

# Методы и средства интеграции информационных систем

- **Интеграция информационных систем** — это процесс установки связей между информационными системами предприятий и организаций для получения единого информационного пространства и организации поддержки сквозных бизнес-процессов предприятий и организаций. является ядром информационного пространства компании.

# Зачем нужна интеграция ИС

- Формирование единых стандартов и подходов при автоматизации бизнес-процессов на уровне функциональных подразделений компаний
- Учет и анализ характеристик используемых ИС в целях формирования программы развития информационно-технологической инфраструктуры
- Повышение прозрачности и управляемости бизнес-процессов, поддерживаемых информационными системами компаний
- Снижение вероятности появления операционных и прочих ошибок, связанных с вводом и передачей деловой информации в ручном режиме


- Сокращение длительности выполнения "сквозных" бизнес-процессов
- Сокращение трудозатрат за счет реализации однократного ввода информации и интегрированного документооборота
- Обеспечение сохранения инвестиций в информационные технологии
- Уменьшение числа ошибок во взаимодействии прикладных систем
- Снижение времени и стоимости внедрения новых систем

# Уровни Интеграционных Процессов



15.12.2013

АКБ РосЕвроБанк, Иванов С

 MyShared

# Уровни интеграции данных

- **Физический** - наиболее простая задача - конверсия данных из различных источников в требуемый единый формат их физического представления.

- **Логический** - предусматривает возможность доступа к данным, содержащимся в различных источниках, в терминах единой глобальной схемы, которая описывает их совместное представление с учетом структурных и, возможно, поведенческих свойств данных.
- **Семантический** - поддержка единого представления данных с учетом их семантических свойств в контексте предметной области.

# Ключевые различия ИС

определяются следующими их архитектурными компонентами:

- схема или модель данных интегрируемых ИС
- технологический стек, на котором реализовано приложение (базовое ПО, СУБД, сервер приложений и т.д.)
- различие в моделях бизнес-процессов и в механизмах их реализации



- **Стек** (англ. *stack* — стопка; читается *стэк*) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. *last in — first out*, «последним пришёл — первым вышел»).
- В 1946 Алан Тьюринг ввёл понятие стека<sup>[1][2]</sup>. А в 1957 году немцы Клаус Самельсон и Фридрих Л. Бауэр запатентовали идею Тьюринга<sup>[3]</sup>.
- В языке C++ стандартная библиотека имеет класс с реализованной структурой и методами<sup>[5]</sup>. И т. д.

- **Алан Мэ́тисон Тью́ринг**, **ОБЕ** (англ. Alan Mathison Turing [ˈtjʊərɪŋ]; 23 июня 1912 — 7 июня 1954) — английский математик, логик, криптограф, оказавший существенное влияние на развитие информатики. Кавалер Ордена Британской империи (1945), член Лондонского королевского общества (1951)<sup>[5]</sup>. Предложенная им в 1936 году абстрактная вычислительная «Машина Тьюринга», которую можно считать моделью компьютера общего назначения<sup>[6]</sup>, позволила формализовать понятие алгоритма и до сих пор используется во множестве теоретических и практических исследований. Научные труды А. Тьюринга — общепризнанный вклад в основания информатики (и, в частности, — теории искусственного интеллекта)



## данных

- Конфликты неоднородности (используются различные модели данных для различных источников)
- Конфликты именованя (в различных схемах используется различная терминология, что приводит к ошибкам)

- Семантические конфликты (выбраны различные уровни абстракции для моделирования подобных сущностей реального мира)
- Структурные конфликты (одни и те же сущности представляются в разных источниках разными структурами данных).

- **Абстракция.**
- При создании ПО значительная часть принадлежит абстракции. Под этим понимается подход, где при анализе какого-либо явления отбрасываются незначительные детали, которые не играют роли в контексте решения данной проблемы.
- В этом смысле абстракция в инфотехнологии схожа с абстракцией в математике, где при определении математических понятий (прямая, число, функция и т. д.) «сохраняются» исключительно те свойства реального объекта-явления, которые имеют значение при построении соответствующей математической теории.

- Примером может служить абстрактное понятие - число, которое имеет смысл, как в математике, так и в языках программирования. Представление числа в компьютере зависит как от ПО, так и от аппаратных средств, однако суть понятия это никак не меняет.
- Обобщив, абстракции в языке программирования можно поделить на две группы:
- абстракция управления (англ. Control abstraction)
- абстракция данных (англ. Data abstraction)
- В случае структурного программирования под абстракцией управления понимается систематическое использование подмодулей и команд управления (итерация, выбор, и т.д.). Под абстракцией данных понимается адекватное отражение реальных данных в структурах данных языков программирования (векторы, записи и т.д.).
- В объектно-ориентированных языках программирования абстракции данных и управления объединены.

# Система интеграции приложений обеспечивает

- Согласование данных, используемых различными приложениями
- Синхронизацию и маршрутизацию информационных потоков в соответствии с определенными бизнес-правилами
- Преобразование данных по заданным алгоритмам
- Поддержку интерфейсов к существующим промышленным системам, системам технологического уровня
- Поддержку удобного интерфейса пользователя для описания бизнес-правил и алгоритмов взаимодействия приложений
- Поддержку промышленных стандартов в области передачи и обработки данных



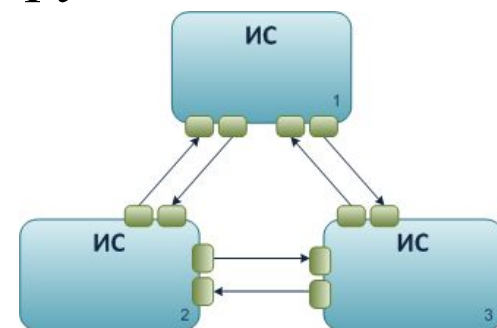
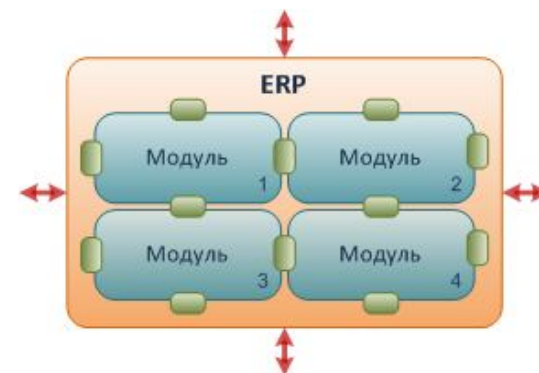
# Задачи разработки систем интеграции

- Разработка архитектуры системы интеграции данных
- Создание интегрирующей модели данных, являющейся основой единого пользовательского интерфейса в системе интеграции
- Разработка методов отображения моделей данных и построение отображений в интегрирующую модель для конкретных моделей, поддерживаемых отдельными источниками данных
- Интеграция метаданных, используемых в системе источников данных
- Преодоление неоднородности источников данных
- Разработка механизмов семантической интеграции источников данных

- **Метаданные** (от лат. *meta* — цель, конечный пункт, предел, край<sup>[1]</sup> и данные) — информация о другой информации, или данные, относящиеся к дополнительной информации о содержимом или объекте. Метаданные раскрывают сведения о признаках и свойствах, характеризующих какие-либо сущности, позволяющие автоматически искать и управлять ими в больших информационных потоках.

# Основные подходы к интеграции ИС

- построение единой корпоративной ИС
- установление прямой связи между интегрируемыми ИС (интеграция отдельных приложений)
- сервисно-ориентированное решение

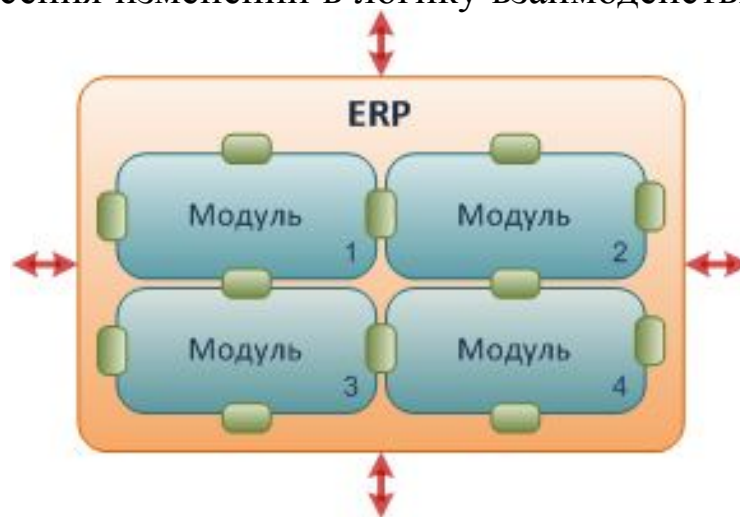


# Построение единой корпоративной ИС

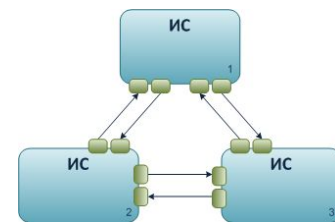
данный подход является достаточно *рискованным*, поскольку с одной стороны такое решение имеет высокую стоимость, а с другой – невозможно точно предугадать, когда вновь потребуется переработка всей структуры.

Несмотря на комплексную автоматизацию и реализацию необходимых связей между модулями, новая система, как правило, не покрывает всю прикладную область, и задачи интеграции с внешними системами остаются, хотя и в меньших объемах.

Неоспоримым плюсом является комплексная реализация необходимых связей между модулями системы еще на этапе проектирования. Отчасти благодаря этому, при использовании **Workflow**-технологий можно существенно упростить процесс разработки бизнес-сценариев и внесения изменений в логику взаимодействия модулей системы.



# Установление прямой связи между ИС



Подходит для объединения *малого количества приложений*, в этом случае стоимость подобного решения невысока, поддержка обходится относительно недорого. Но это до тех пор, пока не наступает момент обновления ИС, поскольку «наладочные работы» при замене приложения сравнимы по объему и затратам с первоначальным проектом по интеграции. Добавление каждого нового приложения ведет к увеличению затрат и существенному усложнению интеграционной модели.

Таким образом, построение интерфейсов по принципу «точка-точка» наиболее удобно в тех случаях, когда необходимо связать небольшое количество ИС, а требования к интеграции ограничиваются синхронизацией данных между ними.

Рост связей при таком подходе идет по формуле  $(N - 1) \cdot N$ , где  $N$  – число интегрируемых ИС. Такой характер увеличения связей между ИС значительно усложняет архитектуру и делает ее трудно эксплуатируемой и развиваемой.

При интеграции каждая ИС должна содержать адресную информацию о том, с кем она взаимодействует, а также выполнять преобразование форматов данных. Таким образом, вопросы согласования форматов данных, передачи данных, обеспечения надежности и безопасности должны решаться отдельно для каждой пары ИС. Это усложняет интеграционную логику при выполнении простых сценариев взаимодействия.

# ESA (Enterprise Service Architecture)

Основная ценность архитектуры ESA (или **ESB** – Enterprise Service Bus, сервисная шина предприятия) состоит в том, что она предлагает принцип более свободного соединения ИС посредством сервисов.

Благодаря таким открытым интерфейсам задача интеграции сводится к реализации единого интегрирующего компонента, который осуществляет связи ИС через их публичные сервисы. Такой подход обеспечивает слабую связанность участников обмена в рамках определенных регламентов, а, следовательно, гибкость и адаптивность всей структуры.



# ESA/ESB



Компонент **ESB** является медиатором между сервисами и обладает достаточно четким набором функциональности.

В первую очередь шина предоставляет транспорт для передачи данных от источника до получателя. Тут вместо топологии «каждый с каждым» получается топология «**hub**», где роль **hub**'а выполняет **ESB**.

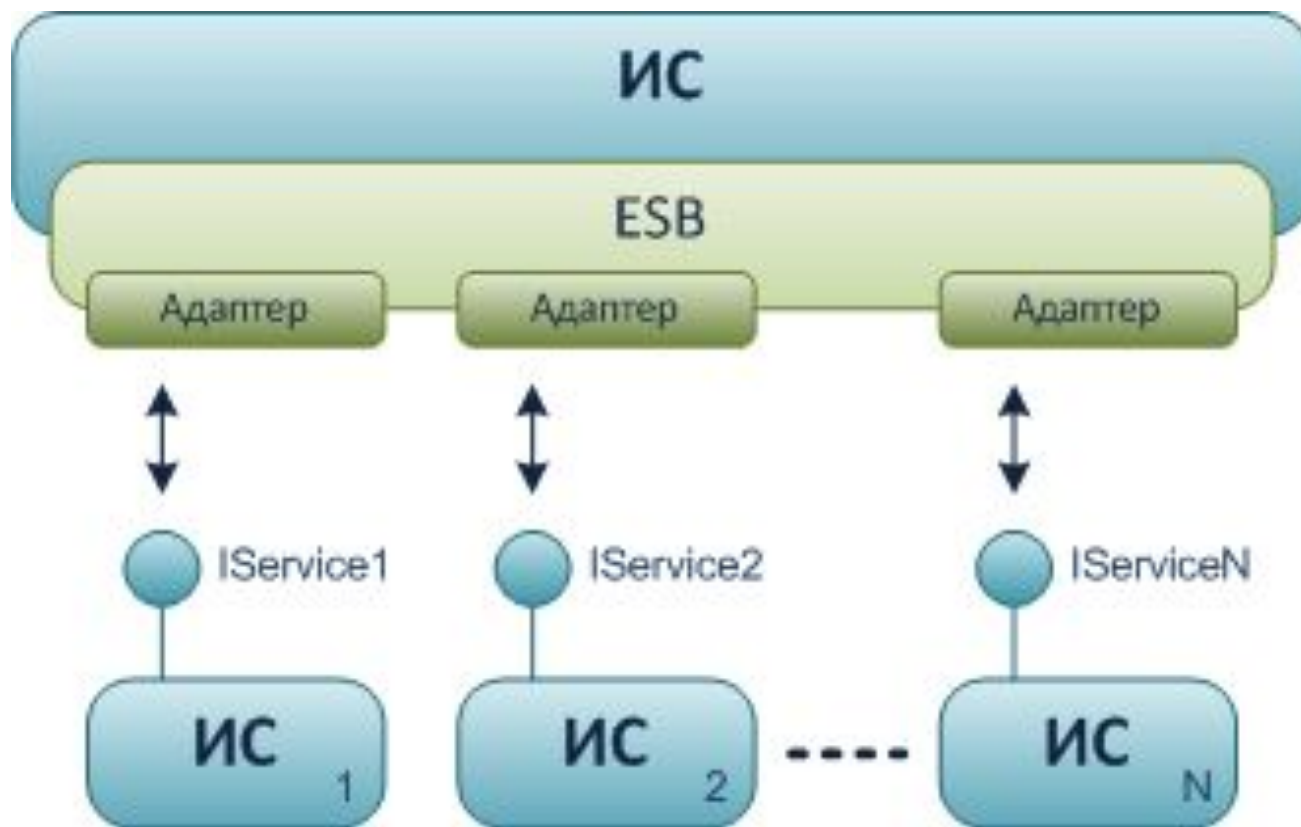
Следующий блок функциональности интегрирующего компонента – это подключение к нему сервисов. Задача **ESB** – обеспечить подключение всех «заинтересованных» систем к центральной шине и обеспечить всех необходимым форматом данных. Внутри самой шины можно использовать промежуточный формат данных (например, **XML**), который обеспечивает унифицированное представление данных и метаданных.

Сервисы подключаются к шине через *адаптеры*, трансформирующие внутреннее представление данных шины в родной формат сервиса. Адаптера может и не быть, если формат сервиса принят как стандарт внутри шины. Таким образом, достигается прозрачное соединение всех ИС через центральный узел, шину данных.

Кроме транспортной функции у шины появляется задача маршрутизации данных между источниками и приемниками. На этой стадии можно задействовать и **Workflow**-технологии.

# Комплексный подход

наиболее распространенный вариант системы интеграции, основанной на ESB



компонент ESB может быть частью одной большой информационной системы, которая осуществляет интеграцию с рядом других систем.



# Популярные методы интеграции ИС

- **обмен файлами**, в которые помещаются общие данные
- **общая база данных (БД)**, в которой сохраняется общая информация
- **удаленный вызов процедур** в рамках систем обмена сообщениями для выполнения действий или обмена данными

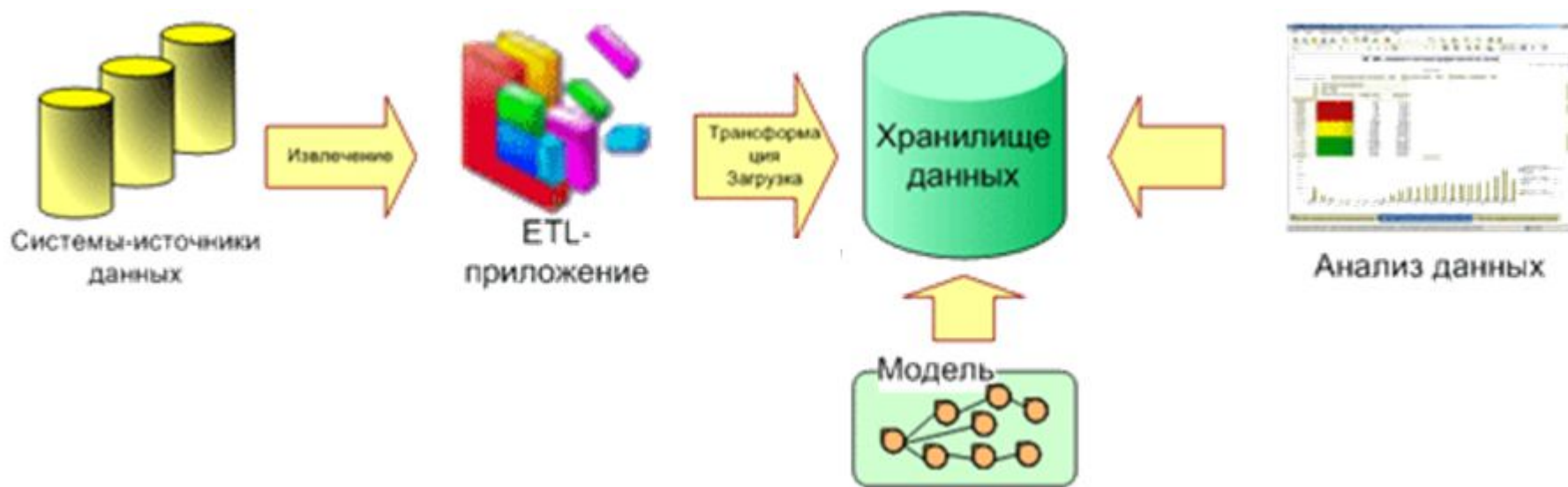
# Архитектура систем интеграции

- **Консолидация** - данные из нескольких источников сливаются в Хранилище, но не распространяются из него обратно в распределенную систему
- **Федерализация** — физического перемещения данных не происходит: данные остаются у владельцев, доступ к ним осуществляется при необходимости (при выполнении запроса)
- **Распространение данных** — двустороннее копирование данных из одного места в другое
- **Сервисно-ориентированный подход** — основан на использовании распределённых, слабо связанных заменяемых компонентов, оснащённых стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам

# Консолидация данных

- данные извлекаются из источников, и помещаются в Хранилище данных. Процесс заполнения Хранилища состоит из трех фаз — извлечение, преобразование, загрузка (**Extract, Transformation, Loading — ETL**). Во многих случаях именно ETL понимают под термином «интеграция данных».
- Распространенная технология консолидации данных — управление содержанием корпорации (**enterprise content management, ECM**). Большинство решений ECM направлены на консолидацию и управление неструктурированными данными, документы, отчеты и web-страницы. Часто консолидированные данные служат основой для приложений бизнес-аналитики (**Business Intelligence, BI**), OLAP-приложений
- При использовании этого метода существует задержка между моментом обновления информации в первичных системах и временем, когда данные изменения появляются в конечном месте хранения.

# Компоненты корпоративного хранилища данных



**ETL - Extract, Transformation, Loading** - извлечение, преобразование, загрузка

# Федерализация

- При использовании медиатора создается общее представление (модель) данных. **Медиатор** — посредник, поддерживающий единый пользовательский интерфейс на основе глобального представления данных, содержащихся в источниках, а также поддержку отображения между глобальным и локальным представлениями данных.
- Пользовательский запрос, сформулированный в терминах единого интерфейса, декомпозируется на множество подзапросов, адресованных к нужным локальным источникам данных. На основе результатов их обработки синтезируется полный ответ на запрос. Две архитектуры с посредником — **Global as View** и **Local as View**.
- Отображение данных из источника в общую модель выполняется при каждом запросе специальной оболочкой. Для этого необходима интерпретация запроса к отдельным источникам и последующее отображение полученных данных в единую модель.
- Технология интеграции корпоративной информации (**Enterprise information integration, EII**) поддерживает федеративный подход к интеграции данных.

# Распространение данных

- Перемещение данных к местам назначения зависят от определенных событий. Обновления в первичной системе могут передаваться в конечную систему синхронно или асинхронно.
- Синхронная передача требует, чтобы обновления в обеих системах происходили во время одной и той же физической транзакции. Независимо от используемого типа синхронизации, метод распространения гарантирует доставку данных в систему назначения. Такая гарантия — это ключевой отличительный признак распространения данных.
- Большинство технологий синхронного распространения данных поддерживают двусторонний обмен данными между первичными и конечными системами. Например, технологии интеграции корпоративных приложений (Enterprise application integration, [EAI](#)) и тиражирование корпоративных данных (Enterprise data replication, [EDR](#)).

# Сервис-ориентированная архитектура

- **SOA, service-oriented architecture** — модульный подход к разработке программного обеспечения, основанный на использовании распределённых, слабо связанных (*loose coupling*) заменяемых компонентов, оснащённых стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам.
- Программные комплексы, разработанные в соответствии с сервис-ориентированной архитектурой, обычно реализуются как набор веб-служб, взаимодействующих по протоколу SOAP, но существуют и другие реализации (например, на базе jini, CORBA, на основе REST).

# Отличия EAI и SOA

- **EAI, Enterprise Application Integration** – Интеграция приложений предприятия — это технологии и приложения, задача которых вовлечь несколько приложений, используемых в одной организации, в единый процесс и осуществлять преобразование форматов данных между ними.
- Под EAI понимается не архитектура, а средства интеграции, которые накладывают архитектурные ограничения.
- SOA предпочитает логику в сервисах логике в промежуточном слое, что прямо противоречит классическим подходам EAI. SOA является не технологическим, а архитектурным понятием. Но, как и любой архитектурный стиль (среди прочего), накладывает технологические ограничения.
- EAI рассматривает разные модели интеграции и чаще всего отделяет бизнес-логику от логики интеграции, то есть доступа к бизнес-сервису, предлагаемому тем или иным приложением, несущим бизнес-логику.



# Удаленный вызов процедур

- *Remote Procedure Call, RPC* — класс технологий, позволяющих программам вызывать функции или процедуры в другом адресном пространстве (на удалённых компьютерах).
- Реализация RPC технологии включает:
  - сетевой протокол для обмена в режиме клиент-сервер
  - язык сериализации объектов (или структур, для необъектных RPC).
- Различные реализации RPC имеют очень отличающуюся друг от друга архитектуру и возможности:
  - **CORBA** - *Common Object Request Broker Architecture* — общая архитектура брокера объектных запросов
  - **DCOM** - *Distributed Component Object Model*
  - **SOAP** – *Service-oriented Architecture*
  - На транспортном уровне RPC используют протоколы **TCP** и **UDP**, однако, некоторые построены на основе **HTTP**.

# Задание!

- Перечислить языки разметки для интеграции ИС и дать описания. (5-6 языков).