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

КОМПОНЕНТ ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЕМ CLAVIRE/GINGER

ОПИСАНИЕ ПРОГРАММЫ

ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.80066-06 13 21-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.
"29" 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 21-ЛУ

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

КОМПОНЕНТ ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЕМ CLAVIRE/GINGER

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ. 80066-06 13 21

ЛИСТОВ 27

2011

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

АННОТАЦИЯ

Документ содержит описание программного компонента взаимодействия с пользователем CLAVIRE/Ginger (Graphical Interface for Grid Environment Resources) RU.СНАБ.80066-06 01 21. Программный компонент реализует визуальную оболочку (доступную посредством web-браузера), предоставляющую пользователю возможность интерактивной работы с ресурсами, доступными в рамках МИТП, в процессе построения и использования композитных приложений, сформированных в терминах соответствующей предметной области, и использующих доступные платформе прикладные сервисы для решения задач компьютерного моделирования.

Программный компонент взаимодействия с пользователем CLAVIRE/Ginger разработан в ходе выполнения «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».

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Программный компонент взаимодействия с пользователем CLAVIRE/Ginger RU.СНАБ.80066-06 01 21 (далее Ginger) – визуальная оболочка (доступная посредством web-браузера), предоставляющая пользователю возможность интерактивной работы с ресурсами, доступными в рамках ПАК, в процессе построения и использования композитных приложений, сформированных в терминах соответствующей предметной области, и использующих доступные платформе прикладные сервисы для решения задач компьютерного моделирования.

Главной функцией компонента Ginger является предоставление доступа к функциональным возможностям МИТП CLAVIRE. Графический интерфейс пользователя имеет особое значение среди компонентов МИТП, так как именно с его помощью происходит управление комплексом в целом.

Компонент Ginger работает в качестве сетевого web-клиента, агрегирующего функциональные возможности распределенных подсистем МИТП, предоставляющих интерфейсы в виде web-сервисов.

Компонент Ginger разработан в виде «тонкого клиента» с использованием языка программирования C# версии 3.0 с применением ряда специализированных библиотек. Ginger функционирует на всех операционных системах, на которых может быть установлена платформа Microsoft Silverlight версии 4 и выше или Mono/Moonlight.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Компонент Ginger представлен в виде набора экранных форм и стандартных (для платформы Silverlight) элементов ввода–вывода. Основным назначением Ginger является предоставление пользователю возможности прямого манипулирования экранными объектами, с обеспечением максимально возможных простоты и удобства и сокрытием деталей взаимодействия с остальными компонентами МИТП CLAVIRE. Ginger организован на основе классической оконной модели построения графических интерфейсов ввиду повсеместного ее использования и интуитивного понимания широким кругом пользователей концепций взаимодействия с ней. Поскольку Ginger агрегирует подмножества элементов взаимодействия с внешними системами для всего комплекса, логически связные сущности в рамках Ginger объединяются в «проекты».

2.1. Решаемые задачи

1) Ginger обеспечивает возможность работы пользователя с МИТП CLAVIRE посредством графической оболочки в web-браузере, предоставляя доступ ко всем функциональным характеристикам с максимальными простотой и удобством.
2) Ginger обеспечивает поддержку работы с компонентом интеллектуальной поддержки принятия решений (CLAVIRE/iKnow) пользователя в форме диалога с последовательным уточнением требований пользователя к вычислениям, а также специфических критериев предметной области.
3) Ginger скрывает от пользователя специфику работы с отдельными пакетами, давая ему, однако, возможность тонко настраивать их вручную по требованию.
4) Ginger обеспечивает взаимодействие с внешними модулями и программными средствами, а именно с сервисами удаленной визуализации и мониторинга.
5) Ginger функционирует в качестве сетевого web-клиента, агрегирующего функциональность распределенных подсистем МИТП, предоставляющих интерфейсы в виде web-сервисов.

2.2. Функциональные возможности

Ginger обеспечивает покрытие следующей функциональности.

1) Взаимодействие через графическую оболочку, обеспечивающую единый подход к вводу и выводу разнородных данных.
2) Однократная авторизация пользователя без необходимости многократного ввода паролей или предоставления сертификатов.
3) Создание, сохранение и закрытие рабочего проекта.
4) Валидация значений вводимых параметров.
5) Выдача сообщений об ошибках и предупреждений.
6) Формирование вычислительного задания с использованием знаний, получаемых от компонента CLAVIRE/iKnow интеллектуальной поддержки пользователя, которое состоит из следующих этапов.

- Запуск сформированного задания на исполнение (путем обращения к компоненту формирования и исполнения workflow).
- Мониторинг исполнения запущенного задания (путем обращения к компоненту автоматизации удаленного исполнения заданий).
- Отображение информации об узлах выполнения задачи на карте мира.

- Останов выполняющегося задания (путем обращения к компоненту формирования и исполнения workflow).
- Передача исходных файлов в удаленное хранилище (путем обращения к компоненту доступа к данным).
- Обзор удаленного хранилища данных, получение результатов и их визуализация (автоматический запуск внешних локальных визуализаторов) путем обращения к компоненту доступа к данным.
- Визуализация полученных в ходе расчетов файлов.
- Просмотр файлов различных форматов: текстовых (любые файлы) и графических (JPG, PNG, GIF, BMP).

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

Ginger реализован в форме web-клиента, представляющего собой программу, выполняющуюся в интернет-браузере пользователя (в рамках подключаемого модуля Silverlight). Достоинствами такого подхода являются: отсутствие необходимости установки приложений, хранение всех данных на удаленных серверах, низкие требования по мощности к машине пользователя.

Данные в Ginger организуются в виде проектов. Проект представляет собой сущность, сохраняющую входные данные и информацию о запуске пакетов. Информация о пакете и входные данные совместно с информацией о запуске пакета (например, id задачи) и результатах выполнения представляют собой сущность *Activity* (задача). Таким образом, один набор данных может быть членом сразу нескольких *Activity*, но каждое «исполнение» является членом лишь одной *Activity*. Создание сущностей *Activity* производится автоматически по мере создания пользователем взаимодействий между входными данными и пакетами. Одной из важных характеристик проекта является его принадлежность определенному пользователю.

3.1. Общий порядок взаимодействия пользователя с МИТП через Ginger

Взаимодействие пользователя с МИТП через Ginger осуществляется манипуляцией отображаемыми графическими элементами. С помощью этих манипуляций пользователь может получать доступ к основным элементам функциональности МИТП CLAVIRE. Графическая среда реализована в виде проблемно-ориентированной оболочки со

структурой, сходной со структурой большинства современных интегрированных сред разработки (IDE) для различных областей применения (общий вид главного окна показан на рис. 3.1), например, Microsoft Visual Studio или Eclipse.

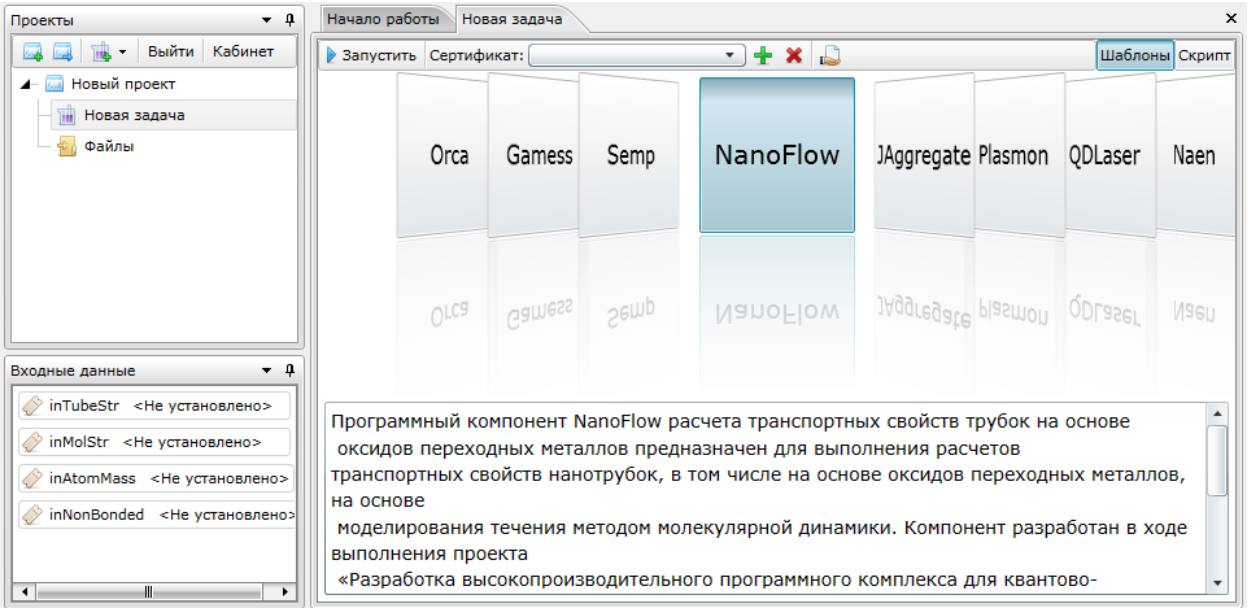


Рисунок 3.1 – Вид главного окна интерфейса человеко-компьютерного взаимодействия

Главное окно состоит из трех областей.

1) *Дерево проектов* – область представления проектов, задач, запусков и файлов пользователя с возможностями навигации и редактирования.
2) *Область документов*: содержит вкладки для каждого из открытых документов (например, для документов задач), в каждом из которых, в зависимости от типа, возможно производить свои действия.
3) *Область данных*: предназначена для связывания файлов, загруженных пользователем, и данных, требуемых пользователем WF.

Описанные области находятся в специальных контейнерах, которые можно перемещать по экрану и «приклеивать» к различным его сторонам (рис. 3.2). Это предоставляет пользователю возможности по свободной компоновке интерфейса.

Помимо главного окна web-интерфейс может создавать дочерние всплывающие окна для выполнения различных действий (например, прохождения опроса интеллектуальной системы). Для взаимодействия с пользователем web-интерфейс использует стандартные компоненты: кнопки, контекстные меню, поля ввода, панели докинга и т.д.

Для более удобной организации работы пользователя в web-интерфейсе применяется модель проектов, задач и запусков. Эта логика в основном отражена в области дерева проектов (рис. 3.3).

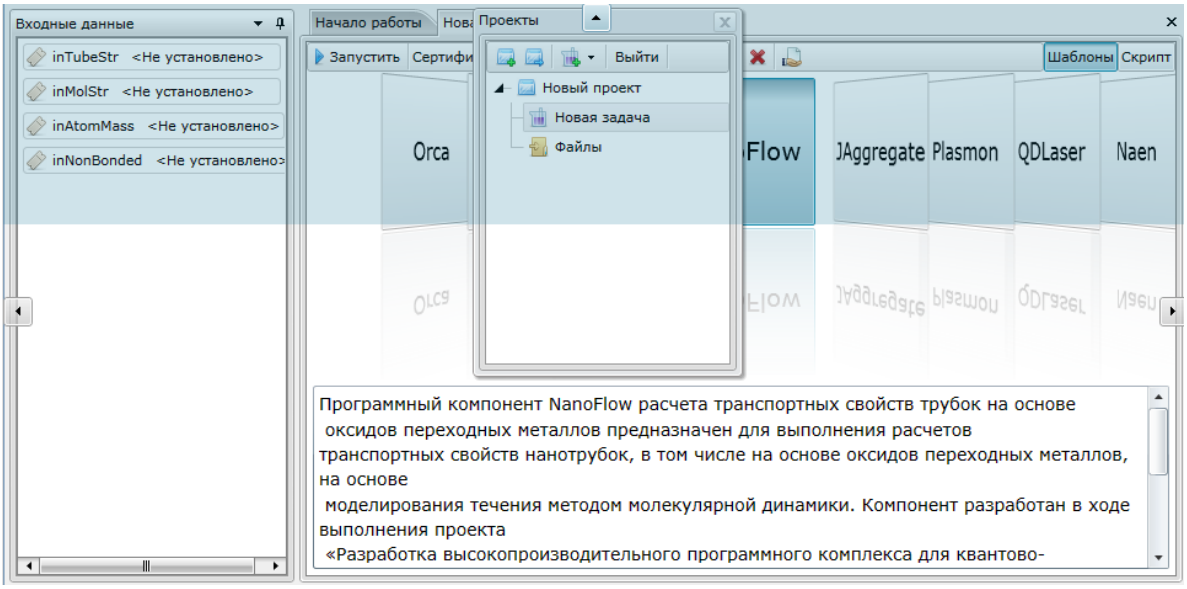


Рисунок 3.2 – Процесс перемещения контейнера к верхней границе экрана

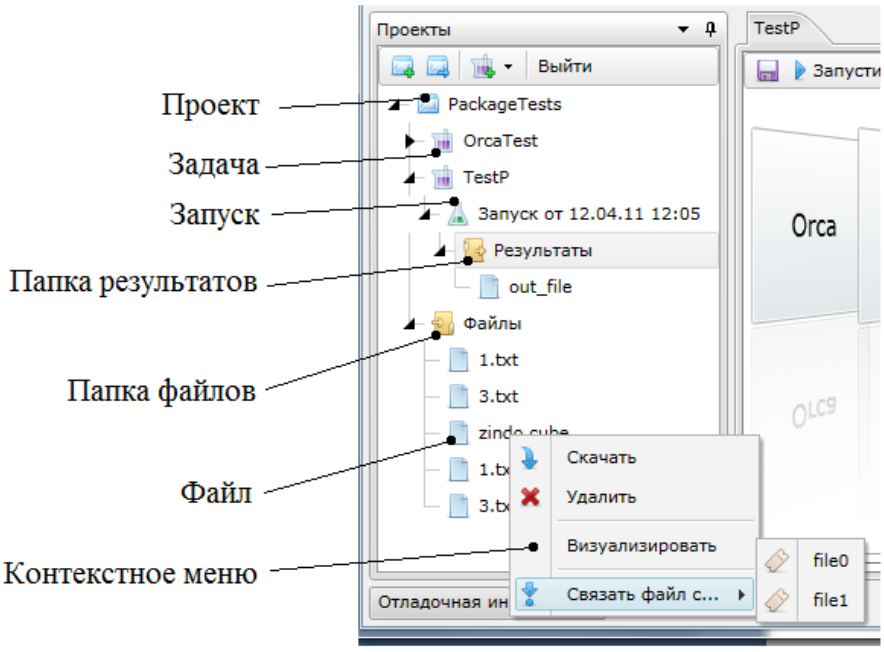


Рисунок 3.3 – Элементы дерева проектов

Проект – это объединение нескольких вычислительных задач и загруженных пользователем входных файлов. В один момент в web-клиенте может быть загружен только один корневой проект.

Задача – это описание вычислительной задачи на языке EasyFlow с возможностью неоднократного запуска. Задача может быть создана как прямым редактированием скрипта, так и выбором из существующих шаблонов или прохождением опроса интеллектуальной системы (см. ниже). Задачи группируются как дочерние элементы корневого проекта в дереве проектов.

Запуск – это сущность, фиксирующая факт запуска задачи. Каждая задача может быть запущена несколько раз (например, после изменения некоторых параметров запуска). Запуски группируются как дочерние элементы задачи в дереве проектов.

Файл – сущность, представляющая реальный файл, находящийся в удаленном хранилище. Файлы группируются в папки. Проект содержит папку «Файлы», в которую пользователь может закачивать собственные входные данные. Любой запуск содержит папку «Результаты», в которую собираются выходные файлы данного запуска.

Каждый элемент дерева проектов имеет собственное контекстное меню, вызываемое нажатием на нем правой кнопки мыши. В меню содержатся действия того или иного элемента (создание, удаление, запуск, открытие и т.д.).

На рис. 3.4 представлена диаграмма использования Ginger с описанием возможных действий пользователя.

Ginger предоставляет пользователю необходимые инструменты для покрытия функциональных возможностей, указанных в разделе 2.2. Ниже описывается типовая последовательность действий при работе с системой.

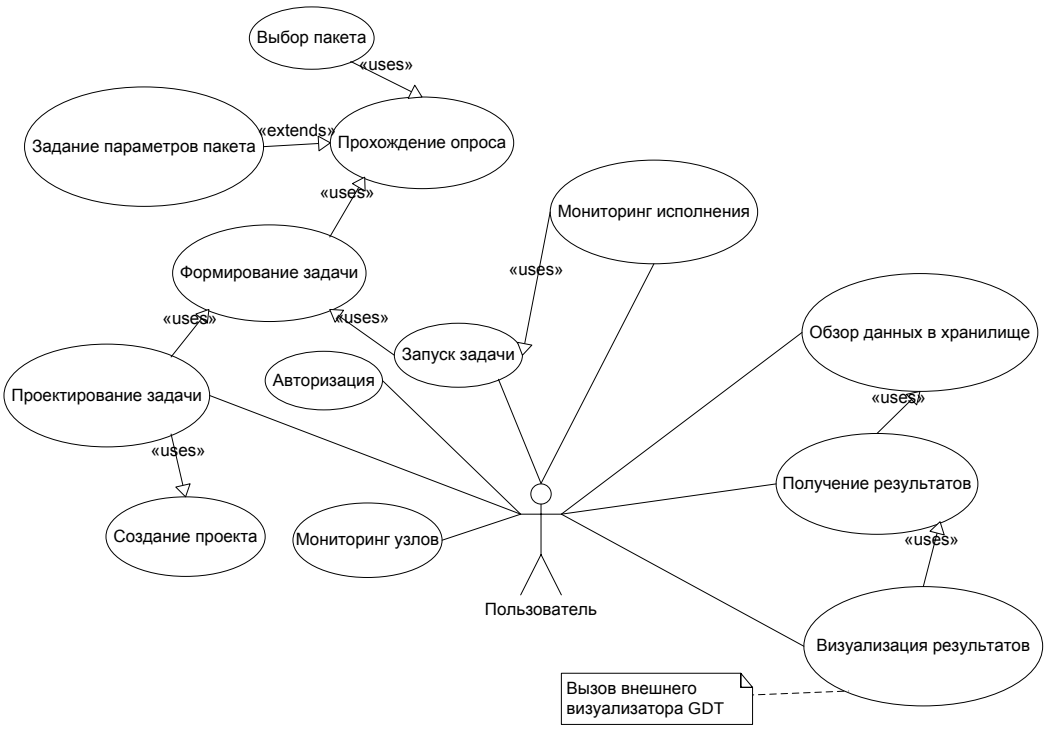


Рисунок 3.4 – Общая диаграмма использования Ginger

После загрузки содержимого в области отображения web-страниц браузера должно появиться окно авторизации web-интерфейса, в которое требуется ввести логин и пароль пользователя (см. рис. 3.5 – на этом и остальном рисунках элементы управления браузера намеренно не указаны, чтобы не осложнять восприятия). В случае установки флажка «Помнить меня» при следующем входе на страницу от пользователя не будет требоваться авторизация.

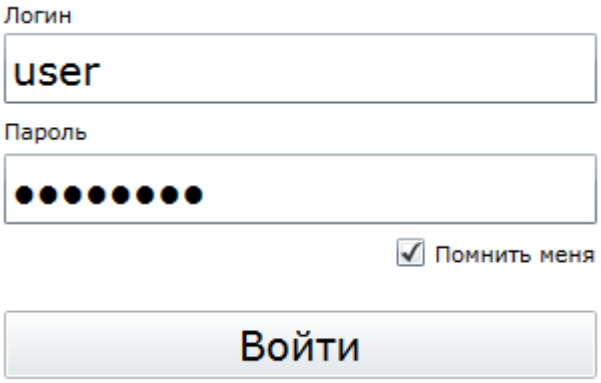


Рисунок 3.5 – Форма авторизации пользователя

В случае успешного входа в систему на странице браузера отобразится основное окно web-интерфейса (см. рис. 3.2).

Работа с системой начинается с создания нового проекта, например, щелчком правой кнопки мыши в области дерева проектов и выбора элемента «Создать проект». При этом в дерево проектов будет загружен новый пустой проект под именем «Новый проект» с пустой папкой «Файлы». При этом будет предложено изменить его имя на желаемое непосредственно в дереве проектов (рис. 3.6). После задания имени можно нажать клавишу Enter либо щелкнуть в любой области клиента: проект будет создан и автоматически сохранен на сервере (далее о сохранении проекта не упоминается, так как оно происходит автоматически при каждом действии по изменению проекта).

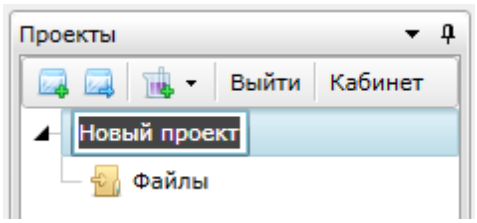


Рисунок 3.6 – Только что созданный проект

Созданный проект является пустым, и для продолжения работы требуется создать новую задачу. Для создания задачи в режиме опроса интеллектуальной системы следует

щелкнуть левой кнопкой мыши на изображении стрелочки рядом с кнопкой «Создать задачу» на панели инструментов дерева проектов (см. рис. 3.6).

В появившемся контекстном меню следует выбрать пункт «Дерево принятия решений». На экране появится дерево опроса интеллектуальной системы (рис. 3.7).

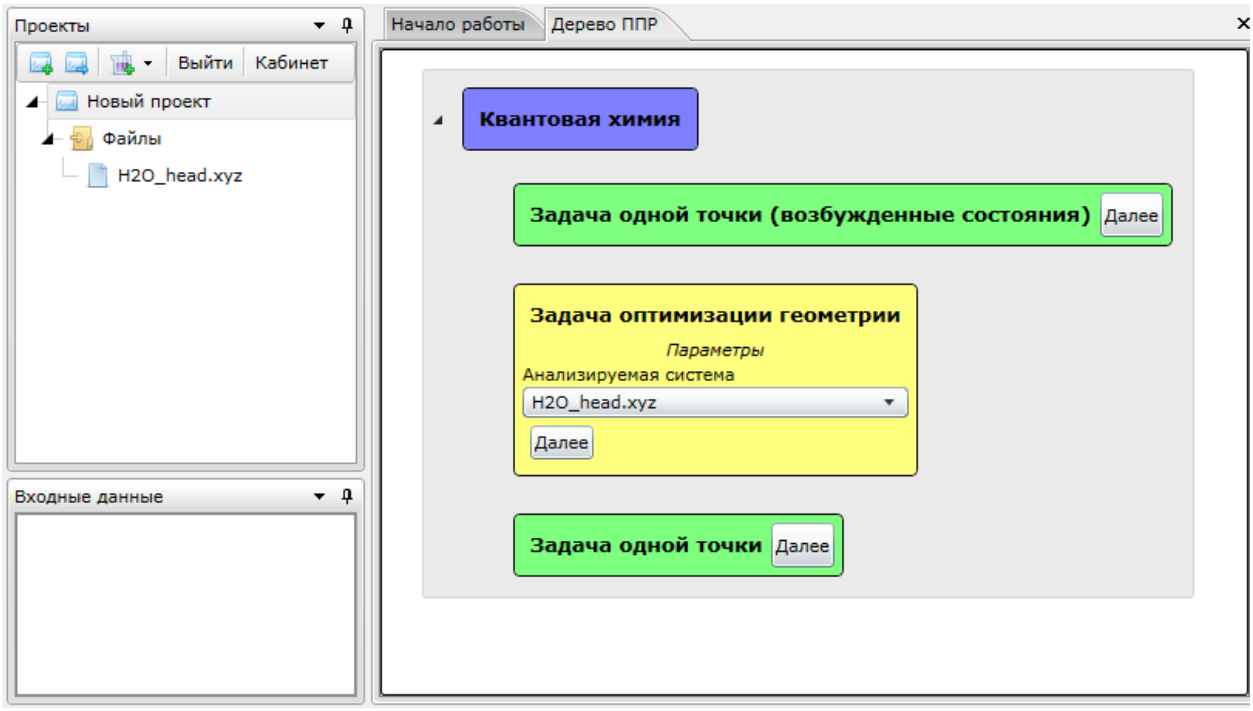


Рисунок 3.7 – Дерево поддержки принятия решений интеллектуальной системы

В случае успешного завершения работы с интеллектуальной системой пользователю предлагается сформированная цепочка вычислений в виде скрипта на языке EasyFlow, который соответствует выбранному системой решению (рис. 3.8).

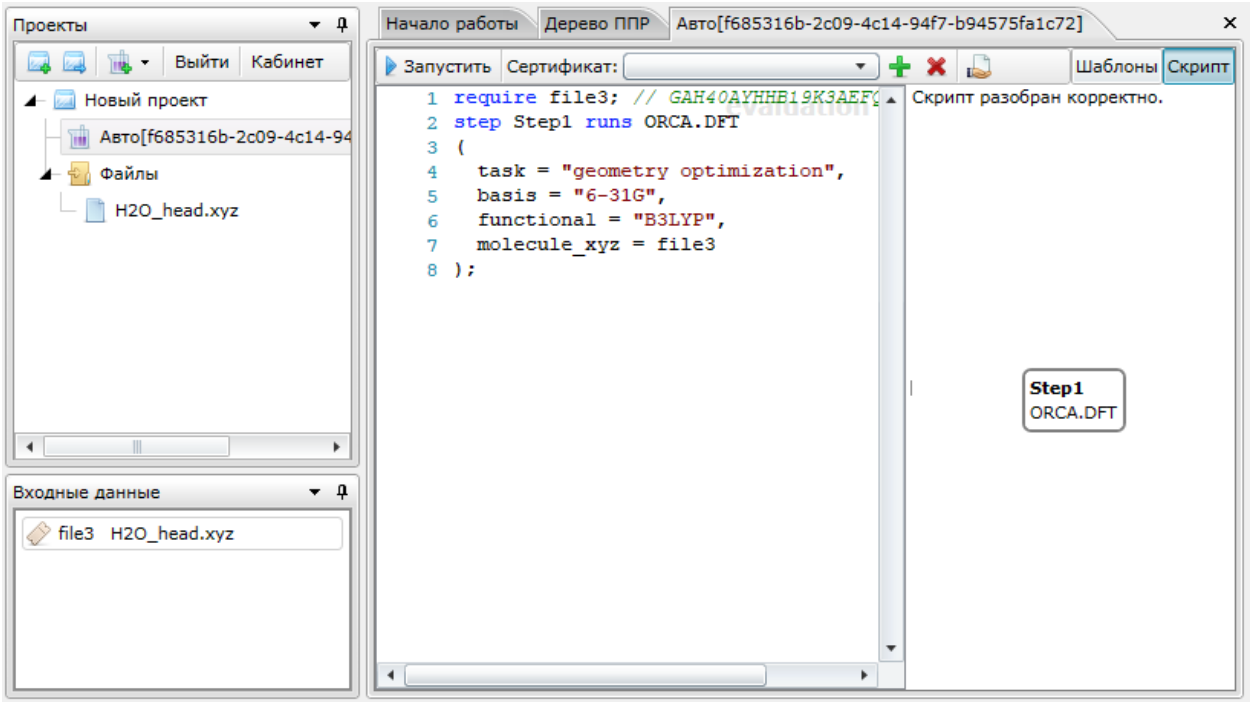
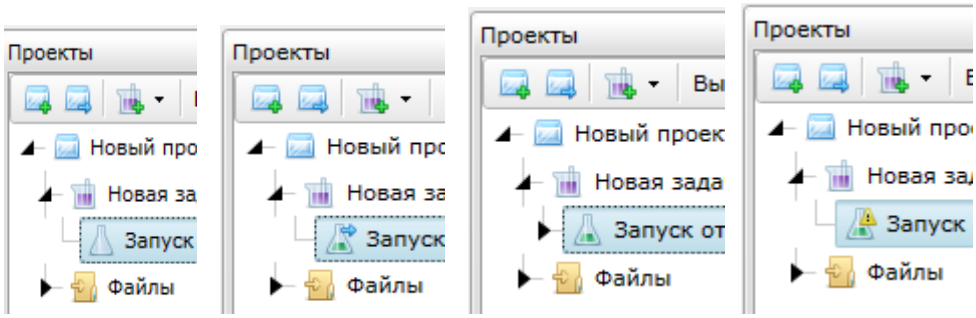


Рисунок 3.8 – Вид окна после опроса интеллектуально системы

На этом этапе становится возможным запуск построенной вычислительной цепочки путем щелчка правой кнопкой мыши на соответствующей задаче в дереве проектов и выбора пункта меню «Запустить…». После этого в дереве проектов под выбранной задачей появится элемент запуска с предложением отредактировать его имя. Запуск начнется автоматически (одну и ту же задачу можно запускать повторно неограниченное число раз).

Для отображения разных состояний задачи у элемента запуска могут отображаться различные значки, обозначающие текущий статус (рис. 3.9): не запущена (а), запускается (б), выполнена успешно (в), выполнена с ошибкой (г).



(а) (б) (в) (г)

Рисунок 3.9 – Статусные иконки задачи

В случае успешного запуска задачи под элементом запуска появится папка «Результаты», элементами которой будут являться выходные файлы запуска.

3.3. Общая архитектура

По своей сути Ginger является сверхтонким клиентом (ultra-thin client) для доступа к функциональности основных компонентов МИТП CLAVIRE, представленных в виде удаленных web-сервисов. Это означает, что подавляющее большинство функций, присущих комплексу в целом, выполняется на удаленных серверах, а взаимодействие Ginger с остальными подсистемами комплекса осуществляется в виде запросов к представляющим их сервисам (web service request) и получения ответных сообщений (web service response). Таким образом, Ginger является частью распределенного приложения и в крупном масштабе может быть представлен как имеющий двухуровневую архитектуру: первый уровень представляет собой собственно графическую оболочку, а второй – совокупность удаленных web-сервисов.

Помимо web-сервисов, входящих в состав иных подсистем, Ginger предоставляет собственные специфические сервисы, обеспечивающие, например, работу с проектами. Общий вид высокоуровневой архитектуры GINGER приведен на рис. 3.10.

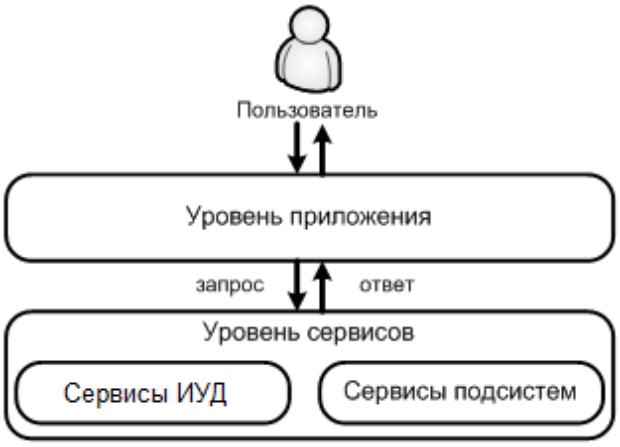


Рисунок 3.10 – Уровни интерфейса человеко-компьютерного взаимодействия

Уровень приложения представляет собой конкретную реализацию графического интерфейса, архитектура которого описана ниже.

3.4. Архитектура уровня приложения

Пользователь работает с МИТП CLAVIRE на уровне приложений Ginger (рис. 3.10). Для этого ему предоставляется графический интерфейс, через который он получает доступ к функциональным возможностям комплекса, используя стандартные компоненты графического интерфейса (кнопки, текстовые поля ввода и т.п.).

Для реализации Ginger использован модульный подход, основанный на применении библиотеки Microsoft Prism совместно с библиотекой Microsoft MEF. Связка Prism+MEF представляет собой набор инструментов для создания композитных пользовательских интерфейсов, состоящих из слабосвязанных динамически загружаемых модулей. Это позволяет создавать расширяемую архитектуру пользовательского интерфейса, основанную на понятии плагинов (plugins) – отдельных программных сущностей (модулей), расширяющих или видоизменяющих функциональность основного приложения. Примерами данных приложений могут служить такие среды разработки, как Microsoft Visual Studio или Eclipse, которые сами по себе являются лишь каркасом с общей функциональностью, расширяемой за счет подключаемых модулей – плагинов.

Таким же образом приложение Ginger представляет собой основное приложение и набор библиотек Silverlight, которые можно разделить на две группы: библиотеки ядра и библиотеки модулей. Библиотеки ядра содержат основные классы, позволяющие компоновать модули и создавать пользовательский интерфейс посредством взаимодействия различных менеджеров (например, ShortcutManager, MenuManager – см. ниже). Библиотеки модулей представляют собой отдельные «куски» функциональных возможностей и представления (графического), которые встраиваются в основное приложение Ginger за счет использования классов библиотеки ядра. Основное приложение Ginger является лишь контейнером для модулей и графических элементов и, по сути, неспособно выполнять какие-либо функции, кроме загрузки модулей и отображения корневого визуального элемента.

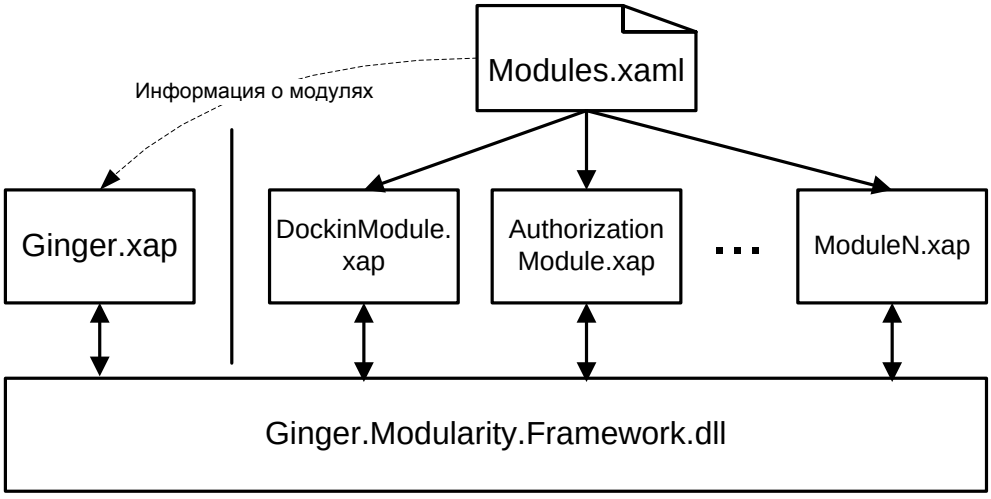


Рисунок 3.11 – Схема модульной архитектуры Ginger

На рис. 3.11 показана общая схема модульной архитектуры Ginger. Ginger.xap – это основное хост-приложение, выполняющее инициализацию среды и загрузку модулей, а

также содержащее корневой визуальный элемент GingerShell, включающий в себя регион (см. документацию Prism) «Ginger.ShellMain», в который загружается главный модуль (в текущей поставке это модуль DockingModule, см. ниже).

Приложение Ginger.xap ссылается на библиотеку Ginger.Modularity.Framework.dll (описана ниже), как и все подключаемые модули. Однако приложение Ginger.xap не имеет сведений о том, какие модули будут в него загружены при запуске. Информация об этом хранится в специальном файле Modules.xaml, который должен находиться на сервере в корневой папке сайта. При запуске основное приложение скачивает этот файл и загружает перечисленные в нем модули (в виде xap-файлов) в порядке зависимостей между ними. Этот файл можно редактировать, изменяя вид и функциональность основного приложения Ginger без перекомпиляции кода.

Примерный вид файла Modules.xaml приведен в листинге 3.1.

Листинг 3.1. Пример файла Modules.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<Modularity:ModuleCatalog
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:sys="clr-namespace:System;assembly=mscorlib"
    xmlns:Modularity="clr-namespace:Microsoft.Practices.Prism.Modularity;
                      assembly=Microsoft.Practices.Prism">
    <Modularity:ModuleInfoGroup>
        <Modularity:ModuleInfo Ref="Ginger.Modules.Authorization.xap"
                               ModuleName="AuthorizationModule"/>

        <Modularity:ModuleInfo Ref="Ginger.Modules.Docking.xap"
                               ModuleName="DockingModule">
            <Modularity:ModuleInfo.DependsOn>
                <sys:String>AuthorizationModule</sys:String>
            </Modularity:ModuleInfo.DependsOn>
        </Modularity:ModuleInfo>

        <Modularity:ModuleInfo Ref="Ginger.Modules.ProjectTree.xap"
                               ModuleName="ProjectTreeModule">
            <Modularity:ModuleInfo.DependsOn>
                <sys:String>DockingModule</sys:String>
            </Modularity:ModuleInfo.DependsOn>
        </Modularity:ModuleInfo>

        <Modularity:ModuleInfo Ref="Ginger.Modules.DebugViewer.xap"
                               ModuleName="DebugViewerModule">
            <Modularity:ModuleInfo.DependsOn>
                <sys:String>DockingModule</sys:String>
            </Modularity:ModuleInfo.DependsOn>
        </Modularity:ModuleInfo>

        ...

    </Modularity:ModuleInfoGroup>
</Modularity:ModuleCatalog>
```

Этот файл содержит перечень модулей, которые должно загрузить основное приложение Ginger, а также зависимости между модулями, чтобы определить порядок загрузки. Каждый модуль должен находиться в отдельной сборке Silverlight с расширением xap, однако один xap-файл может содержать неограниченное количество

модулей. Для указания на то, что требуется загрузить какой-либо модуль, необходимо в файл Modules.xaml добавить тэг <ModuleInfo> со следующими атрибутами: Ref – относительный или абсолютный Uri xap-файла на сервере приложения, содержащего модуль; ModuleName – имя загружаемого модуля (должно совпадать с именем класса, представляющего модуль – см. ниже).

Для указания зависимостей между модулями следует указать в качестве дочерних элементов тэга <ModuleInfo> перечень тэгов <ModuleInfo.DependsOn>, внутри которого должна содержаться строка вида: <sys:String>ModuleName</sys:String>, где ModuleName – имя родительского модуля. В этом случае модуль, зависящий от других модулей, не будет загружен, пока не загружены все его зависимости.

Отдельно стоит сказать, что к модулям предъявляется общее требование независимости друг от друга, которое, однако, может нарушаться в случае, если один модуль должен использовать функциональность другого. В этом случае рекомендуется при написании модуля, от которого, как предполагается, будут зависеть другие модули, разбивать его на две сборки: интерфейсную, на которые будут ссылаться зависимые модули, и главную – содержащую реализацию. Инстанциирование конкретных классов, соответствующих интерфейсам из интерфейсной сборки, должно быть поручено MEF.

3.5. Основные интерфейсы и классы библиотеки ядра

Основной библиотекой компонента Ginger является Ginger.Modularity.Framework.dll, которая содержит основные классы для создания каркаса модульного приложения и поддержки основных действий с пользовательским интерфейсом (например, управление «горячими клавишами», контекстными меню, всплывающими окнами и т.д.).

Библиотека содержит интерфейсы и реализацию логики для следующих функциональных возможностей:

- авторизация пользователя (Authorization);
- управление командами пользовательского интерфейса (Commanding);
- событийное взаимодействие между модулями (Eventing);
- локализация (Localization);
- журналирование (Logging);
- управление меню (Menus);
- управление модулями (Modules);

- управление пользовательскими оповещениями (Notifications);
- управление всплывающими окнами (Popups);
- управление настройками приложения (Settings);
- управление «горячими клавишами» (Shortcuts);
- управление хранением данных пользователя (UserStorage);
- управление интерфейсом (VisualHelpers).

Этот неполный перечень содержит многие необходимые для реализации пользовательского интерфейса техники и подходы, основные из них описаны ниже.

3.5.1. Интерфейс IAuthorizationManager

Интерфейс для классов, реализующих логику авторизации пользователей. Класс-реализатор этого интерфейса (AuthorizationManager) находится в модуле Ginger.Modules.Authorization и реализует логику авторизации по логину и паролю.

Открытые члены

- void *ShowAuthorizationView*() – показать вид авторизации в главном регионе приложения;
- void *BeginAuthorize*(IAuthorizationToken authToken, object userState) – начать процесс авторизации с использованием данных авторизации пользователя (*authToken*) и пользовательским объектом состояния (*userState*);
- EventHandler<AuthorizeCompletedEventArgs> *AuthorizedCompleted* – событие, возникающее при завершении процесса авторизации;
- void *BeginUnauthorize*(object userState) – начать процесс деавторизации с пользовательским объектом состояния (*userState*);
- EventHandler<UnauthorizeCompletedEventArgs> *UnauthorizeCompleted* – событие, возникающее при завершении процесса деавторизации;
- bool *IsAuthorized* – возвращает текущее состояние авторизованности пользователя (true – пользователь авторизован, false – нет);
- IAuthorizationInfo *AuthorizationInfo* – возвращает информацию об авторизованном пользователе, если пользователь авторизован, или null, если пользователь не авторизован.

3.5.2. Интерфейс ICleanupManager

Интерфейс для классов, способных уничтожать связанные с каким-либо объектом визуальные элементы при прекращении существования последнего. Пример: при закрытии проекта требуется закрыть все окна и вкладки, связанные с ним.

Открытые члены

- void *Subscribe*(Action<object> handler) – подписаться на событие очистки;
- void *Unsubscribe*(Action<object> handler) – прекращение подписки на событие очистки;
- void *Cleanup*(object viewModel, bool cleanupChildren) – уничтожить объекты интерфейса, связанные с переданным объектом (*viewModel*). Параметр *cleanupChildren* указывает, требуется ли уничтожать объекты для дочерних элементов.

3.5.3. Интерфейс ICommandManager

Интерфейс для классов, способных управлять поименованными параметризованными командами. Используется для создания единого механизма управления командами в рамках всего приложения. Класс-реализатор: CommandManager.

Открытые члены

- ICommandManager *RegisterCommand*(string commandName, ICommand command, CanExecuteMode canExecuteMode) – зарегистрировать команду *command* под именем *commandName* с режимом вычисления способности выполнения *canExecuteMode*;
- ICommandManager *UnregisterCommand*(string commandName, ICommand command) – удалить регистрацию команды *command* под именем *commandName*;
- bool *CommandRegistered*(string commandName) – проверяет, существует ли команда с именем *commandName*;
- ICommand *this*[string commandName] – возвращает команду по имени *commandName*;
- bool *Execute*(string commandName, object param) – исполняет команду с именем *commandName* и параметром *param*;
- bool *CanExecute*(string commandName, object param) – проверяет, может ли команда с именем *commandName* исполниться с параметром *param*;
- void *InvalidateCommands*() – посылает всем командам сигнал обновить флаг возможности исполнения;

- event EventHandler<CommandEventArgs> *BeforeCommandExecute* – событие, возникающее перед выполнением команды;
- event EventHandler<CommandEventArgs> *AfterCommandExecute* – событие, возникающее после выполнения команды;
- event EventHandler<CommandEventArgs> *CanExecuteChanged* – событие, возникающее при изменении способности команды к выполнению.

3.5.4. Класс FocusManager

Статический класс, управляющий передачей и установкой фокуса приложения.

Открытые члены

- void *SetInject*(UIElement element, bool value) – зарегистрировать графический элемент *element* как обработчик изменения фокуса;
- bool *GetInject*(UIElement element) – проверить, является ли графический элемент *element* обработчиком изменения фокуса;
- object *GetFocusedElement*() – возвращает текущий элемент, имеющий на себе фокус;
- void *EnsureFocus*(Control control) – фокусирует элемент *control* и при необходимости – плагин Silvelright;
- void *FocusPlugin*() – устанавливает фокус для плагина Silverlight.

3.5.5. Интерфейс IMenuManager

Интерфейс для классов, реализующих глобальное поведение контекстных меню, их регистрацию и генерацию. Класс-реализатор: MenuManager.

Открытые члены

- void *RegisterAreaIfMissing*(string areaName) – регистрирует новую зону с контекстным меню под именем *areaName*;
- void *RegisterElementWithArea*(FrameworkElement element, string areaName) – привязывает элемент *element* к зоне *areaName*;
- IEnumerable<FrameworkElement> *GetAreaAttachedElements*(string areaName) – возвращает список элементов, привязанных к зоне *areaName*;
- IEnumerable<string> *GetAreasForElement*(FrameworkElement element) – возвращает список имен зон, к которым привязан элемент *element*;
- IEnumerable<string> *Areas* – возвращает список зарегистрированных зон;

- void *RegisterHandler*(Type elementType, string areaName, GenerateMenuHandler handler) – регистрирует обработчик события (*handler*) нажатия правой клавиши для элементов с типом *elementType*, находящихся в зоне *areaName*;
- IEnumerable<MenuItem> *GenerateContextMenu*(IEnumerable<FrameworkElement> clickedItems) – генерирует контекстное меню для переданных элементов (*clickedItems*), по которым осуществился щелчок кнопкой мыши;
- bool *HasHandlerFor*(Type elementType, string areaName) – проверяет, имеется ли обработчик для пары тип (*elementType*) – зона (*areaName*).

3.5.6. Интерфейс INotificationManager

Интерфейс для классов, реализующих оповещение пользователя о различных событиях. Класс-реализатор: NotificationManager.

Открытые члены

- void *RegisterNotificationType*(INotificationType notificationType) – регистрирует тип оповещения;
- IEnumerable<INotificationType> *NotificationTypes* – возвращает список зарегистрированных типов оповещений;
- ReadOnlyObservableCollection<INotification> *Notifications* – возвращает список активных оповещений;
- void *AddNotification*(INotification notification) – добавляет оповещение;
- event EventHandler<NotificationEventArgs> *NotificationAdded* – событие, возникающее при добавлении оповещения;
- void *RemoveNotification*(INotification notification) – удаляет оповещение;
- event EventHandler<NotificationEventArgs> *NotificationRemoved* – событие, возникающее при удалении оповещения.

3.5.7. Интерфейс IShortcutManager

Интерфейс для классов, управляющих работой «горячих клавиш». Класс-реализатор: ShortcutManager.

Открытые члены

- void *Register*(Shortcut shortcut, ICommand command) – регистрирует «горячую клавишу» *shortcut* с командой *command*;

- void *Unregister*(Shortcut shortcut) – удаляет регистрацию для «горячей клавши» *shortcut*;
- IEnumerable<IShortcutRegistration> *Shortcuts* – возвращает список зарегистрированных «горячих клавиш»;
- bool *HandleKeyEvent*(ModifierKeys modifierKeys, Key key) – обрабатывает событие о нажатии клавиш клавиатуры, пытается найти подходящую «горячую клавишу» и выполнить связанную с ней команду.

3.5.8. Класс UI

Статический класс-помощник для работы с графическим приложением.

Открытые члены

- bool *IsInDesignMode* – возвращает true, если приложение находится в дизайн-режиме (VisualStudio или Expression Blend), и false – в ином случае;
- bool *IsRunningOutOfBrowser* – возвращает true, если приложение запущено в режиме out-of-browser, и false – в ином случае;
- object *RootVisual* – возвращает текущий визуальный корневой элемент;
- void *Dispatch*(Action action) – выполняет действие *action* в потоке графического интерфейса с обработкой ошибок и журналированием.

3.5.9. Класс GingerModule

Корневой абстрактный класс для модулей приложения Ginger.

Открытые члены

- void *Initialize*() – инициализирует модуль;
- bool *Initialized* – возвращает true, если модуль был инициализирован, и false – в ином случае;

Защищенные члены

- void *InitializeModule*() – метод, который должен быть перекрыт в дочерних классах с логикой инициализации модуля.

3.6. Архитектура уровня сервисов

Как было указано в разделе 3.2, Ginger функционирует на основе вызова удаленных web-сервисов, предоставляющих интерфейсы к соответствующим

компонентам МИТП. На рис. 3.13 отражена общая схема взаимодействия Ginger с сервисами подсистем, указаны передаваемые запросы и получаемые ответы.

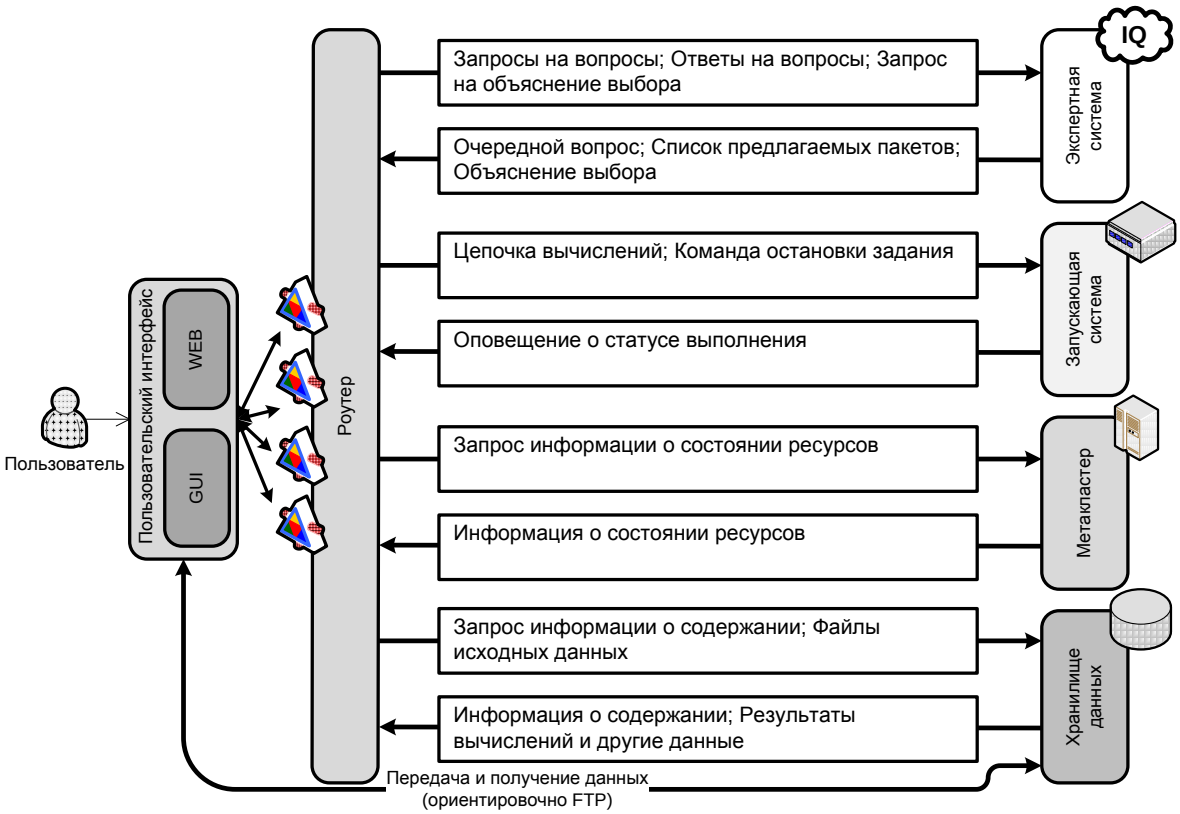


Рисунок 3.13 – Общая схема взаимодействия Ginger с сервисами и хранилищами данных

Для организации доступа к web-сервисам в архитектуру Ginger введен модуль *UiServiceClients*, который содержит прокси-классы для соответствующих сервисов. Они представляют собой «обертки» (wrapper) для клиента SOAP, с помощью которого осуществляются реальная посылка запросов и ожидание ответов. Использование прокси-классов позволяет работать с web-сервисами как с обычными классами. Важно, что для всех методов web-сервисов в прокси-классах реализуются их асинхронные версии, это позволяет делать их вызовы в фоновом режиме.

Важнейшими сервисами, используемыми Ginger, при выполнении проектов являются: *AiService*, *FlowSystemService*, *ExecutionService* и *DataService*. Однако они выходят за рамки данного документа, так как являются частями других компонентов МИТП и описаны в соответствующей программной документации.

В работе сервисов, относящихся непосредственно к Ginger (*ProjectService*), используется концепция web-сервисов «без состояния» (stateless web service), которые реагируют на одни и те же запросы одинаково, вне зависимости от истории предыдущих запросов. В связи с этим хранение данных (информация о пользователях, проектах,

запусках, данных и пр.) осуществляется с помощью БД под управлением MongoDB. По своей сути перечисленные сервисы являются интерфейсами управления данными в этой БД.

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Ginger должен обеспечивать взаимодействие трех классов программно-аппаратных средств: «сервер БД», «сервер приложений» (web-сервер, в рамках которого выполняются web-сервисы) и «клиент», к параметрам которых предъявляются следующие технические требования.

4.1. Требования к аппаратной совместимости

1. ***Сервер БД*:** персональный компьютер или узел кластера на базе процессора не ниже Pentium IV, свободное дисковое пространство – не менее 250 МБ, объем оперативной памяти – не менее 1024 МБ, сетевая аппаратура для соединения по TCP/IP-протоколу.
2. ***Сервер приложений*:** персональный компьютер или узел кластера на базе процессора не ниже Pentium IV, свободное дисковое пространство – не менее 250 МБ, объем оперативной памяти – не менее 2048 МБ, аппаратура для соединения по TCP/IP-протоколу.
3. ***Клиент*:** персональный компьютер на базе процессора – не ниже Pentium III, свободное дисковое пространство – не менее 150 МБ, объем оперативной памяти – не менее 512 МБ, аппаратура для соединения по TCP/IP-протоколу.

4.2. Требования к информационной и программной совместимости

1. ***Сервер БД*:** предназначен для функционирования на персональном компьютере с корректно установленной СУБД MongDB версии 1.8 и выше, исполняющейся в среде ОС, поддерживаемой данной СУБД (Windows не ниже Windows 2000, Linux, FreeBSD).
2. ***Сервер приложений*:** предназначен для функционирования на персональном компьютере в среде ОС Windows (не ниже Windows 2000) или Linux, с установленной средой .NET Framework версии 4 и выше (для Linux требуется реализация «Mono»).

Кроме того, требуется корректная предварительная установка следующих программных средств и библиотек:

- MongoDB C# Driver;
- Microsoft Internet Information Server версии 7.0 и выше.

3. ***Клиент*:** предназначен для функционирования на персональном компьютере в среде ОС Windows (не ниже Windows 2000) или Linux, с установленной платформой Silverlight 4 и выше (для Linux требуется реализация «Moonlight»).

5. ВЫЗОВ И ЗАГРУЗКА

Единственным средством доступа к МИТП для оператора является web-интерфейс, реализованный на базе технологии Microsoft Silverlight 4. Для его запуска требуются только браузер и соединение с Интернетом на скорости не ниже 128 Кбит/с.

Для вызова и загрузки web-интерфейса требуется ввести в строку браузера адрес, по которому развернута МИТП (например, <http://escience.ifmo.ru/WebClient>), и перейти по нему (рис. 5.1).

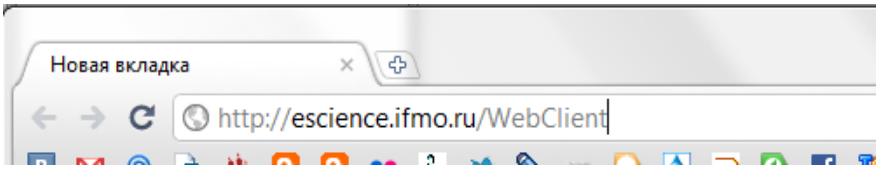


Рисунок – 5.1. Вызов графического web-клиента в браузере

После загрузки содержимого в поле web-страниц браузера должно появиться окно авторизации web-интерфейса, в которое требуется ввести логин и пароль пользователя (см. рис. 3.5). В случае установки флажка «Помнить меня» при следующем входе на страницу от оператора не будет требоваться авторизация.

В случае успешного входа в систему на странице браузера отобразится основное окно web-интерфейса (рис. 3.1).

6. ВХОДНЫЕ ДАННЫЕ

Для запуска и работы с Ginger не требуется специальных видов входных данных. В ходе работы ими могут являться любые входные файлы, соответствующие по формату запускаемому пакету (например, ORCA, GAMESS и т. п.), а также данные, вводимые

пользователем с клавиатуры по запросу МИТП. В случае несоответствия данных условиям их использования будет выдано соответствующее системное сообщение.

7. ВЫХОДНЫЕ ДАННЫЕ

В качестве выходных данных Ginger может предоставлять результаты расчетов, загруженные с удаленного хранилища (в форме текстового или бинарного файла, размещаемого в директории, указываемой пользователем через соответствующее диалоговое окно).

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

IDE	Интегрированная среда разработки
DFT	Метод функционала плотности
MEF	Managed Extensibility Framework
SOAP	Simple Object Access Protocol (простой протокол доступа к объектам)
WF	Workflow (поток заданий)
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**
	измененных	**замененных**	**новых**	**аннулированных**					