

Web Services and LIXI: A Primer

Dr. Vladimir Tasic, NICTA

Copyright NICTA 2007. All rights reserved. Do not distribute without permission.



Australian Government
**Department of Communications,
Information Technology and the Arts**
Australian Research Council

NICTA Members



Department of State and
Regional Development



The University of Sydney



Queensland University of Technology



NICTA Partners

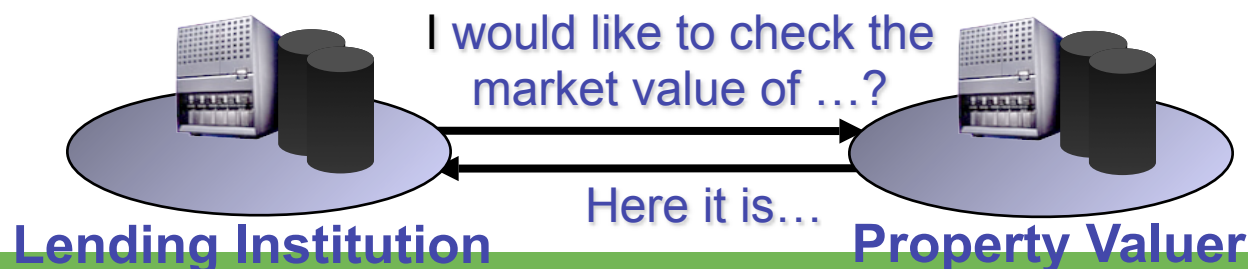
Goals of This Presentation

- Explain usefulness and importance of Web service technologies for LIXI members
- Clarify the most important concepts, standards, and acronyms
- Illustrate implementation of Web services on LIXI Valuations examples

- Motivation: the interoperability problem
- Desired solution: service-oriented architecture
- Web services (WS) approach to the problem
- Main WS standards: WSDL, SOAP, WSBPEL, ...
- Steps to take in WS implementation
- Tools for developing, hosting, and using WS
- Implementing LIXI Valuations with WS
- Summary, list of resources, and discussion

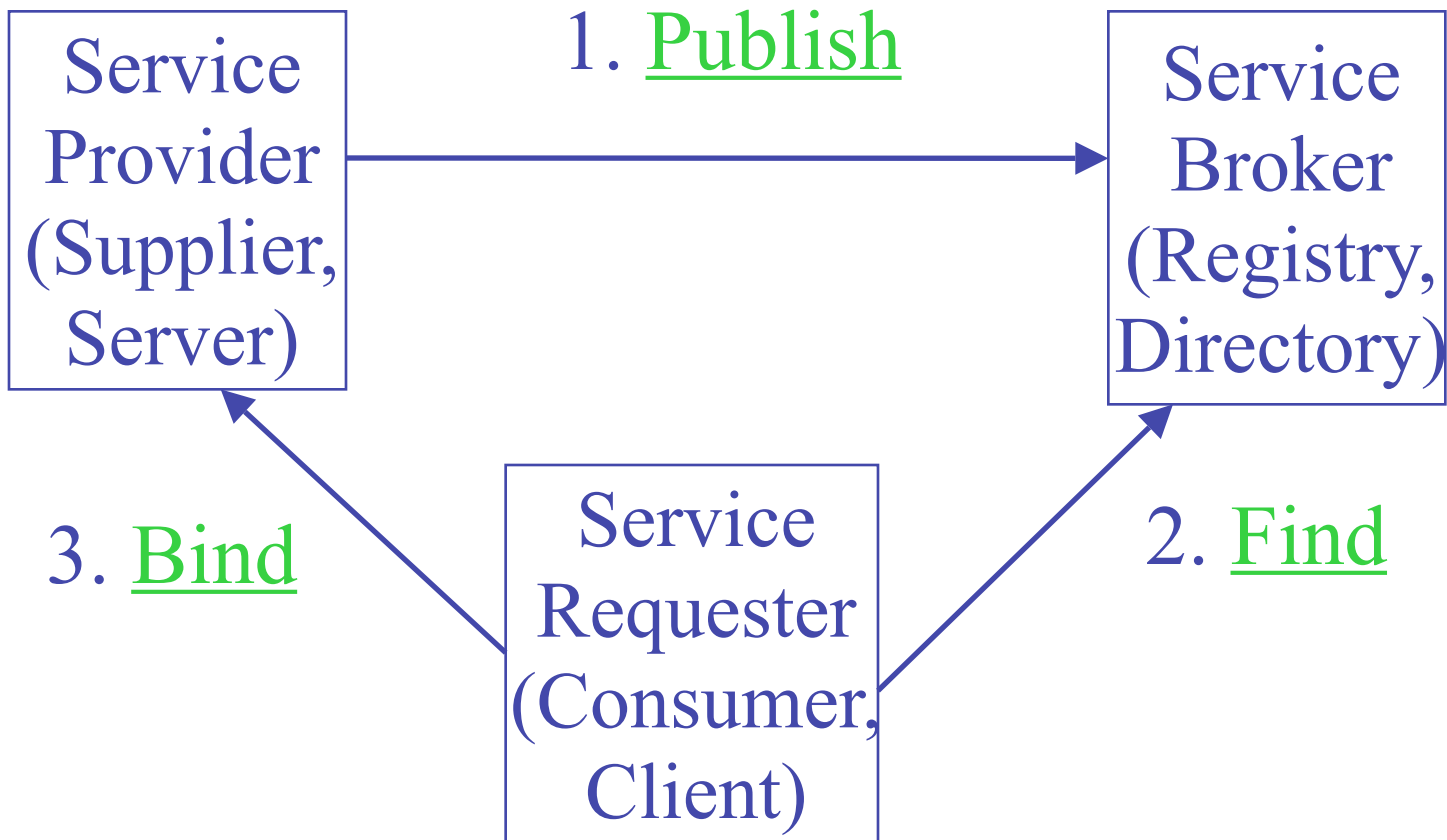
The Interoperability Problem

- Many participants in the lending process
 - Lenders, brokers, valuers, insurers, ...
- Using very diverse technological solutions
 - BPM systems, decision engines, B2B gateways, ...
- Running on heterogeneous platforms
 - Unix/Mainframe, Windows/PC, Linux/PC, ...
- They need to co-operate:



- LIXI's Credit Application Language (CAL)
 - Provides a standard vocabulary, but does not prescribe conversations => insufficient
- Major drawbacks of traditional B2B applications
 - Complex, custom, one-off solutions with proprietary end points
 - Not scalable, costly, and time consuming
- Operation integration standards needed to reduce costs, development & execution times

Software as an on-line service (not product)



- Integration that is dynamic (run-time) and automatic (with minimal human involvement)

- Implement the service-oriented architecture
- Platform, language, and vendor independence
- Leverage existing widely used standards (!)
 - Descriptions (of services, messages, ...): XML
 - Transport: Internet protocols (TCP/IP, HTTP, ...)
- Small set of relatively simple basic standards
 - Complexities in additional, optional technologies
- Supported by all major computing companies (!)
 - Microsoft, IBM, Sun, Oracle, SAP, HP, Verisign, ...

- A Web service is a distributed component of software (and/or hardware) functionality that:
 - can be uniquely identified on the Internet through a URI (Uniform Resource Identifier)
 - can be defined, described, and discovered using XML (Extensible Markup Language)
 - supports exchange of XML messages via Internet-based protocols
- This is a very broad definition, narrower also exist

Example of a Web Service

buyStock.wsdl
buyStockPortType
 buyStockOperation
 buyStockRequest
 symbol
 quantity
 buyStockResponse
 totalStockBuyingCost
 ...
buyStockBinding
 http
buyStockPort
 http://www.nicta.com.au
buyStockService

Web Service Compositions



- To leverage power of Web services
- Multiple providers and requesters of one WS
- Two types of a Web service composition:
 - Orchestration: execution controlled by an orchestrator (e.g., a workflow engine)
 - Choreography: independent WS (e.g., B2B)
- Web service composition – composite service

- Enterprise Application Integration (EAI)
 - Example: Integrating heterogeneous systems that implement lender's business processes
- Business-to-business (B2B) application-to-application (A2A) integration
 - Example: Integrating lenders and valuers
- Management of distributed computing systems
- Several upcoming computing technologies
 - Grid services, Semantic Web services, ...

- The interoperability problem
- Desired solution: service-oriented architecture
- Web services (WS) approach to the problem
- Main WS standards: WSDL, SOAP, WSBPPEL...
- Steps to take in WS implementation
- Tools for developing, hosting, and using WS
- Implementing LIXI Valuations with WS
- Summary, resources, and discussion

- Industrial standards and “standards”
- W3C (World Wide Web Consortium)
 - Basic: XML, WSDL, SOAP
(but also WS-Policy, ...)
- OASIS (Organization for the Advancement of Structured Information Standards)
 - Advanced: WS-BPEL, UDDI, ... (e.g., WSDM)
- WS-I (Web Services – Interoperability)
- Main difference: intellectual property rules
- No agreement about the set of basic standards

Stack of Web Service Technologies

X
M
L

Composition – WS-BPEL

(Web Services Business Process Execution Language)

Discovery – UDDI

(Universal Description, Discovery, and Integration)

Description – WSDL

(Web Services Description Language)

Messaging – SOAP

Communication

(HTTP – Hypertext Transport Protocol, ...)

- XML – for structured data representations
- SOAP – an XML packaging protocol, defines message formats and encoding rules
- WSDL – an XML service description language
- UDDI – for discovering service providers if you do not know their URIs
- WS-BPEL (BPEL, BPEL4WS) – for describing Web Service compositions (orchestrations)
- Many WS-* and WS?L technologies (e.g., WS-Policy, WS-Security, WS-Trust, WSCL)

Extensible Markup Language

- Platform-independent textual representation of structured data (used in CAL for data exchange)
- Looks like HTML, but you can define new tags
- Data and markup/metadata (data about data)
- Element – a piece of data with metadata
 - Between `<...>` and `</...>` tags, empty (`<.../>`)
- Attribute – a (key, value) property of an element
- A few additional concepts (e.g., namespaces)
- XML Schema for defining valid XML data formats
- Parsers check well-formedness and validity

A Simple XML Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<messages>
  <note id="c5336">
    <to>Vladimir</to>
    <from>Patrick</from>
    <heading>Reminder</heading>
    <body>Reminder: Review deadline today</body>
  </note>
  <note id="c5337">
    <to>Patrick</to>
    <from>Vladimir</from>
    <heading/>
    <!-- Comment: The same as: <heading></heading> -->
    <body>Thanks for the reminder</body>
  </note>
</messages>
```

Adapted from: <http://www.w3schools.com/xml/>

- Contractually describes Web service's:
 - Functionality (interface, “port type”): which operations the Web service supports
 - Access mechanisms (“binding”): which protocols the Web service uses and how data is packed
 - Location (URI): Web address of the service
- Can be used with various XML packaging and transport protocols (most often: SOAP over HTTP)
- Version WSDL 1.1 widely used, WSDL 2.0 not
- Does not describe response time, security, ...

A Simple WSDL File Example

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="AirLineWS" xmlns:impl="AirLineWS" xmlns:intf="AirLineWS"
  xmlns:apacheSOAP="http://xml.apache.org/xml-soap" xmlns:wsdlSOAP="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:SOAPENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:XSD="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
-   <wsdl:message name="getSchedulesRequest">
      <wsdl:part name="origin" type="xsd:string" />
      <wsdl:part name="destination" type="xsd:string" />
      <wsdl:part name="day_of_travel" type="xsd:string" />
      <wsdl:part name="month_of_travel" type="xsd:string" />
    </wsdl:message>
-   <wsdl:message name="getSchedulesResponse">
      <wsdl:part name="getSchedulesReturn" type="xsd:string" />
    </wsdl:message>
-   <wsdl:portType name="AirLineWS_IF">
      <wsdl:operation name="getSchedules" parameterOrder="origin destination day_of_travel month_of_travel">
        <wsdl:input name="getSchedulesRequest" message="impl:getSchedulesRequest" />
        <wsdl:output name="getSchedulesResponse" message="impl:getSchedulesResponse" />
      </wsdl:operation>
    </wsdl:portType>
-   <wsdl:binding name="AirLineWSSoapBinding" type="impl:AirLineWS_IF">
      <wsdlSOAP:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="getSchedules">
        <wsdlSOAP:operation soapAction="" />
        <wsdl:input name="getSchedulesRequest">
          <wsdlSOAP:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="AirLineWS" />
        </wsdl:input>
        <wsdl:output name="getSchedulesResponse">
          <wsdlSOAP:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="AirLineWS" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
-   <wsdl:service name="AirLineWS_IFService">
      <wsdl:port name="AirLineWS" binding="impl:AirLineWSSoapBinding">
        <wsdlSOAP:address location="http://www.airlinecompany.com/services/AirLineWS" />
      </wsdl:port>
    </wsdl:service>
  </wsdl:definitions>
```

Example Content of a WSDL File

buyStock.wsdl

```
buyStockPortType
  buyStockOperation
    buyStockRequest
      symbol
      quantity
    buyStockResponse
      totalStockBuyingCost
  ...
buyStockBinding
  http
buyStockPort
  http://www.nicta.com.au
buyStockService
```

- This is a much simplified illustration of a WSDL file
- The real file:
 - is in XML
 - has a different format (this figure only illustrates the content)
 - is very long (even for this simple content)

Structure of a WSDL File

Service interface definition:

<portType>

<operation name="getSchedules">

<operation name="...">

<operation>

<input message="...">

<output message="...">

<message>

<part>

<part>

Service location and access mechanisms definition:

<service>

<port name="AirLineWS">

<port name="...">

<port>

Binding information

Network address

<binding>

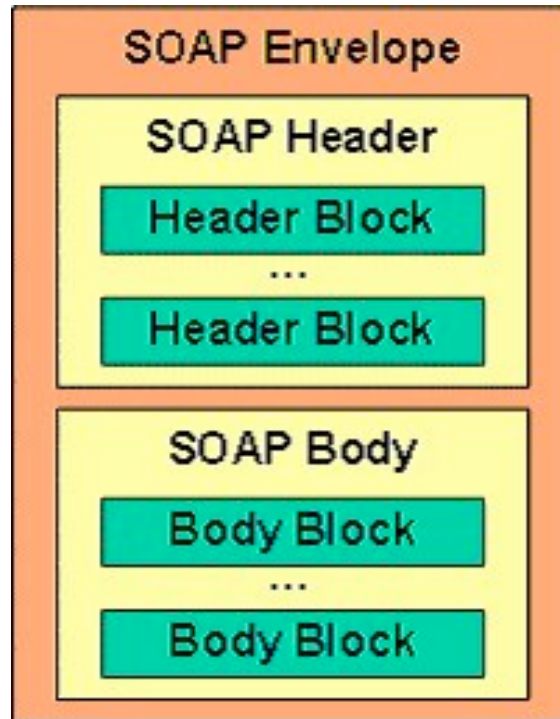
Protocol info

Message format

Operation

- XML-based messaging protocol
- Defines structure of XML messages and XML encoding rules for various data types
 - Envelope: optional headers and mandatory body
- Supports both synchronous (RPC-style) and asynchronous (document-style) interactions
- Can be used with various transport protocols (most often: HTTP, sometimes: SMTP, FTP)
- Version SOAP 1.2 widely used

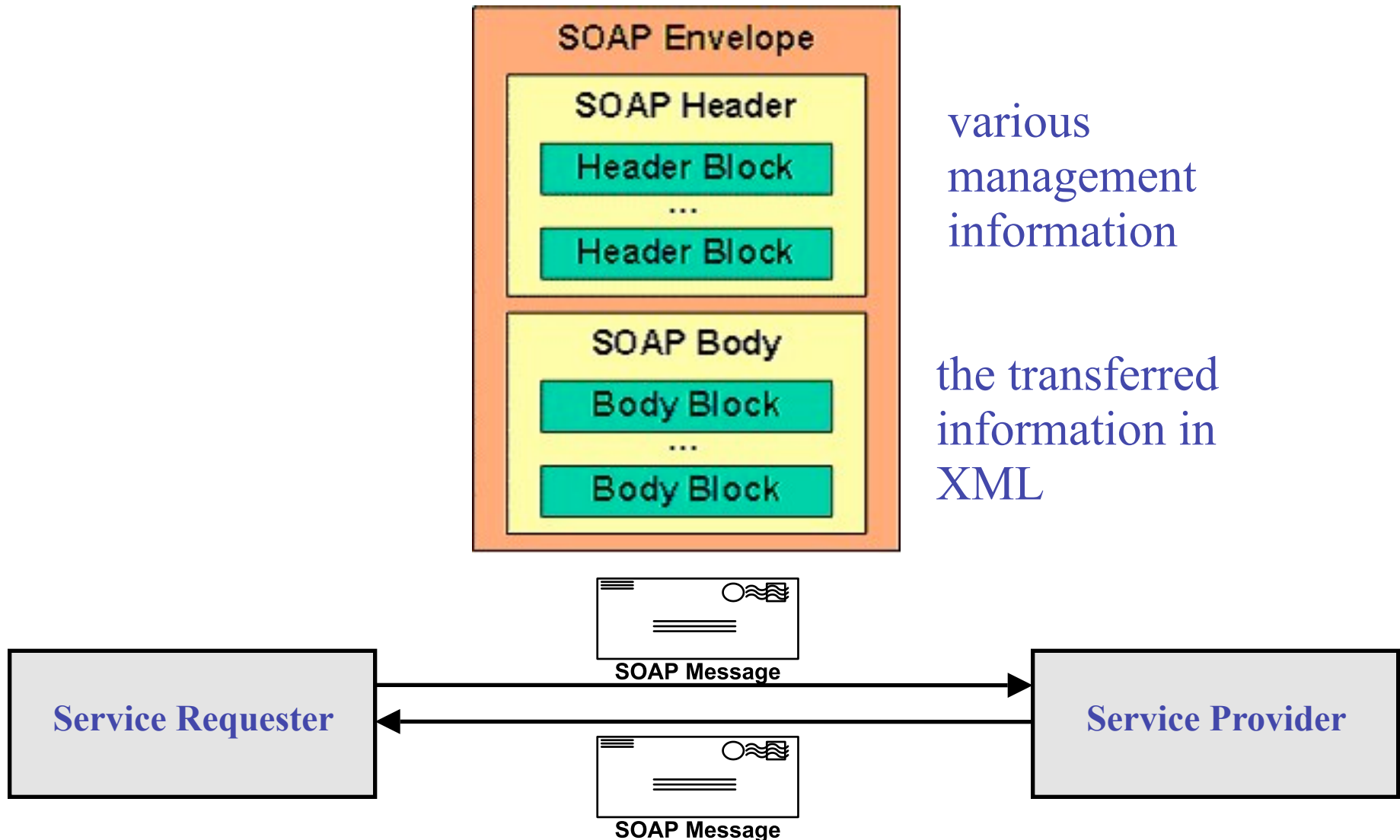
SOAP Message Structure



various
management
information

the transferred
information in
XML

SOAP Message Structure



SOAP Message Example

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
- <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  - <soap:Body>
    - <getSchedules xmlns="http://www.airlinecompany.com/services/AirLineWS">
      <origin>HKG</origin>
      <destination>BOS</destination>
      <day_of_travel>25</day_of_travel>
      <month_of_travel>march</month_of_travel>
    </getSchedules>
  </soap:Body>
</soap:Envelope>
```

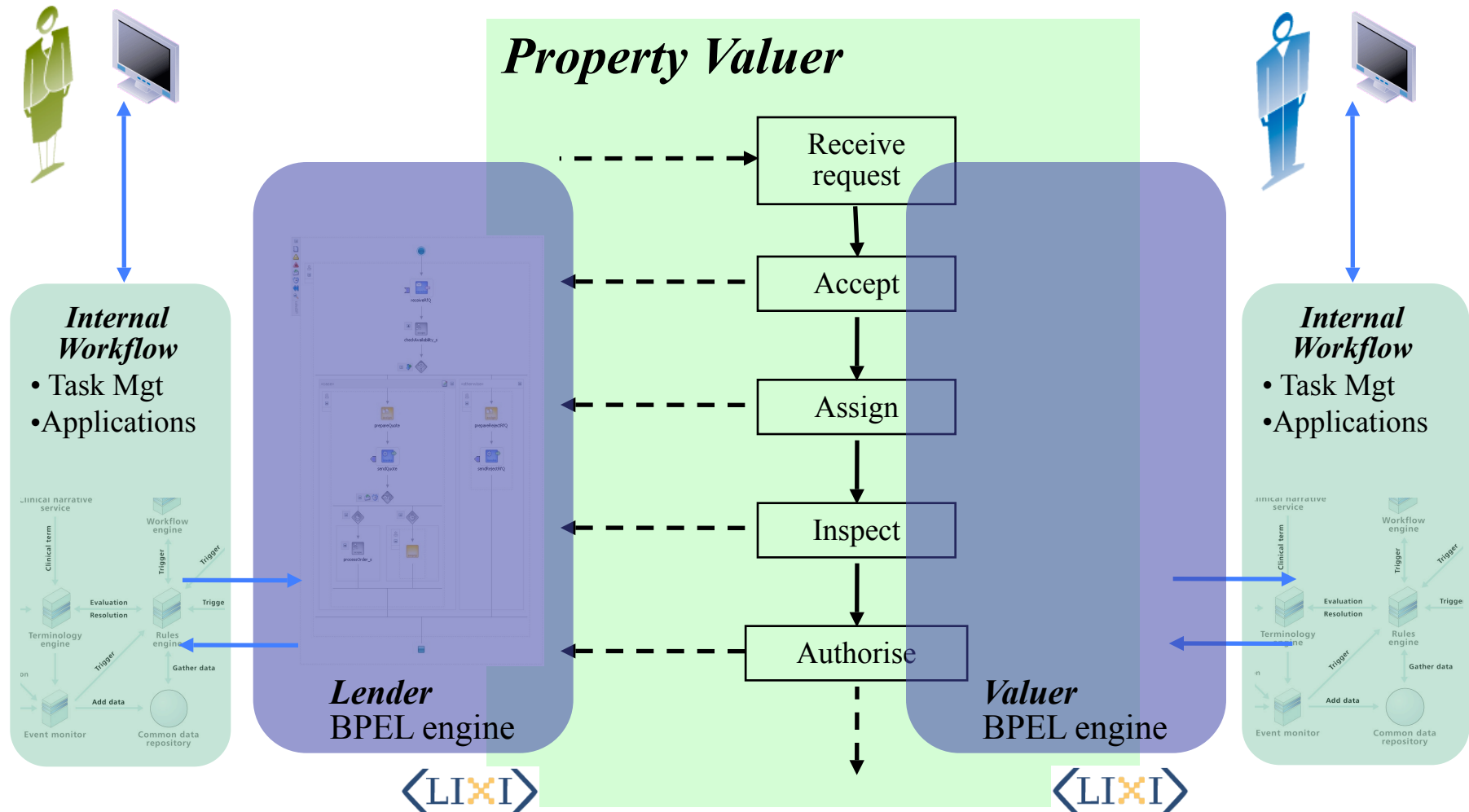
- No SOAP header!

- For an airline reservation Web service
- Most of the text is metadata, data is very short
- WSDL is NOT part of the message (this is usual)

- XML workflow language based on WSDL service descriptions and extension elements
 - Workflow: computer supported business process
 - For orchestrations (pros&cons for choreographies)
- Defines a model and a grammar for describing behaviour of a business process whose activities are implemented with Web services
 - Coordination of activities in the business process
 - Template for creating business process instances
- Version WS-BPEL 2.0 relatively widely used

WS-BPEL Example: LIXI

Valuations



- Partner links (business process participants)
- Variables (in terms of WSDL message types)
- Flow (listing concurrent activities)
- Links (for control structures between activities)
- Receive (blocking wait for messages)
- Invoke (one-way or request-response call)
- Assign (variable update)
- Reply (to message in Receive)
- ...

- For Web service directories (“yellow pages”) – E.g., to list all Web services by LIXI members – ‘UDDI’: both the standard and its implementations
- Defines XML formats for description of:
 - Business information (e.g., contact name)
 - Business and Web services (technical details)
- These formats are not identical to WSDL
- Defines operations to publish/find information
- Not widely used (e.g., there is none for LIXI)
- Latest version UDDI 3.0

Implementation – Provider Side

Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles

Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles



Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles



Describe the Web service in **Web Services Description Language (WSDL)** document

Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles



Describe the Web service in **Web Services Description Language (WSDL)** document



Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles



Describe the Web service in **Web Services Description Language (WSDL)** document



Implement the Web service on a specific development platform

Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles



Describe the Web service in **Web Services Description Language (WSDL)** document



Implement the Web service on a specific development platform



Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles



Describe the Web service in **Web Services Description Language (WSDL)** document



Implement the Web service on a specific development platform



Verify Input and Output **Simple Object Access Protocol (SOAP)** messages

Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles



Describe the Web service in **Web Services Description Language (WSDL)** document



Implement the Web service on a specific development platform



Verify Input and Output **Simple Object Access Protocol (SOAP)** messages



Implementation – Provider Side

Design business logic of the Web Service using service-oriented computing principles



Describe the Web service in **Web Services Description Language (WSDL)** document



Implement the Web service on a specific development platform

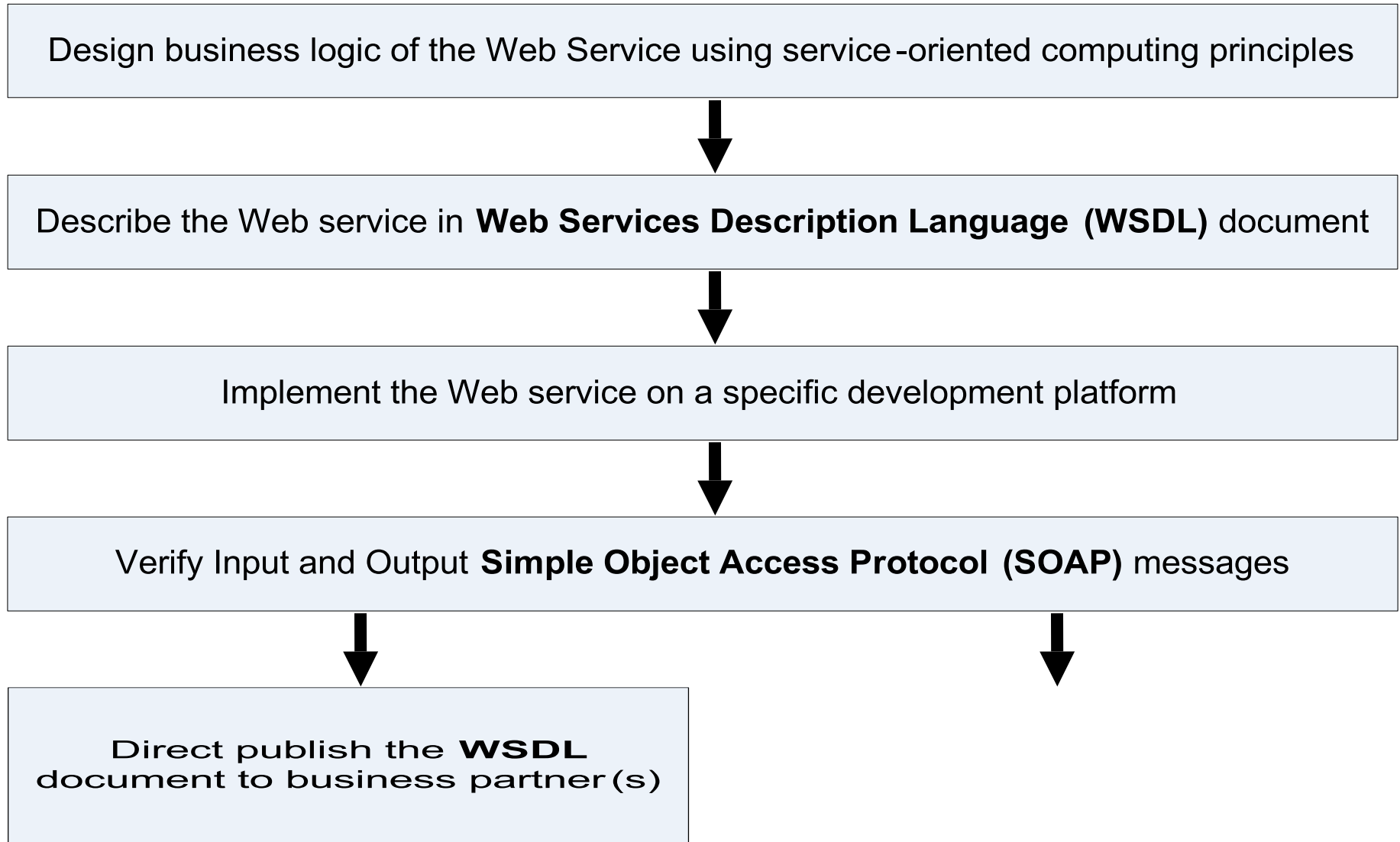


Verify Input and Output **Simple Object Access Protocol (SOAP)** messages

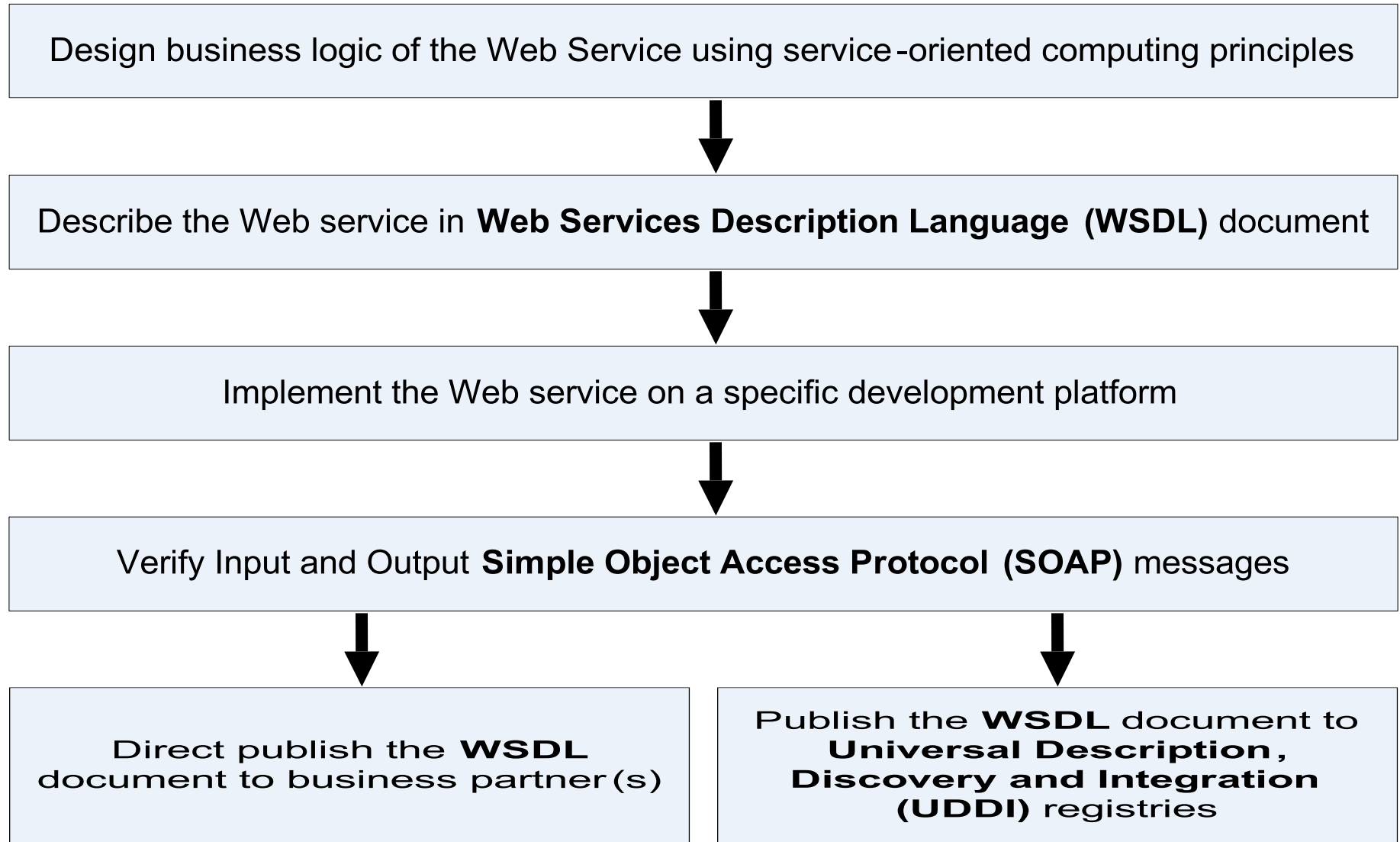


Direct publish the **WSDL** document to business partner(s)

Implementation – Provider Side



Implementation – Provider Side



Implementation – Requester Side

Implementation – Requester Side

Understand your business requirements of a desired Web service

Implementation – Requester Side

Understand your business requirements of a desired Web service



Implementation – Requester Side

Understand your business requirements of a desired Web service



Find the Web service via the
Web Services Description Language (WSDL) documents provided by your business partners

Implementation – Requester Side

Understand your business requirements of a desired Web service



Find the Web service via the
Web Services Description Language (WSDL) documents provided by your business partners



Implementation – Requester Side

Understand your business requirements of a desired Web service

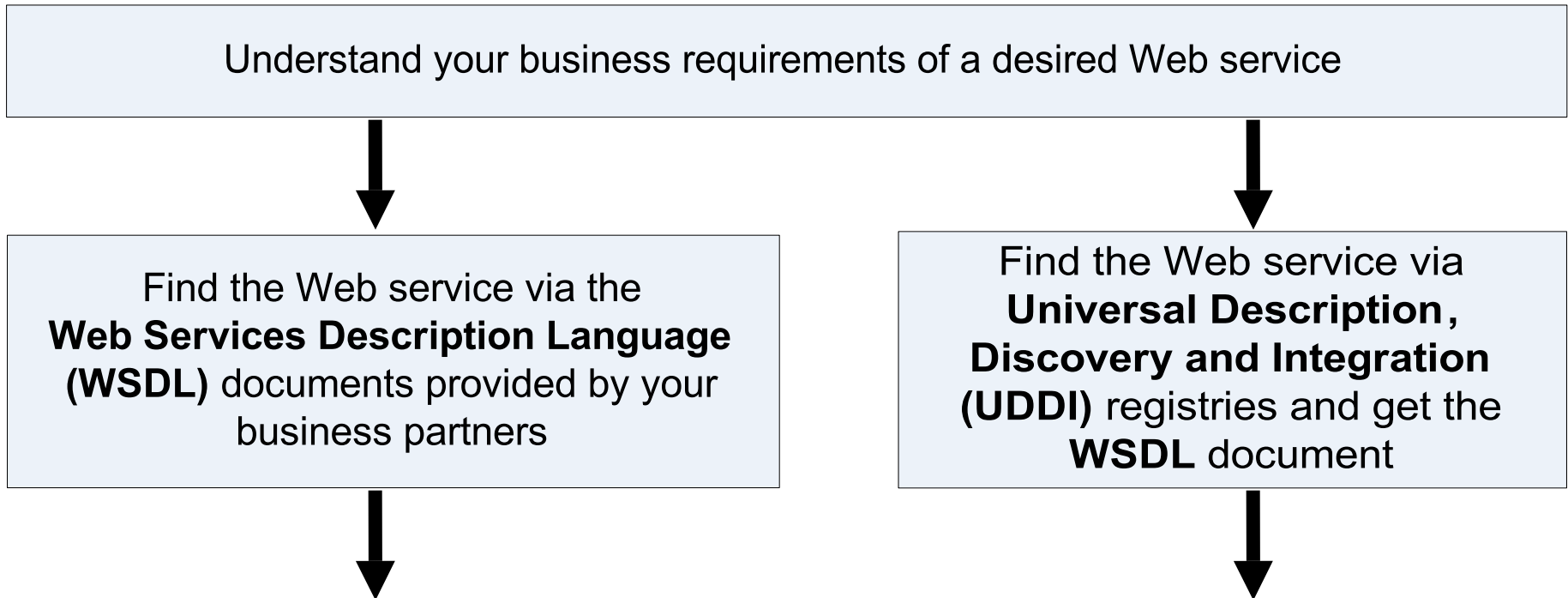


Find the Web service via the **Web Services Description Language (WSDL)** documents provided by your business partners

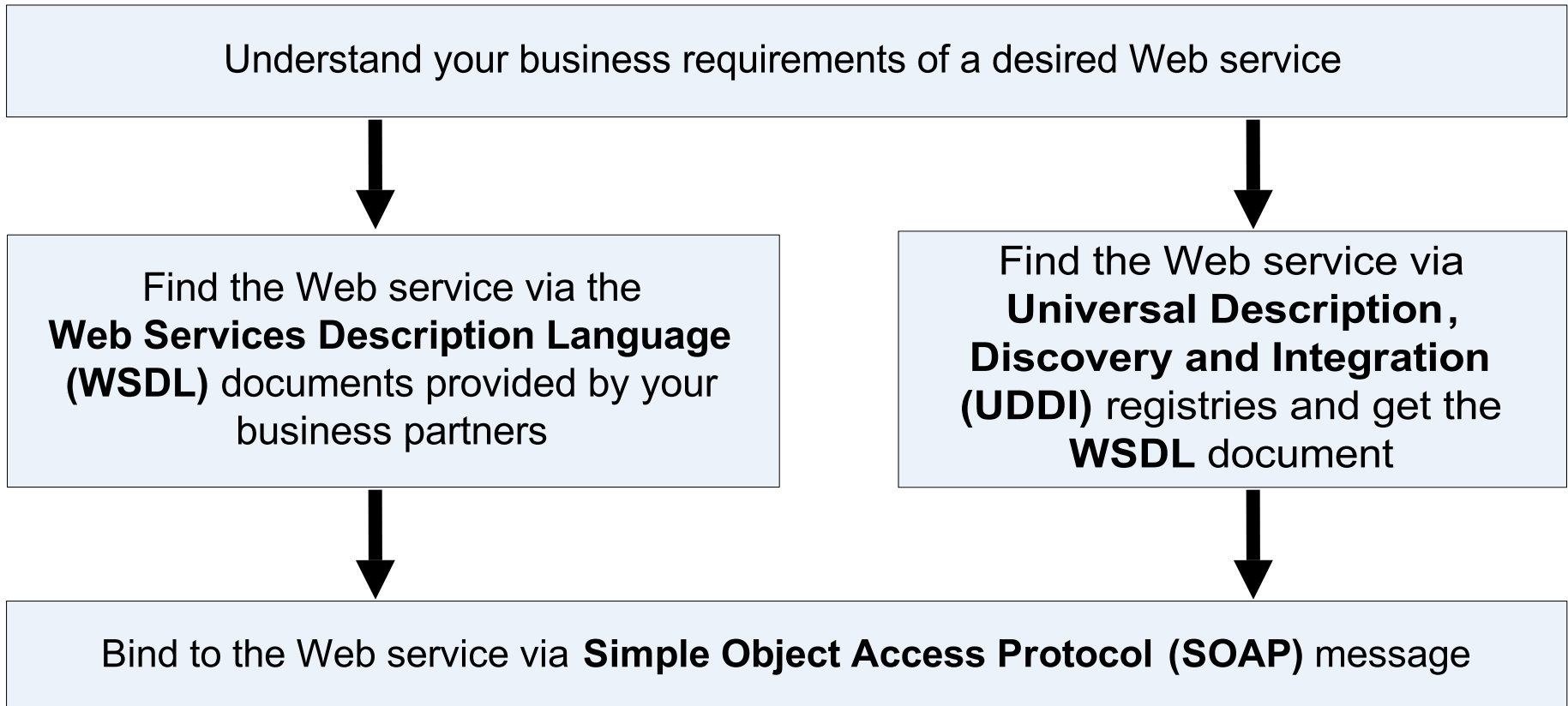


Find the Web service via **Universal Description, Discovery and Integration (UDDI)** registries and get the **WSDL** document

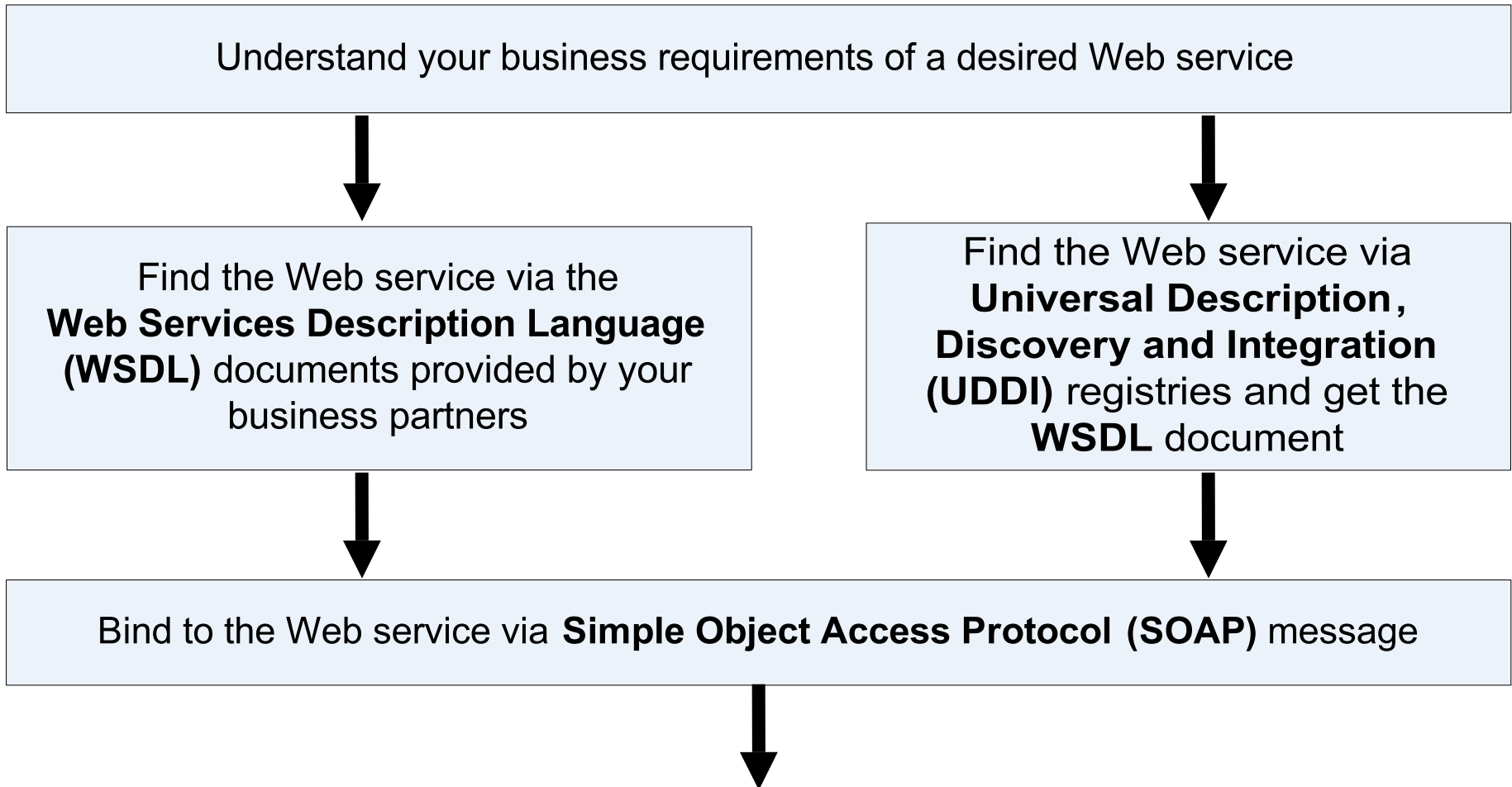
Implementation – Requester Side



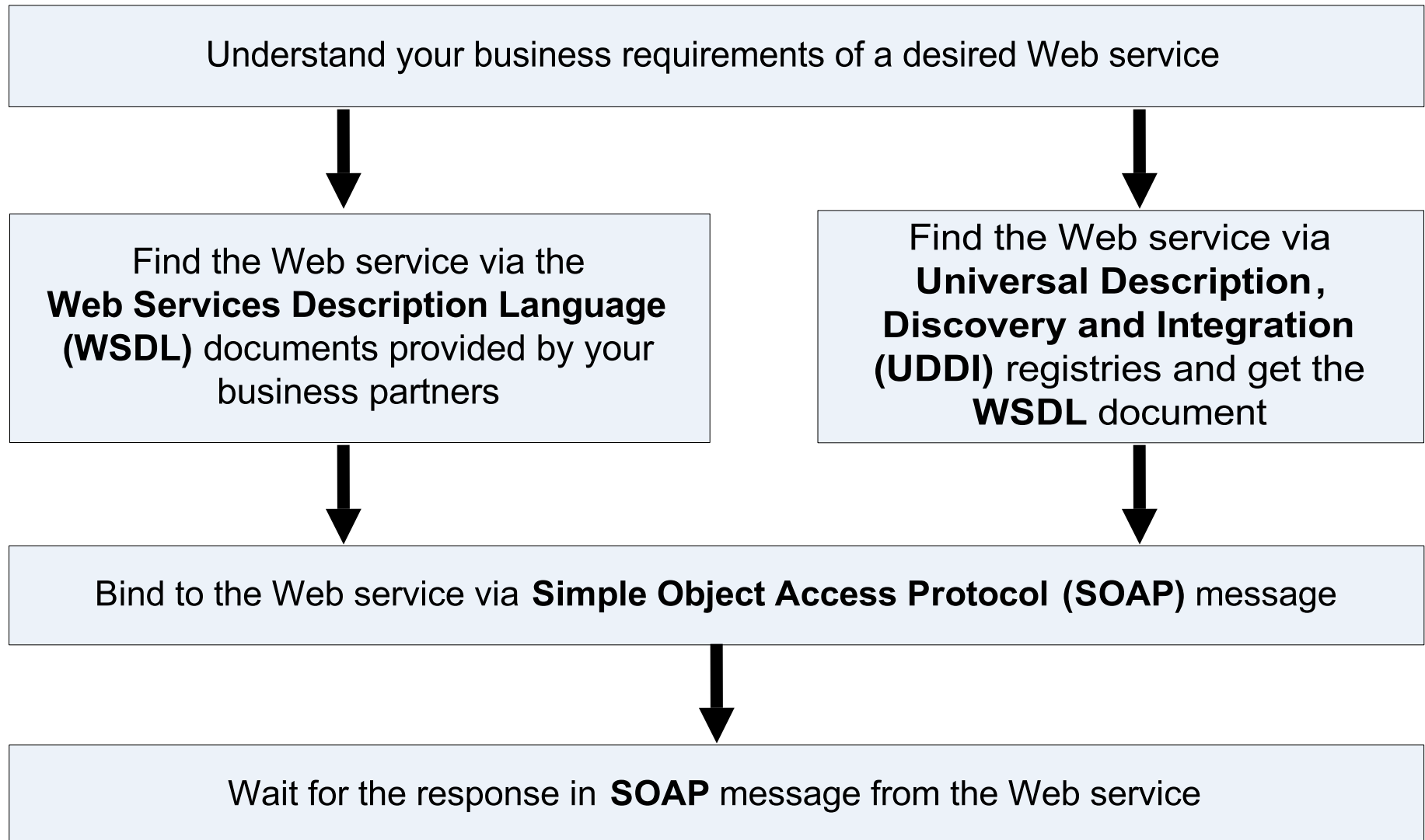
Implementation – Requester Side



Implementation – Requester Side



Implementation – Requester Side



- Most modern software engineering tools have support for writing XML and XML Schema with graphics or tables
- Checking well-formedness (obeying of XML rules) and validity (conformance to an XML schema)
- Specialized software engineering tools for XML
 - Example: Altova XmlSpy
- General commercial software engineering tools
 - Example: Microsoft Visual Studio
- General open-source software engineering tools
 - Example: plug-ins for Eclipse

- Several tools for writing and checking WSDL
- In most cases there is no need to write WSDL manually – some tools automatically generate WSDL from code (in Java, C#, ...)
- Tools generating code skeletons from WSDL
- Previously mentioned example XML tools all support writing and checking of WSDL and some form of WSDL <-> code translation
- Many tools for hosting Web services (SOAP engines) also support WSDL <-> code translation
 - E.g., Apache Axis2 wsdl2java and java2wsdl

Tools for Writing SOAP, BPEL,

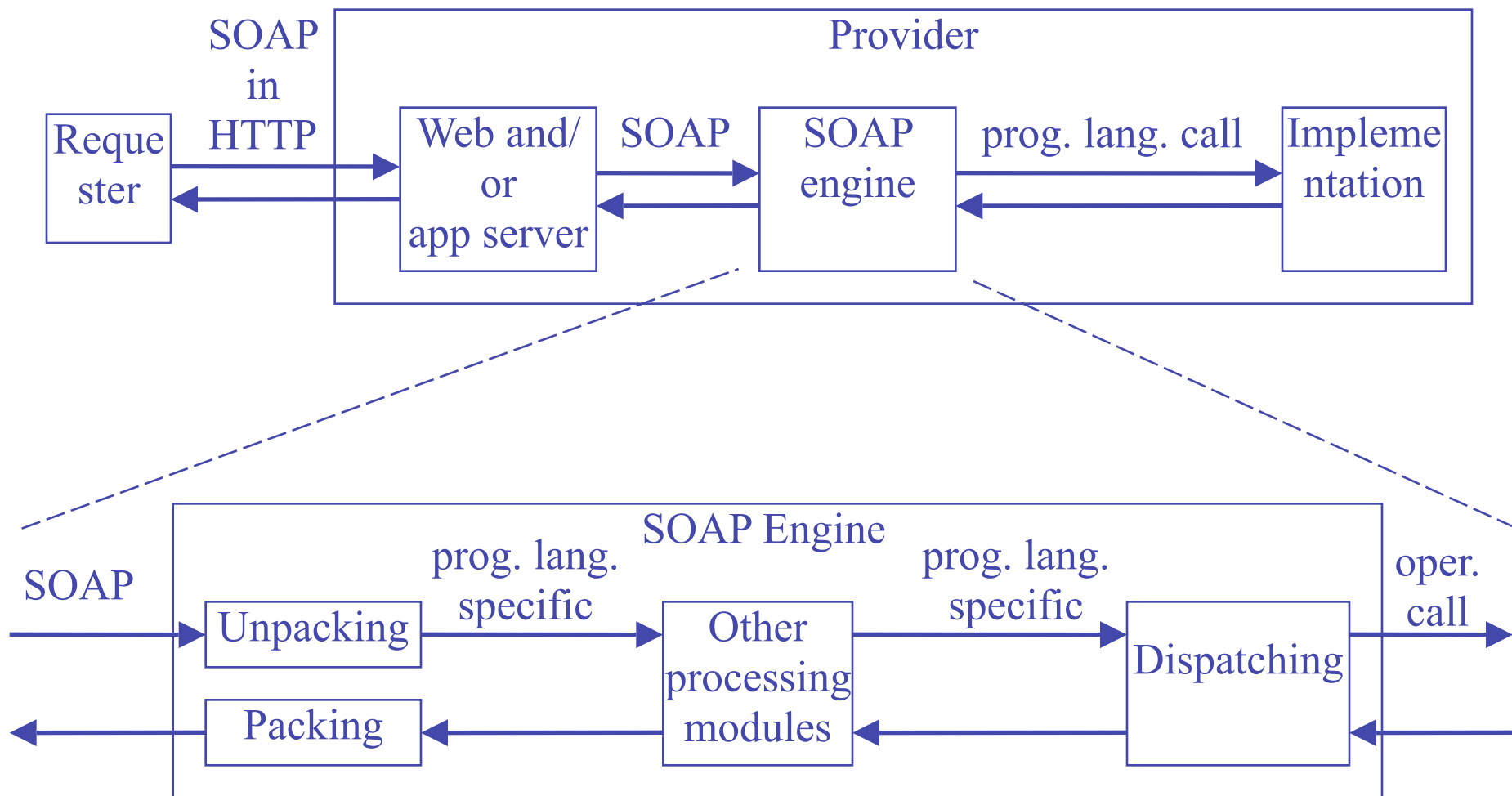
- SOAP is an XML messaging protocol
 - There is no need to write SOAP messages manually
 - They will be automatically generated by Web services
- A few tools enable writing UDDI descriptions
 - Often, their main purpose is hosting UDDI repositories (e.g., UDDI4J)
 - Public UDDI repositories also enable publication
- Several tools enable writing WS-BPEL workflows
 - Most of them are parts of WS-BPEL engines
 - Examples: MS BizTalk, ActiveBPEL (open source)

Tools for Hosting and Using WS

Slide of 48

- SOAP engine is a tool for receiving, processing, and sending SOAP messages
 - The major type of tool for hosting provider WS
 - A lightweight version often used for requestors
- Commercial Microsoft .NET-based SOAP engines
 - E.g., Windows Communication Foundation (WCF)
- Commercial Java-based SOAP engines
 - Examples: IBM WebSphere, BEA WebLogic
- Open-source Java-based SOAP engines
 - Example: Apache Axis2 (for Java or C++)

Typical SOAP Engine Architecture



Next: WS for LIXI Valuations

- The interoperability problem
- Desired solution: service-oriented architecture
- Web services (WS) approach to the problem
- Main WS standards: WSDL, SOAP, WSBPEL, ...
- Steps to take in WS implementation
- Tools for developing, hosting, and using WS
- Implementing LIXI Valuations with WS
- Summary, resources, and discussion

- Centre of excellence established 2002
- 300 research staff, 250 postgrad students,
15 research programs, 6 laboratories, 4 cities

-  Australian Government
Department of Communications,
Information Technology and the Arts
Australian Research Council
 BusinessACT
ACT GOVERNMENT
 NSW
First for Business
Department of State and
Regional Development
 UNSW
 ANU
THE AUSTRALIAN NATIONAL UNIVERSITY

- NICTA partners
 The University of Sydney
 Victoria
The Place To Be
 THE UNIVERSITY OF
MELBOURNE
 Queensland Government
 THE UNIVERSITY
OF QUEENSLAND
AUSTRALIA
 QUT
Queensland University of Technology
 Griffith
UNIVERSITY

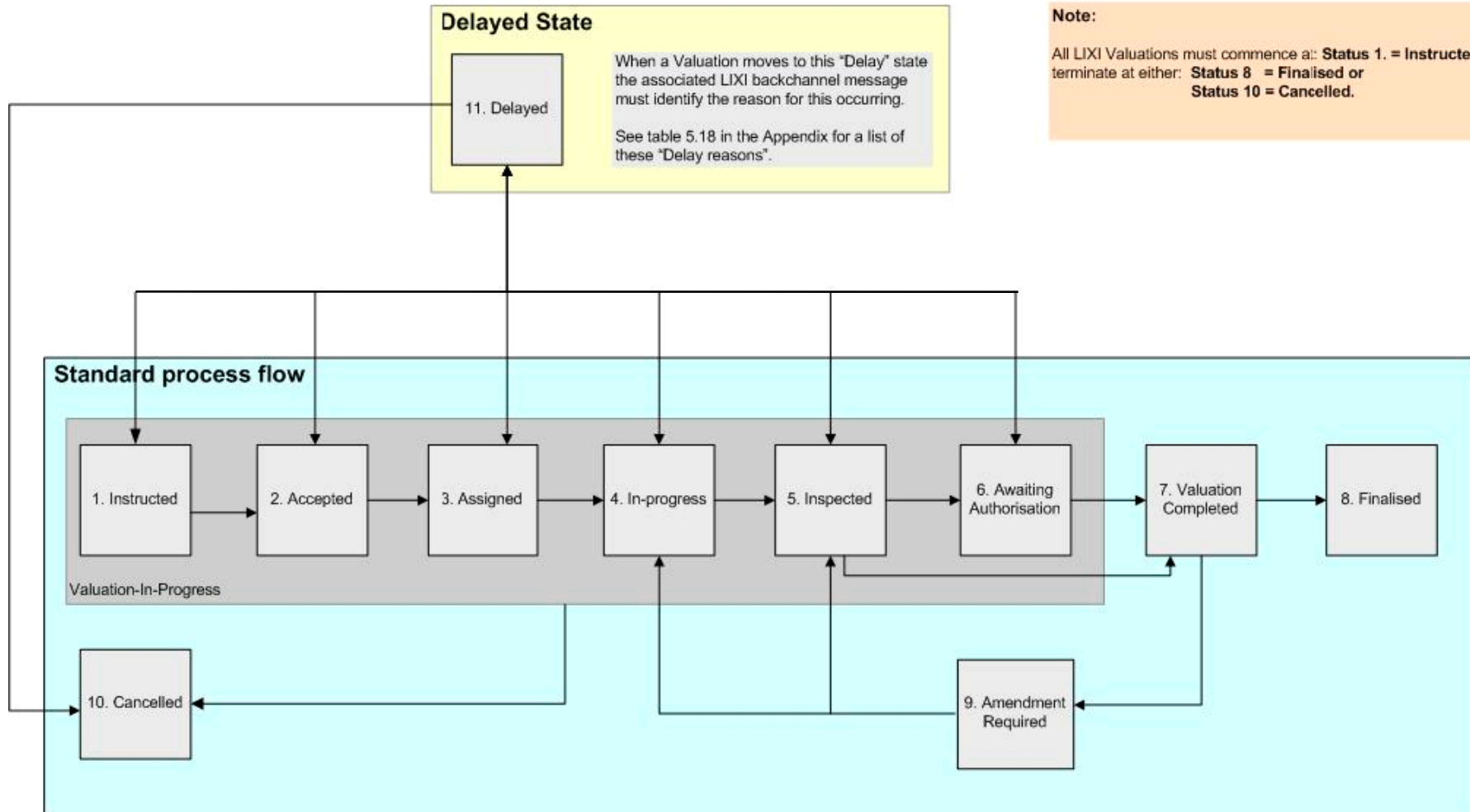
- ~40 people in total:
 - 14 staff (in Sydney and Canberra)
 - 12-15 postgraduate students
 - 10 undergrad thesis students
- Three inter-related research disciplines:
 - Requirements Engineering
 - Software Architectures
 - Processes
- Use-driven fundamental research
 - All projects involve industry collaboration

Our LIXI Projects and Goals

- Our research goals in the collaboration with LIXI:
 - Mapping of business standards to technology
 - Solutions for standardization of conversations of lending (to guide implementation of LIXI standards)
 - Case study application (e.g., for LIXI Valuations)
- Research projects and activities:
 - Business process modelling
 - LIXI Valuations Reference Implementation
 - New project: “Visible Loans”
 - Research and technology seminars

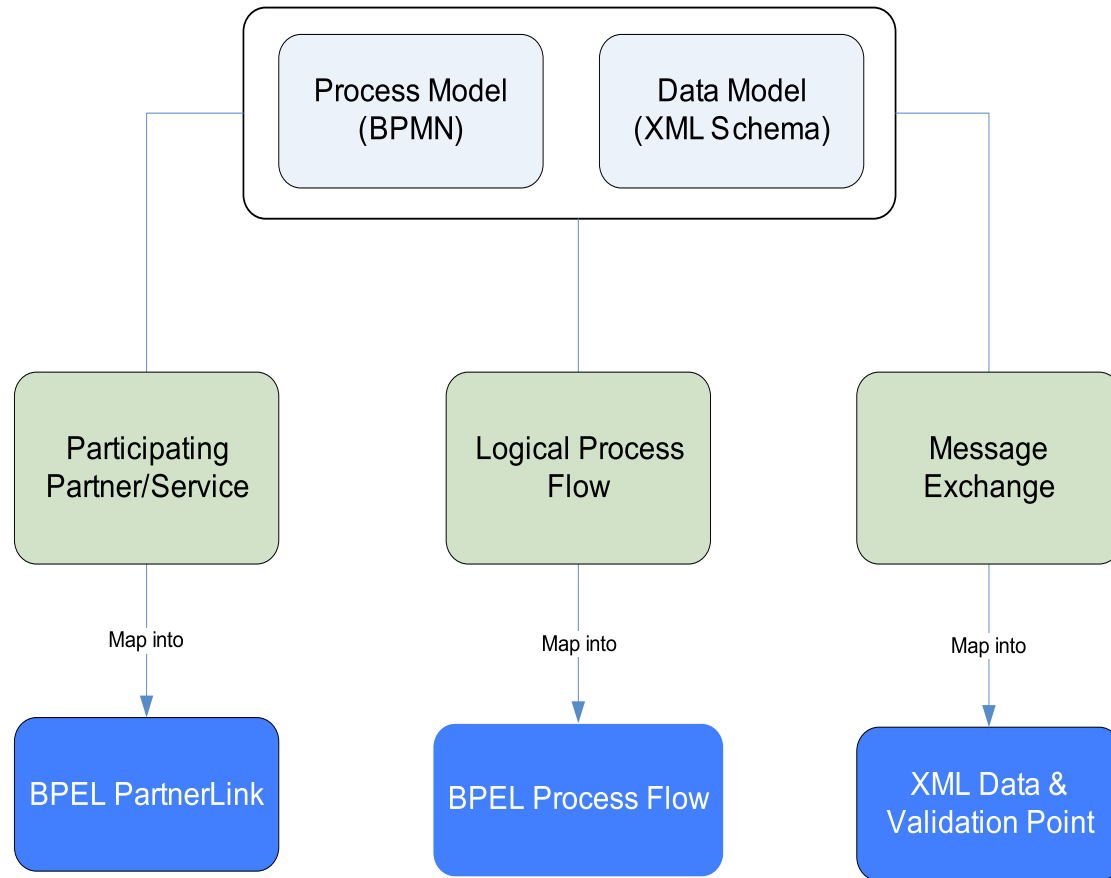


LIXI Valuations: Business Process



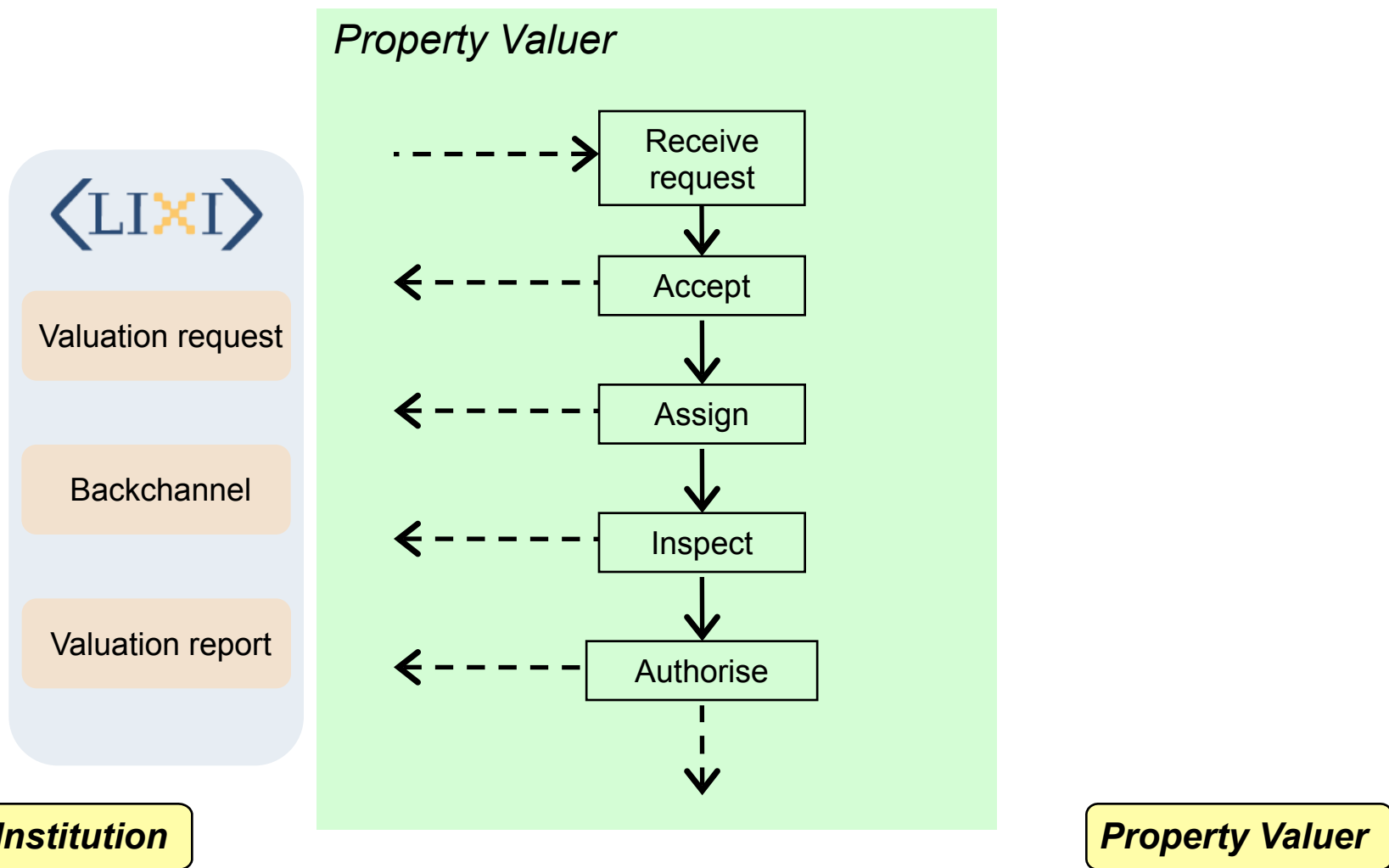
- See the LIXI Process Handbook v2.0

From Model to Web Services



- The LIXI Process Handbook v2.0 also contains a comparative analysis of BPMN tools

LIXI Valuations: Architecture



LIXI Valuations: Architecture

<LIXI>

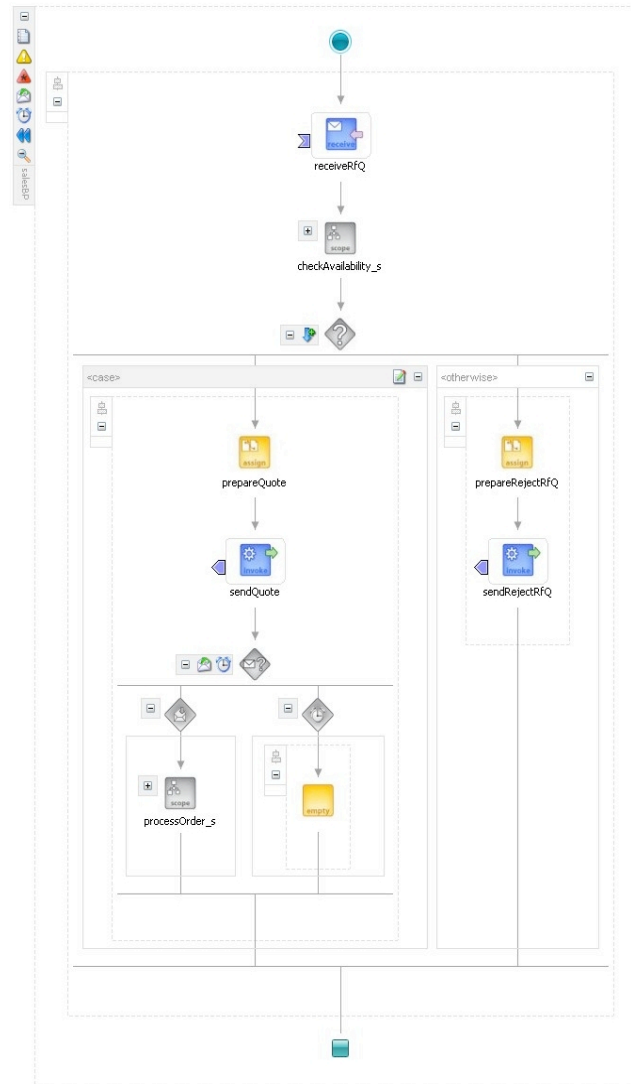
Valuation request

Backchannel

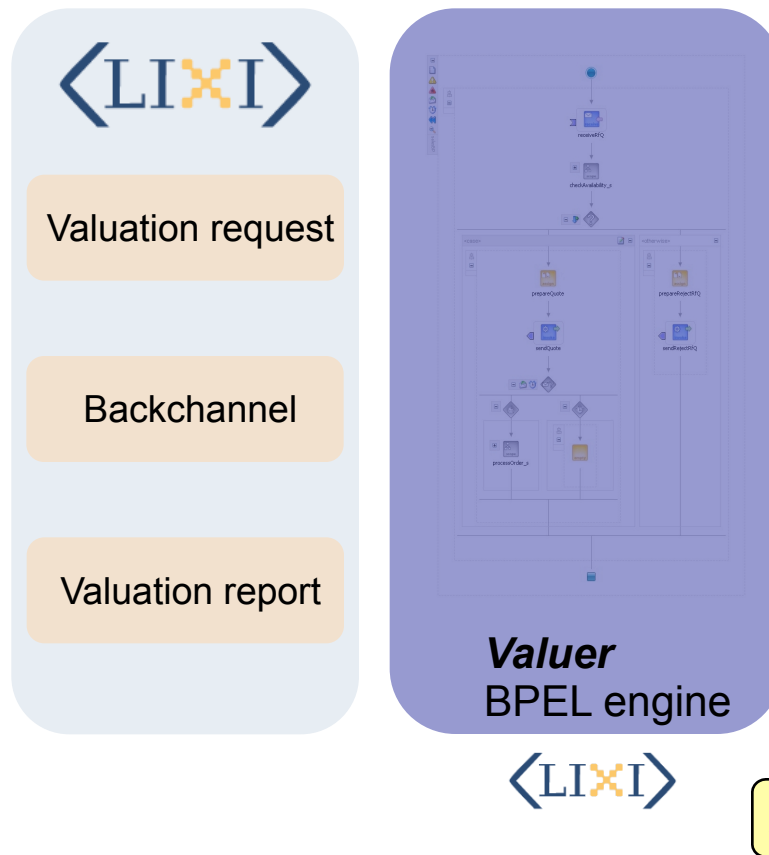
Valuation report

Lending Institution

Property Valuer



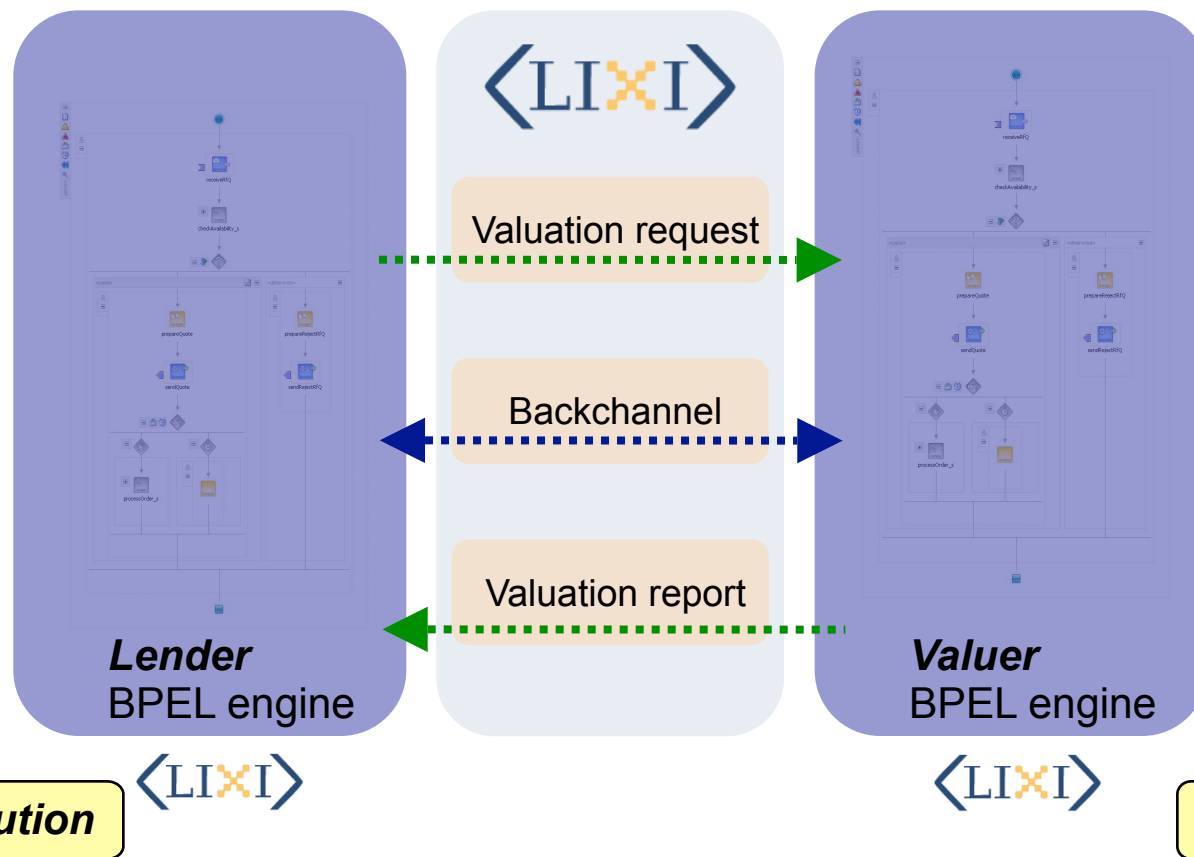
LIXI Valuations: Architecture



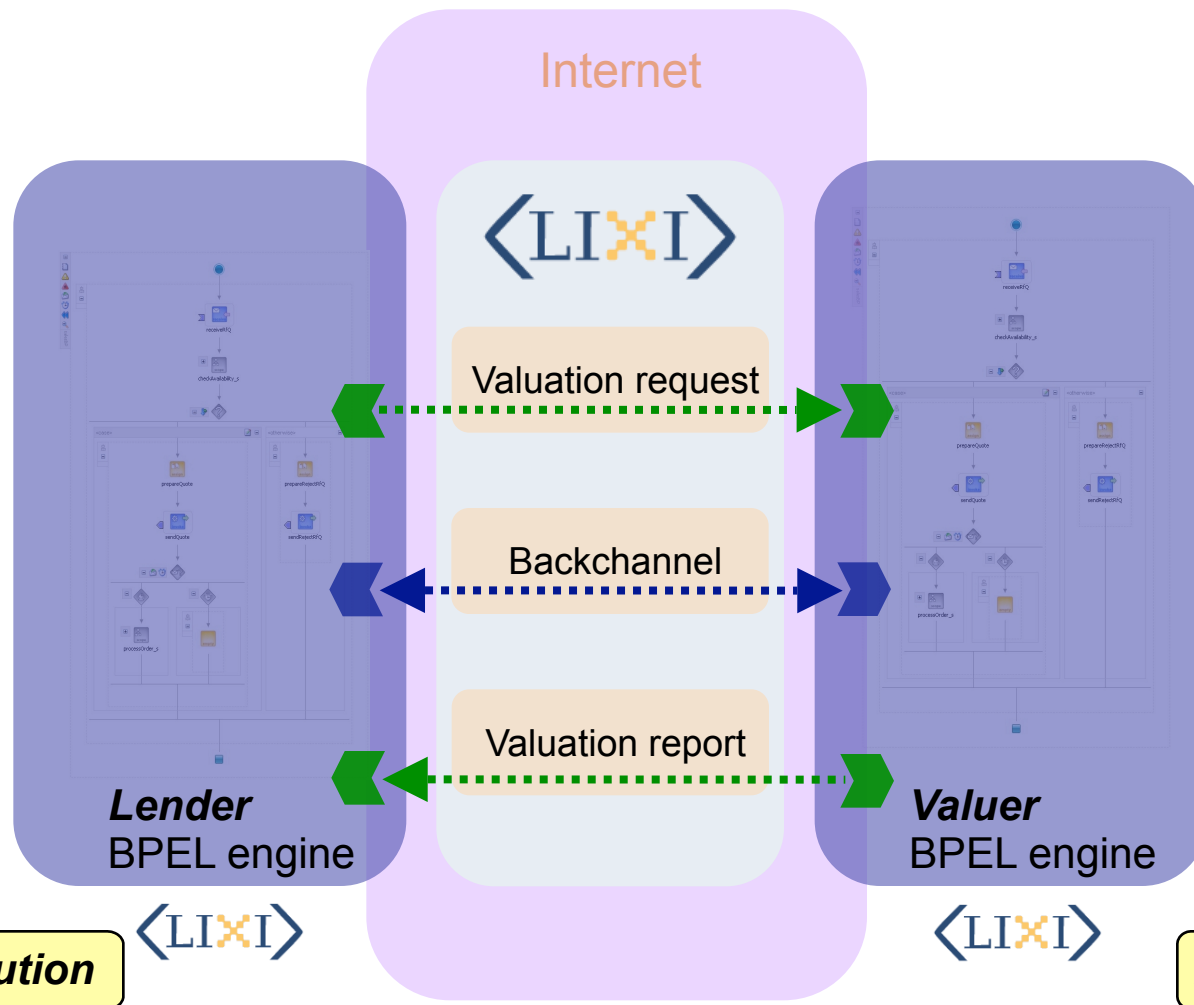
Lending Institution

Property Valuer

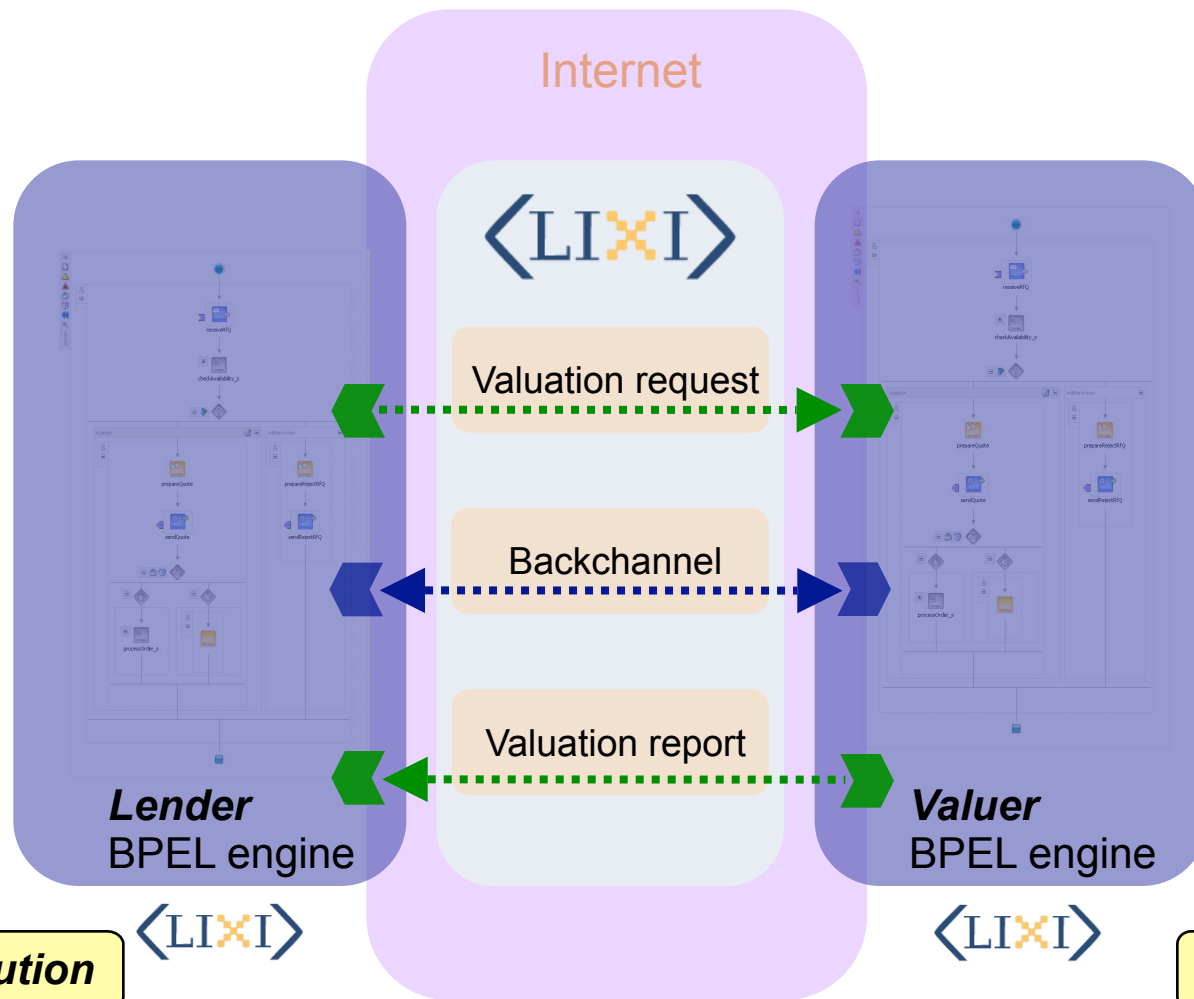
LIXI Valuations: Architecture



LIXI Valuations: Architecture



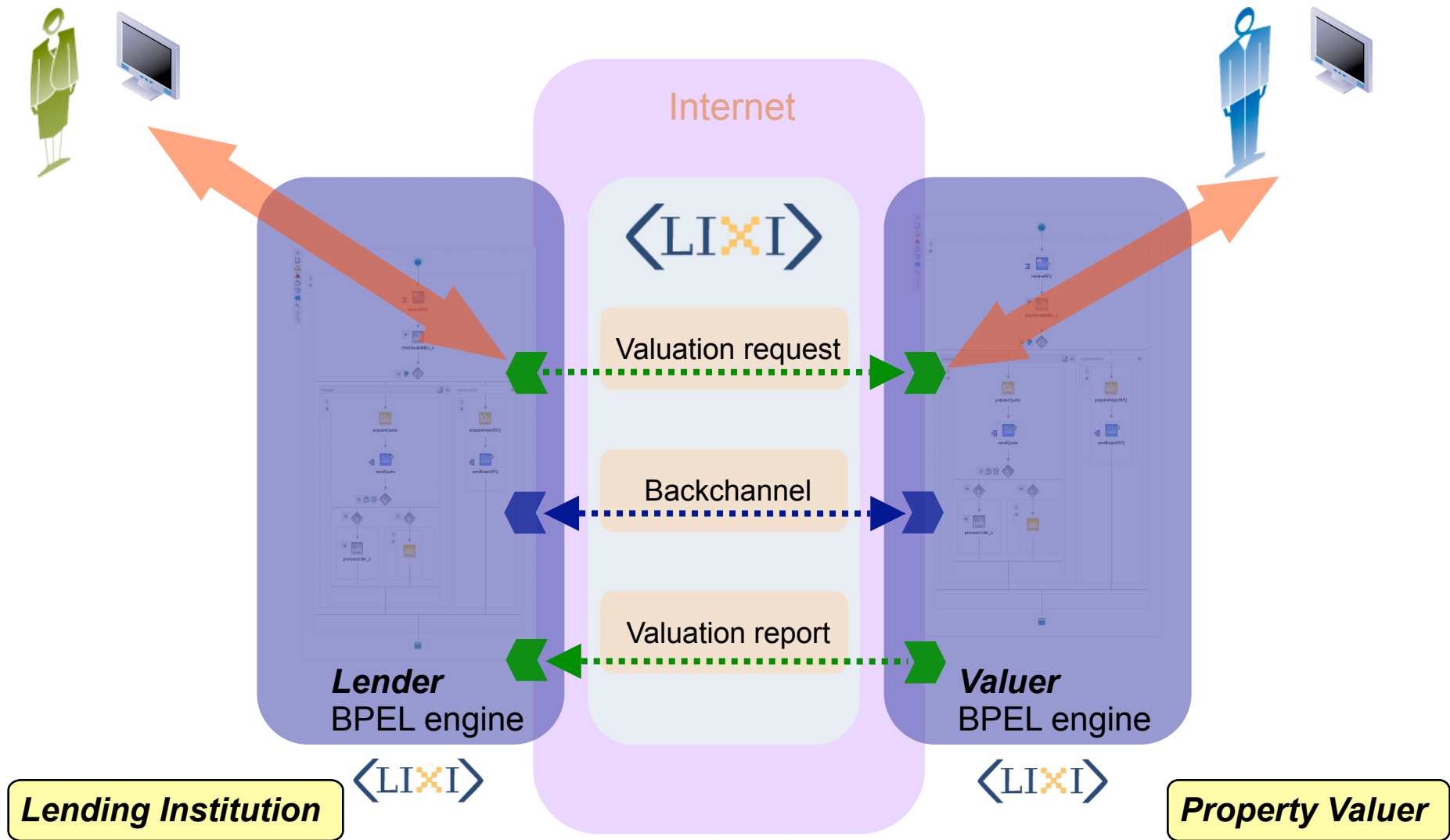
LIXI Valuations: Architecture



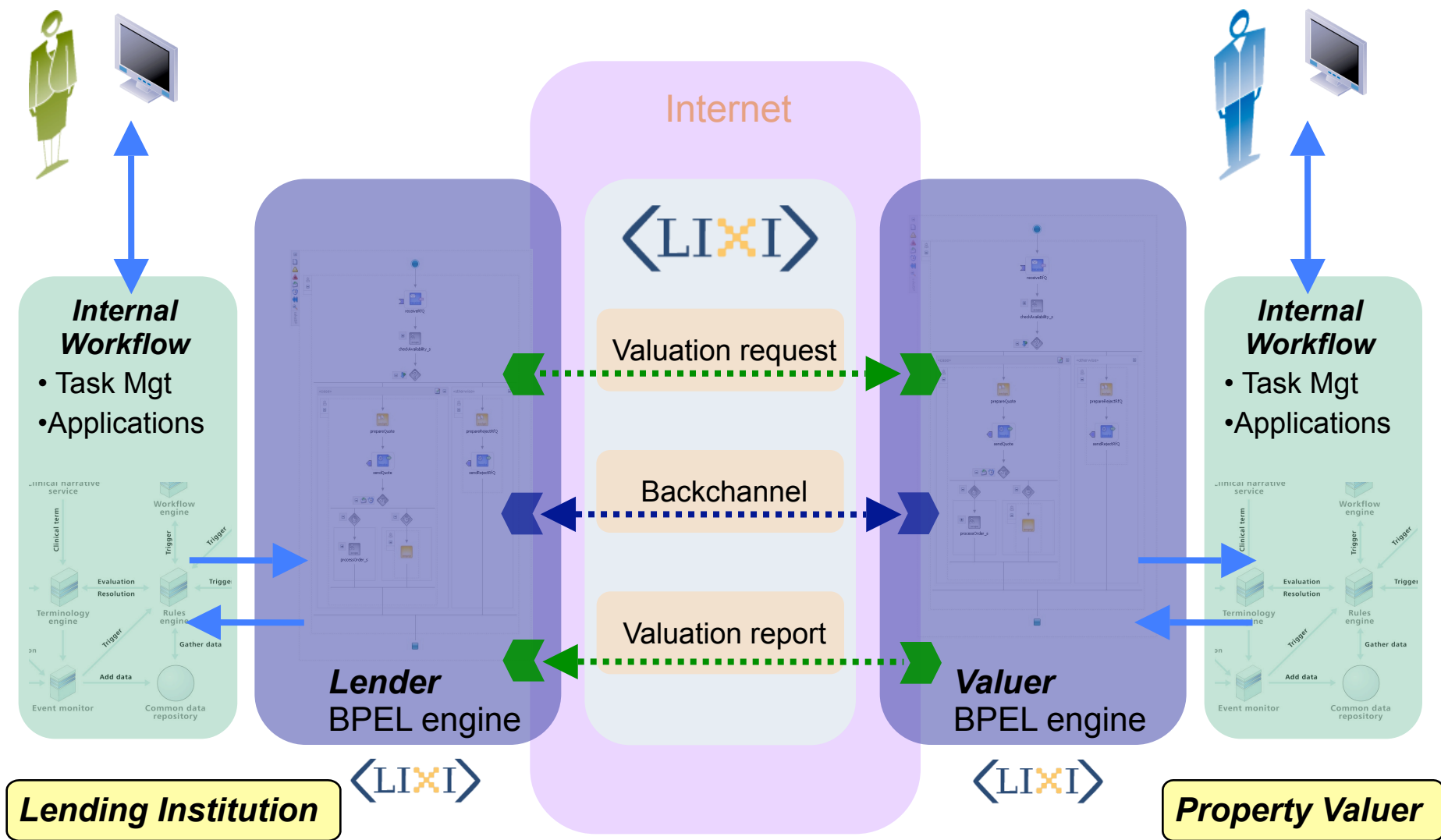
Lending Institution

Property Valuer

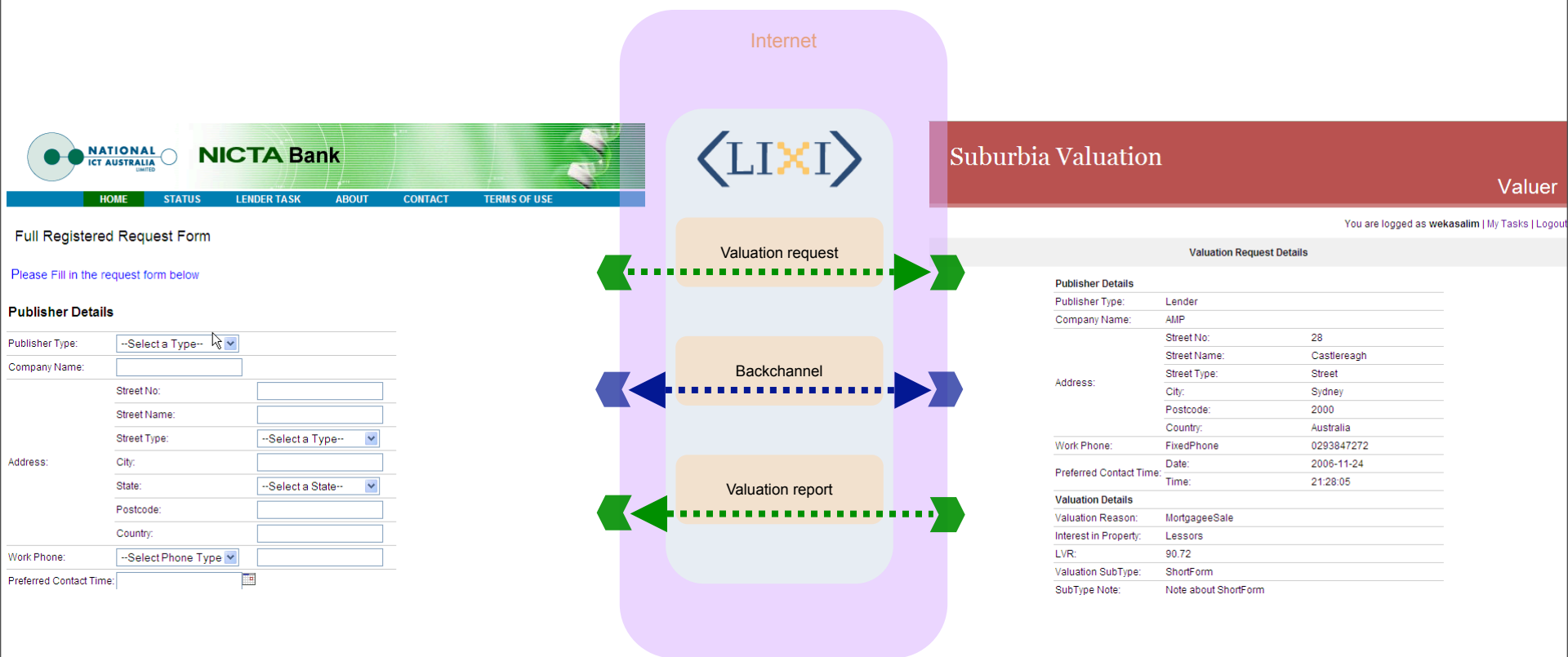
LIXI Valuations: Architecture



LIXI Valuations: Architecture



LIXI Valuations: Web Pages



- The result delivered through application of WS and our methodology has the following qualities:
 - a) Interoperability: Large numbers of different parties can communicate readily
 - b) Flexibility: Participants can comply with another's rules without being locked into these for all interactions
 - c) Maintainability/evolvability: Delivered systems can accommodate new participants and rules over time, without compromising existing interactions

- LIXI Val's Reference Implementation Guide: NOW
- WS-BPEL code for LIXI Valuations Process: NOW
- Web services interfaces for LIXI Valuations: NOW
- Reference implementation case study: ongoing!
- NICTA “extensions”: negotiable
 - Implementation services (expertise)
 - Task management
 - Software interfaces for task management, workflow, applications

- Web service technologies address interoperability
- Service-oriented architecture provides integration that is dynamic and automatic
- Web services have URIs and use XML for description, discovery, and data transfer
- The main technologies: WSDL, SOAP, WS-BPEL, UDDI, but there are many others (WS-*)
- Supported by tools from major computing companies (so they are mostly “under the hood”)
- We use them to address conversations of lending

- On-line communities and portals (e.g. searchwebservices.com, ...)
- Books (by O'Reilly, SAMS, ...)
- Magazines (e.g., Web Services Journal, ...)
- Web sites of companies, open-source projects, and standardization bodies (W3C, OASIS, WS-I)
- Industrial and academic conferences
- NICTA researchers!

- Questions?
- Insights?
- Contact information:
 - Presenter: Vladimir.Tosic at server: nicta.com.au (please, write “LIXI” at the beginning of Subject)
 - NICTA’s LIXI Business Processes team:
Paul.Mackie@nicta.com.au , (02) 8374-5462