Использование компонентных технологий при разработке тестов

В. Кулямин

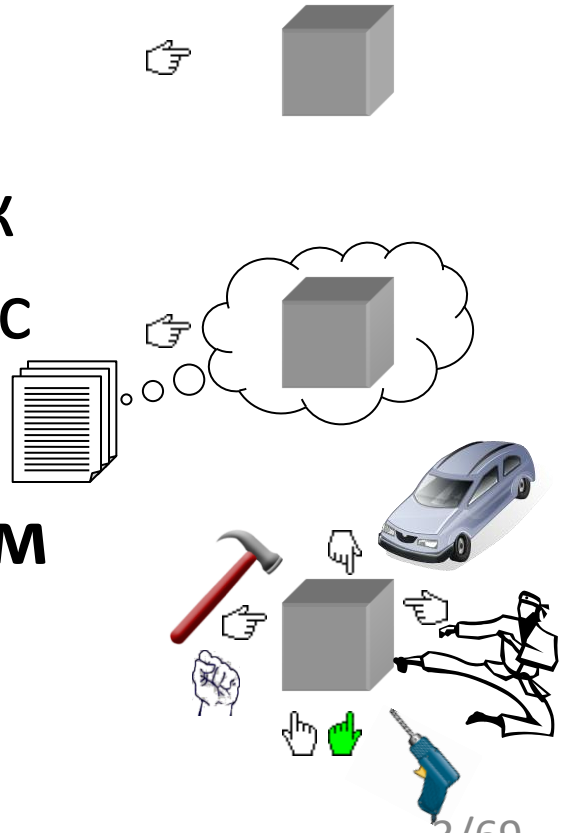
Институт системного программирования РАН

Базовые понятия

- **Тестирование** – проверка корректности (соответствия требованиям) поведения системы на основе наблюдения за ее реальной работой в конечном наборе заранее определенных ситуаций
- **Тестируемая система** (System under Test, SUT)
- **Тестовый вариант** – программа или процедура, нацеленная на создание специфической ситуации в работе тестируемой системы и проверку корректности ее поведения в этой ситуации
- **Тестовый набор** – группа тестовых вариантов, предназначенная для оценки ряда аспектов качества тестируемой системы или ее части

Как устроено тестирование

- **Воздействуем** на тестируемую систему
- Наблюдаем за **ее реакцией**
- Проверяем, такова ли она, как **должна быть** (в соответствии с требованиями)
- Повторяем, пока **не исчерпаем все существенно различные ситуации**



Задачи тестирования

- Найти ошибки
- Оценить качество системы по разным аспектам
- Контролировать развитие системы

Задачи разработки тестов I

- Определить важные для оценки качества системы характеристики ситуаций в ее работе
- Определить критерии корректности работы системы
- Определить критерии достаточности набора ситуаций для оценки нужных характеристик (критерии полноты)
- Определить структуру и состав тестовых отчетов
- Определить структуру набора тестов, удобную для его использования и дальнейшего развития
- …

Задачи разработки тестов II

- …
- Собственно, сделать тесты с учетом принятых решений
  - Придумать и реализовать набор ситуаций, удовлетворяющий критериям полноты
  - В каждой ситуации описать соответствующую процедуру проверки корректности поведения системы
  - Обеспечить построение нужных отчетов
  - Организовать тесты в заданную структуру

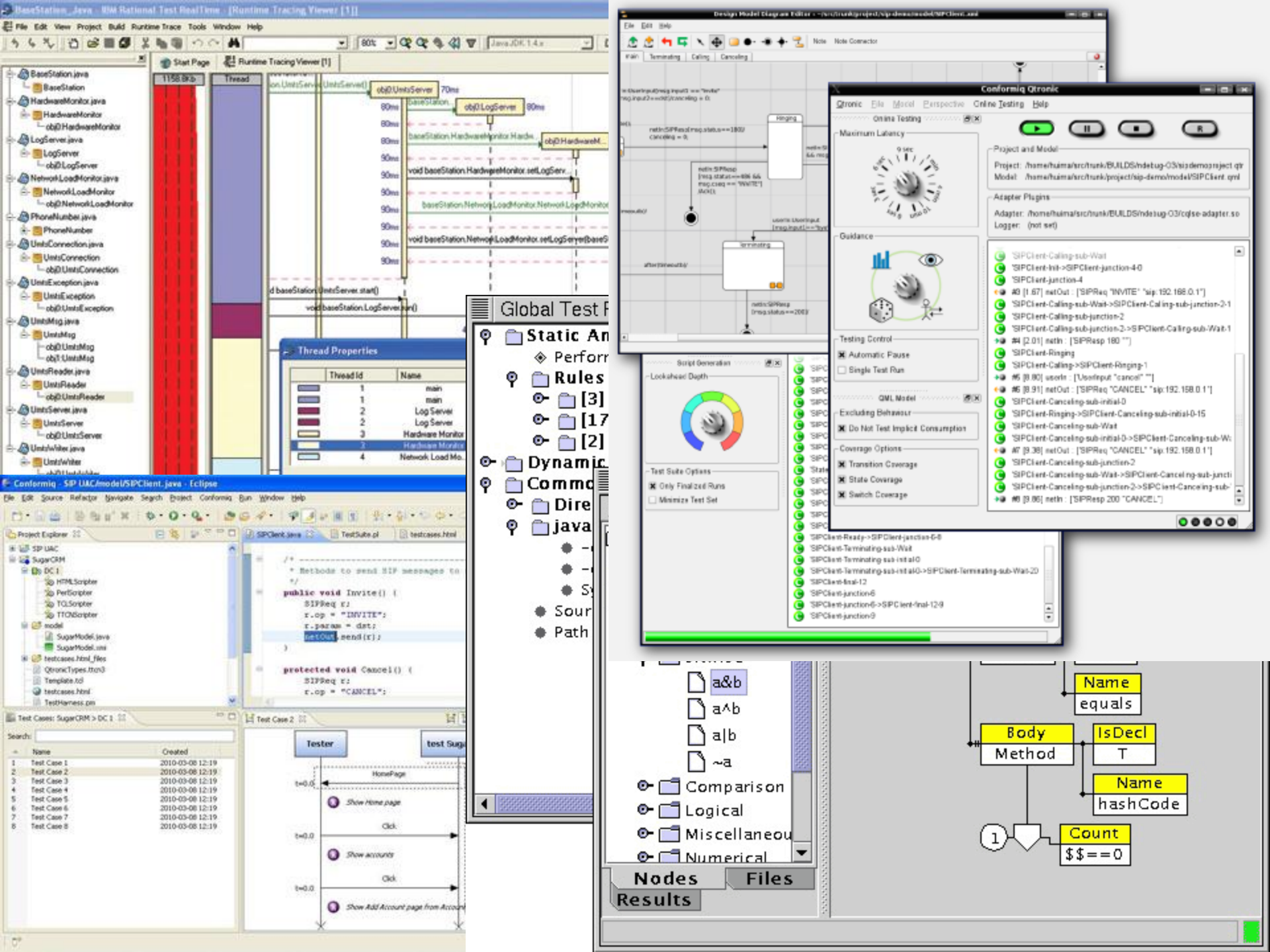
Компонентные технологии

Компоненты

- атомарные элементы сборки системы
- программные модули –
  обладают четко определенным интерфейсом и взаимодействуют только через него
- заменяемы без перекомпиляции и пересборки
  - имеют бинарное представление
- работают в рамках некоторой инфраструктуры
  - компонентной среды

«Обычные» инструменты тестирования

- Часто используют специфические языки и нотации
- С трудом расширяемы
  - Обычно интерфейс для расширений очень узок или отсутствует
  - Поддерживается фиксированный набор техник и пополнить его невозможно или очень трудно
- С трудом включаются в «неподготовленный» процесс разработки
  - Работают изолированно или только с теми инструментами, для которых разработчики предусмотрели некоторую интеграцию


Thread Properties
Global Test
Conformiq Qtronic
Nodes
Files
Results
Comparison
Logical
Miscellaneou
Numerical
Name
equals
Body
Method
IsDecl
T
Name
hashCode
Count
$$==0

Компоненты и разработка тестов

- Компонентная организация делает тестовый набор удобнее
  - при эксплуатации
    (выбор и выполнение нужного подмножества тестов)
  - при развитии
    (модификация тестов, независимая или в связи с развитием тестируемой системы)
- Она также делает более простой интеграцию средств тестирования с другими инструментами

Модульное тестирование

- **Модульное тестирование** (Unit Testing) – проверка работы некоторого куска кода системы в изоляции от всего остального
- **Unit** – минимальный компилируемый кусок кода
  - Процедура/функция – класс – компонент
- Критерии полноты – покрытие структуры кода (инструкций, ветвей, …)

Первые инструменты xUnit

- SUnit [Kent Beck, 1994]
  - автоматизация выполнения модульных тестов на Smalltalk
  - http://sunit.sourceforge.net/

- JUnit [Kent Beck, Erich Gamma, 1998]
  - то же, для Java
  - http://junit.org/
  - Test Driven Development

Основные решения xUnit

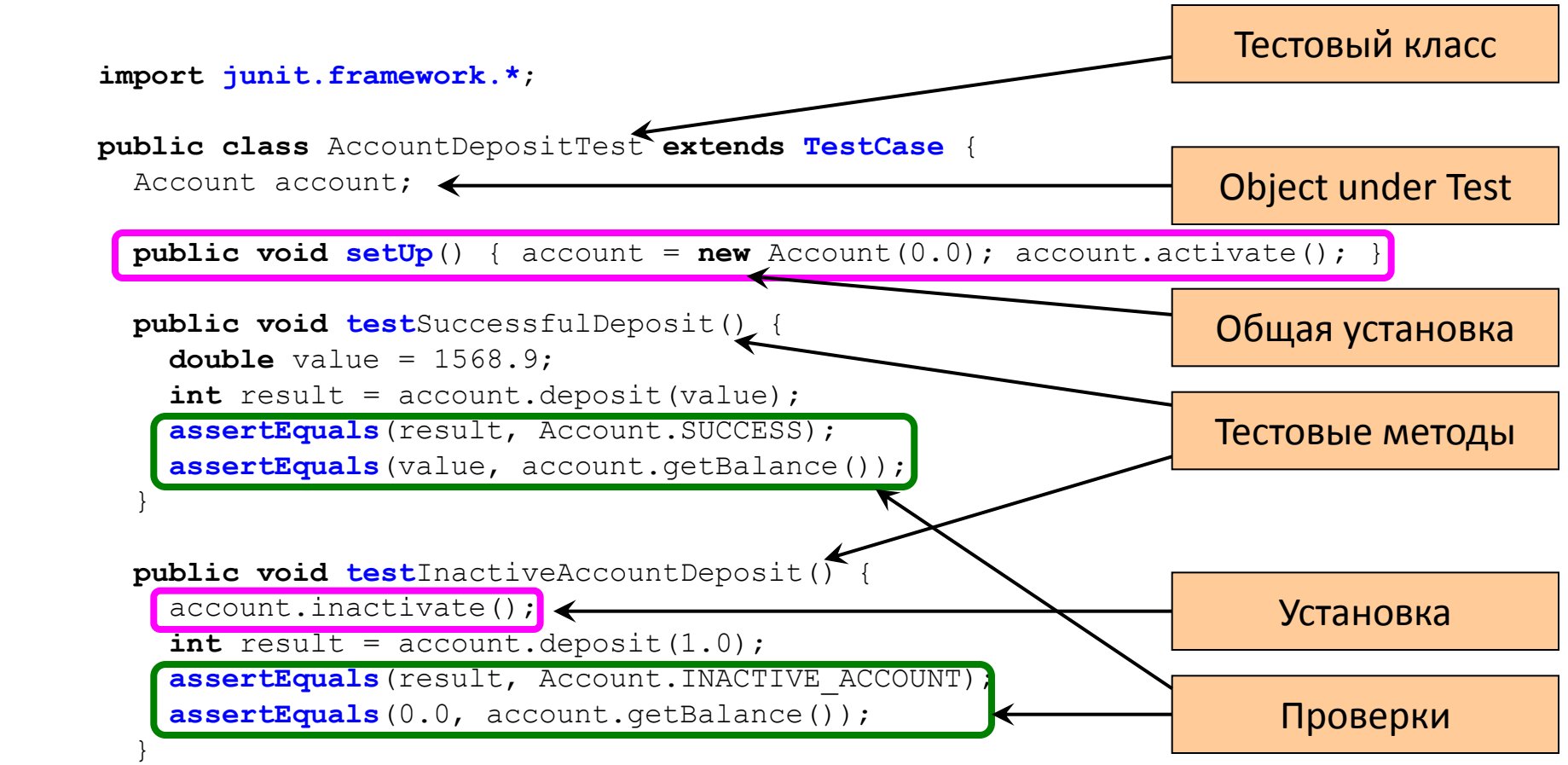
- Структура тестового варианта
  - Инициализация, установка (setup)
  - Выполнение (exercise)
  - Проверка (verify)
  - Финализация, снос (teardown)
- Иерархия тестов
  - Тестовые варианты – методы с именами test***
  - Тестовые классы (включают методы, наследуют TestCase)
  - Тестовые наборы (включают другие наборы и классы)
- Библиотека методов assert*** для оценки корректности
- Общие установка и снос в виде методов setUp() и tearDown()

Пример тестируемого класса

```
public class Account {
  public Account(double initBalance) { ... }

  public void activate() { ... }
  public void inactivate() { ... }
  public double getBalance() { ... }
  public int deposit(double sum) { ... }
  public int withdraw(double sum) { ... }

  public final static int SUCCESS = 0;
  public final static int INACTIVE_ACCOUNT = 1;
  public final static int INVALID_OPERATION = 2;
}
```

Пример теста в JUnit

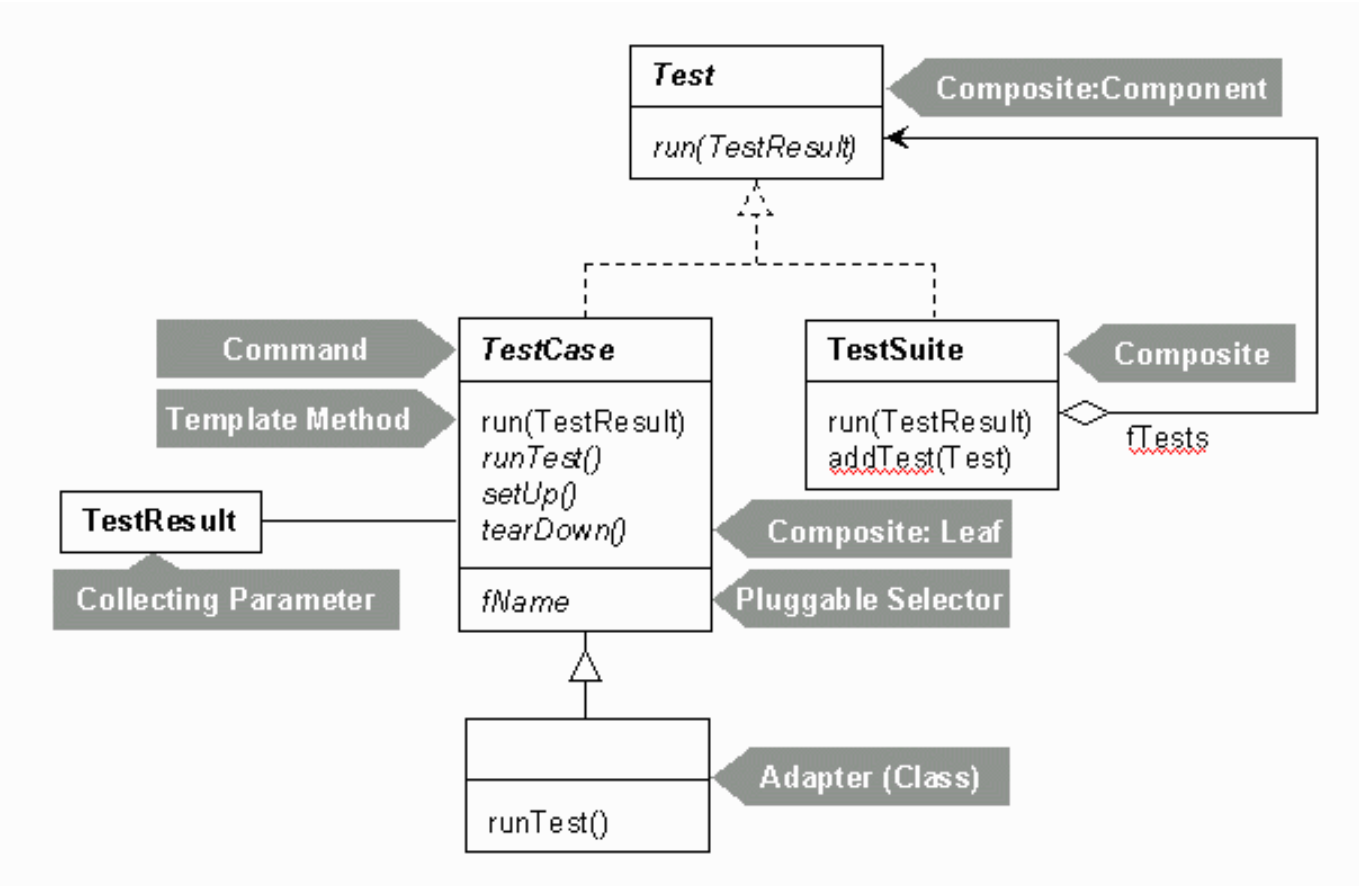
```
import junit.framework.*;

public class AccountDepositTest extends TestCase {
  Account account;

  public void setUp() { account = new Account(0.0); account.activate(); }

  public void testSuccessfulDeposit() {
    double value = 1568.9;
    int result = account.deposit(value);
    assertEquals(result, Account.SUCCESS);
    assertEquals(value, account.getBalance());
  }

  public void testInactiveAccountDeposit() {
    account.inactivate();
    int result = account.deposit(1.0);
    assertEquals(result, Account.INACTIVE_ACCOUNT);
    assertEquals(0.0, account.getBalance());
  }
}
```



Пример тестового набора в JUnit

```
import junit.framework.*;

public class AllTests extends TestSuite
{
  public static Test suite()
  {
    TestSuite suite = new TestSuite("Account Tests");

    suite.addTestSuite (AccountDepositTest.class);
    suite.addTestSuite (InactiveAccountTest.class);
    suite.addTest      (AnotherSuite.suite());

    return suite;
  }
}
```

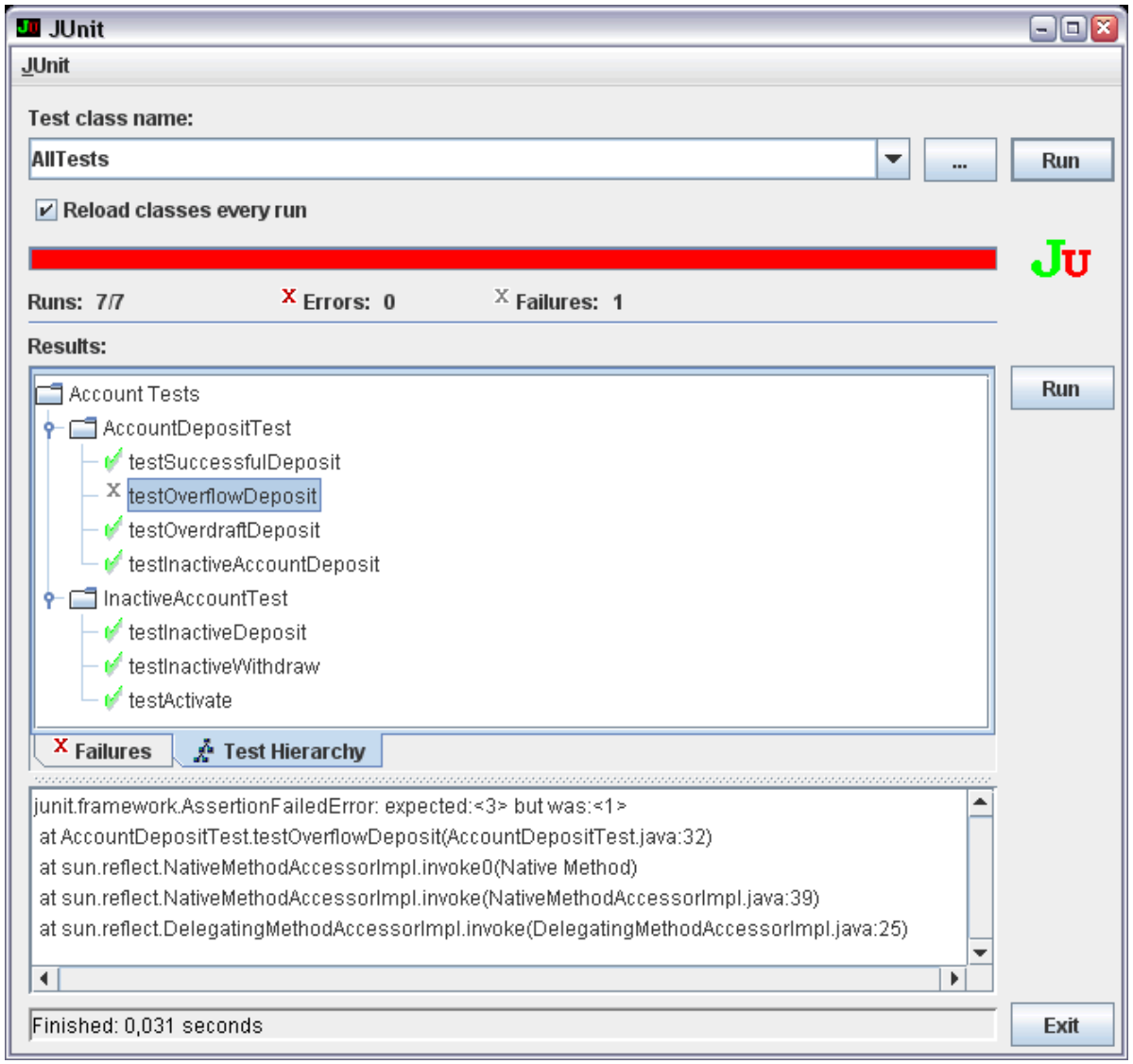
Образцы проектирования в JUnit



Алгоритм выполнения тестов

Элементы набора сложить в список tests
Для каждого элемента из tests
    Если это набор
        Добавить его элементы в конец tests
    Если это тестовый класс
        Выделить тестовые методы (имена начинаются на test)
        Для каждого выделенного метода
            Создать новый объект данного тестового класса
            Выполнить для этого объекта setUp()
            Выполнить для этого объекта данный метод
            Если выпало исключение, занести его в ошибки
            Выполнить tearDown()

Графическая оболочка JUnit



Методы assert

```
fail                ( String message )
assertTrue          ( String message, boolean expression )
assertNotNull ( String message, Object object )
assertEquals    ( String message, Object expected, Object actual )
assertSame      ( String message, Object expected, Object actual )
```

Достоинства и проблемы

- Очень простой инструмент
  - Легко расширяется дополнительными компонентами
  - Легко встраивается в другие инструменты и их цепочки
- Простая и удобная организация тестов
  - Легко пополнять тестовый набор
  - Достаточно легко выделить нужное подмножество
- Однако в больших наборах
  - Много тестов могут отличаться лишь незначительно
  - Для одного метода в разных ситуациях многие проверки могут дублироваться
  - Изменения в SUT (интерфейс и функции) влекут многообразные изменения в тестах («хрупкие» тесты)

Развитие инструментов xUnit

- Несколько сотен инструментов
  - AUnit, Check, FUnit, NUnit, PyUnit, Test::Unit, …
- Большой набор образцов
- Модули для отдельных задач
  - Для специфических областей
    - dbUnit (http://www.dbunit.org)
    - httpUnit (http://www.httpunit.org)
  - Наглядная и обобщенная запись проверок
  - Организация заглушек
- Новые возможности сред
  - TestNG (→ JUnit 4.x)
- Расширения для тестирования на основе моделей

Проверки на «естественном языке»

Behavior Driven Development [Dan North, 2003]

- Шаблон представления требований
  - Given <Некоторое состояние> …
  - When <Некоторое событие> …
  - Then <Что должно произойти и каково итоговое состояние> …
- Инструменты
  - JBehave, NSpecify, RSpec, …
- Библиотеки «наглядного» описания требований

```
specify { robot.moveForward() }.Must.Not.Throw();

Specify.That( max(x, y) ).
  Must.Be.Not.LessThan(x).And.Must.Be.Not.LessThan(y).
  And.Must.Either.Equal(x).Or.Equal(y);
```

Тестовые заглушки

- **Заглушки** (stubs, test doubles) заменяют компоненты, от которых зависит SUT
- Классификация в сообществе xUnit
  - Dummy Object – несущественное значение параметра вызываемого метода SUT
  - Остальные – объекты, к которым SUT обращается
    - Fake Object – простейшая замена реального объекта
    - Test Stub – использует возвращаемые результаты для управления тестом как неявные тестовые данные
      - Test Spy – еще и записывает обращения SUT для проверки в конце теста
      - Mock Object – еще и проверяет обращения SUT при их возникновении

Использование Mocks & Spies

- Среды
  - EasyMock, NMock, Mockito, pMock, CMock, …
- Создание

```
List mockedList = mock(List.class);
```

или

```
@Mock List mockedList;
```

- Определение возвращаемых результатов

```
when( mockedList.get(0) ).thenReturn( "first" );
when( mockedList.get(1) ).thenThrow( new RuntimeException() );
when( mockedList.get( anyInt() )).thenReturn( "element" );
```

- Проверка сделанных вызовов

```
inOrder.verify( mockedList ).add( "one" );
inOrder.verify( mockedList ).get( anyInt() );
verify( mockedList, atLeastOnce() ).add( "three times" );
verify( mockedList, times(2) ).add( "twice" );
verify( mockedList, never() ).add( "never" );
```

Инструмент TestNG

[Cedric Beust, 2004]

- Поддержка больших конфигурируемых тестовых наборов
- http://www.testng.org

Основные решения TestNG I

- Использование аннотаций Java 5
  - Для выделения тестовых методов и методов установки/сноса
  - Для некоторых проверок
    - Ожидаемые исключения (@Test(expectedExceptions="…"))
    - Таймауты (@Test(timeout="…"))
- Расширенная иерархия тестов
  - Тестовые методы (помечаются @Test)
  - Тестовые классы (помечаются @Test)
  - Тесты (определяются конфигурацией)
  - Тестовые наборы (определяются конфигурацией)

Основные решения TestNG II

- Конфигурирование тестов
  - Описание выполняемых тестов в командной строке или в XML
  - Группы и шаблоны выбора тестовых методов
  - Методы общей установки/сноса для всех элементов иерархии и групп
  - Зависимости между методами и группами
  - Настройка выполнения тестов
    - Пропуск метода/теста
    - Многократное выполнение метода
    - Параллельное выполнение методов или тестов
- Определение тестовых данных
  - Для отдельных методов
    - Тестовые методы имеют параметры
    - Задание наборов значений в аннотациях и конфигурации
    - Провайдеры данных
  - Фабрики тестовых объектов

Пример теста в TestNG

```
@Test
public class AccountTest {
  Account account;

  @BeforeClass(groups = {"active", "inactive"})
  public void makeDefaultAccount() { account = new Account(0.0); }

  @Test(groups = "inactive")
  @Parameters("normalSum")
  public void depositInactive(double value) {
    int result = account.deposit( value );
    Assert.assertEquals( result              , Account.INACTIVE_ACCOUNT );
    Assert.assertEquals( account.getBalance(), 0.0 );
  }

  @Test(groups = "active", dependsOnGroups = "inactive", alwaysRun = true)
  public void activateAccount() {
    account.activate();
    Assert.assertTrue( account.isActive() );
  }

  @Test(groups = "active", dependsOnMethods = "activateAccount")
  @Parameters("normalSum")
  public void successfulDeposit(double value) {
    double oldBalance = account.getBalance();
    int result = account.deposit(value);
    Assert.assertEquals(result              , Account.SUCCESS);
    Assert.assertEquals(account.getBalance(), oldBalance + value);
  }
}
```

Пример конфигурационного файла

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="All" verbose="2" parallel="false">

  <parameter name="normalSum" value="15.67" />

  <test name="All">
    <classes>       <class name="tests.AccountTest" />      </classes>
  </test>

  <test name="Active account">
    <groups><run>   <include name = "active"/>    </run>    </groups>

    <classes>       <class name="tests.AccountTest" />      </classes>
  </test>

  <test name="Inactive account">
    <groups><run>   <include name = "inactive"/>  </run>    </groups>

    <classes>       <class name="tests.AccountTest" />
      <methods>     <exclude name=".successful*.*" />       </methods>
    </classes>
  </test>
</suite>
```

Результаты работы TestNG

- Отчет о результатах
  - Выполненные успешно тесты
  - Выполненные с ошибками тесты
  - Пропущенные тесты (которые не удалось выполнить)
- Конфигурация набора, состоящего из выполненных с ошибками и пропущенных тестов

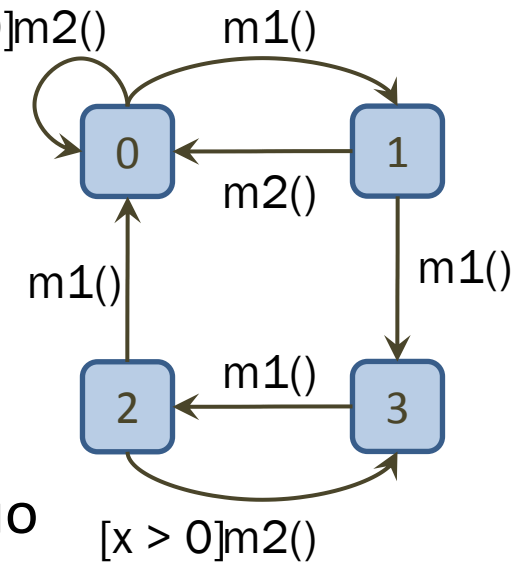
Тестирование на основе моделей

- ModelJUnit [M. Utting, 2004]
  - Waikato University
  - http://czt.sourceforge.net/modeljunit/

- NModel
  [J. Jacky, M. Veanes, C. Campbell, 2007]
  - University of Washington
  - Microsoft Research
  - http://nmodel.codeplex.com/

Основные идеи

- Тест строится как набор путей по конечному автомату, моделирующему поведение SUT
- Тестовый класс интерпретируется как описание расширенного автомата (EFSM)
  - Тестовые методы – возможные воздействия
  - Текущее состояние вычисляется специальным методом или представлено некоторыми полями
  - Действия могут иметь охранные условия (guardians), тоже представленные как методы



Пример модели теста в NModel

```
public class AccountTest {
  public Accout target = new Account(0.0);
  int balance = 0;
  bool active = false;

  [Action]
  void activate() {
    target.Activate();
    active = true;
  }

  bool DepositEnabled() { return active; }
  [Action]
  void Deposit() {
    int sum = 5;
    target.Deposit(sum);
    balance += sum;
  }
}
```

Основные идеи NModel I

- Пространство имен – автоматная модель
  - Состояние – набор значений статических полей классов модели и набор достижимых объектов
    - Правила отождествления состояний
  - Действия – методы, помеченные атрибутом [Action]
    - Много действий может соответствовать одному методу
    - Имеют охранные условия (с именем ***Enabled), может быть несколько
- Тесты
  - Предварительное построение (offline)
    - Строится полная модель – все состояния и переходы
    - Тесты – набор покрывающих ее путей
  - Динамическое построение (online)
    - Настраиваемые стратегии, в т.ч. нацеленные на покрытие
  - Ограничения пространства состояний
  - Тестовые данные
    - Действия могут иметь параметры
    - Провайдеры данных определяются атрибутом [Domain]
  - Адаптеры (steppers)

Основные идеи NModel II

- Анализ моделей
  - Инварианты
    - Достижение нарушающего инвариант состояния – ошибка модели
  - Условия стабильности состояния
    - Недостижимость стабильного состояния – нарушение живучести
- Композиция моделей
  - Компоненты – классы, помеченные [Feature]
  - Одноименные действия выполняются одновременно
  - Действия без партнера выполняются в любой момент
  - Использование
    - Модульность моделей
    - Проверка свойств моделей
    - Ограничение моделей и тестов
    - Выделение целей тестирования

Пример модели – сервер

```
namespace ClientServer {
  [Feature] public partial class Server {
    public static Socket serverSocket = Socket.None;
    public static Phase phase = Phase.Send;

    public static bool ServerSocketEnabled() { return (serverSocket == Socket.None); }
    [Action] public static void ServerSocket() { serverSocket = Socket.Created;  }

    public static bool ServerBindEnabled() { return (serverSocket == Socket.Created); }
    [Action] public static void ServerBind() { serverSocket = Socket.Bound; }

    public static bool ServerListenEnabled() { return (serverSocket == Socket.Bound); }
    [Action] public static void ServerListen() { serverSocket = Socket.Listening; }

    public static bool ServerAcceptEnabled() { return (serverSocket == Socket.Listening); }
    [Action] public static void ServerAccept() { serverSocket = Socket.Connected; }

    public static bool ServerReceiveEnabled()
    { return (serverSocket == Socket.Connected && phase == Phase.ServerReceive);  }
    [Action] public static void ServerReceive() { phase = Phase.Send; }
  }
```

Пример модели – клиент

```
[Feature] public partial class Client {
    public static Socket clientSocket = Socket.None;
    public static double clientBuffer = double.MaxValue;

    public static bool ClientSocketEnabled() { return (clientSocket == Socket.None); }
    [Action] public static void ClientSocket() { clientSocket = Socket.Created; }

    public static bool ClientConnectEnabled() { return (clientSocket == Socket.Created); }
    [Action] public static void ClientConnect() { clientSocket = Socket.Connecting; }

    public static bool ClientSendEnabled() { return (clientSocket == Socket.Connected); }
    [Action] public static void ClientSend() {  phase = Phase.ServerReceive; }

    public static bool ClientReceiveEnabled() { return (clientSocket == Socket.Connected); }
    [Action] public static double ClientReceive(double datum)
    { clientBuffer = datum; return datum; }

    public static bool ClientCloseEnabled()
    { return (clientSocket == Socket.Created || clientSocket == Socket.Connected); }
    [Action] public static void ClientClose() { clientSocket = Socket.Closed; }
  }
```

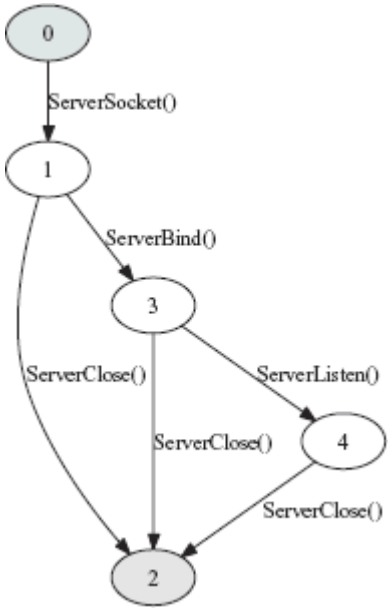
Пример – сервер для композиции

```
[Feature] public partial class Server {

    public static bool ClientConnectEnabled()
    { return (serverSocket == Socket.Listening); }

    public static bool ClientSendEnabled()
    { return (phase == Phase.Send); }
    [Action] public static void ClientSend()
    {  phase = Phase.ServerReceive; }

    public static bool ClientReceiveEnabled()
    { return (phase == Phase.ClientReceive); }
    [Action] public static void ClientReceive()
    { phase = Phase.Send; }
  }

  [Feature] class Values2 {
    readonly static Set<double> Values = new Set<double>(99.9, 100.0);

    [Action] static void ClientReceive([Domain("Values")] double datum) {}
  }
```

Пример – клиент для композиции

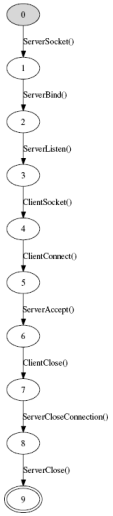
```
[Feature] public partial class Client {

    public static bool ServerAcceptEnabled()
    { return (clientSocket == Socket.Connecting); }
    [Action] public static void ServerAccept()
    { clientSocket = Socket.Connected; }
  }
}
```

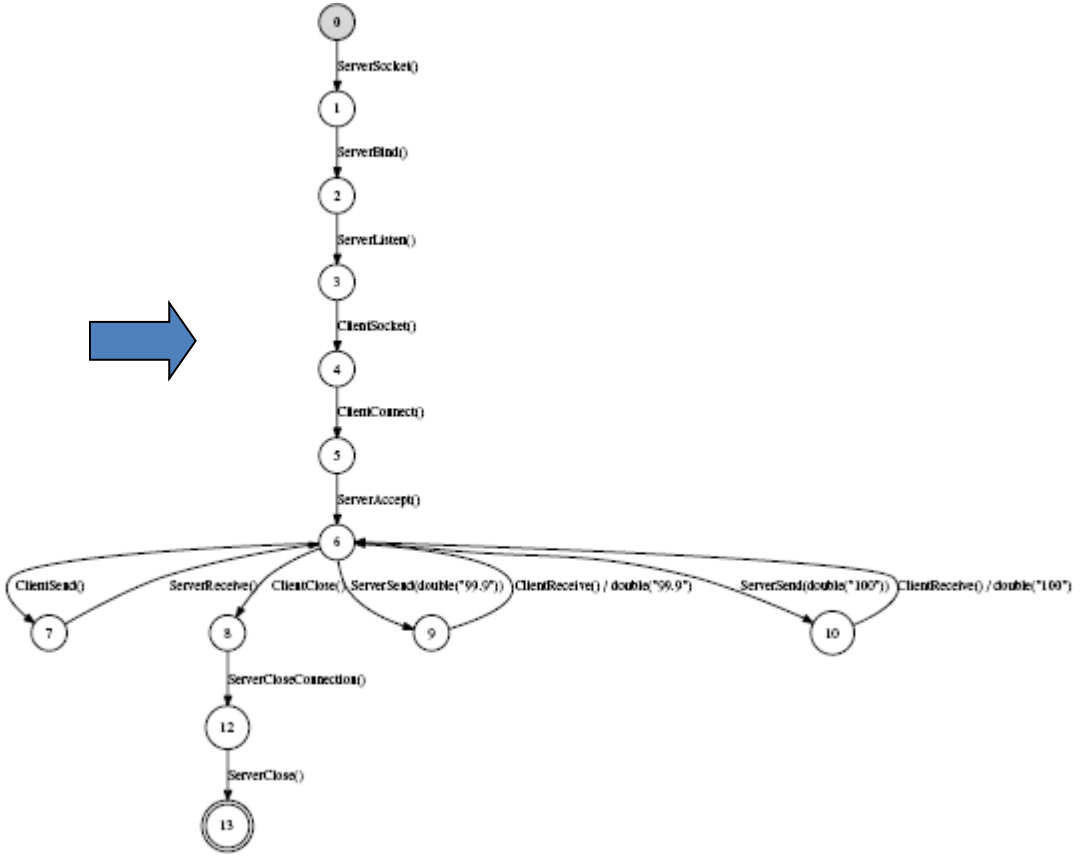
Композиция клиент-сервер



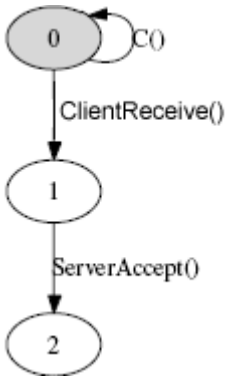
Сервер

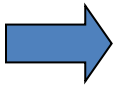
Указание целей тестирования

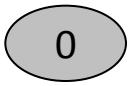




Проверка свойств моделей







Другие полезные элементы

- Программные контракты
  - CodeContracts [2009]
    http://research.microsoft.com/en-us/projects/contracts/
- Использование существующих модулей
  - httpUnit, dbUnit, etc.
  - Заглушки (mocks and spies)
  - Генераторы тестовых данных
- Оценка покрытия модели

Пример CodeContracts

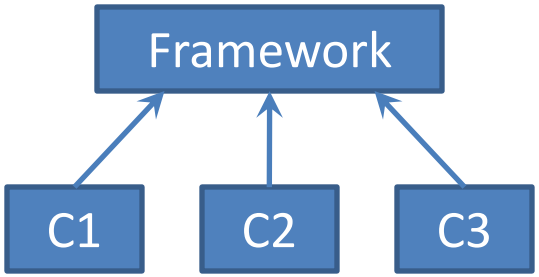
```
[ContractClassFor(typeof(Account))]
public class AccountContract {
  [Pure] double Balance { get; }

  int Deposit(double sum) {
    Contract.Requires
      ( Double.MaxValue - sum >= Balance );
    Contract.Ensures
      ( Balance - sum ==
        Contract.OldValue<double>(Balance) );
  }
}
```

Возможности интеграции

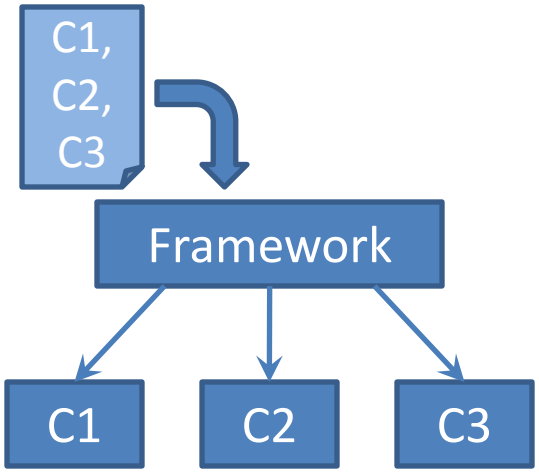
Можно ли объединять различные техники описания (EFSM, контракты, …) при разработке тестов не тратя на это больших усилий?

- Идеи
  - Использование аннотаций и библиотек, а не новых/расширенных языков
  - Внедрение зависимостей
  - Аспектная конфигурация

Внедрение зависимостей



Централизованная интеграция



Интеграция на основе внедрения зависимостей

Аспекты: основные понятия

- **Вставка** (advice) – код, который нужно вставить в определенных местах для реализации некоторой дополнительной функции
- **Точка внедрения** (join point) – место и время вставки дополнительного кода
- **Сечение** (pointcut) – предикат, описывающий множество точек внедрения для реализации определенной функции
- **Аспект** (aspect) – комбинация вставки и сечения, реализующая некоторую дополнительную функцию

Аспектная привязка контрактов

- Метод с контрактом
  - Метод C.M()
  - Предусловие метода preM()
  - Постусловие метода postM()

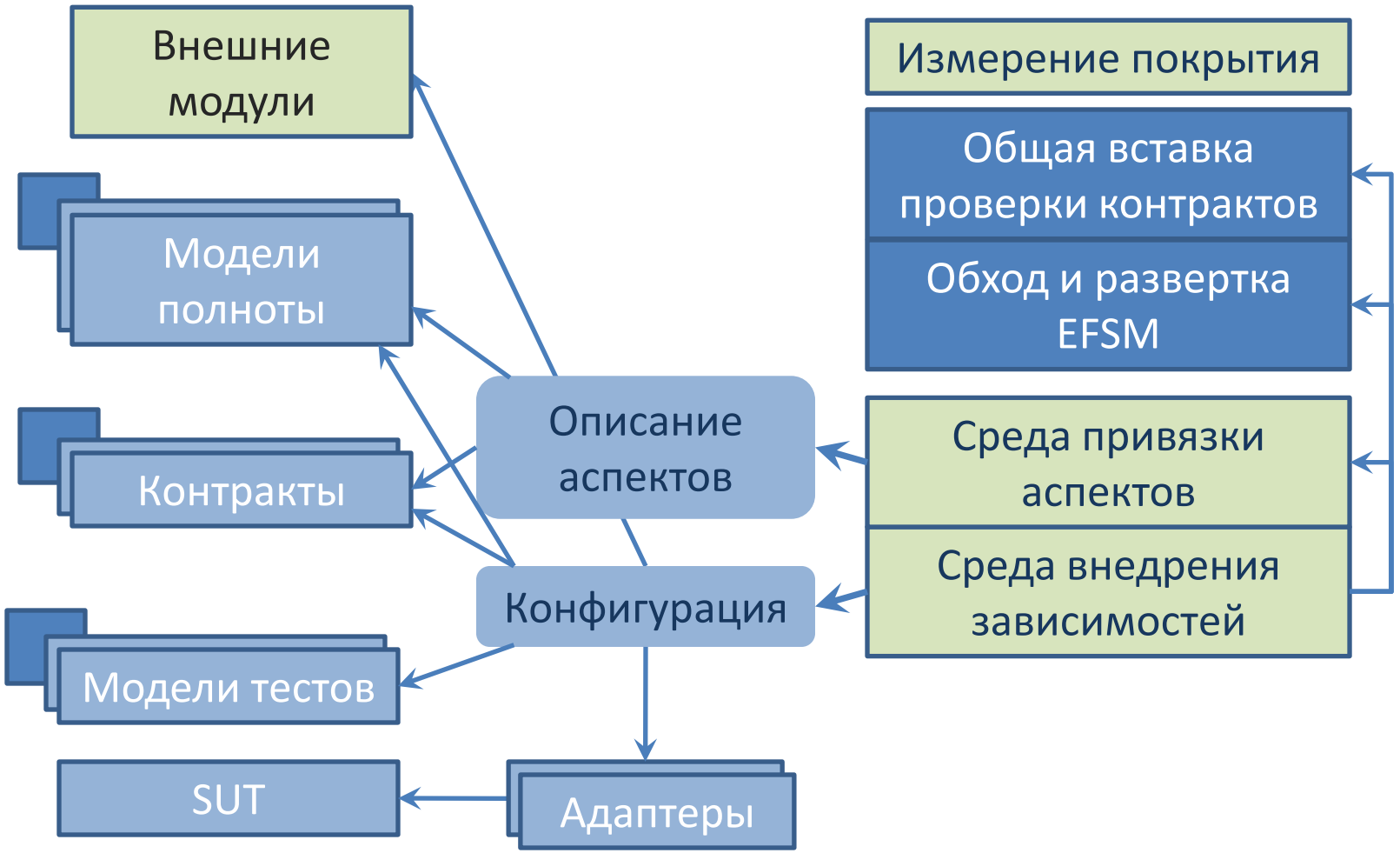
```
obj.M();
```

```
if(!preM()) throw ...;
obj.M();
if(!postM()) throw ...;
```

- Сечение –

  любой вызов метода M() в объектах класса C

Архитектура среды тестирования



Реализация

- Базовый язык – Java
  - Интроспекция (reflection), аннотации
  - Множество инструментов
  - Множество вспомогательных библиотек для поддержки тестирования
- Среда привязки аспектов и внедрения зависимостей – Spring framework
  - http://www.springsource.org

Пример

Account

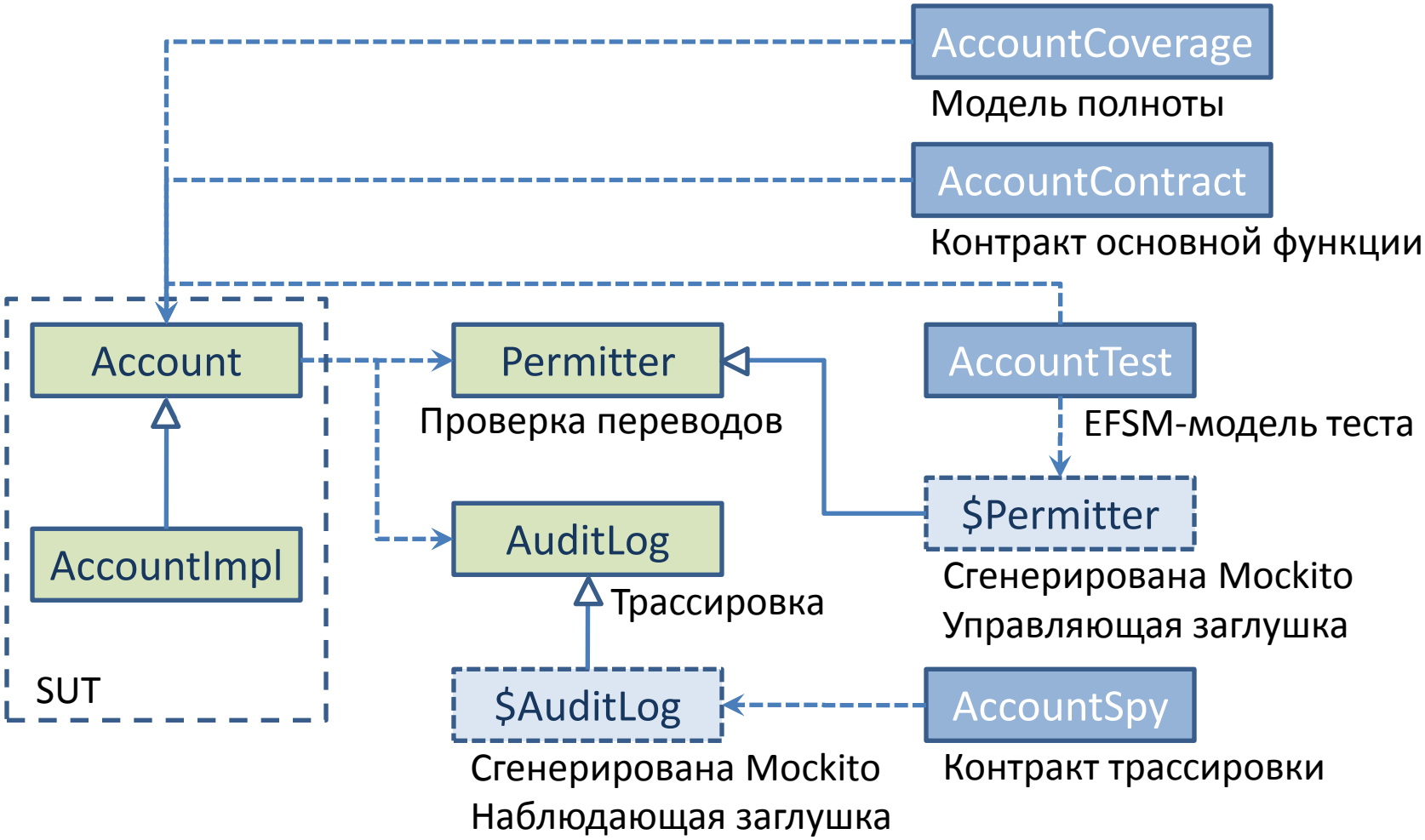
- Одна операция transfer(int sum)
  - sum > 0 : deposit
  - sum < 0 : withdraw
- Возможен кредит (maxCredit ограничивает его величину)
- Все переводы контролируются при помощи внешнего объекта, который может разрешить или запретить перевод
- Параметры и результаты всех вызовов transfer() трассируются для аудита

Конфигурация теста

- EFSM с состоянием (текущий баланс, переводы разрешены/запрещены)
- Тестовые методы
  - transfer
  - Включение/выключение разрешений на переводы с помощью заглушки
- Контракт для основной функции метода transfer
- Заглушка, следящая за трассировкой, и контракт для трассировки
- Модель полноты –
  метод, выделяющий различные ситуации

Заглушки строятся при помощи Mockito [http://code.google.com/p/mockito/]

Диаграмма классов примера



Контракты с состоянием

- Оба контракта используют модельное состояние (значения баланса и границы кредита)
- Для его синхронизации с SUT нужны специальные действия

```
public void transferUpdate(int sum)
{
  if(balance + sum > maxCredit
     && checkedObject.getPermitter()
          .isPermittedTransfer(checkedObject, sum))
    balance += sum;
}
```

Контракт основной функции

```
public boolean transferPost(int sum)
{
  // Validity check result
  boolean permission = checkedObject.getPermitter()
                         .isPermittedTransfer(checkedObject, sum);

  if(Contract.oldBooleanValue(balance + sum > maxCredit) && permission)
    // The transfer is correct and possible
    return    Contract.assertEquals(Contract.intResult(), sum
                , "Result should be equal to the argument")
           && Contract.assertEquals(balance, Contract.oldIntValue(balance) + sum
                , "Balance should be increased by the argument")
           && Contract.assertEquals(maxCredit, Contract.oldIntValue(maxCredit)
                , "Max credit should not change");
  else
    // The transfer is impossible
    return    Contract.assertEquals(Contract.intResult(), 0
                , "Result should be 0")
           && Contract.assertEquals(balance, Contract.oldIntValue(balance)
                , "Balance should not change")
           && Contract.assertEqualsInt(maxCredit, Contract.oldIntValue(maxCredit)
                , "Max credit should not change");
}
```

Контракт трассировки

```
public void transferLogSpy(int sum)
{
  // Validity check result
  boolean permission = checkedObject.getPermitter()
                         .isPermittedTransfer(checkedObject, sum);

  // Whether the transfer possible at all
  boolean possible = (balance + sum > maxCredit) && permission;

  // Calls to spy are verified in order-independent way
  if(possible)
  {
    Mockito.verify(logSpy).logKind("SUCCESS");
    Mockito.verify(logSpy).logNewBalance(balance);
  }
  else if(!permission)
    Mockito.verify(logSpy).logKind("BANNED");
  else
    Mockito.verify(logSpy).logKind("IMPROPER");

  Mockito.verify(logSpy).logOldBalance(oldBalance);
  Mockito.verify(logSpy).logSum(sum);
}
```

Модель полноты

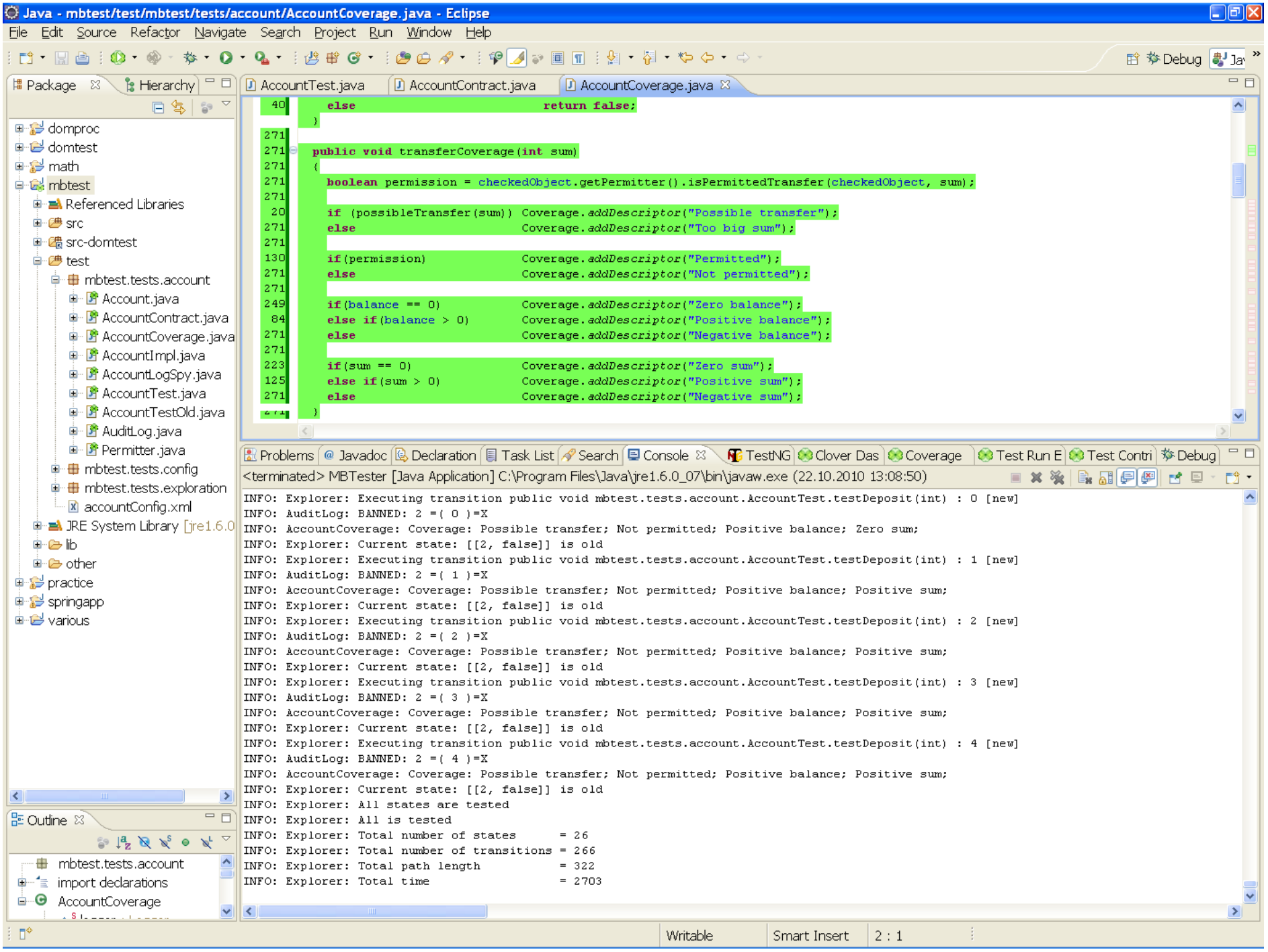
```
public void transferCoverage(int sum)
{
  // Validity check result
  boolean permission = checkedObject.getPermitter()
                         .isPermittedTransfer(checkedObject, sum);

  if(balance + sum > maxCredit) Coverage.addDescriptor("Possible transfer");
  else                          Coverage.addDescriptor("Too big sum");

  if(permission)                Coverage.addDescriptor("Permitted");
  else                          Coverage.addDescriptor("Not permitted");

  if(balance == 0)              Coverage.addDescriptor("Zero balance");
  else if(balance > 0)          Coverage.addDescriptor("Positive balance");
  else                          Coverage.addDescriptor("Negative balance");

  if(sum == 0)                  Coverage.addDescriptor("Zero sum");
  else if(sum > 0)              Coverage.addDescriptor("Positive sum");
  else                          Coverage.addDescriptor("Negative sum");
}
```

Тест : состояние и управл. заглушка

```
@Test public class AccountTest
{
  Account account;
  @Mock Permitter permitterStub;

  boolean permission = true;

  // Init stubs and configure permitterStub to return true on call to isPermittedTransfer()
  public AccountTest() {
    MockitoAnnotations.initMocks(this);
    Mockito.when(permitterStub.isPermittedTransfer(Mockito.<Account>any(), Mockito.anyInt()))
           .thenReturn(permission);
  }

  public void setAccount(Account account) {
    this.account = account;
    account.setPermitter(permitterStub);
  }

  // Current permission and balance are two components of the test state
  @State
  public boolean getPermission() { return permission; }

  @State
  public int getBalance() { return account.getBalance(); }

  ...
}
```

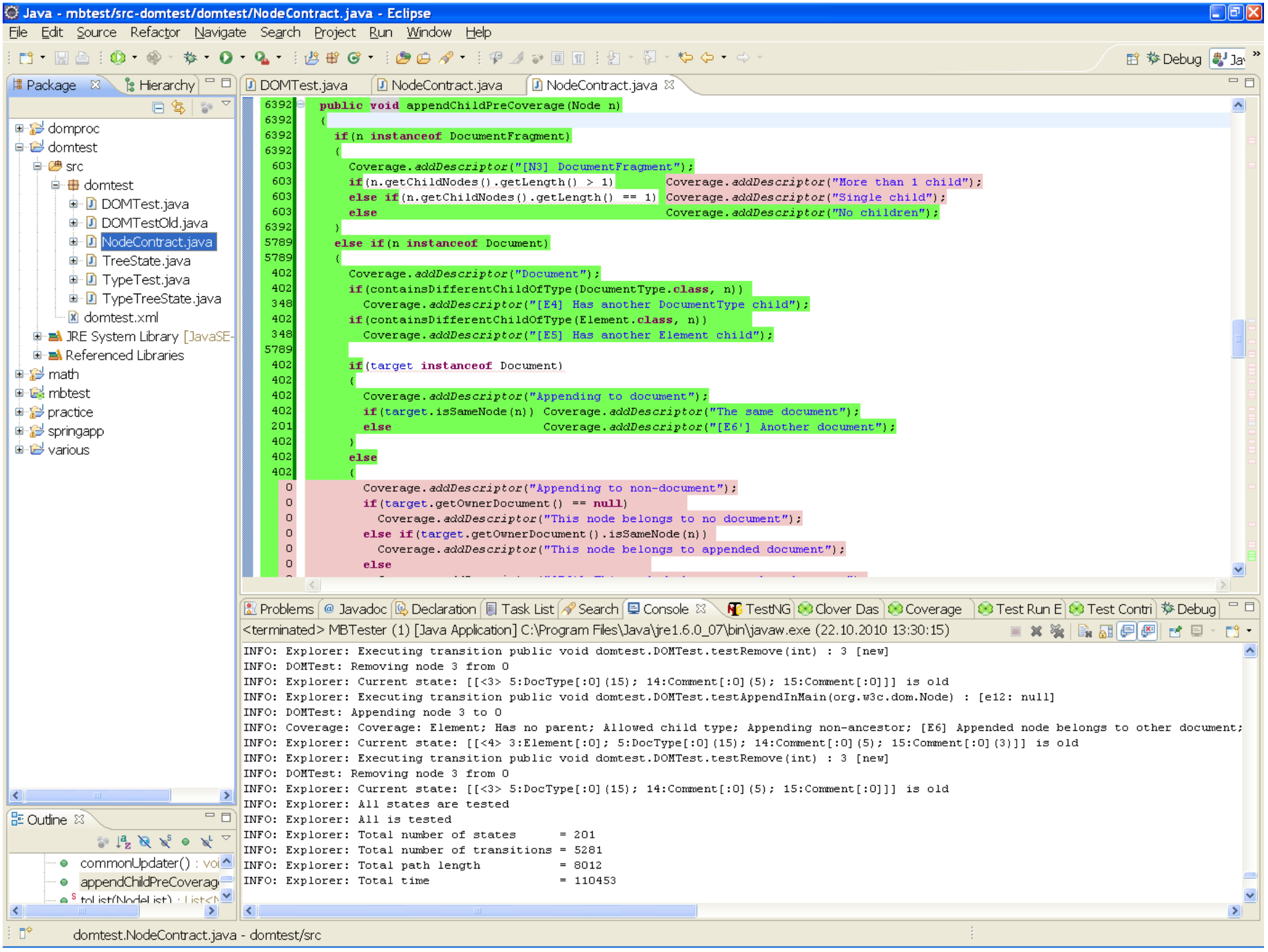
Тест : действия

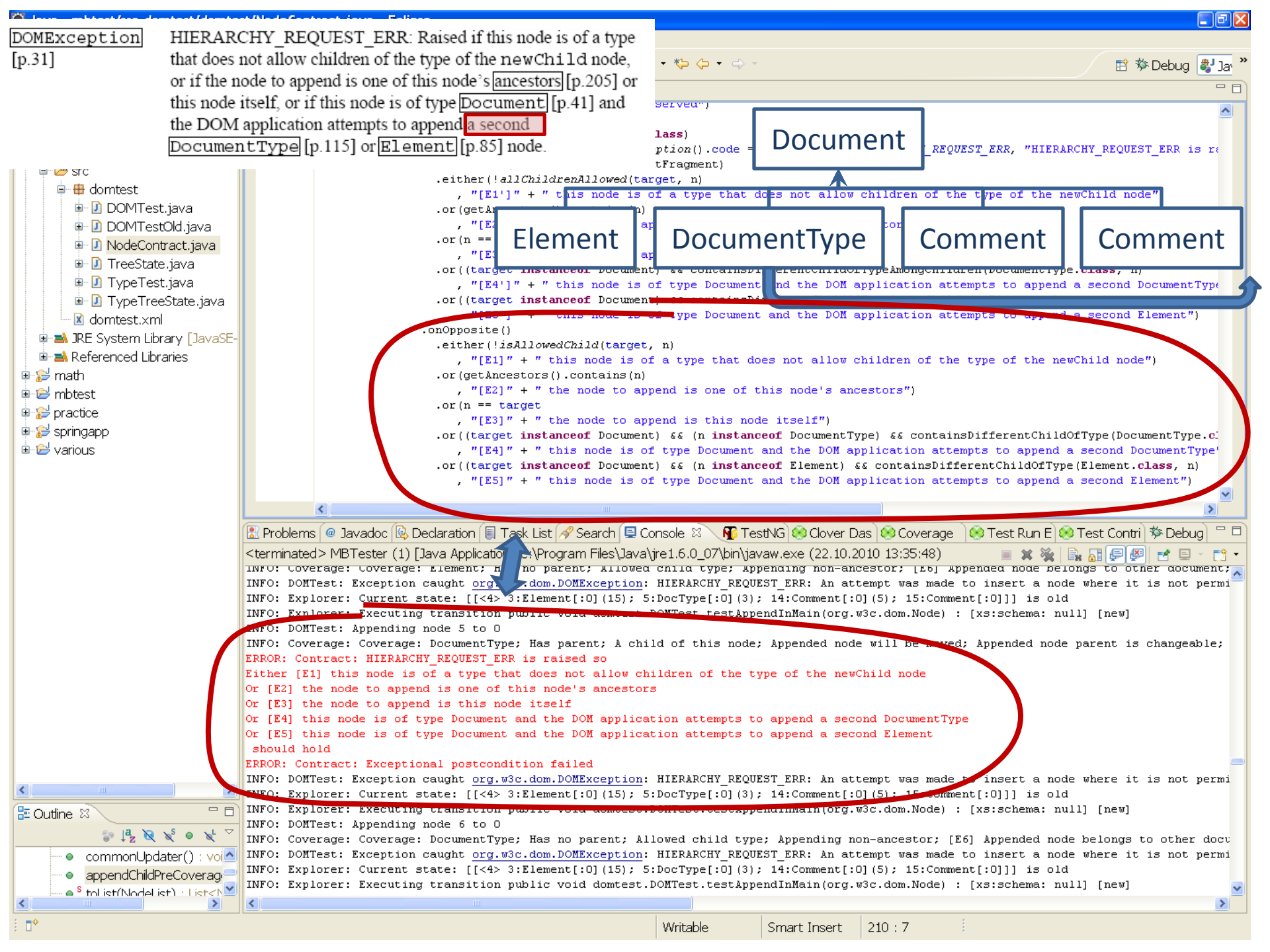
```
@Test public class AccountTest
{
  ...
  @Test(dependsOnMethods="testWithdraw")
  @DataProvider(name = "sumArray")
  @Guard(name = "bound")
  public void testDeposit(int x)   { account.transfer(x);  }

  @Test(dependsOnMethods="switchPermission")
  @DataProvider(name = "sumArray")
  public void testWithdraw(int x)  { account.transfer(-x); }

  // Switch permission and configure permitterStub to its value true on call to isPermittedTransfer()
  @Test
  public void switchPermission()
  {
    permission = !permission;
    Mockito.when(permitterStub.isPermittedTransfer(Mockito.<Account>any(), Mockito.anyInt()))
           .thenReturn(permission);
  }

  // Guardian for deposits to bound the possible balance values
  public boolean bound() { return getBalance() < 5 || !permission; }

  // Source of test data for both transfer test methods
  public int[] sumArray = new int[]{0, 1, 2, 3, 4};
}
```


Java - mbtest/test/mbtest/tests/account/AccountCoverage.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Hierarchy
domproc
domtest
math
mbtest
Referenced Libraries
src
src-domtest
test
mbtest.tests.account
Account.java
AccountContract.java
AccountCoverage.java
AccountImpl.java
AccountLogSpy.java
AccountTest.java
AccountTestOld.java
AuditLog.java
Permitter.java
mbtest.tests.config
mbtest.tests.exploration
accountConfig.xml
JRE System Library [jre1.6.0
lib
other
practice
springapp
various
Outline
mbtest.tests.account
import declarations
AccountCoverage
AccountTest.java
AccountContract.java
AccountCoverage.java
40 else return false;
}
271
271 public void transferCoverage(int sum)
271 {
271 boolean permission = checkedObject.getPermitter().isPermittedTransfer(checkedObject, sum);
271
20 if (possibleTransfer(sum)) Coverage.addDescriptor("Possible transfer");
271 else Coverage.addDescriptor("Too big sum");
271
130 if(permission) Coverage.addDescriptor("Permitted");
271 else Coverage.addDescriptor("Not permitted");
271
249 if(balance == 0) Coverage.addDescriptor("Zero balance");
84 else if(balance > 0) Coverage.addDescriptor("Positive balance");
271 else Coverage.addDescriptor("Negative balance");
271
223 if(sum == 0) Coverage.addDescriptor("Zero sum");
125 else if(sum > 0) Coverage.addDescriptor("Positive sum");
271 else Coverage.addDescriptor("Negative sum");
}
Problems Javadoc Declaration Task List Search Console TestNG Clover Das Coverage Test Run E Test Contri Debug
<terminated> MBTester [Java Application] C:\Program Files\Java\jre1.6.0_07\bin\javaw.exe (22.10.2010 13:08:50)
INFO: Explorer: Executing transition public void mbtest.tests.account.AccountTest.testDeposit(int) : 0 [new]
INFO: AuditLog: BANNED: 2 =(0)=X
INFO: AccountCoverage: Coverage: Possible transfer; Not permitted; Positive balance; Zero sum;
INFO: Explorer: Current state: [[2, false]] is old
INFO: Explorer: Executing transition public void mbtest.tests.account.AccountTest.testDeposit(int) : 1 [new]
INFO: AuditLog: BANNED: 2 =(1)=X
INFO: AccountCoverage: Coverage: Possible transfer; Not permitted; Positive balance; Positive sum;
INFO: Explorer: Current state: [[2, false]] is old
INFO: Explorer: Executing transition public void mbtest.tests.account.AccountTest.testDeposit(int) : 2 [new]
INFO: AuditLog: BANNED: 2 =(2)=X
INFO: AccountCoverage: Coverage: Possible transfer; Not permitted; Positive balance; Positive sum;
INFO: Explorer: Current state: [[2, false]] is old
INFO: Explorer: Executing transition public void mbtest.tests.account.AccountTest.testDeposit(int) : 3 [new]
INFO: AuditLog: BANNED: 2 =(3)=X
INFO: AccountCoverage: Coverage: Possible transfer; Not permitted; Positive balance; Positive sum;
INFO: Explorer: Current state: [[2, false]] is old
INFO: Explorer: Executing transition public void mbtest.tests.account.AccountTest.testDeposit(int) : 4 [new]
INFO: AuditLog: BANNED: 2 =(4)=X
INFO: AccountCoverage: Coverage: Possible transfer; Not permitted; Positive balance; Positive sum;
INFO: Explorer: Current state: [[2, false]] is old
INFO: Explorer: All states are tested
INFO: Explorer: All is tested
INFO: Explorer: Total number of states = 26
INFO: Explorer: Total number of transitions = 266
INFO: Explorer: Total path length = 322
INFO: Explorer: Total time = 2703
Writable
Smart Insert
2 : 1

Еще пример

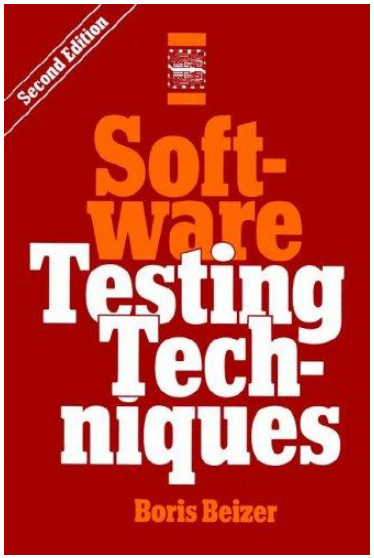
- Реализация DOM API на Java
  - Xerces for Java [http://xerces.apache.org]
- Манипуляции дочерними узлами
  - appendChild(Node n)
  - removeChild(Node n)

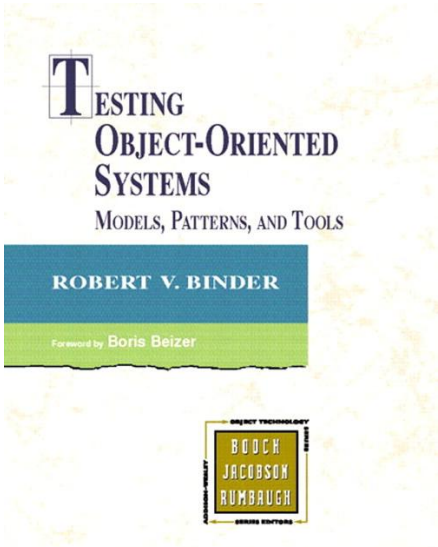

Java - mbtest/src-domtest/domtest/NodeContract.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Hierarchy
domproc
domtest
src
domtest
DOMTest.java
DOMTestOld.java
NodeContract.java
TreeState.java
TypeTest.java
TypeTreeState.java
domtest.xml
JRE System Library [JavaSE-
Referenced Libraries
math
mbtest
practice
springapp
various
DOMTest.java
NodeContract.java
NodeContract.java
6392 public void appendChildPreCoverage(Node n)
6392 {
6392 if(n instanceof DocumentFragment)
6392 {
603 Coverage.addDescriptor("[N3] DocumentFragment");
603 if(n.getChildNodes().getLength() > 1) Coverage.addDescriptor("More than 1 child");
603 else if(n.getChildNodes().getLength() == 1) Coverage.addDescriptor("Single child");
603 else Coverage.addDescriptor("No children");
6392 }
5789 else if(n instanceof Document)
5789 {
402 Coverage.addDescriptor("Document");
402 if(containsDifferentChildOfType(DocumentType.class, n))
348 Coverage.addDescriptor("[E4] Has another DocumentType child");
402 if(containsDifferentChildOfType(Element.class, n))
348 Coverage.addDescriptor("[E5] Has another Element child");
5789
402 if(target instanceof Document)
402 {
402 Coverage.addDescriptor("Appending to document");
402 if(target.isSameNode(n)) Coverage.addDescriptor("The same document");
201 else Coverage.addDescriptor("[E6'] Another document");
402 }
402 else
402 {
0 Coverage.addDescriptor("Appending to non-document");
0 if(target.getOwnerDocument() == null)
0 Coverage.addDescriptor("This node belongs to no document");
0 else if(target.getOwnerDocument().isSameNode(n))
0 Coverage.addDescriptor("This node belongs to appended document");
0 else
Problems Javadoc Declaration Task List Search Console TestNG Clover Das Coverage Test Run E Test Contri Debug
<terminated> MBTester (1) [Java Application] C:\Program Files\Java\jre1.6.0_07\bin\javaw.exe (22.10.2010 13:30:15)
INFO: Explorer: Executing transition public void domtest.DOMTest.testRemove(int) : 3 [new]
INFO: DOMTest: Removing node 3 from 0
INFO: Explorer: Current state: [[<3> 5:DocType[:0](15); 14:Comment[:0](5); 15:Comment[:0]]] is old
INFO: Explorer: Executing transition public void domtest.DOMTest.testAppendInMain(org.w3c.dom.Node) : [e12: null]
INFO: DOMTest: Appending node 3 to 0
INFO: Coverage: Coverage: Element; Has no parent; Allowed child type; Appending non-ancestor; [E6] Appended node belongs to other document;
INFO: Explorer: Current state: [[<4> 3:Element[:0]; 5:DocType[:0](15); 14:Comment[:0](5); 15:Comment[:0](3)]] is old
INFO: Explorer: Executing transition public void domtest.DOMTest.testRemove(int) : 3 [new]
INFO: DOMTest: Removing node 3 from 0
INFO: Explorer: Current state: [[<3> 5:DocType[:0](15); 14:Comment[:0](5); 15:Comment[:0]]] is old
INFO: Explorer: All states are tested
INFO: Explorer: All is tested
INFO: Explorer: Total number of states = 201
INFO: Explorer: Total number of transitions = 5281
INFO: Explorer: Total path length = 8012
INFO: Explorer: Total time = 110453
Outline
commonUpdater() : voi
appendChildPreCoverag
domtest.NodeContract.java - domtest/src


DOMException [p.31]
HIERARCHY_REQUEST_ERR: Raised if this node is of a type that does not allow children of the type of the newChild node, or if the node to append is one of this node's ancestors [p.205] or this node itself, or if this node is of type Document [p.41] and the DOM application attempts to append a second DocumentType [p.115] or Element [p.85] node.
Document
Element
DocumentType
Comment
Comment
.either(!allChildrenAllowed(target, n)
, "[E1']" + " this node is of a type that does not allow children of the type of the newChild node"
.or(getAncestors().contains(n)
, "[E2]" + " the node to append is one of this node's ancestors"
.or(n == target
, "[E3]" + " the node to append is this node itself"
.or((target instanceof Document) && (n instanceof DocumentType) && containsDifferentChildOfType(DocumentType.class, n)
, "[E4']" + " this node is of type Document and the DOM application attempts to append a second DocumentType"
.or((target instanceof Document) && (n instanceof Element) && containsDifferentChildOfType(Element.class, n)
, "[E5]" + " this node is of type Document and the DOM application attempts to append a second Element")
.onOpposite()
.either(!isAllowedChild(target, n)
, "[E1]" + " this node is of a type that does not allow children of the type of the newChild node")
.or(getAncestors().contains(n)
, "[E2]" + " the node to append is one of this node's ancestors")
.or(n == target
, "[E3]" + " the node to append is this node itself")
.or((target instanceof Document) && (n instanceof DocumentType) && containsDifferentChildOfType(DocumentType.c
, "[E4]" + " this node is of type Document and the DOM application attempts to append a second DocumentType'
.or((target instanceof Document) && (n instanceof Element) && containsDifferentChildOfType(Element.class, n)
, "[E5]" + " this node is of type Document and the DOM application attempts to append a second Element")
Problems Javadoc Declaration Task List Search Console TestNG Clover Das Coverage Test Run E Test Contri Debug
<terminated> MBTester (1) [Java Application] C:\Program Files\Java\jre1.6.0_07\bin\javaw.exe (22.10.2010 13:35:48)
INFO: DOMTest: Exception caught org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permi
INFO: Explorer: Current state: [[<4> 3:Element[:0](15); 5:DocType[:0](3); 14:Comment[:0](5); 15:Comment[:0]]] is old
INFO: DOMTest: Appending node 5 to 0
INFO: Coverage: Coverage: DocumentType; Has parent; A child of this node; Appended node will be moved; Appended node parent is changeable;
ERROR: Contract: HIERARCHY_REQUEST_ERR is raised so
Either [E1] this node is of a type that does not allow children of the type of the newChild node
Or [E2] the node to append is one of this node's ancestors
Or [E3] the node to append is this node itself
Or [E4] this node is of type Document and the DOM application attempts to append a second DocumentType
Or [E5] this node is of type Document and the DOM application attempts to append a second Element
should hold
ERROR: Contract: Exceptional postcondition failed
INFO: DOMTest: Exception caught org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permi
INFO: Explorer: Current state: [[<4> 3:Element[:0](15); 5:DocType[:0](3); 14:Comment[:0](5); 15:Comment[:0]]] is old
INFO: DOMTest: Appending node 6 to 0
INFO: Coverage: Coverage: DocumentType; Has no parent; Allowed child type; Appending non-ancestor; [E6] Appended node belongs to other docu
INFO: DOMTest: Exception caught org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permi
INFO: Explorer: Current state: [[<4> 3:Element[:0](15); 5:DocType[:0](3); 14:Comment[:0](5); 15:Comment[:0]]] is old
INFO: Explorer: Executing transition public void domtest.DOMTest.testAppendInMain(org.w3c.dom.Node) : [xs:schema: null] [new]
Outline
commonUpdater() : voi
appendChildPreCoverag
Writable
Smart Insert
210 : 7

Общие книги по тестированию

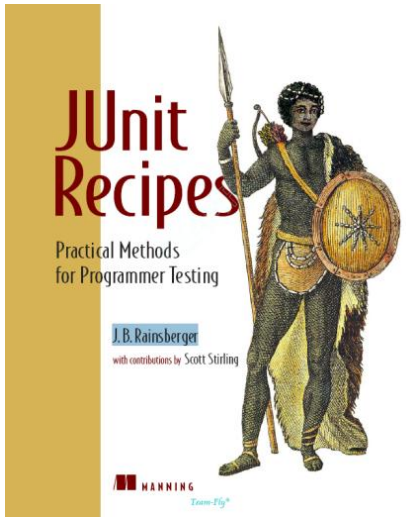
- B. Beizer. Software Testing Techniques. Thomson Computer Press, 1990
- R. Binder. Testing Object-Oriented Systems: Models, Patterns, and Tools. Addison-Wesley, 2000

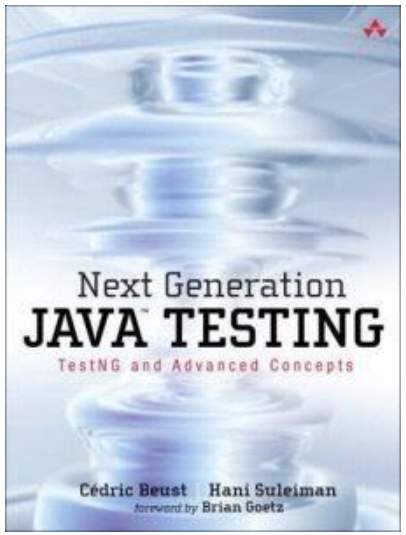




Модульное тестирование (и больше)

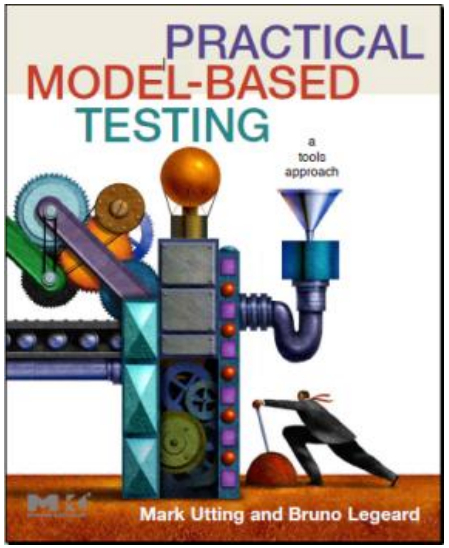
- J.B.Rainsberger. JUnit Recipes. Practical Methods for Programmer Testing. Manning, 2005
- C. Beust, H. Suleiman. Next Generation Java Testing. TestNG and Advanced Concepts. Addison-Wesley, 2007

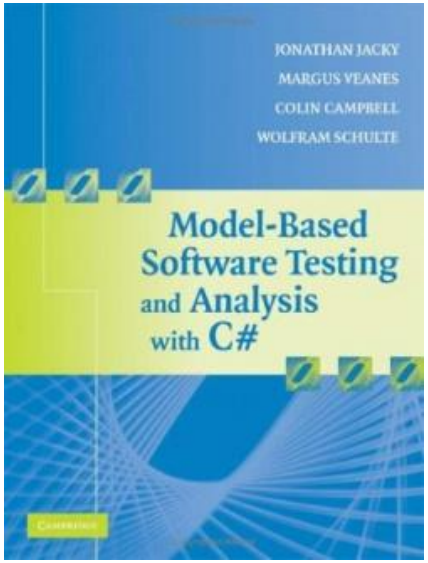




Тестирование на основе моделей

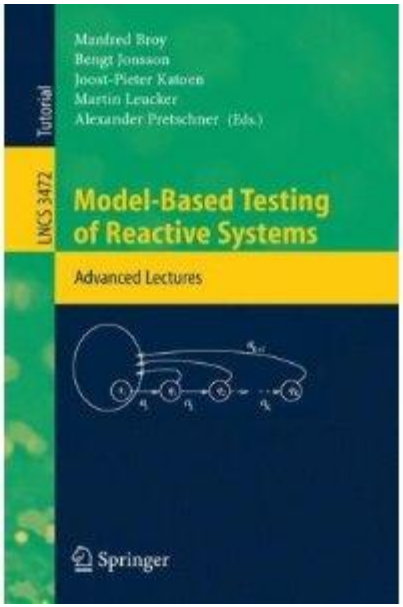
- M. Utting, B. Legeard. Practical Model-Based Testing. A Tools Approach. Morgan Kaufmann, 2006
- J. Jacky, M. Veanes, C. Campbell, W. Schulte. Model Based Analysis and Testing with C#. Cambridge University Press, 2007





Теория тестир-я на основе моделей

- M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, A. Pretschner, editors. Model Based Testing of Reactive Systems. Advanced Lectures. LNCS 3472, Springer, 2005



Книги на русском языке

- Г. Майерс. Искусство тестирования программ. Финансы и статистика, 1982
- Д. Месарош. Шаблоны тестирования xUnit. Рефакторинг кода тестов. Вильямс, 2009
- Мой курс http://mbt-course.narod.ru

Спасибо за внимание!

Виктор Кулямин

kuliamin@ispras.ru
www.ispras.ru/~kuliamin