


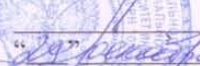
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

СОГЛАСОВАНО
Генеральный директор
ЗАО «АйТи»


Бакнев О.Р.
2011 г.



УТВЕРЖДАЮ
Ректор НИУ ИТМО


Васильев В.Н.
2011 г.



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

Руководство программиста

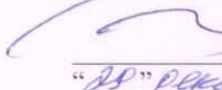
ЛИСТ УТВЕРЖДЕНИЯ

RU.СНАБ.80066-06 33 01-ЛУ

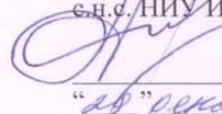
Инв. № подл.	Подп. и дата
Взам. инв. №	Подп. и дата
Инв. № дубл.	Подп. и дата
Подп. и дата	Подп. и дата

Представители
Организации-разработчика

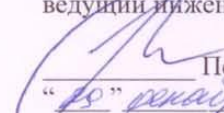
Руководитель разработки,
профессор НИУ ИТМО


Бухановский А.В.
"28" декабря 2011 г.

Ответственный исполнитель,
с.н.с. НИУ ИТМО


Луценко А.Е.
"28" декабря 2011 г.

Нормоконтролер
ведущий инженер НИУ ИТМО


Позднякова Л.Г.
"28" декабря 2011 г.

2011

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ**

**УТВЕРЖДЕН
RU.СНАБ.80066-06 33 01-ЛУ**

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

Руководство программиста

RU.СНАБ.80066-06 33 01

Листов 22

2011

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

АННОТАЦИЯ

Документ содержит руководство программиста многопрофильной инструментально-технологической платформы (МИТП) создания и управления распределенной вычислительной средой CLAVIRE (Cloud Applications Virtual Environment) RU.СНАБ.80066-06. МИТП CLAVIRE предназначена для создания, исполнения и предоставления сервисов доступа к предметно-ориентированным высокопроизводительным композитным приложениям, функционирующим в облаке неоднородных вычислительных ресурсов корпоративного уровня, уровня центров компетенции, центров обработки данных, инфраструктур экстренных вычислений и систем распределенного хранения и обработки данных. МИТП CLAVIRE разработана в ходе выполнения 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».

СОДЕРЖАНИЕ

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ	4
1.1. Назначение программы	4
1.2. Условия применения программы	4
2. ХАРАКТЕРИСТИКИ ПРОГРАММЫ	6
2.1. Технологические функции программы	6
2.2. Порядок использования программы	6
2.3. Режимы взаимодействия пользователя с программой	7
2.4. Общая архитектура	8
2.5. Основные компоненты	11
3. ОБРАЩЕНИЕ К ПРОГРАММЕ	12
3.1. Алгоритм функционирования ядра МИТП	12
3.2. Описание пакетов на языке EasyPackage	18
3.2.1. Структура языка	20
3.3. Регистрация описания в базе пакетов	25
4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ	27
5. СООБЩЕНИЯ	28
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ	31

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1. Назначение программы

Многопрофильная инструментально-технологическая платформа (МИТП) создания и управления распределенной средой облачных вычислений CLAVIRE RU.СНАБ.80066-06 предназначена для создания, исполнения и предоставления сервисов доступа к предметно-ориентированным высокопроизводительным композитным приложениям, функционирующим в облаке неоднородных вычислительных ресурсов корпоративного уровня, уровня центров компетенции, центров обработки данных, инфраструктур экстренных вычислений и систем распределенного хранения и обработки данных.

МИТП представляет собой комплекс программного обеспечения для разработки, настройки и эксплуатации сред распределенных вычислений, предназначенный для:

- 1) эффективного управления вычислительными, информационными и программными ресурсами распределенных неоднородных вычислительных инфраструктур в рамках модели облачных вычислений;
- 2) создания, исполнения, управления и предоставления сервисов доступа к предметно-ориентированным высокопроизводительным композитным приложениям, функционирующим на основе облака распределенных прикладных сервисов¹;
- 3) обеспечения функционирования программно-аппаратных комплексов (ПАК) поддержки инфраструктуры предметно-ориентированных облачных вычислений в различных предметных областях.

В данном документе рассматриваются способы обращения к МИТП с целью описания прикладных сервисов (пакетов), доступных МИТП в ходе ее функционирования. Для описания прикладных пакетов применяется язык EasyPackage.

1.2. Условия применения программы

Для развертывания компонентов МИТП необходима вычислительная система под управлением ОС Windows (XP и выше), с установленной средой Silverlight 4.0, или Linux

¹ Прикладным сервисом называется программа, входные и выходные данные которой интерпретируются в терминах конкретной предметной области и которая в распределенной среде через публичную либо корпоративную сеть передачи данных доступна для выполнения без использования вычислительных и программных ресурсов на стороне пользователя.

(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, WCF, Silverlight и удаленного развертывания сервисов (с использованием технологии WebDeploy). Примером web-сервера, соответствующего предъявленным требованиям, может служить Microsoft IIS версии 7.0 или выше.

Дополнительно для функционирования МИТП должен быть установлен сервер баз данных MongoDB версии 1.6.5. В ходе установки и настройки используются стандартные конфигурации указанных программных средств, не требующие специфической модификации. После установки необходимо осуществить запуск сервера баз данных для локального использования (localhost).

Для работы компонента информационного портала RU.СНАБ.80066-06 01 31 требуется установка СУБД MySQL (версии 5.0 или выше) и поддержка web-сервером интерпретатора языка PHP (версии 5.2 или выше). Для работы компонента хранения знаний RU.СНАБ.80066-06 01 17 требуется установка СУБД Microsoft SQLServer Compact Edition (версии 3.5 или выше). Также должен быть установлен web-сервер Glassfish версии 3.0.1, обеспечивающий поддержку технологии WebServices, необходимой для функционирования варианта реализации хранилища онтологической структуры RunLib. Кроме того, на ту же вычислительную систему должен быть установлен интерпретатор онтологической структуры Pellet версии 2.2.2, необходимый для функционирования хранилища знаний.

Компоненты МИТП функционируют на вычислительной системе – серверной ЭВМ со следующими минимальными характеристиками:

- тип процессоров: Intel-совместимый;
- число ядер – не менее 4;
- число процессоров – не менее 2;
- тактовая частота каждого процессора – не ниже 2.0 ГГц;
- оперативная память (на ядро) – не менее 2.0 ГБ;
- дисковая подсистема – не менее 5×250 ГБ RAID5;
- пропускная способность сетевых интерфейсов – не менее 1 Гбит/с.

Для взаимодействия с другими модулями системы требуется наличие выхода в Интернет или локальную сеть (если web-сервисы других подсистем доступны из локальной сети) с соответствующей поддержкой со стороны оборудования.

В целях повышения производительности и реактивности МИТП отдельные компоненты могут функционировать на разных вычислительных системах в рамках общей локальной сети.

2. ХАРАКТЕРИСТИКИ ПРОГРАММЫ

2.1. Технологические функции программы

МИТП CLAVIRE призвана обеспечить выполнение следующих технологических функций:

- 1) использование вычислительных мощностей разнородных удаленных ресурсов с обеспечением унифицированного доступа к ним;
- 2) использование доступных прикладных пакетов, установленных на удаленных ресурсах для решения задач избранной предметной области;
- 3) формирование композитных приложений, использующих в своем составе вызовы предметно-ориентированных сервисов, реализованных с использованием доступных системе прикладных пакетов;
- 4) подготовка файлов входных данных и расчетных параметров, необходимых для выполнения композитных приложений;
- 5) автоматизированное определение целевых ресурсов, доступных для исполнения сервисов в составе композитного приложения;
- 6) запуск процесса выполнения сформированных композитных приложений в распределенной вычислительной среде;
- 7) мониторинг процесса исполнения пользовательского задания в распределенной вычислительной среде;
- 8) работа с пользовательскими данными в удаленном хранилище.

2.2. Порядок использования программы

Использование МИТП CLAVIRE в режиме функционирования ядра сводится к инструментальному применению отдельных компонентов и управлению набором доступных ядру приложений (прикладных пакетов, зарегистрированных в базе пакетов). При этом последовательность работы программиста следующая.

RU.СНАБ.80066-06 33 01

- 1) После установки ядра МИТП программист осуществляет первоначальную конфигурацию компонентов ядра.
- 2) Программист на основе анализа документации прикладных пакетов, предполагаемых к использованию в составе композитных приложений, производит разработку описаний прикладных пакетов на предметно-ориентированном языке EasyPackage.
- 3) Сформированные описания вносятся в базу пакетов при помощи визуального интерфейса компонента CLAVIRE/AdminTools (см. «Методика использования компонента мониторинга и настройки основных компонентов платформы CLAVIRE/AdminTool» RU.СНАБ.80066-06 ИЗ).
- 4) После регистрации прикладных пакетов в базе пакетов использование программы возможно в двух направлениях:
 - a. разработка композитных приложений с использованием прикладных пакетов, зарегистрированных в базе (см. документы «Руководство программиста» для технологических платформ на базе МИТП RU.СНАБ.80066-06 33 02, RU.СНАБ.80066-06 34 33, RU.СНАБ.80066-06 33 04, RU.СНАБ.80066-06 33 05, RU.СНАБ.80066-06 33 06);
 - b. дальнейшая настройка ядра МИТП с использованием возможностей компонента CLAVIRE/AdminTools (см. документ «Руководство оператора» для МИТП RU.СНАБ.80066-06 34 01).

2.3. Режимы взаимодействия пользователя с программой

Взаимодействие программиста происходит в интерактивном режиме с использованием интерфейса программного компонента CLAVIRE/AdminTool. Использование этого компонента позволяет наряду с прочими функциями формировать и модифицировать описания прикладных программных пакетов, доступных ядру. Описания формируются на языке EasyPackage, предоставляющем пользователю широкие возможности по описанию способов вызова, обращения, настройки, передачи и приема данных, используемых при работе с прикладными пакетами, доступными на вычислительных ресурсах, управляемых МИТП.

2.4. Общая архитектура

На рис. 2.1 приведена общая структура ядра МИТП, отражающая основные элементы (подсистемы и компоненты) платформы.

В состав МИТП входят четыре высокоуровневые подсистемы (они необязательно ассоциируются с отдельными программными системами; форма их реализации может быть различной, в том числе составной).

1. *Подсистема человеко-компьютерного взаимодействия (ЧКВ)*, обеспечивающая диалог между пользователем и системой, работу с системой управления вычислениями и инструментарий для работы с данными. Подсистема ЧКВ отвечает за формирование эффективного интерфейса доступа к остальным подсистемам, интегрируемого в состав информационного портала. Портал помимо предоставления доступа к подсистеме ЧКВ за счет интеграции диалогового интерфейса и средств визуальной обработки данных (в том числе модуля доступа к серверному компоненту визуализации) предоставляет пользователю доступ к структурированной справочной информации и обеспечивает информационно-технологическую поддержку деятельности сообщества пользователей системы.

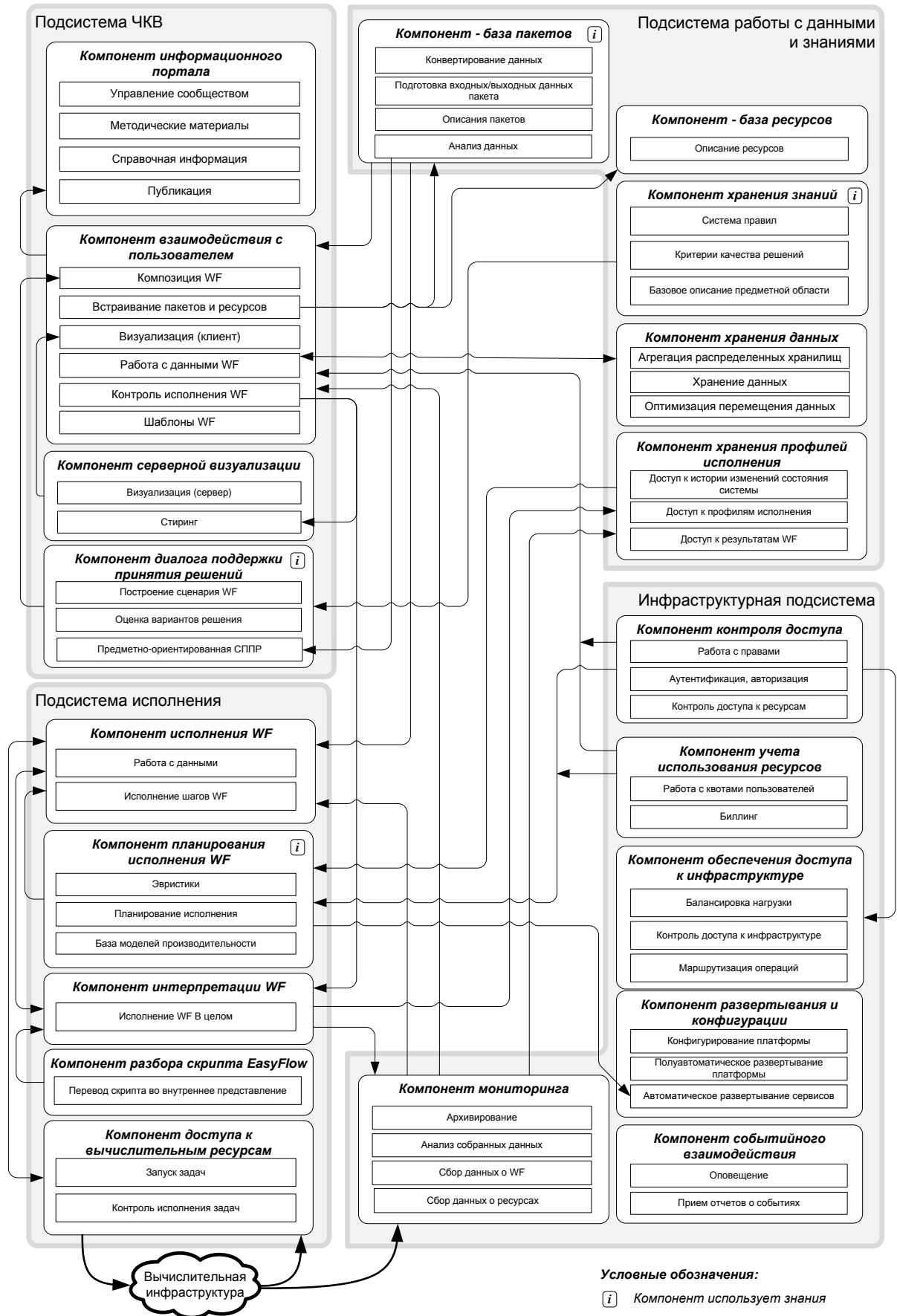


Рисунок 2.1 – Структура ядра МИТП

2. *Подсистема работы с данными и знаниями* отвечает за хранение состояния платформы, характеризуемого данными, накопленными в ходе эксплуатации платформы (куда входят результаты пользовательских экспериментов, внутренняя информация о работе системы), данными о конфигурации платформы (доступных ресурсах и сервисах) и рабочим набором формализованных знаний, используемых в процессе функционирования платформы (о предметной области, о сервисах, о работе платформы). К основным формам представления знаний можно отнести:
а) онтологическую структуру, агрегирующую базовую совокупность декларативных знаний; б) совокупность правил, определяющих основные действия интеллектуальной системы (в т.ч. интерпретацию онтологии, построение различных типов потоков заданий (workflow, WF), анализ состояния вычислительной инфраструктуры и пр.); в) базу моделей производительности, используемых при оценке вариантов реализации конкретных WF (далее – CWF); г) базу пакетов, содержащую описание предметных сервисов и механизмов обработки входных и выходных данных для них. Помимо хранения знаний и данных, а также предоставления доступа к ним пользователям и другим подсистемам, основной функцией данной подсистемы является выполнение процедур обработки данных, конвертирования и автоматического предварительного анализа. Следует обратить внимание на то, что задачи пользователя (в силу своей ключевой роли в процессе функционирования) также представляют собой специфический объект работы данной подсистемы.
3. *Инфраструктурная подсистема* обеспечивает внутреннюю целостность ядра платформы за счет предоставления сервисных механизмов взаимодействия компонентов; она является связующим звеном в работе других подсистем. Кроме того, данная подсистема отвечает за такие сервисные функции, как обеспечение контроля доступа к платформе, обеспечение безопасности работы пользователей и платформы в целом. Помимо этого подсистема играет важную роль в снабжении необходимой информацией других подсистем за счет сбора и агрегации данных о процессах, происходящих внутри управляющей части платформы, а также на вычислительных ресурсах.
4. *Подсистема исполнения* включает в себя модули контроля вычислительной инфраструктуры и управления выполнением заданий на вычислительных сервисах, входящих в ее состав. Обеспечивает доступ к основным

функциональным возможностям вычислительной облачной инфраструктуры. Интегрируя платформы исполнения как реальные и виртуальные вычислительные ресурсы, данная подсистема обеспечивает также интеграцию прикладных сервисов, унифицируя процедуру доступа к ним. Кроме того, база прикладных сервисов и соответствующих вычислительных пакетов может быть использована и на более высоком уровне вплоть до пользовательского интерфейса, обеспечивая возможность детального контроля процесса запуска пакета, задания его входных параметров и контроля выполнения.

2.5. Основные компоненты

В состав ядра МИТП были включены перечисленные ниже компоненты (с указанием десятичных номеров спецификации и названий компонентов). Для каждого компонента указывается список компонентов, от наличия которых зависит его корректное функционирование.

- RU.СНАБ.80066-06 01 21. Компонент взаимодействия с пользователем CLAVIRE/Ginger, реализующий графический интерфейс, предоставляющий возможность интерактивной работы с ресурсами, доступными в рамках ПАК. **Зависимости:** CLAVIRE/FlowSystem, CLAVIRE/Eventing, CLAVIRE/Monitoring, CLAVIRE/Storage, CLAVIRE/GateKeeper, CLAVIRE/PackageBase, CLAVIRE/Farming.
- RU.СНАБ.80066-06 01 20. Компонент интерпретации WF CLAVIRE/FlowSystem, обеспечивающий высокоуровневое представление структуры композитных приложений в форме цепочек заданий. **Зависимости:** нет.
- RU.СНАБ.80066-06 01 28. Компонент планирования исполнения WF CLAVIRE/Scheduler, обеспечивающий поддержку процесса выполнения цепочек заданий в автоматическом режиме с подбором оптимального плана выполнения. **Зависимости:** CLAVIRE/Farming, CLAVIRE/ResourceBase, CLAVIRE/Monitoring.
- RU.СНАБ.80066-06 01 23. Компонент событийного взаимодействия CLAVIRE/Eventing, обеспечивающий поддержку процесса взаимодействия системных сервисов в составе ядра МИТП. **Зависимости:** нет.
- RU.СНАБ.80066-06 01 24. Компонент мониторинга CLAVIRE/Monitoring, обеспечивающий сбор, унифицированное представление и первичный анализ

журналов работы отдельных компонентов в составе ядра МИТП. *Зависимости:* CLAVIRE/Eventing.

- RU.СНАБ.80066-06 01 37. Компонент хранения данных CLAVIRE/Storage, обеспечивающий управление данными в распределенных хранилищах, доступных МИТП. *Зависимости:* нет.
- RU.СНАБ.80066-06 01 26. Компонент контроля доступа CLAVIRE/GateKeeper, обеспечивающий идентификацию, аутентификацию и авторизацию пользователей с использованием политики прав. *Зависимости:* нет.
- RU.СНАБ.80066-06 01 33. Компонент - база ресурсов CLAVIRE/ResourceBase, обеспечивающий структурированное представление информации о доступных вычислительных ресурсах. *Зависимости:* нет.
- RU.СНАБ.80066-06 01 38. Компонент доступа к вычислительным ресурсам CLAVIRE/Farming, обеспечивающие контроль и управление системой разнородных распределенных вычислительных ресурсов. *Зависимости:* CLAVIRE/ResourceBase.
- RU.СНАБ.80066-06 01 35. Компонент - база пакетов CLAVIRE/PackageBase, обеспечивающий унифицированное представление знаний о форматах и способах использования пакетов, доступных в форме прикладных вычислительных сервисов. *Зависимости:* нет.
- RU.СНАБ.80066-06 01 36. Компонент развертывания и конфигурирования CLAVIRE/Deployment, обеспечивающий первоначальную и текущую настройку, а также высокоуровневое администрирование системных и прикладных сервисов МИТП. *Зависимости:* нет.

3. ОБРАЩЕНИЕ К ПРОГРАММЕ

Обращение программиста к программе сводится к двум высокоуровневым шагам: формирование описания прикладного пакета на языке EasyPackage и регистрация этого описания в базе пакетов МИТП.

3.1. Алгоритм функционирования ядра МИТП

Организация процесса построения композитного приложения в МИТП на основе концепции iPSE сводится к последовательной формализации наборов описаний в терминах WF. Верхним уровнем описания приложения является meta-WF (далее MWF). В

нем отдельные блоки содержат лишь описания расчетных задач, нужных пользователю, в виде неявных указаний. Таким образом, составленный MWF представляет собой формальное описание пользовательской задачи в терминах предметной области, без указаний на условия ее реализации. Помимо описания действий и данных, необходимых для вычислений, пользователь может задавать критерии, по которым будут подбираться конкретные сервисы, ресурсы, данные (например, время выполнения или высокая надежность), а также дополнительные ограничения и параметры отдельных действий (например, требуемая точность результата).

Процесс проектирования композитного приложения при таком задании исходных данных будет представлять собой процесс поэтапного уточнения MWF через стадии AWF и CWF вплоть до создания конкретных сценариев запуска сервисов в облачной среде и дальнейшего их выполнения, см. рис. 3.1.

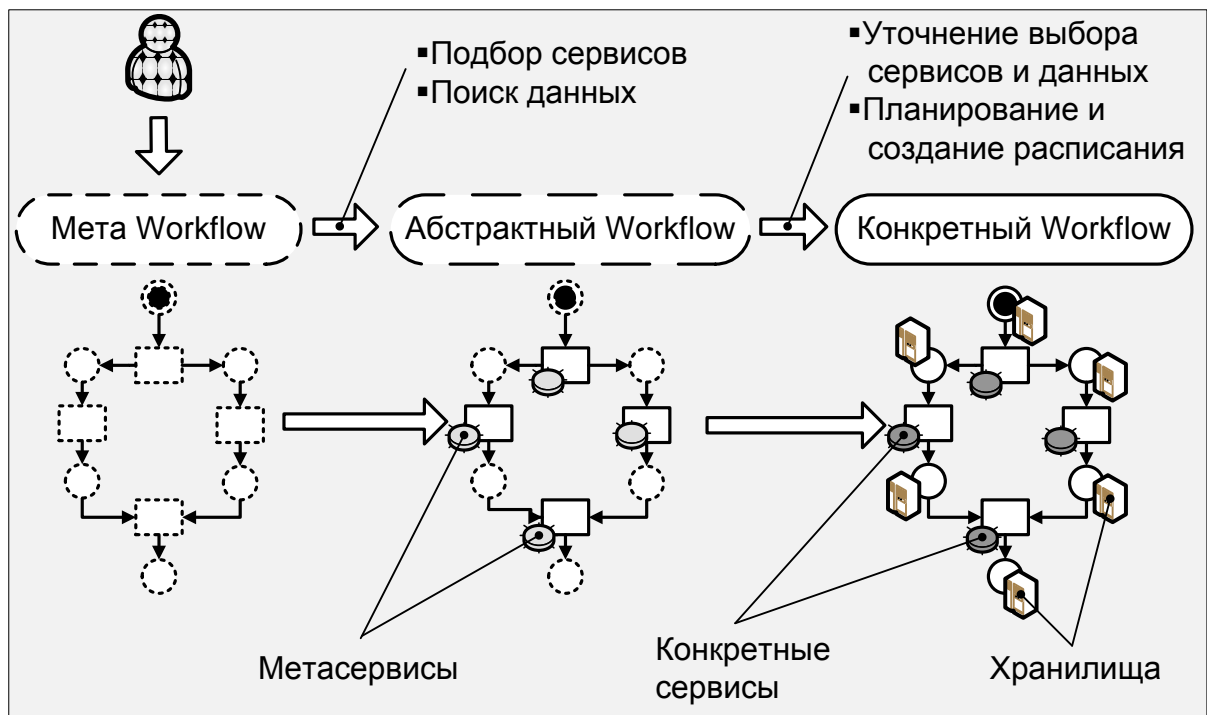


Рисунок 3.1 – Процесс проектирования композитного приложения в МИТП на основе

iPSE

На первом этапе процесса проектирования композитного приложения создается MWF. Пользователь может, например, осуществлять выбор классов сервисов, которые доступны в облачной среде, и уточнять их по мере ввода дополнительной информации. Указанные пользователем классы сервисов будут использоваться на следующем этапе для подбора конкретных сервисов. На его основе создается WF, в котором уже зафиксированы

конкретные реализации вычислительных сервисов, однако еще ничего не известно об условиях их выполнения (AWF). Следующим этапом процесса проектирования является построение расписания и создание сценария выполнения в терминах CWF, который представляет собой WF с полностью определенными блоками. Для блоков действий указаны сервисы и узлы для исполнения, а для блоков данных – конкретные местоположения необходимых данных.

Базовый алгоритм функционирования ядра МИТП. На рис. 3.2 проиллюстрирован базовый алгоритм (в виде временной схемы) формирования WF в процессе взаимодействия пользователя с МИТП, в составе которой выделены три ключевых модуля: интерфейс пользователя, интеллектуальная система (ИС) поддержки принятия решений пользователя и система управления исполнением как отдельные логические блоки МИТП. Рассмотрим основные этапы обработки WF, представленные на рис. 3.2, исходя из особенностей их практической реализации в распределенной среде облачных вычислений.

- *Формальное описание в терминологии базы знаний.* В рамках этого этапа происходит согласование представления пользователя о задаче с формальным описанием предметной области, доступным в базе знаний предметной области МИТП. Задачей данного этапа является определение общей терминологии, однозначно интерпретируемой как пользователем, так и системой. В рамках этой терминологии производится описание задачи, стоящей перед пользователем: определяются имеющиеся у пользователя входные и запрашиваемые выходные данные, формализуются параметры моделирования, определяемые пользователем (параметры предметной области). Результатом выполнения данного этапа является MWF, формализующий требования пользователя к задаче, решение которой требуется реализовать с использованием доступных вычислительных ресурсов.

Построение абстрактного WF включает в себя два базовых этапа. На первом этапе производится *декомпозиция задачи* на подзадачи, решаемые с использованием методов предметной области, описание которых доступно в МИТП. Целью декомпозиции является построение детального описания пути решения для поставленной задачи. При этом необходимо учитывать, что в процессе работы интеллектуальная система может обнаружить несколько путей решения поставленной задачи. В таком случае на этом и последующих этапах необходимо проводить дополнительное ранжирование и фильтрацию методов в соответствии с заданными критериями качества. Пространство

критериев качества и операции на нем должны лежать в основе механизма логического вывода интеллектуальной системы. На следующем этапе построенные варианты декомпозиции *отображаются на набор доступных метасервисов* (обобщенных прикладных сервисов). Результатом этого отображения становится AWF, построенный в соответствии с доступным описанием предметной области и доступным для использования набором вычислительных сервисов.

На следующем этапе производится *техническая настройка WF*, призванная а) отобразить полученный ранее абстрактный WF на набор конкретных сервисов, доступных в рамках вычислительной инфраструктуры; б) провести тонкую настройку параметров запуска сервисов с целью оптимизации производительности вычислительных ресурсов, используемых в процессе решения задачи. Данная задача решается интеллектуальной подсистемой и планировщиком МИТП на основе априорной оценки вычислительного процесса с использованием базы моделей параллельной производительности и набора оптимизирующих правил. При этом в процессе оптимизации учитывается как производительность отдельных вычислительных сервисов, так и накладные расходы на передачу и конвертирование данных, сервисные процессы программного комплекса, системные процессы и пр. Результатом выполнения данного этапа является готовый к исполнению CWF, содержащий ссылки на доступные вычислительные сервисы, находящиеся в хранилище данные, последовательность решения подзадач с использованием сервисов и другую системную информацию.

Сформированный CWF передается на *обработку* системе управления исполнением в форме соответствующего скрипта. Обработка включает в себя запуск WF на указанных ресурсах, контроль потоков данных и управления, мониторинг исполнения, обработку исключительных ситуаций, возникающих в процессе работы. При этом в случае изменения состава и характеристик доступных ресурсов при необходимости производится повторное построение конкретного WF с учетом актуализированных параметров вычислительных сервисов.

После завершения вычислительного процесса проводится дополнительный анализ полученных в результате расчета данных. При этом система позволяет провести анализ в двух режимах: автоматическом – обработка выходных данных под руководством интеллектуальной подсистемы (выделение запрошенных пользователем элементов данных, преобразование их к удобному для восприятия виду, когнитивная визуализация и пр.) и автоматизированный – ручная обработка результатов моделирования с

использованием доступных пользователю инструментальных средств (например, персонального ПО пользователя). Эти режимы могут эффективно дополнять друг друга в случае, если автоматический анализ данных представляет собой предварительный этап обработки.

Создаваемые в ходе последовательности действий на рис. 3.2 AWF и CWF носят для пользователя сугубо «рекомендательный» характер (поскольку МИТП на основе iPSE реализует, по сути, процесс поддержки принятия решений разработчика, а не автоматического управления). Так, пользователь имеет возможность поменять сформированный CWF непосредственно перед запуском. При необходимости может быть организован итеративный процесс определения WF на основе уточнения требований пользователя, выдвинутых после ознакомления с предложенным ему CWF. При этом требования пользователя могут отражать вмешательства разного уровня:

- *Технические требования.* Пожелания пользователя по изменению технических параметров запуска сервисов (например, режима распараллеливания). Таким образом, пользователь влияет на решения интеллектуальной подсистемы, принятые при построении CWF.
- *Требования предметной области.* Пользователь предъявляет дополнительные требования к параметрам предметной области, сформированным в ходе логического вывода, проведенного интеллектуальной подсистемой (т.е. модифицируется AWF).
- *Требования по выбору сервисов.* Пользователь может потребовать поменять состав и структуру метасервисов абстрактного WF (например, на основе личных предпочтений при выборе вычислительных пакетов). При этом пользователь имеет право как заменить отдельный пакет в составе WF, так и существенно поменять его структуру.
- *Требования к знаниям, на основе которых выполняется вывод.* Пользователь дополняет оценку правил, на основе которых был проведен логический вывод (например, на основе методологического недоверия к каким-либо утверждениям, используемым интеллектуальной подсистемой).
- *Требования к процессу планирования.* Могут включать требуемое пользователем время выполнения, указание класса используемых ресурсов, приемлемую стоимость исполнения WF и т.п.

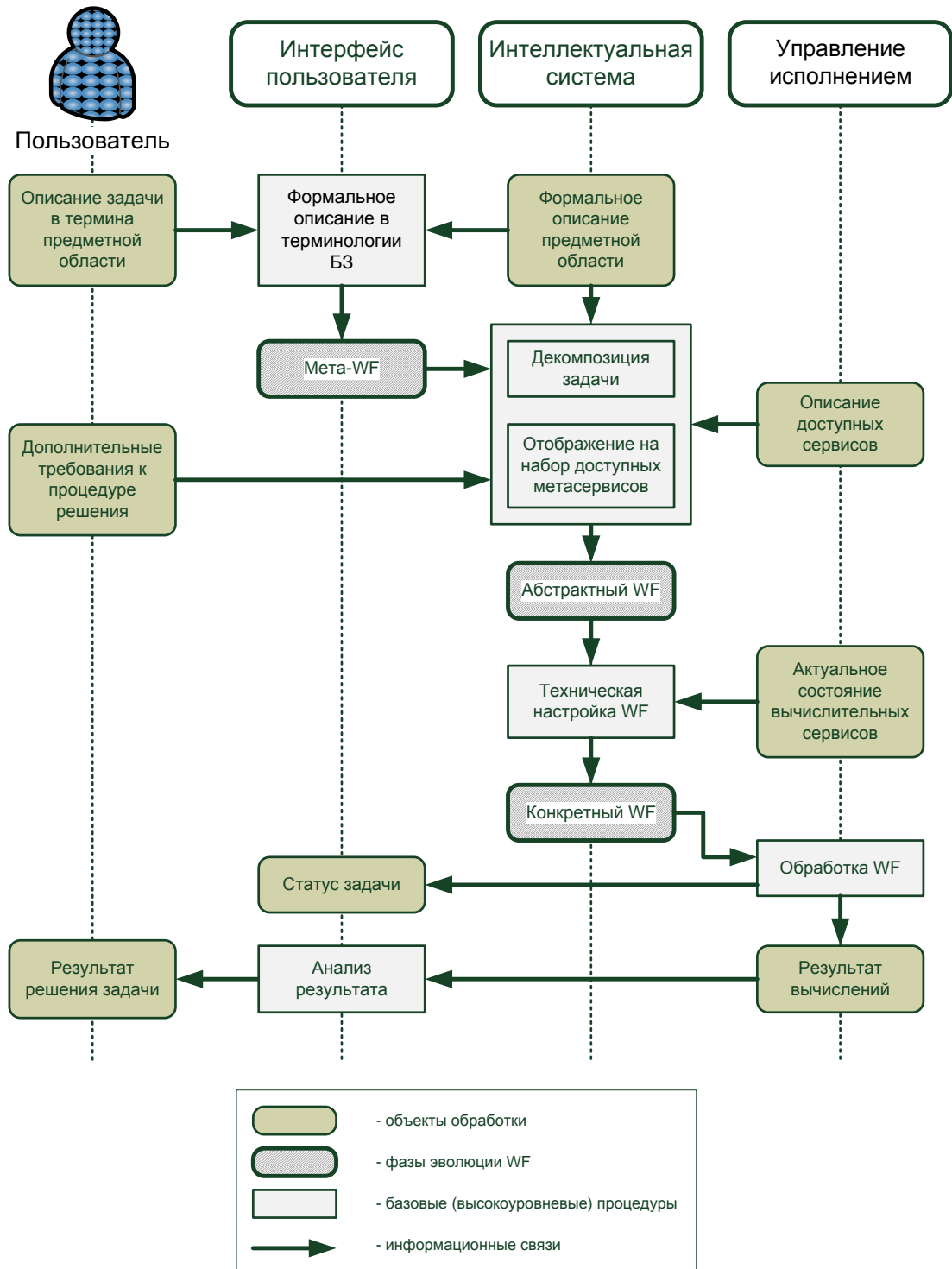


Рисунок 3.2 – Общая схема интерпретации WF в рамках МИТП на основе iPSE

При этом некоторые из требований пользователя (при соответствующем объяснении с его стороны) могут быть впоследствии использованы интеллектуальной системой МИТП в процессе вывода. Дополнительной возможностью является предложение пользователем изменений и расширений в системе знаний, возможное при достаточном уровне компетентности пользователя.

Алгоритм функционирования ядра МИТП, приведенный на рис. 3.2, соответствует выполнению главной задачи программного комплекса – создания и запуска композитного приложения в распределенной среде по заданию пользователя. Однако для отдельных технологических платформ он распадается на ряд детализированных алгоритмов, решающих отдельные подзадачи.

3.2. Описание пакетов на языке EasyPackage

Описать пакет в базе пакетов, значит составить файл описания на языке EasyPackage (обязательный пункт), шаблоны для сборки входных данных, написать расширения для работы с данными пакета.

Рекомендуется следующий алгоритм формирования описания пакета.

1. Подготовить общее описание пакета (название, адрес сайта).
2. Выделить режимы работы пакета и определить, какие параметры влияют на них (часто пакеты содержат параметры-переключатели, которые позволяют выбирать режим работы в зависимости от указанного значения). Режимы можно формировать, используя вычисляемое поле `enabled` – включенность параметра.
3. Декомпозиция параметров. Необходимо выделить достаточный для запуска набор параметров, далее его можно постепенно расширять.
4. Для каждого параметра определить тип, область допустимых значений.
5. Выявление зависимостей между параметрами. Параметры могут быть связаны за счет использования значений внутри процедур (проверки корректности, проверки включенности).
6. Разбор форматов данных пакета и написание расширений для разбора (`extractor`) и сборки (`assembler`) файлов в этих форматах.
7. Формирование параметров исполнения (командная строка, переменные окружения).

Этап заполнения параметров пользователем представляет собой интерактивный цикл модификации интерфейса в ходе реакции на действия пользователя. База пакетов при этом работает с параметрами как с графом, учитывая явные и неявные зависимости между ними. При описании параметров важно учитывать несколько правил формирования базы пакетов.

1. Базовый набор параметров, состоящий как минимум из одного параметра, должен быть изначально активен (`enabled true`).

2. В графе зависимостей параметров не должно быть циклических зависимостей.

Ниже приведены фрагменты описаний пакетов, связанные с описанием контрактного (доступного для редактирования) параметра (рис. 3.3), описанием вычислимого параметра (рис. 3.4), описанием файла (рис. 3.5).

```
public param{
  name "pairlistdist"
  type float
  default 12.0
  validator { |val, ctx| val > 0 }
  display "Pair list distance"
  description "Расстояние добавления атомов в список взаимодействия
Верле, англстрем"
}
```

Рисунок 3.3 – Пример описания контрактного параметра

```
param{
  name "molpro_method"
  display "molpro method"
  type string
  evaluator { |ctx|
    if ctx.calc_method == 'DFT'
      return "rks,b3lyp"
    elsif ctx.calc_method == 'HF'
      return "hf"
    elsif ctx.calc_method == 'MCSCF'
      return "mcscf"
    end
  }
}
```

Рисунок 3.4 – Пример описания внутреннего, вычислимого параметра

```
public file{
  name "XSTfile_output"
  display "Periodic cell discription"
  depends ["sphericalBC"]
  enabled { |ctx| ctx.sphericalBC=="off" }
  filename "cell.xst"
  path "/output/"
}
```

Рисунок 3.5 – Пример описания контрактного файла

При описании форматов входных данных для пакета механизм шаблонов позволяет размечать файл-заготовку директивами вставки параметров управляющих операций. Такой файл легко поддерживать. В языке EasyPackage поддерживается система шаблонизации ERB, которая считается стандартной для языка программирования Ruby.

```
$SYSTEM MWORDS=50 $END
$BASIS <%=w.package_basis%> $END
```

```

$CONTRL <%=w.package_maxit%> RUNTYP=<%=w.package_task%> <%if
w.package_basis!='GBASIS=AM1'%>SCFTYP=<%=w.package_method%><%end%><%=w.packag
e_pseudopotential%> <%if w.useIChange%>ICHARG=<%=w.charge%><%end%>
<%if w.package_basis!='GBASIS=AM1'%>DFTTYP=<%=w.functional%> <%end%>
<%=w.package_nzvar%> $END
<%if w.package_method=='RHF TDDFT=EXCITE'%>$TDDFT NSTATE=<%=w.num_of_state%>
$END<%end%>
<%if w.num_of_steps>0%>$STATPT NSTEP=<%=w.num_of_steps%> $END <%end%>
<%=w.package_FORCE%> <%=w.package_STATPT%>
<%=w.package_ZMAT%>
<%=w.package_DIRSCF%>
<%=w.package_solvent%>
$DATA
Title
C1
<%=w.package_atoms_xyz%> $END

```

Рисунок 3.6 – Пример описания шаблона файла

Ниже приведены некоторые рекомендации по поводу заполнения описания пакета.

1. Для своевременного выявления ошибок рекомендуется по возможности сужать область определения параметров за счет написания валидаторов (validator).
2. Для поддержки пользователей лучше приводить краткое описание параметра и принимаемых им значений – поле комментариев (description).
3. При неправильном ходе вычисления параметров (это можно определить по журналу базы пакетов) необходимо проверить, везде ли указаны зависимости между параметрами (depends).

В следующем разделе представлено подробное описание синтаксиса EasyPackage.

3.2.1. Структура языка

Общие сведения

Язык описания прикладных пакетов (далее – Язык или EasyPackage) предназначен для описания прикладных сервисов (далее – пакет). Описание пакета позволяет формализовать знания эксперта о пакете, его режимах работы, об используемых им данных. Описание пакета совмещает в себе как декларативную информацию (общее описание, описание параметров), так и императивную (процедуры проверки корректности параметров, процедуры работы с форматами файлов). Цель данного языка – предоставить гибкий, расширяемый инструмент описания пакетов для представления в удобном для чтения виде.

Данный язык создан на базе языка программирования общего назначения Ruby и предназначен для интерпретирования средствами компонента базы пакетов.

Основные характеристики языка:

- скрипт представляет собой текстовый файл в кодировке UTF-8;
- зависит от регистра;
- типизация переменных – строгая, динамическая с явным приведением типов;
- язык является платформо-независимым (с точки зрения различия операционных систем и вычислительных платформ);
- язык является интерпретируемым.

Влияние Ruby

Язык был разработан на основе языка Ruby, и как уже было отмечено, весь код на EasyPackage является корректным на Ruby. Базовые элементы Языка идентичны элементам Ruby, а именно: комментарии, идентификаторы, базовые типы данных, элементы объектно-ориентированного программирования (классы), функции. На рис. 3.7 приведены примеры использования данных конструкций в языке. Более подробную информацию о языке Ruby можно найти на его официальном сайте (<http://www.ruby-lang.org/>).

```
# Комментарий

# Базовые типы данных и их инициализация
str = "string example"
int_var = 5
flag = true
double_var = 5.4
list = [ 0, 1, "two" ]
hash = { "a" => "b" }

# Оператор ветвления
if true
  puts "true"
elsif false
  puts "false"
end

# Оператор цикла
for i in [5,4]
  puts i
end

# Объявление функции
def Do
  4
end

# Объявление класса
class A
  # Конструктор
```

```

def initialize()
end
def Print()
  puts "A"
end
end

# Наследование
class B < A
  def Print()
    puts "B"
  end
end
end

```

Рисунок 3.7 – Базовые элементы языка

Грамматика языка

На рис. 3.8 представлена грамматика EasyPackage в нотации РБНФ со следующими соглашениями:

- ::= – отделяет название лексемы от ее описания;
- {A} – ноль или более элементов A;
- [A] – элемент A может входить или не входить;
- (A B) – группа элементов;
- A | B – либо A, либо B.

```

Программа ::= {ОбъявлениеКласса} {(Поле ПереводСтроки)} {Секция}
{ОбъявлениеНабораПараметров> prepare_package
Секция ::= ИмяСекции ОткрывающаяФигурнаяСкобка {ОбъявлениеПараметра}
{ОбъявлениеСпециальногоПараметра} ЗакрывающаяФигурнаяСкобка
ОбъявлениеСпециальногоПараметра ::= cmdline
ОбъявлениеПараметра ::= [Модификатор] ТипПараметра ОткрывающаяФигурнаяСкобка {Поле}
ЗакрывающаяФигурнаяСкобка
Модификатор ::= public
ТипПараметра ::= param | file | char | file_group
Поле ::= ПолеДанных | Процедура
ПолеДанных ::= ИмяПоля [ВыражениеRuby]
Лямбда ::= ОткрывающаяФигурнаяСкобка БлокСПараметрамиRuby ЗакрывающаяФигурнаяСкобка
Процедура ::= ИмяПроцедуры Лямбда
ТипДанных ::= bool | enum [ [<значение перечислителя>, ] ] | int | string | list |
hash

```

Рисунок 3.8 – Грамматика языка описания пакетов

Поля

Поле является основным элементом декларативного описания в Языке и состоит из имени и значения. Значение может быть базовым элементом Ruby либо процедурой. Под

процедурой имеется в виду параметризованное лямбда-выражение Ruby (количество параметров зависит от типа поля). На рис. 3.9 приведены примеры указания полей.

```

name "А" # строковое поле
required # флаг
type int # объявление типа
depends ["В"] # список

# процедура вычисления значения поля
evaluator { |ctx| ctx.A0 + 10 }

# процедура проверки корректности поля
validator { |val, ctx| val >= ctx.startCalcDate}

```

Рисунок 3.9 – Примеры объявления полей

Структура описания пакета

Структура описания пакета состоит из следующих разделов: объявление расширений, общее описание пакета, секционное описание входных и выходных данных пакета. Общее описание пакета предназначено для указания общей информации о пакете (рис. 3.10).

```

name "Swan"
display_as "SWAN"
vendor "Simulating WAVes Nearshore"
url "http://www.wldelft.nl/soft/swan/"
license "GPLv3"
description "SWAN simulates the following physical phenomena:
  1. Wave propagation in time and space, shoaling, refraction due to
current and depth, frequency shifting due to currents and nonstationary
depth.
  2. Wave generation by wind.
  3. Nonlinear wave-wave interactions (both quadruplets and triads).
  4. Whitecapping, bottom friction, and depth-induced breaking.
  5. Blocking of waves by current. "

```

Рисунок 3.10 – Фрагмент общего описания пакета

Для указания доступны следующие поля описания пакета:

- name – название пакета;
- version – версия пакета;
- display – отображаемое название пакета;
- vendor – разработчик пакета;
- url – адрес официального сайта пакета;
- license – лицензия, по которой распространяется пакет;
- description – описание пакета;
- logo – логотип пакета.

Раздел объявления расширений предназначен для написания классов Ruby и функций для расширения функциональности базовой библиотеки языка.

```
class ObjectToSAssembler < FileAsStringAssembler
  def initialize(pname)
    @Pname = pname
  end
  def compile(ctx)
    ctx[@Pname].to_s
  end
end

class IntegerFileExtractor < TextExtractor
  def initialize(pname)
    @Pname = pname
  end
  def extract(str)
    return {@Pname => str.to_i}
  end
end
```

Рисунок 3.11 – Расширение базовой библиотеки по работе с данными на примере экстрактора и сборщика файлов

Раздел секционного описания входных и выходных данных состоит из нескольких секций описания данных, например: inputs, outputs, characteristics.

Параметры

Параметры являются основной абстракцией при работе с данными пакета. Существует несколько типов параметров: обычный параметр, файловый параметр, характеристика, группа файлов. Параметр – это элемент описания пакета, который состоит из полей. Рассмотрим доступные поля для каждого из типов. Общие для всех типов параметров поля:

- name – имя параметра;
- description – описание параметра;
- display – отображаемое имя параметра;
- required – флаг, определяющий параметр;
- enabled – флаг, определяющий, включен ли параметр.

Обычные параметры дополнительно имеют следующий набор полей:

- type – тип;
- default – значение по умолчанию;
- validator – процедура проверки значения параметра на корректность;
- validation_error_msg – сообщение при некорректном значении параметра;
- evaluator – процедура вычисления параметра.

Файловые параметры дополнительно имеют следующий набор полей:

- filename – имя файла;
- place – путь до файла;
- extractor – процедура извлечения данных из файла;
- assembler – процедура сборки файла;
- file_validator – процедура проверки файла на корректность.

Параметр «файловая группа» характеризуется дополнительно одним полем – filters, которое содержит список регулярных выражений для нахождения файлов.

Секции

Секции необходимы для структурирования описания пакета и его данных. Существует несколько типов секций: inputs – секция описания входных данных, outputs – секция описания выходных данных, characteristics – секция описания характеристик. На рис. 3.12 приведен пример описания секции inputs.

```
inputs {
  public param {
    name "startCalcDate"
    required
    display_as "Start Calculation Date"
    type datetime
    validator lambda { |val, ctx| val < Time.now }
  }

  file {
    name "inConfig"
    required
    filename "config.ini"
    assembler old_school_template rtext("swanConfig.erb")
    place "/"
  }

  cmdline { |ctx| "{0}" }
}
```

Рисунок 3.12 – Фрагмент описания секции inputs

3.3. Регистрация описания в базе пакетов

Для регистрации прикладного пакета в базе пакетов следует воспользоваться возможностями, предоставляемыми компонентом CLAVIRE/AdminTools. Для того чтобы добавить новый прикладной пакет в состав комплекса с использованием этого компонента, следует перейти на вкладку «Пакеты» (рис. 3.13), которая предоставляет интерфейс настройки базы пакетов (CLAVIRE/PackageBase).

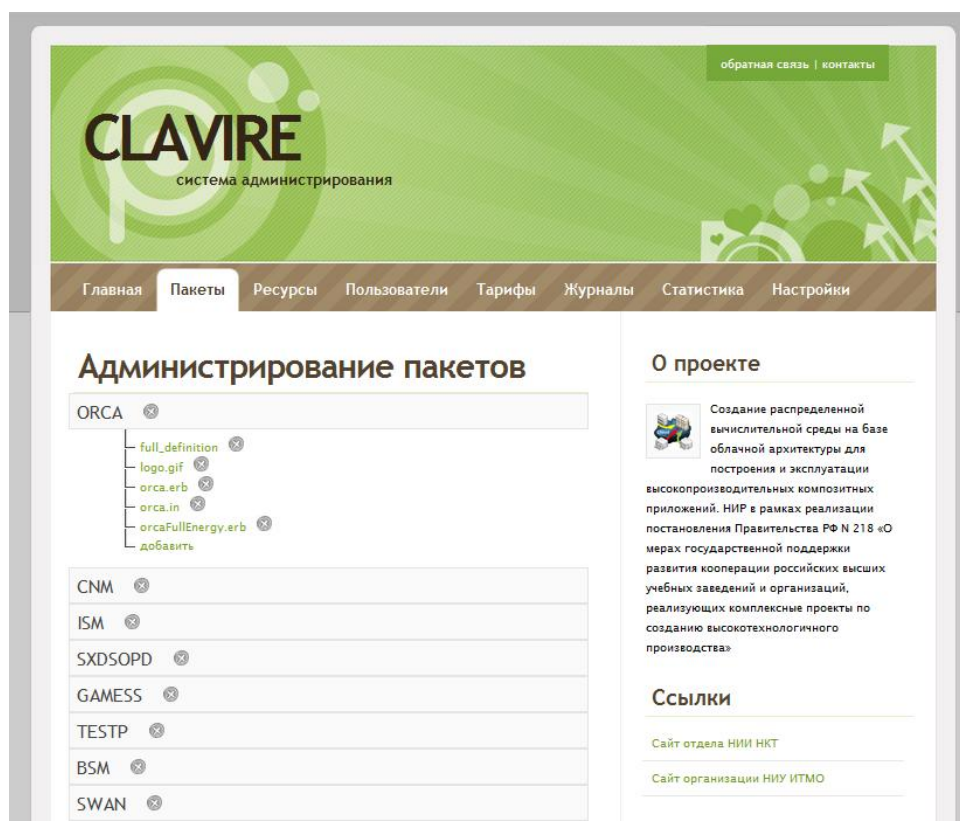


Рисунок 3.13 – Вкладка «Пакеты»

В частности, в этой вкладке можно добавлять и удалять описания пакетов, изменять состав описания пакетов. При нажатии на файл появится окно редактирования его содержания (рис. 3.14), для обеспечения атомарности изменений файлов поддерживается редактирование только его содержимого, но не имени.

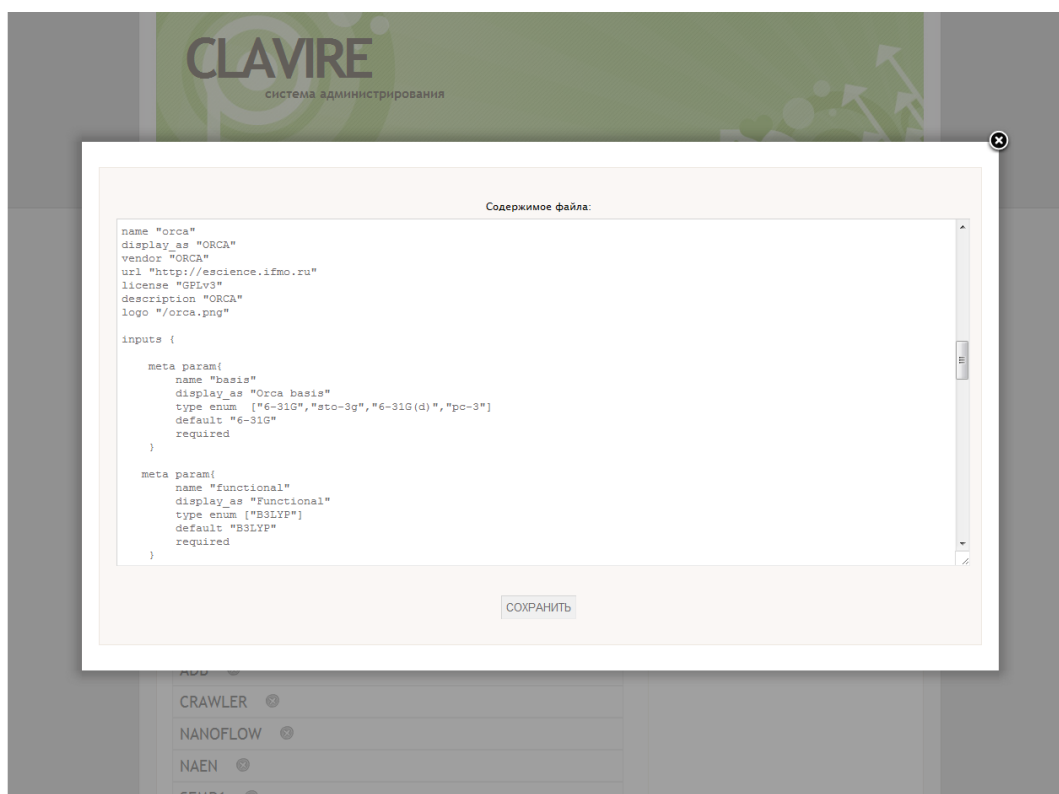


Рисунок 3.14 – Редактирование файла описания пакета

Описание прикладного пакета задается на языке EasyPackage, основные правила использования которого приведены в разделе 3.2.

4. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Для работы с МИТП не требуется специальных видов входных данных. В ходе работы входными данными могут являться любые входные файлы, соответствующие по формату запускаемым прикладным сервисам (в текстовом, цифровом, графическом виде), а также данные, вводимые пользователем с клавиатуры по запросу сервиса. В случае несоответствия данных условиям их использования будет выдано системное сообщение. Для приведения данных к общему формату используется технология описания на основе языка EasyPackage.

В качестве выходных данных МИТП предоставляет результаты расчетов, загруженные с удаленного хранилища CLAVIRE/Storage RU.СНАБ.80066-06 01 25 (в форме текстового, графического или цифрового файла, размещаемого в директории, указываемой пользователем через соответствующее диалоговое окно). Формат файла соответствует тому сервису, посредством которого был произведен расчет. Для

обеспечения единого формата в целях унификации процесса передачи данных между сервисами используется технология описания на основе языка EasyPackage.

5. СООБЩЕНИЯ

В данном разделе определяются основные сообщения программисту, информирующие об ошибке, возникшей в процессе работы с МИТП. Сообщения программисту могут выдаваться двумя способами.

1. *Посредством пользовательского интерфейса.* В этом случае программист получает сообщение в виде всплывающего окна, информирующего о возникновении исключительной ситуации (рис. 5.1).

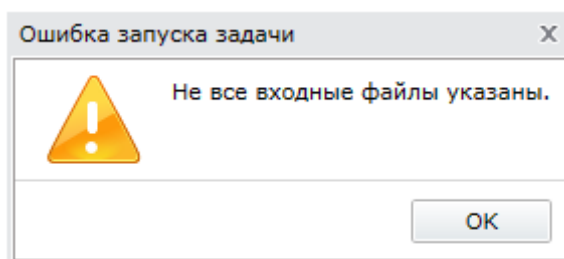


Рисунок 5.1 – Сообщение об исключительной ситуации (пример)

2. *В журнал событий МИТП,* где приводится развернутая информация, необходимая для устранения ошибки. Доступ к журналу сообщений МИТП и ее компонентов программист может получить, используя возможности компонента средств мониторинга и настройки основных компонентов платформы CLAVIRE/AdminTool. Действуя в соответствии с документом «Методика использования компонента мониторинга и настройки основных компонентов платформы CLAVIRE/AdminTool» RU.СНАБ.80066-06 ИЗ 64, программист может получить доступ к просмотру внутренних журналов платформы и ее компонентов (рис. 5.2).

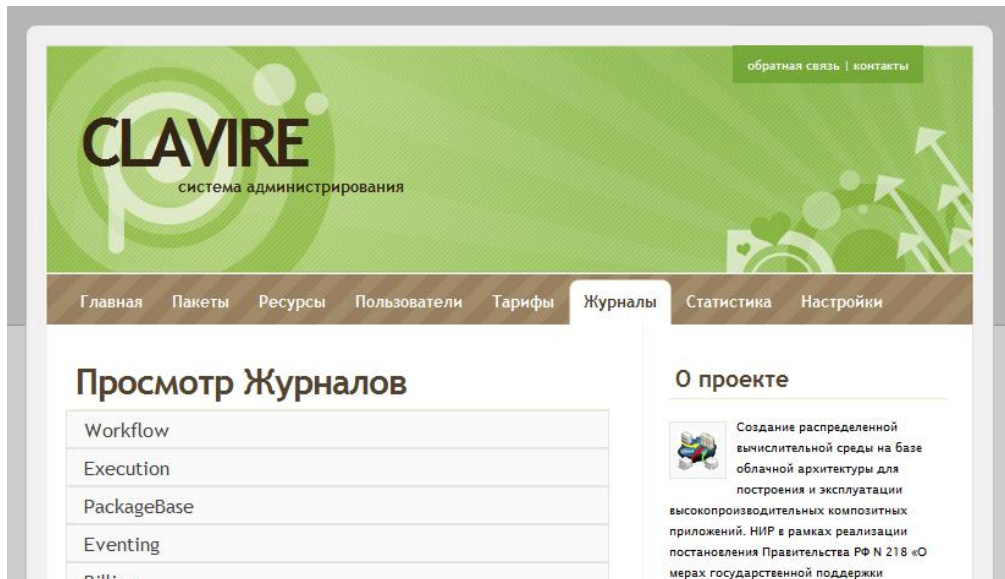


Рисунок 5.2 – Журналы работы компонентов платформы

При выборе компонента появляется окно просмотра журнала этого компонента (рис. 5.3). Особое внимание стоит обращать на строки, выделенные красным – это сообщения об ошибках в работе комплекса.

```

2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.DeclarativeInterpreter Internal event enqueued @block_started(WF#02fc0e8c-0c23-459a-87d1-477876a4fde2.1 +nul
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.GlobalDataScope Shared variable 'Data_base.Result' in global data scope
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.NodeBase Node#1 state changed state_started -> state_pre_section
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.NodeBase Step:Data_base#1(state_pre_section) Pre section is NULL ignoring
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.NodeBase Node#1 state changed state_pre_section -> state_run_start
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.NodeBase RunMode was set to Meta
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.NodeBase [Ignoring temporary] Error while checking package run signature.
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.NodeBase Creating parameter list
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.NodeBase [Ignoring temporary] Error while forming outputs
2011-12-27 19:08:37.4785 Easis.Wfs.Interpreting.NodeBase [Ignoring temporary] Error while forming outputs
2011-12-27 19:08:37.4785 Easis.Wfs.FlowSystemService.DryExecutionStepStarter Defining step using Execution.
2011-12-27 19:08:37.4785 Easis.Wfs.FlowSystemService.DryExecutionStepStarter { "t": "TaskDescription", "ExtensionData": null, "ExecParams": {}, "Input
2011-12-27 19:08:37.4785 Easis.Wfs.FlowSystemService.ExecutionEventConverter Staring step using Execution.
2011-12-27 19:08:39.5126 Easis.Wfs.FlowSystemService.ExecutionEventConverter Found event from Execution for StepId 21909. Trying to find accordance in id di
2011-12-27 19:08:40.5584 Easis.Wfs.FlowSystemService.ExecutionEventConverter Found event from Execution for StepId 21910. Trying to find accordance in id di
2011-12-27 19:08:41.5252 Easis.Wfs.FlowSystemService.ExecutionEventConverter SequenceGetInfo returns valid object
2011-12-27 19:08:41.5252 Easis.Wfs.FlowSystemService.ExecutionEventConverter { "t": "Task", "ExtensionData": {}, "ExecParams": {}, "InputFiles": [{ '
2011-12-27 19:08:41.5252 Easis.Wfs.FlowSystemService.ExecutionEventConverter RunInfo has been successfully fetched
2011-12-27 19:08:41.5252 Easis.Wfs.FlowSystemService.JobExecutor Got PushEvent command for WF#02fc0e8c-0c23-459a-87d1-477876a4fde2. Run_started(WF#02fc0e8c
2011-12-27 19:08:41.5252 Easis.Wfs.Interpreting.DeclarativeInterpreter External event enqueued @run_started(WF#02fc0e8c-0c23-459a-87d1-477876a4fde2.0 +Easis
2011-12-27 19:08:41.5252 Easis.Wfs.Interpreting.NodeBase Node convert action action_set_run_info was called with arg Easis.Wfs.Interpreting.StepRunInfo
2011-12-27 19:08:41.5252 Easis.Wfs.Interpreting.NodeBase Node#0 state changed state_run_start -> state_wait_results
2011-12-27 19:08:42.5710 Easis.Wfs.FlowSystemService.ExecutionEventConverter SequenceGetInfo returns valid object
2011-12-27 19:08:42.5710 Easis.Wfs.FlowSystemService.ExecutionEventConverter { "t": "Task", "ExtensionData": {}, "ExecParams": {}, "InputFiles": [],
2011-12-27 19:08:42.5710 Easis.Wfs.FlowSystemService.ExecutionEventConverter RunInfo has been successfully fetched
2011-12-27 19:08:42.5710 Easis.Wfs.FlowSystemService.ExecutionEventConverter Event Eventing.EventReport converted with Easis.Wfs.FlowSystemService.Execution
2011-12-27 19:08:42.5710 Easis.Wfs.FlowSystemService.JobExecutor Got PushEvent command for WF#02fc0e8c-0c23-459a-87d1-477876a4fde2. Run_started(WF#02fc0e8c
2011-12-27 19:08:42.5710 Easis.Wfs.Interpreting.DeclarativeInterpreter External event enqueued @run_started(WF#02fc0e8c-0c23-459a-87d1-477876a4fde2.1 +Easis
2011-12-27 19:08:42.5710 Easis.Wfs.Interpreting.NodeBase Data_base action action_set_run_info was called with arg Easis.Wfs.Interpreting.StepRunInfo
2011-12-27 19:08:42.5710 Easis.Wfs.Interpreting.NodeBase Node#1 state changed state_run_start -> state_wait_results
2011-12-27 19:08:44.7516 Easis.Wfs.FlowSystemService.ExecutionEventConverter Found event from Execution for StepId 21910. Trying to find accordance in id di
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter SequenceGetInfo returns valid object
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter { "t": "Task", "ExtensionData": {}, "ExecParams": {}, "InputFiles": [],
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter Found output file name:'test_mol.pdbqt' slot:'none' storageid:680X2C1060XY747YC
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter Found output file name:'test_2.pdbqt' slot:'none' storageid:88BXBYIJI44W0468XK
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter Found output file name:'test_3.pdbqt' slot:'none' storageid:88WJNSJSHIFG07GSSB
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter Found output file name:'test_4.pdbqt' slot:'none' storageid:CJQAS67X69S2V7P0K
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter Storage service returned 4 ids for 4 data entries
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter StepResult has been successfully fetched
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.ExecutionEventConverter Event Eventing.EventReport converted with Easis.Wfs.FlowSystemService.Executor
2011-12-27 19:08:46.8041 Easis.Wfs.FlowSystemService.JobExecutor Got PushEvent command for WF#02fc0e8c-0c23-459a-87d1-477876a4fde2. Run_finished(WF#02fc0e8c
2011-12-27 19:08:46.8041 Easis.Wfs.Interpreting.DeclarativeInterpreter External event enqueued @run_finished(WF#02fc0e8c-0c23-459a-87d1-477876a4fde2.1 +Easi
2011-12-27 19:08:46.8041 Easis.Wfs.Interpreting.NodeBase Data_base action action_set_run_results was called with arg Easis.Wfs.Interpreting.StepRunResu
2011-12-27 19:08:46.8041 Easis.Wfs.Interpreting.NodeBase Node#1 state changed state_wait_results -> state_run_finish
2011-12-27 19:08:46.8041 Easis.Wfs.Interpreting.NodeBase Exception while converting output param 'dbl' with value 'null'. Ignoring.
2011-12-27 19:08:46.8041 Easis.Wfs.Interpreting.GlobalDataScope Shared variable 'Data_base.Result' in global data scope
2011-12-27 19:08:46.8041 Easis.Wfs.Interpreting.NodeBase Node#1 state changed state_run_finish -> state_post_section
2011-12-27 19:08:46.8041 Easis.Wfs.Interpreting.NodeBase Step:Data_base#1(state_post_section) Post section is NULL ignoring
2011-12-27 19:08:46.8197 Easis.Wfs.Interpreting.NodeBase Node#1 state changed state_post_section -> state_finished
2011-12-27 19:08:46.8197 Easis.Wfs.Interpreting.NodeBase Node#1 generated event BLOCK_FINISHED
2011-12-27 19:08:46.8197 Easis.Wfs.Interpreting.DeclarativeInterpreter Internal event enqueued @block_finished(WF#02fc0e8c-0c23-459a-87d1-477876a4fde2.1 +nul

```

<</logs

Рисунок 5.3 – Журнал системных сообщений МИТП

Перечень наиболее важных сообщений, выдаваемых компонентами МИТП, приведен в табл. 5.1.

Основные сообщения программисту и оператору

Сообщение	Типовые действия по выявлению и устранению ошибки
Ошибки входа в систему	
Введенная пара логин / пароль неверна	Проверить введенные логин и пароль на корректность, проверить состояние клавиши CapsLock
Ваша учетная запись отключена	Обратиться к администратору комплекса по поводу отключения учетной записи
Система находится на профилактике, попробуйте позднее	Повторить попытку входа через 3–5 минут
Сервер возвратил ошибку: NotFound	Обратиться в службу поддержки комплекса за устранением ошибки
Нет соединения с сетью	Проверить наличие соединения с Интернетом и попробовать повторно выполнить операцию
Истекло время ожидания ответа от сервера	Проверить наличие соединения с Интернетом и попробовать выполнить операцию снова. В случае повторения обратиться в службу поддержки комплекса
Соединение принудительно прервано сервером	Попробовать выполнить операцию снова. В случае повторения обратиться в службу поддержки комплекса
Ошибки работы с проектами и задачами	
Не удалось сохранить проект	Проверить наличие соединения с Интернетом и попробовать выполнить операцию снова. В случае повторения обратиться в службу поддержки комплекса
Не все входные данные задачи указаны	Указать все требуемые для задачи файлы и повторить операции
Невозможно запустить задачу: недостаточно свободных ресурсов	Подождать 3–5 минут и попробовать выполнить операцию снова. В случае повторения обратиться в службу поддержки комплекса
Системные ошибки	
Нехватка места на жестком диске	Проверить корректность работы программных компонентов, установленных на указанной ЭВМ, на предмет бесконтрольного заполнения свободного пространства на жестком диске, осуществить переконфигурацию программных модулей, принять меры к увеличению свободного пространства на жестком диске
Нехватка оперативной памяти	Проверить корректность работы программных компонентов, установленных на указанной ЭВМ, на предмет бесконтрольного заполнения пространства оперативной памяти, осуществить переконфигурацию программных модулей, принять меры к увеличению объема оперативной памяти

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

WF	Поток заданий, workflow
AWF	Абстрактный WF
CWF	Конкретный WF
ННС	Национальная нанотехнологическая сеть
MWF	Мета-WF

