Component-based Framework for Model Based Testing

Victor Kuliamin

Institute for System Programming

Moscow, Russia

Disclaimer and Outline

❖Work in progress

- Motivation – integration problems
- Existing approaches
- Proposed solution
- Some examples

Motivation

- A lot of promising model based testing (MBT) techniques emerge every year
  - They need to be evaluated in various contexts
  - They can be combined in different ways
  - To integrate them all in one tool is very expensive
  - Most of them are not useful in each separate project
  - They also need integration in development process

$\Rightarrow$Need for a component MBT frameworks, which allow rapid integration of emerging techniques in chosen combinations

Additional Observations

- Various other development tools (analyzers, debuggers, monitors, etc.) also evolve quickly
  - It is advantageous to use them along with MBT
  - MBT based on their own languages or non-standard extensions (not supported by standard tools for base language) are hard to maintain

$\Rightarrow$Library- and standard extensions-based solutions are preferable

Perspective Approaches I

Generic unit testing tools (xUnit, JUnit descendants)

- Lightweight, “stealth” frameworks
- Configurable
- Easy to integrate with other tools and components
- TestNG [2003]
  http://www.testng.org
  - Use of annotations instead of naming conventions
  - Expected exceptions and timeouts before checks
  - Elaborated hierarchy of test components, test groups
  - Setup and tear-down methods for each kind of groups
  - Dependencies of test methods and groups
  - Test data as test method arguments, test object factories
  - External configuration in XML files

Perspective Approaches II

MBT tools of the same style

- ModelJUnit [2004]
  http://czt.sourceforge.net/modeljunit/index.html

  JUnit + EFSM-based MBT engine
  - Test class considered as EFSM
  - Test methods are possible actions
  - State is calculated by special method
  - Guardians (also methods) can be set for actions
- NModel [2007]
  http://nmodel.codeplex.com/
  - EFSM-based engine, attributes instead of special names
  - Test data as test method arguments
  - Composition of models
  - Support for model analysis

Additional Features I

- Other modeling techniques than EFSMs
  - Interface contracts and assertions
    - Behavior driven development [2003]

```
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.Be(x).Or.Be(y);
```

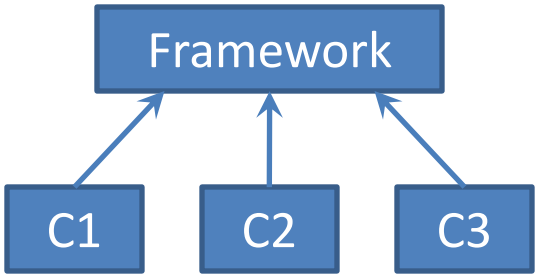
Additional Features II

- Other modeling techniques than EFSMs
  - Interface contracts and assertions
    - Behavior driven development [2003]
    - CodeContracts [2009] http://research.microsoft.com/en-us/projects/contracts/
- Auxiliary components for unit testing
  - httpUnit, dbUnit, etc.
  - Mocks and spies
  - Test data generators
- Model-based test coverage measurement

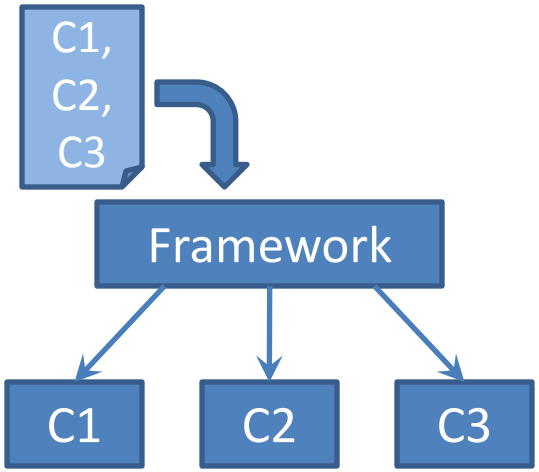
Core Architecture Ideas

- Use of annotations and libraries to implement MBT concepts
  - Easy integration with a lot of development tools
- Dependency injection
  - Easy configurability and extensibility
  - Non-invasive integration of various components
- Aspect-based configuration
  - Non-invasive monitoring of system under test and event processing

Dependency Injection

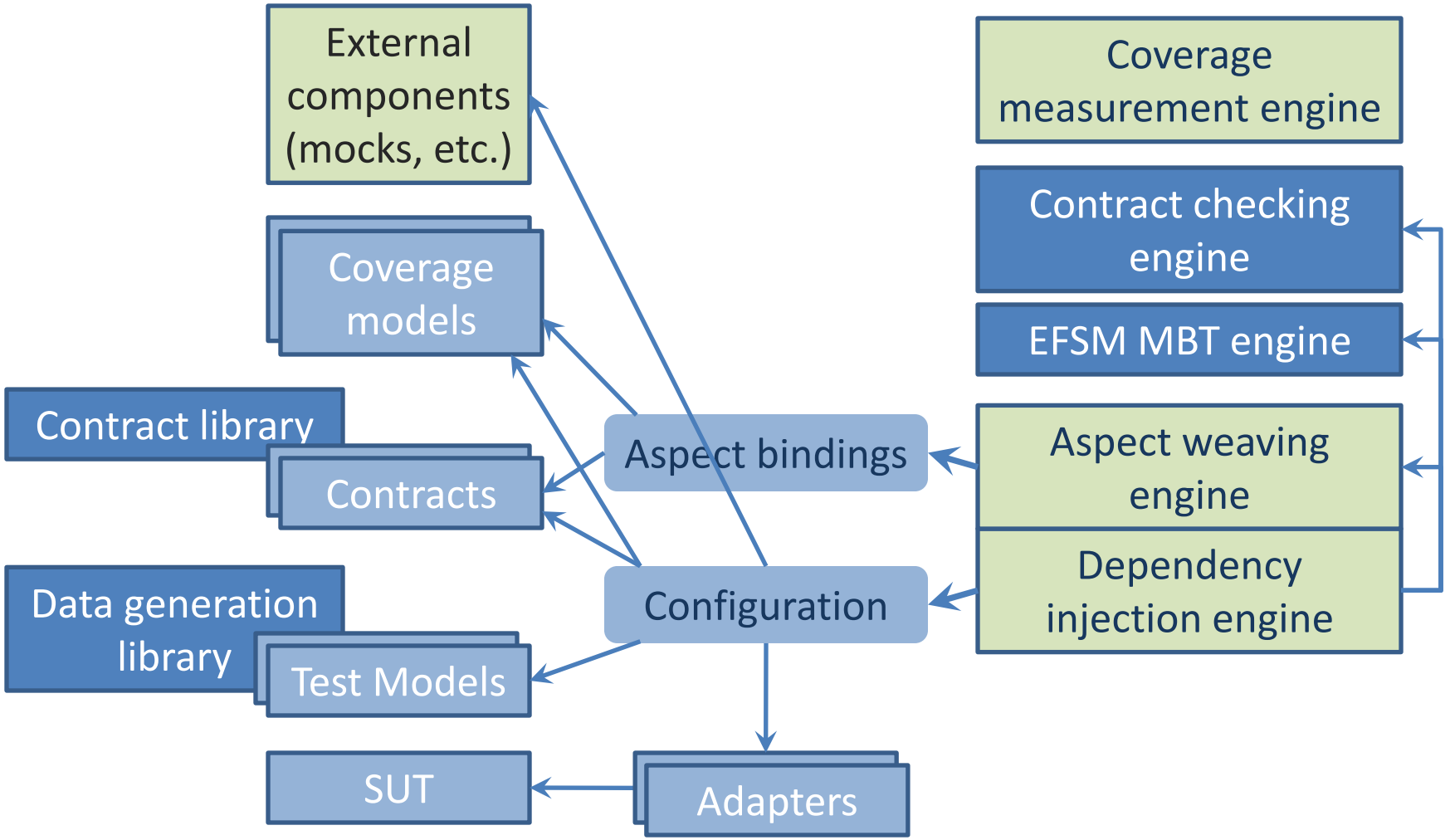


Centralized Integration



Integration based on
Dependency Injection

Architecture Scheme



Prototype Implementation

- Base language – Java
  - Reflection
  - Annotations
  - Plentitude of tools
  - A lot of auxiliary libraries for unit testing
- Dependency injection and aspect engine – Spring framework [www.springsource.org]
  - Long usage and evolution history

Example

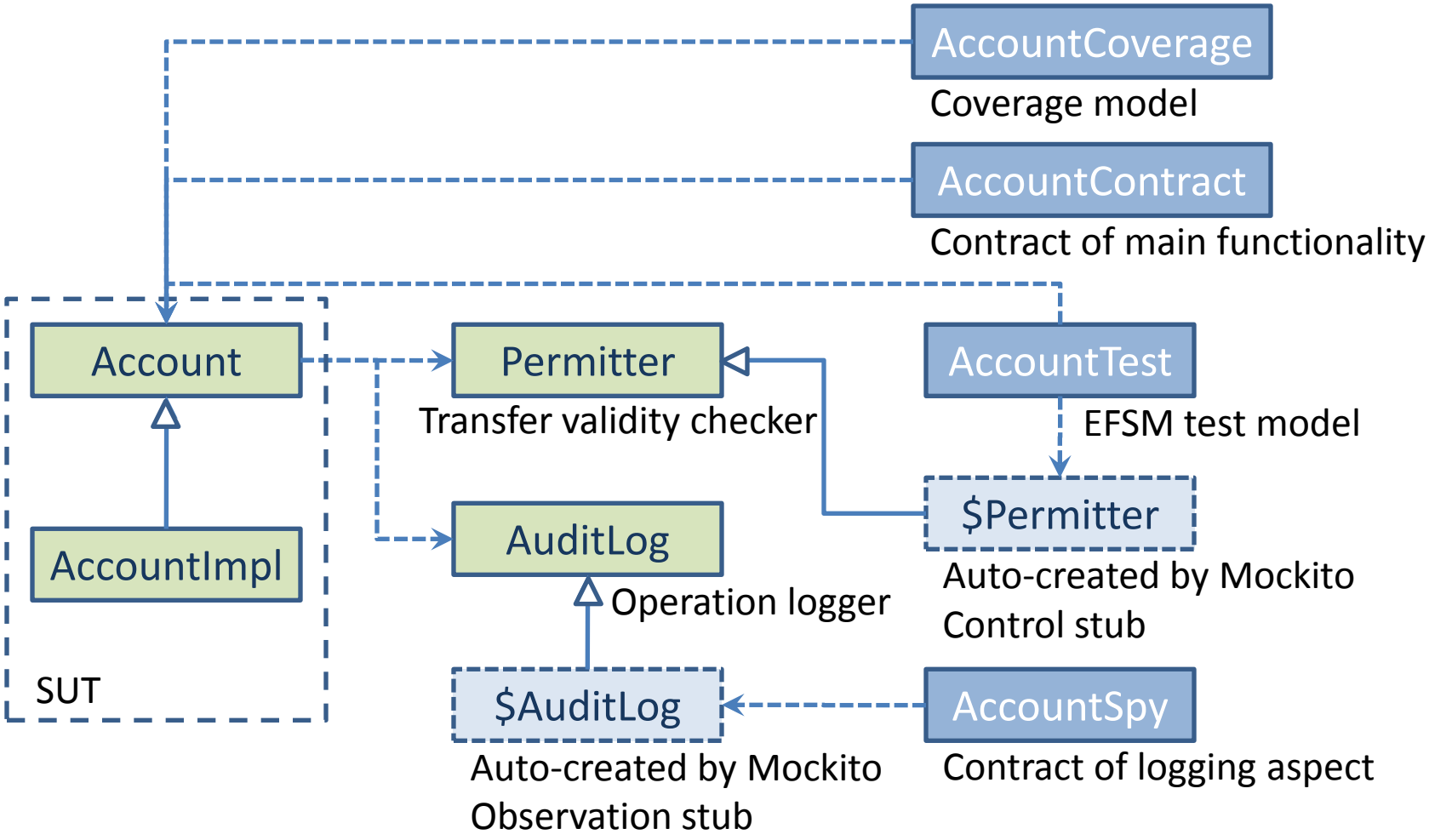
Bank account

- Single operation `transfer(int sum)`
  - sum > 0 : deposit
  - sum < 0 : withdraw
- Stores current balance (`int`, overflow prohibited)
- Credit is possible (`maxCredit` bounds possible withdrawals)
- All transfers are monitored and can be banned by external validity checker
- All transfers and their results are logged for audit

Test system

- EFSM with state (balance, transferAllowed)
- Test methods for transfer, and switching on/off transfer allowance in validity checker stub
- Contract for transfer method
- Multi-aspect coverage model
- Spy on logging
  Stub and spy library – Mockito [http://code.google.com/p/mockito/]

Account Example Class Diagram



Stateful contracts

- Both contracts support their own copy of SUT state (values of `balance` and `maxCredit`)
- To synchronize it with the SUT state they need update methods

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

Contract of Main Functionality

```
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");
}
```

Contract of Logging Aspect

```
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);
}
```

Coverage Model

```
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");
}
```

Test Model : State and Control Stub

```
@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(); }

  ...
}
```

Test Model : Actions

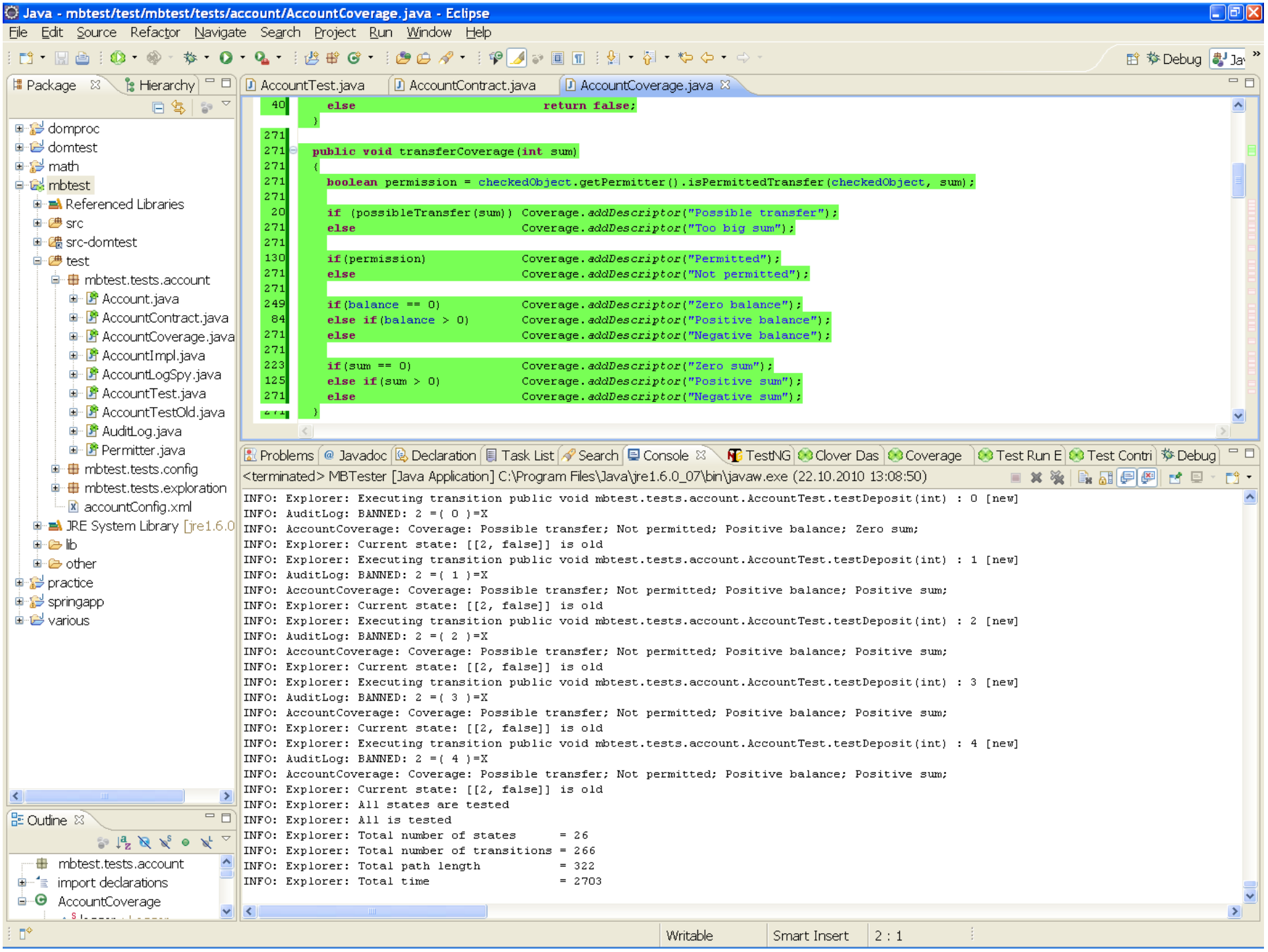
```
@Test public class AccountTest
{
  ...
  @Test(dependsOnMethods="testWithdraw")
  @DataProvider(name = "sumArray")
  @Guard(name = "bound")
  public void testDeposit(int x)   { account.transfer(x);  }

  @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

More Realistic Example

- DOM API in Java
  - SUT – Xerces for Java [xerces.apache.org]
- Node children manipulation
  - appendChild(Node n)
  - removeChild(Node n)

DOM Standard Requirements

appendChild modified in **DOM Level 3**

Adds the node `newChild` to the end of the list of children of this node. If the `newChild` is already in the tree, it is first removed.

Parameters

`newChild` of type `Node` [p.56]

The node to add.

If it is a `DocumentFragment` [p.40] object, the entire contents of the document fragment are moved into the child list of this node

Return Value

`Node` [p.56] The node added.

Exceptions

`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.

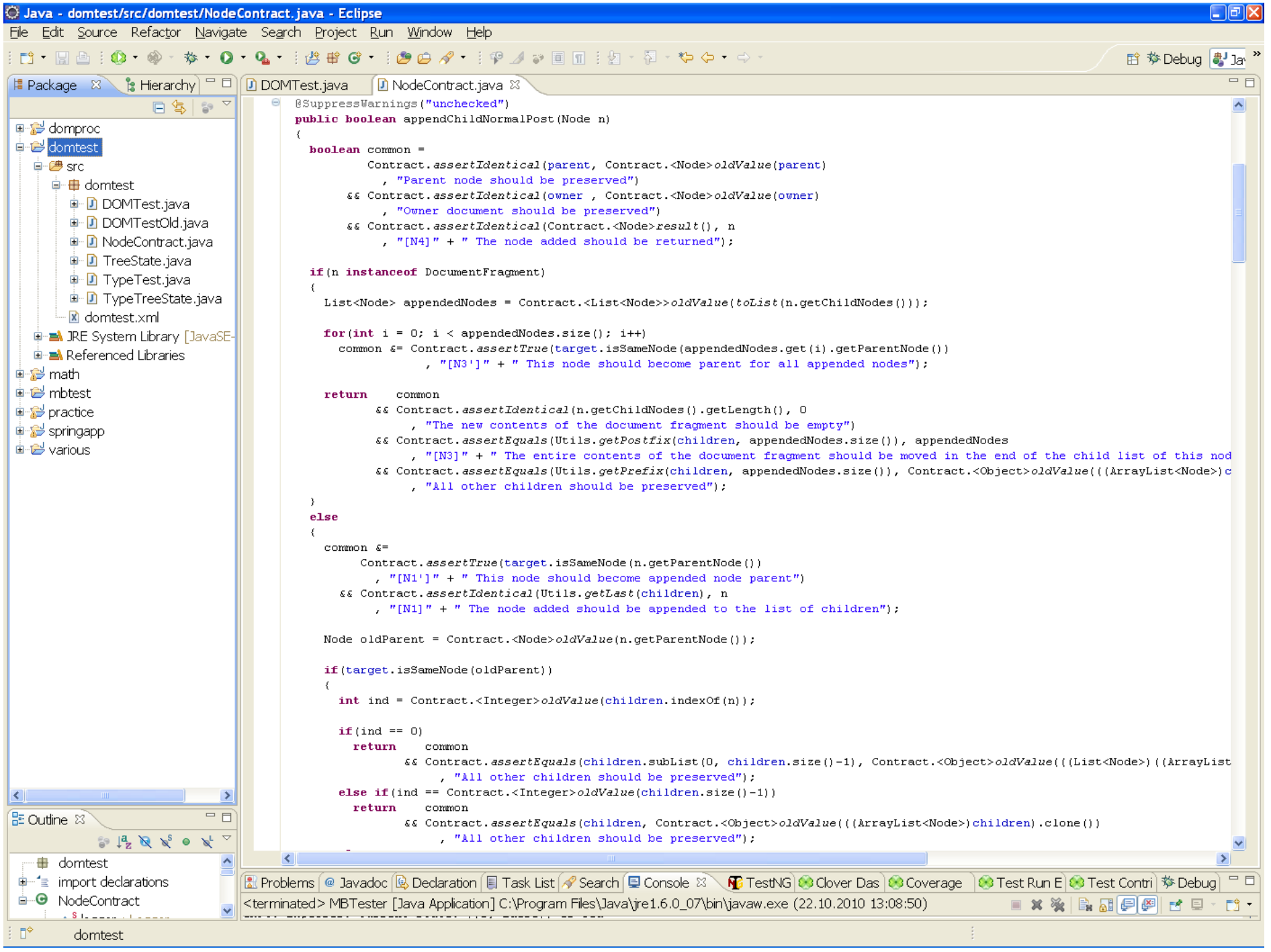
WRONG_DOCUMENT_ERR: Raised if `newChild` was created from a different document than the one that created this node.

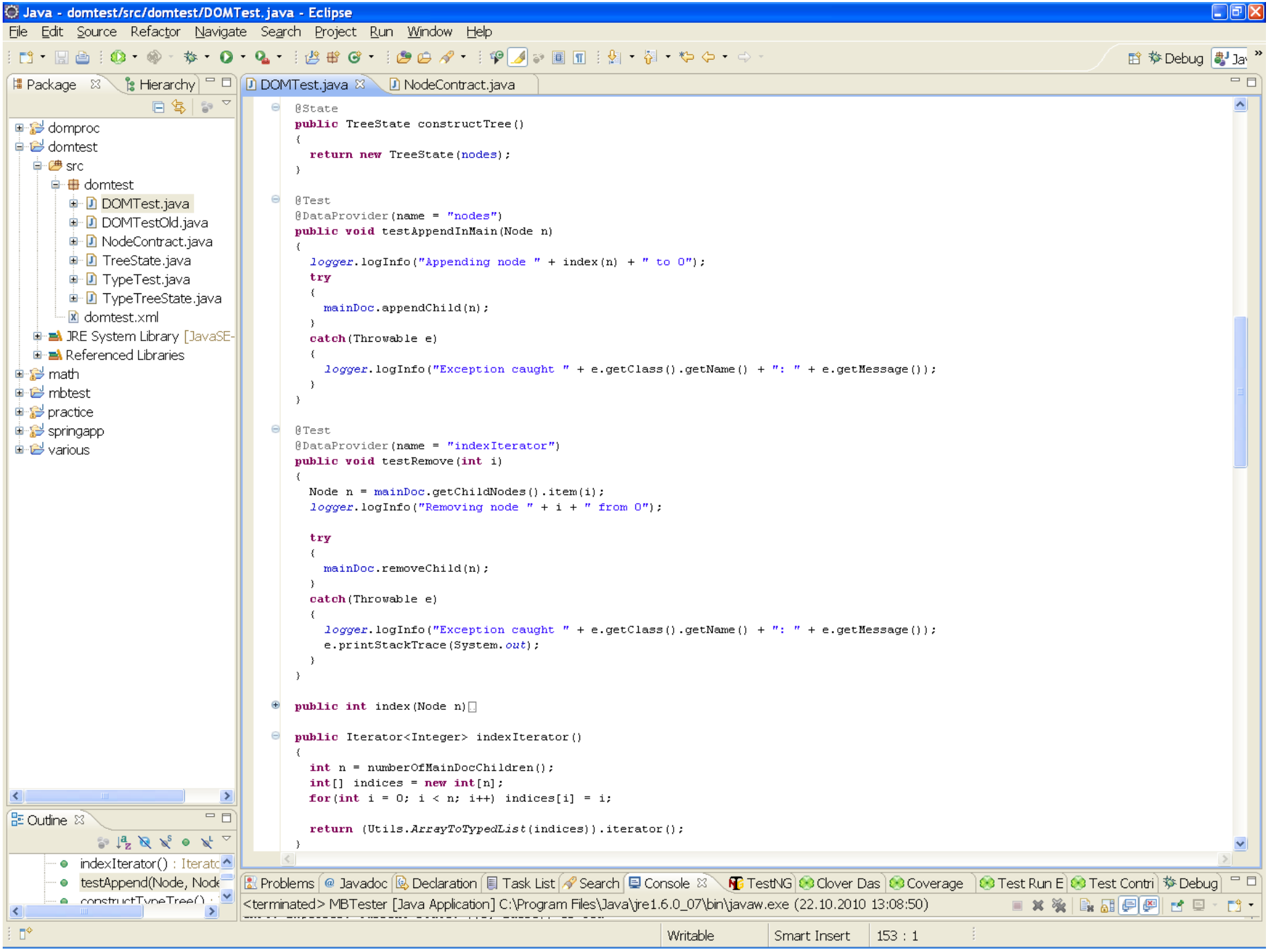
NO_MODIFICATION_ALLOWED_ERR: Raised if this node is readonly or if the previous parent of the node being inserted is readonly.

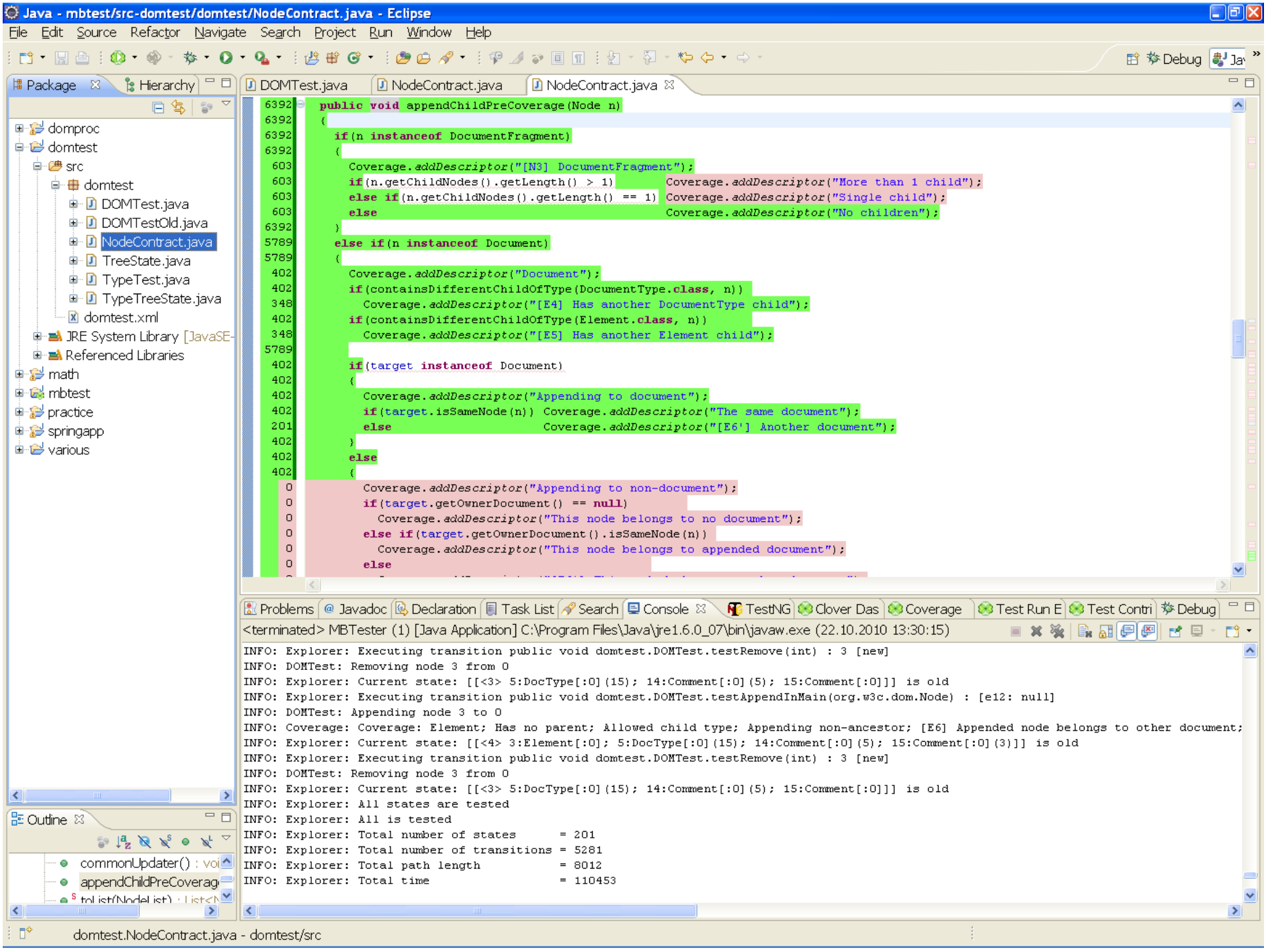
NOT_SUPPORTED_ERR: if the `newChild` node is a child of the `Document` [p.41] node, this exception might be raised if the DOM implementation doesn't support the removal of the `DocumentType` [p.115] child or `Element` [p.85] child.

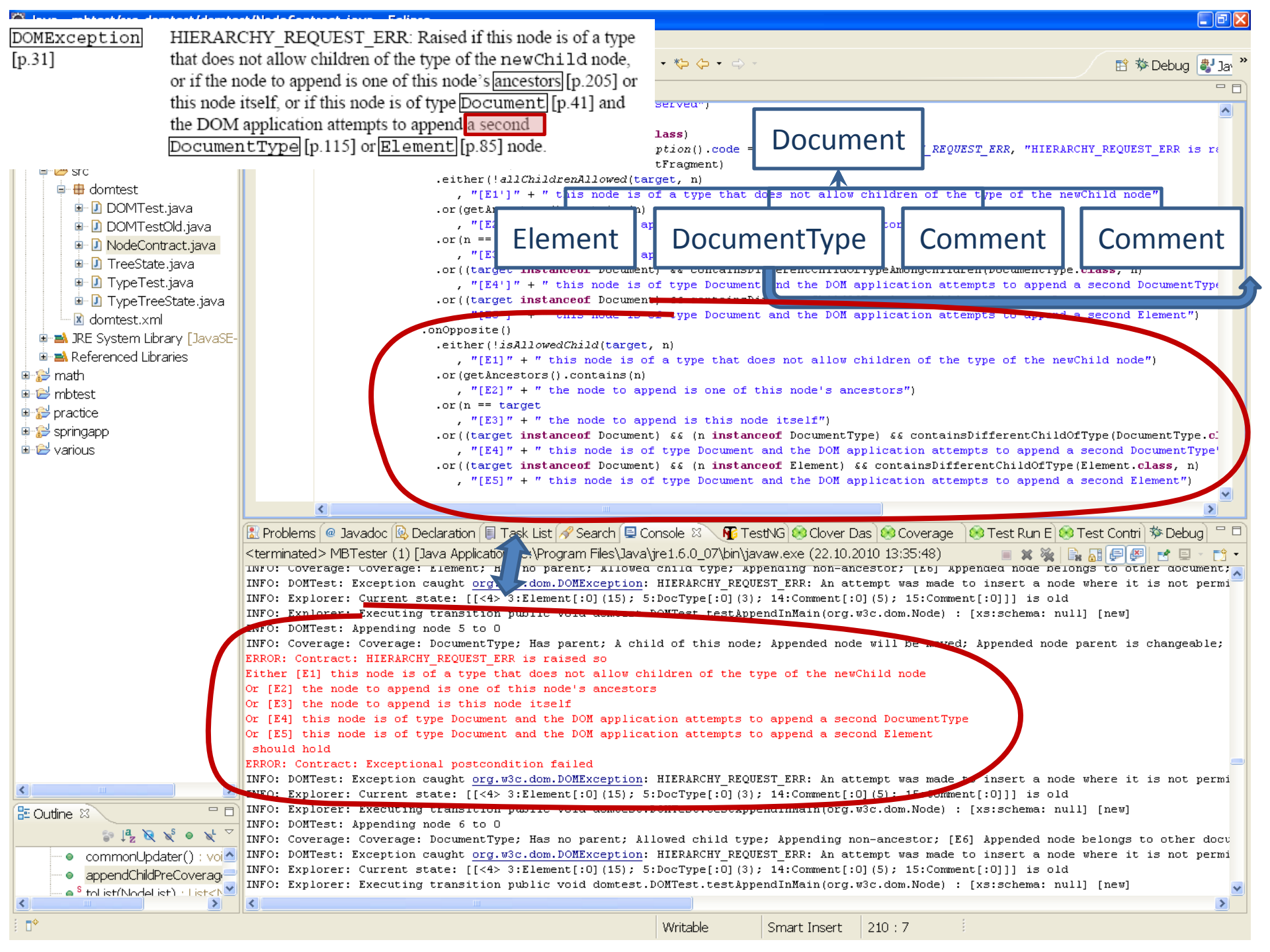
N1 N2 N3 N4

E1 E2 E3 E4 E5 E6 E7 E8 E9 E10


Java - domtest/src/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
Outline
domtest
import declarations
NodeContract
DOMTest.java
NodeContract.java
@SuppressWarnings("unchecked")
public boolean appendChildNormalPost(Node n)
{
boolean common =
Contract.assertIdentical(parent, Contract.<Node>oldValue(parent)
, "Parent node should be preserved")
&& Contract.assertIdentical(owner , Contract.<Node>oldValue(owner)
, "Owner document should be preserved")
&& Contract.assertIdentical(Contract.<Node>result(), n
, "[N4]" + " The node added should be returned");
if(n instanceof DocumentFragment)
{
List<Node> appendedNodes = Contract.<List<Node>>oldValue(toList(n.getChildNodes()));
for(int i = 0; i < appendedNodes.size(); i++)
common &= Contract.assertTrue(target.isSameNode(appendedNodes.get(i).getParentNode())
, "[N3']" + " This node should become parent for all appended nodes");
return common
&& Contract.assertIdentical(n.getChildNodes().getLength(), 0
, "The new contents of the document fragment should be empty")
&& Contract.assertEquals(Utils.getPostfix(children, appendedNodes.size()), appendedNodes
, "[N3]" + " The entire contents of the document fragment should be moved in the end of the child list of this nod
&& Contract.assertEquals(Utils.getPrefix(children, appendedNodes.size()), Contract.<Object>oldValue(((ArrayList<Node>)c
, "All other children should be preserved");
}
else
{
common &=
Contract.assertTrue(target.isSameNode(n.getParentNode())
, "[N1']" + " This node should become appended node parent")
&& Contract.assertIdentical(Utils.getLast(children), n
, "[N1]" + " The node added should be appended to the list of children");
Node oldParent = Contract.<Node>oldValue(n.getParentNode());
if(target.isSameNode(oldParent))
{
int ind = Contract.<Integer>oldValue(children.indexOf(n));
if(ind == 0)
return common
&& Contract.assertEquals(children.subList(0, children.size()-1), Contract.<Object>oldValue(((List<Node>)((ArrayList
, "All other children should be preserved");
else if(ind == Contract.<Integer>oldValue(children.size()-1))
return common
&& Contract.assertEquals(children, Contract.<Object>oldValue(((ArrayList<Node>)children).clone())
, "All other children should be preserved");
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)
domtest


Java - domtest/src/domtest/DOMTest.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
Outline
indexIterator() : Iterat
testAppend(Node, Node
DOMTest.java
NodeContract.java
@State
public TreeState constructTree()
{
return new TreeState(nodes);
}
@Test
@DataProvider(name = "nodes")
public void testAppendInMain(Node n)
{
logger.logInfo("Appending node " + index(n) + " to 0");
try
{
mainDoc.appendChild(n);
}
catch(Throwable e)
{
logger.logInfo("Exception caught " + e.getClass().getName() + ": " + e.getMessage());
}
}
@Test
@DataProvider(name = "indexIterator")
public void testRemove(int i)
{
Node n = mainDoc.getChildNodes().item(i);
logger.logInfo("Removing node " + i + " from 0");
try
{
mainDoc.removeChild(n);
}
catch(Throwable e)
{
logger.logInfo("Exception caught " + e.getClass().getName() + ": " + e.getMessage());
e.printStackTrace(System.out);
}
}
public int index(Node n)
public Iterator<Integer> indexIterator()
{
int n = numberOfMainDocChildren();
int[] indices = new int[n];
for(int i = 0; i < n; i++) indices[i] = i;
return (Utils.ArrayToTypedList(indices)).iterator();
}
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)
Writable
Smart Insert
153 : 1


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

Conclusion

- There is a lot of features to implement yet
- But many are implemented with low effort
- (Almost) All external tools and libraries are open source (exception – clover is taken for its good integration in Eclipse)

- Dependency injection and aspects allow non-invasive composition of models

Thank you!
Questions?

Victor Kuliamin

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

Institute for System Programming,
Software Engineering Department

www.unitesk.com
petrenko@ispras.ru