

MACHOLD
Systemhaus 21

V4 Connect IMS applications with Java/.NET

The easy and working way for IMS connection to Java/.NET/... and SOA integration

LIVE Demo included using Ivory

Axel Rittershaus, Machold Systemhaus 21, Stuttgart

Why SOA for the mainframe

- Today the mainframe is often seen as a fast but inflexible system if you have to integrate it into modern architectures
- Not only technological boundaries: There is more than a world between the mainframe and the java developers – it's like living on different planets
- BUT if you enable your mainframe to be a part of a SOA and any java developer can use a mainframe based functionality without any knowledge what's really going on (he only calls a Web Service) the existing borders will break down
- AND if you enable your mainframe developers to use their broad knowledge of the „old“ existing (but perfect working) applications AND the business logic and they produce Web Services without having to learn SOAP, Java, XML, ... you can accelerate everything

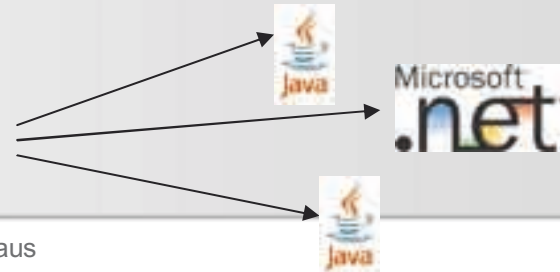
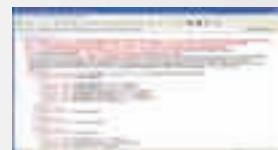
How to connect IMS applications with Java or .NET

- The traditional approach is a direct connection between a specific IMS transaction and a Java or .NET application
 - Requires Mainframe- and Java/.NET skills while creating the connection
 - Creates one more 1:1-connection between two systems
 - Has to handle data format issues, transaction issues, ...
 - Different technologies and lack of standards causes more problems
 - Not reusable for other applications
 - Each interface only usable for one technology

How to connect IMS applications with Java or .NET

- The SOA-approach

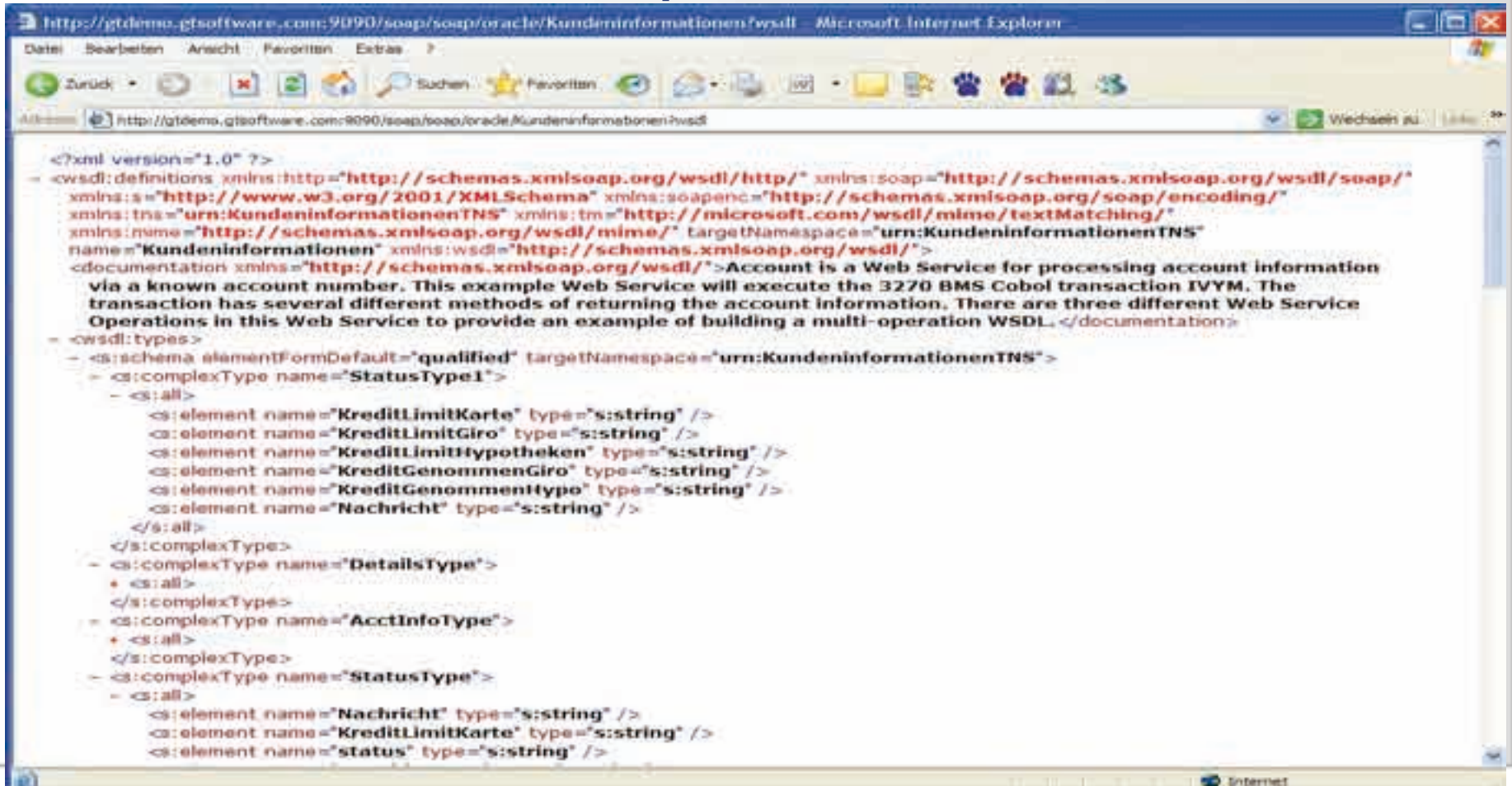
- Define the service once (on mainframe side) and publish it
- Interface is based on standards (WSDL, SOAP, XML, ...)
- Mainframe and Java/.NET developers discuss the required functionality instead of long lasting discussions about technological issues and pros and cons of their preferred technology
- Reusable for many other applications
- Reusable for different technologies (since Web Services can be used by Java, .NET, C#, ...)



Web Services for mainframe integration

- Web Services = standard based interface
 - Invoking using HTTP/HTTPS
 - Data exchange using SOAP = XML
 - Any kind of functionality can be implemented (read, write, delete, update, with/without return data)
 - Very easy integration into Java-based applications
 - Very easy integration into other technologies (.NET, SAP, ...)
 - Each Web Service can be used by several other applications without additional effort (but take care about governance!)

Web Service – WSDL example



```

<?xml version="1.0" ?>
- <wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:tns="urn:KundeninformationenTNS" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:name="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="urn:KundeninformationenTNS"
  name="Kundeninformationen" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">Account is a Web Service for processing account information
  via a known account number. This example Web Service will execute the 3270 BMS Cobol transaction IVYM. The
  transaction has several different methods of returning the account information. There are three different Web Service
  Operations in this Web Service to provide an example of building a multi-operation WSDL.</documentation>
- <wsdl:types>
- <xs:schema elementFormDefault="qualified" targetNamespace="urn:KundeninformationenTNS">
- <xs:complexType name="StatusType1">
- <xs:all>
  <xs:element name="KreditLimitKarte" type="s:string" />
  <xs:element name="KreditLimitGiro" type="s:string" />
  <xs:element name="KreditLimitHypotheiken" type="s:string" />
  <xs:element name="KreditGenommenGiro" type="s:string" />
  <xs:element name="KreditGenommenGiro" type="s:string" />
  <xs:element name="Nachricht" type="s:string" />
  </xs:all>
</xs:complexType>
- <xs:complexType name="DetailsType">
+ <xs:all>
</xs:complexType>
- <xs:complexType name="AcctInfoType">
+ <xs:all>
</xs:complexType>
- <xs:complexType name="StatusType">
- <xs:all>
  <xs:element name="Nachricht" type="s:string" />
  <xs:element name="KreditLimitKarte" type="s:string" />
  <xs:element name="status" type="s:string" />

```

But....

„You can't buy SOA from a vendor!
SOA is a lifestyle“

Anne Thomas Manes (VP, Burton Group)

Is there any solution for the
SOA integration of mainframes?

Residing on the mainframe?

Easy to use?

Supporting Web Services standards?

Ivory principles

 Ivory™ Studio



Development environment
on XP, W2000, ...

 Ivory™ Server

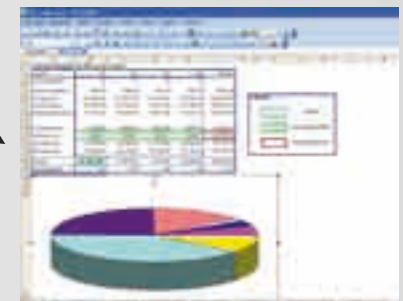


Server component for IBM mainframes
using MVS, z/OS, VSE, OS/390

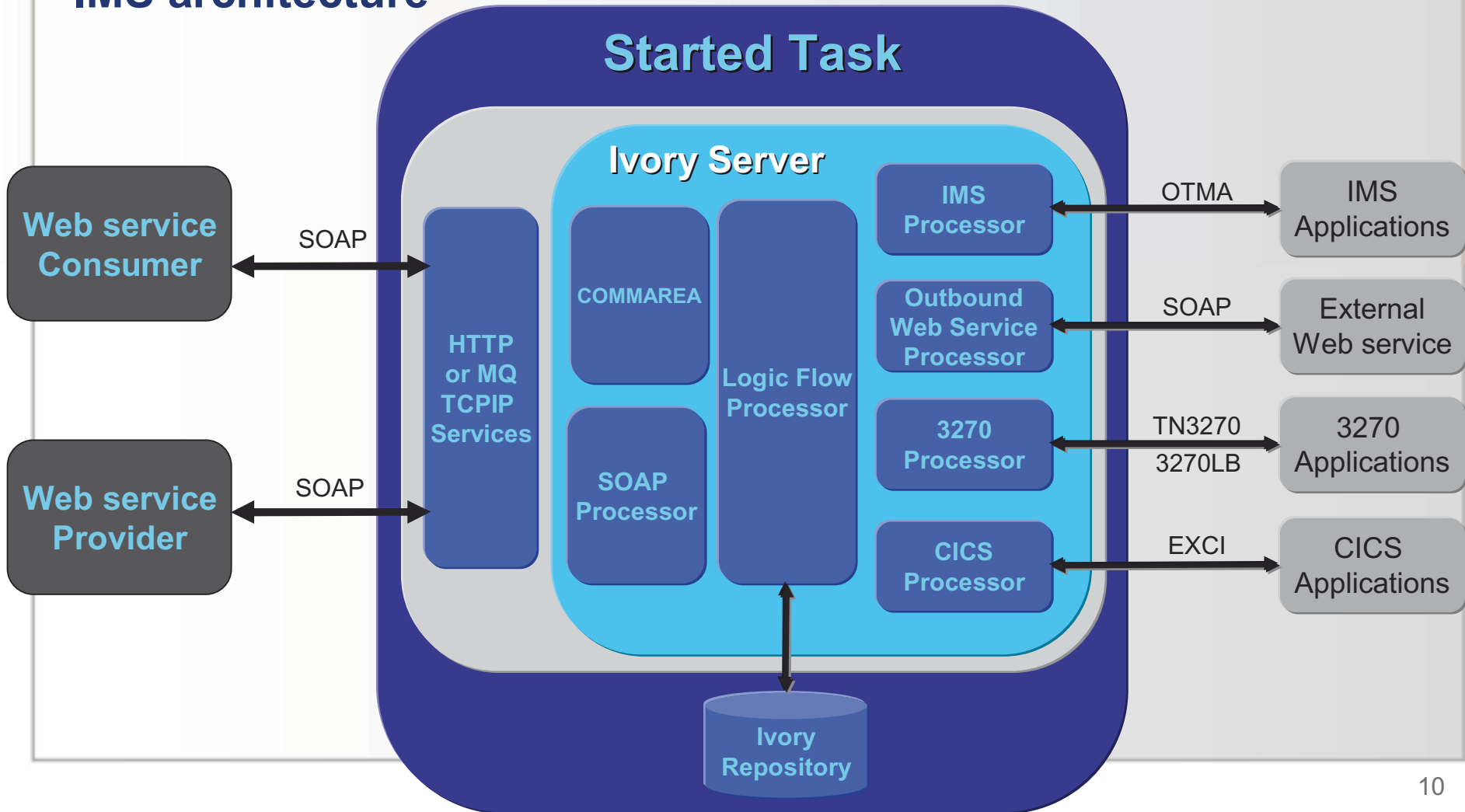
SOAP

SOAP

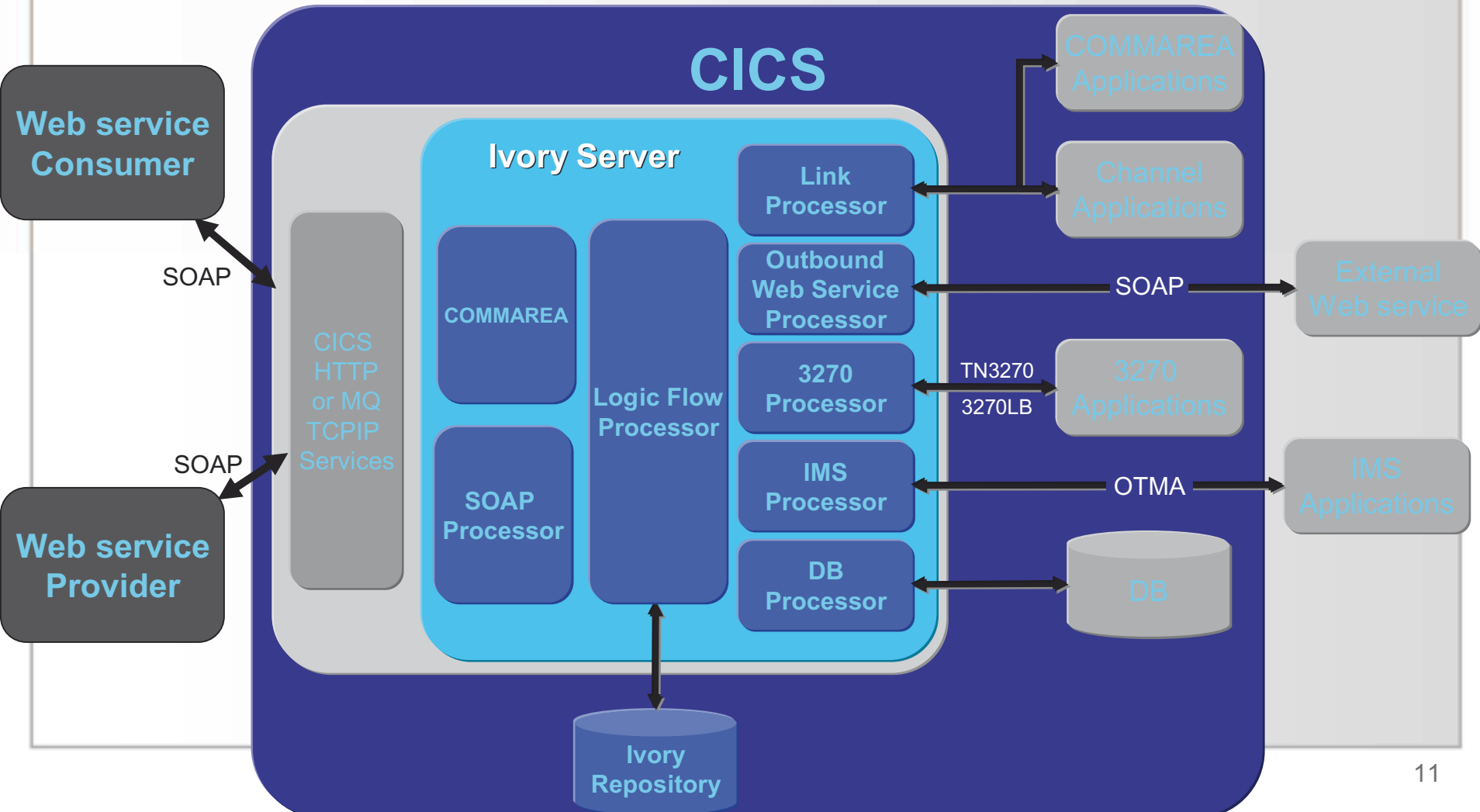
SOAP



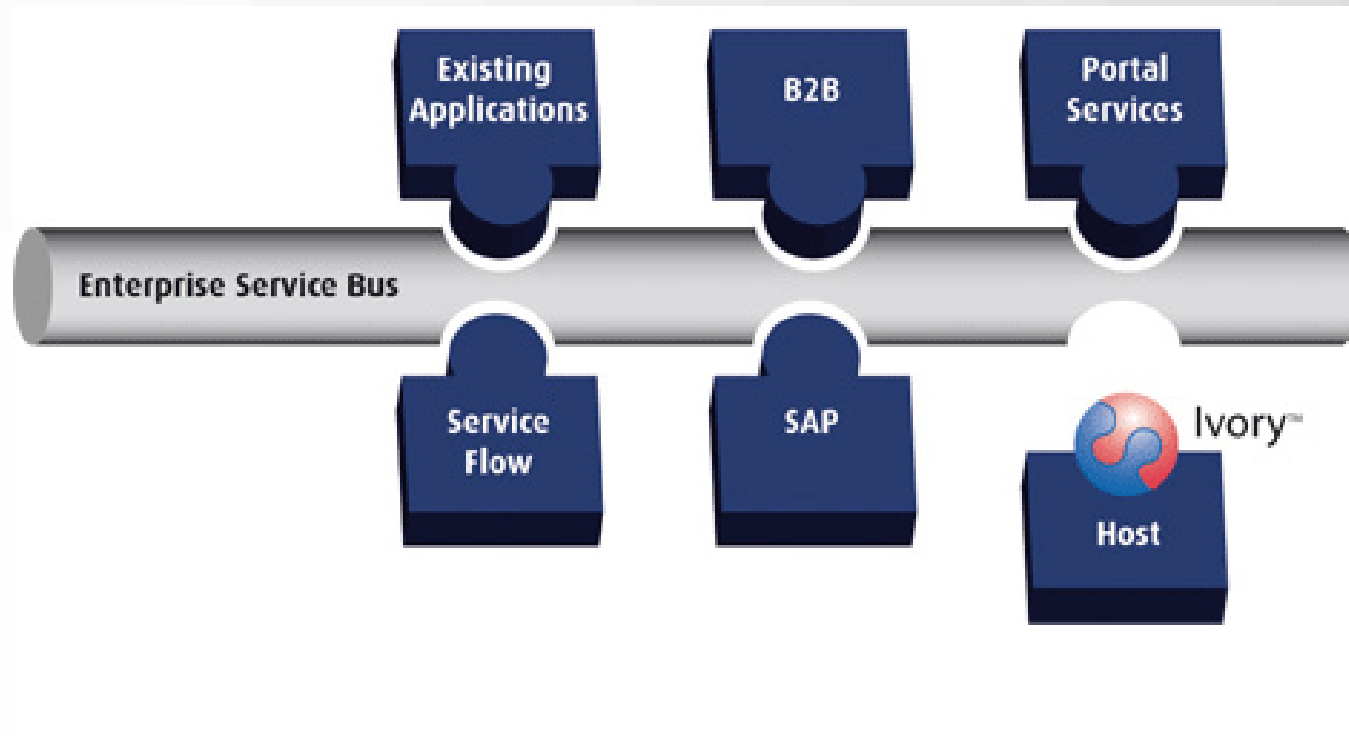
IMS architecture



CICS architecture



SOA architecture using Ivory



Example 1: Financial institute

- The issue:
Demand for a very fast realisation (less than 6 month) of a new user front end with java for an existing 3270 based application
- The situation:
Not much time for implementation, existing application can be used but has to be modified for additional requirements, not enough budget or time for traditional approach.
And: Another project was „on hold“ following the actual project and also needing several of these functions

Example 1: Financial institute

- The approach:
A comparison of the traditional approach (self developed solution, IMS connect) with an Ivory based solution during a PoC pointed the advantages of Ivory out.
The main issues have been the development speed, the easy to use IDE, the reusability with Web Services and the options for composite web services.
- The result:
All involved people (Java developers AND mainframe developers) could communicate very easy about the required interfaces, the mainframe developers were able to react very fast on changing requirements from the front end guys and the Java developers could use existing frameworks (Axis) and increase the speed to use the mainframe functions via Web Services (Ivory processing time request-response < 0.03 sec)
- Project has been delivered in time and quality

Example 2: Federated Insurance Companies

- Multi-lines Mutual Insurance Company
- \$1.4 Billion in Premium
- Operations in 48 States
- 2,600+ Employees
- Home Office in Owatonna, Minnesota



Example 2: Federated Insurance Companies

Problem

- Make legacy services available to new composite applications
- Developers spending 50%+ time on “plumbing”
- Slowing development efforts
- Reuse opportunities lost

Objective

- Refocus on the business problem
- Expose and consume Web Services
- Reuse legacy when possible ...or build new
- *Active* approach to mainframe SOA

Example 2: Federated Insurance Companies

Results

- Return on investment in Ivory already achieved
 - ◉ Developers now spend <5% on infrastructure
- 14 Mainframe Developers now Service Developers
 - ◉ ***Only 2 hours informal training for 1 person***
- Snapshot production stats
 - ◉ Over 100 services
 - Over 255 Web service operations
 - Over 270 COBOL programs
 - ◉ Serving 10 Applications across 7 business areas
 - ◉ Over 300,000 Ivory Web services requests / day

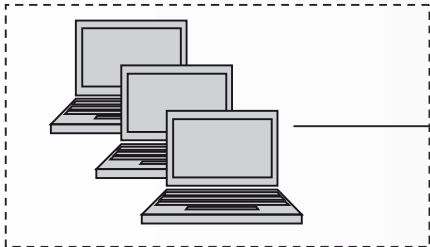
Ivory at a glance

- Delivers existing and proven (!) mainframe business functions to any kind of .NET, J2EE, ... Web Services-consuming application
- Two components:
 - Graphical IDE (Ivory Studio)
 - Mainframe based server component (IMS: Started task, CICS: Transaction)
- Combination of multiple mainframe transactions into one single Web Service practicable
- Import wizard for COBOL, PL/1, Copybooks, Includes, BMS maps, MFS, ...
- Delivers the real advantages of mainframe like performance, scalability, stability, security and the existing business logic to a SOA

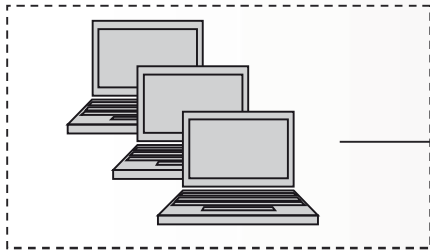
- NO additional middleware (hard- and software) required
- NO modification of your existing applications
- NO coding
- NO additional skills required for XML, SOAP, .. required

Realisation: Step 1

User(group) 1



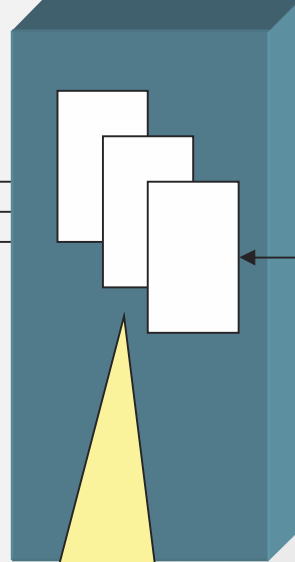
User(group) 2



User(group) x



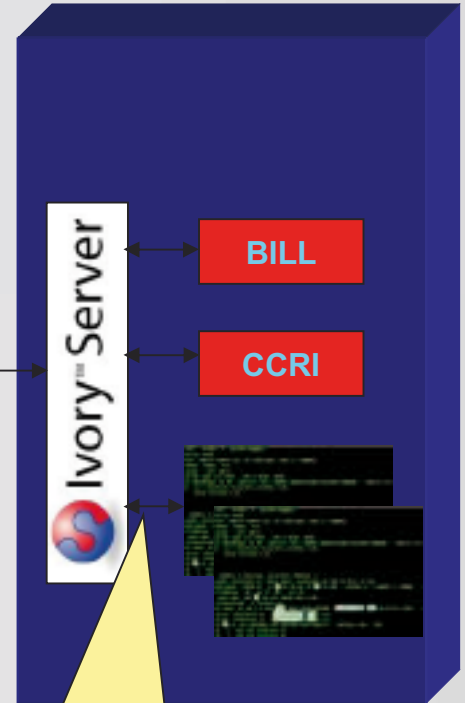
Web/Applic-
Server



Java application
using JSPs, ...
ONLY user interface!

SOAP

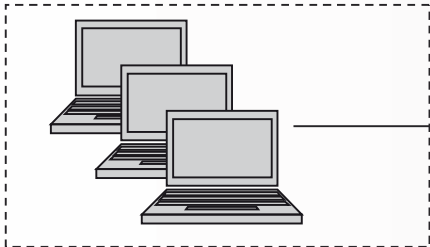
Host



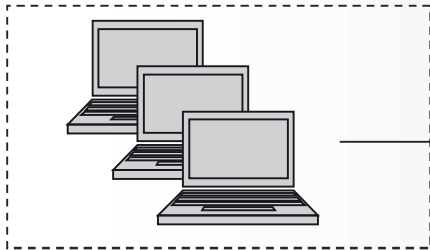
Only mainframe
communication

Realisation: Step 2

User(group) 1



User(group) 2

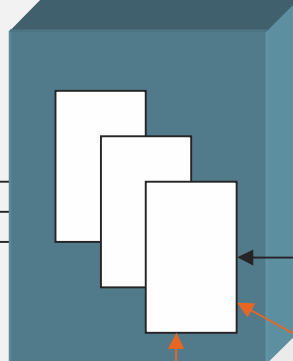


User(group) y

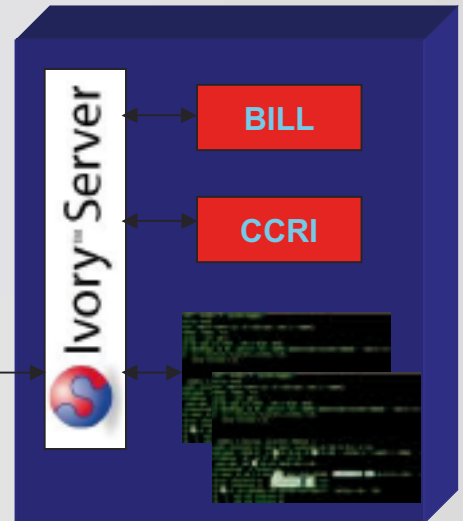


Maybe external users

Web/Applic-Server

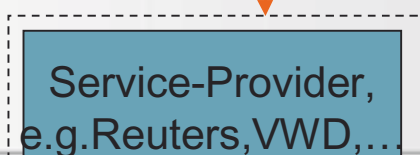


Mainframe



Other systems like
Unix, Portal, SAP

External Provider



Service-Provider,
e.g. Reuters, VWD, ...

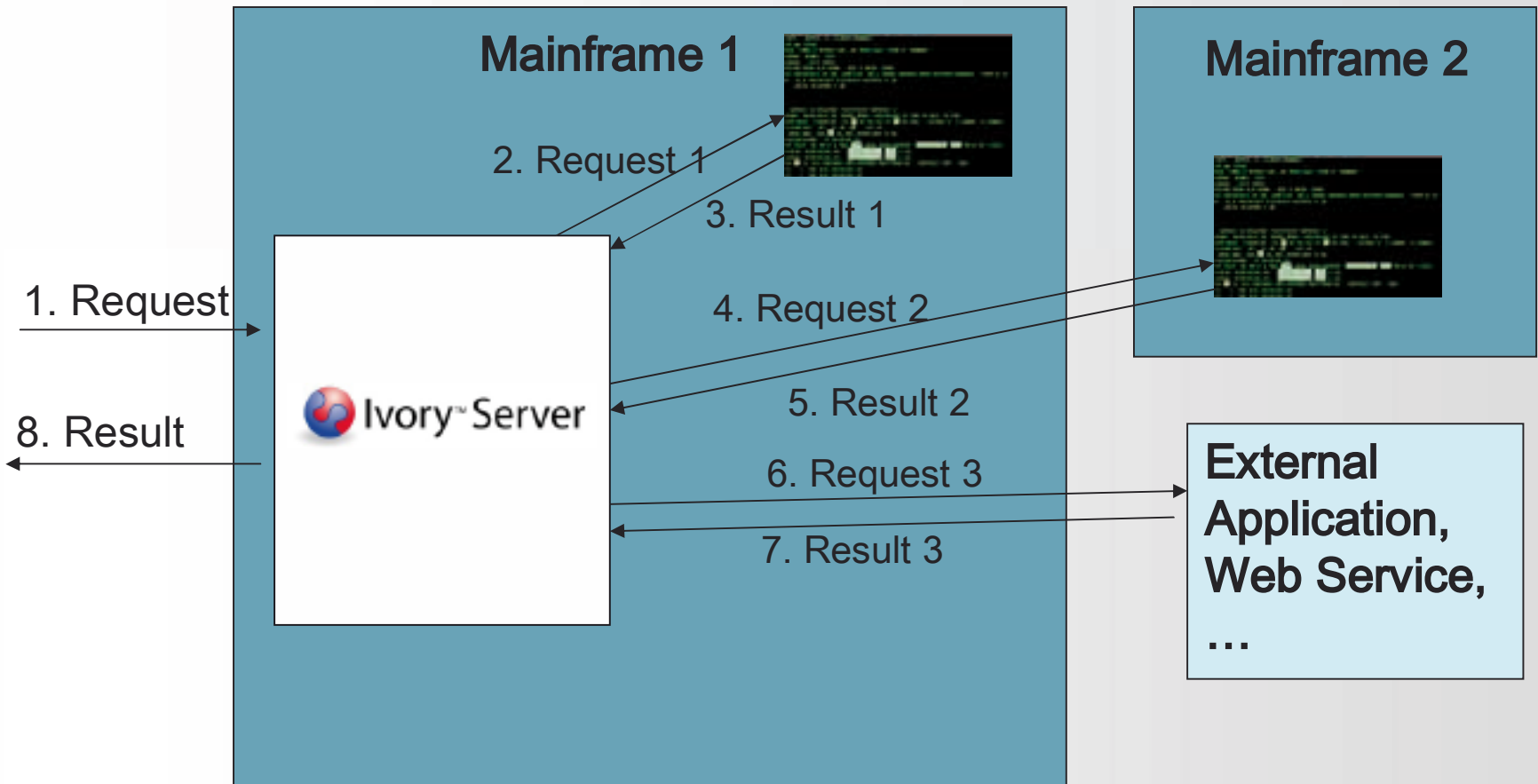
SOAP

SOAP

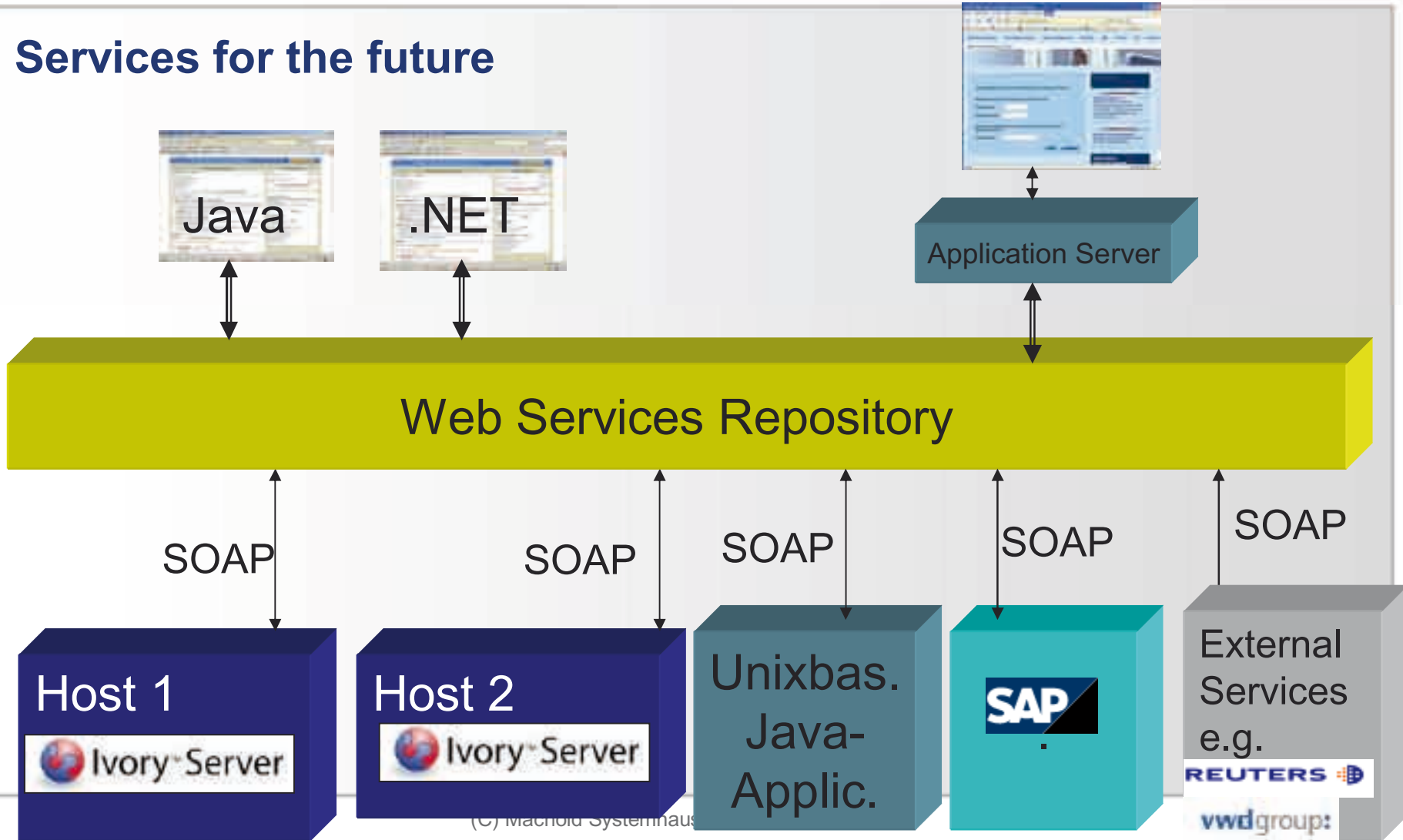
SOAP

Adding other functions to the user interface can be done without blocking the system. An existing function can also be redirected to another provider without letting the user know.

Ivory – why composite Web Services are important



Services for the future



LIVE Demo

using a mainframe located at GT Software office
in Atlanta, GA, USA

hosting Ivory Server and IMS transactions

Ivory in Europe

Responsible for Germany, Austria, Switzerland, Danmark:

- Machold Systemhaus 21

Nordbahnhofstr. 17

70191 Stuttgart

Germany

+49 711 25772-0

www.machold.de

www.rapid-soa.de

Other countries: www.gtsoftware.com