

# Exposing Business Value with VisualWorks Web Services

**Kirk D. Blackburn**

**Qwest Communications, International**

[KDBlack@qwest.com](mailto:KDBlack@qwest.com)

**Stan Benda**

**Qwest Communications, International**

[SBenda@qwest.com](mailto:SBenda@qwest.com)

**Robert Michaud**

**Qwest Communications International**

[RMichau@qwest.com](mailto:RMichau@qwest.com)



# Introduction

- ❑ **Nice to be back at ESUG!**
- ❑ **ESUG '98 – Smalltalk and Java Interoperability**
- ❑ **This year: Web Services**
- ❑ **My Team**
- ❑ **Project background**

# Structure of Presentation

- ❑ **Web Services Overview**
- ❑ **Problem (Trouble Ticket Service)**
- ❑ **Objectives (Qwest IT)**
- ❑ **Project Approach**
- ❑ **Architecture, Implementation and Mappings**
- ❑ **Lessons Learned**
- ❑ **Summary and Conclusions**




# Why “Exposing Business Value ...?”

# Web Services Definition

- ❑ “XML-based information exchange systems that use the internet for direct application to application interaction. These systems can include programs, objects, messages, or documents. Web services provide a data-independent mechanism to programmatically expose business services on the Internet using standard XML protocols and formats. Web services can be accessed using browsers, but do not require the use of either browsers or HTML. “

# XML Web Services

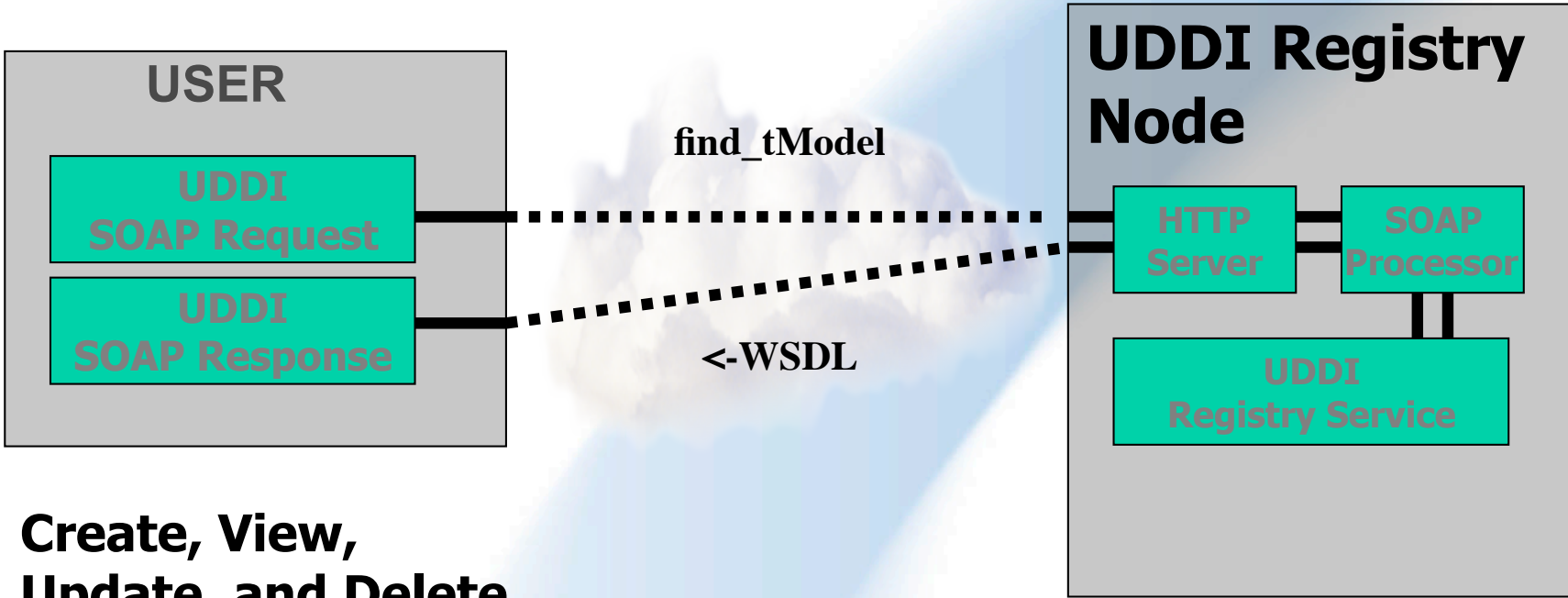
- ❑ Reusable & programmable web components
- ❑ Built on Open Standards 
- ❑ Vendor, platform, & language independent
- ❑ Platform interoperability by providing application to application connectivity
- ❑ Applications will be comprised of web services (Unix Commands)
- ❑ Differences from CORBA?



# Web Service Technologies

- ❑ **Set of XML-based Technologies**
  - ❑ SOAP (simple object access protocol)
  - ❑ UDDI (Universal Description, Discovery and Integration)
  - ❑ WSDL (Web Service Description Language)
  - ❑ XSD (XML Schema)
  
- ❑ **Hype (promise to make EAI easy)**
  - ❑ Integration of heterogeneous systems
  
- ❑ **Reality (security, Xactions, workflow)**

# UDDI and SOAP



**Create, View,  
Update, and Delete  
registrations**

**Implementation-  
neutral**





# Registry APIs (SOAP Messages)

## Inquiry API

- **Find things**
  - find\_business
  - find\_service
  - find\_binding
  - find\_tModel
- **Get Details about things**
  - get\_businessDetail
  - get\_serviceDetail
  - get\_bindingDetail
  - get\_tModelDetail

## Publishers API

- **Save things**
  - save\_business
  - save\_service
  - save\_binding
  - save\_tModel
- **Delete things**
  - delete\_business
  - delete\_service
  - delete\_binding
  - delete\_tModel
- **security...**
  - get\_authToken
  - discard\_authToken



# VisualWorks UDDI Search Tool

The screenshot shows a window titled "UDDI Search" with a standard Windows XP-style title bar. The main content area has a header that reads "Search the UDDI Registry for Web Services". Below this, there is a "URL:" label followed by a dropdown menu currently showing "Microsoft\_Test\_Site\_UDDIInquiry v1". Underneath is a "Search for:" section containing a text input field with "microsoft", the word "in", another dropdown menu showing "Business name", and a "Go" button. A large, empty rectangular area with scrollbars occupies the lower half of the window. At the bottom right of the window is a button labeled "Business Details".

# HTTP Settings Tool

HTTP Settings

Keep Alive Connection       Redirect Request

Proxy Configuration

Use Proxy

Address:

Port:

Proxy User:

Advanced

Add User...      Accept      Help

# WSDL

## What is a Web Service Description Language?

It is a “**simple**” XML document that contains set of definitions to define a web service \*

Major elements of a WSDL document are:

Element	Defines
<portType>	The operations performed by the web service
<message>	The messages used by the web service
<types>	The data types used by the web service
<binding>	The communication protocols used by the web service

\* We will discuss “**Simple**” later in the presentation



# WSDL Example

```
<message name="getTermRequest">
  <part name="term" type="xs:string" />
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string" />
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest" />
    <output message="getTermResponse" />
  </operation>
</portType>
```

**\* We will compare WSDL with IDL later in the presentation**



# IDL and WSDL and ST Signatures

## IDL

```
void createTicket(inout TroubleTicket troubleTicket, out ErrorStruct error);
```

## WSDL

```
<operation name="createTicket" parameterOrder="target troubleTicket  
  error">  
  <input message="tns:createTicket"/>  
  <output message="tns:createTicketResponse"/>  
</operation>
```

## Smalltalk

```
createTicket: aTicket error: errorParameter
```



# IDL and WSDL Complex Types

## IDL

```
#pragma class ErrorStruct Qwest.WebServices.TroubleTicketErrorStruct
struct ErrorStruct {
    long code;
    string description;
};
```

## WSDL

```
</xsd:complexType>
  <xsd:complexType name="troubleTicketErrorStruct">
    <xsd:sequence>
      <xsd:element maxOccurs="1" minOccurs="1" name="code" type="xsd:int"/>
      <xsd:element maxOccurs="1" minOccurs="1" name="description" nillable="true"
        type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
```



# Problem Description



# Qwest IT Objectives

- ❑ Everything will be a Web Service
- ❑ “all businesses will provide Web Service access to existing Qwest applications without re-writing the existing applications. “
- ❑ “We will use tools to create web services from existing applications and services without changing the underlying implementations....”
- ❑ Short-term, web services will tend to be developed with BEA’s EJB technology or similar technology. We will use the C# and Microsoft.NET in addition to the EJB technologies.



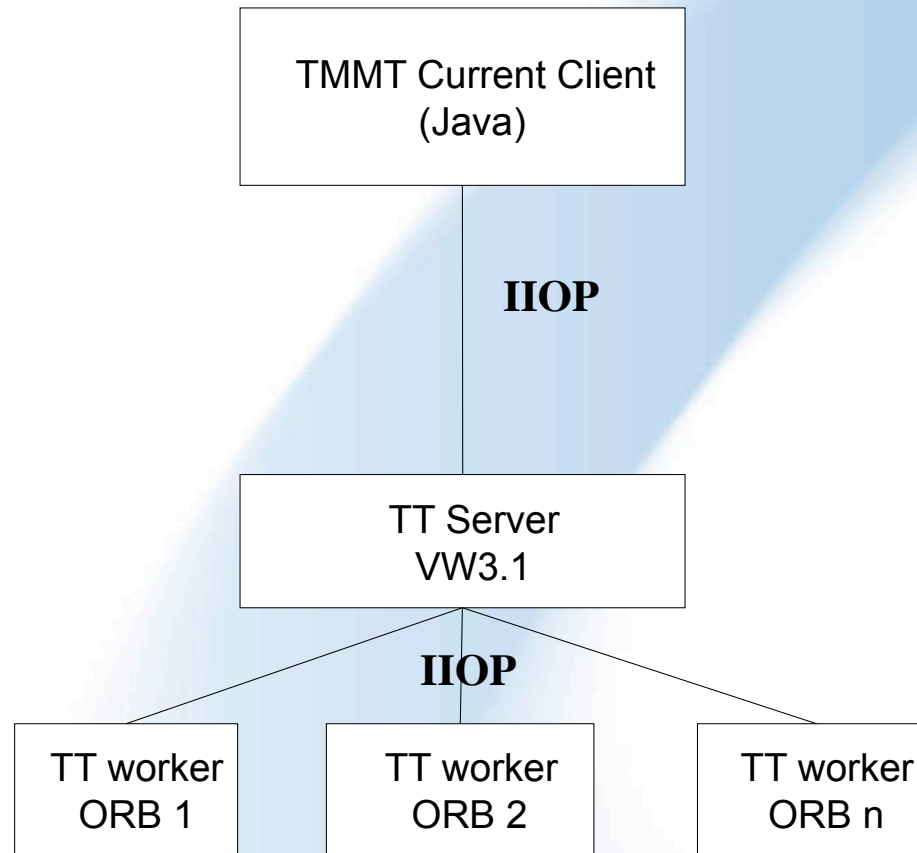
# Problem (early 2002)

- ❑ **How to expose Trouble Ticket Service as Web Service**
- ❑ **Trouble Ticket Service was CORBA-based**
  - ❑ **findTicketByID (in Int out Ticket);**
  - ❑ **createTicket(inout TroubleTicket troubleTicket, out ErrorStruct error);**

# Existing Application

- ❑ **Trouble Ticket Application**
- ❑ **Promia ORB**
- ❑ **VW3.1 Implementation**
- ❑ **Java clients**
- ❑ **Interface defines protocol for creating and finding trouble tickets**

# Trouble Ticket Service Architecture



# Options for Web Service Enabling

- ❑ Build on limited XML and HTTP support in VW3
- ❑ Use a shareware SOAP to CORBA Bridge \*
- ❑ Use .NET and VisualWorks Com Connect
- ❑ Use VW7 Beta Bits

\* [http://soap2corba.sourceforge.net/html/fo\)design.html](http://soap2corba.sourceforge.net/html/fo)design.html)



# Chose to Use VW7 Beta

❑ All Smalltalk Solution



❑ We Wanted VW7 Experience

❑ Partner with Cincom to Beta Test VW7

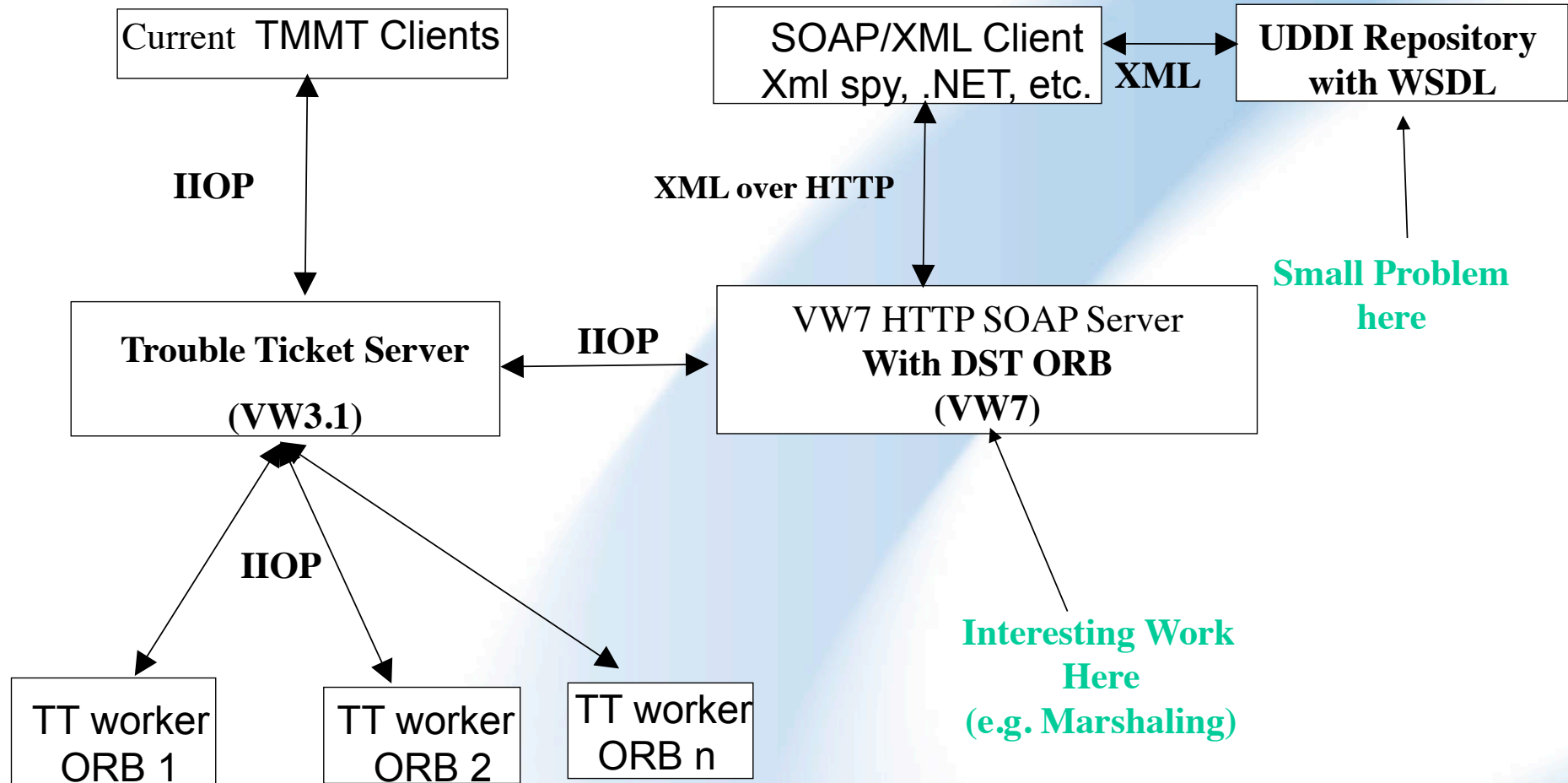


# VW7 Support for Web Services

- ❑ SOAP Server
- ❑ WSDL Client
- ❑ UDDI
- ❑ XSD
- ❑ XMLMarshaling
- ❑ XMLToObject Bindings
- ❑ WSDL tool (as of 8/20/02 ... **last week**)



# VW7 Option





# Where does WSDL come from?

- ❑ Did not have WSDL
- ❑ Had IDL
- ❑ Tools exist to create WSDL from
  - ❑ Java
  - ❑ .C# classes in .NET
- ❑ So...

# Found Cape Connect

- ❑ Cape Connect from Cape Clear Software \*
- ❑ Generates WSDL from IDL !
- ❑ Mainly for Java and C#
  - ❑ Generates WSDL from Classes
- ❑ Did an ok job of generating WSDL from our IDL
  - ❑ (2.0 CORBA, IDL imports, etc)
- ❑ Still had to modify the WSDL (namespaces, etc)



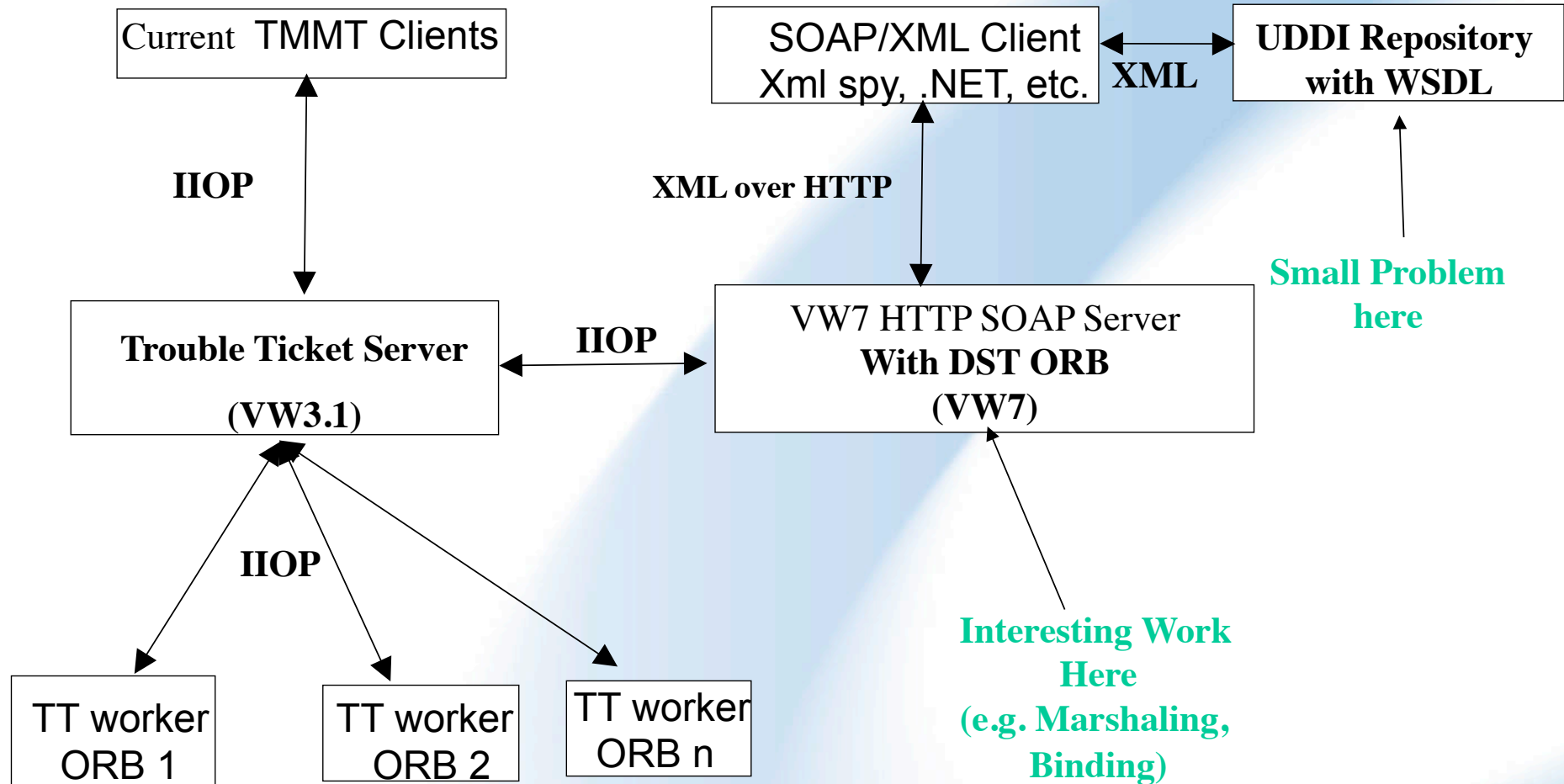
\* [www.capeclear.com](http://www.capeclear.com)



# Steps to Create a Web Service

- ❑ **Create WSDL**
- ❑ **Create Class for the Web Service**
- ❑ **Create Classes for the XSD types**
- ❑ **Write XML to Smalltalk Bindings**
  - ❑ **WSDLClient creates default bindings**
  - ❑ **Can then modify to bind XSD types into Objects instead of default structs**

# VW7 Option



# Performance

## ❑ Preliminary

- ❑ CORBA – 30 to 100s of message sends/sec.
- ❑ SOAP – 800 msec per request
  - ❑ VW7 vs. .NET

# New VW7 WSDL Capabilities

## Lessons Learned

- ❑ **WSDL is not easy to write**
- ❑ **WSDL takes time to learn**
- ❑ **VW7 SOAP works. Can interop with ORBs**
- ❑ **VW7 Web Services support saved us time**
- ❑ **ObjRefs don't exist in Web Services**
- ❑ **Effort was well supported by Cincom**

## Summary and Conclusions

- ❑ **VW7 provides good support for Web Services**
- ❑ **Support requires more coding than say .NET**
- ❑ **Interop issues with .NET**
- ❑ **VW7 WS have potential to leverage entire Domain**
- ❑ **Selling Web Services implemented VW still an obstacle (internally)**
- ❑ **Web Services in VW may help overcome arguments that limited viability of VW in Web-based software architectures**

