

# Объектно-ориентированный Анализ и Дизайн

## Часть 3

Шаблоны проектирования  
Архитектурные шаблоны  
Rational Unified Process  
Документирование бизнес-процессов на UML  
Управление конфигурацией ИТ проекта

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 1

## 7. Шаблоны проектирования

✓ Шаблон проектирования – стандартное решение стандартной проблемы.

- Название
- Проблема – описание ситуаций, в которых применяется шаблон
- Решение
- Последствия

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 2

## Виды шаблонов

E. Gamma:

- Конструкционные шаблоны – абстрагируют создание объектов
  - Abstract Factory, Factory Method, Singleton
- Структурные шаблоны – решают проблемы композиции
  - Adapter, Bridge, Decorator, Proxy
- Поведенческие шаблоны – алгоритмы и распределение ответственности между объектами
  - Command, Iterator, Observer, State, Strategy

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 3

## Abstract Server

Проблема:  
- Button нельзя использовать в контексте не использующем Light

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 4

## Решение

Решение: разорвать зависимость между Button and Light путем вставки интерфейса

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 5

## Abstract Server Pattern

Плюсы:  
- устраняет зависимость клиента от сервера  
- сервера могут изменяться не влияя на клиентов  
- устраняет нарушение DIP

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 6

## Adapter

Проблема:  
- Light уже существует и не может наследовать Device  
- Device может уже иметь методы activate/deactivate

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 7

## Решение

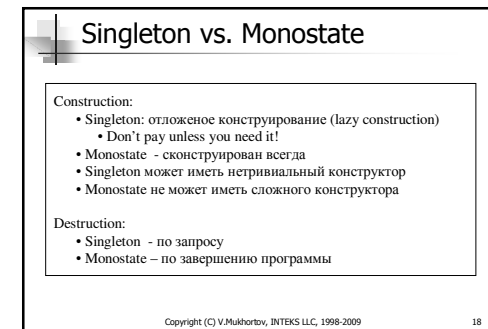
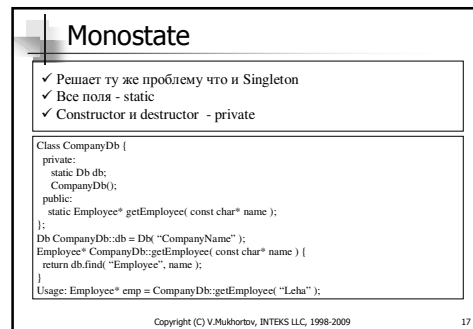
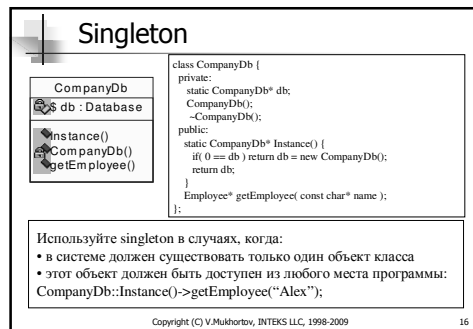
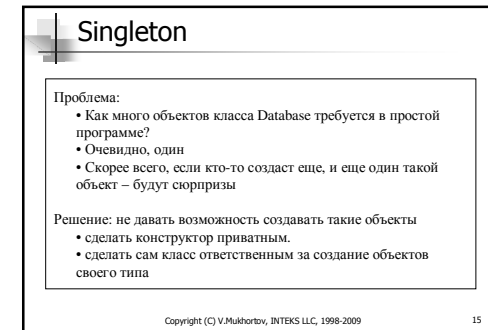
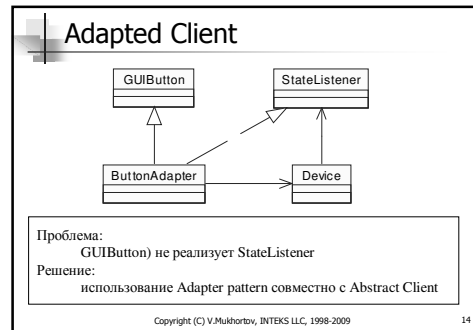
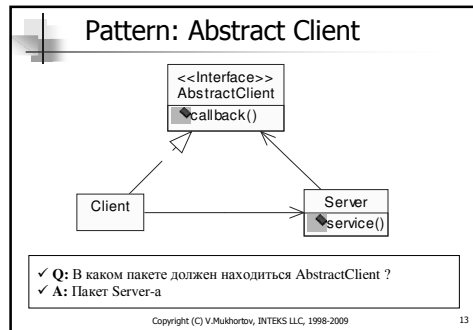
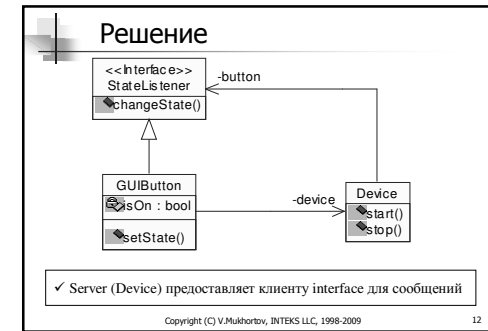
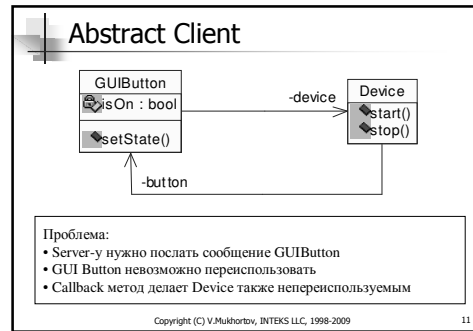
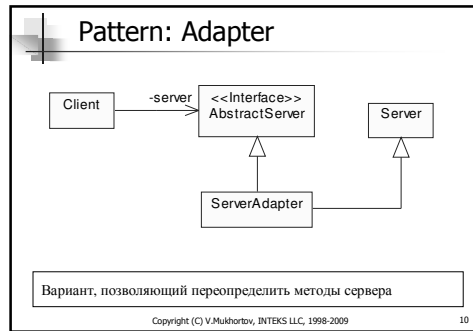
- Адаптер конвертирует один интерфейс в другой

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 8

## Pattern: Adapter

Также известен как: Wrapper  
Плюсы:  
- разрыв зависимости между client and server когда сервер уже существует  
- позволяет подменять сервера

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 9



### Strategy

Проблема:

- различные сотрудники оплачиваются по-разному
- как добавить hourly manager ?

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 19

### Решение

✓ Алгоритм начисления зарплаты вынесен в PaymentPolicy

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 20

### Strategy Pattern

Также известен как: Policy

Плюсы:

- можно добавлять различные стратегии
- Context закрыт от модификации стратегий (выполняется OCP)

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 21

### Bridge

Проблема:

- Код зависит от платформы
- Для поддержки новой платформы надо воспроизвести всю иерархию

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 22

### Solution

✓ Все операции подклассов Window реализованы в терминах операций из WindowImp

✓ Window и его подклассы платформо-независимы

Q: Как сделать код независимым от подклассов WindowImp?

A: использовать фабрику

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 23

### Bridge Pattern

Также известен как: Handle/Body

Плюсы:

- Устраняет нарушение DIP & OCP

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 24

### Abstract Factory

Проблема:

- создание объектов есть зависимость от производных типов
- но все прочие действия делаются через интерфейс
- что мы выиграли от использования интерфейса?

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 25

### Решение

Создать класс, отвечающий за создание объектов

Избегать создания объектов нельзя, но можно локализовать

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 26

### Factory Pattern

Изолирует конкретные типы данных, упрощает внесение изменений

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 27

### Proxy

```

classDiagram
    class Client
    class RealEmployee {
        db_dependent_code()
    }
    Client --> RealEmployee
    
```

Проблема:  
Нам нужно хранить объекты в базе данных, но нет желания зависеть от схемы базы

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 28

### Решение

```

classDiagram
    class Employee {
        <<interface>>
    }
    class CompanyDatabase
    class RealEmployee
    class EmployeeProxy {
        db_specific_code()
    }
    Employee <|-- RealEmployee
    Employee <|-- EmployeeProxy
    EmployeeProxy ..> CompanyDatabase : <<creates>>
    
```

Решение:  
- использовать суррогат который знает о схеме базы  
- клиенты могут считать что работают с RealEmployee

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 29

### Proxy Pattern

```

classDiagram
    class Client
    class Subject {
        request()
    }
    class RealSubject {
        request()
    }
    class Proxy {
        request()
    }
    Client --> Subject
    Subject <|-- RealSubject
    Subject <|-- Proxy
    Proxy ..> RealSubject
    
```

Также известен как: Surrogate  
Применность:  
- remote proxy (удаленный доступ к объекту)  
- virtual proxy (создание реальных объектов «на лету»)

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 30

### Stairway to Heaven

```

classDiagram
    class Type1
    class Type2
    class Type3
    class PersistentObject {
        read()
        save()
    }
    Type1 <|-- Type2
    Type2 <|-- Type3
    
```

Проблема:  
- Нужно сделать иерархию персистентной, но нет желания зависеть от производителя OR-маппера или OODB

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 31

### Решение

```

classDiagram
    class Type1
    class Type2
    class Type3
    class PersistentType1
    class PersistentType2
    class PersistentType3
    class PersistentObject {
        read()
        save()
    }
    Type1 <|-- PersistentType1
    Type2 <|-- PersistentType2
    Type3 <|-- PersistentType3
    PersistentType1 <|-- PersistentType2
    PersistentType2 <|-- PersistentType3
    PersistentType1 ..> PersistentObject
    PersistentType2 ..> PersistentObject
    PersistentType3 ..> PersistentObject
    
```

Применить Adapter к каждому уровню иерархии  
❖ Требуется виртуальное множественное наследование. Как его избежать?

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 32

### 8. Архитектурный шаблон

- хорошо зарекомендовавшее себя решение определенной проблемы в определенном контексте
- Прежде чем изобретать новый «велосипед» – стоит изучить, как устроены и насколько хорошо работают отдельные узлы уже существующих «велосипедов»

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 33

### Архитектурные шаблоны

- Client/server : Клиент/Сервер
- N-tier : Многоуровневая архитектура
- Peer-to-peer (P2P) : Одноранговая сеть
- Pipes and filters : Каналы и фильтры
- ACL security : Списки контроля доступа
- MVC (Model-View-Controller) : Модель-Представление-Управление

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 34

### Архитектура клиент/сервер

```

graph LR
    Client[Клиент] -- "+" --> Server[Сервер]
    Server -- "1" --> Client
    
```

**Характерные черты:**

- Единый сервер
- Отсутствие прямого взаимодействия между клиентами

**Преимущества:**

- Централизованное обслуживание

**Недостатки:**

- Сервер может оказаться «узким местом» системы

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 35

### Многоуровневая архитектура

```

graph LR
    Client[Клиент] --> L1[Уровень представления]
    L1 --> L2[Уровень логики и управления объектами]
    L2 --> L3[Уровень бизнес-логики]
    L3 --> L4[Уровень данных]
    L4 --> L5[Уровень хранения]
    
```

**Характерные черты:**

- каждый слой предоставляет сервисы следующему и использует сервисы предыдущего
- взаимодействуют только соседние слои
- каждый слой реализует четко определенную часть функциональности

**Преимущества:**

- возможность независимой разработки
- сужение набора необходимых для создания каждого слоя знаний

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 36

### Одноранговая сеть

**Характерные черты:**

- отсутствие центрального сервера
- равные права участников

**Преимущества:**

- Надежность

**Недостатки:**

- Сложность распространения изменений (updates)

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 37

### Одноранговая сеть

Применение

- Collaborative Computing
- Instant Messaging (совместно с Client-Server)
- P2P networks (сервера UseNet news, SMTP)
- Simple file sharing (MS Windows network w/o domain)

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 38

### Каналы и фильтры

Применение: системы обработки потоков данных

Ситуации:

- система создается разными разработчиками (возможно, часть системы уже существует)
- задача естественно разбивается на независимые этапы обработки
- задачи могут меняться, требуя декомпозиции шагов обработки

**Характерные черты:**

- фильтры – модули, получающие на вход поток(и) данных и выдающие поток(и) данных
- Простота конфигурирования системы фильтров (соединение их с помощью каналов)
- фильтры работают параллельно

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 39

### Списки контроля доступа (ACL или ACE)

- Subject – соответствует сессии пользователя
- Principal – некоторый идентификатор лица (пользователя) которому можно поставить в соответствие права доступа (например LDAP), может быть не один, если у пользователя несколько ролей в системе
- Resource – ресурс, доступ к которому ограничен
- ACL – список идентификаторов лиц, которым разрешен доступ к ресурсу

При каждом обращении к ресурсу производится проверка, есть ли у обращающегося subject'a principal, которому разрешено данное действие

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 40

### Model-View-Controller

- Разделяет представление, данные и логику, и обработку событий UI
- Model: информация, бизнес-правила.
- View: элементы UI, получающие данные из модели
- Controller: обработка событий UI, часто через callback, передача изменений в модель

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 41

### MVC

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 42

### 9. Rational Unified Process

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 43

### Концепция (Vision)

➤ Vision – представляет собой общее описание проекта и является базисом для уточнения требований к системе

➤ Содержит:

- Цели проекта
- Stakeholders & Users (*описание инициаторов проекта и конечных пользователей*)
- Перспективы и возможности системы
- Особенности
- Ограничения

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 44

### RUP: Business modeling

- Задачи:
  - Идентификация бизнес-процессов (business use-cases)
  - Идентификация бизнес-акторов и сущностей (business entity)
  - Улучшение (refine) бизнес-процессов
- Модели:
  - business use-case model
  - business object model

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 45

### RUP: Требования (Requirements)

**Задачи:**

- сбор и анализ требований к системе
- классификация use-cases
- оценки затрат и рисков

**Модели:**

- Use-case model

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 46

### SRS (Спецификация требований)

- SRS (Software Requirements Specification) - полностью определяет требования к системе, зависит от Vision
- Содержит:
  - Функциональные требования (что должна делать система, роли пользователей, фактически, описание use-cases)
  - Нефункциональные требования (производительность, ограничения по используемым технологиям и т.д.)

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 47

### RUP: Анализ и проектирование

**Задачи:**

- Трансформировать требования собранные на предыдущем этапе в дизайн системы
- Проработать архитектуру системы
- Адаптировать дизайн к среде исполнения

**Модели:**

- Analysis model
- Design model

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 48

### SAD (Архитектурный документ)

➢ SAD (Software Architecture Document) – содержит полное описание архитектуры системы

➢ Содержит:

- Use-case view
- Logical View (архитектурно важные части Design model)
- Process View
- Deployment View (диаграмма размещения системы)
- Implementation View
- Open issues (список известных проблем, например, с производительностью или масштабируемостью, возможные пути решения)
- Quality issues (любые проблемы в качестве)

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 49

### RUP: Реализация (Implementation)

**Задачи:**

- Структурирование системы
- Реализация компонент системы

**Артефакты:**

- SAD – приведение в соответствие с реализацией
- Implementation model – модель реализации системы в терминах компонент и процессов

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 50

### RUP: Ключевые роли

- Project Manager
- Analyst
- Test Designer
- System Architect
- Designer
- Implementer
- Tester

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 51

### Роль RUP

- Высокоадаптивный процесс: минимум обязательных практик.
- Систематизация знаний в области процессов разработки ПО

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 52

### Бизнес-анализ

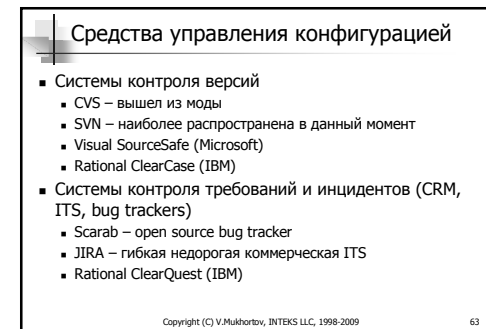
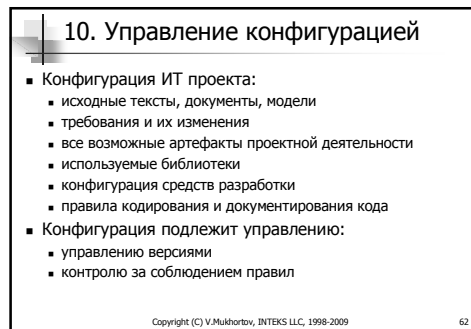
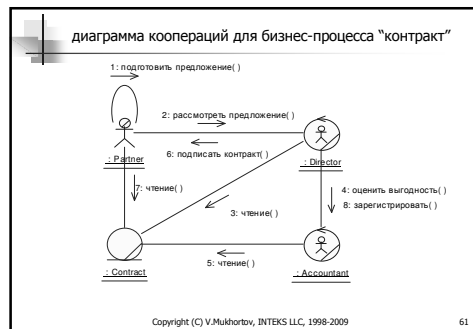
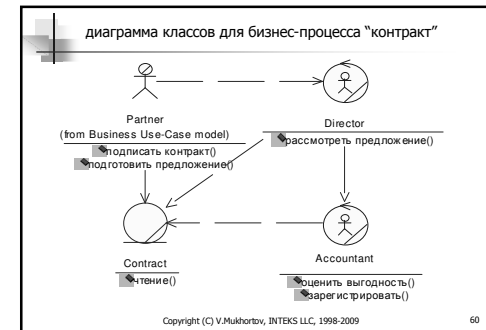
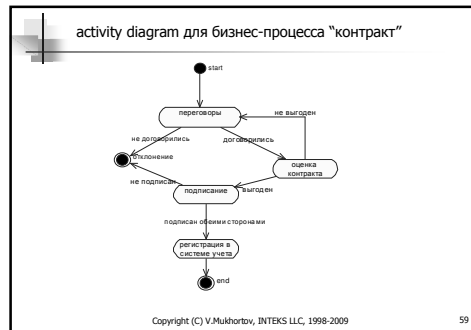
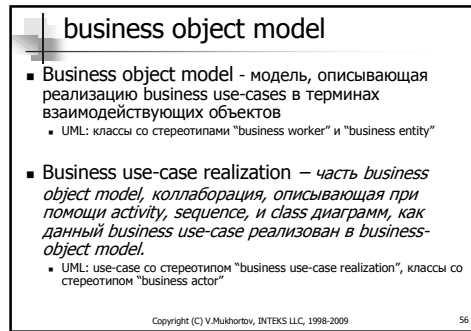
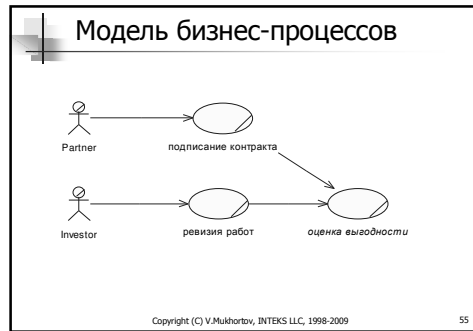
- **Задачи:**
  - Идентификация бизнес-процессов (use-cases)
  - Идентификация бизнес-актеров и сущностей (entity)
  - Улучшение (refine) бизнес-процессов
- **Нужен для того, чтобы:**
  - Лучше понять предметную область
  - Понять, как изменяются процессы в бизнесе в случае внедрения системы, описываемой в use-case model
- **UML модели:**
  - business use-case model
  - business object model

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 53

### business use-case model

- Модель, описывающая бизнес процессы в терминах *business-actors* и *business use-cases*
- *Business actor* – некто или нечто **вовне** бизнеса, взаимодействующее с ним
  - UML: класс со стереотипом <<business actor>>
- Business use-case – бизнес-процесс, представляющий ценность для *business actor*
  - UML: use-case со стереотипом <<business use-case>>

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009 54



### Терминология: контроль версий

- **Репозиторий** – место хранения артефактов проекта в системе управления версиями
- **Релиз** – любая версия ПО, вышедшая за пределы проектной команды; подлежит обязательному присвоению номера версии и тега
- **Бранч** – версия ПО, начавшая изменяться параллельно с текущей версией
- **Тег** – моментальный снимок состояния репозитория

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009

64

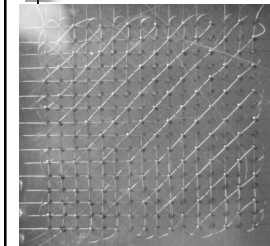
### Терминология

- **Требование (requirement)** – любое функциональное или нефункциональное требование
- **Изменение (change)** – изменение исходного требования
- **Уточнение (adjustment)** – уточнение требования, не влияющее на оценку проекта
- **Ошибка (bug)** – подтвержденная ошибка
- **Ticket, Issue** – проблема, нерешенный вопрос, может стать ошибкой, может изменением

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009

65

### Термин Bug (жук)



- Есть мнение, что термин возник из-за настоящих жуков, иногда попадавших на проволочки памяти на ферритовых кольцах, и мешавших считывать информацию

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009

66

### Несколько советов

- Не путайте бранч с тегом, у них разные задачи
- Commit в репозиторий делается ежедневно. No commit – по excuse.
- Автоматический бекап репозитория:
  - Ежедневно.
  - Обязательно на другой физический диск;
  - лучше всего – на диск, расположенный на другом сервере;
  - еще лучше – на диск на сервере, расположенном в другом здании.
  - Регулярное копирование бекапа на долговечный носитель (CD, DVD, streamer).

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009

67

### Несколько советов

- Используйте единый метод сборки проекта всеми участниками, желательно из командной строки (build.xml, makefile), т.к это позволит, при необходимости, легко внедрить регулярную автоматическую сборку
- Не допускайте использования разных сред разработки в одном проекте (то, что программисту «нравится» - пусть использует у себя дома, на работе он должен использовать то, что записано в описании проектной среды)

Copyright (C) V.Mukhortov, INTEKS LLC, 1998-2009

68