Организация взаимодействия клиентских приложений с веб-сервисами в гетерогенных средах

В.Н. Гридин, Г.Д. Дмитревич, Д.А. Анисимов

Аннотация. Рассматриваются вопросы внедрения веб-технологий в системы автоматизированного проектирования. Приводится сравнительная оценка возможных способов построения программного обеспечения распределенных систем автоматизированного проектирования на основе технологии веб-сервисов. Дается сравнительная оценка возможных способов построения программного обеспечения распределенных платформенно-независимых систем автоматизированного проектирования на основе технологии веб-сервисов, что позволяет строить независимые от технологий разработки и платформ системы, при этом клиентские приложения, работающие на одних платформах, могут вызывать стандартным способом сервисы, работающие на других платформах. Показывается, что при организации функционирования клиентских приложений и веб-сервисов необходимо учитывать возможные проблемы взаимодействия распределенных приложений, вызываемых неопределенностью в спецификациях WSDL, предназначенных для реализации в гетерогенных средах.

Ключевые слова: системы автоматизированного проектирования, веб-технологии, веб-сервисы, сервис-ориентированная архитектура, распределенные системы.

Введение

К числу актуальных задач в области информационных технологий относится внедрение в системы автоматизированного проектирования веб-технологий, позволяющих реализовать построение систем с сервисно-ориентированной архитектурой [1-3]. При этом в отличие от традиционных САПР, распределенная система автоматизированного проектирования может состоять из отдельных модулей, функционирующих независимо и взаимодействующих друг с другом посредством одного из SOA-ориентированных протоколов по схеме «запрос – ответ» [4, 5].

Системы, основанные на сервис-ориентированной архитектуре, независимы от технологий разработки и платформ, при этом приложения, работающие на одних платформах, могут вызывать стандартным способом сервисы, работающие на других платформах. Основная нагрузка по выполнению вычислительных операций при такой архитектуре ложится на веб-сервисы, решающие все задачи моделирования проектируемых систем. На клиентские приложения возлагаются только простейшие функции подготовки данных и отображения результатов моделирования. Вследствие слабосвязанности отдельных подсистем это позволяет проводить адаптацию существующих приложений к меняющимся условиям проектирования.

Поскольку веб-сервисы могут функционировать на более высоком уровне абстракции, анализируя и обрабатывая типы данных динамическим образом, то отдельным компонентам программного обеспечения предоставляется возможность взаимодействовать более открыто. При использовании универсально описанных интерфейсов можно использовать программные компоненты повторно, что позволяет снизить трудоемкость разработки САПР и более эффективный возврат инвестиций в программное обеспечение. Учитывая высокую

стоимость программного обеспечения современных САПР, указанная возможность имеет большое экономическое значение.

1. Архитектура распределенной САПР

Интеграция на базе веб-сервисов при разработке децентрализованных САПР позволяет перейти к описанию интерфейсов и взаимодействий на базе XML, обеспечивая возможность модификации и развития построенного программного обеспечения в условиях сохранения выбранного интерфейса. Сервис-ориентированная архитектура (SOA) основана на модульном подходе к построению программного обеспечения со стандартными интерфейсами. SOA использует принципы унификации типовых процессов, неоднократного использования элементов и организацию на базе выбранной платформы. Компоненты программного обеспечения можно распределять по различным узлам, и они предлагаются для применения как слабо связанные, независимые приложения. Хотя архитектура SOA не привязывается к какой-либо одной технологии удаленного вызова методов (COM, DCOM, COM+, .NET REMUTING, Java RMI, CORBA и др.), программные комплексы, построенные согласно SOA, как правило, реализуются как некоторая совокупность веб-сервисов, которые интегрированы согласно с известными стандартными протоколами (WSDL, SOAP). Веб-сервис описывает некоторый набор методов, каждый из которых может быть вызван в сети через стандартизированные XML-сообщения. На основании таких сообщений можно описать требуемые данные способом, не зависящим от платформы, и реализовать обмен информацией между приложениями, что обеспечивает слабосвязанность приложений.

При разработке САПР с использованием веб-сервисов могут быть применены следующие типы клиентских приложений: консольного типа, оконного типа и веб-приложение [6].

Особенностью консольных приложений является отсутствие графического интерфейса, однако их использование может оказаться полезным при реализации простейших САПР для планшетных компьютеров с небольшой площадью экрана.

Приложения оконного типа дают возможность в наибольшей степени реализовать графические средства и наилучшим образом подходят для разработки распределенных систем на базе веб-сервисов. Для любого веб-сервиса предоставляется возможность построения нескольких клиентских приложений с различными способами реализации диалогового взаимодействия.

Веб-приложение обеспечивает возможность целиком разместить все используемое программное обеспечение САПР в сети. Достоинством приложения этой структуры является открытый доступ к использованию распределенной САПР через броузер любого типа, недостатком такого приложения является увеличение времени, которое требуется для описания компонентов проектируемой системы из-за ожидания реакции на отдельных шагах ввода данных.

Для клиентского приложения любого вида вызов веб-сервисов осуществляется одинаковым способом, и для каждого веб-сервиса возможно использование любых способов реализации клиентских приложений, написанных на различных языках. Если необходимо, такие клиентские приложения можно легко модифицировать согласно с изменяющимися условиями проектирования. Возможно также расширение веб-сервиса за счет включения в него дополнительных методов.

Любое клиентское приложение может включать в себя локальную базу данных с информационными ресурсами, содержащими описание используемых при проектировании компонентов. Возможна также реализация в клиентском приложении встроенного броузера, посредством которого без выхода из приложения осуществляется прямое обращение к удаленным базам данных в сети Интернет.

Поскольку концепция распределенных сервис-ориентированных САПР предполагает создание и постоянное совершенствование клиентских приложений программистами, которые не имеют информации о структуре программного обеспечения веб-сервисов, то все разрабатываемые веб-сервисы должны быть самодокументируемыми [7,8]. Последнее означает, что каждый веб-сервис, помимо WSDL-документа, должен включать в себя информационный метод (например, getInf()), не имеющий параметров и возвращающий простую строковую пе-

ременную, в тексте, связанном с которой, должна содержаться следующая информация.

- Описание компонентов схемы (число компонентов данного типа, массив включения, массив значений параметров и порядок формирования этих массивов для каждого типа компонента).
- Алгоритм преобразования массивов описания компонентов схемы к унифицированному виду, обеспечивающему гарантированное функционирование распределенной системы в гетерогенных средах.
- Имя рабочей директории для размещения прокси-объекта, обеспечивающего взаимодействие веб-сервиса с клиентскими приложениями.
- Операторы формирования прокси-объекта для взаимодействия веб-сервиса с клиентскими приложениями.
- Заголовки рабочих методов веб-сервиса.
- Описание возвращаемых результатов работы методов веб-сервиса в унифицированной форме.
- Описание преобразования унифицированной формы возвращаемых результатов методов веб-сервиса к виду, требуемому для отображения результатов работы веб-сервиса в клиентских приложениях.

Возможная архитектура распределенной сервис-ориентированной системы автоматизированного проектирования на основе технологии веб-сервисов приведена на Рис. 1.Согласно этой архитектуре все веб-сервисы, решающие вычислительные задачи моделирования путем удаленного вызова соответствующих методов, располагаются на сервере веб-сервисов, а клиентские приложения в зависимости от их типа, размещаются либо непосредственно на пользовательских компьютерах ПК (локальные клиентские приложения), либо на сервере веб-приложений (клиентские веб-приложения).

При этом вся работа, связанная с вводом и отображением данных, осуществляется клиентскими приложениями, что позволяет осуществлять их модификацию и адаптацию к условиям проектирования независимо от веб-сервисов. В свою очередь, при условии сохранения выбранного интерфейса возможно развитие и совершенствование методов веб-сервисов, выполняющих основные задачи моделирования.

При построении веб-сервисов применяется ряд спецификаций, основанных на открытых стандартах, при этом стек протоколов выполняется в следующей последовательности: поиск веб-сервиса при помощи протокола UDD, описание веб-сервиса на основе WSDL,

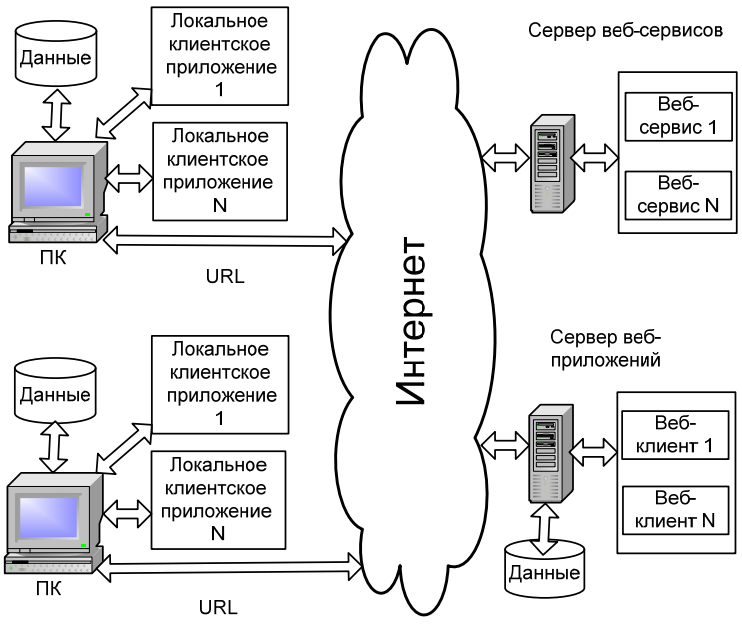

Рис. 1. Архитектура распределенной сервис-ориентированной системы автоматизированного проектирования

вызов веб-сервиса через протокол SOAP, кодирование данных XML, транспортировка HTTP.

Основным стандартом технологии веб-сервисов является протокол WSDL. Он используется для создания описания интерфейса веб-сервиса. При этом WSDL-документ описывает информационные блоки сервиса и имеет информацию об интерфейсе и методах, доступных для общего пользования, дает описание типов данных, использующихся при передаче в запросах и ответах по соответствующему протоколу, а также информацию о транспортном протоколе, который используется для работы с сервисом и адресную информацию о расположении описываемого сервиса. Следует заметить, что стандарт документа WSDL является независящим от какого-либо определенного протокола обмена XML-данными, но содержит встроенные средства поддержки SOAP.

Протокол SOAP – это межплатформенный стандарт, используемый для форматирования сообщений, которыми обмениваются веб-сервис и клиентское приложение. SOAP описывает XML-конверт для сообщений веб-сервиса, модель обработки и алгоритм кодирования информации перед ее отправкой. В технологии веб-сервисов используются два типа SOAP-сообщений. Первый является сообщением запроса, которое клиент направляет веб-сервису для инициализации выбранного метода веб-сервиса. Второй тип является ответным сообщен ием, которое веб-сервис возвращает клиентскому приложению.

Транспортный протокол HTTP применяется для обмена информацией с веб-сервисом через SOAP-сообщения. Возможно также в качестве транспортного протокола использование стандарта SMTP.

Протокол UDDI используется для создания бизнес-журналов, фиксирующих имена компаний, предоставляющих веб-сервис и соответствующие URL-адреса их WSDL-контрактов. В качестве альтернативы можно использовать стандарт DISCO, который применяется для создания документов поиска, содержащих ссылки на множество конечных пунктов веб-сервисов.

Необходимость в использовании протокола UDDI (или DISCO) возникает только тогда, когда необходимо осуществлять поиск требуемого для решения задач проектирования веб-сервиса. Если же URL-адрес веб-сервиса уже известен, то его можно просто жестко закодировать или поместить в файл конфигурации, а потом вызвать сервис и осуществить обмен информацией, применяя стандартный формат обмена сообщениями.

Обращение к реестру осуществляется единственный раз и после получения адреса веб-сервиса повторно не выполняется. Один раз выполняется также и процесс получения WSDL-документа. Если URL-адрес веб-сервиса известен, то обращение к реестру выполнять не обязательно, так как файл WSDL можно вызвать непосредственно из веб-сервиса. Поскольку в реальных условиях работы с распределенными системами на основе веб-сервисов их URL-адреса, как правило, известны, обращение к UDDI осуществляется сравнительно редко.

Вызов метода веб-службы осуществляется из клиентского приложения через прокси-класс, объект которого реализует требуемую заглушку (stub). Прокси-класс осуществляет преобразование передаваемых методу аргументов в SOAP-сообщение и направляет это сообщение веб-сервису. После выполнения сервисом вызываемого метода, прокси-класс получает соответствующий SOAP-ответ и преобразует его в соответствующие типы данных, что позволяет клиентскому приложению отобразить результаты выполнения метода

2. Организация функционирования веб-приложений в средах с гетерогенной структурой

Работающие на одной платформе SOA-приложения могут вызывать стандартным способом сервисы, функционирующие на иных платформах и написанных на различных языках. Важно отметить, что сервисы не располагают информацией о вызывающем их приложении, а клиентские приложения не имеют информации о способе, которым сервисы выполняют свои задачи. SOA-программы предоставляет инкапсуляцию всех деталей реализации каждого приложения от иных компонентов, предоставляя гибкий способ неоднократного использования и комбинирова-

ния приложений для построения распределенных информационных систем.

Любой язык программирования, позволяющий создавать документы XML и посылать информацию при помощи HTTP, дает возможность взаимодействовать с веб-службой. Можно получить результат работы веб-сервиса из любой системы, отличающейся от той, в которой создан веб-сервис, при этом нет необходимости оставаться на определенном уровне совместимости, так как веб-службы включены в общие стандарты. Таким образом, стандартизация SOAP обеспечивает возможность соединения слабосвязанных приложений, созданных на любой платформе их реализации. Такое свойство веб-сервисов обеспечивает эффективное и оптимальное использование широкого ряда гетерогенных, слабосвязанных ресурсов в распределенных приложениях. Поэтому веб-сервисная архитектура не просто некоторый вариант реализации распределенных систем для решения нужд обмена информацией и интеграции различных приложений, а качественно новый уровень абстракции, который можно представить как интеграцию технологий взаимодействия, а не приложений или платформ. Операционная система или аппаратная платформа, а также язык программирования скрываются от потребителя веб-сервисов стандартным представлением WSDL-документа. При этом современные средства Java/J2EE и .NET позволяют осуществлять автоматическую генерацию WSDL-документа для конкретной системы.

Вместе с тем документация стандартов нередко допускает разную интерпретацию, что может порождать различное толкование одних и тех же документов. Если серверные и клиентские приложения веб-сервисов используются в одной среде, то вопросы их совместимости не возникают. Однако при совместной работе клиентского и серверного приложений на различных платформах, возможно возникновение проблем, вызываемых неопределенностью в спецификациях WSDL, а также и SOAP, которые предписывают, что содержимое сообщений, направляемых веб-сервисом, должно иметь формат XML, но не включать жестких требований к этому формату. Вместо этого заложена поддержка нескольких форматов сообщений, наиболее важными среди которых являются форматы документа (document style) и RPC (RPC style). Базовым для всех форматов SOAP-сообщений в системах разработки среды .NET является формат документа (document style), где он применяется по умолчанию, но в среде Java/J2EE в разное время в качестве рекомендуемых считались как стили document style, так и RPC style. Сейчас в среде Java/J2EE стиль RPC постепенно вытесняется методикой document style, как более гибкой и не накладывающей на разработчика ограничений.

Учитывая возможные проблемы взаимодействия распределенных приложений, предназначенных для реализации в различных гетерогенных средах, при построении веб-сервисов следует использовать стиль document style. При этом более предпочтительной следует считать комбинацию document/literal с рекомендацией, чтобы SOAP-тело содержало только один дочерний элемент документа. Еще более надежной для обеспечения совместимости клиентских и серверных приложений является введенная компанией Microsoft комбинация document/literal wrapped, которая может использоваться и в Java-приложениях при развертывании веб-сервисов восходящим методом. Поскольку комбинация document/literal wrapped обеспечивает гарантированное взаимодействие приложений клиента и сервера при работе в средах гетерогенного типа, то именно эту комбинацию и следует выбирать при построении веб-сервисов.

Возможны два различных подхода к выбору сервис-ориентированной архитектуры распределенных приложений, функционирующих в гетерогенных средах. Первый основан на построении веб-сервисов в среде Java/J2EE с возможностью построения клиентских приложений либо в среде Java/J2EE, либо в среде .NET на основе языка C#. Второй предусматривает построение веб-сервисов в среде .NET с возможностью построения клиентских приложений в среде Java/J2EE или в среде .NET. Оба подхода к выбору сервис-ориентированной архитектуры обеспечивают платформенную независимость как серверного, так и клиентского приложений, и трудоемкость их реализации при различных подходах примерно одинакова.

Каждая архитектура требует установки соответствующего типа сервера (например, Apache Tomcat для первого случая и Internet Information Services во втором случае). Оба сервера легко доступны и их установка не вызывает каких-либо затруднений. Однако следует учитывать, что при размещении на сервере веб-сервисов, построенных в среде .NET, для возможности их функционирования на этом сервере должна быть установлена лицензионная среда .NET Framework, что накладывает определенные ограничения на выбор провайдера. Вместе с тем, для функционирования на сервере веб-сервисов, разработанных в среде Java/J2EE, требуется только поддержка виртуальной Java-машины, которая является свободно распространяемым программным продуктом и установлена практически на всех компьютерах. Именно поэтому в качестве среды разработки веб-сервисов целесообразно выбирать среду Java/J2EE и поэтому первый подход к выбору сервис-ориентированной архитектуры распределенных приложений, способных функционировать в гетерогенных средах, является наиболее предпочтительным.

При организации взаимодействия клиентских приложений, созданных в среде .NET с сервисами, созданными в среде Java/J2EE, должны решаться следующие основные задачи:

- построение клиентского Windows-приложения или веб-приложения в интегрированной среде .NET (например, в Visual Studio);
- организация взаимодействия объекта прокси-класса приложения среды .NET с веб-сервисом среды Java/J2EE;
- развертывание клиентского Windows-приложения или веб-приложения.

Пусть веб-сервис WS_Java среды Java/J2EE имеет имя класса Class_Name и в нем имеется метод get_Name, получающий от клиентского приложения некоторый список аргументов Arg_Name и возвращающий массив данных, который необходимо записать в клиентском приложении в массив Arr_Name.

Тогда для реализации первого этапа следует создать в интегрированной среде разработки приложения (например, в Visual Studio) проект с выбранным именем Client_Net, и сформировать в шаблоне проекта требуемый пользовательский интерфейс клиентского Windows-приложения среды .NET.

Взаимодействие объекта прокси-класса клиентского приложения среды .NET с веб-сервисом среды Java/J2EE реализуется в среде .NET путем создания в клиентском приложении объекта ws прокси-класса Class_NameService и вызова метода get_Name с соответствующими аргументами:

```
//Создание объекта прокси класса
 Class_NameService ws = new
Class_NameService();
 //Вызов метода get_Name
 Arr_Name = ws. get_Name (Arg_Name).
```

После включения приведенного кода в описание одного из классов проекта Client_Net необходимо связать клиентское приложение с сервисом, в качестве которого в данном примере выступает веб-сервис WS_Java, размещенный на серверном компьютере с некоторым адресом IP_0. Чтобы осуществить такую связь следует открыть в интегрированной среде Visual Studio пункт меню **Project | Add Web Reference** и в появившемся окне **Add Web Reference** ввести в поле URL адрес

http://IP_0/ WS_Java /services/ Class_Name?wsdl

После этого следует щелкнуть на кнопке Go, а затем на кнопке Add Reference. В результате выполненных операций будет построен прокси-класс Class_NameService, при помощи которого можно вызывать требуемый метод веб-сервиса WS_Java.

Для развертывания клиентского Windows-приложения Client_Net достаточно открыть папку bin\Debug проекта этого приложения и скопировать файл Client_Net.exe в рабочую папку клиента.

Следует отметить, что синтаксис построения объекта прокси-класса при реализации в гетерогенной среде отличается от стандартного синтаксиса операторов, решающих аналогичную задачу, как в однородной Java-среде, так и в однородной среде .NET.

Заключение

Сервис-ориентированная архитектура позволяет строить независимые от технологий разработки и платформ системы, при этом клиент-

ские приложения, работающие на одних платформах, могут вызывать стандартным способом сервисы, работающие на других платформах. При организации функционирования клиентских приложений и веб-сервисов необходимо учитывать возможные проблемы взаимодействия распределенных приложений, вызываемых неопределенностью в спецификациях WSDL, предназначенных для реализации в различных средах. Реализация изложенных концепций обеспечивает гарантированное взаимодействие приложений клиента и сервера на различных платформах при работе в средах гетерогенного типа.

Литература

1. Гридин В.Н., Анисимов В.И. Методы построения систем автоматизированного проектирования на основе Интернет-технологий и компактной обработки разреженных матриц // Информационные технологии в проектировании и производстве. №1, 2009.
2. Гридин В.Н., Дмитревич Г.Д., Анисимов Д.А. Построение систем автоматизированного проектирования на основе Web-сервисов // Автоматизация в промышленности. №1, 2011.
3. Алмаасали С.А., Анисимов В.И. Построение распределенных систем автоматизированного проектирования на основе методов диакоптики // Известия СПбГЭТУ. 2014. № 1. С. 15-19.
4. Дей Н., Мандел Л., Райман А.. Eclipse: Платформа Web-инструментов. пер. с англ. - М.: КУДИН-ПРЕСС, 2008.
5. Морган С., Райан Б., Хорн Ш., Бломсма М. Разработка распределенных приложений на платформе Microsoft.Net. М., СПб.:Питер. 2008.
6. Гридин В.Н., Дмитревич Г.Д., Анисимов Д.А. Построение систем автоматизированного проектирования на основе Web-технологий// Информационные технологии. №5, 2011.
7. Гридин В.Н., Дмитревич Г.Д., Анисимов Д.А. Построение веб-сервисов систем автоматизации схемотехнического проектирования // Информационные технологии и вычислительные системы. №4, 2012.
8. Анисимов Д.А. Методы построения систем автоматизации схемотехнического проектирования на основе веб-сервисов// Известия СПбГЭТУ «ЛЭТИ». №10, 2012.

Гридин Владимир Николаевич. Директор Центра информационных технологий в проектировании РАН. Окончил Московский авиационный технологический институт в 1972 году. Доктор технических наук, профессор. Автор 200 печатных работ. Область научных интересов: информационные технологии в проектировании, безопасность систем. E-mail: info@ditc.ras.ru.

Дмитревич Геннадий Даниилович. Профессор Санкт-Петербургского электротехнического университета. Окончил Ленинградский электротехнический институт в 1960 году. Доктор технических наук. автор 150 печатных работ. Область научных интересов: моделирование систем, методы оптимизации.E-mail: gddm@inbox.ru.

Анисимов Д.А. Аспирант Санкт-Петербургского электротехнического университета. Окончил Санкт-Петербургский электротехнический университет в 2010 году. Автор 14 печатных работ. Область научных интересов – моделирование систем, веб-ориентированные системы. E-mail: anisimovdenis2009@yandex.ru.