**Cодержание**

Введение 3

Обзор 4

Химия 4

История распределенных систем 5

Основные требования к распределенным системам 11

Открытость 11

Прозрачность 12

Масштабируемость 13

Классификация распределенных систем 14

Распределенные системы баз данных 14

Основные принципы 14

Типы распределённых баз данных 17

Распределенные системы компьютеров 17

Классификация распределенных систем компьютеров 18

Распределенные системы контроля версий 20

Распределенные операционные системы 21

Параллелизм 22

Параллелизм на уровне битов 23

Параллелизм на уровне инструкций 24

Параллелизм данных 24

Параллелизм задач 24

Распределенные системы управления 24

GRID 25

Концепция грид 26

Сравнение грид-систем и обычных суперкомпьютеров 27

Типы грид-систем 28

Требования к системе 29

Оболочка информационного наполнения 30

Редакторы 30

Система автоматического построения решателей класса вычислительных задач 31

Система поддержки библиотеки решателей задач разного класса 31

Библиотека решателей задач разного класса 31

Информационное наполнение 31

Архитектура системы 32

Порядок взаимодействия узлов 32

Архитектура узла “Лаборатория” 33

Архитектура системы “Институт химии” 34

Требования к редакторам информационных компонентов 37

Cписок литературы 39

**Введение**

Одним из классов программных систем являются системы, основанные на знаниях, отличительная особенность которых состоит в том, что знания, необходимые для выполнения профессиональной деятельности, отделены в этих системах от программ для решения прикладных задач.

В рамках диссертационной работы д.т.н., профессора кафедры ПО ЭВМ ДВФУ – Артемьевой Ирины Леонидовны «Многоуровневые модели сложно-структурированных предметных областей и их использования при разработке систем, основанных на знаниях» автором были разработаны теоретические положения и получено практическое решение проблемы создания расширяемых специализированных оболочек систем, основанных на знаниях, для сложно-структурированных предметных областей.

На настоящий момент имеется необходимость в создании распределенной системы, выполняющей те же функции, что и специализированная оболочка интеллектуальных систем для физической химии, а пользователями этой системы будут специалисты ПО «Физическая химия».

**Обзор**

**Цель обзора**

Целью данного обзора является рассмотрение типов и технологий организации распределенных вычислительных систем, а также поверхностный анализ предметной области «Физическая химия».

**Химия**

Физи́ческая хи́мия — наука об общих законах физики и химии. Исследует химические явления с помощью теоретических и экспериментальных методов химии и физики.

Разделы физической химии:

* Учение о строении вещества, свойствах молекул, ионов, радикалов, природе химической связи — В этот раздел входит учение о строении атомов и молекул и учение об агрегатных состояниях вещества. Учение о строении атома, относящееся в большей степени к физике, в курсах физической химии необходимо для выяснения вопросов образования молекул из атомов, природы химической связи.
* Химическая термодинамика — В этом разделе физической химии рассматриваются основные соотношения, вытекающие из первого закона термодинамики, которые позволяют рассчитать количество выделяемой или поглощаемой теплоты и определить, как будет влиять на него изменение внешних условий. На основе второго закона термодинамики определяется возможность самопроизвольного течения процесса, а также условия положения равновесия и его смещения под влиянием изменения внешних условий. Внутри данного раздела можно выделить несколько подразделов:
* Термодинамика газов
* Термодинамика растворов рассматривает природу растворов, их внутреннюю структуру и важнейшие свойства, зависимость свойств от концентрации и химической природы компонентов и вопросы растворимости.
* Термодинамика адсорбции
* Статистическая термодинамика позволяет получать термодинамические параметры системы исходя из строения компонентов системы и внешних условий.
* Химическая кинетика — изучает скорость химических реакций, её зависимость от внешних условий (температура, концентрации). Является одним из важнейших разделов химии, показывает какой именно продукт образуется в сложной системе
* Электрохимия изучает некоторые особенности свойств растворов электролитов, электропроводность растворов, процессы электролиза)
* Звукохимия (акустохимия) изучает химические процессы, протекающие при действии звуковых волн.
* Потенциометрия — междисциплинарная область физической химии, подразумевающая использование различных электрохимических и термодинамических методов, методов аналитической химии, — широко применяемых в научных исследованиях различной принадлежности, — в производственной практике; в том числе — ионометрии, pH-метрии, а также при создании измерительной аппаратуры используемой в них (см. также: периодический закон (окислительный потенциал), водородный показатель, редокс-потенциал, pH-метр, стеклянный электрод).

**История распределенных систем [4]**

По самым смелым оценкам, начало распределенным вычислениям было положено два столетия назад. Безусловно, ни о каких компьютерах тогда не могло быть и речи, но это было время, когда перед учеными уже ставили задачи крупномасштабных вычислений.

В конце 18-го века правительство Франции решило существенно улучшить логарифмические и тригонометрические таблицы. Работа была связана с огромными по тем временам объемами расчетов, а потому ее поручили профессионалу, барону Гаспару де Прони, состоявшему главным вычислителем при французском правительстве в 1790-1800 гг. В результате появилась знаменитая "вычислительная мануфактура" барона де Прони.

Барон смело взял на вооружение идею о разделении труда и перенес ее принципы на вычислительный процесс. Исполнители проекта были распределены на три уровня. Низший уровень в системе занимали обыкновенные люди-вычислители ("компьютеры", от которых требовалось только производить аккуратные арифметические действия. На втором уровне стояли образованные технологи ("серверы" проекта), которые организовывали рутинный процесс, распределяя задания и обрабатывая полученные вычислителями данные. Высшую ступень занимали выдающиеся французские математики ("администраторы-программисты" проекта), среди которых были Адриен Лежандр и Лазар Карно. Они готовили математическое обеспечение - по существу писали "программы". В результате барону де Прони удалось организовать процесс таким образом, чтобы свести очень сложные задачи к набору рутинных операций, благодаря четкой системе контроля и хорошо отлаженной системе распределения работы между вычислителями.

Идеи де Прони подтолкнули Чарльза Бэббиджа к созданию первой в истории вычислительной машины, а основанные на прототипе барона "вычислительные мануфактуры" использовались в исследовательских проектах вплоть до середины 20 века, с их помощью, в частности, рассчитывались первые ядерные бомбы в США, Великобритании и Советском Союзе.

В 1973 году Джон Шох и Джон Хапп из калифорнийского научно-исследовательского центра Xerox PARC написали программу, которая по ночам запускалась в локальную сеть PARC и заставляла работающие компьютеры выполнять вычисления .

В 1978 году советский математик Виктор Глушков работал над проблемой макроконвейерных распределённых вычислений. Он предложил ряд принципов распределения работы между процессорами. На базе этих принципов им была разработана ЭВМ ЕС-2701.

В 1988 году Арьен Ленстра и Марк Менес написали программу для факторизации длинных чисел. Для ускорения процесса программа могла запускаться на нескольких машинах, каждая из которых обрабатывала свой небольшой фрагмент.

В 1994 году Дэвидом Джиди была предложена идея по организации массового проекта распределённых вычислений, который использует компьютеры добровольцев (т. н. добровольные вычисления) — SETI@Home. Научный план проекта, который разработали Дэвид Джиди и Крейг Каснофф из Сиэтла был представен на пятой международной конференции по биоастрономии в июле 1996 года.

В январе 1996 года стартовал проект GIMPS по поиску простых чисел Мерсенна, также используя компьютеры простых пользователей как добровольную вычислительную сеть.

28 января 1997 года стартовал конкурс RSA Data Security на решение задачи взлома методом простого перебора 56-битного ключа шифрования информации RC5. Благодаря хорошей технической и организационной подготовке проект, организованный некоммерческим сообществом distributed.net, быстро получил широкую известность.

17 мая 1999 года стартовал SETI@home на базе Grid, а в начале 2002 года завершилась разработка Калифорнийского Университета в Беркли открытой платформы BOINC (Berkeley Open Infrastructure for Network Computing), разрабатываемой с апреля 2000 года первоначально для SETI@Home, но первым на платформе BOINC стал проект Predictor@home запущенный 9 июня 2004 года.

Распределенная система — это набор независимых компьютеров, представляющийся их пользователям единой объединенной системой.

Рассмотрим наиболее важные характеристики распределенных систем. Первая из таких характеристик состоит в том, что от пользователей скрыты различия между компьютерами и способы связи между ними. То же самое относится и к внешней организации распределенных систем. Другой важной характеристикой распределенных систем является способ, при помощи которого пользователи и приложения единообразно работают в распределенных системах, независимо от того, где и когда происходит их взаимодействие.

Распределенные системы должны также относительно легко поддаваться расширению, или масштабированию. Эта характеристика является прямым следствием наличия независимых компьютеров, но в то же время не указывает, каким образом эти компьютеры на самом деле объединяются в единую систему. Распределенные системы обычно существуют постоянно, однако некоторые их части могут временно выходить из строя. Пользователи и приложения не должны уведомляться о том, что эти части заменены или починены или что добавлены новые части для поддержки дополнительных пользователей или приложений.

Для того чтобы поддержать представление различных компьютеров и сетей в виде единой системы, организация распределенных систем часто включает в себя дополнительный уровень программного обеспечения, находящийся между верхним уровнем, на котором находятся пользователи и приложения, и нижним уровнем, состоящим из операционных систем, как показано на рис. 1. Соответственно, такая распределенная система обычно называется системой промежуточного уровня (middleware).

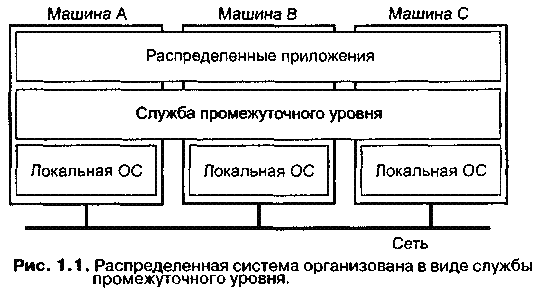


Рисунок 1 - Распределенная система организована в виде службы промежуточного уровня

Рассмотрим теперь некоторые примеры распределенных систем. В качестве первого примера рассмотрим сеть рабочих станций в университете или отделе компании. Вдобавок к персональной рабочей станции каждого из пользователей имеется пул процессоров машинного зала, не назначенных заранее ни одному из пользователей, но динамически выделяемых им при необходимости. Эта распределенная система может обладать единой файловой системой, в которой все файлы одинаково доступны со всех машин с использованием постоянного пути доступа. Кроме того, когда пользователь набирает команду, система может найти наилучшее место для выполнения запрашиваемого действия, возможно, на собственной рабочей станции пользователя, возможно, на простаивающей рабочей станции, принадлежащей кому-то другому, а может быть, и на одном из свободных процессоров машинного зала. Если система в целом выглядит и ведет себя как классическая однопроцессорная система с разделением времени (то есть многопользовательская), она считается распределенной системой. В качестве второго примера рассмотрим работу информационной системы, которая поддерживает автоматическую обработку заказов. Обычно подобные системы используются сотрудниками нескольких отделов, возможно в разных местах. Так, сотрудники отдела продаж могут быть разбросаны по обширному региону или даже по всей стране. Заказы передаются с переносных компьютеров, соединяемых с системой при помощи телефонной сети, а возможно, и при помощи сотовых телефонов. Приходящие заказы автоматически передаются в отдел планирования, превращаясь там во внутренние заказы на поставку, которые поступают в отдел доставки, и в заявки на оплату, поступающие в бухгалтерию. Система автоматически пересылает эти документы имеющимся на месте сотрудникам, отвечающим за их обработку. Пользователи остаются в полном неведении о том, как заказы на самом деле курсируют внутри системы, для них все это представляется так, будто вся работа происходит в централизованной базе данных.

В качестве последнего примера рассмотрим World Wide Web. Web предоставляет простую, целостную и единообразную модель распределенных документов. Чтобы увидеть документ, пользователю достаточно активизировать ссылку. После этого документ появляется на экране. В теории (но определенно не в текущей практике) нет необходимости знать, с какого сервера доставляется документ, достаточно лишь информации о том, где он расположен. Публикация документа очень проста: вы должны только задать ему уникальное имя в форме унифицированного указателя ресурса (Uniform Resource Locator, URL), которое ссылается на локальный файл с содержимым документа. Если бы Всемирная паутина представлялась своим пользователям гигантской централизованной системой документооборота, она также могла бы считаться распределенной системой. К сожалению, этот момент еще не наступил. Так, пользователи сознают, что документы находятся в различных местах и распределены по различным серверам.

Основные требования к распределенным системам:

* Открытость
* Прозрачность
* Масштабируемость

**Открытость**

Открытость – это использование синтаксических и семантических правил, основанных на стандартах. Для распределенной системы – это, прежде всего, использование формализованных протоколов. Службы, входящие в распределенную систему, определяются через интерфейсы, которые часто описываются при помощи языков описания интерфейсов IDL. Языки IDL касаются почти исключительно синтаксиса (имена доступных функций, типы параметров, возвращаемых значений, исключительные ситуации). Семантика чаще задается неформально, на естественном языке.

Если интерфейс описан правильно, возникает возможность правильной совместной работы одного произвольного процесса, нуждающегося в интерфейсе, с другим произвольным процессом, представляющим интерфейс. Один и тот же интерфейс может быть также реализован в разных распределенных системах (независимо друг от друга), но работать обе системы будут одинаково. Для обеспечения переносимости и способности к взаимодействию в интерфейсе должно быть все, что нужно для его реализации, но он не должен определять внешний вид реализации. Переносимость характеризует, насколько приложение, сделанное для одной распределенной системы, может работать в составе другой системы, а способность к взаимодействию показывает, насколько две реализации систем или компонентов, выполненных разными людьми, в состоянии работать совместно.

Открытые системы обладают очень важной характеристикой – гибкостью. Гибкость есть легкость конфигурирования системы, состоящей из различных компонентов. Достижения необходимого уровня гибкости приводит к тому, что открытая распределенная система становится расширяемой.

**Прозрачность**

Распределенная система должна скрывать разницу в способах представления данных и в способах доступа к ресурсам. Такое свойство распределенных систем называется прозрачностью доступа к данным.

Распределенная система должна обеспечивать прозрачность местоположения ресурса, то есть скрывать его физическое расположение. Важно, чтобы ресурсы имели только логические имена. Примером такого имени может служить универсальный указатель ресурса URL, в котором нет никакой информации о том, где находится файл, который ищется в Интернете.

Ресурс может время от времени менять свое расположение, и при следующем вызове может быть обнаружен в другом месте (но по тому же логическому адресу). Распределенная система, позволяющая ресурсам менять свое расположение от вызова к вызову, обладает свойством прозрачности смены местоположения ресурса (пример – система ICQ).

Иногда ресурсу было позволено менять свое положение непосредственно в процессе его использования (пример такого ресурса – мобильные пользователи с беспроводной связью, не отключающиеся от сети при переходе в другую зону обслуживания). Это более сильное свойство называется прозрачностью динамической смены местоположения ресурса.

Для балансировки использования ресурсов они могут быть реплицированы, то есть, размножены по нескольким физическим адресам. Прозрачность репликации скрывает это. Из наличия этого свойства сразу следует и прозрачность местоположения.

Часто совместное использование ресурсов достигается за счет действительно совместной работы и тесного взаимодействия пользователей системы. Однако пользователь распределенной системы не должен знать, что он является не единственным ее пользователем. Например, при работе с системой управления базой данных (СУБД) пославший запрос пользователь не должен знать, что одновременно СУБД получает и обрабатывает запросы многих других пользователей. Такое прозрачное параллельное использование должно быть непротиворечивым, для чего составляются специальные правила блокировок, когда пользователи поочередно получают исключительные права на ресурс. Этой же цели достигают с помощью отправки транзакций.

В распределенной системе должны быть приняты меры, чтобы часть аппаратуры брала на себя выполнение работы в случае выхода из строя другой части системы. Основная трудность достижения прозрачности поломки в том, чтобы отличить по-настоящему сломанные части от медленно работающих фрагментов системы.

Данные могут размещаться на различных физических носителях, в том числе таких, которые могут сохранять их значения в период выключения системы. Действия системы, обладающей свойством прозрачности сохранности данных, при обработке таких объектов должны быть скрыты от пользователя. Сохранность важна для любых информационных систем, не только распределенных.

Существуют ситуации, когда полностью скрыть распределенность (то есть достичь абсолютной прозрачности) не удается. При сильной удаленности узлов системы друга от друга возникают заметные задержки передачи информации. Существует проблема часовых поясов. Существует связь между прозрачностью и производительностью распределенной системы. Необходимо соблюдать баланс этих системных свойств.

**Масштабируемость**

Распределенные системы программного обеспечения обладают свойством масштабируемости, которая может проявляться по отношению к размеру, к географическому положению, к административному устройству систем. Достижение масштабируемости связано с решением проблем, возникающих из-за наличия узких мест по обслуживанию (один сервер для множества клиентов), данным (множественный доступ к одному файлу данных), и алгоритмам (перегрузка коммуникаций из-за использования централизованных алгоритмов).

Требование масштабируемости часто является препятствием распространения систем, реализованных для локальных сетей, на уровень сетей глобальных (корпоративных или Интернета). В глобальных сетях время получения ответа может значительно превышать локальные задержки, поэтому там чаще используется асинхронная связь. Кроме того, в локальных сетях службы часто распределены по компьютерам фиксированно, а в глобальных сетях местоположение необходимой службы заранее неизвестно.

Еще одно следствие масштабируемости: аппаратные решения для распределенных систем могут быть гетерогенными. В отличие от гомогенных систем, построенных на единой технологии, гетерогенные системы могут состоять из частей, построенных на разных физических принципах, и обладающих разными свойствами и характеристиками.

Разнородность в сочетании с независимостью приводят к важнейшему свойству распределенных систем: сами системы могут существовать продолжительное время, но отдельные их части могут время от времени отключаться. При этом пользователи и приложения не должны уведомляться о том, что эти части вновь подключены, что добавлены новые части для поддержки дополнительных пользователей или приложений.

**Классификация распределенных систем [4]**

* Распределенные системы баз данных
* Распределенные системы компьютеров
* Распределенные системы контроля версий
* Распределенные файловые системы
* Распределенные операционные системы
* Распределенные системы управления
* Распределенные системы документов
* GRID

Основная задача распределенных систем программного обеспечения – облегчить их пользователям доступ к удаленным ресурсам, а также контролировать совместное использование этих ресурсов. Ресурсы могут быть виртуальными, но могут быть и традиционными – компьютерами, принтерами, устройствами хранения файлов, файлами и данными.

**Распределенные системы баз данных [2]**

Распределённые базы данных (РБД) — совокупность логически взаимосвязанных баз данных, распределённых в компьютерной сети.

**Основные принципы [8]**

РБД состоит из набора узлов, связанных коммуникационной сетью, в которой:

* каждый узел — это полноценная СУБД сама по себе;
* узлы взаимодействуют между собой таким образом, что пользователь любого из них может получить доступ к любым данным в сети так, как будто они находятся на его собственном узле.

Каждый узел сам по себе является системой базы данных. Любой пользователь может выполнить операции над данными на своём локальном узле точно так же, как если бы этот узел вовсе не входил в распределённую систему. Распределённую систему баз данных можно рассматривать как партнёрство между отдельными локальными СУБД на отдельных локальных узлах.

Фундаментальный принцип создания распределённых баз данных («правило 0»): Для пользователя распределённая система должна выглядеть так же, как нераспределённая система.

Фундаментальный принцип имеет следствием определённые дополнительные правила или цели. Таких целей всего двенадцать:

1. Локальная независимость. Узлы в распределённой системе должны быть независимы, или автономны. Локальная независимость означает, что все операции на узле контролируются этим узлом.
2. Отсутствие опоры на центральный узел. Локальная независимость предполагает, что все узлы в распределённой системе должны рассматриваться как равные. Поэтому не должно быть никаких обращений к «центральному» или «главному» узлу с целью получения некоторого централизованного сервиса.
3. Непрерывное функционирование. Распределённые системы должны предоставлять более высокую степень надёжности и доступности.
4. Независимость от расположения. Пользователи не должны знать, где именно данные хранятся физически и должны поступать так, как если бы все данные хранились на их собственном локальном узле.
5. Независимость от фрагментации. Система поддерживает независимость от фрагментации, если данная переменная-отношение может быть разделена на части или фрагменты при организации её физического хранения. В этом случае данные могут храниться в том месте, где они чаще всего используются, что позволяет достичь локализации большинства операций и уменьшения сетевого трафика.
6. Независимость от репликации. Система поддерживает репликацию данных, если данная хранимая переменная-отношение — или в общем случае данный фрагмент данной хранимой переменной-отношения — может быть представлена несколькими отдельными копиями или репликами, которые хранятся на нескольких отдельных узлах.
7. Обработка распределённых запросов. Суть в том, что для запроса может потребоваться обращение к нескольким узлам. В такой системе может быть много возможных способов пересылки данных, позволяющих выполнить рассматриваемый запрос.
8. Управление распределёнными транзакциями. Существует 2 главных аспекта управления транзакциями: управление восстановлением и управление параллельностью обработки. Что касается управления восстановлением, то чтобы обеспечить атомарность транзакции в распределённой среде, система должна гарантировать, что все множество относящихся к данной транзакции агентов (агент — процесс, который выполняется для данной транзакции на отдельном узле) или зафиксировало свои результаты, или выполнило откат. Что касается управления параллельностью, то оно в большинстве распределённых систем базируется на механизме блокирования, точно так, как и в нераспределённых системах.
9. Аппаратная независимость. Желательно иметь возможность запускать одну и ту же СУБД на различных аппаратных платформах и, более того, добиться, чтобы различные машины участвовали в работе распределённой системы как равноправные партнёры.
10. Независимость от операционной системы. Возможность функционирования СУБД под различными операционными системами.
11. Независимость от сети. Возможность поддерживать много принципиально различных узлов, отличающихся оборудованием и операционными системами, а также ряд типов различных коммуникационных сетей.
12. Независимость от типа СУБД. Необходимо, чтобы экземпляры СУБД на различных узлах все вместе поддерживали один и тот же интерфейс, и совсем необязательно, чтобы это были копии одной и той же версии СУБД.

**Типы распределённых баз данных [2]**

1. Распределённые базы данных
2. Мультибазы данных с глобальной схемой. Система мультибаз данных — это распределённая система, которая служит внешним интерфейсом для доступа ко множеству локальных СУБД или структурируется, как глобальный уровень над локальными СУБД.
3. Федеративные базы данных. В отличие от мультибаз не располагают глобальной схемой, к которой обращаются все приложения. Вместо этого поддерживается локальная схема импорта-экспорта данных. На каждом узле поддерживается частичная глобальная схема, описывающая информацию тех удалённых источников, данные с которых необходимы для функционирования.
4. Мультибазы с общим языком доступа — распределённые среды управления с технологией «клиент-сервер»

**Распределенные системы компьютеров [7]**

Распределенная система компьютеров (компьютерная сеть, вычислительная сеть, сеть передачи данных) — система связи компьютеров или компьютерного оборудования (серверы, маршрутизаторы и другое оборудование). Для передачи информации могут быть использованы различные физические явления, как правило — различные виды электрических сигналов, световых сигналов или электромагнитного излучения.

**Классификация распределенных систем компьютеров [11]**

***По территориальной распространенности***

* PAN (Personal Area Network) — персональная сеть, предназначенная для взаимодействия различных устройств, принадлежащих одному владельцу.
* LAN (Local Area Network) — локальные сети, имеющие замкнутую инфраструктуру до выхода на поставщиков услуг. Термин «LAN» может описывать и маленькую офисную сеть, и сеть уровня большого завода, занимающего несколько сотен гектаров. Зарубежные источники дают даже близкую оценку — около шести миль (10 км) в радиусе. Локальные сети являются сетями закрытого типа, доступ к ним разрешен только ограниченному кругу пользователей, для которых работа в такой сети непосредственно связана с их профессиональной деятельностью.
* CAN (Campus Area Network — кампусная сеть) — объединяет локальные сети близко расположенных зданий.
* MAN (Metropolitan Area Network) — городские сети между учреждениями в пределах одного или нескольких городов, связывающие много локальных вычислительных сетей.
* WAN (Wide Area Network) — глобальная сеть, покрывающая большие географические регионы, включающие в себя как локальные сети, так и прочие телекоммуникационные сети и устройства. Пример WAN — сети с коммутацией пакетов (Frame relay), через которую могут «разговаривать» между собой различные компьютерные сети. Глобальные сети являются открытыми и ориентированы на обслуживание любых пользователей.
* Термин «корпоративная сеть» также используется в литературе для обозначения объединения нескольких сетей, каждая из которых может быть построена на различных технических, программных и информационных принципах.

***По типу функционального взаимодействия***

* Клиент-сервер
* Смешанная сеть
* Одноранговая сеть
* Многоранговые сети

***По типу сетевой топологии***

* Шина
* Кольцо
* Двойное кольцо
* Звезда
* Ячеистая
* Решётка
* Дерево
* Fat Tree

***По типу среды передачи***

* Проводные (телефонный провод, коаксиальный кабель, витая пара, волоконно-оптический кабель)
* Беспроводные (передачей информации по радиоволнам в определенном частотном диапазоне)

***По функциональному назначению***

* Сети хранения данных
* Серверные фермы
* Сети управления процессом
* Сети SOHO, домовые сети

***По скорости передачи***

* низкоскоростные (до 10 Мбит/с),
* среднескоростные (до 100 Мбит/с),
* высокоскоростные (свыше 100 Мбит/с);

***По сетевым операционным системам***

* На основе Windows
* На основе UNIX
* На основе NetWare
* На основе Cisco

***По необходимости поддержания постоянного соединения***

* Пакетная сеть, например Фидонет и UUCP
* Онлайновая сеть, например Интернет и GSM

**Распределенные системы контроля версий [6]**

## Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

## Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. В частности, системы управления версиями применяются в САПР, обычно в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools).

Также известны как англ. Distributed Version Control System, DVCS. Такие системы используют распределённую модель вместо традиционной клиент-серверной. Они, в общем случае, не нуждаются в централизованном хранилище: вся история изменения документов хранится на каждом компьютере, в локальном хранилище, и при необходимости отдельные фрагменты истории локального хранилища синхронизируются с аналогичным хранилищем на другом компьютере. В некоторых таких системах локальное хранилище располагается непосредственно в каталогах рабочей копии.

Когда пользователь такой системы выполняет обычные действия, такие как извлечение определённой версии документа, создание новой версии и тому подобное, он работает со своей локальной копией хранилища. По мере внесения изменений, хранилища, принадлежащие разным разработчикам, начинают различаться, и возникает необходимость в их синхронизации. Такая синхронизация может осуществляться с помощью обмена патчами или так называемыми наборами изменений (англ. change sets) между пользователями.

**Распределенные операционные системы [3]**

Распределенные операционные системы (параллельные вычислительные системы) — это физические компьютерные, а также программные системы, реализующие тем или иным способом параллельную обработку данных на многих вычислительных узлах.

Например, для быстрой сортировки массива на двухпроцессорной машине можно разделить массив пополам и сортировать каждую половину на отдельном процессоре. Сортировка каждой половины может занять разное время, поэтому необходима синхронизация.

Распределённая ОС, динамически и автоматически распределяя работы по различным машинам системы для обработки, заставляет набор сетевых машин обрабатывать информацию параллельно. Пользователь распределённой ОС, вообще говоря, не имеет сведений о том, на какой машине выполняется его работа.

Распределённая ОС существует как единая операционная система в масштабах вычислительной системы. Каждый компьютер сети, работающей под управлением распределённой ОС, выполняет часть функций этой глобальной ОС. Распределённая ОС объединяет все компьютеры сети в том смысле, что они работают в тесной кооперации друг с другом для эффективного использования всех ресурсов компьютерной сети.

**Параллелизм**

Идея распараллеливания вычислений основана на том, что большинство задач может быть разделено на набор меньших задач, которые могут быть решены одновременно. Обычно параллельные вычисления требуют координации действий. Параллельные вычисления существуют в нескольких формах: параллелизм на уровне битов, параллелизм на уровне инструкций, параллелизм данных, параллелизм задач. Параллельные вычисления использовались много лет в основном в высокопроизводительных вычислениях, но в последнее время к ним возрос интерес вследствие существования физических ограничений на рост тактовой частоты процессоров. Параллельные вычисления стали доминирующей парадигмой в архитектуре компьютеров, в основном в форме многоядерных процессоров.

Писать программы для параллельных систем сложнее, чем для последовательных, так как конкуренция за ресурсы представляет новый класс потенциальных ошибок в программном обеспечении (багов), среди которых состояние гонки является самой распространённой. Взаимодействие и синхронизация между процессами представляют большой барьер для получения высокой производительности параллельных систем. В последние годы также стали рассматривать вопрос о потреблении электроэнергии параллельными компьютерами. Характер увеличения скорости программы в результате распараллеливания объясняется законом Амдала.

***Типы параллелизма***

* Параллелизм на уровне битов
* Параллелизм на уровне инструкций
* Параллелизм данных
* Параллелизм задач
* Распределённые операционные системы

**Параллелизм на уровне битов**

Эта форма параллелизма основана на увеличении размера машинного слова. Увеличение размера машинного слова уменьшает количество операций, необходимых процессору для выполнения действий над переменными, чей размер превышает размер машинного слова. К примеру: на 8-битном процессоре нужно сложить два 16-битных целых числа. Для этого вначале нужно сложить нижние 8 бит чисел, затем сложить верхние 8 бит и к результату их сложения прибавить значение флага переноса. Итого 3 инструкции. С 16-битным процессором можно выполнить эту операцию одной инструкцией.

Исторически 4-битные микропроцессоры были заменены 8-битными, затем появились 16-битные и 32-битные. 32-битные процессоры долгое время были стандартом в повседневных вычислениях. С появлением технологии x86-64 для этих целей стали использовать 64-битные процессоры.

**Параллелизм на уровне инструкций**

Компьютерная программа — это, по существу, поток инструкций, выполняемых процессором. Но можно изменить порядок этих инструкций, распределить их по группам, которые будут выполняться параллельно, без изменения результата работы всей программы. Данный приём известен как параллелизм на уровне инструкций. Продвижения в развитии параллелизма на уровне инструкций в архитектуре компьютеров происходили с середины 1980-х до середины 1990-х.

Современные процессоры имеют многоступенчатый конвейер команд. Каждой ступени конвейера соответствует определённое действие, выполняемое процессором в этой инструкции на этом этапе. Процессор с N ступенями конвейера может иметь одновременно до N различных инструкций на разном уровне законченности. Классический пример процессора с конвейером — это RISC-процессор с 5-ю ступенями: выборка инструкции из памяти (IF), декодирование инструкции (ID), выполнение инструкции (EX), доступ к памяти (MEM), запись результата в регистры (WB). Процессор Pentium 4 имеет 35-тиступенчатый конвейер.

Некоторые процессоры, дополнительно к использованию конвейеров, обладают возможностью выполнять несколько инструкций одновременно, что даёт дополнительный параллелизм на уровне инструкций. Возможна реализация данного метода при помощи суперскалярности, когда инструкции могут быть сгруппированы вместе для параллельного выполнения (если в них нет зависимости между данными). Также возможны реализации с использованием явного параллелизма на уровне инструкций: VLIW и EPIC.

**Параллелизм данных**

Основная идея подхода, основанного на параллелизме данных, заключается в том, что одна операция выполняется сразу над всеми элементами массива данных. Различные фрагменты такого массива обрабатываются на векторном процессоре или на разных процессорах параллельной машины. Распределением данных между процессорами занимается программа. Векторизация или распараллеливание в этом случае чаще всего выполняется уже на этапе компиляции — перевода исходного текста программы в машинные команды. Роль программиста в этом случае обычно сводится к заданию настроек векторной или параллельной оптимизации компилятору, директив параллельной компиляции, использованию специализированных языков для параллельных вычислений.

**Параллелизм задач**

Стиль программирования, основанный на параллелизме задач, подразумевает, что вычислительная задача разбивается на несколько относительно самостоятельных подзадач и каждый процессор загружается своей собственной подзадачей.

**Распределенные системы управления** [13]

Распределённая система управления (англ. Distributed Control System, DCS) — система управления технологическим процессом, отличающаяся построением распределённой системы ввода-вывода и децентрализацией обработки данных.

РСУ применяются для управления непрерывными и гибридными технологическими процессами (хотя сфера применения РСУ только этим не ограничена). К непрерывным процессам можно отнести те, которые должны проходить днями и ночами, месяцами и даже годами, при этом остановка процесса, даже на кратковременный период, может привести к порче изготавливаемой продукции, поломке технологического оборудования и даже несчастным случаям. Классическим примером непрерывного процесса является изготовление стекла в стекловаренной печи.

Сферы применения РСУ многочисленны:

1. Химия и нефтехимия.
2. Нефтепереработка и нефтедобыча.
3. Стекольная промышленность.
4. Пищевая промышленность: молочная, сахарная, пивная.
5. Газодобыча и газопереработка.
6. Металлургия.
7. Энергоснабжение и т. д.

Требования к современной РСУ:

1. Отказоустойчивость и безопасность.
2. Простота разработки и конфигурирования.
3. Поддержка территориально распределённой архитектуры.
4. Единая конфигурационная база данных.
5. Развитый человеко-машинный интерфейс.

**GRID**

Грид-вычисления (англ. grid — решётка, сеть) — это форма распределённых вычислений, в которой «виртуальный суперкомпьютер» представлен в виде кластеров соединённых с помощью сети, слабосвязанных, гетерогенных компьютеров, работающих вместе для выполнения огромного количества заданий (операций, работ). Эта технология применяется для решения научных, математических задач, требующих значительных вычислительных ресурсов. Грид-вычисления используются также в коммерческой инфраструктуре для решения таких трудоёмких задач, как экономическое прогнозирование, сейсмоанализ, разработка и изучение свойств новых лекарств.

Грид с точки зрения сетевой организации представляет собой согласованную, открытую и стандартизованную среду, которая обеспечивает гибкое, безопасное, скоординированное разделение вычислительных ресурсов и ресурсов хранения информации, которые являются частью этой среды, в рамках одной виртуальной организации.

**Концепция грид**

Грид является географически распределённой инфраструктурой, объединяющей множество ресурсов разных типов (процессоры, долговременная и оперативная память, хранилища и базы данных, сети), доступ к которым пользователь может получить из любой точки, независимо от места их расположения.

Идея грид-компьютинга возникла вместе с распространением персональных компьютеров, развитием интернета и технологий пакетной передачи данных на основе оптического волокна (SONET, SDH и ATM), а также технологий локальных сетей (Gigabit Ethernet). Полоса пропускания коммуникационных средств стала достаточной, чтобы при необходимости привлечь ресурсы другого компьютера. Учитывая, что множество подключенных к глобальной сети компьютеров большую часть рабочего времени простаивает и располагает ресурсами, большими, чем необходимо для решения их повседневных задач, возникает возможность применить их неиспользуемые ресурсы в другом месте.

**Сравнение грид-систем и обычных суперкомпьютеров [11]**

Распределённые или грид-вычисления в целом являются разновидностью параллельных вычислений, которое основывается на обычных компьютерах (со стандартными процессорами, устройствами хранения данных, блоками питания и т. д.) подключенных к сети (локальной или глобальной) при помощи обычных протоколов, например Ethernet. В то время как обычный суперкомпьютер содержит множество процессоров, подключенных к локальной высокоскоростной шине.

Основным преимуществом распределённых вычислений является то, что отдельная ячейка вычислительной системы может быть приобретена как обычный неспециализированный компьютер. Таким образом можно получить практически те же вычислительные мощности, что и на обычных суперкомпьютерах, но с гораздо меньшей стоимостью.

**Типы грид-систем**

В настоящее время выделяют три основных типа грид-систем:

1. Добровольные гриды — гриды на основе использования добровольно предоставляемого свободного ресурса персональных компьютеров;
2. Научные гриды — хорошо распараллеливаемые приложения программируются специальным образом (например, с использованием Globus Toolkit);
3. Гриды на основе выделения вычислительных ресурсов по требованию (коммерческий грид, англ. enterprise grid) — обычные коммерческие приложения работают на виртуальном компьютере, который, в свою очередь, состоит из нескольких физических компьютеров, объединённых с помощью грид-технологий.

Требования к системе

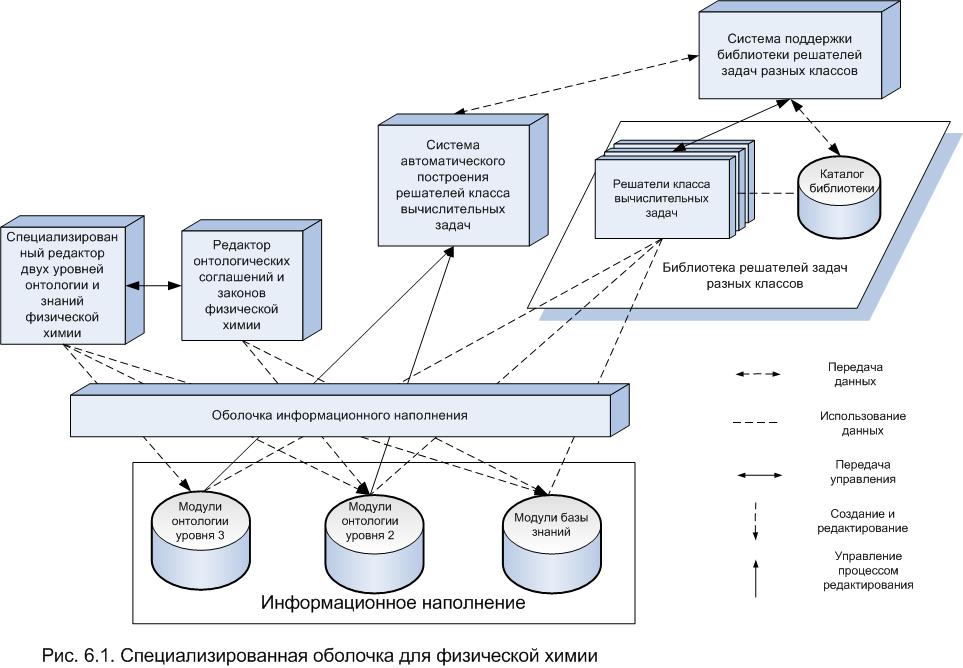


Рисунок 2 – Архитектура специализированной оболочки систем для физической химии

Исходя из архитектуры специализированной оболочки систем для физической химии (рис. 2) распределенная система должна содержать следующие функциональные модули:

1. Оболочка информационного наполнения
2. Редакторы

* Редактор онтологий и знаний физической химии
* Редактор онтологиеских соглашений и законов физической химии

1. Система автоматического построения решателей класса вычислительных задач
2. Система поддержки библиотеки решателей задач разных классов
3. Библиотека решателей задач разных классов
4. Информационное наполнение

* Модули онтологий
* Модули базы знаний

На основе рассмотренных технологий организации распределнных систем реализация системы опирается на технологию GRID.

Распределенная система должна работать на нескольких машинах, объединенных в сеть (проводную или беспроводную). Компьютеры могут быть гетерогенными, т. е. иметь разные операционные системы (Windows, Linux).

Основными модулями, которые требуют распределения между машинами являютcя библиотеки решателей и информационное наполнение, т.к. они являются наиболее требовательными к ресурсам.

Остальные функциональные системы являются одинаковыми на всех машинах системы, т.о. не требуют распределения между машинами.

Узлы системы должны иметь возможность предоставлять доступ к своему информационному наполнению и библиотекам решателей для других узлов системы. Каждый узел системы, если он имеет доступ к информационному наполнению или библиотекам решателей другого узла, должен иметь возможность взаимодействия с ними как со своими.

Оболочка информационного наполнения

Этот модуль должен присутствовать на каждом узле системы, т.к. он позволяет взаимодействовать с другими модулями системы и отвечает за ввод, вывод информации и решений.

Редакторы

Эти модули должны присутствовать на каждом узле системы , использующей распределенные данные, т.к. они отвечают за редактирование информационного наполнениния узла, на котором они находятся.

Система автоматического построения решателей класса вычислительных задач

Этот модуль должен присутствовать на каждом узле системы , т.к. получает данные из модулей информационного наполнения, строит решатели и записывает их в библиотеку решателей.

Система поддержки библиотеки решателей задач разного класса

Этот модуль должен присутствовать на каждом узле системы, т.к. работает с решателями, добавляя построенные решатели в библиотеку решателей или извлекая их из библиотеки для вывода пользователю.

Библиотека решателей задач разного класса

Этот модуль хранит в себе множество построенных решателей. Он должен быть распределен между узлами, и обладать следующими свойствами:

1. Модуль хранится на узле, создавшем решатель.
2. Недоступен для других узлов, если локальный узел не разрешил доступ к модулю.
3. Пользователь должен видеть решатели, расположенные на других узлах так, как будто они расположены на его собственном узле.

Информационное наполнение

Этот модуль хранит в себе онтологии и знания. Он должен быть распределен между узлами, и обладать следующими свойствами:

1. Информационное наполнение хранится на узле, создавшем его.
2. Недоступен для других узлов, если локальный узел не разрешил доступ к модулю.
3. Пользователь должен видеть информационное наполнение, расположенные на других узлах так, как будто они расположены на его собственном узле

**Архитектура системы**

Рисунок 3 – модель системы

Каждый узел обладает своей базой знаний.

Главный узел – “Институт химии” хранит данные об узлах “Лабораториях” необходимые для их идентификации и осуществления связи между ними.

**Порядок взаимодействия узлов**:

1. Узел «Лаборатория» подключается к базе знаний узла «Институт химии»;
2. Узел «Лаборатория» посылает запрос на получение данных о других узлах системы узлу «Институт химии»;
3. Узел «Институт химии» возвращает данные для поключения к остальным узлам системы;
4. Узел «Лаборатория» осуществляет подключения ко всем узлам, данные для подключения к которым были предоставдены узлом «Институт химии»

**Архитектура узла «Лаборатория»**

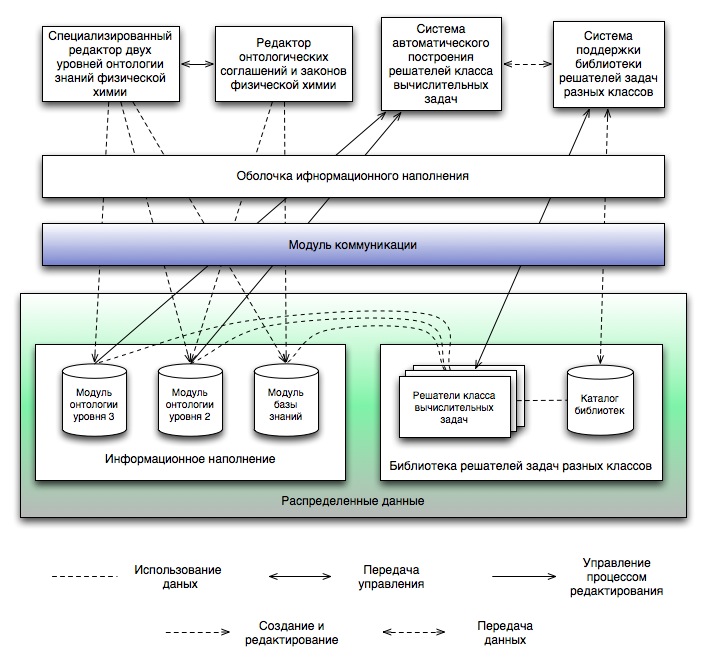


Рисунок 4 – архитектура узла «Лаборатория»

Модуль коммуникации отвечает за установление подключения к распределенным компонентам других узлов системы.

**Архитектура узла «Институт химии»**

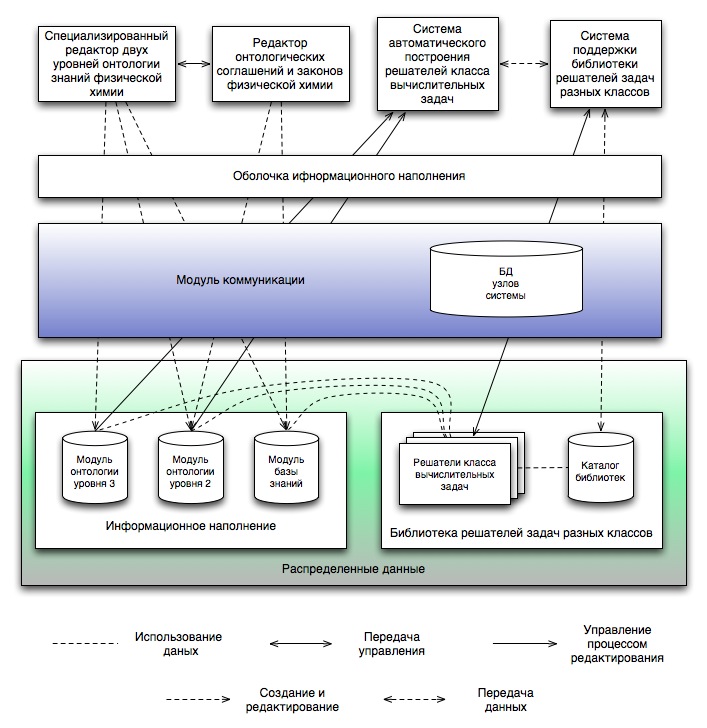
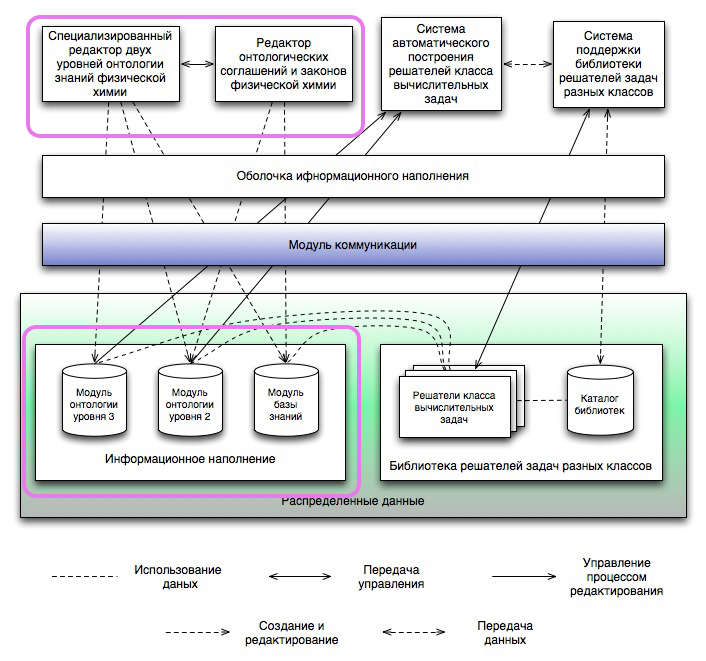


Рисунок 4 – архитектура узла «Лаборатория»

Модуль коммуникации данного типа узла содердит в себе БД, хранящую ифнормацию обо всех узлах системы.

Модуль коммуникации узла «Лаборатория» посылает запрос модулю коммуникации узла «Институт химии». В ответ он получает данные для подключения к другим узлам системы. Затем он подключается к каждому из узлов системы (к модулю коммуникации), данные о которых были переданы ему. После осуществления подключения узел «Лаборатория» может использовать распределяемые данные других узлов системы.



На рисунке (выше) приведена та часть системы, которую планируется реализовать в рамках дипломной работы.

А именно:

* Специализтированный редактор двух уровней онтологий знаний физической и органической химии
* Редактор онтологических соотношений и законов физической и органической химии
* Модуль онтологии уровня 3
* Модуль онтологии уровня 2
* Модуль базы данных

**Модель предметной области «Химия»**

Ниже приведена структура четырех уровневой модели химии, охватывающая физическую и органическоую химию в объеме университетских курсов.

Модель была построена на основе анализа данной предметной области при помощи метода «снизу-вверх»[1] и представлена как четырехуровневая необогащенная система логических соотношений с параметрами.

Третий уровень модел ей физической и органической химии построены с использованием метода "сверху вниз"[1], т. е. посредством задания обогащений для модели онтологии четвертого уровня. Модели онтологий второго уровня представлены средствами языка прикладной логики.

**Модель онтологии четвертого уровня**

Модель онтологии четвертого уровня представляет собой необогащенную систему логических соотношений с параметрами O4 = <Онтология четвертого уровня(ST, Интервалы, Категории), {Типы сущностей, Типы компонентов сущности, Типы компонентов сущности, задаваемых количеством, Типы сущностей процесса, Подмножества компонентов сущности}, Определение конструкторов>. Прикладная логическая теория Онтология четвертого уровня(SТ, Интервалы, Категории) = <Ø, SS>. Данная прикладная логическая теория не использует предложений других прикладных логических теорий .

**Модель онтологии третьего уровня**

Онтология третьего уровня для химии есть композиция онтологий третьего уровня для физической и органической химии.

**Модель онтологии третьего уровня для физической химии**

Модель онтологии третьего уровня для физической химии представляет собой нееобогащенную систему логических соотношений без параметров Оф3 = <*Онтология третьего уровня для физической химии*(ST, Интервалы, Категории), Ø, Определение конструкторов для физической химии>. Прикладная логическая теория *Онтология третьего уровня для физической химии*(ST, Интервалы, Категории) = <{*Онтология четвертого уровня, Константы онтологии физической химии*}, SS>.

Модель онтологии третьего уровня для физической химии получается из модели четвертого уровня с помощью обогащения kф3 = <3начение параметров четвертого уровня для физической химии, Определение сортов имен третьего уровня для физической химии, Множество онтологических соглашений для физической химии, Определение конструкторов для физической химии, Ø>.

**Модель онтологии третьего уровня для органической химии**

Модель онтологии третьего уровня для органической химии представляет собой нееобогащенную систему логических соотношений без параметров Оф3 = <*Онтология третьего уровня для органической химии*(ST, Интервалы, Категории), Ø, Определение конструкторов для органической химии>. Прикладная логическая теория *Онтология третьего уровня для органической химии*(ST, Интервалы, Категории) = <{*Онтология четвертого уровня, Константы онтологии органической химии*}, SS>.

Модель онтологии третьего уровня для органической химии получается из модели четвертого уровня с помощью обогащения kф3 = <3начение параметров четвертого уровня для органической химии, Определение сортов имен третьего уровня для органической химии, Множество онтологических соглашений для органической химии, Определение конструкторов для органической химии, Ø>.

**Модель онтологии второго уровня**

Онтология второго уровня (и модель) состоит из двух онтологий второго уровня: для физической и органической химии. Модели онтологий второго уровня получены посредством задания обогащений для модели онтологии третьего уровня.

Онтология второго уровня для физической химии и ее модель состоят из 9 связанных друг с другом модулей. Модулями онтологии являются: «Свойства химических элементов», «Свойства химических веществ», «Свойства химических реакций», «Основы термодинамики», «Термодинамика. Химические свойства», «Термодинамика. Физические свойства», «Термодинамика. Связь физических и химических свойств», «Химическая кинетика». Все модули используют вспомогательный модуль «Константы онтологии физической химии». Модули 1-3 определяют термины, используемые при описании свойств элементов, веществ и реакций. В модуле 4 определяются термины, используемые при описании общих свойств термодинамических систем и их компонентов. Состояния термодинамической системы могут изменяться в ходе физико-химического процесса. Состояния процесса задаются в дискретные моменты наблюдения. В модуле 5 определены термины, используемые при описании фазовых превращений веществе в ходе процесса, без учета химических превращений. В модуле 6 определены термины, используемые при описании химических превращений веществ в ходе процесса без учета фазовых превращений. И, наконец, в модуле 7 определены термины, используемые при описании физико-химических процессов. В модуле 8 определены термины, описывающие динамику прохождения процессов.

Для физической химии самостоятельное значение имеют еще 4 онтологии первого уровня, состоящие из следующих разделов (название онтологии соотвтствует названию последнего раздела в каждой строке ):

1. "Элементы", "Вещества", "Реакции", "Термодинамика. Химиче ские свойства";

2. "Элементы", "Вещества", "Основы термодинамики";

3. "Элементы", "Вещества", "Основы термодинамики", "Термодинамика. Физические свойства";

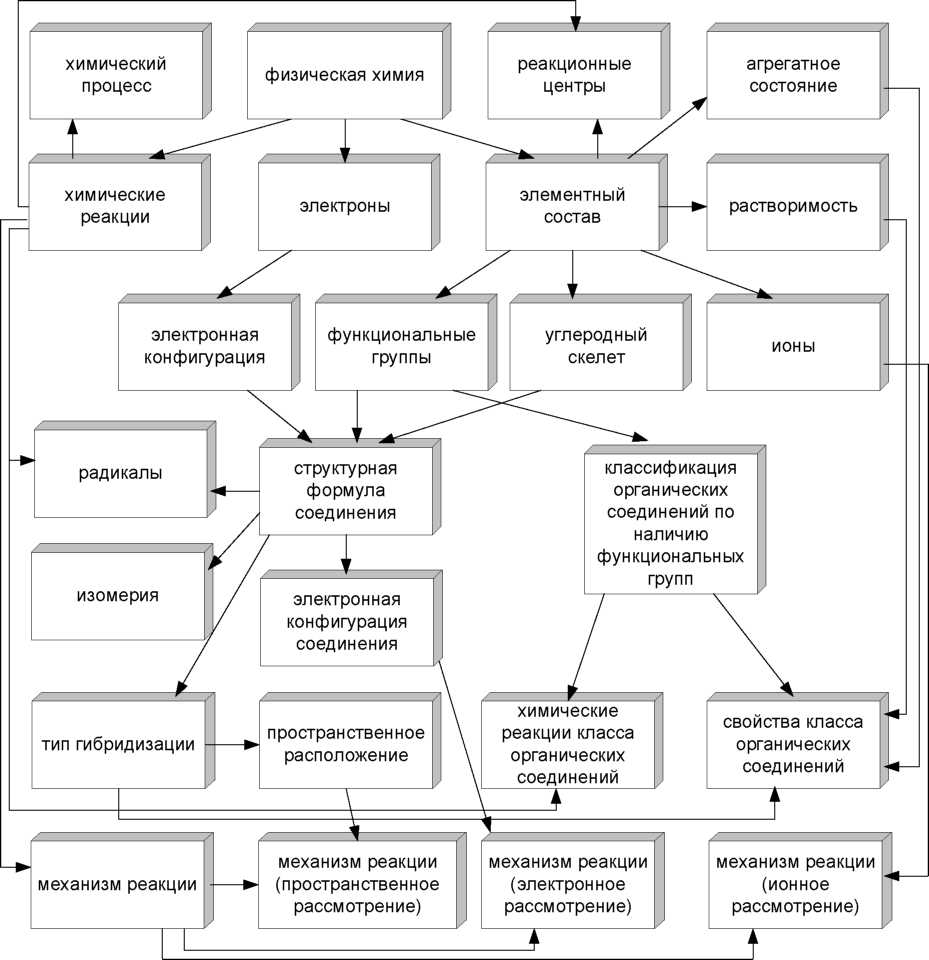
4. "Элементы", "Вещества", "Реакции, "Термодинамика. Химические свойства", "Химическая кинетика".

Каждая из этих онтологий является упрощением онтологии второго уровня для физической химии. Все указанные онтологии второго уровня получаются из общей онтологии второго уровня отбрасыванием терминов и соглашений невходящих в нее разделов.

Структура модульной онтологии уровня 2 для физической химии представлена на схеме ниже.



Онтология второго уровня и ее модель для органической химии состоит из 26 связанных друг с другом модулей. На рисунке ниже представлены 24 модуля. Все модули используют модуль "Константы онтологии". В состав онтологии и модели входит вспомогательный модуль "Номенклатура", непредставленный в работе и на рисунке. Три модуля «Электроны», «Элементный состав» и «Химические реакции» используют терминологию онтологии физической химии и определяют термины для задания электронного строения химических элементов, элементного состава органических соединений и свойств химических реакций, соответственно. Модули «Агрегатное состояние», «Растворимость», «Углеродный скелет», «Функциональные группы», «Ионы» и «Реакционные центры» используют терминологию модуля «Элементный состав». Модуль «Реакционны центры» также использует терминологию модуля «Химические реакции». Перечисленные модули описывают терминологию, используемую при задании агрегатного состояния органических соединений, их растворимости в различных растворителях, представлеения их углеродного скелета, различных функциональных групп, ионов и реакционных центров соединений в химических реакциях. Модуль «Электронная конфигурация» использует терминологию модуля «Электроны» и описывает термины для задания электронной конфигурации химических элементов. Модуль «Структурная формула соединения» использует терминологию модулей «Электронная конфигурация», «Функциональные группы» и «Углеродный скелет» и определяет термины для описания структурных формул органических соединений. Его терминологию используют модули «Номенклатура», «Тип гибридизации», «Изомерия», «Электронная конфигурация соединения», «Радикалы» и «Связь». Модуль «Связь» также использует терминологию модуля «Ионы», а модуль «Радикалы» - модуля «Химические реакции». Модуль «Пространственно расположение » использует терминологию модуля «Тип гибридизации» и определяет терминологию для задания положения молекулы соединения в пространстве. Модуль «Классификация органических соединений по наличию функциональных групп» использует терминологию модуля «Функциональны группы» и описывает термины для задания классов органических соединений. Модули «Радикалы» и «Реакционные центры» используют также терминологию модуля «Химические реакции». Модуль «Химический процесс» определяет терминологию для задания свойств химических процессов. Модуль «Химические реакции класса органических соединений» использует терминологию модуля «Классификация органических соединений по наличию функциональных групп» и описывает термины для задания свойств химических реакций для классов органических соединений. Последний модуль «Свойства класса органических соединений» использует терминологию модулей «Классификация органических соединений по наличию функциональных групп», «Тип гибридизации», «Агрегатное состояние» и «Растворимость» и описывает свойства классов со единений.



**Пользователи системы**

Можно выделить следующие группы пользователей специализированной оболочки:

* сопровождающий программист;
* инженер знаний;
* эксперт;
* специалист предметной области.

Сопровождающий программист обеспечивает развитие программных компонентов, инженер знаний и эксперт - развитие информационных компонентов, а специалист предметной области использует готовые программны и информационны компон нты при р ш нии своих прикладных задач.

**Редакторы информационных компонентов**

Для редактирования информационных компонентов в состав оболочки входят: специализированный редактор многоуровневой модульной онтологии, специализированны редакторы знаний, управляемые онтологиями уровня 2, системы ввода/вывода значений нестандартных величин, редактор утверждений, позволяющий задавать законы и онтологические соглашения предметной области.

Специализированный редактор многоуровневой онтологии предназначен для использования инженерами знаний. Он обеспечивает возможность создания и редактирования нового модуля онтологии уровня i под управлением модуля онтологии уровня i+1, а также повторное использование существующих модулей онтологии каждого уровня.

Специализированные редакторы знаний, управляемые онтологией, предназначены для использования экспертами сложно-структурированной области. Они обеспечивают возможность создания и редактирования нового модуля знаний под управл ни м модуля онтологии уровня 2. Редакторы отличаются способом интерпретации онтологии уровня 2. Создание модуля знаний состоит в задании значений терминов онтологии уровня 2, а также законов предметной области в виде утверждений. С использованием системы поддержки библиотеки редакторов знаний эксп ерты выбирают требуемый им р едактор знаний для раздела (подраздела) сложно-структурированной предметной области. Редактор законов и онтологических соглашений предназначен для использования инженером знаний и экспертом предметной области. Он обеспечивает возможность создания и редактирования законов и онтологических соглашений предметной области в виде утверждений некоторого логического языка.

Системы для ввода/вывода значений нестандартных величин используются эксп ртами сложно-структурированной области при создании модуля знаний. Они позволяют использование графической нотации при задании значений терминов онтологии уровня 2. При вводе заданное графические значение нестандартной величины преобразуется в вербально представление в соответствии со специализированной онтологией нестандартной величины. При выводе вербальное представление преобразуется к графическому.

Система ввода/вывода значений нестандартной величины вызывается автоматически редактором знаний, если при определении термина онтологии уровня 2 использовано названи е не стандартной величины. Выбор требуемой системы ввода/вывода производится с помощью системы поддержки библиотеки подпрограмм для работы с нестандартными величинами. Каталог данной библиотеки содержит название нестандартной величины (оно используется при определении терминов онтологии уровня 2 редактором онтологий), а также информацию о сопоставленной этой величине системе ввода/вывода.

**Требования к редакторам информационных компонентов**

*Требование 1*: специализированная оболочка для сложно-структурированной предметной области должна позволять создание и изменение информационных компонентов, которые включают многоуровневую онтологию и знания предметной области.

Данное требование является следствием того, что сложно-структурированная предметная область, особенно связанная с наукой, постоянно развивается, причем такое развитие предполагает изменение не только знаний, но и онтологии. Изменение предметной области предполагает также появление новых разделов и, как следствие, новых онтологий разных уровней (соответствующих разделам и подразделам области).

*Требование 2*: редакторы многоуровневых онтологии и знаний должны позволять создание и редактирование модульных онтологий и знаний, а также обеспечивать возможность повторного использования модулей при создании онтологий и знаний новых разделов и подразделов области.

Для реальных предметных областей большими являются и база знаний, и онтология, что требует разбиения их на повторно используемые модули. Каждый модуль соответствует одному разделу (подразделу) области. База знаний также состоит из модулей. Разные разделы предметной области могут использовать онтологии и знания других разделов.

*Требование 3.1*: для сложно-структурированных предметных областей процесс создания и редактирования онтологии уровня i-1 должен управляться онтологией уровня i, а процесс создания знаний - онтологией уровня 2.

*Требование 3.2*: при создании нового модуля онтологии уровня i-1 должна обеспечиваться возможность выбора того из существующих модулей онтологии уровня i, который управляет процессом редактирования создаваемого модуля. Аналогично при редактировании модуля знаний должна обеспечиваться возможность выбора «управляющего» модуля онтологии уровня 2.

Создание и изменение онтологии уровня i-1 всегда выполняется в терминах некоторой онтологии уровня i, т.е. редактор онтологии интерпретирует онтологию уровня i при создании и изменении онтологии уровня i-1. Создание и изменение знаний всегда выполняется в терминах некоторой онтологии уровня 2, т. е. редактор знаний интерпретирует онтологию уровня 2 при создании и изменении базы знаний.

*Требование 4*: специализированная оболочка должна обеспечивать возможность задания структурированной и неструктурированной части онтологии, а также структурированной и неструктурированной части знаний в виде утверждений на некотором логическом языке.

Как отмечено в главе 1, онтологии и знания предметных областей могут быть структурированными или нет. Это требует обеспечения возможности задания нетолько терминов онтологии (при редактировании онтологии) и их значений (при редактировании онтологий и знаний), но также и неструктурированной части онтологических соглашений и знаний в вид утверждений.

*Требование 5*:оболочка должна обеспечивать возможность ввода/вывода значений нестандартных величин при редактировании знаний.

Наличие нестандартных величин является важным свойством предметных областей (в том числ сложно-структурированных). Значения, принадлежащие нестандартным величинам, используются при задании значений терминов, используемых при задании структурированной части знаний.

*Требование 6.1*: оболочка должна обеспечивать возможность использования принятого в предметной области графического способа представления значений нестандартных величин при создании и редактировании знаний.

*Требование 6.2*: оболочка должна обеспечивать автоматический выбор (управляемый онтологией) средств для графического представления значений нестандартных величин при редактировании знаний.

Для значений нестандартных величин в предметной области может существовать способ их графического представления. Например, для химии графически может быть задана краткая структурная формула или структурная формула органического соединения. Для косметологии графически может быть задана форма глаз или форма лица. Используемая при редактировании знаний нестандартная величина задается онтологией уровня 2. Если при определнии онтологии уровня 2 выбрана нестандартная величина, для представления значений которой в предметной области существует способ графического представления, то должна использоваться специализированная система ввода таких значений.

*Требование 7.1*: оболочка должна позволять использование редакторов, поддерживающих разные способы интерпретации модуля онтологии уровня 2.

*Требование 7.2*: оболочка должна предоставлять возможность эксперту выбора требуемого ему редактора знаний.

Как сказано выше, при создании знаний оболочка интерпретирует некоторую онтологию уровня 2. Редактор знаний специализированной оболочки должен позволять интерпретацию любой онтологии уровня 2, созданной с помощью оболочки, т.е. позволять интерпретацию класса онтологий уровня 2.

Вместе с тем, одна и та же онтология может интерпретироваться разными способами в разных редакторах знаний. Редакторы знаний могут отличаться не только способом интерпретации знаний, но и интерфейсом. Очевидно, что более удобный интерфейс и более понятный эксперту способ интерпретации можно обеспечить для редактора, предназначенного для интерпретации одной онтологии, а не класса онтологий.

# **Список литературы**

1. И.Л. Артемьева, Многоуровневые модели сложно-структурированных предметных областей и их использование при разработке систем, основанных на знаниях, 2008 г., - 478 стр.
2. К. Дж. Дейт , Введение в системы баз данных = An Introduction to Database Systems. — 7-е изд. - «Вильямс» , 2001
3. В. А. Крюков «Операционные системы распределённых вычислительных систем (распределённые ОС)». Лекции для 4 курса факультета ВМК МГУ
4. Эндрю Таненбаум, Мартин ван Стеен Распределенные системы. Принципы и парадигмы = Andrew S. Tanenbaum, Maarten van Steen. "Destributed systems. Principles and paradigms". — Санкт-Петербург: Питер, 2003. — 877 с.
5. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб: БХВ-Петербург, 2002. — 608 с.
6. Проблемы вычислений в распределенной среде. Распределенные приложения, коммуникационные системы, математические модели и оптимизация: Афанасьев А.П. — Москва, КомКнига, 2007 г.- 224 с.
7. Сетевые распределенные вычисления: достижения и проблемы: Макс К. Гофф — Санкт-Петербург, КУДИЦ-Образ, 2005 г.- 320 с.
8. Коннолли, Т., Бегг, К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание.: Пер. с англ. —  М.: Издательский дом «Вильяме», 2003. – 433 с.
9. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. – 2-е изд. – М.: Финансы и статистика, 1989. – 350 с.
10. Дейт К. Дж. Введение в системы баз данных.: Пер. с англ. – 6-е изд. – Киев: Диалектика, 1998. – 784 с.
11. Сетевые распределенные вычисления: достижения и проблемы: Макс К. Гофф — Москва, КУДИЦ-Образ, 2005 г.- 320 с.
12. Интернет и распределенные многоагентные системы: Р. Э. Асратян, В. Н. Лебедев, Р. И. Дмитриев — Москва, Ленанд, 2007 г.- 72 с.
13. Введение в распределенные алгоритмы: Ж. Тель — Санкт-Петербург, МЦНМО, 2009 г.- 616 с.