Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

GlassFish™ ESB: get your apps on the bus!

OpenESB
The *Open* Enterprise Service Bus

Frank Kieviet
OpenESB Community Manager

Sujit Biswas
Senior Software Engineer

Sun Microsystems

Agenda

- Why use an ESB?
  - ESB = “Ant for integration/SOA”
- Components
- Demo
- Architecture
- Open Source: OpenESB and GlassFish ESB
- What is next?
- Q&A

Why use an ESB?

- You can do anything with Java code and an application server (e.g. EE), no?
  - EE provides transactions, JAX WS
  - Use third party jars for misc. tasks
  - Add JPA for state management

- Question is akin to “Why use Ant?”

Analogy: builds

- Characteristics of builds
  - Each project has its own build: hundreds of builds per enterprise
  - Many repetitive tasks (e.g. compiling, jarring, …)
  - Builds must be maintained over time
  - Builds must be understandable by Configuration Management, not just developers
  - Build process must be automated and managed

Doing builds in Java or in Ant?

- Yes, you can write a build script with just Java, but:
  - Code duplication, repetitive tasks, lots of code
  - Java is not at the right expression level
    - Difficult to get an overview, difficult to quickly understand, difficult to maintain
  - Nobody does it!
- Everybody uses Ant
  - Provides a higher level abstraction
  - Gets a lot done with just a few lines
    - Dozens of built in tasks (some simple, some do heavy lifting)
    - Extensible through custom plug-ins
  - Favors declarative programming (configuration) over coding
  - IDE support
  - Leads to fast development

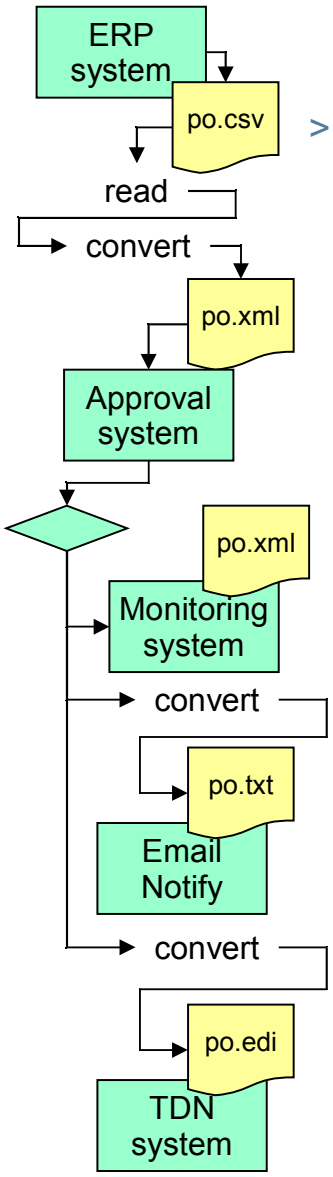
Characteristics of Integration and SOA

- Hundreds of integration projects per enterprise
- Lots of commonality (some simple, some hard):
  - Connectivity
  - Data transformation
  - Business logic / orchestration
- Integration projects evolve over time
- Runtime needs to be managed

ESB = “Ant for integration/SOA”

- Important benefits:
  - Many off the shelf components (some simple, some hard)
    - Connectivity
    - Data transformation
    - Business logic / orchestration
  - Extensible through plug-ins
  - Provides a higher level abstraction to wire components together
  - Few or no lines of code + configuration accomplishes a lot
  - IDE support
- Makes managing large numbers of sub-projects possible

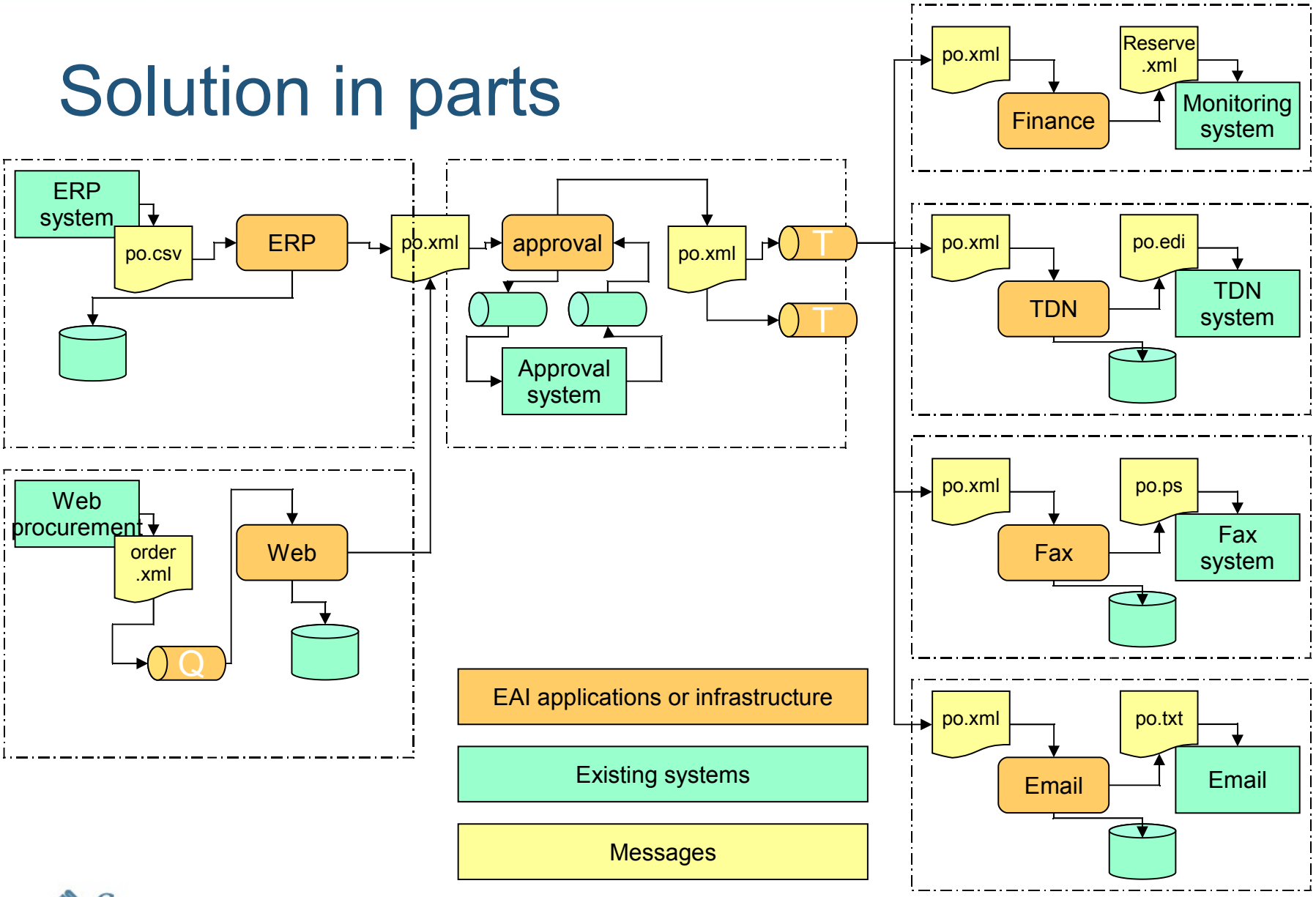
Example



> Following an order from the ERP system

- ERP system creates a file with a batch of orders in CSV format
- File needs to be picked up, read and converted to a canonical format
- The CSV file does not contain all information necessary for the canonical format so it needs to be enriched
- All orders in the CSV file need to be approved by an approval system
- Each order needs to be sent out to the Internet Gateway (TDN) in EDI format, or to a Fax gateway in PS format
- A financial monitoring application needs to get a copy of the order
- An email needs to be sent to the submitter of the order when the order goes out
- The department that generally takes delivery of shipments needs to be notified of the order

Solution in parts



Components: connectivity

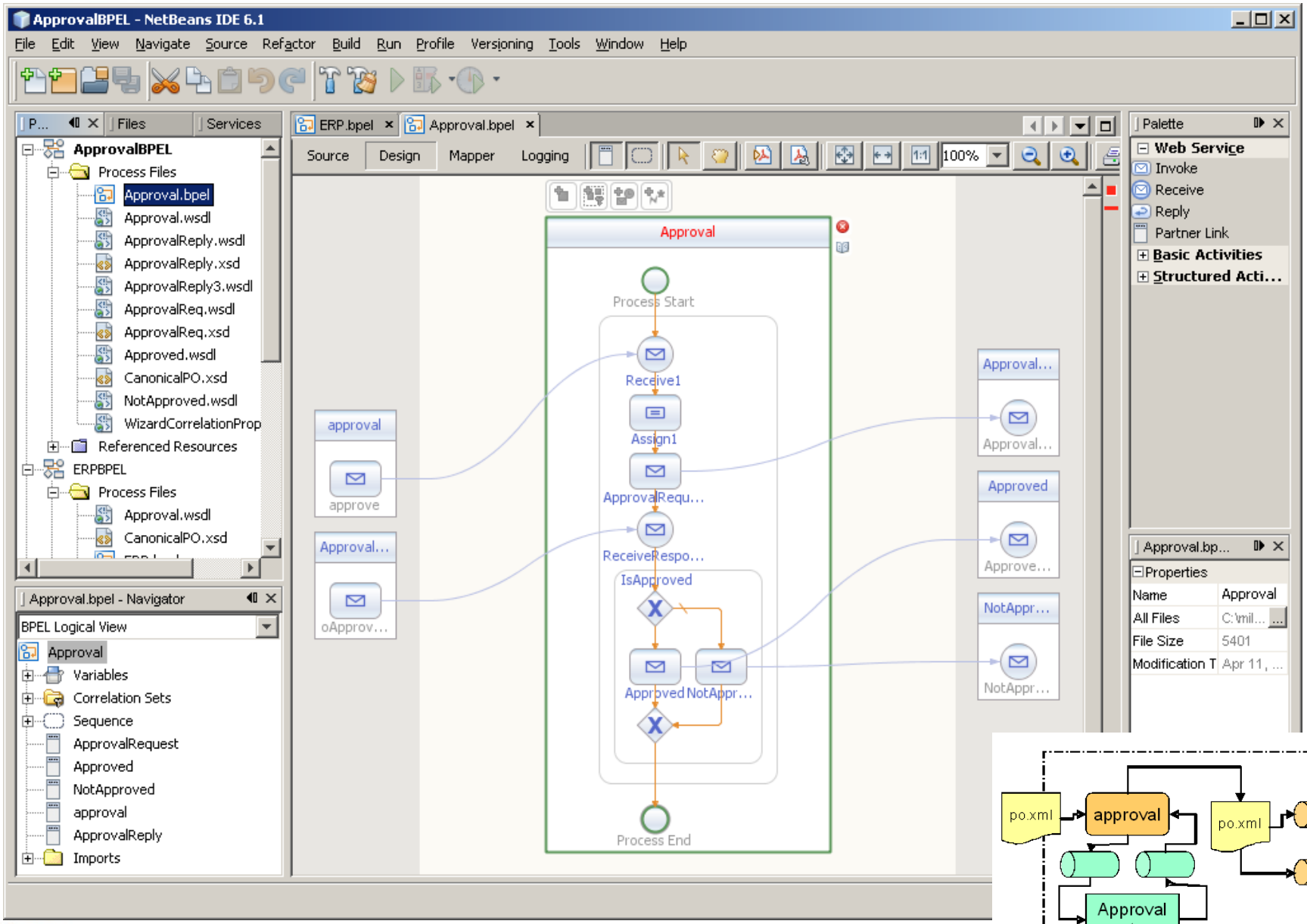
- Typically for an ESB:
  - HTTP, File, FTP, JMS, Email, Scheduler, database
- OpenESB also has:
  - ETL, Data Mashup, MDM, Asterisk, Exec BC, HL7, LDAP BC, RSS BC, SIP BC, SNMP BC, TCPIP BC, UDDI BC, XMPP BC, CICS BC, CORBA BC, DCOM BC, EJB BC, IMS BC, MSMQ BC, MQSeries BC, ScreenScraping SE, SWIFT BC, SAP BC

Components: data transformation

- Typical for an ESB:
  - XSLT SE
- OpenESB also has:
  - Encoding SE
  - Visual mapping in BPEL editor
  - Java based mapping (test/sample driven)

Components: business logic / orchestration

- ESB typically has:
  - Java?
- Open ESB also has:
  - BPEL SE: Long running transactions, Asynchronous interactions (correlation), Persistence, clustering
  - IEPSE: Complex Event Processing / Event Driven Architectures
  - EE SE: Makes EJBs accessible from within JBI
  - POJO SE: Use simple Java classes for business logic
  - Scripting SE: Use scripts for business logic
  - WLM SE: human workflow
  - Camel SE


ApprovalBPEL - NetBeans IDE 6.1
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help
Files
Services
ApprovalBPEL
Process Files
Approval.bpel
Approval.wsdl
ApprovalReply.wsdl
ApprovalReply.xsd
ApprovalReply3.wsdl
ApprovalReq.wsdl
ApprovalReq.xsd
Approved.wsdl
CanonicalPO.xsd
NotApproved.wsdl
WizardCorrelationProp
Referenced Resources
ERPBPEL
Process Files
Approval.wsdl
CanonicalPO.xsd
Approval.bpel - Navigator
BPEL Logical View
Approval
Variables
Correlation Sets
Sequence
ApprovalRequest
Approved
NotApproved
approval
ApprovalReply
Imports
ERP.bpel
Approval.bpel
Source
Design
Mapper
Logging
100%
Approval
Process Start
Receive1
Assign1
ApprovalRequ...
ReceiveRespo...
IsApproved
Approved
NotAppr...
Process End
approval
approve
Approval...
oApprov...
Approval...
Approval...
Approved
Approve...
NotAppr...
NotAppr...
Palette
Web Service
Invoke
Receive
Reply
Partner Link
Basic Activities
Structured Acti...
Approval.bp...
Properties
Name
Approval
All Files
File Size
5401
Modification T
Apr 11, ...

po.xml
approval
po.xml
T
T
Approval system

Example: BPEL SE

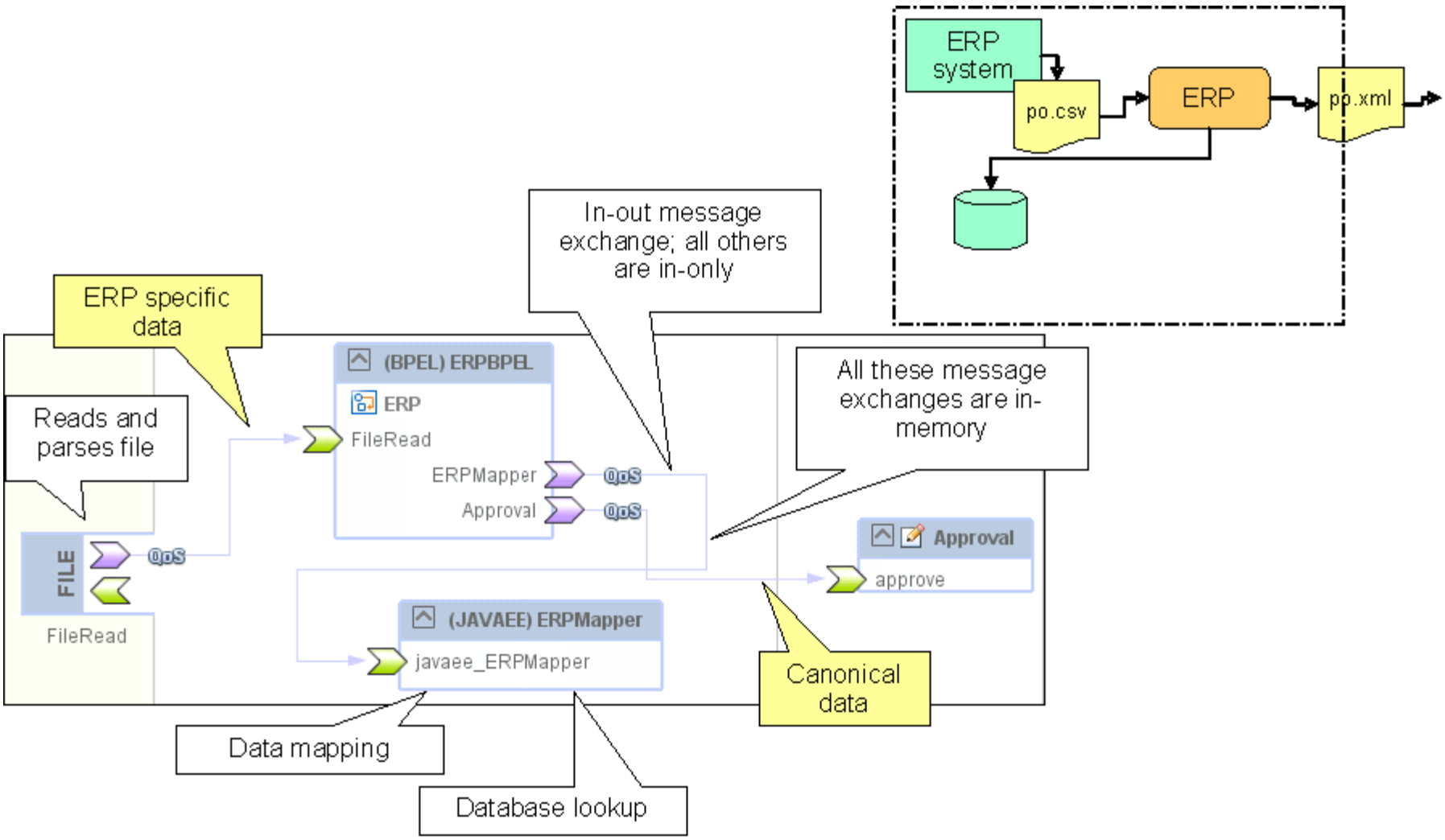
- BPEL: highly expressive language for
  - Orchestration
  - Asynchronous (correlation), concurrent interactions
  - Compensating transactions
  - etc.
- Strong IDE support for BPEL (graphically)
- Reliability through database persistence (transparent to user)
- Clustering support

Components: extensibility

- ESB typically provides
  - Proprietary way to add custom components
- OpenESB provides
  - Standards based components

Wiring together components

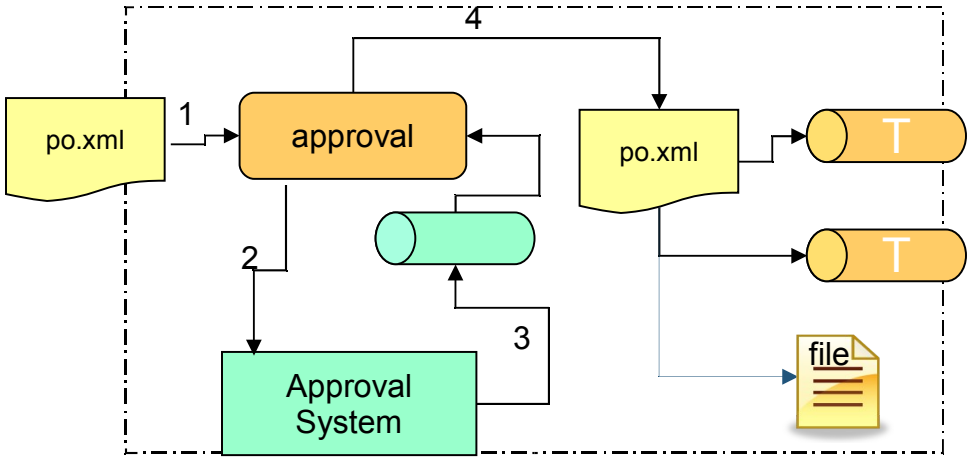
- ESB typically provides:
  - Higher level abstraction to express message flow / connectivity.
  - Typically XML file
- OpenESB also provides:
  - IDE to graphically wire components together


ERP system
po.csv
ERP
po.xml
In-out message exchange; all others are in-only
ERP specific data
All these message exchanges are in-memory
Reads and parses file
(BPEL) ERPBPEL
ERP
FileRead
ERPMapper
Approval
QoS
QoS
QoS
FILE
FileRead
Approval
approve
(JAVAEE) ERPMapper
javaee_ERPMapper
Canonical data
Data mapping
Database lookup

… leads to

- Less code
- Faster development
- Easier management

DEMO



Open standards

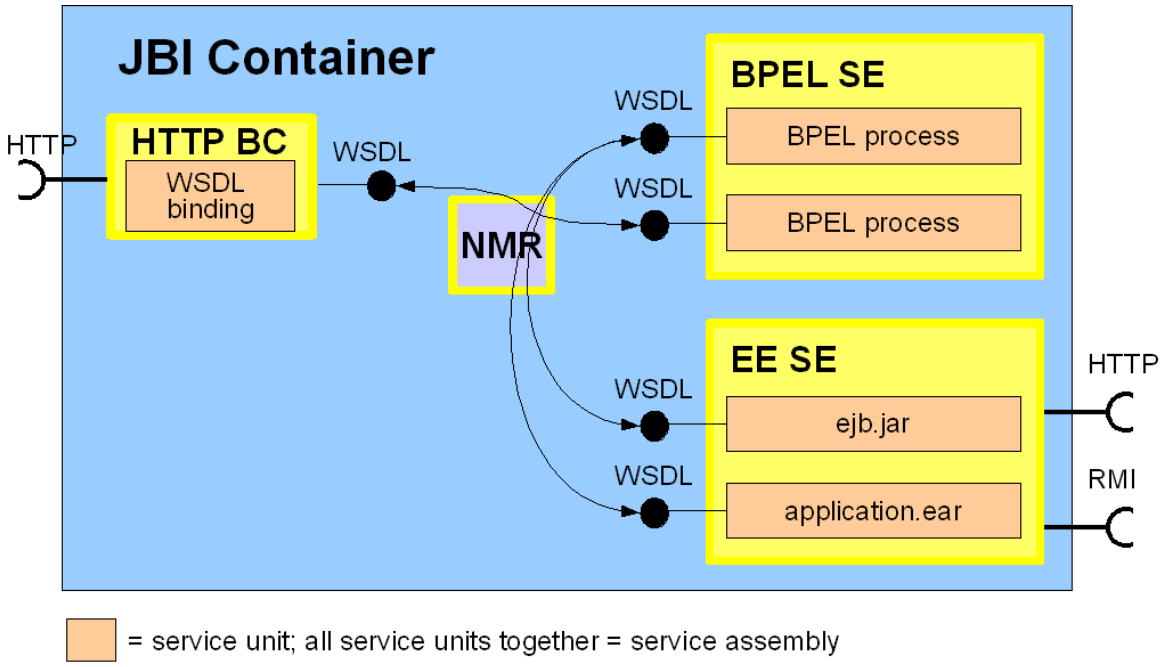
- Interoperability
  - Allows connectivity with other ESBs / systems
  - Typical for ESB: done through bridge / bolted on
  - OpenESB: architecture is based on open standards
- Component standards
  - Typical for ESB: components based on proprietary standards
  - OpenESB: JBI based

OpenESB architecture

- Components interact typically within the same VM
  - Homogeneous clustering
  - Propagation of transaction context, security context, etc
- Interactions with other nodes:
  - Typically done through HTTP / standards
  - Also for heterogeneous clustering: proxy binding
- Component interactions described with WSDL:
  - Describes message types

Architecture

- Components are installed into runtime
- Service units are deployed onto components
- Service assembly
  - contains service units
  - represents a composite application



Open source: project OpenESB

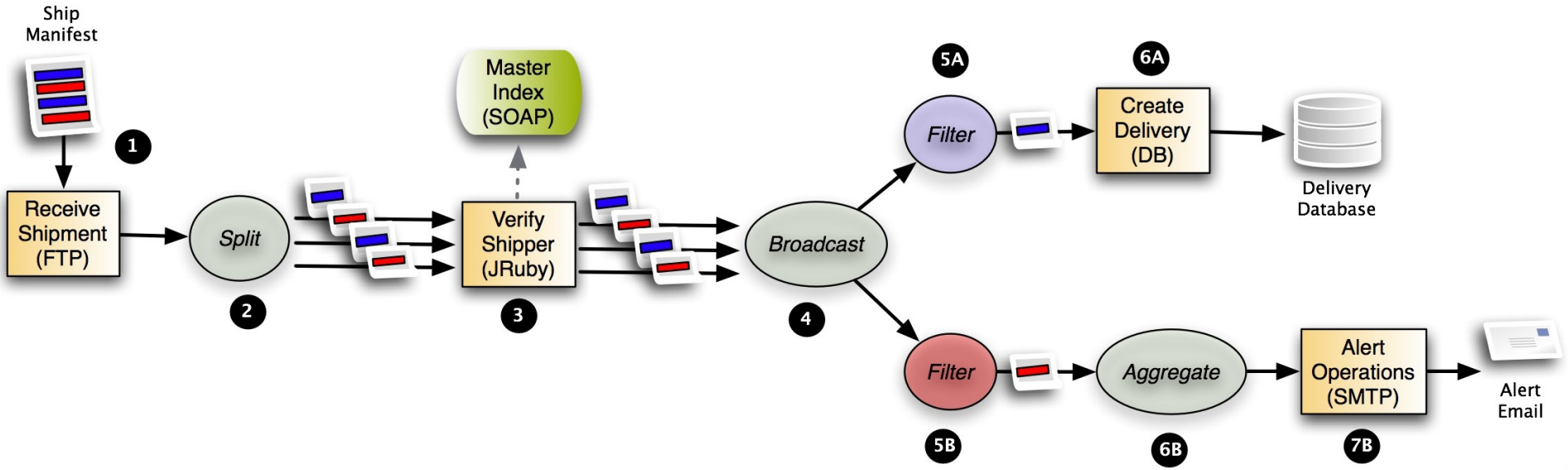
- Started in 2005
- Led by Sun Microsystems
- Many other partners and contributors:
  - E.g. contributors from Advantech, Imola, ChainBuilder, eviware, Gestalt
  - Implementation partners
- A vibrant community

GlassFish ESB

- Large community = large number of components
  - Humongous installer
  - Problems with synchronizing release dates
  - Components in different stages of development
- Solution:
  - Smaller distribution (installer) called GlassFish ESB
  - Subset of components
    - Most commonly used
    - Consistent quality
    - Same release cycle
- Commercial support on GlassFish ESB is available from Sun Microsystems

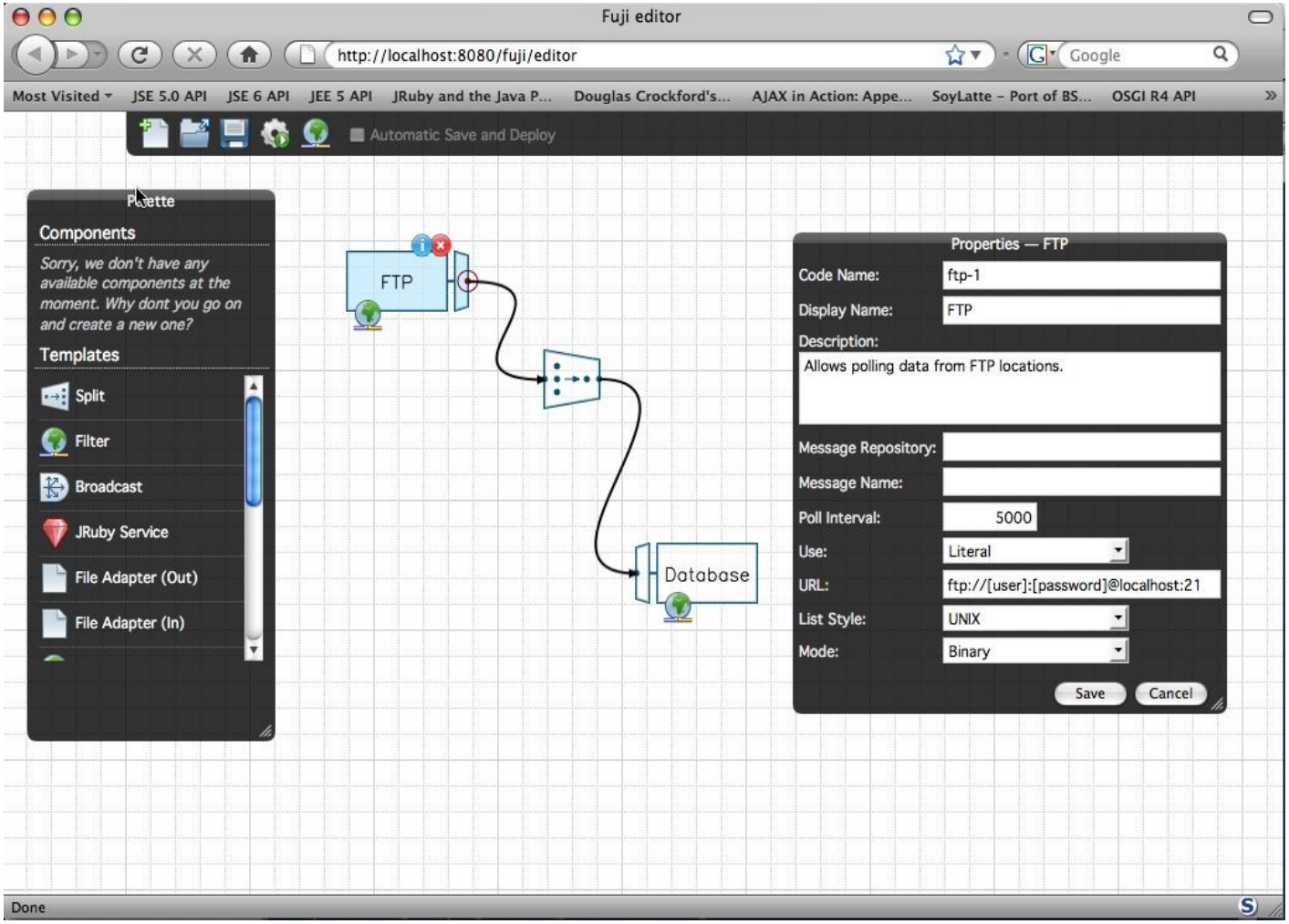
What is next? GlassFish ESB v3

- Runs on GlassFish v3
- OSGi
- Maven
- Simplify user interface to service artifacts
- Convention, Configuration, Code ... (In that order)
- New way of specifying interactions between components: Integration Flow Language
- Editable text
- Browser based visual editor
- Enterprise Integration Patterns out-of-the-box
- Interceptors
- New way of distribution over multiple machines


Ship Manifest
1
Receive Shipment (FTP)
Split
2
Master Index (SOAP)
Verify Shipper (JRuby)
3
Broadcast
4
5A
Filter
6A
Create Delivery (DB)
Delivery Database
5B
Filter
6B
Aggregate
7B
Alert Operations (SMTP)
Alert Email

```
#IFL - Integration Flow Language
ftp "receive-manifest"
jruby "verify-address"
database "create-delivery"
smtp "email-operations"

route do
    from "receive-manifest"
    split xpath ("//Manifest/Container")
    to "verify-address"
    broadcast do
        route do
            filter xpath ("count(//ShipTo/euid) > 0")
            to "create-delivery"
        end
        route do
            filter xpath ("count(//ShipTo/euid) = 0")
            aggregate set ("wait=60m,header=<alarm>,trailer=</alarm>")
            to "email-operations"
        end
    end
end
```


Fuji editor
http://localhost:8080/fuji/editor
Google
Most Visited
JSE 5.0 API
JSE 6 API
JEE 5 API
JRuby and the Java P...
Douglas Crockford's...
AJAX in Action: Appe...
SoyLatte – Port of BS...
OSGI R4 API
Automatic Save and Deploy
Palette
Components
Sorry, we don't have any available components at the moment. Why dont you go on and create a new one?
Templates
Split
Filter
Broadcast
JRuby Service
File Adapter (Out)
File Adapter (In)
FTP
Database
Properties — FTP
Code Name:
ftp-1
Display Name:
FTP
Description:
Allows polling data from FTP locations.
Message Repository:
Message Name:
Poll Interval:
5000
Use:
Literal
URL:
ftp://[user]:[password]@localhost:21
List Style:
UNIX
Mode:
Binary
Save
Cancel
Done

Thank You

Frank Kieviet
frank.kieviet@sun.com
Sujit Biswas
sujit.biswas@sun.com

http://open-esb.org