СИСТЕМНАЯ ПОДДЕРЖКА РЕШЕНИЯ БИЗНЕС-ЗАДАЧ В ГЛОБАЛЬНОЙ ИНФОРМАЦИОННОЙ СЕТИ[3]

Е. М. Лаврищева, Л. Е. Карпов, А. Н. Томилин

Е. М. Лаврищева <lavr@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
Московский физико-технический институт (технический университет),
141700, Московская область, г. Долгопрудный, Институтский пер., 9.
Л. Е. Карпов <mak@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
Московский государственный университет им. М. В. Ломоносова,
119991, Москва, ГСП-1, Ленинские горы, д. 1.
А. Н. Томилин <tom11@bk.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
Московский государственный университет им. М. В. Ломоносова,
119991, Москва, ГСП-1, Ленинские горы, д. 1.

Аннотация. Рассмотрены сервисы как системная поддержка процессов разработки программных систем и сервисы Интернет (или веб-сервисы) для решения бизнес-задач и приложений в среде Интернет. Дано описание набора системных сервисов CORBA для управления объектами и интерфейсами (stub и skeleton) между клиентом и сервером. Представлены архитектуры сервисов SOA и SCA, используемые для композиции систем из сервисов и компонентов в Интернет среде. Приведены языки описания сетевых сервисов (XML, SOAP, BPMN, WSDL, BPEL и др.) консорциума W3C и протоколы Icontract WCF для передачи сообщений между распределенными системами Интернет. Рассмотрены сервисные протоколы, и технологии обработка Web-сервисов в Grid-системе, обеспечивающей поддержку научного сервиса в глобальной сети e-sciences.

Ключевые слова: сервис, системный сервис, веб-сервис, модели сервисных архитектур, распределенная система; сетевая служба; протоколы взаимодействия; языки описания сервисов, бизнес задачи и приложения.

1. ВВЕДЕНИЕ

Системная поддержка процесса программирования задач началась с создания отдельных наборов последовательностей фрагментов программ, которые полезны всем пользователям вычислительной машины и относятся к службам управления обработкой прикладных программ, занесения их в

[3] Работа поддержана грантами Российского фонда фундаментальных исследований № 14-07-00606 и № 15-07-02355.

долговременную память, в стандартные библиотеки, а также предоставления их многим пользователям. Постепенно сформировалось понятие системного сервиса (или "службы"), которое со временем приобрело стандартное представление, содержащее информацию о реализованной системной функции, форме ее применения при решении разных вычислительных задач и бизнес приложений. Одним из наиболее известных средств обеспечения системного сервиса по управлению процессом разработки объектных программ и систем является система CORBA (http://www.corba.org). Она предоставляет сервисы по управлению разработкой систем на разных языках программирования, обеспечивает взаимодействие разных систем при решении некоторой задачи и передачи данных через интерфейсы IDL, обрабатываемые брокером объектных запросов ORB. Система находит применение как самостоятельно, так и внутри других общих систем [1-6].

В связи с распространением сети Интернет появилось огромное количество других видов сервисов, которыми управляют Web сервера. В e-sciences возник научный проект, под названием Grid (2000 г.) с целью создания компьютерной инфраструктуры нового типа, обеспечивающей глобальную интеграцию информационных и вычислительных ресурсов, специального программного обеспечения и набора стандартизованных служб для обеспечения совместного доступа к географически распределенным ресурсам. Разработан принципиально новый подход к обработке огромных объемов экспериментальных данных, к моделированию сложнейших физических процессов и созданию бизнес-приложений с большими объемами вычислений. Grid включает набор систем по реализации разных научных задач. Одной из систем является ETICS. Для построения сетевых приложений (Web Application) в ней представлены сетевые службы (Web Services), доступные глобальным пользователям [6-9].

За последние 10 лет комитет W3C разработал набор стандартов (протоколов) и языков описания программ, процессов и технологий для решения разного рода задач и бизнес-приложений в среде Интернет [10-12].

Далее рассматриваются системные и прикладные сервисы, которые представляются в рамках систем CORBA, W3C и Grid.

2. СИСТЕМНЫЕ СЕРВИСЫ

Системные сервисы обеспечивают решение разного рода прикладных, деловых и бизнес-задач. К ним относятся:

- общие сервисы системных сред для поддержки процессов и функций обработки программ и данных (например, службы именования, каталогизации и др.);
- объектные сервисы, которые управляют объектами, классами и услугами по формированию и обработке объектно-ориентированных

систем (например, службы диспетчеризации объектными запросами, управления интерфейсом и др.);
- сетевые сервисы стандартной модели OSI, моделей SOA (Service-oriented Architecture), SCA (Service-Component Architecture), как инструменты представления и обработки ресурсов в сети Интернет, которые реализуют деловые, финансовые, экономические и другие услуги при решении соответствующих задач;
- готовые программные и информационные ресурсы (services, artifacts, reuses, assets и др.), используемые как многоразовые услуги при решении разных задач в e-science и других прикладных областях.

Некоторые из сервисов стали обязательной частью общесистемных средств (VS.Net, IBM, Intel, Linux и др.), другие используются в специальных областях (например, медицина, биология) в плане предоставления услуг при работе с современными данными (FDT, GDT, Big Data).

Каждая служба определяется именем, по которому осуществляется поиск в распределенной среде пространства имен через транзакции, устанавливающие соответствие “имя-объект” для организации и управления отдельными сервисными ресурсами глобальной сети, а также с помощью сообщений для визуального общения с требуемыми представителями отдельных ресурсов.

Перечисленные виды сервисов используются при моделировании программных систем (ПС) из готовых ресурсов сети Интернет. Для их использования при создании ПС требуется проводить поиск подходящего сервисного ресурса, его апробацию и встраивание в прикладную программу решения задачи, либо использовать его в динамическом режиме (см. [5, 6-10]).

2.1. Системные сервисы CORBA

Полный набор сервисов для объектов создан в системе CORBA (см. 1-6]). В ней впервые реализована объектная модель и системные сервисы для работы с объектами:

- брокер объектных запросов (*Object Request Broker* — ORB), обеспечивающий взаимодействие объектов в разных языках программирования;
- общие объектные сервисы (*Common Object Services* — COS), обеспечивающие сервис всем объектам по вопросам управления изменениями, реализациями, контролем,
- транзакциями, подпроцессами и т.п.;
- общие средства обслуживания (*Common Facilities* — CF) или общие услуги, предоставляющие ряд общих прикладных функций, которые могут объединяться в различные конфигурации с учетом заданных требований (например, средства печати, БД, электронная почта и др.);

- объектные приложения (*Application Objects* — AO), к которым относятся приложения и их компоненты, реализующие задачи и объекты пользователя, функционирующие в объектной среде, и над которыми могут производиться операции типа - открыть, инсталлировать, переместить и поместить.

Общие объектные сервисы и общие средства обслуживания способствуют разбиению приложения на функции — базовые для большинства приложений, либо достаточно общие для широкого класса приложений. Объектные приложения и общие средства обслуживания обеспечивают функции и сервисы объектными интерфейсами, а также позволяют адаптироваться к новым поколениям сетей, языков и сред.

Система CORBA ориентирована на обеспечение взаимодействия удаленных объектов в распределенной среде типа OpenStep. Для задания взаимодействия объектов используется язык интерфейсов IDL (Interface Definition Languages) объектов. Интерфейсы в IDL запоминаются в репозитории интерфейсов (*Interface Repository*), а реализации объектов — в репозитории реализаций (*Implementation Repository*). Независимость интерфейсов от реализаций объектов позволяет использовать их разными приложениями статически и динамически.

Объект-клиент и объект-сервер обмениваются между собой с помощью запросов, каждый из которых исполняется брокером ORB с помощью компонентов, создаваемых на основе описания интерфейсов клиента, сервера и ядра ORB.

Интерфейс клиента (*Client Interface*) обеспечивает взаимодействие с объектом-сервером и состоит из трех базовых интерфейсов:

- stub-интерфейса, содержащего описание внешне видимых параметров и операций объекта в IDL;
- интерфейс динамического вызова (Dynamic Invocation Interface — DII) объекта, определяемого во время выполнения программы клиента посредством поиска описания интерфейса в репозитории интерфейсов или в репозитории реализаций;
- интерфейса сервисов ORB (ORB Services Interface), содержащего набор сервисных функций, которые клиент запрашивает у сервера через брокер ORB.

Stub-интерфейс обеспечивает взаимосвязь клиента с ORB. Прикладная программа клиента через посредника (заместителя) *stub* — статической части программы клиента, посылает в запросе параметры, которым сопоставляются соответствующие описания из репозитория интерфейсов. Брокер ORB выполняет полученный запрос и пересылает результаты клиенту.

Рассмотренная схема взаимодействия клиента с объектом соответствует схеме вызова удаленных процедур через RPC-механизм.

Интефейс DII обеспечивает доступ объектов и их интерфейсов во время выполнения. Этот интерфейс предоставляет механизм запроса к объектам, интерфейс которых становится известным во время выполнения. Он становится доступным с помощью вызовов брокера ORB. В каждом вызове указывается тип объекта, тип запроса и параметры. Эту информацию посылает прикладная программа, но она может извлекаться из репозитория интерфейсов или репозитория реализаций.

Объекты специфицируются средствами языков программирования (см. [1, 4, 5]) и могут быть реализованы на разных платформах и в разных средах. Интерфейсы программ-посредников описываются на языке IDL. Заместитель клиента (stub) выполняет сервисные функции, связанные с преобразованием типов данных клиентских компонентов к стандартным системным типам, а заместитель сервера (skeleton) преобразует стандартное представление данных в типы данных сервера.

Описание интерфейса в IDL начинается с ключевого слова **interface**, за которым следует: имя интерфейса, описание типов параметров и операций (op_dcl) вызова объектов:

interface A { ... }
interface B { ... }
interface C: B, A { ... }.

Параметры операций (op_dcl) в задании интерфейсов это:

1) тип данных (type_dcl);
2) константа (const_dcl);
3) название исключительной ситуация (except_dcl), которая может возникнуть в процессе выполнения метода объекта;
4) атрибуты параметров (attr_dcl).

Описание типов данных (ТД) начинается ключевым словом typedef, за которым следует базовый или конструируемый тип и его идентификатор. В качестве константы может быть некоторое значение типа данного или выражение, составленное из констант. ТД и константы описываются как фундаментальные типы данных: integer, boolean, string, float, char и др.

Описание операций op_dcl передачи данных включает в себя:

1) наименование операции интерфейса;
2) список параметров (от нуля и более);
3) типы аргументов и результатов, иначе – *void*;
4) управляющий параметр или описание исключительной ситуации и др.

Атрибуты передаваемых параметров начинаются служебными словами: **in** – при отсылке параметра от клиента к серверу; **out** – при отправке параметров-результатов от сервера к клиенту; **inout** – при передаче параметров в оба направления (от клиента к серверу и обратно).

Описание интерфейса для одного объекта может наследоваться другим объектом и тогда это описание становится базовым, например:

```
const long l=2
interface A { void f (in float s [l]); }
interface B { const long l=3; }
interface C: B, A { }.
```

Интерфейс ***C*** использует интерфейс ***B*** и ***A***. Это означает, что интерфейс ***C*** наследует описание типов данных ***A*** и ***B***, которые по отношению к ***C*** являются внешними. При этом синтаксис и семантика остаются неизменными. Согласно приведенному примеру - операция функции *f ( )* в интерфейсе ***C*** наследуется из ***A***.

Механизм наследования интерфейса состоит в сохранении имен объектов без их переопределения. Это касается описания операций, которые должны иметь уникальные обозначения. Имена операций могут использоваться динамически во время выполнения интерфейса *Skeleton.*

Общая структура описания модуля с интерфейсом в языке IDL имеет вид:

```
Request Operations
module CORBA {
        interface Request {
                Status add-arg (
                        in Identifier name,
                        in Flags arg_flags
                );
                Status invoke (
                        in Flags invoke_flags // invocation flags
                );
                Status send (
                        Status get_response (
                                out Flags response_flags // response flags
                        );
                );
};
```

Тип данных описывается в классе FDT, GDT, которые передаются через параметры операторов RPC, RMI, а также протоколами в WCF VS.Net, либо еще какими-нибудь. ТД описывается на объектно-ориентированном языке программирования (C#, VBasic, Pascal, и др.).

Входные и выходные интерфейсы (например) для программ P_1, P_2 (рис.1) имеют разную семантику, но одинаковое синтаксическое описание на некотором языке программирования. Передача данных от этих программ для P_3 осуществляется через функции $F_1(\ldots)$, $F_2(\ldots)$ и интерфейсы ***In***, ***Out***, с помощью которых осуществляется преобразование ТД переданных между P_1, P_3 и P_2, P_3 туда и обратно.

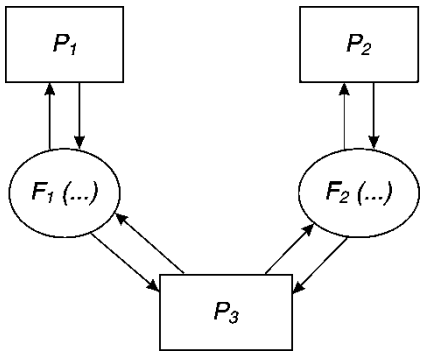

Рис. 1. Схема вызова функции объектов

Данный аппарат интерфейса реализован в системах CORBA, ОС IBM, Microsoft и др. Его основу составляют библиотеки VS.Net (CRL, CTS, FCL, CIL и др.) преобразования ТД, которые применяется при интеграции разнородных программных объектов.

Общие объектные службы предоставляют набор операций для работы с разными категориями объектных приложений:

- наименование и поиск объектов по именам, либо по свойствам и атрибутам;
- управление жизненным циклом объектов;
- параллельного обращения к объектам;
- определение очередей запросов к объектам;
- иерархия транзакций;
- защиту объектов от несанкционированного доступа (авторизация и аутентификация клиентов и др.).

Интерфейсные средства IDL используются для обеспечения взаимодействия распределенных систем прикладного, коммерческого и бизнес типа, а также входят в состав общесистемных систем (VS.Net, Java и др.),

3. СЕТЕВЫЕ ТЕХНОЛОГИЙ ИНТЕРНЕТ. СТАНДАРТЫ W3C

Сеть Интернет базируется на стеке протоколов TCP/IP, за которые отвечает международная некоммерческая организация ISOC (Internet Society).

ISOC насчитывает более чем 20 тысяч представителей, свыше 100 организаций из 180 стран мира и предоставляет организационную основу для других консультативных и исследовательских групп в том числе:

IETF открытое международное сообщество ученых, инженеров, провайдеров которое занимается развитием протоколов и архитектуры Интернет;

ICANN (Corporation for Assigned Names and Numbers) — международная некоммерческая организация, которая координирует предоставление имен и адресов в Интернете.

Структура TCP/IP базируется на четырехуровневой модели сетевого взаимодействия, которое разработано Министерством обороны США и в основном отвечает 7 уровневой модели OSI.

В Интернете используется общий язык для передачи данных глобальной информационной среды, стандартные методики и форматы представления данных .и обмен протоколами. На рис.2 приведены организации, генерирующие сетевые Интернет технологии.

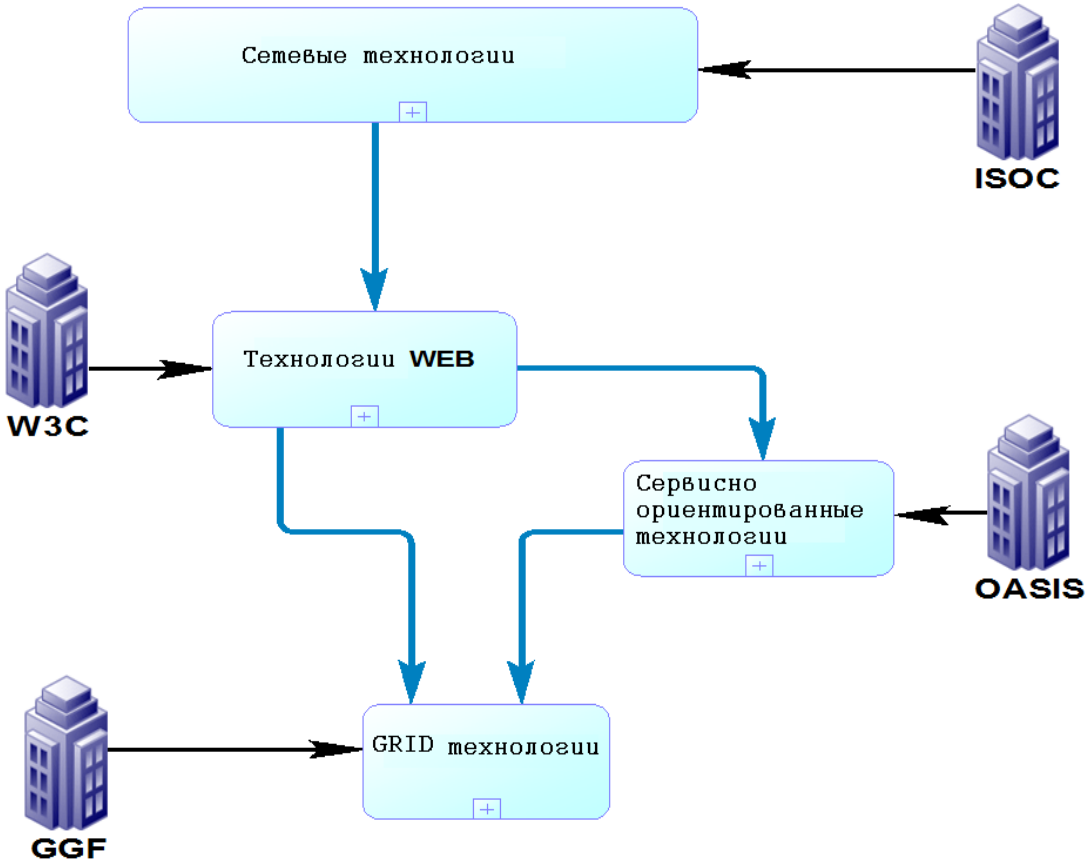

Рис.2. Организация формирования стандартных Интернет технологий

3.1. WEB технологии

Глобальная информационная сеть (WEB-среда) объединяет десятки миллионов документов по всему миру и развивается благодаря стандартам консорциум W3C [12-20]. Консорциум W3C объединяет наиболее крупных производителей программного обеспечения для WEB технологий (Web серверов и WEB браузеров). W3C обеспечивает компьютерным программам взаимодействие в сети (так называемая «сетевая интероперабельность»).

Стандарты глобальной информационной сети можно подразделить на 4 группы.

1). Представление форматных данных пользователя

HTML язык разметки гипертекста. и документов. в виде .ASCII текста, фрагменты которого облагаются специальными пометками (тегами);

XHTML расширяемый язык разметки гипертекста (Extensible Hypertext Markup Language) на основе принципов и синтаксиса XML:

CSS каскадные таблицы стилей (Cascading Style Sheets) для писания внешнего вида документа (страницы);

MathML (Mathematical Markup Language) – язык математических формул, который использует формат XML для отображения математических формул. Используется для отображения формул в Web браузерах MathML;

SVG (Scalable Vector Graphics) – язык масштабируемой двумерной векторной графики, которая использует формат XML.

2). Представление структурированных данных

XML Schema – язык для определения правил, которым должен удовлетворять документ и состоять из набора тегов и их атрибутов для описания соответствия документа его предметной отрасли.;

XLink (XML Linking Language) – XML схема, которая рекомендована W3C для организации ссылок между ресурсами;

XInclude (XML Inclusions) – XML схема, которая рекомендована W3C и предоставляет механизм включения в XML-документы текстовых файлов и др.;

XSL (eXtensible Stylesheet Language) – семейство рекомендаций (стандартов) которое определяет методологию превращения XML документов для визуализации или другой обработки соответствующими программными средствами;

XSLT декларативный язык, который интерпретирует правила и применяет их к входным документам;

DOM (Document Object Model) – не зависимая от платформы и языка программирования модель, который позволяет создавать динамические Веб страницы посредством скриптовых языков (JScipt, JavaScript);

XML Encryption – определяет порядок шифровки и дешифровки содержания элементов XML документа;.

XKMS XML Key Management Specification) – определяет порядок безопасной передачи и регистрации открытых ключей для шифровки и дешифровки XML документов за протоколом XML Encryption;.

PNG (portable network graphics) – формат для сохранения растровой графики;

SMIL (Synchronized Multimedia Integration Language) и язык мультимедийной интеграции на синтаксической основы формата XML..

3). Протоколы удаленного выполнения программ (сервисов)

SOAP (Simple Object Access Protocol) [15, 16] – определяет порядок обмена сообщениями (данными) в WEB среде.;

WSDL (Web Services Description Language) [16] – язык для описания Веб сервисов и доступа к ним на основе XML.

4). Представление семантических данных

RDF (Resource Description Framework) – ряд стандартов, который определяет базовые методы формального представления знаний для машинной обработки;

OWL (Web Ontology Language) – язык, который определяет правила описания онтологии предметных отраслей и предназначен для обеспечения одинаковой и однозначной интерпретации документов разными агентами в распределенной среде.

Наиболее распространенным является протокол SOAP**.** При описании сервиса указывается адрес URI (Uniform Resource Identifier) и транспортный протокол (например, HTTP). Средством описания функциональности сервиса является язык WSDL (Web-service description language, см. [15]). Для представления данных, в особенности метаданных, используется модель RDF (см. [16]). Описание процессов представления и обработки запросов на сервисы в графическом виде осуществляется языками:

WSCI (Web Services Choreography Interface,

WSCL (Web Services Conversation Language,
BPMN (Business process and model and notation,
BPEL (Business Process Execution Language for Web Services, см. [20-24]) и др.

3.2. Сервисно-ориетированная архитектура (SOA) и SCA

Модель SOA (Service-oriented Architecture) задает сервисно-ориентированную архитектуру программной системы, а ***модель SCA*** (Service-Component Architecture) – архитектуру на основе сервисов и компонентов. К средствам моделирования сложных систем относится система WebSphere Integration Developer компании IBM. Она предоставляет сервис-ориентированную архитектуру SOA и SCA, в виде use case языка UML Эта система обеспечивает интеграцию сервисов SCA через модель интерфейсов JAVA, задаваемую в языке WSDL и классах JAVA. Эта модель дает доступ к сервисным компонентам и определяет зависимость между ними через аппарат ссылок. Они упаковываются в модуль для выполнения на сервисном модуле WebSphere Process Server, который эквивалентен EAR-файлу J2EE и некоторым другим. Подмодули J2EE и артефакты упаковываются с модулем SCA, который затем запускает сервис и передает данные для их интеграции [24].

Механизмы, которые используются для вызова внешнего сервиса, названного *импортом* и *экспортом*, связаны с другими технологиями, такими как JMS, Enterprise JAVA Beans или веб-сервисы. SCA модуль может обратиться к существующему Enterprise JAVA Bean для обеспечения релевантного представления в *универсальной модели данных*, а также обмена данными друг с другом через SDO. В этой модели объекты данных представлены в JAVA common.sdo.DataObject и включают в себя метод, который позволяет пользователям получать свойства данных. WebSphere Integration Developer используется на платформе Eclipse 3.0.

3.2.1. Модель SOA

Модель SOA предоставляет набор принципов и средств создания системного ПО и прикладных ПС из совместимых и унифицированных сервисов.

Модель группируется на серверной стороне из некоторого количества согласованно реализованных сервисов и их служб. Группы задают открытый интерфейс, содержащий описание типов входных/выходных параметров каждого сервиса и портов обмена метаданными в языке WSDL. Для WSDL созданы компиляторы, позволяющие получать серверные и клиентские заместители и учитывающие особенности конкретных программных платформ, в том числе языки программирования на этих платформах, которые описывают

реализуемые операции.

Унификация сервисов состоит в типизации функциональности сервиса и его характеристик, а также языков описания сервисов и их взаимодействия. Объект SOA обладает специфицированной функциональностью и качеством. В его модели используются две технологии, которые обеспечивают функциональность (Functions) и качество сервисов (Quality service). Эти технологии вынесены на уровень IT-стандартов комитета W3C.

Технология обеспечения *функциональности* веб-сервисов включают в себя:

1) транспортный уровень (transport layer) для обмена данными;
2) коммуникационный уровень (service communication layer) протоколов;
3) сервисный уровень (service description layer) и связанные с ним интерфейсы;
4) уровень бизнес-процессов (business process layer) для реализации бизнес-процессов и потоков работ через механизмы веб-сервисов;
5) уровень реестра сервисов (service registry layer), который обеспечивает библиотеку веб-сервисов для их публикации, поиска и вызова WSDL-описаний интерфейсов.

Технология обеспечения *качества* веб-сервисов имеет следующие уровни:

6) политики (policy layer) для описания правил и условий применения веб-сервисов;
7) безопасности (security layer) для описания вопросов безопасности веб-сервисов и функционирования (авторизация, аутентификация и распределение доступа);
8) транзакций (transaction layer) для установления параметров обращения к веб-сервисам и обеспечению надежности их функционирования;
9) управление (management layer) веб-сервисами.

Технологическую основу составляют: XML, SOAP, UDDI, WSDL. С их помощью осуществляется реализация базовых свойств веб-сервиса и механизмов взаимодействия между собой веб-сервисов в среде SOA (рис.3).

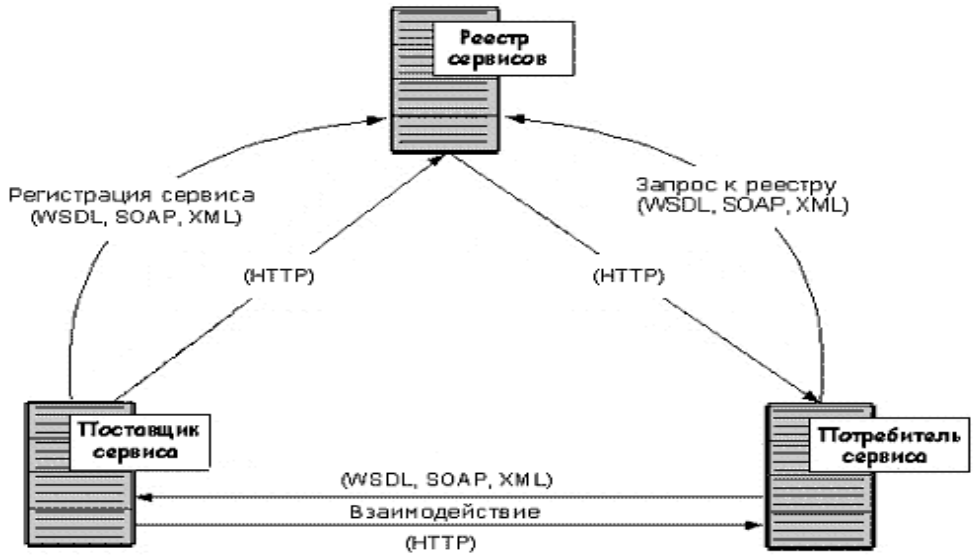

Рис. 3. Поставщики и потребители сервисов в сети

К ним относятся:

1) провайдер сервиса осуществляет реализацию сервиса, прием и выполнение запросов пользователей, публикацию сервиса, от из реестра сервисов;
2) реестр сервисов содержится библиотеку сервисов для поиска и вызова сервиса по запросам от поставщика или провайдера сервисов, предоставляющих сервисы;
3) потребитель или пользователь сервиса осуществляет поиск и вызов необходимого сервиса из реестра описания сервисов, а также использует сервис, предоставленный поставщиком в соответствии с его интерфейсом (рис. 3).

Посредником между этими службами и приложениям является *провайдер*, который обеспечивает взаимодействие меэжу поставщиками и провайдерами с помощью средств описания и передачи сервисов WSDL, SOAP, XML.

Для получения сервиса в архитектуре SOA выполняются следующие операции:

- публикация сервиса WSDL с целью обеспечения доступности пользователю сервиса и его интерфейса;
- поиск сервиса в реестре с помощью протокола SOAP и заданных критериев;
- связь с реестром UDDI через описание пользователем необходимого сервиса.

При этом предусматривается, что в реестре архитектуры SOA содержится описание сервиса с форматом запросов пользователя к провайдеру, содержащему в себе перечень описаний сервисов, которые могут быть вызваны соответственно с опубликованным интерфейсом сервиса.

К базовым функциям управления компонентами и службами в операционной среде относятся:

- поиск необходимых ресурсов (компонентов, reuses, assets, artifacts, служб и др.);
- доступ к названным ресурсам;
- организация обмена информации между компонентами, ресурсами и службами;
- динамическое управление функционированием заданной совокупности ресурсов.

Модель сервисов ПС базируется на унификации и совместимости, что позволяет рассматривать ПС как набор сервисов, функциональности и взаимодействия. Унификация достигается путем:

- типизации функциональности сервиса и их других характеристик;
- применения унифицированных языков для описания сервиса и их взаимодействия;
- использования стандартных базовых технологий.

Отдельный класс средств унификации составляет онтология – модели и словари, которые обеспечивают согласование терминов и понятий языка описания сервисов на уровне семантики. В качестве стандартных базовых технологий при реализации сервисов используются: модель клиент-сервер, унифицированные коммуникационные протоколы, компонентные модели и т. д.

Смысл SOA состоит в том, чтобы создавать небольшие компоненты и собирать их в большой распределенный по глобальной сети комплекс под бизнес-задачи конкретного клиента. Функциональные сервисы, как обычные программные модули, могут быть распределены по вычислительным системам и обладать способностью к взаимодействию посредством локальных и/или глобальных сетей. Интерфейс таких модулей не зависит от технологии или платформы, в рамках которой они реализованы.

Бизнес-процесс определяется как набор взаимосвязанных задач, относящихся к деловой деятельности, имеющий начальные и конечные точки для повторения задачи.

3.2.2. Сервисно-компонентная архитектура (SCA)

Модель **SCA** предназначена для работы с компонентами со спецификациями, разработанными различными компаниями: компонентами EJB сервера приложений J2EE, сетевыми сервисами, объектами планирования, доступа к базам данных, к информационной системе предприятия и др. [2, 24].

Архитектура SCA обеспечивает доступ к сервисным компонентам и определяет зависимости между ними через аппарат ссылок. Механизмы, которые используются для вызова внешнего сервиса, называются импортом и экспортом. Элементы SCA могут компоноваться и обмениваться данными друг с другом, пересылая сервисные объекты данных, подготовленные в необходимом виде. Этот интерфейс включает определение метода получения и установления свойства данных. В рамках модели SCA сервисы могут собираться в различные образования (хореографии). Они используют архитектуру SOA и/или создают новые сервисы для их комбинирования и конфигурирования (рис.4).

Характеристики приложения можно согласовывать со специфическими потребностями клиента (конфигурирование), используя такие элементы, как стандарты бизнеса, бизнес-правила, соглашения о бизнес-сервисе и параметры конфигурации (рис. 2).

Модель SCM представляет собой обобщение объектно-компонентной модели семейства программных продуктов (см. [1]). В ней каждый член является: системой удаленных reuses, которые обмениваются гетерогенными данными и предоставляют сервисы реализации с множеством общих свойств; композицией сетевых сервисов, которые поддерживают некоторый деловой процесс. Данная модель ориентирована на обеспечение адаптивности ПС к переменным условиям использования запросов к функциям и обрабатываемым ими гетерогенных данных. В интересах этой поддержки для программной реализации ПС используются механизмы сервисных объектов данных и сервисов доступа к ним. Они позволяют размежевать код ПС и код доступа к обрабатываемым данным. В состав модели SCM входят подмодель *абстрактных сервисов, интерфейсная* подмодель и подмодель объектных *ресурсов*.

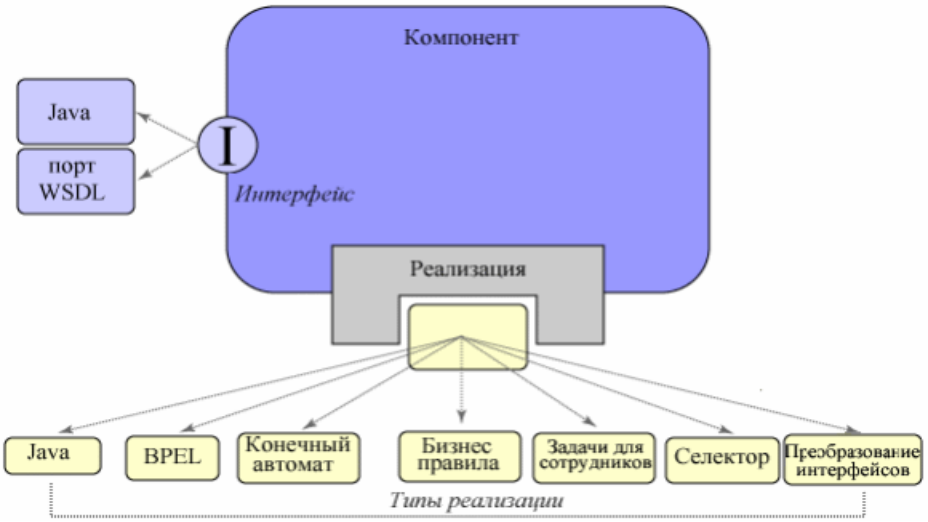

Рис. 4. Общая схема сервера SCA IBM

Сервер приложений Java Enterprise Edition содержит набор спецификаций на языке Java, которые необходимы при работе с сетевыми программами и средствами. К ним относятся:

- динамическая генерация серверных страниц (Java Server Pages);
- сетевые службы;
- компоненты повторного использования (Enterprise Java Beans);
- служба обмена сообщениями (Java Message Queue) и другие сервисные технологии.

Сетевая служба идентифицируется с помощью универсального ресурсного идентификатора URI, ее ресурсы (свойства и методы) описываются на специальном языке WSDL. Доступ к ресурсам осуществляется через протокол SOAP, который представляет собой XML-запросы, передаваемые посредством интернет-протоколов относительно высокого уровня (HTTP, SMTP). Сетевые службы соответствуют объектам объектно-ориентированных языков программирования с некоторыми важными отличиями.

Ключевым понятием сетевой службы **SCA** является сообщение, которое состоит из одной или нескольких переменных. Вместо методов классов в сетевые сервисах используются операции, которые определяются входным и выходным сообщениями. Для описания общедоступных ресурсов сетевых сервисов в язык WSDL, построенный на синтаксической основе языка разметки XML, введены возможности описания данных различных типов. В качестве переменных для сообщений можно использовать последовательности, созданные из фиксированного количества переменных простых типов, причем типы, которые будут использоваться в службе, декларируются заранее.

3.2.3. Среда WCF для взаимодействия систем

Программная среда Windows Communication Foundation (WCF) входит в состав среды .NET Framework (см. [25]) и является логическим развитием технологий сетевых служб, .NET Remoting и DCOM. В основе WCF лежит модель SOA, которая обеспечивает на сервере работу некоторого количества сервисов, определенных в интерфейсе для задания абстрактных входных/исходных параметров. Эти операции описываются на WSDL и могут быть сделаны доступными через, так называемые mex-endpoints (Metadata Exchange Endpoints), что позволяет получить "метаданные" сервиса. Подключаясь к этому интерфейсу; можно получить описание сервиса и всех его операций, а также сгенерировать соответствующий прокси-класс (класс-заместитель) для заданного языка или платформы. Сервис описывается в языке WCF и используется с языками Java / Python / Ruby и т. п. Клиенты в свою очередь имеют на своей стороне прокси-классы, которые содержат ссылку на операции к сервису.

В WCF MS.Net работает технология фабрик программ (AppFab), основанная на нескольких веб-службах и фабрике сервисов. Основу технологии составляют схемы, "рецепты", методы и средства построения программ разного назначения. Фабрика программ включает в себя набор ресурсов, блоков кода, документации, образцов приложений, инструментов и патернов VSIP для создания из них разных пакетов, которые накапливаются в глобальном хранилище Global Bank. Фабрика сервисов содержит рекомендации, схемы и методы, а также стандарты по проектированию и конструированию продуктов, Функционирование WCF определяется так называемыми конечными точками (endpoint), которые устанавливают "ABC" или "Address-Binding-Contract" (см. рис.5).

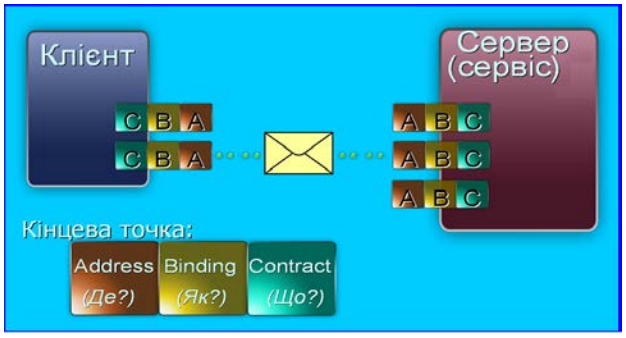

Рис.5 Точки в WCF

Каждая точка выполняет определенную роль:

"Address" задает место расположения конечной точки по абсолютному или относительному адресу;

"Binding" задает связь с транспортным протоколом, на основе которого будет происходить взаимодействие. Связь в виде классов-привязок (BasicHttpBinding в HTTP, NetTcpBinding в TCP и т. д.);

"Contract" задает контракт, на основе которого будет происходить взаимодействие клиента и сервиса с помощью операций сервиса, который строит класс-прокси на стороне клиента.

На сервисной стороне задается множество конечных точек. В результате, сервисная сторона распределенного приложения будет выглядеть как совокупность конечных точек.

Инструменты распределенного действия WCF или .Net Remoting построены по принцип «слоенного пирога», каждый слой которого отвечает за свой конкретный уровень абстракции и не знает ничего о нижележащих уровнях. Инфраструктура WCF состоит из двух главных уровней: Service Model Layer и Channel Layer. Первый уровень расположен ближе к самому сервису клиента и обеспечивает превращение метода с параметрами в сообщение для передачи более низкому уровню. Канальный уровень (Channel Layer) инкапсулирует канал передачи данных, которых может быть много, такие как транспорт TCP, Http, Named Pipes и т. д. Каждый из этих уровней содержит подуровни, и может включиться в каждый из них.

Контракты представляют собой описание сообщений, переданных конечными службами с возвратом. Конечная точка должна специфицироваться и может выполняться в формате ожидаемых данных. Совокупность этих спецификаций и есть контракт.

WCF содержат три вида контрактов:

1) контракты сервисов для описания функциональных операций, реализованных сервисом. Внутри контракта имеются операции сервиса, которые реализуют функции;
2) контракты данных определяют формат данных, которыми будут обмениваться сервисы. Это относится к запросу на сервис и октету сервиса. Если используются примитивные типы – ***int***, ***string*** и др., то контракт не нужен, потому что в .Net имеется возможность сериализации и десериализации типов. Для комплексных типов – *Customers*, *Order* и др., необходимо указать принцип сериализации и десериализации;
3) контракты сообщений, как тип контракта, который используется для того, чтобы получить контроль над заголовком SOAP (рис.6).

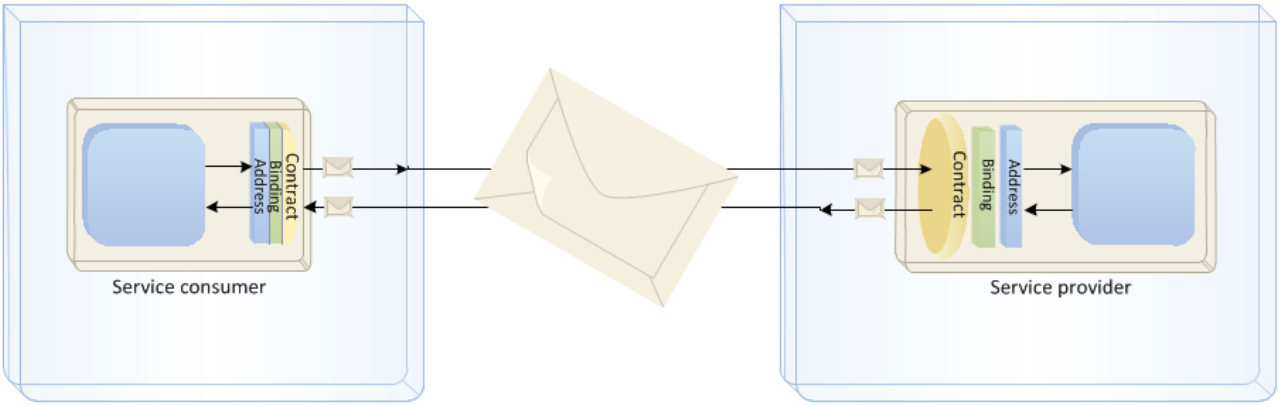

Рис. 6 Передача пакета сообщения между потребителем и поставщиком

В пакете передается протокол SOAP в виде:

<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://www.cbsystematics.com">
<!-Конверт протокола SOAP->
<env:Header>
<!- Заголовок протокола SOAP->
</env:Header> <env:Body>
<!-Тело протокола SOAP->
</env:Body> </env:Envelope>.

При взаимодействии систем общего назначения друг с другом через сообщения в конверте могут возникнуть разного конфликты передачи данных (рис. 7).

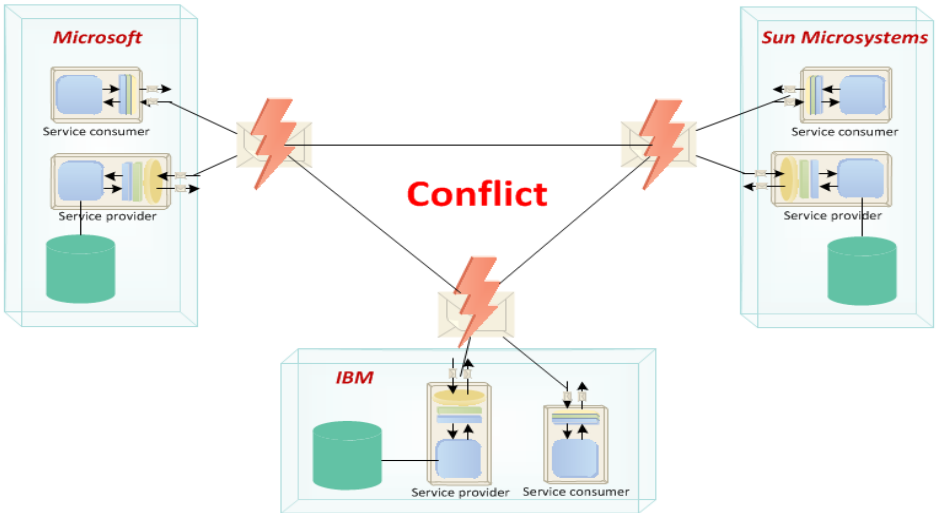

Рис.7. Схема взаимодействия систем с возникновением конфликтов

Например:

1) Несовместность типов данных, которые передаются в контрактных интерфейсах потребителя (вместо целого присылается символьный).
2) Несоответствие в порядке (или количестве) параметров в протоколе передачи;
3) Разлица в архитектуре платформ объектов клиента и сервера.

Для обеспечения интероперабельности контрактов в широком диапазоне систем, используются языки WSDL и XSD. При написании кода службы поставляется класс с определенными в WCF атрибутами [ServiceContract], [OperationContract], [FaultContract], [MessageContract] и [DataContract] и утилита *svcutil.exe*, которая вызывает конечную точку сервиса для возврата данных после генерации WSDL-документа по атрибутам.

На этапе выполнения вызывается метод, определенный в интерфейсе сервиса, WCF сериализует типы CLR и вызов метода в формат XML и посылает сообщение в сеть для привязки к схеме кодировки, согласованной с WSDL. При этом участвуют четыре конструкции: две со стороны .NET и две со стороны XML. В .NET имеется тип CLR, который определяет структуры данных и функциональные возможности создания объект такого типа. Со стороны XML задается XSD-описание структуры данных, но сообщение осуществляется лишь после того, как будет создан экземпляр XML (XML Instance).

Язык WSDL. Для описания общедоступных веб-ресурсов используется язык WSDL (Web Service Definition Language) на основе XML. Среда программирования Eclipse позволяет автоматически создавать описания на основе классов Java. В языке определены следующие основные типы данных:

1) строки (xsd:string);
2) целые числа (xsd:int, xsd:long, xsd:short, xsd:integer, xsd:decimal), числа с плавающей запятой (xsd:float, xsd:double);
3) логический тип (xsd:boolean);
4) последовательности байт (xsd:base64Binary, xsd:hexBinary);
5) дата и время (xsd:time, xsd:date, xsd:g);
6) объекты (xsd:anySimpleType).

В качестве переменных для сообщений могут использоваться последовательности, созданные из фиксированного количества переменных простых типов.

Типичный WSDL-файл имеет следующую структуру.

```
<wsdl:definitions [.]>
<!-- Декларация типов, которые используются в сервисе —>
<wsdl:types>
<element name="someMethod">
```

<complexType>
<sequence>
<element name="arg0" type="xsd:*double"*/>
<element name="arg1" type="xsd:*boolean"*/>
</sequence>
</complexType>
</element>
<element name="someMethodResponse">
<complexType>
</wsdl: types> …

Приведенное WSDL-описание определяет веб-сервис MyService с единственным методом String someMethod(double arg0, boolean arg1). На его основе можно сгенерировать два типа данных, которые отвечают входным и исходным аргументам метода. Эти типы применяются в описаниях someMethodRequest и someMethodResponse – входного и выходного сообщений для операции someMethod.

Операции декларируются в описании интерфейса сервиса (декларация wsdl:portType) и дальше в описании привязки сервиса к SOAP (декларация wsdl:binding), причем во втором случае также оговаривается способ вызова (<wsdlsoap:body use ="literal"/>). За счет этого при вызове операции используются те же названия параметров, что и в методе класса. В конце WSDL-файла находится декларация веб-сервиса (<wsdl:service>), в которой содержится информация относительно его расположения (параметр location).

3.3. Технологии Grid

Документ Open Grid Services Architecture (OGSA) [2, 8, 9], определяет концептуальную основу системы Grid и ее основных компонентов (рис.8): архитектуру, данные, инфраструктуру, компьютеры и приложения. Система Grid объединяет локальные ресурсы, которые настраиваются и администрируются автономно (нижний слой), объединяются, руководствуются и динамически отслеживаются за счет программного обеспечения промежуточного слоя (middleware).

Прикладное программное обеспечение через стандартизованные интерфейсы имеет доступ к локальным и распределенным ресурсам через средний слой, который скрывает от пользователя внутреннюю сложность и физическое отображение ресурсов. В основу стандарта OGF положена архитектура SOA, согласно которой вся функциональность промежуточного слоя реализуется путем комбинации взаимоувязанных Web сервисов (SOAP, WSDL). Служба OGSA (Basic Execution Service) определяет интерфейс к сервисам, которые инициируют вычислительные процессы в Grid, отслеживают и руководят вычислительной деятельностью. Кроме того определены модели

жизненного цикла (состояния) процессов, а также информационные модели вычислительного процесса.

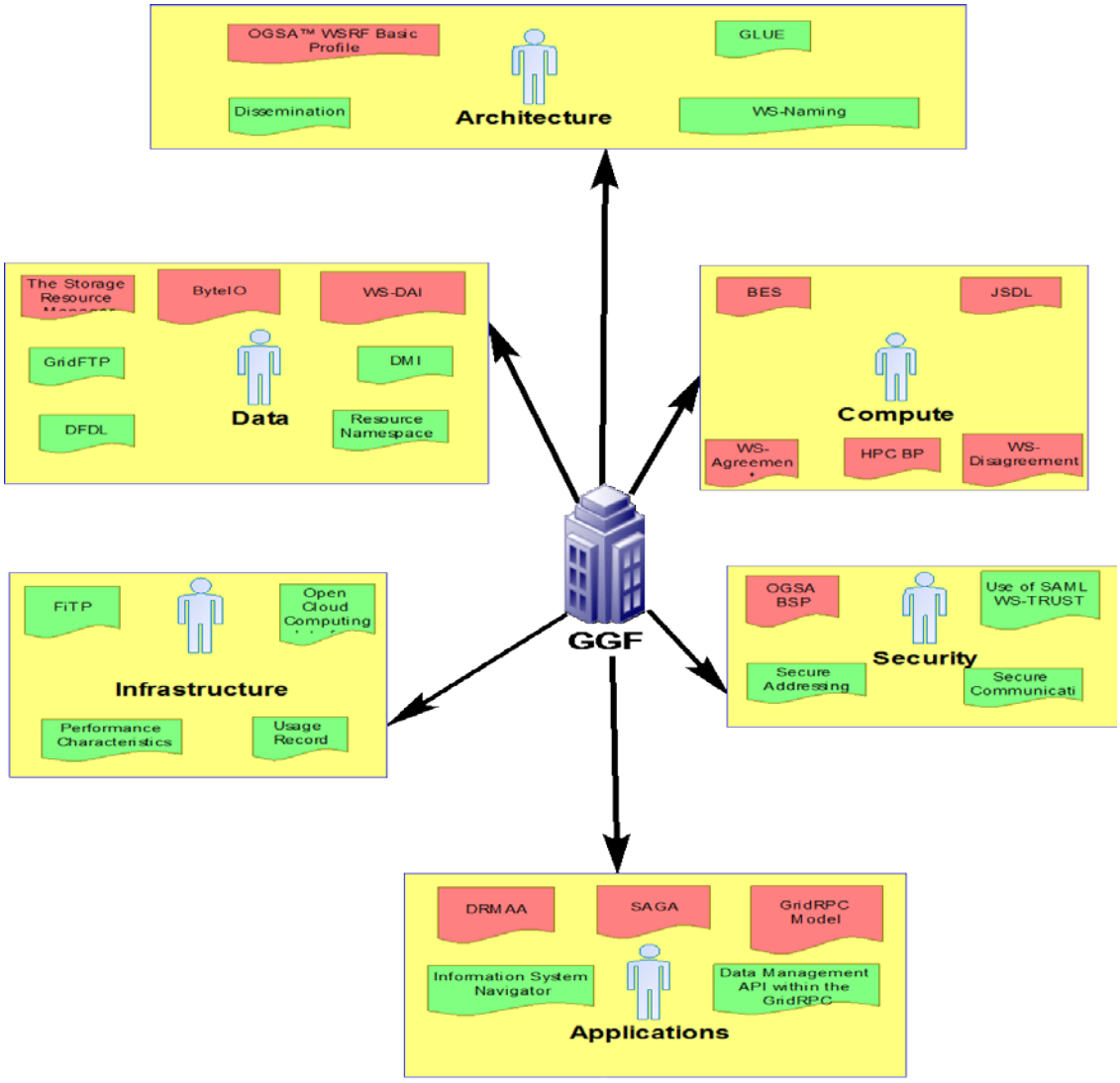

Рис. 8. Структура стандартов Grid

В Grid реализован программный интерфейс SAGA (Simple API for Grid Applications). Он определяет программную модель для описания компонентов Grid и сигнатуру методов для поддержки процесса выполнения заданий, управления ресурсами и др. Целью спецификации является предоставление стандартизировавшего доступа к функциям Grid на языках высокого уровня (C, Java, Python и т.п.) и тем самым обеспечение их совместимости с разными middleware.

Одной из Grid систем является ETICS. Она предоставляет e-инфраструктуру (Web Application и Web Services) для тестирования, интеграции и конфигурирования, занесения в репозиторий готовых программ и ПС (рис. 9). ETICS включает процессы сборки (builds) и управления тестами на компьютерах пользователей, обеспечивают отправку удаленных сборок на разные платформы пользователей [26]. Программные объекты (пакеты, компоненты и системы) разрабатываются по модели CIM (Common Information Model).

Сценарии работы пакетов и систем задаются на языке моделирования UML. Сервисный набор процедур Etics обеспечивает построение и тестирование новых пакетов и систем с помощью механизмов плагинов. Каждый плагин включает в себя публичный интерфейс и описание услуг.

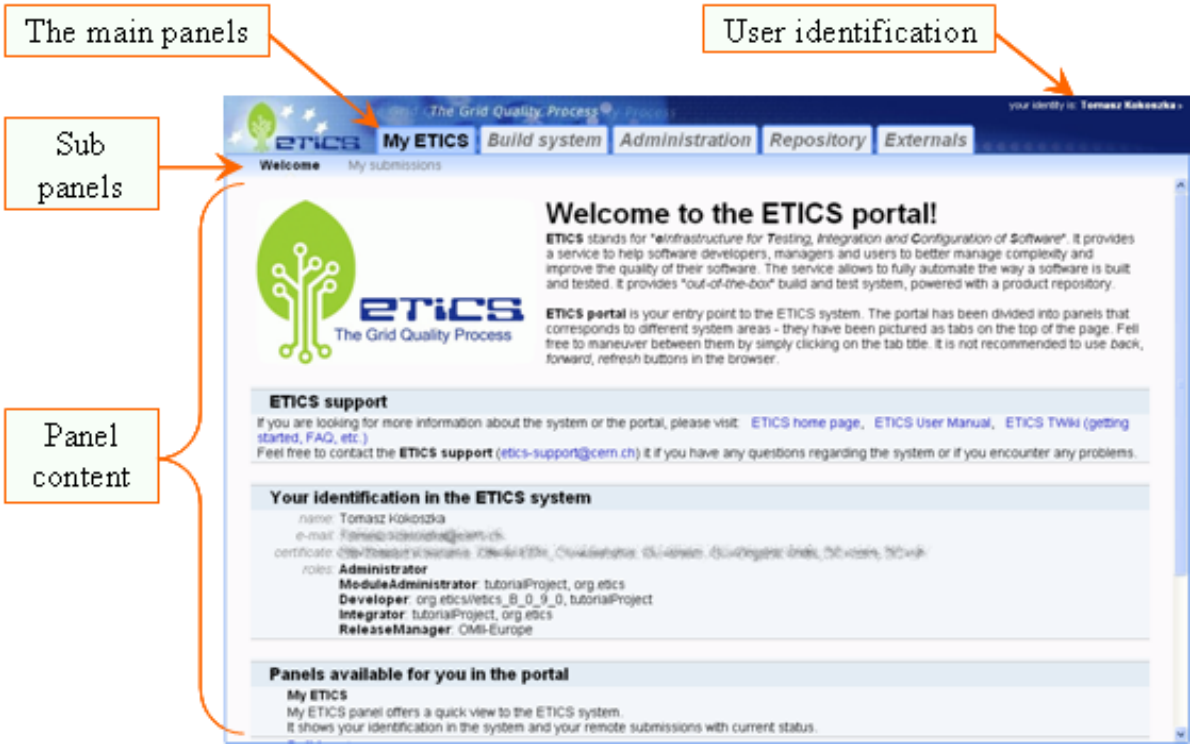

Рис. 9 Портал ETICS

В набор характеристик системы входят механизмы спецификации зависимостей между разными пакетами и их тестами. Функциональные плагины обеспечивают проверку договоров, тестов для выполнения разных элементов систем, генерацию документации и ведения готовых объектов в оперативном или постоянном репозитории. В ETICS разработаны наборы пакетов из комбинаций перекомпилированных двоичных элементов, размещенных в репозитории. Компоненты регистрируются в репозитории в стандартном языке WSDL консорциума W3C (рис.10). Стандартное описание компонентов используется при проведении сборки и тестирования (Build and Test Web Application) групп компонентов. При конфигурации совокупности

компонентов в результирующий файл поступают команды контроля версий (Command-Line Client), тестирования свойств и зависимостей в выходном коде системы. Этот сервис можно задать командной строкой задания сборки и тестирования (Build and Test Service). *Удаленная сборка/тестирование* выполняется на разных платформах с генерацией ряда отчетов о сборке, статическом и динамическом тестировании отдельных компонентов в разных средах глобальных пользователей.

Каждый новый компонент содержит сведения (имя, лицензия, URL репозитория), глобальный уникальный идентификатор – ID (GUID); информацию о версиях, GUID (свойства, среде выполнения и зависимости), тестовых команд и GUIDs.

Типы данных пакетов и систем объединяется плагинами, в которых задаются услуги для потребителей или поставщиков, доступ к OCAM, архитектуре CPU, компиляторам с языков программирования и средствам спецификации зависимостей между разными пакетами, используемым при сборке программ и их развертывании. Плагины обеспечивают проверку контрактов, тестов выполнения разных элементов систем, генерацию документации, ведения готовых КПИ в репозиториях.

Главная проблема в ETICS – преобразование некоторых компонентов систем для альтернативной платформы гетерогенной среды компьютеров путем ссылок с 16-, 32-разрядной платформы на 64-разрядную платформу среды Grid.

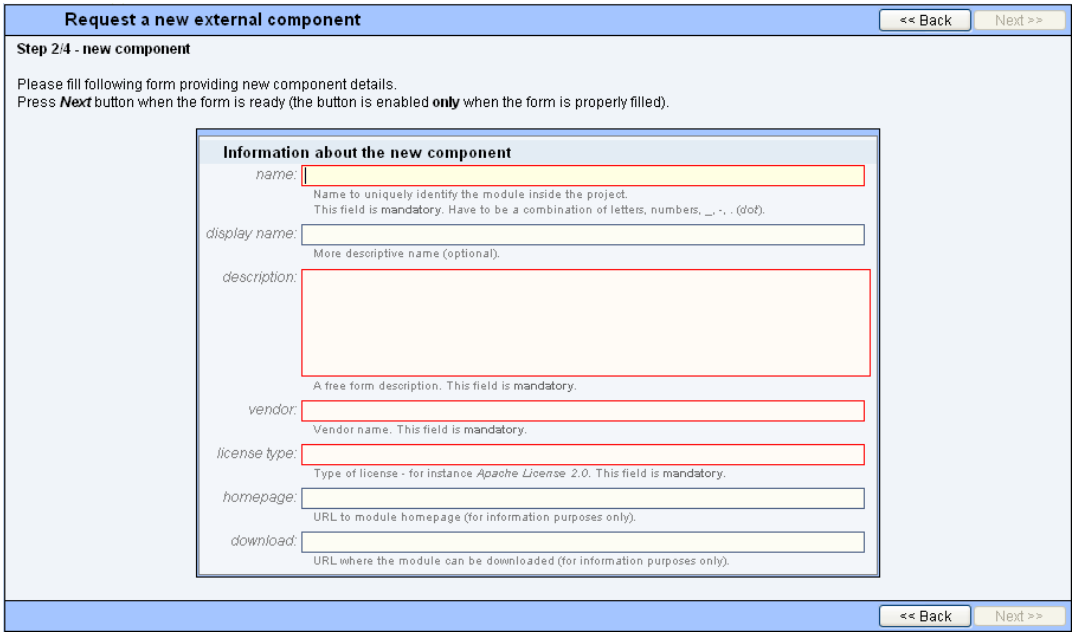

Рис. 10. Стандарт описания нового компонента в Grid

4. ЗАКЛЮЧЕНИЕ

В работе рассмотрены сервисы, которые являются системной поддержкой процессов разработки программных систем и веб-сервисы Интернет как информационные ресурсы, используемые для решения научных и бизнес-задач в среде Интернет. Дано описание набора системных сервисов CORBA для управления объектами при обмене данными (через stub и skeleton) между клиентом и сервером. Представлены модели сервисов SOA и SCA, используемые для композирования систем из сервисов и последующего их выполнения в распределенной среде. Дано описание системы Grid, в состав которого входит ряд систем поддержки физического эксперимента и реализации прикладных систем в e-sciences (физики, биологии, математики, медицины и др.). В ней реализованы операций сборки (building), тестирования (testing) и конфигурации (configurating) и работа с данными, согласно универсальной модели данных. Приведено описание ряда стандартов W3C, включая протоколы и языки описания сервисов для функционирования в среде Интернет (XML, SOAP, BPMN, WSDL, BPEL и др.). Приведено описание протоколов (Icontract) в системе WCF для передачи сообщений между распределенными системами и обеспечения взаимодействия между разнородными системами в среде Интернет.

ЛИТЕРАТУРА

1. Андон Ф.И. Методы инженерии компьютерных распределенных приложений.-К: Наук.думка.-1997. 271 с.
2. Лаврищева Е. М. Software Engineering компьютерных систем. Парадигмы, Технологии, CASE-средства программирования. – К.: Наук. Думка, 2014 – 284 с.
3. Лаврищева Е. М., Грищенко В.. Сборочное программирование. Основы индустрии программных систем.- Наук.Думка, К.: 2009.-370с.
4. Карпов Л. Е. Архитектура распределенных систем программного обеспечения – М., МАКС Пресс, 2007. – 130 с.
5. Карпов Л. Е., Юдин В. Н. Обмен данными в распределённой системе поддержки решений. Труды Института системного программирования, т. 19, М., Институт системного программирования РАН, 2010, стр. 71-80, ISBN 978-0-543-57630-9, ISBN 978-5-4221-0085-9, ISSN 2220-6426 (Online), ISSN 2079-8156 (Print), http://www.ispras.ru/ru/proceedings/docs/2010/19/isp_19_2010_71.pdf
6. Jon Siegel. "Quick CORBA™ 3". Wiley Computer Publishing, John Wiley & Sons, Inc., 2001 (Джон Сигел, "CORBA 3", М., МАЛИП, 2002).
7. Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju. Web Services. Concepts, Architectures and Applications. Springer-Verlag, 2004.

8. А.П. Демичев, В.А. Ильин, А.П. Крюков, Введение в грид-технологии. Препринт НИИЯФ МГУ - 2007 - 11/832 2007, http://www.sinp.msu.ru
9. А.М. Ходжибаев*, Ф.Т.Адылова, Новейшие информационные ГРИД-технологии в электронной медицине, Донецк, журнал телемедицинита медичної телематики том 3, №1, 2005
10. Lim Gray. A transformed scientific method. http://research.microsoft.com/en-us/collaboration/fourthparadigm/4th_paradigm_book_jim_gray_transcript.pdf
11. Дрёмин И.М. Физика на большом адронном коллайдере. УФН. Июнь 2009. т. 179 № 6 Fusion for energy. Annual report 2011. av. At http://fusionforenergy.europa.eu/downloads/mediacorner/publications/reports/ANNUAL_2011.pdf
12. http://www.w3.org/TR/2008/REC-xml-20081126/
13. Гладцын В. А., Кринкин К. В. Яновский В. В. Сервис-ориентированная архитектура: стандарты, алгоритмы, протоколы – Санкт-Петербург: СПб ГЭТУ ЛЭТИ, 2006 – 108 с.
14. Papazoglou M. P., Dubray J.-J. A Survey of Web Service Technologies, Technical Report DIT-04-058, Ingegneria e Scienza dell'Informazione, University of Trento, 2004.
15. http://www.w3.org/TR/soap/
16. http://www.w3.org/TR/soap12-part1/wsdl20/RDF/wsci
17. http://www.omg.org/spec/BPMN
18. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
19. http://www.oasis-open.org/specs/index.php#wsbpelv2.0
20. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm
21. http://www.oasis-open.org/specs/index.php#uddiv2
22. http://www.oasis-open.org/specs/index.php#uddiv3
23. http://www.ibm.com/developerworks/websphere/techjournal
24. Джим Амсден, Моделирование SOA: часть 1. Идентификация сервисов, IBM® Rational® Application
25. http://127.0.0.1:4000/IContract
26. https://web.infn.it/etics-support/index.php/documentation