отдельный класс в целях упрощения работы критериальной оценки, оперирующей сокращенными множествами $\tilde{C}$ и $\tilde{R}$.

Целью работы блока логического вывода является определение способа решения поставленной задачи. При этом способ решения определяется как кортеж $S=(s_1,s_2\ldots s_N)$ фиксированной структуры, i-м элементом которого являются множества вида $s_i=\left\{\tilde{c}\in\tilde{C}\mid gn^{(C)}(\tilde{c})=c_i\right\}$, где последовательность классов $c_i\in C$ и требований к множествам s_i определяют общую структуру решения. Для оценки построенного решения по системе критериев применяется анализ графа $\tilde{O}'=\left\langle\tilde{C}',\tilde{R}'\right\rangle:\tilde{C}'=\bigcup_i s_i\cup\tilde{C}_S$, где $\tilde{C}_S$ – присоединенная система классов $\tilde{C}_S=\left\{\tilde{c}_S\mid\tilde{c}_S\notin\bigcup_i s_i,\exists\tilde{c}_1\in\bigcup_i s_i:rch(\tilde{c}_S,\tilde{c}_1)\right\}$, $rch(\tilde{c}_1,\tilde{c}_2)$ – отношение достижимости на графе $\tilde{O}'$. Оценка $\tilde{O}'$ осуществляется в пространстве критериев Ψ, определяемом пересечением множеств критериев, описывающих пространства $\Psi^{(C)}$ и $\Psi^{(R)}$. С другой стороны, задача ранжирования способов запуска требует определения отображения $\psi:\Psi\to R$, дающего возможность вычисления интегрального критерия качества оцениваемого решения.

3.2. Структура компонента

В текущей реализации данного сервиса используется модель представления знаний, использующая концепты 3–5 уровней иерархии. Концепты уровня 1–2 группируются в рамках понятия ***«задача»***, определяющего базовую схему (базовые входные и выходные данные, типовые условия и критерии оценки качества) для работы специалиста в рамках избранной предметной области.

В целом использование данного сервиса возможно в нескольких режимах.

1) *Интеллектуальная поддержка*. В данном режиме ключевой задачей хранилища знаний в форме онтологической структуры является поддержка логического вывода, обеспечивающего поддержку принятия решения пользователя в процессе

использования высокопроизводительных распределенных программных комплексов для компьютерного моделирования.

2) *Информационная поддержка*. Систематизированные экспертные знания могут (будучи снабженными соответствующим материалом публикаций) послужить основой для структурированного справочного пособия по решению задач данной предметной области.
3) *Методическая поддержка*. Базовая структура онтологии, построенная на основе предложенной иерархии, может служить инструментом для обучения специалистов в рассматриваемой предметной области как в ходе лабораторных работ, так и в процессе систематизации теоретических знаний.

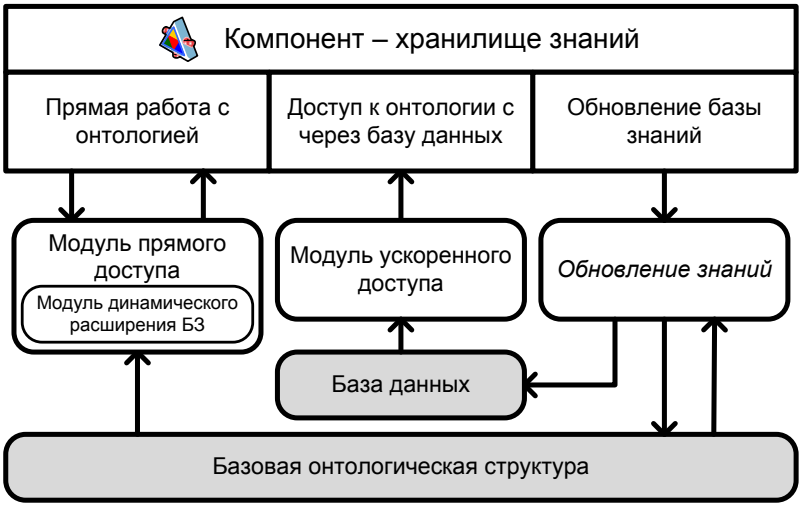

Рисунок 3.2 – Высокоуровневая архитектура компонента хранения знаний

Высокоуровневая архитектура сервиса представлена на рис. 3.2, она включает в себя два уровня хранения знаний.

1) Непосредственно онтологическая структура. База, наиболее полно отражающая набор доступных экспертных знаний, реализована с использованием возможностей языка OWL для описания базовых структур классов и индивидов и языка SWRL – для описания правил построения динамических связей между индивидами. Использование языка правил позволяет осуществлять динамическое расширение базы знаний пользовательскими фактами, определяющими условия срабатывания правил. Таким образом, появляется возможность задания базовых принципов логического вывода, определяемых экспертными знаниями, уже на уровне онтологической структуры.
2) База данных, хранящая фиксированный упрощенный вариант онтологической структуры и обеспечивающая более быстрый доступ к хранимым знаниям.

Используются средства СУБД Microsoft SQL Server Compact Edition для хранения упрощенной онтологии, построенной на базе концептуальной иерархии индивидов, представленной выше. Использование базы данных позволяет за счет некоторого снижения гибкости обеспечить существенно более быстрый способ работы с базой знаний. Автоматическое формирование базы знаний, синхронизированной по наполнению с текущей версией онтологической структуры, является одной из задач модуля обновления знаний.

Разработанная архитектура предполагает три основных способа использования данного сервиса, что определяет базовые алгоритмы взаимодействия модулей в составе сервиса.

1) *Прямой доступ к онтологии*. Представляет наиболее полный функционал для работы с онтологической структурой. Используя модуль динамического расширения БЗ, данный вариант использования позволяет формировать дополнительную онтологию, состоящую из описания активных фактов, полученных в ходе сессии работы с пользователем. Дополнительная онтология может быть добавлена к базовой для выполнения логического вывода на основе полученной расширенной онтологической структуры. При этом кроме высокоуровневого логического вывода, выполняемого в ходе внешнего анализа онтологической структуры, происходит автоматический логический вывод в форме построения динамических связей на основе правил, заданных в базовой онтологической структуре. Данный алгоритм реализован в форме WCF-сервиса, доступного для использования другими компонентами ядра МИТП.
2) *Доступ с использованием базы данных.* Обеспечивает в целом эквивалентный п.1 функционал в рамках режима интеллектуальной поддержки принятия решений пользователем. При этом механизм динамического расширения онтологии может быть исключен за счет более сложной схемы анализа базы знаний в ходе построения пользовательской сессии. Тем не менее такое усложнение допустимо, поскольку: а) при данном способе используется фиксированный сценарий взаимодействия с пользователем, что упрощает процесс работы анализа базы знаний; б) уменьшение времени реакции, полученное за счет применения более быстрой технологии доступа к знаниям обеспечивает возможность построения диалога с пользователем без заметных для него задержек. Данный алгоритм реализован в форме подключаемой библиотеки (сборки), реализующей доступ к базе данных с применением технологии ADO.NET Entity Framework.

3) *Обновление базы знаний*. В рамках данного способа использования сервиса выполняются расширение и модификация базовой онтологической структуры, а также автоматическое обновление базы данных для синхронизации с текущей версией онтологической структуры. Этот алгоритм реализуется вручную с соответствующим использованием инструментальных средств для доступа к СУБД и онтологическим структурам.

3.3. Реализация онтологической структуры

Для онтологии, описывающей основные методы, реализуемыми вычислительными моделями, доступными в качестве сервисов в составе программно-аппаратного комплекса (проблемно-ориентированной среды удаленного доступа), используется базовая структура классов (рис. 3.3).

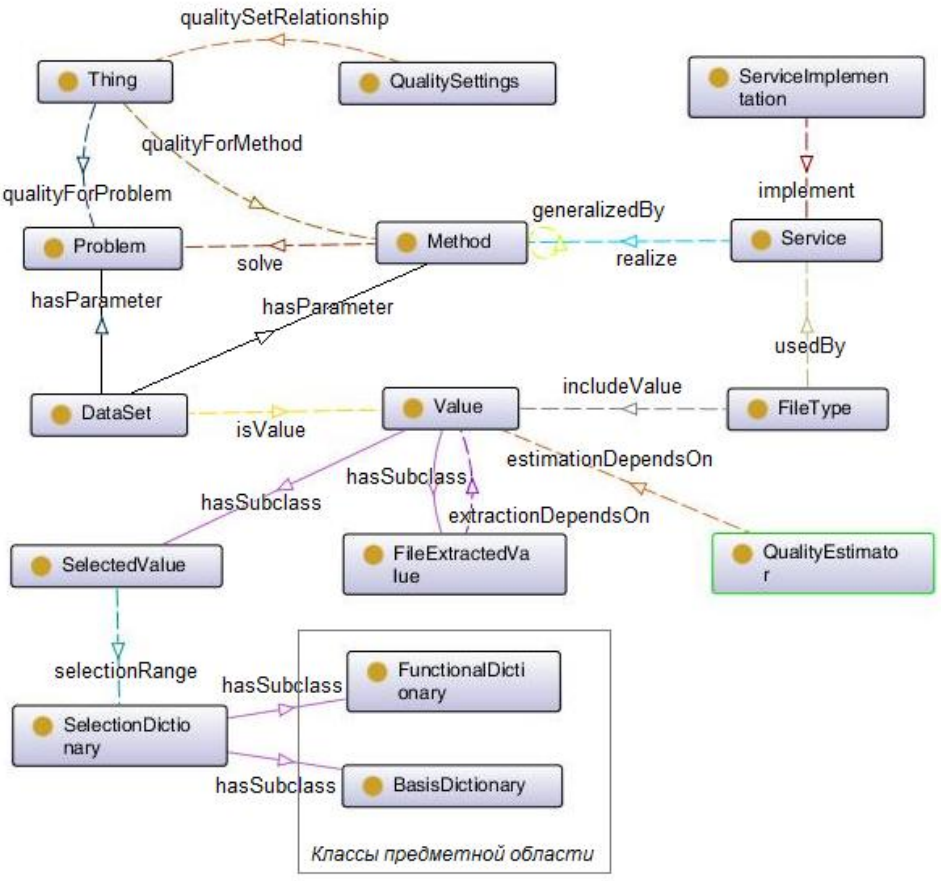

Рисунок 3.3 – Структура основных классов онтологии

К ним относятся:

- *Problem – задача, решаемая в рамках предметной области;*
- *Method – метод, обеспечивающий решение поставленной задачи;*
- *Service – вычислительный сервис, реализующий данный метод;*

- *ServiceImplementation – экземпляр сервиса, доступный в составе программно-аппаратного комплекса;*
- *DataSet – набор входных или выходных данных для заданного метода или решаемой задачи;*
- *Value – величины предметной области, используемые в качестве входных и* выходных данных для решения задач. Выделяются два специфических класса величин, различающихся способом их задания:
  - *FileExtractedValue* – извлекаемые из файлов величины. Способ извлечения должен быть описан в виде класса (в исходном коде компонента), реализующего унифицированный интерфейс IFileValueExtractor;
  - *SelectedValue* – величины, выбираемые из списка доступных. Список доступных значений задается в составе онтологии индивидами, относящимися к подклассам класса *SelectionDictionary*. В состав данной онтологии введены классы, описывающие возможные значения «Базис» и «Функционал плотности», определяющие процесс расчета в методах квантовой химии *BasisDictionary* и *FunctionalDictionary* соответственно.
- *FileType – файл, содержащий доступные для извлечения значения.*

Ключевым моментом для работы компонента CLAVIRE/iKnow является оценка качества доступных решений. Для этой оценки используются индивиды класса *QualityEstimator*, описывающие вариант оценки реализуемого метода с указанием зависимостей от входных данных и ссылкой на класс в исходном коде компонента, реализующем унифицированный интерфейс IQualityEstimator. Кроме того, для оценки конкретных задач, методов и величин используются наборы параметров, сгруппированные в индивидах класса *QualitySettings*.

3.4. Реализация базы данных

Структура базы данных, заполняемой в соответствии с набором знаний, формализованных в онтологической структуре, организованна на базе иерархии, приведенной в разделе 3.1. На рис. 3.4 приведена структура базы данных в форме модели, используемой платформой ADO.NET Entity Framework.

Ключевым элементом базы является таблица *LayerEntity*, хранящая информацию об индивидах, напрямую соотносимых с уровнями используемой иерархии. Сами уровни иерархии описываются в таблице *LayerClass*. В текущей версии компонента используются следующие уровни: предметная область, задача, метод, пакет, сервис. При этом уровень

иерархии «предметная область», не описанный в разделе 4.3.1, исполняет лишь группирующую роль.

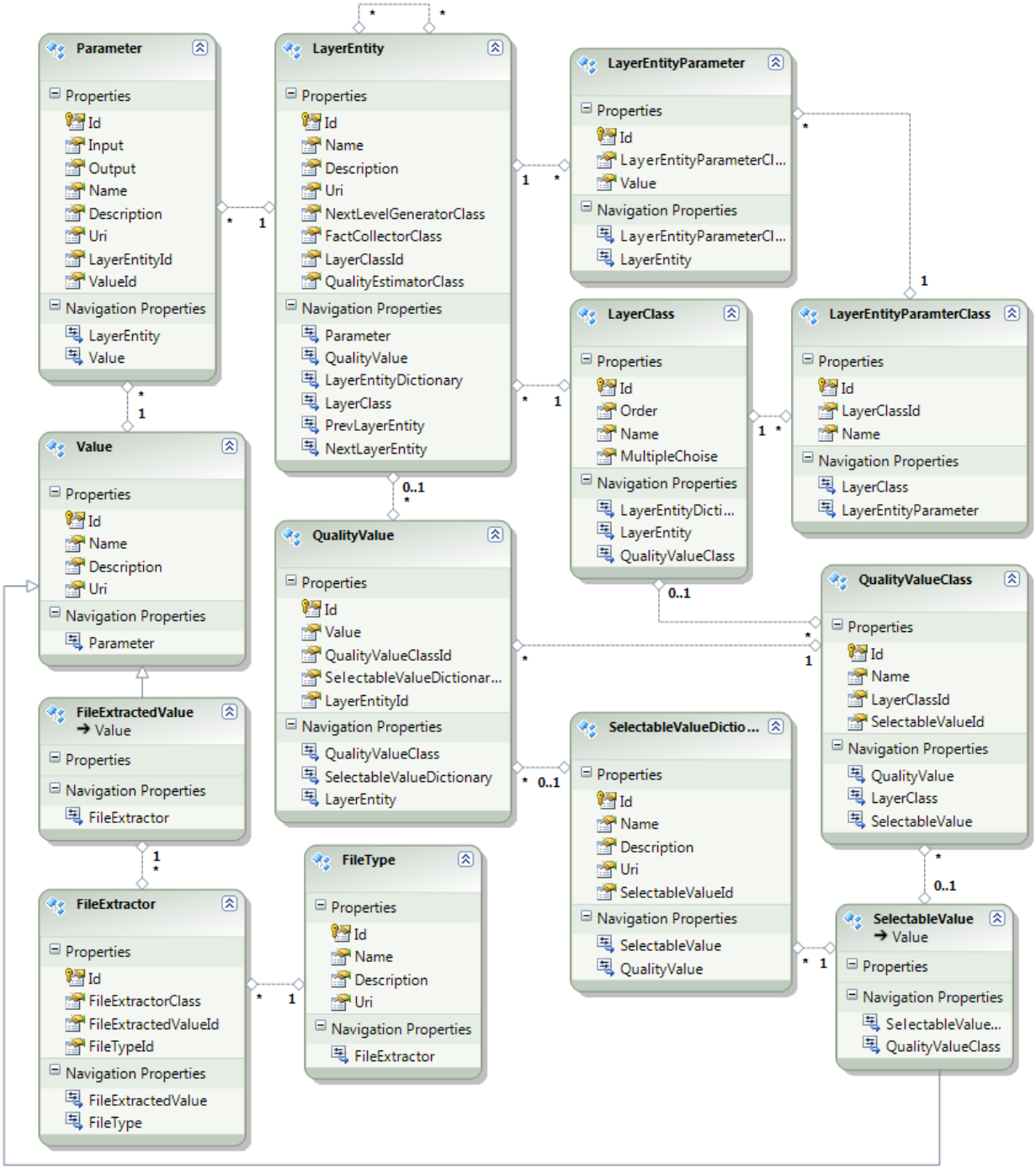

Рисунок 3.4 – Структура базы данных

Другим важным элементом базы данных является пара таблиц *Parameter* и *Value*, описывающих параметры (входные и выходные) и общие величины предметной области соответственно. При этом сущность *Value* генерализует частные случаи величин, различающихся способом их задания. Выделяются величины, извлекаемые из файлов, – *FileExtractedValue* (которые могут не являться входными или выходными для какого-либо

элемента базовой иерархии, а служить, например, для анализа качественных характеристик предлагаемого решения) и величины, выбираемые из списка допустимых, – *SelectableValue*. Данная иерархия повторяет иерархию, организованную в базовой онтологической структуре. При этом с первым классом сущностей ассоциированы сущности *FileType* и *FileExtractor*, описывающие доступный тип файлов и способ извлечения конкретной величины из него соответственно. Со вторым классом связан словарь допустимых значений *SelectableValueDictionary*. Для хранения параметров, позволяющих оценить качество решения, используется таблица *QualityValue*, элементы которой могут быть ассоциированы с сущностями класса *EntityLayer* или *SelectableValueDictionary*. Таким образом, коэффициенты могут быть ассоциированы либо с сущностями базовой иерархии (сервис, пакет, метод, задача), либо с вариантами значений величин. Более подробно структура классов описана в разделе 3.5.

3.5. Основные классы базы знаний

Для организации доступа к базе данных была построена библиотека, обеспечивающая доступ к экземплярам сущностей, хранящимся в базе данных, посредством ADO.NET Entity Framework. Этот подход обеспечил возможность обращения к базе данных как к набору взаимосвязанных коллекций, хранящих экземпляры классов, эквивалентных сущностям базы данных. Основным классом для доступа к базе знаний является автоматически сгенерированный класс-модель *EscienceModelContainer*.

3.5.1. Класс EscienceModelContainer

Данный класс обеспечивает доступ к набору коллекций объектов, соответствующих основным таблицам базы данных. Коллекции типа *ObjectSet* при этом представляются в виде свойств (*Property*) данного класса. Ниже перечислены основные свойства, обеспечивающие доступ к классам, соответствующим элементам базы знаний (см. рис. 3.4):

- FileExtractorSet,
- FileTypeSet,
- LayerClassSet,
- LayerEntityParameterSet,
- LayerEntityParameterClassSet,
- LayerEntitySet,
- ParameterSet ,

- QualityValueClassSet,
- QualityValueSet,
- SelectableValueDictionarySet,
- ValueSet.

3.6. Реализация прямой работы с онтологией

В результате исследований, проведенных на предыдущем этапе данного проекта, была выбрана реализация прямого доступа к онтологической структуре с использованием API программного средства Pellet (вариант RunLib). Реализуемый данным модулем интерфейс включает в себя базовые методы работы с онтологической структурой, соответствующие предъявленным функциональным требованиям. Ниже перечислены основные методы (приведена сигнатура методов на псевдокоде, объясняющая состав основных параметров методов).

CreateSession() – создание пользовательской сессии. Метод возвращает строковый идентификатор сессии.

AddOWLModel(<ID сессии>, <онтология>) – расширение онтологической структуры в рамках пользовательской сессии за счет добавления дополнительных элементов, которое задается на языке OWL в форме отдельной онтологии с возможными ссылками на существующие элементы.

ExecuteQuery(<ID сессии>, <запрос>) – метод позволяет выполнить запрос к онтологической структуре, расширенной в рамках текущей пользовательской сессии. Запрос задается на языке SPARQL, ответ представляет собой строку, содержащую результаты в формате XML.

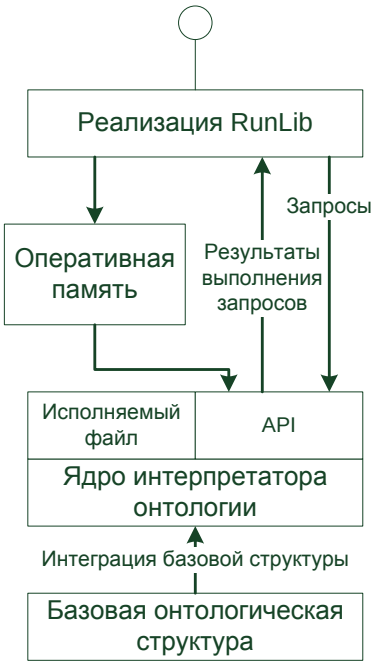

Рисунок 3.5 – Общая схема взаимодействия реализаций с интерпретатором онтологий

На рис. 3.5 представлена общая схема взаимодействия реализации RunLib с интерпретатором онтологии. Следует отметить, что важным отличием реализаций является то, что для хранения сессионной информации (онтологического расширения) отклоненная реализация RunExe использует файловую систему, в то время как RunLib не требует записи дополнительных файлов на диск – все операции производятся сервисом в оперативной памяти. Этот подход обеспечивает меньшее время отклика компонента при работе с наиболее важными операциями доступа к онтологической структуре.

Использование интерпретатора онтологии Pellet не является обязательным. Он может быть заменен на произвольное программное решение эквивалентной функциональности, причем вследствие стандартности используемых языков представления онтологии и запросов к ней такая замена не является сложной операцией. Так, в состав реализации RunExe был включен альтернативный интерпретатор ARQ [7].

На рис. 3.6 представлена диаграмма основных классов реализации RunLib. Важнейшим классом, реализующим заданный интерфейс, в данном случае является PelletService.

Рисунок 3.6 – Структура классов реализации RunLib

Используя более простую схему взаимодействия, реализация RunExe содержит значительно меньше классов. К ее основным структурным элементам относятся интерфейс IQueryProcessor, классы PelletQueryProcessor и ArqQueryProcessor, реализующие указанный интерфейс для интерпретаторов онтологии Pellet и Arq соответственно.

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 меньше 256 МБ, на котором должна быть установлена платформа .NET (версии 3.5 и выше) либо 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

Использование библиотеки доступа к базе данных осуществляется непосредственно из программного кода, требующего использования знаний, хранимых в базе. При этом основным способом доступа в рамках текущей версии компонента является статическое поле Model класса Easis.iKnow.EntytyAccess.EntityStorageDb, доступ к экземпляру базы которого реализован в соответствии с шаблоном «одиночка» (singleton). При этом полями класса являются коллекции, соответствующие основным сущностям модели базы (см. разделы 3.4–3.5).

Установка и настройка работы сервиса реализации RunLib осуществляются с помощью визуального мастера, запускаемого от имени администратора сервера Glassfish. Для корректной инсталляции сервиса необходимо наличие файла с дистрибутивом Pellet.war. Более подробно процесс инсталляции сервиса рассмотрен в документации сервера Glassfish.

Взаимодействие с web-сервисом осуществляется с использованием программных средств, доступных в составе большинства современных языков высокого уровня. В большинстве случаев это взаимодействие скрыто от разработчика прикладного ПО с помощью прокси-класса, реализующего идентичный сервису интерфейс.

6. ВХОДНЫЕ ДАННЫЕ

Входные данные в процессе работы с библиотекой доступа к базе данных формируются программным способом при вызове соответствующих методов, реализованных классами в составе данного модуля.

В табл. 6.1 приведено описание основных типов входных данных, используемых методами реализованного интерфейса web-сервиса работы с онтологической структурой базы знаний.

Таблица 6.1

Входные данные сервиса

Метод	Входные данные
CreateStession	–
AddOWLModel	Идентификатор сессии: строка; Текст расширения на языке OWL: строка
ExecuteQuery	Идентификатор сессии: строка; Текст запроса на языке SPARQL: строка

7. ВЫХОДНЫЕ ДАННЫЕ

Выходные данные в процессе работы с библиотекой доступа к базе данных формируются программным способом при вызове соответствующих методов, реализованных классами в составе данного модуля.

В табл. 7.1 приведено описание основных типов выходных данных, используемых методами реализованного интерфейса web-сервиса работы с онтологической структурой базы знаний.

Таблица 7.1

Выходные данные сервиса

Метод	Выходные данные
CreateStession	Идентификатор сессии: строка
AddOWLModel	–
ExecuteQuery	Результат выполнения запроса в формате XML: строка

ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Бухановский А.В., Ковальчук С.В., Марьин С.В. Интеллектуальные высокопроизводительные программные комплексы моделирования сложных систем: концепция, архитектура и примеры реализации // Изв. вузов. Приборостроение. 2009.– №10. с. 5-24.

[2] Pellet: OWL 2 Reasoner for Java [http://clarkparsia.com/pellet/].

[3] OWL Web Ontology Language Overview [http://www.w3.org/TR/owl-features/].

[4] Microsoft SQL Server Compact 3.5 [http://www.microsoft.com/sqlserver/2005/en/us/compact.aspx].

[5] Платформа ADO.NET Entity Framework [http://msdn.microsoft.com/ru-ru/library/bb399572.aspx].

[6] ORCA [http://www.thch.uni-bonn.de/tc/orca/].

[7] ARQ – A SPARQL Processor for Jena [http://jena.sourceforge.net/ARQ/].

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

API	Application programming interface (Ьнтерфейс программирования приложений)
DFT	Метод функционала плотности
WCF	Windows Communication Foundation (программный фреймворк)
WF	Workflow (поток заданий)
XML	eXtensible Markup Language (расширяемый язык разметки)
БЗ	База знаний
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
	изме-ненных	заме-ненных	но-вых	Аннули-рован-ных					