# Сервисные средства интернет для решения бизнес-задач[1]

*Е. М. Лаврищева <lavr@ispras.ru>*
*Л. Е. Карпов <mak@ispras.ru>*
*А. Н. Томилин <tom11@bk.ru>*
*Институт системного программирования РАН,*
*109004, Россия, г. Москва, ул. А. Солженицына, д. 25.*

**Аннотация**. Описываются различные виды сервисов и служб, используемых в современных программных и распределенных системах. Дается характеристика широко распространенных и внедренных в практику систем со спектром системных и функциональных сервисов. Рассмотрены модели сервисной, сервисно-компонентной архитектур и системы сервисной поддержки WCF для представления прикладных систем из готовых сервисов с целью решения бизнес-задач. Приведен пример решения бизнес задачи обработки данных в калькуляторе, реализованном с помощью сервисной службы WCF комплекса ИТК.

**Ключевые слова:** распределенная система; глобальная сеть; модели сервисных архитектур, сетевой сервис, сетевая служба; протоколы взаимодействия; языки описания сервисов, бизнес задачи.

**DOI:** 10.15514/ISPRAS-2015-27(1)-7

**Для цитирования:** Лаврищева Е.М., Карпов Л.Е., Томилин А.Н. Сервисные средства интернет для решения бизнес-задач. Труды ИСП РАН, том 27, вып. 1, 2015 г., стр. 125-150. DOI: 10.15514/ISPRAS-2015-27(1)-7.

## *1. Введение*

Системная поддержка программирования, начавшись со слабоструктурированных наборов полезных последовательностей команд, записанных в долговременную память (например, на магнитный барабан или магнитную ленту), со временем прошла интенсивный путь через развитие библиотек программ, классов и компонентов к возникновению понятия "служба" (часто используется как термин "сервис"). Это понятие стало настолько популярным в среде программистов и пользователей их программ, что уже давно появилось выражение: "Все есть служба". Однако не всякая

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





системная поддержка, тем более поддержка распределенных программ, работающих в глобальной информационной сети, например, в сети Интернет, представляет собой службу в строгом значении этого термина. К настоящему времени промышленные стандарты требуют, чтобы правильно организованные службы содержали информацию о реализованной системной функции и форме ее применения. При этом службы работают автоматически – программы вызывают другие программы (службы).

Постепенно сформировались службы следующих видов:

- общие системные сервисы, имевшиеся сейчас во всех системных средах для поддержки процессов и функций этих сред (DCE, COM, DCOM, CORBA, MQ Series, J2EE, .NET и множество других), обработки программ и данных (например, службы именования, каталогов, коллекций, безопасности, событий и каналов);
- объектные сервисы, которые управляют объектами, классами и услугами по формированию и обработке объектно-ориентированных систем (например, службы жизненного цикла объектов);
- сетевые сервисы стандартных моделей SOA (Service-oriented Architecture), SCA (Service-Component Architecture), как инструменты представления и обработки ресурсов Интернет, которые реализуют деловые, финансовые, экономические и другие услуги для решения соответствующих задач в среде Интернет (аналогично в стандарте CORBA введен механизм Common Facility, выполняющий функции управления данными, системами, интерфейсами и др.);
- готовые программные ресурсы (reuses, assets, artifacts, resources, компоненты повторного использования), которые предоставляют программные сервисные услуги в некоторой прикладной области.

***Системные сервисы*** необходимы для организации построения и функционирования стандартных функций систем, а также для управления конфигурацией прикладных программ. Некоторые из сервисов стали обязательной частью системной поддержки, другие используются для специальных областей (например, медицина), а некоторые предоставляют необходимые услуги работы с данными, используемыми в языках программирования. В состав общего множества сервисов обязательно или часто могут входить:

1. Служба именования (naming) готовых компонентов для обеспечения их поиска в распределенной среде пространства имен.
2. Служба связи (binding), предназначенная для определения соответствия “имя-объект” применительно к найденным компонентам. По-видимому, эти две первые из перечисленных служб

– единственные, которые с уверенностью можно отнести к обязательному для каждой системы набору служб.

3. Служба транзакций (transaction), которая обеспечивает организацию и управление функционированием совокупности объектов и данных.
4. Служба обмена сообщениями (messaging), необходимая для организации общения между компонентами, как составной элемент в модели асинхронных транзакций. Транзакции могут выполняться как с использованием синхронного способа взаимодействия, так и в асинхронном виде. Однако асинхронное взаимодействие имеет большие преимущества почти во всем, за исключением простоты.

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 четыре вида сервисов и их служб используются при реализации моделей программной системы (ПС). В настоящее время в Интернете имеется более 100 миллионов сервисных ресурсов. Для их использования при создании программы решении конкретной задачи требуется проводить поиск подходящего сервисного ресурса, его апробацию и встраивание в прикладную программу решения задачи, либо использовать его в динамическом режиме (см. [1-7]).

## 2. Подходы к представлению и реализации сервисов

Первый вариант достаточно полного набора сервисов, основанный на объектно-ориентированном подходе, был сделан в проекте ***Common Object Request Broker Architecture*** (CORBA, см. [3, 5, 8-9]), который первоначально предполагал, что в состав системы поддержки должны входить компоненты (1994):

- *брокер объектных запросов* (Object Request Broker — ORB), обеспечивающий взаимодействие распределенных объектов;
- *объектная модель* (OM), объекты которой взаимодействуют между собой через интерфейсные объекты –посредники stub для клиента и skeleton для сервера, описываемые в специальном языке IDL (Interface Definition language) и обрабатываемые брокером ORB;
- *общие объектные сервисы* (Common Object Services), предоставляющие услуги всем объектам по управлению данными, изменениями программ и подпроцессов, а также по обработке транзакций и т. п.;
- *общие средства обслуживания* (Common Facilities) или общие





средства, предоставляющие ряд общих прикладных функций для любых приложений (средства печати, управление БД, электронная почта и др.);

- *объектные приложения* (Application Objects), к которым относятся приложения и их компоненты, реализующие задачи пользователя.

Объекты специфицируются средствами языков программирования (Smalltalk, Cobol, Ada-95, Lisp, PL/1, C++, Python, Java, IDLScript и др., см. [5, 8-9]) и могут быть реализованы на разных платформах и средах. Интерфейсы программ-посредников, которые тождественны интерфейсам клиентских и серверных компонентов, описываются в языке IDL (см. [8-9]). 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. При этом по сравнению с более ранними системами, ориентированными на работу в рамках процедурной парадигмы, в архитектуре CORBA сделан существенный шаг в сторону системной симметрии: обратные вызовы позволяют серверным компонентам выступать в роли клиентов, а клиентам, обрабатывающим обратные вызовы, обслуживать их, как если бы они были серверными компонентами.
Общие объектные службы предоставляют набор операций для работы с разными категориями объектных приложений. Наиболее известными являются следующие службы:

- именование и поиск объектов по именам, либо по свойствам и атрибутам;
- поддержка жизненного цикла объектов (создание объектов, их копирование и уничтожение);
- поддержка параллелизма обращения к объектам;
- поддержка очередей запросов к объектам;
- поддержка иерархия транзакций;
- обеспечение защиты объектов от несанкционированного доступа, поддержка авторизации и аутентификации клиентов и др.

Система CORBA предоставила типовые методы реализации и диспетчеризации объектов. Они широко используются многими современными операционными системами и средами, а также стали промышленными стандартами и применяются в системах Cloud Computing, Grid и др. Служба транзакций и безопасности CORBA используется в транзакционном мониторе CICS компании IBM, системах управления базами данных Oracle, Sybase, Mybase, сервере приложений J2EE компании Sun Microsystems и др.

## 2.1 Представление сетевых служб современными средствами

С широким распространением глобальной сети Интернет, интерес к сервисам постоянно растет. Это прежде всего связано с проявившимся пониманием того факта, что без стандартизации адекватное функционирование распределенных приложений, опирающихся на распределенную системную поддержку, просто невозможно. Понятие "сетевой службы" (см. [5]). Не всякая программа, доступная в Интернете, имеет право называться сетевым сервисом. Чтобы получить это право, сервис должен удовлетворять целой серии стандартов, описывающих разные аспекты его состояния и поведения. С общей точки зрения модель сетевой службы имеет типизированную структуру и интерфейс. Модель "клиента" и "поставщика сервиса" сближены, то есть тенденция, наметившаяся в процессе развития более традиционных сервисов, еще более усилена. Для представления разных аспектов описания сервисов в качестве синтаксической основы используется язык XML (см. [10]). К этим аспектам относятся описания структуры и семантики данных, механизмов взаимодействия сетевых служб и функциональных услуг поиска необходимых сервисов (см. [11, 12]).

В качестве базового протокола доступа к объектам в информационной сети, определяющего форматы и методы упаковки информации, используется простой протокол доступа к объектам, который называется SOAP (Simple Object Access Protocol, см. [13, 14]). Этот протокол не детализирует свойства конкретного обмена информацией и задаёт шаблон обобщённого сообщения.

Интерфейс сетевых служб, как правило, описывается на языке IDL, а различные виды взаимодействия описываются с помощью схем XML. При описании сервиса необходимо указывать его адрес (Uniform Resource Identifier – URI) 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]). Для описания процессов представления и обработки запросов на сервисы в графическом виде предложены языки:

- WSCI (Web Services Choreography Interface, см. [17]),
- WSCL (Web Services Conversation Language, см. [18]),
- BPMN (Business process and model and notation, см. [19]),
- BPEL (Business Process Execution Language for Web Services, см. [20-21]) и другие.

В качестве адреса объектов в сети используются универсальные идентификаторы ресурсов URI и интерфейс, задаваемый для организации связи с другими сервисами с помощью XML-документов.





Таким образом, к основным средствам описания, разработки и взаимодействия систем в глобальных сетях относятся:

- язык XML, предназначенный для разметки и описания структуры и способов взаимодействия компонентов распределённой архитектуры;
- протокол SOAP для определения форматов запросов к сетевым службам;
- язык WSDL, основанный на XML и предназначенный для описания интерфейсов сервисов, типов данных, сообщений, операций, типов портов, моделей взаимодействия и протоколов связи сервисов между собой;
- служба регистрации UDDI (Universal Description, Discovery and Integration, см. [22-25]) для универсального описания, выявления и интеграции служб, обеспечения их хранения, поиска и сопровождения, упорядочения деловой служебной информации в специальном реестре с указателями на конкретные интерфейсы и адреса сервисов;
- язык описания протоколов прикладного уровня ("бизнес-протоколов") BPEL для описания наборов правил, регулирующих взаимодействия сетевых служб ("разговоры");
- графическая нотация языка BPMN для нотации прикладных процессов; описания структуры системы на языке моделирования UML (Unified Modeling Language, см. [26-27]); детализации объектной структуры и последующей разработки текстов прикладных программ; представления процесса выполнения системы в BPEL, включая бизнес-процессы на сервере приложений;
- модель SOA (Service-oriented Architecture) для описания сервисно-ориентированной архитектуры программной системы;
- модель SCA (Service-Component Architecture) для создания сложных систем на основе сервисов и компонентов.

Модели SOA и SCA реализованы в .NET (Microsoft), WebSphere (IBM), Weblogic (BEA, Oracle), Java Enterprise Edition (Sun), SAP, Soap WS и др.

## 2.2 Модель сервисов SOA (Service-oriented Architecture)

Идея, предлагаемая архитектурой SOA, заключается в группировании на серверной стороне некоторого количество согласованно реализованных сервисов и их служб. Группы должны задавать открытый интерфейс, содержащий описание типов входных/выходных параметров каждого сервиса. Эта информация задается с помощью языка WSDL и описания так называемых портов обмена метаданными (Metadata Exchange Endpoints). Для

работы с языком WSDL созданы компиляторы, позволяющие на основе текстов, написанных на этом языке, серверные и клиентские заместители (прокси-классы). Они учитывают особенности конкретных программных платформ, в том числе языки программирования на этих платформах, которые описывают реализуемые операции.

Данный механизм хорошо отработан и для систем, построенных на традиционных принципах. Он обеспечивает согласованность, языковую независимость и интероперабельность серверной и клиентской частей распределённой системы. От разработчика требуется написать контрактный сервис средствами WCF (Window Communication Foundation) и использовать его в реализации клиентов, написанных в одном из языков – Java, Python, Ruby или др. Клиенты в свою очередь имеют на своей стороне собственные заместители сервера.

Таким образом, на новом витке развития информационных технологий можно наблюдать воспроизведение достижений, полученных еще разработчиками принципов, положенных в основу метода RPC удаленного вызова процедуры: клиент обращается к локальному заместителю сервера (на этот выполненному в виде прокси-класса), затем сетевая служба вызывает метод, реализованный на удаленном сервере. При этом непосредственно к сервису обращается локальный заместитель клиента (прокси-класс).

***Модель SOA*** – это набор принципов и средств создания системного программного обеспечения и прикладных программных систем (ПС) из совместимых и унифицированных сервисов. К принципам SOA относятся:

1. Архитектура, которая не привязана к определённой технологии.

2. Независимость организации системы от используемой вычислительной платформы (платформ) и от применяемых языков программирования.

3. Использование сервисов независимо от конкретных приложений выполняется единообразным интерфейсом и является инструментом для построения систем из сервисов.

Унификация проявляется как типизация функциональности сервиса и его характеристик, а также языков описания сервисов и их взаимодействия. Объект SOA – сервис, который обладает специфицированной функциональностью (Function) и качеством (Quality service). Такие сервисы определены стандартами комитета W3C и имеют следующие уровни коммуникации:

- *транспортный уровень* (transport layer) для обмена данными;
- *уровень коммуникационного сервиса* (service communication layer) для определения высокоуровневых протоколов;
- *уровень описания сервиса* (service description layer) и связанных с ним





интерфейсов;

- *уровень бизнес-процессов* (business process layer) для реализации бизнес-процессов и потоков работ средствами сетевых служб;
- *уровень регистрации сервисов* (service registry layer) в реестрах сетевых служб для их публикации, поиска и вызова с использованием их WSDL интерфейсов.

Технология обеспечения качества сетевых служб имеет следующие уровни:

- *политика* (policy layer) для описания правил и условий применения сервисов;
- *безопасность* (security layer) для описания безопасности служб (авторизация, аутентификация и распределенный доступ);
- *транзакции* (transaction layer) для установки параметров обращения к сервисам и обеспечения надежности их функционирования;
- *управление* (management layer) службами.

Технологический фундамент сетевых служб составляют (рис.1):

- *набор языков* – XML, WSDL, BPREL, BPMN и др. для реализации базовых свойств сетевых сервисов, обеспечения их взаимодействия между собой в соответствующих средах (SOA, SCA и др.);
- *поставщик услуги*, который осуществляет ее реализацию в виде сетевой службы, прием и выполнение запросов пользователей, а также публикацию сведений о сервисе в соответствующем реестре;
- *реестр* (каталог) служб UDDI содержит в себе библиотеку сервисов для пользователей, а также средства их поиска и вызова с помощью запросов, которые поступают от поставщиков сервисов на получение сервисов; реестр служб оформляется как стандартная сетевая служба, адрес и другие параметры доступа к которой заранее известны поставщикам служб и их пользователям;
- пользователь или *потребитель сервиса* (приложение, программный модуль и др.), который осуществляет поиск и вызов необходимого сервиса из реестра описания сервисов, а также использует сервис, предоставленный провайдером в соответствии с заданным интерфейсом.

Связь между поставщиком и потребителем (рис. 1) осуществляется через HTTP и XML-сообщения сетевой среды, которая использует интерфейсы сетевых служб. Посредником между этими сервисами и приложениям является *провайдер*, который обеспечивает взаимодействие между поставщиками и потребителями с помощью средств описания и передачи сервисов WSDL, SOAP, XML.

Для получения сервиса в архитектуре SOA выполняются следующие операции:

1) публикация сервиса WSDL с целью обеспечения доступности (через вызов) пользователю сервиса и его интерфейса;
2) поиск сервиса в реестре с помощью протокола SOAP и заданных критериев;
3) связь с реестром UDDI через описание пользователем необходимого сервиса, который может предоставляться в таких моделях как DCOM, CORBA, DBMS, .NET и т. п.

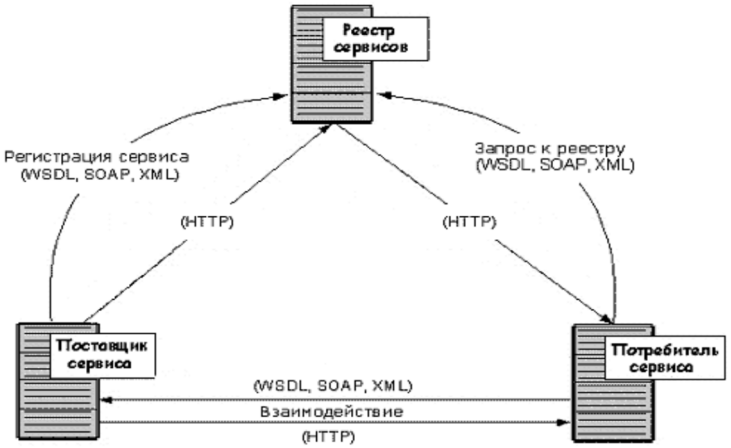


*Рис.1. Поставщики и потребители сервисов в сети.*

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 в ПС.





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.

Модель службы сервисов обеспечивает:

- динамическое расширение функциональности ПС за счет поиска и привлечения в сферу обработки новых сервисов;
- повышение масштаба и улучшение возможностей коммуникации между отдельными элементами системы за счет стандартного механизма подключения сервисов через их интерфейсы;
- повышение языкового уровня коммуникации системы с конечными пользователями.

Принципы разработки ПС из готовых компонентов повторного использования (КПИ), сетевых (и не только сетевых) служб, интерфейсов, данных, артефактов и т. п.:

- композиционность систем и ПС из компонентов, интерфейсов и сервисов с их свойствами и характеристикам, а также механизмами их композиции (агрегации) и правилами взаимодействия в интегрированных средах;
- компонентная инженерия (CBSE), как деятельность по созданию ПС из готовых "деталей" КПИ, базированная на системе классификации и каталогизации КПИ, средствах их унификации, стандартизации и интеграции их в ПС с помощью стандартного жизненного цикла (ЖЦ), начиная от процессов инженерии требований, разработки, эксплуатации и уничтожения ПС;
- интероперабельность КПИ и ПС, которая базируется на интерфейсах и стандартных правилах взаимодействия компонентов и сервисов между собой в процессе их интеграции для работы в разных гетерогенных средах;
- вариантность как способность ресурсов КПИ и сервисов служб к изменениям (удаление незавершенных функций или добавление

новых функциональных КПИ в конфигурационную структуру ПС и т. п.).

Сервис SOA – это функция, являющаяся четко определенной, самодостаточной и не зависящей от контекста или состояния других служб. Сервис определяется как единица работы, выполняемая от имени некоторого информационного субъекта-пользователя или программы. Клиент может избирать сервисы от разных поставщиков. Каждая служба может развиваться и быстро доставляться клиенту отдельно от других.

Главная идея сервисного подхода SOA состоит в том, чтобы создавать небольшие независимые компоненты (как сервисы) и собирать их в большой распределенный по глобальной сети комплекс (архитектуру) под 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 некоторой задачи.

## 2.3 Модель SCA (Service-Component Architecture) в IBM Sphere

Сервисно-компонентная архитектура (SCA, см. [11-12, 28]) предназначена для работы с компонентами, подчиняющимися самым разным спецификациям, разработанным различными компаниями: с компонентами EJB сервера приложений J2EE компании Sun Microsystems, с сетевыми сервисами, объектами планирования, компонентами доступа к базам данных, к информационной системе предприятия (Enterprise Information System, EIS) и др.

***Архитектура SCA*** обеспечивает доступ к сервисным компонентам и определяет зависимости между ними через аппарат ссылок. Компоненты SCA системы IBM WebSphere Integration Developer (инструментарий для разработки приложений на платформе Eclipse) могут быть упакованы в модуль для выполнения сервисного модуля с WebSphere Process Server – эквивалентного EAR-файлу J2EE и некоторым другим (рис.2). Подмодули J2EE и артефакты упаковываются с модулем SCA. Это позволяет запустить сервис через модель SCA и передавать данные при интеграции.

Механизмы, которые используются для вызова внешнего сервиса, называются импортом и экспортом. Они связаны с другими технологиями, такими как JMS, Enterprise JavaBeans или технологиями сетевых служб. SCA модуль позволяет обратиться к Enterprise JavaBean. Элементы SCA могут компоноваться и обмениваться данными друг с другом, пересылая сервисные объекты данных (Service Data Objects – SDO), 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.

Ключевые процессы для сложных приложений – разработка КПИ и внедрение SOA-сервисов и SCA. Для банковской сферы в IBM разработан набор компонентов в рамках портала WebSphere Portal (Process Server, WebSphere Service Registry and Repository, WebSphere Enterprise Bus, WebSphere Portlet Factory, WebSphere Application Server), а также соответствующие инструменты разработки новых приложений из сервисных компонентов и служб. Основные особенности предлагаемой технологии – наличие в ней и широкое использование точек изменчивости, вариантов использования и среды выполнения.

Каждое приложение состоит из одного или более компонентов или их объединений, связанных между собой интерфейсами сервисов. Для реализации бизнес-сервисов в конкретное приложение используются определенные бизнес-цели, которые облегчают разработку, применение и повторное использование в них готовых ресурсов.

Характеристики приложения можно согласовывать со специфическими потребностями клиента (то есть проводить конфигурирование приложения), используя такие элементы, как стандарты бизнеса, бизнес-правила, соглашения об уровне бизнес-сервиса и параметры конфигурации. Так, функция Dynamic Profiles WebSphere Portlet Factory обеспечивает высокую динамическую конфигурацию пользовательского интерфейса.

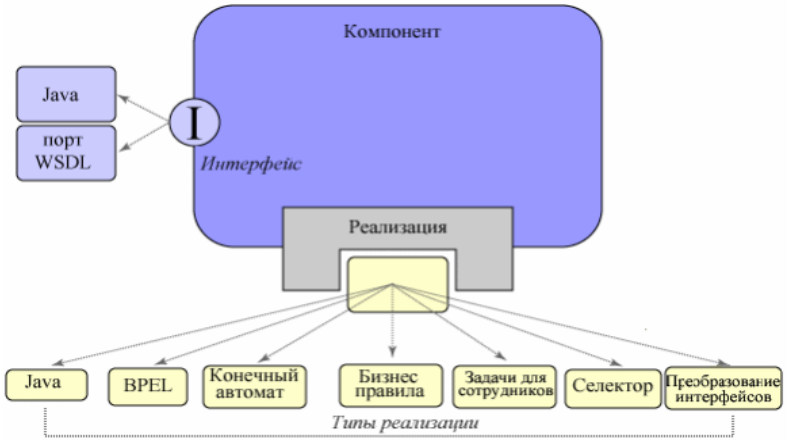


*Рис.2. Общая схема сервера SCA IBM.*

Сервисы уровня предприятия запускаются на узле сервисов предприятия, который содержит продукты, обеспечивающие сервисы данных, безопасность и другие службы инфраструктур, например, системный реестр служб. Сервер каталогов Tivoli (Tivoli Directory Server) обеспечивает инфраструктуру облегченного протокола доступа к каталогам (Lightweight Directory Access Protocol, LDAP), являющегося основой для управления идентификацией. Реестр WebSphere Service Registry and Repository дает возможность провайдерам регистрироваться, а клиентам – выбирать сервисы.

***Модель SCM*** представляет собой обобщение объектно-компонентной модели семейства программных продуктов (СПП, см. [1]). В ней каждый член является системой удаленных КПИ, 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 (SDO) и сервисов доступа к ним (DAS). Они позволяют размежевать код ПС и код доступа к обрабатываемым данным.
Модель SCM для реализации ПС представлена в виде:

$$SCM = \langle N, ID, MD(N), MI(ID), SR \rangle,$$

где $N$ – имя ПС,

$ID$ – идентификатор SCA;

$MD(N)$ – подмодель *абстрактных сервисов,* отображающая потребности ПС в функциях/данных на множестве бинарных отношений, которые задают взаимосвязи между абстрактными сервисами;

$MI(ID)$ – *интерфейсная* подмодель на множестве интерфейсов, которые реализуют абстрактные сервисы на множестве бинарных отношений и задают взаимосвязанные интерфейсы;

$SR = \langle R, RR \rangle$ – подмодель *сервисных ресурсов* реализации интерфейсов *MI* (*ID*) на множестве $R$ ресурсов и $RR$ бинарных отношений, которые задают взаимосвязи между этими ресурсов. Элементы множества $R$ – сервисы удаленных КПИ, сетевые службы и *композиты* (composites) SCA в виде объектов специального типа, которые соединяют свойства КПИ и сетевые службы.





## 2.4 Сетевые службы сервера приложений Java Enterprise Edition (Java EE)

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) описаны на специальном языке WSDL. Доступ к ресурсам осуществляется через протокол SOAP, который представляет собой XML-запросы, передаваемые посредством интернет-протокола относительно высокого уровня (HTTP, SMTP). Сетевые службы соответствуют объектам объектно-ориентированных языков программирования с некоторыми важными отличиями.
Ключевым понятием сетевой службы является сообщение (message), которое состоит из одной или нескольких переменных. Вместо методов классов в сетевых сервисах используются операции, которые определяются входным и выходным сообщениями. Далее приведен пример запроса, который вызывает операцию сервиса MyService.someMethod.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:q0 ="http://webservice.demo"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:someMethod>
      <q0:arg0>32.5</q0:arg0>
      <q0:arg1>true</q0:arg1>
    </q0:someMethod>
  </soapenv:Body>
</soapenv:Envelope>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <someMethodResponse xmlns="http://webservice.demo">
      <someMethodReturn>Nothing to see here</someMethodReturn>
    </someMethodResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Для описания общедоступных ресурсов сетевых сервисов в язык WSDL, построенный на синтаксической основе языка разметки XML, введены возможности описания данных следующих типов:

строка (xsd:string),
целые числа (xsd:int, xsd:long, xsd:short, xsd:integer, xsd:decimal),
числа с плавающей запятой (xsd:float, xsd:double),
логический тип (xsd:boolean),
последовательность байтов (xsd:base64Binary, xsd:hexBinary),
дата и время (xsd:time, xsd:date, xsd:g),
объекты (xsd:anySimpleType).

В качестве переменных для сообщений можно использовать последовательности, созданные из фиксированного количества переменных простых типов, причем в начале декларируются типы, которые будут использоваться в службе. Типичный WSDL-файл имеет такую примерную структуру.

```
<wsdl:definitions […]>
  <!-- Декларация типов 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>
        <sequence>
          <element name="someMethodReturn" type="xsd:
string"/>
        </sequence>
      </complexType>
    </element>
  </wsdl: types>
  <! -- Декларация сообщения -->
    <wsdl: message name="someMethodResponse">
      <wsdl:part element="impl:someMethodResponse"
name="parameters">
      </wsdl:part>
    </wsdl:message>
    <wsdl:message name="someMethodRequest">
      <wsdl:part element="impl:someMethod"
name="parameters">
      </wsdl:part>
    </wsdl:message>
```





```
    <!-- Декларации операций -->
      <wsdl:portType name="MyService">
        <wsdl:operation name="someMethod">
          <!-- Входные сообщения -->
          <wsdl:input message="impl:someMethodRequest"
                      name="someMethodRequest">
          </wsdl:input>
          <!-- Выходное сообщение -->
          <wsdl:output message="impl:someMethodResponse"
                       name="someMethodResponse">
          </wsdl: output>
        </wsdl:operation>
      </wsdl:portType>
    <!-- Декларация связи с SOAP -->
    <wsdl:binding name="MyServiceSoapBinding"
type="impl:MyService">
      <wsdlsoap:binding […]/>
      <wsdl:operation name="someMethod">
        <wsdlsoap:operation soapAction=""/>
        <wsdl:input name="fahrenheitToCelsiusRequest">
          <wsdlsoap:body use="literal"/>
        </wsdl:input>
        <wsdl:output name="fahrenheitToCelsiusResponse">
          <wsdlsoap:body use="literal"/>
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <!-- Декларация веб-сервиса -->
    <wsdl:service name="MyServiceService">
      <wsdl:port binding="impl:MyServiceSoapBinding"
name="MyService">
        <wsdlsoap:address location=[…]/>
       </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

Приведенное далее WSDL-описание принадлежит сетевой службе MyService с единственным методом *String someMethod(double arg0, boolean arg1)*. Для вызова метода сгенерированы два типа данных, которые соответствуют входным и исходным аргументам метода. Эти типы применяются в описаниях someMethodRequest и someMethodResponse – входного и выходного сообщения для операции someMethod. Операции декларируются в описании интерфейса службы (декларация wsdl:portType) и описание привязки службы к SOAP (декларация wsdl:binding). Причем во втором случае также оговаривается способ вызова (<wsdlsoap:body use="literal"/>). За счет этого разрешается при вызове операции использовать те же названия параметров, что и в методе класса. В конце WSDL-файла

находится декларация сетевой службы (<wsdl:service>), в которой содержится информация о расположении (параметр location).

На основе WSDL в среде Eclipse имеется возможность описывать клиенты сетевых служб в виде пяти Java-файлов:

- локатор сервиса (service locator), который выполняет нахождение веб-сервиса;
- интерфейс локатора;
- клиентский заместитель для SOAP-связи (SOAP binding stub) , предназначенный для составления и разбора SOAP–сообщения;
- интерфейс сервиса;
- серверный заместитель (прокси-класс), который реализует интерфейс; использующий клиентский заместитель и локатор для доступа к операциям службы.

Для сервиса MyService автоматически создаются классы и интерфейсы - MyServiceServiceLocator, MyServiceService, MyServiceSoapBindingStub, MyService, MyServiceProxy (в порядке их описания). Обращение к операции MyService.someMethod выглядит таким образом:

```
My Service svc = new MyServiceProxy ();
try { System.out.println (svc.someMethod (32.5, true));
}
catch (Remote Exception e) {
      System.err.println("Error occurred while accessing web
service");
      e.printStackTrace();
}
```

## 2.5 Сервисы IContact WCF MS.NET

Программная среда Windows Communication Foundation (WCF) используется для обмена данными, которые входят в состав среды .NET Framework [29-30]. Она является логическим развитием технологий сетевых служб, .NET Remoting и DCOM.

В основе WCF лежит архитектура SOA. Тем самым предполагается, что на стороне сервера работает некоторое количество сервисов, которые представляют собой группу операций, определенных в некотором интерфейсе, и которые получают абстрактные входные/выходные параметры. Все это описывается на языке WSDL и может быть сделано доступным удаленным клиентским приложениям через порты обмена метаданными (Metadata Exchange Endpoints – mex-endpoints). Это позволяет получить "целевые данные" служб путем подключения к этому интерфейсу, а также получить описания служб и всех их операций через серверные заместители (прокси-классы), специфические для заданного языка или платформы. Клиенты в свою





очередь имеют на своей стороне клиентские заместители, которые содержат ссылки на соответствующие операции на серверной стороне. Таким образом, вызов локальных методов прокси-класса приводит к вызову методов соответствующей службы.

В составе программной среды WCF MS.NET имеется технология сетевых служб, а также фабрика сервисов. Основу технологии составляют схемы, “рецепты”, методы и средства построения фабрик разного назначения. Фабрика сервисных программ в WSF включает набор полезных ресурсов, блоков кода, документации, образцы приложений, автоматизированные инструменты и пакеты Visual Studio Industry Partners (VSIP) для создания на их основе программных комплексов для нестандартных аппаратных устройств. Компания Microsoft создала набор рекомендации, схем и методов, а также стандарты их выполнения разработчиками готовой продукции.

Фабрика сервисов предоставляет рекомендации для использования сервисов при проектировании и конструировании некоторого приложения, которые накапливаются в хранилище Global Bank. К ним относятся ASP.NET для использования в WCF. Данные фабрики программ и сервисов базируются на готовом наборе программных элементов и разного рода сервисов, специфических для MS.NET (см. [1, 2]).

В основе функционирования программной среды WCF лежат так называемые конечные точки (endpoint), что составляет связь "Address – Binding – Contract" ("ABC"). Каждая составляющая играет важную роль в определении конечной точки. Компонент "Address" содержит указание места расположения конечной точки (порта). Адрес может быть как абсолютным, так и относительным (относительно базового адреса). Компонент "Binding" задает привязку и, фактически, определяет транспортный протокол, на основе которого будет происходить взаимодействие. В модели WCF определен ряд классов-привязок, например, BasicHttpBinding (простая привязка на основе HTTP), NetTcpBinding (привязка на основе транспорта TCP ) и т.д.

Компонент "Contract" задает контракт, на основе которого будет происходить взаимодействие клиента и сервера. Фактически, определение контракта регламентирует операции для оказания серверной услуги. На основе контракта на стороне клиента строится прокси-класс. Интерфейс *Icontract* содержит описание атрибутов и операций передачи данных от одного сервисного объекта клиента (*Service consumer*) к другому (*Service provider*). Их описание задается в языке XML. Передача интерфейсов между ними выполняет протокол, в котором задаются атрибуты и операции интерфейса. Интерфейс *Icontract* задается сообщениями вида:

1. сервис операций, вызываемых клиентом;
2. фундаментальных данных (int, float, string и др.) для передачи их через сервисные службы;

3. об ошибках, которые могут содержаться при передаче контрактов клиенту и обратно;
4. операций прямого взаимодействия объектов между собой.

Эти сообщения задаются в языке XML с помощью протокола SOAP в конверте (рис.3) следующего вида:

```
<?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>
```

В конверте передаются параметры для обмена данными. Через них происходит взаимодействие между пользователем и провайдером. Параметры могут содержать ошибки, которые приводят к возникновению разного рода конфликтов в сетевой среде следующего вида:

1. Несовместимость переданных типов данных в контрактных интерфейсах (например, тип "целое", а параметр описан как "символьный") или некорректное описание некоторых типов данных в компонентах на ЯП.
2. Различие в порядке задания параметров в протоколе передачи информации между сервисными объектами.
3. Различие в архитектуре платформ объектов клиента и сервера или в конфигурационных файлах взаимодействующих систем.

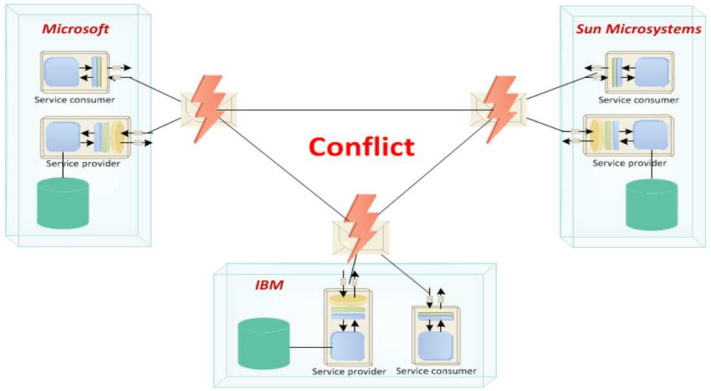


*Рис. 3. Схема взаимодействия систем в Интернет и возможность возникновения конфликтов.*





Снятие таких конфликтов решается специальными сервисными средствами взаимодействующих систем, расположенных на серверной стороне и отмеченных конечными точками. Клиент в этом случае создает соединение с той конечной точкой, которая требуется именно ему. Пример схемы соединения сервиса и клиентов показан на рис. 4.

Архитектура WCF для удобства построения распределенных приложений следует принципу "слоенного" пирога, когда каждый слой отвечает за свой конкретный уровень абстракции и не знает нижележащие уровни. Инфраструктура WCF состоит из двух главных уровней: уровня серверной модели (Service Model Layer) и канального уровня (Channel Layer). Первый уровень относится и к самому серверу, и к клиенту, он отвечает за преобразование метода и его параметров в сообщение для передачи более низкому канальному уровню. Канальный уровень инкапсулирует в себе множество каналов передачи данных, использующих транспортные протоколы TCP, HTTP, Named Pipes и т. д. Каждый из этих уровней содержит подуровни, в которые может войти любой пользователь.

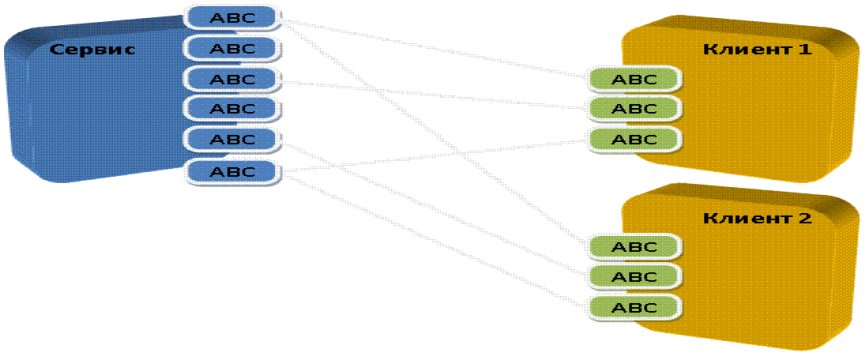


*Рис.4. Пример соединения сервиса и клиентов.*

Контракты в WCF представляют собой описания сообщений, переданных конечным службам для получения от них ответа. Конечная точка специфицирует операции, которые она может выполнять, и формат ожидаемых данных. Совокупность этих спецификаций и есть контракт.
В WCF содержится три вида контрактов:

- функциональные операции, реализованные сервером. Внутри контракта сервера имеются контракты на операции, которые описывают сервисы, реализующие требуемые функции;
- формат данных, которыми сервисы будут обмениваться, относится как к запросу на сервис, так и к октету сервиса. Если используются примитивные типы данных – целые (int), строки (string) и им подобные, то контракт не требуется, потому что

среда .NET изначально позволяет работать с этими типами данных;

- контроль заголовка SOAP

Чтобы контракты были интероперабельными для широкого диапазона систем, они должны описываться на языке WSDL. Однако на практике контракты описываются на языках WSDL и XSD, а программа обычно работает с типами данных библиотеки CLR, для отображения одной системы типов в требуемую другую. В WCF эта задача решается в три этапа. Сначала при написании кода сервиса поставляется класс, определенный в WCF атрибутами [ServiceContract], [OperationContract], [FaultContract], [MessageContract] и [DataContract]. После создания текста клиентской программы следует запрос к сервису контракта. Утилита svcutil.exe вызывает инфраструктуру конечной точки сервера, чтобы ответить передачей данных, необходимых для генерации WSDL документа с атрибутами. На этапе выполнения метода и определенного в интерфейсе сервиса, WCF сериализует типы CLR и вызов в формате XML, а затем посылает сообщение в сеть для привязки к схеме через WSDL. В этом процессе участвуют четыре конструкции: две со стороны .NET и две со стороны XML. Со стороны .NET имеется тип CLR, который определяет структуры данных и функциональные возможности. После того, как будет создан объект этого типа со стороны XML (XML Instance), сообщение начинает существовать.

**Пример создания сервиса WCF в MS.** Для наглядности дается пример работы сервиса калькулятора с функциями «+», «-», «*» и «/», выполненного студентом МФТИ (см. [1] и раздел «взаимодействие» на сайте http://sestudy.edu-ua.net).

**Использование служб WCF**

1. В меню Файл надо выбрать пункт «Создать», а затем команду «Проект».
2. В диалоговом окне «Новый проект» развернуть узел Visual Basic или Visual C# и WCF, Библиотека службы WCF. Нажать кнопку «ОК» для открытия проекта.
3. В обозревателе решений дважды щелкнуть файл IService1.vb или IService1.cs и найти следующий интерфейс IService1.
4. Добавить следующие коды:

```
[OperationContract()]
int Add(int, int b);
[OperationContract()]
int Multiply(int, int b);
[OperationContract()]
int Divide(int, int b);
[OperationContract()]
int Substract(int, int b);
```





5. В обозревателе решений дважды щелкнуть файл Service1.cs и найти Service1 и добавить коды:

```
public int Add(int, int b) {
  return + b;
}
public int Multiply(int, int b) {
  return * b;
}
public int Divide(int, int b) {
  return / b;
}
public int Substract(int, int b) {
  return - b;
}
```

**Доступ к службе WCF**

1. В меню Файл последовательно выбрать пункты «Добавить» и «Новый проект».
2. В диалоговом окне «Новый проект» развернуть узел Visual C#, выбрать пункт Windows и элемент «Добавление» Windows Forms. Нажать кнопку «ОК» для открытия проекта.
3. Щелкнуть правой кнопкой мыши WindowsApplication1 и выбрать ссылку «Добавить». Появится диалоговое окно и надо добавить ссылку на службу.
4. В диалоговом окне «Добавить» выбрать «Найти», появится Служба1.
5. Нажать кнопку «ОК» для добавления этой ссылки на службу.

**Построение клиентского приложения**

1. Если конструктор Windows Forms еще не открыт, дважды необходимо щелкнуть файл Form1.cs в обозревателе решений задач и открыть его (рис. 5).
2. На панели элементов перетянуть в форму следующие элементы управления: 2 TextBox, Label и 4 Button. Переименуйте кнопки как показано на рис 4
3. Для каждой кнопки добавить обработчик события Click. Для этого по каждой кнопке щелкнуть два раза.

## 2.6 Принцип реализации сетевых служб в ИТК

Сетевая служба ИТК [3] идентифицируется программой (сложение и вычитание чисел) из символов URI, свойства и методы которой описаны в языке WSDL [1]. Доступ к ресурсам осуществляется через протокол SOAP,

который представляется XML-запросами, передаваемые HTTP Интернет–протоколом. Сетевые службы близки классам сервера приложений Java Enterprise Edition. Ключевым понятием сетевой службы является сообщение из одной или нескольких переменных. Методы классов задаются операциями с входным и выходным значениями сообщений. После вызова операции переменной входного сообщения протокола SOAP, интерпретируются как параметры соответствующего метода класса, который лежит в основе службы. После завершения работы метода формируется исходное сообщение, которое содержит возвращенное методом значение, после чего оно отправляется клиенту по протоколу SOAP. Сформулированные принципы и понятия создания распределенных приложений с использованием сервисно-компонентных архитектур представлены в комплексе ИТК в виде готовых КПИ и операций их реализации.

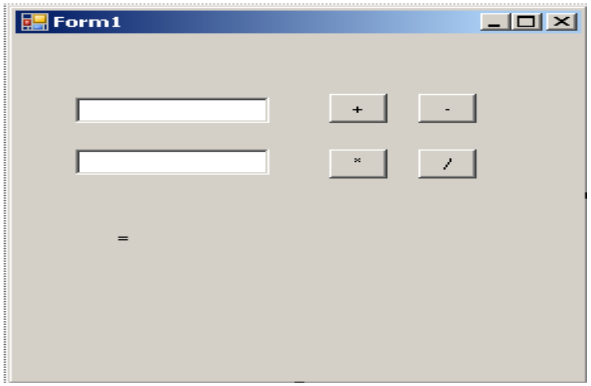


*Рис.5. Вид клиентского приложения сервиса калькулятора*

## Список литературы

[1]. Лаврищева Е. М. Software Engineering компьютерных систем. Парадигмы, Технологии, CASE-средства программирования. – К.: Наук. Думка, 2014 – 284 с.
[2]. Lavrischeva, E.: Formal Fundamentals of Component Interoperability in Programming. In: Cybernetics and Systems Analysis, vol. 46, no. 4, pp. 639–652. Springer, Heidelberg (2010), http://link.springer.com/article/10.1007%2Fs10559-010-9240-z
[3]. Лаврищева Е. М., Зинькович В. М., Куцаченко Л. И. и др. Инструментально-технологический комплекс для разработки и обучения приемам производства программных систем. Госслужба интеллектуальной собственности Украины. – Свидетельство о регистрации №45292 от 27.08.2012. –108 с. (украинский).
[4]. Andrew S. Tanenbaum, Maarten van Steen. "Distributed Systems. Principles and paradigms". Prentice Hall, Inc., 2002 (Таненбаум Э., ван Стеен М.. "Распределённые системы. Принципы и парадигмы". СПб.: Питер, 2003 – 878 с.)
[5]. Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju. Web Services. Concepts, Architectures and Applications. Springer-Verlag, 2004





[6]. Карпов Л. Е. Архитектура распределенных систем программного обеспечения – М., МАКС Пресс, 2007. – 130 с.
[7]. 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
[8]. http://www.corba.org/
[9]. Jon Siegel. "Quick CORBA™ 3". Wiley Computer Publishing, John Wiley & Sons, Inc., 2001 (Джон Сигел, "CORBA 3", М., МАЛИП, 2002).
[10]. http://www.w3.org/TR/2008/REC-xml-20081126/
[11]. Гладцын В. А., Кринкин К. В. Яновский В. В. Сервис-ориентированная архитектура: стандарты, алгоритмы, протоколы – Санкт-Петербург: СПб ГЭТУ ЛЭТИ, 2006 – 108 с.
[12]. 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.
[13]. http://www.w3.org/TR/soap/
[14]. http://www.w3.org/TR/soap12-part1/
[15]. http://www.w3.org/TR/wsdl20
[16]. http://www.w3.org/RDF/
[17]. http://www.w3.org/TR/wsci/
[18]. http://www.w3.org/TR/wscl10/
[19]. http://www.omg.org/spec/BPMN
[20]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[21]. http://www.oasis-open.org/specs/index.php#wsbpelv2.0
[22]. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm
[23]. http://www.oasis-open.org/specs/index.php#uddiv2
[24]. http://www.oasis-open.org/specs/index.php#uddiv3
[25]. http://www.oasis-open.org/specs/index.php#uddiv3.0.2
[26]. http://www.omg.org/spec/UML/ISO/19505-1/PDF
[27]. http://www.omg.org/spec/UML/ISO/19505-2/PDF
[28]. http://www.ibm.com/developerworks/websphere/techjournal – IBM WebSphere Developer Technical Journal.
[29]. Архитектура IT-ландшафта на базе корпоративных сервисов. Разработка маршрутной карты – 2011 – 16 с., http://www1.sap.com/cis/pdf/ESARoadmap.pdf.
[30]. http://www.ivk.ru/po/upiter/

# Internet services for solving business problems

*E. Lavrischeva <lavr@ispras.ru>*
*L. Karpov <mak@ispras.ru>*
*A. Tomilin <tom11@bk.ru>*
*Institute for System Programming of the Russian Academy of Sciences,*
*25, Alexander Solzhenitsyn str., Moscow, 109004, Russia.*

**Abstract**. Different types of global information network services, used in the modern distributed software systems are described. Several approaches to service description and to service interaction are shown. Description of wide-spread and inculcated approaches to creation of distributed object interaction is given in the wide practice of the systems with spectrum of system and functional services. Web-service international standards stack (SOAP protocol, WSDL language, UDDI interface and service, XML, BPEL and BPMN notations, CORBA object broker and services, JEE application server) is briefly described. Models of web services, service-oriented approach (SOA), service-component architectures (SCA), and Windows Communication Foundation (WCF) service support application systems are considered for presentation of the business systems by the services ready to the decision of business-tasks. Features that support interoperability and multilanguage interaction are underlined. Principles of distributed software design (service composition, component engineering, interoperability, variance) that are of special care in SOA are listed. Some examples of presenting information using standard notations are given. An additional example of calculator service with functions of the data processing in ITK service-environment (WCF-based system) and the concept of web service implementation are presented.

**Keywords**: distributed system; worldwide network; models of service architectures; network service; co-operation protocols; languages of services and interconnection; system business.

**DOI:** 10.15514/ISPRAS-2015-27(1)-7

**For citation:** Lavrischeva E., Karpov L., Tomilin A. Internet services for solving business problems. *Trudy ISP RAN/Proc. ISP RAS*, vol. 27, issue 1, 2015, pp. 125-150 (in Russian). DOI: 10.15514/ISPRAS-2015-27(1)-7

## *References*

[1]. Lavrishcheva E. M. Software Engineering komp'yuternykh sistem. Paradigmy, Tekhnologii, CASE-sredstva programmirovaniya. [Software Engineering for computer systems. Paradigms, Methods, CASE technology]– K.: Nauk. Dumka, 2014 – 284 pp. (in Russian).

[2]. Lavrischeva, E. Formal Fundamentals of Component Interoperability in Programming. In: Cybernetics and Systems Analysis, vol. 46, no. 4, pp. 639–652. Springer, Heidelberg (2010), http://link.springer.com/article/10.1007%2Fs10559-010-9240-z

[3]. Lavrishcheva E. M., Zin'kovich V. M., Kutsachenko L. I. et al. Instrumental'no-tekhnologicheskii kompleks dlya razrabotki i obucheniya priemam proizvodstva programmnykh sistem [Instrumental technology for design and teaching procedures of prgramming system creation]. Gossluzhba intellektual'noi sobstvennosti Ukrainy [Ukraine state service for intellectual property support]. – Svidetel'stvo o registratsii [Registry sertificate] #45292 27.08.2012. – 108 p. (in Ukrainian).

[4]. Andrew S. Tanenbaum, Maarten van Steen. "Distributed Systems. Principles and paradigms". Prentice Hall, Inc., 2002

[5]. Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju. Web Services. Concepts, Architectures and Applications. Springer-Verlag, 2004

[6]. Karpov L. E. Arkhitektura raspredelennykh sistem programmnogo obespecheniya [Distributed software systems architecture] – Moscow, MAKS Press, 2007, 130 p.





[7]. Karpov L. E., Yudin V. N. Obmen dannymi v raspredelennoi sisteme podderzhki reshenii [Data exchange in distributed software system for decision support]. Trudy ISP RAN [The Proceedings of ISP RAS], 2010, vol. 19, pp. 71-80 (in Russian) http://www.ispras.ru/ru/proceedings/docs/2010/19/isp_19_2010_71.pdf

[8]. http://www.corba.org/

[9]. Jon Siegel. "Quick CORBA™ 3". Wiley Computer Publishing, John Wiley & Sons, Inc., 2001.

[10]. http://www.w3.org/TR/2008/REC-xml-20081126/

[11]. Gladtsyn V. A., Krinkin K. V. Yanovskii V. V. Servis-orientirovannaya arkhitektura: standarty, algoritmy, protokoly [Service-oriented architecture: standards, algorithms, protocols] – Sankt-Peterburg: SPb GETU LETI, 2006 – p. 108 (in Russian).

[12]. 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.

[13]. http://www.w3.org/TR/soap/

[14]. http://www.w3.org/TR/soap12-part1/

[15]. http://www.w3.org/TR/wsdl20

[16]. http://www.w3.org/RDF/

[17]. http://www.w3.org/TR/wsci/

[18]. http://www.w3.org/TR/wscl10/

[19]. http://www.omg.org/spec/BPMN

[20]. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[21]. http://www.oasis-open.org/specs/index.php#wsbpelv2.0

[22]. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm

[23]. http://www.oasis-open.org/specs/index.php#uddiv2

[24]. http://www.oasis-open.org/specs/index.php#uddiv3

[25]. http://www.oasis-open.org/specs/index.php#uddiv3.0.2

[26]. http://www.omg.org/spec/UML/ISO/19505-1/PDF

[27]. http://www.omg.org/spec/UML/ISO/19505-2/PDF

[28]. http://www.ibm.com/developerworks/websphere/techjournal – IBM WebSphere Developer Technical Journal.

[29]. Arkhitektura IT-landshafta na baze korporativnykh servisov. Razrabotka marshrutnoi karty [IT landscape architecture based on corporate services. Building a roadmap] – 2011 – pp. 16, http://www1.sap.com/cis/pdf/ESARoadmap.pdf (in Russian).

[30]. http://www.ivk.ru/po/upiter/