

# 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
  - Relations with other tools
- Case studies
  - SMTP protocol
  - Part of DOM API
  - Part of Google Web UI

# THE TOOL

---

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

# 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); ... }
```

```
}
```

# RELATED TOOLS AND ADDITIONS

---

- ❑ TestNG (one of most powerful xUnit tools, Java, Cedric Beust, 2004)
  - 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#, Microsoft Research, 2007)
  - State-based testing
  - Guard conditions
- ❑ 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.<Integer>oldValue(balance) - sum > maxCredit)
            return assertEquals(Contract.<Integer>result(), sum
                , "Result should be equal to the argument")
                && assertEquals(balance, Contract.<Integer>oldValue(balance) - sum
                    , "Balance should be increased on the argument");
        else
            return assertEquals(Contract.<Integer>result(), 0
                , "Result should be 0")
                && assertEquals(balance, Contract.<Integer>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

---

- ❑ SMTP protocol implementations (against SMTP RFC)
- ❑ Part of Xerces DOM implementation (against DOM API standard)
- ❑ Part of Google WebUI (against simple intuitive constraints)
- ❑ We wanted to check that
  - Flexibility of component architecture facilitates usage of generic tools for various systems under test
  - Modular testware (separate components : test oracles, test sequence generator, test data generators, test coverage measurement) helps to achieve good tests maintainability and extensibility



# SMTP 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 }

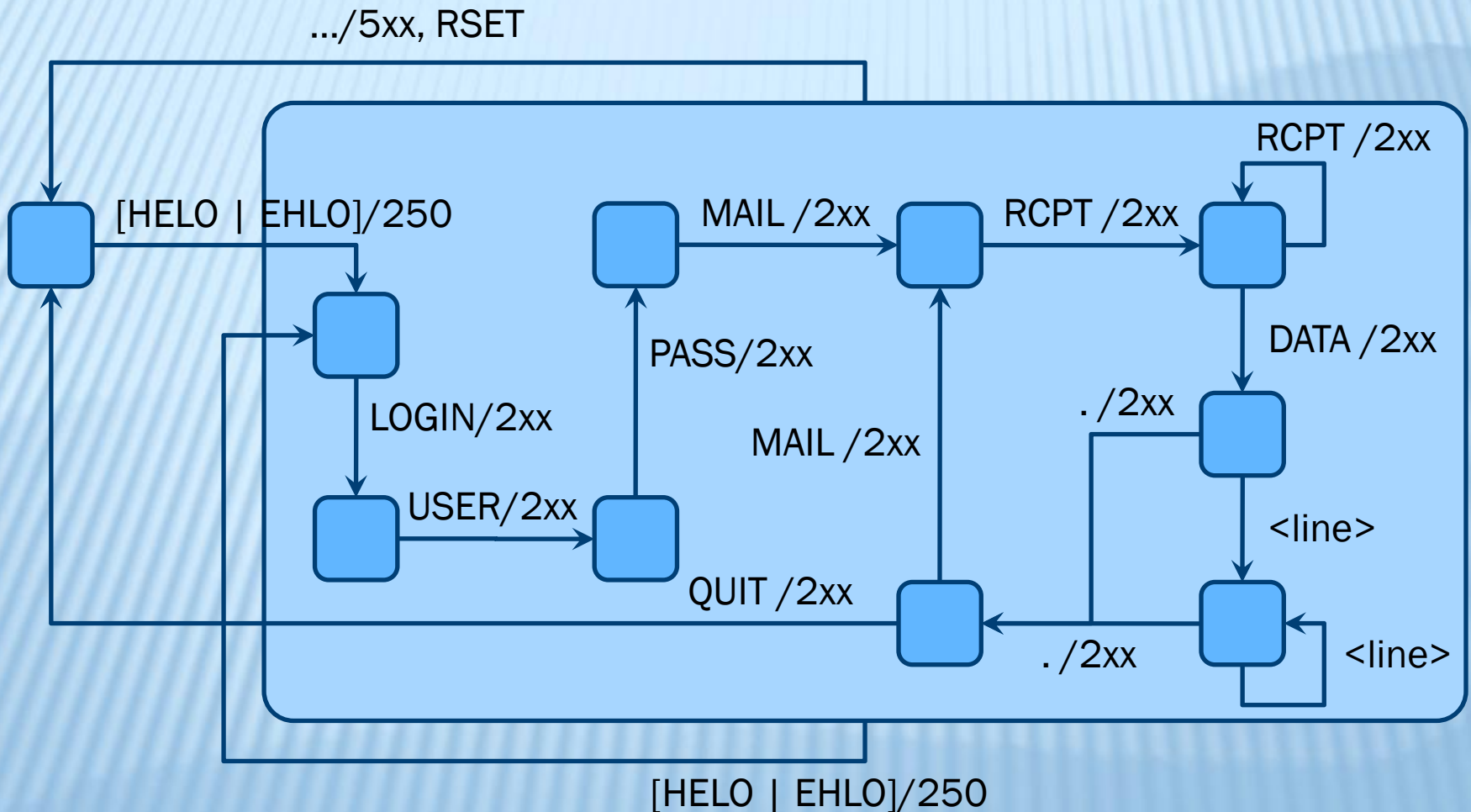
# MODULAR TEST MODEL

---

- ❑ CONNECT-DISCONNECT test model
  - Can be used with other over-transport protocols
- ❑ Basic SMTP test model
- ❑ AUTH PLAIN test model

Possibility to add other extensions

# 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](http://xerces.apache.org)]

### 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.

Parameter  
`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.

# TRACING REQUIREMENTS

N1

N2

N3

N4

E1

E2

E3

E4

E5

E6

E7

E8

E9

E10

DOMException [p.31]

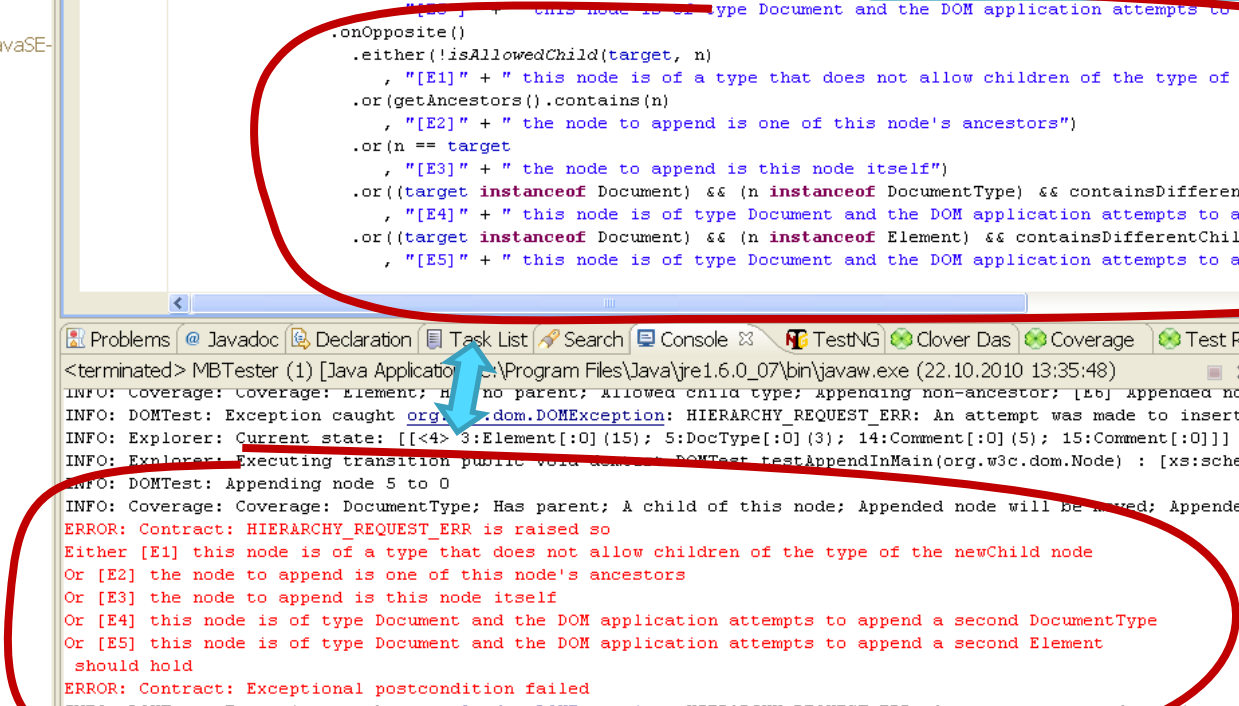
HIERARCHY\_REQUEST\_ERR: Raised if this node is of a type that does not allow children of the type of the newChild node, or if the node to append is one of this node's ancestors [p.205] or this node itself, or if this node is of type Document [p.41] and the DOM application attempts to append a second DocumentType [p.115] or Element [p.85] node.

- src
  - domtest
    - DOMTest.java
    - DOMTestOld.java
    - NodeContract.java
    - TreeState.java
    - TypeTest.java
    - TypeTreeState.java
  - domtest.xml
- JRE System Library [JavaSE-6]
  - Referenced Libraries
- math
- mbtest
- practice
- springapp
- various

```

class Document {
    ...
    .either(!allChildrenAllowed(target, n)
        , "[E1]" + " this node is of a type that does not allow children of the type of the newChild node"
    ).or(getAncestorOf(n) != null
        , "[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")
    ...
}

```



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: Coverage: Coverage: Element; Has no parent; Allowed child type; Appending non-ancestor; [E6] Appended node belongs to other document;
INFO: DOMTest: Exception caught org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted
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]
INFO: DOMTest: Appending node 5 to 0
INFO: Coverage: Coverage: DocumentType; Has parent; A child of this node; Appended node will be allowed; 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 precondition failed
INFO: DOMTest: Exception caught org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted
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]
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 document;
INFO: DOMTest: Exception caught org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted
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

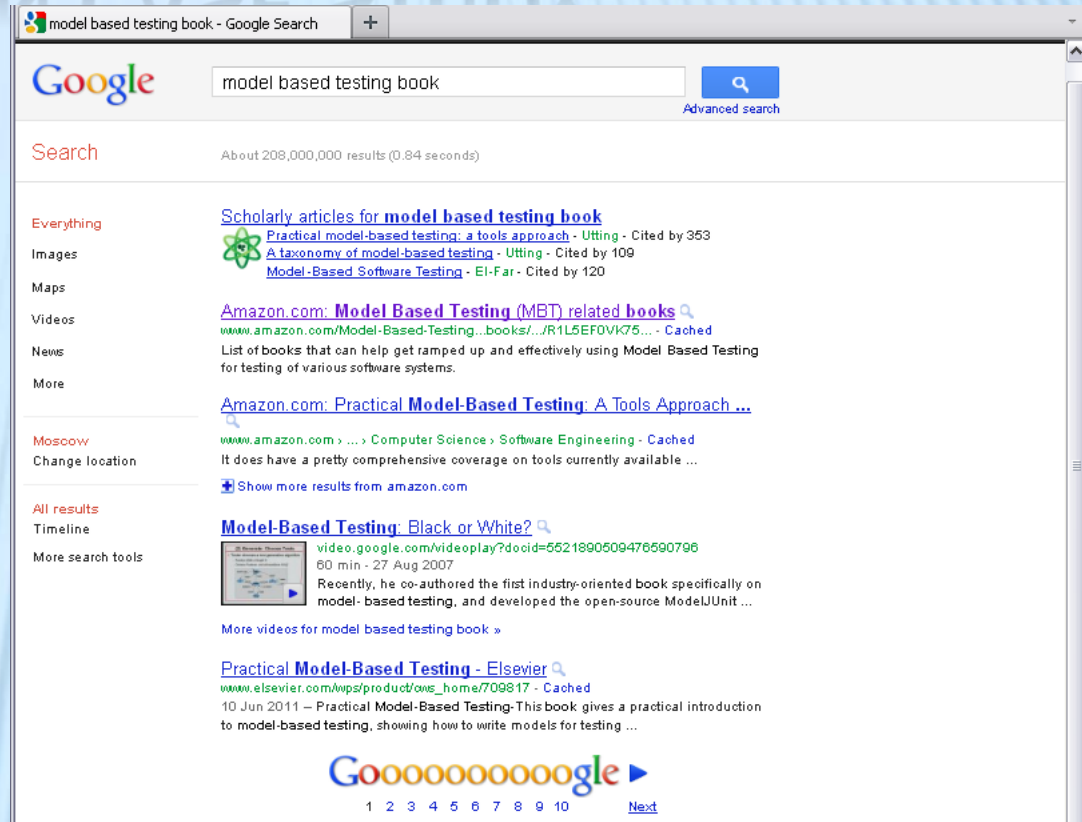
- commonUpdater() : void
- appendChildPreCoverage() : void
- testAppendInMain(Node) : void

# WEB APPLICATION CASE STUDY

□ Google Web UI

□ No ready Java API  
so, use  
WebUI Driver

- Selenium RC
- <http://seleniumhq.org/>





# CONCLUSION

---

- ❑ Flexibility of component architecture facilitates usage of generic tools for various SUTs
  - Seems to be true : API components, protocols, Web UI can be tested by uniform mechanisms
- ❑ Modular testware helps to achieve tests maintainability and extensibility
  - By construction?
  - Experiments give empiric evidence

# Thank you for attention!

## Questions?

[kuliamin@ispras.ru](mailto:kuliamin@ispras.ru)

<http://www.ispras.ru/~kuliamin>

<http://www.ispras.ru/en/se/index.php>

ISP RAS Software Engineering Department