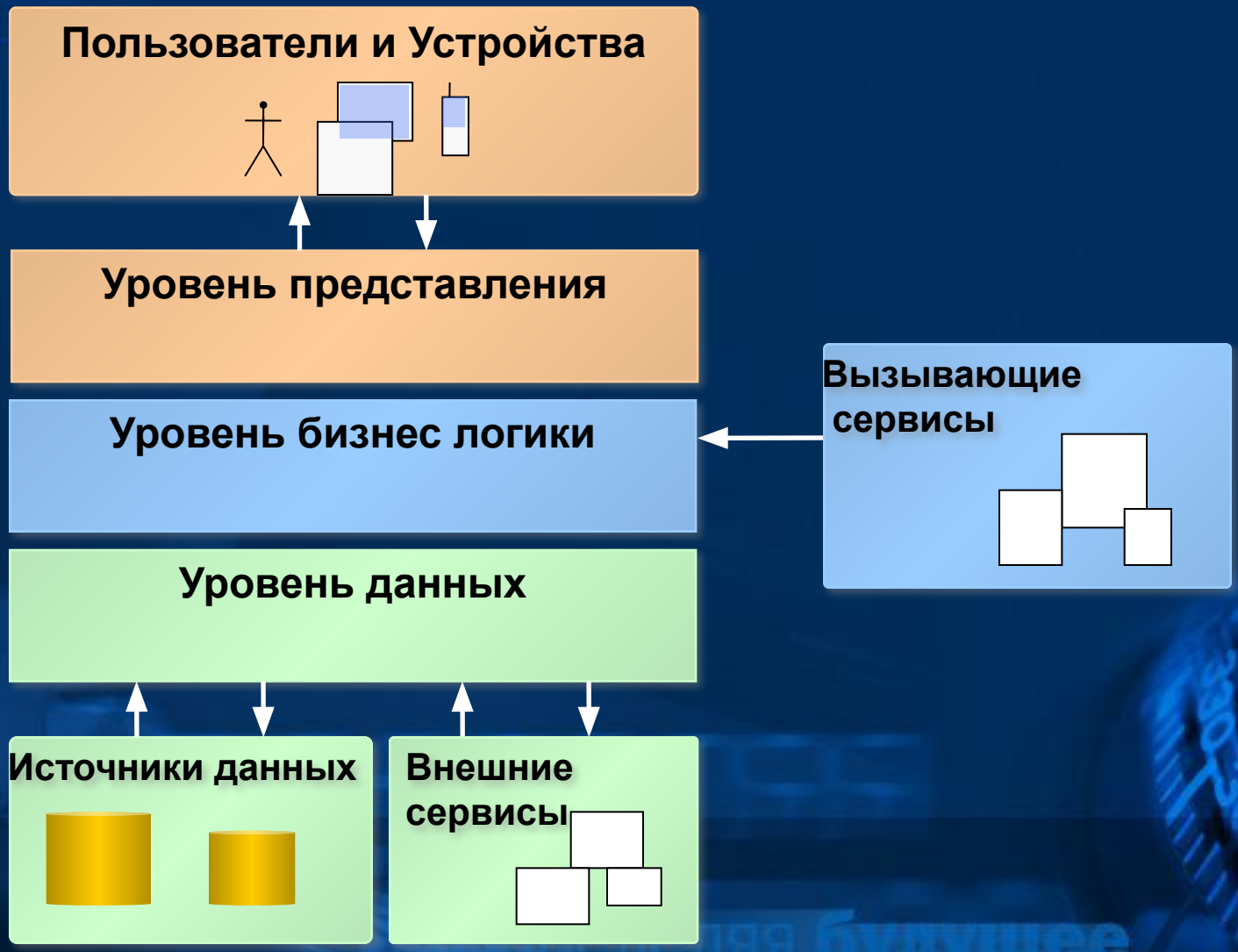
Microsoft Платформа 2003

Архитектура и проектирование распределенных .NET приложений

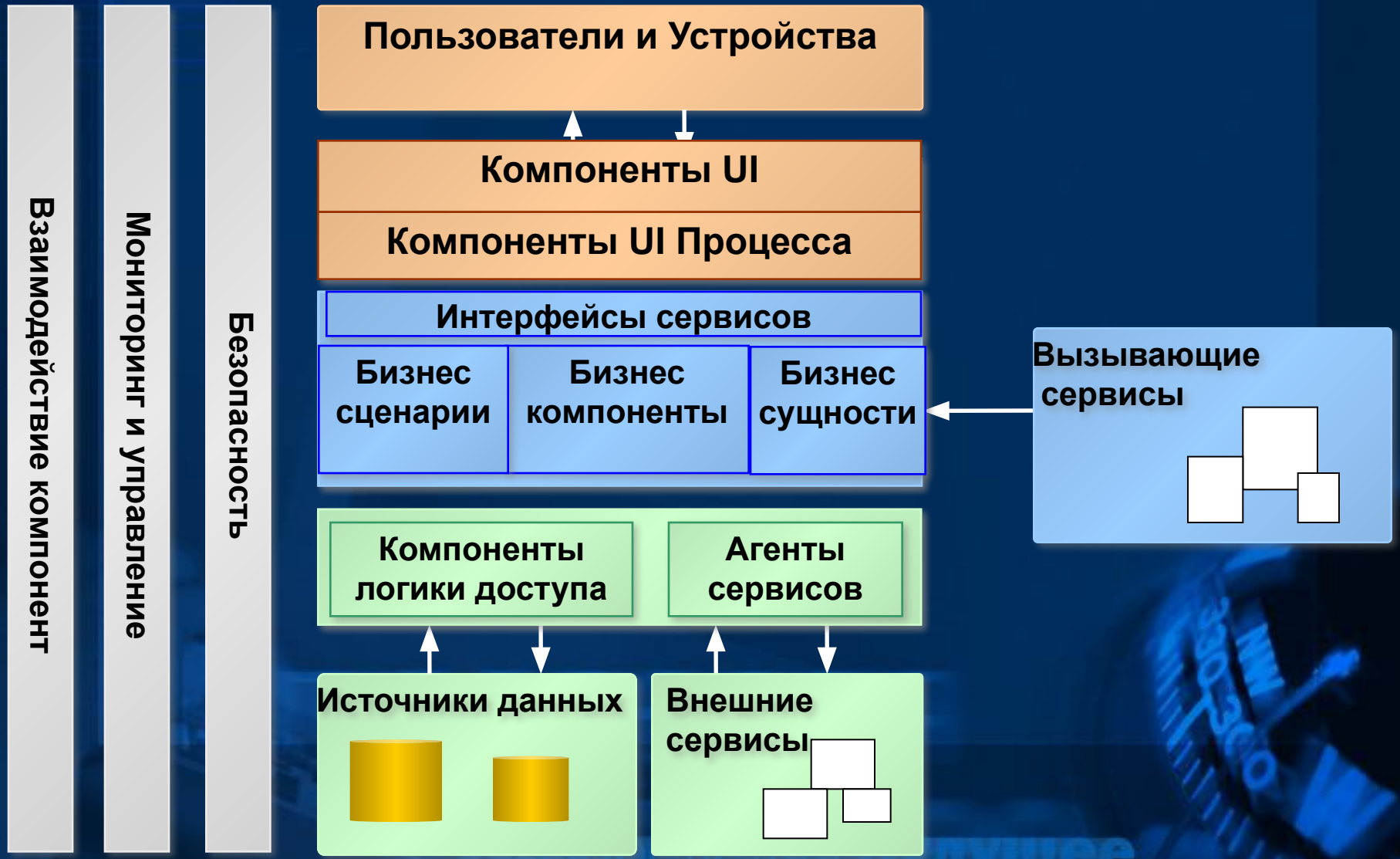
Дмитрий Старостин

Определяя будущее

Многоуровневое приложение



Категории компонент



Содержание

- Категории компонент на уровнях
  - **Представления**
  - Бизнес логики
  - Обращения к данным
- Политики
  - Безопасности
  - Управления и мониторинга
  - Взаимодействия

Компоненты пользовательского интерфейса

- Компоненты пользовательского интерфейса:
  - Отображают данные
  - Принимают введенные данные от пользователя
  - Проверяют введенные данные
  - Не инициализируют, не участвуют и не голосуют в транзакциях
  - Могут содержать ссылку на компоненту UI процесса
  - Могут содержать и функциональность отображения и функциональность управления

Пользовательский интерфейс Windows

- Развитый интерфейс настольного приложения, построенный с Windows Forms
- Внедренный HTML
- Дополнительные модули к приложениям

Пользовательский интерфейс Windows

- Используйте связывание для синхронизации данных между одновременно открытыми формами
- Избегайте использования в коде жестких связей между формами
- Реализуйте обработчики ошибок в формах
- В пользовательских контролах делайте публичными только необходимые методы
- Не реализуйте функции управления внутри обработчиков событий

Web пользовательский интерфейс

- ASP.NET предоставляет мощный framework
  - Единая интегрированная среда для разработчика
  - Событийная модель программирования
  - Связывание данных на уровне пользовательского интерфейса
  - Доступ к интегрированной модели безопасности в .NET Framework
  - Богатые возможности кеширования и управления
  - Доступность, производительность и масштабируемость Web обработки

Web пользовательский интерфейс

- Реализуйте пользовательскую страницу обработки ошибки и глобальный обработчик ошибок в Global.asax
- Используйте инфрастуктуру проверки ASP.NET для оптимизации задачи проверки вводимых данных
- Используйте Web User Controls
- Осуществляйте переходы между страницами из компонент UI процесса
  - Получите контекст и вызовите Redirect
- Функции управления реализуйте как отдельный функции на ASP.NET странице или в отдельных .NET классах
- Используйте ASP.NET view state для хранения состояния страницы

Web пользовательский интерфейс

- Расширяемый framework для объединения Web приложений
  - Microsoft Content Management Server
  - Microsoft SharePoint Portal™ Server 2002
  - IBuySpy Portal
    - http://msdn.microsoft.com/library/en-us/dnbda/html/bdasampibsport.asp

Интерфейс для мобильных устройств

Web интерфейс

- Mobile Internet Toolkit
  - В модели ASP.NET
  - Элементы управления Web-форм
  - Один исходный код – множество целевых устройств (HTML, WML, cHTML)
  - Поддержка множества мобильных устройств
    - Определение возможностей устройства
    - Шаблоны для более чем 180 устройств
    - Обобщенные шаблоны
    - Расширяемая модель под новые типы устройств
  - Add on к Visual Studio .NET - Mobile Internet Designer

Интерфейс для мобильных устройств

интерфейс для интеллектуального мобильного устройства

- **Smart Device Extensions**
  - .NET Compact Framework
  - Интеграция с Visual Studio.NET
  - Оптимально для online и offline решений
  - Использует все возможности Pocket PC
  - Отличная интеграция с SQL Server™ CE
  - Мощная локальная обработка, включая графику, мультимедиа

Реализация интерфейса на основе документов

- **Работа с документом извне**
  - Ввод: заполнение документа, отправка в приложение, разбор документа
  - Представление: генерация документа
- **Работа с документом изнутри**
  - Ввод: формы и макросы внутри документа, передача данных компонентам бизнес логики
  - Представление: расширение документа, Smart Tags

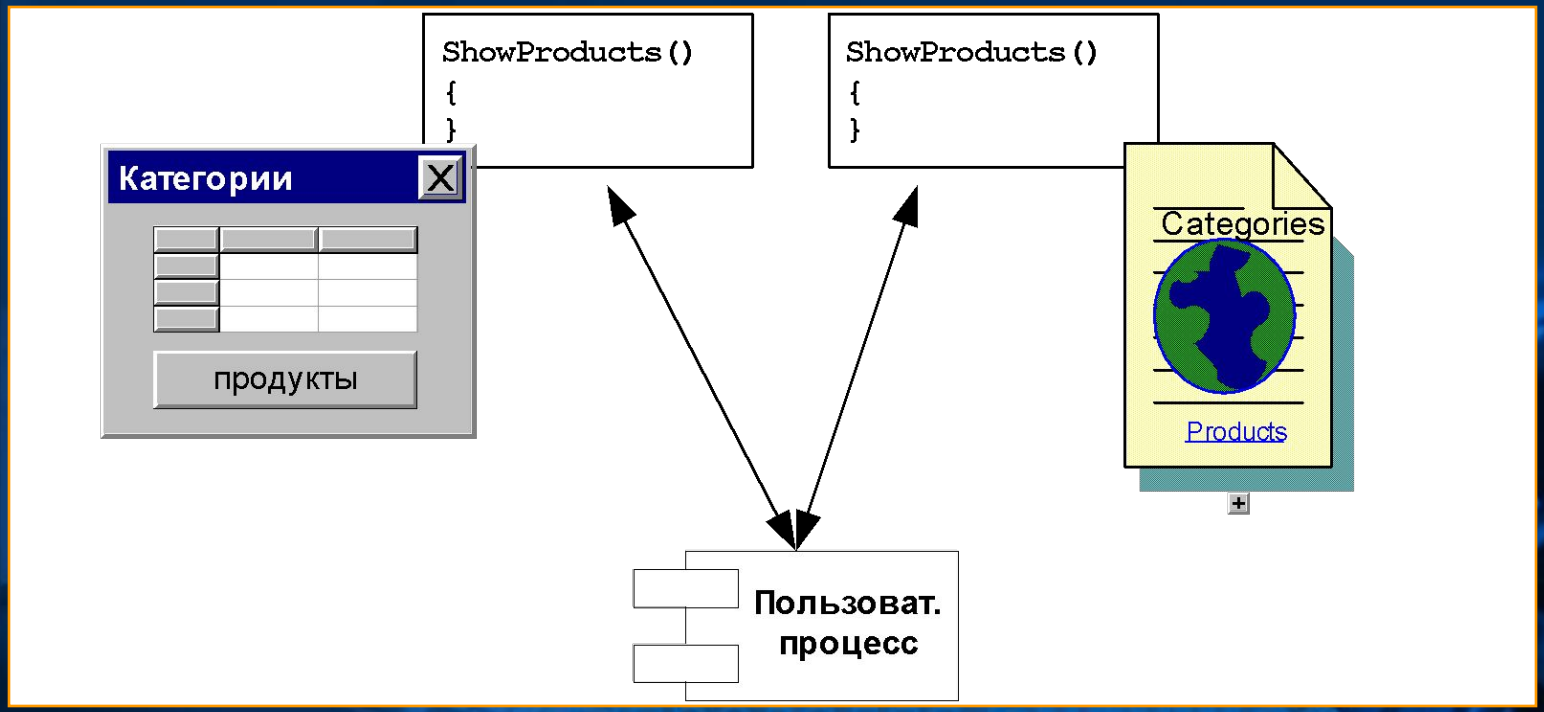
Уровень представления

Обращение к компонентам доступа данным из уровня представления

- Может применяться в определенных сценариях
  - Семантика пользовательского интерфейса тесно связана со схемами и методами доступа данных
  - Физически компоненты UI и компоненты доступа к данным расположены вместе
  - Для доступа к данным не требуется авторизации или дополнительной обработки на уровне бизнес логики

Разработка компонент процесса UI

- Поддержка состояния длинных UI взаимодействий с пользователем
- Использование несколькими интерфейсами одного UI процесса



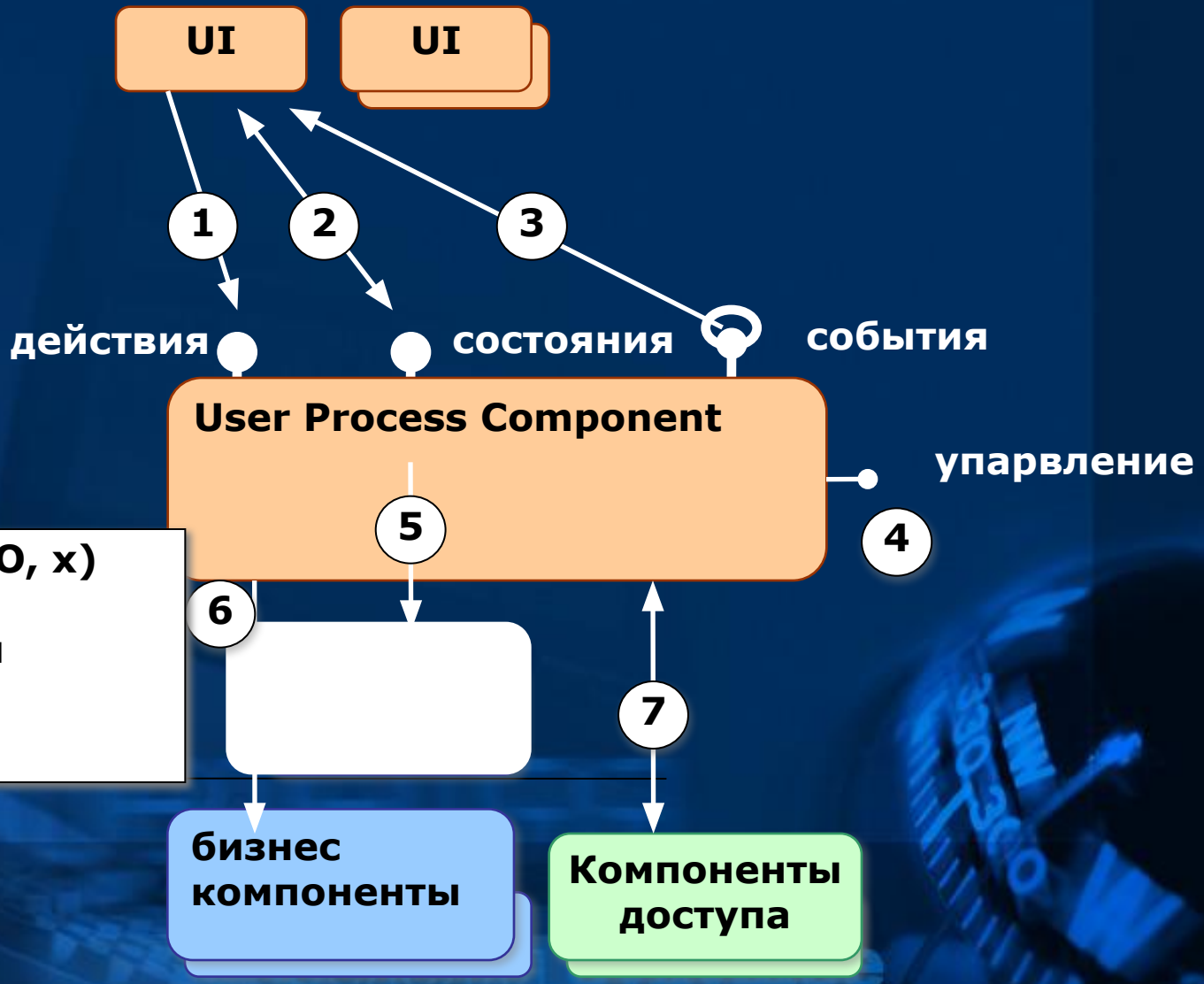
Разработка компонент процесса UI

- Обработка конкурентных пользовательских активностей
- Использование нескольких панелей отображения для одной активности
- Изолирование длинных пользовательских интерфейсных активностей от состояния бизнес компонент

Отделение UI процесса от UI интерфейса

- Идентифицируйте бизнес процессы, в которых участвует пользовательский интерфейс
- Идентифицируйте данные, необходимые для бизнес процессов
- Идентифицируйте дополнительное состояние, необходимое для поддержания пользовательской активности
- Разработайте визуальное представление пользовательского процесса

Проектирование компонент процесса интерфейса




```
Void Buttin1_Click(O, x)
{
  // код обработки
}
```

Общие рекомендации для компонент процесса интерфейса

- Выделение UI компонент процесса при
  - Насыщенном UI
  - Высокой вероятности изменения UI
- Выбор хранилища состояний для компонент UI
- Проектируйте компоненты сериализуемыми
- Включайте обработку исключений и распространяйте исключения на уровень UI компонент

Содержание

- Категории компонент на уровнях
  - Представления
  - **Бизнес логики**
  - Обращения к данным
- Политики
  - Безопасности
  - Управления и мониторинга
  - Взаимодействия

Бизнес компоненты и сценарии

- Используйте бизнес сценарии когда :
  - Нужно управлять процессом, состоящем из множества шагов и длинных транзакций
  - Нужно раскрывать интерфейс, позволяющий Вашему приложению участвовать во взаимодействии с другими приложениями или сервисами
  - Есть возможность использования для доступа к приложениям адаптеров из широкого списка поддерживаемых серверами интеграции
- Реализуйте бизнес процессы только через бизнес компоненты, когда:
  - Бизнес функциональность может быть реализована в виде атомарной транзакции
  - Нужно инкапсулировать функциональность и логику, которая может использована во внешних бизнес процессах
  - Бизнес логика требует интенсивных вычислений или тонкого контроля над структурами данных API

Рекомендации для компонент бизнес логики

- Максимально используйте ориентированные на передачу сообщений взаимодействия
- Обеспечьте устойчивость интерфейсов к непоследовательным воздействиям
- Тщательно выбирайте границы транзакций
- Обеспечьте возможность вызова из сервиса (без передачи контекста пользователя)
- Последовательно используйте форматы данных для входных и выходных параметров

Использование шаблона конвейерной обработки

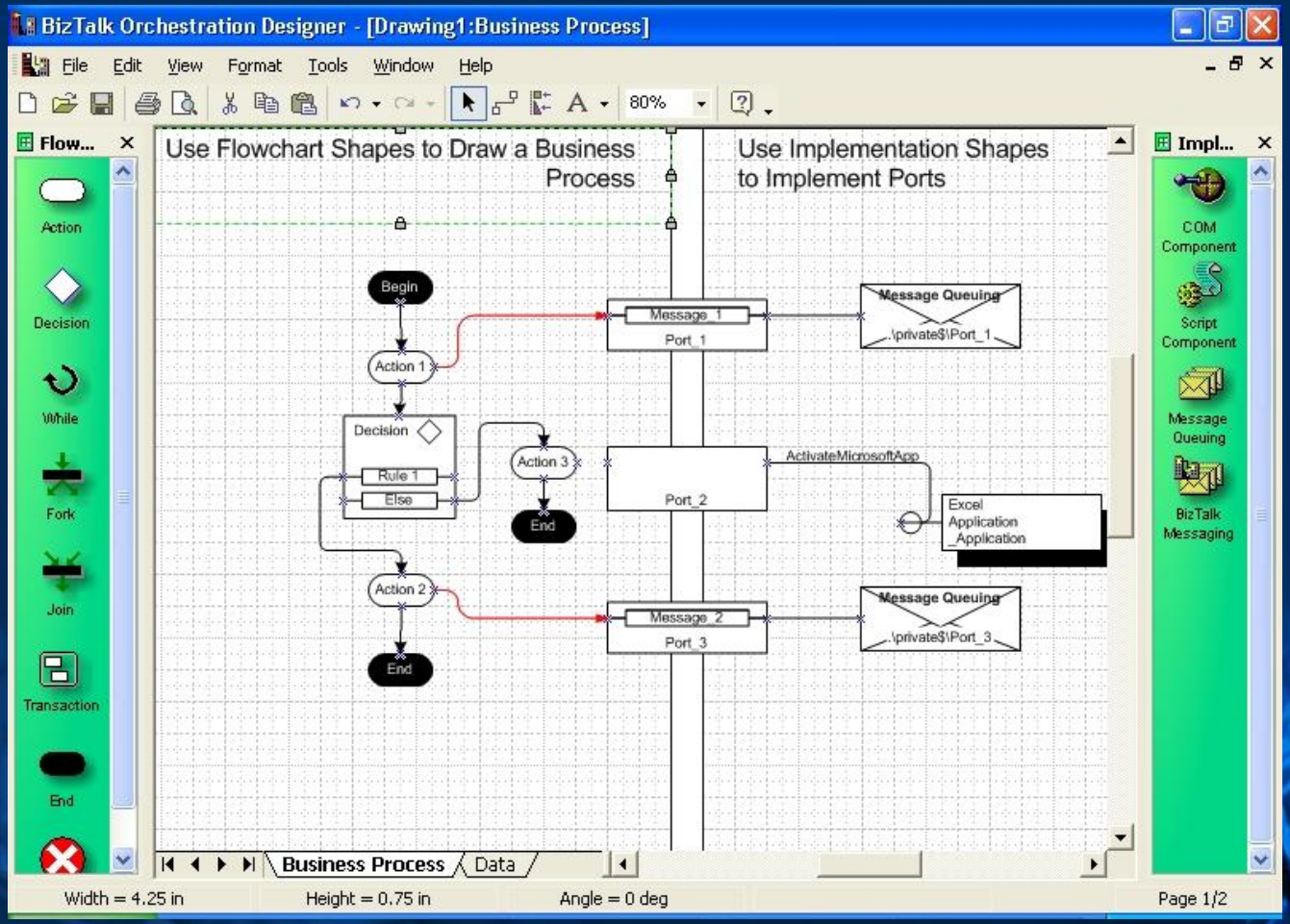
- Используйте шаблон конвейерной обработки, когда:
  - Возможно специфицировать последовательность заранее известных шагов
  - Нет необходимости ожидания асинхронного ответа на каждом шаге
  - Все нижележащие по уровню компоненты могут преобразовывать данные от вышележащих компонент
- Преимущества конвейерной обработки :
  - Простота понимания и реализации
  - Усиливает последовательную обработку
  - Легко оборачивать в атомарные транзакции
- Недостатки конвейерной обработки :
  - Слишком упрощает ситуацию, особенно для сложных сценариев взаимодействия компонент
  - Не предоставляет возможности обработки условных конструкций, циклов и других элементов управления потоком выполнения
  - Добавление нового шага может воздействовать на производительность всей цепочки

Использование шаблона событийной обработки

- **Используйте шаблон событийной обработки**:
  - необходимо управлять независимыми и изолированными реализациями специфической функциональности
  - Ответ от одной реализации не воздействует на работу другой
  - Все реализации только сохраняют результаты или вызывают метод без ожидания результата
  - Результат бизнес процесса не определяется ни одном шагом либо определяется только на одном специфическом шаге
- **Преимущества событийной обработки:**
  - Позволяет поддерживать несвязанные бизнес процессы независмо
  - Позволяет параллельную обработку, что может значительно повышать производительность
  - Легко оборачивать в атомарную транзакцию
  - Нейтральна по отношению к синхронным и асинхронным реализациям – не ожидает ответ
- **Недостатки событийной обработки**:
  - Не позволяет строить комплексные ответы для бизнес функций
  - Компонент не может использовать данные или статус другой компоненты

Реализация бизнес процесса

Реализация бизнес процесса через BizTalk Orchestration



Разработка интерфейса сервиса

- Интерфейс сервиса является точкой входа в приложение
- Реализация интерфейс сервиса должны быть абстрагирована от реализации бизнес логики
- Для компоненты бизнес логики может быть разработано и опубликовано несколько интерфейсов, предназначенных для разных типов вызывающих приложений
- Различные сервисные интерфейсы могут реализовываться для разных
  - протоколов взаимодействия
  - форматов сообщений
  - схем аутентификации
  - Service Level Agreement (SLA)
  - Возможности поддержки транзакций

Разработка интерфейса сервиса

Интерфейсы сервисов

- Помогают изолировать изменения в бизнес логики
  - Повышают эксплутационную устойчивость
  - Могут использовать возможности .NET инфрастуктуры для прозрачного преобразования
    - Выставление Enterprise Services объектов как Web services в Windows .NET Server
- Могут реализовывать
  - Кеширование
  - Отображение, простые преобразования форматов и схем
- Не должны реализовывать бизнес логику
- Могут включать поддержку транзакций
  - Зависит от протокола
  - Воздействует на стратегию управления ошибками и транзакциями
- Должны разрабатываться с максимальной возможностью организации взаимодействия с другими сервисами (в том числе и на других платформах)
  - Опора на стандарты
  - Реализация платформо независимых схем аутентификации
- Могут выполняться под собственным идентити без имперсонации оригинального вызывающего

Бизнес фасад с интерфейсами сервиса

Представление данных и передача их между уровнями

- XML
- DataReader
- DataSet
- типизированный DataSet
- Пользовательский объект

Компоненты бизнес сущностей

- Не являются обязательной части любого приложения
- Являются локальным кешем снимка данных
- Добавляют к данным модели поведения с данными
- Предоставляют программный доступ к данным через свойства и методы сущности
- Могут содержать данные из разных источников
- Должны использовать компоненты логики доступа данных

Содержание

- Категории компонент на уровнях
  - Представления
  - Бизнес логики
  - **Обращения к данным**
- Политики
  - Безопасности
  - Управления и мониторинга
  - Взаимодействия

Уровень доступа к данным

- Вопросы для решения
  - Используемые хранилища данных
  - Проектирование компонент доступа
  - Проектирование вспомогательных компонент
  - Формат передачи данных между компонентами

Хранилища данных

- Реляционные Базы Данных
- Базы данных сообщений
- Файловая система
- Системные Каталоги

Компоненты логики доступа к данным

Функциональность компонент логики доступа к данным

- Реализация методов доступа для бизнес сущностей
  - Create, Read, Update и Delete (CRUD)
- Могут использовать вспомогательные компоненты доступа к данным для централизованного управления:
  - API доступа к данным
  - Пула соединений
  - Параметров кеширования
- **Data Access Management Application Block на MSDN**

Содержание

- Категории компонент на уровнях
  - Представления
  - Бизнес логики
  - Обращения к данным
- **Политики**
  - Безопасности
  - Управления и мониторинга
  - Взаимодействия

Проектирование политики безопасности

Безопасность

- **Аутентификация**
- **Авторизация**
- **Безопасность каналов**
- **Аудит**
- **Управление профилем**

Проектирование политики операционного управления

Управление исключениями

- Перехват и генерация исключений
- Разработка классов исключений
  - Бизнес исключения в отличие от системных исключений
- Передача информации об исключениях через уровни приложения
- Публикация информации об исключениях

- Exception Management Application Block for .NET на MSDN

Мониторинг

- Корректное исполнение:
  - Все компоненты работаю?
  - Нет ли блокировок?
  - Нет ли аварийных завершений?
- Соответствие SLA :
  - Удовлетворяет ли система ожидаемым параметрам?
  - Нет проблем с внешними взаимодействиями?
  - Удовлетворяет производительность?
- Управление масштабированием:
  - Удовлетворяют ли физические характеристики компьютеров возложенным на них задачам?
  - Предсказуема ли производительность отдельных ресурсов?
- Повышение бизнес производительности

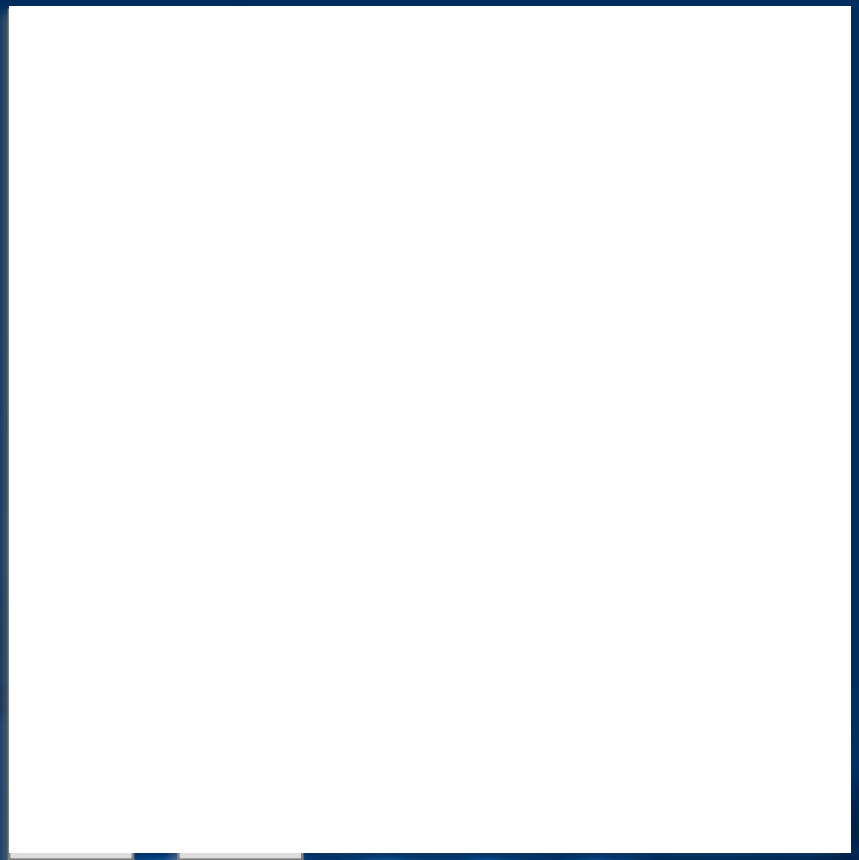
Конфигурация

- Последовательный подход
- Шифруйте критически важные настройки
  - Используйте DPAPI
- Типы хранилищ:
  - XML конфигурационный файлы
  - SQL Server
  - Active Directory
  - Метаданные Enterprise Services
  - Другие хранилища
    - Windows Registry
    - Windows Local Security Authority (LSA) store
    - Собственные реализации

Метаданные

- Информация о самом приложении
- Более гибкая реакция на меняющиеся условия
- Могут использоваться
  - На этапе разработки
  - Во время выполнения

Проектирование политики коммуникаций



Физическое развертывание и операционные требования

- Физическое развертывание компонент
  - Физическое окружение
  - Распределение компонент по сборкам
  - Распространение и установка сборок по уровням
- Операционные требования
  - Управляемость
  - Доступность
  - Масштабируемость
  - Производительность

Заключение

- Определите распределение компонент приложения по уровням
- Выберите технологии взаимодействия компонент с учетом поддержки транзакций
- Спроектируйте политики безопасности и операционного управления
- Спланируйте развертывание приложения

Дополнительные ресурсы

- Российский веб-сервер компании Microsoft
  http://www.microsoft.com/rus
- Платформа Microsoft .NET
  http://www.gotdotnet.ru/
- .NET Architecture Center
  http://msdn.microsoft.com/architecture
- Microsoft TechNet
  http://www.microsoft.com/rus/technet/
- Партнёры Microsoft
  http://www.microsoft.com/rus/licensing/where/certified_partners/
- Обучение и сертификация
  http://www.microsoft.com/rus/ctec/

Microsoft Платформа 2003

Вопросы?

Определяя **будущее**

Microsoft Платформа 2003

Where do you want to go today?®

Определяя будущее

Microsoft®