DEVELOPMENT OF INFORMATION SYSTEMS BASED ON FLEXIBLE THREE-TIER ARCHITECTURE

Voychenko O.
International Research and Training Center for Information Technologies and Systems, Kiev, Ukraine

Nowadays development of complex information systems often occurs through the integration and further simultaneous use of several previously developed autonomous subsystems based on different technologies, thus it may cause significant problems due to the software incompatibility. One approach to solution of the information systems integration problem is the use of a flexible three-tier architecture based on the technology of Web service.

РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ НА БАЗЕ ГИБКОЙ ТРЕХУРОВНЕВОЙ АРХИТЕКТУРЫ

Войченко А.П.
Международный научно-учебный центр информационных технологий и систем, Киев, Украина

В настоящее время создание сложных информационных систем зачастую происходит путем интеграции и дальнейшей одновременной эксплуатации нескольких ранее созданных автономных подсистем, разработанных на основе различных технологий, вследствие чего могут возникать значительные проблемы в связи с их программной несовместимостью. Одним из путей решения проблемы интеграции информационных систем, использующих различные технологии, является использование гибкой трехуровневой архитектуры на базе механизма веб-сервисов.

Последние годы характеризуются все более широким внедрением информационных систем (ИС) в большинстве областей человеческой деятельности. Активно развивается сетевая инфраструктура, с каждым годом растет степень проникновения интернета и число пользователей, применяющих информационные технологии (ИТ) в своей профессиональной деятельности.

Соответственно, возрастает и уровень требований, предъявляемых пользователи к применяемым ИС. Для обеспечения

эффективного функционирования ИС в новых условиях проектировщикам приходится решать ряд задач, которые ранее оставались за рамками рассмотрения по причине своей технической сложности, а так же сравнительно более низких темпов создания и внедрения высокотехнологичных инновационных решений в области ИТ.

В частности, все большую актуальность приобретают проблемы системной интеграции, когда для удовлетворения возрастающих информационных потребностей пользователей становится необходимым обеспечить эффективное взаимодействие между различными ИС в процессе решения некоторых общих задач.

Другой актуальной проблемой является во многих случаях невозможность четкой формулировки полного набора технических требований к конкретной ИС на этапе ее проектирования. Данная проблема обычно обусловлена постоянным совершенствованием и усложнением логики процессов, для информационного обеспечения которых конкретная ИС разрабатывается.

Еще одной проблемой, с которой приходится сталкиваться проектировщикам современных ИС является быстрое обновление базовых технологий и средств разработки. На этапах проектирования и разработки не всегда очевидно, какие новшества могут появиться в ближайшем будущем, и, соответственно, какие шаги необходимо предпринять в данный момент для обеспечения совместимости создаваемой системы с технологическими инновациями, которые целесообразно будет применять при ее усовершенствовании в будущем.

Одним из возможных путей решения описанных выше проблем является использование гибкой трехуровневой архитектуры ИС.

Для понимания ее особенностей и отличий от альтернативных решений рассмотрим сначала традиционные двух- и трехуровневые клиент-серверные архитектуры.

Двухуровневая клиент-серверная архитектура включает в себя два базовых структурных элемента: клиент и сервер, которые взаимодействуют (обмениваются данными) с помощью инфраструктуры компьютерной сети.

Схематически такая архитектура изображена на рисунке 1.

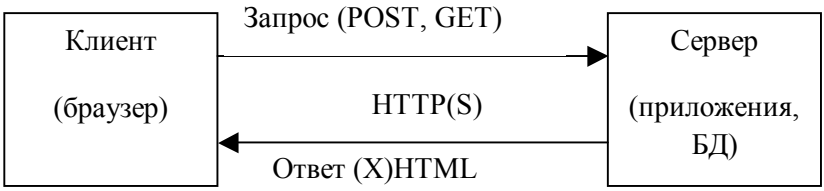

Рисунок 1. Двухуровневая клиент-серверная архитектура.

В качестве клиента в подавляющем большинстве случаев на ПК пользователя выступает стандартный веб-браузер, хотя возможны архитектурные решения, где в роли клиентского программного обеспечения используется специализированное приложение.

Северная компонента содержит определенный набор приложений(скриптов) для реализации бизнес-логики а так же базу данных (БД) для хранения информации.

Система работает следующим образом: клиент отправляет на сервер некоторый запрос, параметры которого передаются методом GET или POST, сервер обрабатывает запрос, анализирует параметры и в зависимости от их значений генерирует ответ средствами установленных приложений в соответствии с заложенной бизнес логикой.

При этом в качестве хранилища информации используется находящаяся на сервере БД. Ответ в большинстве случаев представляет собой сгенерированную динамически веб-страницу, которая пересылается клиенту для последующего отображения на ПК пользователя.

С ростом числа пользователей и усложнением бизнес-логики у систем с подобной архитектурой проявляются следующие недостатки:

Производительность: Так как количество пользователей растет, производительность начинает ухудшаться. Ухудшение производительности прямо пропорционально количеству пользователей, каждый из которых имеет собственное подключение

к серверу, что означает, что сервер должен поддерживать все эти соединения, даже когда операции с базой данных не ведутся.

Безопасность: Каждый пользователь должен иметь собственный индивидуальный доступ к базе данных, и обладать правами, предоставленными для работы с приложениями. Для этого необходимо хранить права доступа для каждого пользователя в базе данных. Когда нужно добавить функциональность одному из приложений, может возникнуть необходимость обновления системы прав пользователя.

Функциональность: Независимо от того, какой тип клиента используется, большая часть обработки данных осуществляется с помощью базы данных, это означает, что она полностью зависит от возможностей предусмотренных в базе данных производителем. Это может серьезно ограничить функциональность приложения, поскольку различные базы данных поддерживают различные функции, используют различные языки программирования и даже реализуют некоторые основные функции по-разному.

Мобильность: Двухуровневая архитектура настолько зависит от конкретной реализации базы данных, что перенос существующих приложений для различных СУБД, становится серьезной проблемой.

Для того чтобы снять ограничения, налагаемые рассмотренной выше двухуровневой архитектурой, был введен дополнительный уровень. Цель введения в архитектуру дополнительного уровня – разделить функционал, реализующий бизнес-логику и системные компоненты, обеспечивающие функционирование базы данных.

Схематически трехуровневая архитектура изображена на рисунке 2.

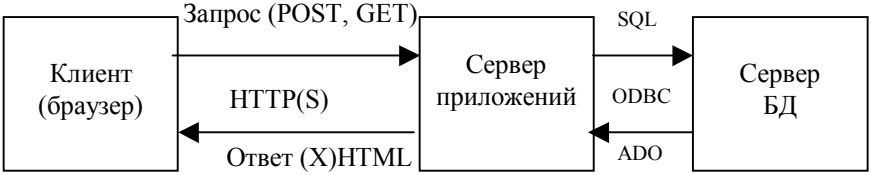

Рисунок 2. Трехуровневая архитектура.

Как видно из рисунка, серверная часть в данной архитектуре представлена двумя компонентами, причем прикладные

приложения, реализующие бизнес-логику, логически отделены от системы хранения данных (БД). Взаимодействие сервера приложений и БД осуществляется средствами API (ODBC) путем отправки SQL-запросов и получения соответствующих выборок данных из базы с помощью интерфейса ADO. Технологии ODBC, SQL и ADO не являются единственным возможным решением для связи сервера приложений и сервера БД и зависят от выбора конкретной СУБД и средств разработки.

По сравнению с двухуровневой архитектурой, трехуровневая обеспечивает следующие преимущества:

- Масштабируемость;
- конфигурируемость – изолированность уровней друг от друга позволяет быстро и простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней;
- высокая безопасность;
- высокая надёжность;
- низкие требования к скорости канала между клиентами и сервером приложений.

При использовании этой архитектуры клиенты соединяются только с сервером приложений, а не непосредственно с сервером БД, нагрузка на БД снижается, и отпадает необходимость реализации бизнес-логики в пределах сервера БД. Сервер БД теперь может выполнять только функции хранения и обработки данных, а задачу обработки запросов пользователей выполняет средний уровень трехуровневой архитектуры.

Данная архитектура успешно реализует свои преимущества в случаях, когда разработка информационной системы на ее основе реализуется «с нуля». В случаях, когда создание информационной системы происходит путем интеграции и дальнейшей одновременной эксплуатации нескольких ранее созданных подсистем, разработанных на основе различных технологий, в рамках данной архитектуры могут возникать значительные проблемы в связи с их программной несовместимостью.

Аналогичные проблемы возникают и в случаях, когда наращивание функционала системы в какой-то момент ограничивается возможностями выбранных технологий разработки,

а новые потенциально перспективные технологии не являются в достаточной степени совместимыми с использовавшимися ранее.

Одним из путей решения проблемы интеграции информационных систем, использующих различные технологии является использование механизма веб-сервисов.

Веб-сервис – информационная система, чьи общедоступные интерфейсы определены на универсальном кросс платформенном языке XML.

Веб-сервис является единицей модульности при использовании сервис-ориентированной архитектуры.

Основные достоинства веб-сервисов:

· Веб-сервисы обеспечивают взаимодействие информационных систем независимо от платформы;

· Веб-сервисы основаны на базе открытых стандартов и протоколов. Благодаря использованию XML достигается максимальная простота разработки и отладки веб-сервисов;

· Использование интернет-протокола HTTP обеспечивает взаимодействие информационных систем через межсетевые экраны.

Реализация веб-сервисов тесно связана с использованием протокола SOAP.

SOAP – протокол обмена структурированными сообщениями в распределённой вычислительной среде. SOAP является одним из стандартов, на которых базируются технологии веб-сервисов.

Еще одной технологией, позволяющей существенно упростить интеграцию информационных систем и оптимизировать интерфейс пользователя, является AJAX.

AJAX – подход к построению интерактивных пользовательских интерфейсов, заключающийся в асинхронном обмене данными браузера с веб-сервером.

В результате, при обновлении данных, веб-страница не перезагружается полностью, и интерфейсы становятся более быстрыми и удобными с точки зрения эргономики.

Использование совокупности рассмотренных выше технологий в качестве основы для усовершенствования стандартной трехуровневой архитектуры информационных систем позволяет реализовать гибкую трехуровневую архитектуру.

Схематически гибкая трехуровневая архитектура изображена на рисунке 3.

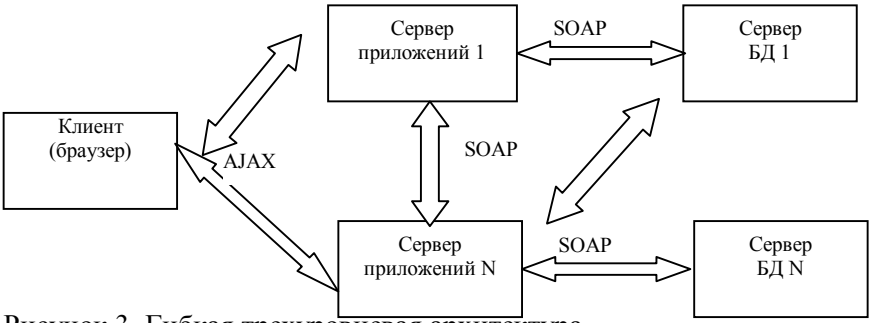

Рисунок 3. Гибкая трехуровневая архитектура.

Как видно из рисунка, серверная часть в данной архитектуре представлена совокупностью серверов приложений и серверов баз данных, которые в общем случае могут быть реализованы на основе использования различных платформ и технологий разработки, и взаимодействовать друг с другом в зависимости от решаемой задачи, как автономные веб-сервисы. Таким образом, решаются проблемы кросс-платформенности и совместимости различных технологий, использованных при разработке отдельных компонентов информационной системы.

Кроме того, использование распределенной серверной инфраструктуры существенно облегчает интеграцию информационной системы с различными облачными сервисами.

Клиентская часть строится на основе динамического AJAX-интерфейса, расширяя возможности интерактивного интерфейса пользователя и позволяя динамически манипулировать данными. При этом разные категории пользователей могут использовать различные сервера приложений для работы с системой и решать свои задачи в рамках изолированных технологических цепочек.

Например, при проведении модификации или регламентного обслуживания одного из серверов приложений, только те группы

пользователей, которые взаимодействуют непосредственно с данным сервером, могут испытывать временные трудности в работе, в то время как для остальных групп пользователей функционал системы останется неизменным.

Разделение серверов приложений и баз данных в зависимости от выбранной бизнес-логики в полной мере позволяет использовать преимущества процессного подхода к организации функционирования информационной системы, а так же существенно повысить ее защищенность от несанкционированного доступа.

Данная архитектура дает возможность реализовать максимальную степень гибкости при управлении данными и перераспределении вычислительных ресурсов в процессе функционирования информационной системы. Использование рассмотренной архитектуры позволяет также добиться масштабируемости информационной системы, эффективно осуществлять хранение резервных копий данных и дублировать критически важные компоненты системы.

Литература

1. Войченко А.П. "Методы управления непрерывным функционированием информационных систем на основе конвергенции традиционных и облачных технологий". Тезисы международной конференции ITEA-2010, Киев, 2010.