ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

РАДЧЕНКО Глеб Игоревич

СЕРВИСНО ОРИЕНТИРОВАННЫЙ ПОДХОД К ИСПОЛЬЗОВАНИЮ СИСТЕМ ИНЖЕНЕРНОГО ПРОЕКТИРОВАНИЯ И АНАЛИЗА В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДАХ

Специальность 05.13.11 - математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени кандидата физико-математических наук

Научный руководитель:
СОКОЛИНСКИЙ Леонид Борисович,
доктор физ.-мат. наук, профессор

Челябинск - 2009

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

АКТУАЛЬНОСТЬ ТЕМЫ

Системы компьютерного проектирования (CAE - Computer Aided Engineering), ориентированные на разработку сложных технологических процессов, конструкций, и материалов, являются сегодня одним из ключевых факторов обеспечения конкурентоспособности любого высокотехнологического производства. Использование таких систем дает возможность проводить *виртуальные* эксперименты, которые в реальности выполнить затруднительно или невозможно. Это позволяет значительно повысить точность анализа вариантов проектных решений и в десятки раз сократить путь от генерации идеи до ее воплощения в реальном промышленном производстве [102].

Точность результатов компьютерного моделирования во многом зависит от степени детализации сеток, используемых для проведения вычислительных экспериментов. На сегодняшний день размер сеток, используемых в задачах инженерного анализа, может составлять десятки миллионов элементов [1]. В связи с этим постоянно возрастает вычислительная сложность задач, и требуются значительные вычислительные ресурсы для выполнения инженерного моделирования. Решение этой проблемы заключается в использовании многопроцессорных систем. Практически все современные CAE-пакеты имеют параллельные реализации для многопроцессорных систем, в том числе и для систем с кластерной архитектурой.

На сегодняшний день процесс решения задач инженерного проектирования с использованием суперкомпьютерных ресурсов для рядового пользователя сопряжен с определенными трудностями. С одной стороны, от него требуется наличие специфических знаний, умений и навыков в области высокопроизводительных вычислений, таких как: архитектура суперкомпьютеров, навыки работы в Unix-подобных операционных системах,

настройка и администрирование удаленного доступа, умение работать с очередями приложений и др. С другой стороны, современные системы инженерного проектирования представляют собой многофункциональные программные комплексы, состоящие из множества отдельных программных подсистем со сложным пользовательским интерфейсом [32, 36]. Для решения задач инженерного проектирования пользователю требуется изучить интерфейс и особенности работы всех программных компонентов, входящих в технологический цикл решения задачи (формирование геометрии задачи, генерация вычислительной сетки, определение граничных условий, проведение компьютерного моделирования, визуализация и анализ результатов решения). Проблема сопряжения компонент существенно усложняется при использовании одновременно двух и более различных инженерных пакетов для решения одной задачи. Все эти факторы затрудняют широкое внедрение систем компьютерного инженерного проектирования в практику НИОКР.

Еще одним важным фактором, препятствующим быстрому внедрению систем инженерного проектирования на промышленных предприятиях, является высокая стоимость приобретения, владения и поддержки суперкомпьютерных систем. Для создания суперкомпьютерного центра по инженерному проектированию необходимо:

1) подготовить помещение и инфраструктуру для суперкомпьютерной системы;
2) подготовить персонал для поддержки и администрирования суперкомпьютерной системы;
3) приобрести суперкомпьютер, на базе которого будет производиться моделирование и анализ продукции;
4) приобрести лицензии на пакеты инженерного проектирования;
5) обучить пользователей работе с суперкомпьютером и инженерными пакетами.

Каждый этап данного процесса требует значительных материальных и людских ресурсов. По этой причине руководители предприятий часто ставят под сомнение целесообразность внедрения систем инженерного проектирования в заводских лабораториях.

Рациональной альтернативой созданию собственного суперкомпьютерного центра является аренда вычислительных и программных ресурсов в режиме удаленного доступа у центров коллективного пользования, функционирующих при крупных университетах, академических институтах и других организациях. Однако при этом возникает целый комплекс проблем, связанных с обеспечением безопасности вычислительных систем и данных.

Указанный комплекс проблем можно решить посредством применения концепции грид вычислений (Grid Computing) [60] и родственной ей концепции облачных вычислений (Cloud Computing) [69, 123] в соответствии с которыми, пользователю предоставляется конечный проблемно-ориентированный сервис, обеспечивающий решение задач на базе ресурсов распределенных вычислительных систем.

В соответствии с этим *актуальной* является задача разработки сервисно ориентированных методов использования систем инженерного проектирования и анализа в распределенных вычислительных средах. В настоящее время эффективные комплексные решения в этой области отсутствуют.

ЦЕЛЬ И ЗАДАЧИ ИССЛЕДОВАНИЯ

Цель данной работы: на основе концепции облачных вычислений разработать методы и алгоритмы, обеспечивающие автоматизированную генерацию проблемно-ориентированных грид-сервисов, позволяющих использовать программные системы для инженерного проектирования и анализа в распределенных вычислительных средах. Для достижения этой цели необходимо было решить следующие задачи:

1) разработать модель проблемно-ориентированного сервиса для решения задач инженерного проектирования и анализа в грид в виде РаВИС (Распределенного Виртуального Испытательного Стенда);
2) разработать архитектуру и принципы структурной организации РаВИС;
3) разработать методы и алгоритмы автоматизированного построения РаВИС и реализовать их в виде *программной системы CAEBeans*, обеспечивающей создание и использование РаВИС;
4) провести испытания системы CAEBeans путем создания и внедрения РаВИС на промышленном предприятии.

МЕТОДЫ ИССЛЕДОВАНИЯ

В исследованиях, проводимых в диссертационной работе, используются методы объектно-ориентированного программирования. Для проектирования систем и алгоритмов применяется аппарат UML.

НАУЧНАЯ НОВИЗНА

Научная новизна работы заключается в следующем:

1) предложен комплексный подход к интеграции ресурсов современных систем инженерного проектирования и анализа в распределенные вычислительные среды, обеспечивающий высокую степень автоматизации разработки и исполнения распределенных виртуальных испытательных стендов на базе концепции облачных вычислений;
2) разработана модель проблемно-ориентированного сервиса для решения задач инженерного проектирования и анализа;
3) предложена концепция распределенного виртуального испытательного стенда, обеспечивающая прозрачность предоставления конечному

пользователю ресурсов инженерных систем на базе распределенных вычислительных сред;

4) разработана программная система CAEBeans для автоматизированного создания и исполнения распределенных виртуальных испытательных стендов.

ТЕОРЕТИЧЕСКАЯ И ПРАКТИЧЕСКАЯ ЦЕННОСТЬ

Теоретическая ценность работы состоит в том, что в ней предложена концептуальная модель распределенного виртуального испытательного стенда, и на ее основе предложен комплекс методов и алгоритмов для представления ресурсов систем инженерного проектирования и анализа в виде грид-сервисов. *Практическая ценность* работы заключается в том, что программный комплекс CAEBeans, представленный в данной работе, может быть использован для автоматизированного создания и исполнения распределенных виртуальных испытательных стендов, обеспечивающих решение различных классов задач инженерного моделирования посредством вычислительных ресурсов, предоставляемых вычислительными грид-сетями.

СТРУКТУРА И ОБЪЕМ РАБОТЫ

Диссертация состоит из введения, четырех глав, заключения и библиографии. Объем диссертации составляет 124 страницы, объем библиографии – 125 наименований.

СОДЕРЖАНИЕ РАБОТЫ

Первая глава, «Инженерное проектирование в распределенных вычислительных средах», посвящена описанию и исследованию современных подходов к формированию распределенных вычислительных сред. Исследованы основные особенности распределенных вычислительных сетей, построенных на базе технологий распределенных объектов; одноранговых вычислений; грид; облачных вычислений. Производится обзор тех-

нологий предоставления ресурсов в распределенных вычислительных средах и выполняется анализ их применимости к задачам инженерного проектирования в качестве базовой технологии.

Во второй главе, «Технология CAEBeans», производится описание технологии CAEBeans, представляющей собой комплекс моделей, методов и алгоритмов, направленных на автоматизированное создание иерархий распределенных проблемно-ориентированных оболочек (Beans) над инженерными (CAE) пакетами на основе сервисно ориентированного подхода и концепции облачных вычислений. Раскрывается понятия задачи инженерного моделирования и распределенного виртуального испытательного стенда.

В третьей главе, «Система CAEBeans», рассматривается архитектура программного комплекса обеспечивающего разработку и исполнение распределенных виртуальных испытательных стендов. Приводится описание базовых сущностей системы CAEBeans. Рассматривается архитектура предложенной программной системы и основные модули: CAEBeans Toolbox, CAEBeans Portal, CAEBeans Server, CAEBeans Broker, CAE-ресурс.

В четвертой главе, «Испытания системы CAEBeans», приводится описание испытательных задач, на основе которых производились исследования возможности применения системы CAEBeans для решения реальных задач инженерного моделирования на базе различных базовых инженерных пакетов.

В заключении суммируются основные результаты диссертационной работы, выносимые на защиту, приводятся данные о публикациях и апробациях автора по теме диссертации, и рассматриваются направления дальнейших исследований в данной области.

ГЛАВА 1. ИНЖЕНЕРНОЕ ПРОЕКТИРОВАНИЕ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДАХ

На сегодняшний день, существует большое число технологий, обеспечивающих разработку и поддержку распределенных вычислительных систем. Каждая такая технология ориентирована на некоторый класс задач и обладает определенными достоинствами и недостатками. В этой главе дается обзор технологий предоставления ресурсов в распределенных вычислительных средах и выполняется анализ их применимости к задачам инженерного проектирования в качестве базовой технологии.

1.1. Технологии распределенных вычислений

Проблема выбора между централизованной и децентрализованной моделями предоставления вычислительных ресурсов является одной из центральных проблем организации вычислительных систем [47, 81]. До середины 70-х годов прошлого века по причине высокой стоимости телекоммуникационного оборудования и относительно слабой мощности вычислительных систем доминировала централизованная модель [95, 30]. В конце 70-х годов появление систем разделения времени и удаленных терминалов, явилось предпосылкой возникновения клиент-серверной архитектуры, обеспечивающей предоставление ресурсов мейнфреймов конечным пользователям посредством удаленного соединения [81]. Дальнейшее развитие телекоммуникационных систем и появление персональных компьютеров дало толчок развитию клиент-серверной парадигме обработки данных [97].

Согласно парадигме *клиент-серверной архитектуры* один или несколько клиентов и один или несколько серверов совместно с базовой операционной системой и средой взаимодействия образуют единую систему, обеспечивающую распределенные вычисления, анализ и представление данных [109]. Использование клиент-серверного подхода позволило поль-

зователю персонального компьютера получить доступ к различным ресурсам удаленных серверов, таких как базы данных, файлы, принтеры, процессорное время и др. [40]. Однако использование клиент-серверного подхода при разработке распределенных приложений было связано со значительными трудностями. Как правило, клиент-серверные системы основывались на закрытых протоколах информационного обмена, реализуемых посредством низкоуровневых сетевых протоколов (TCP/IP, NetBIOS), что затрудняло разработку таких систем и организацию взаимодействия между системами различных производителей [109]. Появление новых сервисов требовало обновление программного кода клиента и реализацию новых механизмов и протоколов для их использования [44]. Особые проблемы возникали при необходимости объединения приложений в гетерогенных вычислительных средах, состоящих из систем, реализованных на базе различных аппаратных и программных платформ [71].

Развитие клиент-серверной архитектуры в начале 1990-х привело к формированию *объектно-ориентированной концепции* распределенных систем, ориентированной на инкапсуляцию механизма распределенных взаимодействий и уменьшение сложности разработки распределенных приложений посредством методов объектно-ориентированной разработки и удаленных вызовов методов объектов [38]. Наиболее известными представителями данного направления являются технологии CORBA и Java RMI. Основными достоинствами данного подхода стали [39, 44]:

- простота разработки распределенных приложений по сравнению с классическим клиент/серверным подходом;
- возможность разработки приложений для гетерогенных вычислительных сред, обеспечиваемая виртуальной машиной Java и независимым описанием интерфейсов взаимодействующих компонентов;
- возможность отделения интерфейса удаленного объекта от его непосредственной реализации.

Однако использование объектного подхода было ориентировано на создание корпоративных систем, работающих в масштабе отдельной организации или предприятия [120]. Технические ограничения значительно затрудняли использование этих технологий для разработки гетерогенных распределенных вычислительных сред глобального масштаба. Развитие технологий распределенных вычислений в конце 1990-х годов привело к разработке концепции грид, которая позволила объединить географически-распределенные по всему миру гетерогенные ресурсы для решения сложных задач.

Архитектура *грид* ориентирована на стандартизованное совместное использование автономных географически-распределенных компьютерных ресурсов в зависимости от их доступности, производительности, цены и иных характеристик, важных для конечного пользователя [33]. Само понятие грид сформировалось на сопоставлении вычислительного грид (grid – с англ. сеть) с сетями электроснабжения [112]. На основе этого сопоставления были выведены следующие три основных требования, которым должны удовлетворять грид-системы [60].

1. *Гетерогенность*. Вычислительная среда грид состоит из множества различных ресурсов, обладающих различными характеристиками и параметрами.
2. *Масштабируемость*. Грид может состоять из произвольного числа ресурсов. Требуется учитывать, что при увеличении количества задействованных ресурсов, значительно возрастают накладные расходы на передачу данных между компьютерами.
3. *Адаптируемость*. При работе с грид-системами необходимо учитывать, что ошибки при работе с ресурсами – это не исключение, а правило. Среда Грид может состоять из сотен компьютеров, и ошибки в работе десятка из них не должны влиять на возможность получения результатов.

Грид-системы ориентированы на формирование виртуального пространства для прозрачного совместного использования распределенных ресурсов в рамках виртуальных организаций [122]. *Виртуальная организация* – это ряд людей и/или организаций, объединенных общими правилами коллективного доступа к определенным вычислительным ресурсам [61].

В 2000-2005 гг. произошло смещение тренда разработки распределенных приложений с объектно-ориентированного подхода на *сервисно ориентированную парадигму*, обеспечивающую организацию и использование распределенных сервисов, которые могут принадлежать различным владельцам [106]. Согласно [98] *сервисами* называются открытые, самоопределяющиеся программные компоненты, обеспечивающие прозрачную сетевую адресацию и поддерживающие быстрое построение распределенных приложений. В последствие сервисно ориентированный подход был принят разработчиками грид-систем [29, 52, 100] и реализован в виде архитектуры OGSA (Open Grid Service Architecture) [62].

Параллельно с этим, в начале 2000 г., с появлением систем Napster и gnutella, получила развитие концепция *P2P-сетей* (от англ. peer-to-peer – равный-к-равному), обеспечивающая формирование сетей на базе принципов децентрализации. P2P-сети в какой-то степени являются противоположностью тесно-связанным клиент-серверным сетевым архитектурам [23, 86, 117]. P2P – это разделение вычислительных ресурсов и сервисов посредством прямого обмена ресурсами между участниками сети [99]. В отличие от традиционной клиент-серверной архитектуры, в P2P-сетях каждый узел, входящий в вычислительную сеть, может являться как клиентом, так и сервером, предоставляя или используя ресурсы сети. Применение такого подхода позволило обойти основные ограничения клиент-серверных технологий, связанные со значительным возрастанием нагрузок на выделенные сервера при увеличении количества клиентов и значительно повы-

сить отказоустойчивость распределенной вычислительной среды за счет отсутствия «узких мест» в виде централизованных серверов [58].

В 2008 г. сформировалась новая концепция предоставления вычислительных ресурсов, названная «*облачными вычислениями*». До сих пор ведутся дискуссии о точном определении данной концепции и ее отличий от сервисно ориентированной архитектуры и грид [34, 69, 93, 118]. В [119] облаком называется пул виртуальных ресурсов (таких как аппаратное обеспечение, платформы разработки или сервисы), для которых обеспечены легкий доступ и простота использования. Этот пул может быть динамически реконфигурирован для поддержки меняющейся нагрузки, обеспечивая оптимальное использование ресурсов. Тарификация предоставляемых ресурсов производится на базе модели «оплата по факту потребления». Провайдер инфраструктуры гарантирует качество сервиса посредством соглашений об уровне предоставляемых услуг. Основным отличием облачных вычислений от концепции грид называют ориентированность грид на совместное использование ресурсов в рамках виртуальной организации, тогда как облако предоставляет ресурсы произвольному пользователю и ориентировано на коммерческое представление компьютерных ресурсов [34, 119].

1.1.1. CORBA

CORBA (Common Object Request Broker Architecture - общая архитектура брокера объектных запросов) [1, 26, 96] – это технология разработки распределенных приложений, ориентированная на интеграцию распределенных изолированных систем. Основные компоненты, составляющие архитектуру CORBA, представлены на рисунке 1.

Технологический стандарт CORBA определяет язык IDL [3, 72], применяемый для унифицированного описания интерфейсов распределенных объектов, и его отображения на языки Ada, C, C++, Java, Python, COBOL,

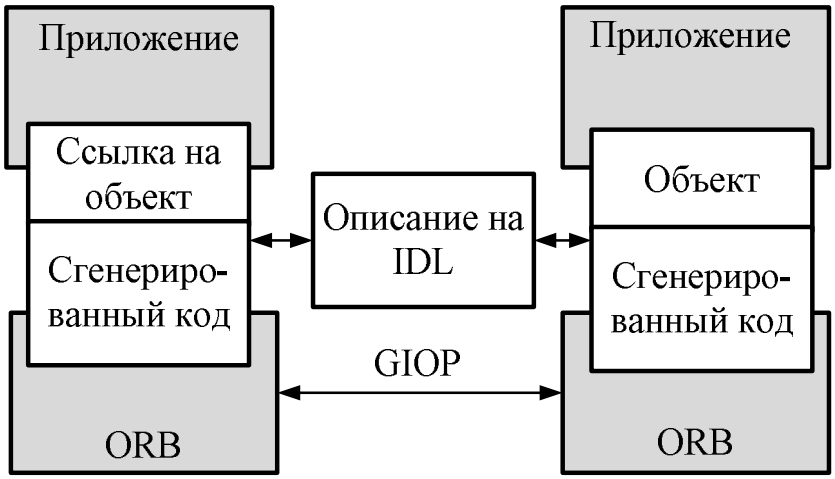


Рис. 1. Ядро архитектуры CORBA.

Lisp, PL/1 и Smalltalk [22]. Для преобразования описания интерфейса на языке IDL на требуемый язык программирования используется специальный компилятор [125, 117]. В дальнейшем построенный с его помощью программный код может быть преобразован любым стандартным компилятором в исполняемый код.

Главной особенностью CORBA является использование компонента ORB (Object Resource Broker - брокер ресурсов объектов) для создания экземпляров объектов и вызова их методов. Данный компонент формирует «мост» между приложением и инфраструктурой CORBA [94]. ORB поддерживает удаленное взаимодействие с другими ORB, а также обеспечивает управление удаленными объектами, включая учет количества ссылок и времени жизни объекта [105]. Для обеспечения взаимодействия между ORB используется протокол GIOP (General Inter-ORB Protocol - общий протокол для коммуникации между ORB) [79]. Наиболее распространенной реализацией данного протокола является протокол IIOP (Internet Inter-ORB Protocol - протокол взаимодействия ORB в сети интернет), обеспечивающий отображение сообщений GIOP на стек протоколов TCP/IP [41, 72].

По сравнению с классическим клиент-серверным подходом, использование технологии CORBA для разработки распределенных приложений имеет следующие преимущества:

- использование IDL для описания интерфейсов позволяет разрабатывать программные компоненты независимо от языка программирования и базовой операционной системы;
- поддержка богатой инфраструктуры распределенных объектов;
- прозрачность вызова удаленных объектов.

Однако программные решения на базе технологии CORBA редко выходят за рамки отдельных предприятий. Разработка крупномасштабных меж-организационных систем на базе технологии CORBA сопряжена со следующими трудностями:

- плохая совместимость различных реализаций технологии CORBA от различных поставщиков [39];
- проблемы взаимодействия узлов CORBA через Интернет [70];
- несогласованность многих архитектурных решений CORBA и отсутствие компонентной модели, которая могла бы значительно упростить разработку [71].

На смену технологии CORBA, пришли стандартизованные протоколы веб-сервисов, такие как XML, WSDL, SOAP и др. [70]. В настоящее время CORBA используется для реализации узкого круга приложений и является фактически нишевой технологией [71].

1.1.2. Java RMI

Технология *Java RMI (Remote Method Invocation – удаленный вызов метода)* [94] позволяет обеспечить прозрачный доступ к методам удаленных объектов, обеспечивая доставку параметров вызываемого метода, сообщение объекту о необходимости выполнения метода и передачу возвращаемого значения клиенту обратно [68].

Распределенное приложение, разработанное на базе технологии Java RMI, состоит из двух отдельных программ: клиента и сервера. Серверное приложение создает удаленный объект, публикует ссылки на него и ожида-

ет, когда клиенты произведут вызов метода данного удаленного объекта. Приложение-клиент получает с сервера ссылку на удаленный объект на сервере, после чего может вызывать его методы. Технология RMI обеспечивает механизм, при помощи которого производится обмен информацией между клиентом и сервером [77]. Процесс публикации ссылки на удаленный объект может быть реализован с помощью специального регистра или же посредством передачи удаленной объектной ссылки как части обычной операции.

Достоинствами использования технологии Java RMI для разработки распределенного приложения можно назвать возможность разрабатывать систему целиком основываясь на объектно-ориентированной концепции, не погружаясь в разработку собственных протоколов взаимодействия между распределенными компонентами систем, а также кроссплатформенность, предоставляемую виртуальной машиной Java. К недостаткам данного подхода можно отнести:

– строгую ограниченность данной технологии платформой Java;
– необходимость обработки соединений между распределенными компонентами приложения ограничивает масштабируемость используемого подхода.

1.1.3. OGSA

Термин «грид» был введен в обращение в начале 1998 года публикацией книги «Грид. Новая инфраструктура вычислений» [60]. Хотя в последнее десятилетие базовая идея грид не претерпела существенных изменений, всеобъемлющего определения грид не существует до сих пор [113].

С точки зрения архитектуры приложение грид состоит из множества различных компонентов, отвечающих за те или иные аспекты функционирования. Например, типовое приложение грид может содержать следующие сервисы [111]:

1) сервис управления виртуальными организациями;
2) сервис поиска и управления ресурсами;
3) сервис управления заданиями.

Ключевым моментом в разработке грид приложений является *стандартизация*, позволяющая организовать поиск, использование, размещение и мониторинг различных компонентов, составляющих единую виртуальную систему, даже если они предоставляются различными поставщиками услуг или управляются различными организациями [62].

В качестве базы для создания стандарта архитектуры грид приложений была выбрана сервисно ориентированная концепция и технология веб-сервисов. Данный выбор был обусловлен двумя основными достоинствами данной технологии. Во-первых, язык описания интерфейсов веб-сервисов WSDL (Web Service Definition Language) поддерживает стандартные механизмы для определения интерфейсов отдельно от их реализации, что в совокупности со специальными механизмами связывания (транспортным протоколом и форматом кодирования данных) обеспечивает возможность динамического поиска и компоновки сервисов в гетерогенных средах. Во-вторых, широко распространенная адаптация механизмов веб-сервисов означает, что инфраструктура, построенная на базе веб-сервисов, может использовать различные утилиты и другие существующие сервисы, такие как различные процессоры WSDL, системы планирования потоков задач и среды для размещения веб-сервисов [51].

Разработанный стандарт архитектуры грид получил название *OGSA* (Open Grid Services Architecture – Открытая Архитектура Грид Сервисов) [53]. Он основывается на понятии грид-сервиса. *Грид-сервисом* называется сервис, поддерживающий предоставление полной информации о текущем состоянии (потенциально временного) экземпляра сервиса, а также поддерживающий возможность надежного и безопасного исполнения, управления временем жизни, рассылки уведомлений об изменении со-

стояния экземпляра сервиса, управления политикой доступа к ресурсам, управления сертификатами доступа и виртуализации [51]. Грид-сервис поддерживает следующие стандартные интерфейсы.

– *Поиск*. Грид приложениям необходимы механизмы для поиска доступных сервисов и определения их характеристик.
– *Динамическое создание сервисов*. Возможность динамического создания и управления службами – это один из базовых принципов OGSA, требующий наличия сервисов создания новых сервисов.
– *Управление временем жизни*. Распределенная система должна обеспечивать возможность уничтожения экземпляра грид-сервиса.
– *Уведомление*. Для обеспечения работы грид приложения наборы грид сервисов должны иметь возможность асинхронно уведомлять друг друга о изменениях в их состоянии.

Первая реализация модели OGSA, разработанная в 2003 г., называлась *OGSI* (Open Grid Service Infrastructure). В связи с тем, что существовавшие тогда стандарты веб-сервисов (к которым относились WSDL, SOAP, UDDI) не могли обеспечить всех требований, предъявляемых разработчиками к функциональным возможностям грид-сервисов, при создании OGSI потребовалось модифицировать и расширить соответствующие стандарты [55]. Это привело к тому, что совместное использование веб-сервисов и грид-сервисов в одной среде стало невозможным, из-за несовместимости базовых стандартов [19].

Дальнейшие совместные усилия сообщества грид и организаций по разработке стандартов веб-сервисов привело к определению стандартов, соответствующих требованиям грид. В предложенном стандарте *WSRF* (*Web Service Resource Framework*) [43, 54, 57, 116] специфицированы универсальные механизмы для определения, просмотра и управления состоянием удаленного ресурса, что является критически-важным с точки зрения

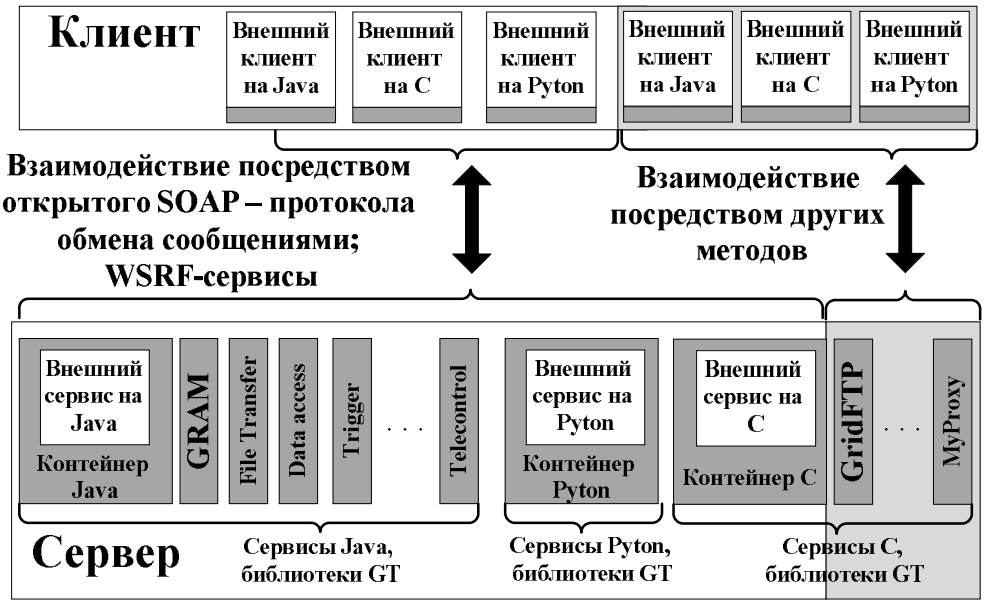


Рис. 2. Общая схема взаимодействия компонентов Globus Toolkit 4.0.

грид [63]. На сегодняшний день реализация модели OGSA посредством стандарта WSRF (и сопутствующих стандартов, таких как WS-Notification и WS-Addressing) является наиболее распространенной в среде грид.

В настоящее время, существуют две системы, обеспечивающие инфраструктуру разработки грид-систем в соответствии со стандартами OGSA, реализованными посредством WSRF: Globus Toolkit [56] и UNICORE [90].

Проект Globus начал разрабатываться в конце 1990-х годов для создания и поддержки сервисно ориентированной инфраструктуры для грид-вычислений [59]. В составе системы Globus Toolkit 4.0 [56], первой системы грид-вычислений, обеспечившей полноценную поддержку стандарта WSRF, можно выделить следующие функциональные группы (см. рис. 2):

– базовые сервисы (GRAM – управление ресурсами; File Transfer, GridFTP - передача файлов; Trigger, Index – поиск и каталог ресурсов и др.);
– контейнеры для пользовательских сервисов, поддерживающие аутентификацию, управление состоянием, поиск и т.п. обеспечивающие поддержку стандартов WSRF, WS-Security, WS-Notification;

– библиотеки, для обеспечения взаимодействия сторонних приложений с GTK 4.0 и/или пользовательскими сервисами.

Проект UNICORE (Uniform Interface to Computing Resources – единый интерфейс к вычислительным ресурсам) зародился в 1997 году, и к настоящему моменту представляет собой комплексное решение, ориентированное на обеспечение прозрачного безопасно5го доступа к ресурсам грид [114].

Архитектура UNICORE 6 [115] формируется из клиентского, сервисного и системного слоев (см. рис. 3). Верхним слоем в архитектуре является клиентский слой. В нем располагаются различные клиенты, обеспечивающие взаимодействие пользователей с грид средой:

– UCC (Unicore Command Line Client – клиент командной строки для UNICORE): клиент, обеспечивающй интерфейс командной строки для постановки задач и получения результатов;
– URC (Unicore Rich Client – многофункциональный клиент UNICORE): клиент, основанный на базе интерфейса среды Eclipce, предоставляет в графическом виде полный набор всех функциональных возможностей системы UNICORE;
– HiLA (High Level API for Grid Applications – высокоуровневый программный интерфейс для приложений грид): обеспечивает разработку клиентов к системе UNICORE;
– Порталы: доступ пользователей к грид-ресурсам через интернет, посредством интеграции UNICORE и систем интернет-порталов.

Промежуточный сервисный слой содержит все сервисы и компоненты системы UNICORE, основанные на стандартах WSRF и SOAP. Шлюз – это компонент, обеспечивающий доступ к узлу UNICORE посредством аутентификации всех входящих сообщений [90]. Компонент XNJS обеспечивает управление задачами и исполнение ядра UNICORE 6. Регистр сервисов

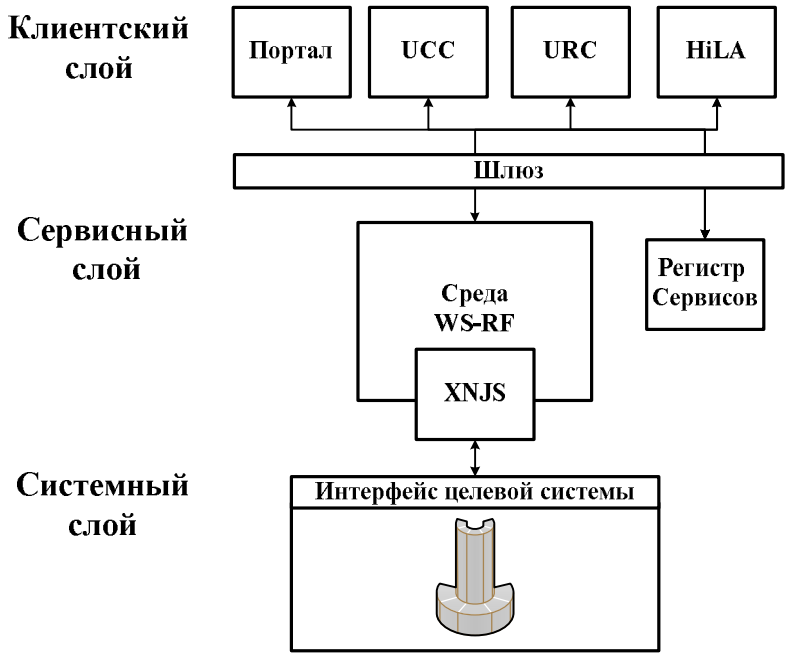


Рис. 3. Архитектура UNICORE 6.

обеспечивает регистрацию и поиск ресурсов, доступных в грид-среде. Также, на уровне сервисного слоя обеспечивается поддержка безопасных соединений, авторизации и аутентификации пользователей.

В основании архитектуры UNICORE лежит системный слой. Интерфейс целевой системы (TSI – Target System Interface) обеспечивает взаимодействие между UNICORE и отдельным ресурсом грид-сети. Он обеспечивает трансляцию команд, поступающих из грид-среды локальной системе.

Основным достоинством использования системы UNICORE 6 для разработки распределенных вычислительных систем можно считать наличие богатого арсенала различных клиентов, обеспечивающих взаимодействие пользователя с ресурсами вычислительной сети, а также развитых средств обеспечения безопасности при разработке грид-приложений.

1.1.4. P2P-технологии (одноранговые сети)

Технология P2P (анг. peer-to-peer - равный-к-равному) обеспечивает формирование сетей на базе принципов децентрализации. Одноранговые вычислительные сети, в какой-то степени, являются противоположностью тесно-связанным клиент-серверным архитектурам (таким как CORBA,

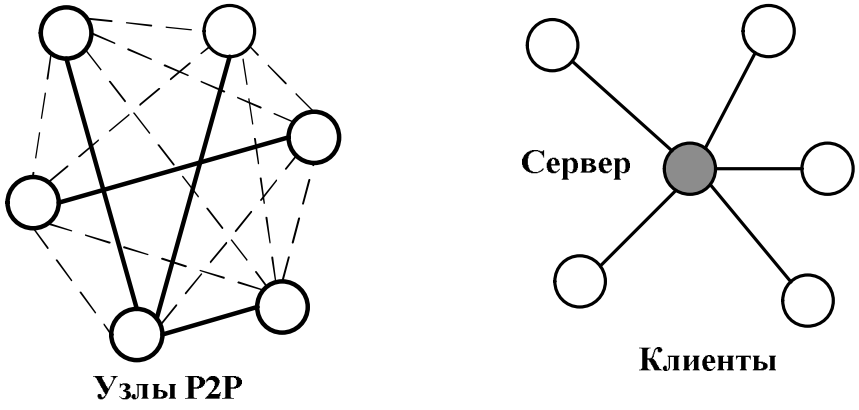


Рис. 4. Сравнение P2P и централизованной (клиент-серверной) архитектур.

RMI) [117]. P2P – это разделение вычислительных ресурсов и сервисов посредством прямого взаимодействия между участниками сети [99]. В отличие от традиционной клиент-серверной архитектуры в P2P-сетях каждый узел, входящий в вычислительную сеть, может являться как клиентом, так и сервером, предоставляя или используя ресурсы сети. На рис. 4 представлены связи в сетях с P2P и с централизованной архитектурой [92].

Основными преимуществами одноранговых вычислений являются:

- высокая масштабируемость, связанная с равномерным распределением вычислительной нагрузки на всех участников сети;
- стабильность работы сети, обусловленная отсутствием «узкого места» - выделенного сервера, обрабатывающего все сетевые запросы;
- возможность объединения ресурсов отдельных участников сети, и их предоставление другим участникам;
- распределение совокупных затрат на предоставление ресурсов между участниками сети.

В настоящее время наблюдается значительный скачек в развитии одноранговых сетей. По статистическим данным на конец 2006 года объем трафика, генерируемого файлообменными сетями на базе P2P-сетей, составил более 70% всего сетевого трафика [49]. Наибольшее распространение одноранговых сетей наблюдается в системах, обрабатывающих большие

объемы данных и обеспечивающих индивидуальный обмен информацией между пользователями:

- системы хранения и разделения информации (Gnutella [66], eDonkey, BitTorrent [27]);
- системы распределенных вычислений (проекты на платформе Boinc [28]);
- системы IP-телефонии (Skype [67,110]);
- системы трансляции видео (Joost [24], PPLive [86]);
- системы поддержки совместной работы (Groove [50]).

Структура P2P сети определяет принципы поиска новых узлов и замены узлов вышедших из состава сети новыми. Можно выделить два основных типа P2P сетей: централизованные и децентрализованные [86].

Централизованная структура P2P сети подразумевает наличие выделенного индексного сервера (трекера) собирающего информацию об узлах, входящих в P2P-сеть и обеспечивающего поиск и предоставление необходимых сервисов одних узлов другим. Примером централизованной структуры является сеть BitTorrent. Центральным узлом данной сети является трекер, содержащий информацию о списке узлов, подключенных к сети, и сервисах, предоставляемых каждым узлом (список файлов, доступных для загрузки с данного узла). Для получения необходимого файла, узел посылает трекеру запрос, содержащий уникальный идентификатор необходимого файла. На данный запрос трекер возвращает список узлов, на которых доступен требуемый файл.

Децентрализованная структура P2P сети предполагает отсутствие выделенного сервера. Поиск и предоставление сервисов производится путем процедуры пошагового поиска, в которой могут участвовать все узлы, входящие в сеть. Типичным примером одноранговой сети с децентрализованной структурой является система gnutella. В данной сети, обнаружение и подключение к узлам сети происходит посредством процедуры случайного

обхода. Каждый узел содержит таблицу соседей, содержащую IP адрес и порт известного узла gnutella. При запуске новый узел gnutella переходит в режим начальной загрузки, в котором посредством одного из доступных источников (список узлов на одном из узлов интернет; внутренний предустановленный список узлов и др.) формирует начальный список соседей. После чего, соседям высылается сообщение обнаружения, пересылаемое далее по цепочке систем. Таким образом обеспечивается обнаружение ресурсов, предоставляемых всеми узлами подключенными к сети.

Архитектурные особенности одноранговых сетей в комплексе с заложенными в них механизмами функционирования обуславливают определенные недостатки, присущие таким сетям:

- в одноранговых сетях не может быть обеспечено гарантированное качество обслуживания: любой узел, предоставляющий те или иные сервисы, может быть отключен от сети в любой момент;
- индивидуальные технические характеристики узла, могут не позволить полностью использовать ресурсы P2P сети;
- при работе того или иного узла через брандмауэр может быть значительно снижена пропускная способность передачи данных в связи с необходимостью использования специальных механизмов обхода;
- участниками одноранговых сетей в основном являются индивидуальные пользователи а не организации, в связи с чем возникают вопросы безопасности предоставления ресурсов: владельцы узлов P2P-сети, сокрее всего, не знакомы друг с другом лично, предоставление ресурсов происходит без предварительной договоренности;
- при значительном увеличении числа участников P2P сети возникает большая нагрузка на сеть (как с централизованной, так и с децентрализованной структурой).

Таким образом, одноранговые сети ориентированы на анонимное предоставление ресурсов одних индивидуальных пользователей другим.

Они не ориентированы на безопасное, стандартизованное предоставление вычислительных ресурсов крупных организаций и сообществ.

1.2. Системы совместного проектирования

Интеграция систем инженерного проектирования в распределенные вычислительные среды посредством сервисно ориентированной концепции, в настоящее время является одним из основных направления развития отрасли инженерного анализа [42]. Перспективы применения сервисно ориентированной архитектуры для построения инфраструктуры систем инженерного проектирования хорошо согласуются с распределенным характером этих систем, и позволяет обеспечить «бесшовное» сопровождение разрабатываемого продукта на всех этапах его жизненного цикла, начиная с концепции дизайна, и заканчивая его утилизацией и переработкой [84].

В мировом научном сообществе считается перспективным направление, связанное с применением грид-технологий для решения ресурсоемких задач инженерного анализа. В тоже время наблюдается тенденция перехода от разработки специализированных грид-ориентированных систем инженерного проектирования к интеграции существующих классических CAE-систем в сервисно ориентированные грид-среды [9]. Существуют специальные надстройки для системы UNICORE, позволяющие осуществлять удаленную постановку задачи инженерным и научным системам Gaussian, Nastran, Fluent, Star-CD [82].

1.2.1. Распределенная оптимизация инженерных систем

В последнее десятилетие было проведено множество большое число исследований по разработке систем, ориентированных на различные аспекты внедрения пакетов инженерного проектирования в распределенные вычислительные среды. В работах [73, 80, 85, 124] представлены подходы к разработке грид-сервисов, обеспечивающих проведение численной оптимизации параметров инженерных систем на базе ресурсов, предоставляемых

гетерогенными распределенными вычислительными сетями. Принципы массового параллелизма, заложенные в алгоритмах многокритериальной оптимизации, позволяют достичь высоких показателей эффективности использования разнородных вычислительных систем. Применение модели грид-сервисов позволило обеспечить взаимодействие между различными целевыми системами, независимо от языков программирования и базовых операционных систем. Однако в этих работах не рассматривается возможность интеграции в грид CAE-пакетов, а также отсутствует комплексный подход по переносу всего технологического цикла решения задачи инженерного проектирования и анализа в распределенную вычислительную среду.

1.2.2. Потоки задач в инженерном проектировании

В настоящее время наибольшее распространение для организации взаимодействия распределенных грид-сервисов получил подход, основанный на понятии потока задач (Workflow) [76]. В работе [65] предлагается следующее определения *потока задач*: автоматизация процессов, заключающаяся в объединении грид-сервисов для решения определенной задачи или для определения нового сервиса. В работе [36] была предложена архитектура системы, поддерживающей совместную работу нескольких групп инженеров над задачами инженерного проектирования посредством грид (на примере задач моделирования соударения). Основным протоколом организации взаимодействия грид-сервисов в гетерогенной вычислительной среде был выбран стандартный протокол описания потоков задач BPEL4WS (Business Process Execution Language for Web Services) [31]. Основным недостатком данного подхода является жесткая детерминированность потока задач, не позволяющая обеспечить динамическое предоставление требуемых вычислительных ресурсов в зависимости от конкретных входных данных.

В связи со специфическим характером потоков задач, возникающих в процессе инженерного проектирования, в работе [78] представлена концепция взаимодействия компонентов комплекса инженерного проектирования, основанная на потоках данных, возникающих между ними в процессе решения задач. Потоки задач в области CAE фактически определяются потоками данных. Обеспечивается передача данных от постоянного хранилища на вычислительный ресурс, обрабатывающий эти данные; производится разбиение данных для подготовки к параллельной обработке; производится трансформация данных, когда различные шаги обработки требуют различных форматов данных.

1.2.3. Внедрение инженерных систем в распределенные среды

В работах [45, 75] описываются системы инженерного проектирования и анализа, построенные на технологии CORBA, обеспечивающие совместную распределенную работу пользователей над единым проектом. В качестве основного недостатка такого подхода можно отметить, что технология CORBA не была ориентирована на интеграцию систем инженерного проектирования в распределенную вычислительную среду, в связи с чем требуется разработка версий инженерных систем на базе этой технологии.

В работах [37, 91] приводится пример разработки систем совместного проектирования на базе технологии удаленного вызова методов Java RMI. Достоинством данного подхода является независимость от платформы исполнения и простота организации доступа к таким системам посредством веб-доступа, что вытекает из использования базовой технологии Java. Но недостатком данного подхода является отсутствие стандартной инфраструктуры, которая позволила бы обеспечить унифицированные механизмы интеграции ресурсов в гетерогенные вычислительные среды.

В работах [48, 121] раскрываются аспекты разработки системы совместного проектирования, ориентированной на решение задач инженерно-

го проектирования в динамических распределенных группах разработчиков. В связи с потребностью постоянного взаимодействия между участниками команды инженеров в процессе разработки, в качестве базовой платформы данной системы была выбрана концепция P2P-вычислений. Такой подход позволяет обеспечить беспрепятственный обмен информацией во время разработки проекта, прозрачно использовать динамические вычислительные ресурсы коллектива разработчиков. Но применение такого подхода ориентировано не на предоставление конечному пользователю готового продукта, предоставляющего ресурсы распределенной сети, а на поддержку процесса совместного проектирования командой разработчиков. Такие системы не ориентированы на предоставление проблемно-ориентированного интерфейса для решения конкретного класса задач инженерного проектирования.

В работах [87, 88] представлена концепция грид-среды для совместного проектирования (Collaborative Manufactoring Grid) система совместной разработки, основанная на стандартах грид-систем. Представлены уровни взаимодействия разработчиков на уровне CAD/CAE/CAM сервисов. Взаимодействие с системой реализовано в виде грид-портала. Описание взаимодействия компонентов системы инженерного проектирования реализовано посредством потоков задач. Реализованы интерфейсы постановки задач, обертывающие классические системы инженерного проектирования. Реализована возможность решения задач инженерного моделирования посредством указания параметров моделирования и получения конечных результатов через веб-портал. В качестве недостатков этой системы можно отметить отсутствие стандартизированного подхода к созданию виртуальных испытательных стендов; отсутствие возможности конкретизации проблемно-ориентированных оболочек, обеспечивающих «подгонку» задач к конкретным требованиям пользователя; отсутствие реализации стандарта WSRF. Также, отсутствует брокер ресурсов обеспечивающий прозрачное

для пользователя предоставление ресурсов распределенной вычислительной среды, наиболее соответствующих требованиям текущей задачи.

1.3. Выводы по главе 1

Проведенный анализ литературы показывает, что на сегодняшний день отсутствуют универсальные системы, обеспечивающие прозрачное внедрение ресурсов систем инженерного проектирования в распределенные вычислительные среды. В соответствии с этим остается актуальной задача разработки комплексного технологического подхода для интеграции систем инженерного проектирования и анализа в грид на базе сервисно ориентированной концепции с поддержкой всех этапов создания и использования распределенных инженерных приложений. В качестве платформы для реализации системы, целесообразно использовать технологию грид-вычислений на базе системы UNICORE 6.

ГЛАВА 2. ТЕХНОЛОГИЯ CAEBEANS

В данной главе описывается технология CAEBeans, представляющей собой комплекс моделей, методов и алгоритмов, направленных на автоматизированное создание иерархий распределенных проблемно-ориентированных оболочек (Beans) над инженерными (CAE) пакетами на основе сервисно ориентированного подхода и концепции облачных вычислений (cloud computing).

2.1. Основные концепции технологии CAEBeans

2.1.1. Задача инженерного моделирования

Задача инженерного моделирования включает в себя:

- геометрическую модель объекта исследования и (или) вычислительную сетку, разбивающую моделируемую область на дискретные подобласти;
- граничные условия, физические характеристики и параметры взаимодействия отдельных компонентов исследуемой области;
- описание неизвестных величин, значения которых требуется получить в результате решения задачи;
- требования к аппаратным и программным ресурсам, обеспечивающим процесс моделирования.

Параметром задачи инженерного моделирования назовем величину, значение которой влияет на конечный результат моделирования и может варьироваться в определенном диапазоне значений. Каждый параметр задачи инженерного моделирования имеет определенную семантику, отражая свойство проблемной области, либо некоторую особенность процесса решения задачи [14].

Полным дескриптором задачи назовем множество параметров, однозначно описывающих задачу инженерного моделирования.

Для решения задачи инженерного моделирования, пользователь должен пройти сложный технологический цикл, который обычно включает в себя следующие этапы:

– подготовка геометрии модели;
– построение вычислительной сетки;
– определение физических свойств, граничных условий и протекающих процессов в анализируемой области;
– численное решение поставленной задачи моделирования;
– визуализация и анализ полученных результатов.

Каждый из этих этапов может быть реализован средствами различных инженерных пакетов. Например, геометрия задачи может быть выполнена в пакете SolidWorks, после чего отправлена в систему ICEM CFD для генерации качественной сетки для дальнейшего решения задачи средствами пакета ANSYS CFX. В соответствии с этим для решения задачи инженерного моделирования необходимо скоординировать совместную работу множества программных компонентов, отвечающих за различные этапы технологического цикла. *Логическим планом* решения задачи инженерного моделирования назовем ориентированный граф, в вершинах которого могут находиться *узлы* двух типов:

– *действия*, выполняемые отдельными инженерными пакетами;
– *узлы управления* потоком решения задачи.

С каждым узлом логического плана связаны два подмножества параметров полного дескриптора задачи

– множество параметров, определяющих входные данные для узла;
– множество параметров, в которых сохраняются результаты работы узла.

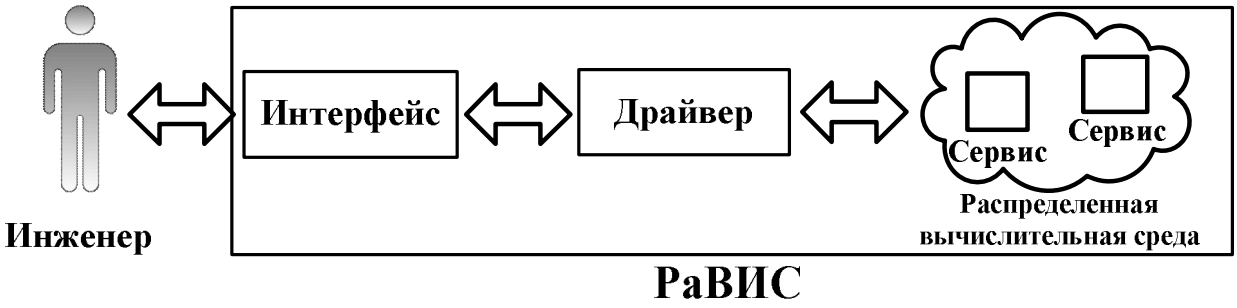


Рис. 5. Обобщенная схема распределенного виртуального испытательного стенда.

Определим *класс задач инженерного моделирования* как множество задач, у которых совпадает структура полного дескриптора задачи и для которых можно описать единый логический план, приводящий к решению.

2.1.2. Распределенный виртуальный испытательный стенд

Внедрение ресурсов CAE-пакетов в распределенные вычислительные среды основывается на концепции распределенного виртуального испытательного стенда. *Распределенный виртуальный испытательный стенд (РаВИС)* – это программно-аппаратный комплекс, обеспечивающий проведение работ инженерного моделирования в распределенной вычислительной среде в рамках определенного класса задач.

РаВИС включает в себя (см. рис. 5):

- *интерфейс*, обеспечивающий постановку определенного класса задач инженерного моделирования;
- *драйвер*: набор программных средств, обеспечивающих использование сервисов распределенной вычислительной среды для проведения виртуального эксперимента;
- *сервисы распределенной вычислительной среды*: множество вычислительных систем, входящих в распределенную вычислительную среду, в совокупности с установленными на них программными компонентами, обеспечивающими решение задач инженерного моделирования

и поддерживающими безопасные стандартизованные методы удаленного взаимодействия.

Пользовательский интерфейс РаВИС обеспечивает возможность *проблемно-ориентированной* постановки конкретного класса задач инженерного моделирования. Проблемно-ориентированная постановка задачи позволяет сделать прозрачной гетерогенную структуру распределенной вычислительной среды и абстрагировать пользователя от механизмов решения задачи посредством ресурсов, предоставляемых удаленными грид-сервисами. При постановке и решении задачи инженерного моделирования пользователю предоставляется возможность взаимодействовать с РаВИС в терминах проблемной области посредством указания значений параметров, описывающих интересующий его класс задач.

Драйвер РаВИС выполняет следующие действия:

- автоматизирует процесс декомпозиции задачи на типовые действия;
- обеспечивает поиск вычислительных ресурсов, обеспечивающих оптимальное решение поставленной задачи в распределенной вычислительной среде;
- анализирует и отслеживает лицензии на программное обеспечение для инженерного моделирования, доступные сервисам распределенной вычислительной среды;
- обеспечивает безопасное соединение и обмен информацией между сервисами распределенной вычислительной среды;
- производит постановку, мониторинг и получение результатов решения задач на удаленных вычислительных сервисах.

Сервисы распределенной вычислительной среды представляют собой грид-сервисы, реализованные на базе архитектуры OGSA [62], доступ к которым организуется посредством стандартов WSRF [43, 116]. Сервисы предоставляют безопасный стандартизованный доступ к ресурсам CAE-

пакетов, установленных на вычислительных системах распределенной вычислительной среды.

Процессы разработки и функционирования РаВИС определяются технологией CAEBeans. *Технология CAEBeans* – это совокупность теории и практической техники, на которые опирается процесс создания и использования распределенных виртуальных испытательных стендов. Технология CAEBeans включает в себя:

1) *концептуальные средства,* которые определяют методы разработки и структуру РаВИС;
2) *организационные средства*, которые определяют форму труда и распределение обязанностей в команде разработчиков и пользователей РаВИС;
3) *программные средства* разработки и среду исполнения РаВИС.

2.2. Архитектура CAEBeans

Оболочка CAEBean – это основная структурная единица, формирующая РаВИС. В соответствии с технологией CAEBeans выделяются четыре слоя структуры РаВИС, каждый из которых представляется своим типом оболочек CAEBeans (см. рис. 6) [13]:

1) концептуальный слой (проблемные CAEBeans);
2) логический слой (потоковые CAEBeans);
3) физический слой (компонентные CAEBeans);
4) системный слой (системные CAEBeans).

2.2.1. Концептуальный слой

Концептуальный слой РаВИС формируется на основе оболочек CAEBeans, которые мы будем называть *проблемными*. Пользовательский интерфейс, предоставляемый проблемным CAEBean, является основным средством взаимодействия пользователя с системой CAEBeans. Посредством проблемного CAEBean, ориентированного на решение конкретного

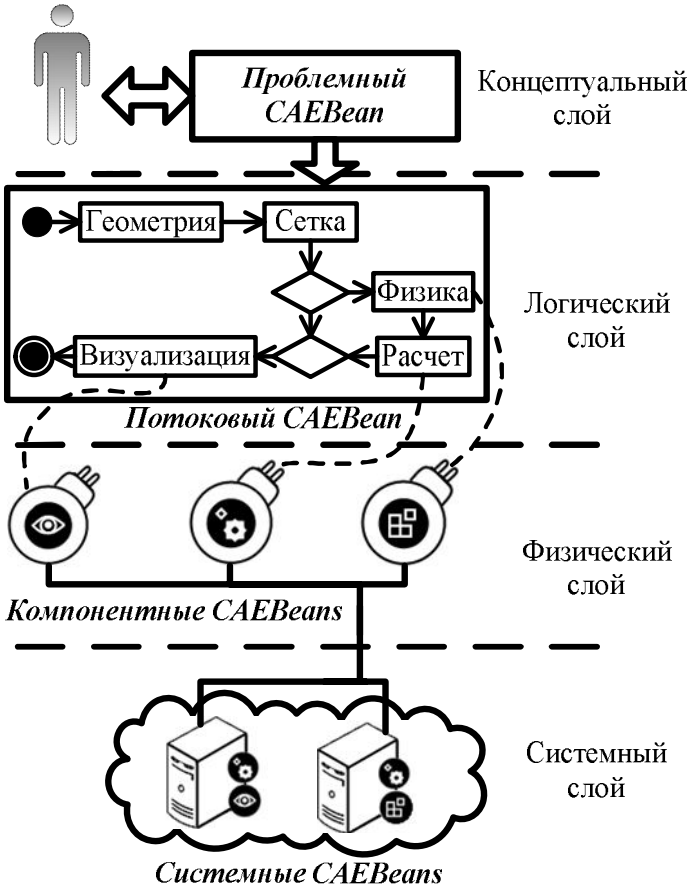


Рис. 6. Обобщенная схема слоев РаВИС.

класса задач инженерного моделирования, пользователь может произвести постановку задачи; проследить за ходом решения поставленной задачи; получить результаты решения.

При постановке задачи посредством проблемного CAEBean пользователю предоставляется возможность оперировать терминами той проблемной области, в рамках которой выполняется решение задачи. Как отмечалось в п. 2.1.1, каждая задача инженерного моделирования может быть описана некоторым набором значений входных параметров. В качестве примера таких параметров можно привести следующие: температура жидкости, протекающей в системе труб при моделировании трубопровода; шаг и профиль резьбы при моделировании резьбового соединения труб; скорость поступательного движения и скорость вращения трубы при моделировании процесса закалки.

Параметры задачи инженерного моделирования можно разделить на несколько категорий:

– *проблемные параметры* описывают задачу инженерного моделирования в терминах предметной области;
– *параметры вычислительной среды* определяют желаемые характеристики среды исполнения виртуального эксперимента: требования к аппаратным ресурсам, минимальное время ожидания отклика удаленного сервиса, конкретные версии программного обеспечения и лицензий;
– *служебные параметры:* значения этих параметров не устанавливаются пользователем, а формируются автоматически в процессе исполнения РаВИС. Они используются для обмена промежуточными данными между различными сервисами, а также для передачи результатов решения пользователю.

Интерфейс, предоставляемый пользователю проблемным CAEBean, ориентирован на решение определенного класса задач инженерного моделирования. С его помощью можно произвести постановку задачи посредством указания значений параметров, характеризующих данный класс задач [16]. Таким образом обеспечивается проблемно-ориентированная постановка задачи, не зависящая от функциональных особенностей программных продуктов, на основе которых будет производиться ее решение. Проблемный CAEBean скрывает от пользователя «лишнюю» функциональность CAE-пакета, распределенный характер вычислительной среды, структуру аппаратных, программных и лицензионных ресурсов [12]. Пользователю достаточно освоить простой интерфейс проблемного CAEBean для того, чтобы иметь возможность поставить и решить задачу инженерного моделирования с помощью сервисов грид-среды.

Проблемно-ориентированный пользовательский интерфейс решения задачи очень трудно разработать таким образом, чтобы он подходил под требования всех возможных пользователей системы одновременно. Для решения данной проблемы была предложена концепция *дерева проблемных*

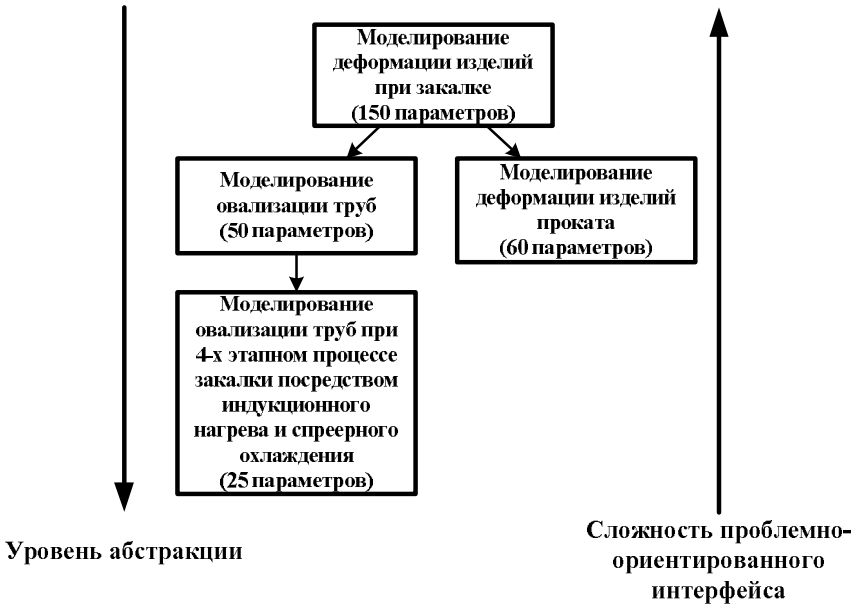


Рис. 7. Пример иерархии проблемных оболочек.

оболочек [13, 14]. Она позволяет адаптировать проблемно-ориентированный интерфейс конкретного класса задач инженерного моделирования под нужды определенной категории пользователей. В основе данного подхода лежит процесс формирования иерархии классов задач и построения дерева проблемных оболочек CAEBeans на основе этой иерархии (см. рис. 7).

Корневой элемент дерева, задающего иерархию проблемных оболочек, представляется проблемной оболочкой, соответствующей наиболее широкому классу решаемых задач. С одной стороны, она обеспечивает решение наиболее широкого класса задач инженерного моделирования, но с другой стороны, обладает очень сложным интерфейсом с большим количеством параметров, значения которых необходимо указать пользователю для того, чтобы поставить задачу.

Процедуру формирования дерева проблемных оболочек можно описать как движение от корневого элемента к листьям. Технология CAEBeans позволяет дочерним проблемным оболочкам конкретизировать класс задач, решаемых родительской оболочкой, путем выделения и инкапсуляции

групп проблемных параметров, значения которых определяются на данном уровне абстракции. Скрываемый проблемный параметр дочерней оболочки может быть задан в виде константы или в виде функции, зависящей от значений других проблемных параметров, входящих в интерфейс данной оболочки. Пользователю не требуется указание значений таких параметров вручную при постановке задачи. Таким образом обеспечивается упрощение пользовательского интерфейса.

Для решения конкретной инженерной задачи, пользователю предлагается оболочка соответствующая листу дерева проблемных оболочек. Такая оболочка имеет наиболее простой интерфейс и требует от пользователя минимальной квалификации. В случае если пользователю необходимо задействовать большее количество параметров, он осуществляет переход к родительской оболочке, тем самым переходя на более низкий уровень абстракции. Так пользователь может дойти до корневого проблемного CAEBean, который охватывает максимально широкий класс задач в данной предметной области.

При постановке задачи листовой проблемный CAEBean формирует *полный* дескриптор задачи: часть проблемных параметров задает пользователь, часть определяется внутри CAEBean [14]. При этом набор параметров, входящих в полный дескриптор задачи, одинаков для всех проблемных CAEBean, входящих в одно дерево проблемных оболочек [12]. Если пользователь ставит задачу средствами корневой проблемной оболочки, то все значения входных параметров указываются им вручную. Если же для постановки задачи использовалась оболочка более высокого уровня абстракции, значения проблемных параметров, зафиксированных на данном уровне, вычисляются по формулам, заданным разработчиком РаВИС. Сформированный полный дескриптор задачи передается в потоковый CAEBean, представляющий логический слой РаВИС.

2.2.2. Логический слой

Процесс решения задачи инженерного моделирования можно представить в виде совокупности взаимосвязанных *действий инженерного моделирования*, таких как определение геометрии моделируемой области, генерация расчетной сетки, определение физики решаемой задачи, проведение компьютерного моделирования, визуализация и анализ результатов компьютерного моделирования.

Потоковый CAEBean представляет собой *логический план* решения задач инженерного моделирования [13]. Для формирования логического плана используются элементы нотации *диаграммы деятельности* стандарта UML 2.0 [4, 5, 17, 35, 104, 108]. В соответствии с определением данным в п. 2.1.1 логический план решения задачи представляет собой ориентированный граф, в вершинах которого могут находиться узлы двух типов: действия и узлы управления, а ребра представляют собой отношения, управляющие потоками управления между двумя узлами (см. рис. 8).

Поток управления – это ребро, задающее протекание управления, но не данных. Информация, необходимая для исполнения любого узла логического плана содержится в полном дескрипторе задачи. В ходе решения задачи, любой узел может обратиться к дескриптору для получения значений входных параметров и сохранения результатов в соответствующих выходных параметрах.

Процесс выполнения деятельности можно представить себе как обход экземпляра графа логического плана, содержащего маркеры управления. *Маркер управления* указывает на наличие управления на некотором этапе процесса вычисления, представленным узлом или потоком. Маркеры управления не имеют внутренней структуры. Выполнение действия разрешается, если во всех его входных потоках присутствуют маркеры. Некоторые виды узлов управления могут начинать работу в момент, когда на

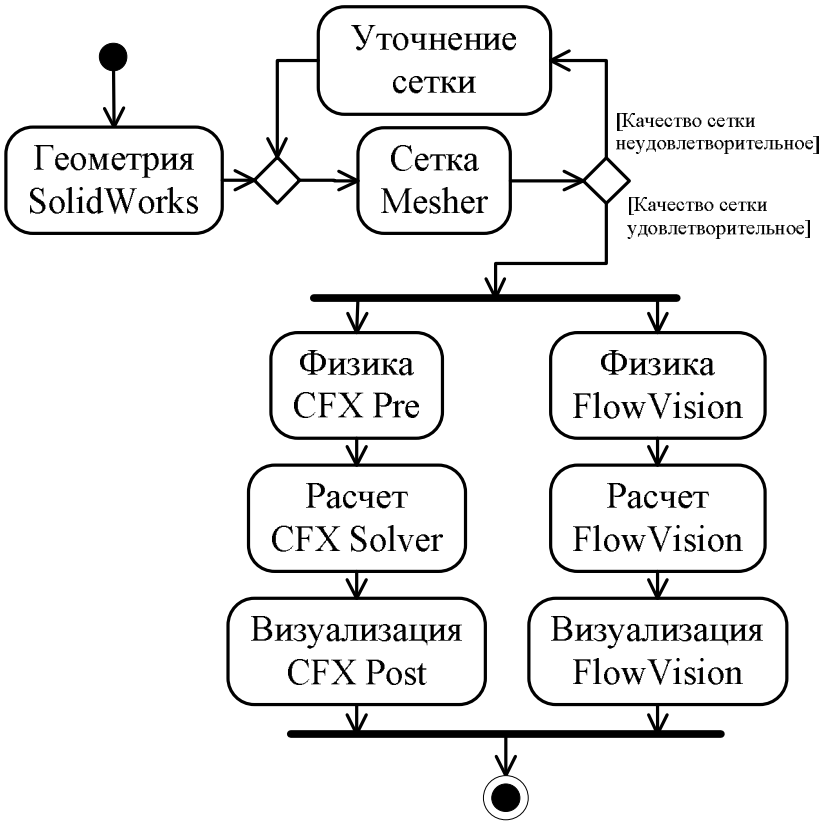


Рис. 8. Потоковый CAEBean.

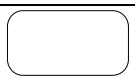
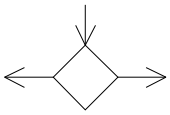
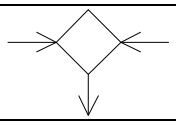
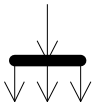
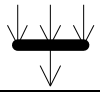
заданном подмножестве входных ребер появятся маркеры управления. Когда работа узла завершается, в каждом из его исходящих потоков управления появляется маркер управления.

Виды узлов логического плана решения задачи представлены в таблице 1.

Начальный узел является управляющим узлом, указывающим место начала исполнения логического плана. Начальный узел не может иметь ни одного входа, но может иметь один выход. Когда начинается выполнение логического плана, на начальный узел помещается маркер управления. Логический план может иметь несколько начальных узлов. В этом случае, в начале исполнения маркеры управления помещаются на каждый из них. Такая конструкция имитирует параллельность исполнения.

Выполнение конечного узла вызывает принудительное завершение всех потоков в данном логическом плане и завершение выполнения логического плана в целом.

Таблица 1. Виды узлов логического плана решения задач

Название	Обозначение	Описание
Начальный узел		Управляющий узел, с которого начинается выполнение логического плана.
Конечный узел		Управляющий узел, обозначающий завершение выполнения логического плана.
Действие		Узел, обеспечивающий выполнение определенного действия инженерного моделирования.
Разветвление		Управляющий узел, обеспечивающий выбор дальнейшего маршрута исполнения логического плана, в зависимости от значения булева выражения.
Объединение		Управляющий узел, в котором сходятся два или более альтернативных путей управления.
Разделение		Управляющий узел, обеспечивающий копирование входного маркера на каждый из выходов и инициирующий исполнение нескольких параллельных потоков управления.
Слияние		Управляющий узел, синхронизирующий несколько параллельных потоков управления.

Разветвление – это узел принятия решения, имеющий один входной поток несколько выходных потоков. На каждом из выходных потоков помещается ожидаемое значение условия. Маркер управления помещается в тот выходной поток, условие которого совпадает со значением выражения, вычисляемого на входе в узел разветвления. При этом необходимо, чтобы значения всех параметров, участвующих в логической формуле, были установлены в дескрипторе задачи до вызова узла разветвления, иначе результат вычислений будет не определен. Условия выходных потоков не должны перекрываться (иначе поток управления будет неоднозначным), но при этом должны учитываться все возможные варианты.

Объединение – это соединение двух или более маршрутов управления. Узел объединения имеет несколько входных потоков и один исходящий. Если маркер приходит на узел объединения по одному из входов, он немедленно копируется на выход. Допускается комбинация из разветвления и объединения, когда у узла есть несколько входящих и несколько помеченных исходящих переходов.

Разделение обеспечивает расщепление одного потока управления на несколько параллельных. Входной маркер управления копируется на каждый из выходов. После разделения деятельность, ассоциированная с каждым потоком, выполняется параллельно. При этом реализация параллельной деятельности должна быть такова, чтобы подмножество входных и выходных параметров узлов в каждом параллельном потоке не пересекалось с подмножествами выходных параметров узлов других потоков этой параллельной деятельности.

Слияние представляет синхронизацию нескольких параллельных потоков управления. Узел соединения может иметь несколько входящих потоков управления и один исходящий. Когда по каждому входному ребру приходит маркер управления, они потребляются, а на выходное ребро помещается новый маркер. В момент соединения, параллельные потоки синхронизируются, то есть каждый из них ждет, когда все остальные достигнут точки соединения, начиная с которой продолжит выполняться один поток управления. Между точками разделения и слияния должен поддерживаться баланс. Это означает что число потоков, исходящих из точки разделения, должно быть равно числу потоков, приходящих в соответствующую точку слияния.

Узел действия реализует определенное действие инженерного моделирования. Каждый узел действия обладает списком входных и выходных параметров, значения которых хранятся в полном дескрипторе задачи. При исполнении узла действия, потоковый CAEBean обеспечивает взаимодействие с дескриптором задачи: получение значений входных параметров для инициации работы действия и запись значений выходных параметров, вычисленных в результате исполнения действия. Действие реализуется соответствующим компонентным CAEBean, находящимся на физическом слое РаВИС.

2.2.3. Физический слой

Компонентные CAEBeans, представляющие физический слой РаВИС, отвечают за процесс постановки и решения отдельных действий инженерного моделирования средствами конкретных инженерных пакетов. При его разработке прикладной программист ориентируется на стандарты постановки задач, поддерживаемые конкретным CAE-пакетом: форматы файлов постановки задачи и результатов решения; методы автоматизации процесса решения; языки описания процесса моделирования; параметры входной строки и др.

Основная функция компонентного CAEBean – преобразование проблемно-ориентированного описания действия инженерного моделирования в *компонентно-ориентированную форму*. На основе значений входных параметров, компонентный CAEBean формирует набор файлов, содержащих описание действия в формате, поддерживаемом целевым CAE-пакетом. Также, на основе информации о формате и семантике параметров командной строки CAE-пакета, компонентный CAEBean формирует и передает команду запуска процесса решения.

По окончании процесса решения, компонентный CAEBean обеспечивает преобразование компонентно-ориентированных результатов решения задачи в проблемно-ориентированные. На основе информации о формате результатов, компонентный CAEBean обеспечивает извлечение значений выходных параметров действия из файлов решения задачи.

Компонентный CAEBean задает абстрактное действие, которое не может быть реализовано само по себе, так как не обеспечено реальными программными, аппаратными и лицензионными ресурсами. В процессе исполнения, каждому компонентному CAEBean должен быть найден и предоставлен физический ресурс, отвечающий всем требованиям, необходимым для реализации специфицированной в нем деятельности. Поиск и пре-

доставление физических ресурсов обеспечивается брокером ресурсов [12], описание которого приведено в п. 3.6. В рамках системы CAEBeans, физические ресурсы реализуются посредством системных CAEBeans, описываемых в следующем разделе.

2.2.4. Системный слой

Системным CAEBean называется оболочка системного слоя РаВИС, предоставляющая функциональные возможности физического ресурса в распределенной вычислительной среде и обеспечивающая сервисно ориентированный подход к постановке задач и получению результатов.

В рамках технологии CAEBeans, для любого инженерного пакета создается набор системных CAEBeans, каждый из которых представляет реализацию определенного действия инженерного моделирования (см. п. 2.2.2). Например, для пакета ANSYS CFX может быть создано три системных CAEBeans:

1. Pre_CAEBean (определение граничных условий и физических свойств сред моделируемой области);
2. Solver_CAEBean (расчет задачи вычислительной гидродинамики);
3. Post_CAEBean (анализ и визуализация результатов моделирования).

При этом для реализации таких оболочек могут быть использованы различные средства, в зависимости от программного интерфейса, предоставляемого конкретным CAE-пакетом. В качестве примера таких средств можно привести: лог-файлы и макросы, поддерживаемые CAE-пакетом ANSYS Mechanical; файлы сценариев на языке Python, поддерживаемые CAE-пакетом Abaqus; текстовый файл CCL (Command Language File - Файл языка команд) поддерживаемый решателем ANSYS CFX-Solver [13].

Системный CAEBean является проблемно-независимым элементом архитектуры РаВИС. С одной стороны, множество компонентных CAE-Beans, обеспечивающих реализацию различных действий инженерного мо-

делирования, могут обращаться к одному и тому же системному CAEBean, если предоставляемый им физический ресурс поддерживает решение задач такого типа. С другой стороны, в распределенной вычислительной среде может находиться сколь угодно много однотипных системных CAEBean. В этом случае выбирается тот, который свободен в данный момент.

Системный CAEBean предоставляет следующие возможности доступа к физическому ресурсу:

1) обеспечение изолированного рабочего пространства для каждого действия инженерного моделирования, реализуемого на вычислительной системе;
2) загрузка исходных данных, необходимых для реализации действия инженерного моделирования;
3) предоставление программного интерфейса для удаленного запуска действия инженерного моделирования;
4) предоставление служебной информации и промежуточных результатов реализации действия;
5) передача результатов реализации действия на требуемый узел грид-сети.

2.2.5. Взаимодействие слоев РаВИС

Рассмотрим процедуру постановки и решения задачи инженерного моделирования в соответствии с архитектурой CAEBeans (см. рис. 9).

1. При постановке задачи инженерного моделирования проблемный CAEBean формирует полный дескриптор задачи, занося туда значения проблемных параметров, указанные пользователем, и вычисляя значения зафиксированных параметров.

   Процесс решения задачи инженерного моделирования реализуется потоковым CAEBean посредством выполнения логического плана

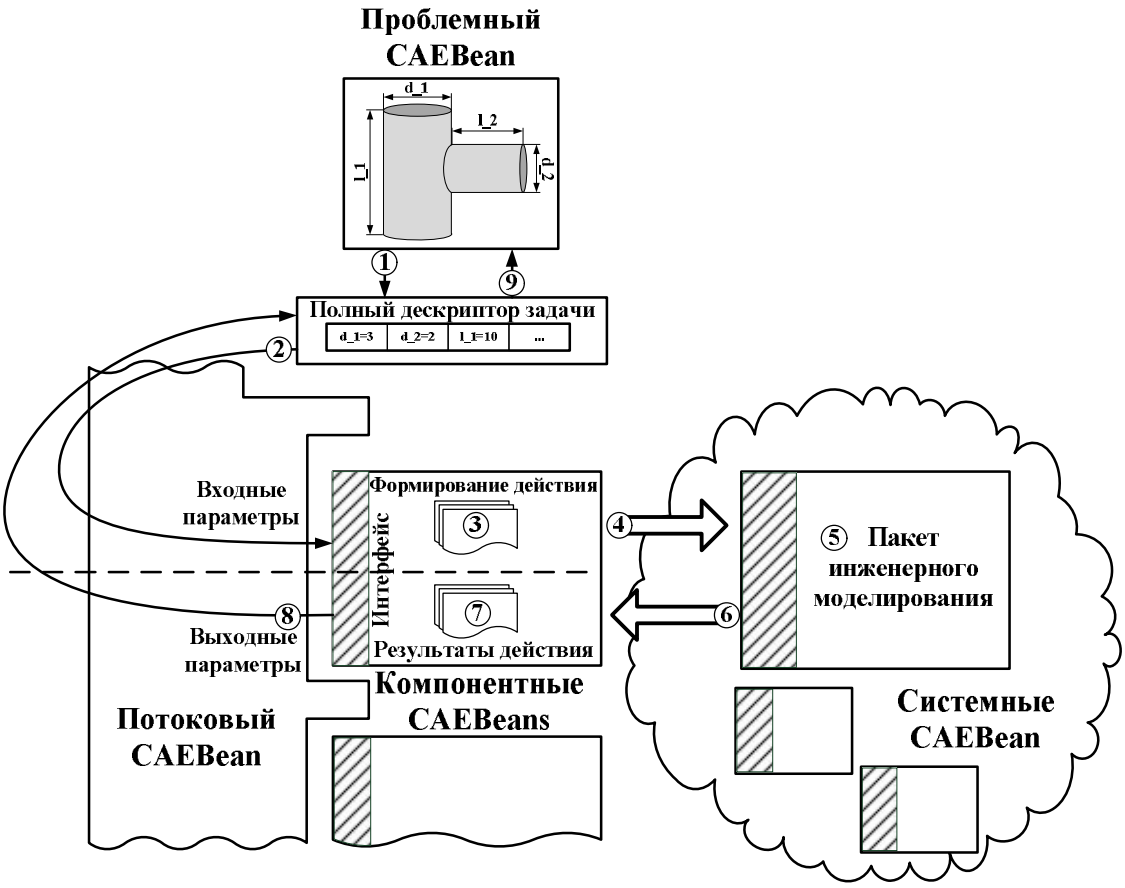


Рис. 9. Взаимодействие оболочек РаВИС в процессе решения задачи инженерного моделирования**.**

решения задачи. При реализации очередного действия инженерного моделирования, потоковый CAEBean получает из полного дескриптора задачи и передает компонентному CAEBean значения требуемых входных параметров. Значения всех входных параметров должны быть определены до момента реализации действия.

3. На основе значений входных параметров компонентный CAEBean формирует постановку действия для конкретного CAE-пакета, в соответствии с поддерживаемыми форматами файлов и команд.
4. Посредством брокера ресурсов производится поиск и предоставление системного CAEBean, обеспечивающего наилучшие возможности по реализации текущего действия. На предоставленный грид-ресурс производится копирование файлов постановки действия и запускается исполнение действия инженерного моделирования.

5. Инженерный пакет производит решение поставленной задачи и формирует набор файлов с результатами.
6. Компонентный CAEBean получает от удаленного сервиса необходимые ему файлы с результатами решения.
7. Компонентный CAEBean производит разбор результатов действия и извлекает значения требуемых выходных параметров из файлов, полученных в результате решения. Значения данных параметров передаются потоковому CAEBean.
8. Потоковый CAEBean помещает полученные значения выходных параметров в полный дескриптор задачи и завершает исполнение компонентного CAEBean. Производится вызов следующего узла логического плана решения задачи.
9. При завершении исполнения логического плана результаты моделирования в виде значений выходных параметров, сохраненных в полном дескрипторе задачи, передаются проблемному CAEBean.

2.3. Программные средства CAEBeans

Технологию разработки и исполнения распределенных испытательных стендов невозможно эффективно реализовать без использования специализированных программных средств, ориентированных на поддержку данной технологии. Программные средства CAEBeans можно разбить на два типа: средства исполнения и средства разработки РаВИС (см. рис. 10).

2.3.1. Разработка РаВИС

Разработка проблемно-ориентированной части распределенного виртуального испытательного стенда ведется в системе *Конструктор*. Посредством Конструктора прикладной программист формирует оболочки концептуального, логического и физического слоя РаВИС, отвечающие за решение конкретного класса задач инженерного моделирования.

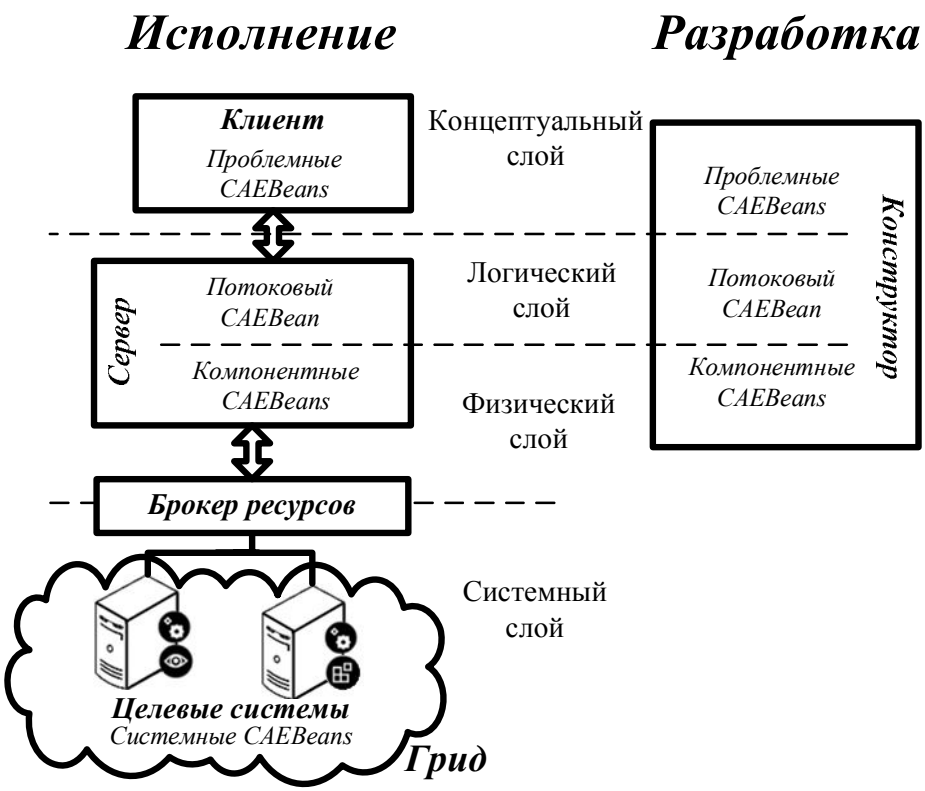


Рис. 10. Программные средства CAEBeans.

На концептуальном слое поддерживается создание корневого проблемного CAEBean, описание параметров задачи инженерного моделирования и полного дескриптора задачи. Также обеспечивается возможность формирования дерева проблемных оболочек посредством наследования и конкретизации наборов параметров родительских проблемных оболочек.

На логическом слое поддерживается визуальное проектирование логического плана решения задачи в соответствии с ограничениями и семантикой узлов, определенных в пункте 2.2.2. Поддерживается подключение физических CAEBean к узлам действий.

На физическом слое поддерживается описание интерфейса физического CAEBean, выбор входных и выходных параметров данного CAEBean, выбор необходимых файлов, обеспечивающих реализацию действия. Более детально реализация конструктора рассмотрена в пункте 3.2.

2.3.2. Исполнение РаВИС

В соответствии с технологией CAEBeans работа РаВИС осуществляется с помощью следующих четырех программных систем:

1) клиент;
2) сервер;
3) брокер ресурсов;
4) набор целевых систем.

Клиент предоставляет унифицированный интерфейс конкретного РаВИС, который был описан в п. 2.1.2. Клиент обеспечивает постановку и получение результатов решения задачи инженерного моделирования. Поддерживается возможность автоматической генерации пользовательского интерфейса РаВИС на основе описания параметров требуемого проблемного CAEBean. Подробное описание клиента будет дано в п. 3.3.

Сервер отвечает за хранение и исполнение РаВИС. На сервере производится анализ введенных пользователем значений параметров РаВИС и запускается процесс исполнения логического плана решения задачи. Сервер обеспечивает исполнение логического плана решения задачи. Более подробно архитектура сервера будет описана в п. 3.4.

Брокер ресурсов – это автоматизированная система регистрации, анализа и предоставления ресурсов распределенной вычислительной среды. Брокер представляет собой промежуточное программное обеспечение, обеспечивающее оптимальный выбор и виртуализацию доступа к наборам ресурсов в гетерогенной вычислительной среде. Архитектура брокера ресурсов и алгоритмы предоставления физических ресурсов описаны в п. 3.6. В совокупности, сервер и брокер ресурсов реализуют драйвер распределенного виртуального испытательного стенда, определенный в п. 2.1.2.

Целевая система – это совокупность грид-сервисов, которые имеют доступ к пространству программных, аппаратных и лицензионных ресурсов

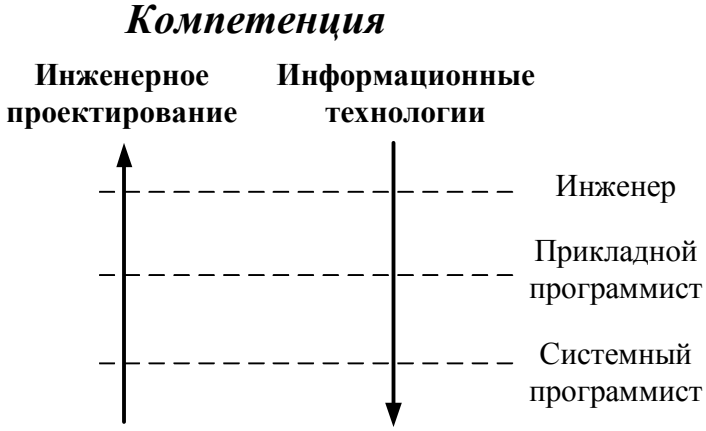


Рис. 11. Распределение компетенции пользователей РаВИС.

некоторого узла грид, и поддерживает аутентификацию и авторизацию пользователей. Сервисы, предоставляемые каждой отдельной целевой системой можно разделить на две категории:

1. системные CAEBeans;
2. служебные сервисы.

Системные CAEBeans инкапсулируют ресурсы, предоставляемые отдельными инженерными пакетами. Системные CAEBeans обеспечивают удаленную постановку действия инженерного моделирования и получение результатов его исполнения.

Служебные сервисы, входящие в целевую систему, поддерживают работу вычислительной системы в грид-среде совместно с программным обеспечением системы CAEBeans. Более подробно структура целевой системы будет описана в п. 3.5.

2.4. Организация работ в системе CAEBeans

Разработка РаВИС – это сложный процесс, который невозможен без участия специалистов в различных областях информационных технологий и инженерного проектирования и анализа. Можно выделить три основные роли в разработке и использовании РаВИС (см. рис. 11):

1) инженер;
2) прикладной программист;
3) системный программист.

2.4.1. Инженер

Инженер – это пользователь системы CAEBeans, решающий задачу инженерного моделирования на основе созданного для этой цели распределенного виртуального испытательного стенда. Доступ к возможностям РаВИС инженер получает посредством Клиента системы CAEBeans, предоставляющего пользовательский интерфейс соответствующего проблемного CAEBean. В связи с этим, инженер должен обладать минимальной компетенцией в области информационных технологий, которой достаточно чтобы запустить клиента системы, произвести постановку задачи инженерного моделирования и получить результаты решения.

При разработке РаВИС инженер представляет собой постановщика задачи инженерного моделирования, обладающего информацией о физической сущности решаемой задачи. Таким образом, инженер обладает максимальной компетенцией в сфере инженерного моделирования поставленной задачи среди всех пользователей системы CAEBeans.

2.4.2. Прикладной программист

Прикладной программист – это специалист в области разработки РаВИС посредством технологии CAEBeans. Это пользователь системы, который производит разработку РаВИС в Конструкторе CAEBeans на основе информации, полученной от инженера. Прикладной программист должен обладать широкими знаниями как в области инженерного моделирования, так и в области информационных технологий.

С точки зрения инженерного моделирования прикладной программист должен достаточно глубоко разбираться в поставленной задаче, чтобы выявить возможные пути ее параметризации и создать дерево проблемных CAEBean. Прикладной программист должен уметь описать оптимальный технологический цикл решения данной задачи при формировании потоко-

вого CAEBean и выбрать CAE-пакеты, обеспечивающие решение поставленной задачи.

С точки зрения информационных технологий, прикладной программист должен обладать глубокими знаниями о методах взаимодействия с CAE-пакетами, методах автоматизированной постановки задач инженерного моделирования, форматах входных и выходных файлов, методах их автоматического формирования и извлечения результатов решения. Также прикладной программист должен иметь хорошее представление о технологиях распределенных вычислений, чтобы адекватно оценить и по возможности минимизировать время исполнения логического плана решения задачи.

2.4.3. Системный программист

Системный программист – это специалист в области информационных технологий, отвечающий за формирование распределенной вычислительной среды для исполнения РаВИС. Он обеспечивает установку и настройку программных средств системы CAEBeans, а также CAE-пакетов, средствами которых производится решение задач инженерного моделирования.

При разработке РаВИС системный программист отвечает за создание и установку системных CAEBeans, обеспечивающих удаленную постановку и решение задач средствами инженерных пакетов. Программный интерфейс разработанного системного CAEBean передается прикладному программисту для возможности дальнейшей разработки компонентного CAEBean на базе ресурсов, предоставляемых данным системным CAEBean.

2.5. Параметрические модели производительности Грид

Принцип работы и функциональность грид приложений значительно отличаются от обычных последовательных и параллельных систем. Основное отличие – это возможность агрегирования и совместного использова-

ния больших наборов гетерогенных ресурсов, распределенных между географически-разделенными областями. Во многих случаях это приносит большие выгоды, например, когда приложение требует ресурсов, недоступных в рамках одного узла, оно может затребовать ресурсы у других узлов, подключенных к грид [46].

Но такое сложное поведение несет в себе и определенные проблемы. К высоко-гетерогенной, динамически-формируемой распределенной среде очень трудно напрямую применить такие традиционные метрики производительности, как скорость вычислений, пропускная способность канала и др. В связи с этим, для оценки качества предоставляемого сервиса требуется использование специализированных метрик.

Предположим, что в грид-среде доступно m ресурсов и существует система распределения заданий τ, обеспечивающая распределение поставленных задач $j \in \tau$ на доступные ресурсы. В рамках данной системы, каждое задание может быть разбито на действия $k \in j$. Количество заданий в системе $|\tau|$; количество действий в задаче $|j|$. При постановке задачи, указывается время d_j, до которого пользователь желает получить результаты решения.

Каждая задача j и все ее действия $k \in j$ поступают в грид в момент времени r_j. В связи с тем, что грид работает в “online”-режиме, значение r_j не известно заранее для большинства задач. Как только появляется определенная задача, производится планирование ее работы, после чего производится поиск и выделение ресурсов необходимых для запуска.

2.5.1. Метрики, зависящие от времени

Предположим, что в результате финального распределения S, каждое действие $k \in j$ будет исполнено за время $C_k(S)$. Таким образом, задача j может быть решена не раньше, чем за время

$$C_j(S) = \max_{k \in j} C_k(S).$$

Определим время реализации действия $k \in j$ как p_k. Таким образом, время решения p_j задачи j может быть вычислено следующим образом:

$$p_j = C_j(S) - \min_{k \in j}(C_k(S) - p_k).$$

Полученные величины позволяют оценить интегральные свойства грид-среды. Для анализа качества сервиса, предоставляемого грид-средой можно использовать показатель максимального опоздания задач:

$$L_{max} = \max_{j \in \tau}\big(C_j(S) - d_j\big).$$

При оптимизации работы распределенной среды необходимо стремиться к минимизации значения данного показателя. Также, можно использовать показатель TJ, определяемый как количество опоздавших задач: $j \in \tau \wedge C_j > d_j$. Такой показатель предоставляет информацию о количестве невыполненных пользовательских запросов.

Потребление ресурсов RC_k определенной подзадачей определим как произведение соответствующего времени решения на количество используемых ресурсов:

$$RC_k = p_k \cdot m_k.$$

Следовательно, мы можем определить потребление ресурсов определенным заданием (1), и всего перечня задач планировщика соответственно (2).

$$RC_k = \sum_{k \in j} RC_k\,; \tag{1}$$

$$RC(S) = \sum_{j \in r} RC_j. \tag{2}$$

Используя определение суммарного потребления ресурсов, можно определить величину использования U (3) доступных ресурсов.

$$U = \frac{RC(S)}{m \cdot \left(\max\limits_{j\in\tau} C_j(S) - \min\limits_{j\in\tau}\left(C_j(S) - p_j\right)\right)}. \quad (3)$$

Данная величина характеризует насколько оптимально используются ресурсы, доступные в распределенной сети.

В процессе работы грид, очень часто возникают ситуации, когда в процессе исполнения задачи происходит сбой. Тогда задание должно быть запущено несколько раз для того, чтобы успешно выполниться. Вследствие этого, мы можем определить полное потребление ресурсов $RC\{k,j\}^{true}$ и полную величину использования ресурсов U^{true}, как соответствующую величину плюс затраты на исполнение сбойных заданий. В связи с этим, можно определить метрику растрат:

$$WASTE = U^{true} - U,$$

которая определяет динамическую величину ошибок грид системы и должна быть минимизирована владельцем вычислительных ресурсов.

Так как пользователи и администраторы часто выдвигают различные (и даже конфликтующие) требования к грид системе, очень трудно подобрать метрику, которая удовлетворяла бы всех. С точки зрения пользователя, возможно выделить метрики среднего времени ответа (4) (Average Response Time – ART) и среднего времени ожидания (5) (Average Wait Time – AWT) [107]:

$$ART = \frac{1}{|\tau|}\sum_{j\in\tau}\left(C_j(S)\right), \quad (4)$$

$$AWT = \frac{1}{|\tau|}\sum_{j\in\tau}\left(C_j(S) - p_j\right). \quad (5)$$

Значение параметра ART характеризует, насколько быстро происходит решение задач пользователей. С другой стороны, значение параметра AWT интересно пользователям, которые производят постановку относительно небольших заданий.

Относительно хороший и простой метод измерения справедливости использования ресурсов это расчет девиации среднего времени ожидания:

$$AWTD = \frac{1}{|\tau|}\sqrt{\sum_{j\in\tau}(WT_j)^2 - \left(\sum_{j\in\tau}\frac{WT_j}{|\tau|}\right)^2},$$

где $WT_j = (\mathrm{C}_j(S) - p_j)$.

Для достижения оптимальных результатов работы грид, необходимо добиться минимизации параметра $AWTD$ каждым владельцем ресурсов.

Также, важна обработка результатов мониторинга работы грид системы. В этом случае, можно использовать метрику эффективности грид (Grid Efficiency – GE) [107]:

$$GE = \frac{\sum_{j\in\tau}((EndTime_j - StartTime_j)\times CPUs_j \times CPUSpeed_j)}{(EndTime_{lastJob} - SubmitTime_{firstJob})\times\sum_{m\in M}(CPUs_m \times CPUSpeed_m)}\times 100\%, \quad (6)$$

где $(EndTime_{lastJob} - SubmitTime_{firstJob})$ – это время работы системы;

$CPUs_j$ и $CPUSpeed_j$ – это количество процессоров использованных задачей j и их производительность;

$CPUs_m$ и $CPUSpeed_m$ – это количество процессоров в машине m и их производительность.

2.5.2. Метрики, зависящие от объема работы

В современных грид системах, возможность завершить исполнение данного объема работы может быть даже более важным, чем ускорение, полученное посредством такого исполнения (требуется отметить, что задачи, исполняемые в грид средах, могут быть значительно сложнее тех заданий, которые исполняются в традиционных параллельных системах, например потоки заданий обладают значительно более сложной логической структурой, чем пакеты задач). Грид требует переопределения понятия ошибки приложения: грид приложение, которое не смогло успешно выполниться в рамках отведенного ей бюджета, генерирует сообщение об ошиб-

ке, как только обнаружится невозможность успешного исполнения. Например, сбой может произойти вследствие того, что не могут быть найдены ресурсы для выполнения вычислений или в связи с наступлением крайнего срока работы приложения. Используя это понятие, отказоустойчивость можно определить как возможность на как можно больший срок переносить время появления ошибки, пока есть хоть какие-то шансы того, что приложение завершится успешно [74].

Определим метрику *завершенного объема работы* (Workload Completion) как отношение успешно завершенных задач к объему всех задач, поставленных планировщику грид-среды.

$$WC = \frac{\sum_{j \in \tau \wedge (j\ complited)} 1}{|\tau|}.$$

Данная метрика позволяет определить ограничения грид системы, и ее максимизация может быть основной целью как пользователей, так и владельцев ресурсов. С другой стороны, WC имеет некоторые ограничения с токи зрения владельцев ресурсов, так как задачи с меньшим количеством действий имеют большее влияние на данную величину.

В качестве дополнительной метрики предлагается ввести метрику *завершения действий* (Task Completion), которую можно определить как количество завершенных действий к общему количеству действий, исполненных в рамках системы распределения заданий:

$$TC = \frac{\sum_{j \in \tau \wedge k \in j \wedge (k\ complited)} 1}{\sum_{j \in \tau} |j|}.$$

Также, можно ввести понятие завершения разблокированных действий (Enabled Task Completion), где под понятием разблокированного действия мы будем понимать те действия, которые могут быть выполнены только после того, как все зависимости для данного действия будут выполнены:

$$ETC = \frac{\sum_{j \in \tau \wedge k \in j \wedge (k\ complited)} 1}{\sum_{j \in \tau \wedge k \in j \wedge (k\ enabled)} 1}.$$

Таким образом, владельцы ресурсов должны стремиться к максимизации ETC. Если метрики TC и ETC значительно разнятся, то необходимо принять специальные меры для обеспечения выполнения критичных действий (таких действий, от которых зависит исполнение большого количества других действий).

2.5.3. Адаптация моделей производительности

При использовании моделей производительности в грид для базовых вычислительных приложений применительно к проектируемой системе CAEBeans необходимо учитывать нюансы функционирования грид-сервисов данной системы.

Потоковый CAEBean**,** в соответствии с логическим планом решения поставленной задачи, производит последовательную реализацию действий посредством соответствующих компонентных оболочек CAEBeans. Следовательно, обращение к потоковой оболочке инициирует формирование задачи инженерного моделирования, которая разбивается на атомарные действия, соответствующие компонентным CAEBean, участвующим в решении задачи. Таким образом, любая задача CAEBeans разворачивается в последовательность действий, которые, скорее всего, зависят друг от друга по данным и решаются, в общем случае, на различных вычислителях.

Таким образом, возможно определить базовые параметры исполнения системы CAEBeans следующим образом. В рамках системы распределения заданий τ, каждое задание $j \in \tau$ представлено отдельной задачей, поставленной потоковому CAEBean. Каждая задача разбита на действия $k \in j$, в простейшем случае, соответственно количеству вызовов компонентных CAEBeans. Особенность процесса решения типовых задач инженерного моделирования состоит в том, что отдельные действия, на которые раскладывается задача, в большинстве случаев зависят друг от друга по данным, в

связи с чем высока степень последовательно-исполняемых участков логического плана.

Одним из наиболее важных показателей является показатель среднего времени ответа ART (4). Предположив, что последовательные участки логического плана являются доминирующими при решении задачи инженерного моделирования, показатель ART можно расписать следующим образом:

$$ART = \frac{1}{|\tau|}\sum_{j\in\tau}\left(EndTime_j - SubmitTime_j\right) = \frac{1}{|\tau|}\sum_{j\in\tau}\left(\sum_{k\in j} SolveTime_k\right), \quad (7)$$

где $SolveTime_k$ – это, суммарные временные затраты на реализацию действия k.

Применительно к модели CAEBeans мы можем расписать значение величины $SolveTime_k$ следующим образом:

$$SolveTime_k = PreparationTime_k + UploadTime_k + ProcessingTime_k + DownloadTime_k, \quad (8)$$

где $PreparationTime_k$ – время подготовки к решению и поиска необходимых вычислительных ресурсов; $UploadTime_k$ - время загрузки начальных данных для реализации действия на найденный вычислительный узел; $ProcessingTime_k$ – время непосредственной реализации действия; $DownloadTime_k$ – время загрузки результатов действия.

2.5.4. Оценка производительности технологии CAEBeans

Для оценки приемлемых величин параметра ART был произведен анализ текущих задач, решаемых суперкомпьютерным центром ЮУрГУ [18]. В результате, исходя из статистики использования вычислительных ресурсов суперкомпьютерного центра за последние 6 месяцев, были получены следующие значения параметров их использования:

– средний объем затрат на решение расчетной задачи: $\overline{v}$=1500 машино-минут;
– среднее количество процессоров, выделяемых на одну задачу: $\overline{n} = 10$ процессоров (из 50 доступных);
– среднее время получения результата вычисления: $\overline{t}_{\text{реш}} = \frac{\overline{v}}{\overline{n}} = 150$ минут;
– среднее время подготовки к запуску задания: $\overline{t}_{\text{зап}} = 2$ минуты
– среднее количество одновременно выполняемых заданий: $\overline{|\tau|} = 4$.

При оценке параметра *ART*, были использованы следующие допущения:

– среднее количество действий соответствует количеству шагов типового технологического цикла решения задач численного моделирования: формирование геометрии задачи, построение расчетной сетки, формирование физики задачи, решение задачи, анализ результатов решения. Таким образом, $\overline{|j|} = 5$;
– объем данных, передаваемых для решения задачи составляет порядка 100Mb. Таким образом, пересылка данных для постановки задачи и для получения результатов решения на скорости 1,5MB/c составляет порядка $\overline{t}_{\text{передачи}}$= 2 минуты.

Также, при оценке общих затрат необходимо учитывать, что в настоящее время посредством вычислительных ресурсов суперкомпьютерного центра ЮУрГУ в основном решаются только наиболее ресурсоемкие этапы задач компьютерного моделирования. Временные затраты на решение других этапов составляют порядка $\frac{1}{100}$ от временных затрат на наиболее ресурсоемкий этап. Таким образом, учитывая один ресурсоемкий этап и четыре относительно нересурсоемких этапа вычислений, мы получаем следующую оценку параметра ART:

$$\overline{ART} = 4 \cdot \left(\overline{t}_{\text{зап}} + 2 \cdot \overline{t}_{\text{передачи}} + \frac{\overline{t}_{\text{реш}}}{100}\right) + 1 \cdot \left(\overline{t}_{\text{зап}} + 2 \cdot \overline{t}_{\text{передачи}} + \overline{t}_{\text{реш}}\right) = \quad (9)$$

$$= 4 \cdot (2 + 2 \cdot 2 + 15) + 1 \cdot (2 + 2 \cdot 2 + 150) = 84 + 156 = 240 \text{ (мин)}.$$

Таким образом, полученная оценка параметра *ART* для системы CAEBeans позволяет судить о потенциально хорошей эффективности использования данной системы для автоматизации решения типовых задач инженерного моделирования. Также полученная оценка позволяет определить потенциальные возможности повышения эффективности использования разрабатываемой системы посредством кэширования результатов предыдущих этапов вычислений для их возможного повторного использования на последующих этапах моделирования определенных классов задач и повышения степени параллельности реализуемых действий.

2.6. Выводы по главе 2

В главе 2 была рассмотрена архитектура CAEBeans, определяющая структуру и принципы работы РаВИС. В основе архитектуры CAEBeans лежит понятие задачи инженерного моделирования, базирующееся на понятиях параметра задачи инженерного моделирования, полного дескриптора задачи и логического плана решения задачи. Архитектура CAEBeans определяет четыре слоя РаВИС: концептуальный, логический, физический и системный. Данным слоям соответствуют четыре типа оболочек CAEBeans. Разработка и исполнение РаВИС поддерживается программным комплексом «система CAEBeans». Выделено три основных роли пользователей, работающих с РаВИС: инженер, прикладной программист, системный программист. Произведена оценка показателя среднего времени ответа для решения задачи инженерного моделирования средствами суперкомпьютерных ресурсов Южно-Уральского государственного университета.

ГЛАВА 3. СИСТЕМА CAEBEANS

Для поддержки технологии CAEBeans, описанной в главе 2, был разработан программный комплекс *система CAEBeans*, обеспечивающий разработку и исполнение РаВИС.

3.1. Структура системы CAEBeans

3.1.1. Состав системы CAEBeans

В систему CAEBeans [11] входят следующие компоненты (см. рис. 12):

1. CAEBeans Constructor – интегрированная среда разработки распределенных виртуальных испытательных стендов для грид;
2. CAEBeans Portal – веб-приложение, обеспечивающее выбор, запуск и получение результатов моделирования РаВИС;
3. CAEBeans Server – хранилище и среда исполнения РаВИС;
4. CAE-ресурсы – грид-сервисы, обеспечивающие удаленную постановку и решение задач инженерного моделирования;
5. CAEBeans Broker – автоматизированная система регистрации, анализа и предоставления CAE-ресурсов.

3.1.2. CAE-проект

В основе системы CAEBeans лежит понятие CAE-проекта. `CAEProject` – это класс, объединяющий в себе взаимосвязанные сущности проблемных, потоковых и компонентных оболочек CAEBeans, ориентированные на решение конкретного класса задач инженерного моделирования (см. рис. 13).

Класс `CAEProject` обладает следующими атрибутами:

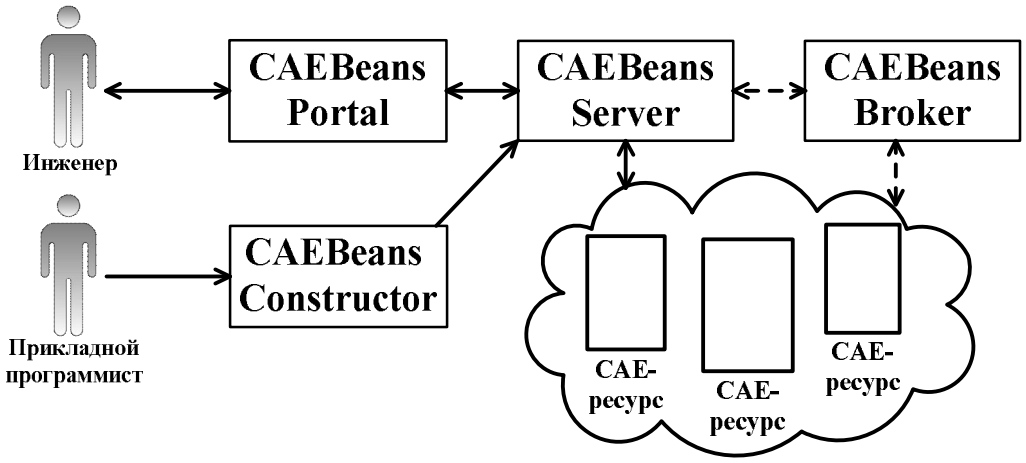


Рис. 12. Общая схема взаимодействия компонентов системы CAEBeans.

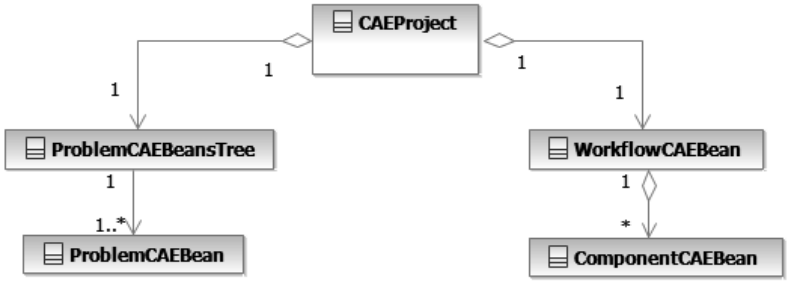


Рис. 13. Сущности, составляющие CAE-проект.

– `GUID projectId:` уникальный идентификатор [83] CAE-проекта, обеспечивающий однозначную идентификацию [89] всех проектов рамках системы CAEBeans;
– `String name:` имя CAE-проекта, заданное прикладным программистом в процессе разработки.

В класс `CAEProject` входят экземпляры следующих классов:

– `ProblemCAEBeansTree:` контейнер, содержащий дерево проблемных оболочек CAEBeans входящих в текущий CAE-проект;
– `WorkflowCAEBean:` потоковый CAEBean.

Хотя класс `CAEProject` не реализует методов, обеспечивающих решение задачи инженерного моделирования, он обеспечивает взаимодействие между входящими в него компонентами и объединяет их в рамках единого CAE-проекта.

3.1.3. CAE-параметр

`CAEParameter` – это класс, содержащий информацию об отдельном параметре задачи инженерного моделирования. Можно выделить следующие основные атрибуты класса `CAEParameter`:

- `String name`: имя параметра, является его уникальным идентификатором и не может дублироваться в рамках одного CAE-проекта;
- `CAEParameterType parameterType`: тип CAE-параметра (целочисленный, число с плавающей запятой, множество, строка и др.);
- `String default`: значение параметра, заданное по умолчанию;
- `String units`: единицы измерения параметра;
- `String comment`: комментарии, описывающие особенности и область применения CAE-параметра в терминах проблемной области CAE-проекта;
- `String value`: значение параметра (поддерживается использование стандартных арифметических операций, использующих в качестве параметров имена других параметров задачи);
- `Bool visible`: видимость параметра пользователю.

Унифицированное описание CAE-параметра позволяет автоматически формировать пользовательский интерфейс для постановки любой задачи инженерного моделирования на основе множества проблемных CAE-параметров, значения которых должен указать инженер.

3.1.4. Проблемный CAEBean

Класс `ProblemCAEBean`, реализующий сущность проблемного CAEBean, объединяет CAE-параметры задачи инженерного моделирования и формирует полный дескриптор задачи, на основе которого осуществляется постановка и решение задачи.

Можно выделить следующие основные атрибуты класса `ProblemCAEBean`:

- `GUID problemCAEBeanId`: идентификатор проблемного CAEBean;
- `CAEProject* parentProject`: указатель на базовый CAE-проект, в который входит текущий проблемный CAEBean;
- `String author`: имя разработчика проблемного CAEBean;
- `String version`: версия проблемного CAEBean;
- `CAEParamterCategory[] categories`: контейнер, содержащий в себе описание категорий, на которые разбиты параметры текущего проблемного CAEBean;
- `CAEParameter[] parameters`: множество параметров, объединенных данным проблемным CAEBean, включающее в себя параметры как с предопределенными значениями, так и со значениями определяемыми пользователем при постановке задачи.

Можно выделить следующий основной метод `ProblemCAEBean`:

- `void execute(ParameterValues[] userDefinedValues)`: вызов данного метода инициирует процесс решения задачи инженерного моделирования:
  1) производится вычисление значений всех входных параметров задачи инженерного моделирования и формирование полного дескриптора задачи;
  2) полный дескриптор задачи передается потоковому CAEBean как параметр при вызове его метода `execute()`.

Входные параметры: множество значений CAE-параметров, указанных инженером при постановке задачи инженерного моделирования.

3.1.5. Потоковый CAEBean

Класс `WorkflowCAEBean`, реализующий сущность потокового CAEBean, содержит информацию о логическом плане решения задачи инженерного моделирования и обеспечивает его исполнение. Типы и семантика узлов логического плана решения задачи основаны на нотации диаграммы деятельности стандарта UML 2.0.

Класс `WorkflowCAEBean` содержит следующие атрибуты:

- `ProblemDescriptor mainProblemDescriptor`: полный дескриптор задачи, содержащий все CAE-параметры, необходимые для решения задачи;
- `Workflow workflow`: граф деятельности.

Интерфейс `WorkflowCAEBean` определяет методы, обеспечивающую исполнение логического плана решения задачи:

- `void execute(ProblemDescriptor problemDescriptor)`: запуск исполнения логического плана решения задачи.

  Входные параметры: полный дескриптор поставленной задачи.
- `void monitorLoop()`: процедура исполнения *монитора логического плана*.

Перед тем как описать алгоритм монитора логического плана, необходимо рассмотреть основные методы, предоставляемые узлами логического плана.

Граф деятельности, реализуемый классом `Workflow`, представляет собой связный список, элементы которого представляют собой узлы логического плана а связи между ними устанавливают очередность исполнения данных узлов в процессе решения задачи инженерного моделирования. Узлы логического плана потокового CAEBean, являются наследниками абстрактного класса `AbstractWorkflowNode` (см. рис. 14), что обеспечивает

Рис. 14. Диаграмма классов, реализующих виды узлов логического плана.

универсальный механизм исполнения любого узла логического плана, независимо от его типа.

Рассмотрим атрибуты класса `AbstractWorkflowNode`:

- `AbstractWorkflowNode[] sourceNodes`: набор узлов логического плана, из которых выходят потоки управления, входящие в текущий узел;
- `AbstractWorkflowNode[] targetNodes`: набор узлов логического плана, в которые ведут исходящие потоки управления из текущего узла;
- `ProblemDescriptor* descriptor`: указатель на структуру, в которой хранится полный дескриптор решаемой задачи инженерного моделирования;

`AbstractWorkflowNode` определяет следующие виртуальные методы:

- `Boolean ready()`: проверка, готов ли узел к исполнению;
- `void sendToken(sourceNode* AbstractWorkflowNode)`: получить маркер управления от узла `sourceNode`;

```
// пока происходит исполнение логического плана
while (running) {
     // цикл по всем узлам логического плана
     foreach (AbstractWorkflowNode node in workflow) {
          if node.ready(){    //Если узел готов к исполнению
               node.reset();  //Сброс информации о поступивших
                              //маркерах управления
               new Thread(node).start();  //Исполнение узла
          } //if                          //в отдельном потоке
     }//foreach
}//while
```

Рис. 15. Алгоритм работы монитора потокового CAEBean**.**

– `void run()`: исполнить текущий узел логического плана;

– `void reset()`: сбросить информацию о всех полученных маркерах управления.

На рис. 15 приведен алгоритм работы монитора логического плана, реализованный в методе `WorkflowCAEBean.monitorLoop()`.

Процесс исполнения логического плана поддерживается посредством монитора логического плана. Он постоянно производит проверку состояния всех узлов логического плана. Готовность узла к исполнению зависит от наличия маркеров управления в потоках управления, входящих в узел. В зависимости от типа узлов, может требоваться наличие одного или нескольких маркеров управления на входных потоках управления. В связи с этим, проверка готовности узла к исполнению инкапсулируется методом `ready()`.

Как только появляется узел, готовый к исполнению, монитор формирует отдельную нить, в которой запускает исполнение метода `run()` данного узла. При завершении исполнения данного метода узел логического плана обязан сформировать маркеры управления на потоках управления, выходящих из него.

В зависимости от подхода к обработке поступающих маркеров управления и формирования выходных маркеров, выделены три класса уз-

лов логического плана: DecisionMergeNode, InitialNode и абстрактный класс ParallelNode.

Класс InitialNode реализует начальный узел логического плана решения задачи. Исполнение данного узла инициируется один раз за все время исполнения логического плана решения задачи.

Единственный атрибут данного класса Boolean executed является флагом, отмечающим то, что данный узел уже был исполнен.

Метод sendToken() является заглушкой виртуального метода, описанного в родительском классе AbstractWorkflowNode, и не несет никакой функциональной нагрузки, в связи с отсутствием входных потоков в начальный узел.

Рассмотрим остальные методы, реализуемые данным классом:

- void run(): вызывает метод putTokens();
- Boolean ready(): возвращает значение флага executed;
- void reset(): устанавливает значение флага executed равным «истина».
- void putTokens(): устанавливает маркеры деятельности во все потоки деятельности исходящие из текущего узла.

Класс DecisionMergeNode объединяет в себе семантику узлов разветвления и объединения (особенности узлов разветвления и объединения описаны в п. 2.2.2). Это позволяет формировать единый узел разветвления-объединения, имеющий несколько входных и выходных потоков управления. В состав класса входят следующие атрибуты:

- String[] guardConditions: охранные условия, количество которых соответствует количеству потоков управления targetNodes, исходящих из текущего узла;

- `String criteria`: выражение, значение которого сравнивается с охранными условиями для выбора дальнейшего пути исполнения логического плана;
- `CAEParameter[] inCAEParameters`: множество CAE-параметров участвующих в вычислении `criteria`;
- `Boolean token`: флаг, отмечающий, что данному узлу пришел маркер управления.

Класс `DecisionMergeNode` реализует абстрактные методы `sendToken()`, `run()`, `ready()` и `reset()` определенные в родительском классе `AbstractWorkflowNode`, а также собственный метод `putToken()`:

- `void sendToken()`: определить значение флага `token` равным «истина»;
- `void putToken(AbstractWorkflowNode targetNode)`: передать маркер управления последующему узлу логического плана `targetNode`;
- `void reset()`: установить значение флага `token` равным «ложь»;
- `Boolean ready()`: возвращает значение флага `token`.
- `void run()`: исполнить текущий узел логического плана;

Метод `DecisionMergeNode.run()` обеспечивает интерпретацию выражения, содержащегося в атрибуте `criteria`. Результат вычисления выражения сравнивается с граничными условиями `guardConditions`. После сравнения маркер управления передается по тому потоку управления, охранные условия которого удовлетворяют значению вычисленного выражения. Если значение атрибута `criteria` не определено, считается, что объект класса `DecisionMergeNode` реализует исключительно узел слияния, и значения аргументов `guardConditions`, `criteria` и

inCAEParameters не учитываются при его исполнении. В этом случае маркер управления передается единственному последующему узлу логического плана.

Абстрактный класс ParallelNode обеспечивает возможность синхронизации потоков управления на входе и формирования множества потоков управления на выходе узла логического плана.

Атрибут NodesDictionary tokens формирует словарь, ведущий учет входящих маркеров управления. В tokens каждому входящему потоку управления поставлен в соответствие элемент типа Boolean, значение которого показывает, был ли получен по данному потоку маркер управления. При инициации узла, словарь tokens заполняется значениями «ложь».

В данном классе реализуются абстрактные методы sendToken(), ready(), reset(), описанные в классе AbstractWorkflowNode, а также метод putTokens():

- void sendToken(): получение входного маркера управления от предшествующего узла логического плана. При вызове данного метода, текущий узел получает указатель на узел-источник sourceNode* AbstractWorkflowNode из параметра метода. Элементу словаря tokens, соответствующему данному узлу, присваивается значение «истина».
- Boolean ready(): возвращает «истина», если значение всех элементов в словаре tokens равно «истина». Иначе возвращает значение «ложь».
- void reset(): установить значения всех элементов словаря tokens равными «ложь».

Класс `FinalNode` описывает конечный узел логического плана. Исполнение метода `run()` этого узла инициирует процесс завершения решения задачи.

Класс `ForkJoinNode` объединяет в себе семантику узлов разделения и слияния. Такой подход позволяет формировать единый узел слияния-разделения, поддерживаемый стандартом UML 2.0. Метод `run()` данного класса вызывает метод `putTokens()`.

Класс `ActionNode` является реализацией узла действия. Основная задача данного класса – подготовить входные параметры для компонентного CAEBean, содержащегося в атрибуте `action`, и экспортировать результаты его работы обратно в полный дескриптор задачи. Информация о требуемых параметрах доступна непосредственно из атрибута `ComponentCAEBean action`. Метод `ActionNode.run()` вызывает метод `action.execute(ParameterValues[] inParameterValues)`.

`ActionNode` передает значения входных параметров компонентному CAEBean и инициирует исполнение действия. После исполнения действия, `ActionNode` записывает значения выходных параметров в полный дескриптор задачи, ссылка на который хранится в атрибуте `descriptor`, и вызывает метод `putTokens()`.

3.1.6. Компонентный CAEBean

Класс `ComponentCAEBean` реализует функциональные возможности, предоставляемые компонентным CAEBean. Можно выделить следующие основные атрибуты класса `ComponentCAEBean`:

– `String name`: уникальное имя компонентного CAEBean, однозначно определяющее его в рамках текущего CAE-проекта.

– `SystemCAEBeanInterface* systemCAEBeanInterface`: указатель на описание интерфейса системного CAEBean, средствами которого реализуется действие компонентного CAEBean;
– `CAEParameter[] inCAEParameters`: набор CAE-параметров, значения которых требуются для постановки и реализации действия компонентного CAEBean.
– `CAEParameter[] outCAEParameters`: набор CAE-параметров, значения которых устанавливаются в результате реализации текущего действия.

Класс `ComponentCAEBean` поддерживает следующие методы:

– `void generateActionDefinition(ParameterValues[] inParameterValues)`: формирование системно-ориентированной постановки действия в формате, соответствующем интерфейсу базового системного CAEBean `systemCAEBeanInterface`;
входные параметры: массив значений входных параметров;
– `URI findCAEResource()`: данный метод инкапсулирует процесс взаимодействия с Брокером Ресурсов, обеспечивая формирование запроса, поиск и предоставление CAE-ресурса, соответствующего требованиям данного компонентного CAEBean;
результат: адрес предоставляемого CAE-ресурса;
– `GUID setAction(URI CAEResource)`: обеспечивает постановку действия на выбранный CAE-ресурс, включая пересылку файлов постановки задачи и запуск процесса решения в соответствии с интерфейсом базового системного CAEBean;
входные параметры: адрес CAE-ресурса;
результат: идентификатор, однозначно определяющий контекст исполняемого действия на удаленной вычислительной системе;

- `ActionStatus checkActionStatus(GUID action, URI CAEResource)`: метод обеспечивает проверку состояния текущего действия на удаленном CAE-ресурсе;
  входные параметры: `action` – идентификатор действия, `CAEResource` – адрес CAE-ресурса;
  результат: статус действия;
- `void getResults(GUID action, URI CAEResource)`: обеспечивает получение результатов (промежуточных или конечных, в зависимости от статуса действия) с удаленного CAE-ресурса;
  входные параметры: `action` – идентификатор действия, `CAEResource` – адрес CAE-ресурса;
- `ParameterValues[] translateResults()`: метод анализирует результаты решения задачи и извлекает значения выходных параметров;
  результат: массив значений выходных параметров компонентного CAEBean;
- `ParameterValues[] execute(ParameterValues[] inParameterValues)`: метод обеспечивает автоматизированную генерацию, постановку и получение результатов действия;
  входные параметры: массив значений входных параметров;
  результат: массив значений выходных параметров.

3.1.7. Интерфейс системного CAEBean

Системный CAEBean - это грид-сервис, обеспечивающий предоставление функциональных возможности конкретной инженерной системы в грид-среде. Класс `SystemCAEBeanInterface` представляет интерфейс системного CAEBean. Он включает в себя описание формата входных и

выходных файлов, а также параметры интерфейса командной строки для запуска процесса исполнения.

- `String name`: имя, однозначно идентифицирующее системный CAEBean;
- `CLIParameter[] parameters`: описание параметров командной строки, необходимых для запуска процесса решения задачи инженерного моделирования средствами данного системного CAEBean;
- `FileFormat[] inFiles`: описание формата входных файлов, необходимых для постановки задачи;
- `FileFormat[] outFiles`: описание формата выходных файлов, формирующихся в результате решения задачи.

Методы класса `SystemCAEBeanInterface` обеспечивают реализацию системно-зависимых операций при постановке и реализации действия инженерного моделирования, инкапсулируя процесс взаимодействия с сервисом системного CAEBean:

- `GUID createAction(URI CAEResource)`: формирует контекст исполнения для определенного действия на удаленном системном CAEBean;

  входные параметры: адрес CAE-ресурса;

  результат: массив значений выходных параметров.
- `void putFiles(GUID action, URI CAEResource, File[] inFiles)`: копирует файлы постановки задачи, необходимые для решения задачи средствами данного системного CAEBean;

  входные параметры: `action` – идентификатор действия, `CAEResource` – адрес CAE-ресурса, `inFiles` – массив файлов, содержащих постановку задачи, в формате, соответствующем базовой CAE-системе;

- `void executeCLICommand(GUID action, URI targetSystem, CLIParameter[] parameters)`: передает на удаленный системный CAEBean значения параметров командной строки и инициирует процесс реализации действия инженерного моделирования средствами инженерного пакета;

  входные параметры: `action` – идентификатор действия, `CAEResource` – адрес CAE-ресурса, `inFiles` – массив файлов, содержащих постановку задачи, в формате, соответствующем базовой CAE-системе;
- `void getFiles(GUID action, URI CAEResource)`: осуществляет получение файлов с результатами решения с системного CAEBean;

  входные параметры: `action` – идентификатор действия, `CAEResource` – адрес CAE-ресурса;
- `ActionStatus checkActionStatus(GUID action, URI CAEResource)`: метод обеспечивает проверку состояния текущего действия;

  входные параметры: `action` – идентификатор действия, `CAEResource` – адрес CAE-ресурса;
- `void destroyAction(GUID action, URI CAEResource)`: остановка исполнения действия;

  входные параметры: `action` – идентификатор действия, `CAEResource` – адрес CAE-ресурса.

3.1.8. CAE-задание

Класс `CAEJob` реализует сущность *CAE-задания.* CAE-задание содержит информацию об определенной инженерной задаче, поставленной пользователем. Когда пользователь заполняет значения всех необходимых

параметров проблемного CAEBean и отправляет CAE-проект на решение, система CAEBeans формирует CAE-задание, которое поддерживает весь дальнейший процесс решения задачи инженерного моделирования. Можно выделить следующие основные атрибуты класса `CAEJob`:

- `GUID jobId`: уникальный идентификатор CAE-задания;
- `CAEProject project`: CAE-проект, содержащий полную информацию о поставленной задаче;
- `PorblemCAEBean* problemBean`: указатель на проблемный CAEBean, содержащий информацию о значениях параметров поставленного CAE-задания;
- `DateTime submitTime`: время постановки CAE-задания;
- `DateTime destroyTime`: время принудительной остановки исполнения CAE-задания;
- `String status`: текущий статус CAE-задания.

`CAEJob` поддерживает следующие методы:

- `GUID createJob(CAEProject* project, PorblemCAEBean* problemBean, ParameterValues[] userDefinedValues)`: метод обеспечивает формирование задания на основе определенного проблемного CAEBean существующего CAE-проекта с учетом значений входных параметров `userDefinedValues` установленных пользователем; метод возвращает уникальный идентификатор CAE-задания;

  входные параметры: `project` – указатель на базовый CAE-проект, `problemBean` – указатель на базовый проблемный CAEBean, `userDefinedValues` – значения входных параметров, указанные инженером при постановке задачи;

  результат: уникальный идентификатор сформированного CAE-задания;

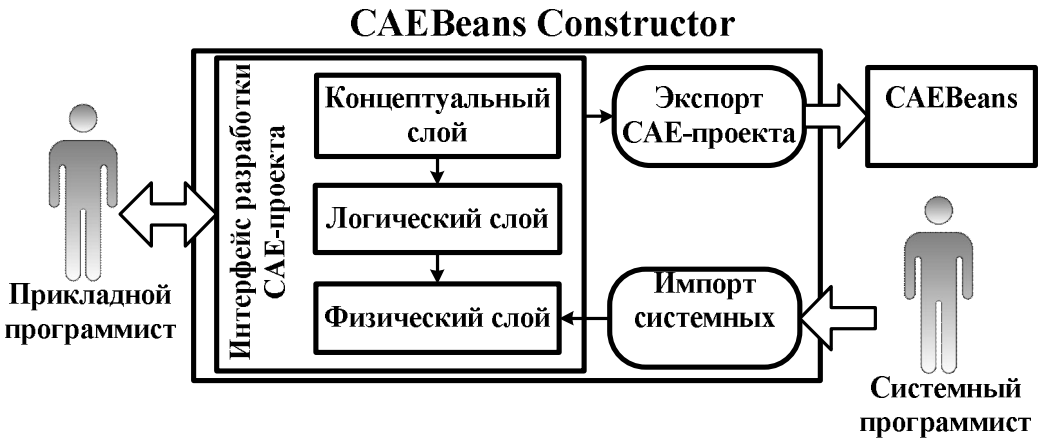


Рис. 16. Общая схема CAEBeans Constructor.

- `String updateStatus()`: обновить информацию о статусе CAE-задания;
- `void terminateJob()`: остановить исполнение задания;
- `void fetchOutcome()`: метод обеспечивает предоставление результатов исполнения задания.

3.2. Конструктор

CAEBeans Constructor – это интегрированная среда разработки РаВИС на основе CAE-проектов. CAEBeans Constructor предоставляет прикладному программисту пользовательский интерфейс для разработки оболочек CAEBeans концептуального, логического и физического слоев. В соответствии с этим, пользовательский интерфейс, обеспечивающий разработку CAE-проектов в среде CAEBeans Constructor, разделен на 3 секции, обеспечивающих разработку проблемных, логических и физических оболочек CAEBean соответственно (см. рис. 16).

Разработка CAE-проекта начинается с формирования концептуального слоя и корневого проблемного CAEBean в дереве проблемных оболочек. Пользовательский интерфейс концептуального уровня позволяет прикладному программисту сформировать список категорий и указать свойства CAE-параметров входящих в каждую из категорий (см. рис. 17).

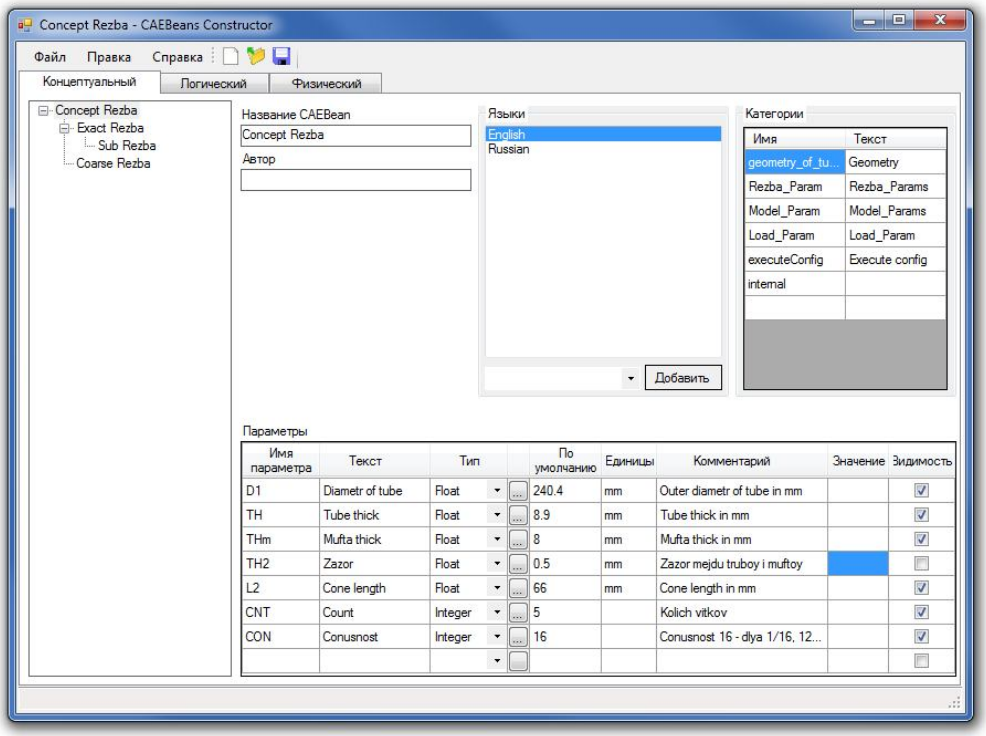


Рис. 17. Пользовательский интерфейс для редактирования концептуального слоя CAE-проекта.

Также, CAEBeans Constructor предусматривает возможность формирования иерархии проблемных CAEBean, посредством наследования дочерних проблемных CAEBean от родительского. CAE-параметры в наследуемом проблемном CAEBean можно зафиксировать определенными значениями (константой, либо выражением, значение которого зависит от других CAE-параметров).

Пользовательский интерфейс логического слоя CAEBeans Constructor обеспечивает визуальное редактирование графа логического плана потокового CAEBean (см. рис. 18). Разработчик РаВИС может создавать, редактировать и удалять узлы логического плана. В зависимости от типа добавляемого узла, может потребоваться указать его дополнительные свойства: при создании узла ветвления требуется указать условие ветвления, при создании узла ветвления управления требуется указать количество параллельных ветвей. При создании нового узла деятельности, пользователю предоставляется возможность выбора одного из существующих или создание нового компонентного CAEBean.

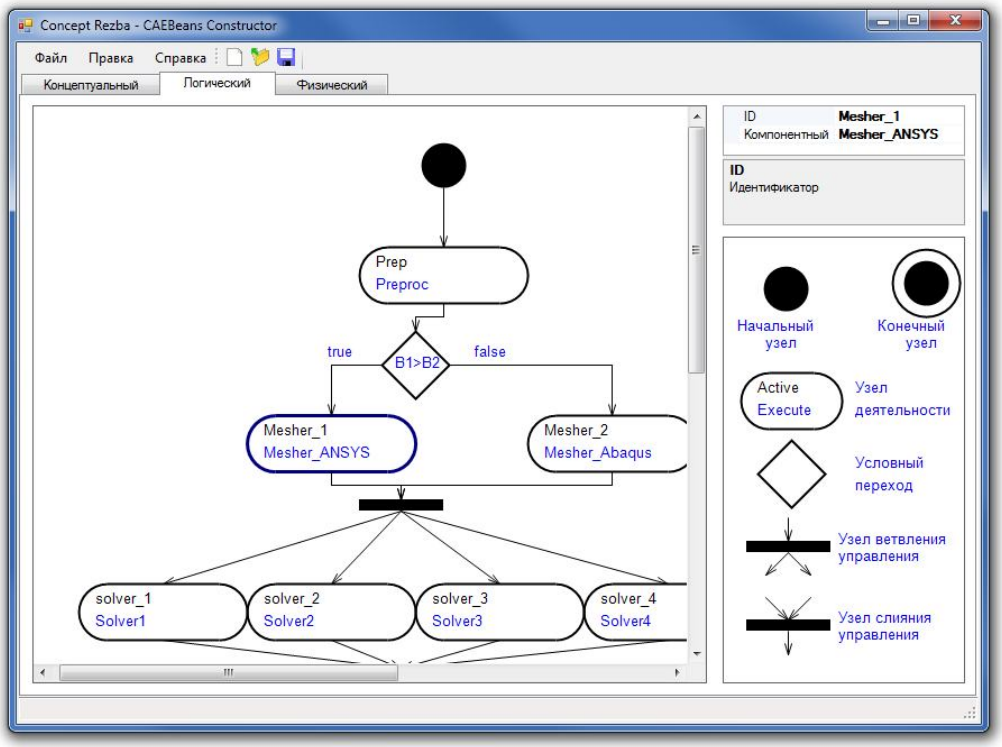


Рис. 18. Пользовательский интерфейс для редактирования логического слоя CAE-проекта.

Для создания нового или редактирования существующего компонентного CAEBean программисту предоставляется пользовательский интерфейс физического слоя. Программисту необходимо выбрать базовый системный CAEBean, интерфейсу которого будет соответствовать новый компонентный CAEBean. Система CAEBeans Constructor поддерживает возможность импорта описания новых системных CAEBeans, разработанных системным программистом. Далее, разработчик указывает входные и выходные параметры для созданного компонентного CAEBean, включая параметры-файлы, обеспечивающие постановку задачи инженерного моделирования и анализ полученных результатов.

Разработанный CAE-проект можно сохранить на компьютере прикладного программиста, или внедрить в систему CAEBeans, экспортировав CAE-проект в CAEBeans Server, посредством метода `putCAEProject()`.

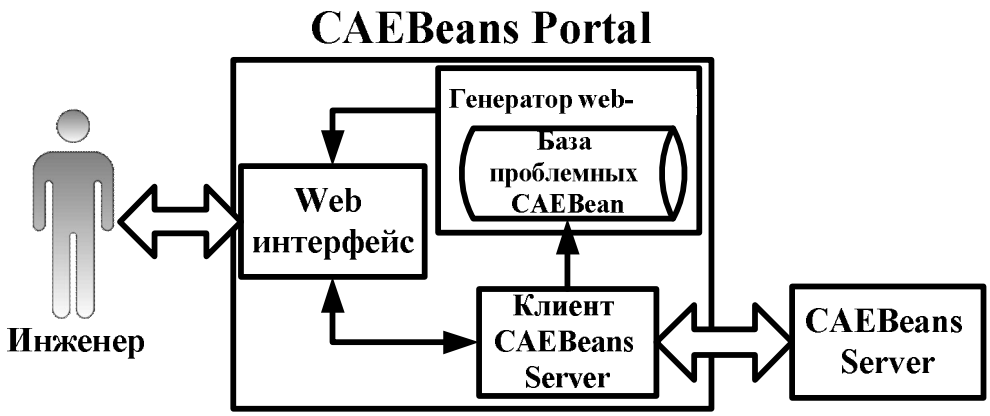


Рис. 19. Схема CAEBeans Portal.

3.3. Клиент

CAEBeans Portal – это веб-приложение, доступное через интернет, обеспечивающее пользовательский интерфейс для постановки и решения задач инженерного моделирования средствами системы CAEBeans.

В соответствии с функциональной нагрузкой, можно выделить три части CAEBeans Portal: пользовательский веб-интерфейс, генератор веб-форм и клиент CAEBeans Server (см. рис. 19).

При входе в систему, производится авторизация пользователя, после чего ему предоставляется список РаВИС, доступных для моделирования, а также список задач, запущенных пользователем ранее с указанием их текущего статуса. Для решения новой задачи инженерного моделирования, пользователь должен выбрать РаВИС, на основе которого будет произведена постановка задачи. После выбора РаВИС, пользователю предлагается интерфейс проблемного CAEBean, соответствующего выбранной задаче.

Генератор веб-форм обеспечивает хранение проблемных оболочек CAEBeans, импортированных из CAEBeans Server и автоматическую генерацию веб-форм для постановки CAE-заданий на основе описания CAE-параметров соответствующих проблемных CAEBean.

Графический интерфейс постановки задачи инженерного моделирования соответствует и полностью определяется описанием параметров ин-

женерного моделирования базового проблемного CAEBean. Параметры разделены на группы, соответствующие группам параметров проблемного CAEBean. На основе информации о типе параметра инженерного моделирования, выбирается метод ввода параметра: выпадающий список, переключатель, или же поле ввода. Атрибуты класса `CAEParameter` `units` и `comment` позволяют привести описание сущности указанного CAE-параметра в пользовательском интерфейсе. Значение параметра по умолчанию выставляется в соответствии со значением атрибута `default`. Таким образом, прикладному программисту не требуется дополнительно формировать интерфейс для инженера для каждого нового проблемного CAEBean.

Клиент CAEBeans Server обеспечивает постановку и получение результатов решения CAE-заданий от CAEBeans Server, а также импорт существующих проблемных оболочек CAEBeans для дальнейшей генерации веб-форм для новых РаВИС.

3.4. Сервер

CAEBeans Server – это грид-сервис, обеспечивающий хранение и интерпретацию CAE-проектов. Стуктурно, CAEBeans Server состоит из следующих подсистем (см. рис. 20):

- *интерфейс* предоставляет возможности по взаимодействию с системой CAEBeans Server со стороны CAEBeans Portal и импорт CAE-проектов из CAEBeans Constructor;
- *хранилище* обеспечивает хранение CAE-проектов, библиотеки системных CAEBean, промежуточных и конечных результатов решения задач инженерного моделирования;
- *менеджер CAE-задач* отвечает за создание, управление работой и уничтожение CAE-заданий;

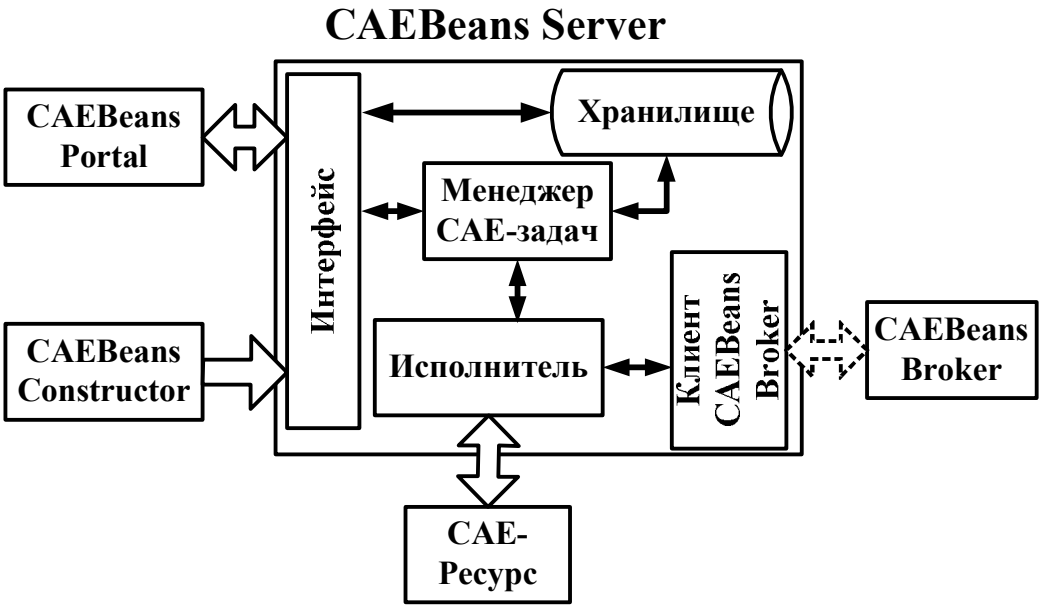


Рис. 20. Структура CAEBeans Server.

– *исполнитель* обеспечивает исполнение отдельного CAE-задания на базе определенного CAE-проекта с уникальным набором значений входных параметров;
– *клиент CAEBeans Broker* отвечает за взаимодействие с системой CAEBeans Broker для поиска и предоставления CAE-ресурсов.

Можно выделить следующие основные методы, предоставляемые интерфейсом CAEBeans Server:

– `GUID createJob(GUID projectId, GUID problemCAEBeanId, ParameterValues[] userDefinedValues)`: создать CAE-задание.

  входные параметры: `projectId` – идентификатор CAE-проекта, `problemCAEBeanId` – идентификатор проблемного CAEBean, `userDefinedValues` – значения входных параметров, указанные инженером при постановке задачи;

  результат: уникальный идентификатор сформированного CAE-задания;

– `CAEJobProperties getJobProperties(GUID jobId)`: получить свойства указанного CAE-задания;

  входные параметры: `jobId` – уникальный идентификатор CAE-задания;

  результат: структура, содержащая свойства CAE-задания, такие, как текущий статус, время постановки задачи, время окончания решения, ссылки на файлы, содержащие текущие результаты решения;

– `void terminateJob()`: остановить исполнение задания;

– `void putCAEProject(CAEProject project)`: импортировать CAE-проект, разработанный в CAEBeans Constructor;

– `ProblemCAEBean getProblemCAEBean(GUID problemCAEBeanId)`: экспортировать проблемный CAEBean в CAEBeans Portal для формирования пользовательского интерфейса.

Функционально, CAEBeans Server представляет собой грид-сервис на базе стандарта WSRF:

– при решении задачи инженерного моделирования, каждый запрос к CAEBeans Server происходит в контексте CAE-задания, уникальный идентификатор которого обеспечивает формирование контекста исполнения запроса по стандарту WS-Addressing;

– свойствами ресурса, предоставляемые CAEBeans Server по стандарту WS-ResouceProperties, являются свойства текущего CAE-задания;

– обеспечивается подписка на базе стандарта WS-Notification об изменении состояния свойств CAE-задания;

– время жизни CAE-задания управляется на базе стандарта WS-ResourceLifetime.

3.5. CAE-ресурс

CAE-ресурсом является экземпляр системного CAEBean, предоставляющий ресурсы некоторого инженерного пакета на базе конкретной целе-

вой системы. Интерфейс, предоставляемый CAE-ресурсом, определяется классом `SystemCAEBeanInterface`, описание которого приведено в п. 3.1.7, и обеспечивает:

– получение данных для решения задачи средствами базового инженерного пакета из CAEBeans Server или внешнего источника данных;
– запуск и автоматизированное решение задачи инженерного моделирования;
– передачу результатов решения CAEBeans Server или во внешнее хранилище данных.

Целевая система – это совокупность грид-сервисов, которые имеют доступ к пространству программных, аппаратных и лицензионных ресурсов некоторого узла грид, и поддерживает аутентификацию и авторизацию пользователей. Целевая система CAEBeans:

– формируется на базе целевой системы UNICORE;
– обеспечивает авторизацию и аутентификацию пользователей по протоколам безопасности грид;
– является контейнером CAE-ресурсов и обеспечивает их динамическую конфигурацию;
– обеспечивает взаимодействие грид-узла с CAEBeans Server (постановка и получение результатов решения подзадач, оповещение об изменении статуса подзадач и др.);
– обеспечивает взаимодействие грид-узла с CAEBeans Broker (предоставление информации о текущем состоянии (статических и динамических характеристиках) целевой системы, резервирование и освобождение вычислительных ресурсов и др.).

Целевая система UNICORE предоставляет набор стандартных сервисов UNICORE Atomic Services [25, 103], поддерживающих управление задачами и обмен данными:

– *TargetSystemFactory*: обеспечивает создание новых сервисов целевых систем;
– *TargetSystemService*: сервис, предоставляющий интерфейс для постановки задач на удаленную вычислительную систему;
– *JobManagementService*: обеспечивает управление и мониторинг исполнения задач на целевой системе;
– *StorageManagementService*, *FileTransferService*: обеспечивают обмен файлами постановки при постановке задачи и получении результатов решения.

Системные CAEBeans реализуются в виде специальных сервисов целевой системы, обеспечивающих исполнение действий инженерного моделирования. Комплект системных CAEBeans разрабатывается отдельно для каждого пакета инженерного моделирования. Также, реализация системных CAEBeans одного вида может различаться в зависимости от версии инженерного пакета, или от базовой операционной системы.

3.6. Брокер

CAEBeans Broker обеспечивает автоматизированную регистрацию, поиск и выделение CAE-ресурсов для реализации действий инженерного проектирования. В основе архитектуры CAEBeans Broker лежит понятие CAE-ресурса, обеспечивающее виртуализацию физических ресурсов, доступных в грид. При решении задач инженерного проектирования в грид необходимо учитывать значительные ограничения, возлагаемые на вычислительные ресурсы, реализующие отдельные действия. Они могут быть вызваны различными причинами: различного типа лицензионные ограничения; низкий уровень масштабируемости конкретного класса задач на суперкомпьютерные системы определенной архитектуры и др. В связи с этим, становится неоптимальным предоставление ресурсов всей целевой системы под нужды определенной задачи. Данная проблема решается посредством

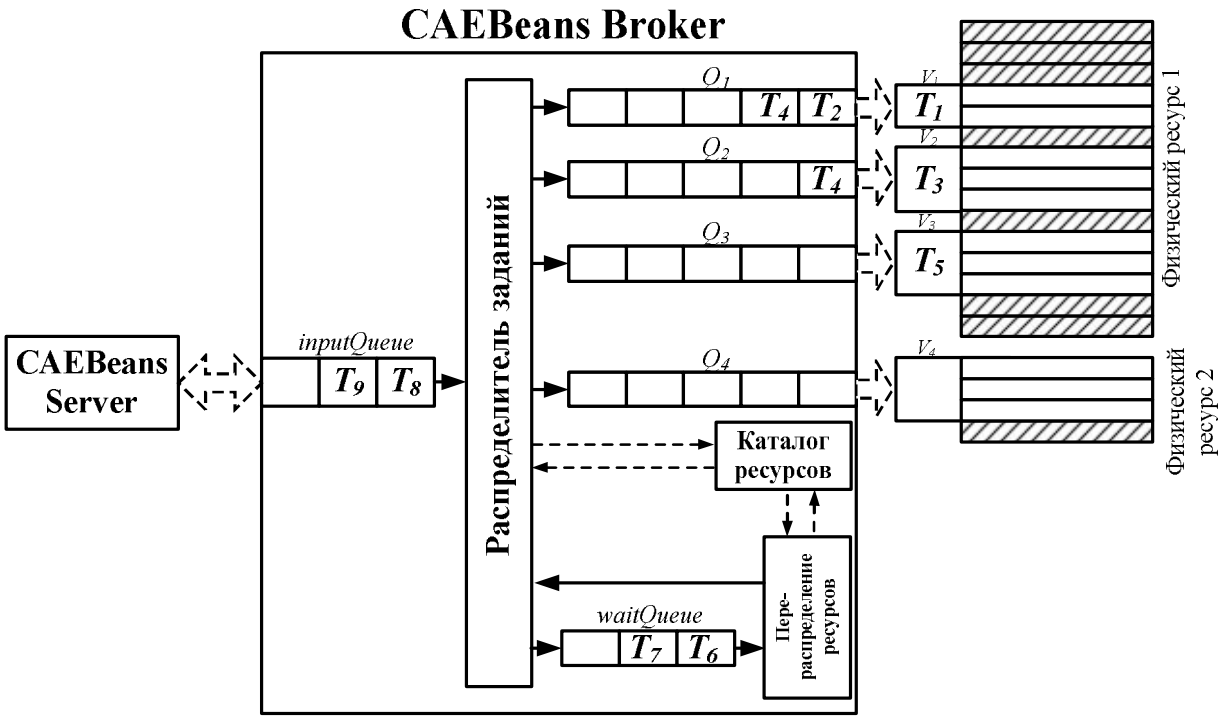


Рис. 21. Структура очередей CAEBeans Broker.

реализации виртуальных CAE-ресурсов, которые инкапсулируют определенную часть физических ресурсов (вычислительных и лицензионных) отдельного узла грид.

Система CAEBeans Broker обеспечивает распределение действий инженерного моделирования по виртуальным CAE-ресурсам. В структуре системы CAEBeans Broker можно выделить следующие основные блоки (см. рис. 21):

- *входная очередь заданий (inputQueue)* обеспечивает получение от сервера запросов на предоставление CAE-ресурсов;
- *распределитель заданий* обеспечивает анализ поступающих запросов и их распределение по очередям заданий;
- *каталог ресурсов* хранит информацию о характеристиках всех физических вычислительных узлов, входящих в грид, и виртуальных CAE-ресурсах находящихся на данных узлах;

– *блок перераспределения ресурсов* обеспечивает возможность удаления старых и формирования новых CAE-ресурсов на основе доступных физических ресурсов;
– *очередь ожидания (waitQueue),* в которую попадают задания, которые не могут быть исполнены на существующем наборе CAE-ресурсов, но можно использовать перераспределение для получения CAE-ресурсов, удовлетворяющих требованиям.

Интерфейс CAEBeans Broker предоставляет следующие методы:

– `GUID resourceQuery(Query query)`: найти ресурс соответствующий требованиям действия;

входные параметры: запрос, содержащий требования к предоставляемому ресурсу, такие как тип необходимого системного CAEBean, базовая операционная система, максимальное время исполнения задачи, требования к аппаратной платформе;

результат: идентификатор задачи в брокере ресурсов;

– `void removeTask(GUID taskId)`: удалить задание из всех очередей;

входные параметры: уникальный идентификатор задания брокера;

– `void lockResource(GUID taskId, GUID CAEResourceId)`: закрепить CAE-ресурс для выполнения определенного действия;

входные параметры: `taskId` - уникальный идентификатор задания брокера; `CAEResourceId` - уникальный идентификатор CAE-ресурса;

– `void unlockResource(GUID taskId, GUID CAEResourceId)`: освободить CAE-ресурс после исполнения действия;

входные параметры: `taskId` - уникальный идентификатор задания брокера; `CAEResourceId` - уникальный идентификатор CAE-ресурса.

Рассмотрим *алгоритм предоставления CAE-ресурса* для реализации действия.

1. При исполнении очередного действия, физический CAEBean формирует запрос `query` на предоставление CAE-ресурса, и CAEBeans Server передает его CAEBeans Broker при вызове метода `resourceQuery();`
2. CAEBeans Broker анализирует доступные виртуальные и физические ресурсы. Если существует возможность решить задачу существующими CAE-ресурсами, он ставит задачу ко всем ресурсам, подходящим по запросу и возвращает CAEBeans Server уникальный идентификатор задания брокера `taskId`. Если существует возможность перераспределить ресурсы для формирования нового CAE-ресурса, отвечающего требованиям запроса, CAEBeans Broker ставит задачу в очередь *waitQueue* и возвращает CAEBeans Server уникальный идентификатор задания `taskId`. Если нет физических ресурсов, обеспечивающих решение данной задачи, CAEBeans Broker возвращает ошибку «*Нет ресурсов, соответствующих запросу*».
3. Если задача занимает первое место в очереди к CAE-ресурсу, CAEBeans Broker производит оповещение CAEBeans Server о том, что существует возможность решить задачу на выделенном ресурсе, указывая при этом место текущей задачи в очередях ко всем остальным ресурсам.
4. Если CAEBeans Server решает захватить требуемый ресурс, он вызывает метод `lockResource()`, захватывая предложенный ресурс.

При этом задача автоматически снимается со всех очередей к альтернативным CAE-ресурсам. Дальнейшая постановка и решение задачи на захваченном ресурсе производится без участия CAEBeans Broker. По окончании решения, CAEBeans Server вызывает метод `unlockResource()`, сигнализируя CAEBeans Broker о том, что ресурс освободился.

5. Во время работы CAEBeans Broker, периодически производится процедура перераспределения CAE-ресурсов для обеспечения оптимального времени решения задач. На основе анализа задач в очереди *waitQueue* может быть принято решение на формирование более крупного CAE-ресурса. В этом случае:
   a) производится поиск ресурсов, суммарная мощность которых может обеспечить решение поставленной задачи;
   b) они маркируются, после чего в очереди к ним не производится постановка новых задач;
   c) как только очереди ко всем помеченным ресурсам очищены от задач, производится удаление этих CAE-ресурсов и формирование нового более крупного ресурса. К нему в очередь ставятся соответствующие задачи из очереди *waitQueue*.

3.7. Взаимодействие компонентов системы CAEBeans

Рассмотрим, каким образом компоненты системы CAEBeans обеспечивают процесс постановки и решения задачи инженерного моделирования (см. рис. 22).

1. Инженеру предоставляется доступ к веб-страницам CAEBeans Portal. Он авторизуется на данном портале и выбирает РаВИС. Посредством пользовательского интерфейса, предоставляемого CAEBeans Portal, инженер указывает значения CAE-параметров и интересующие его результаты решения, после чего запускает процесс решения задачи.

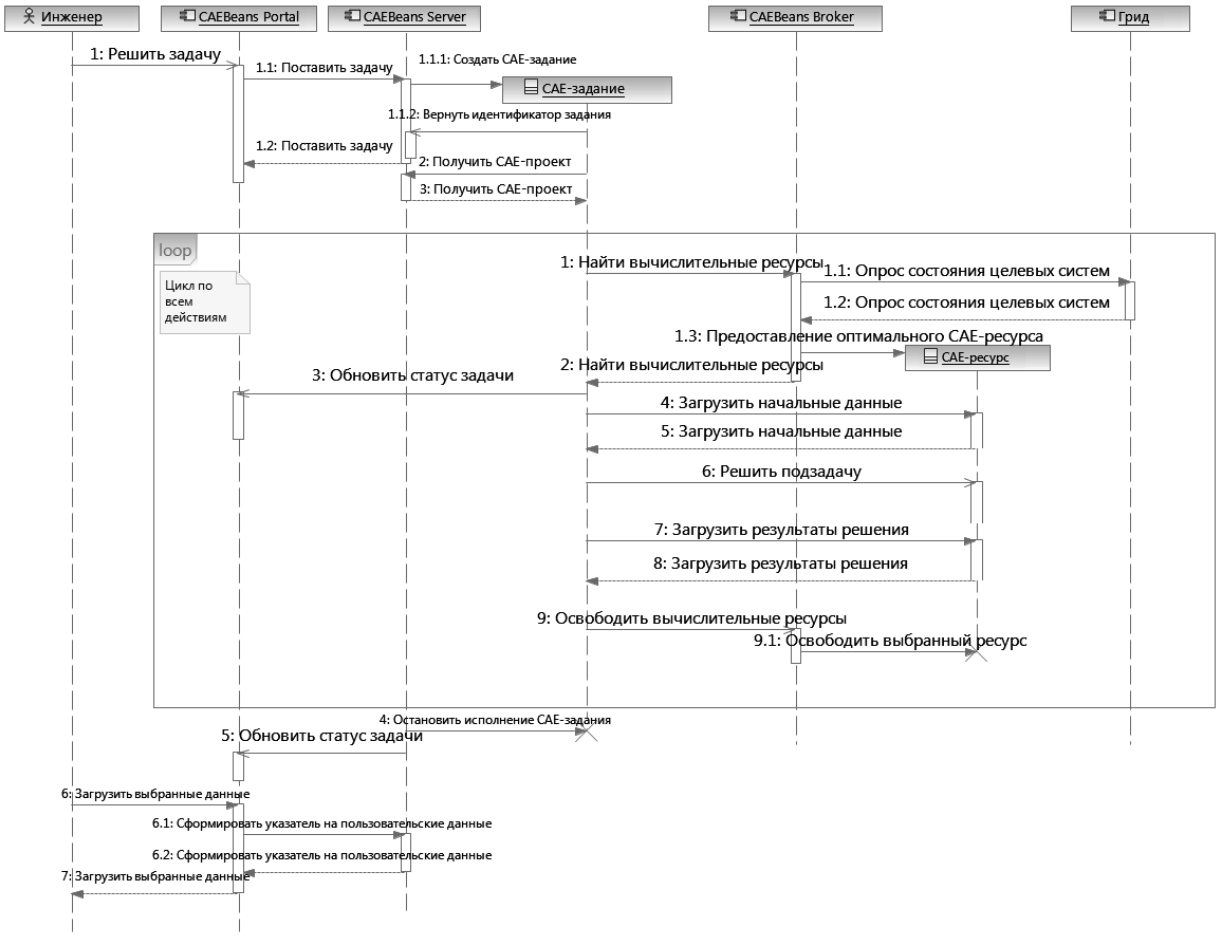


Рис. 22. Диаграмма последовательности постановки и решения задачи инженерного моделирования в системе CAEBeans.

2. CAEBeans Portal формирует проблемный CAEBean, устанавливая значения всех CAE-параметров, введенные пользователем, и отправляет запрос CAEBeans Server на создание нового CAE-задания с соответствующими входными значениями. В ответ на данный запрос, CAEBeans Server формирует новый контекст исполнения задачи, и передает уникальный идентификатор CAE-задания (в терминах WSRF – идентификатор ресурса) системе CAEBeans Portal. В дальнейшем, любую информацию о ходе решения задачи можно получить, указав данный идентификатор CAE-задания в запросе CAEBeans Server.

3. В CAE-задание загружаются данные о соответствующем CAE-проекте, включая дерево проблемных оболочек CAEBeans, и производится инициализация всех CAE-параметров задачи. После этого начинается последовательное исполнение логического плана решения задачи. Дальнейшие шаги последовательно повторяются для каждого узла деятельности в составе логического плана вплоть до окончания решения задачи:
   3.1. Компоненту CAEBeans Broker отправляется запрос на выбор оптимального вычислительного ресурса. Он производит анализ состояния доступных узлов грид и соответствующих целевых систем, после чего резервирует и передает адрес наиболее подходящего CAE-ресурса системе CAEBeans Server.
   3.2. CAEBeans Server обновляет статус задачи и оповещает об этом CAEBeans Portal.
   3.3. CAEBeans Server производит загрузку начальных данных на выделенный CAE-ресурс, после чего, инициирует процесс решения подзадачи.
   3.4. По окончании решения подзадачи, CAE-ресурс производит выгрузку результатов решения на выделенный сервер хранения результатов (в зависимости от задачи это может быть CAE-Beans Server; узел грид, ответственный за следующий этап решения CAE-задания или же какой-либо другой узел грид).
   3.5. CAEBeans Server отправляет запрос CAEBeans Broker на освобождение текущего CAE-ресурса.
4. CAEBeans Server обновляет статус задачи на «Успешно решена» и останавливает исполнение CAE-задания.

5. Инженер заходит на CAEBeans Portal и выбирает данные, которые хочет загрузить на локальный компьютер. CAEBeans Portal обращается к CAEBeans Server и получает ссылки на запрошенные данные, после чего, производится загрузка результатов решения пользователю.

3.8. Выводы по главе 3

В рамках диссертационного исследования был разработан программный комплекс система CAEBeans, обеспечивающий поддержку всех этапов разработки и исполнения РаВИС. Была разработана архитектура программного комплекса, описаны основные интерфейсы, предоставляемые компонентами системы CAEBeans. Также, были исследованы процедуры взаимодействия между компонентами системы CAEBeans в процессе постановки и решения задачи инженерного моделирования средствами РаВИС.

ГЛАВА 4. ИСПЫТАНИЯ СИСТЕМЫ CAEBEANS

Для проверки возможностей, предоставляемых системой CAEBeans в области создания и исполнения виртуальных испытательных стендов, был разработан набор РаВИС, обеспечивающих решение различных задач инженерного моделирования средствами различных CAE-пакетов. Были произведены исследования методов взаимодействия с современными инженерными пакетами, и проанализированы API, предоставляющие методы автоматизированного решения задач инженерного моделирования. Для испытания системы CAEBeans были разработаны оболочки для решения задач средствами наиболее распространенных пакетов инженерного моделирования: ANSYS CFX [15], ANSYS Mechanical [8], ABAQUS [10], DEFORM [6, 7].

В качестве основной испытательной задачи была выбрана задача моделирования процесса закалки и охлаждения труб и анализа влияния различных аспектов процесса закалки на качество производимой продукции («Распределенный виртуальный испытательный стенд «Термообработка»), решаемая по заказу ОАО «Челябинский трубопрокатный завод».

4.1. Испытание системы CAEBeans на базе DEFORM

Основные испытания системы CAEBeans производились на базе виртуального испытательного стенда, моделирующего процесс овализации труб при их закалке и последующем отпуске на ОАО «Челябинский трубопрокатный завод» [6]. Процесс изготовления цельнокатаных труб является технологически сложным и недетерминированным по такому параметру, как разнотолщинность стенок изготавливаемой трубы. После изготовления труба поступает в цех термической обработки, где производится ее закалка и отпуск на индукционной установке. При разогреве трубы и последующем охлаждении в ряде случаев возникает эффект овализации концов трубы.

Трубы с подобным дефектом технически невозможно сваривать в трубопровод. Челябинский трубопрокатный завод использует прессовые установки, для устранения брака продукции. Однако подобный способ решения проблемы брака является устаревшим и экономически нецелесообразным, так как приводит к большим финансовым потерям [6]. В соответствии с этим актуальной является задача создания виртуального испытательного стенда как средства проведения вычислительных экспериментов, с целью поиска оптимального решения по изменению существующей технологии производства труб.

Перед созданием компьютерной модели процесса термической обработки были произведены тепловизионные исследования процесса закалки труб непосредственно на производстве. Был произведен сбор информации о геометрии заготовок и температурных полях, величине разностенности с разверткой по длине и окружности заготовки, величине начальной овальности с разверткой по длине и кривизне оси заготовки.

Для создания компьютерной модели процесса термической обработки труб на индукционных установках был выбран инженерный пакет DEFORM. В рамках поставленной задачи пакет DEFORM имеет ряд преимуществ, так как предназначен для анализа процессов обработки металлов давлением, термической и механической обработки. Кроме того, в пакете DEFORM есть модуль Microstructure 3D, который предназначен для моделирования трехмерных процессов термической обработки и позволяет моделировать микроструктурные превращения в заготовке при ее деформировании.

Пакет DEFORM предоставляет возможность производить постановку задачи, ее решение и обработку результатов без запуска графического интерфейса, путем использования пакетного режима препроцессора, решателя и постпроцессора [7]. Пакетный режим препроцессора позволяет производить весь цикл постановки задачи, на основе текстового `*.key` файла с ее

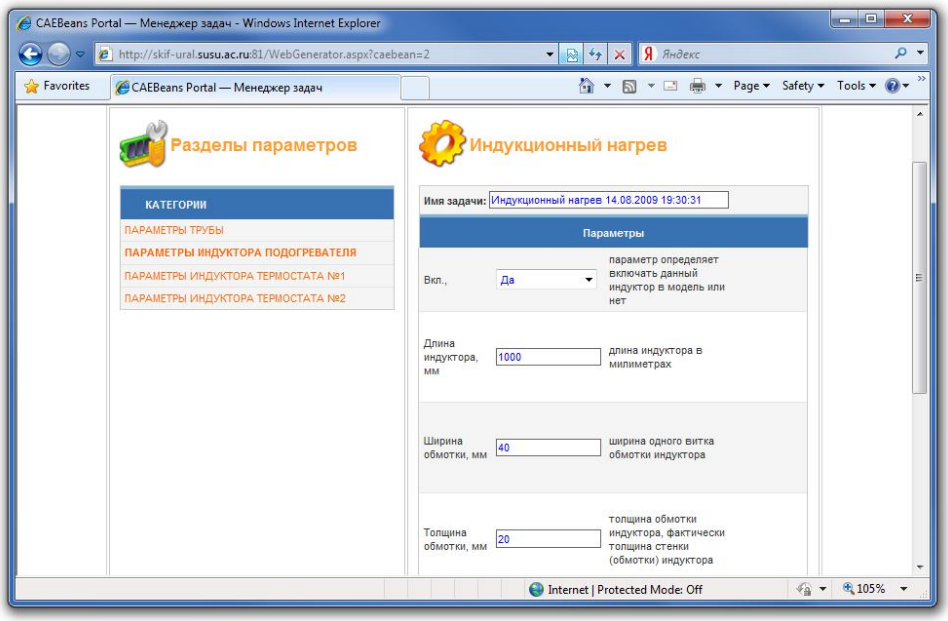


Рис. 23. Интерфейс постановки задачи инженерного моделирования предоставляемый CAEBeans Portal.

описанием. Решатель позволяет произвести запуск решения задачи в пакетном режиме, указав имя базы данных задачи. Постпроцессор в пакетном режиме позволяет сохранять изображения модели на разных шагах расчета, с выбором требуемых данных для отображения: температура, фазовый состав, деформации и т.д.

При разработке проблемного CAEBean была предусмотрена возможность изменения следующих технических параметров индукционной установки: количество индукторов, частота и сила тока, длина индукторов, количество и конфигурация водяных струй, давление и расход воды, скорость движения трубы через индукционную установку, частота вращения труб (см. рис. 23). Также, была предусмотрена возможность моделирования термообработки труб из различных марок сталей (путем указания физических характеристик материала).

В результате моделирования из файла базы данных пакета DEFORM сохраняются все необходимые результаты: данные о распределении температур в трубе, данные о фазовом составе вещества трубы, данные о деформациях трубы, графики смещения выбранной точки трубы (см. рис. 24).

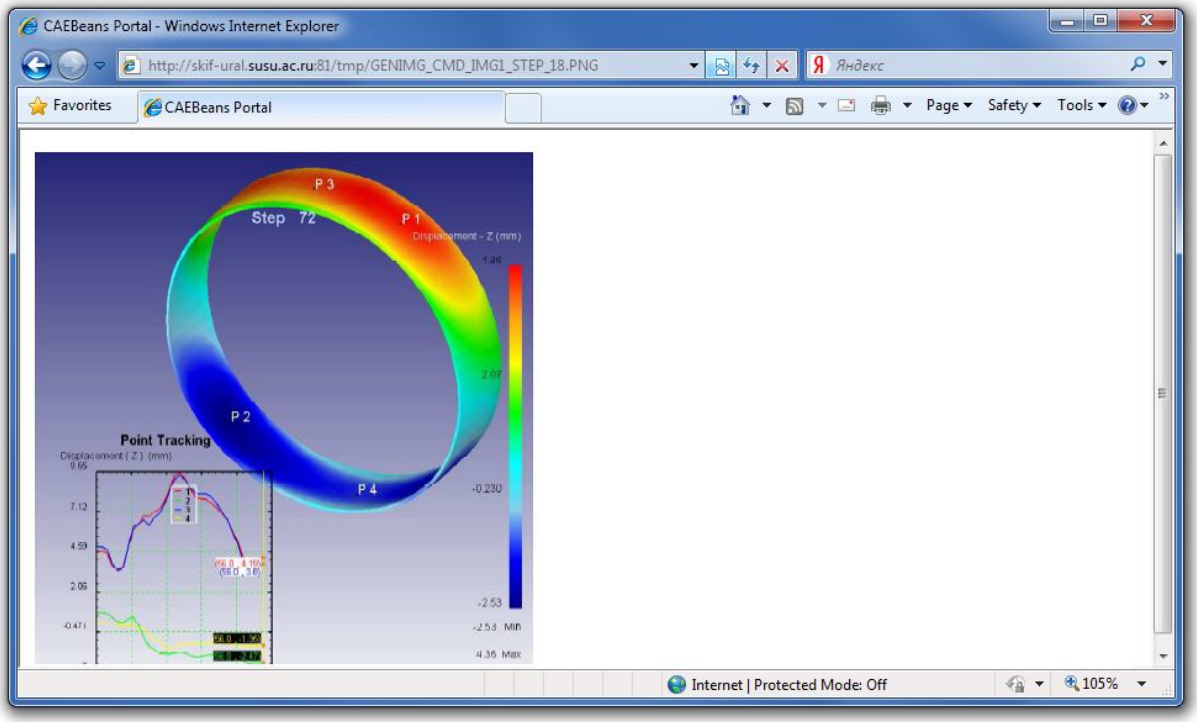


Рис. 24. Пример результата моделирования задачи овализации трубы при закалке.

Результаты предоставляются инженеру в виде изображений формата PNG и табличных данных.

Система CAEBeans была развернута на базе суперкомпьютерных ресурсов СКЦ ЮУрГУ [18]. Системы CAEBeans Portal, CAEBeans Server и CAEBeans Broker были установлены на вычислительных узлах суперкомпьютера Infinity. В системе CAEBeans Constructor был разработан CAE-проект РаВИС. В CAEBeans Portal был загружен проблемный CAEBean созданного виртуального испытательного стенда. На CAEBeans Portal была создана тестовая учетная запись и для нее предоставлены права на разработанный виртуальный испытательный стенд.

На узлах суперкомпьютера СКИФ Урал была установлена и настроена грид-система Unicore [25]. В Unicore была создана целевая система, реализующая системный CAEBean для взаимодействия с пакетом DEFORM. На вычислительных узлах суперкомпьютера СКИФ Урал был установлен пакет DEFORM.

Доступ инженера к CAEBeans Portal осуществляется посредством интернет-обозревателя. После авторизации менеджер задач отображает список доступных испытательных стендов и список запущенных расчетов. В менеджере задач можно создать и запустить новую задачу моделирования индукционного нагрева. По окончанию расчетов изображения с результатами моделирования доступны в разделе результатов задачи.

4.2. Испытание системы CAEBeans на базе ANSYS Mechanical

Для тестирования возможности формирования РаВИС на базе пакета ANSYS Mechanical был разработан виртуальный испытательный стенд для моделирования резьбового соединения обсадных и насосно-компрессорных труб для нефтяных скважин [8].

Обсадные трубы используются для защиты оборудования, опущенного в скважину, от внешних воздействий, таких как загрязнение, механическое воздействие. Обсадные трубы соединяются с помощью муфт. При углублении скважины на обсадную колонну навинчивают очередную трубу и опускают глубже. Данный вид труб свинчивается однократно и подвергается изгибающим и растягивающим нагрузкам во время эксплуатации.

Насосно-компрессорные трубы имеют диаметр от 80 до 140 мм, располагаются внутри обсадных и используются для подачи напора воды и получения нефти. Эти трубы подвергаются многократному свинчиванию и развинчиванию, что приводит к сильному износу резьбы и невозможности эксплуатации труб.

Поэтому, одной из задач, возникающих на практике в нефтяной промышленности, является задача построения параметризованной модели резьбового соединения двух труб с помощью муфты. Модель предназначена для разработки новых видов резьбовых соединений для обсадных и насосно-компрессорных труб, которые могли бы обеспечить более совершенные характеристики соединения.

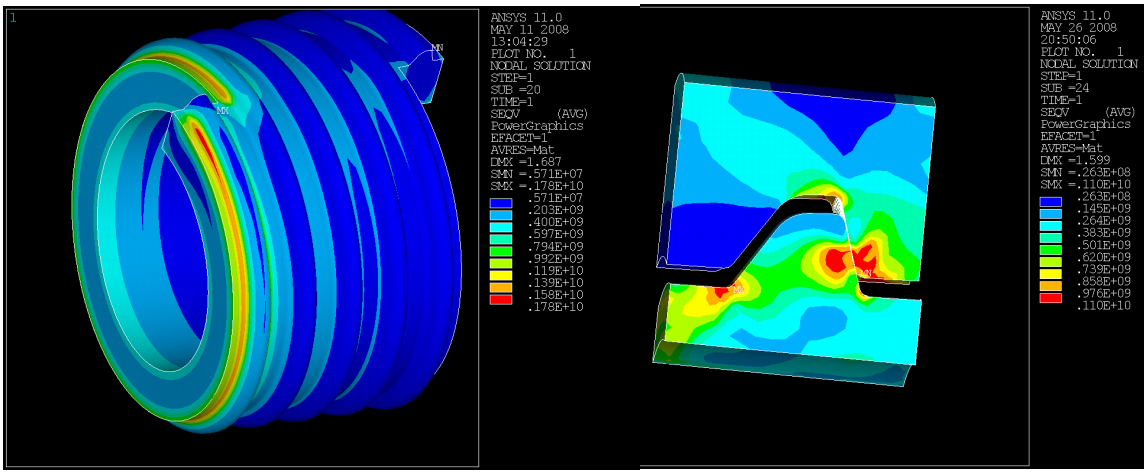


Рис. 25. Пример результатов моделирования резьбы обсадной трубы.

Моделирование резьбового соединения было произведено в пакете ANSYS Mechanical. Для создания системных CAEBeans, использовался режим пакетной обработки, поддерживаемый ANSYS Mechanical. В качестве входных параметров постановки задачи используется log-файл. Log-файл представляет собой набор команд, выполняя которые пошагово в консольной строке ANSYS Mechanical, можно в воспроизвести действия, производимые в GUI за время работы с ANSYS Mechanical.

В разработанном проблемном CAEBean выделены три группы проблемных параметров:

1. параметры трубы (такие как наружный диаметр, толщина стенки, длина конусной части);
2. параметры резьбы (шаг резьбы, высота профиля);
3. нагрузка (растяжение, свинчивание, коэффициент трения).

Результаты моделирования предоставляются пользователю в графическом (см. рис. 25) и табличном виде.

4.3. Испытание системы CAEBeans на базе Abaqus

Для испытания системы CAEBeans на базе конечно-элементного CAE-пакета ABAQUS, был разработан РаВИС для моделирования напряженно-

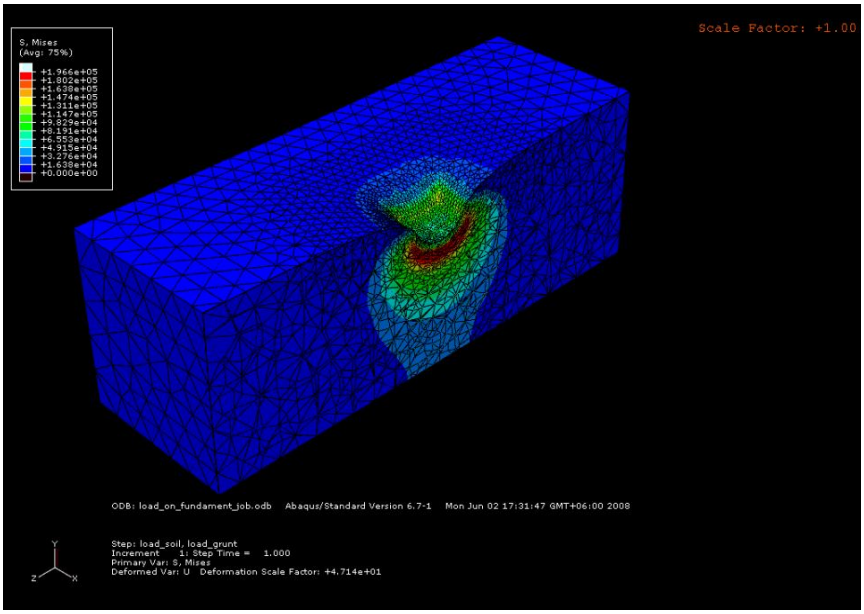


Рис. 26. Пример результатов моделирования грунтового массива.

деформированного состояния грунтового массива под массивным зданием или сооружением [10].

Взаимодействие с CAE-пакетом ABAQUS было реализовано посредством Интерфейса Сценариев ABAQUS. Интерфейс Сценариев ABAQUS является расширением объектно-ориентированного языка Python [101], и обеспечивает взаимодействие с функциональными возможностями, предоставляемыми компонентами, входящими в пакет ABAQUS. Интерфейс Сценариев ABAQUS обеспечивает:

– создание и модификацию компонентов модели ABAQUS, таких как детали, материалы, нагрузки, шаги расчета;
– создание, изменение и запуск вычислительного процесса в ABAQUS;
– чтение и запись базы данных модели;
– просмотр и анализ результатов расчета.

Проблемный CAEBean обеспечивает изменение таких параметров модели как размеры грунтового массива, нагрузка, размеры нагружаемой области. Результаты моделирования предоставляются пользователю в графическом виде (см. рис. 26).

4.4. Испытание системы CAEBeans на базе ANSYS CFX

Для испытания взаимодействия системы CAEBeans и комплекса ANSYS CFX был разработан тестовый РаВИС, обеспечивающий моделирование обдувания дымовой трубы.

В ходе анализа пакета ANSYS CFX, было выявлено несколько способов взаимодействия внешних систем с компонентами, составляющими данный пакет. Наиболее приемлемый способ автоматизации взаимодействия с компонентами пакета ANSYS CFX – это запуск и исполнение его компонентов в командном режиме [15].

Все подсистемы пакета ANSYS CFX (ANSYS CFX-Pre, ANSYS CFX-Solver, ANSYS CFX-Post) поддерживают работу в пакетном режиме посредством указания специального флага в командной строке. Также, каждый компонент пакета ANSYS CFX поддерживает возможность автоматизированной постановки действий инженерного моделирования посредством специальных форматов импортируемых файлов:

– файлы сессии ANSYS CFX-Pre (`*.pre`);
– файл CCL («Command Language File»), обеспечивающий отображение параметров задачи инженерного моделирования в текстовом формате;
– файлы сессии ANSYS CFX-Post (`*.cse`).

Таким образом, указав необходимые параметры в командной строке или входных файлах, можно поставить требуемое действие инженерного моделирования компоненту пакета ANSYS CFX и получить результаты ее решения.

Проблемный CAEBean обеспечивает изменять скорость и температуру воздушных потоков (как ветра, так и потока, исходящего из трубы). Результаты моделирования предоставляются пользователю в графическом виде (см. рис. 27).

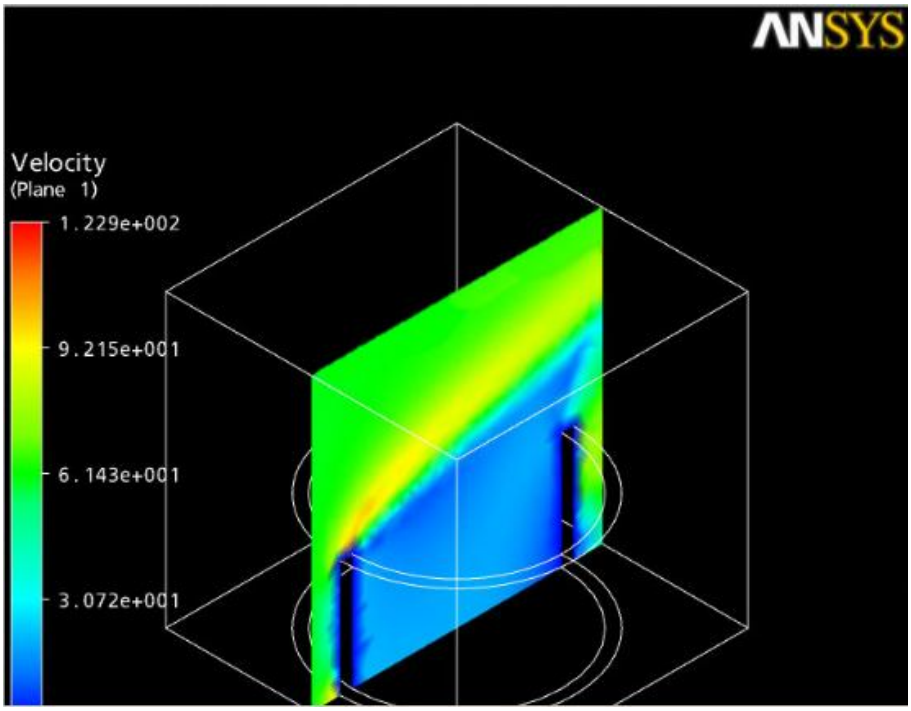


Рис. 27. Пример результатов моделирования обтекания трубы воздушным потоком.

4.5. Выводы по главе 4

Для тестирования технологии использования компонентных CAEBeans были созданы оболочки для решения следующих задач инженерного моделирования:

1) моделирование резьбовых соединений труб для нефтяных скважин (на базе инженерного пакета ANSYS Mechanical);
2) моделирование эффекта овализации труб при термической обработке (на базе инженерного пакета DEFORM);
3) моделирование напряженно-деформированного состояния грунтового массива (на базе инженерного пакета ABAQUS);
4) моделирование обтекания дымовой трубы воздушным потоком (на базе инженерного пакета ANSYS CFX).

Разработка распределенных испытательных стендов для решения данных задач показала возможность применения системы CAEBeans для решения задач из различных областей инженерного моделирования.

ЗАКЛЮЧЕНИЕ

В диссертационной работе были рассмотрены вопросы, связанные с внедрением систем инженерного проектирования и анализа в распределенные вычислительные среды. Было произведено исследование современных подходов по организации распределенных вычислительных систем. Были исследованы основные аспекты внедрения систем инженерного проектирования и анализа в распределенную вычислительную среду. На основе проведенных исследований была предложена концепция распределенного виртуального испытательного стенда, обеспечивающая проблемно-ориентированный подход к решению конкретных классов задач инженерного проектирования посредством ресурсов, предоставляемых вычислительными грид-средами. Была предложена технология CAEBeans, в соответствии с которой, выделяются четыре слоя архитектуры РаВИС, каждый из которых представляется своей оболочкой CAEBeans: Концептуальный слой (проблемный CAEBean), Логический слой (потоковый CAEBean), Физический слой (компонентный CAEBean), Системный слой (системный CAEBean). Разработан прототип комплекса программных средств «Система CAEBeans», обеспечивающий поддержку разработки и исполнения РаВИС. В состав системы входят компоненты: CAEBeans Constructor, CAEBeans Portal, CAEBeans Server, CAEBeans Broker, CAE-ресурсы. На основе разработанного прототипа произведено испытание системы посредством разработки виртуальных испытательных стендов, ориентированных на задач инженерного моделирования на базе ряда наиболее распространенных пакетов инженерного моделирования.

Работа выполнялась при поддержке *Роснауки* (гос. контракт 2007-4-1.4-20-01-026) и *научно-технической программы Союзного государства Россия-Белоруссия "СКИФ-ГРИД"* (контракты 2007-СГ-04/4 и 2009-СГ-03).

В заключение перечислим основные полученные результаты диссертационной работы, приведем данные о публикациях и апробациях, и рассмотрим направления дальнейших исследований в данной области.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ ДИССЕРТАЦИОННОЙ РАБОТЫ

На защиту выносятся следующие новые научные результаты.

1. Разработана модель проблемно-ориентированного сервиса для решения задач инженерного проектирования и анализа в распределенных вычислительных средах.
2. Разработаны архитектура и принципы структурной организации распределенного виртуального испытательного стенда (РаВИС), предоставляющего сервис для решения задач инженерного анализа на основе грид-технологий.
3. Разработан комплекс методов и алгоритмов, позволяющих автоматизировать процесс построения специализированных РаВИС для решения прикладных задач с использованием различных CAE-пакетов.
4. Разработан прототип программной системы CAEBeans, включающий в себя средства автоматического создания и исполнения РаВИС. Произведены испытания системы CAEBeans путем создания РаВИС на базе инженерных пакетов ANSYS CFX, ANSYS Mechanical, ABAQUS, DEFORM. Распределенный виртуальный испытательный стенд «Термообработка», внедрен в опытную эксплуатацию на предприятии ОАО «Челябинский трубопрокатный завод».

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

Основные результаты диссертации полностью опубликованы в следующих работах автора.

1. Радченко Г.И. Технология построения проблемно-ориентированных иерархических оболочек над инженерными пакетами в грид-средах // Системы управления и информационные технологии. № 4(34). 2008. С. 57-61.

2. Радченко Г.И., Соколинский Л.Б. Технология построения виртуальных испытательных стендов в распределенных вычислительных средах // Науч.-техн. вест. СПбГУ ИТМО. № 54. 2008. С. 134-139.

3. Радченко Г.И. Методы организации грид-оболочек системного слоя в технологии CAEBeans // Вестник ЮУрГУ. Серия "Математическое моделирование и программирование" № 15 (115). Вып. 1. 2008. С. 69-78.

4. Радченко Г.И. Грид-система CAEBeans: интеграция ресурсов инженерных пакетов в распределенные вычислительные среды // Параллельные вычислительные технологии (ПаВТ’2009): Тр. междунар. науч. конф. (Н.Н., 30 марта – 3 апреля 2009 г.). Челябинск: Изд-во ЮУрГУ, 2009. С. 281-292.

5. Радченко Г.И., Дорохов В.А., Насибулина Р.С., Соколинский Л.Б., Шамакина А.В. Технология создания виртуальных испытательных стендов в грид-средах // Вторая Международная научная конференция "Суперкомпьютерные системы и их применение" (SSA'2008): доклады конференции (27-29 октября 2008 года, Минск) Минск: ОИПИ НАН Беларуси, 2008. С. 194-198.

6. Радченко Г.И., Соколинский Л.Б., Шамакина А.В. Разработка компонентно-ориентированных CAEBean-оболочек для пакета ANSYS CFX // Параллельные вычислительные технологии (ПаВТ'2008): Труды международной научной конференции (28 января - 1 февраля 2008 г., г. Санкт-Петербург). Челябинск: Изд-во ЮУрГУ, 2008. С. 438-443.

7. Радченко Г.И., Соколинский Л.Б., Кутепов И.С. BeanShells: интеграция CAE-пакетов в GPE // Параллельные вычислительные технологии (ПаВТ'2007): Труды международной научной конференции (Челябинск, 29 января – 2 февраля 2007 г.). Челябинск: Изд. ЮУрГУ, 2007. Т.2., С. 15

8. Свидетельство Роспатента об официальной регистрации программы для ЭВМ № 2008612879. «CAEBeans Toolbox: программная среда для разработки проблемно-ориентированных оболочек для грид» / Юрков В.В., Дорохов В.А., Радченко Г.И., Насибулина Р.С., Шамакина А.В; Заяв. 03.10.2008.

9. Свидетельство Роспатента об официальной регистрации программы для ЭВМ № 2008614998. «CAEBeans Sphere: программное средство для поддержки распределенных вычислительных сред на базе платформы Microsoft.NET» / Юрков В.В., Дорохов В.А., Радченко Г.И., Насибулина Р.С., Шамакина А.В.; Заяв. 03.10.2008.

10. Свидетельство Роспатента об официальной регистрации программы для ЭВМ № 2008612879. «Пакет проблемно-ориентированных оболочек CAEBeans для решения типовых инженерных задач» / Радченко Г.И., Насибулина Р.С., Шамакина А.В., Юрков В.В., Федянин О.Н., Дорохов В.А.; Заяв. 04.05.2008.

Работы 1, 2 опубликованы в журналах, включенных ВАК в перечень журналов, в которых должны быть опубликованы основные результаты диссертаций на соискание ученой степени доктора наук.

АПРОБАЦИЯ РАБОТЫ

Основные положения диссертационной работы, разработанные модели, методы, алгоритмы и результаты вычислительных экспериментов докладывались автором на следующих международных и всероссийских научных конференциях:

- на VI Всероссийской межвузовской конференции молодых ученых (14 - 17 апреля 2009г., Санкт-Петербург);
- на V Всероссийской межвузовской конференции молодых ученых (15 - 18 апреля 2008 г., Санкт-Петербург);
- на Международной научной конференции «Параллельные вычислительные технологии» (30 марта – 3 апреля 2009 г., Нижний Новгород);
- на Международной научной конференции «Параллельные вычислительные технологии» (29 января – 2 февраля 2007 г., Челябинск);
- на Всероссийской научной конференции «Научный сервис в сети Интернет: технологии параллельного программирования» (24-29 сентября 2007 г., Новороссийск).

НАПРАВЛЕНИЯ ДАЛЬНЕЙШИХ ИССЛЕДОВАНИЙ

Теоретические исследования и практические разработки, выполненные в рамках этой диссертационной работы, предполагается продолжить по следующим направлениям.

1. Исследование методов и алгоритмов внедрения систем многокритериальной оптимизации, обеспечивающих работу в грид, в архитектуру распределенного виртуального испытательного стенда.
2. Разработка методов и алгоритмов автоматической генерации исходных кодов РаВИС для определенных классов задач инженерного проектирования и анализа.

ЛИТЕРАТУРА

1. Афанасьев А.П., Волошинов В.В., Кривцов В.Е. О возможных принципах организации доступа к удаленным вычислительным ресурсам на основе CORBA. Тез. докл. 1-й Московской конференции «Декомпозиционные методы в математическом моделировании». ВЦ РАН, Москва, 2001. С. 11-14.
2. Бегунов А.А. Применение результатов моделирования для оптимизации и управления технологическими процессами // Параллельные вычислительные технологии: тр. Междунар. науч. конф. (28 янв. – 1 февр. 2008 г., г. Санкт-Петербург). Челябинск: Изд. ЮУрГУ. 2008. С. 31-38.
3. Бурмакин Е.М., Tuominen J.O. Интеграция технологии CORBA и объектных баз данных // XXIX Неделя науки СПбГТУ. Материалы межвузовской научной конференции. Ч.V. СПб.: Изд-во СПбГТУ, 2001. С. 62-63.
4. Буч Г., Рамбо Д., Якобсон И. Язык UML. Руководство пользователя. 2-е изд. М.: ДМК Пресс, 2007. 496 с.
5. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS. 2-е изд. Спб.: Питер, 2006. 736 с.
6. Дорохов В.А., Маковецкий А.Н., Соколинский Л.Б. Разработка проблемно-ориентированной GRID-оболочки для решения задачи овализации труб при закалке // Параллельные вычислительные технологии: Труды международной научной конференции (28 января - 1 февраля 2008 г., г. Санкт-Петербург). -Челябинск: Изд-во ЮУрГУ. -2008. С. 520.

7. Дорохов В.А. Разработка виртуального испытательного грид-стенда для исследования эффекта овализации труб при термической обработке // Параллельные вычислительные технологии (ПаВТ'2009): Труды международной научной конференции (Нижний Новгород, 30 марта - 3 апреля 2009 г.). Челябинск: Изд-во ЮУрГУ, 2009. С. 457-462.

8. Лёушкин В.В., Соколинский Л.Б., Чайко К.А., Юрков В.В. Разработка проблемно-ориентированной Grid-оболочки для моделирования резьбовых соединений труб для нефтяных скважин в распределенных вычислительных средах // Параллельные вычислительные технологии: Труды международной научной конференции (28 января - 1 февраля 2008 г., г. Санкт-Петербург). Челябинск: Изд-во ЮУрГУ. 2008. С. 534.

9. Лукичев А.С. Интеграция SOA- и классических высокопроизводительных приложений // Научный сервис в сети Интернет: технологии распределенных вычислений: Труды Всероссийск. науч. конф. (18-23 сентября 2006 г., г. Новороссийск). М.: Изд-во МГУ. 2006. С. 42-44.

10. Насибулина Р.С. и др. Методы организации программных интерфейсов к инженерным пакетам в среде GPE // Параллельные вычислительные технологии: Труды международной научной конференции (28 января - 1 февраля 2008 г., г. Санкт-Петербург). Челябинск: Изд-во ЮУрГУ, 2008. С. 537.

11. Радченко Г.И. Грид-система CAEBeans: интеграция ресурсов инженерных пакетов в распределенные вычислительные среды // Параллельные вычислительные технологии (ПаВТ'2009): Труды международной научной конференции (Нижний Новгород, 30 марта - 3 апреля 2009 г.). Челябинск: Изд-во ЮУрГУ, 2009. С. 281-292.

12. Радченко Г.И. Методы организации грид-оболочек системного слоя в технологии CAEBeans // Вестник ЮУрГУ. Серия "Математическое моделирование и программирование". 2008. № 15 (115). Вып. 1. С. 69-80.

13. Радченко Г.И. Технология построения проблемно-ориентированных иерархических оболочек над инженерными пакетами в грид-средах // Системы управления и информационные технологии. № 4(34). 2008. С. 57-61.

14. Радченко Г.И., Соколинский Л.Б. Технология построения виртуальных испытательных стендов в распределенных вычислительных средах // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. № 54. 2008. С. 134-139.

15. Радченко Г.И., Соколинский Л.Б., Шамакина А.В. Разработка компонентно-ориентированных CAEBean-оболочек для пакета ANSYS CFX // Параллельные вычислительные технологии (ПаВТ'2008): Труды международной научной конференции (28 января - 1 февраля 2008 г., г. Санкт-Петербург). Челябинск: Изд-во ЮУрГУ, 2008. С. 438-443.

16. Радченко Г.И., Соколинский Л.Б. CAEBeans: иерархические системы структурированных проблемно-ориентированных оболочек над инженерными пакетами// Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ: Труды Всероссийск. науч. конф. (24-29 сентября 2007 г., г. Новороссийск). М.: Изд-во МГУ, 2007. С. 54-57.

17. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. СПб.: Питер, 2007. 544 с.

18. Суперкомпьютерный центр Южно-Уральского государственного университета. URL: http://supercomputer.susu.ru/ (дата обращения: 19.05.2009)

19. Черняк Л. Web-сервисы, grid-сервисы и другие // Открытые системы. СУБД. №12. 2004. С. 20-27.

20. Шамакина А.В. Организация брокера ресурсов в системе CAEBEANS // Научный сервис в сети Интернет: решение больших задач: Труды Всероссийск. науч. конф. (22-27 сентября 2008 г., Новороссийск). М.: Изд-во МГУ, 2007. С. 326-327.

21. Шамакина А.В. Организация брокера ресурсов в системе CAEBeans // Вестник Южно-Уральского государственного университета. Серия "Математическое моделирование и программирование". 2008. № 27(127). Вып. 2. С. 110-116.

22. Aleksy M., Gitzel R. Relationship between the standard compliance of IDL-compilers and interoperability of CORBA-based applications // Proceedings of the 1st international symposium on Information and communication technologies. Dublin, Ireland. Dublin: Trinity College, 2003. P. 512-517.

23. Aberer K., Hauswirth M. Peer-to-peer information systems: concepts and models, state-of-the-art, and future systems // Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering. New York: ACM, 2001. P. 326-327.

24. Alhaisoni M., Liotta A. Characterization of signaling and traffic in Joost // Peer-to-Peer Networking and Applications. 2008. Vol. 2, No. 1. P. 75-83.

25. Architectural Overview of OGSA-BES Adoption in UNICORE 6 // Forschungszentrum Jülich, 2008. 18 p.

26. Bastide Remi et al. Formal specification of CORBA services: experience and lessons learned // Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. New York: ACM. 2000. P. 105-117.

27. BitTorrent. URL: http://www.bittorrent.com/ (дата обращения: 19.05.2009).

28. BOINC - Berkeley Open Infrastructure for Network Computing. URL: http://boinc.berkeley.edu/ (дата обращения: 19.05.2009).

29. Brebner P., Emmerich W. Two Ways to Grid: The Contribution of Open Grid Services Architecture (OGSA) Mechanisms to Service-Centric and Resource-Centric Lifecycles // Journal of Grid Computing. Vol. 5, No. 1. 2006. P. 151-131.

30. Bucci G., Streeter D.N. A methodology for the design of distributed information systems // Communications of the ACM. Vol. 22, Issue 4. 1979. P. 233-245.

31. Business Process Execution Language for Web Services Version 1.1, URL: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel (дата обращения: 19.05.2009)

32. Buriola T.M., Scheer S. CAD and CAE Integration Through Scientific Visualization Techniques for Illumination Design // Tsinghua Science & Technology. Vol. 13, No. 1. 2008. P. 26-33.

33. Buyya R., Abramson D., Venugopal S. The Grid Economy. Proceedings of the IEEE. Vol. 93, Issue 3. 2005 P. 698-714.

34. Buyya R. et al. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility // Future Generation Computer Systems. 2009. № 25. P. 599-616.

35. Canevet C. et. al. Analysing UML 2.0 activity diagrams in the software performance engineering process // Proceedings of the 4th international workshop on Software and performance. New York: ACM, 2004. P. 74-78.

36. Chao K.-M., Younas M., Griffiths N. BPEL4WS-based coordination of Grid Services in design // Computers in Industry. Vol. 57. 2006. P. 778-786.

37. Cheng H.-C., Fen C.-S. A web-based distributed problem-solving environment for engineering applications // Advances in Engineering Software. Vol. 37, Issue 2. 2006. P. 112-128.

38. Chin R.S., Chanson S.T. Distributed, object-based programming systems // ACM Computing Surveys (CSUR). Vol. 23, Issue 1. 1991. P. 91-124.

39. Coatta T. Corba: Gone but (Hopefully) Not Forgotten // Queue. Vol. 5, Issue 4. 2007. P. 3.

40. Cols D.R. Business-structured client/server: an architecture for distributed applications // Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: software engineering. IBM Press, 1993. Vol. 1. P. 41-53.

41. Coulson G., Baichoo S. Implementing the CORBA GIOP in a high-performance object request broker environment // Distributed Computing. Vol. 14, No. 2. 2001. P. 113-126.

42. Credle R. et al. SOA Approach to Enterprise Integration for Product Lifecycle Management. IBM, 2008. 506 p.

43. Czajkowski K., Ferguson D., Foster I. et al. From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution. – The Globus Project Whitepaper. 2004. URL: http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf (дата обращения: 02.06.2009).

44. Dogac A., Dengi C., Oszu M.T. Distributed Object Computing Platforms // Communications of the ACM. Vol. 41, Issue 9. 1998. P. 95-103.

45. Domazet D. et al. An infrastructure for inter-organizational collaborative product development // Proceedings of the 33rd Annual Hawaii International Conference on System Sciences. 2000. 10 pp.

46. Ernemann C. et al. On advantages of grid computing for parallel job scheduling. // Proc. of the 2nd IEEE Int'l. Symp. on Cluster Computing and the Grid (CCGrid). Washington, DC: IEEE Computer Society, 2002. P 39-49.

47. Evaristo J.R., Desouza K.C., Hollister K. Centralization momentum: the pendulum swings back again // Communications of the ACM. Vol. 48, Issue 2. 2005. P. 66-71.

48. Fan L.Q. et al. Development of a distributed collaborative design framework within peer-to-peer environment. // Computer-Aided Design. Vol. 40. 2008. P. 891–904.

49. Ferguson D. Trends and statistics in peer-to-peer. Workshop on technical and legal aspects of peer-to-peer television. Amsterdam, Netherlands, 2006, Mar. 17. 2006. URL: http://www2.noticiasdot.com/publicaciones/2006/0406/1804 /noticias/images/CacheLogic_AmsterdamWorkshop_Presentation_v1.0.ppt (дата обращения: 02.06.2009).

50. Finlayson N., Morrison J. P2P and Client-Server Hybrids: Groove-enabling a J2EE portal using web services // Networking and Electronic Commerce Research Conference (NAEC 2005), October 6-9, 2005, Riva Del Garda, Italy. URL: http://www.atlanticshack.com/resources/P2P_and_Client-Server_NAEC2005_final.pdf (дата обращения: 15.05.2009).

51. Foster I.T., Kesselman C., Nick J., Tuecke S. The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration. / Global Grid Forum. 2002. URL: http://www.globus.org/ogsa/ (дата обращения: 14.05.2009).

52. Foster I.T. Service-Oriented Science // Science. 2005. Vol. 308, No. 5723. P. 814–817.

53. Foster I. et al. Grid Services for Distributed System Integration // Computer. Vol. 35, Issue 6. 2002. P. 37-46.

54. Foster I. et al. How Do I Model State? Let Me Count the Ways // Queue. Vol. 7, Issue 2. 2009. P. 54-64.

55. Foster I. et al. Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF // Proceedings of the IEEE. Vol. 93, Issue 3. 2005. P. 604-612.

56. Foster I. Globus Toolkit Version 4: Software for Service-Oriented Systems // IFIP International Conference on Network and Parallel Computing. Springer, 2005. P. 2-13.

57. Foster I., Frey J., Graham S. et al. Modeling Stateful Resources with Web Services / The Globus Project Whitepaper, 2004 URL: http://www.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf (дата обращения: 16.05.2009).

58. Foster I., Iamnitchi A. On death, taxes, and the convergence of peer-to-peer and grid computing // In 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03). Springer, 2003. P. 118-128.

59. Foster I., Kesselman C. Globus: A Metacomputing Infrastructure Toolkit // International Journal of Supercomputer Applications Vol. 11, Issue 2. 1997. P. 115-128.

60. Foster I., Kesselman C. The Grid. Blueprint for a new computing infrastructure. San Francisco: Morgan Kaufman, 1999. 677 p.

61. Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations // International Journal of Supercomputer Applications and High Performance Computing. 2001. Vol. 15, No 3. P. 200-222.

62. Foster I., Kishimoto H., Savva A. et al. The Open Grid Services Architecture. 2005. 62 p. URL: http://forge.gridforum.org/projects/ogsa-wg pdf (дата обращения: 16.05.2009).

63. Foster I., Jennings N.R., Kesselman C. Brain Meets Brawn: Why Grid and Agents Need Each Other // Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems. Washington, DC: IEEE Computer Society, 2004. Vol. 1. P. 8-15.

64. Foster I. What is the Grid? A Three Point Checklist. 2002. URL: http://www.fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf (дата обращения: 17.05.2009).

65. Fox G. C., Gannon D. Workflow in Grid Systems // Concurrency and Computation: Practice & Experience. Vol. 18. No. 10. 2006. P. 1009-1019.

66. Gnutella. URL: http://www.gnutella.com/ (дата обращения: 25.05.2009)

67. Guha S., Daswani N., Jain R. An experimental study of the Skype peer-to-peer VoIP system. In: Proceedings of the IPTPS'06. Santa Barbara, CA, Feb. URL: http://iptps06.cs.ucsb.edu/talks/guha-skype-talk.pdf (дата обращения: 25.05.2009).

68. Gray N.A.B. Comparison of Web Services, Java-RMI, and CORBA service implementations // Australian Software Engineering Conference (ASWEC 2004). Melbourne, Australia (April 13 and 14, 2004). Melburne: Swinburne Univercity Of Technology. 2004. P. 1-12.

69. Hayes B. Cloud computing // Communication of the ACM. Vol. 51. Issue 7. 2008. P. 9-11.

70. Haynos M. Perspectives on grid: Grid computing – Next-generation distributed computing / IBM Whitepaper. 2004. 11 p. URL: http://www-128.ibm.com/developerworks/grid/library/gr-heritage/ (дата обращения: 19.06.2009).

71. Henning M. The Rise and Fall of CORBA // ACM Queue. Vol. 4, Num. 5. 2006. P. 28-34.

72. Henning M, Vinoski S. Advanced CORBA programming with C++. New York: Addison-Wesley, 1999. 1120 p.

73. Hollingsworth J., Powell D. The Implementation of an Evolutionary-Based Engineering Optimization Framework for the Grid / Elon University. 2007. URL: http://www.cs.uncc.edu/~abw/ITCS4146S07/uncc.pdf (дата обращения: 19.05.2009).

74. Iosup A., Epema D. et al. On Grid Performance Evaluation using Synthetic Workloads // CoreGRID Technical Report Number TR-0039. 2006. 18 p.

75. Jagannathan V., Almasi G., Suvaiala A. Collaborative infrastructures using the WWW and CORBA-based environments // Proceedings of the IEEE Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WETICE'96). 1996. P. 292–297.

76. Jalonski S., Bussker C. Workflow management: modeling concepts, architecture and implementation. London: International Thomson Computer Press, 1996. 351 p.

77. Java RMI Tutorial. URL: http://java.sun.com/docs/books/tutorial/rmi/index.html (дата обращения: 02.06.2009).

78. Joncheere N., Deridder D., Van Der Straeten R., Jonckers V. A Framework for Advanced Modularization and Data Flow in Workflow Systems // Service-Oriented Computing – ICSOC 2008. Vol. 5364. 2008. P. 592-598.

79. Kamel M., Leue S. Formalization and validation of the General Inter-ORB Protocol (GIOP) using PROMELA and SPIN // International Journal on Software Tools for Technology Transfer (STTT). Vol. 2, No. 4. 2000. P. 394-409.

80. Kim J.-H. et al. A Problem Solving Environment Portal for Multidisciplinary Design Optimization. // Advances in Engineering Software. Vol. 40. Issue 8. 2009. P. 623-629.

81. King J.L. Centralized versus decentralized computing: organizational considerations and management options // ACM Computing Surveys. Vol. 15, Issue 4. 1983. P. 319-349.

82. Krüger M. Grid Computing & UNICORE. 2006. URL: http://www.be.itu.edu.tr/news/grid_atolye/UNICOREIntroduction.pdf (дата обращения: 16.05.2009).

83. Leach P., Mealling M., Salz R. A Universally Unique IDentifier (UUID) URN Namespace. RFC 4122. July 2005.

84. Lee B.-E., Suh S.-H. An architecture for ubiquitous product life cycle support system and its extension to machine tools with product data model // The International Journal of Advanced Manufacturing Technology. Vol. 1. 2009. -P. 15.

85. Lee H.-J., Lee J.-W., Lee J.-O. Development of Web services-based Multidisciplinary Design Optimization framework. // Advances in Engineering Software. Vol. 40. No. 3. 2009. P. 176-183.

86. Li J. On peer-to-peer (P2P) content delivery // Peer-to-Peer Networking and Applications. 2008. Vol. 1, No. 1. P. 45-63.

87. Li Z. et al. Architecture of collaborative design grid and its application based on LAN // Advances in Engineering Software. Vol. 38, № 2. 2007. P. 121-132.

88. Li Z. et al. Conception and implementation of a collaborative manufacturing grid // The International Journal of Advanced Manufacturing Technology. Vol. 34, № 11-12. 2007. P. 1224-1235.

89. Lutteroth C., Weber G. Efficient Use of GUIDs // Proceedings of the 2008 Ninth international Conference on Parallel and Distributed Computing, Applications and Technologies (December 01 - 04, 2008). Washington, DC: IEEE Computer Society, 2008. P.115-120.

90. Memon M. S. et al. Enhanced resource management capabilities using standardized job management and data access interfaces within UNICORE Grids // 13th International Conference on Parallel and Distributed Systems. Julich: Central Inst. of Appl. Math, 2007. Vol. 2. P. 1-6.

91. Mervyn F., Senthil Kumar A., Bok S.H., Nee A.Y.C. Developing distributed applications for integrated product and process design // Computer-Aided Design. Vol 36, Issue 8. 2004. P. 679–689.

92. Milojicic D. S. et al. Peer-to-Peer Computing, Hewlett-Packard, Tech. Rep. HPL-2002-57R1. 2003. URL: http:// www.hpl.hp.com/ techreports/ 2002/HPL-2002-57R1.html (дата обращения: 19.05.2009).

93. Milojicic D. Cloud computing: Interview with Russ Daniels and Franco Travostino // IEEE Internet Computing. Vol. 15, Issue 5. 2008. P. 7–9.

94. Munoz C., Zalewski J. Archictecture and Performance of Java-Based Distributed Object Models: CORBA vs RMI // Real-Time Systems. Vol. 21, No. 1-2. 2001. P. 43-75.

95. Nolan R.L. Managing the computer resource: a stage hypothesis // Communications of the ACM. Vol. 16, Issue 7. 1973. P. 399-405.

96. Object Management Group (OMG), The Common Object Request Broker Architecture (CORBA). URL: http://www.corba.org (дата обращения: 19.05.2009).

97. Olson M.H. Remote office work: changing work patterns in space and time // Communications of the ACM. Vol. 26, Issue 3. 1983. P. 182-187.

98. Papazoglou M.P., Georgakopoulos D. Service-Oriented Computing // Communications of the ACM. Vol. 46, No. 10. 2003. P. 25-28.

99. Peer-to-peer Working Group. URL: http://www.p2pwg.org (дата обращения: 19.05.2009).

100. Prodan R., Fahringer T. From Web Services to OGSA: Experiences in Implementing an OGSA-based Grid Application // Proceedings of the 4th International Workshop on Grid Computing. Washington, DC: IEEE Computer Society, 2003. P. 2-10.

101. Python Programming Language - Official Website. URL: www.python.org (дата обращения: 19.05.2009)

102. Raphael B., Smith I.F.C. Fundamentals of computer aided engineering. John Wiley, 2003. 306 p.

103. Riedel M., Mallmann. D. Standardization Processes of the UNICORE Grid System // Proceedings of 1st Austrian Grid Symposium 2005, Schloss Hagenberg, Austria. Austrian Computer Society, 2005. P: 191–203.

104. Russell N. et. al. On the suitability of UML 2.0 activity diagrams for business process modelling // Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling. Hobart: Australian Computer Society, Inc, 2006. Vol. 53. P. 95-104.

105. Schnekenburger T. Load Balancing in CORBA: A Survey of Concepts, Patterns, and Techniques // The Journal of Supercomputing. Vol. 15, No. 2. 2000. P. 141-161.

106. Service Oriented Architecture (SOA) Reference Model Public Review Draft 1.0 (Feb) / Organization for the Advancement of Structured Information Standards (OASIS), 2006. 28 p.

107. Shan H., Oliker L., Biswas R. Job superscheduler architecture and performance in computational grid environments // Proceedings of the ACM/IEEE conference on Supercomputing. Washington, DC: IEEE Computer Society, 2003. P. 44–54.

108. Silva V.T., Noya R.C., Lucena C.J.P. Using the UML 2.0 activity diagram to model agent plans and actions // Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. New York: ACM, 2005. P. 594-600.

109. Sinha A. Client-server computing // Communications of the ACM. Vol. 35, Issue 7. 1992. P. 77-98.

110. Skype. URL: http://www.skype.com (дата обращения: 02.06.2009).

111. Sotomayor B. The Globus Toolkit 4 Programmer's Tutorial / University of Chicago, Department of Computer Science, 2005. URL: http://gdp.globus.org/gt4-tutorial/download/progtutorial-pdf_0.2.1.tar.gz (дата обращения: 02.06.2009).

112. Stevens R. et. al. From the I-WAY to the National Technology Grid // Communications of the ACM. Vol. 40, Issue 11. 1997. P. 50-60.

113. Stockinger H. Defining the Grid: A Snapshot on the Current View // The Journal of Super-computing. 2007. № 42(1). P. 3-17.

114. A. Streit. UNICORE: Getting to the heart of Grid technologies // eStrategies. Vol. 3. 2009. P. 8-9.

115. A. Streit. UNICORE - What lies beneath Grid functionality? // eStrategies. Vol. 7. 2008. P. 38-39.

116. Sundaram B. Understanding WSRF, Part 1: Using WS-ResourceProperties. 2005. URL: http://www.ibm.com/developerworks/edu/gr-dw-gr-wsrf1-i.html (дата обращения: 02.06.2009).

117. Taylor I., Harrison A. From P2P and Grids to Services on the Web. Springer. 2008. 462 p.

118. Twenty-One Experts Define Cloud Computing. URL: http://cloudcomputing.sys-con.com/read/612375_p.htm (дата обращения: 18.05.2009).

119. Vaquero L. M. et al. A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review. Vol. 39, Issue 1. 2009. P. 50-55.

120. Vogels W. Web Services Are Not Distributed Objects // IEEE Internet Computing. Vol. 7, No. 6. 2003. P. 59-66.

121. Yin J.W., Zhang W.Y., Li Y., Chen H.W. A peer-to-peer-based multi-agent framework for decentralized grid workflow management in collaborative design. // The International Journal of Advanced Manufacturing Technology. Vol. 41, № 3-4. 2009. P. 407-420.

122. Wasson G., Humphrey M. Policy and Enforcement in Virtual Organizations // Proceedings of the 4th International Workshop on Grid Computing. Washington, DC: IEEE Computer Society, 2003. P. 125-133.

123. Weiss A. Computing in the Clouds // netWorker. № 11(4). 2007. P. 16-25.

124. Xue G., Song W., Cox S.J., Keane An. Numerical Optimisation as Grid Services for Engineering Design // Journal of Grid Computing. 2004. Vol. 2. P. 223-238.

125. Zalila B., Hugues J., Pautet L. An improved IDL compiler for optimizing CORBA applications // Proceedings of the 2006 annual ACM SIGAda international conference on Ada. Albuquerque, New Mexico, USA. New York: ACM, 2006. P. 21-28.