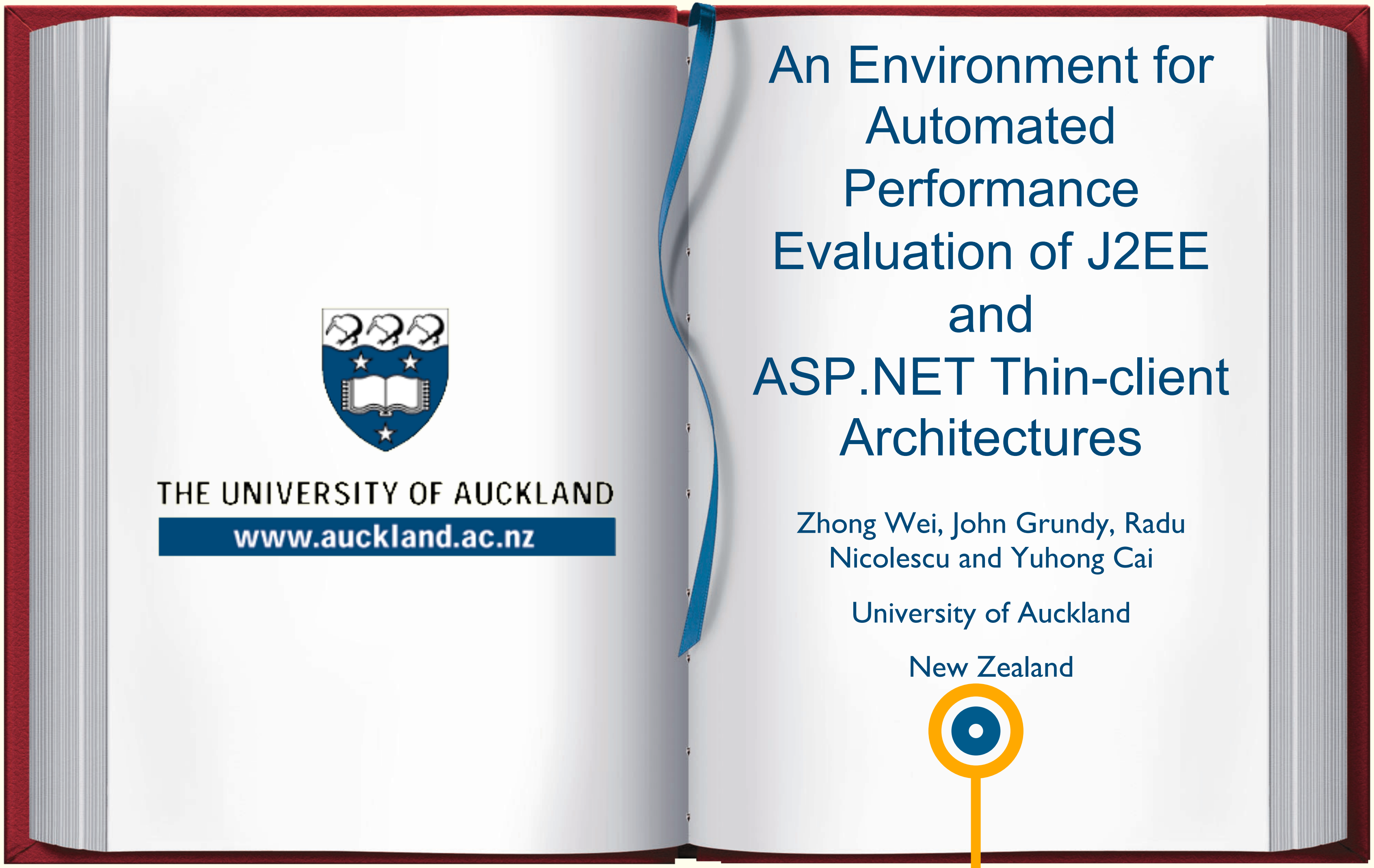
2004

SOFTWARE ENGINEERING

The University of Auckland New Zealand



THE UNIVERSITY OF AUCKLAND

www.auckland.ac.nz

An Environment for Automated Performance Evaluation of J2EE and ASP.NET Thin-client Architectures

Zhong Wei, John Grundy, Radu Nicolescu and Yuhong Cai

University of Auckland

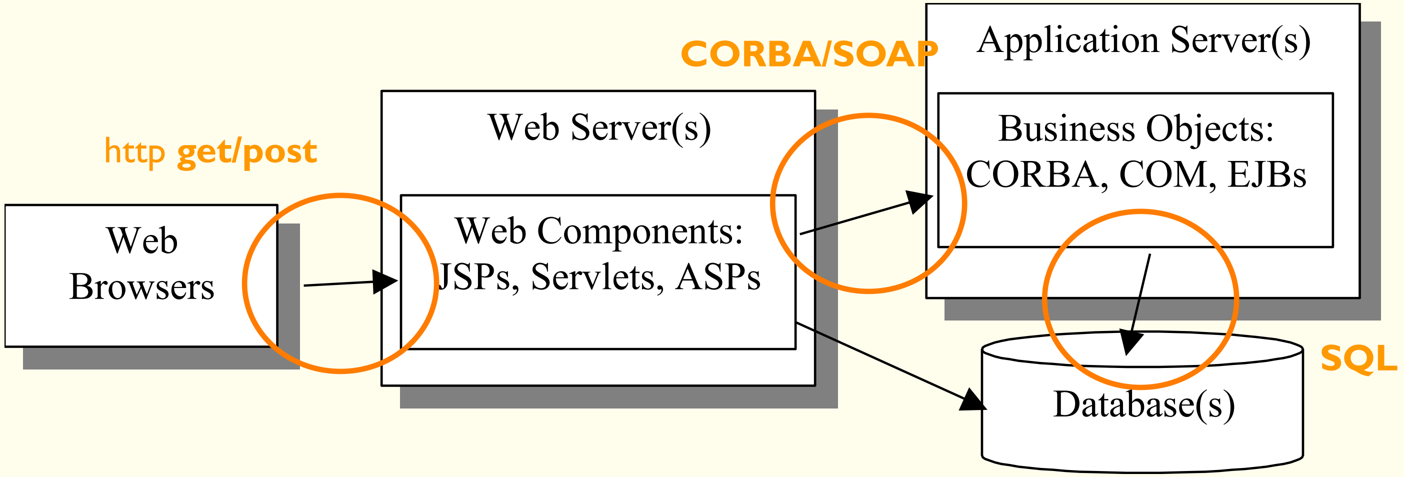
New Zealand

Outline

- Motivation
- Overview of our SoftArch/Thin approach
- Example usage of SoftArch/Thin for ASPs/JSPs
- Performance results examples
- Comments on results & experiences to date
- Conclusions and Future Research

Motivation

- Thin-client application architecture performance – critical for system success
- Very difficult in general to estimate/design for

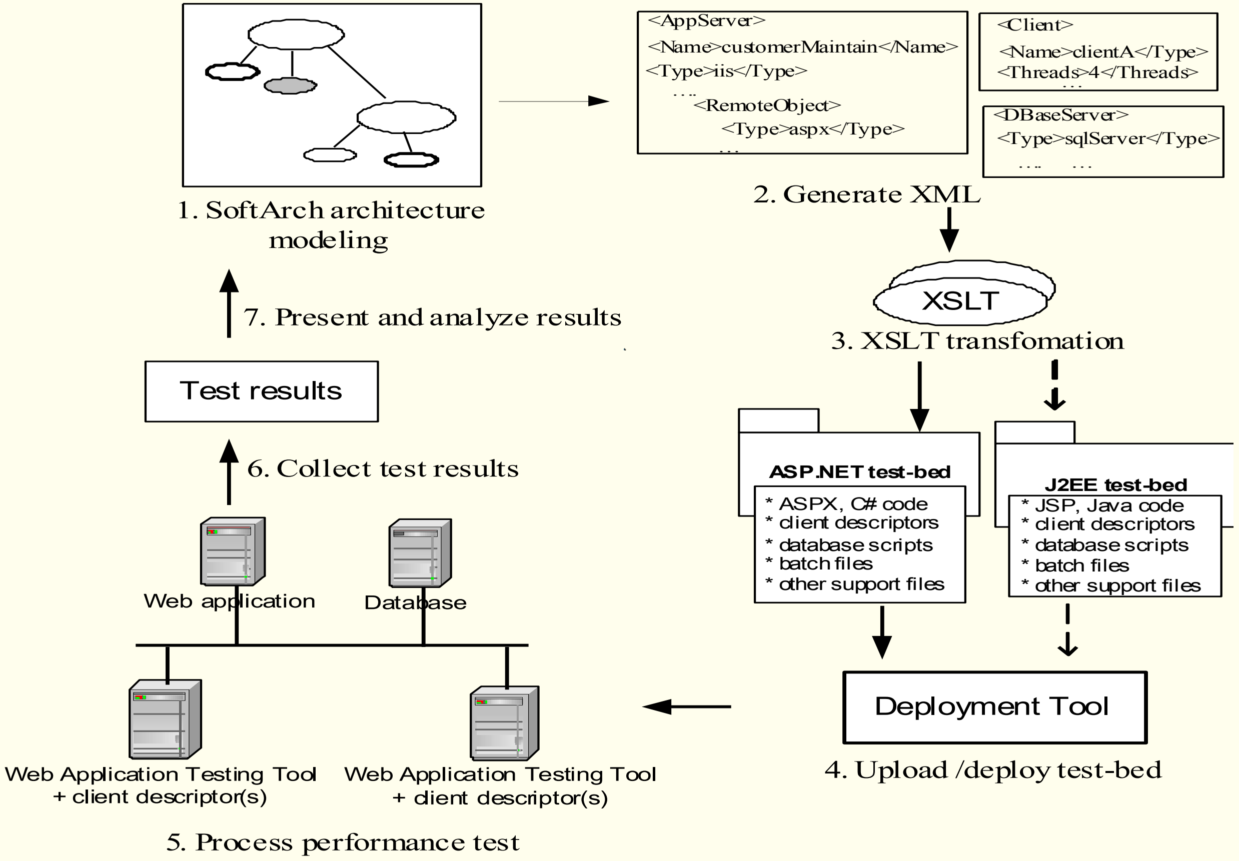


Motivation (cont.)

- Issues:
  - Response time, throughput, resource utilisation, …
  - Where is the time spent typically?
  - When designing system architecture, how meet performance non-functional requirements?

- Approaches:
  - Massively over-engineer
  - Simulation from architecture models
  - Rapid prototyping
  - Benchmarks and existing application profiling

Our Approach: Performance Test-bed Generation and Evaluation



1. SoftArch architecture modeling

```
<AppServer>
<Name>customerMaintain</Name>
<Type>iis</Type>
  ….
    <RemoteObject>
      <Type>aspx</Type>
      …
```

```
<Client>
 <Name>clientA</Type>
<Threads>4</Threads>
      …
```

```
<DBaseServer>
<Type>sqlServer</Type>
  ….   …
```

2. Generate XML

XSLT

3. XSLT transfomation

ASP.NET test-bed
- * ASPX, C# code
- * client descriptors
- * database scripts
- * batch files
- * other support files

J2EE test-bed
- * JSP, Java code
- * client descriptors
- * database scripts
- * batch files
- * other support files

Deployment Tool

4. Upload /deploy test-bed

Web application

Database

Web Application Testing Tool + client descriptor(s)

Web Application Testing Tool + dient descriptor(s)

5. Process performance test

6. Collect test results

Test results

7. Present and analyze results

Performance Test-bed Approach

- Model architecture at high level of abstraction, but include middleware/DB configuration etc
- Generate ASP, JSP, web component, DB table etc. from model – includes request/response calls between components
- Compile, deploy, run tests
- Present results to user for analysis
- Essentially automated form of rapid prototyping

Example usage of SoftArch/MTE: Modelling a thin-client architecture

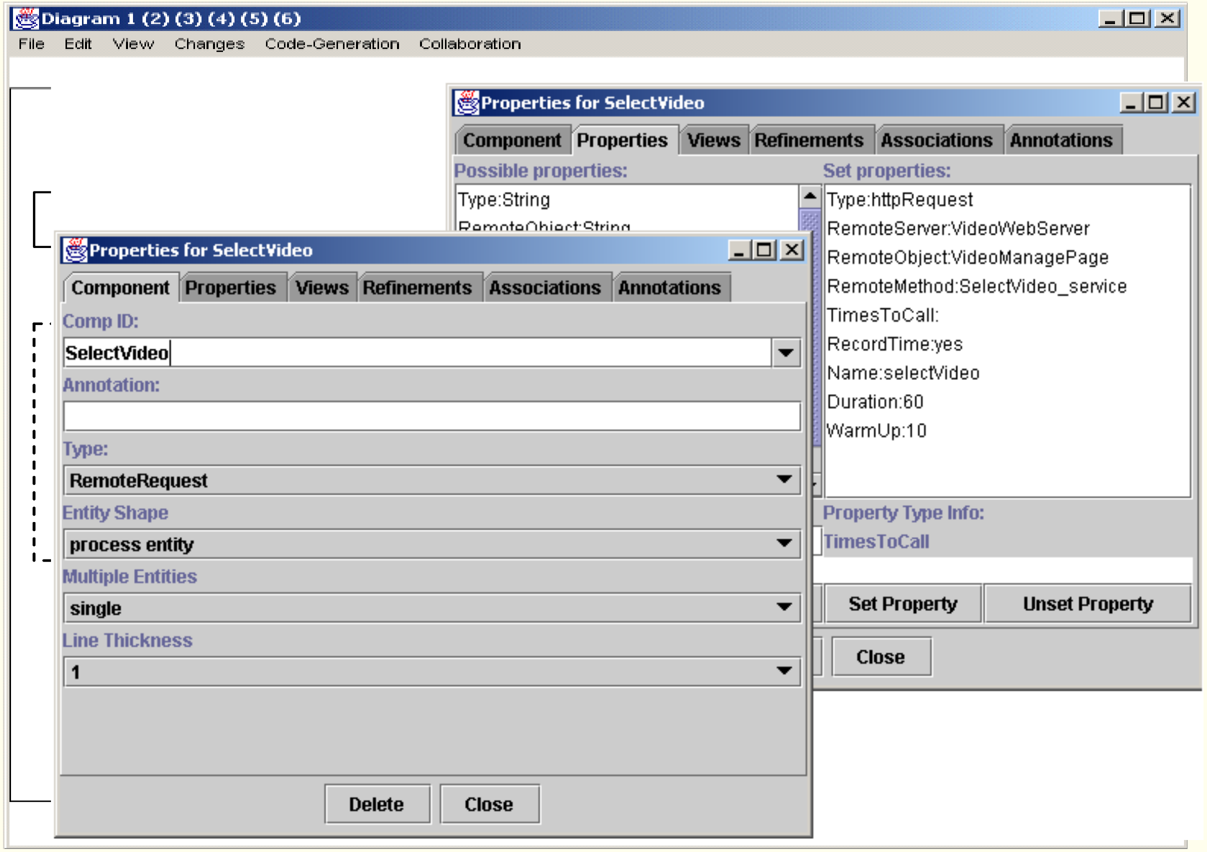


Diagram 1 (2) (3) (4) (5) (6)
File Edit View Changes Code-Generation Collaboration
Properties for SelectVideo
Component Properties Views Refinements Associations Annotations
Set Property
Unset Property
Close
Properties for SelectVideo
Component Properties Views Refinements Associations Annotations
Delete
Close

Possible properties:
Type:String
RemoteObject:String

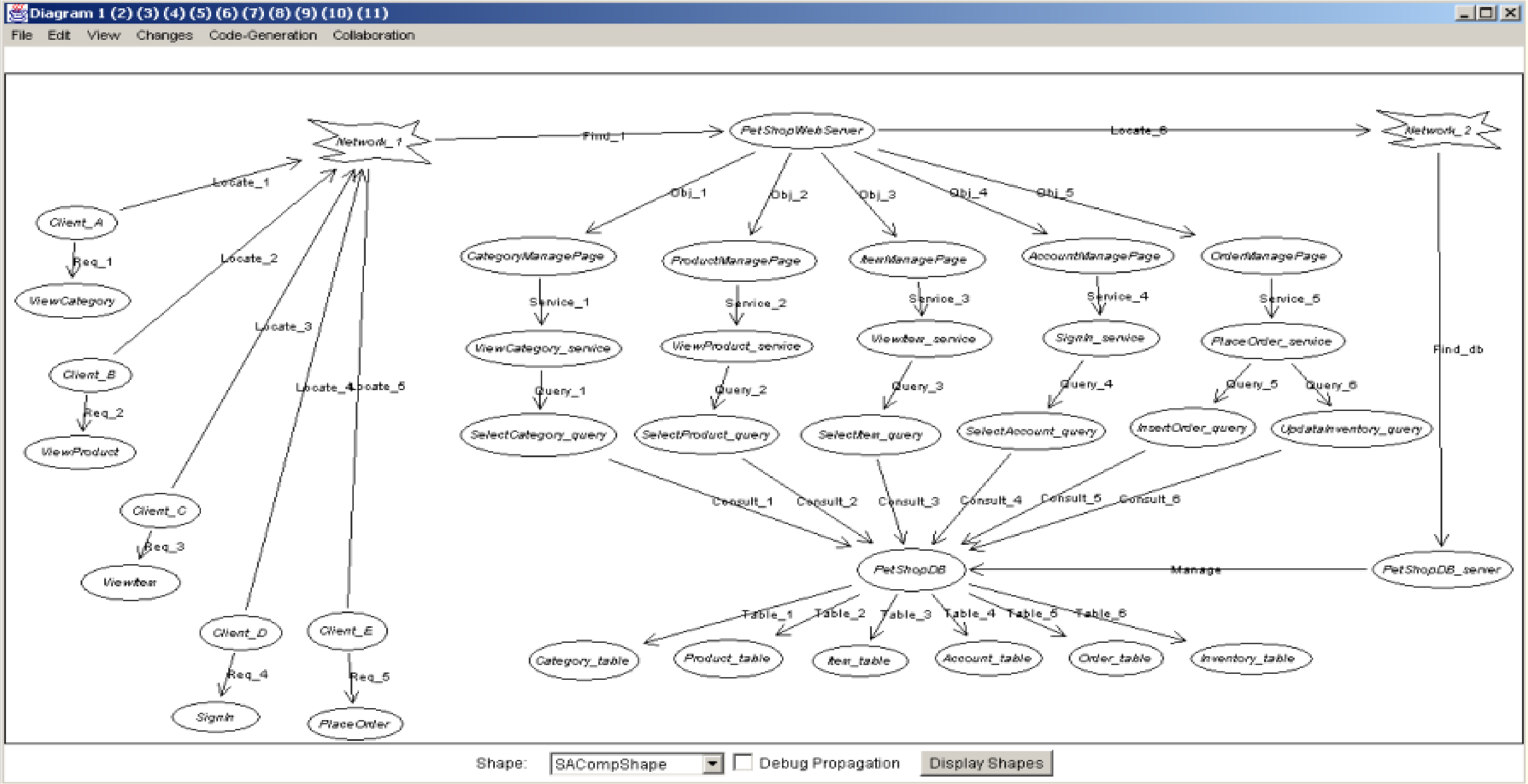
Set properties:
Type:httpRequest
RemoteServer:VideoWebServer
RemoteObject:VideoManagePage
RemoteMethod:SelectVideo_service
TimesToCall:
RecordTime:yes
Name:selectVideo
Duration:60
WarmUp:10

Property Type Info:
TimesToCall

Comp ID: SelectVideo
Annotation:
Type: RemoteRequest
Entity Shape: process entity
Multiple Entities: single
Line Thickness: 1

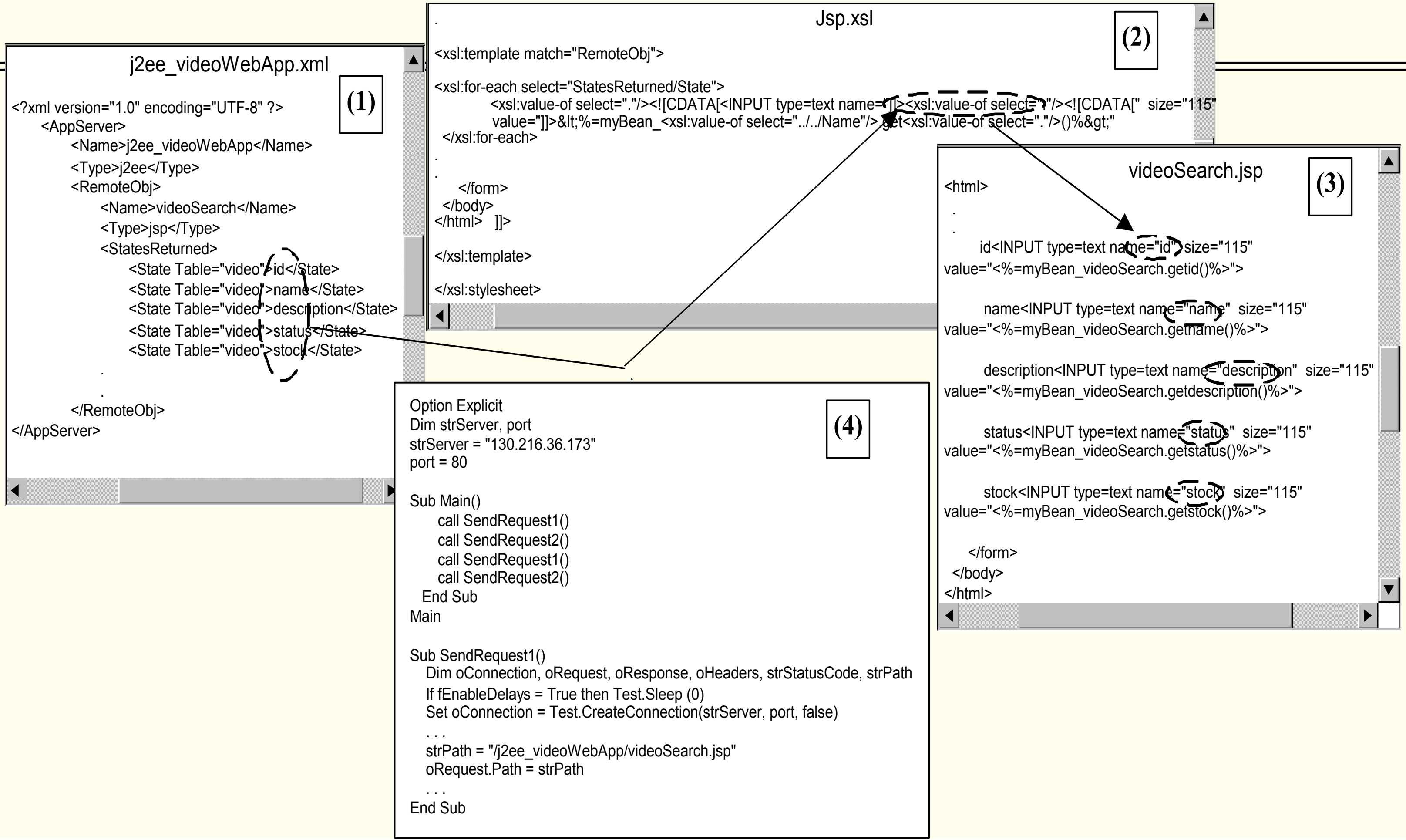
- Model clients – requests to servers
- Model servers, server objects, object services
- Model service requests to other services, databases
- Model database elements

Model example: PetShop J2EE Reference application



Shape: SACompShape ▾ □ Debug Propagation

Generating Test-bed Code



j2ee_videoWebApp.xml (1)

```
<?xml version="1.0" encoding="UTF-8" ?>
    <AppServer>
        <Name>j2ee_videoWebApp</Name>
        <Type>j2ee</Type>
        <RemoteObj>
            <Name>videoSearch</Name>
            <Type>jsp</Type>
            <StatesReturned>
                <State Table="video">id</State>
                <State Table="video">name</State>
                <State Table="video">description</State>
                <State Table="video">status</State>
                <State Table="video">stock</State>
                        .
                        .
        </RemoteObj>
</AppServer>
```

Jsp.xsl (2)

```
.
<xsl:template match="RemoteObj">

<xsl:for-each select="StatesReturned/State">
         <xsl:value-of select="."/><![CDATA[<INPUT type=text name="]]><xsl:value-of select="."/><![CDATA["  size="115"
         value="]]>&lt;%=myBean_<xsl:value-of select="../../Name"/>get<xsl:value-of select="."/>()%&gt;"
 </xsl:for-each>
.
.
    </form>
  </body>
</html>   ]]>

</xsl:template>

</xsl:stylesheet>
```

videoSearch.jsp (3)

```
<html>
  .
  .
      id<INPUT type=text name="id" size="115"
value="<%=myBean_videoSearch.getid()%>">

       name<INPUT type=text name="name"  size="115"
value="<%=myBean_videoSearch.getname()%>">

       description<INPUT type=text name="description"  size="115"
value="<%=myBean_videoSearch.getdescription()%>">

       status<INPUT type=text name="status"  size="115"
value="<%=myBean_videoSearch.getstatus()%>">

       stock<INPUT type=text name="stock"  size="115"
value="<%=myBean_videoSearch.getstock()%>">


    </form>
  </body>
</html>
```

(4)

```
Option Explicit
Dim strServer, port
strServer = "130.216.36.173"
port = 80

Sub Main()
     call SendRequest1()
     call SendRequest2()
     call SendRequest1()
     call SendRequest2()
  End Sub
Main

Sub SendRequest1()
   Dim oConnection, oRequest, oResponse, oHeaders, strStatusCode, strPath
   If fEnableDelays = True then Test.Sleep (0)
   Set oConnection = Test.CreateConnection(strServer, port, false)
   . . .
   strPath = "/j2ee_videoWebApp/videoSearch.jsp"
   oRequest.Path = strPath
   . . .
End Sub
```

Compile/Deploy/Run Tests

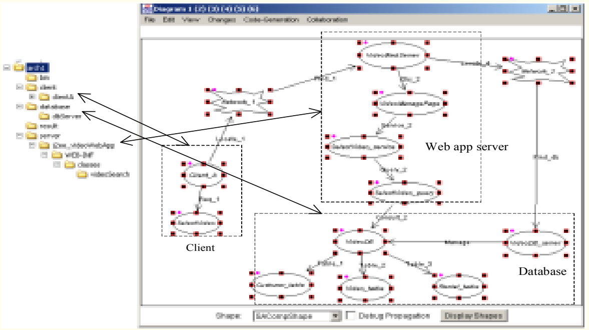
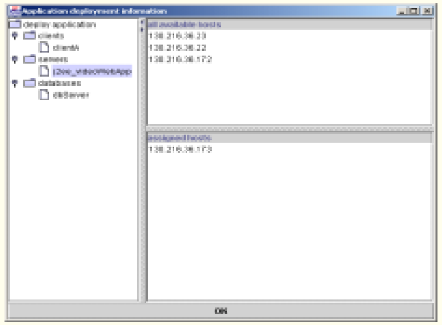


Diagram 1 (2) (3) (4) (5) (6)
File Edit View Changes Code-Generation Collaboration
Web app server
Client
Database
Shape:
Debug Propagation
Display Shapes

1. Generate .jsp/.asp, .java/.c, .bat, .sql, .war etc files



Application deployment information

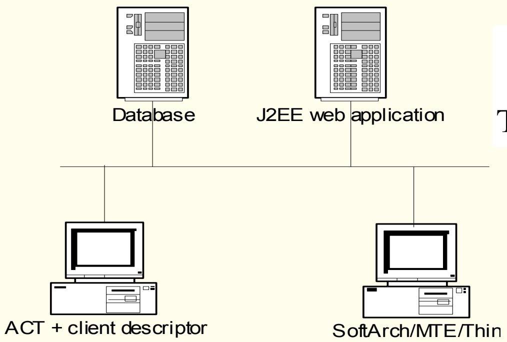
- deploy application
  - clients
    - clientA
  - servers
    - j2ee_videoWebApp
  - databases
    - dbServer

all available hosts
138.216.36.23
138.216.36.22
138.216.36.172

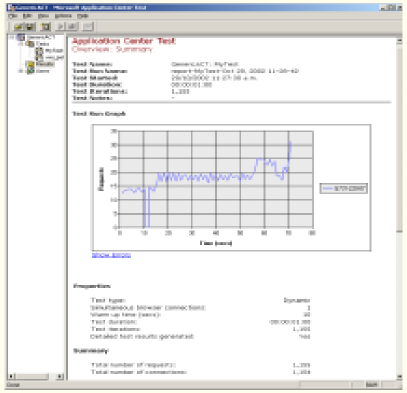
assigned hosts
138.216.36.173

OK

2. Deploy generated test-bed files to client, server host machines

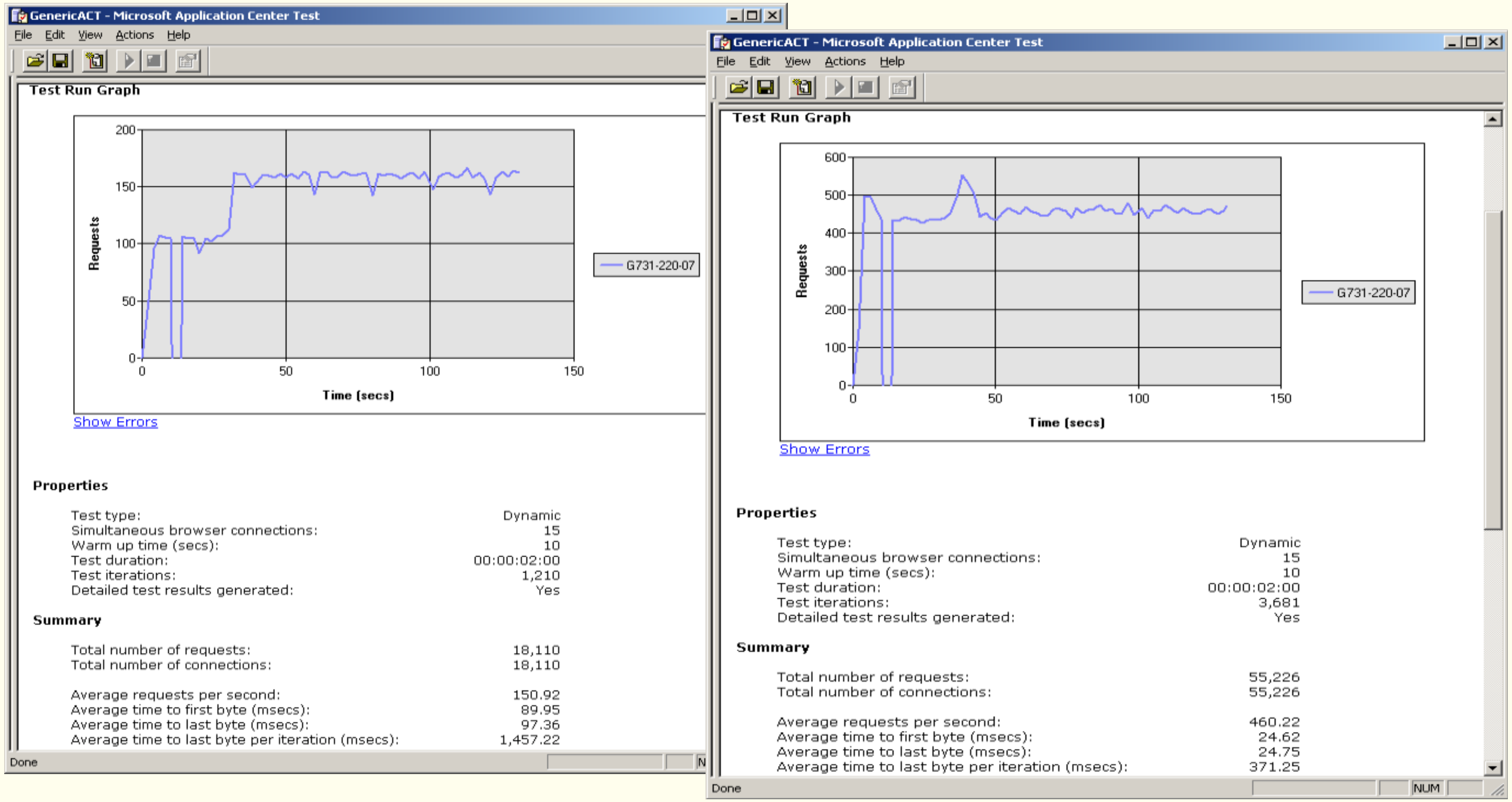


3. Instruct Application Centre Test to perform tests



4. Display and Analyse results

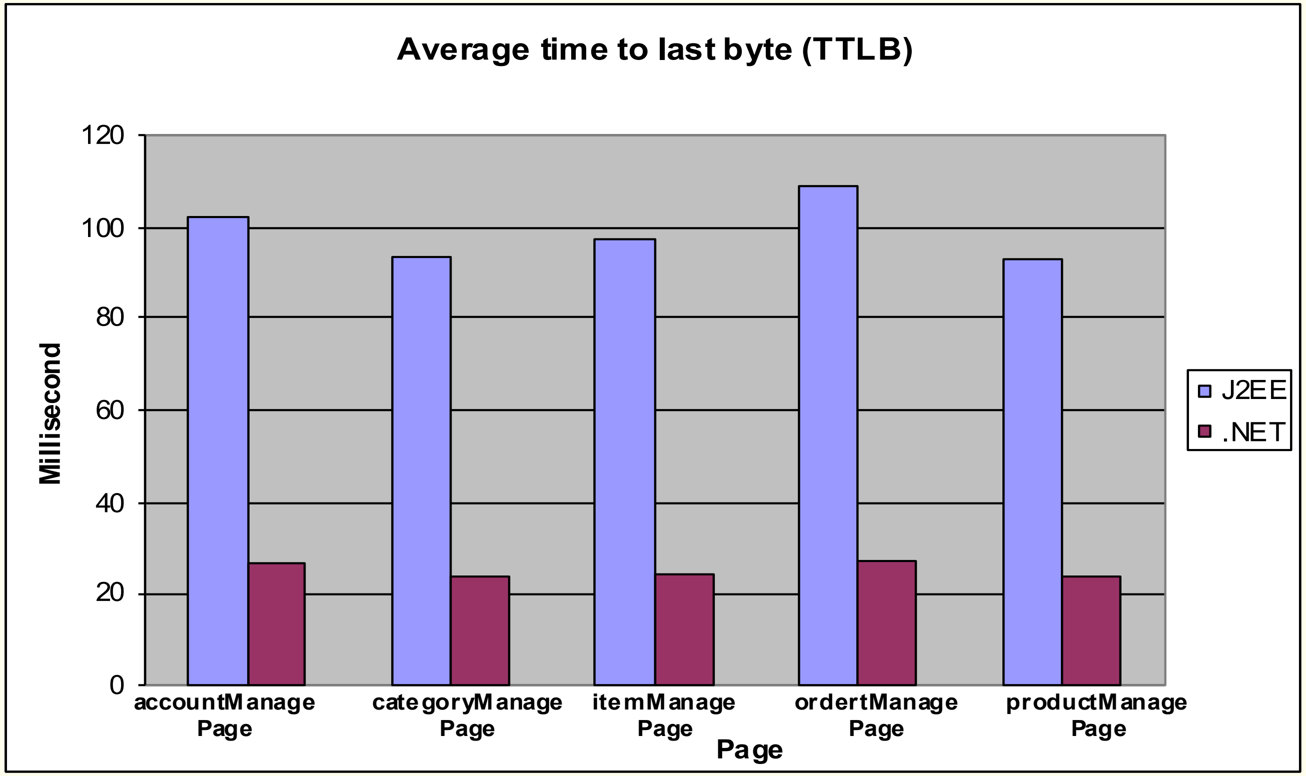
Examples of Test Results via ACT



```
Properties

    Test type:                                        Dynamic
    Simultaneous browser connections:                      15
    Warm up time (secs):                                   10
    Test duration:                                00:00:02:00
    Test iterations:                                    1,210
    Detailed test results generated:                      Yes

Summary

    Total number of requests:                          18,110
    Total number of connections:                       18,110

    Average requests per second:                       150.92
    Average time to first byte (msecs):                 89.95
    Average time to last byte (msecs):                  97.36
    Average time to last byte per iteration (msecs):  1,457.22
```

```
Properties

    Test type:                                        Dynamic
    Simultaneous browser connections:                      15
    Warm up time (secs):                                   10
    Test duration:                                00:00:02:00
    Test iterations:                                    3,681
    Detailed test results generated:                      Yes

Summary

    Total number of requests:                          55,226
    Total number of connections:                       55,226

    Average requests per second:                       460.22
    Average time to first byte (msecs):                 24.62
    Average time to last byte (msecs):                  24.75
    Average time to last byte per iteration (msecs):   371.25
```

Summarised Results (caveats next… ☺)



Results validity – comparison to "real" PetShop application profiling

Performance parameters	**Real ASP.NET PetShop**	**SoftArch/Thin-generated PetShop**
Overall average RPS (requests/second)	419.56	460.22
Overall average (ms)	28.67	24.75
accountManagePage	27.34	26.36
categoryManagePage	23.42	23.56
itemManagePage	23.74	24.34
orderManagePage	***39.15***	***27.34***
productManagePage	23.63	24.01

Some Comments/Observations/ Limitations

- J2EE example uses Sun "free" servers etc; ASP ones use MS IIS commercial server etc
- Generated code only as good as the model used – if model wildly inaccurate; so are results
- Can evolve model and re-run tests as evolve design/application code
- Can model existing application and proposed new architecture and do performance tests
- Some code e.g. application logic, caching, complex middleware v. hard to generate

Conclusions

- Performance test-bed generation and analysis a promising approach to performance engineering
- To date comparison of test-bed generated code to real code performance is generally good
- Challenging to develop code generators – need IDE; need to import parts of model from CASE
- Integrating with Argo/UML CASE tool…

References

- Cai, Y., Grundy, J.C., Hosking, J.G., Dai, X. Software Architecture Modelling and Performance Analysis with Argo/MTE, In Proceedings of the 2004 Conference on Software Engineering and Knowledge Engineering, Baniff, Canada, June 20-24 2004.
- Grundy, J.C., Cai, Y. and Liu, A. SoftArch/MTE: Generating Distributed System Test-beds from High-level Software Architecture Descriptions, Automated Software Engineering, Kluwer Academic Publishers, vol. 12, no. 1, January 2005, pp. 5-39.
- Cai, Y., Grundy, J.C. and Hosking, J.G. Experiences Integrating and Scaling a Performance Test Bed Generator with an Open Source CASE Tool, In Proceedings of the 2004 IEEE International Conference on Automated Software Engineering, Linz, Austria, September 20-24, IEEE CS Press, pp. 36-45.
- Cai, Y., Grundy, J.C., Hosking, J.G., Dai, X. Software Architecture Modelling and Performance Analysis with Argo/MTE, In Proceedings of the 2004 Conference on Software Engineering and Knowledge Engineering, Baniff, Canada, June 20-24 2004.
- Grundy, J.C., Wei, Z., Nicolescu, R. and Cai, Y. An Environment for Automated Performance Evaluation of J2EE and ASP.NET Thin-client Architectures, In Proceedings of the 2004 Australian Software Engineering Conference, Melbourne, Australia, April 14-17 2004, IEEE CS Press.
- Grundy, J.C., Cai, Y. and Liu, A. Generation of Distributed System Test-beds from High-level Software Architecture Descriptions, In Proceedings of the 16th International Conference on Automated Software Engineering, San Diego, 26-29 Nov 2001, IEEE CS Press, pp. 193-200.