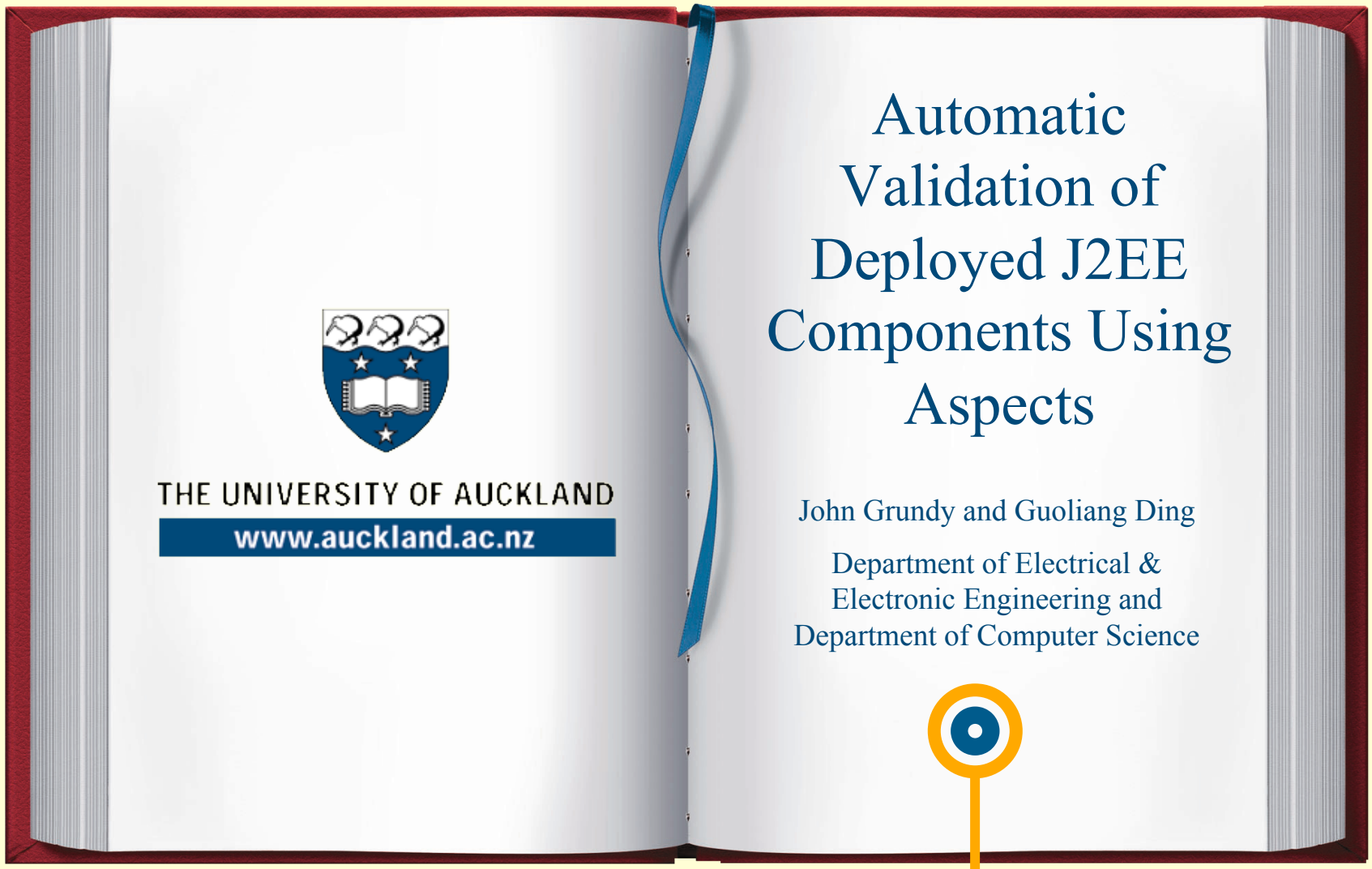


Automatic Validation of Deployed J2EE Components Using Aspects

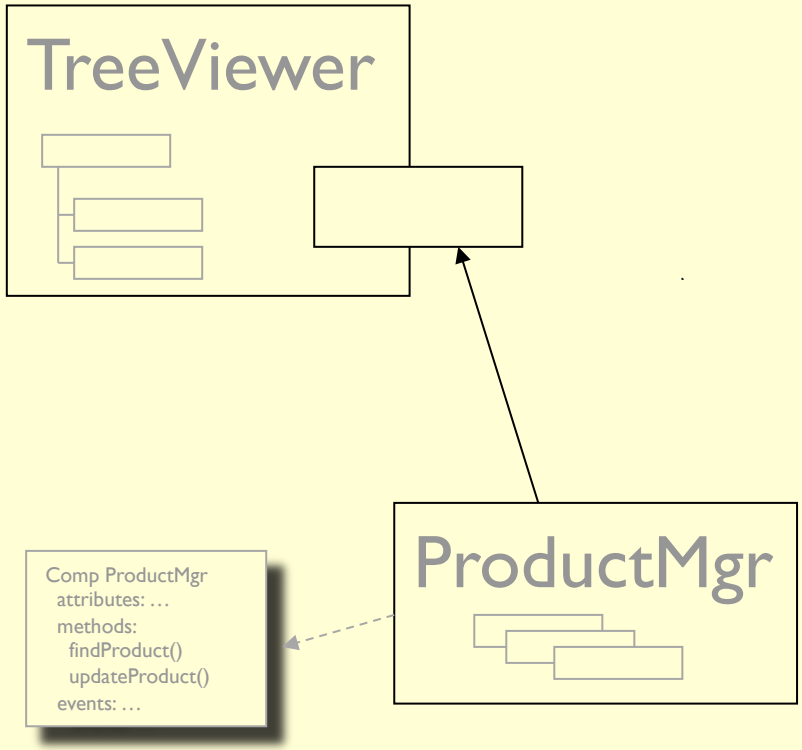
John Grundy and Guoliang Ding

Department of Electrical & Electronic Engineering and Department of Computer Science

Outline

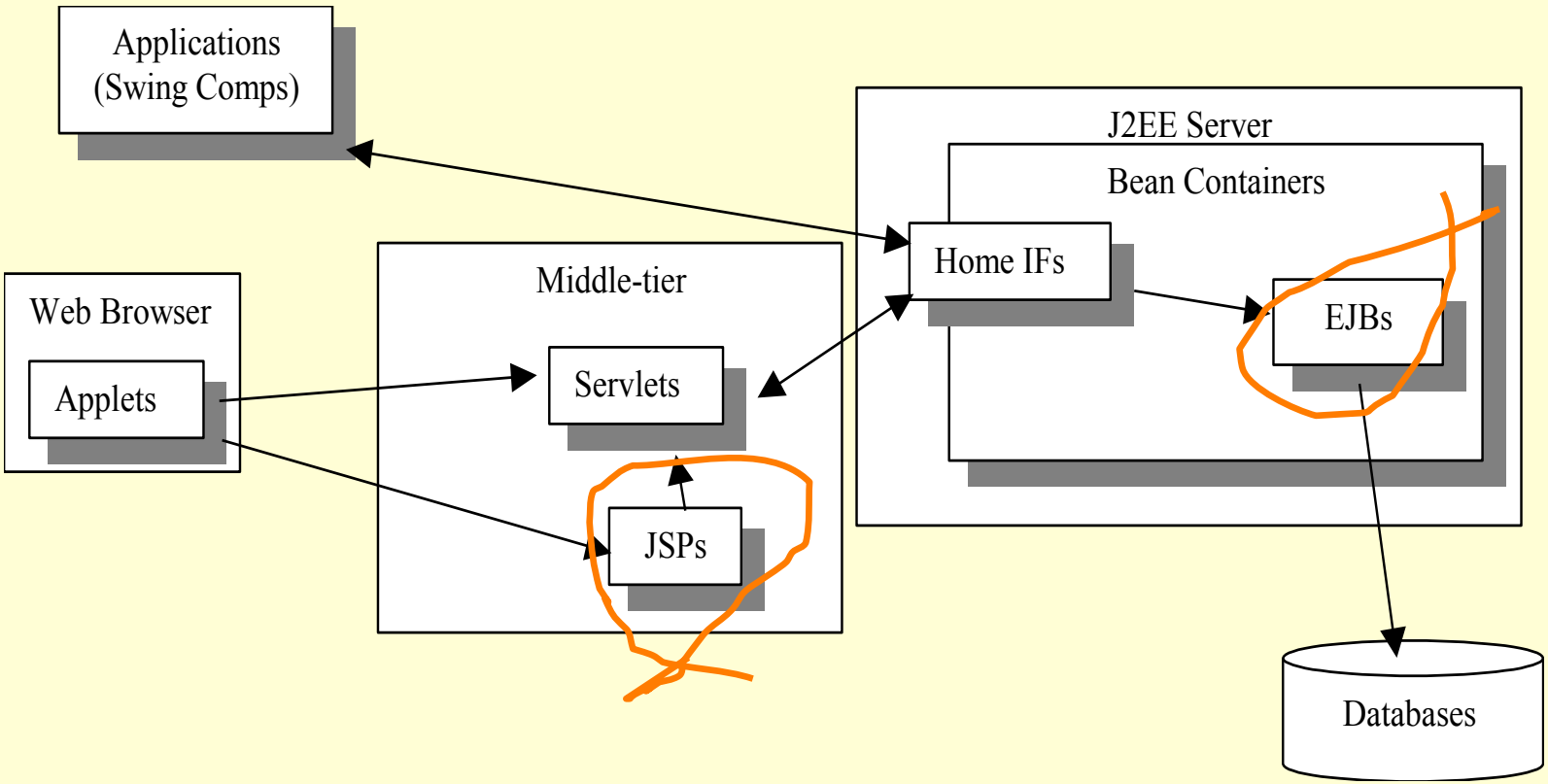
- Motivation
- Our Approach
- Example System
- Characterising software components
- Deploying and validating components
- Summary & Future Work

What are Software Components?



- Data/behaviour
- Interfaces
- Self-describing
- Plug-and-play
- Various constraints…
- e.g. J2EE, COM, .NET, CORBA, web services

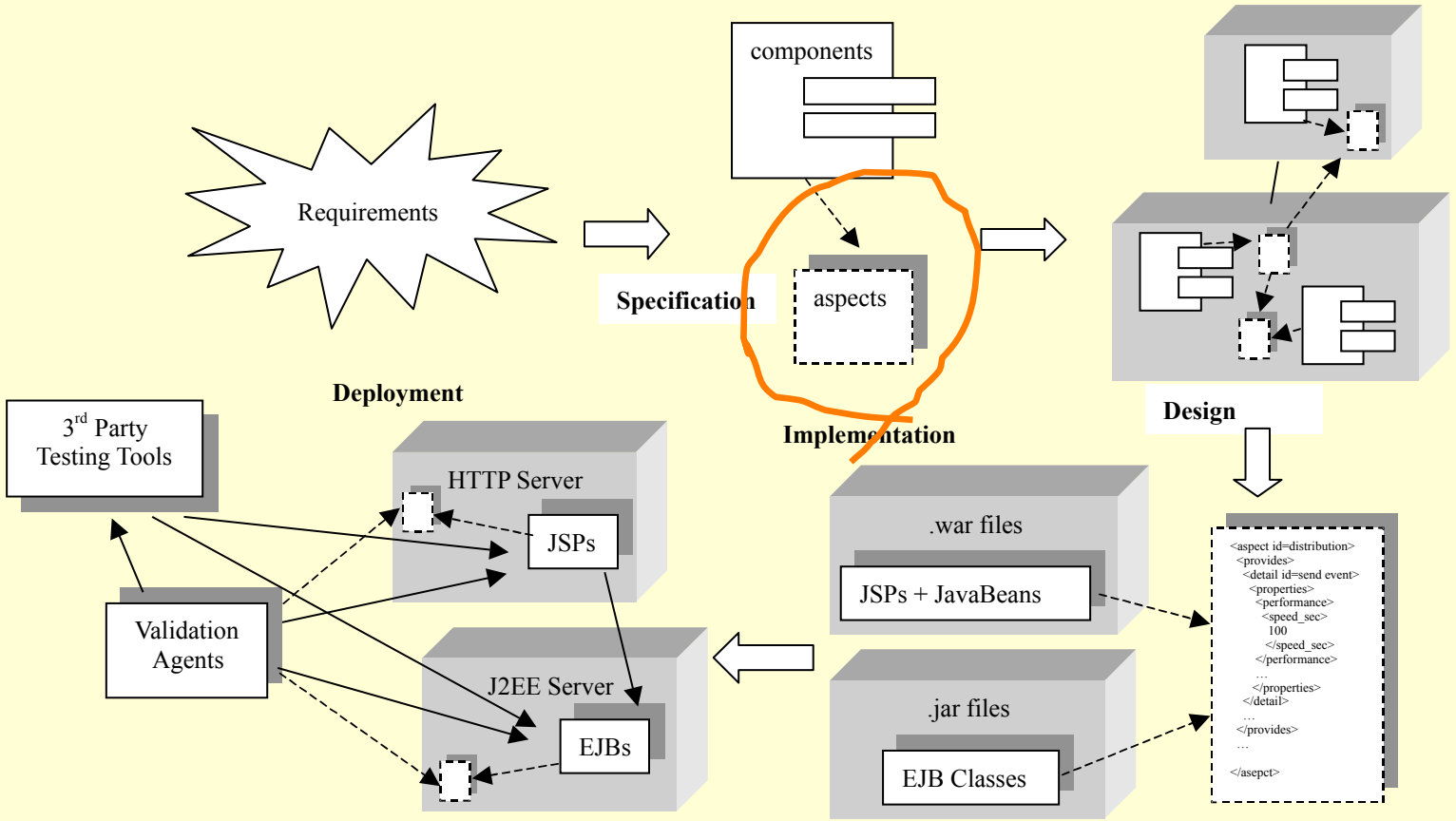
Java 2 Enterprise Edition Components



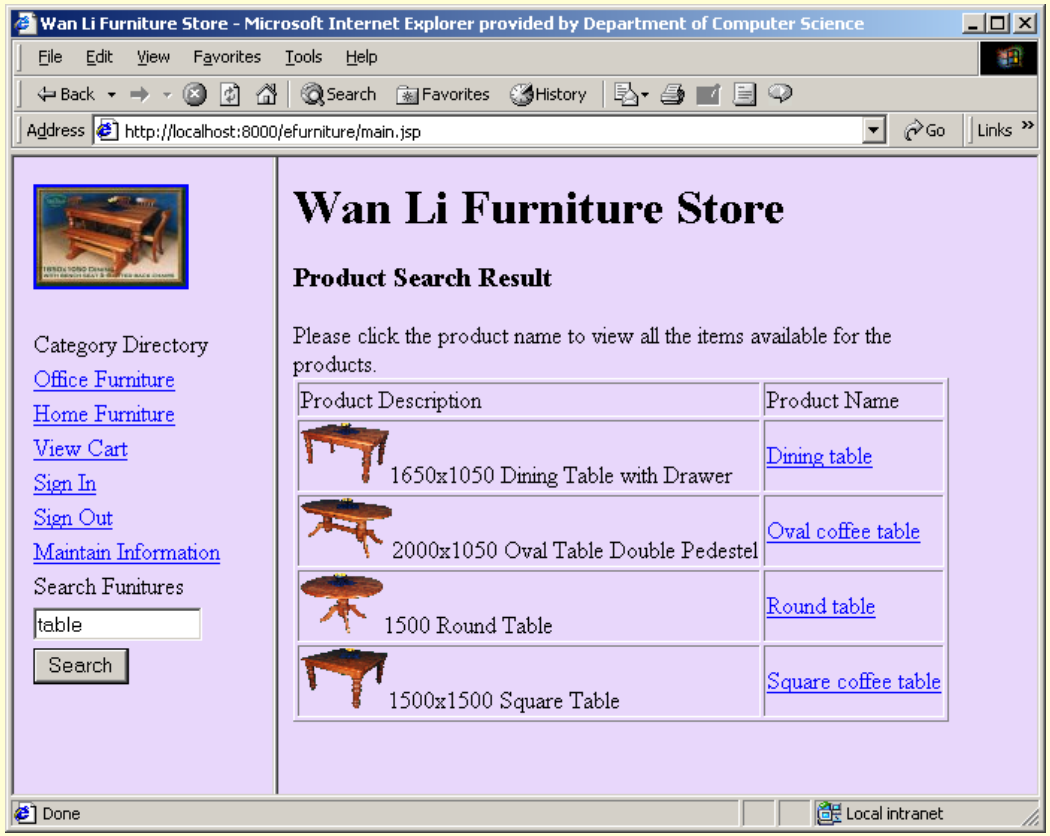
Ensuring Deployed Components Meet Specifications

- Component requirements/constraints:
  - Functional behaviour – data & processing they embody
  - Transaction processing, data integrity, security , resource usage
  - Performance, # users, reliability, trustability, …
  - Composite components – combine these constraints…
- Issues:
  - Static checking of constraints limited in usefulness…
  - Deployment scenario effects how component behaves
  - Depends on hardware, networking, applications, loading etc etc
- So how do we ensure deployed components meet their specifications???

Our Approach



An Example…

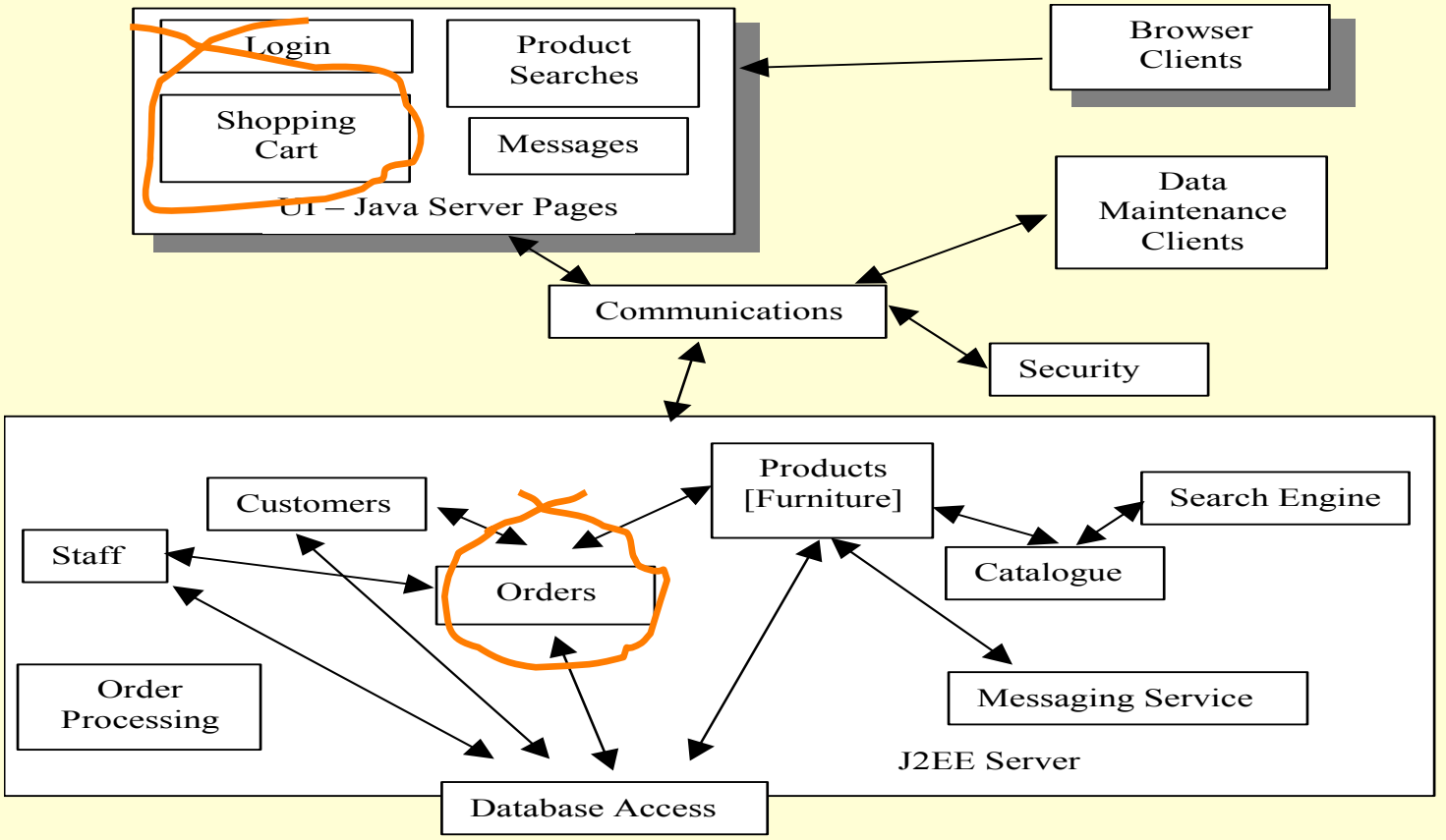


- Catalogue of products
- Searching
- Product info
- Orders
- Payment
- Messaging

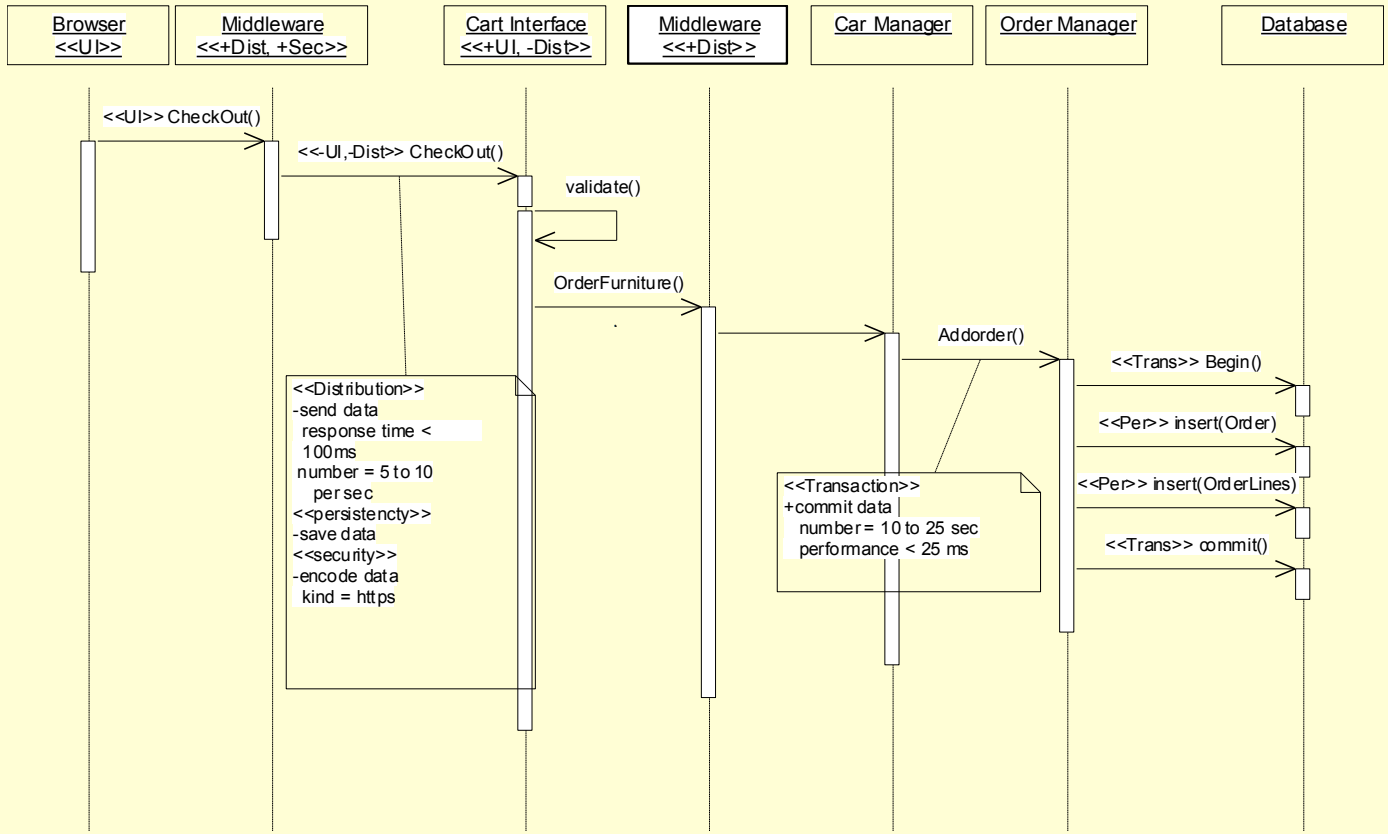
Software Engineering
The University of Auckland

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Some components



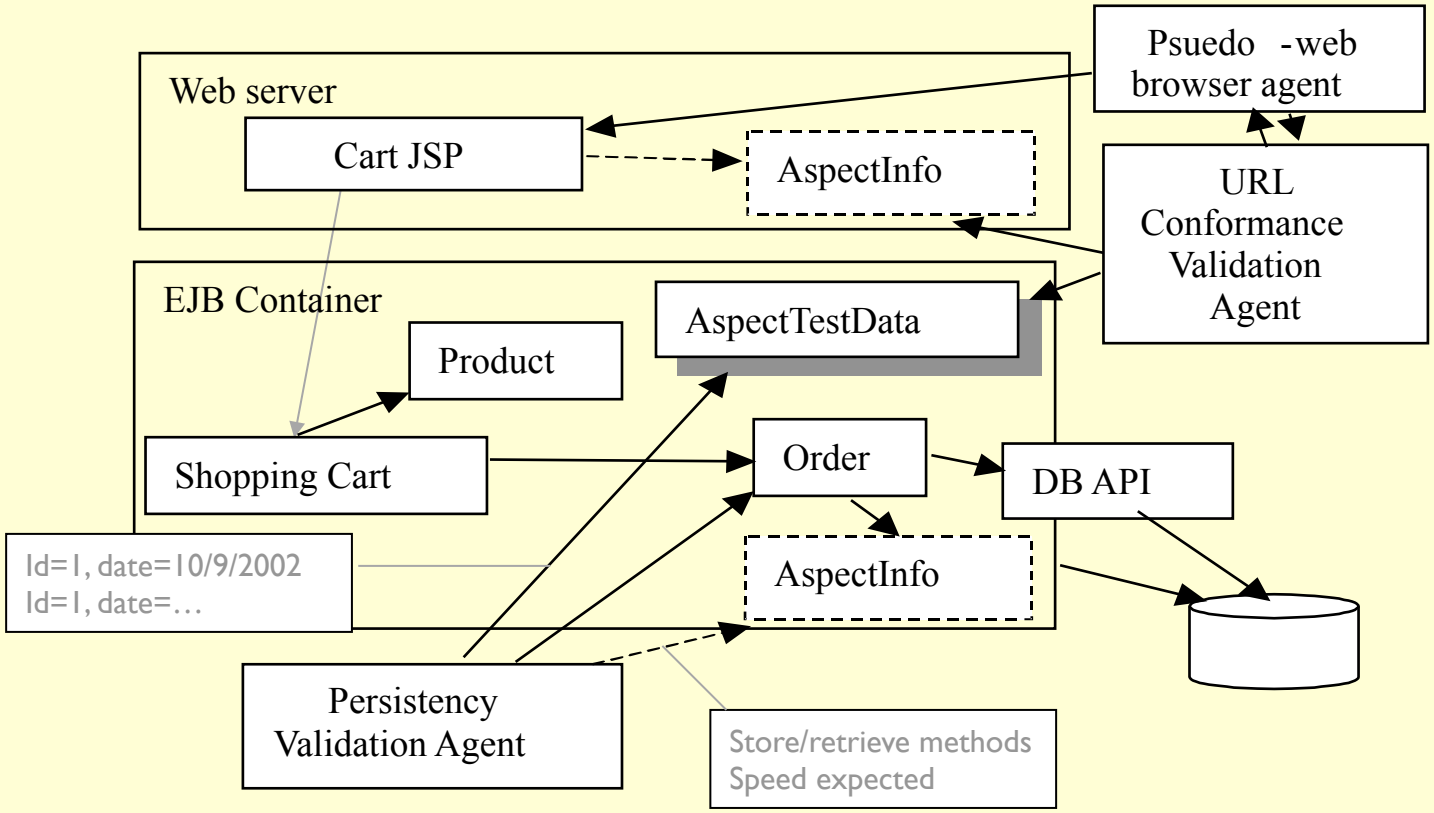
Characterisations of Components – via “Aspects”



XML Encoding of Aspects

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT (Component|ComponentGroup)+>
<!ELEMENT Component (CompName, MappingName, CompProperties,
CompMethods, CompEvents, CompAspects)>
...
<!ELEMENT Aspects (Aspect)+>
<!ELEMENT Aspect (AspectName,Details)>
<!ELEMENT Details (Detail)+>
<!ELEMENT Detail (DetailName, DetailType, Provided, DetailProperties,
ImpactedMethods, DetailInfo)>
...
<!ELEMENT DetailProperty (DetailPropName, DetailPropType,
DetailPropConstraint, DetailTestMethods, DetailPropInfo)>
...
<!ELEMENT DetailPropConstraint Expr>
...
<!ELEMENT DetailTestMethod (MethodCall|URLCall)>
<!ELEMENT MethodCall (MethodName, MethodArgumentData)>
<!ELEMENT URLCall (URLName, URLArgumentData)>
```

Deploy Components/Validation Agents – an Example



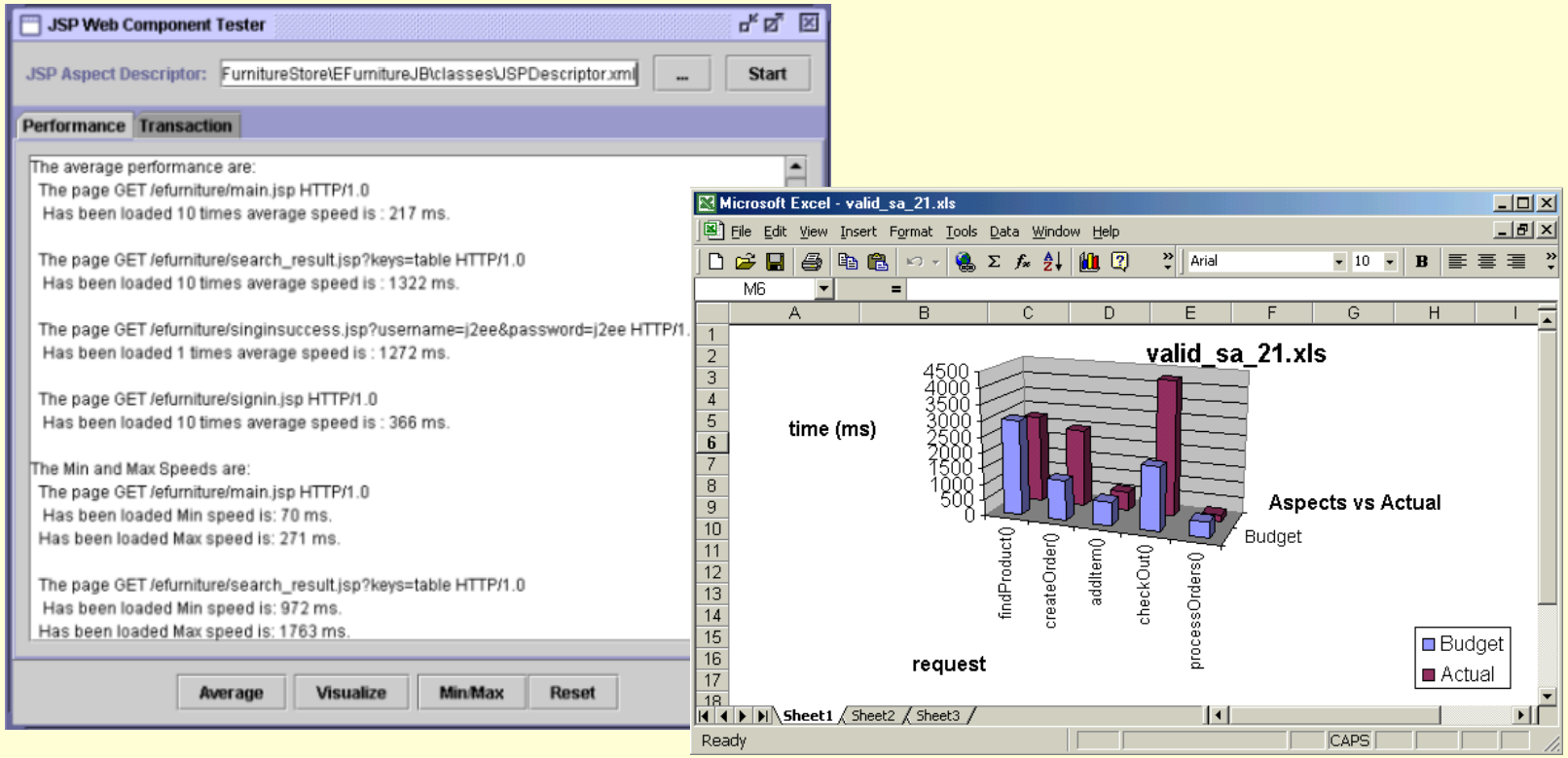
Example Aspects Extracted

```
<Component CompName="OrderEJB">
 <Aspects>
   <Aspect AspectName="Persistency">
    <Detail DetailName="Store" DetailType="StoreData" Provided='true'>
     < ImpactedMethod Name='ejbCreate' /><ImpactedMethod Name='ejbStore' />
    <DetailProperties>
     <DetailProperty DetailPropName="StoreSpeed" DetailType="ResponseTime">
      <DetailPropType>Milliseconds</DetailPropType>
      <DetailPropConstraint><Expr><LessThan>50</LessThan>...
      <DetailTestMethods>
        <DetailTestMethod MethodName="ejbStore"
            MethodArgumentData=' java:comp/env/ejb/staff_testdata'>
...

<Component CompName="ShoppingCart.jsp">
...  <Aspect AspectName="UserInterface" ...>
   ... <Detail DetailName="PostData" DetailType="DataOutput" Provided='true' ...>
     ... <DetailProperty DetailPropName="PostData" ...>
       .... <DetailTestMethod><URLCall URLName="ShoppingCart.jsp"
            URLArgumentData= java:comp/env/ejb/ShoppingCart_testdata'
              URLResultData = ... />
```

- XML gives full component spec – data, functions, non-functional constraints…
- Xpath expressions used by agents to obtain ones they want…

Results Visualisation



Evaluation

- Advantages:
  - Deployment-time assessment of components
  - Can deploy validation agents in different situations for reuse
  - Preliminary results compare well to manual testing efforts
- Disadvantages:
  - Extra effort of specifying aspect information (need tools)
  - Composite component aspects – if run-time composition? If dynamic architecture??
  - Aspect "ontologies" – what if compose comps with diff aspects?

Summary

- Validating deployed software components – checking that they meet their functional/non-functional constraints – is HARD!
- Static analysis not sufficient; manual testing too hard/error-prone; JUnit-style test drivers require much coding
- Aspect characterisations of components capture rich information about their functions/constraints
- Validation agents using component aspects promising approach
- We have developed J2EE-based validation agents; a number of different validation agents prototyped; have tried on some relatively simple J2EE applications/components…

References

- Grundy, J. and Patel, R. Developing Software Components with the UML, Enterprise Java Beans and Aspects, In Proceedings of the 2001 Australian Software Engineering Conference, Canberra, Australia, 26-28 August 2001, IEEE CS Press.
- Grundy, J.C. Multi-perspective specification, design and implementation of components using aspects, International Journal of Software Engineering and Knowledge Engineering, Vol. 10, No. 6, December 2000, World Scientific.
- Grundy, J.C., Mugridge, W.B. and Hosking, J.G. Constructing component-based software engineering environments: issues and experiences, Information and Software Technology Vol 42, No. 2, Special Issue on Constructing Software Engineering Tools, Elsevier Science Publishers.
- Grundy, J.C. Storage and retrieval of Software Components using Aspects, In Proceedings of the 2000 Australasian Computer Science Conference, Canberra, Australia, Jan 30-Feb 3 2000, IEEE CS Press, pp 95-103.
- Grundy, J.C. Aspect-oriented Requirements Engineering for Component-based Software Systems, 1999 IEEE Symposium on Requirements Engineering, Limmerick, Ireland, 7-11 June, 1999, IEEE CS Press.
- Grundy, J.C. and Hosking, J.G. Serendipity: integrated environment support for process modelling, enactment and work coordination, Automated Software Engineering: Special Issue on Process Technology, Vol. 5, No. 1, January 1998, Kluwer Academic Publishers, pp. 27-60.
- Grundy, J.C., Mugridge, W.B., Hosking, J.G., Apperley, M.D., Tool integration, collaboration and user interaction issues in component-based software architectures, In Proceedings of TOOLS Pacific'98, Melbourne, Australia, 24-26 Nov, IEEE CS Press.