Heterogeneous Systems Integration

Iván García-Magariño

Introduction

- IoT often has heterogeneous applications
- EAI (*Enterprise Application Integration*) facilitates systems integration
- Service integration practice in different languages
- Standard documents for information exchange (e.g. XML)

EAI (*Enterprise Application Integration*) (1)

- EAI uses intermediate software known as *middleware*
- Integration with EAI can reduce time-to-market for novel products.
- EAI allows you to recover your investment by reusing different applications.
- EAI allows you to overcome chaotic architectures by managing their complexity, properly training your personnel

EAI (2)

- EAI is the creation of enterprise solutions by combining applications using middleware
- Middleware are the application-independent services that mediate between applications.
- EAI with middleware allows:
  - Package functionality as services for other applications (e.g. an IoT garbage collection application can determine which is the nearest empty garbage can)
  - Applications can share information with other applications
  - Coordinate business processes with different applications

EAI facilitates new critical solutions

- Improve customer relations
  - It allows a more global vision of each customer, and the customer perceives an integrated service.
- Improved supply chain relationships
  - XML (*eXtensible Markup Language*) facilitates communication between applications.
- EAI improve internal processes
- Reduces time to market for new applications

The Web at EAI

- The web has been a key factor in the rise of the EAI
- Dot-com refers to companies that do most of their business on the Internet.
- The web is ubiquitous and enables rapid business growth

Examples of Middleware

- Message Oriented Middleware (MOM)
- Common Object Request Broker Architecture (CORBA)
- Microsoft Distributed Component Object Model (DCOM)
- Enterprise Java Beans (EJB)

Barriers to effective EAI

- Chaotic architectures
  - APIs (Application Programming Interface) allow you to provide other applications with access to your functionality or data.
- Lack of staff skills
  - Middleware skills such as MOM, CORBA, DCOM and EJB are required.
- Safety Failures

Types of Integration (1)

- Integration can occur at three points:
  - Presentation
  - Functionality
  - Data layer
- One objective of integration is to reduce coupling
- Tightly coupled integration can be a serious maintenance problem
- Integration into the presentation can be easy but very limiting

Types of Integration (2)

- Data-level integration provides more comprehensive solutions than presentation-level integration, but may require rewriting some of the software.
- Functional integration is the most important model but the most complex one
- There are three types of functional integration:
  - Consistency of data
  - Multi-step processes (multistep)
  - Plug-and-play components

Integration Model

- An integration model is an approach and setup for integrating software
- The most relevant aspects to be taken into account are:
  - Simplicity of integration
  - Reusability of integration for different configurations
  - Breadth of possible approaches to integration
  - Experience required to perform the integration
- An integration model defines how the applications will be integrated indicating the nature and mechanisms of integration

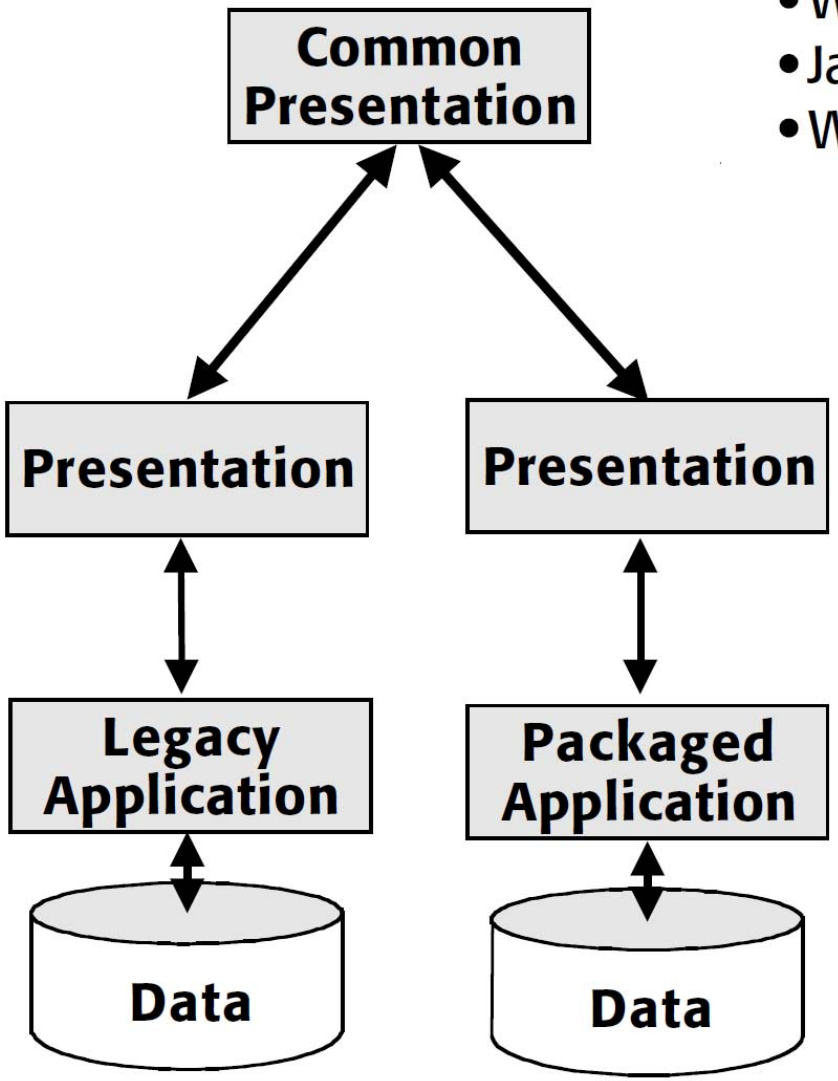
Presentation Integration Model (1)

- The presentation integration model integrates new software through existing presentations.
- This is usually done to create a new user interface.
- Can be used to integrate various applications

Presentation Integration Model (2)

- When to use it:
  - To provide more usable UIs (user interfaces)
  - For the user to perceive as one application the composition of several applications, simply with a common UI.
  - When the only useful point of integration is through its presentation.
- Examples of Integration:
  - Providing a Windows interface for accessing various applications
  - Providing a unified HTML interface for various applications
  - Providing a Java-based interface for access to multiple applications

Presentation Integration Model(3)



- Web browser
- Java
- Windows GUI

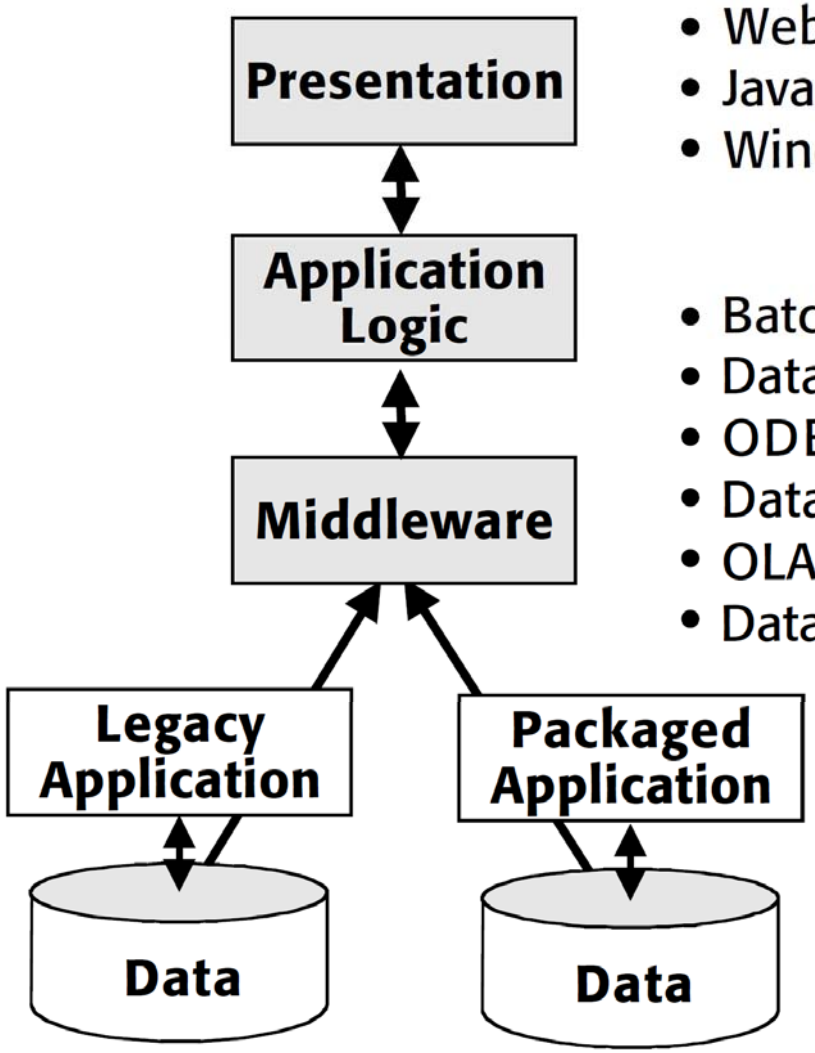
Data Integration Model

- The data integration model directly integrates databases and data structures.
- Sometimes, it is as simple as accessing the database management system (DBMS) of another application for the coordination
- A data access middleware facilitates database access through the use and creation of connectors.

Tools and middleware for data integration

- Batch file transfer
- ODBC (Open Database connectivity)
- Middleware for access to distributed databases
- Data transformation: It allows passing data from one database to another destination for use and transfer of information.

Integration in the data access layer



- Web browser
- Java
- Windows GUI

- Batch file transfer
- Database gateway
- ODBC
- Data warehousing
- OLAP
- Data transformation

When to use integration at the data layer

- To combine information from different sources for analysis and decision making
- To provide multiple applications with read access to a common source of information
- To allow information to be extracted from one source of information, and reformatted and updated in another source of information.

Functional Integration Model

- Business logic is the implementation of the business process in a programming language.
- Business logic contains the rules necessary to properly interpret or construct the data and is not always available through presentation.
- Functional integration integrates business logic

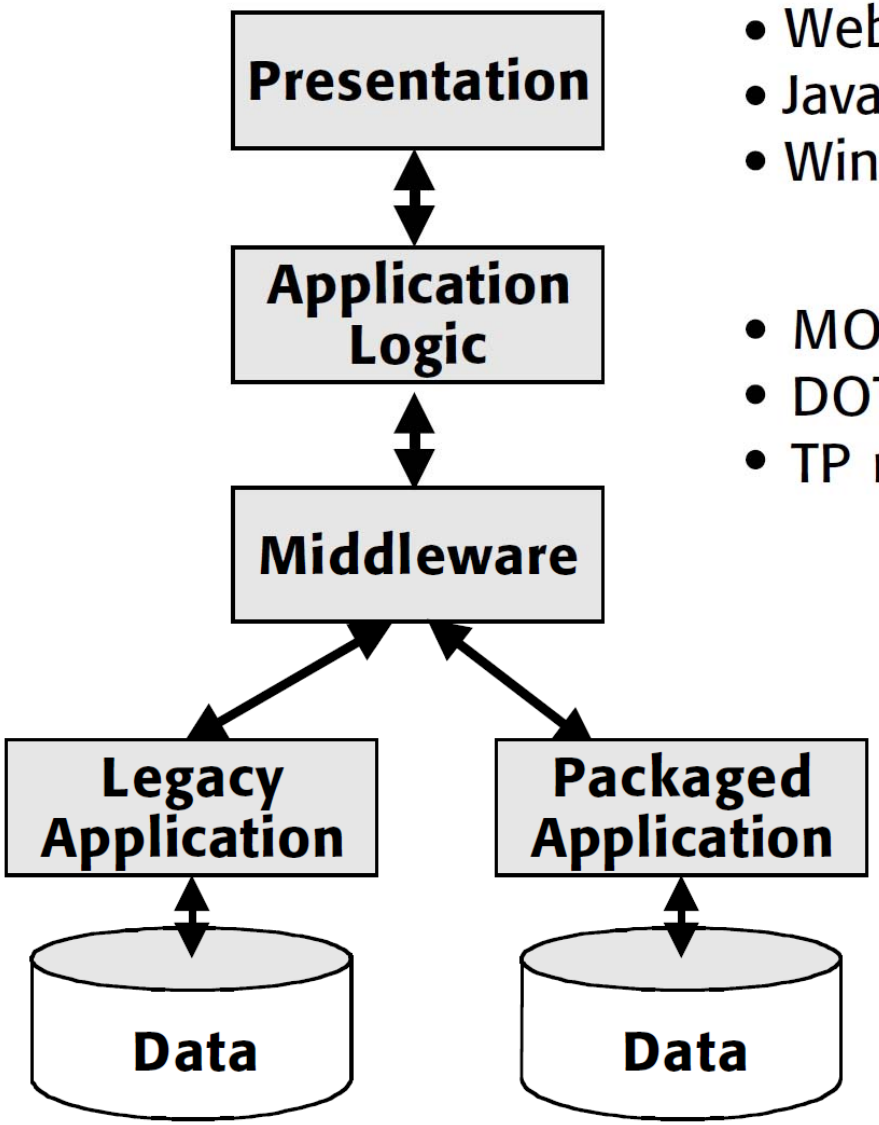
Distributed processing middleware

- Distributed processing middleware is software that facilitates the communication of requests between different software components through the use of interfaces or messages.
- Provides the execution environment for handling requests between software components

Types of distributed processing middleware

- Message-oriented middleware (MOM): Performs integration by means of message passing between applications.
- Distributed object technology: This middleware allows software components to be used as objects. A well-known example is CORBA
- Processing transaction monitors: Enables distributed architectures by performing transactions with concepts such as "two-phase commit" of a transaction.

Functional Integration



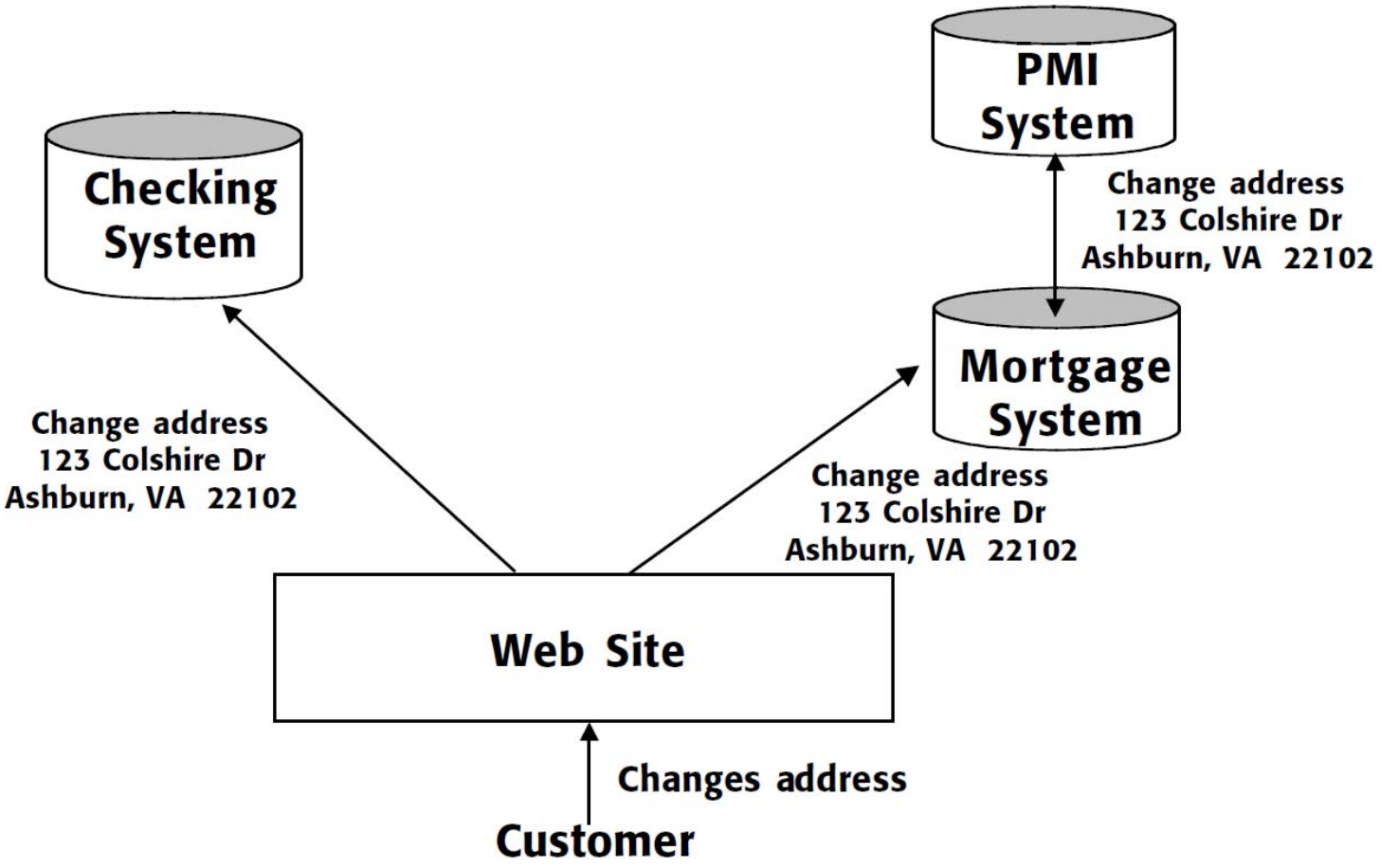
- Web browser
- Java
- Windows GUI

- MOM
- DOT
- TP monitor

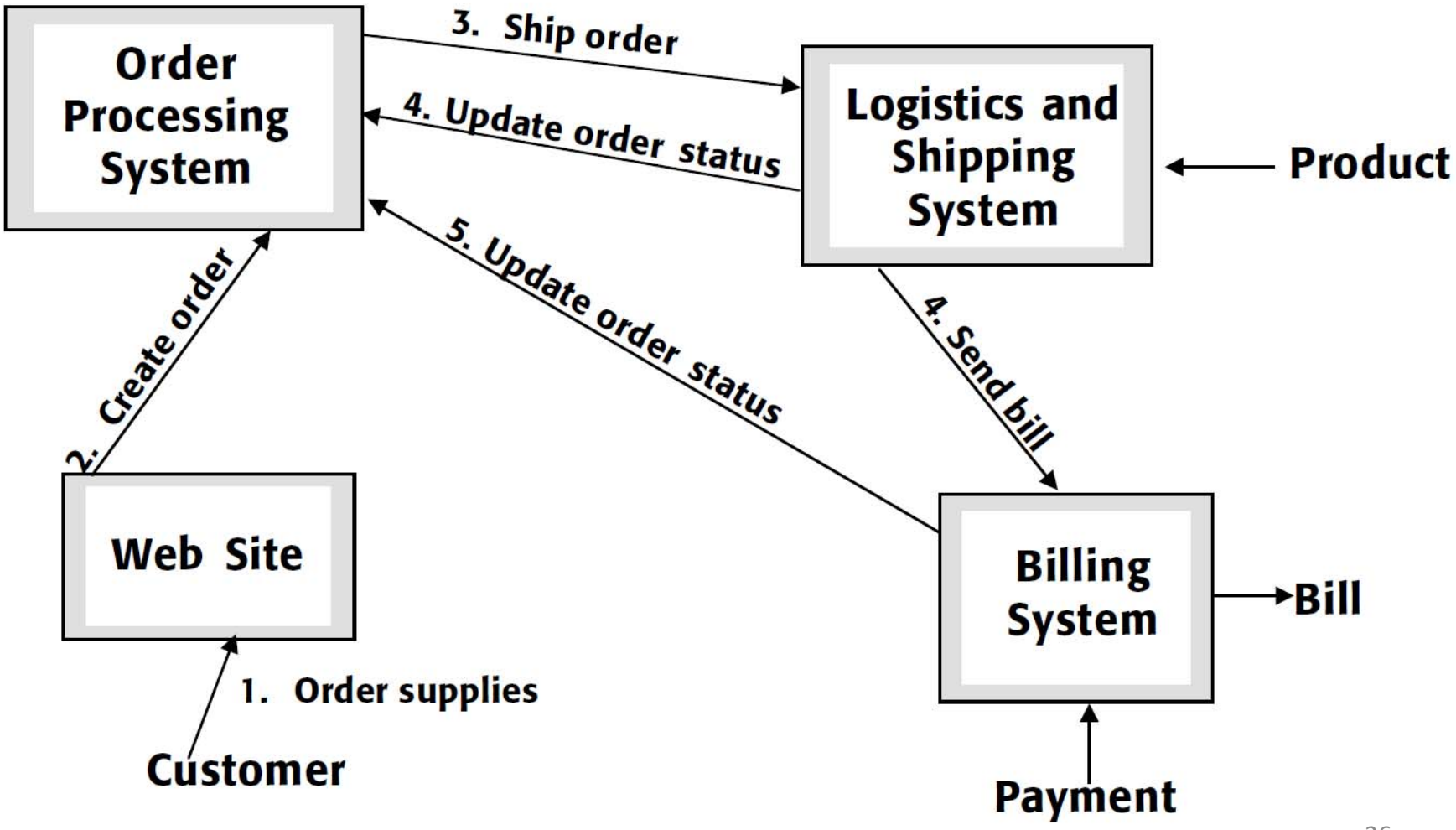
Types of functional integration

- Data consistency: The updating of information in one or more sources is coordinated among the various integrated applications.
- Multi-step processes: A set of applications are coordinated to perform coordinated actions executed in the different integrated applications.
- Plug-and-play components: Creation of reusable interfaces that simplify the creation of new applications.

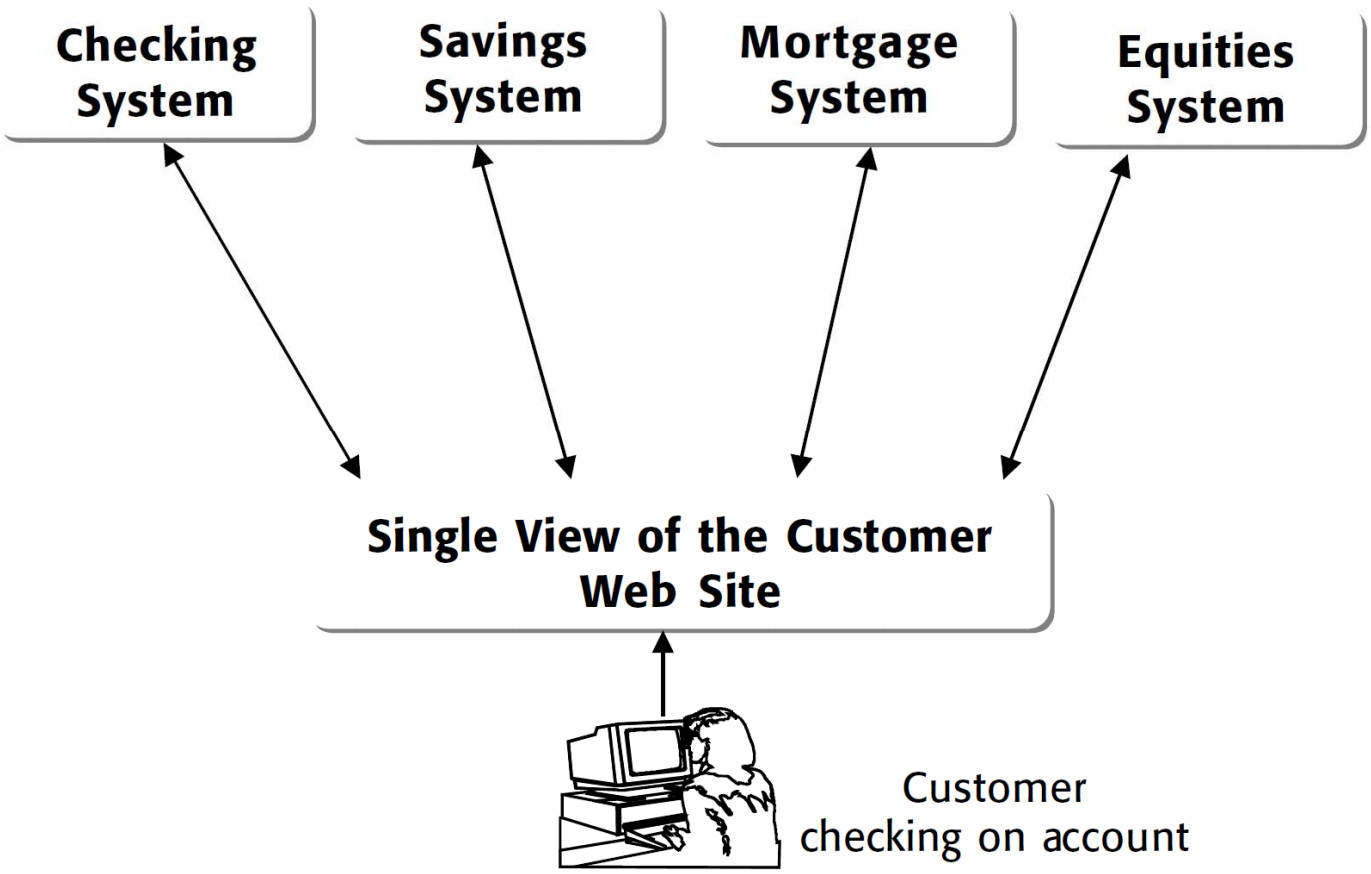
Example of Functional Integration for data consistency



Example of multi-step Process Integration



Connect-and-Ready Component Integration Example



Practical example of connector between Java and Python

- Download JPython jar file (standalone version)
- Creating a Java project with NetBeans or similar
- Include Jpython jar file in Java project
- Creating a class in Java (see following slides)
- Creating a Java program (see the following slides)
- Calling from Java program to Python program

Example of Class that facilitates communication between Java and Py

```
package examplejavapythonv1;
import org.python.core.PyInstance;
import org.python.util.PythonInterpreter;
/** * Example of using Connector of between Java and Python
 * @author Iván García-Magariño
 */
public class InterpreterExample
{
   PythonInterpreter interpreter = null;

   public InterpreterExample()
   {
      PythonInterpreter.initialize(System.getProperties(),
                                   System.getProperties(), new String[0]);
      this.interpreter = new PythonInterpreter();
   }

   void execfile( final String fileName )
   {
      this.interpreter.execfile(fileName);
   }

   PyInstance createClass( final String className, final String opts )
   {
      return (PyInstance) this.interpreter.eval(className + "(" + opts + ")");
   }
```

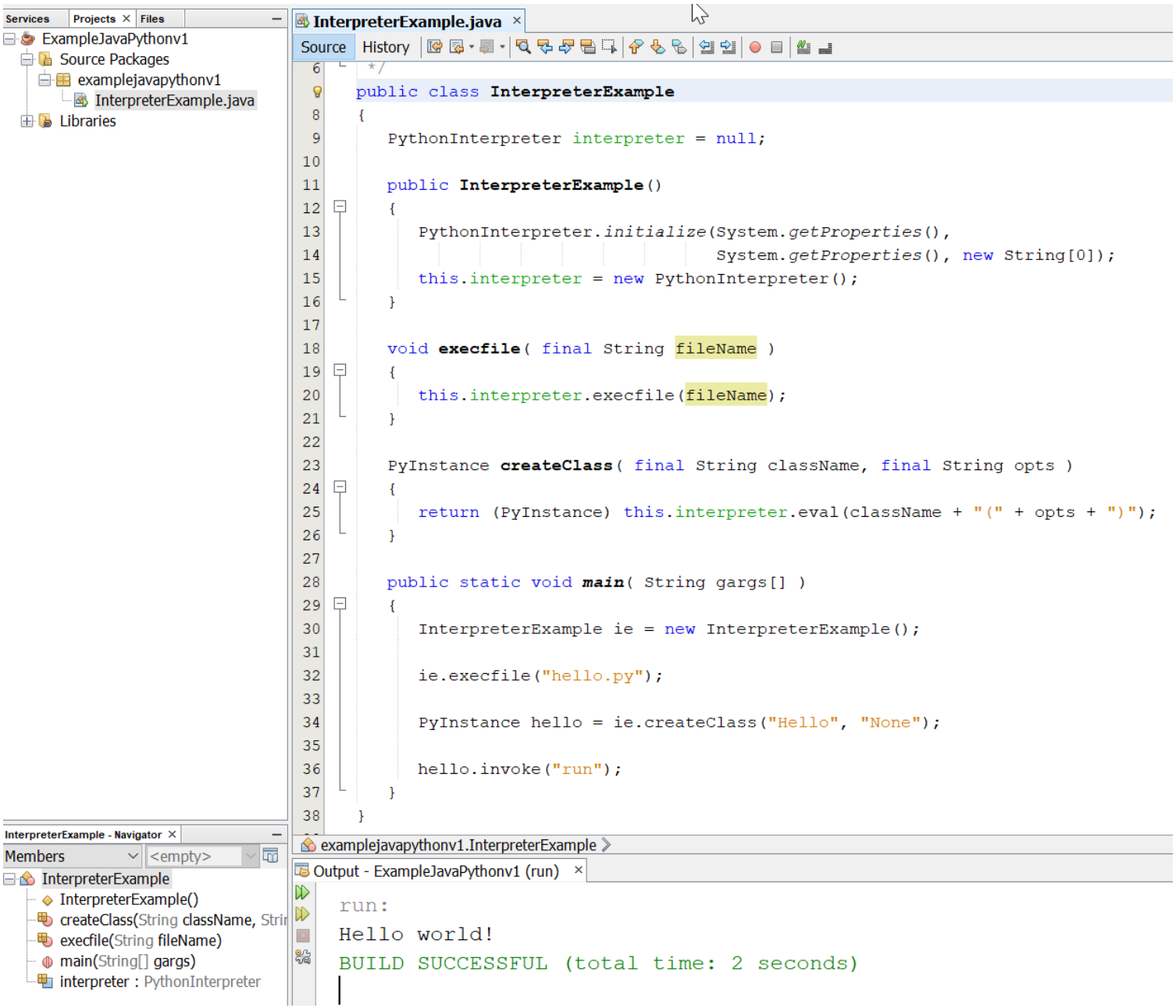
Using the interpreter to call the Python program from Java

```
    public static void main( String gargs[] )
    {
        InterpreterExample ie = new InterpreterExample();

        ie.execfile("hello.py");

        PyInstance hello = ie.createClass("Hello", "None");

        hello.invoke("run");
    }
}
```

You need the code of the previous class

Python program called from Java

```
class Hello:
    __gui = None

    def __init__(self, gui):
        self.__gui = gui

    def run(self):
        print 'Hello world!'
```


Services
Projects
Files
ExampleJavaPythonv1
Source Packages
examplejavapythonv1
InterpreterExample.java
Libraries
InterpreterExample.java
Source
History
*/
public class InterpreterExample
{
PythonInterpreter interpreter = null;
public InterpreterExample()
{
PythonInterpreter.initialize(System.getProperties(),
System.getProperties(), new String[0]);
this.interpreter = new PythonInterpreter();
}
void execfile(final String fileName)
{
this.interpreter.execfile(fileName);
}
PyInstance createClass(final String className, final String opts)
{
return (PyInstance) this.interpreter.eval(className + "(" + opts + ")");
}
public static void main(String gargs[])
{
InterpreterExample ie = new InterpreterExample();
ie.execfile("hello.py");
PyInstance hello = ie.createClass("Hello", "None");
hello.invoke("run");
}
}
InterpreterExample - Navigator
Members
<empty>
InterpreterExample
InterpreterExample()
createClass(String className, Stri
execfile(String fileName)
main(String[] gargs)
interpreter : PythonInterpreter
examplejavapythonv1.InterpreterExample
Output - ExampleJavaPythonv1 (run)
run:
Hello world!
BUILD SUCCESSFUL (total time: 2 seconds)

Disadvantages of Jython connector and alternatives

- Cannot run some libraries
  - For example Scikit-learn
- Alternative:
  - Expose scikit-learn or other libraries in a HTTP/Json service
  - Use microframeworks such as flask, bottle or cornice

Bottle: Python Web Framework

- Bottle's website: http://bottlepy.org/docs/dev/

- Installation, run from console:

```
pip install bottle
```

- Write a python program like the following example (in the next slide)

-

Python program using Bottle

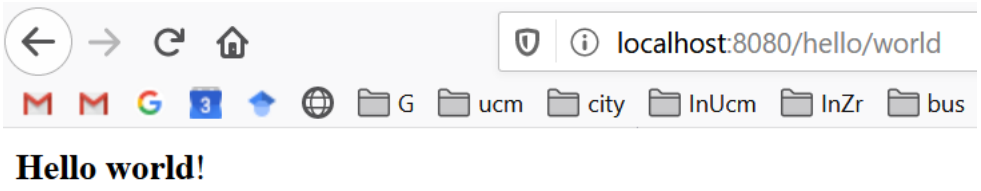
```
from bottle import route, run, template

@route('/hello/<name>')
def index(name):
    return template('<b>Hello {{name}}</b>!', name=name)

run(host='localhost', port=8080)
```

@route indicates the route by ascocicating it with a function

Access from HTTP

Web access:

Example browser access



Python console output when bottle is run

```
Bottle v0.12.17 server starting up (using WSGIRefServer())...
Listening on http://localhost:8080/
Hit Ctrl-C to quit.

127.0.0.1 - - [21/Nov/2019 12:54:11] "GET /hello/world HTTP/1.1" 200 19
127.0.0.1 - - [21/Nov/2019 12:54:11] "GET /favicon.ico HTTP/1.1" 404 742
127.0.0.1 - - [21/Nov/2019 12:59:31] "GET /hello/world HTTP/1.1" 200 19
```

Using the Bottle framework to integrate a neural network with Python

```
from bottle import route, run, template, request
from sklearn.neural_network import MLPClassifier

@route('/integration')
def classify():
    # Creation of the neural network
    clf = MLPClassifier(solver = 'lbfgs', alpha = 1e-5, hidden_layer_sizes=(5, 2), random_state=1)
    # Training the Neural Network
    X=[[0,1], [3,4], [5,3], [7,0], [9,1]]
    Y=[0, 1, 0, 1, 1]
    clf.fit(X, Y)
    # Taking the input point from GET parameters
    x1 = int(request.query['x1'])
    x2 = int(request.query['x2'])
    # Classifing the input
    y = clf.predict([[x1,x2]])[0]
    # Returning the output
    return str(y)

run(host='localhost', port=8080)
```

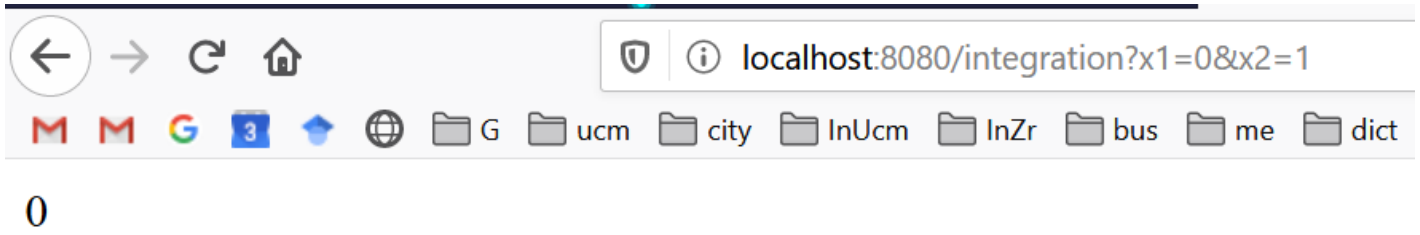
Check the way in which

- **The input GET parameters are received,**
- **Neural network is created/trained/applied with Scikit-learn library, and**
- **Output is returned**

It can be tested in a browser using GET parameters:

- URL with parameters x1 and x2:

http://localhost:8080/integration?x1=0&x2=1



In this case, "0" is the result of sorting (x1,x2) = (0,1)

Integration of a Java System with neural network classification in Python

- In any Java program
- Add GET parameters to the URL
- Make an HTTP request to the localhost or to the corresponding website.
- Set a time limit (i.e. timeout)
- Read program output in Python
- In this example, the output of a neural network (Multi-layer Perceptron) implemented with the Scikit-learn library is read.
- Scikit-learn website:

Example of an integrated Java program with the neural network in Python

```
package integrationrequesthttp;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.logging.Level;
import java.util.logging.Logger;

/** Example of integration with Python neural network
 * @author Iván García-Magariño */
public class IntegrationRequestHTTP {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://localhost:8080/integration?x1=0&x2=1");
            HttpURLConnection con = (HttpURLConnection) url.openConnection();
            con.setRequestMethod("GET");
            con.setConnectTimeout(5000);
            con.setReadTimeout(5000);
            BufferedReader in = new BufferedReader(
                    new InputStreamReader(con.getInputStream()));
            String result= in.readLine();
            System.out.println ("The classification result is "+result);
        } catch (MalformedURLException ex) {
            Logger.getLogger(IntegrationRequestHTTP.class.getName())
                    .log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(IntegrationRequestHTTP.class.getName())
                    .log(Level.SEVERE, null, ex);
        }
    }
}
```

MOM (Message-oriented Middleware) (1)

- Secure communication
- Message queuing
- Directory support
- Distributed communications
- MOM does not model communications as remote method invocations (unlike CORBA for example).

MOM (2)

- Support message passing
- Message passing is non-blocking
- MOM allows posting and subscribing to channels to send and receive messages
- Basic MOMs allow only direct messages
- Advanced MOMs enable cross-platform switching

MOM(3)

MOM Advantages:

- Simple, just publish and subscribe
- Easy to install and use, unlike remote calling methods with CORBA.
- Generic and can be used between heterogeneous applications
- Flexible, it is not restrictive in what it sends, it simply sends it.

MOM (4)

Disadvantages of MOM

- Because it is so generic, applications are in charge of interpreting the messages.
- Developers are often unfamiliar with
- Asynchronous, it can be difficult to work with programs with blocking messages in a natural way, since MOM is non-blocking.
- Messages have no structure and can be more difficult to interpret.
- Non-standard: MOM only facilitates communication but messages must be interpreted by the application.

MOM (5)

When to use MOMs:

- When applications need messages
- When you do not want to mix programming with call definition.
- When CORBA or similar are very complex
- When remote calls are very rudimentary

MOM design considerations

- The Message Bus connects various applications
- The "**Channel**" class is the message bus interface.
- The "**ChannelListener**" classes are used to **listen for messages**
- The "**ChannelsUpdateListener**" class is used to receive notifications from the listening channel.
- MessageBus interface and MessageBusSocketImpl class allows **communication with the world outside** the app.

Use of MOM in client (1)

- Establish an instance of the message bus:

Channel. **setMessageBus** (new MessageBusSocketImpl (BROKER_NAME, BROKER_PORT));

- **Subscribe to a channel**

Channel textChannel = Channel.subscribe ("text_channel", this);

- **Post a message in the channel**:

textChannel. **publish** (new String (myID + " says Hello!"));

Use of MOM in client (2)

- Unsubscribe from a channel:

textChannel. **unsubscribe** (ChannelListener);

- Get a list of channel names:

Enumeration Channel. **getChannelNames** ();

MMO Interfaces

The **ChannelListener** class should be implemented by any object that wants to Be notified when a message is received from a calna

```
public interface ChannelListener {
  public void messageReceived (Channel channel, Object message);
}
```

The ChannelsUpdateListener interface has to be implemented by any object you want to be notified when a channel is added

```
public interface ChannelsUpdateListener {
  public void channelAdded (String name);
}
```

Example of MOM

```java
/**
 * MOM Client Interface.
 * <p>
 *     The MOM Client is the client side part of the message broker, it allows client code to subscribe to messages and
 *     send messages.
 * </p>
 */
public interface IMomClient {

    /**
     * Subscribe to a particular message class.
     * @param <T> the class of the message object that will be processed
     * @param messageClass the class of the message to subscribe to
     * @param messageProcessor the processor that will handle the message
     */
    <T> void subscribe(Class<T> messageClass, IMessageProcessor<T> messageProcessor);

    /**
     * Send a message.
     * <p>
     *     If the object to be sent is a modified object that has already been sent it must be cloned otherwise the
     *     receiver on the other end will not get the modifications.
     * </p>
     * @param destination the destination
     * @param message the message to send
     */
    void sendMessage(String destination, Serializable message);
}
```

Example implementation of the above interface

```
@Override
public final <T> void subscribe(@NotNull Class<T> messageClass, @NotNull IMessageProcessor<T> subscriber) {
    Set<IMessageProcessor<?>> processorsForThisType = messageProcessors.get(messageClass);
    if (processorsForThisType == null) {
        processorsForThisType = new HashSet<>();
        messageProcessors.put(messageClass, processorsForThisType);
    }
    processorsForThisType.add(subscriber);
}

@Override
public final void sendMessage(@NotNull String destination, @NotNull Serializable messageObject) {
    Message message = new Message(destination, messageObject);
    messages.add(message);
}
```

```java
/**
 * Process a Message.
 * <p>
 *     Calls processMessage on all processors that are registered for this message class.
 * </p>
 * @param <T> the class of the message object that will be processed
 * @param body the message body
 */
@SuppressWarnings("unchecked")
private <T> void processMessage(@NotNull T body) {
    Class<?> messageClass = body.getClass();
    Set<IMessageProcessor<?>> processorsForThisType = messageProcessors.get(messageClass);
    if (processorsForThisType != null) {
        for (IMessageProcessor<?> processor : processorsForThisType) {
            IMessageProcessor<T> typedProcessor = (IMessageProcessor<T>) processor;
            if (eventQueue == null) {
                typedProcessor.processMessage(body);
            }
            else {
                eventQueue.invoke(() -> typedProcessor.processMessage(body));
            }
        }
    }
}
```

```java
/**
 * In Loop.
 * <p>
 *     Gets messages from the input stream and processes them.
 * </p>
 */
private class InLoop implements Runnable {

    @Override
    public void run() {
        LOGGER.info("Input loop is running");
        try {
            while (running) {
                try {
                    Message message = (Message) inStream.readObject();
                    processMessage(message.getBody());
                }
                catch (ClassNotFoundException e) {
                    LOGGER.error("Could not read message", e);
                }
            }
        }
        catch (IOException e) {
            if (running) {
                LOGGER.error("In stream closed", e);
                stop();
            }
        }
        LOGGER.info("Input loop is stopped");
    }
}
```

```java
/**
 * Out Loop.
 * <p>
 *     Gets messages from the queue and sends them to the MOM Server.
 * </p>
 */
private class OutLoop implements Runnable {

    @Override
    public void run() {
        LOGGER.info("Output loop is running");
        try {
            while (running) {
                Message message = messages.poll(1000, TimeUnit.MILLISECONDS);
                if (message != null) {
                    outStream.reset();
                    outStream.writeObject(message);
                }
            }
        }
        catch (IOException e) {
            if (running) {
                LOGGER.error("Output stream closed", e);
                stop();
            }
        }
        catch (InterruptedException e) {
            if (running) {
                LOGGER.error("Out loop interrupted", e);
                stop();
            }
        }
        LOGGER.info("Output loop is stopped");
    }
}
```

Bibliography

- Ruh, W. A., Maginnis, F.X., Brown, W.J., "Enterprise Application Integration", Wiley e-book.
- Bottle: http://bottlepy.org/docs/dev/
- Scikit-learn: https://scikit-learn.org
- MOM in Java: https://docs.oracle.com/cd/E19148-01/819-4470/gbpdl/index.html