

---

# **Технические аспекты интеграции бизнес-приложений и построения SOA-решений**

Боев Олег

---

17.04.2010



# Основные определения

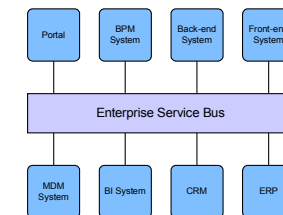
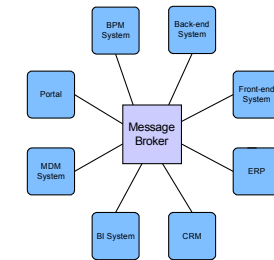
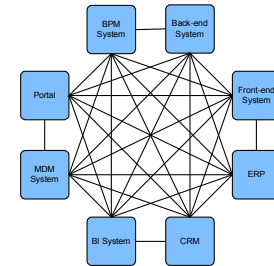
---

- **Интеграция корпоративных приложений (Enterprise Application Integration, EAI)** – это технологии и приложения, обеспечивающие взаимодействие и совместную работу нескольких приложений, используемых в одной организации, в рамках общих бизнес-процессов организации.
- **Сервис-ориентированная архитектура (Service-Oriented Architecture, SOA)** – это парадигма организации и использования распределенных информационных ресурсов, таких как приложения и данные, находящихся в сфере ответственности разных владельцев, для достижения желаемых результатов потребителем, которым может быть конечный пользователь или другое приложение.\*
- **Сервис-ориентированная интеграция (Service-Oriented Integration, SOI)** – это подход к интеграции корпоративных приложений (EAI), основанный на взаимодействии сервисов в парадигме SOA.
- **Сервис в SOA** – независимый компонент IT-инфраструктуры, реализующий бизнес-функциональность и предоставляющий стандартизованный и независимый от реализации интерфейс.

\* OASIS Reference Model for Service Oriented Architecture 1.0

# Топологии интеграции

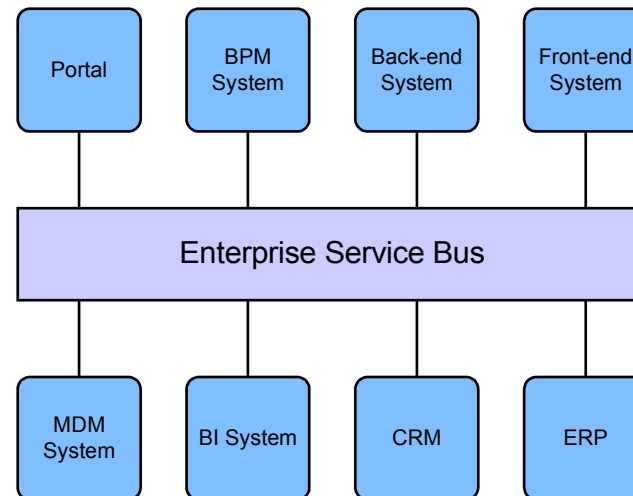
- Прямая интеграция приложений (Point-to-Point Integration)
- Брокер сообщений (Message Broker, Hub-and-Spoke)
- Интеграционная шина предприятия (Enterprise Service Bus, ESB)



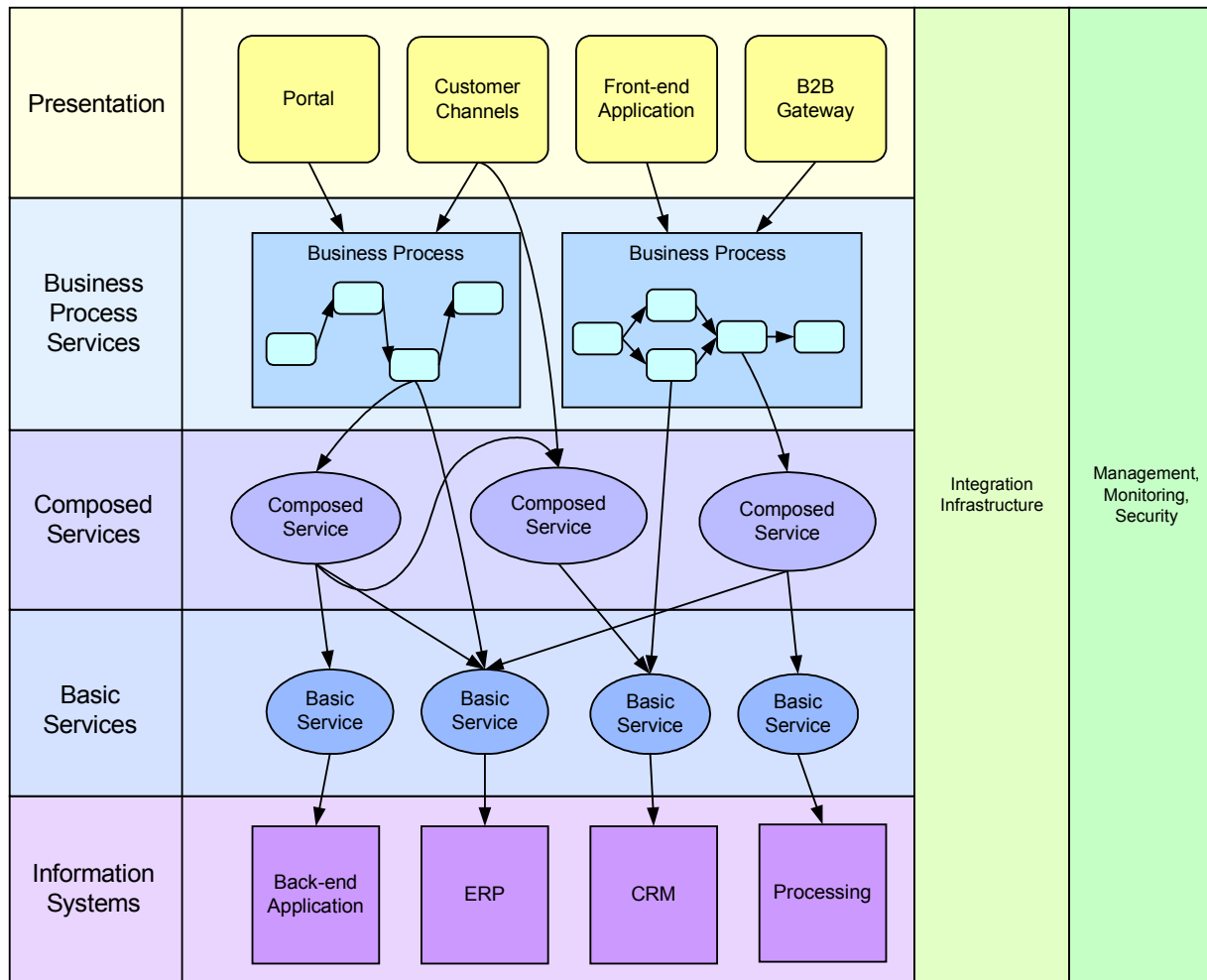
# ESB – основной компонент инфраструктуры SOA

Основная функциональность ESB:

- Подключение информационных систем
- Трансформация протоколов
- Трансформация данных
- Маршрутизация
- Обеспечение независимости от размещения, протоколов и интерфейсов при интеграции систем
- Обеспечение инфраструктуры:
  - Управление сервисами
  - Надежность, доступность и масштабируемость
  - Безопасность
  - Мониторинг
  - Логирование

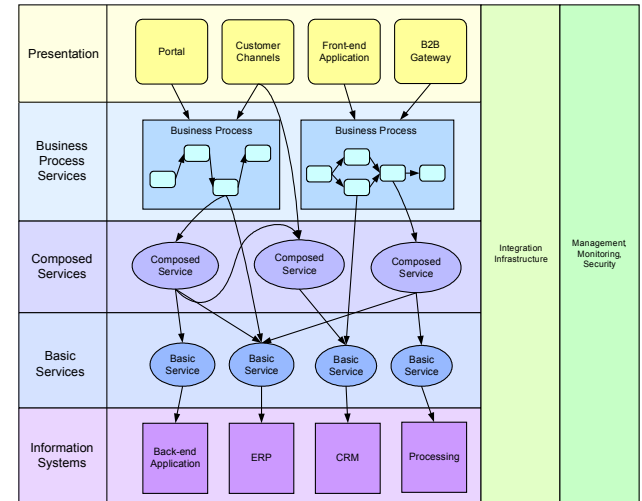


# Классификация сервисов (1)



# Классификация сервисов (2)

- Бизнес-процессы. Реализуют высокоуровневые бизнес-операции, используют композитные и базовые сервисы
  - Долгоживущие процессы с сохранением состояния
  - Для обеспечения транзакционной семантики используется механизм компенсаций
- Композитные сервисы. Предоставляют функциональность, реализованную с помощью оркестровки сервисов (orchestration)
  - Как правило, короткоживущие процессы без сохранения состояния (stateless)
  - Поддерживается транзакционная семантика операций (транзакции базовых сервисов, компенсации)
- Базовые сервисы. Предоставляют базовые операции над данными, используются сервисами более высокого уровня
  - Без сохранения состояния (stateless)
  - Поддерживается транзакционная семантика операций (транзакции информационных систем, 2PC)



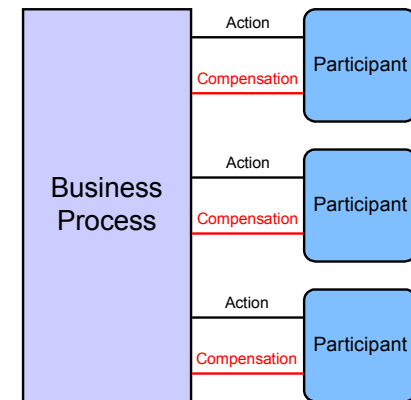
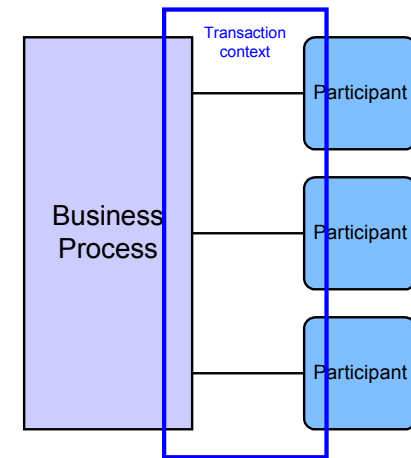
# Управление бизнес-процессами

---

- Управление бизнес-процессами (Business Process Management, BPM)  
– подходы и технические средства, предназначенные для моделирования, разработки и выполнения сквозных бизнес-процессов организации
  
- Классификация процессов:
  - По назначению:
    - Человеко-ориентированные (workflow) – обеспечивают взаимодействие сотрудников для выполнения бизнес-функций
    - Документо-ориентированные – поддерживают реализацию жизненного цикла документов
    - Интеграционные – реализуют процессы взаимодействия между информационными системами
  - По времени жизни:
    - Короткоживущие – не имеют состояния, используются для реализации композитных сервисов
    - Долгоживущие – имеют состояние, реализуют сквозные бизнес-процессы организации и сложную интеграционную логику

# Обработка транзакций

- Атомарные транзакции
  - Требует поддержки механизмов управления транзакциями со стороны участников, общего транзакционного контекста
  - Как правило, используются в короткоживущих процессах и при синхронном взаимодействии
- Механизм компенсаций («логические транзакции», «undo» вместо «rollback»)
  - Требует наличия компенсационных операций, отменяющих выполненные действия
  - Как правило, используются в долгоживущих процессах и при асинхронном взаимодействии



# Производительность

---

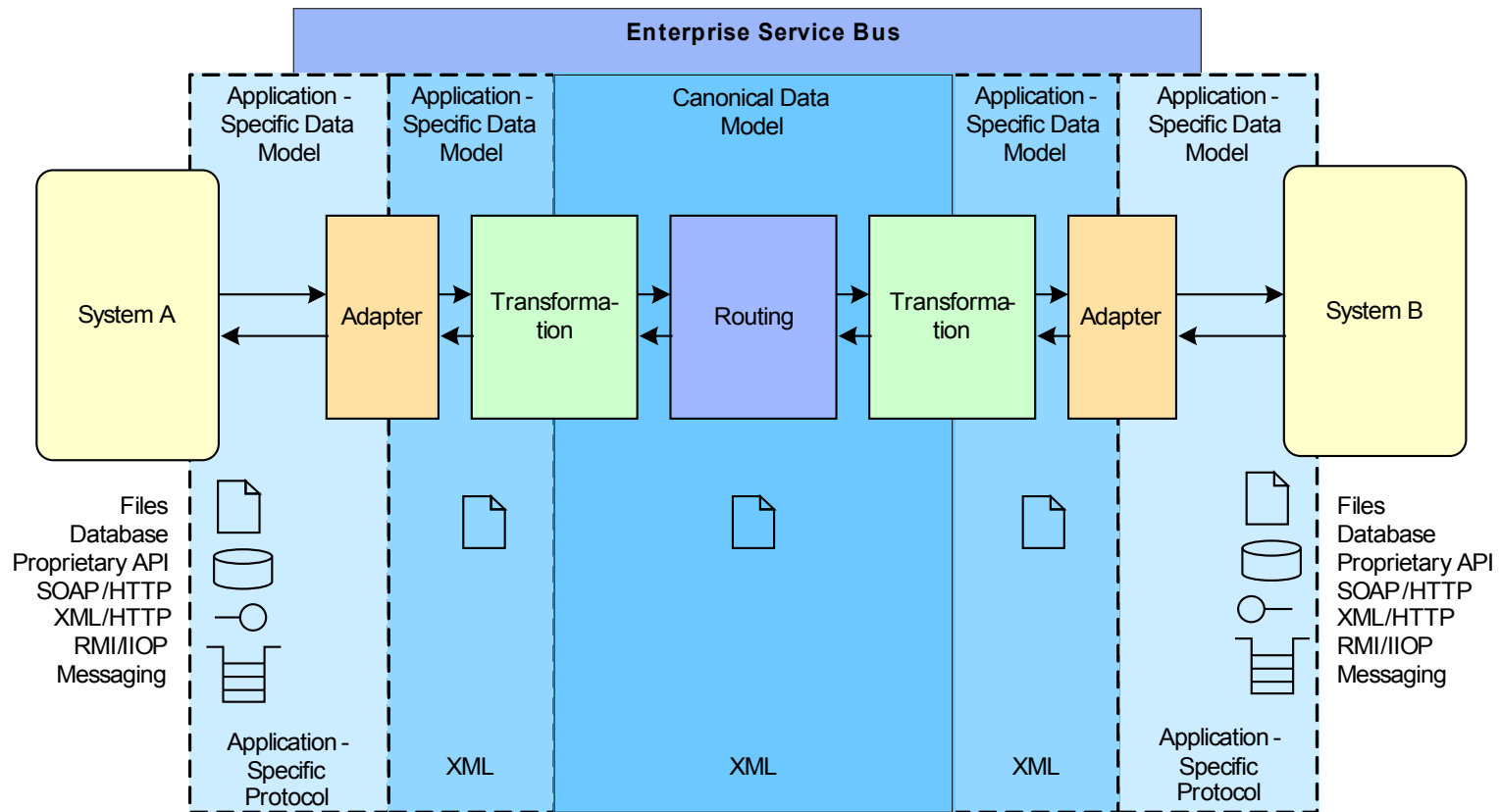
- Общая производительность интеграционного решения зависит от производительности операций интегрируемых систем
- Способы увеличения производительности:
  - Укрупнение интерфейсов сервисов
  - Переход от синхронной к асинхронной модели взаимодействия
  - Переход от атомарных транзакций к компенсациям
  - Уменьшение накладных расходов на преобразование протоколов и форматов данных, упрощение модели данных
  - Использование альтернативных механизмов интеграции:
    - Прямая интеграции между системами (point-to-point) для уменьшения накладных расходов
    - Интеграция через механизмы ETL (Extract, Transform, Load) для передачи больших объемов данных

# Идемпотентность

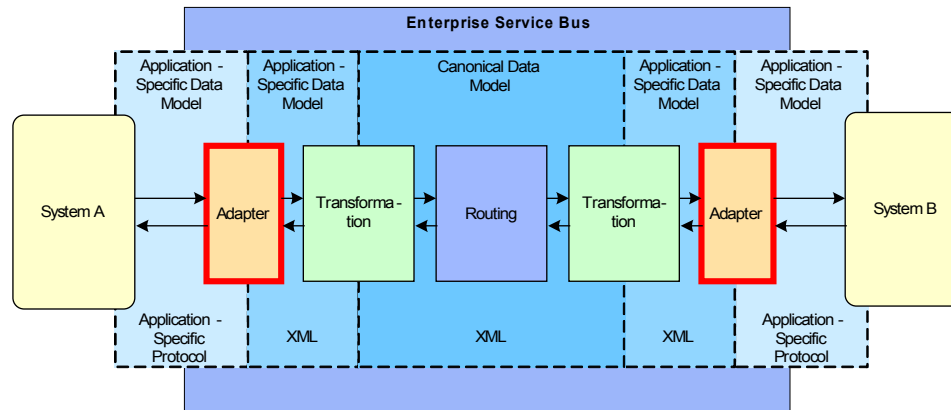
---

- Идемпотентность означает возможность повторного вызова операции без появления побочных эффектов
- Поддержка идемпотентности важна для операций, изменяющих данные
- При отсутствии подтверждения об успешном завершении операции может произойти повторный вызов этой операции
- Для предотвращения таких ситуаций сервис должен быть идемпотентным
- Примеры потенциальных проблем:
  - Повторное создание сущностей
  - Многократное изменение данных
- Возможные механизмы решения:
  - Использование уникальных идентификаторов сообщений
  - Поддержка идемпотентности на уровне семантики операции

# Поток сообщений при взаимодействии через ESB

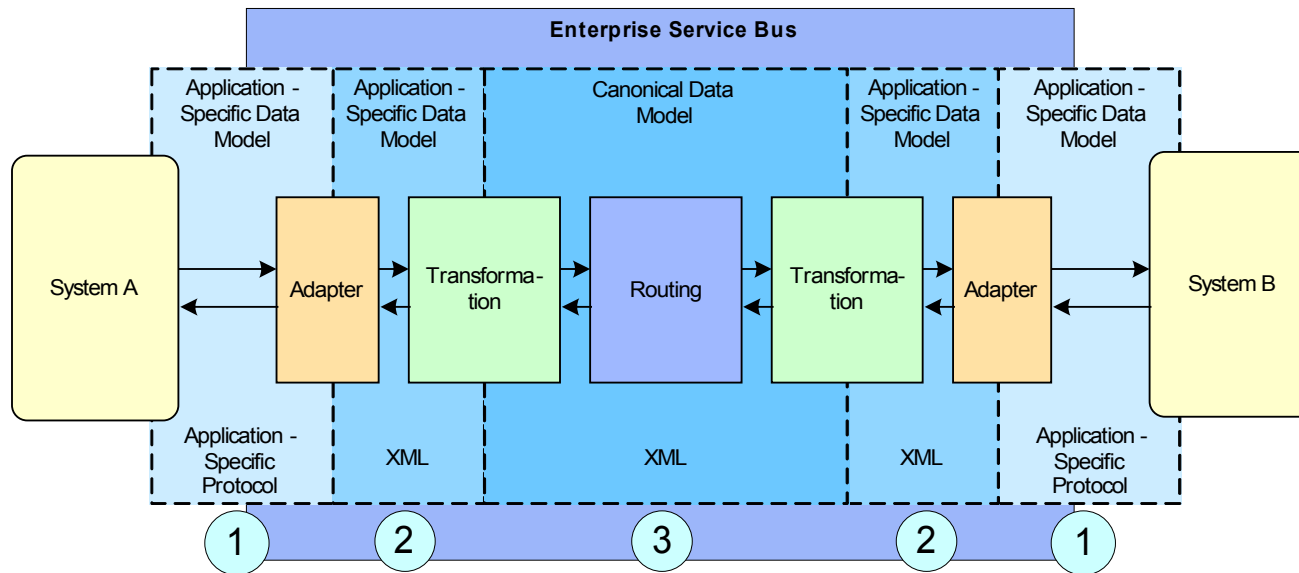


# Адаптеры



- Назначение:
  - Подключение системы к интеграционной инфраструктуре
  - Преобразование специфических протоколов интегрируемых систем к XML-сообщениям, содержащим исходные данные
  - Фасад к низкоуровневому API интегрируемой системы
  - Преобразование моделей обмена сообщениями (Message Exchange Patterns, MEP)
- Реализация:
  - Отдельный компонент интегрируемой системы
  - Отдельно размещенный программный компонент
  - Стандартный адаптер, предоставляемый ESB
  - Программный компонент (модуль), размещенный на ESB

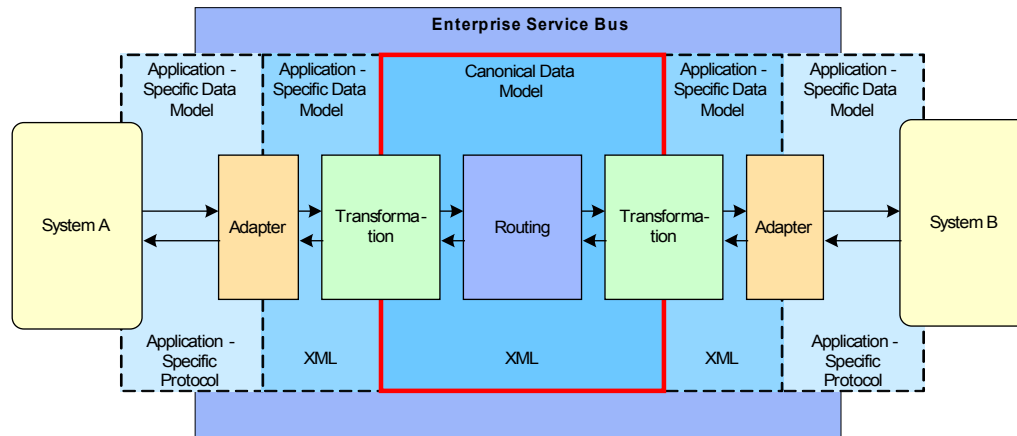
# Модели и форматы данных



№	Модель данных	Формат
1	Модель данных API приложения	Определяется API
2	Модель данных приложения	XML
3	Каноническая модель данных	XML



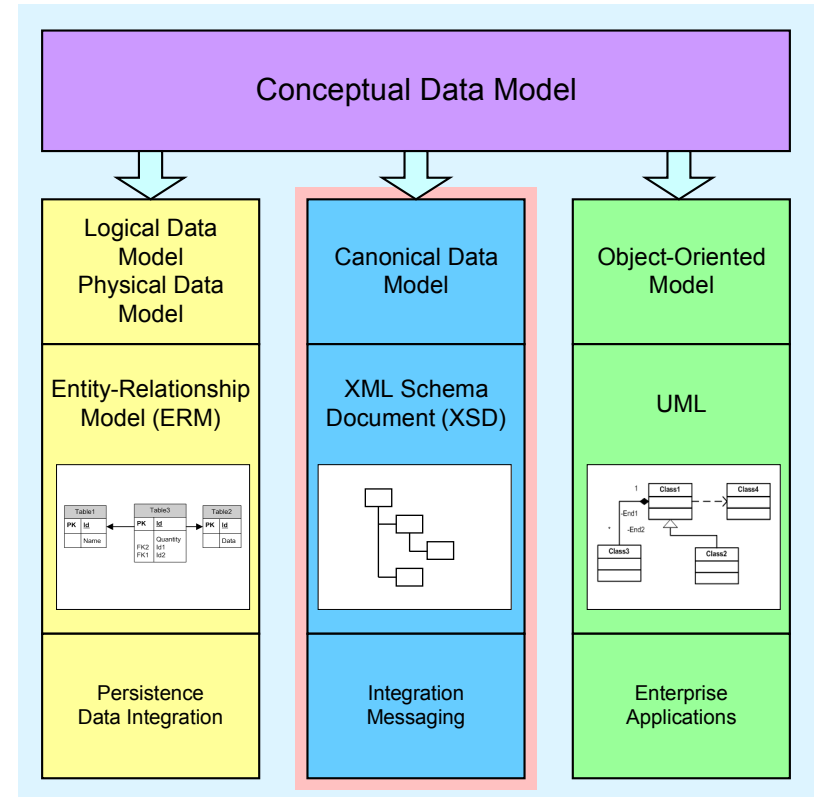
# Каноническая модель данных (1)



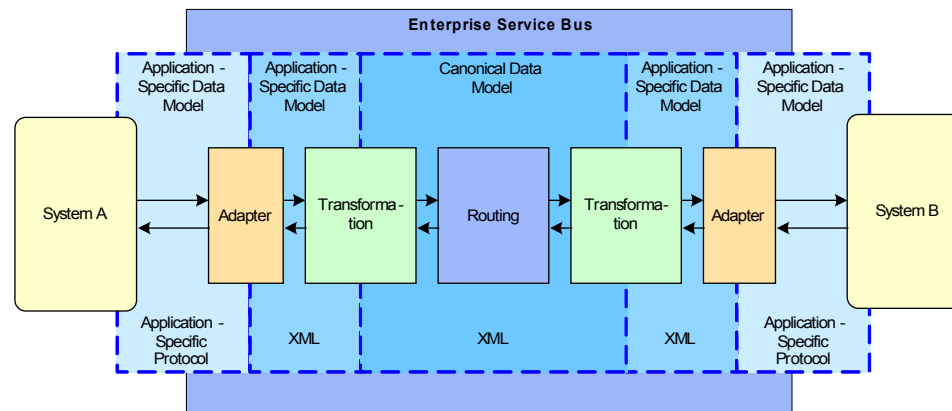
- Используется для описания интерфейсов, операций, типов данных сервисов
- Описывает основные бизнес-сущности и связи между ними, атрибуты, типы данных и т. д.
- Обеспечивает единое описание бизнес-сущностей, независимое от моделей данных отдельных приложений

# Каноническая модель данных (2)

- Основана на концептуальной модели данных предметной области организации
- Используется иерархическое и денормализованное представление данных
- Основной способ описания – документы XML Schema, которые могут импортироваться в WSDL
- Поддерживает механизмы расширения и версииности



# Версионность интерфейсов сервисов и моделей данных



- Механизмы обеспечения версионности позволяют управлять изменениями интерфейсов сервисов и моделей данных.
- Это особенно важно в интеграционной среде, в которой взаимодействуют независимые компоненты и может отсутствовать централизованное управление.
- Жизненные циклы отдельных компонентов интеграционной среды (информационных систем, приложений, бизнес-процессов, сервисов, моделей данных) не зависят друг от друга
- Подходы к обеспечению версионности:
  - Классификация возможных изменений (обратно совместимые и несовместимые, прямо совместимые и несовместимые)
  - Использование описания модели данных и интерфейсов
  - Использование технической инфраструктуры интеграции, обеспечивающей поддержку версионности (реестр сервисов, управление версиями экземпляров бизнес-процессов)
  - Использование интеграционных шаблонов (Canonical Data Model, Content-Based Routing + Format Indicator, Message Filter + Format Indicator, Message Translator)

# Совместимость версий

---

- Отсутствие совместимости



- Прямая совместимость



- Обратная совместимость



# Совместимость изменений интерфейсов (WSDL)

---

- Обрато совместимые изменения интерфейса:
  - Добавление новых операций
  - Добавление новых XML-типов, которые не включаются в существующие ранее типы
  - Добавление опциональных элементов и атрибутов
  - Преобразование обязательного элемента или атрибута в опциональный
  
- Обрато несовместимые изменения интерфейса:
  - Удаление операций
  - Переименование операций
  - Изменение параметров операции (типы данных и / или порядок)
  - Изменение структуры типов данных
  - Удаление значения enumeration
  - Добавление элементов в сообщение-ответ
  
- Обеспечение прямой совместимости - проектирование интерфейса с учетом поддержки будущих изменений:
  - Опциональные элементы и атрибуты
  - `<xsd:any>` и `<xsd:anyType>`

# Обозначение версий в описании интерфейсов (WSDL)

---

- Использование targetNamespace

```
<xsd:schema targetNamespace=  
"http://www.somecompany.com/services/service/v2.5" />
```

- Подходит для обозначения несовместимых версий
- Не поддерживает расширяемость типов, в каждой версии создаются новые типы

- Использование стандартного атрибута version в XML Schema

```
<xsd:schema version="4.1" />
```

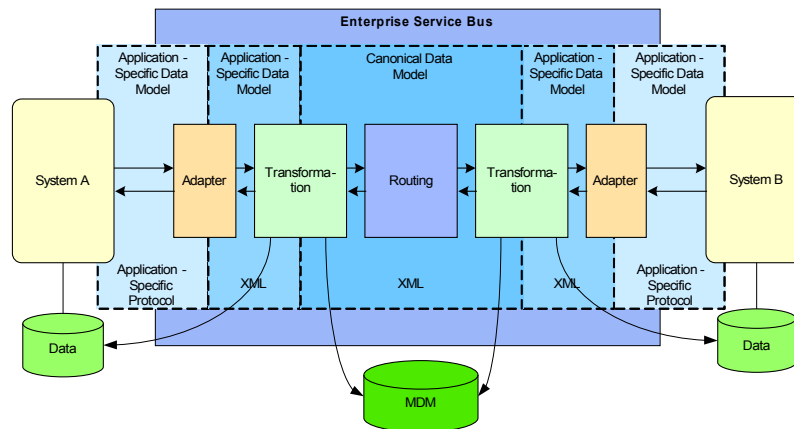
- Позволяет повторно использовать существующие типы данных
- Опциональный элемент, может игнорироваться XML-парсерами

- Использование специального элемента для обозначения версии

```
<xsd:element name="version" type="xsd:string"/>
```

- Может использоваться для маршрутизации на основе содержимого (content-based routing)

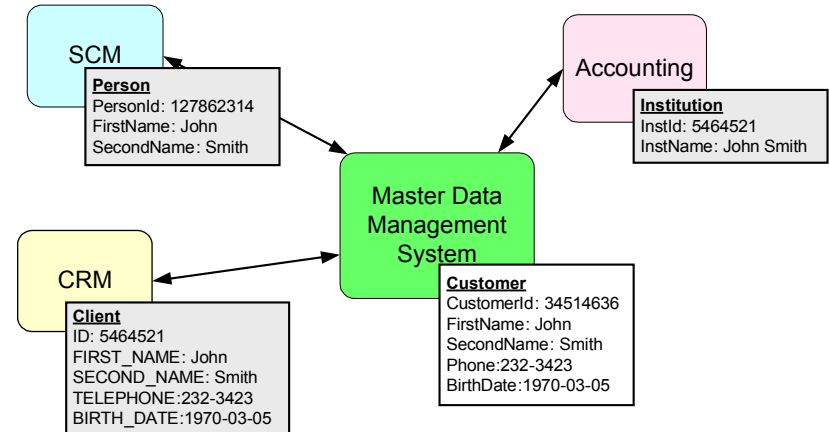
# Обогащение данных (data enrichment)



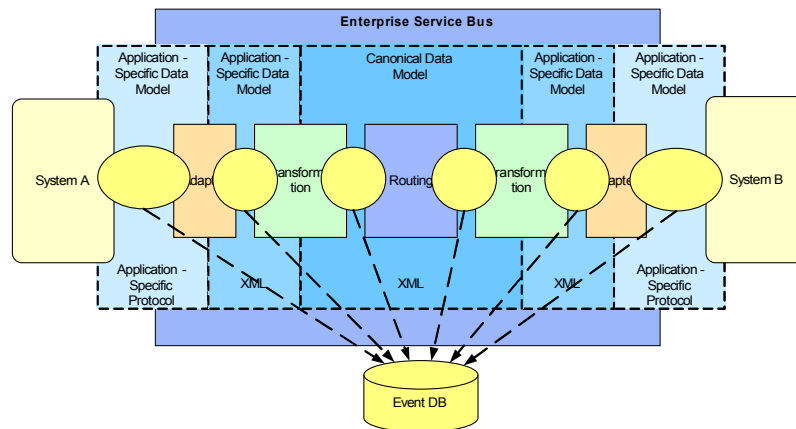
- На этапе преобразования данных (Transformation) может потребоваться выполнить обогащение данных (data enrichment):
  - Заполнение и преобразование значений справочников
  - Преобразование идентификаторов сущностей
  - Наполнение сообщений дополнительными данными
- Реализация:
  - Использование системы управления мастер-данными (Master Data Management, MDM)
  - Доступ к данным отдельных информационных систем (система-источник, система-получатель)
  - Хранение подмножества мастер-данных на уровне интеграционной инфраструктуры (ESB)

# Управление мастер-данными

- Мастер-данные (также основные данные, нормативно-справочная информация, НСИ) – единый синхронизированный набор основных данных бизнеса, используемых практически во всех информационных системах и бизнес-процессах организации. Примеры:
  - справочные данные
  - данные об основных бизнес-сущностях (продукты, клиенты, поставщики, сотрудники и т. д.)
  - данные об организации (сотрудники, подразделения, организационная структура)
- Управление мастер-данными (Master Data Management, MDM) – централизованное хранение, сбор, обработка и распространение мастер-данных на уровне всей организации.
- Система управления мастер-данными – ключевой компонент инфраструктуры SOA, обеспечивающий механизмы интеграции корпоративной информации

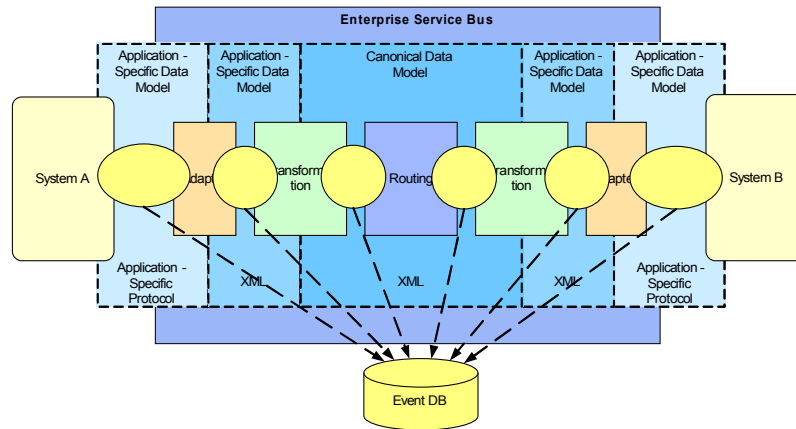


# Журналирование и аудит событий



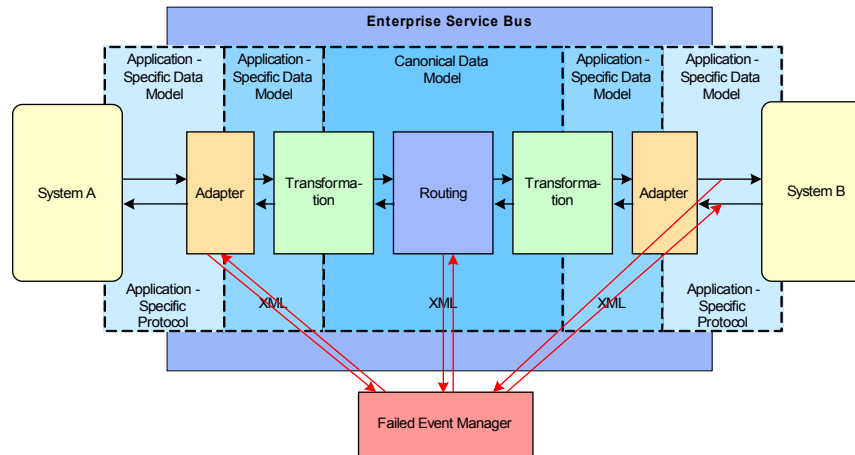
- Важная часть интеграционной инфраструктуры – журналирование и трассировка событий:
  - Интеграционная инфраструктура является центральным звеном, реализующим критичную для бизнеса функциональность и сквозные бизнес-процессов
  - Возможность анализа причин некорректной работы
  - Возможность использования для реализации бизнес-мониторинга и мониторинга работоспособности решения
- Реализация:
  - Как минимум, журналирование сообщений между ESB и информационными системами.
  - Использование функциональности журналирования интеграционной платформы (ESB)
  - Реализация специальных сервисов журналирования, БД и отчетов

# Информация для журналирования



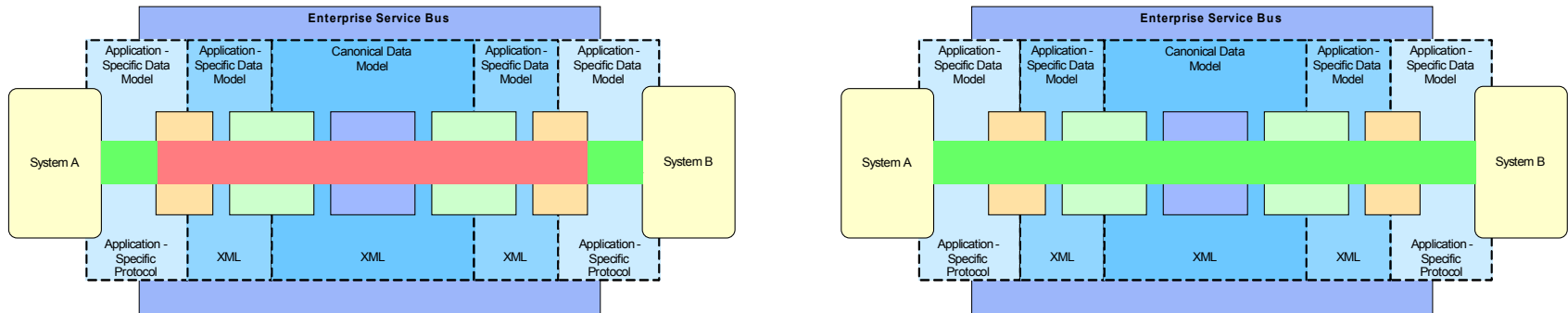
- Рекомендуемый набор информации о событии для журналирования:
  - Время события
  - Идентификатор сообщения
  - Идентификатор процесса
  - Идентификатор бизнес-сущности
  - Источник сообщения
  - Приемник сообщения
  - Выполняемая операция
  - Содержимое сообщения

# Обработка ошибок



- **Функциональность:**
  - Журналирование информации об ошибках
  - Сохранение исходного сообщения в постоянном хранилище
  - Возможность повторной отправки сообщения
  - Возможность корректировки данных сообщения
- **Реализация:**
  - Использование механизмов интеграционной платформы (Failed Event Manager, FEM)
  - Реализация потоков обработки ошибок и механизмов хранения ошибочных сообщений (например, используя шаблон Dead Letter Channel)

# Безопасность передачи данных



- **Безопасность на уровне транспорта (Transport-level Security)**
  - Реализуется на уровне транспортного протокола (SSL)
  - Обеспечивает безопасную передачу данных только в пределах одного соединения
  - Не предотвращает доступ к конфиденциальной информации сообщения при передаче через посредников
  - Сообщение с конфиденциальной информацией может быть сохранено во временных очередях, БД, журналах, что увеличивает риск несанкционированного доступа
- **Безопасность на уровне сообщения (Message-level Security)**
  - Реализуется с использованием стандартов XML Encryption, WS-Security
  - Как правило, интеграционная платформа поддерживает встроенную реализацию XML Encryption, WS-Security
  - Обеспечивает шифрование конфиденциальной информации на уровне сообщения
  - Поддерживает независимую защиту разных элементов одного сообщения

# Выводы, рекомендации

---

- Применение централизованного подхода к управлению SOA (SOA Governance) и управлению интеграцией
- Стратегическое планирование развития интеграционной инфраструктуры
- Использование общих стандартов, подходов и методов
- Применение подходящих технологий и средств, которые потенциально могут иметь пересекающиеся возможности (ESB, BPM, ETL, ECM и др.)
- Понимание основных отличий разработки интеграционных решений от разработки корпоративных приложений:
  - Реализация критичной для бизнеса функциональности
  - Распределенная среда
  - Множество заинтересованных лиц
  - Отсутствие централизованного контроля за всеми участниками интеграционной среды
  - Независимые жизненные циклы участвующих информационных систем
  - Различие в данных, форматах, протоколах, технологиях и т. д.

# Дополнительные источники

---

1. Arsanjani A. Toward a pattern language for Service-Oriented Architecture and Integration, Part 1: Build a service eco-system. <http://www.ibm.com/developerworks/webservices/library/ws-soa-soi/>
2. Bieberstein N., Bose S., Fiammante M, Jones K., Shah R. Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap. IBM Press, 2005.
3. Byrne B., Kling J., McCarty D., Sauter G., Worcester P. The information perspective of SOA design, Part 4: The value of applying the canonical modeling pattern in SOA. <http://www.ibm.com/developerworks/data/library/techarticle/dm-0803sauter/index.html>
4. Dichmann D. PowerDesigner for Enterprise Informaiton Architecture Implementation. <http://www.sybase.ru/dl/files/10Feb10/2%20PowerDesigner%20for%20Enterprise%20Information%20Architecture%20Implementation%20v1.pdf>
5. Erl T. SOA Design Patterns. Prentice Hall, 2009.
6. Erl T. SOA: Principles of Service Design. Prentice Hall, 2007.
7. Hohpe G., Woolf B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, 2003.
8. Josuttis N. SOA in Practice. The Art of Distributed System Design. O'Reilly, 2007.
9. OASIS Reference Model for Service Oriented Architecture 1.0. [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm)
10. Roshen W. SOA-Based Enterprise Integration: A Step-by-Step Guide to Services-Based Application Integration. McGraw-Hill Osborne Media, 2009.
11. ZapThink's Service-Oriented Architecture Roadmap. <http://www.zapthink.com/2005/10/31/zapthinks-service-oriented-architecture-roadmap/>

---

**Благодарю за внимание**

Боев Олег  
oboyev@gmail.com

---

**17.04.2010**

