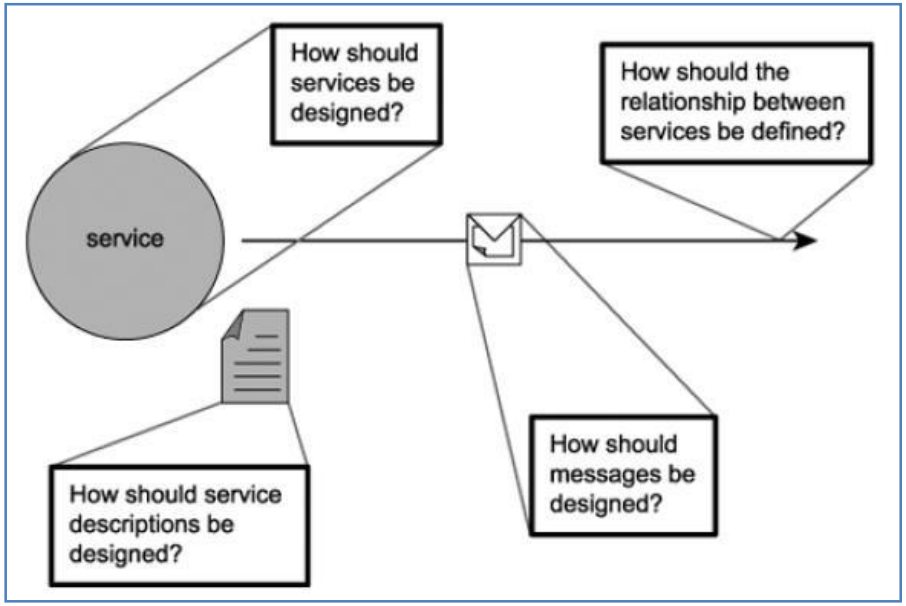
MTIS

SOA
Arquitectura Orientada a Servicios
III

Profesor: Alejandro Sirvent Llamas
Curso: 2019-2020

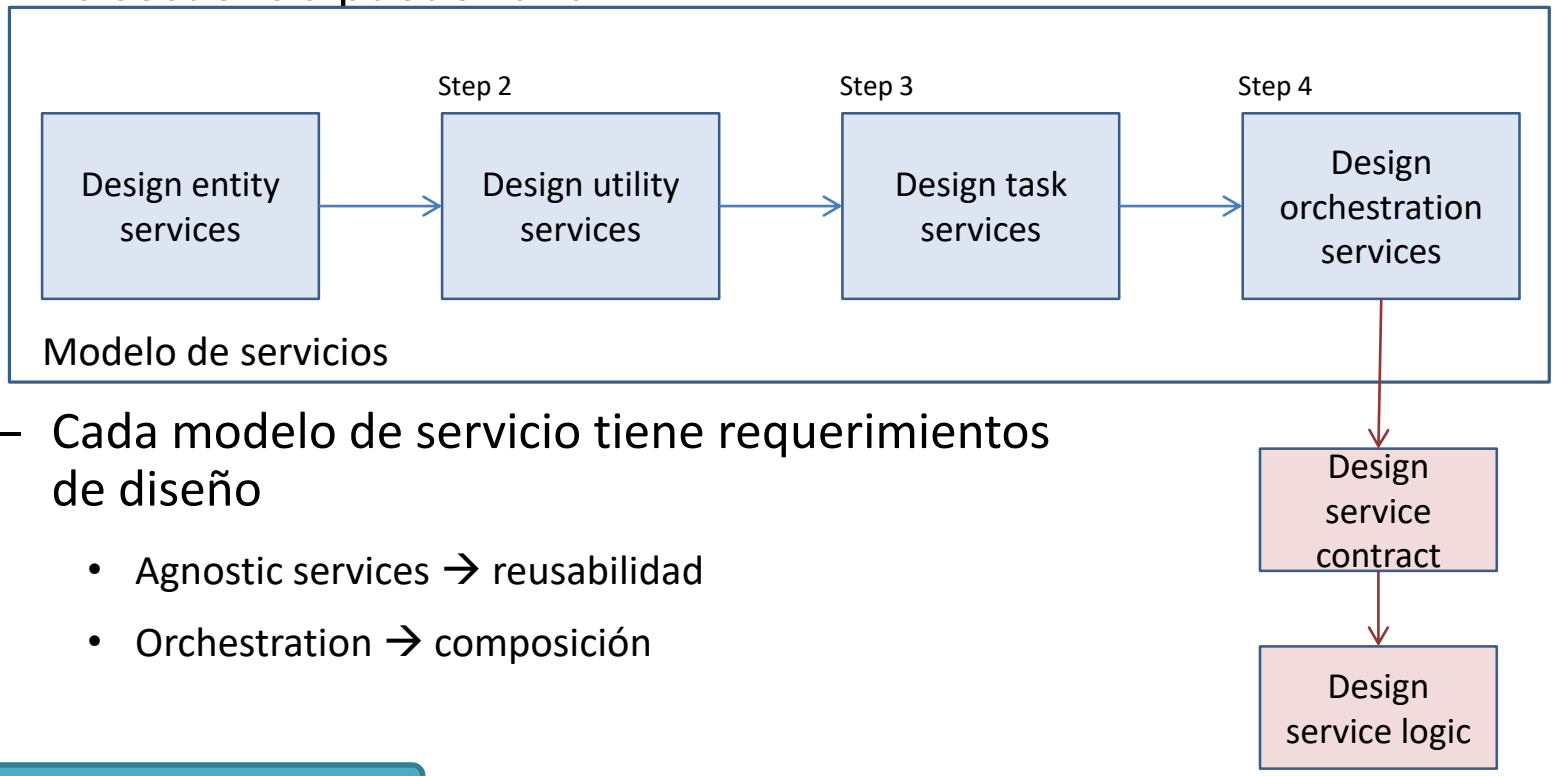
Diseño

- Diseño
  - Logical service candidates → Physical service designs
  - Composición para implementar procesos de negocio
    - Ensamblaje de servicios
  - Cuestiones
    - Cómo definir interfaces de servicios físicas desde servicios candidatos?
    - Qué características SOA quiero implementar y soportar?
    - Qué estándares y tecnologías usaremos para implementar el diseño?



- Diseño
  - Logical service candidates → Physical service designs
    - Teniendo en cuenta los cuatro tipos principales de capas de servicios que hemos identificado anteriormente, la secuencia de diseño sería:
      - » Entity-centric business services
      - » Application services
      - » Task-centric business services
      - » Process services

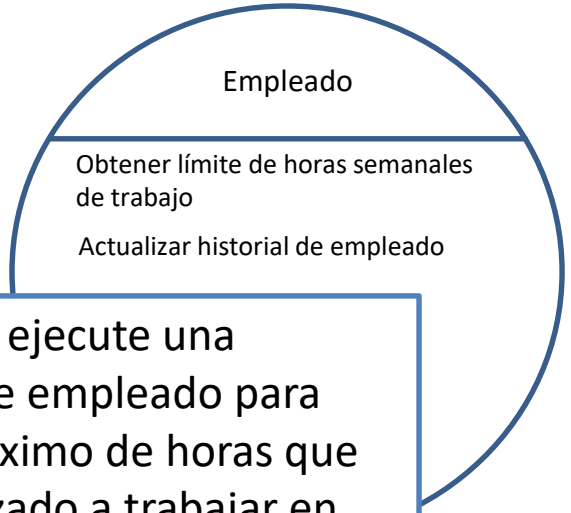
- Diseño
  - Una fase por cada tipo de servicio
  - La secuencia puede variar



  - Cada modelo de servicio tiene requerimientos de diseño
    - Agnostic services → reusabilidad
    - Orchestration → composición

- Diseño
  - Dentro del caso de estudio tenemos los siguiente servicios candidatos:
    - TLS
      - Proceso de envío de partes de horas (Proceso).
      - Empleados (Entity service).
      - Partes de horas (Entity service).
      - Facturas (Entity service).
      - Notificaciones (Application Service)
    - RailCo
      - Proceso de facturación (Task-centric).
      - Proceso de OP (Task-centric).
      - Sistemas Heredados (Application Service).
      - Notificación de sondeo (Application Service).
      - Transformar documentos de contabilidad (Application Service).

- Diseño
  - Entity Service
    - Revisar servicios existentes (candidatos)
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la
    - Extender el diseño de
    - Identificar requerimi



- La primera requiere que ejecute una consulta en el registro de empleado para recuperar el número máximo de horas que el empleado está autorizado a trabajar en una semana.
- La otra parte de la funcionalidad, es la posibilidad de publicar las actualizaciones del historico de los empleados.

- Diseño
  - Entity Service
    - Revisar servicios existentes (candidatos)
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Extender el diseño de servicios
    - Identificar requerimientos de procesamiento

- Diseño
  - Entity Service

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace=
      "http://www.xmltc.com/tls/employee/schema/hr/">
   <xsd:element name="EmployeeUpdateHistoryRequestType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="ID" type="xsd:integer"/>
            <xsd:element name="Comment" type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="EmployeeUpdateHistoryResponseType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="ResponseCode"
               type="xsd:byte"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace=
      "http://www.xmltc.com/tls/employee/schema/accounting/">
   <xsd:element name="EmployeeHoursRequestType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="ID" type="xsd:integer"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="EmployeeHoursResponseType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="ID" type="xsd:integer"/>
            <xsd:element name="WeeklyHoursLimit"
               type="xsd:short"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
xsd:schema>
```

- Extender el diseño de servicios
- Identificar requerimientos de p

```
<types>
   <xsd:schema targetNamespace=
      "http://www.xmltc.com/tls/employee/schema/">
      <xsd:import namespace=
         "http://www.xmltc.com/tls/employee/schema/accounting/"
         schemaLocation="Employee.xsd"/>
      <xsd:import namespace=
         "http://www.xmltc.com/tls/employee/schema/hr/"
         schemaLocation="EmployeeHistory.xsd"/>
   </xsd:schema>
</types>
```

Diseño - Caso de estudio

- Diseño
  - Entity Service
    - Revisar serv
    - Definir sche
    - Parte abs
    - Aplicar orier
      - Reusab
      - Autono
      - Sin esta
      - Descub
    - Estandariza
    - Extender el
    - Identificar r

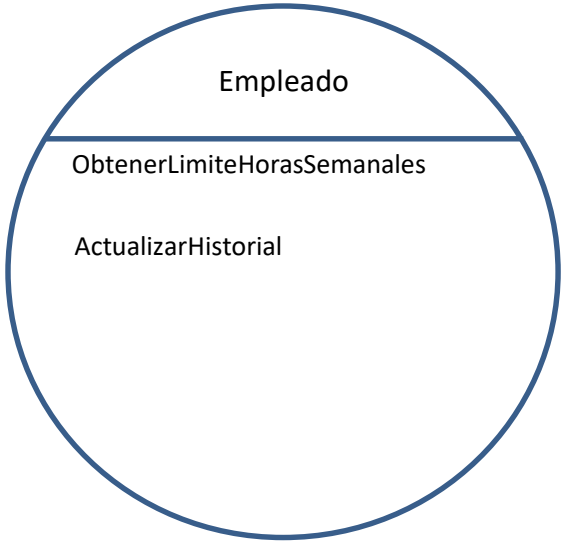
```
<message name="getEmployeeWeeklyHoursRequestMessage">
   <part name="RequestParameter"
      element="act:EmployeeHoursRequestType"/>
</message>
<message name="getEmployeeWeeklyHoursResponseMessage">
   <part name="ResponseParameter"
      element="act:EmployeeHoursResponseType"/>
</message>
<message name="updateEmployeeHistoryRequestMessage">
   <part name="RequestParameter"
      element="hr:EmployeeUpdateHistoryRequestType"/>
</message>
<message name="updateEmployeeHistoryResponseMessage">
   <part name="ResponseParameter"
      element="hr:EmployeeUpdateHistoryResponseType"/>
</message>
<portType name="EmployeeInterface">
   <operation name="GetEmployeeWeeklyHoursLimit">
      <input message=
         "tns:getEmployeeWeeklyHoursRequestMessage"/>
      <output message=
         "tns:getEmployeeWeeklyHoursResponseMessage"/>
   </operation>
   <operation name="UpdateEmployeeHistory">
      <input message=
         "tns:updateEmployeeHistoryRequestMessage"/>
      <output message=
         "tns:updateEmployeeHistoryResponseMessage"/>
   </operation>
</portType>
```

- Diseño
  - Entity Serv
    - Revisar
    - Definir s
    - Parte ab
    - Aplica
      - Re
      - Au
      - Sin
      - De
    - Estandarización de la interfaz
    - Extender el diseño de servicios
    - Identificar requerimientos de procesamiento

```
<portType name="EmployeeInterface">
   <documentation>
      GetEmployeeWeeklyHoursLimit uses the Employee
      ID value to retrieve the WeeklyHoursLimit value.
      UpdateEmployeeHistory uses the Employee ID value
      to update the Comment value of the EmployeeHistory.
   </documentation>
   <operation name="GetEmployeeWeeklyHoursLimit">
      <input message=
         "tns:getEmployeeWeeklyHoursRequestMessage"/>
      <output message=
         "tns:getEmployeeWeeklyHoursResponseMessage"/>
   </operation>
   <operation name="UpdateEmployeeHistory">
      <input message=
         "tns:updateEmployeeHistoryRequestMessage"/>
      <output message=
         "tns:updateEmployeeHistoryResponseMessage"/>
   </operation>
</portType>
```

- Diseño
  - Entity Service
    - Revisar servicios existentes (candidatos)
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Extender el diseño de servicios
    - Identificar requerimientos de procesamiento

```
<operation name="GetWeeklyHoursLimit">
  <input message="tns:getWeeklyHoursRequestMessage"/>
  <output message="tns:getWeeklyHoursResponseMessage"/>
</operation>
<operation name="UpdateHistory">
  <input message="tns:updateHistoryRequestMessage"/>
  <output message="tns:updateHistoryResponseMessage"/>
</operation>
```



El uso de estándares de nomenclatura ofrece soporte nativo para la interoperabilidad intrínseca.

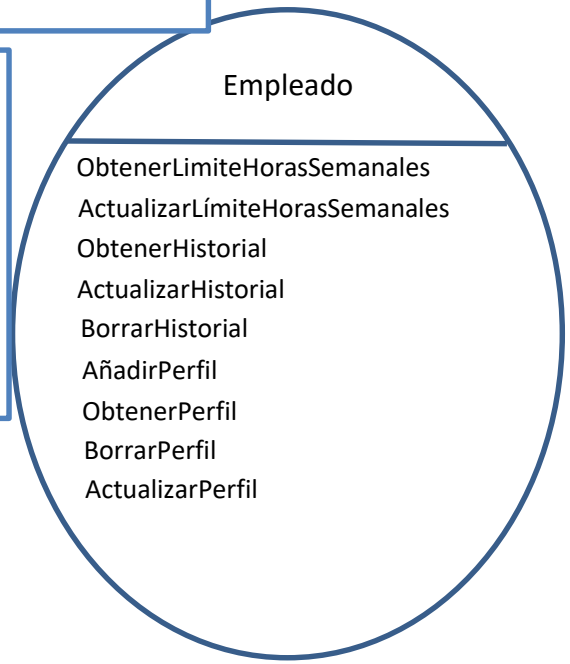
- Diseño
  - Entity Service
    - Revisar servicios existentes (candidat
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Estandarización de la interfaz
    - Extender el diseño de servicios
    - Identificar requerimientos de procesamiento

Operaciones típicas:

GetSomething
UpdateSomething
AddSomething
DeleteSomething

Este paso consiste en la realización de un análisis especulativo sobre que otras prestaciones debe ofrecer este servicio, dos formas de implementar nuevas funcionalidades:

- Añadir nuevas operaciones.
- Agregar nuevos parámetros para las operaciones existente. (Se pierde la intuición de que hace)



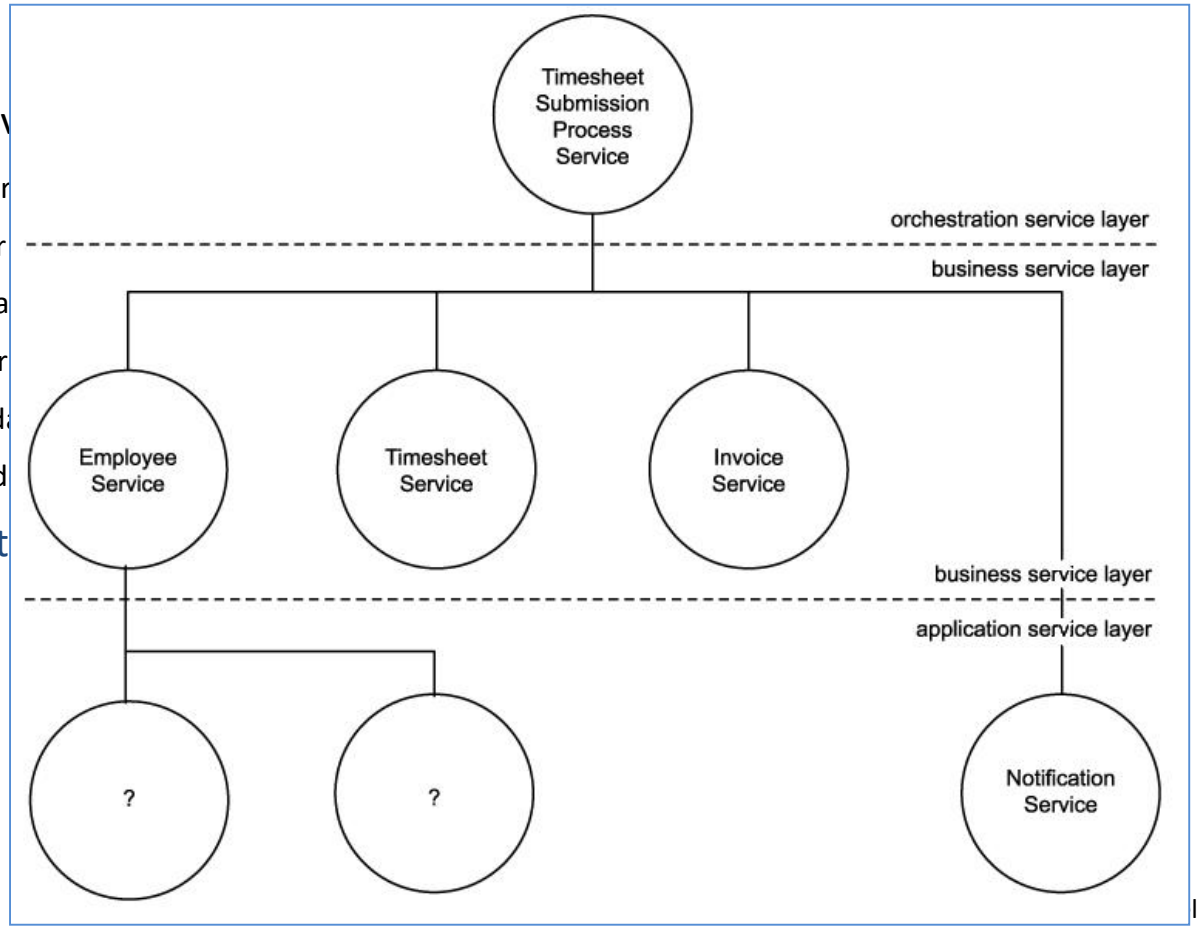
MTIS

- Diseño
  - Entity Service
    - Revisar servicios existentes (can
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Extender el diseño de ser
    - Identificar requerimientos de pr

```
<portType name="EmployeeInterface">
   <operation name="GetWeeklyHoursLimit">
      <input message="tns:getWeeklyHoursRequestMessage"/>
      <output message="tns:getWeeklyHoursResponseMessage"/>
   </operation>
   <operation name="UpdateWeeklyHoursLimit">
      <input message=
         "tns:updateWeeklyHoursRequestMessage"/>
      <output message=
         "tns:updateWeeklyHoursResponseMessage"/>
   </operation>
   <operation name="GetHistory">
      <input message="tns:getHistoryRequestMessage"/>
      <output message="tns:getHistoryResponseMessage"/>
   </operation>
   <operation name="UpdateHistory">
      <input message="tns:updateHistoryRequestMessage"/>
      <output message="tns:updateHistoryResponseMessage"/>
   </operation>
   <operation name="DeleteHistory">
      <input message="tns:deleteHistoryRequestMessage"/>
      <output message="tns:deleteHistoryResponseMessage"/>
   </operation>
   <operation name="AddProfile">
      <input message="tns:addProfileRequestMessage"/>
      <output message="tns:addProfileResponseMessage"/>
   </operation>
   <operation name="GetProfile">
      <input message="tns:getProfileRequestMessage"/>
      <output message="tns:getProfileResponseMessage"/>
   </operation>
   <operation name="UpdateProfile">
      <input message="tns:updateProfileRequestMessage"/>
      <output message="tns:updateProfileResponseMessage"/>
   </operation>
   <operation name="DeleteProfile">
      <input message="tns:deleteProfileRequestMessage"/>
      <output message="tns:deleteProfileResponseMessage"/>
   </operation>
</portType>
```

- Diseño
  - Entity Service
    - Revisar servicios existentes (candidatos)
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
    - Estandarización de la interfaz
    - Extender el diseño de servicios
    - Identificar requerimientos de procesamiento
      - De dónde se saca la información?
        - » Información de contabilidad de usuario e información de contacto → Sistema de contabilidad
        - » Información de empleados → Sistema de Recursos Humanos
        - » Schema definido
          - Expresa información de empleado
          - La estructura deriva de dos repositorios físicos
        - » Consulta a la base de datos de contabilidad obteniendo el límite de horas semanales a partir del identificador de empleado

- Diseño
  - Entity Serv
    - Revisar
    - Definir
    - Parte a
    - Aplicar
    - Estand
    - Extend
    - Ident
      –



l identificador de empleado

- Diseño
  - Application Service
    - Revisar servicios existentes (candidatos)
    - Validar el contexto
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Añadir posibles características futuras

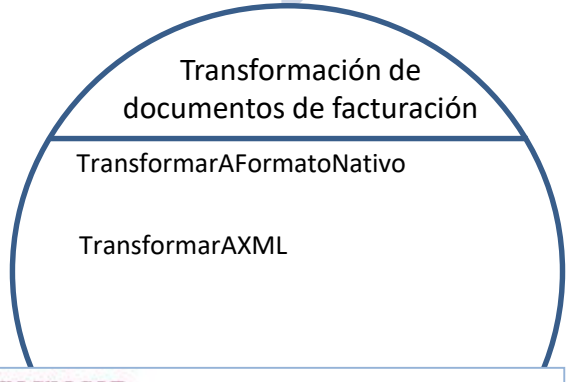
Este tipo de proceso presenta varios pasos similares al entity-centric anterior.

- Diseño
  - Application Service
    - Revisar servicios existentes (candidatos), de los más independientes a los más dependientes:

- Diseño
  - Application Service
    - Validar el contexto
      - Es posible que una o más operaciones pertenezcan a otra aplicación, por eso, la necesidad de revisar y validar el contexto

- Diseño
  - Application Service
    - Revisar servicios existentes (candidatos)
    - Validar el contexto
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Añadir posibles características futuras

- Diseño




```
<xsd:element name="TransformToXMLType">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="SourcePath"
            type="xsd:string"/>
         <xsd:element name="DestinationPath"
            type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
<xsd:element name="TransformToXMLReturnCodeType">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="Code"
            type="xsd:integer"/>
         <xsd:element name="Message"
            type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

datos)

```
<xsd:schema targetNamespace=
   "http://www.xmltc.com/railco/transform/schema/">
   <xsd:element name="TransformToNativeType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="SourcePath"
               type="xsd:string"/>
            <xsd:element name="DestinationPath"
               type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="TransformToNativeReturnCodeType">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Code"
               type="xsd:integer"/>
            <xsd:element name="Message"
               type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

- Estandarización de la interfaz
- Añadir posibles características futu

- Diseño
  - Application Service
    - Revisar servicios existentes (candida
    - Validar el contexto
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía

Aunque coinciden los datos para cada operación se definen unas estructuras concretas.

s futur

```
<message name="transformToNativeRequestMessage">
   <part name="RequestParameter"
      element="trn:TransformToNativeType"/>
</message>
<message name="transformToNativeResponseMessage">
   <part name="ResponseParameter"
      element="trn:TransformToNativeReturnCodeType"/>
</message>

<message name="transformToXMLRequestMessage">
   <part name="RequestParameter"
      element="trn:TransformToXMLType"/>
</message>
<message name="transformToXMLResponseMessage">
   <part name="ResponseParameter"
      element="trn:TransformToXMLReturnCodeType"/>
</message>

<portType name="TransformInterface">
   <operation name="TransformToNative">
      <input message=
            "tns:transformToNativeRequestMessage"/>
         <output message=
            "tns:transformToNativeResponseMessage"/>
   </operation>
   <operation name="TransformToXML">
      <input message=
         "tns:transformToXMLRequestMessage"/>
      <output message=
         "tns:transformToXMLResponseMessage"/>
   </operation>
</portType>
```

- Diseño
  - Application Service
    - Revisar servicios existentes (candidatos)
    - Validar el contexto
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
        - » Se podrían juntar las operaciones anteriores en una?
        - » Es preferible la semántica?
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Añadir posibles características futuras

TransformToNative
TransformToXML

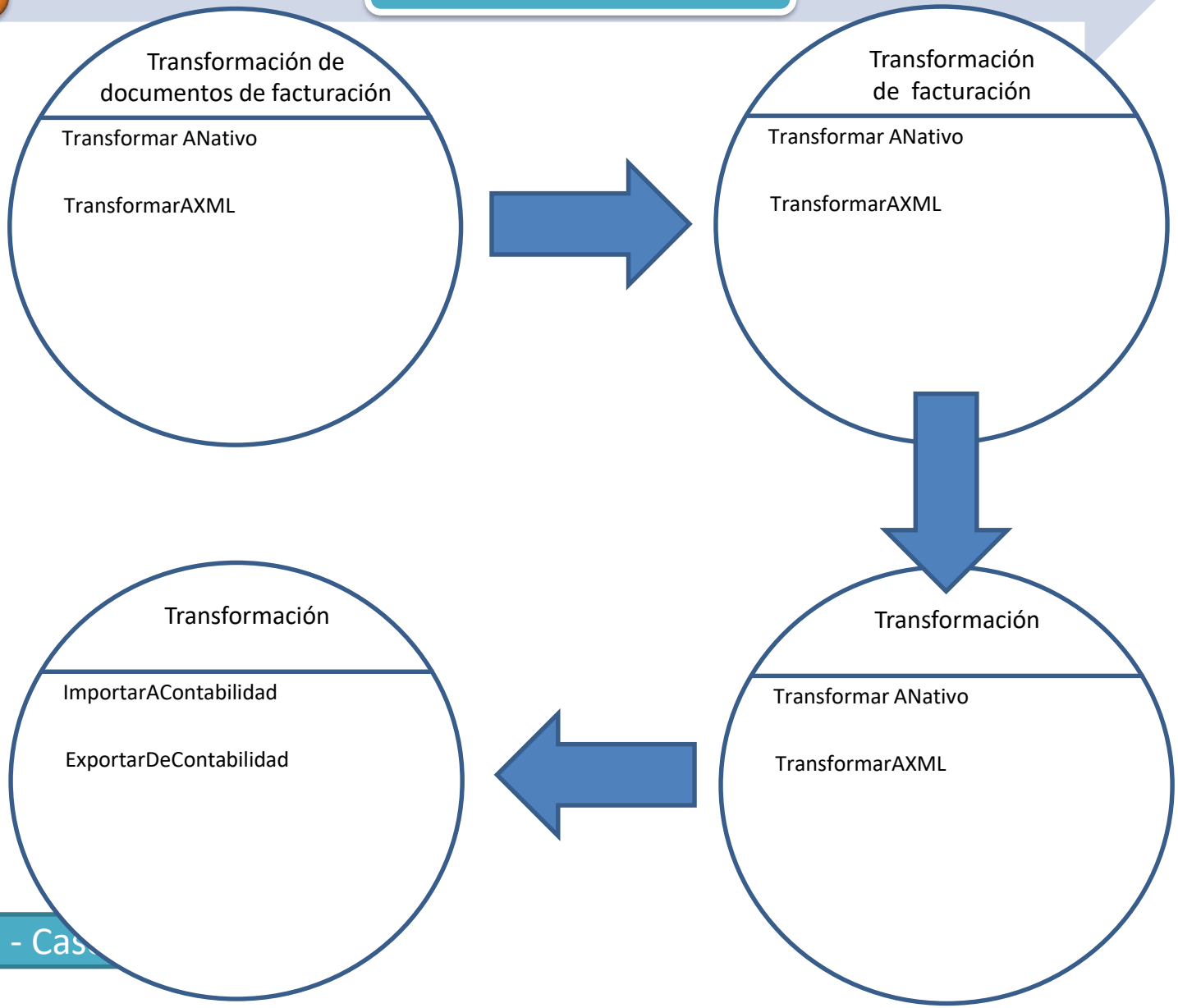
- Diseño
  - Application Service
    - Revisar servicios existentes (candidatos)
    - Validar el contexto
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Añadir posibles características futuras

- Diseño
  - Application Service
    - Revisar servicios existentes (c
    - Validar el contexto
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicios
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Añadir posibles características futuras

```
<portType name="TransformInterface">
   <documentation>
      Retrieves an XML document and converts it
      into the native accounting document format.
   </documentation>
   <operation name="TransformToNative">
      <input message=
         "tns:transformToNativeRequestMessage"/>
      <output message=
         "tns:transformToNativeResponseMessage"/>
   </operation>
   <documentation>
      Retrieves a native accounting document and
      converts it into an XML document.
   </documentation>
   <operation name="TransformToXML">
      <input message=
         "tns:transformToXMLRequestMessage"/>
      <output message=
         "tns:transformToXMLResponseMessage"/>
   </operation>
</portType>
```

MTIS

SOA II

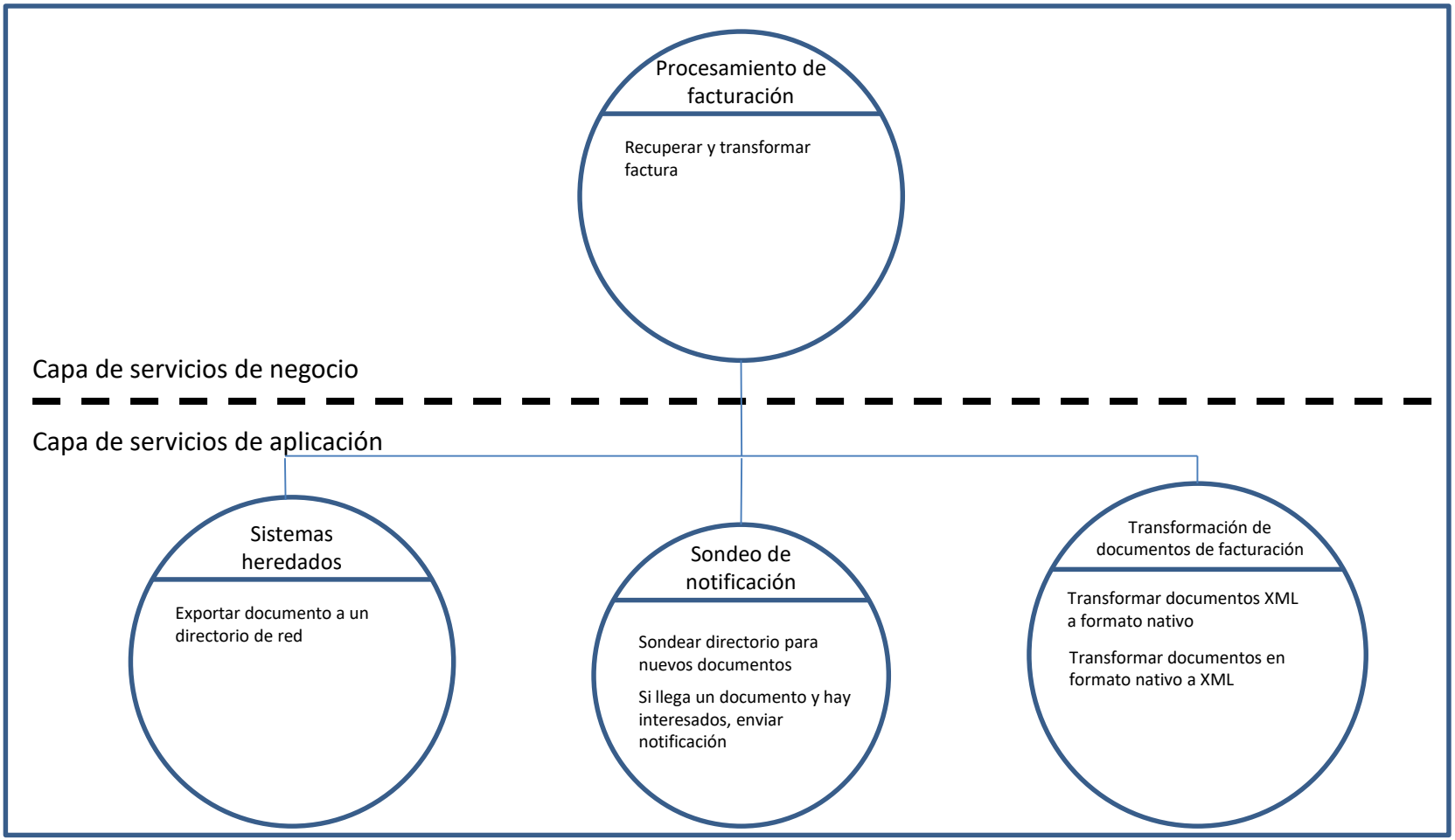


Diseño - Cas

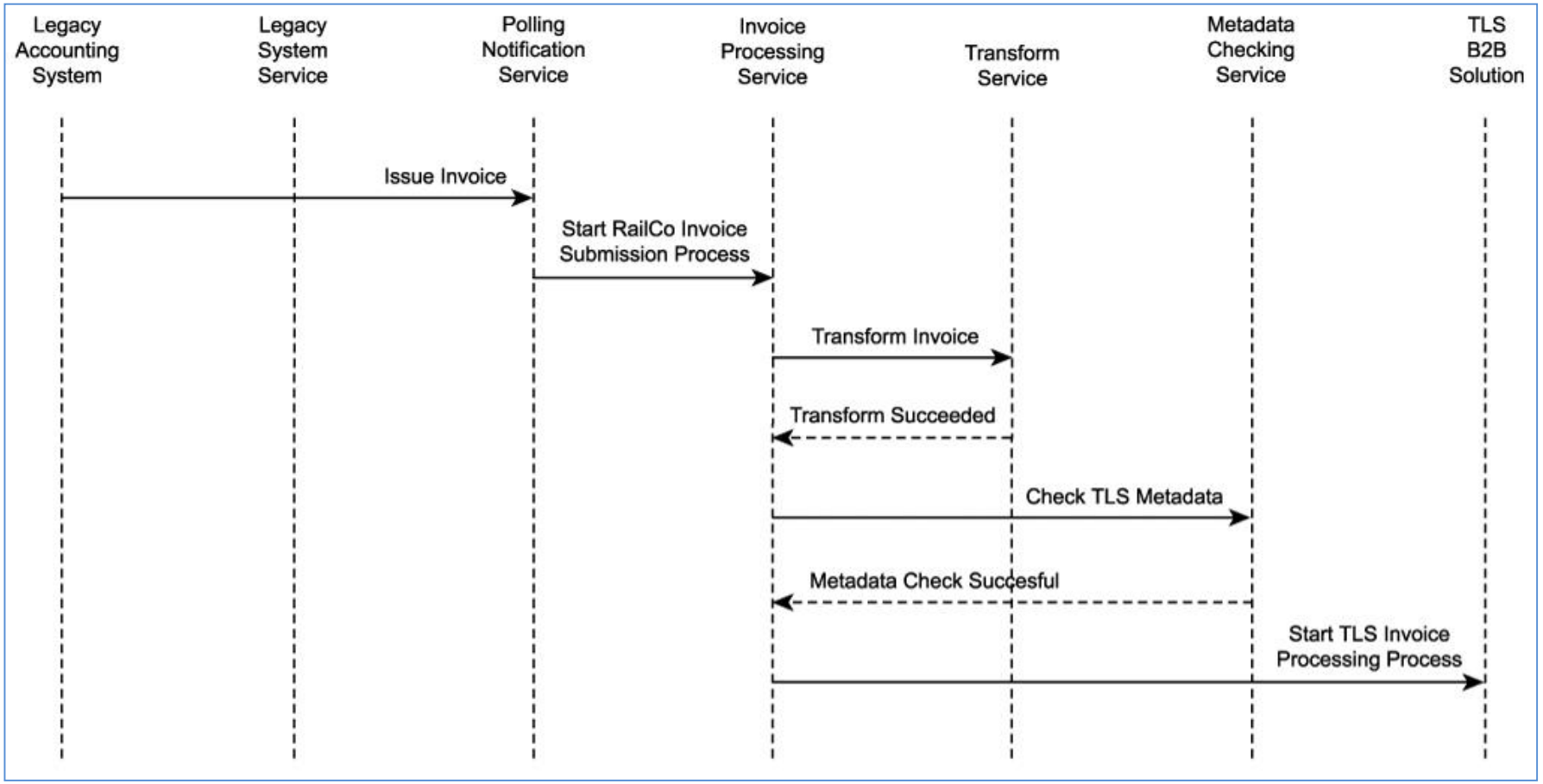
- Diseño
  - Application Service
    - Revisar servicios existentes (
    - Validar el contexto
    - Definir schema de la entidad
    - Parte abstracta de la interfaz
    - Aplicar orientación a servicio
      - Reusabilidad
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz
    - Añadir posibles características futuras

```
<types>
   <xsd:schema targetNamespace=
      "http://www.xmltc.com/railco/transform/schema/">
      <xsd:element name="ForImportType">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="SourcePath"
                  type="xsd:string"/>
               <xsd:element name="DestinationPath"
                  type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="ForImportReturnCodeType">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="Code"
                  type="xsd:integer"/>
               <xsd:element name="Message"
                  type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
```

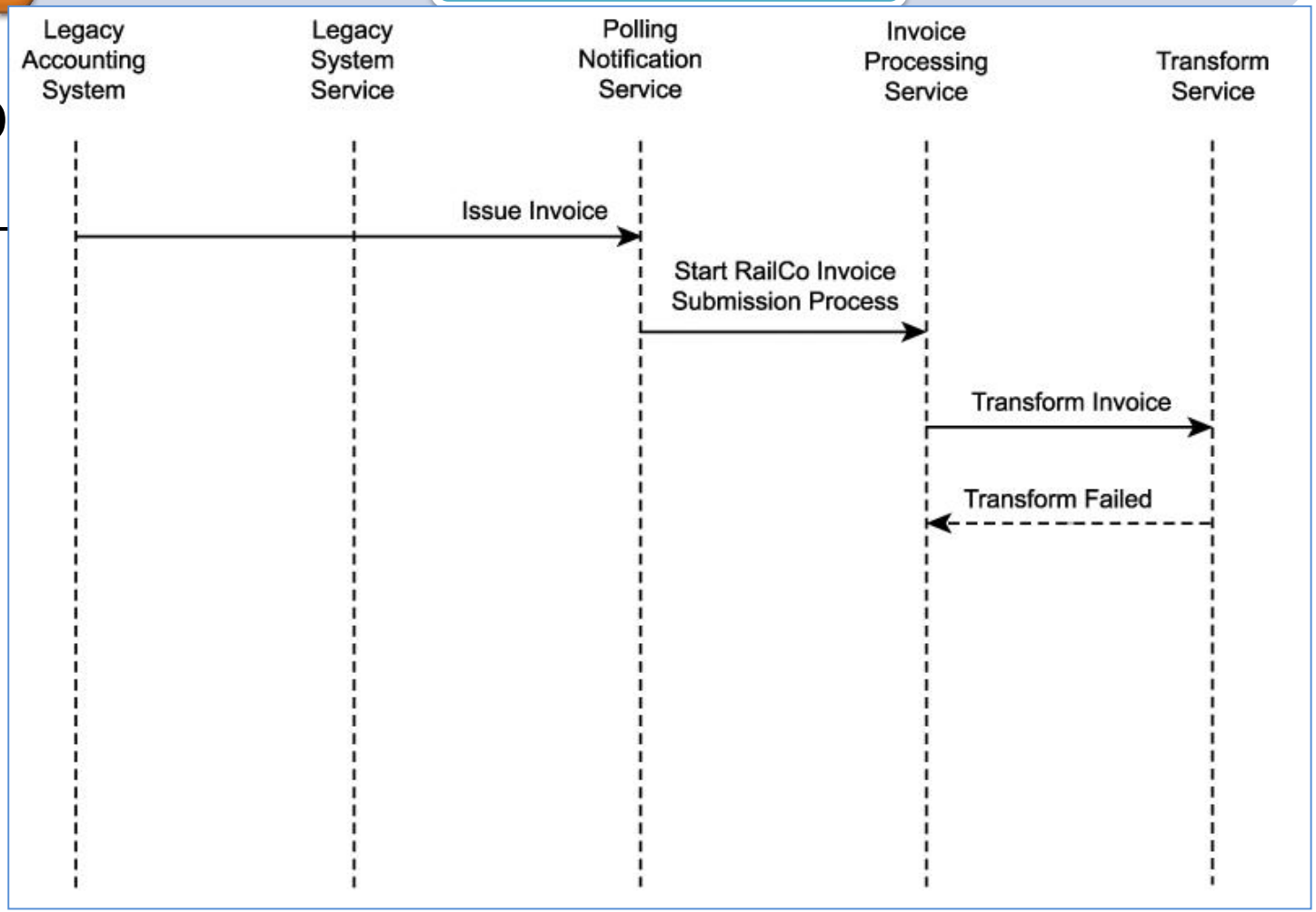
- Diseño
  - Task-Centric Service
    - Definir lógica de flujo
      - Diagramas de secuencia
    - Interfaz inicial
    - Aplicar orientación a servicios
      - Reusabilidad (no aplicable, generalmente)
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz



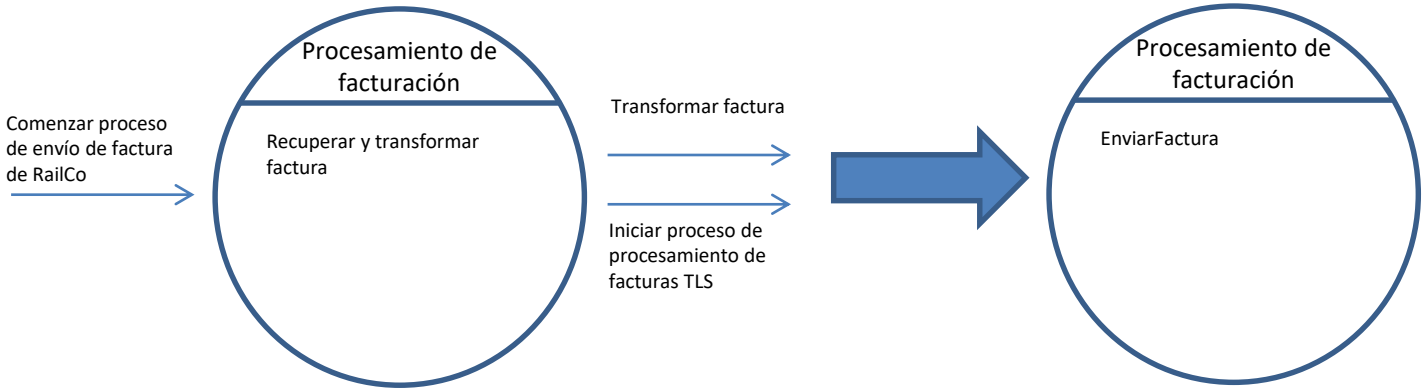
Diseño - Caso de estudio


Legacy Accounting System
Legacy System Service
Polling Notification Service
Invoice Processing Service
Transform Service
Metadata Checking Service
TLS B2B Solution
Issue Invoice
Start RailCo Invoice Submission Process
Transform Invoice
Transform Succeeded
Check TLS Metadata
Metadata Check Succesful
Start TLS Invoice Processing Process

- D


Legacy Accounting System
Legacy System Service
Polling Notification Service
Invoice Processing Service
Transform Service
Issue Invoice
Start RailCo Invoice Submission Process
Transform Invoice
Transform Failed

- Diseño
  - Task-Centric Service
    - Definir lógica de flujo
    - Interfaz inicial



- Diseño
  - Task-Centric Service
    - Definir lógica de flujo
    - Interfaz inicial
    - Aplicar orientación a servicios
      - Reusabilidad (no aplicable generalmente)
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz

```
<types>
   <xsd:schema targetNamespace=
      "http://www.xmltc.com/railco/invoiceservice/schema/">
      <xsd:element name="SubmitInvoiceType">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="ContextID"
                  type="xsd:integer"/>
               <xsd:element name="InvoiceLocation"
                  type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:schema>
</types>
```

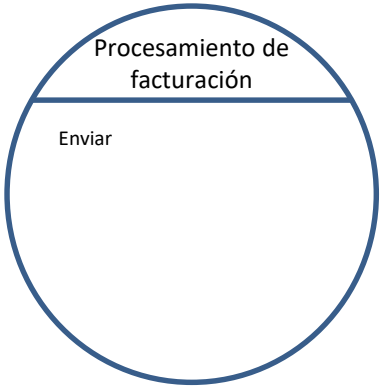
```
<message name="receiveSubmitInvoiceMessage">
   <part name="RequestParameter"
      element="invs:SubmitInvoiceType"/>
</message>
<portType name="InvoiceProcessingInterface">
   <operation name="SubmitInvoice">
      <input message="tns:receiveSubmitInvoiceMessage"/>
   </operation>
</portType>
```

- Diseño
  - Task-Centric Service
    - Definir lógica de flujo
    - Interfaz inicial
    - Aplicar orientación a servicios
      - Reusabilidad (no aplicable generalmente)
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz

```
<portType name="InvoiceProcessingInterface">
   <documentation>
      Initiates the Invoice Submission Process.
   </documentation>
   <operation name="SubmitInvoice">
      <input message="tns:receiveSubmitInvoiceMessage"/>
   </operation>
</portType>
```

- Diseño
  -

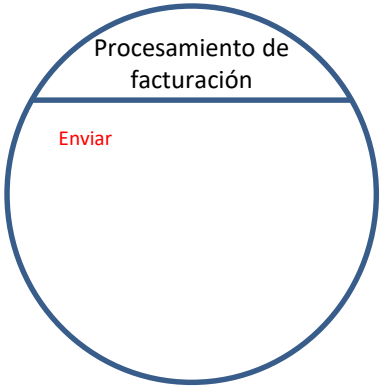
```
<types>
   <xsd:schema targetNamespace=
      "http://www.xmltc.com/railco/invoiceservice/schema/">
      <xsd:import namespace=
         "http://www.xmltc.com/railco/invoice/schema/"
         schemaLocation="Invoice.xsd"/>
      <xsd:element name="SubmitInvoiceType">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="ContextID"
                  type="xsd:integer"/>
               <xsd:element name="InvoiceLocation"
                  type="xsd:string"/>
               <xsd:element name="InvoiceDocument"
                  type="inv:InvoiceType"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:schema>
</types>
```




```
<documentation>
   Initiates the Invoice Submission Process.
   Requires either the invoice document location
   or the document.
</documentation>
```

- Diseño
  - Task-Centric Service
    - Definir lógica de flujo
    - Interfaz inicial
    - Aplicar orientación a servicios
      - Reusabilidad (no aplicable generalmente)
      - Autonomía
      - Sin estado
      - Descubrimiento
    - Estandarización de la interfaz

```
<message name="receiveSubmitMessage">
   <part name="RequestParameter"
      element="invs:SubmitInvoiceType"/>
</message>
<portType name="InvoiceProcessingInterface">
   <documentation>
      Initiates the Invoice Submission Process.
      Requires either the invoice document location
      or the document.
   </documentation>
   <operation name="Submit">
      <input message="tns:receiveSubmitMessage"/>
   </operation>
</portType>
```

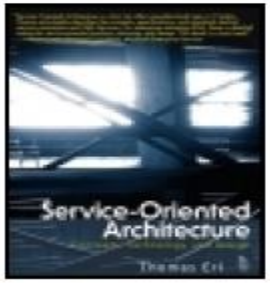


- Diseño
  - Business Process Service
    - Definir los escenarios de actuación
      - Diagramas de secuencia
      - Similar al diseño de task-centric
    - Interfaz inicial
      - WSDL
    - Formalizar la interacción de servicios asociados
    - Definir lógica del proceso
    - Alinear escenarios de actuación y refinar la definición de procesos

```
<plnk:partnerLinkType name="TimesheetSubmissionType">
   <plnk:role name="TimesheetSubmissionService">
      <plnk:portType
         name="tns:TimesheetSubmissionInterface"/>
   </plnk:role>
</plnk:partnerLinkType>
```

- Diseño
  - Business Process Service
    - Definir los escenarios de actuación
    - Interfaz inicial
    - Formalizar la interacción de servicios asociados
      - Definir los servicios asociados (partners) participantes en el proceso y los roles asociados
      - Añadir el tipo de socio (partnerLinkType) al final de la WSDL de cada socio
      - Crear el partnerLink para cada socio en la definición del proceso
      - Definir las variables de entrada y salida para representar el intercambio de mensajes
    - Definir lógica del proceso
    - Alinear escenarios de actuación y refinar la definición de procesos

- Diseño
  - Business Process Service
    - Definir los escenarios de actuación
    - Interfaz inicial
    - Formalizar la interacción de servicios asociados
    - Definir lógica del proceso
      - Usar elementos estructurales de BPEL
    - Alinear escenarios de actuación y refinar la definición de procesos



Service-Oriented Architecture: Concepts, Technology, and Design

By Thomas Erl

.....

Publisher: **Prentice Hall PTR**
Pub Date: **August 04, 2005**
ISBN: **0-13-185858-0**
Pages: **792**

Bibliografía