Rey Juan Carlos University

Kybele Research Group

Representing Web Services with UML: A Case Study

First International Conference on Service Oriented Computing

Trento, Italy

E. Marcos, V. de Castro and **B. Vela**

{emarcos, vcastro, bvela}@escet.urjc.es

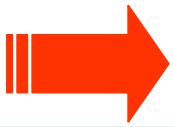
Index

- Introduction
- MIDAS Framework
- The WSDL Metamodel
- Case Study
- Conclusions
- Future Works

Introduction

- **Services: one of the most important issues in the scope of the WIS development**
  - Applications: *collection of services available through the Web*
- **Technologies: JAVA, .Net, …**
- **No solid methodological basis for development of:**
  - Service-Oriented Systems
  - Web Services

  ➔New methods and modeling techniques needed to guarantee quality
- **New modeling techniques and methodologies for WIS and Service-Oriented WIS development**



MIDAS

Index

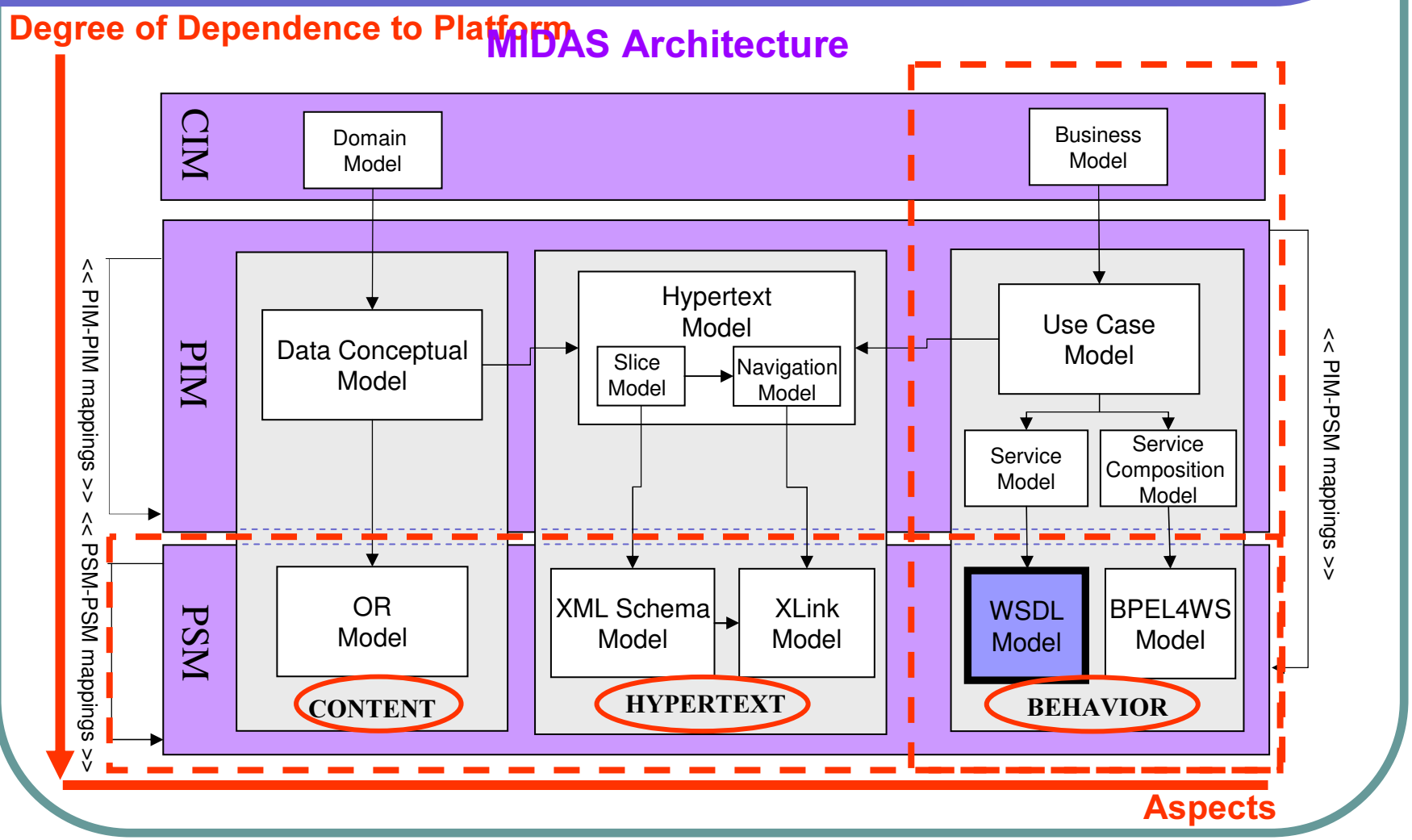
- *Introduction*
- MIDAS Framework
- The WSDL Metamodel
- Case Study
- Conclusions
- Future Works

MIDAS Framework

- **MIDAS**
  - a Model-Driven Methodology for WIS development
  - specific application of MDA metamodel to Web platform
  - defines CIMs, PIMs, PSMs and mappings
  - Main characteristics:
    - an iterative and incremental process based on prototyping
    - use techniques based on agile methodologies
    - use of standards: UML, XML y SQL:1999
    - **unique notation to model the whole system: UML**

Representing Web Services with WSDL ⇨ **UML**

MIDAS Framework



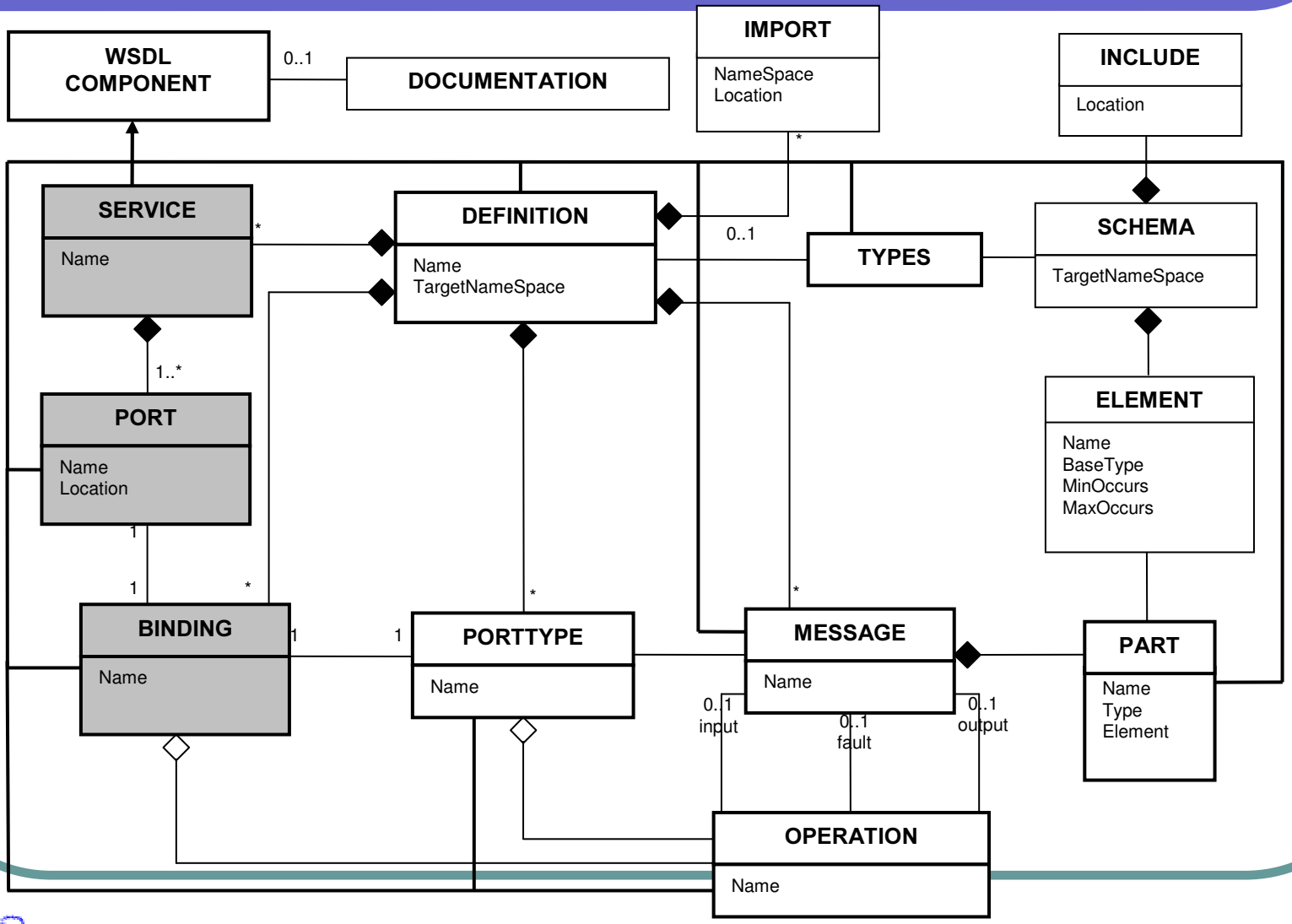
Index

- *Introduction*
- *MIDAS Framework*
- The WSDL Metamodel
- Case Study
- Conclusions
- Future Works

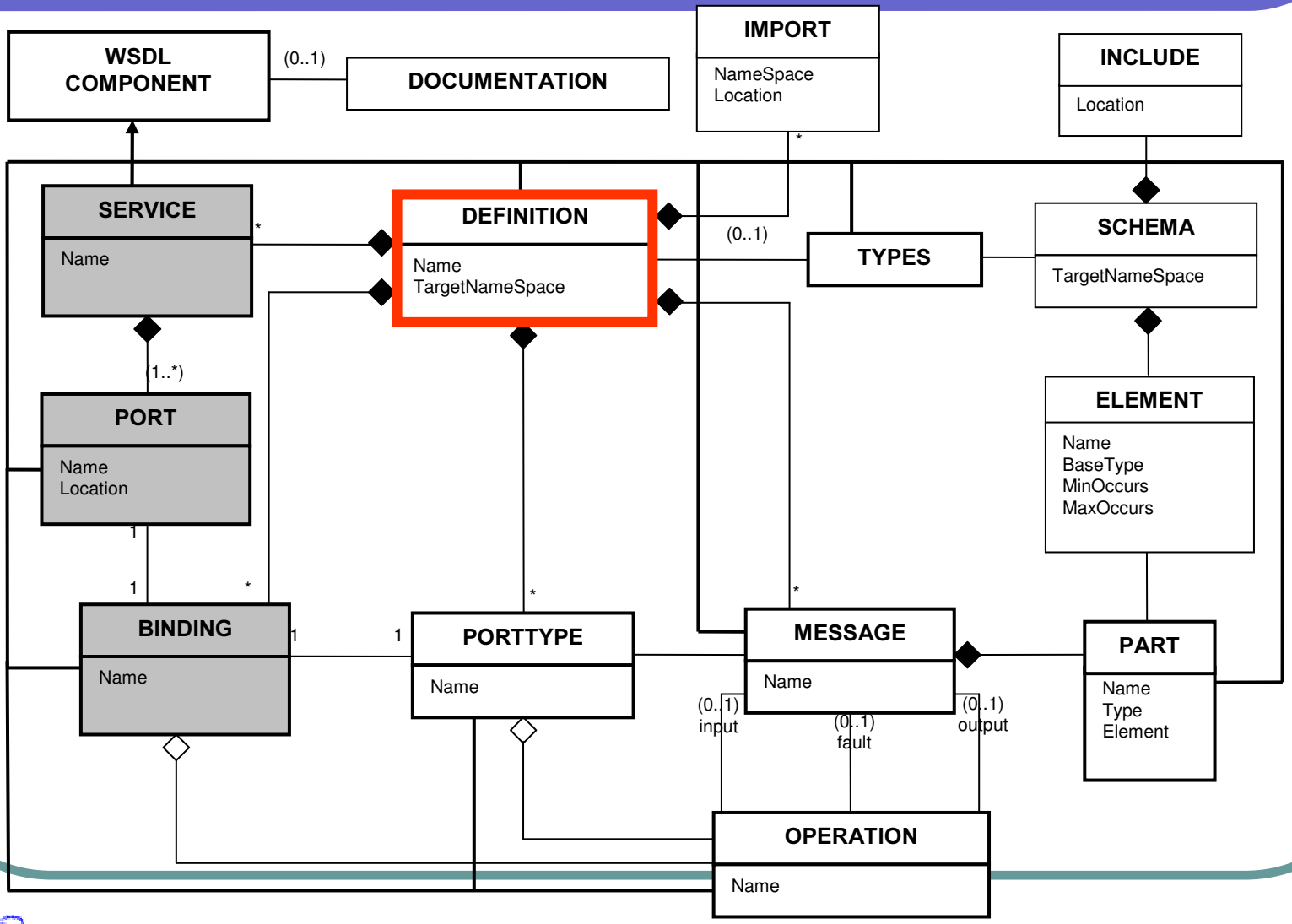
The WSDL Metamodel (I)

- WSDL is proposed by the W3C
- Provides a XML format to describe Web services
- Describes:
  - *Operations*
  - *Messages (Service Provider ↔ Service Requestor)*
  - *Protocols to access service operations*
  - *Localization of the service (URL)*
- Enables to separate:
  - *"Abstract" description (PortType)*
  - *"Concrete" details description (message format and transmission protocol: SOAP, HTTP or MIME)*

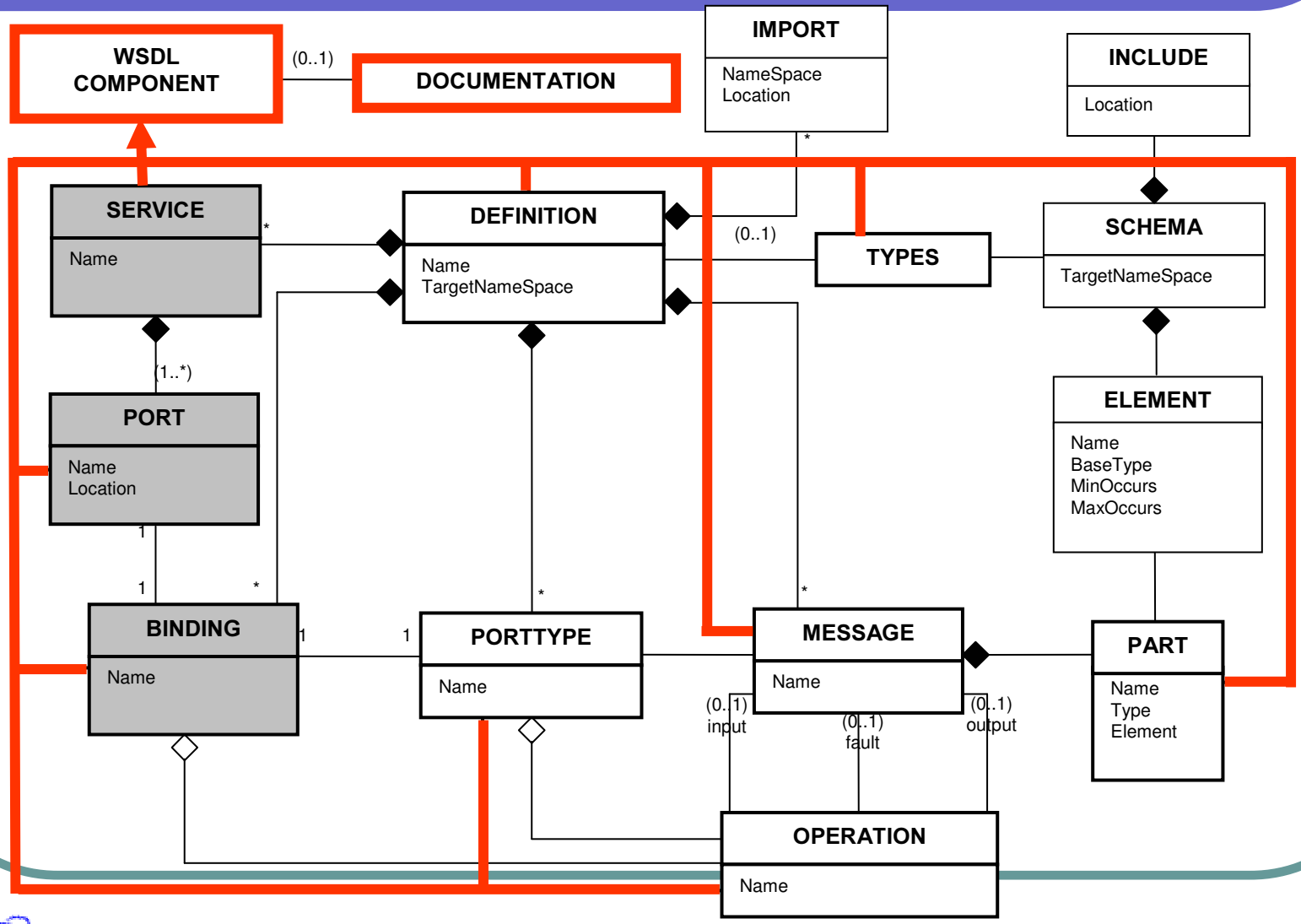
The WSDL Metamodel (II)



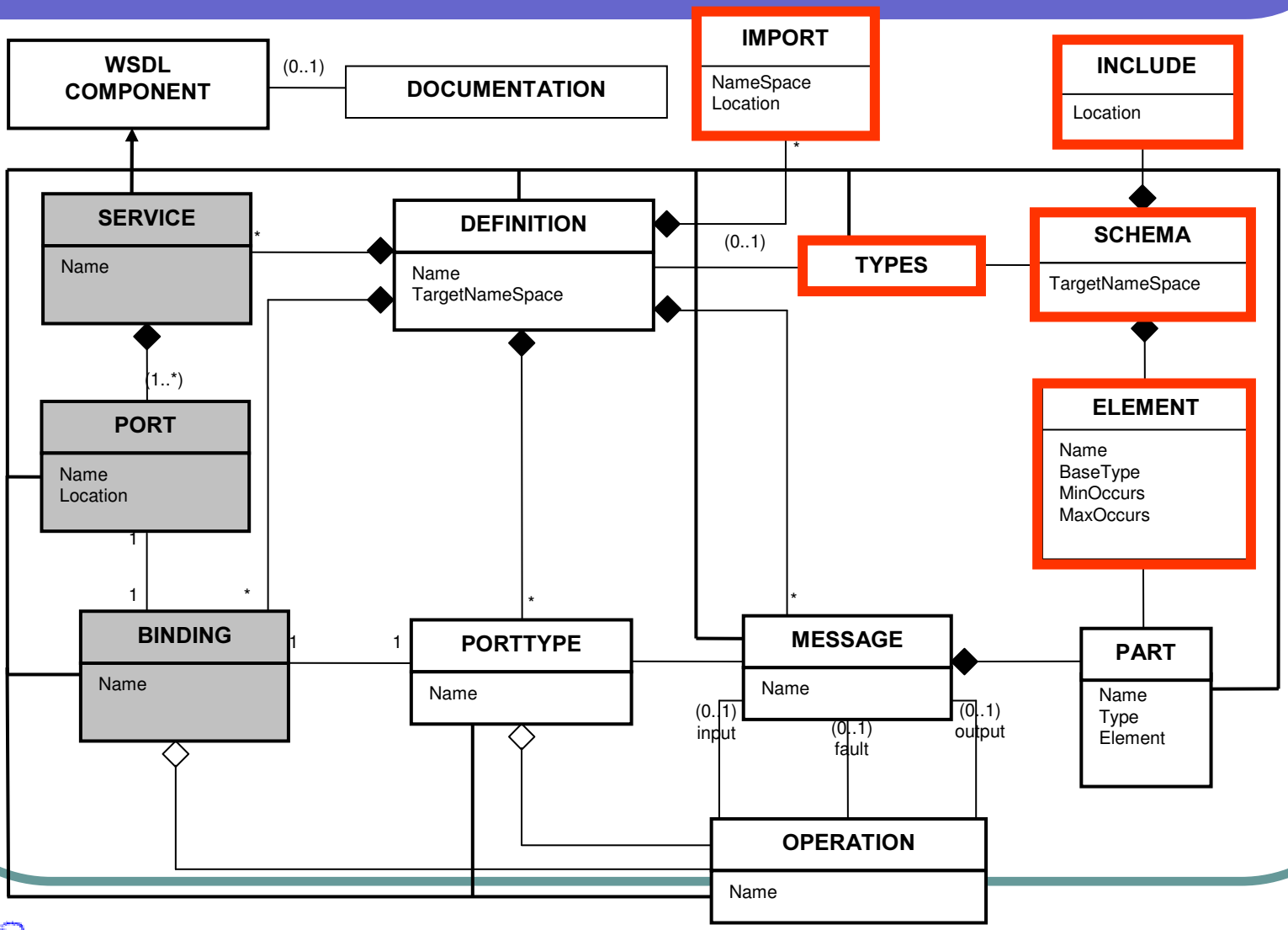
The WSDL Metamodel (II)



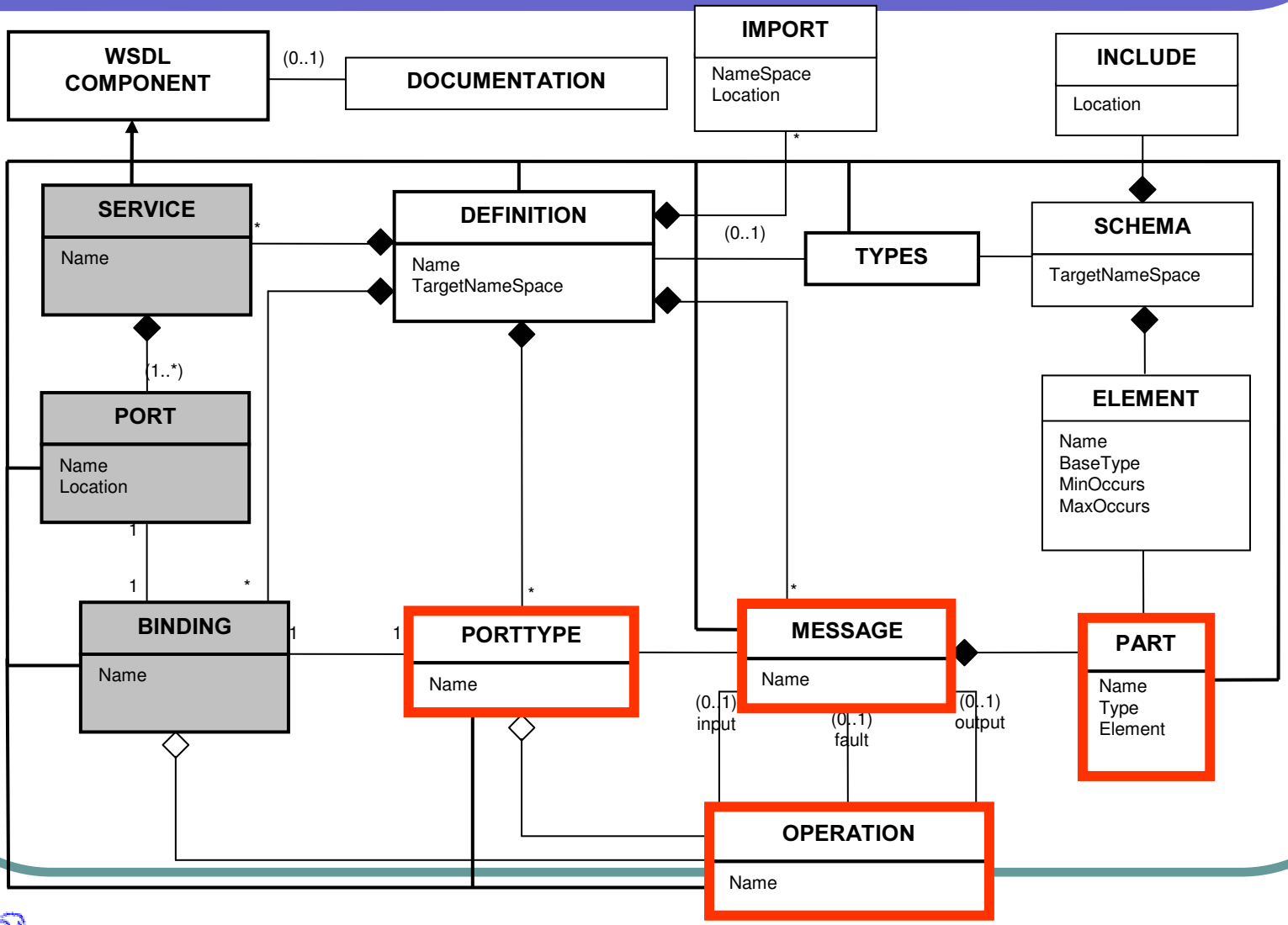
The WSDL Metamodel (II)



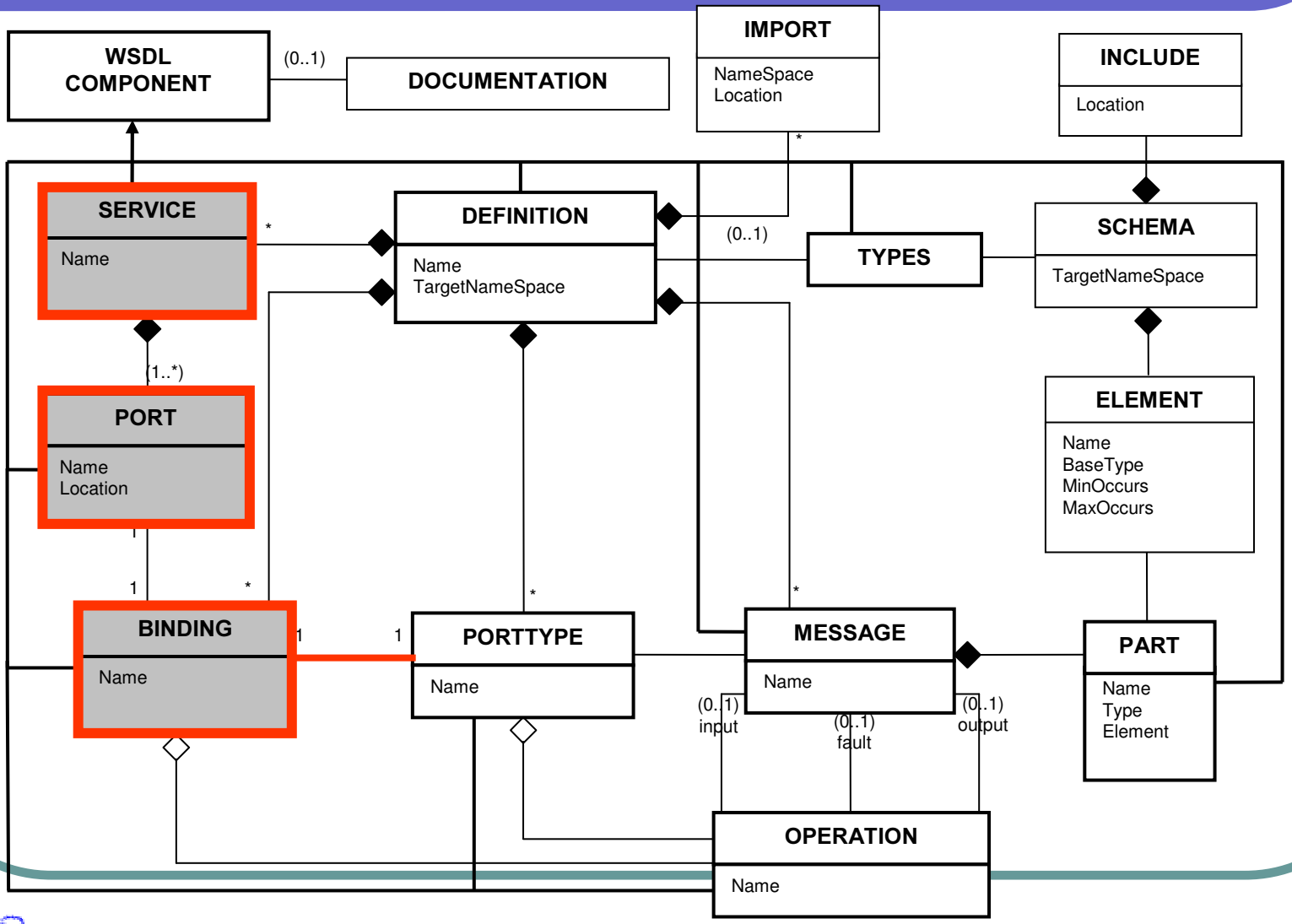
The WSDL Metamodel (II)



The WSDL Metamodel (II)



The WSDL Metamodel (II)



Index

- *Introduction*
- *MIDAS Framework*
- *The WSDL Metamodel*
- Case Study
- Conclusions
- Future Works

Case Study (I)

```
<?xml version="1.0"?>
<definitions name="FlightService"
  targetNamespace="http://example.com/flightinfo.wsdl"
  xmlns:tns="http://example.com/flightinfo.wsdl"
  xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap
  xmlns:xsd1="http://example.com/flightinfo.xsd"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace=http://example.com/flightinfo.xsd
    xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TypeFlightInfo">
        <complexType>
            <all>
            <element name="Gate" type="int"/>
            <element name="State" type="string"/>
            </all>
        </complexType>
      </element>
      <element name="Ticket">
        <complexType>
            <all>
            <element name="AirlineName" type="string"/>
            <element name="FlightNum" type="int"/>
            <element name="DepartureDate" type="date"/>
            <element name="DepartureTime" type="time"/>
            </all>
        </complexType>
      </element>
```

```
<portType name="AirportServPortType">
    <operation name="GetFlightInfo">
        <input message="tns:GetFlightInfoInput"/>
        <output message="tns:GetFlightInfoOutput"/>
    </operation>
    <operation name="CheckIn">
        <input message="tns:CheckInInput"/>
    </operation>
</portType>
```

```
<message name="GetFlightInfoInput">
    <part name="AirlineName" type="xsd:string"/>
    <part name="FlightNum" type="xsd:int"/>
</message>
<message name="GetFlightInfoOutput">
    <part name="FlightInfo" type="xsd1:TypeFlightInfo"/>
</message>
<message name="CheckInInput">
    <part name="Body" element="xsd1:Ticket"/>
</message>
```

```
        <soap:operation style="rpc"
            soapAction="http://example.com/FlightInfo"/>
        <input>
            <soap:body use="encoded"
            namespace=" http://example.com/FlightInfo"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
        <output>
            <soap:body use="encoded"
            namespace=" http://example.com/FlightInfo"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </output>
    </operation>
    <operation name="CheckIn">
        <soap:operation style="document"
            soapAction=" http://example.com/CheckIn"/>
        <input>
            <soap:body use="literal"/>
        </input>
    peration>

    me="FlightService">
    ame="FlightServicePort"
    ding="tns: AirportServBinding">
    ap:address location=" http://example.com/flightinfo"/>

    s>
```

Case Study (I)

```
<?xml version="1.0"?>
<definitions name="FlightService"
  targetNamespace="http://example.com/flightinfo.wsdl"
  xmlns:tns="http://example.com/flightinfo.wsdl"
  xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap
  xmlns:xsd1="http://example.com/flightinfo.xsd"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace=http://example.com/fligh
    xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TypeFlightInfo">
        <complexType>
            <all>
            <element name="Gate" type="int"/>
            <element name="State" type="string"/>
            </all>
        </complexType>
      </element>
      <element name="Ticket">
        <complexType>
            <all>
            <element name="AirlineName" type="string"
            <element name="FlightNum" type="int"/>
            <element name="DepartureDate" type="dat
            <element name="DepartureTime" type="tim
            </all>
        </complexType>
      </element>
    </schema>
  </types>

<message name="GetFlightInfoInput">
    <part name="AirlineName" type="xsd:string"/>
    <part name="FlightNum" type="xsd:int"/>
</message>
<message name="GetFlightInfoOutput">
    <part name="FlightInfo" type="xsd1:TypeFlight
</message>
<message name="CheckInInput">
    <part name="Body" element="xsd1:Ticket"/>
</message>
```

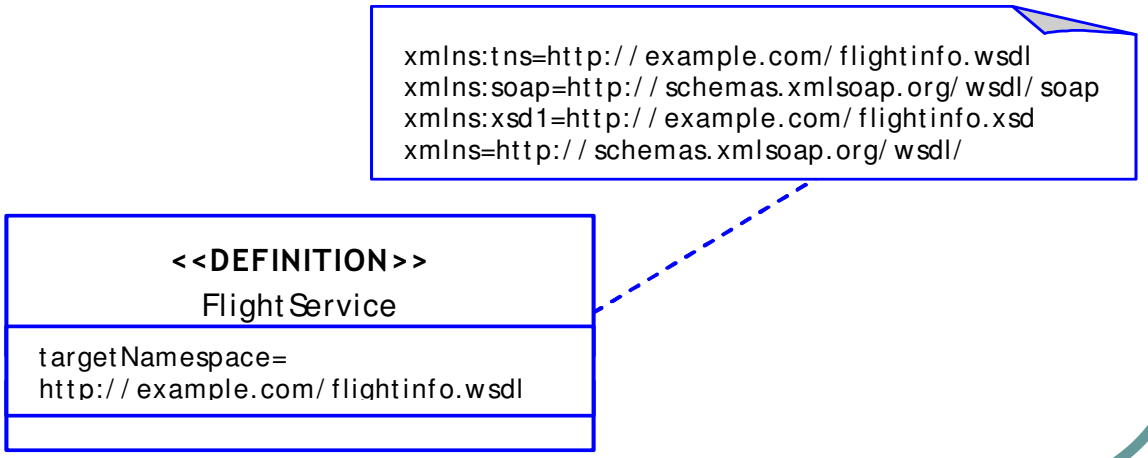
```
<portType name="AirportServPortType">
    <operation name="GetFlightInfo">
        <input message="tns:GetFlightInfoInput"/>
        <output message="tns:GetFlightInfoOutput"/>
    </operation>
    <operation name="CheckIn">
```

```
<binding name="AirportServBinding"
    type="tns:AirportServPortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetFlightInfo">
            <soap:operation style="rpc"
                soapAction="http://example.com/FlightInfo"/>
            <input>
                <soap:body use="encoded"
                namespace=" http://example.com/FlightInfo"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </input>
            <output>
                <soap:body use="encoded"
                namespace=" http://example.com/FlightInfo"
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
            </output>
        </operation>
        <operation name="CheckIn">
            <soap:operation style="document"
                soapAction=" http://example.com/CheckIn"/>
            <input>
                <soap:body use="literal"/>
            </input>
        </operation>
```

```
<service name="FlightService">
    <port name="FlightServicePort"
        binding="tns: AirportServBinding">
        <soap:address location=" http://example.com/flightinfo"/>
    </port>
</service>
```

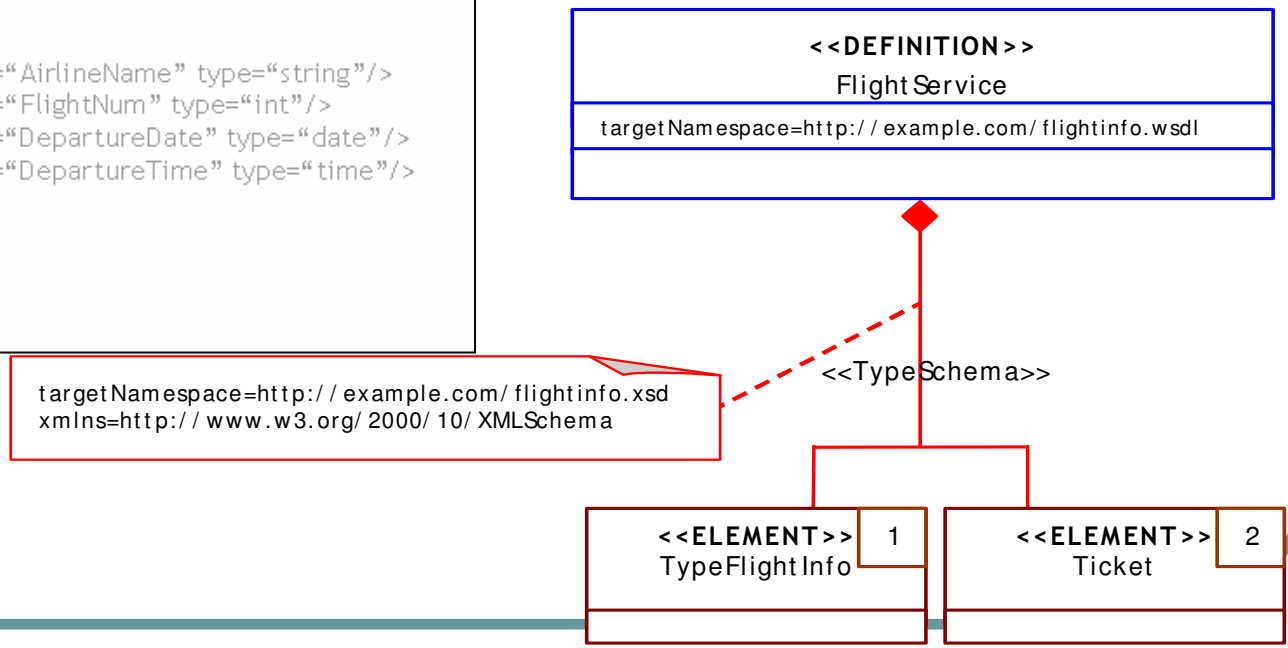
Case Study (II)

```
<?xml version="1.0"?>
<definitions name="FlightService"
   targetNamespace="http://example.com/flightinfo.wsdl"
   xmlns:tns="http://example.com/flightinfo.wsdl"
   xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap
   xmlns:xsd1="http://example.com/flightinfo.xsd"
   xmlns="http://schemas.xmlsoap.org/wsdl/">
...
</definitions>
```



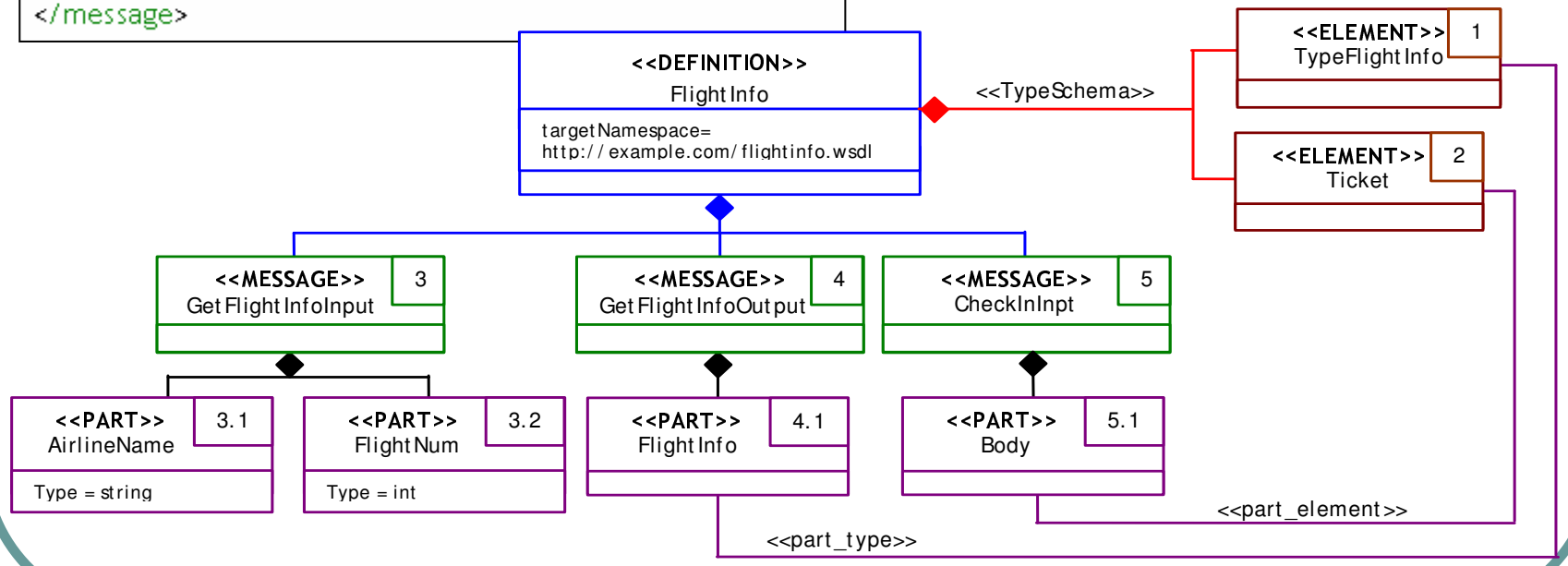
Case Study (III)

```
<types>
  <schema targetNamespace=http://example.com/flightinfo.xsd
  xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="TypeFlightInfo">
      <complexType>
        <all>
          <element name="Gate" type="int"/>
          <element name="State" type="string"/>
        </all>
      </complexType>
    </element>
    <element name="Ticket">
      <complexType>
        <all>
          <element name="AirlineName" type="string"/>
          <element name="FlightNum" type="int"/>
          <element name="DepartureDate" type="date"/>
          <element name="DepartureTime" type="time"/>
        </all>
      </complexType>
    </element>
  </schema>
</types>
```



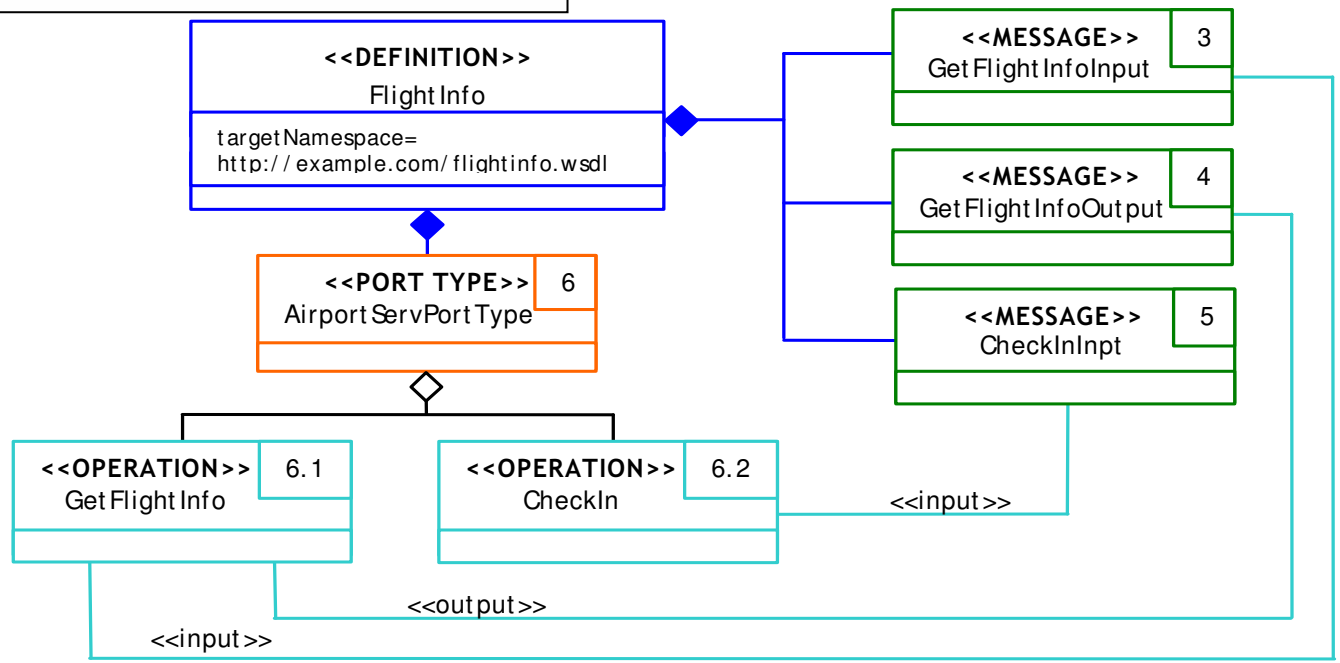
Case Study (IV)

```
<message name="GetFlightInfoInput">
      <part name="AirlineName" type="xsd:string"/>
      <part name="FlightNum" type="xsd:int"/>
</message>
<message name="GetFlightInfoOutput">
      <part name="FlightInfo" type="xsd1:TypeFlightInfo"/>
</message>
<message name="CheckInInput">
      <part name="Body" element="xsd1:Ticket"/>
</message>
```



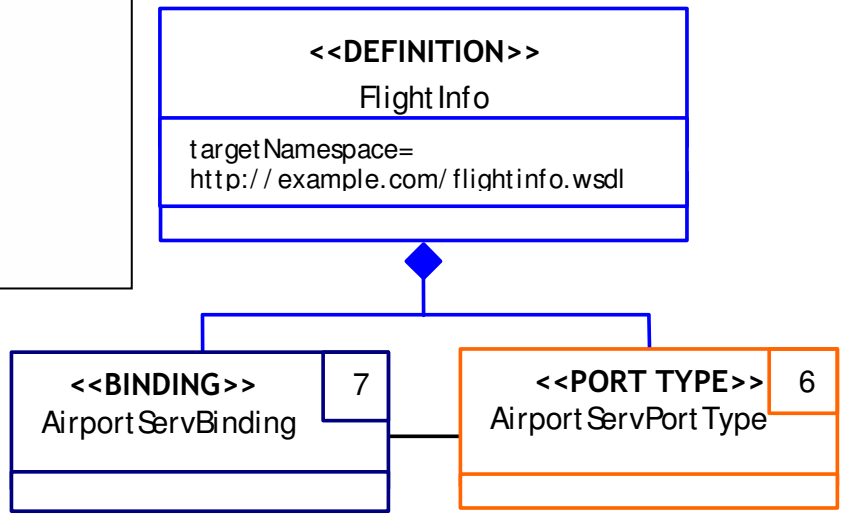
Case Study (V)

```
<portType name="AirportServPortType">
    <operation name="GetFlightInfo">
        <input message="tns:GetFlightInfoInput"/>
        <output message="tns:GetFlightInfoOutput"/>
    </operation>
    <operation name="CheckIn">
        <input message="tns:CheckInInput"/>
    </operation>
</portType>
```



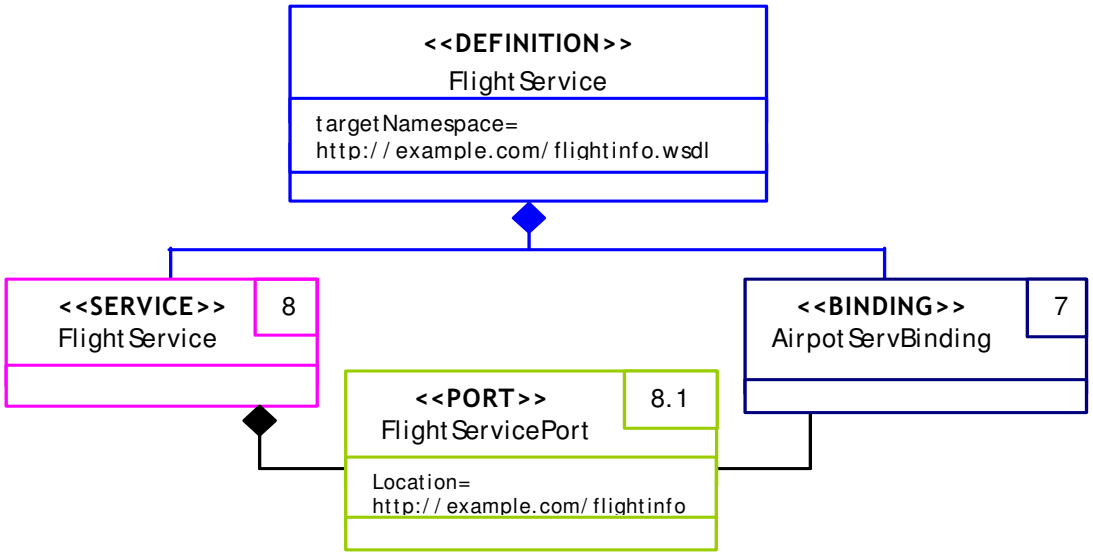
Case Study (VI)

```
<binding name="AirportServBinding"
      type="tns:AirportServPortType">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
            <operation name="GetFlightInfo">
                  <soap:operation style="rpc"
                        soapAction="http://example.com/FlightInfo"/>
                  <input>
                        <soap:body use="encoded"
                        namespace=" http://example.com/FlightInfo"
                        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                  </input>
                  <output>
                        <soap:body use="encoded"
                        namespace=" http://example.com/FlightInfo"
                        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
                  </output>
            </operation>
            <operation name="CheckIn">
                  <soap:operation style="document"
                        soapAction=" http://example.com/CheckIn"/>
                  <input>
                        <soap:body use="literal"/>
                  </input>
            </operation>
</binding>
```

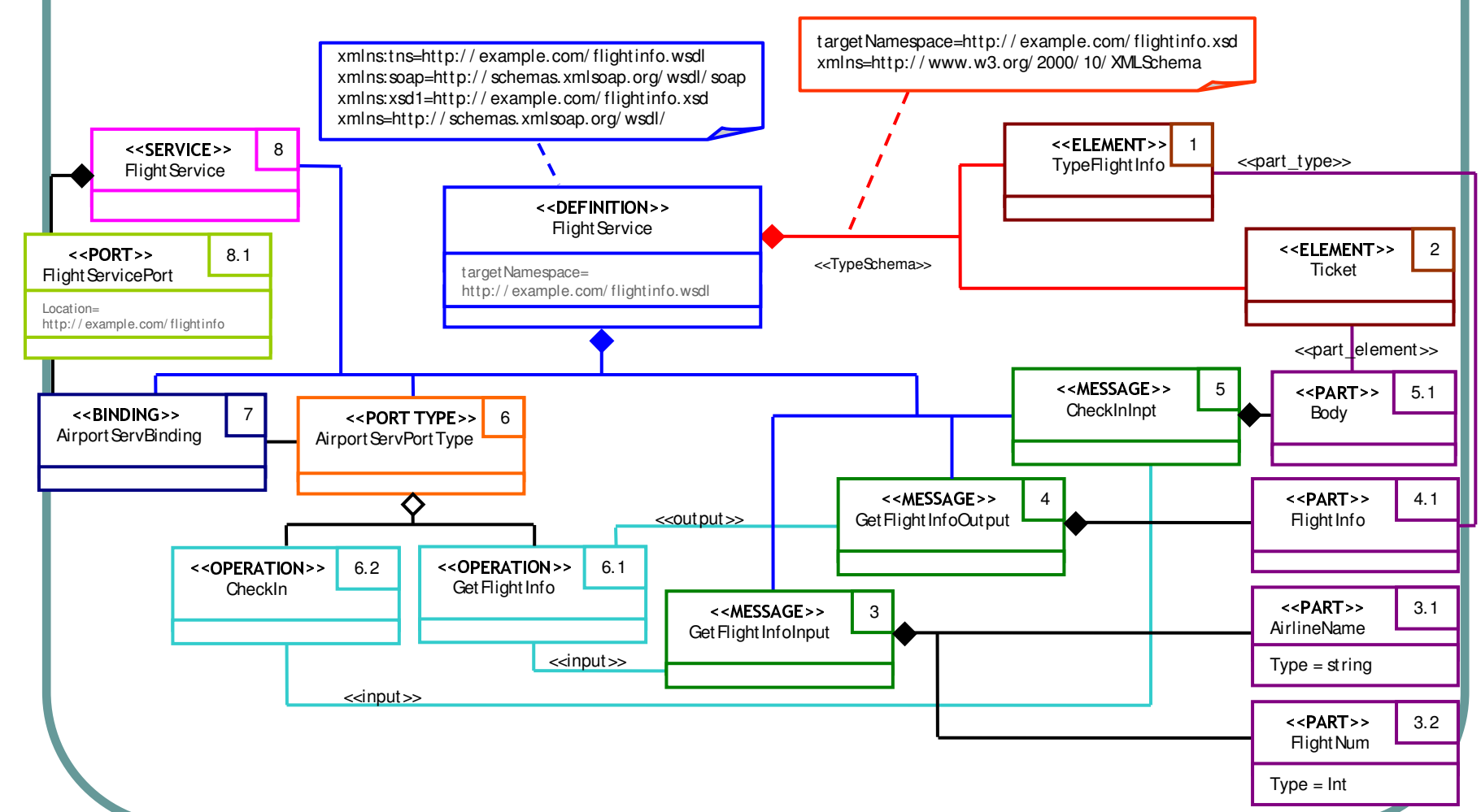


Case Study (VII)

```
<service name="FlightService">
      <port name="FlightServicePort"
            binding="tns: AirportServBinding">
            <soap:address location=" http://example.com/flightinfo"/>
      </port>
</service>
```



Case Study (VIII)

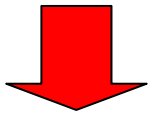


Index

- *Introduction*
- *MIDAS Framework*
- *The WSDL Metamodel*
- *Case Study*
- Conclusions
- Future Works

Conclusions

- MIDAS: a methodological framework for WIS development
  - UML as unique notation to model the whole system



An UML extension to Model Web Services

⇨ The WSDL Metamodel

⇨ Web Service: *"FlightService"*

⇨ Graphical representation in extended UML

Future Works

- Automatic generation of services in WSDL from a UML diagram
- Definition of new stereotypes (binding to specific protocols such as SOAP, HTTP o MIME)
- Incorporate integration techniques in MIDAS
- CASE Tool for semi-automatic generation of WIS development

Rey Juan Carlos University
Kybele Research Group

Representing Web Services with UML: A Case Study

First International Conference on Service Oriented Computing
Trento, Italy

E. Marcos, V. de Castro and **B. Vela**
{emarcos, vcastro, bvela}@escet.urjc.es