Принципы стандартизации программного обеспечения ГРИД[1]

В.Н. Коваленко, Е.И. Коваленко, Л.В. Ухов

Аннотация. Статья посвящена анализу основополагающих принципов организации программного обеспечения грида, представленных в Открытой архитектуре грид-служб (OGSA). На современном этапе разработка программного обеспечения грида[2] осуществляется различными коллективами и наличие общепризнанных архитектурных решений создает необходимые условия для создания открытой и расширяемой программной платформы, реализующей типовые функции для работы с множествами распределенных ресурсов. В статье рассмотрены мотивы использования в OGSA ориентированных на службы архитектурных подходов (SOA и Web-служб), а также ряд новых стандартов, которые отражают особенности грида по работе с распределенными ресурсами.

Ключевые слова: грид, архитектура программного обеспечения, служба, ресурс, стандарт, WSRF.

Введение

Стандартизация полезна во многих сферах, но в области распределенных систем она является необходимостью. В полной мере это относится и к рассматриваемому в данной работе направлению *грид*, в котором ставится задача создания аппарата для работы с множеством пространственно-распределенных *ресурсов* [1]. Назначение этого аппарата - поддержка дистанционного выполнения операций, которые соответствуют типу ресурсов и обычно применяются к ним в локальной среде одного компьютера. Семантика этих операций прямо отражает конкретную функцию работы с ресурсами, например, запуск программ, передачу файлов или получение информации от приборов наблюдения.

Постановка задачи доступа к распределенным ресурсам в гриде характеризуется общностью. Во-первых, подход грида рассчитан на различные типы ресурсов: аппаратные (компьютеры, устройства хранения данных, источники данных) и ресурсы данных (файлы, структурированная информация). Во-вторых, в рамках одного типа ресурсы могут быть гетерогенными: размещаться на разных компьютерных платформах, управляться разными системами, использовать разные способы представления данных. С учетом этих условий предложен общий принцип обеспечения доступа, состоящий в том, что ресурсы интегрируются в распределенную инфраструктуру – грид - путем установки дополнительного слоя *программного обеспечения грида* (ПГО, middleware). Это позволяет работать с удаленными ресурсами посредством стандартизированных и не зависящих от внутренней организации ресурсов операций, а ПГО выполняет их преобразование в реальные операции, которые поддерживаются управляющими системами ресурсов.

В созданных на начальном этапе развития грида комплексах ПГО работа с ресурсами была унифицирована на основе частных решений, и в результате эти комплексы оказывались несовместимыми друг с другом. Рассматриваемая

[1] Работа выполнена при поддержке фонда РФФИ (грант № 09-07-00335-а), программы фундаментальных исследований Президиума РАН, гранта Президента РФ для ведущих научных школ НШ-2139.2008.9.

[2] Здесь и далее по тексту авторское написание термина.

в данной статье деятельность по разработке общих стандартов открыла возможность создания *программной платформы грида*, в которой реализуются базовые функции работы с множествами распределенных ресурсов разных типов. Совокупность спецификаций об устройстве платформы и заложенных в ней функциях получила название *Открытая архитектура грид-служб* (Open Grid Service Architecture – OGSA).

Проблема выработки программной архитектуры грида была впервые сформулирована в [2] и развивается в серии работ, запланированных в [3]. Современное представление об OGSA, изложенное в работе [4], имеет статус информационного документа, который одобрен организацией Open Grid Forum (OGF) [5].

Руководящим принципом для выбора архитектуры грида является необходимость наделения его платформы двумя свойствами: открытости и расширяемости. Свойство открытости означает, что платформа должна предоставлять программные интерфейсы для обращения к реализуемым в ней функциям из приложений. Платформа также должна быть ориентирована на перспективу, обладая свойством расширяемости состава собственных функций.

Разработка архитектуры грида ведется с учетом архитектурных решений других подходов распределенного компьютинга, в том числе RMI, CORBA, DCOM, JINI. Принимались во внимание и практический опыт создания вычислительных гридов, и современные тенденции разработки производственных приложений. Развитие OGSA, которым в рамках OGF руководит рабочая группа OGSA-WG, направлено на создание серии нормативных документов, специфицирующих способ организации и функциональные возможности платформы грида.

Базовый документ [4] выделяет два аспекта архитектуры OGSA: структурный и функциональный. Структурный аспект стандартизирует форму компонент платформы и способы взаимодействия с ними. Здесь главное положение состоит в том, что OGSA основывается на Архитектуре Web-служб – WSA [6], используя набор ее спецификаций. Тем самым компоненты платформы грида представляют собой Web-службы, а реализуемые ими функции доступны извне посредством формально описанных интерфейсов. Поддержка функционирования Web-служб осуществляется частью платформы - *инфраструктурой служб OGSA* (OGSA Infrastructure). В то же время OGSA дополняет архитектуру WSA спецификациями для работы с ресурсами, внося, таким образом, существенный и ориентированный на грид вклад в ее развитие.

Потребность в стандартизации функционального аспекта архитектуры вытекает из задачи грида как средства интеграции множества ресурсов и обеспечения к ним технологичного доступа. Набор служб платформы может развиваться, но расширяемость платформы, тем не менее, предполагает определенность ее состава. Сложившийся состав установлен на основе анализа различных предметных областей [7] и включает следующие классы служб:

- службы инфраструктуры;
- службы обеспечения технологических свойств;
- службы интеграции компьютерных ресурсов, ресурсов данных, информационных ресурсов;
- службы управления обработкой;
- службы управления ресурсами.

Дальнейшее изложение построено следующим образом. В части 1 рассматривается структурный аспект архитектуры OGSA: принципы организации ПГО, на основе которых программная платформа грида может развиваться эволюционно путем объединения компонент, решающих отдельные задачи. В части 2 демонстрируется продуктивность применения этих принципов для создания важнейшей функциональной составляющей платформы – инфраструктуры безопасности.

1. Стандартизация формы программного обеспечения грида

Для грида, как и для любой другой распределенной системы, ключевой архитектурный вопрос - обеспечение *интероперабельности*, то есть возможности взаимодействия между любыми программными компонентами ПГО и приложений, независимо от платформ хостинга, языка реализации и программной среды, в которой они выполняются [2].

На начальном этапе развития грида известный уровень интероперабельности был достиг-

нут за счет унификации протоколов доступа к ресурсам. Был предложен многоуровневый стек, базирующийся на интернет-протоколах TCP/IP и заканчивающийся уровнем коллективных протоколов (анализ этой архитектуры дан в [8]). Наиболее важным результатом стала спецификация протоколов доступа к ресурсам: GRAM (управление заданиями), GridFTP (передача файлов), GRIP (управление информацией), GSI (безопасность). Эти протоколы были реализованы в первых двух версиях инструментального комплекса Globus Toolkit [9], причем на основе второй версии создано большинство современных гридов.

Такой унификации оказывается, однако, недостаточно: обеспечивается возможность работы с множеством ресурсов в рамках одной системы ПГО, но системы различных разработчиков оказываются несовместимыми друг с другом. Причина в том, что протоколы задают правила сетевого взаимодействия, а конкретные операции (создание процессов, перемещение файлов) предоставляются потребителям посредством интерфейсов программных компонент-служб, реализующих протоколы. Поэтому для интероперабельности разных систем ПГО требуется не только стандартизация протоколов доступа к ресурсам, но, кроме того, и стандартизация служб.

В проблеме интероперабельности служб можно выделить две части:

- взаимодействие со службами должно осуществляться посредством стандартизированных интерфейсов (в дополнении к платформенно-нейтральным протоколам);
- для обеспечения возможности использования служб в различных системах должен быть определен универсальный способ описания интерфейсов служб, который допускает отображение в конкретные конструкции разных языков программирования.

Усилия по обеспечению интероперабельности ПГО по существу совпадают с современными тенденциями индустрии прикладного программирования, которые также направлены на декомпозицию монолитных, ориентированных на фиксированные ресурсы приложений и представление их в виде множества взаимодействующих по сети служб. Как для ПГО, так и для приложений весьма важно обеспечить композицию служб, которая в перспективе должна стать основной формой разработки распределенных систем.

В промышленных системах архитектура служб существует на протяжении последнего десятилетия. Достаточно широкое распространение получили несколько программно-инструментальных сред: Sun ONE [10], J2EE [11], Websphere [12] и .NET [13], которые поддерживают разработку и функционирование ориентированных на службы приложений. Однако во всех этих средствах проблема интероперабельности решается ограниченно - в рамках одного языка (Java, C#) или платформы (Sun, IBM, Microsoft).

В то же время развита теоретическая база, выраженная в архитектурах SOA (Service Oriented Architecture) и Web-служб, которая решает существенно более сложную задачу обеспечения интероперабельности между службами независимо от языка реализации, программной модели (процедурная, объектно-ориентированная), платформы установки и исполнительной среды. Опора на архитектуру Web-служб рассматривается как наиболее эффективный путь развития грида, позволяющий рассчитывать, что принятые решения будут широко восприняты, станут индустриальными стандартами и расширят сферу применения грида с научно-технических на производственные приложения.

1.1. Ориентированная на службы архитектура (SOA)

Мотивы и основополагающие принципы разработки распределенных систем в виде взаимодействующих служб берут начало в концепции Ориентированной на службы архитектуры SOA (Service Oriented Architecture) [14]. SOA предлагает принципы проектирования хорошо структурированных модульных приложений, которые могут быть динамически образованы из распределенно функционирующих компонент – служб, причем одна и та же служба может использоваться в составе многих приложений.

Для фундаментального понятия SOA–службы известно множество различных определений, воспользуемся следующим [15]:

Служба (service) – это программный объект, который реализует некоторую совокупность функций и предоставляет их по сети в соответствии с формально описанными интерфейсами, определяющими схему обмена сообщениями для обращения к функциям.

Устройство типичной службы представлено на Рис. 1. Она состоит из аппарата обработки сообщений, прикладной логики и ресурсов. Поступающие службе сообщения обрабатываются в соответствии с прикладной логикой, в процессе обработки при необходимости используются ресурсы.

Подход SOA формулирует ряд принципов, среди которых важнейшими представляются: интероперабельность, слабая связанность и обнаруживаемость.

Интероперабельность обеспечивается формальным определением интерфейсов службы в виде схемы обмена сообщениями между службой и обращающимися к ней клиентами. Способ определения не зависит от внутреннего устройства службы, включая платформу функционирования, язык и способ ее реализации. Сообщения передаются в платформенно нейтральном, стандартизованном формате. Благодаря этому, в одном приложении совместно со службами, разработанными на C# и работающими на платформе .Net, могут использоваться и Java-службы на платформе J2EE. Точно также множество разноплатформенных приложений могут использовать одну службу. Одно из важных достоинств абстрагирования от способа реализации – возможность создания служб на базе наследуемых систем путем заключения их в оболочку обработки сообщений.

Слабая связанность (loose coupling) предписывает минимизацию взаимозависимости служб. Каждая служба реализуется независимо от всех остальных, выполняя автономный набор функций. Слабая связанность способствует тому, что взаимодействие служб в SOA ориентировано на обмен сообщениями, а не на наличие постоянного соединения. В общем случае обмен сообщениями осуществляется асинхронно, так что клиент, обращающийся к службе, может продолжать свое выполнение, не дожидаясь ответа, время получения которого может быть длительным и сильно варьироваться.

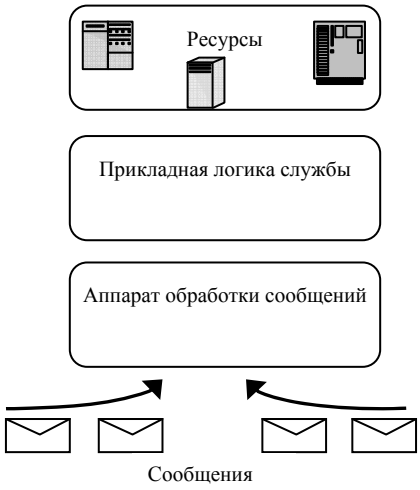

Рис. 1. Основные составляющие службы

Связи между службами устанавливаются динамически посредством механизма обнаружения (discovering), который основан на описании службы в такой форме и таким способом, чтобы потенциальный потребитель мог узнать о ее существовании и возможностях. Помимо формального определения интерфейсов описывается также семантика служб в форме метаданных, пригодных для компьютерной обработки.

1.2. Архитектура Web-служб

SOA представляет собой концептуальную модель архитектуры распределенных систем и допускает воплощение с помощью различных программных моделей, например, с применением такие технологии распределенного компьютинга, как CORBA и DCOM. В архитектуре грида - OGSA реализация SOA осуществляется на основе технологий WSA. Если SOA служит аппаратом проектирования распределенных систем, то WSA, предлагая не только стандарты, но и конкретный программный аппарат, становится средством реализации распределенных систем. В то же время, конкретизируя SOA, архитектура Web-служб проводит в жизнь свойства этого подхода: независимость от платформы, слабую связанность и т.д.

Необходимо подчеркнуть, что, в отличие от DCE, CORBA и Java RMI, WSA базируется на широко распространенных стандартах Интер-

нета (eXtensible Markup Language - XML [16], Simple Object Access Protocol - SOAP [17]), которые применимы в условиях гетерогенных распределенных сред. WSA находится в стадии развития – разработка спецификаций продолжается, на сегодняшний день их общее число превышает 40 наименований [18]. Часть из них принята в качестве стандартов (стандартизацией WSA занимается главным образом организация W3C), но до сих пор появляются новые предложения. Однако уже современное состояние развития WSA служит основой для нескольких платформ программного обеспечения: .NET (Microsoft), Dynamic e-Business (IBM), Sun ONE (Sun).

Количество спецификаций, определяющих WSA, выглядит обескураживающе большим. Этому, однако, есть объективная причина: при разработке различных распределенных систем должны учитываться разнообразные факторы (безопасность, транзакционность, надежность обмена), причем WSA позволяет это сделать разными способами. В связи с этим, важное значение имеет принцип композиционности (composability), который выдерживается при проектировании спецификаций. Его можно выразить в следующих положениях.

- Каждая спецификация прямо адресована конкретному свойству и имеет самостоятельное значение.

- Спецификации разрабатываются таким образом, чтобы их можно было комбинировать для совместного использования.

- Комбинирование спецификаций обеспечивается благодаря соответствующему устройству базовых спецификаций WSA - SOAP и WSDL, которые поддерживают эту возможность.

Достоинства этого подхода можно проиллюстрировать на примере такой фундаментальной конструкции, как сообщение. В WSA сообщение имеет регулярную структуру с любым количеством частей - элементов. Новое свойство может быть просто добавлено к сообщению, причем это не меняет обработку остальных элементов. Так, например, можно независимо добавить идентификаторы транзакций или порядковые номера сообщений, задействуя свойство надежного обмена. Оба расширения не будут конфликтовать друг с другом и совместимы с изначальной структурой сообщения.

Благодаря принципу композиционности разработчикам приложений требуется освоить лишь те спецификации WSA, которые существенны для их задач, так что сложность решения зависит только от проблемы и “избыток” технологий не является помехой.

Ядро WSA составляют две спецификации, использование которых является обязательным: описания интерфейсов Web-служб и протокола обмена сообщениями. Независимость этих, так же как и других средств WSA, от платформ и прочих реализационных аспектов обеспечивается использованием языка XML и связанных с ним стандартов: XML Information Model [19], XML Base и XML Schema [20].

Интерфейсы Web-служб описываются посредством языка Web Services Description Language (WSDL) [21], производного от XML. WSDL-описание выступает в роли метаданных и применяется для разработки обращений к службе, а также для ее обнаружения.

В качестве протокола обмена сообщениями выступает SOAP, который выполняет функции упаковки/распаковки при передаче сообщений между посылающей и принимающей стороной. Применение SOAP имеет два преимущества. Во-первых, существует возможность использования разных нижележащих транспортных протоколов (обычно HTTP, но возможны SMTP, RMI и др.), причем выбор (связывание) может производиться во время выполнения. Во-вторых, SOAP, как отмечалось выше, имеет механизмы расширения, которые позволяют адаптировать модель обмена к различным требованиям (безопасности, надежности и т.д.).

К ядру WSA относят также средства обнаружения, способствующие реализации принципа гибкого связывания служб. В более традиционных процедурных или объектных моделях программирования связывание происходит по ссылкам или именам. WSA предполагает, что связи должны быть динамичными и основываться на обнаружении служб, чьи интерфейсы, семантика и условия использования соответствуют ожиданиям запрашивающей стороны. Механизм обнаружения основывается на спецификациях UDDI или WS-Discovery.

На Рис. 2 приведена общая структура спецификаций WSA [22], показывающая основные функциональные группы: транспорт, передача сообщений и описание служб. Дополнительные технологии включают: безопасность, надежные сообщения, транзакции и композицию служб. Детальному обзору спецификаций WSA посвящена работа [23]. Здесь мы ограничим рассмотрение технологиями, которые родились в контексте и в связи с потребностями грида, отмечая, что все остальные также значимы в этой области.

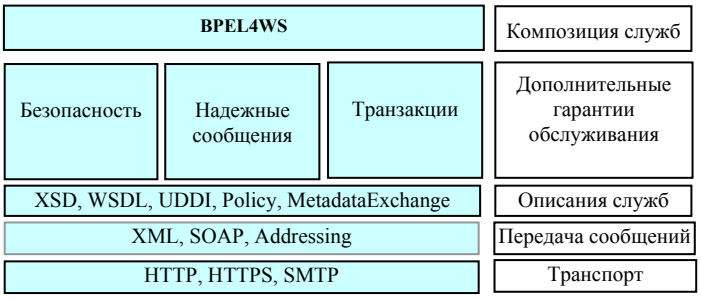

Рис. 2. Стек протоколов и спецификаций WSA

1.3. Web-службы в гриде

В архитектуре грида WSA выступает как средство, определяющее принципы разработки, благодаря которым ПГО наделяется необходимыми для условий распределенного функционирования свойствами. Специфическая для грида задача - работа с распределенными ресурсами - также подкреплена архитектурными решениями. Предлагаемая стандартизация для работы с ресурсами посредством Web-служб выражена в наборе спецификаций Web Services Resource Framework (WSRF) и WS-Notification [24], принятых организацией OASIS [25].

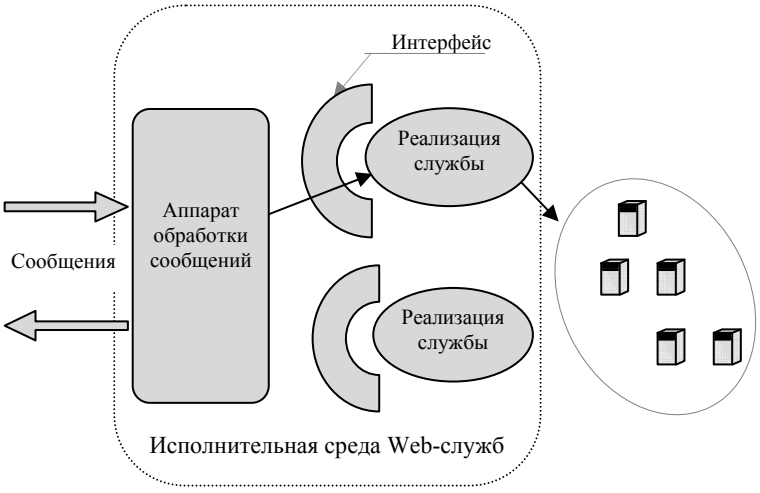

Рис. 3. Обработка обращений к операциям Web-службы

Исходное положение подхода заключается в моделировании ресурса Web-службой, так что доступ к любым ресурсам осуществляется в стандартизированной форме: исходя из формально описанных интерфейсов, унифицированных для всех ресурсов одного типа, и с помощью известных протоколов. Идущие от грида стандарты направлены на решение проблемы унификации доступа к ресурсам, которые обладают *состоянием* (stateful resource). Такие ресурсы представляют особый интерес в гриде, хотя они типичны и в общем плане.

Для того, чтобы разобраться в отношениях Web-служб и ресурсов, рассмотрим устройство Web-службы (Рис. 3).

- Web-служба представляет собой программную компоненту, которая может входить составной частью в различные приложения, построенные в соответствии с распределенной архитектурой WSA. Web-служба устанавливается в некоторой *исполнительной среде*, например, на сервере Web-приложений.

- Web-служба имеет *интерфейс*, описывающий ее функциональность, т.е. набор операций, которые могут быть выполнены компонентами приложений, выступающих в роли клиентов. Операции описываются на языке WSDL, при этом определяется формат сообщения, используемого для ее вызова, формат ответного сообщения, а также сообщения об ошибках.

- Исполнительная среда располагает *аппаратом обработки сообщений*, который, принимая сообщение по какому-либо из транспортных протоколов, допускаемых SOAP, преобразует его в формат, на который рассчитана служба, и передает сообщение ее реализации.

- *Реализация службы* выполняет обработку сообщения, возможно, обращаясь при этом к другим службам и используя ресурсы исполни-

тельной среды (компьютеры, файловую систему, базу данных).

На первый взгляд, реализационная часть службы, полностью экранированная от клиентов описанием интерфейсов, может разрабатываться произвольным способом и какая-либо регламентация здесь излишня. Это верно для служб, относящихся к категории *не сохраняющих состояние* (stateless): такие службы при обработке сообщения не используют никакой информации, кроме содержащейся в самом сообщении. Не сохраняющие состояние службы считаются наиболее предпочтительным вариантом реализации, так как в этом случае повышается надежность и масштабируемость: поскольку предшествующие обращения не имеют значения, то в случае сбоя служба может быть просто рестартована, а при возрастании нагрузки могут быть запущены новые копии.

Большинство Web-служб реальных приложений устроены более сложно: они порождают и меняют состояние ресурсов, время жизни которых превышает выполнение одной операции. Службы такого рода особенно характерны для грида, поскольку в нем порождаются, перемещаются, удаляются ресурсы данных, а также меняется состояние физических ресурсов. Так, при обращении к операции запуска задания в исполнительной среде соответствующей службой порождаются новые прикладные и системные процессы, создаются файлы результатов и диагностики, изменяется загрузка компьютеров, на которых происходит выполнение.

Это обстоятельство создает две проблемы при реализации служб в архитектуре WSA. Первая проблема связана с тем, что за исключением простейших случаев взаимодействие со службой редко сводится к вызову единственной операции. Обычно это *сеанс*, в котором выполняется некоторая последовательность операций, работающих с созданными ресурсами (например, после того как задание запущено, должна присутствовать возможность узнать о его состоянии или снять его). В WSA каждое сообщение, посылаемое для выполнения операции, обрабатывается независимо от всех других, причем связь между запрашивающей стороной и службой разрывается после обработки каждого сообщения. Для реализации же сеансов необходимо, чтобы при обработке всех сообщений служба работала со “своими” ресурсами, или, как принято говорить, сохраняла состояние.

Вторая проблема: в промежутках между обменом сообщениями ресурсы, порожденные службой, могут менять свое состояние. Изменение состояния ресурса можно проиллюстрировать на примере цикла жизни вычислительного задания: запуск - ожидание в очереди ресурсного центра - передача на исполнительный компьютер – выполнение – передача результатов – завершение. Отслеживание состояния задания существенно для пользователя грида. Помимо того, состояние ресурсов является необходимой информацией для программных механизмов управления гридом: сведения о работоспособности, загрузке и прочих характеристиках физических ресурсов являются основой для принятия решений об их распределении заданиям. Наблюдение за состоянием задания позволяет реализовать надежную обработку посредством, например, миграции.

Подход WSRF направлен на моделирование состояний ресурсов и кодифицирует соотношение между Web-службами и ресурсами с состоянием, опираясь на спецификации XML, WSDL и WS-Addressing [26]. Предлагаемое решение представляет собой компромисс между необходимостью поддержки состояния, с одной стороны, и выгодностью, с точки зрения реализации, служб без состояния. Этот компромисс был выработан путем согласования спецификаций Web-служб и предложений по Открытой инфраструктуре грид-служб OGSI [27], которые требовали достаточно существенной модификации принятых спецификаций WSA.

1.3.1. Модель WS-ресурса

В основе подхода WSRF лежит обычная практика программирования Web-служб: в реализации службы выделяется часть, которая ответственна за сохранение состояния, используя для этого базу данных или файловую систему. Тогда содержательная часть Web-службы, выполняющая прикладную обработку, может быть реализована как служба, не сохраняющая состояние, со всеми вытекающими из этого достоинствами. Поддержка состояния в течение

сеанса (в промежутках между обменом сообщениями) обеспечивается тем, что:

- при каждом обмене в сообщениях передается информация о созданных службой ресурсах – в виде дескриптора, устанавливающего связь с компонентой реализации, ответственной за сохранение состояния;
- служба реализуется таким образом, что состояние извлекается с помощью компоненты сохранения и восстанавливается в соответствии со значением дескриптора.

Стандартизируя такой способ реализации, WSRF определяет способ представления состояния в виде так называемого *Ресурса, обладающего состоянием* (stateful resource), далее *S-ресурс*. Согласно WSRF S-ресурс – это совокупность именованных данных, структура которых может быть описана XML-документом. Такое весьма абстрактное представление состояния позволяет применять его к реальным ресурсам разной природы: файлам, строкам реляционной базы данных или процессам операционной системы.

В ходе функционирования Web-службы может порождаться несколько экземпляров S-ресурса, соответствующих разным сеансам взаимодействия, поэтому каждый экземпляр имеет уникальный (для службы) идентификатор (resourceID).

Спецификации WSRF определяют правила адресации S-ресурсов и их передачи между клиентами и службой.

1.3.2. Адресация ресурсов

S-ресурс имеет смысл только в ассоциации с Web-службой, которая с ним работает, поэтому адресуется он с помощью составной конструкции WS-ресурс, включающей Web-службу и собственно S-ресурс. Когда в результате обработки сообщения некоторая операция службы создает S-ресурс, она обязана сформировать ссылку (Endpoint Reference - EPR) на конструкцию WS-ресурс, содержащую, во-первых, адрес точки приема сообщений самой службы, и, во-вторых, идентификатор S-ресурса. Способ порождения идентификаторов S-ресурсов зависит от реализации, но должен обеспечивать их уникальность, по крайней мере, в пределах службы.

Адресация посредством WS-ресурса обладает тем достоинством, что становится возможным явно указать использование S-ресурса в WSDL-описании службы в форме, удовлетворяющей требованиям спецификации WS-Addressing: идентификатор S-ресурса задается как метаданные, ассоциированные с элементом ReferenceProperties:

```
<wsa:EndpointReference>
<wsa:Address>
        http://someOrg.com/WebService
</wsa:Address>
<wsa:ReferenceProperties>
        <tns:resourceID> C </tns:resourceID>
</wsa:ReferenceProperties>
</wsa:EndpointReference>
```

Образованная таким образом EPR-ссылка WS-ресурса передается службой в ответном сообщении запрашивающей стороне. Здесь она никак не интерпретируется, но используется при всех последующих обращениях к службе. Получая EPR, служба использует ее для восстановления запомненного состояния.

Абстрактный способ адресации ресурсов службы с помощью ссылки EPR на WS-ресурс имеет несколько достоинств. Единственная такая ссылка может адресовать любое количество физических ресурсов, используемых службой, причем они остаются приватными. EPR WS-ресурса может передаваться между приложениями, позволяя им работать с одними и теми же ресурсами. При разработке службы, однако, необходимо учитывать, что сама по себе EPR не имеет какого-либо смысла, ее интерпретация составляет часть реализации службы.

1.3.3. Представление состояния ресурсов и операции доступа

Еще одна проблема, решаемая WSRF, - обеспечение доступа к описанию состояния, содержащемуся в S-ресурсе. Как сказано выше, S-ресурс представляет собой набор данных, структура которого определяется некоторой XML-схемой. Эта схема называется *Документом свойств*, а отдельные ее элементы соответствуют компонентам состояния и называются свойствами. Документ свойств является

внешним по отношению к WSDL-описанию службы (это документ с глобальной декларацией – GED в некотором пространстве имен XML). Однако связь между ними описывается в явном виде: пользователь может узнать, какой Документ ассоциирован со службой, поскольку ссылка на него указывается в ее WSDL-описании (в стандартном атрибуте Resource-Properties). WSRF вводит ряд спецификаций, в которых предложен аппарат для доступа к свойствам, содержащимся в Документе.

Спецификация WS-ResourceProperties определяет обмен сообщениями для операций чтения, изменения, установки и уничтожения значений свойств. Правило, регламентирующее работу со свойствами, состоит в том, что любая операция, изменяющая значение в Документе свойств, должна адекватно отражать изменение в реальном состоянии ресурсов.

WSRF не вводит явно понятие сеанса, однако спецификация WS-ResourceLifetime устанавливает механизмы завершения существования WS-ресурса, включая обмен сообщениями для немедленного или отложенного по времени уничтожения.

Имеющая более широкое назначение, но обычно соотносимая с WSRF спецификация WS-Notification задает механизмы подписки на события, такие, например, как изменение свойств, для получения асинхронных уведомлений.

Полный список спецификаций WSRF приводится в таблице.

Список спецификаций WSRF

Спецификация	Определяет
WS-ResourcePrope	Способ связывания WS-ресурса с описанием интерфейса Web-службы, а также обмен сообщениями для получения, изменения и уничтожения свойств.
WS-ResourceLifetim	Механизмы завершения существования WS-ресурса, включая обмен сообщениями для немедленного или отложенного по времени уничтожения.
WS-RenewableRefe	Описывает механизмы обновления ссылки на WS-ресурс в случае, когда она становится недействительной.
WS-ServiceGroup	Интерфейс к гетерогенной совокупности Web-служб, доступных по ссылке.
WS-BaseFaults	Схема описания ошибок, возникающих при обмене сообщениями.

WSRF имеет несколько реализаций. Три из них выполнены для четвертой версии инструментального комплекса Globus Toolkit в виде контейнеров Web-служб для языков Java, C, Python [28]. WSRF.NET [29] является реализацией WSRF и WS-Notification на платформе .NET. Известна также основанная на языке Perl реализация WSRF::Lite [30].

2. Реализация функций поддержки функционирования грида на основе архитектуры web-служб

Как отмечалось во введении, OGSA не ограничена формой реализации ПГО, в ней также ставится задача стандартизации функциональных возможностей служб платформы грида. В данном разделе мы рассматриваем группу служб поддержки функционирования, имея в виду показать на примере функции безопасности, каким образом они могут быть реализованы на основе архитектуры Web-служб.

Назначение служб поддержки функционирования - сделать грид технологичной средой, в которой выполнение приложений безопасно, надежно и контролируемо, несмотря на то, что оно происходит в условиях распределенности и децентрализованного управления, при большом количестве пользователей и ресурсов.

Элементарные единицы работы в гриде – задания – обрабатываются распределенно некоторой совокупностью служб, которые размещены в различных административных доменах. Для контроля за ходом обработки на всех этапах предназначена служба *мониторинга*, позволяющая получить в каждый момент статус обработки задания. Ценность этой службы не только в информировании пользователей, она служит основой для механизмов обеспечения *надежности* обработки: при ошибке на каком-то этапе может быть проинформирован предыдущий и, если сбой вызван техническими причинами, обработка может быть возобновлена на альтернативных ресурсах инфраструктуры.

Поскольку обработка задания происходит в разных местах, *протоколирование (Logging&Bookkeeping)* осуществляет сбор диагностической информации на каждом шаге обработки и делает ее доступной в целом.

Учет потребления ресурсов (Accounting) детально регистрирует количество ресурсов, потребленных заданиями при их выполнении. Сбор этих данных необходим для выстраивания экономических отношений между поставщиками ресурсов и потребителями.

Функционирование грида невозможно без наличия достоверной информации о ресурсах. *Информационная служба* производит, во-первых, сбор данных об их составе и характеристиках. Состав грида подвержен частым изменениям (ресурсы могут подключаться и отключаться), поэтому информационная служба имеет механизм автоматической (soft state) регистрации. Кроме того, она осуществляет перманентный *мониторинг ресурсов*, обновляя сведения об их состоянии. На основе совокупности данных, собранных информационной службой, осуществляется виртуализация ресурсов: исходя из их характеристик, исправности, загруженности происходит распределение заданий.

В OGSA с наибольшей полнотой разработаны вопросы поддержки больших пользовательских коллективов и обеспечения безопасности, которые рассматриваются далее.

2.1. Поддержка коллективной деятельности и безопасность

Грид образуется из ресурсных центров, принадлежащих разным организациям, но потенциально все множество ресурсов должно быть доступным пользователям независимо от их административной принадлежности. Это положение лежит в основе грида, и ему соответствует понятие *виртуальной организации* (ВО), которое введено в [31, 2]. Виртуальная организация определяется как способ объединения множества индивидуумов и/или учреждений, совместно использующих ресурсы и службы в соответствии с набором правил и политик, которые регламентируют условия потребления. Назначение ВО - снять барьеры между административными доменами и обеспечить возможность использования ресурсов многих реальных организаций. ВО могут образовываться на временной или постоянной основе, поддерживая деятельность распределенных пользовательских коллабораций. Предполагается, что:

- членами одной ВО могут быть работники различных реальных организаций;
- пользователь грида одновременно может быть членом разных ВО;
- члены ВО могут использовать ресурсы многих провайдеров, принадлежащих разным доменам;
- одни и те же ресурсы могут использоваться несколькими ВО.

Как и реальная организация, действующая в границах своего домена, ВО устанавливает политику безопасности: какие ресурсы доступны ее членам, кто может получить к ним доступ и на каких условиях осуществляется разделение ресурсов между пользователями. Более формально под *политикой безопасности* понимается [32] набор правил, определяющих отношения между субъектами и объектами грида. Здесь *субъект* – активная сторона в выполнении операций. Субъектом может быть: пользователь; процесс, инициированный пользователем; служба, действующая от имени пользователя или ресурса. *Объект* – ресурс, защищаемый политикой безопасности.

Решение проблем безопасности в ВО осложняется тем, что она представляет собой оверлей над реальными доменами, в связи с чем политика ВО должна проводиться в жизнь путем согласования с существующими инфраструктурами безопасности доменов и не должна противоречить установленной в них политике. Подход грида исходит из того, что в сложившихся реалиях домены безопасности различаются по многим аспектам.

- Различен состав задействованных в них функций безопасности. Если аутентификация и авторизация применяется всегда, то аудит или протоколирование входов обычно не считаются обязательными.
- Одинаковые функции отличаются механизмами реализации (пароли, SSH, Kerberos [33], PKI [34]).
- Используются различные идентифицирующие документы (сертификаты X.509, билеты Kerberos).

Подход к проблеме безопасности грида впервые сформулирован в работе [32], там же определен ряд требований к способу ее решения.

- Однократная регистрация. Освобождает субъект, успешно прошедший аутентифика-

цию, от необходимости участия в повторных аутентификациях при последующем доступе к ресурсам в течение некоторого периода времени. Это требование должно выполняться при обработке запросов, которая охватывает несколько доменов безопасности.

- Динамическое установление доверительных отношений. При обработке запросов, охватывающей несколько доменов, объекты одного домена должны иметь возможность обращаться к службам, расположенным в других доменах. Для этого необходимо, чтобы между различными доменами существовали доверительные отношения (trust). В условиях грида образование областей доверия должно происходить динамически, отражая, например, включение в грид новых доменов и их исключение.

- Делегирование полномочий**.** Обработка запросов производится последовательностью служб, поэтому необходимо делегирование прав доступа от запрашивающей стороны к службе с тем, чтобы ее выполнение происходило от имени запрашивающей стороны (в конечном счете, от имени пользователя). Для этого необходимы средства частичного делегирования прав и способы определения политики делегирования.

Удовлетворяя этим специфическим для грида требованиям, его система безопасности должна решать традиционные задачи: контроль доступа, целостность и конфиденциальность сообщений, отражение внешних и внутренних вторжений, аудит, протоколирование.

Исходя из перечисленных требований и условий были разработана инфраструктура безопасности для ВО – Grid Security Infrastructure (GSI), решающая задачу интеграции гетерогенных доменов безопасности. GSI основана на технологиях открытых ключей PKI и была реализована как составная часть комплекса Globus Toolkit. Дальнейшее развитие – переход на архитектуру OGSA и технологии Web-служб – привело к пересмотру этих решений по безопасности. Современное представление отражают работы [35, 36].

2.2. Средства безопасности архитектуры WSA

Архитектура WSA дает мощную базу для решения проблемы интероперабельности служб, функционирующих в различных инфраструктурах хостинга, в том числе и в аспекте безопасности.

Доступ к Web-службам может осуществляться по разнообразным протоколам, и уже на уровне сообщений поддерживаются такие функции безопасности как конфиденциальность, целостность, аутентификация (на основе спецификаций WS-Security, SAML [37]).

Службы могут публиковать свою политику безопасности (WS-Policy, XACML [38]), показывая тем самым требования для вступления с ними в контакт. Политика описывает поддерживаемый службой механизм аутентификации, необходимый уровень обеспечения конфиденциальности и целостности, способы установления доверительных соглашений и прочее. Благодаря этому вызывающая сторона имеет возможность динамически адаптироваться к требованиям безопасности вызываемых служб и устанавливать с ними доверительные соглашений (WS-SecureConversation, WS-Trust).

Имена объектов и субъектов, так е как и удостоверяющие их данные (credentials), имеют смысл лишь в пределах одного домена ВО, в котором они определены. Возможность выхода за пределы “домашнего” домена требует междоменного отображения этих атрибутов с помощью посредников, которые реализуют политику ВО и которым доверяют обе стороны. Механизм отображения является ключевым для однократной аутентификации, делегирования полномочий и авторизации. Способ объединения доменов безопасности на основе этого механизма определяется спецификацией WS-Federation.

Модель авторизации, соответствующая спецификации WS-Authorization, предполагает, что доступ к ресурсам происходит от лица запрашивающей стороны и возможность доступа определяется исходя из прав, ассоциированных с этим лицом. Наиболее типична авторизация, основанная на идентификаторах субъектов, и именно в этом случае необходимо, чтобы идентификаторы имели смысл в домене поставщика ресурсов – это поддерживается механизмом междоменного отображения имен и удостоверяющих данных. В то же время допускаются и другие способы, основанные на ролях, прави-

лах и списках контроля доступа. В любом случае результатом успешной авторизации становится отображение запрашивающей стороны в некоторый профиль домена, в который направляется запрос.

Таким образом, спецификации безопасности WSA с известной полнотой удовлетворяют требованиям грида. На основе этих спецификаций в рамках OGSA предложена модель, в которой детально адресованы вопросы реализации и состава инфраструктуры безопасности.

2.3. Поддержка безопасности в платформе грида

Модель OGSA ставит целью исключение кодов, отвечающих за безопасность, из приложений и служб, возлагая эти функции на платформу грида. Для этого, прежде всего, функции безопасности, как и все другие функции платформы, реализуются в форме Web-служб. Результатом становится повышение уровня абстракции, поскольку механизмы безопасности представляются стандартизированными интерфейсами, а детали реализации остаются скрытыми.

Состав служб безопасности OGSA определен в [39]. Помимо служб аутентификации, авторизации, аудита и прочих, состав включает также следующие службы.

- Служба преобразования удостоверяющих данных (Credential Conversion service) – служит связующим средством между доменами с разными механизмами безопасности, выполняя преобразование удостоверений.

- Служба отображения идентификаторов (Identity Mapping service) – исходя из идентификатора субъекта/объекта в одном домене, получает идентификатор в другом. Например, отображает идентификатор X.509 в имя Kerberos.

- Служба управления политикой ВО (VO Policy Management) – хранит и представляет по запросам описания элементов политики ВО. Она служит информационной базой для всех остальных служб безопасности, которые осуществляют интеграцию доменов.

Совокупность этих служб позволяет динамически устанавливать междоменные доверительные отношения, адаптируя приложения, пересекающие границы доменов, к специфике механизмов безопасности. Более того, в [35] показано, что выполнение функций безопасности может быть практически полностью возложено на среду хостинга служб - контейнеры, такие как J2EE или .NET. Продемонстрируем работу модели безопасности на примере (Рис. 4).

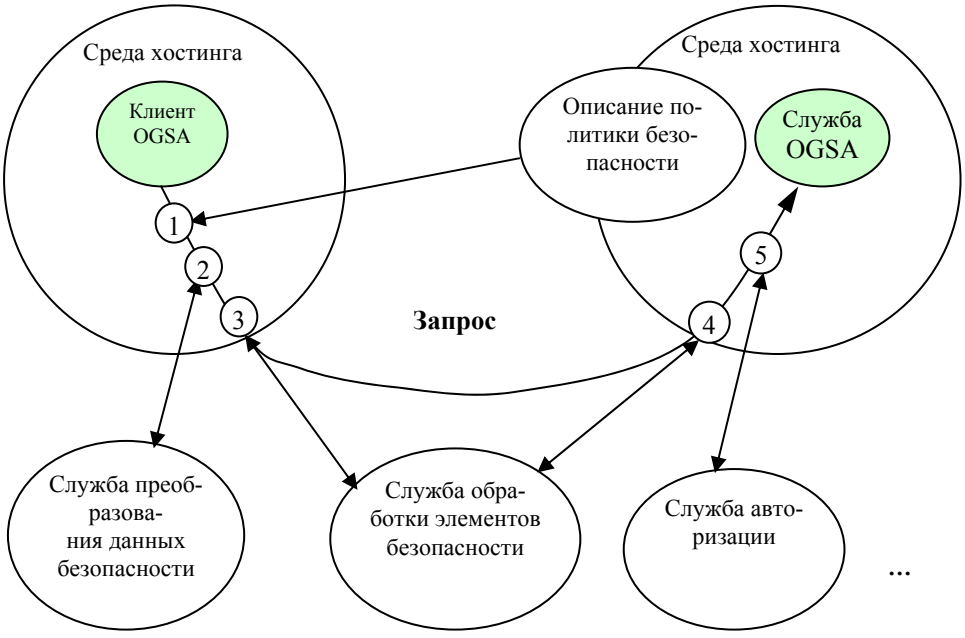

Рис. 4. Реализация функций безопасности в среде хостинга служб

В этом примере клиент и служба размещены в средах хостинга, которые самостоятельно обслуживают безопасность. При обработке запроса, направляемого клиентом службе, безопасность выполнения запроса обеспечивается следующими шагами.

1. Клиент формирует запрос и передает его службе, установленной в его среде хостинга. Эта служба получает политику безопасности вызываемой службы, определяя, какие механизмы она использует и какие удостоверяющие данные требуется послать в запросе.

2. Если удостоверяющие данные клиента не соответствуют требованиям вызываемой службы, среда хостинга клиента связывается со службой конверсии, которая преобразует имеющиеся данные безопасности в нужный формат. Например, может производиться преобразование из экаунта пользователя в удостоверения ВО или между механизмами Kerberos и PKI.

3. Среда хостинга клиента подготавливает в соответствии со стандартами WSA и вставляет в запрос данные безопасности для обмена с вызываемой службой.

4. На стороне службы среда хостинга вызывает службу обработки элементов безопасности для распаковки и проверки аутентификационных атрибутов сообщения.

5. После аутентификации и получения идентификатора клиента среда хостинга службы представляет информацию о клиенте своей службе авторизации для его отображения в локальную ресурсную среду домена.

6. При условии, что все шаги выполняются успешно, среда хостинга целевой службы направляет авторизованный запрос стоящему за ним приложению.

Как видно из этой схемы, клиентское приложение и вызываемая служба полностью освобождаются от выполнения функций безопасности. Тем самым, они становятся независимыми от конкретных механизмов безопасности, а их разработка сводится к реализации прикладной логики.

Заключение

В статье представлен анализ результатов по выработке стандартов программного обеспечения грида, которые составляют содержание OGSA - Открытой архитектуре грид-служб. Стандартизация уже дала положительные эффекты. Наличие общих правил, определяющих способы разработки, позволяет участвовать в создании служб ПГО различным коллективам. Имеющиеся реализации таких спецификаций OGSA, как инфраструктура служб (в комплексе Globus Toolkit), составляют базу, на которую могут опираться разработчики ПГО и приложений для грида.

Предложенный в OGSA способ организации ПГО согласован с современными принципами проектирования распределенных систем (SOA) и технологиями их реализации (Web-службы). Это делает более простым создание приложений для грида, а грид становится для них операционной средой, в которой обеспечиваются важные технологические свойства.

В OGSA введены расширения стандартов Web-служб, ориентированные на работу с ресурсами: ресурсы моделируются в гриде Web-службами, интерфейсы которых стандартизированы и соответствуют типу ресурса. Такие службы имеют, однако, специфику: они, как правило, должны сохранять состояние и представлять его вовне. Реализация аппарата, основанного на стандартах управления состоянием, облегчает разработку важных для грида механизмов мониторинга заданий и ресурсов. Функции управления службами (инициация, завершение) оказались полезными и востребованными во многих областях распределенного компьютинга.

В статье также рассмотрены функциональные возможности служб OGSA по обеспечению безопасности. Безопасность – одно из ряда необходимых свойств, реализуемых ПГО и превращающих грид в технологичную среду, в которой выполнение приложений надежно и контролируемо, несмотря на то, что оно происходит в условиях распределенности и децентрализованного управления, при большом количестве пользователей и ресурсов.

Для поддержки безопасности в OGSA применяется особый подход. Он также основывается на службах, но предполагает полное исключение из приложений кодов, отвечающих за безопасность. Приведенная модель показывает, что все требуемые операции могут выполняться

непосредственно инфраструктурой служб без участия приложений. Подобный подход может быть применен и к другим службам поддержки технологических свойств.

Литература

1. В.Н. Коваленко, Д.А. Корягин. Базовые принципы и способы применения грида. Программирование, № 1, 2009, с. 26-49.
2. I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Global Grid Forum, June 22, 2002.
3. Kishimoto, H., Treadwell, J. (eds.) Defining the Grid: A Roadmap for OGSA Standards 1.0. Global Grid Forum, GFD-I.053, September 2005. http://www.ggf.org/documents/final.htm
4. I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, J. Von Reich. The Open Grid Services Architecture, Version 1.0. Informational Document, Global Grid Forum, January 29, 2005.
5. Open Grid Forum. http://www.ogf.org/
6. D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard. Web Services Architecture. W3C, Working Group Note, February 2004. http://www.w3.org/TR/ws-arch/.
7. Open Grid Services Architecture Use Cases. Editors: I. Foster, D. Gannon, H. Kishimoto, Jeffrin J. Von Reich, Global Grid Forum, Oct. 28, 2004, GFD-I.029.
8. В.Н. Коваленко, Е.И. Коваленко, Д.А. Корягин, Э.З. Любимский, Е.В.Хухлаев. Современное состояние и направления развития программного обеспечения GRID. Информационные технологии и вычислительные системы, № 4, Москва, РАН, 2003, стр. 23- 36.
9. Globus Toolkit Homepage. http://globus.org/toolkit/
10. Sun Open Net Environment. http://www.sun.com/software/sunone/index.xml
11. Java 2 Enterprise Edition (J2EE). http://java.sun.com/javaee/
12. IBM WebSphere Software. http://www.ibm.com/software/websphere/
13. Microsoft .NET. http://www.microsoft.com/net/
14. OASIS Reference Model for Service Oriented Architecture V 1.0. Aug 2, 2006. http://www.oasis-open.org /committees/download.php/16587/wd-soa-rm-cd1ED.pdf
15. Atkinson, M. P., De Roure, D., Dunlop, A. N., Fox, G., Henderson, P., Hey, A. J. G., Paton, N. W., Newhouse, S., Parastatidis, S., Trefethen, A. E., Watson, P. and Webber, J. Web Service Grids: an evolutionary approach. Concurrency and Computation: Practice and Experience, 2005, 17 (2-4). pp. 377-389.
16. Extensible Markup Language (XML). http://www.w3.org/XML/
17. Simple Object Access Protocol (SOAP). http://www.w3.org/TR/soap/
18. Web Services Specifications. http://msdn2.microsoft.com/ru-ru/webservices/aa740689.aspx
19. XML Information Set, WorldWide Web Consortium, February 2004, http://www.w3.org/TR/xml-infoset/.
20. XML Schema, WorldWide Web Consortium, http://www.w3.org/XML/Schema.
21. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation, June 2007. http://www.w3.org/TR/wsdl20/
22. D. F. Ferguson, B. Lovering, T. Storey, J. Shewchuk. Secure, reliable, transacted web services: Architecture and composition, 2003. http://www.ibm.com/developerworks/webservices/library/ws-securtrans/
23. Cabrera L.F., Kurt C., Box D. An Introduction to the Web Services Architecture and Its Specifications. October 2004. http://msdn2.microsoft.com/en-us/library/ms996441.aspx
24. K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe. The WS-Resource Framework. March 5, 2004. http://www.globus.org/wsrf/specs/ws-wsrf.pdf
25. OASIS Web Services Resource Framework (WSRF) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf
26. Web Services Addressing (WS-Addressing). W3C Member Submission 10 August 2004. http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/
27. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, D. Snelling. Open Grid Services Infrastructure (OGSI) Version 1.0. Global Grid Forum Draft Recommendation, 6/27/2003. http://www.globus.org/alliance/publications/papers/Final_OGSI_Specification_V1.0.pdf
28. Ian Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. H. Jin, D. Reed, and W. Jiang (Eds.): NPC 2005, LNCS 3779, pp. 2 – 13, 2005. © IFIP International Federation for Information Processing 2005.
29. Wasson, G. WSRF.NET Programmer's Reference Manual. 2004. http://www.ws-rf.net/WSRFdotNet/WSRFdotNet_programmers_reference.pdf
30. WSRF::Lite - Perl Grid Services. http://www.sve.man.ac.uk/Research/AtoZ/ILCT.
31. I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001.
32. I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids. Proc. 5th ACM Conference on Computer and Communications Security Conference, pp. 83-92, 1998.
33. Kohl, J., and Neuman, C. The Kerberos Network Authentication Service (V5), RFC 1510, IETF, 1993. http://www.ietf.org/rfc/rfc1510.txt
34. Housley, R., Polk, W., Ford, W., and Solo, D., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, IETF, April 2002.
35. V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman,

S. Tuecke. Security for Grid Services. Twelfth International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Press, June 2003.
36. N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, Von Welch, I. Foster, S. Tuecke. The Security Architecture for Open Grid Services. July 17, 2002, Version 1. http://www.cs.virginia.edu/~humphrey/ogsa-sec-wg/OGSA-SecArch-v1-07192002.pdf
37. Security Assertion Markup Language (SAML). http://xml.coverpages.org/saml.html
38. eXtensible Access Control Markup Language (XACML). OASIS, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
39. Frank Siebenlist, Von Welch, Steven Tuecke, Ian Foster, Nataraj Nagaratnam, Philippe Janson, John Dayka, Anthony Nadalin. OGSA Security Roadmap, 2002. http://www.cs.virginia.edu/~humphrey/ogsa-sec-wg/ogsa-sec-roadmap-v13.pdf.

Коваленко Виктор Николаевич. Заведующий сектором ИПМ им. М.В.Келдыша РАН. Окончил Московский физико-технический институт в 1973 году. Автор более 60 научных работ. Область научных интересов: распределенные системы, технологии грида.

Коваленко Евгения Ивановна. Научный сотрудник ИПМ им. М.В.Келдыша РАН. Окончила Московский физико-технический институт в 1973 году. Автор 35 научных работ. Область научных интересов: распределенные вычисления, технологии грида, нейрокомпьютинг.

Ухов Лев Васильевич. Старший научный сотрудник ИПМ им. М.В.Келдыша РАН. Окончил механико-математический факультет Московского государственного университета им. М.В.Ломоносова в 1962 году. Автор более 11 научных работ. Область научных интересов: теория программирования, распределенные вычисления, технологии грида.