



vs. **Mule ESB** vs.





Consulting
Developing
Speaking
Coaching
Writing

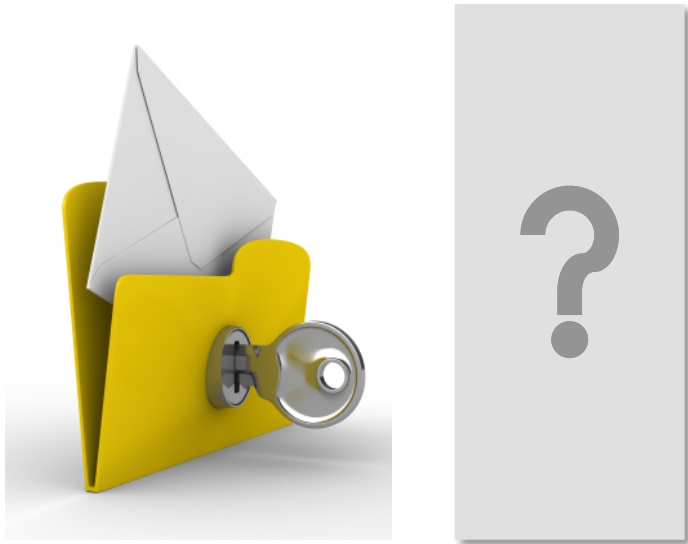
Main Tasks

Evaluation of Technologies and Products
Requirements Engineering
Enterprise Architecture Management
Business Process Management
Architecture and Development of Applications
Planning and Introduction of SOA
Integration of Legacy Applications
Cloud Computing

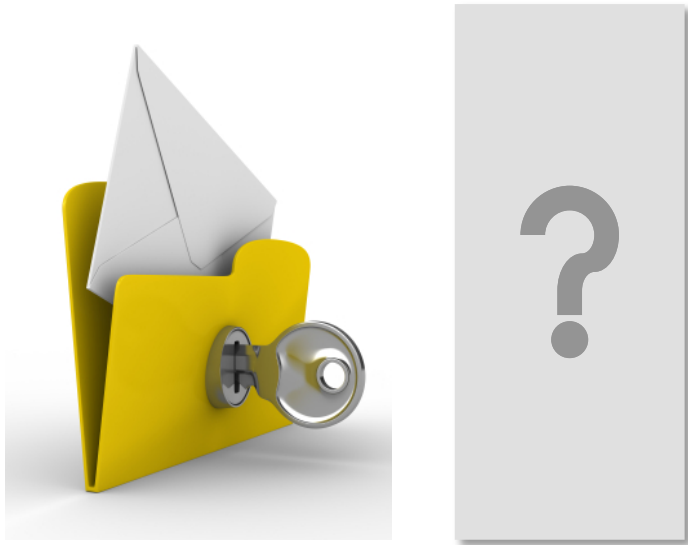
Contact

Email: kai.waehner@mwea.de
Blog: www.kai-waehner.de/blog
Twitter: @KaiWaehner
Social Networks: Xing, LinkedIn



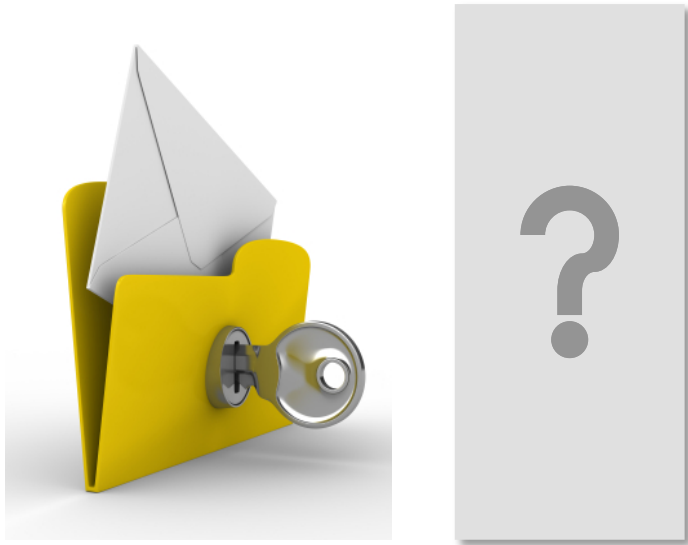


Do not reinvent the „integration wheel“!



Do not reinvent the „integration wheel“!

There are some good alternatives for Integration!



Do not reinvent the „integration wheel“!

There are some good alternatives for Integration!

Often an ESB is the wrong Choice!

- 1) Systems Integration
- 2) Integration Frameworks
- 3) Spring Integration
- 4) Mule ESB
- 5) Apache Camel
- 6) And the Winner is ...

- 1) **Systems Integration**
- 2) Integration Frameworks
- 3) Spring Integration
- 4) Mule ESB
- 5) Apache Camel
- 6) And the Winner is ...



Growth

- Applications
- Interfaces
- Technologies
- Products

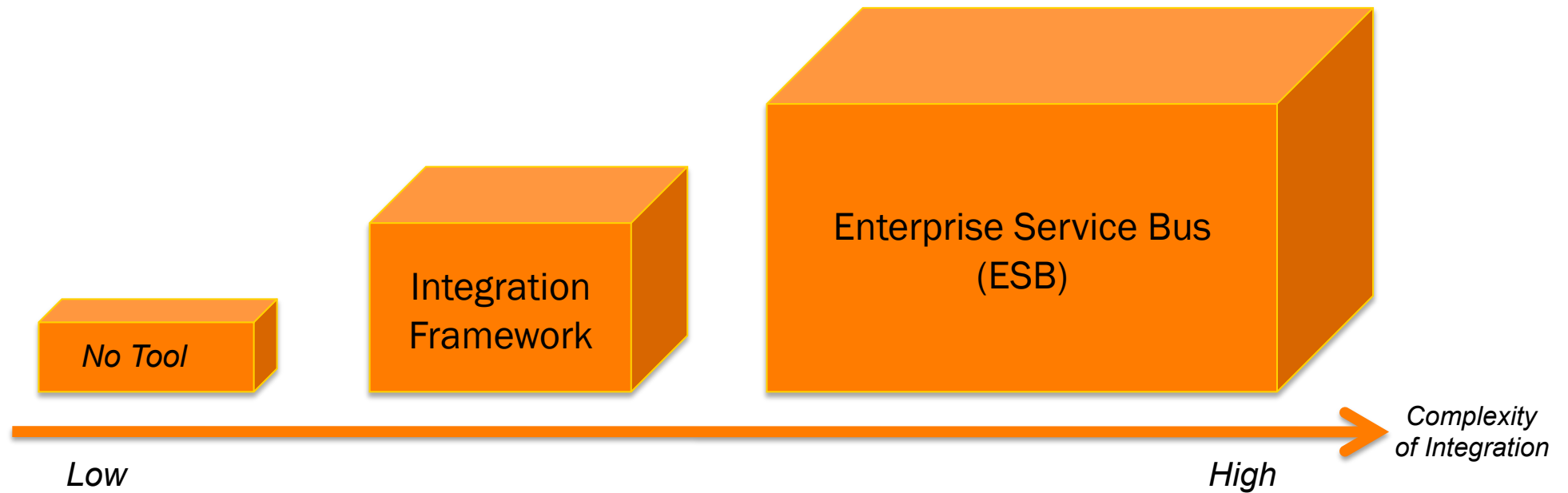


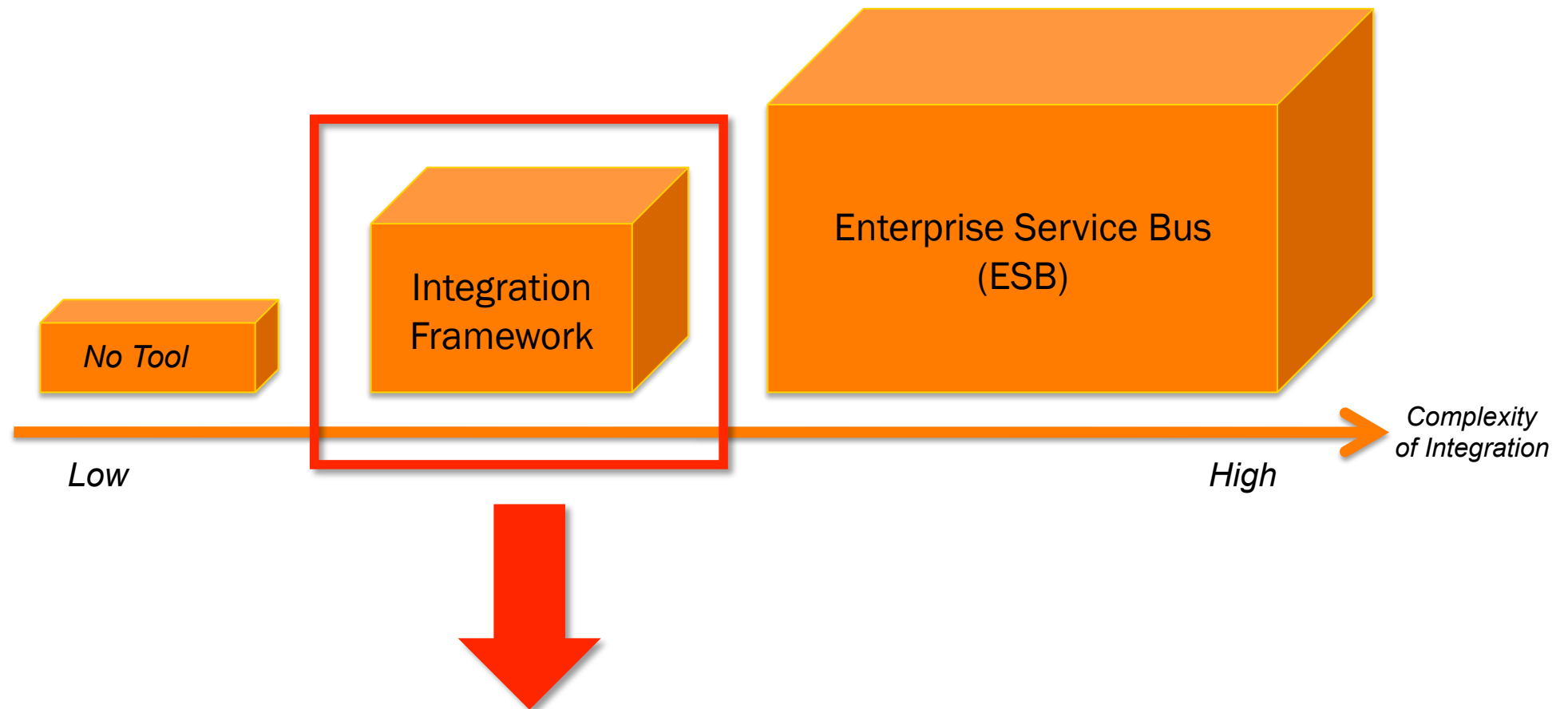


Everybody communicates to everybody



All Roads
lead to Rome ...



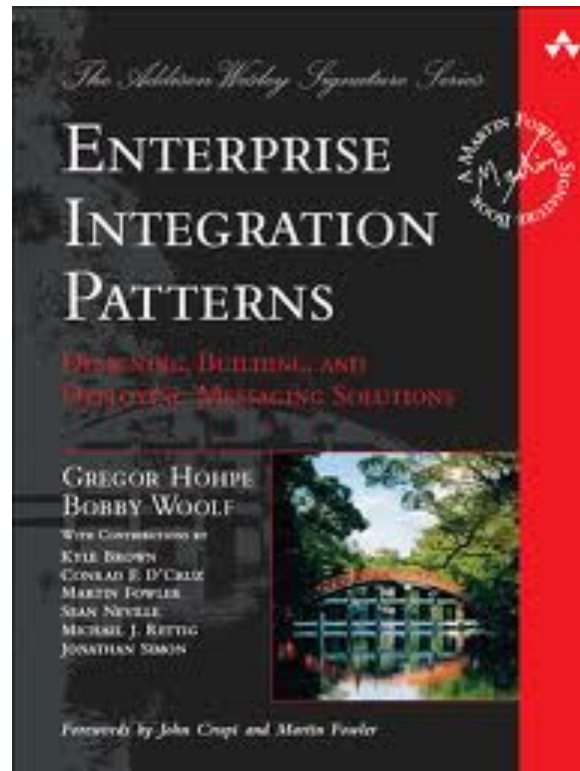


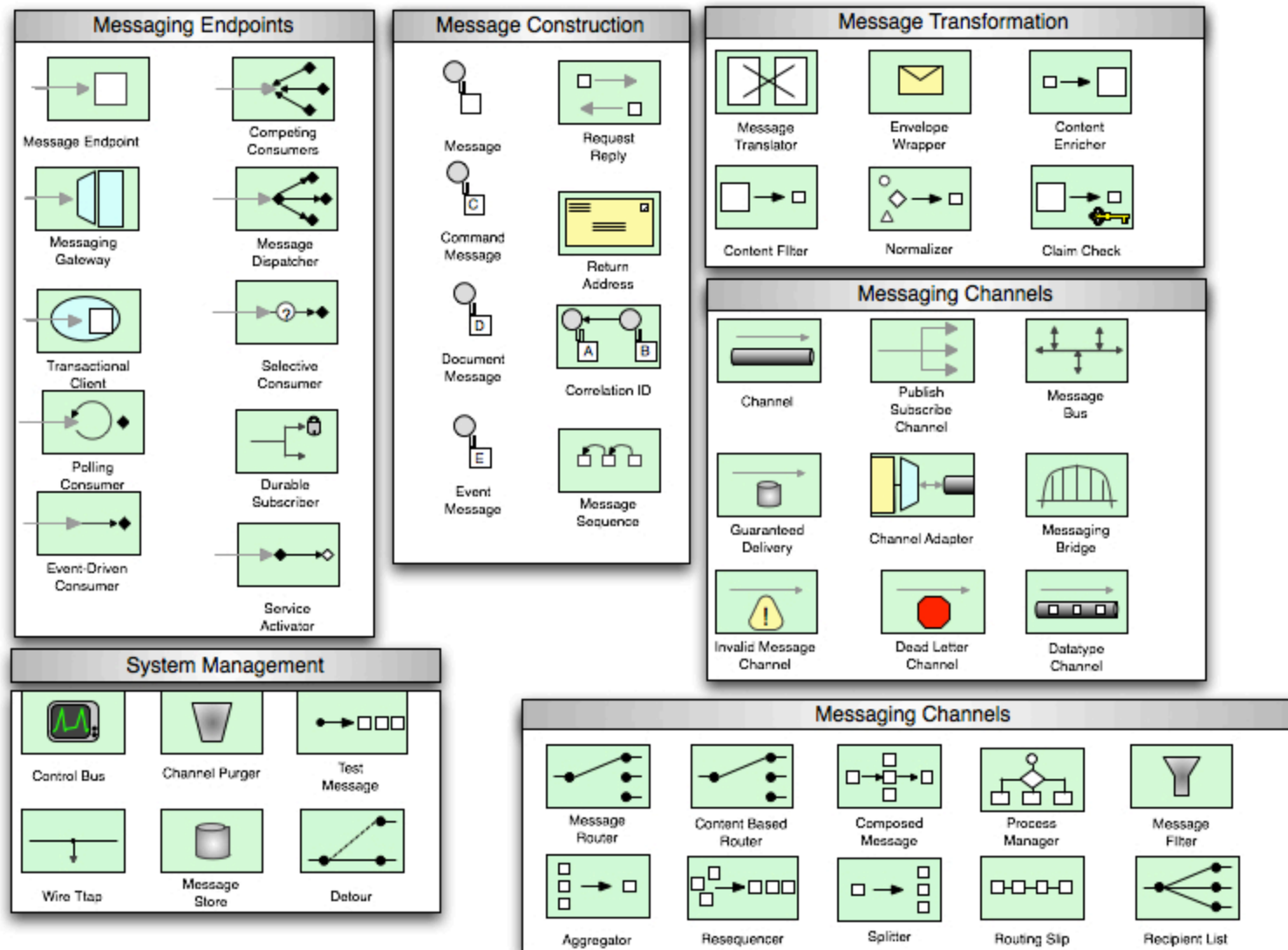
Spring Integration vs. Mule ESB vs. Apache Camel

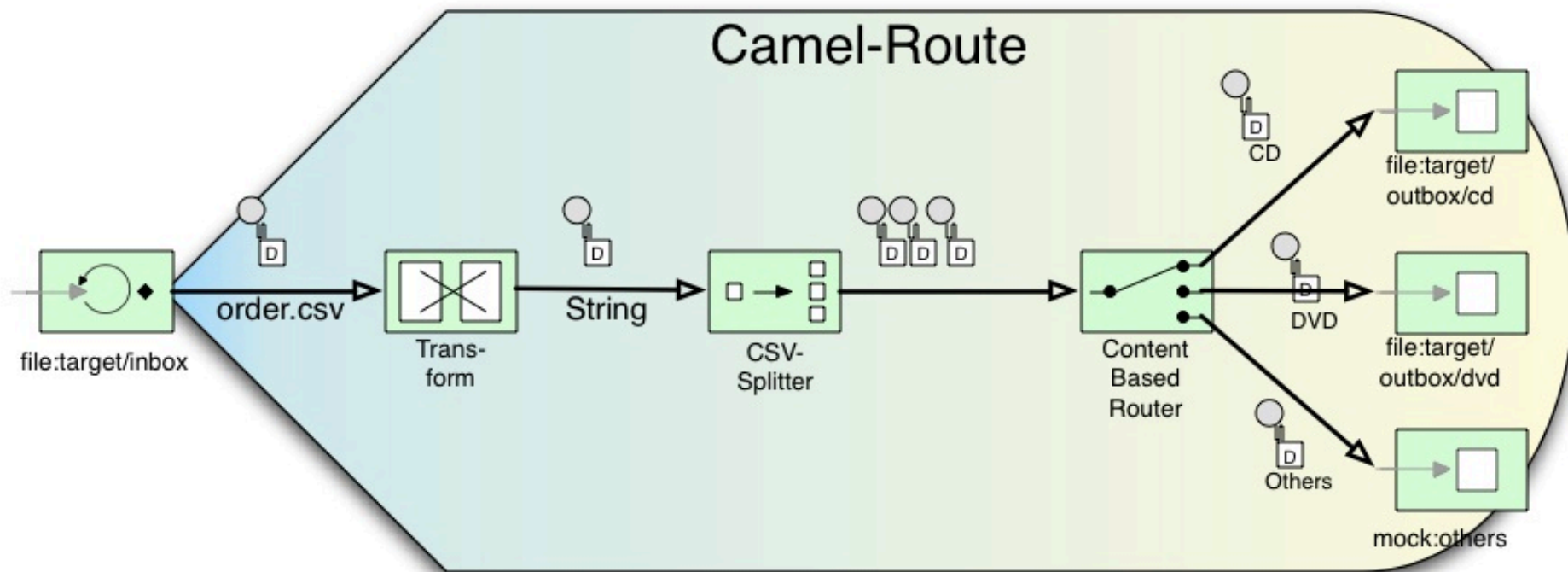
- 1) Systems Integration
- 2) Integration Frameworks
- 3) Spring Integration
- 4) Mule ESB
- 5) Apache Camel
- 6) And the Winner is ...

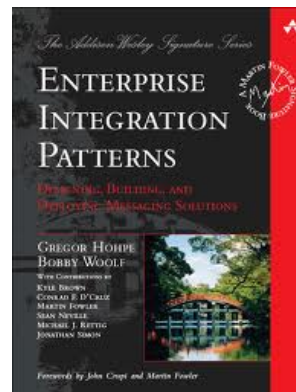


- Standardized Modeling
- Efficient Realization
- Automatic Testing









- Open source
- Basic concepts / architecture
- Testability
- Commercial support
- Error handling
- Monitoring
- Enterprise readiness
- Developer-centric vs. designer-centric
- Expandability
- Deployment
- Popularity
- Tool support
- Connectivity
- Domain specific language (DSL)



Integration of different Technologies



*„A **domain-specific language (DSL)** is a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and / or a particular solution technique.“*

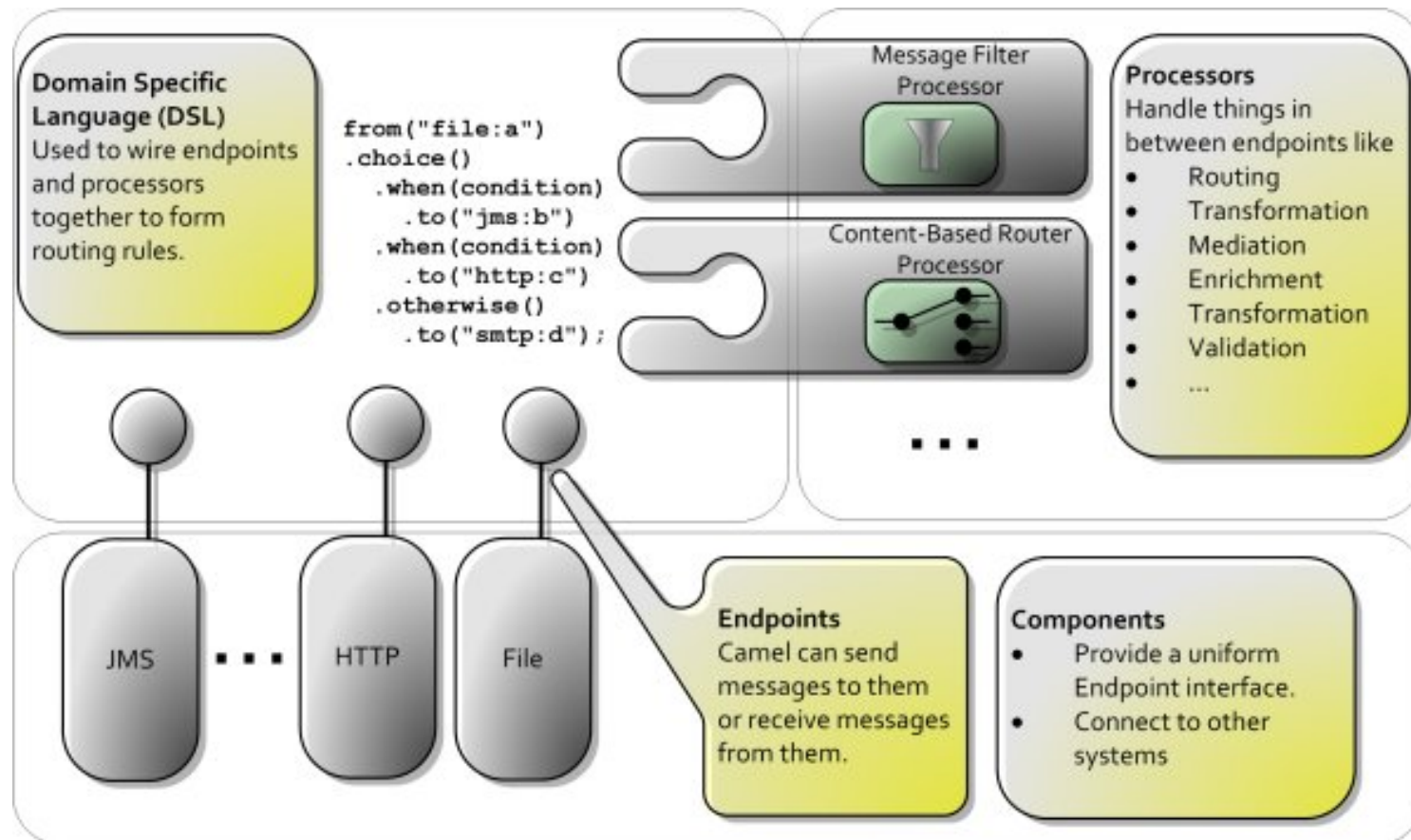
Wikipedia

*„A **domain-specific language (DSL)** is a programming language or specification language dedicated to a particular problem domain, a particular problem representation technique, and / or a particular solution technique.“*

Wikipedia



Internal DSL



<http://java.dzone.com/articles/apache-camel-integration>

(Exemplarily: Apache Camel => Concepts are all the same, only different names)

Standalone **Application Server**

Web Container

Spring Container

OSGi

Cloud





- Maturity
- Transactions
- Concurrency
- Error handling
- Monitoring
- Testability


spring
Integration

vs. **Mule ESB** vs.

Apache

Camel



Apache License



Common Public
Attribution License
(CPAL)



Apache License


Integration

vs. **Mule ESB** vs.


Apache Camel




spring
source



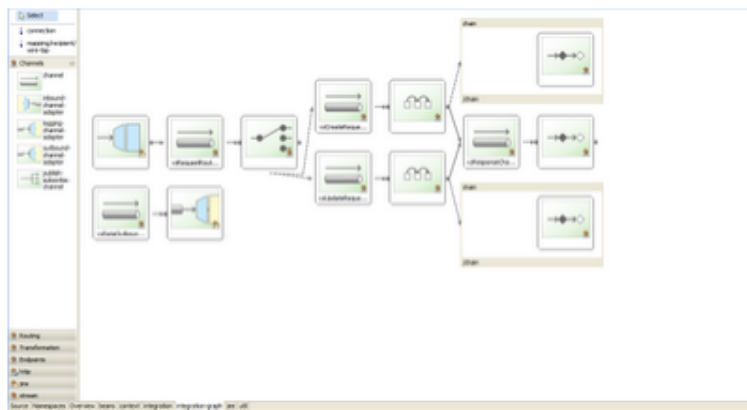
 MuleSoft™



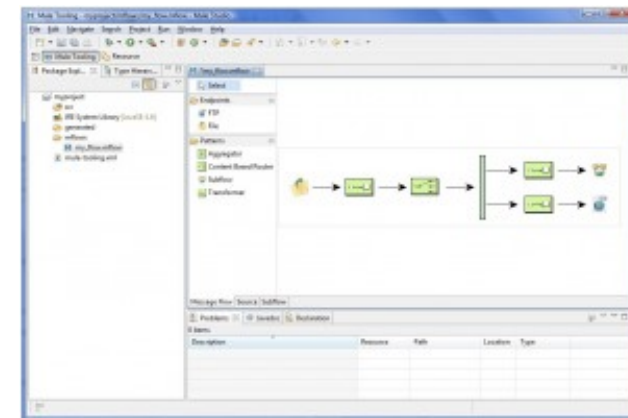
FuseSource
A Progress Software Company


talend
open data solutions

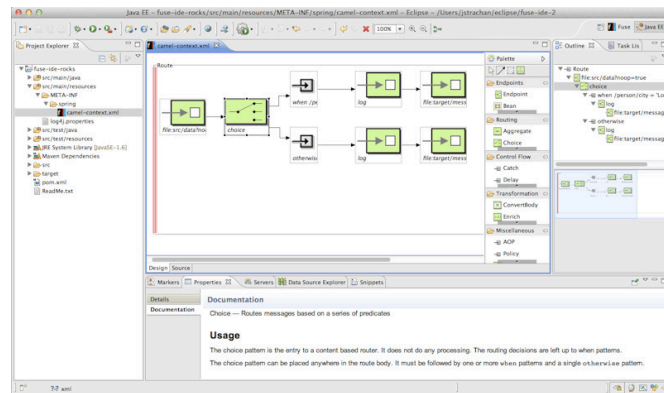
Integration Graph for Spring Integration



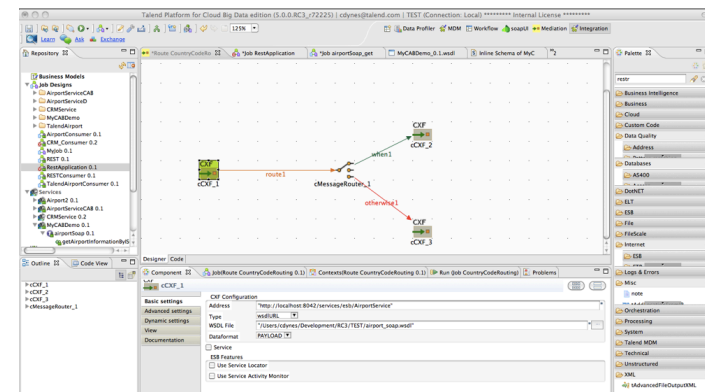
Mule Studio for Mule ESB



Fuse IDE for Apache Camel



Talend Studio for Apache Camel

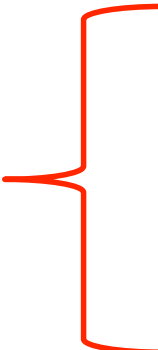


- Concepts of each Framework
- Code Example
- Live Demo



- Open source
- Basic concepts / architecture
- Testability
- Commercial support
- Error handling
- Monitoring
- Enterprise readiness
- Developer-centric vs. designer-centric

Focus

- 
- **Expandability**
 - **Deployment**
 - **Popularity**
 - **Tool support**
 - **Connectivity**
 - **Domain specific language (DSL)**

- 1) Systems Integration
- 2) Integration Frameworks
- 3) Spring Integration**
- 4) Mule ESB
- 5) Apache Camel
- 6) And the Winner is ...

AMQP
Feed
File
FTP(S)
GemFire
HTTP
TCP
UDP
JDBC
JMS
Mail
MongoDB
Redis
RMI
SFTP
Stream
Twitter
Web Service
XML
XMPP

How to create own custom adapter?



Dashboards ▾ Projects ▾ Issues ▾ Agile ▾

Spring Integration Samples / INTSAMPLES-73

There is no sample how to create an own custom adapter

Comment Planning Board Task Board More Actions ▾

▼ Details

Type:	 Task	Status:	 Open
Priority:	↓ Minor	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Component/s:	None	Security Level:	Public
Labels:	None		

Spring Forum Reference:: <http://forum.springsource.org/showthread.php?125793-How-to-create-a-new-custom-adapter-for-Spring-Integration&p=410641#post410641>

XML



(Not production-ready yet)



(Not production-ready yet)


```
<file:inbound-channel-adapter
  id="incomingOrders"
  directory="file:incomingOrders"/>

<payload-type-router input-channel="incomingOrders">
  <mapping type="com.kw.DvdOrder"
    channel="dvdOrders" />
  <mapping type="com.kw.VideogameOrder"
    channel="videogameOrders" />
  <mapping type="com.kw.OtherOrder"
    channel="otherOrders" />
</payload-type-router>

<file:outbound-channel-adapter
  id="dvdOrders"
  directory="dvdOrders"/>

<jms:outbound-channel-adapter
  id="videogamesOrders"
  destination="videogameOrdersQueue"
  channel="videogamesOrders"/>

<logging-channel-adapter id="otherOrders" level="INFO"/>
```

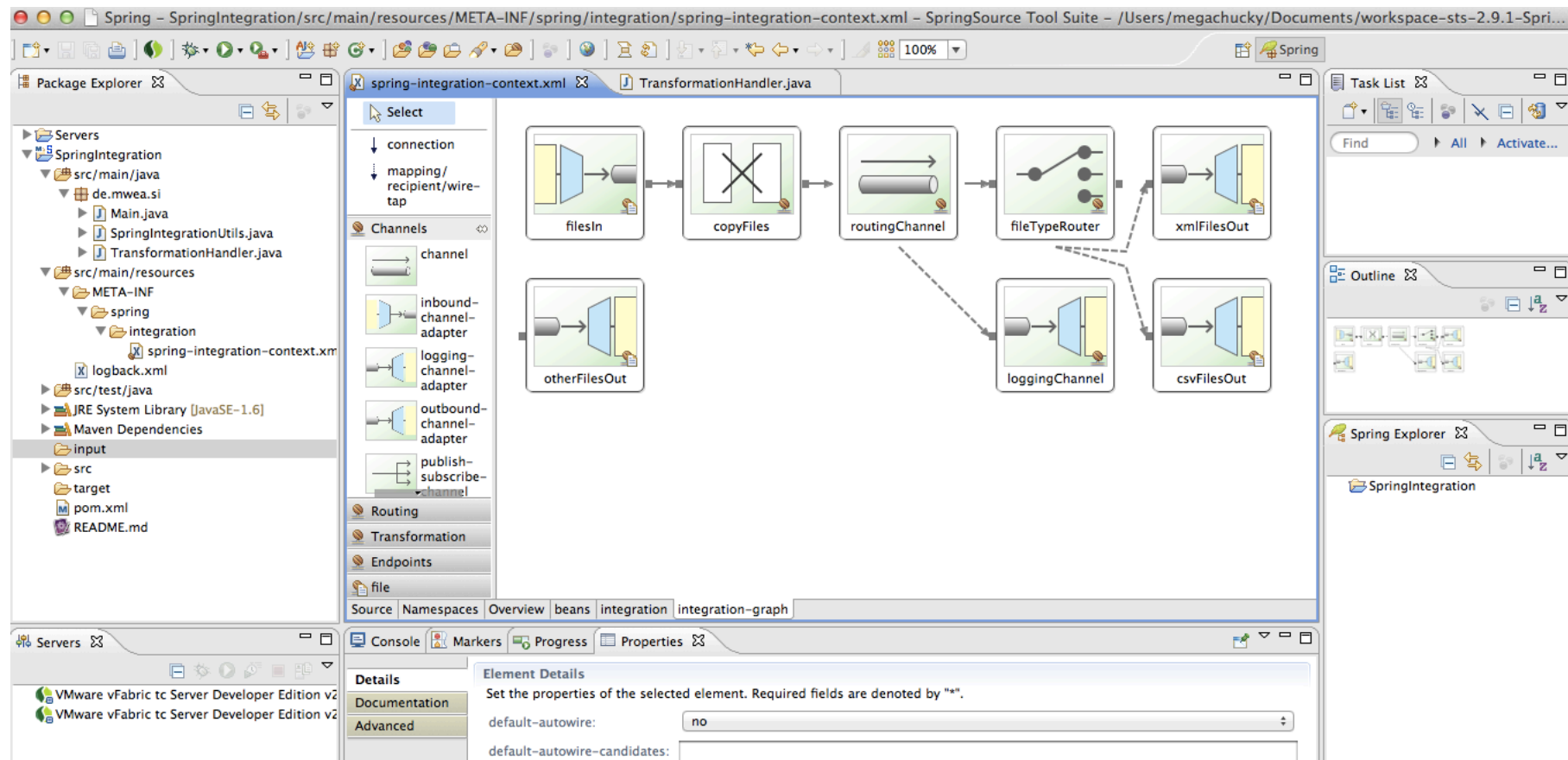
```
val messageFlow =  
    filter.using { payload: String => payload == "World" } -->  
    transform.using { payload: String => "Hello " + payload } -->  
    handle.using { payload: String => println(payload) }  
  
messageFlow.send("World")
```

<http://blog.springsource.org/2012/03/05/introducing-spring-integration-scala-dsl/>





Spring Integration in Action



Pro

- Visual Designer for Spring Integration flows
- Vice versa Editing (code generation vs. coding by hand)
- Open Source (part of Spring IDE at github)
- Simple Eclipse plugin – „Just Spring Integration“
- Export of flow diagrams

Contra

- Non-intuitive usability
- Unclear diagrams
- Immature (e.g. missing documentation, problems with code examples)

- 1) Systems Integration
- 2) Integration Frameworks
- 3) Spring Integration
- 4) Mule ESB**
- 5) Apache Camel
- 6) And the Winner is ...



Embedding Mule in a Java Application or Webapp

This page describes how to start and stop Mule from a Java application or to embed it in a Webapp (such as a JSP or servlet), and how to interact with Mule from your code in both scenarios.

Starting Mule from a Java Application

To start Mule from any Java application, you can call one of its configuration builders. To use Mule XML configuration:

```
DefaultMuleContextFactory muleContextFactory = new DefaultMuleContextFactory();  
SpringXmlConfigurationBuilder configBuilder = new SpringXmlConfigurationBuilder("mule-c  
muleContext = muleContextFactory.createMuleContext(configBuilder);
```

Make sure you store a reference to the MuleContext, as you will need it to stop Mule.

If you have multiple configuration files, you can provide a comma-separated list or an array of configuration files:

```
SpringXmlConfigurationBuilder configBuilder =  
    new SpringXmlConfigurationBuilder(new String[] { "mule-config.xml", "another-confi
```

You then call the `start` method to start the server:

```
muleContext.start();
```

Stopping Mule from a Java Application

To stop Mule, you stop its context like this:

```
muleContext.stop();  
muleContext.dispose();
```

<http://www.mulesoft.org/documentation/display/MULE3USER/Embedding+Mule+in+a+Java+Application+or+Webapp>

AS400 Data Queue

Abdera

Amazon SQS

jBPM

CICS CTG

CXF

Email

FTP

Hibernate

HTTP/S

Legs4Mule

IMAP/S

Servlet

SFTP

SMTP/S

SOAP

STDIO

TCP

UDP

VM

XMPP

WebSphere MQ

WSDL

Atom

Base64 encoded

Byte arrays

CSV

Encrypted

GZIP

Hex Strings

HTML/ XHTML

Java Objects

JSON

EDI

COBOL Copybook

XML

Amazon S3

Authorize.net

Apple Push

Bit.ly

CMIS

CyberSource

Facebook

Flickr

HBase

Magento

JCR

JDBC

Jersey

Jetty

JMS

LDAP

Multicast

POP3/S

Quartz

Restlet

RMI

SAP

**Many more Connectors +
easy to create own Connectors**

Several Proprietary Connectors available, for instance:



OSGi? No Thanks

 Ross Mason on Tuesday, November 9, 2010
[41 Comments](#)

114   3  1  36  11

There have been bubbles of interest about [OSGi](#) in the Java community over recent years. I for one was very excited about the advent of a modular Java platform that freed us from the classloader issues in the JDK manifested best by Jakarta commons-logging (clogging our app servers).

OSGi was going to change everything, dependencies would be completely isolated (no more tripping over conflicting dependency versions), visibility and would be strictly enforced between each 'bundle'. I was so bought into the promise of OSGi, like many others, I focused our engineering team on making Mule OSGi-enabled.

„OSGi adds another complexity to building applications. [...] OSGi is a great specification for middleware vendors, but a terrible specification for the end user.“

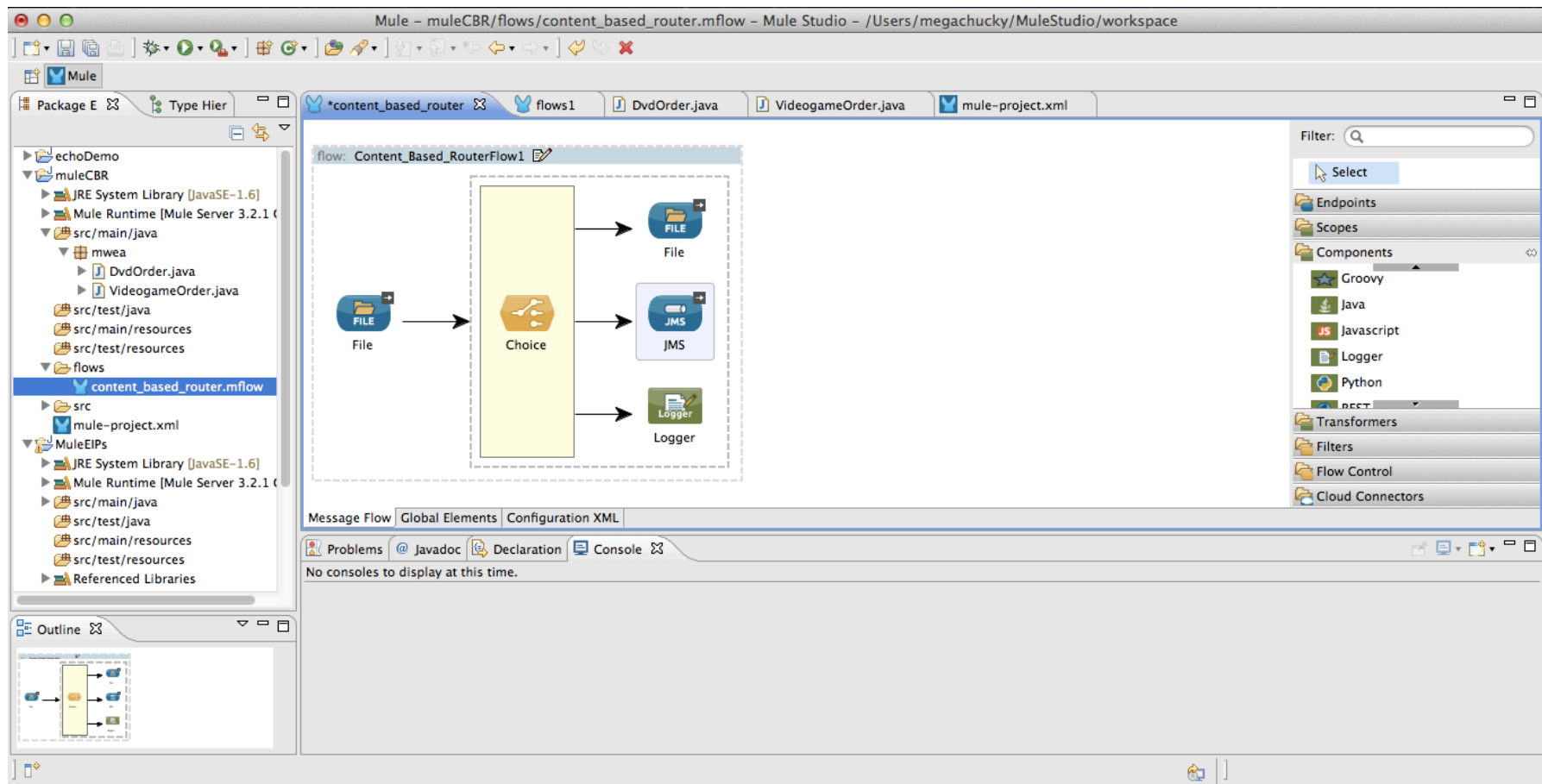
Ross Mason, MuleSoft, November 2010

XML

```
<flow name="muleFlow">
  <file:inbound-endpoint path="incomingOrders"/>
  <choice>
    <when expression="payload instanceof com.kw.DvdOrder"
      evaluator="groovy">
      <file:outbound-endpoint path="incoming/dvdOrders"/>
    </when>
    <when expression="payload instanceof com.kw.DvdOrder"
      evaluator="groovy">
      <jms:outbound-endpoint queue="videogameOrdersQueue"/>
    </when>
    <otherwise>
      <logger level="INFO"/>
    </otherwise>
  </choice>
</flow>
```



Mule in Action



Pro

- Visual Designer for Mule Flows
- Visual „Live Monitoring“
- Vice versa Editing (Code generation vs. coding by hand)
- Simple Eclipse plugin – „Just Mule“
- Intuitive GUI

Contra

- Proprietary
- Subscription required for many enterprise features (such as monitoring)

- 1) Systems Integration
- 2) Integration Frameworks
- 3) Spring Integration
- 4) Mule ESB
- 5) Apache Camel**
- 6) And the Winner is ...



XML



(Not production-ready yet)

SQL TCP LDAP XSLT
SMTP JMS
Netty Jetty
RMI
FTP Lucene JDBC EJB
Twitter
Bean-Validation MQ IRC
JMX Quartz
CXF RSS AMQP jclouds
AWS Atom
Akka
File HTTP
MongoDB

**Many more Components +
easy to create own Components**

Mailing Lists?

Blogs?

Articles?

Conference talks?

ESBs?

Jobs?

...



Mailing Lists?

Blogs?

Articles?

Conference talks?

ESBs?

JOBS?

...



**Even JBoss' new ESB
„SwitchYard“ uses Camel inside!**



```
<route>
  <from uri="file:incomingOrders"/>
  <choice>
    <when>
      <simple>${in.header.type} is 'com.kw.DvdOrder'</simple>
      <to uri="file:incoming/dvdOrders"/>
    </when>
    <when>
      <simple>${in.header.type} is 'com.kw.VideogameOrder'</simple>
      <to uri="jms:videogameOrdersQueue"/>
    </when>
    <otherwise>
      <to uri="log:OtherOrders"/>
    </otherwise>
  </choice>
</route>
```

```
from("file:incomingOrders ")
    .choice()
        .when(body().isInstanceOf(com.kw.DvdOrder.class))
            .to("file:incoming/dvdOrders")
        .when(body().isInstanceOf(com.kw.VideogameOrder.class))
            .to("jms:videogameOrdersQueue ")
        .otherwise()
            .to("mock:OtherOrders ");
```



Apache Camel in Action

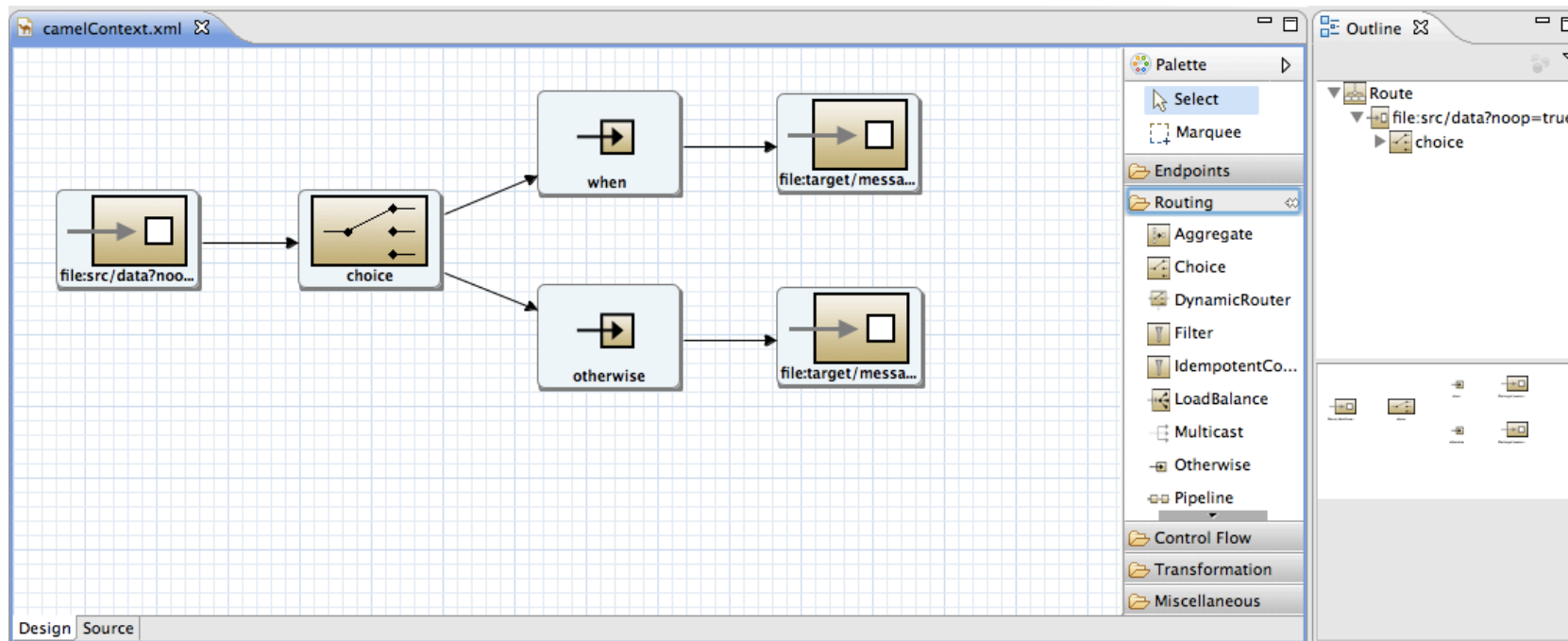


Fuse IDE

vs.

Talend Open Studio for ESB



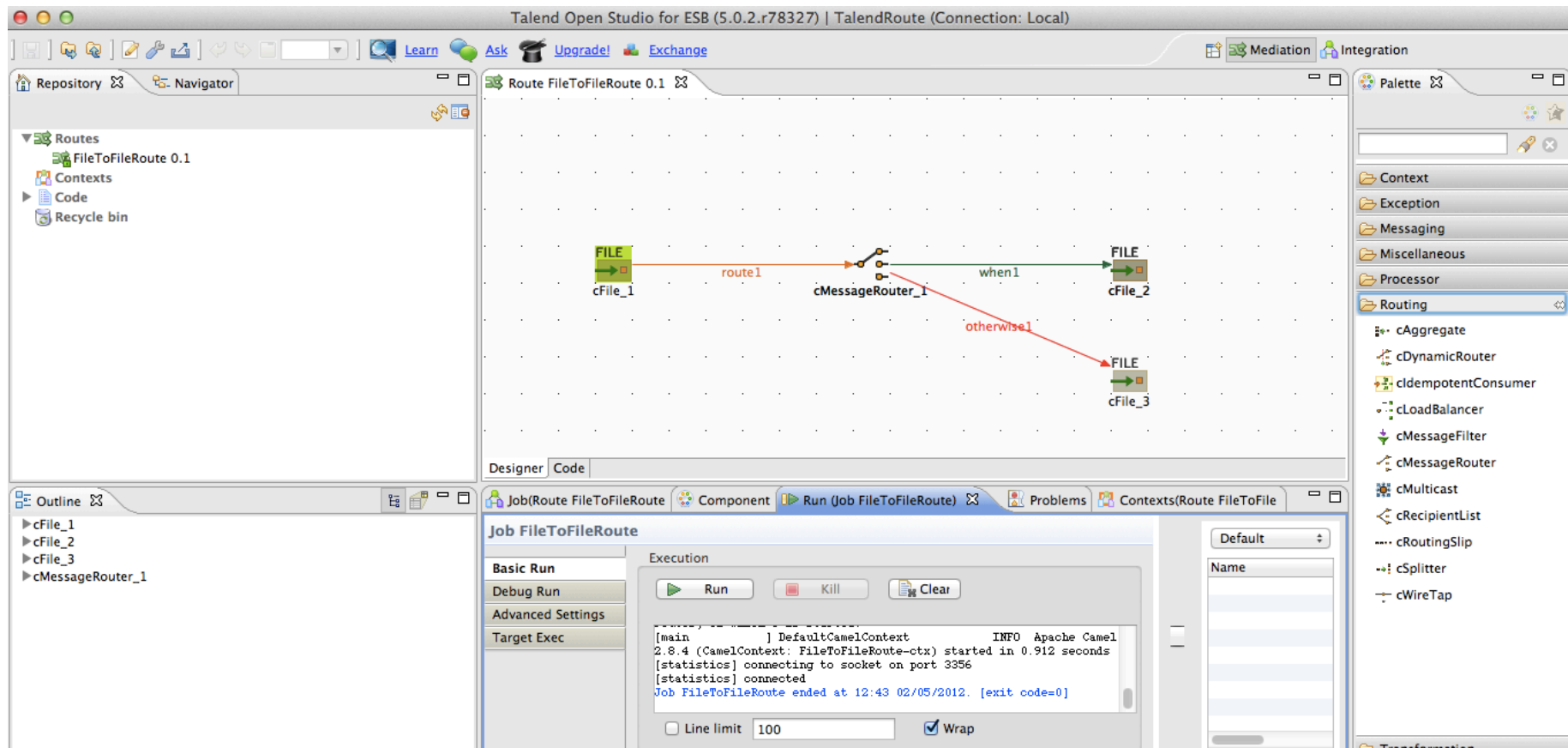


Pro

- Visual Designer for Camel routes
- Visual „Live Monitoring“ (for debugging, browsing, tracing)
- Vice versa Editing (code generation vs. coding by-hand)
- Just a simple Eclipse-Plugin – „Just Camel“
- Intuitive GUI
- Export of route diagrams
- JUnit Test Wizard (generates scaffolding for tests)

Contra

- Proprietary
- Subscription required
- Only XML DSL (this is not really a contra, because only Camel offers other DSLs)



Pro

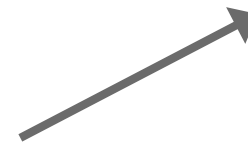
- Visual Designer for Camel routes
- Visual „Live Monitoring“
- Open Source (at github)
- Community Edition (not all features)
- Zero Coding

Contra

- Zero Coding
- Only Java DSL (plus a lot of boilerplate code)
- No vice versa code editing (only code generation)
- No intuitive user interface
 - => no simple Eclipse plugin
 - => based on Eclipse, but it is more an own product (1GB)
 - => **full ESB only**

- 1) Systems Integration
- 2) Integration Frameworks
- 3) Spring Integration
- 4) Mule ESB
- 5) Apache Camel
- 6) And the Winner is ...

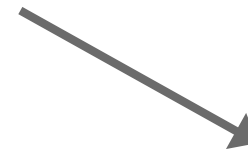
And the Winner is ...




spring
Integration



Mule ESB



Apache

Camel


spring
Integration

Mule ESB


Apache
Camel



- Spring Project
- „Typical“ JVM Technologies
- No additional Framework




spring
Integration

Mule ESB

Apache

Camel


spring
Integration

- One of the available proprietary Connectors is required



Mule ESB

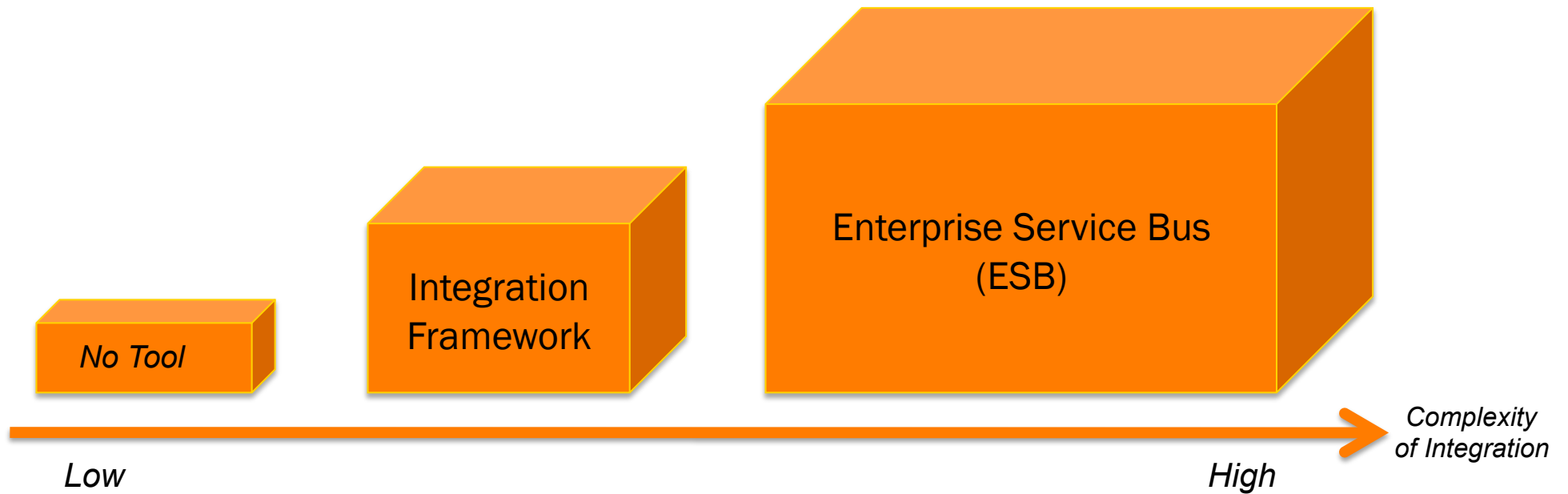

Apache
Camel

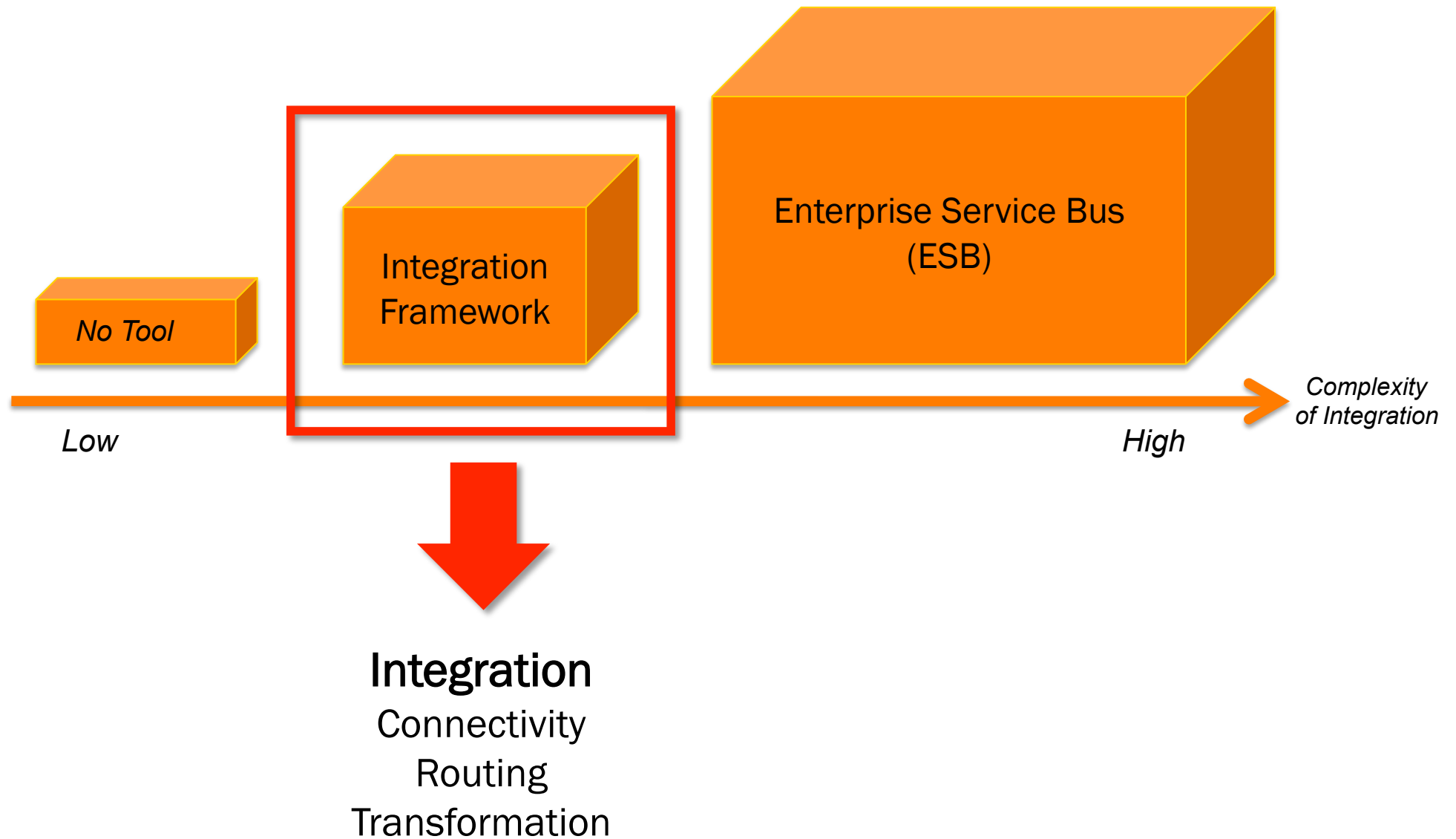

spring
Integration

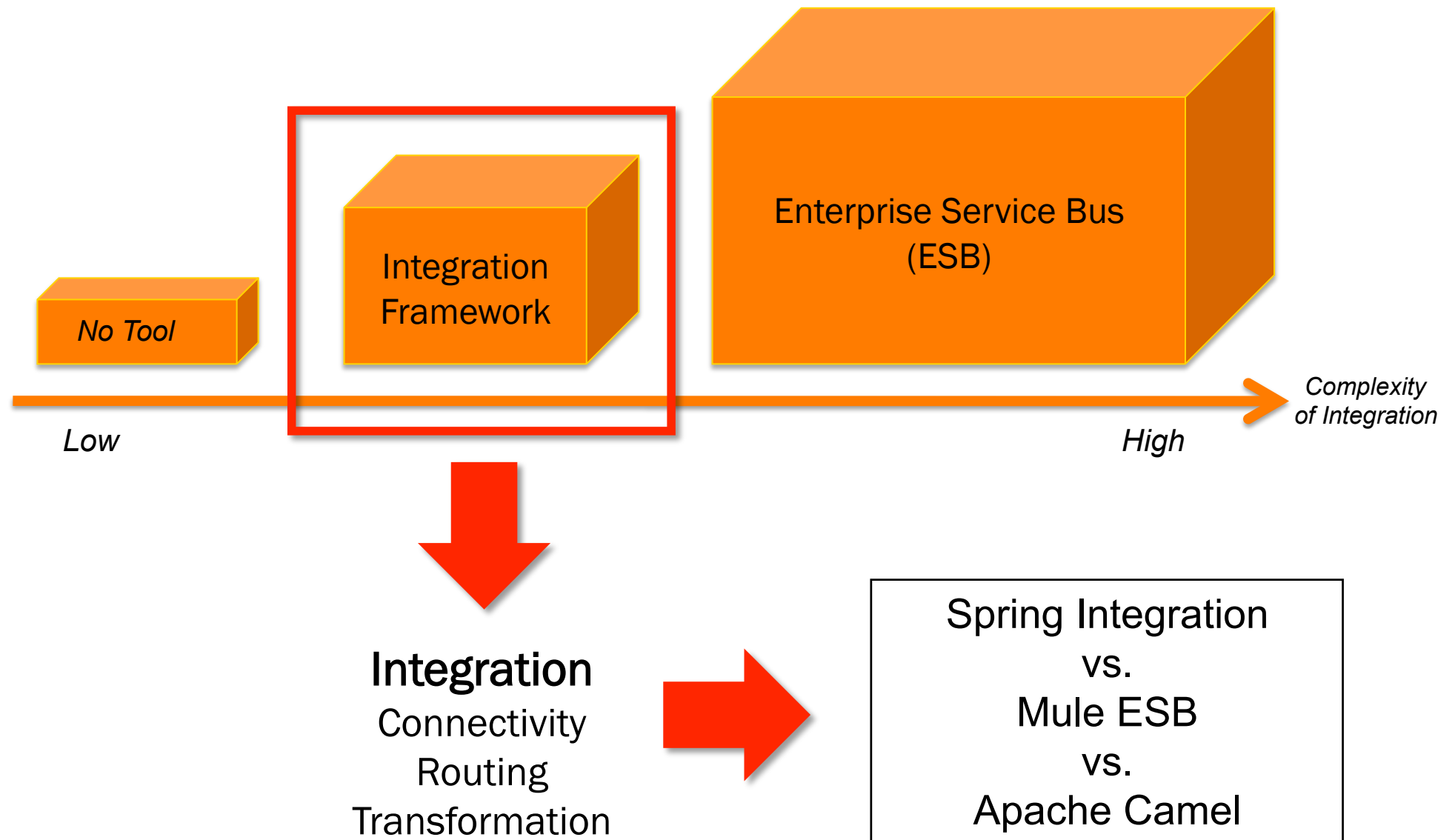
Mule ESB

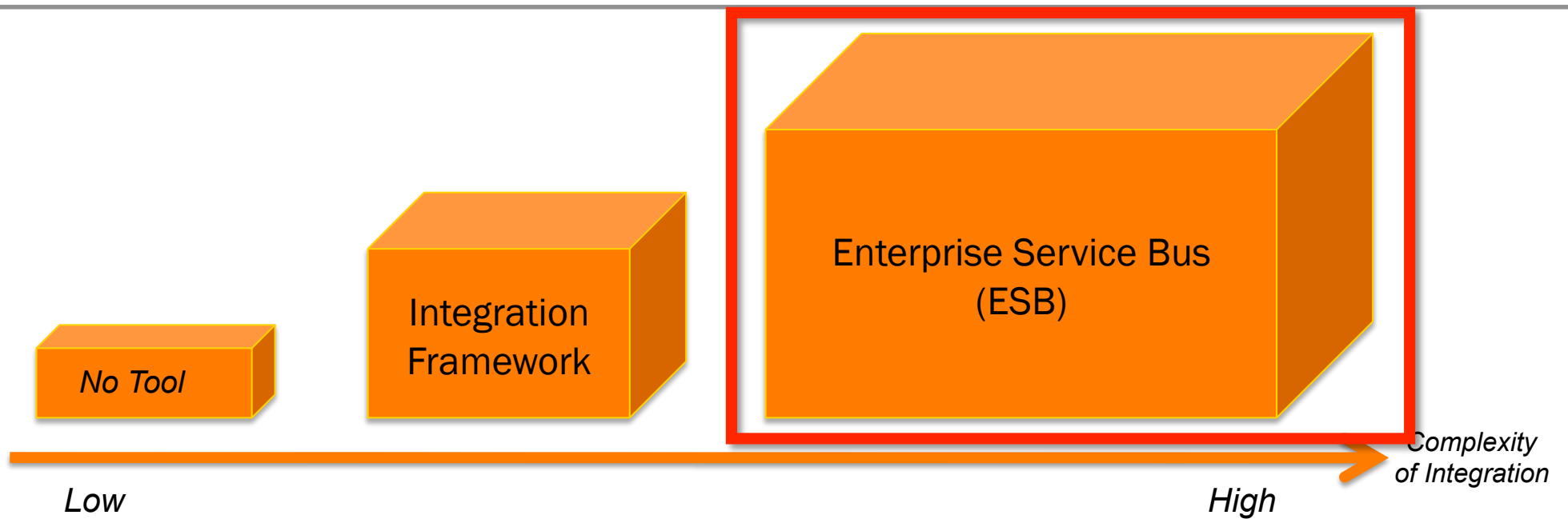
- In all other Cases

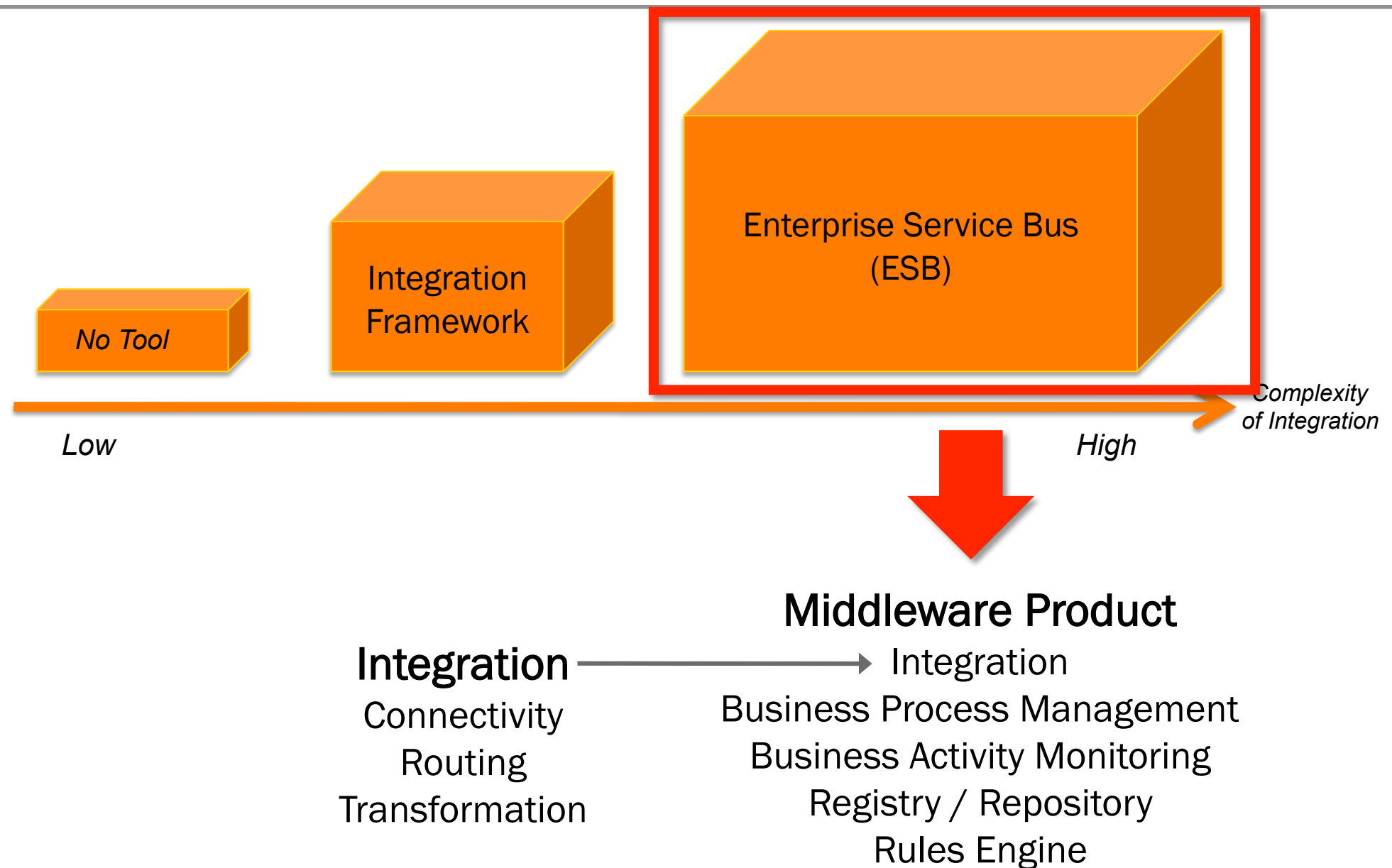


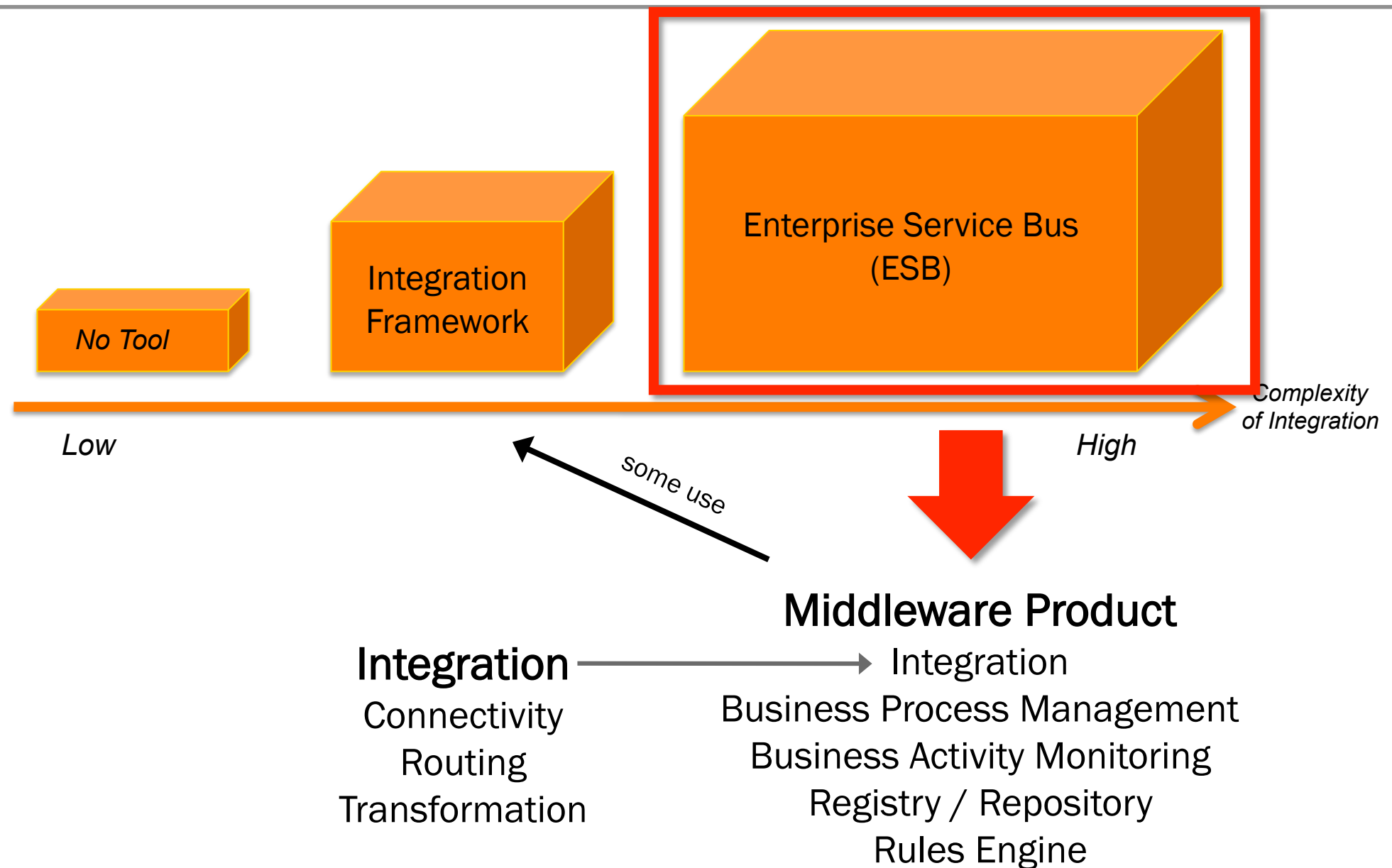


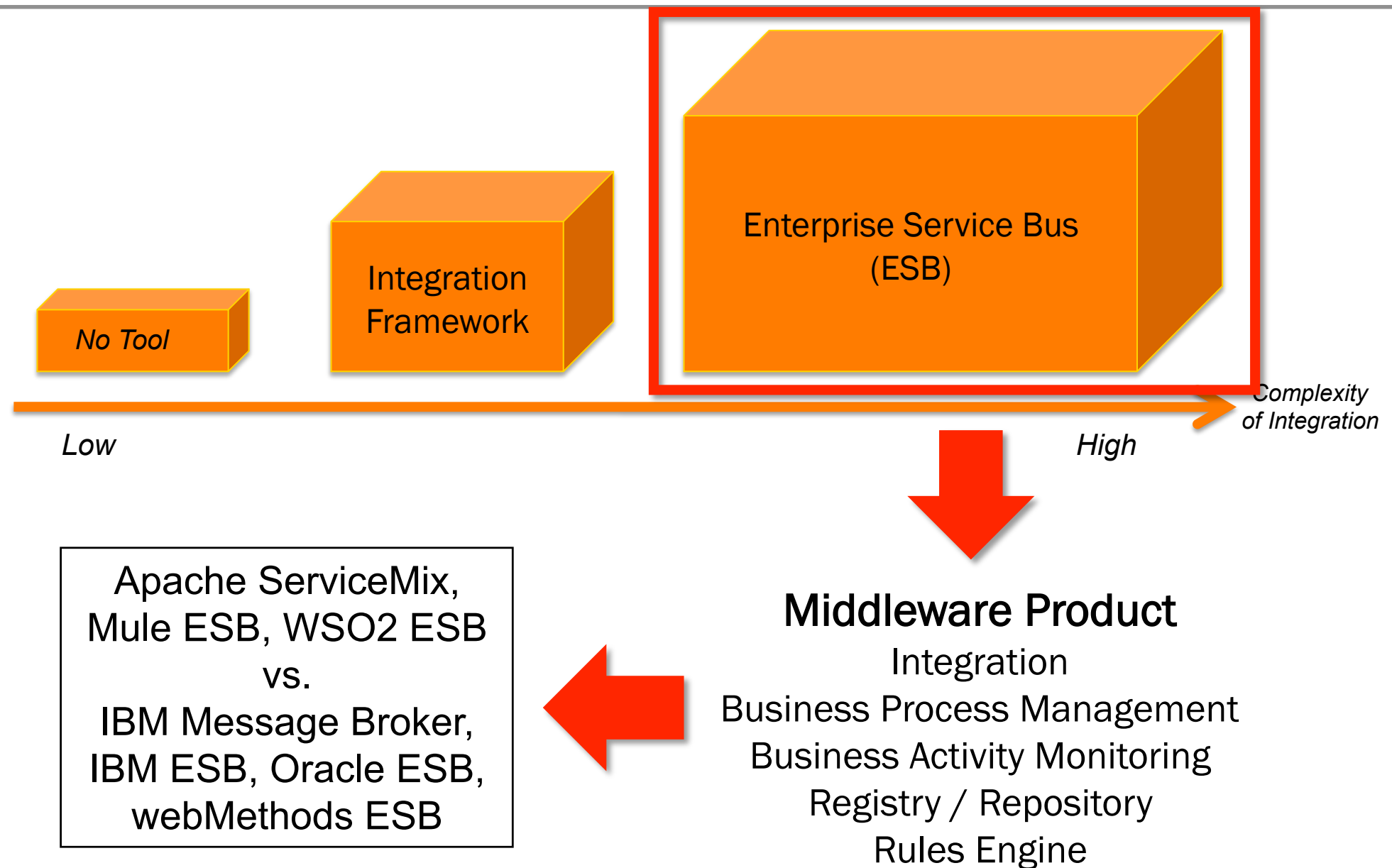




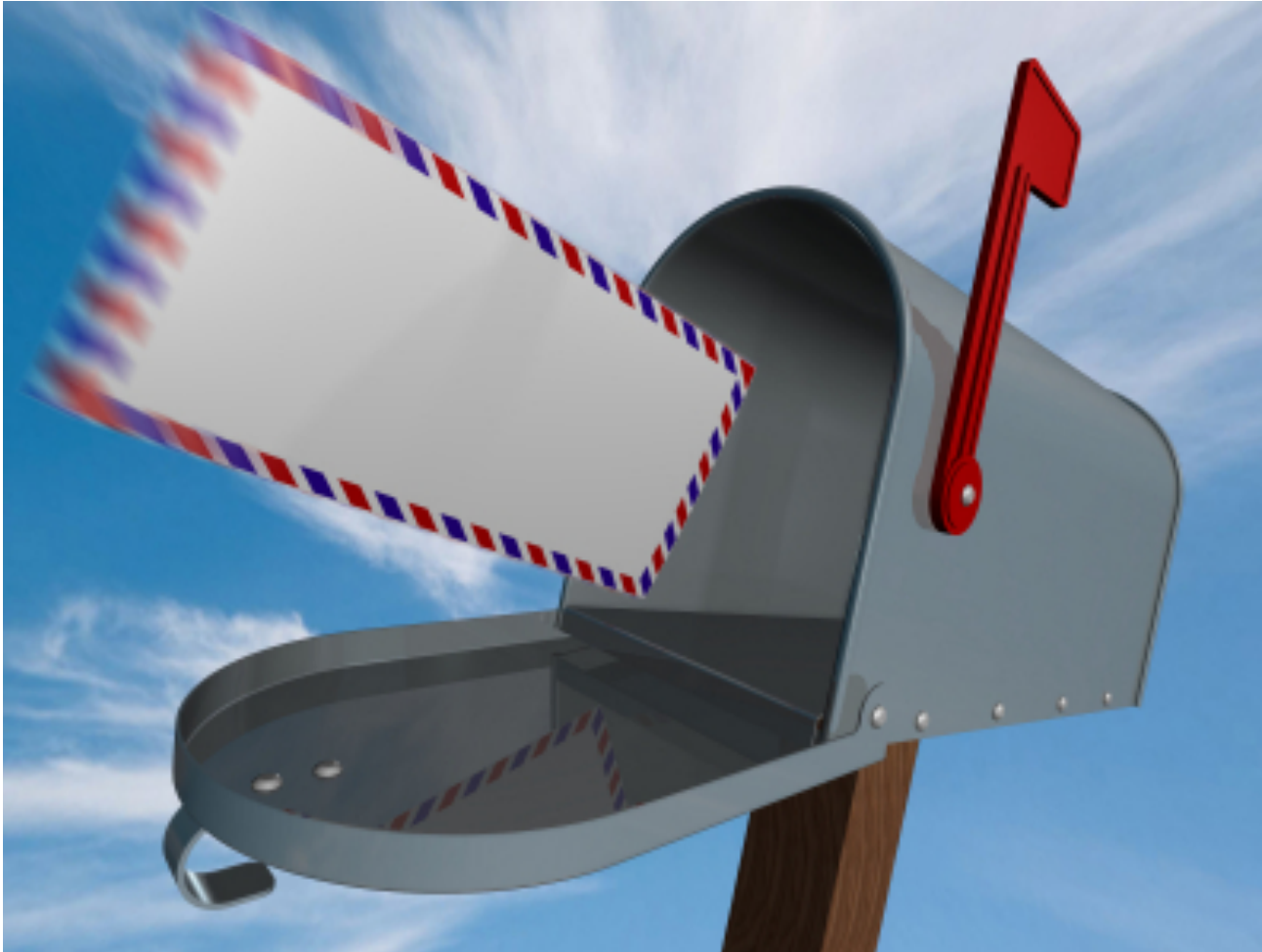


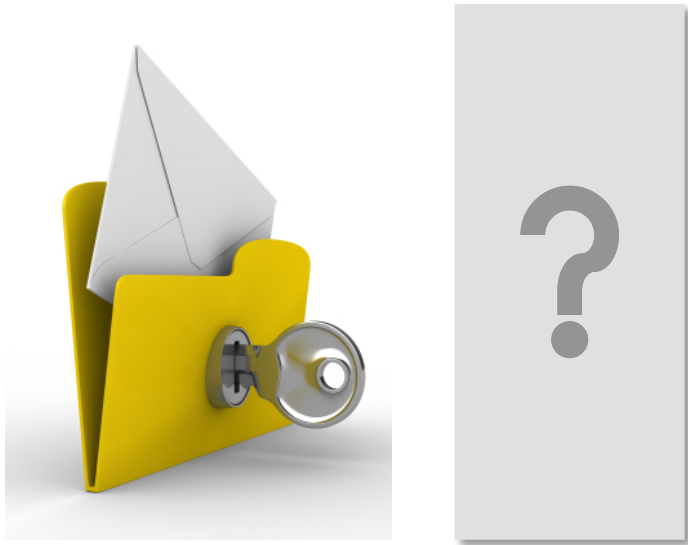






Did you get the Key Messages?





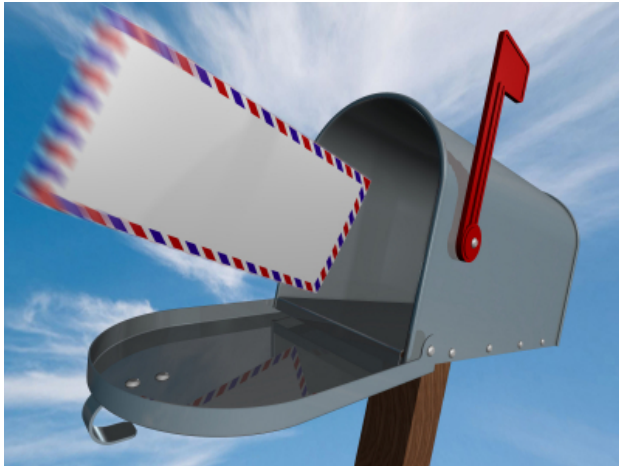
Do not reinvent the „integration wheel“! 

There are some good alternatives for Integration! 

Often an ESB is the wrong Choice! 

Did you get the Key Messages?

MaibornWolff  et al





Thank you for your Attention. Any Questions?



or

Mule ESB

or



=

Smart EAI

Kai Wähler

MaibornWolff et al: www.mwea.de

Email: kai.waehner@mwea.de

Twitter: @KaiWaehner

Blog: www.kai-waehner.de/blog

