CASE STUDIES OF SUMMER MODEL-BASED TESTING FRAMEWORK

V. Kuliamin, N. Pakulin, A. Tugaenko

Institute for System Programming, Russian Academy of Sciences

OUTLINE

- Description of the tool
  - Intended domain
  - Main features
  - Similarities and differences with other tools
- Case studies
  - SMPT protocol
  - Part of DOM API
  - Part of Google Web UI

THE TOOL

- Summer
  Test development and execution framework
- Features
  - Java API testing (and where possible to link up)
  - xUnit-like test presentation
  - Unit – Component levels
  - Black-box testing
  - MBT features
    - EFSM-based testing
    - Software contracts as test oracles

VERY SIMPLE TEST EXAMPLE (JUST XUNIT)

```
@Test
public class TestClass {

  Account target = new Account(); // object under test

  @Test
  public void testDeposit() {
    int oldBalance = target.getBalance();
    int deposited  = 5;

    tagret.deposit(deposited);
    assertEquals(target.getBalance(), oldBalance + deposited
                 , "Balance should grow by deposited sum");
  }
}
```

ELABORATED TEST EXAMPLE

```
@Test
public class TestClass {

  Account target = new Account();

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

  int[] sums = new int[]{0, 1, 2, 3, 5, 17, 238};
  public boolean bound() {  return balance() < 350; }

  @Test
  @DataProvider(name = "sums")
  @Guard(names = "bound")
  public void testDeposit(int sum) { … target.deposit(sum); … }

  @Test
  @DataProvider(name = "sums")
  public void testWithdraw(int sum) { … target.withdraw(sum); … }
}
```

SIMILAR TOOLS AND ADDITIONS

- TestNG (one of most powerful xUnit tools, Java)
  - Testware hierarchy : test suites – tests – test classes – test methods
  - Setup-teardown methods on all hierarchy levels
  - Test methods grouping and selection by groups
  - Test methods sequencing
  - Test data providers
- NModel (MBT tool extending xUnit, C#)
  - State-based testing
  - Guard conditions
  - Compositions of test classes
- Additions
  - Stateful software contracts (described separately of tests)
  - Aspect-based linking of external components : contracts, coverage models, etc.
  - Combinations of data providers for parameters
  - More flexible data providers, guard conditions, state definitions

CONTRACT EXAMPLE

```
public class AccountContract {
  int balance;
  int maxCredit;

  public boolean withdrawPost(int sum) {
    if (Contract.oldValue(balance) - sum > maxCredit)
      return  assertEquals(Contract.intResult(), sum
                  , "Result should be equal to the argument")
           && assertEquals(balance, Contract.oldValue(balance) - sum
                  , "Balance should be increased on the argument");
    else
      return  assertEquals(Contract.intResult(), 0
                  , "Result should be 0")
           && assertEquals(balance, Contract.oldValue(balance)
                  , "Balance should not change");
  }
}
```

ASPECT-BASED LINKING

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns= ... >
  ...
  <bean id="accountContract" class="mbtest.tests.account.AccountContract">
    <property name="checkedObject" ref="accountImpl"/>
  </bean>

  <bean id="accountContractExecutor" class="mbtest.contracts.ContractExecutor">
    <property name="postcondition"
              value="mbtest.tests.account.AccountContract.withdrawPost(int)"/>
    <property name="updater"
              value="mbtest.tests.account.AccountContract.transferUpdate"/>
    <property name="contract" ref="accountContract"/>
  </bean>

  <aop:config>
    <aop:aspect id="accountContractAspect" ref="accountContractExecutor">
      <aop:pointcut id="accoutTransfer"
           expression="execution(* mbtest.tests.account.Account.withdraw(..))"/>
      <aop:around pointcut-ref="accoutTransfer" method="execute"/>
    </aop:aspect>
  </aop:config>
</beans>
```

CASE STUDIES

- Testing
  - SMTP protocol implementations (against SMPT RFC)
  - Part of Xerces DOM implementation (against DOM API standard)
  - Pat of Google WebUI (against simple intuition)
- Main goal : to check the following ideas
  - Idea 1
    Flexibility of component architecture facilitates usage of generic tools in various domains
  - Idea 2
    Modular testware (separate components : test oracles, test sequence generator, test data generators, test coverage measurement) helps to achieve comprehensive testing with less effort

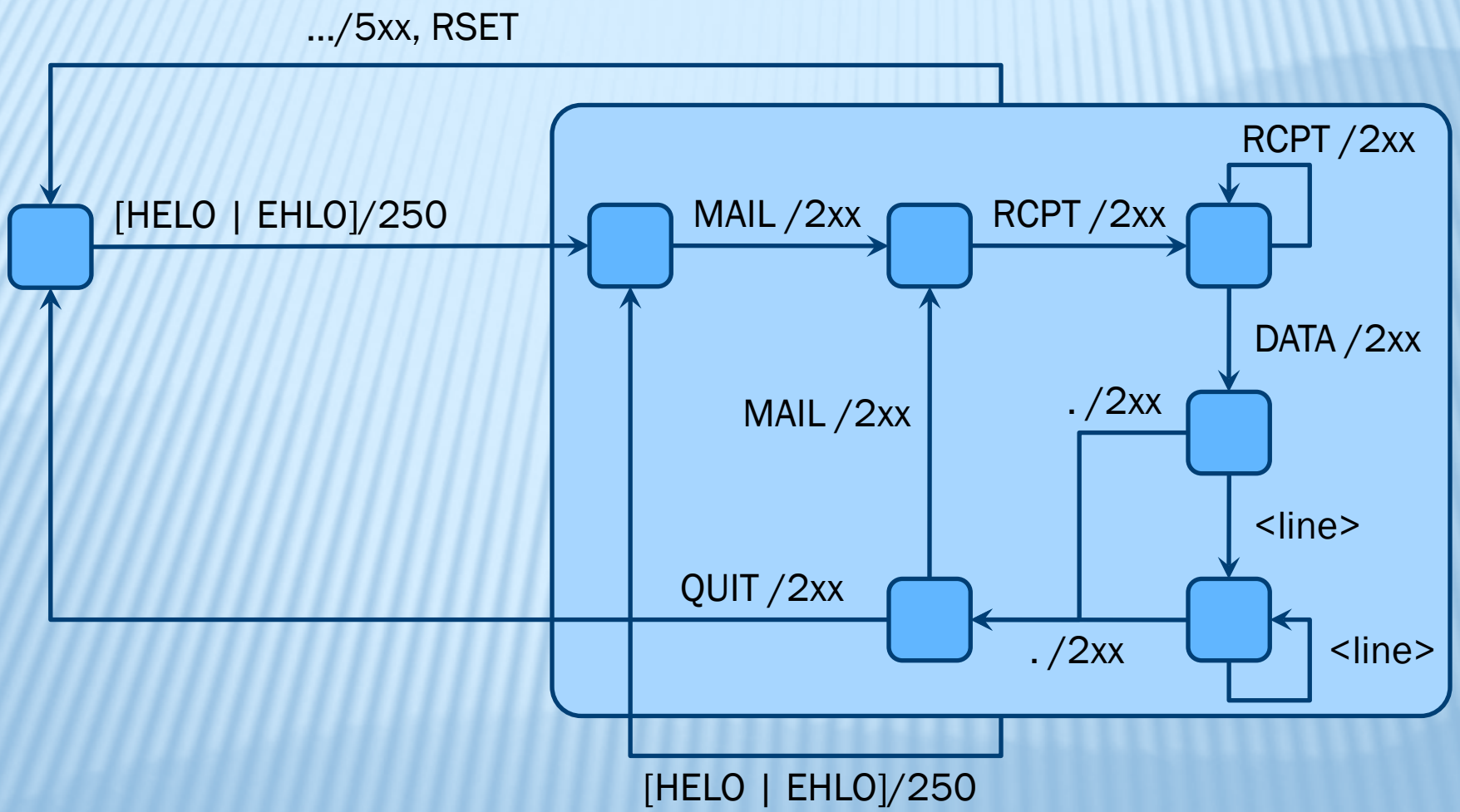
SMPT CASE STUDY

- Simple Mail Transfer Protocol
  RFC 5321 [2008]
- Client
  - Basic actions:
    <connect>
    [HELO | EHLO] …
    (MAIL FROM: <…> (RCPT TO: <…>)+ DATA (<line>)* .)+
    QUIT
  - Additional : NOOP, RSET { VRFY, EXPN, HELP }
- Server responses : [2-5][0-2|5][0-9] …
- Extensions
  RFC 4954 (AUTH) { RFC 1652, 1879, 2034, 2920, 3030, 3207, 3461, 3463, 3865, 3885, 4095, 4405, 4865, 4954, 5336 }

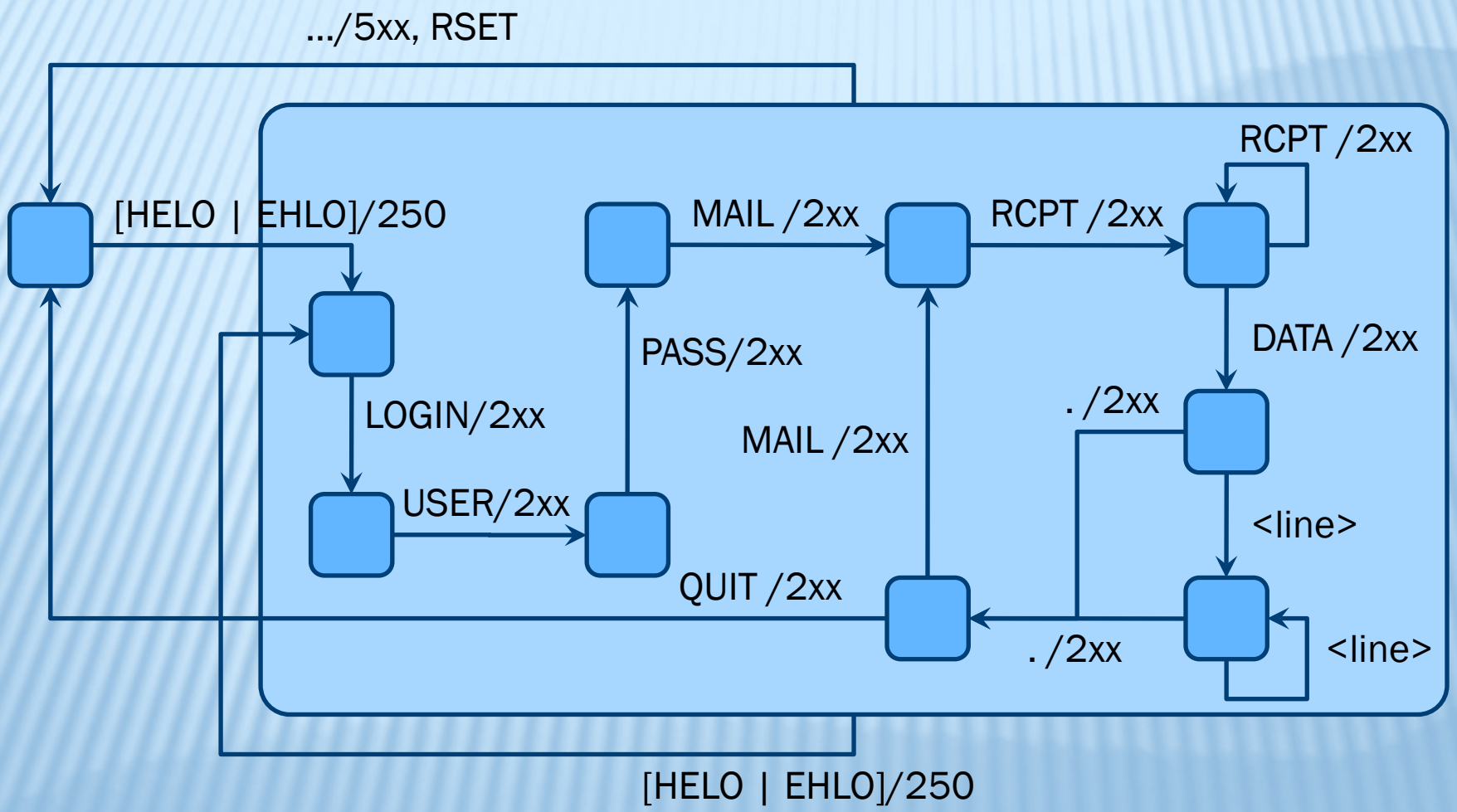
MODULAR TEST MODEL

- Separate CONNECT-DISCONNECT test model
  - Can be used with other over-transport protocols
- Separate basic SMTP test model
- Separate AUTH PLAIN test model
- Possibility to add other extensions

TEST STATE MACHINE (SLIGHTLY SIMPLIFIED)



TEST STATE MACHINE WITH AUTH PLAIN



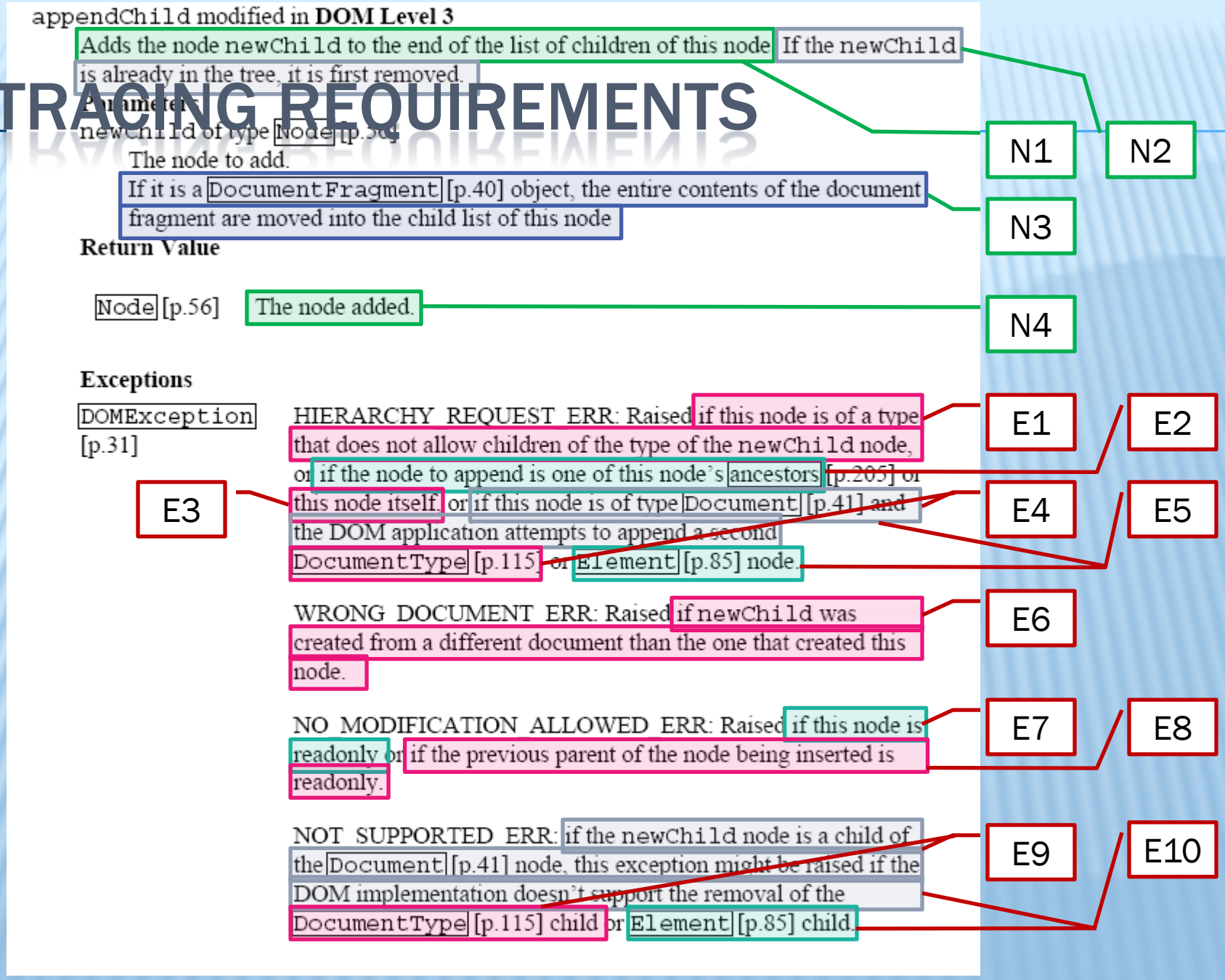
SMTP CASE STUDY STATISTICS (LOC)

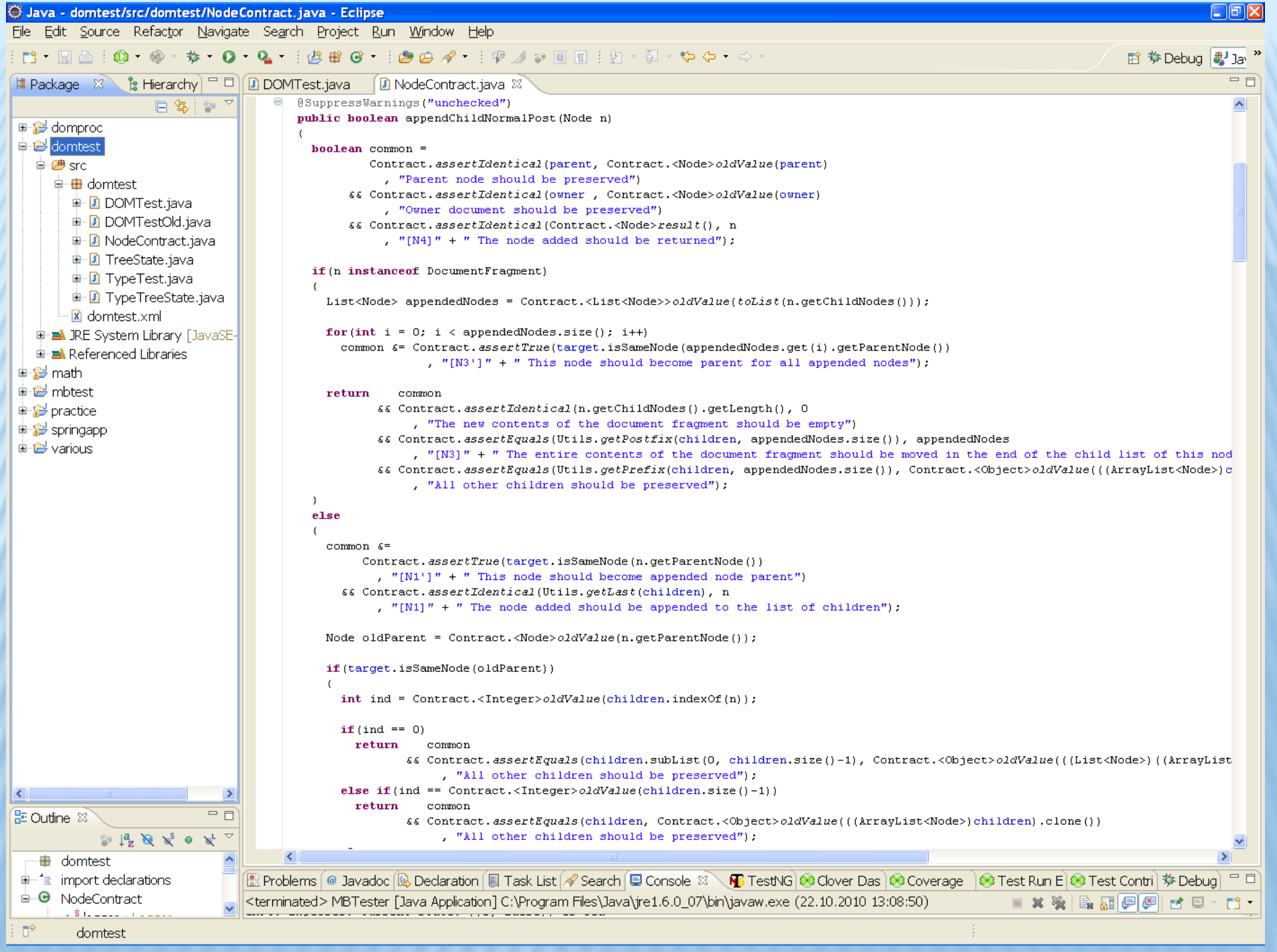
Testware module	Test model	Contract	Other	Total
Connect-Disconnect	90	140		230
Basic SMTP	200	480		680
Authentication	140	300		440
Auxiliary			680	680
Configuration			230	230
Total	430	920	910	2260

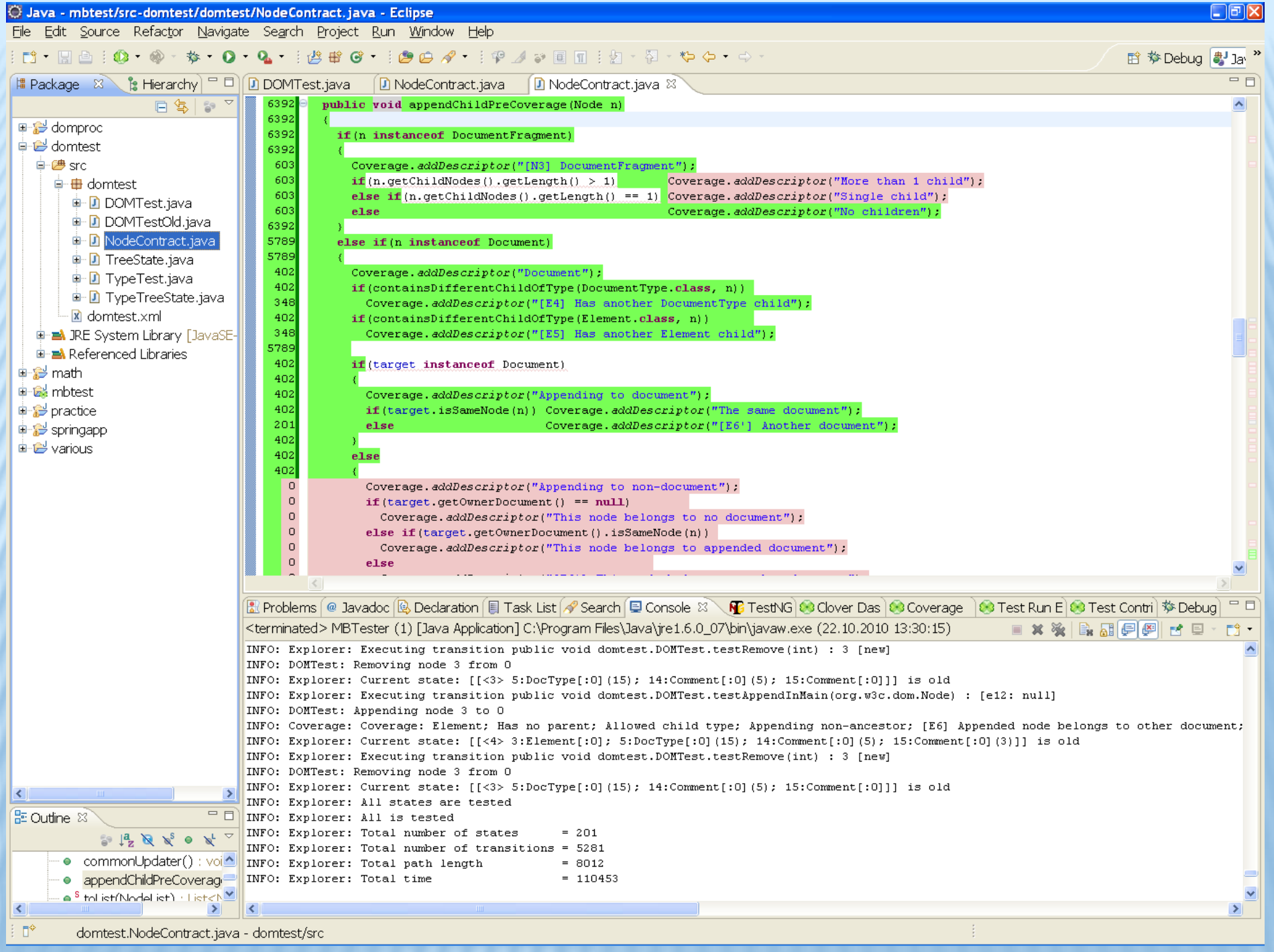
DOM CASE STUDY

- DOM API Standard
  - Document Object Model – internal representation of web pages in browsers
- Node interface
- SUT – Xerces for Java [xerces.apache.org]

TRACING REQUIREMENTS

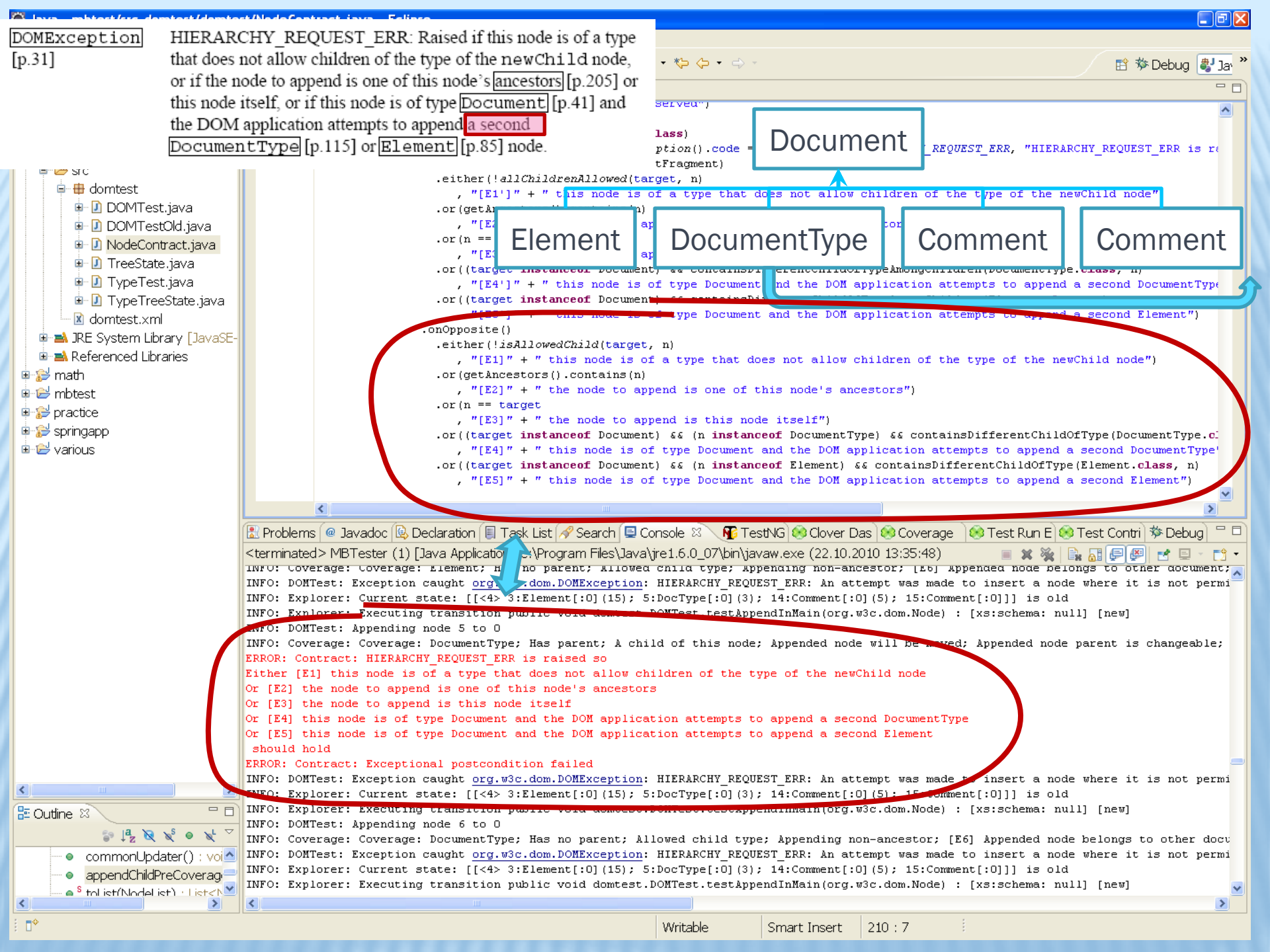



Java - domtest/src/domtest/NodeContract.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Hierarchy
domproc
domtest
src
domtest
DOMTest.java
DOMTestOld.java
NodeContract.java
TreeState.java
TypeTest.java
TypeTreeState.java
domtest.xml
JRE System Library [JavaSE-
Referenced Libraries
math
mbtest
practice
springapp
various
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");
Outline
domtest
import declarations
NodeContract
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 - 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
Outline
commonUpdater() : voi
appendChildPreCoverag
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
domtest.NodeContract.java - domtest/src

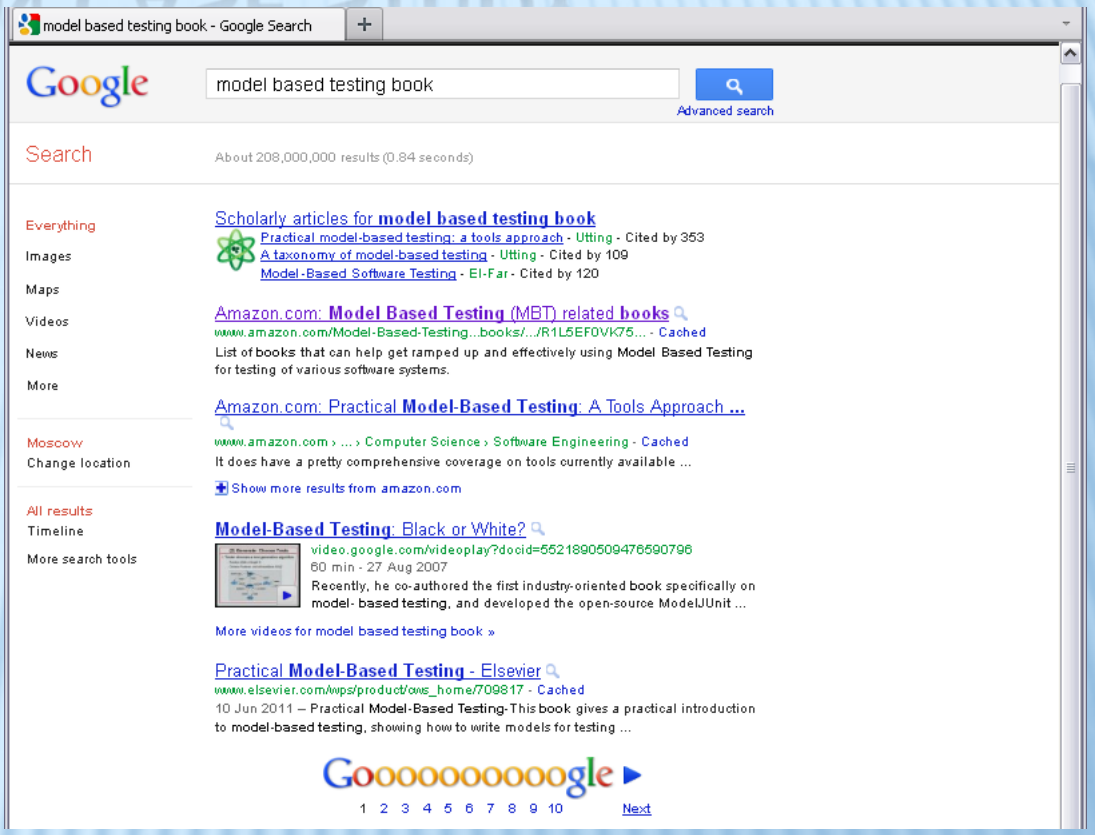
DOMException [p.31]

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.



WEB APPLICATION CASE STUDY

- Google Web UI



- No ready Java API so, use WebUI Driver
  - Selenium RC
  - http://seleniumhq.org/

TEST EXCERPT

```
@Test public class GoogleTest {
  Selenium  sdriver;
  int       page  = 0;
  @State public int getPage() { return page; }

  @BeforeTest public void init() {
    sdriver = new DefaultSelenium("localhost", 4444, "*firefox", "http://www.google.com");
    sdriver.start();
    sdriver.open("/");
  }

  boolean containsTransitionTo(int n) { return sdriver.isElementPresent("link=" + n); }

  @Test @DataProvider(name = "pageNumber")
  @Guard(names = {"isResultsPage", "containsTransitionTo"})
  public void moveToPage(int n) {
    sdriver.click("link=" + n);
    if(n != 1) sdriver.waitForCondition("selenium.isTextPresent(
                                        'Page " + n + " of about results')==true", "20000");
    else       sdriver.waitForCondition("selenium.isTextPresent(
                                        'Page " + page + " of about results')!=true", "20000");
    page = n;
    commonCheck();
  }
}
```

CONCLUSION

- Idea 1
  Flexibility of component architecture facilitates usage of generic tools in various domains
  - Seems to be true : API components, protocols, Web UI can be tested by uniform mechanisms
- Idea 2
  Modular testware helps to achieve comprehensive testing with less effort
  - Not enough data, maybe wrong
  - Note : tests' maintainability requires additional effort, but modular tests are more maintainable by construction
  - So, such an idea seems to be formulated inadequately

Thank you for attention!