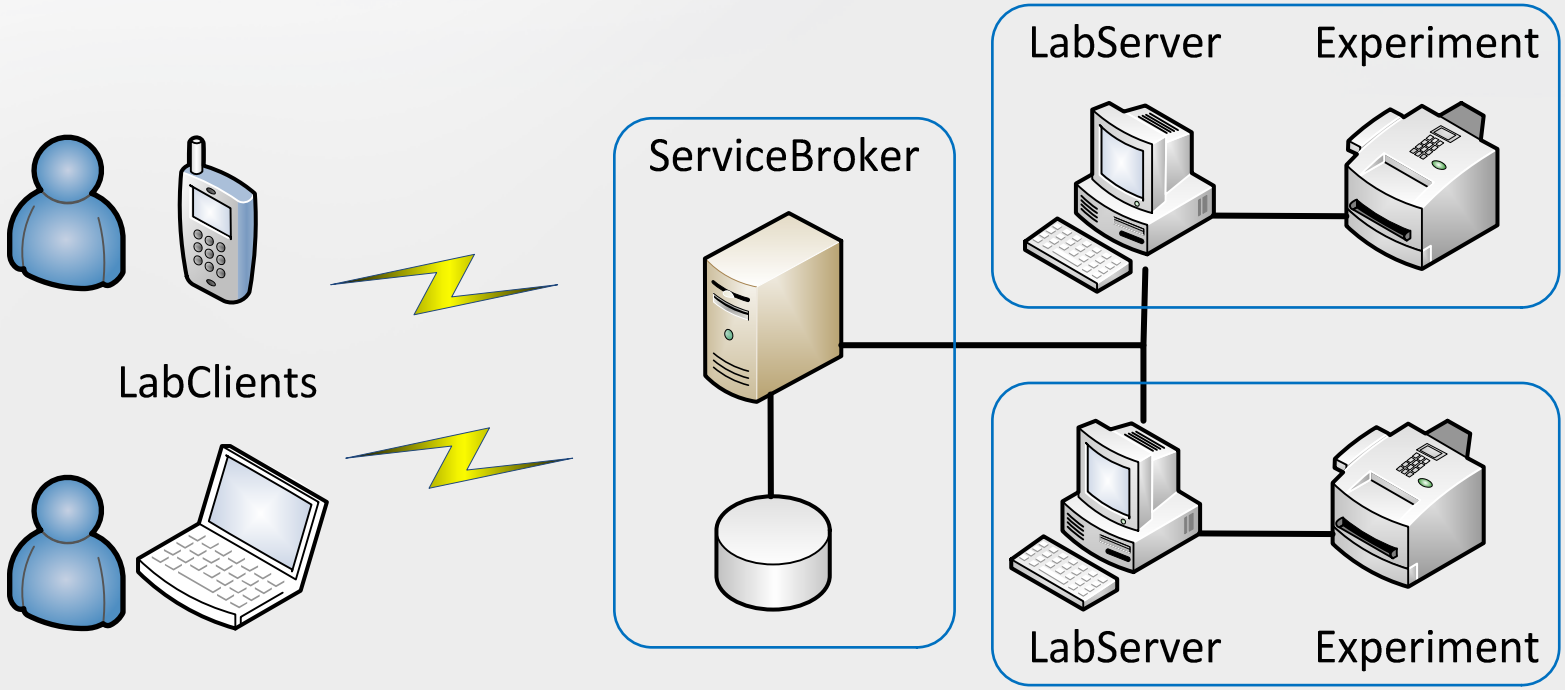
JAVA Implementation of the Batched iLab Shared Architecture

Lenard J. Payne, Mark F. Schulz
The University of Queensland
uqlpayne@uq.edu.au

MIT's Batched iLab Shared Architecture

- Microsoft Windows only
- Microsoft DotNet Web Services
- Microsoft SQL Database



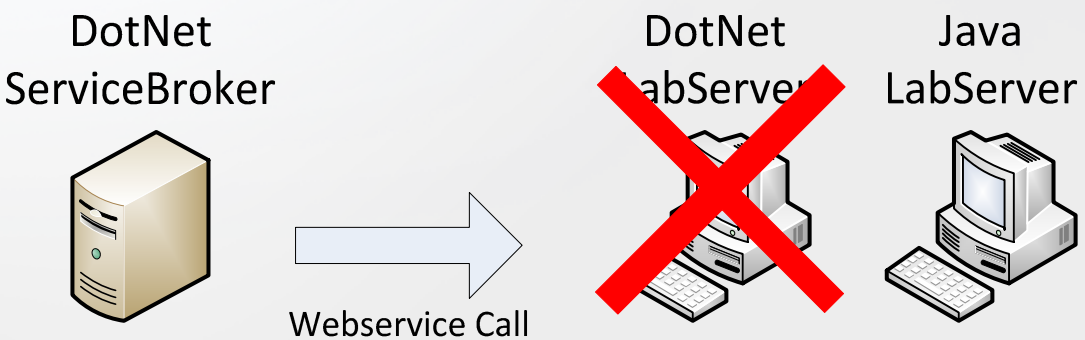
Development Enviroment

- Microsoft
  - Visual Studio IDE (C#) and SQL Server Database
  - Licensed and expensive
  - Windows only

Alternative?

- Goals
  - Open source software
  - Operating system independence
- Solution
  - Java + NetBeans IDE
  - PostgreSQL Database

DotNet LabServer Web Service



Webservice Call

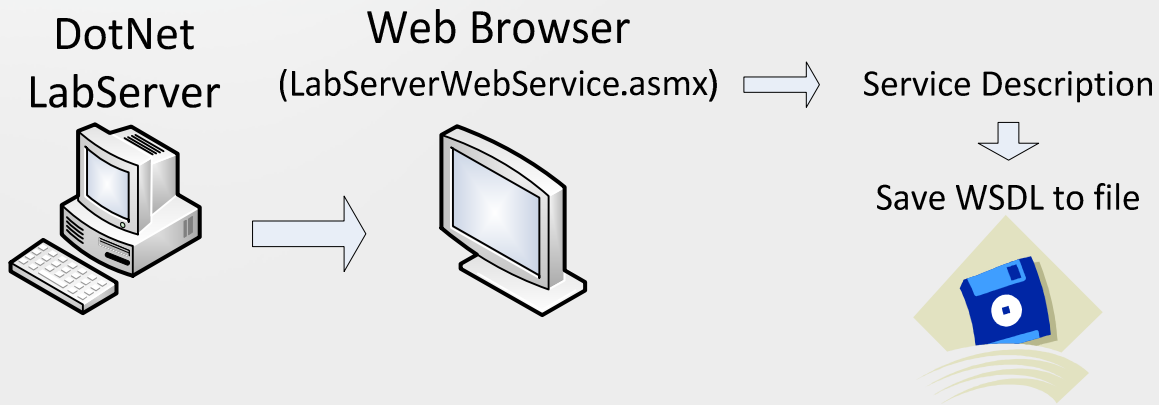
- Validate()
- Submit()
- RetrieveResult()

SOAP Header

- ServiceBroker's GUID
- Outgoing Passkey

Java Interoperability with DotNet

- Java provides *jax-ws* framework
- Framework requires web service Java classes
- Java classes generated from DotNet WSDL file



Java LabServer Web Service

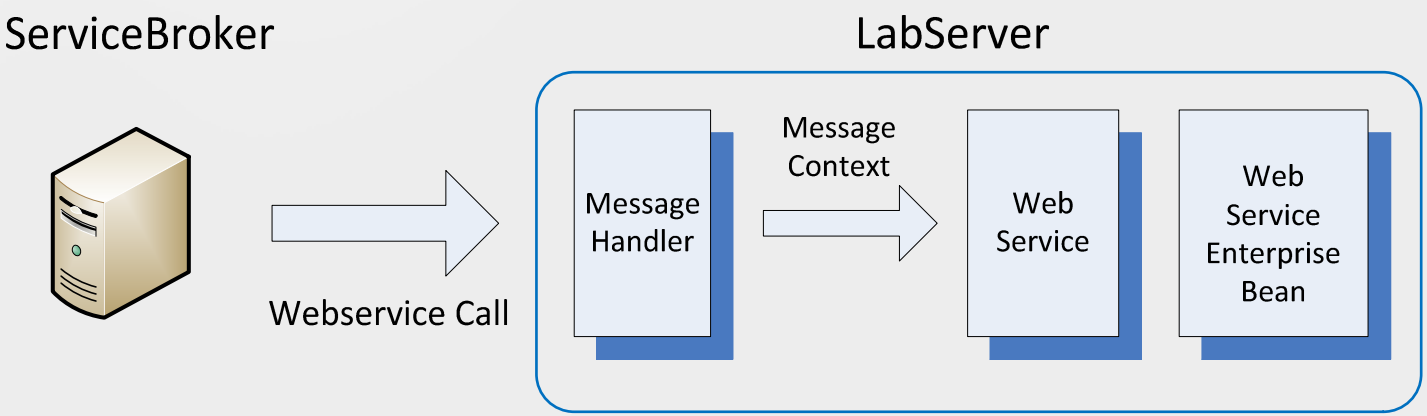
- Using NetBeans IDE
- Create new *Java Web Application* project
- Create new *Web Service from WSDL*
- Select WSDL file generated from DotNet
- Create an *Enterprise Bean* for the web service

What about the SOAP header?

- Create new *Web Services->Message Handler*
- *Web Services->Configure Handlers … Add*

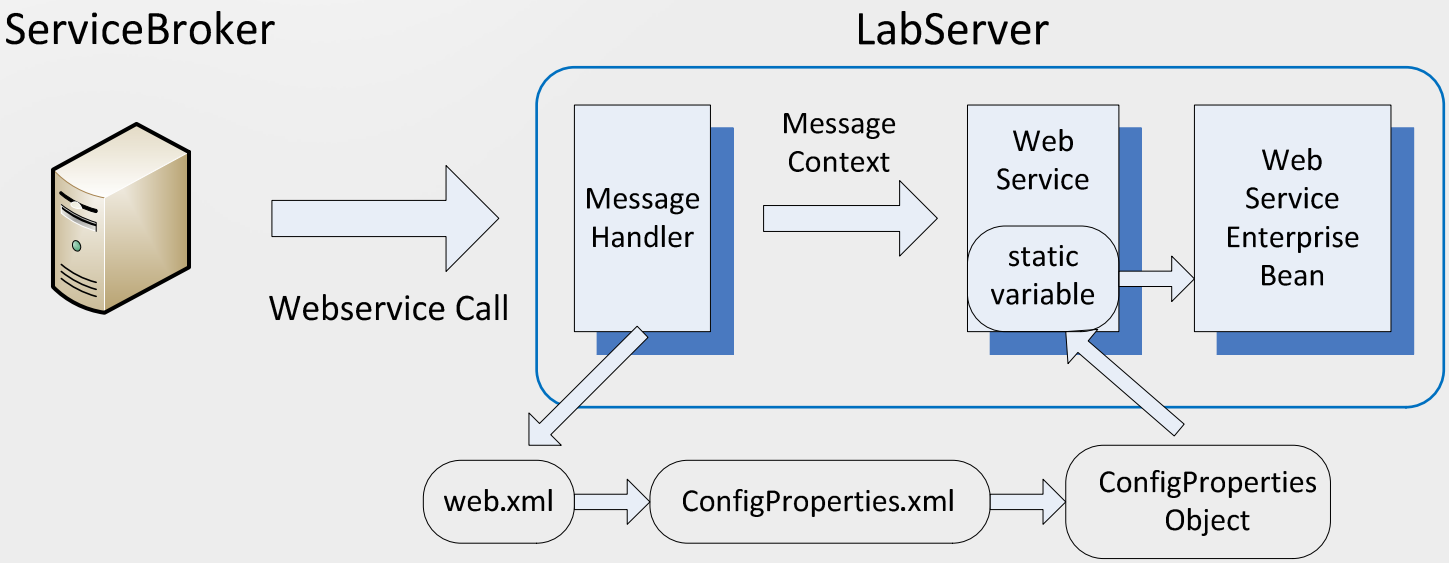
Message Handler

- First point of contact with web service
- Extract information from SOAP header
- Pass information to web service – Message Context
- Inbound messages only



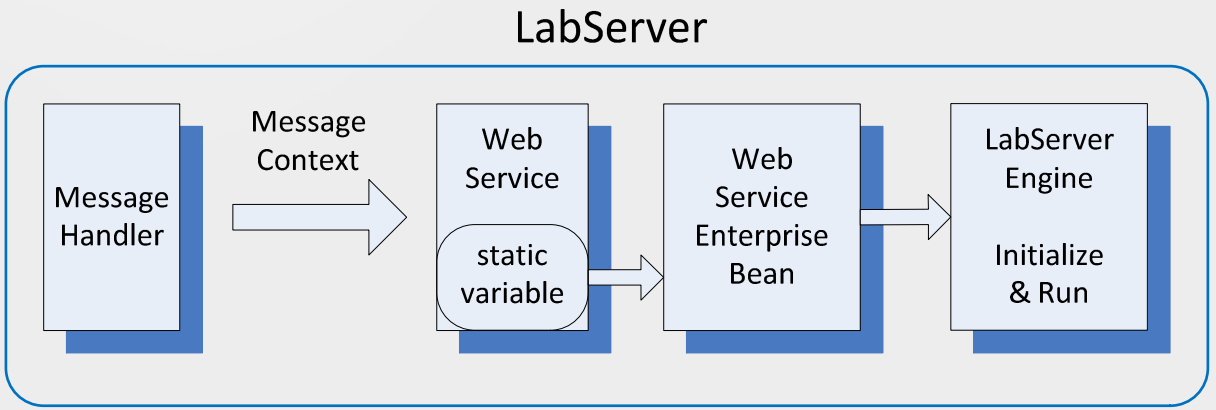
Web Service Initialization

- Message handler – first point of contact
- Read configuration file location from *web.xml*
- Create *ConfigProperties* object
- Set *static* variable in web service



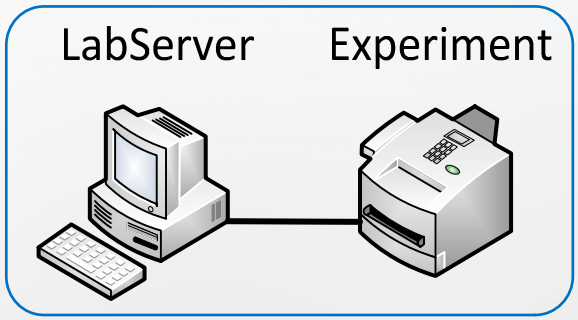
Web Service Initialization - continued

- Web service bean gets *static* variable
- Finishes initialization
  - Creates LabServer Engine
    - Database access methods, execution threads, etc

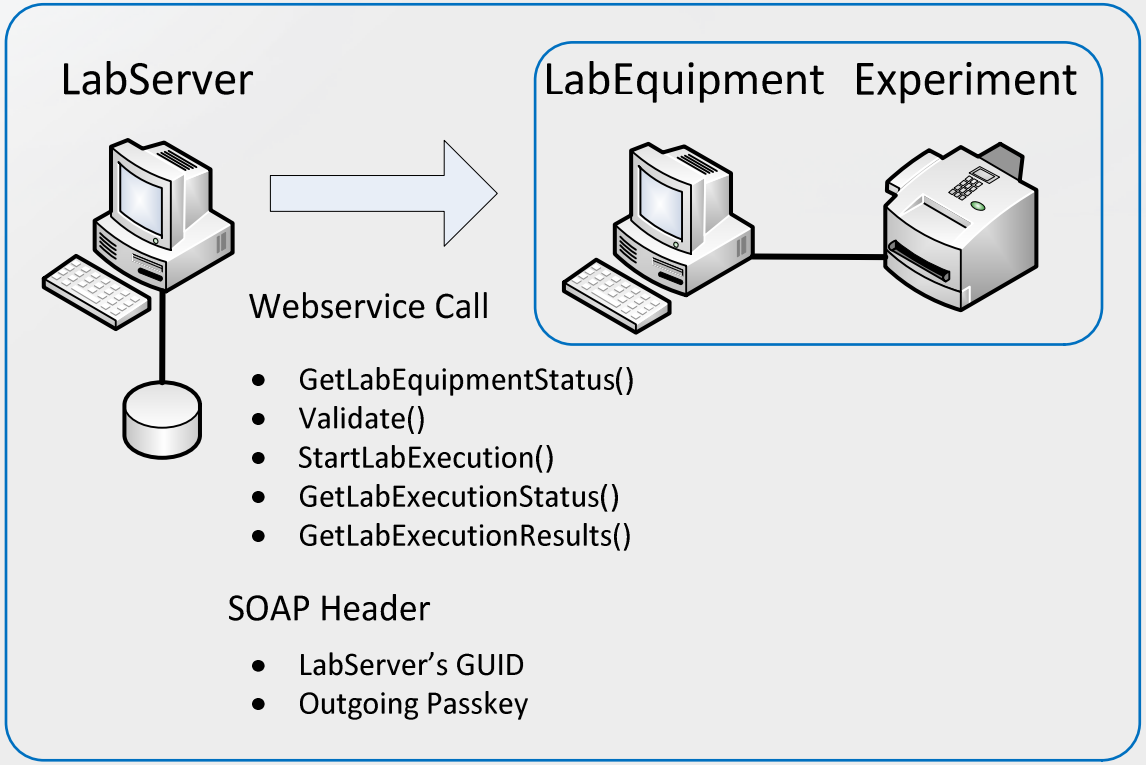


Revised LabServer Model

- LabServer split into LabServer + LabEquipment



BECOMES

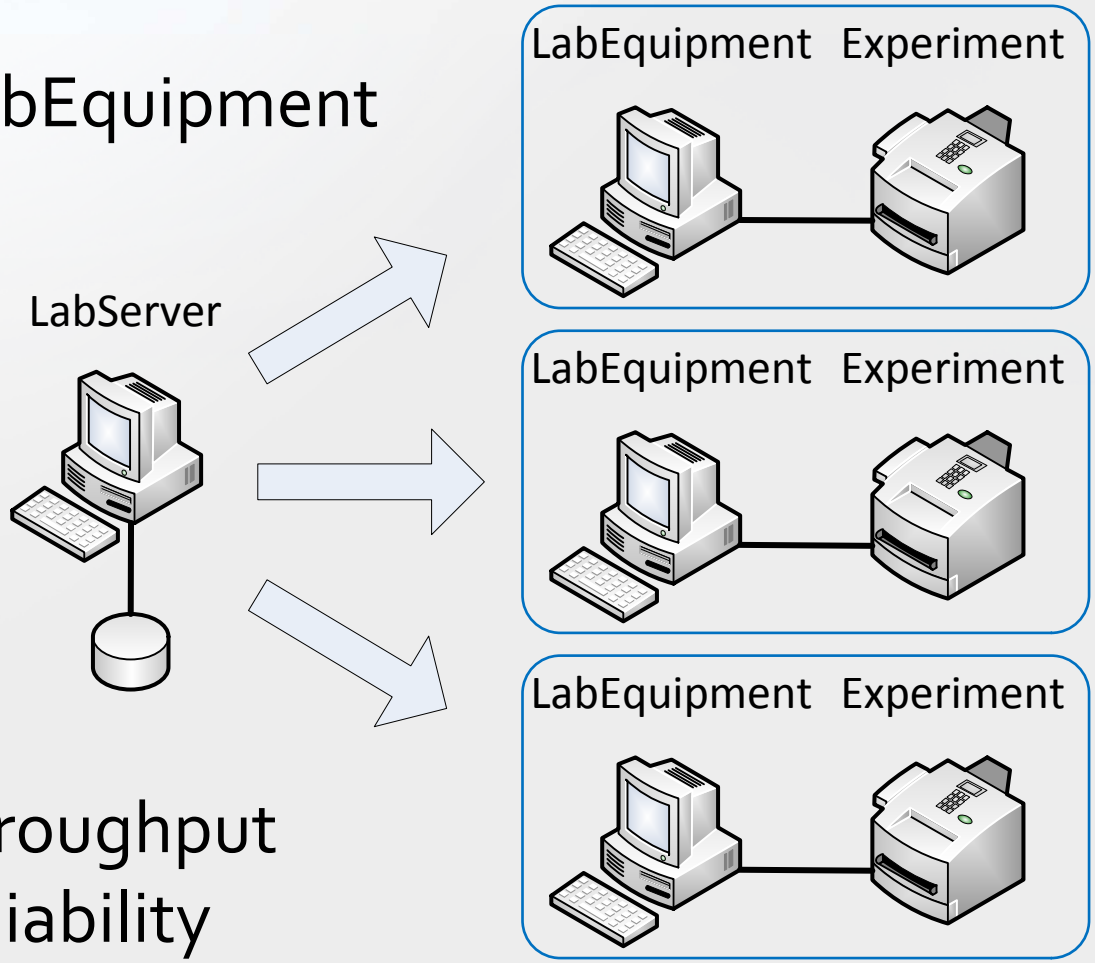


Revised LabServer Model - continued

- LabServer
  - Experiment hardware independent
  - Handles experiment submission
  - Local database queues experiments
  - Can be remote from equipment
  - Becomes manager not worker
- LabEquipment
  - Experiment hardware specific (Windows only?)
  - Handles experiment execution
  - Powers up /down equipment
  - Resides with equipment

LabEquipment Farm

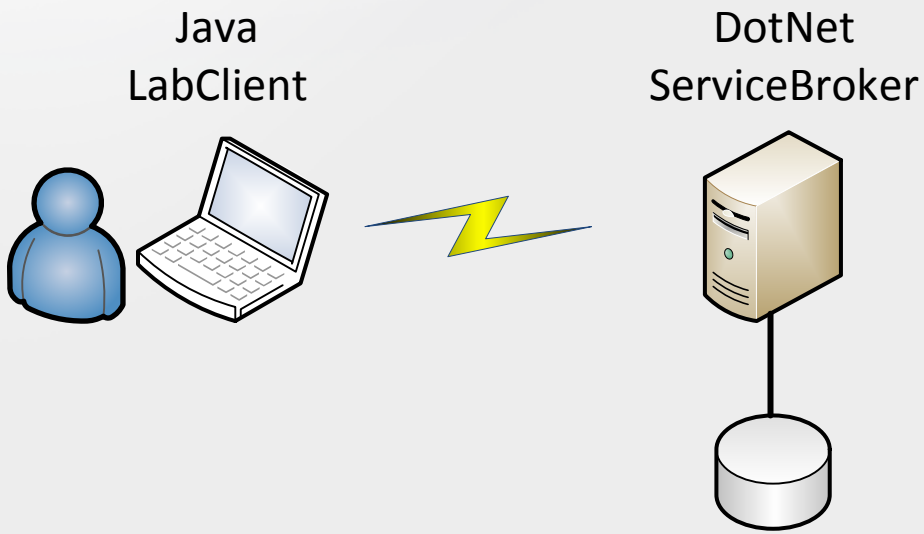
- Duplicate LabEquipment



- Increased throughput
- Improved reliability

Java LabClient

- JavaServer Faces
- Applet



Java ServiceBroker Web Reference

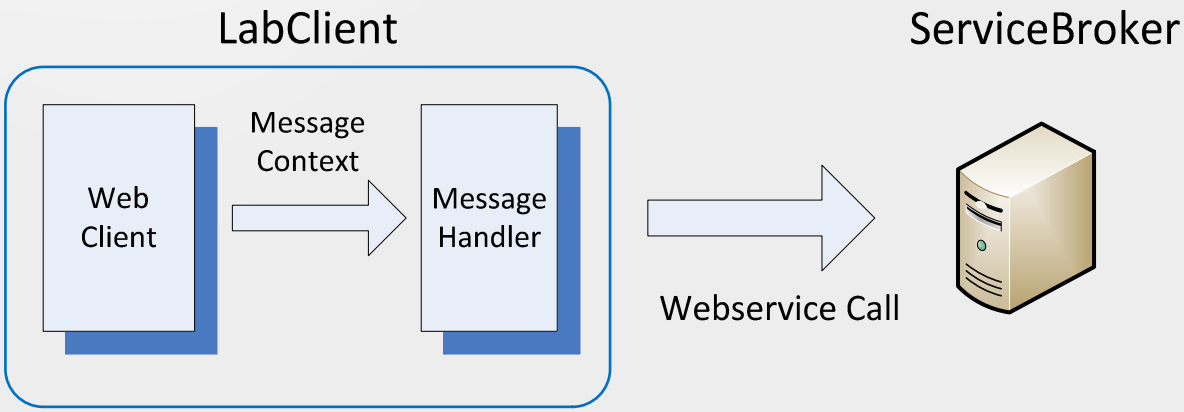
- Using NetBeans IDE
- Create new *Java Class Library* project
- Create new *Web Service Client*
- Select WSDL file generated from DotNet ServiceBroker

What about the SOAP header?

- Create new *Web Services->Message Handler*
- *Web Service References->Configure Handlers … Add*

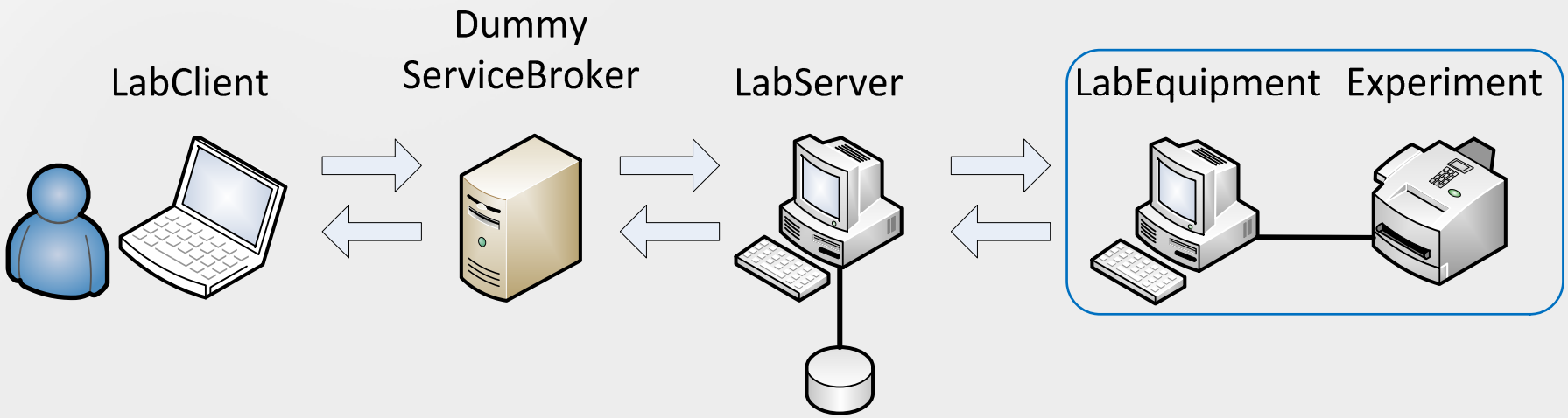
Message Handler

- Insert information into SOAP header
- Pass information from web client – Message Context
- Outbound messages only



Dummy ServiceBroker

- LabClient/LabServer development use only
- Pass-through methods: LabClient -> LabServer
- Debug from LabClient to LabEquipment and back
- Only generates an experiment Id



Three-Tier Code Development

- Bottom Tier
  - Libraries common to all LabServers / LabEquipment
  - Base classes, Database routines, Engine threads
  - For LabServers: LabEquipment communication
  - For LabEquipment: Powers up/down equipment
- Middle Tier
  - Library specific to each experiment
  - Experiment execution drivers, LabEquipment devices
- Top Tier
  - Web service application
  - Same for all LabServers / LabEquipment

Three-Tier Code Development - continued

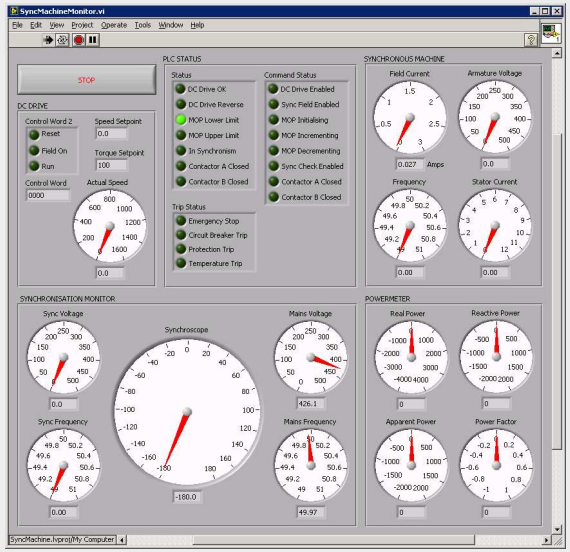
- Advantages
  - Reuse common code libraries
  - Focus only on experiment specific implementation
  - Previous experiments become templates for new experiments
  - Reduced overall time and effort

Examples

- AC Machines and DC Machines
  - For each, five sets of identical equipment in a farm
  - Lab camera on each set

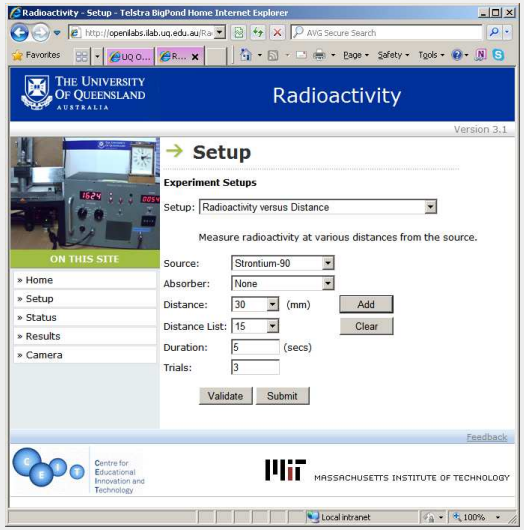
Examples

- Synchronous Machine
  - Two sets of identical equipment in a farm
  - Lab camera on each set



Examples

- Radioactivity
  - One set of equipment without absorbers
  - One set of equipment with absorbers
  - Lab camera on each set



Conclusions

- Host iLab experiments on non-Windows platforms
- Java *jax-ws* framework interoperates with DotNet
- Three-Tier code development reduces time & effort

- Code available as open source from:

  https://github.com/uqlpayne/UQ-iLab-BatchLabServer
- UQ OpeniLabs experiments available at:

  http://openilabs.ilab.uq.edu.au/ServiceBroker/

Coming soon …

- Java implementation of the iLab ServiceBroker