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
  - SMPT 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
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
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");
    }
}
<?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="accountTransfer" expression="execution(* mbtest.tests.account.Account.withdraw(..))"/>
      <aop:around pointcut-ref="accountTransfer" method="execute"/>
   </aop:aspect>
</aop:config>
</beans>
CASE STUDIES

- SMTP protocol implementations (against SMPT 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
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
## SMTP CASE STUDY STATISTICS (LOC)

<table>
<thead>
<tr>
<th>Testware module</th>
<th>Test model</th>
<th>Contract</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect-Disconnect</td>
<td>90</td>
<td>140</td>
<td></td>
<td>230</td>
</tr>
<tr>
<td>Basic SMTP</td>
<td>200</td>
<td>480</td>
<td></td>
<td>680</td>
</tr>
<tr>
<td>Authentication</td>
<td>140</td>
<td>300</td>
<td></td>
<td>440</td>
</tr>
<tr>
<td>Auxiliary</td>
<td></td>
<td></td>
<td>680</td>
<td>680</td>
</tr>
<tr>
<td>Configuration</td>
<td></td>
<td></td>
<td>230</td>
<td>230</td>
</tr>
<tr>
<td>Total</td>
<td>430</td>
<td>920</td>
<td>910</td>
<td>2260</td>
</tr>
</tbody>
</table>
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]
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.

The node to add.

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

Return Value

Node 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 of this node itself or if this node is of type Document or DocumentType and the DOM application attempts to append a second DocumentType or Element 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 node, this exception might be raised if the DOM implementation does not support the removal of the DocumentType child or Element child.
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 or this node itself, or if this node is of type Document or the DOM application attempts to append a second DocumentType or Element node.

`Document` 

`Element` 

`DocumentType` 

`Comment` 

`Comment`
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@isprras.ru
http://www.isprras.ru/~kuliamin

ISP RAS Software Engineering Department