электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.418

МЕТОДЫ ИНТЕГРАЦИИ КОРПОРАТИВНЫХ ПРИЛОЖЕНИЙ

Леонов В.С., студент
кафедра «Системы обработки информации и управления»
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Научный руководитель: С.Б. Спиридонов, доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
chernen@bmstu.ru

Внедрение новых технологий всегда встречали сопротивление со стороны современных технологий. Тем не менее, «технология на данный момент – это просто средство для достижения цели. Использование информации, ее эксплуатация и максимизация является стратегическим подходом». Это означает, что прежде чем мы сможем применять интеграцию корпоративных приложений, мы должны оправдать ее использование. Чтобы сделать это, мы рассмотрим, зачем изначально она нужна, начиная с краткой истории развития компьютерных систем. Как только мы предоставим обоснование ее использования, мы можем разобрать конкретные требования для успешной реализации. Затем мы рассмотрим две различные архитектуры интеграции и выделим их преимущества и недостатки. На следующем шаге выделим четыре общих методов интеграции, и какие критерии являются полезными в его выборе.

Краткая история

На протяжении многих тысяч лет, человек использовал инструменты для выполнения некоторой работы. Пещерные люди прикрепляли заостренные камни к деревянным палкам, что позволяло им атаковать добычу на расстоянии. Современный человек использует компьютеры для выполнения многих задач – от торговли и банковского дела до освоения космоса и медицины. Первые компьютеры в первую очередь использовались для автоматизации ручных задач. Эти задачи, как правило, разделены на небольшие части, как, например, фирмы разделены на департаменты. Поскольку системы разрабатывались, либо для конкретного отдела, либо конкретным отделом, они часто делают то же, что и ручные операции, а это очень мало. Эти системы не будут зависеть от других; никакой интеграции

не будет. Они были известны как "дымоходные" системы, похожие на то, что мы видим на рисунке 1.

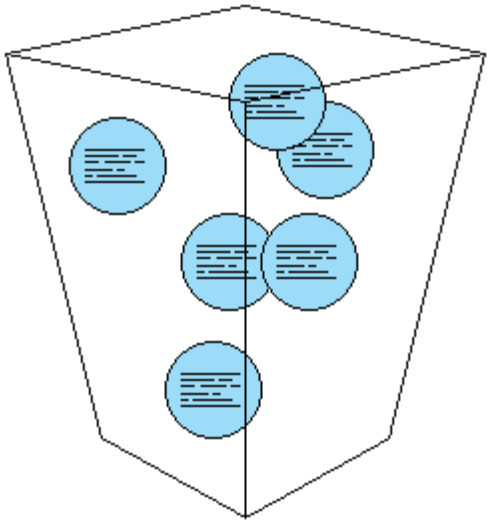

Рис. 1. Независимые «островки» вычислений

Эта модель «островков автоматизации» была постепенно забыта из-за трех причин:

1. Взаимодействие систем стало необходимым. Это означало, что существующие системы должны взаимодействие.

2. Осознание того, что информации о клиентах в рамках «дымоходной» системы было много, особенно если смотреть в целом. Например, поставщик программного обеспечения может иметь отдельные системы для дома, для бизнеса и правительственных клиентов, но не иметь возможности получить глобальное представление информации.

3. Желание интегрировать ключевые системы с поставщиками и клиентами.

По мере того как интеграция становится все более и более важной, системы будут выглядеть как на рисунке 2.

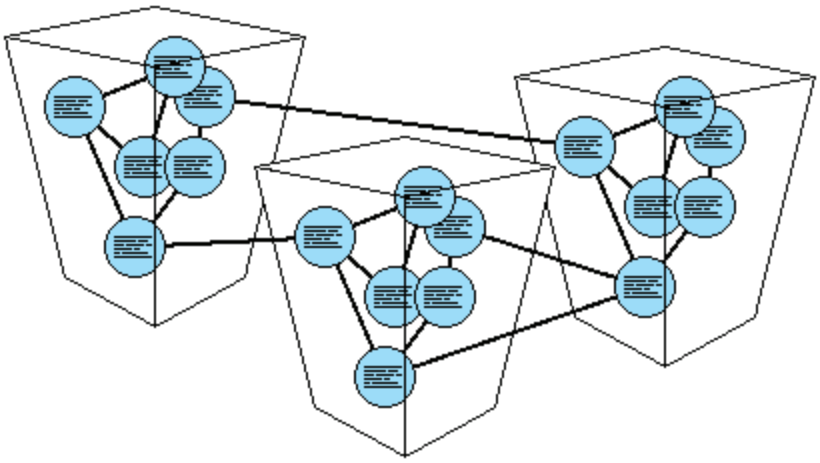

Рис. 2. Интегрированное предприятие

В последнее десятилетие, коробочные версии ПО стали очень популярными. Программного обеспечения, таких как SAP, Oracle ERP, PeopleSoft JDEdwards, Siebel и Clarify, как правило, хорошо работают по отдельности, но опять же, создают "информационные островки" как и раньше, хотя и в более широком масштабе. Когда обычные данные должны измениться, приходится обновлять их вручную. Например, в немецком банке, конторский персонал будет читать с одного экрана и вносить данные в пользовательский интерфейс другого приложения. Это был долгим, и порождало много ошибок. Также, «дальнейшее распространение готовых приложений, приложений, которые были обращены к потенциальным проблемам 2000 года, управление цепочками поставок / бизнес-бизнес (B2B) интеграции, обтекаемые бизнес-процессы, интеграция веб-приложений, и весь технологический прогресс» означало, что долгосрочные масштабируемые решения стали необходимы. Эти решения называются методами «Интеграции Корпоративных Приложений» (ИКП).

Требования ИКП

Теперь, когда необходимость ИКП была признана, мы должны объяснить, что ИКП должна включать для того, чтобы быть реалистичным решением. Она должна охватывать каждую часть системы предприятия, в том числе архитектуру, аппаратные средства, программное обеспечение и процессы.

• Интеграция бизнес-процессов (BPI): Принципиально важно для корпорации указать процессы, связанные с обменом корпоративной информацией. «Это позволяет организациям оптимизировать операции, сократить расходы и улучшить чувствительность к требованиям клиентов». Это может включать в себя управление процессами, моделирование процессов и документооборот. Здесь мы подразумеваем сочетание задач, процедуры, организации, необходимую входную и выходную информацию, а также инструменты, необходимые для каждого этапа бизнес-процесса.

• Интеграция приложений: Здесь целью является «соединить данные или функции из одного приложения вместе с данными из другого приложения». Это может включать бизнес-бизнес интеграцию, системы управления взаимоотношениями с клиентами (CRM), которые могут быть интегрированы с backend-приложениями компании, веб-интеграцию и создание веб-сайтов, которые взаимодействуют с несколькими бизнес системами.

• Интеграция данных: Если мы хотим, чтобы методы, указанные выше, преуспели, мы должны также интегрировать данные. Их расположение должно быть идентифицировано, зарегистрировано, и модель метаданных должна быть построена (основное руководство для

различных хранилищ данных). Теперь данные могут быть разделены или распределены по базе данных систем, обеспечивающих нахождение данных в стандартных форматах, таких как COM + / DCOM, CORBA, EDI, JavaRMI и XML.

• Интеграция платформы: Наконец, отдельные потребности гетерогенной сети должны быть интегрированы. Интеграция платформы имеет дело с процессами и инструментами, которые требуются, чтобы эти системы общались, как оптимально, так и надежно, так чтобы данные могли быть переданы через различные приложения без труда. Например, нахождение, как Apple, может передавать данные к беспроводной КПК, является частью всей корпоративной системной интеграции.

Архитектура интеграции ИКП

В ИКП, существуют два типа интеграции архитектуры: прямая точка-точка (PTP) и на основе промежуточного уровня.

Точка-точка

Это основной, более традиционный подход. Он используется, поскольку он прост и быстр, жизнеспособен в ситуациях, когда у нас есть несколько систем для интеграции. Например, новый веб-сайт, может нуждаться во взаимодействии с существующей системой заказа на продажу и интеграция типа точка-точка является подходящей. Однако, если вы интегрируете дополнительные приложения, вы получаете ситуации, как показано на рисунке 3.

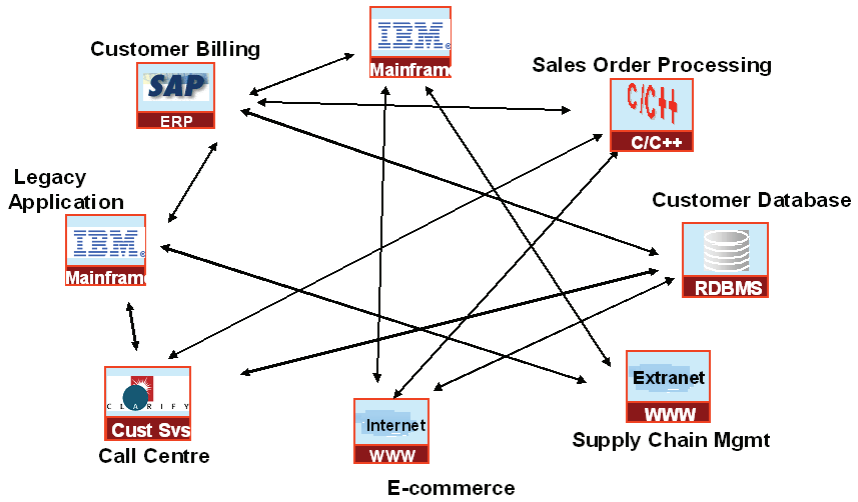

Рис. 3. Поздние стадии интегрирования «точка-точка»

По мере того как это решение увеличивается в масштабах, инфраструктура становится хрупкой. Тесная связь, зависимости, и количество точек интеграции - основные недостатки. Восемь приложений, показанных на рисунке 3, используют в общей сложности 12 точек интеграции, каждая из которых нуждается в поддержке. В теории, необходимо удвоить

число точек интегрирования в сравнении с количеством приложений, как показано на рисунке 4, где пяти приложениям нужно десять точки интеграции.

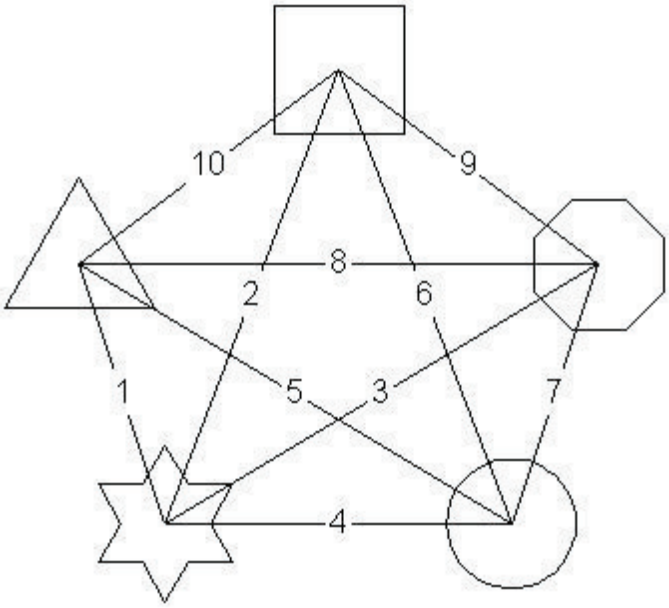

Рис. 4. Число соединений «точка-точка»

Если мы хотим избежать этого, мы должны обеспечить промежуточный слой, который может изолировать изменения между приложениями, эффективно уменьшая количество связей. Для этого, мы используем промежуточного уровень.

Middleware

Как мы уже говорили, мы должны предоставить что-либо в качестве посредника между приложениями. С помощью промежуточного ПО, мы можем предоставить универсальные интерфейсы, которые позволяют приложениям передавать сообщения друг другу. Каждый из этих интерфейсов определяет процесс, обеспечиваемый приложением. На рисунке 5 мы видим логическое описание этого принципа.

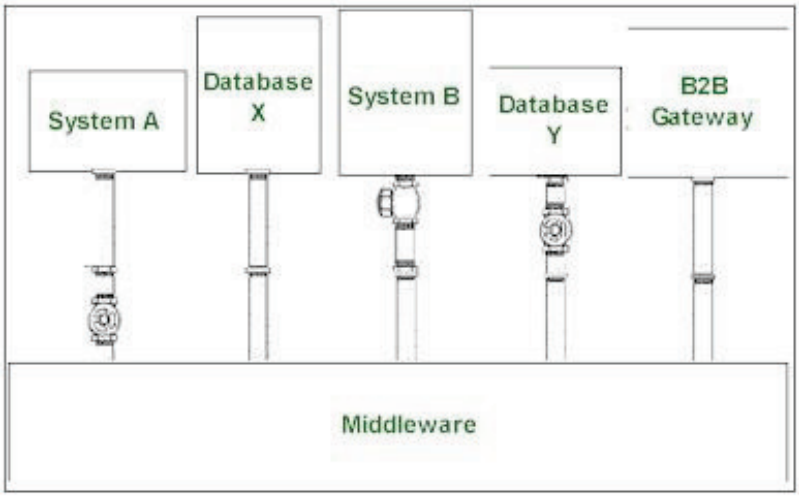

Рис. 5. Интеграция на основе промежуточного уровня

Теперь пять приложений имеют только пять точек интеграции. Мы можем также добавить и заменить приложения таким образом, что это не повлияет на другие приложения. Промежуточный уровень сам по себе может выполнять операции, такие как маршрутизация, преобразования, объединения, разделение и преобразование данных. Тем не менее, существуют дополнительные сложности в плане создания промежуточного ПО, и преобразование приложения для использования промежуточного API.

Четыре метода интеграции

Как только логическая архитектура ИКП была выбрана, мы можем перейти к фактическому методу интеграции, который пригоден для использования. Есть четыре общих методов интеграции:

- Интеграция уровня данных
- Интеграции уровня пользовательского интерфейса (UI)
- Интеграция уровня приложений
- Интеграция уровня метода

Интеграция уровня данных

Здесь интегрируются хранилища бэкенд-данных соответствующих приложений, и могут быть прямого или обратного действия. При использовании интеграции прямого действия, одно приложение делает SQL-запросы на таблицы базы данных другого приложения. Это происходит через связи баз данных или хранимые процедуры, и данные вставляются в базе данных другого приложения. А интеграция обратного действия использует триггеры. Триггеры захватывают изменения данных и пишут идентифицирующую информацию в интерфейсные таблицы. Интегрция обратного действия

используется, когда приложение требует пассивного уведомления об изменениях в данных другого приложения.

Когда приложение, которое должно быть интегрировано не обеспечивает API, либо клиентских интерфейсов, вы должны использовать интеграцию уровня данных. Вы также должны иметь хорошее понимание бизнес-операций, которые могут повлиять на модель данных приложения.

Интеграция уровня пользовательского интерфейса

Этот метод связывает логику с кодом пользовательского интерфейса. Интеграционный код встраивается в компоненты пользовательского интерфейса.

В случаях, когда прямой доступ к базе данных труден или невозможен, или когда бизнес-логика встроена в пользовательский интерфейс, это правильный метод интеграции. Мэйнфреймы и клиент-серверные приложения часто являются хорошими кандидатами на это. Мэйнфреймы как правило, не имеют доступа к дружелюбным хранилищам данных, и не обеспечивают открытых API. Тем не менее, интеграция уровня пользовательского интерфейса, как правило, используется в качестве последней надежды. Если вы добавите скрипты для отлова событий клиент / серверных приложений, они становятся очень трудно поддерживаемыми, так как уровень интеграции увеличивается и происходят больше изменений. Создается постоянная связь между поддержкой пользовательского интерфейса и интеграционного кода.

Интеграция уровня приложений

Этот метод считается наилучшим для интеграции приложений и API интегрируемых приложений. Этот метод удобен, так как он прозрачен для интегрируемых приложений и сохраняет целостность данных приложения. Интерфейс приложения позволяет вызывать бизнес-логику для сохранения целостности данных.

Интеграция уровня метода

Это менее часто используемый метод интегрирования на уровне приложений, описанный выше. Здесь мы собираем общие операции над несколькими приложениями в одно приложение, которое взаимодействует с интегрируемыми приложениями. Он обычно используется, когда каждое интегрируемое приложение имеет аналогичный набор API или функциональных методов. Интегрированные приложения должны поддерживать удаленный вызов процедур (RPC) или технологию распространяемых компонентов. Основным недостатком этого подхода является вновь плотная связь между компонентами. Проблемы

возникают, когда вносятся изменения в API интегрируемых приложений, и эти проблемы будут распространяться на другие приложения, которые зависят от них.

Как выбрать метод интеграции?

Необходимо посмотреть на каждую систему и определить возможные интерфейсы в этом приложении. В некоторых случаях приложение не имеет API, поэтому серверное хранилище данных представляет единственный вариант. В случаях, когда существует API, используем интеграцию на уровне приложений.

Список литературы

1. G. Hohpe, B. Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, 2003. p. 683.
2. R. Miles, K. Hamilton. Learning UML 2.0. O'Reilly Media Inc., 2008. p. 290.
3. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). WC3, 2007. // URL http://www.w3.org/TR/soap12-part1 (дата обращения: 01.06.2013).
4. S. Chatterjee. Messaging Patterns in Service-Oriented Architecture. Cap Gemini Ernst & Young, 2004. // URL http://msdn.microsoft.com/en-us/library/aa480027.aspx (дата обращения: 01.06.2013).
5. E. Thomas. Service Oriented Architecture: Concepts, Technology, and Design. Indiana: Pearson Education, 2003. p. 171.
6. Andre Yee, "Demystifying Business Process Integration." EaiQ // URL
7. http://www.gartner.com (дата обращения: 01.06.2013).
8. А. Добровольский. Интеграция приложений: методы взаимодействия, топология, инструменты, 2006. // URL http://www.osp.ru/os/2006/09/3776464 (дата обращения: 01.06.2013).