ТРЕХЗВЕННЫЕ АРХИТЕКТУРЫ ДЛЯ ПОСТРОЕНИЯ БАНКОВСКОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Сафронов В. Д.

Сиротко С.И. – канд. техн. наук, доцент

Объектные трехзвенные архитектуры DCE, CORBA, DCOM предлагают реально действующие стандарты для построения трехзвенных приложений. Существующие серверы приложений, которые не поддерживают этих архитектур, предлагают свои внутренние механизмы для построения трехзвенной архитектуры. Перед тем, как выбрать, в какой архитектуре строить свою информационную систему, корпоративный разработчик обязан ясно представлять себе, какую цену он заплатит за соответствие архитектуре, то есть необходимо хорошо представлять себе не только плюсы соответствующей архитектуры, но и ее минусы.

Минусы трехзвенных архитектур

Трехзвенные архитектуры обладают рядом преимуществ. Это гибкость, масштабируемость, многоплатформенность, распределенность технологий, управляемость, сочетаемость технологий, безопасность данных, доступность, надежность. Но также они имеют и ряд недостатков:

– непроработанность архитектуры;
– тяжелые в реальности решения;
– несоответствие с уже имеющимися технологиями;
– неустойчивость версий стандартов, а, следовательно, потенциальная несовместимость;
– недоразвитость инструментов (неудобство, ошибки);
– неоправданная дороговизна средств (или обучения специалистов, или высокая цена администрирования).

Рассмотрим два основных вида объектных архитектур DCE и DCOM. DCE – это распределенная архитектура появившаяся раньше DCOM.

Исходная UNIX-ориентированность технологии DCE, ее некоторая громоздкость, ее ориентированность только на язык C, отсутствие системы управления приложениями – это очевидные минусы.

С другой стороны – надежность, поддержка архитектуры многими производителями, надежный сервис безопасности, масштабируемость и ориентация архитектуры для работы с тысячами пользователей, использующих сотни источников данных – это плюсы. Расширяемость DCE доказывает и то, что при помощи хорошо устроенных продуктов можно скомпенсировать недостатки архитектуры, достроив в рамках архитектуры недостающие механизмы.

DCOM – закрытая архитектура с закрытым протоколом. Может использоваться только в рамках данной реализации, соотношения между объектными сервисами обладают очевидными недостатками. Производителем DCOM является компания Microsoft.

Но недостатки архитектуры так же, как и в случае DCE, можно исправить удачно сделанными продуктами. Inprise MIDAS вносит необходимую гибкость в архитектуру, снабжая ее необходимым инструментарием и утилитами.

Тонкие и толстые клиенты

В системе, построенной на основе трехзвенной архитектуры, клиентское приложение часто называют тонким клиентом. Имеется в виду то, что клиентское приложение трехзвенной архитектуры освобождено от кода обращения к данным, и поэтому гораздо тоньше по объему.

Тонким клиентом называют также и стандартные internet-клиенты, которые в интрасетях действительно занимаются только отображением / представлением данных, хотя и не являются объектами, соответствующими архитектурам DCE, CORBA, DCOM. Эти два типа клиентов различаются не столько по объему кода, сколько по способу их применения в течение жизни информационной системы. Трехзвенная архитектура предназначена для того, чтобы внести расширяемость и масштабируемость в информационные системы. Системы, которым нужны эти качества, никогда не бывают полностью завершены, и в течение жизненного цикла всегда подвергаются изменениям. Тонкие клиенты первого типа также подвергаются изменениям с изменениями системы и, должны время от времени заменяться новыми, более модифицированными версиями. Тонкие клиенты второго типа (ультратонкие) могут не заменяться в течение жизненного цикла системы, поэтому обслуживание интранет-системы несравненно проще трехзвенной системы, построенной без применения стандартных тонких клиентов. В принципе, никакого противоречия тут нет, и можно было бы построить ультратонкого клиента и для DCE, CORBA, DCOM.

Первоначально может показаться, что ультратонкий клиент не может быть достаточно функциональным по сравнению с просто тонким. Действительно, ультратонкий клиент не меняется в течение своей жизни, однако способен интерпретировать скрипты, получаемые с сервера.

В том случае, если при установлении соединения (или в течение рабочего сеанса, что тоже возможно) приложение серверного слоя снабжает ультратонкого клиента правилами работы с бизнес-логикой, правилами отображения и манипулирования информацией, мы имеем дело с процессом доставки кода. В предельном случае готовое клиентское приложение, хранящееся на сервере, просто

инсталлируется на клиентский компьютер. И это наиболее опасная ситуация, поскольку на клиентский компьютер может быть доставлено разрушительное приложение, снабженное вирусом или «троянским конем».

Следующий шаг по ужесточению контроля – это введение на клиента интерпретатора, который контролирует опасные ситуации. Например, на клиентский компьютер подгружается только описание формы – расположение кнопок, полей ввода и других контрольных элементов, что позволяет достичь компромисса между требованиями безопасности, функциональности презентационной логики и требованиями нулевого администрирования для ультратонкого клиента.

Движение в сторону достижения максимальной безопасности в пределе останавливается на варианте, когда клиентскими рабочими станциями являются терминалы либо X-терминалы, а вся информация между сервером и терминалами курсирует с шифрованием трафика.

Удобство реализации ультратонкого клиента с подгружаемым со стороны сервера приложений скриптом или кодом может быть как удачным, так и неудачным в зависимости от реализации.

Список использованных источников:

1. https://ru.wikipedia.org
2. Информационные системы. Учебник /Петров В.Н. – СПб.: Питер, 2008.
3. Информационное обеспечение систем управления. Учебное пособие/Голенищев Э.П., Клименко И.В. - Ростов н/Д: Феникс, 2009.