

Distributed Systems: Lessons Learned & Challenges Remaining

Kanai Award Lecture

Kanai Award



Dr. Alfred Z. Spector

Vice President, Services & Software Research

IBM Corporation

Abstract

The design of most distributed systems has settled down into a reasonably coherent architectural framework, with only moderate perturbations occurring from system to system, and from generation to generation. I'll describe the framework, some lessons learned as this framework has progressed over the last 25 years, and some important research and engineering challenges ahead. I'll extrapolate to the types of amazing systems we might successfully construct, if we can meet these challenges



Outline

- Distributed Systems
- A Historical Perspective
- Web Services
- Lessons Learned
- Future Challenges
- The 2001 Kanai Award

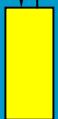
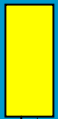
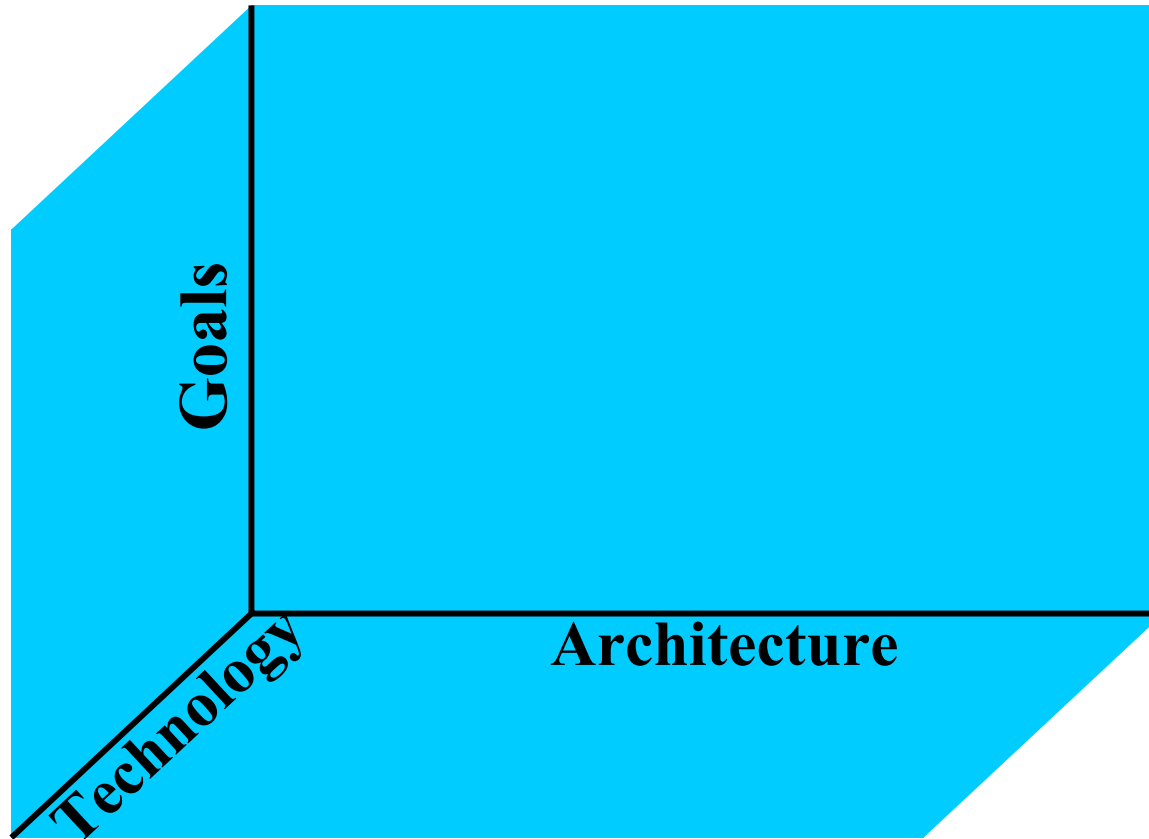


Distributed Systems

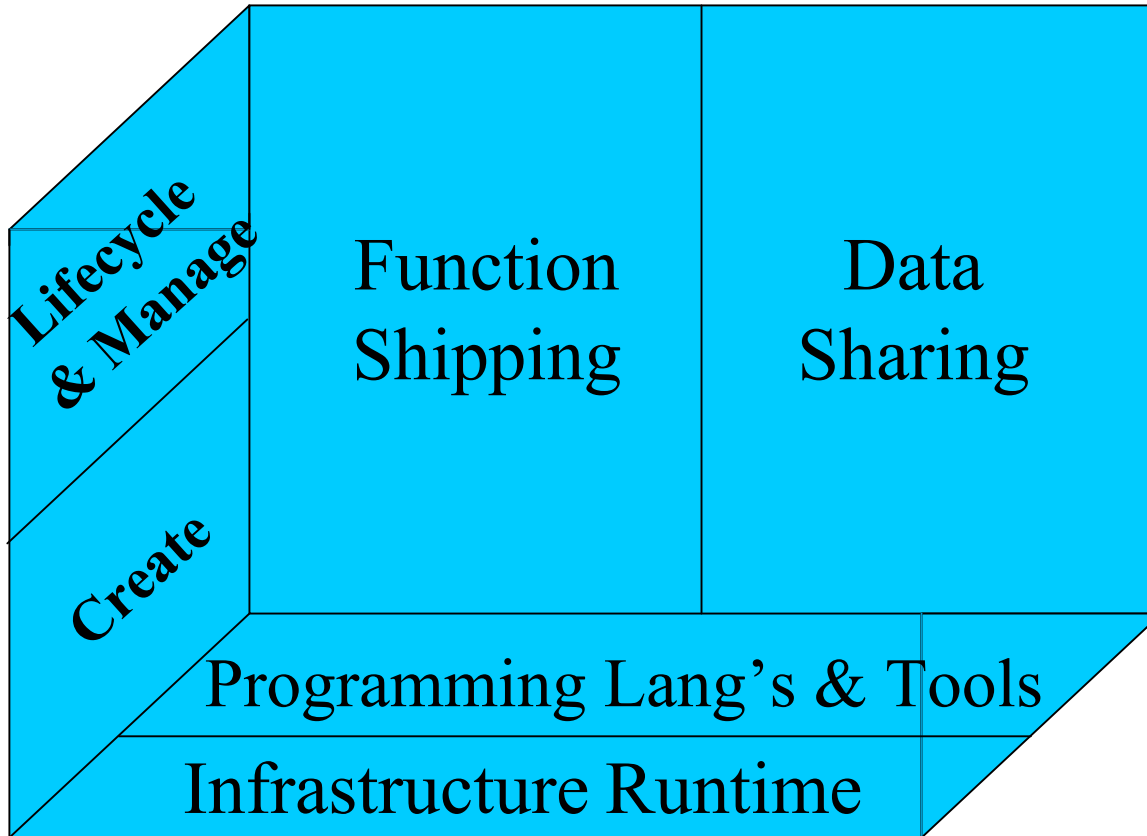
- Three definitions:
 - A computer science discipline aimed at reducing the cost of *developing & managing* applications that utilize multiple, networked systems
 - Particular technology (infrastructure, specifications, or software) aimed at reduced development & management cost
 - A complete system that implements a particular distributed application



3 Dimensions of Distributed Systems



3 Dimensions of Distributed Systems



The Distributed Systems Cube

Architecture

Function Shipping

- Advantages:
 - OO Nature
 - Flexibility
- Examples:
 - DCE, Corba, RMI, DCOM
 - Eventual Web Appeal

Data Sharing

- Ease of Use
 - Predefined capabilities
- Examples:
 - Distr. File Systems
 - Distr. Database
 - Initial Web Appeal

- Note: There really is a continuum
 - Databases have stored procedure
 - Object-oriented systems have dual flavors
- The Web today utilizes both

Goals

Create

- Primary focus of academic computer science
- Plethora of techniques now required
- Blending of runtime & tools

Lifecycle & Manageability

- The greater cost
- Plethora of tools now required
- Blending of runtime & tools

- Note: This research community has tended to focus on dev't
 - Lifecycle & Manageability are *at least* equal problem
 - Albeit, proper development is also key to manageability
 - Autonomic Computing* must be a greater academic focus

Technology

Programming Langs & Tools

- Traditionally, a community unto itself
- Often overly focused on programming issues, while ignoring essential distributed semantics

Runtime Infrastructure

- Traditionally, a community unto itself
- Not as focused on programming, but rather on algorithms, architecture & semantics

- Note these are hard problems:
 - Programming gets far harder if one tries to include issues adaptability, reliability and availability, concurrency, ...
 - Runtime infrastructures can be remarkably difficult to use



A Historical Perspective

- Distributed computing fundamentals known for at least 25 years
- Early great papers & collections – some favorites:
 - Notes on Database Operating Systems, Jim Gray, in *Operating Systems: an Adv. Course*, Springer-Verlag, 1978
 - Using Encryption for Authentication in Large Networks of Computers, R. Needham and M. Schroeder, CACM, 21(12), 12/78
 - On the Duality of Operating System Structures, H. Lauer, & R. Needham, *OSR* 13(2), 4/79
 - Distributed Systems Architecture Model, Richard Watson, in *Distributed Systems, Architecture & Implementation*, Springer-Verlag, 1980
 - Remote Procedure Call, Bruce Nelson, CMU Ph.D. Dissertation, 1981
- The Distributed Systems summer course 1982-1989

1980's Western Institute of Computer Science Distributed Systems Course Topics

- A. Spector / D. Gifford / R. Rashid, (J. Gray replaced Spector in late 80's)
- Threading
- Function shipping: Message Passing & Remote Procedure Call
- Authentication, authorization, privacy
- Integrity/availability
 - Transactions
 - Distributed Replication
- Name Service, Time Service, ...
- Data sharing techniques, and caching
- Related programming language issues

Personal Contributions

- Remote Reference/Remote Operation Model & extremely efficient RPC system
- Extensive work on Transactional Remote Procedure Call
- Wide-area file systems & contributions to stateful, file systems
- Contributions to Decorum (later, DCE) Architecture
- Contributions to Application Server Architectures: TABS, Camelot, Encina/DCE
- Commercialization of the above

Results of Our Research?

- Many attempts to create a successful world-wide distributed computing standard
 - Mach Projects
 - DCE
 - CORBA/IIOP
 - DCOM
 - RMI
 - And more..., but they did not have desired, universal impact
- **HTTP & HTML with DNS had impact!**
 - *Not* based on breadth of distributed computing research
 - And yet, wildly successful!

Why Limited Impact of “Correct” Systems?

- Providing a *complete* programming, operating, & management environment for distributed computing is remarkably difficult
- There was then minimal economic incentive to extend technology and & work out details
- The technology was not fully understood
- The technical community tended to fragment along traditional lines

Why the Great Impact of HTML/HTTP?

- Simple goals → an elegant implementation
 - No security implied, paradoxically, lessened deployment resistance
 - Simple models supported initial use with few, if any, tools
 - Retooling of programs not required
- Significant economic value realizable without massive expense
- Worked with extant technology
- Limited need for a concerted effort by worldwide technical community

Limitations of HTTP/HTML

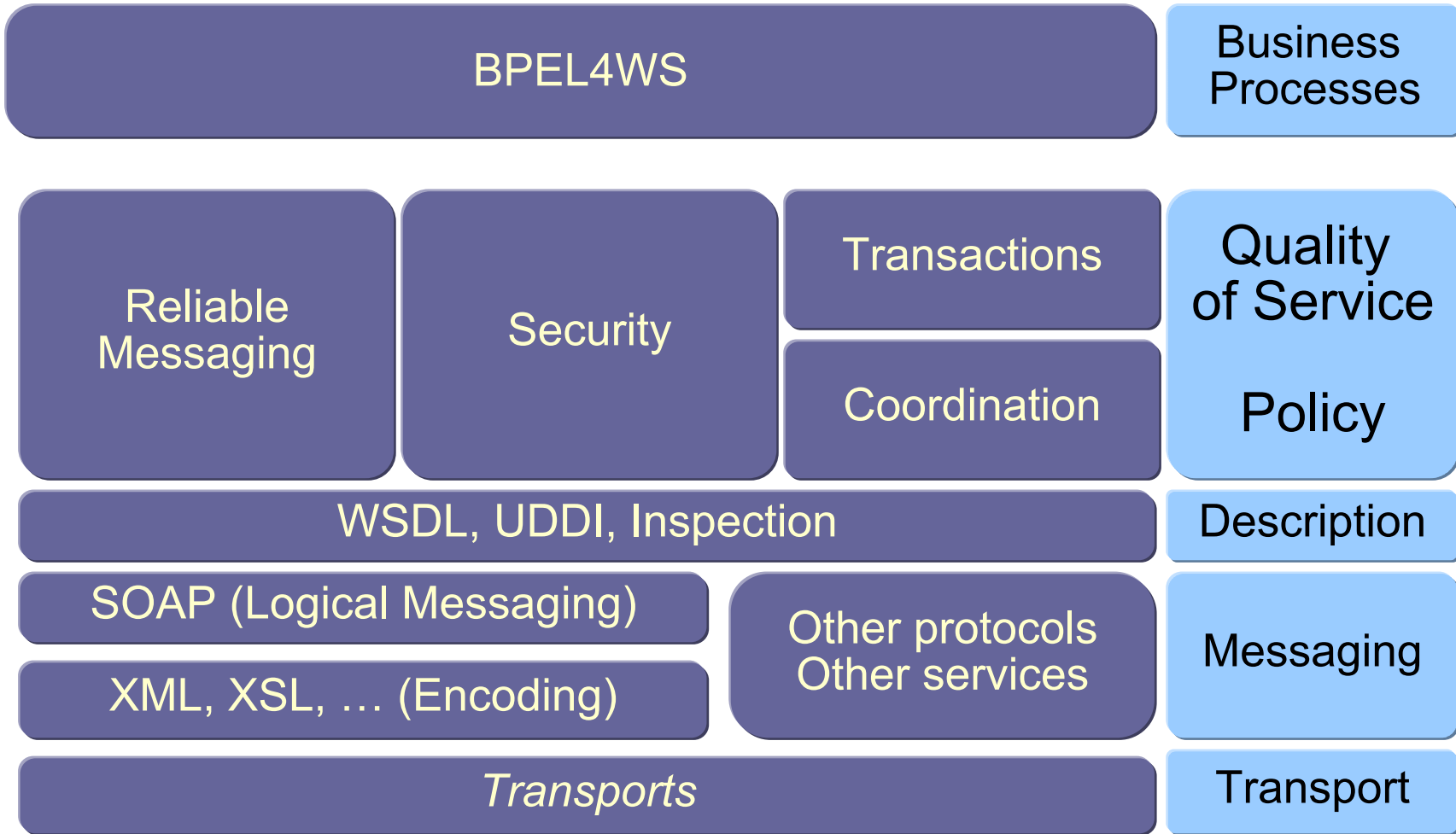
- Data presentation formats focused on presentation, not meaning
- Implied programming model ad hoc
 - Difficult to use state-of-the-art programming models
 - In fact, much programming a throwback to the past
- Weak security
- Behind the curve in supporting multi-domain services models



Enter Web Services

- Standards addressing many aspects of the cube
 - Greater focus on heterogeneity, than previously achieved
 - Support of multiple communication protocols
 - Support for arbitrary object hierarchies
 - Seamless, extensible data model
 - From storage abstraction
 - To communication standard
 - Breadth of focus
 - Supported by economic incentive
- Protocol and Programming Model, alike

Web Services (partial picture)



Some Observations (1)

- There is greater care and creativity in Web Services than in previous approaches:
 - E.g. Service Level Agreement
 - Distributed systems research traditionally discussed *transparency* of services
 - In reality, transparency of services not exactly desired: Service Level Guarantees are desired
 - E.g., The world is inevitably heterogeneous at one or more levels
 - E.g., XML is well-grounded, audacious, & creative
 - E.g., Coming security standards are more complete

Some Observations (2)

- Web Services stds have significant complexity and success requires endurance
 - Performance not trivially achieved
 - Breadth of standards are not easily implemented
 - Complex integration required with other worlds; e.g., J2EE, Windows
 - Architecture and collection of requisite standards still growing in size
 - **Most importantly, great tools are needed**



Combining Tools & Web Services

- At IBM, this point seems to be getting understood.
- For example, IBM's Websphere Studio Tool makes implementing and deploying Web Services easy
- An example of a simple application





Navigator

- UpperCase
 - source
 - UpperCase.java
 - webApplication
 - theme
 - WEB-INF
 - classes
 - UpperCase.class
 - lib
 - ibm-web-bnd.xmi
 - ibm-web-ext.xmi
 - web.xml
 - .classpath
 - .websettings
- UpperCaseEAR
 - META-INF
 - .modulemaps
 - application.xml
 - ibm-application-ext.xmi

Gallery

- Image
- Wallpaper
- Webart
- Sound
- Style Sheet
- Script

```

UpperCase.java x
*/ Author: John Smith
*/
*/ This is a simple example to demonstrate the Web Service
*/ capabilities of Eclipse platform
*/ =====
*/ /

public class UpperCase {
    /**
     * This method converts the input parameter string to
     * uppercase and returns it to the caller
     */
    public String convertToUpperCase(String inputString)
    {
        if(inputString != null)
        {
            return inputString.toUpperCase();
        }
        else
        {
            return null;
        }
    }
}

```

Tasks (0 items)

C	I	Description	Resource	In Folder	Location

New

Close Ctrl+F4

Close All Ctrl+Shift+F4

Save UpperCase.java Ctrl+S

Save UpperCase.java As...

Save All Ctrl+Shift+S

Print Ctrl+P

Import...

Convert Links...

Export...

1 UpperCase/source/UpperCase.java

Exit

- Project...
- Web Project
- Folder
- HTML File
- JSP File
- CSS File
- Image File
- Servlet
- Web Service**
- Other...

```

smith
le example to demonstrate the Web Service
of Eclipse platform
=====
erCase {
    ...method converts the input parameter string to
    * uppercase and returns it to the caller
    */
    public String convertToUpperCase(String inputString)
    {
        if(inputString != null)
        {
            return inputString.toUpperCase();
        }
        else
        {
            return null;
        }
    }
}

```

Deploying Web Service From WebSphere Studio

application.xml

ibm-application-ext.xml

Gallery

- Image
- Wallpaper
- Webart
- Sound
- Style Sheet
- Script

Gallery Outline

Tasks (0 items)

C	I	Description	Resource	In Folder	Location

Tasks Properties Links Thumbnail Styles Colors

Web Service Creation Step

Web Service

Web Service Java Bean Identity
Configure the Java bean as a Web service.

Web service URI:

Scope:

Use static methods
 Use secure SOAP (WebSphere only)

Folder:

ISD file name:

WSDL service document name:

WSDL binding document name:

WSDL schema document name:

< Back Next > Finish Cancel

Navigator

- UpperCase
 - source
 - UpperCase.java
 - webApplication
 - theme
 - WEB-INF
 - classes
 - UpperCase.clas
 - lib
 - ibm-web-bnd.xmi
 - ibm-web-ext.xmi
 - web.xml
 - .classpath
 - .websettings
- UpperCaseEAR
 - META-INF
 - .modulemaps
 - application.xml
 - ibm-application-ext.xmi

Gallery

- Image
- Wallpaper
- Webart
- Sound
- Style Sheet
- Script

Create the Web Service

Namespace

parameter string to caller

string (inputString)

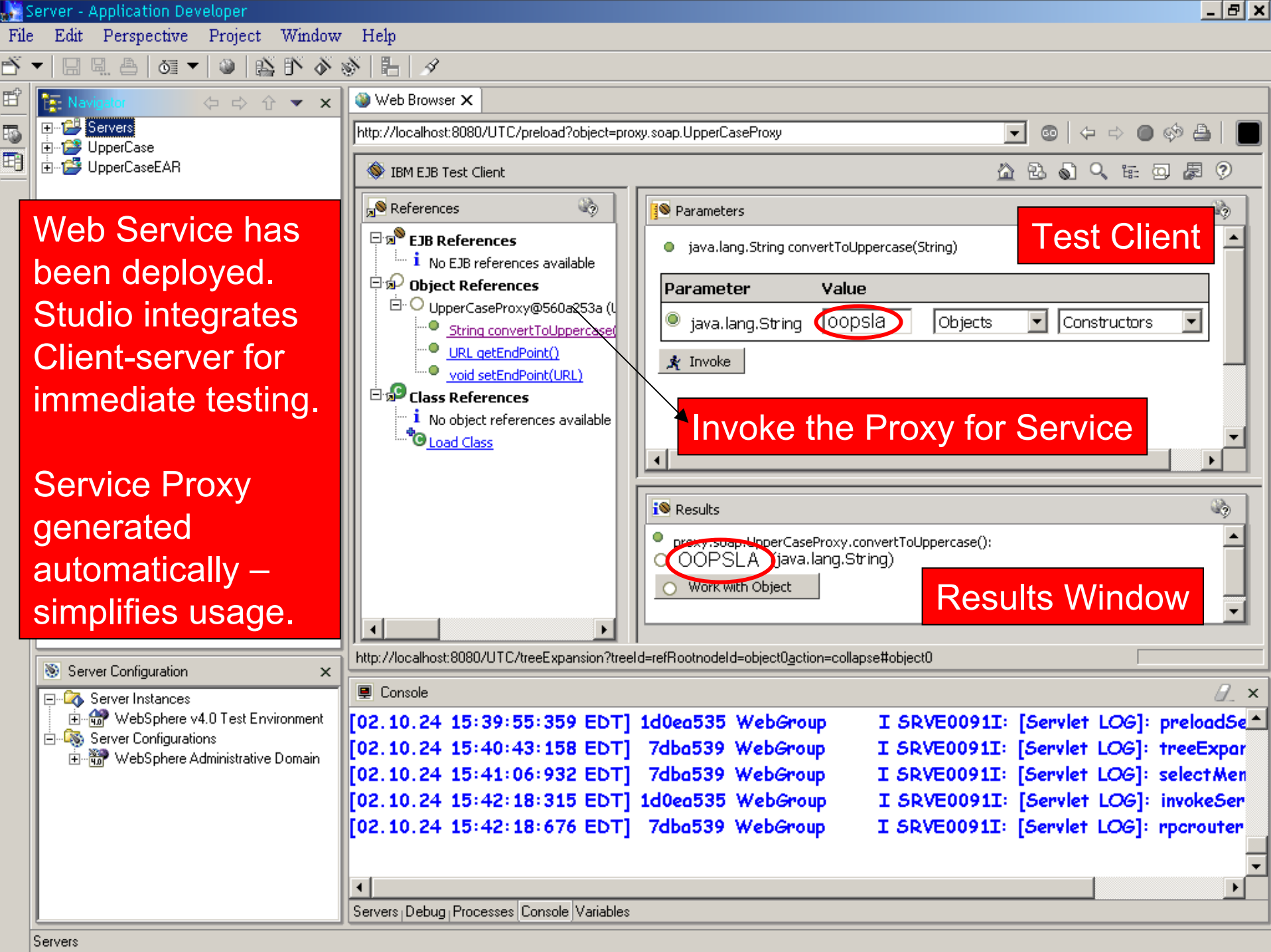
SOAP deployment descriptor

Web Service Descriptions

Data Mapping

Tasks (0 items)

C	I	Description	Resource	In Folder	Location



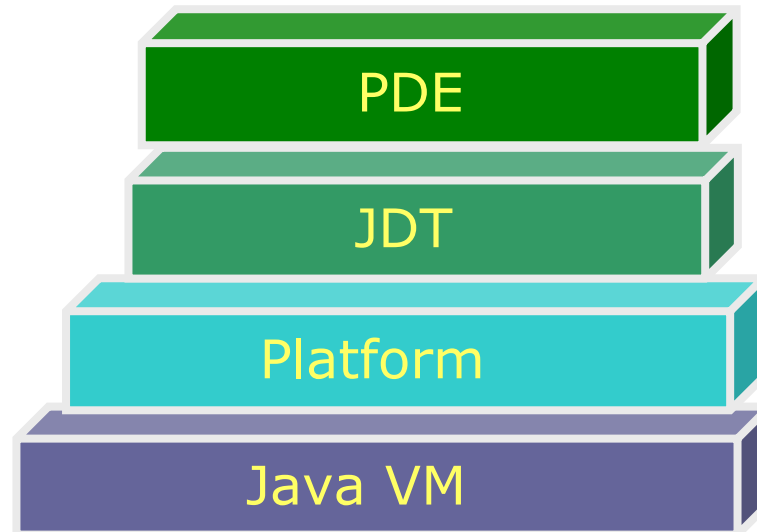
The Tools Dimension

- Many more types of tools will be required to make Web Services easy enough:
 - Service modeling and definition
 - Deployment
 - XML
 - Data modeling and definition
 - Transformation
 - Debugging
 - Performance evaluation
 - Quality of service specification
 - *Tools specialized for particular customer segments*
 - *Industry-specific schemas will often have their own tools*
- ⇒ *Quality tools will be required*

Example: Eclipse tools

www.eclipse.org

- Because of the diversity and creativity required, an open-source development community has great potential
- Eclipse Project has built a universal platform for integrating development tools
- Its open, extensible architecture is based on plug-ins
- Brought to market by commercial offerings
- With *source* licensed for royalty free world-wide distribution

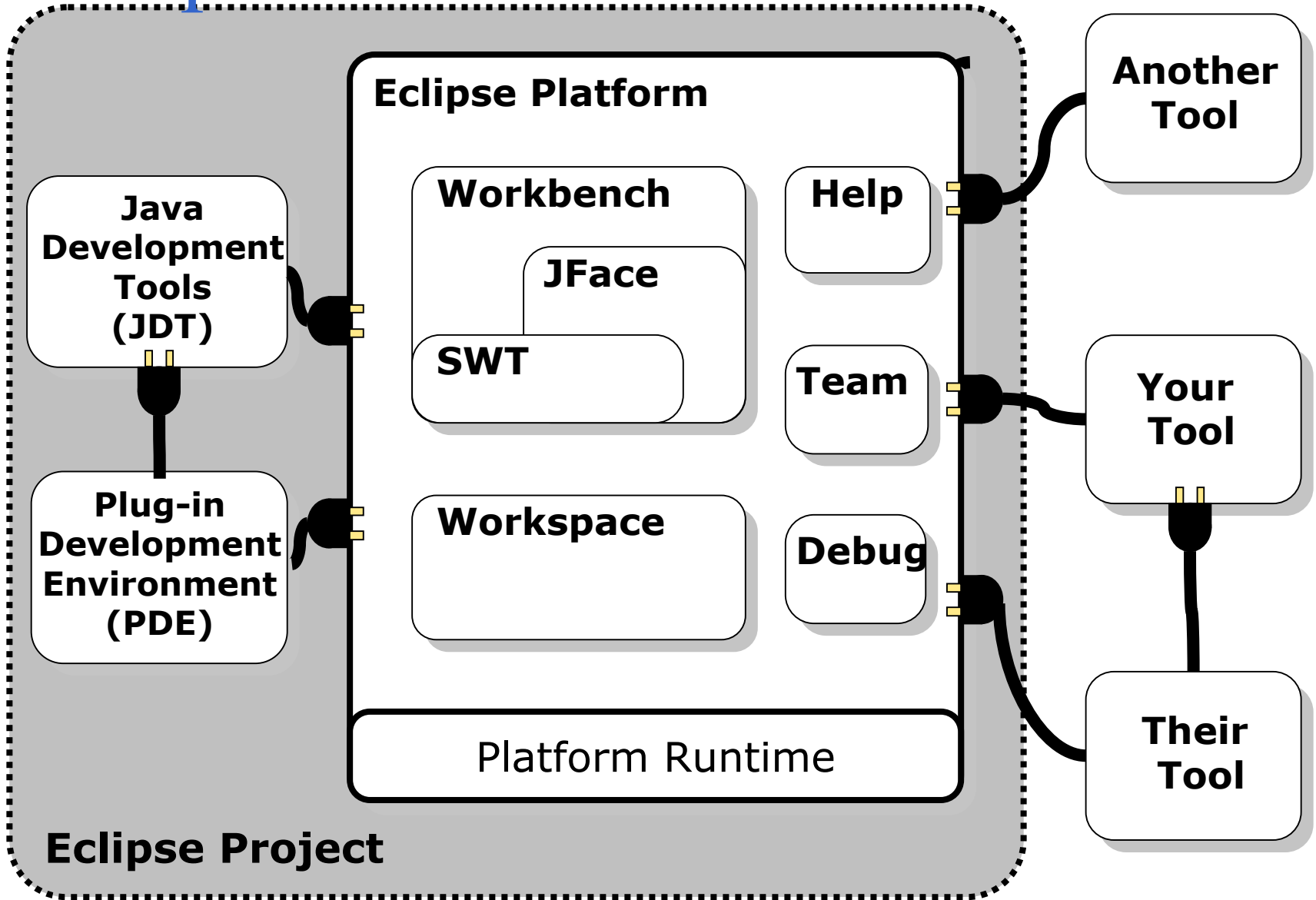


aspector@us.ibm.com

April 9, 2003



Eclipse Overview



Challenges – Core Architecture

- Generality & heterogeneity seem to lead to complexity
- But, we must mitigate complexity
- While allowing scale to much greater dimensions



Additional Challenges Beyond Core

- Quality of Service
- Autonomic Computing
- Components
- Security
- Scale
- Role of Artificial Intelligence



Opportunity

Distributed Systems become “The World’s Operating System”

- The basis for an “on-demand” world
 - e-utilities
 - Continual Optimization
- Revolution in education



About the Kanai Award

- In my opinion, this Award given for work in the science, engineering, and productization of the “orthodox” distributed computing architecture
- I had an important role, but *very many* others had leading roles, also
- I think this Kanai award recognizes their role



Conclusions

- **Distributed computing has been broadly understood for 25+ years**
 - But, the quest requires more technology than had been initially thought
 - Web Services aimed at the breadth of the problem
 - Economic incentive finally exists
 - Complexity of core architecture is interesting
 - Seamless integration of tools and runtime essential
 - Proof of initial theses (e.g., from 1970s) finally to occur
- **There are many key challenges remain**
 - Quality of Service
 - Autonomic Computing
 - Components
 - Security
 - Scale
 - Role of Artificial Intelligence
- **Opportunities abound to change the world**