Oxygenating The Web Service Platform

Three SOA Case Studies understanding what to use - where

Paul Fremantle
Chief Technology Officer
WSO2 Inc

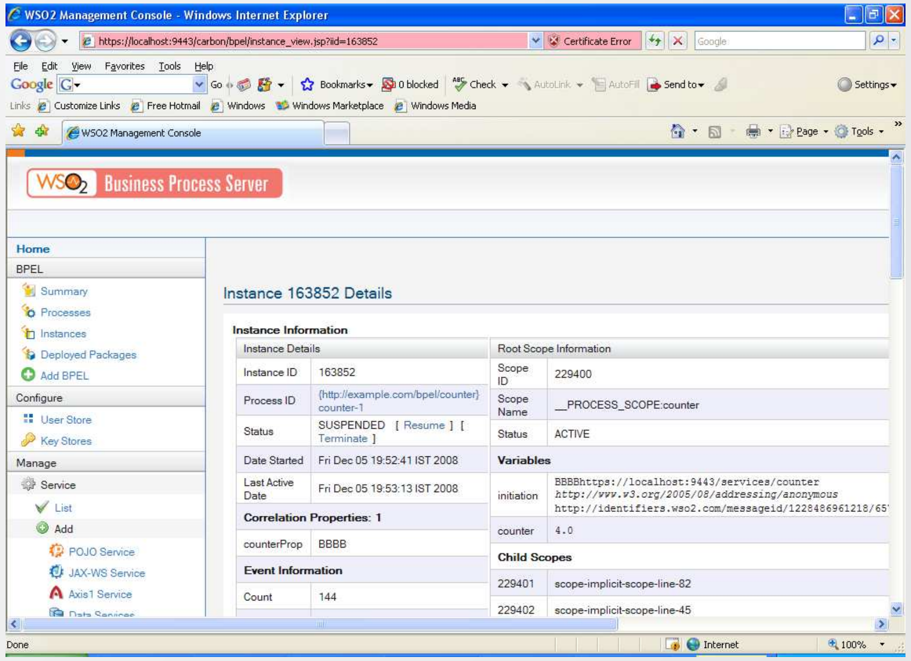
Introduction

- Paul Fremantle, CTO, WSO2
  - Co-Chair, OASIS WSRX TC
  - VP, Apache Synapse
  - Previously STSM in WebSphere Architecture

- This is based on projects I've worked on at WSO2
  - Case study #1
    - Integrating legacy systems for reporting at Concur
  - Case study #2
    - Building a National SOA – OIO SOI
  - Case study #3
    - Using SOA to integrate IT Management systems
  - **Anti-study**
    - *Some lessons learnt NOT on WSO2 projects!*

A very short plug for WSO2

- Open Source SOA Startup
  - Since 2005
- A complete SOA platform available under the Apache License
- WSO2 Carbon – OSGi-based runtime including
  - ESB
  - Service Hosting – Web Services Application Server
  - Data Services
  - Registry / SOA Governance
  - Business Process Server
- No Gimmicks / Gotchas
- Full 24x7 support
- Training and Consultancy
- **Hear more tomorrow**
  at 16:45 SkillsMatter booth



Instance 163852 Details

Instance Information

Instance Details	
Instance ID	163852
Process ID	{http://example.com/bpel/counter}counter-1
Status	SUSPENDED [Resume] [Terminate]
Date Started	Fri Dec 05 19:52:41 IST 2008
Last Active Date	Fri Dec 05 19:53:13 IST 2008
Correlation Properties: 1	
counterProp	BBBB
Event Information	
Count	144

Root Scope Information	
Scope ID	229400
Scope Name	__PROCESS_SCOPE:counter
Status	ACTIVE
Variables	
initiation	BBBBhttps://localhost:9443/services/counter *http://www.w3.org/2005/08/addressing/anonymous* http://identifiers.wso2.com/messageid/1228486961218/65
counter	4.0
Child Scopes	
229401	scope-implicit-scope-line-82
229402	scope-implicit-scope-line-45

Oxygenating The Web Service Platform

Case study 1

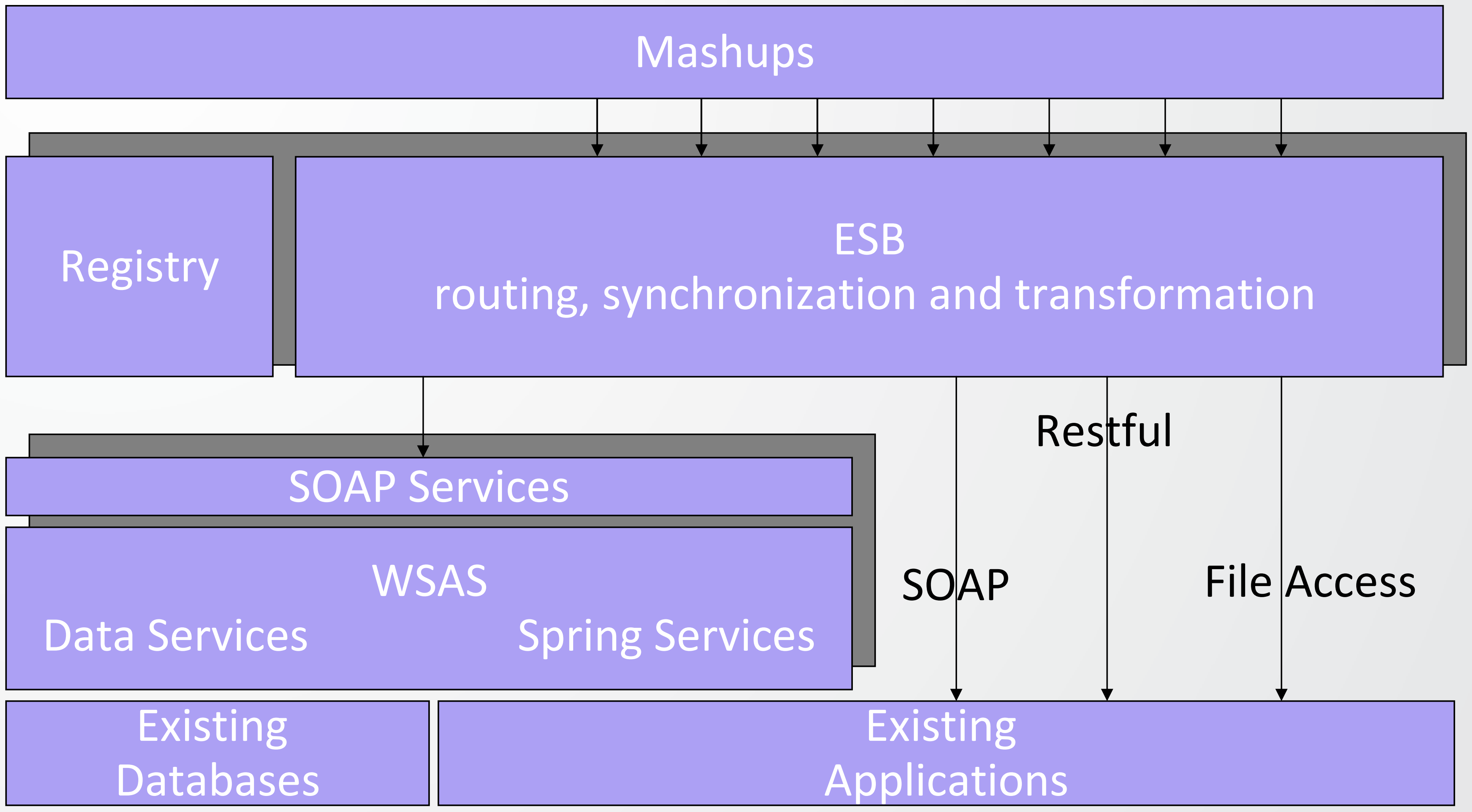
Concur™
click. done.™

Integration at the glass

Concur

- Concur is an online expense management company
  - >$200m revenue
  - Multiple legacy systems:
    - Customer Relationship Management
    - ERP
    - Sales Force Automation
    - In house HR employee application
  - Main requirement – enable better reporting across applications
    - Internal project only – not in the direct flow of external customer systems
  - Needed an approach that supported:
    - Iterative development
    - Support changes to the underlying systems
    - Flexible

Architecture

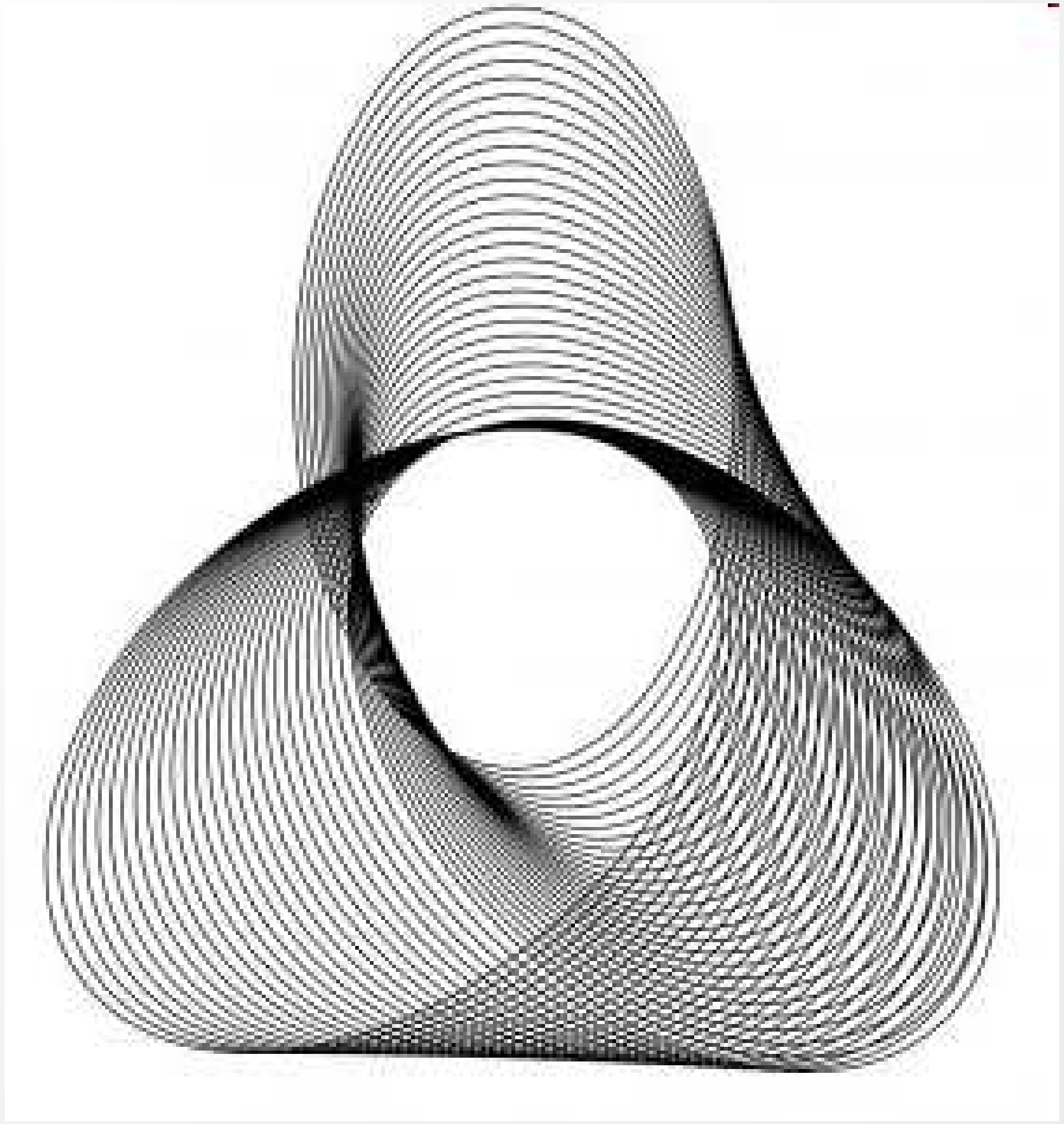


Bug Tracking / ITIL Ticket / CRM / SFA / HR / (10 systems in all and growing)

Technical details

- Everything deployed on Windows 2003 running on VMWare
- Internal systems so limited security
  - Basic authentication
  - Some use of digital signature
- Running in a blade server to simplify test and scaling
  - Currently Hot/Cold but moving to Hot/Hot
- ~75,000 transactions a day
  - 95% SOAP, 5% Restful at this point
- WSDLs and Schema’s stored in WSO2 Registry
  - Embedded in the ESB
- Currently 18 services across 10 backends with 120 operations
  - Growing
- Looking at moving to a more event-based approach in the future

Iterative development



Project Approach

- Planned for iterative development over phases
- Staff self-educated on SOA and looked at Open Source systems before talking to vendors
- One week “kickstart” education and POC session
  - Built a data synchronization application
- Proof to the business:
  - Concur built a prototype that offered real value to executives:
    - Single customer view mashup – pulled open CRM tickets, ERP and CRM data.
    - The demo was an “instant hit” – gaining an executive sponsor
- Team identified re-usable services
  - Put extra effort into the design
- Several refactoring iterations

Benefits

- Lower cost of licenses/users on SaaS systems
  - Previously were using licenses for occasional users
- Intermittent users were being trained on systems that they rarely used – the new mashups replaced this requirement
- The SOA design has allowed incremental replacement of some legacy systems
  - Existing test plans for Sarbanes-Oxley could be re-used
- Open source meant that a POC could prove the benefits to the business without upfront expenditure

Lessons Learnt

- Keep it Simple
- In-house expertise has paid off
  - Steeper learning curve but
  - Better technology selection
  - Lower overall cost
  - More agility
- Use of open source projects has
  - Reduced cost
  - Been more flexible
  - Given better access to the community and developers

Business to Government

Case Study 2

IT- og Telestyrelsen

Ministeriet for Videnskab
Teknologi og Udvikling

OIO SOI

OIO SOI

- Danish Government wanted to simplify electronic business
  - Especially for Business-to-Government (B2G)
- Potential savings of 630m Euros by digitalizing business
- Requirements
  - Reliable delivery
  - Secure – encrypted and signed messages
  - Support small businesses

OIO SOI

- Several aspects
  - A registry for service lookup
  - A profile of transport protocols
  - Open Source toolkits for Java and .NET
  - A reference implementation of a message handler
  - A legal framework
- Some existing framework
  - A nationwide digital certificate framework
  - A standard XML syntax for invoices and orders (UBL2)

Registry

- A profile of OASIS UDDI v3.0
- A central registry run by the Danish Government
  - https://publish.uddi.ehandel.gov.dk:12443/registry/uddi/web
- Designed to be used by electronic clients
  - Not to be browsed by humans!
- Requires a Danish Certified Certificate to publish

RASP

RASP
Reliable Asynchronous Secure Profile

- A profile of
  - SOAP 1.2
  - WS-Security 1.1
  - WS-ReliableMessaging 1.0
  - WS-Addressing
- Two bindings: HTTP and SMTP

- Why SMTP?
  - To allow small businesses to communicate
  - No requirement to host a web server
    - No 24x7 operation
    - No firewall configuration
  - Only an email address

RASP capabilities

- Authentication
- Confidentiality
- Integrity
- Non-repudiation / proof of delivery
- Support for intermediaries
- Asynchronisity

Interoperability

- RASP includes libraries for both
  - .NET – based on WCF 3.0
  - Java – based on Apache Axis2
- Defined a set of tests and run using a continuous test environment
- Biggest problems were found with
  - WSRM and SMTP

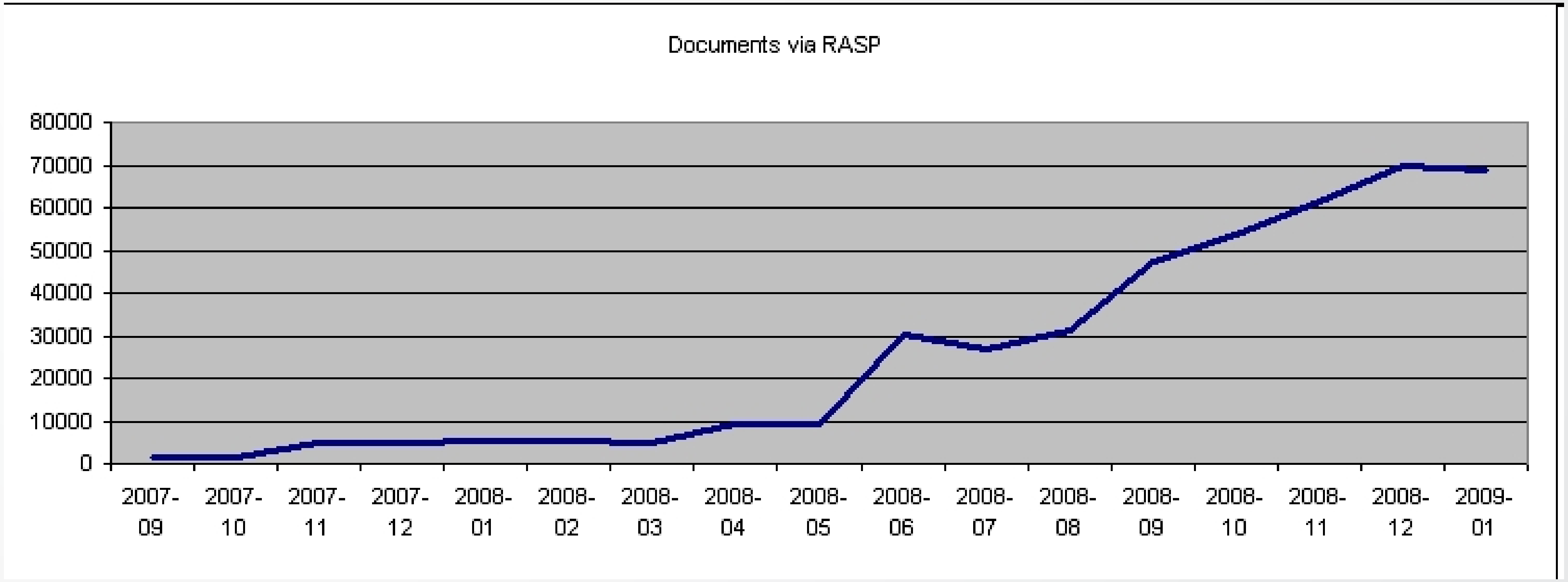
NITA Interop

No RM, No Sec		HTTP		SMTP	
Scenario	Description	Axis2->.NET	.NET->Axis2	Axis2->.NET	.NET->Axis2
1	Basic success	Yes	Yes	Yes	Yes
2	Resending	NA	NA	NA	NA
3	Timeout	NA	NA	NA	NA
4	Incomplete stack fault	NA	NA	NA	NA
5	Clock Skew	NA	NA	NA	NA
6	Custom Headers	Yes	Yes	Yes	Yes
7	Mail Binding validity	NA	NA		
RM Only		**HTTP**		**SMTP**	
Scenario	**Description**	**Axis2->.NET**	**.NET->Axis2**	**Axis2->.NET**	**.NET->Axis2**
1	Basic success	Yes	Yes	Yes	Yes
2	Resending	Yes	Yes	Yes	Yes
3	Timeout	Yes	Yes	Yes	Yes
4	Incomplete stack fault	Yes	Yes	Yes	Yes
5	Clock Skew	NA	NA	NA	NA
6	Custom Headers	Yes	Yes	Yes	Yes
7	Mail Binding validity	NA	NA		
Sec only		**HTTP**		**SMTP**	
Scenario	**Description**	**Axis2->.NET**	**.NET->Axis2**	**Axis2->.NET**	**.NET->Axis2**
1	Basic success	Yes	Yes	Yes	Yes
2	Resending	NA	NA	NA	NA
3	Timeout	NA	NA	NA	NA
4	Incomplete stack fault	Yes	Yes	Yes	Yes
5	Clock Skew	Yes	Yes	Yes	Yes
6	Custom Headers	Yes	Yes	Yes	Yes
7	Mail Binding validity	NA	NA		
RM+Sec		**HTTP**		**SMTP**	
Scenario	**Description**	**Axis2->.NET**	**.NET->Axis2**	**Axis2->.NET**	**.NET->Axis2**
1	Basic success	Yes	Yes	Yes	Yes
2	Resending	Yes	Yes	Yes	Yes
3	Timeout	Yes	Yes	Yes	Yes
4	Incomplete stack fault	Yes	Yes	Yes	Yes
5	Clock Skew	Yes	Yes	Yes	Yes
6	Custom Headers	Yes	Yes	Yes	Yes
7	Mail Binding validity	NA	NA		

Logical architecture

- This is logically a complete peer-to-peer architecture
  - With only a central registry
- Any company can talk to any other company
- Even those with only mail accounts
- Cannot track all the requests!

Results



18,500 companies sending invoices via RASP
Mandatory to send invoices to all government agencies
Scanning companies and a web gateway allow bridging

Lessons learnt

- SMTP in the real world is tricky
  - Spam filters can modify or drop messages
  - Our email accounts got shut down for “spamming”
    - i.e. sending many messages in a short time
  - Timeouts were too long for the RM system
  - We made mistakes layering SMTP and WS-Addressing
- Publishing interoperable reference implementations was a big win
  - Proved interoperability
  - Formed the basis for other implementations to test against
- The RASP team is now working on a European initiative:
  - PEPPOL http://peppol.eu
  - Trying to bring the same results across Europe

Resources

- RASP specs and pointers to implementations
  - http://tinyurl.com/azwhx5
- Peppol
  - http://peppol.eu

Case Study #3

Enterprise IT Management

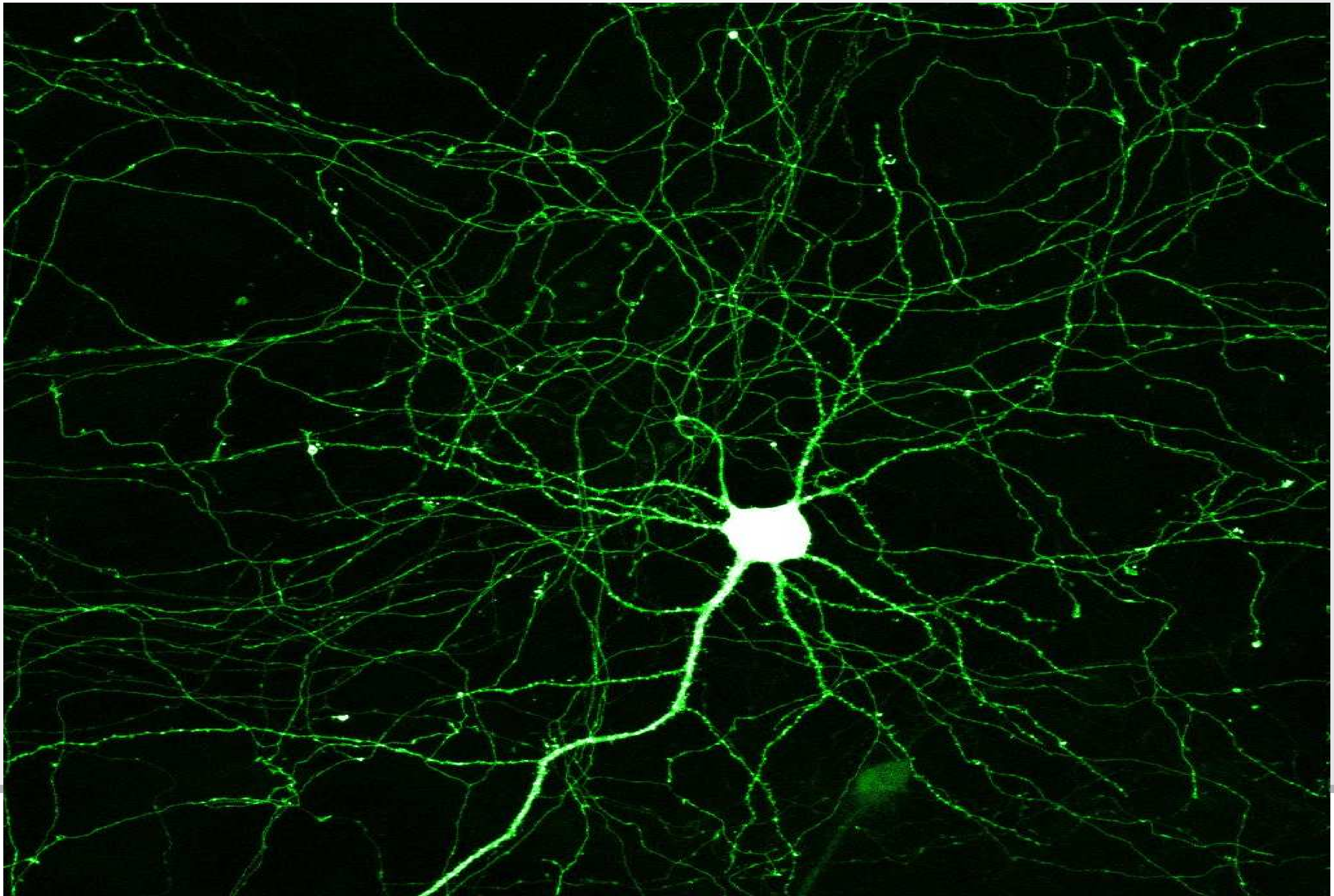
Enterprise IT Management

- Problem statement:
  - Customers have multiple installed management systems
    - Network Management
    - User Management
    - Systems Management
  - All from the same vendor!
  - These are not just “stock” systems – each has been customized for each installation
  - Customers have to keep these systems in sync
    - By data entry
  - Any solution needs to be flexible, extensible, modifiable

This is a difficult problem!

- Synchronizing multiple different systems
- But:
  - Systems have different underlying formats
  - Some of the systems may be more accurate than others
  - Need to be able to scale to different numbers of systems
  - Must be extensible / reprogrammable

Event based models



Actuators and Sensors

- An actuator emits an event
- A sensor accepts events

- Each of the systems produces events when something changes within the system
- An Adapter converts the event into an XML and publishes it
  - The XML can be in an “Application Specific” format
  - These events are transformed by the ESB into “Generic”

Managing the Event Subscriptions

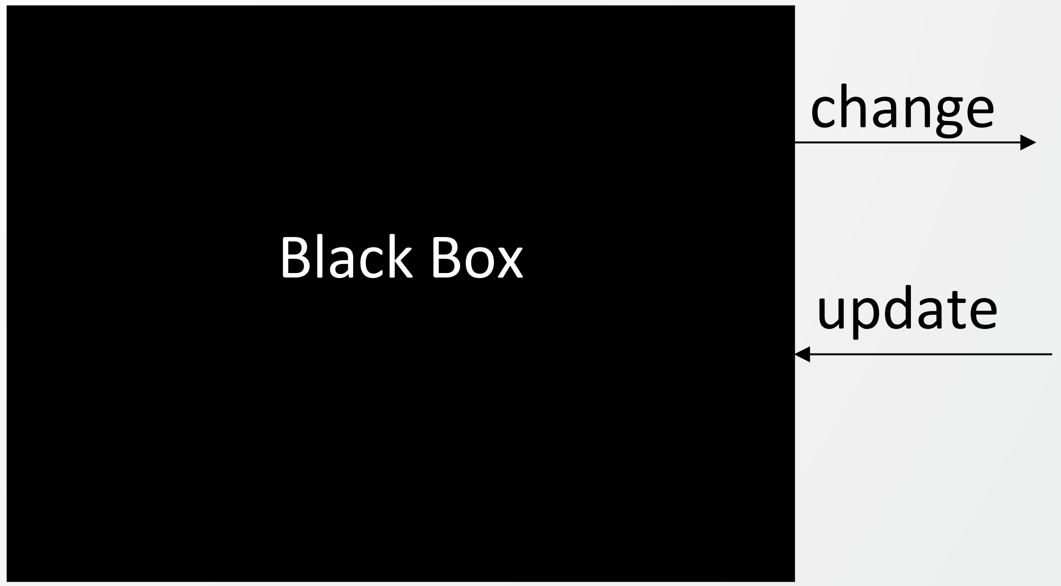
- A header carries the “Topic”
- E.g.
  - /config/hardware/server/windows/xp
- Subscribers can subscribe to a specific topic, or all sub-events
- The topic space is represented as a tree in the registry
  - Subscriptions are simply URLs stored as entries at a point in the tree

```
/config/
   /software/
   /hardware/
        /server/
         /linux/
         /windows
              /xp/ URL1
                   / URL2 (etc)
```

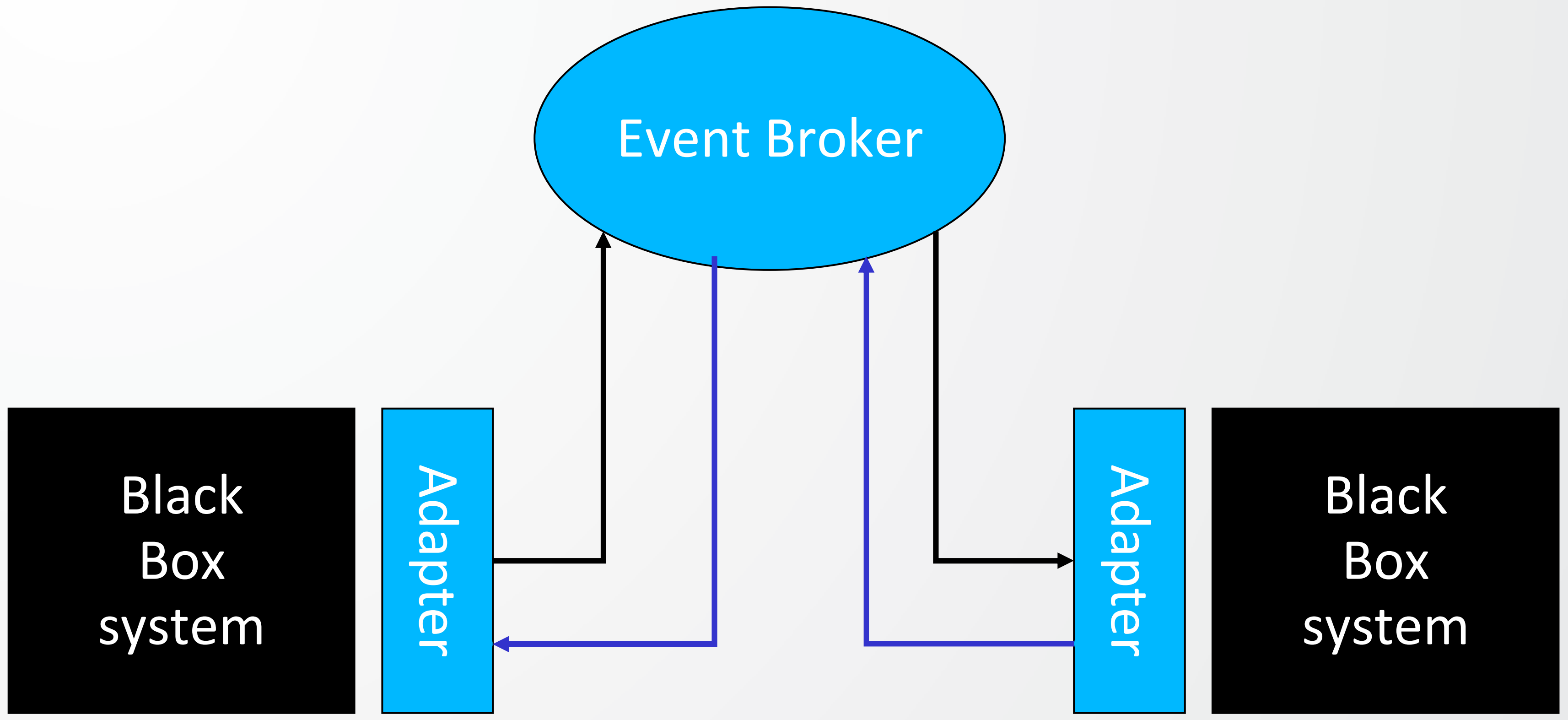
- The G-message schemas match the structure
  - /config/hardware/server extends /config/hardware
  - The master data services are all generated from a schema-driven DSL

Feedback!

Feedback problems

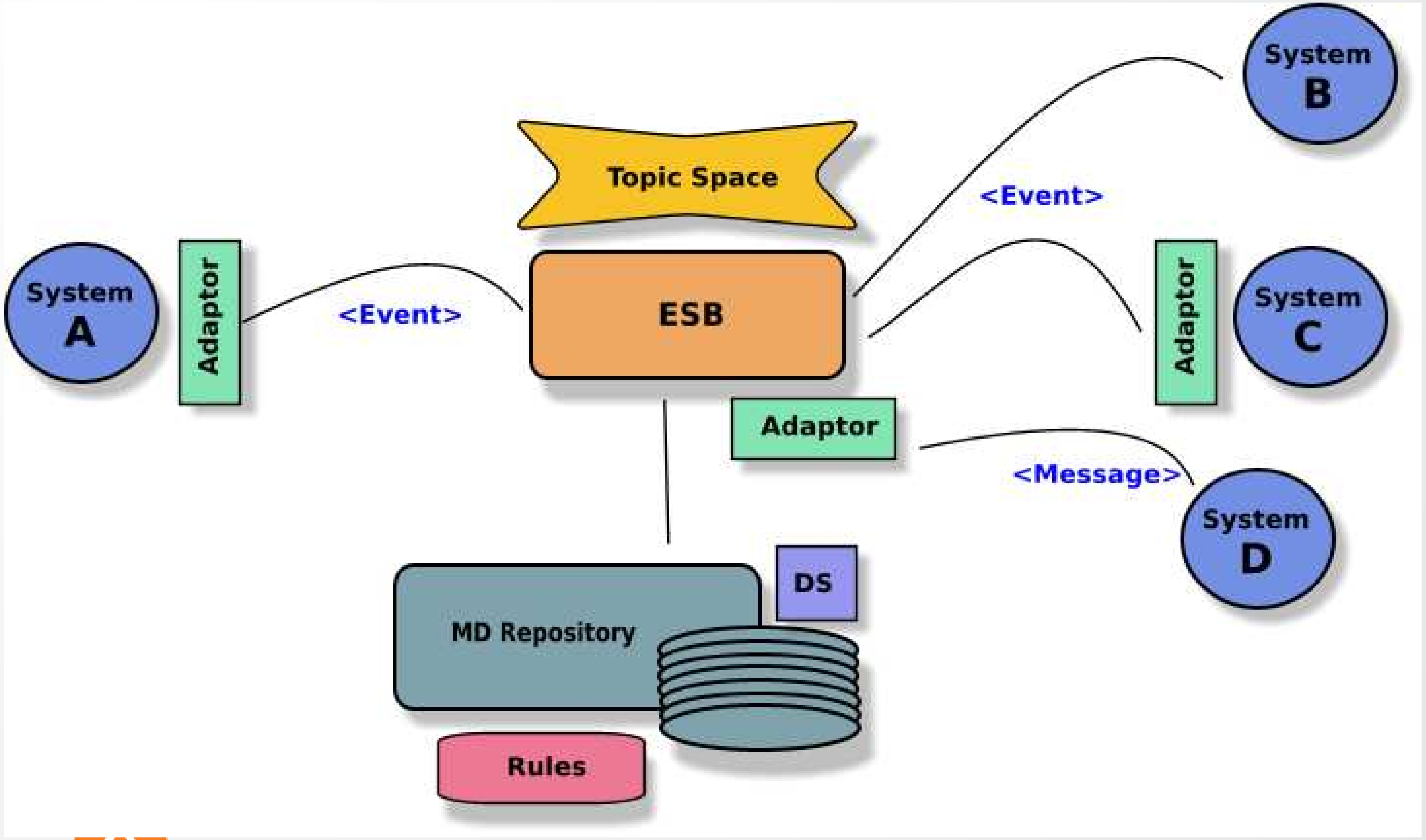


Feedback loops



http://pzf.fremantle.org/2008/09/interesting-problem-in-event-driven.html

Adding Master Data into an Event Based Architecture



Understanding the flow

- Adapter produces an AS-Event
- ESB transforms to a G-Event and sends to subscribers
- Master receives the event
  - Decides if it is an echo (and drops)
  - Executes policy based on the topic/message
    - This may execute a business process or ruleset
- Master updates the master db
- Republishes in a second topic space using a G-Event
  - This is now the master event
  - This gets transformed to an update of the other systems using the AS-schema

Technologies used

- SOAP
- WS-Transfer for the updates
  - Both the adapters and the master data
- WS-Eventing for the events
- WS-Security for authentication, encryption, signatures
- WS-ReliableMessaging for reliable message delivery
- The system is manageable using JMX
  - But can also be managed by logging events with a new subscriber

Project approach

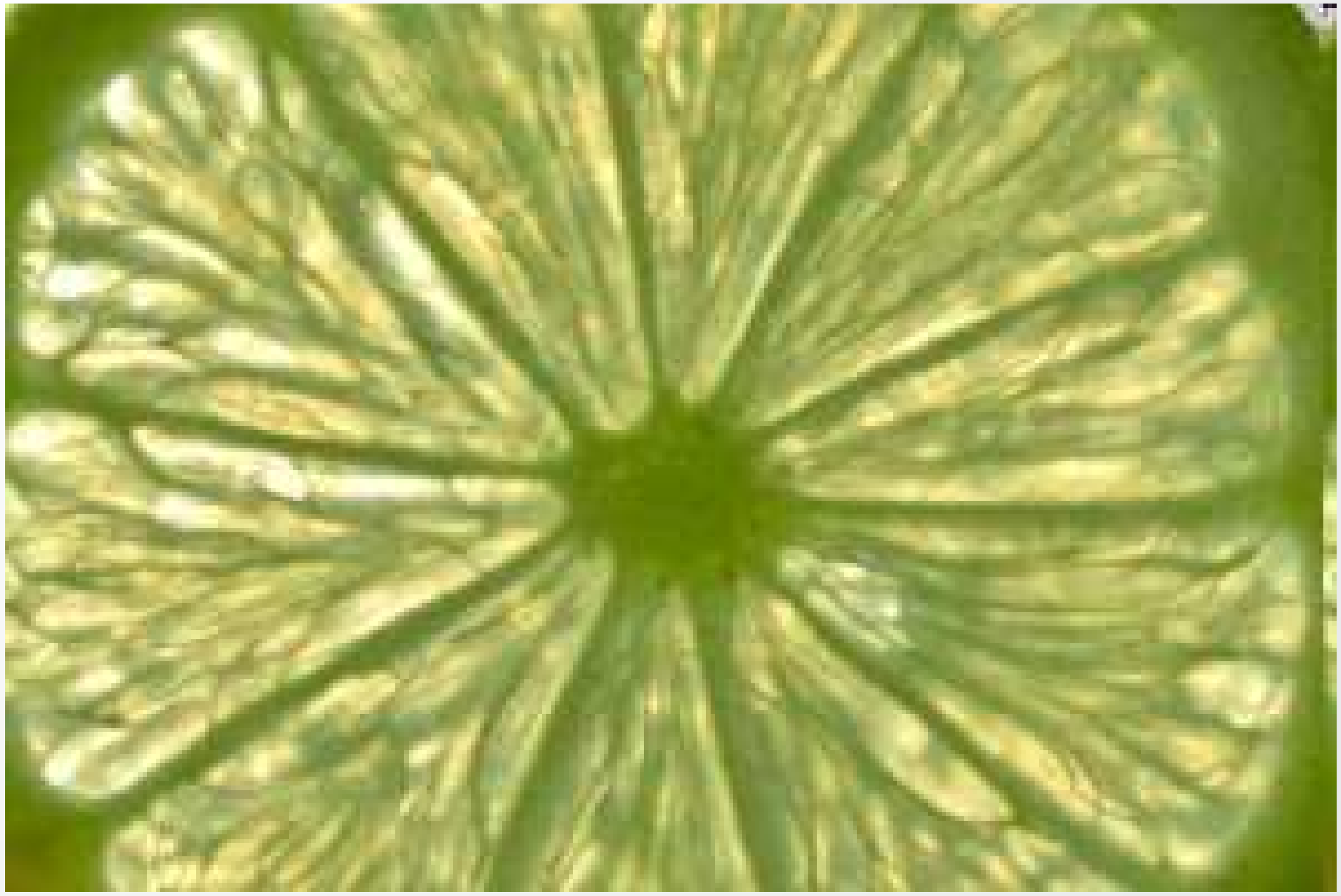
- Kickstart 1 week
  - "Thin Slice" end-to-end
  - Several teams
    - Adapter
    - Master Data
    - Eventing
    - Transformation
  - Integrated
- Iterative development
  - Start with two key Use Cases
- Open Source
  - In close partnership with WSO2 for support and consultancy

Anti-patterns

- Use a full waterfall model
- Don’t budget time for integration test
  - Assume that standard coding unit test->integration test will work
- Build unit tests that don’t test interoperability
  - E.g. Simulate XML request/response inside the calling system rather than calling a remote system
- Wait until all the systems are ready before starting any integration test
  - A delay to one system will hold up testing all the others
- Don’t bother with continuous build and test
  - Even better build by hand
  - **Even better** test by hand too
- Have a nice complex process to hand over from development to test
  - That way each defect will take a long time
- Wait until the project is failing to find out your team doesn’t have the skills

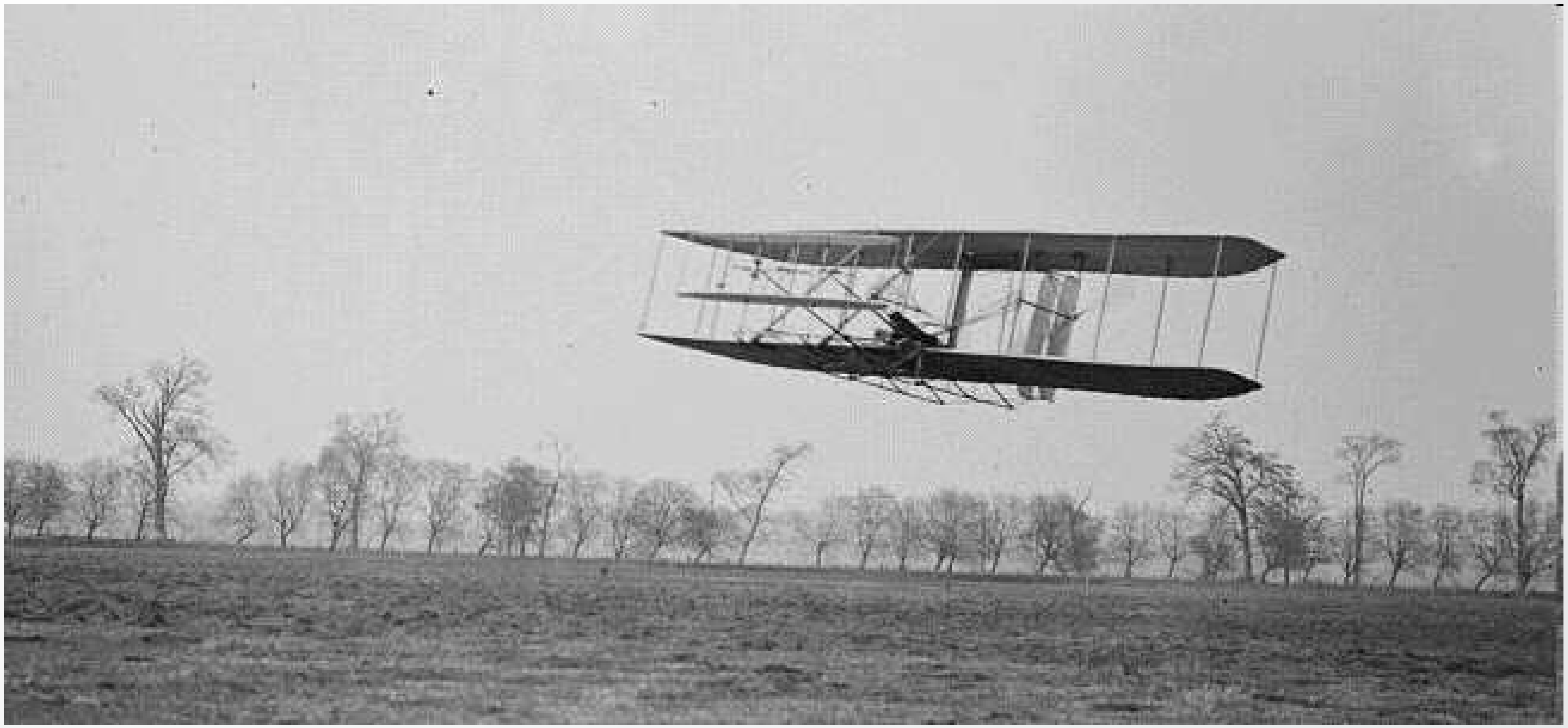
Conclusions

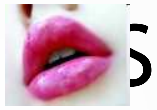
Thin slice prototyping is always a good idea



Iterative project plans are essential

Prove the concept to the business





S

Keep it Simple, Stupid!

Questions?