УДК 658.512.011.56.005:91:004.9

О.И. Жуковский, Н.Б. Рыбалов

Архитектура корпоративной WEB-ориентированной ГИС

Рассмотрены подходы к построению пользовательского интерфейса средств Web-публикации ГИС, проведен анализ существующих принципов реализации серверной и клиентской составляющих классических Web-приложений, предложены усовершенствованные архитектуры клиентского и серверного уровней для Web-ориентированных ГИС.

Ключевые слова: ГИС, архитектура ГИС, WEB-ориентированные системы.

Введение

В ходе повсеместного внедрения современных геоинформационных технологий, которые становятся основой создания крупных территориальных и промышленных ГИС, все более востребованной становится возможность публикации геопространственных данных в Интернет. Наряду с этим при построении комплексных ГИС возникает проблема создания наиболее эффективной архитектуры, которая должна обеспечить производительность, масштабируемость и надежность решения. В широком смысле построение архитектуры сводится к выбору основных составляющих системы: базовой ГИС-технологии, средства хранения пространственных данных. В узком смысле – это применение наиболее эффективных архитектурных решений на каждом уровне проектируемой геоинформационной системы, где центральное место занимают средства Web-публикации.

Многоуровневая архитектура Web-ориентированных ГИС

В основе большинства современных ГИС лежит многоуровневая архитектура. В таких системах обычно разделяется три уровня: обработка данных, бизнес-логика и представление. В Web-ориентированных ГИС можно условно выделить две части – клиентскую и серверную, где каждая часть может иметь более сложную организацию и, оставаясь в рамках архитектуры, подразделяться на несколько уровней.

При разработке клиентской части средств Web-публикации акцент делается на приложение, реализованное на основе Web-браузеров. Интерфейс, реализованный в среде браузера, должен предоставлять средства просмотра картографических данных, обеспечивать выполнение пространственных запросов и вывод атрибутивной информации. Насыщение клиентского приложения богатой функциональностью порождает новый класс клиентских приложений, получивший название «богатый клиент», который в отличие от «тонкого клиента» помимо уровня представления получает уровень бизнес-логики.

Применение архитектурных шаблонов, предназначенных для организации приложений, упрощает процесс разработки и сокращает время, затраченное на весь процесс создания Web-приложения. Одним из наиболее используемых является шаблон «Модель-Вид-Контроллер» (Model-View-Controller – MVC), достоинства которого включают в себя разделение ответственности и уменьшение количества избыточного кода.

Создание эффективного пользовательского интерфейса

Одной из причин низкой эффективности приложений, влияющих на степень интерактивности, является задержка между действием пользователя и реакцией на него. Непостоянное время отклика, связанное с задержками при передаче данных по сети, понижает практичность и затрудняет оценку качества приложения. Таким образом, низкое время отклика становится основным требованием, предъявляемым к пользовательскому интерфейсу [1]. Существует несколько моделей взаимодействия пользователя и Web-приложения, применение одной из которых позволит достичь нового уровня интерактивности:

– переходная модель (на основе синхронного взаимодействия);

– независимая модель (на основе асинхронного взаимодействия).

Независимая модель благодаря принципам асинхронного взаимодействия освобождена от недостатков переходной модели. Каждая операция считается длительной и выполняется асинхронно в отдельном потоке (в фоновом режиме), а пользователь в это время может выполнять другие действия. При этом время отклика сводится к минимуму, уменьшается влияние задержек, связанных с длительными вычислениями и с взаимодействием по сети. Например, при асинхронном взаимодействии пользователь, активировав функцию по-

иска объекта и не дожидаясь оповещения об успешном выполнении операции, может продолжить использовать другие функции.

Выбор архитектуры приложения на стороне сервера

Одним из ключевых этапов проектирования приложения является процесс разделения функциональности на несколько уровней. Эта задача решается посредством применения различных многоуровневых архитектур. После логического расслоения, возникает задача физического распределения уровней приложения. В классических Web-приложениях уровень представления доставляется клиенту, а остальные уровни находятся на сервере. В независимых приложениях уровень бизнес-логики распределен между клиентом и сервером, поэтому для функционирования распределенного приложения сервер должен обеспечивать доставку клиентской части приложения браузеру. Рассмотрим варианты архитектур, предназначенных для построения классических Web-приложений. С учетом достоинств и недостатков каждого подхода попытаемся выделить наиболее существенные детали, использование которых позволит создать оптимальную серверную архитектуру Web-ориентиро-ванной ГИС.

Архитектура Model2

Архитектура Model2 является разновидностью шаблона проектирования MVC. Как и MVC, данная архитектура разделяет модель данных приложения, пользовательский интерфейс и управляющую логику на три отдельных компонента. При этом модификация одного из компонентов оказывает минимальное воздействие на другие компоненты. Отличие архитектуры Model2 от MVC в том, что котроллер имеет единственную точку входа и единственное определение последовательности действий пользователя. Данное решение, именуемое как «Контроллер запросов» (Front Controller), объединяет все действия пользователя по обработке запросов в одном месте, распределяя их выполнение посредством одного объекта-обработчика [2]. Из каждого запроса, поступающего к Web-серверу, извлекаются название Web-обработчика и иерархия команд. Web-обработчик – это объект, который выполняет фактическое получение POST- или GET-запросов, поступивших на Web-сервер. Контроллер запросов извлекает необходимую информацию из адреса URL и входных данных запроса, после чего решает, какое действие необходимо инициировать, и делегирует его выполнение соответствующей команде.

Данная архитектура хорошо подходит для доставки данных, но не определяет средств доставки клиентского приложения, что может быть существенным недостатком при разработке «богатой» клиентской части Web-ориентированной ГИС.

Архитектура на основе компонентов

Архитектура на основе компонентов позволяет повысить уровень абстракции при программировании пользовательских интерфейсов. Благодаря этому в распоряжении разработчика серверных программ предоставляются элементы, интерфейс которых не уступает компонентам графического интерфейса для настольных систем. Для отображения таких компонентов необходима автоматическая генерация потока HTML-данных и JavaScript-программ. В результате в среде браузера реализуются функции, типичные для настольных систем и разработчик избавляется от необходимости программировать низкоуровневые операции. Преимуществом компонентного подхода является возможность реализации системы воспроизведения на базе обычного HTML-кода, в результате чего исчезает необходимость специального изучения разработчиками особенностей языка сценариев. Применение компонентной архитектуры «в чистом виде» для создания серверной части ГИС имеет ряд недостатков:

– подобное решение хорошо подходит лишь для тех приложений, для которых требуются только стандартные типы компонентов. Создание сложного приложения, такого как картографический вьювер, потребовало бы написания собственного набора компонент от прокручиваемой карты до элементов;

– для построения сложного приложения на основе независимой модели, архитектура должна обеспечивать эффективную передачу данных в процессе работы. Это может быть достигнуто посредством механизмов обработки событий серверной частью приложения. Однако контроллер в данной архитектуре жестко привязан к уровню сервера, ориентирован на работу с компонентами и лишен гибкости для определения собственных обработчиков событий.

Архитектура на основе Web-служб

Архитектура на основе Web-служб ориентирована на работу с информацией, предоставляемой бизнес-логикой приложения. В данном случае служба – это нечто, к чему можно обратиться по сети и получить в качестве ответа структурированный документ. Программа, использующая Web-службу как источник данных, обладает высокой степенью автономности и более широкими возможностями повторного использования элементов

приложения. Служба определяется один раз и может быть использована различными клиентами, работающими независимо друг от друга. Таким образом, данная архитектура разделяет средства, предназначенные для генерации клиентской части приложения и для ее обслуживания. Создание комплексной геоинформационной системы, интегрирующей множество технологий, может потребовать от серверной архитектуры большей гибкости. Например, в случае синхронизации модели данных клиента и сервера, как правило, используются механизмы обработки событий, а при использовании множества систем, требующих доставки клиентской части приложения, эффективна архитектура на основе контроллера запросов.

Усовершенствованная архитектура на основе Web-служб с использованием развитого контроллера

С учетом достоинств и недостатков вышеописанных архитектур выделим основные требования к серверной архитектуре Web-ориентированной ГИС: поддержка обработки событий; использование Web-служб; поддержка средств доставки клиентской части приложения. Удовлетворить данные требования позволяет усовершенствованная архитектура, разработанная на основе подхода MVC (рис. 1).

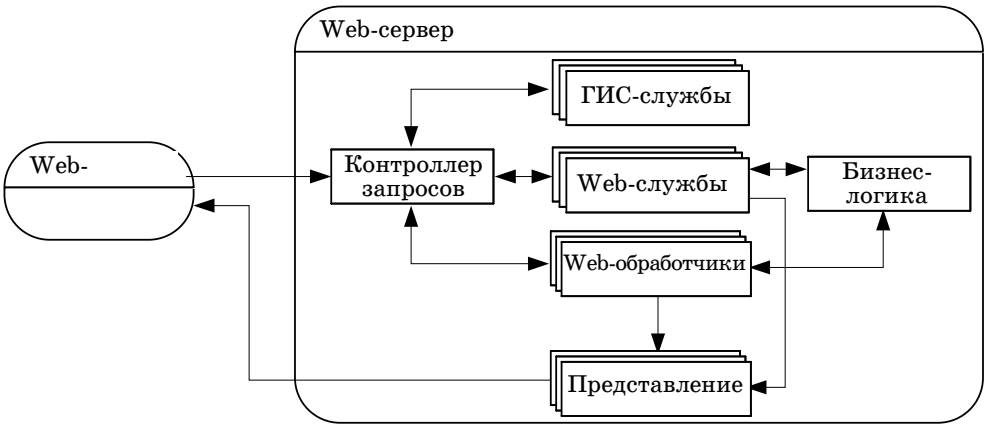

Рис. 1. Усовершенствованная архитектура на основе Web-служб

Рассмотрим более подробно организацию уровня управляющей логики. Контроллер запросов реализован в виде цепочки фильтров.

Фильтры – это программные компоненты, позволяющие разбить процесс обработки запроса на отдельные этапы и выделить различные вспомогательные действия. Логика каждого фильтра реализована таким образом, что переход к последующему фильтру зависит от успешного выполнения действий предыдущего (рис. 2).

Рис. 2. Работа контроллера на основе фильтров

В предложенной авторами архитектуре запрос обрабатывается фильтрами в следующей последовательности:

1) Инициализация – создание или открытие сеанса работы пользователя и обновления статистической информации о пользователе.

2) Диспетчеризация запроса – анализ и фильтрация некорректных URL-запросов, выбор контроллера, иерархии команд, Web-обработчика и выделение запросов к ГИС-службам.

3) ГИС-обработчик – подпрограмма, которая получает запросы картографического вьювера, перенаправляет их соответствующим ГИС-службам и передает результат браузеру.

4) Авторизация – проверка прав доступа на выполнение действий.

5) Генерация HTTP-ответа – отправка данных и HTTP-заголовков.

6) Обработка запроса – запуск команды, которую обрабатывает указанный контроллер и Web-обработчик.

7) Генерация представления – отвечает за формирование представления данных или содержимого в заданном формате и генерацию клиентской части приложения.

Данная архитектура обладает преимуществами архитектуры с использованием Web-служб и достоинствами применения подхода MVC. Применение развитого контроллера на базе фильтров позволяет управлять ходом работы приложения, более эффективно обрабатывать события и исключения, а также интегрировать в серверное приложение средства публикации картографических данных и ГИС-службы.

Выбор архитектуры приложения на стороне клиента

Проблема создания сложного и масштабируемого клиента обострилась с ростом популярности реализаций технологии асинхронного взаимодействия. Теперь к клиенту предъявляются иные требования: он не только должен обладать богатой функциональностью, но и интеллектуальностью реализуемой логики, что выражается в способности выполнять большинство операций вместо сервера и способности самостоятельно отвечать на действия пользователя. Приложение, способное удовлетворить данное требование, должно строиться на основе независимой модели взаимодействия и обеспечивать малое время отклика, где основную роль играет эффективность обмена данными между клиентом и сервером.

Данные, возвращаемые сервером, можно разделить на три типа [1]:

– *Содержимое.* В качестве данных передается HTML-поток генерируемый сервером, который может отображаться в составе элемента IFrame или быть встроенным в структуру документа средствами DHTML. Данный подход эффективен при передаче статических данных, например справочной информации.

– *Сценарии.* В качестве данных передаются JavaScript-сценарии, которые позволяют динамически генерировать представление, либо содержать в себе структуры данных модели предметной области. Недостатком данного подхода является тесная связь между уровнями приложения, так как для генерации кода на стороне сервера необходимо иметь информацию о программном интерфейсе клиента.

– *Структурированные данные.* Сервер передает только низкоуровневые данные (XML/JSON), которые обрабатываются уровнем клиента и могут использоваться для обновления клиентской модели данных и пользовательского интерфейса. При использовании данного подхода удается разделить уровни сервера и клиента, что позволяет изменять код клиента и сервера независимо друг от друга.

Предлагаемая авторами архитектура (рис. 3) основана на принципах MVC и ориентирована на использование структурированных данных, поставщиками которых являются Web- и ГИС-службы.

Картографический вьювер, интегрированный в клиентское приложение, реализован в виде компонента, управление которым осуществляется посредством API-интерфейса. Контроллер в приложении включает в себя обработчики событий, перехватывающие действия пользователя, генератора запросов, подготавливающего данные к отправке на сервер, и набор Callback-функций, которые выполняют обновление модели данных и представления при получении асинхронного ответа.

Данная архитектура обладает следующими преимуществами:

– применение архитектуры MVC обеспечивает разделение ответственности и уменьшение количества избыточного кода;

– использование независимой модели взаимодействия позволяет повысить интерактивность приложения;

– ориентированность на структурированные данные позволяет разделить код клиента и сервера, а также повлиять на уменьшение времени отклика;

– разделение приложения на отдельные компоненты способствует коллективной разработке приложения, помогая разработчикам сосредоточиться на выполнении функций, требующих определенной специализации, и взаимодействовать через четкие интерфейсы.

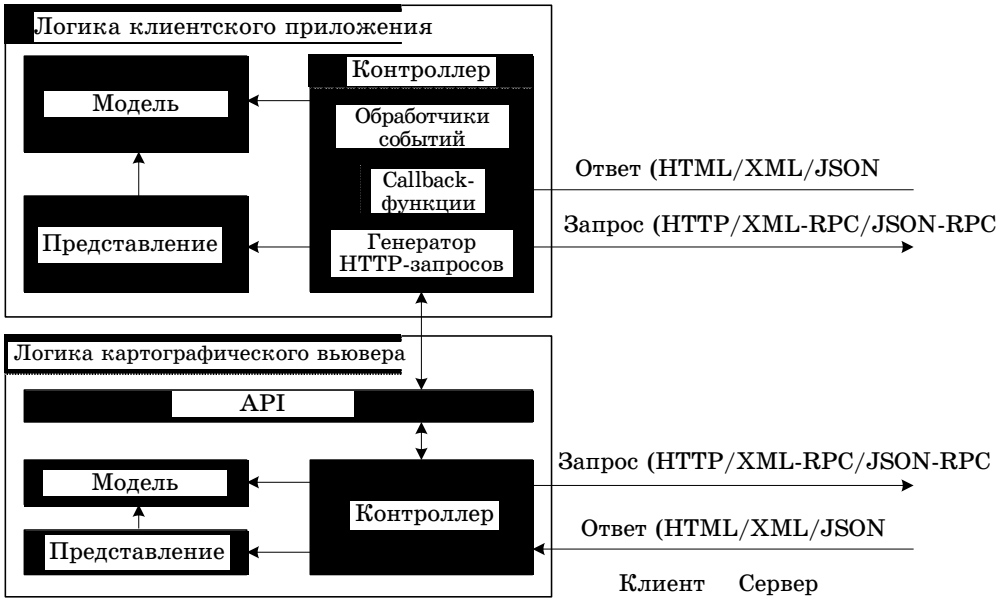

Рис. 3. Архитектура приложения на стороне клиента

Заключение

Эффективность предложенных архитектур была подтверждена в ходе их использования для обеспечения средств Web-публикации геоинформационных систем промышленных предприятий ООО «Томскнефтехим» (г. Томск) и ОАО «НКМК» (Новокузнецкий металлургический комбинат, г. Новокузнецк), а также ГИС Томского университета систем управления и радиоэлектроники, что позволило повысить эффективность разработки проектов [4].

Литература

1. Джеймс Д. Ajax в действии (пер. с англ., под ред. Вейтмана В.В., Назаренко А.В.) / Д. Джеймс, Д. Крейн, Э. Паскарелло. – М.: ИД «Вильямс», 2006. – 640 с.

2. Фаулер М. Архитектура корпоративных программных приложений (пер. с англ.). – К.; М.; СПб.: Вильямс, 2004. – 544 с.

3. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2001. – 368 с.

4. Ехлаков Ю.П. Принципы построения Web-ориентированной ГИС промышленного предприятия / Ю.П. Ехлаков, О.И. Жуковский, Н.Б. Рыбалов // Изв. Том. политех. ун-та. – 2006. – Т. 309, № 7. – С. 146–152.

Жуковский Олег Игоревич
Канд. техн. наук, доцент кафедры автоматизации обработки информации и управления ТУСУРа
Тел.: (3822) 41-44-70
Эл. почта: ol@muma.tusur.ru

Рыбалов Никита Борисович
Аспирант кафедры автоматизации обработки информации и управления ТУСУРа
Тел.: (3822) 41-47-09
Эл. почта: fisher@muma.tusur.ru

O.I. Zhukovsky, N.B. Rybalov

Architecture of corporate WEB-based GIS

Some approaches intended for construction of the user interface of GIS Web-publishing tools are considered. An analysis of the existing principles for implementation of server and client components of the classic Web-applications has been performed. The improved architectures of the client and server levels for Web-based GIS are suggested.

Key words: GIS, architecture of GIS, Web-based applications.