Arquitectura Orientada a Servicios y su impacto en el desarrollo de aplicaciones

Tesina de Grado: **Román Rollié**
Director: **Lic. Patricia Bazán**

Temario

- Introducción
- Evolución de las arquitecturas distribuidas
- Arquitectura Orientada a Servicios
- Servicios Web
- BMP (*Business Process Management*)
- Sistema de Ventanilla Única
- Conclusión
- Líneas de trabajo futuras

Introducción

- Análisis de la evolución de las principales tecnologías destinadas a la implementación de arquitecturas orientadas a servicios.
- Desarrollo de los conceptos relacionados al modelo SOA, los Servicios Web como herramienta de implementación y el lenguaje BPEL para la gestión de los procesos de negocios y orquestación de los Servicios Web involucrados en el proceso.
- Aplicación práctica en un caso de estudio para un Sistema de Ventanilla Única.

Introducción

Motivación

La creciente necesidad de las organizaciones por establecer la interacción entre sus sistemas *legacy* y aprovechar los activos de software existentes concebidos como servicios.

Temario

- Introducción
- Evolución de las arquitecturas distribuidas
- Arquitectura Orientada a Servicios
- Servicios Web
- BMP (*Business Process Management*)
- Sistema de Ventanilla Única
- Conclusión
- Líneas de trabajo futuras

Evolución de las arquitecturas distribuidas

¿En que consiste?

- Orientación a un diseño cada vez más débilmente acoplado a la arquitectura subyacente.
- Distribución de las tareas a realizar entre varios computadores.
- Lograr independencia de la plataforma.
- Alcanzar un menor costo de desarrollo.

Soluciones alcanzables por algunos paradigmas:

- CORBA / DCOM
- RMI
- RPC

Temario

- Introducción
- Evolución de las arquitecturas distribuidas
- Arquitectura Orientada a Servicios
- Servicios Web
- BMP (*Business Process Management*)
- Sistema de Ventanilla Única
- Conclusión
- Líneas de trabajo futuras

Arquitectura Orientada a Servicios

SOA es una metodología cuyo propósito es establecer la composición de servicios de software, entre empresas, PC, dispositivos móviles, etc., con el objetivo de lograr la reusabilidad de las aplicaciones existentes.

- **Reusalibilidad.**
- **Modularidad.**

Arquitectura Orientada a Servicios
Beneficios de SOA

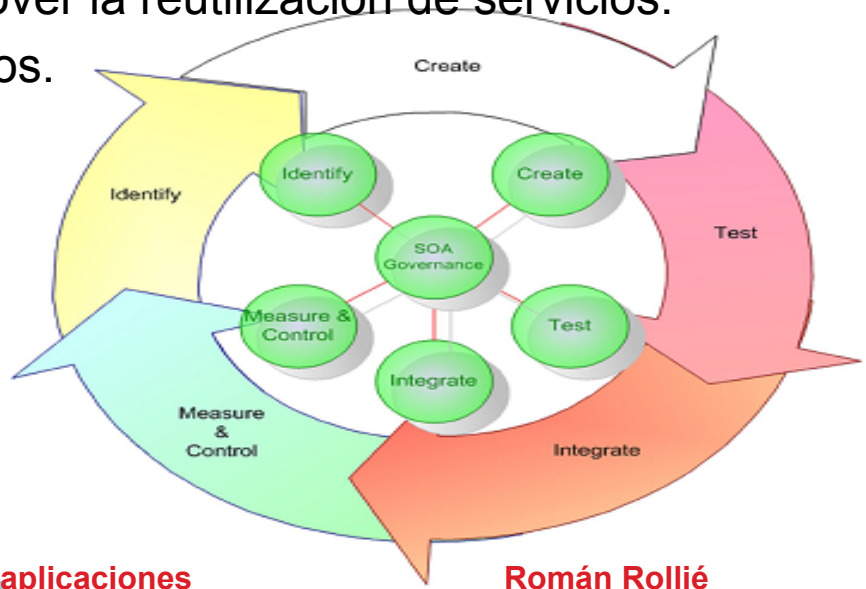
- Reutilización de servicios en múltiples aplicaciones.
- Creación de nuevos servicios de manera rápida y sencilla a partir de servicios existentes.
- Abstracción del entorno de ejecución, concentrándonos en el desarrollo del servicio.
- División de tareas, asignando responsabilidades particulares a cada grupo de desarrollo.

Arquitectura Orientada a Servicios

Modelo de desarrollo

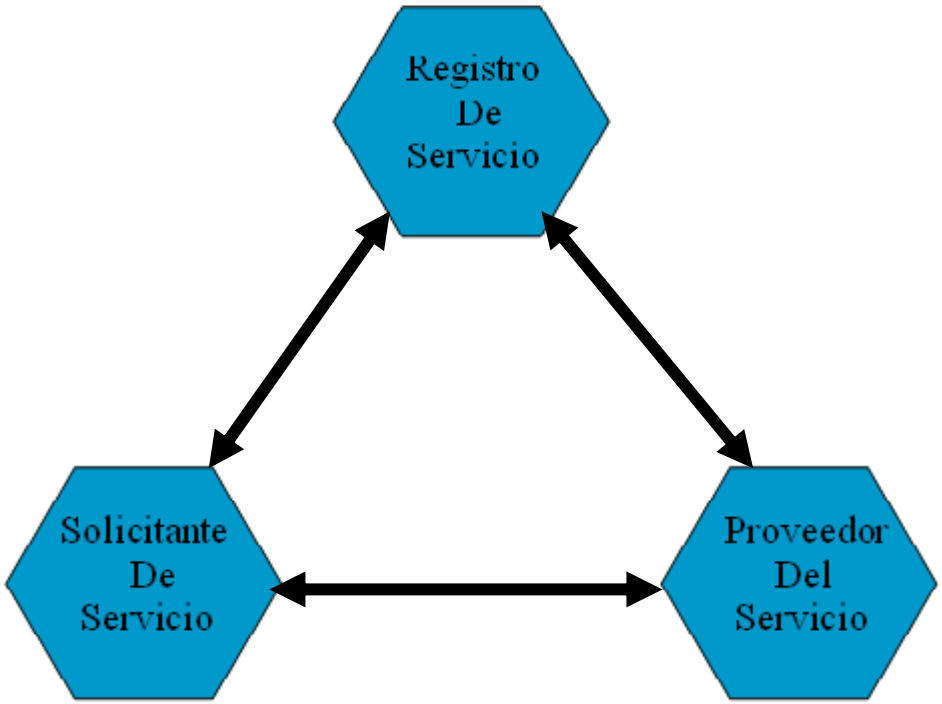
El desarrollo de un modelo SOA respetará un ciclo de vida en el cual algunos de los aspectos de control serán:

- Determinar como el proyecto SOA será puesto en producción, controlado y consumido desde otras aplicaciones.
- Desarrollar e implementar las prácticas correctas para alcanzar la interoperabilidad.
- Asegurar el crecimiento del proyecto y promover la reutilización de servicios.
- Definir la granularidad correcta de los servicios.



Arquitectura Orientada a Servicios

Modelo Operacional



SOA sigue un paradigma de búsqueda, enlace y ejecución.

- Quienes proveen servicios los publican en registros públicos.
- Los registros públicos son utilizados por quienes buscan algún servicio.
- El registro le devolverá a quien consulte por un servicio su interfaz y ubicación.

Temario

- Introducción
- Evolución de las arquitecturas distribuidas
- Arquitectura Orientada a Servicios
- Servicios Web
- BMP (*Business Process Management*)
- Sistema de Ventanilla Única
- Conclusión
- Líneas de trabajo futuras

Servicios Web

Los **Servicios Web** son básicamente componentes de software que pueden ser accedidos a través de Internet por protocolos que forman parte de un estándar creado para tal fin. Los mismos, deben ser lo suficientemente flexibles como para hacer de forma casi transparente el cambio de servidor de ejecución sin tener que modificar demasiados aspectos.

Independencia del lenguaje de programación y de la plataforma. Permiten abstraernos de las capas subyacentes.

Interoperabilidad por medio de la utilización de estándares para la comunicación: XML, SOAP, WSDL, UDDI.

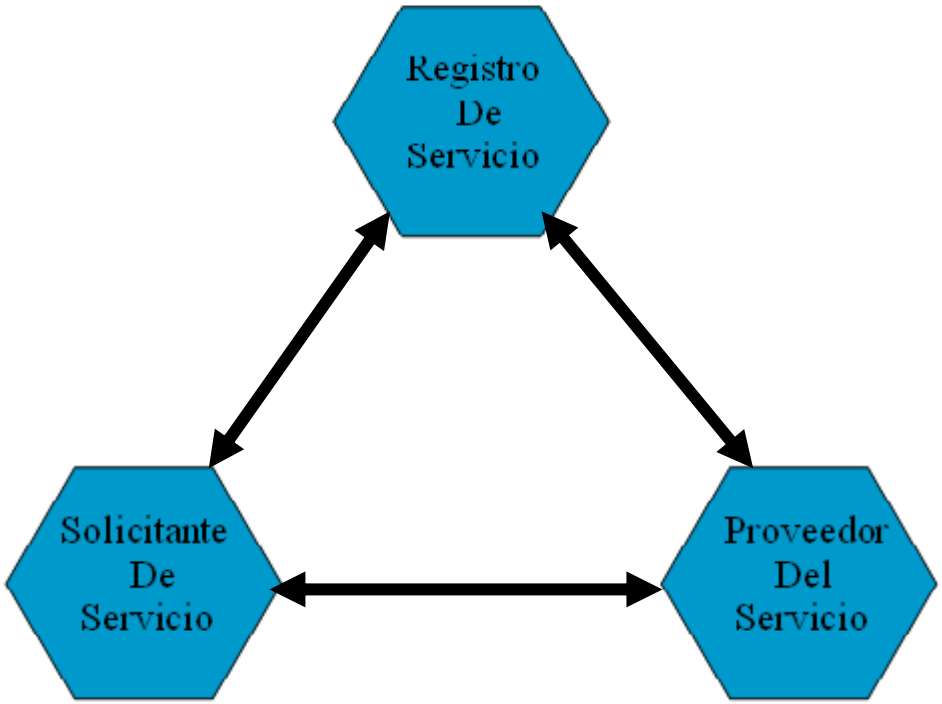
Débil acoplamiento, la interacción entre los sistemas se hace a través de Internet por medio de mensajes (interacciones síncronas y asíncronas), sin control centralizado.

Modularidad y reusabilidad el desarrollo SOA facilita la reutilización de los servicios de negocio. las funciones de negocio pueden ser expuestas como Servicios Web y ser reutilizadas para cubrir nuevas necesidades de negocio.

Escalabilidad las aplicaciones que usan Servicios Web escalan fácilmente debido a que se encuentran débilmente acoplados.

Servicios Web

Modelo operacional basado en Servicios Web



UDDI centraliza los servicios en un registro común (*publish* / *find*)

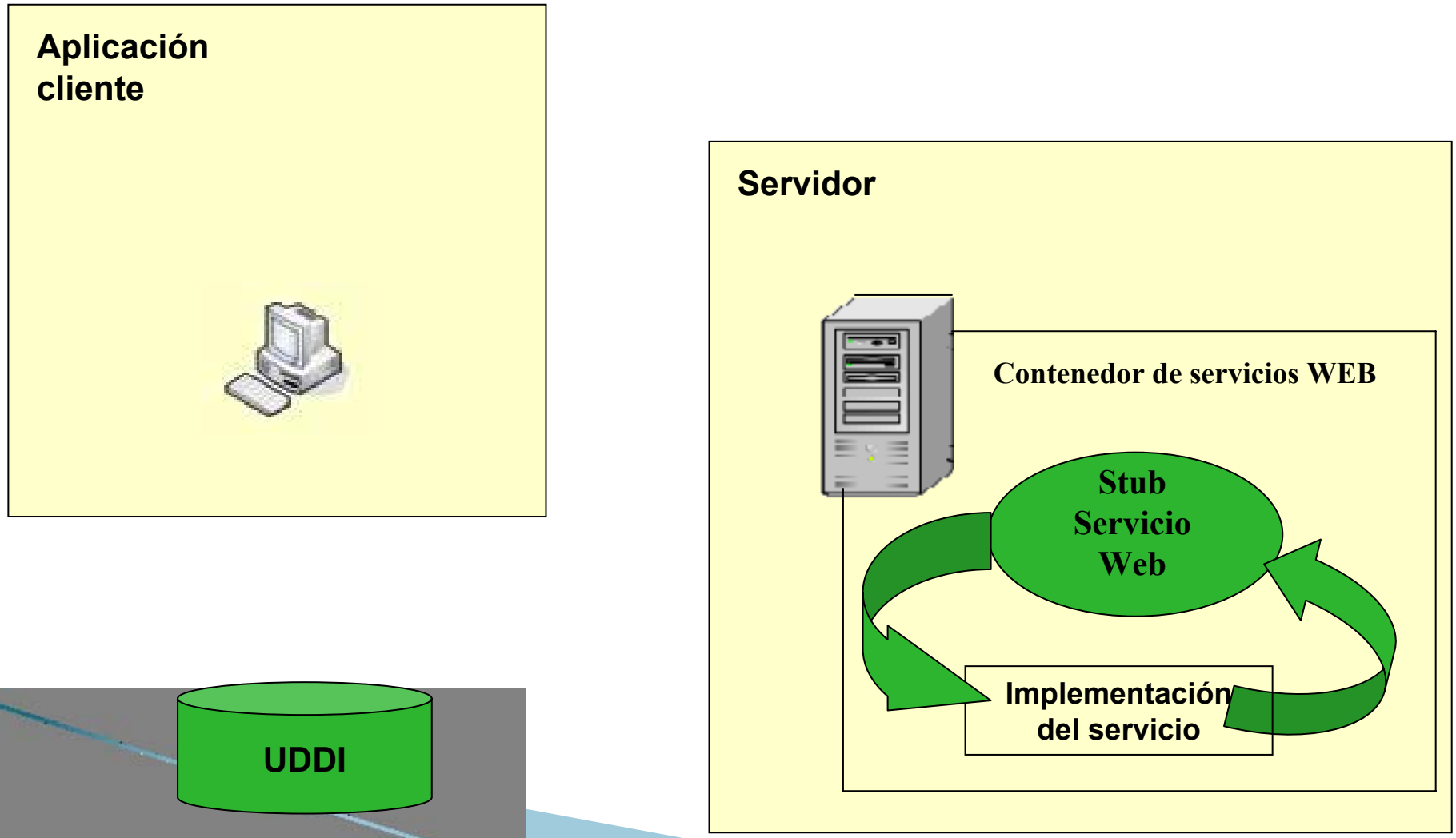
WSDL define la interfaz de los Servicios Web

SOAP codifica los mensajes en un formato común de intercambio

HTTP, SMTP, FTP son algunos de los protocolos de transporte posibles

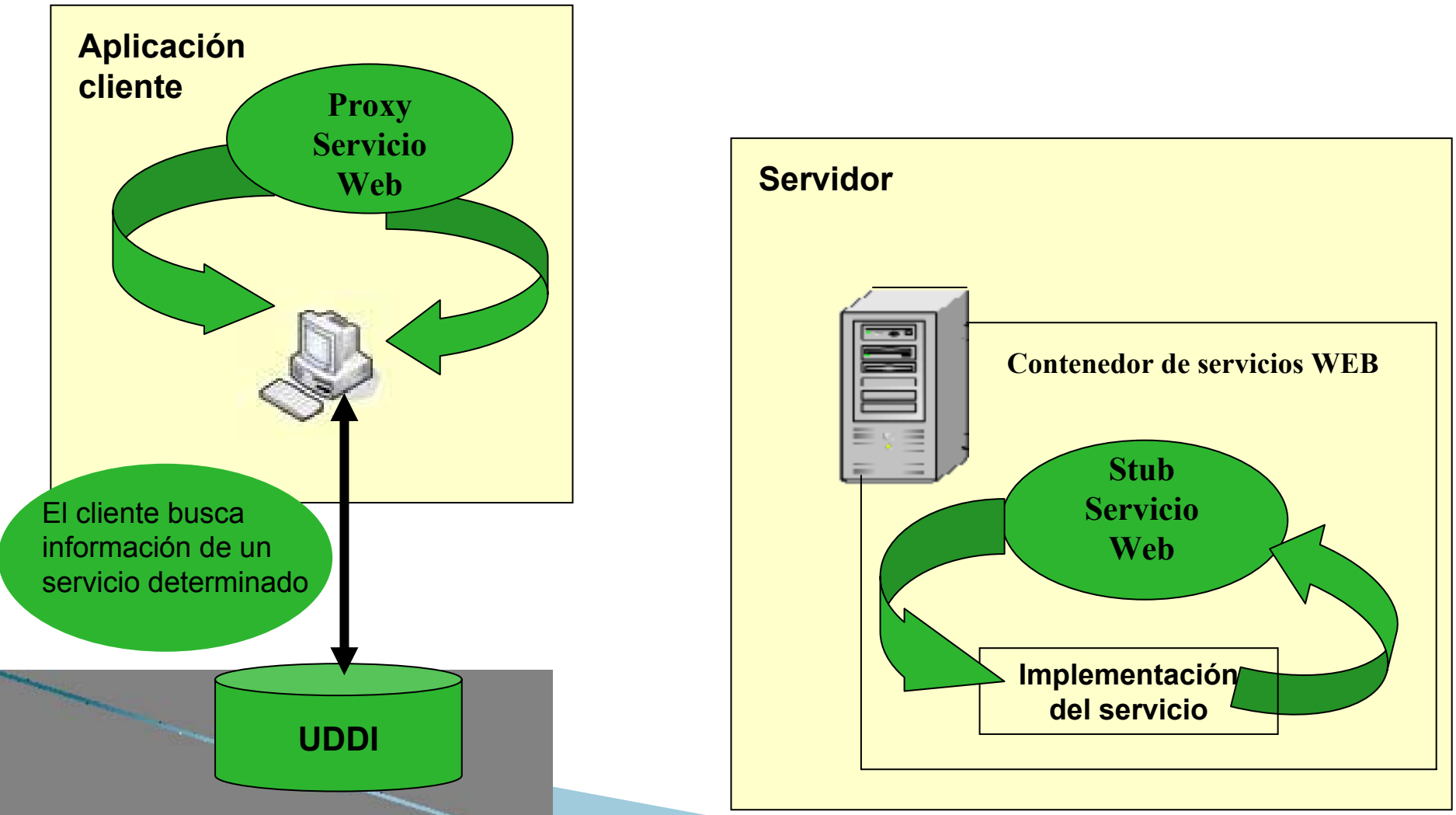
Servicios Web

Invocación de un Servicio Web



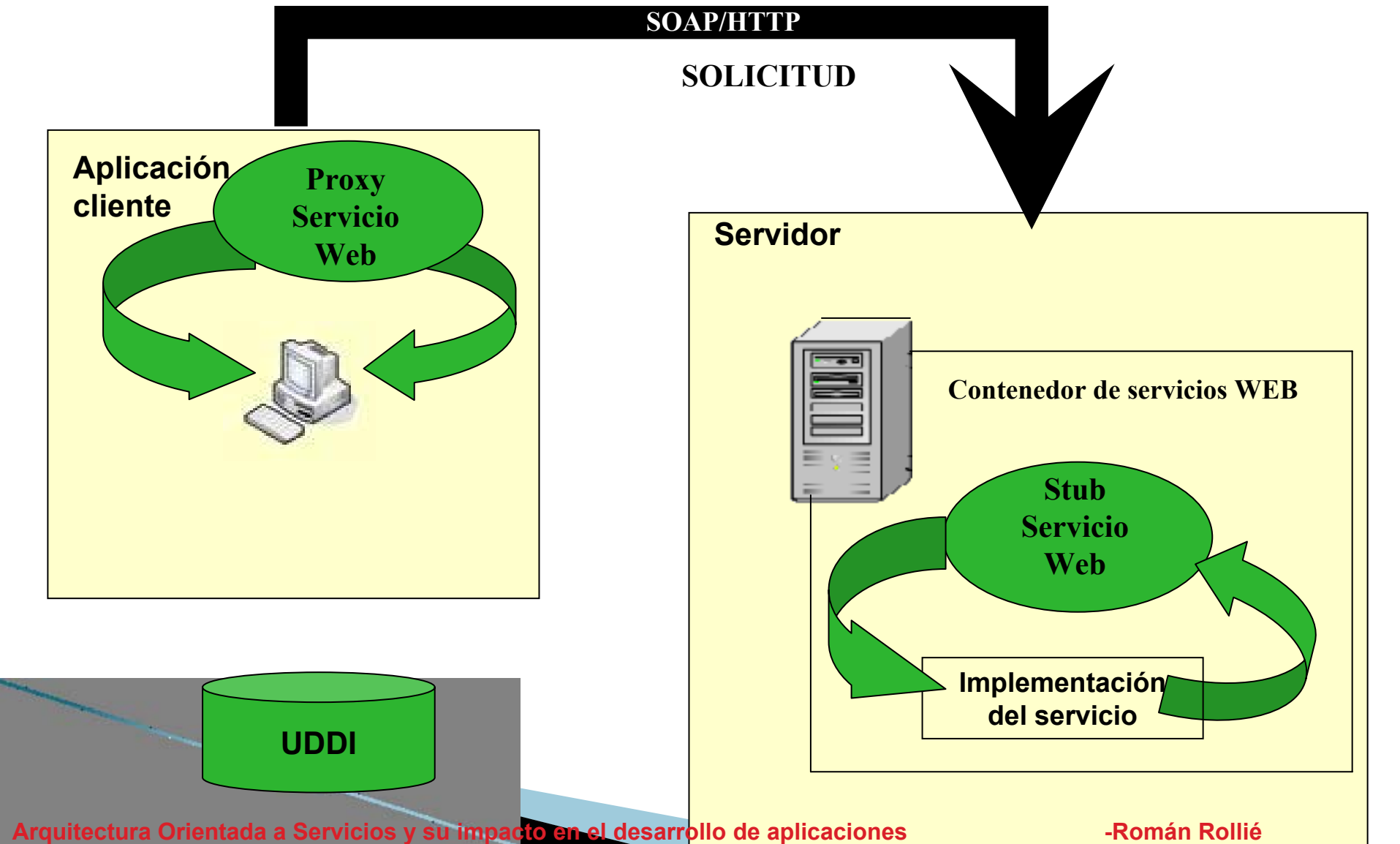
Servicios Web

Invocación de un Servicio Web (2)



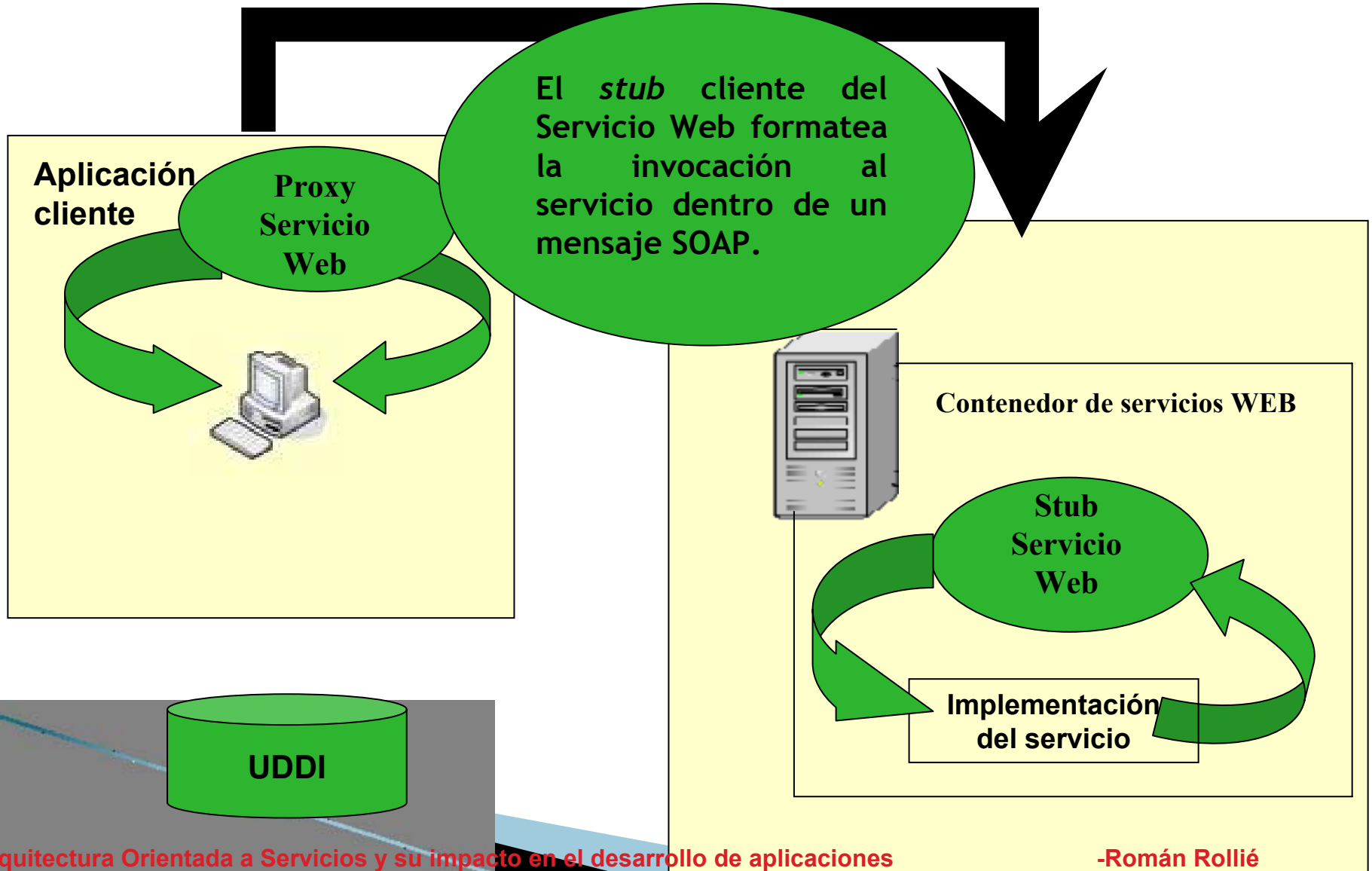
Servicios Web

Invocación de un Servicio Web(3)



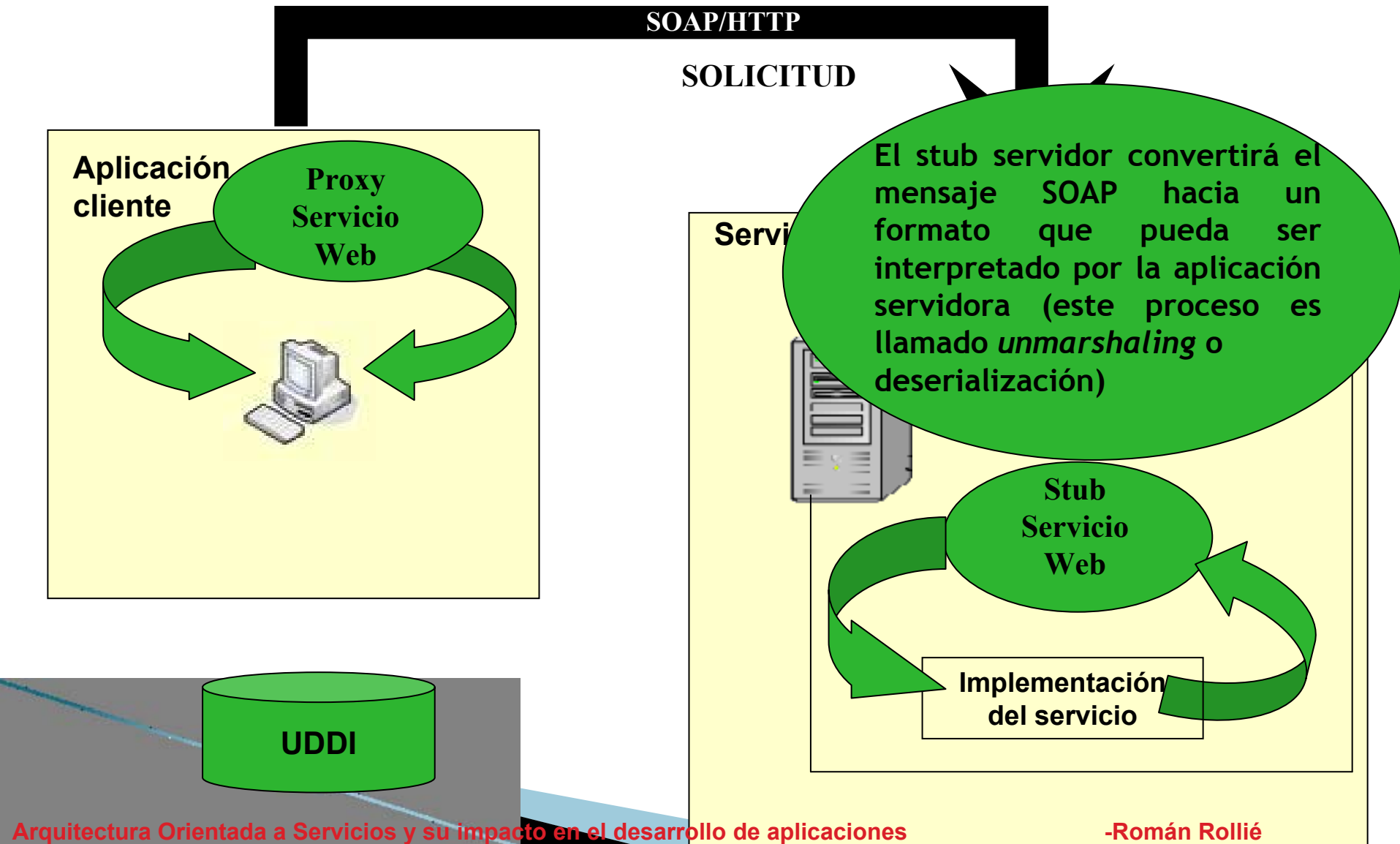
Servicios Web

Invocación de un Servicio Web(4)



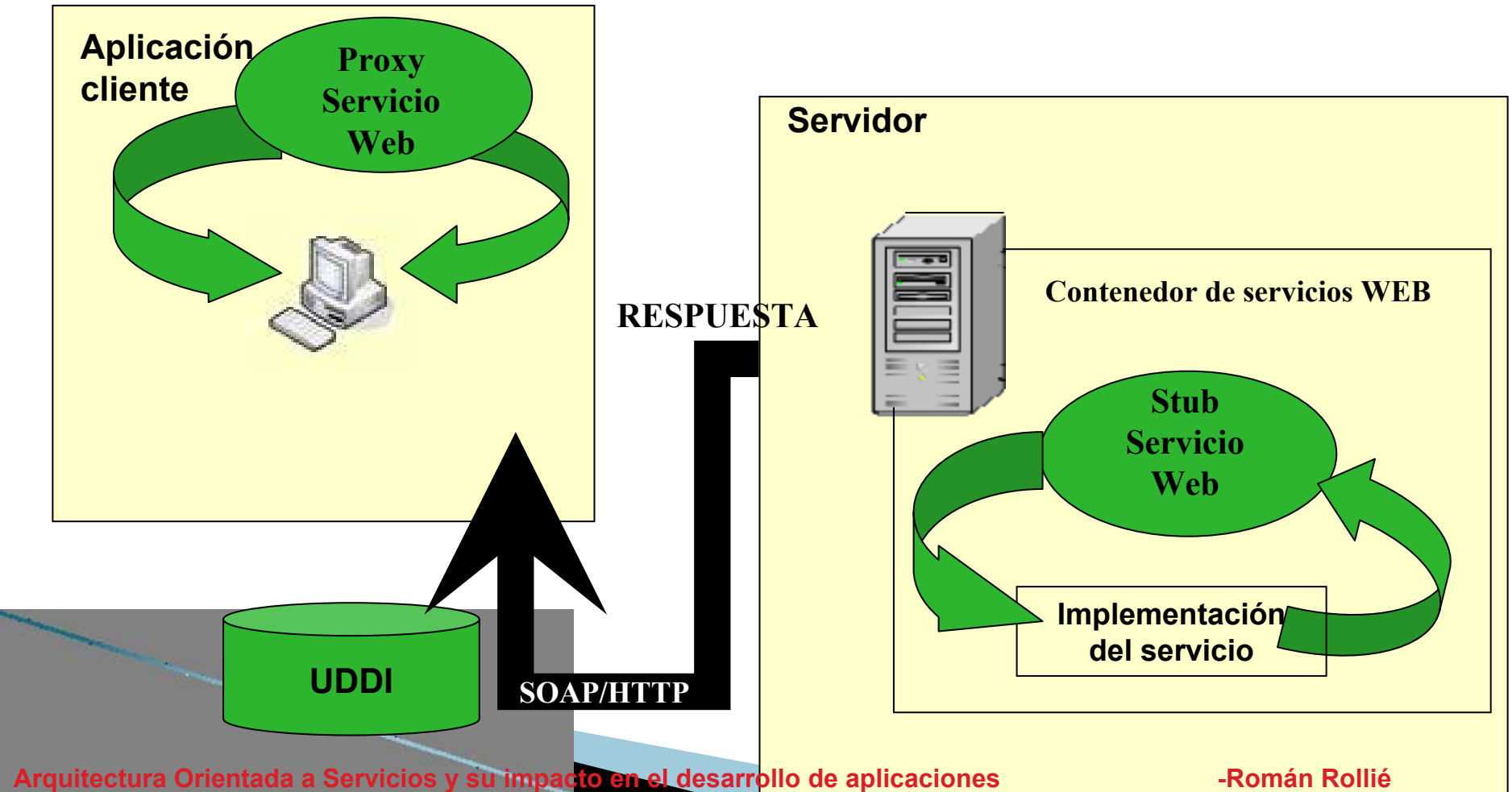
Servicios Web

Invocación de un Servicio Web(5)



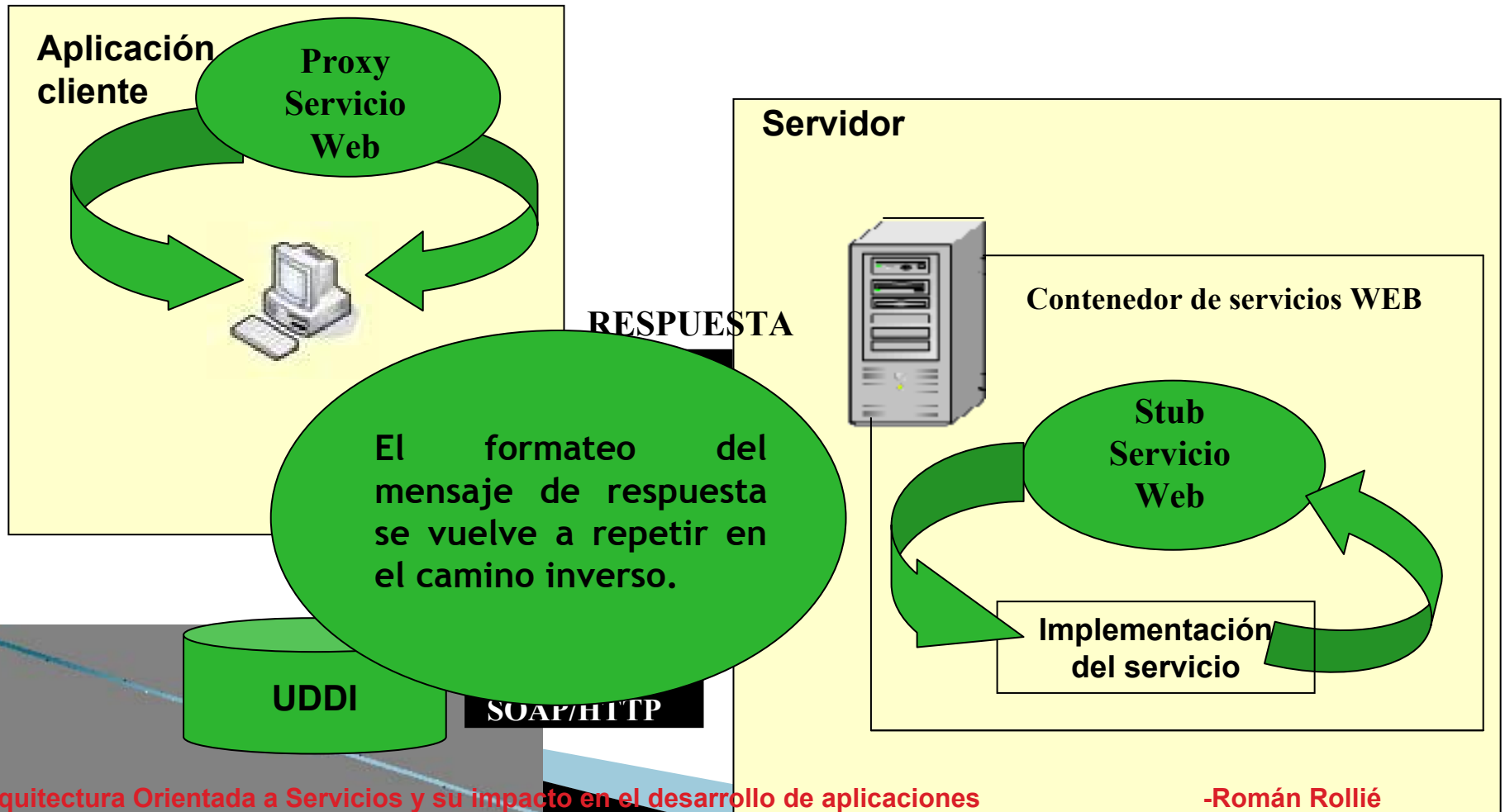
Servicios Web

Invocación de un Servicio Web(6)



Servicios Web

Invocación de un Servicio Web(7)



Servicios Web

Tecnología utilizada

- Lenguaje de programación JAVA.
- API JAX-WS para la creación de Servicios Web sobre JAVA.
- Servidor de aplicaciones JBOSS 2.3.

Servicios Web

Generación de Servicios Web con la API JAX-WS

1. Definición de encabezado.
2. Protocolo comunicación a utilizar.
3. Métodos a exponer.
4. Generación de proxy cliente.

Servicios Web

Generación sobre clase JAVA

```
@SOAPBinding(parameterStyle=ParameterStyle.WRAPPED, style=Style.DOCUMENT,use=Use.ENCODED)
public class ObtencionInformeTerritorial{

     public ObtencionInformeTerritorial(){}

    @WebMethod(operationName="recuperarInforme")
    @WebResult(name="RespuestaInformeTerritorial")
      public InformeTerritorial getInformeTerritorial(String nroPartida){
      InformeTerritorial informeTerritorial = new InformeTerritorial();

      HibernateInformeTerritorialDAO hitDAO = new HibernateInformeTerritorialDAO();
      informeTerritorial =hitDAO.getInformeTerritorial(nroPartida);
      return      informeTerritorial;
   }
```

La anotación @WebService indicará que la clase JAVA definida implementa un Servicio Web.

Atributos

- *name* define el nombre *de wsdl:portType*.
- *portName* determina la definición del *wsdl:portName*.
- serviceName especifica el nombre del Servicio Web: *wsdl:service*.
- wsdlLocation especifica la dirección Web del documento WSDL que define el Servicio Web.

Servicios Web

Generación sobre clase JAVA(2)

```
@WebService(name="ObtencionInformeTerritorial",portName="ObtencionInformeTerritorial",
serviceName="InformeTerritorial",wsdlLocation="WEB-INF/wsdl/InformeTerritorial.wsdl")

public class ObtencionInformeTerritorial{

     public ObtencionInformeTerritorial(){}

    @WebMethod(operationName="recuperarInforme")
    @WebResult(name="RespuestaInformeTerritorial")
      public InformeTerritorial getInformeTerritorial(String nroPartida){
      InformeTerritorial informeTerritorial = new InformeTerritorial();

      HibernateInformeTerritorialDAO hitDAO = new HibernateInformeTerritorialDAO();
      informeTerritorial =hitDAO.getInformeTerritorial(nroPartida);
      return      informeTerritorial;
   }
```

La anotación @*SOAPBinding* especifica el protocolo de mensajes SOAP utilizado en la comunicación con el Servicio Web.

Atributos

- *style* define el estilo de codificación para los mensajes enviados a y desde el Servicio Web.
- *use* define el formato utilizado para los mensajes enviados a y desde el Servicio Web.
- *parameterStyle* determina si los parámetros del método representan todo el cuerpo del mensaje o si los parámetros son elementos envueltos en el elemento.

Servicios Web

Generación sobre clase JAVA(3)

```
@WebService(name="ObtencionInformeTerritorial",portName="ObtencionInformeTerritorial",
serviceName="InformeTerritorial",wsdlLocation="WEB-INF/wsdl/InformeTerritorial.wsdl")

@SOAPBinding(parameterStyle=ParameterStyle.WRAPPED, style=Style.DOCUMENT,use=Use.ENCODED)
public class ObtencionInformeTerritorial{

    public ObtencionInformeTerritorial(){}

    public [illegible]eTerritorial(String nroPartida){
    InformeTerritorial informeTerritorial = new InformeTerritorial();

    HibernateInformeTerritorialDAO hitDAO = new HibernateInformeTerritorialDAO();
    informeTerritorial =hitDAO.getInformeTerritorial(nroPartida);
    return      informeTerritorial;
  }
```

La anotación @*WebMethod* permite poder exponer un método de una clase java como una operación del Servicio Web.

Atributo

operationName especifica el nombre de wsdl:operation que coincide con este método.

La anotación @*WebResult* personaliza la correlación de un valor de retorno con una parte WSDL o elemento XML.

Atributo

name especifica el nombre del valor de retorno como figura en el archivo WSDL y aparece en los mensajes de la comunicación.

Servicios Web

Generación del proxy cliente

Para poder realizar la generación del *proxy* cliente, se deberá tener información acerca de la ubicación del documento WSDL que define al Servicio.

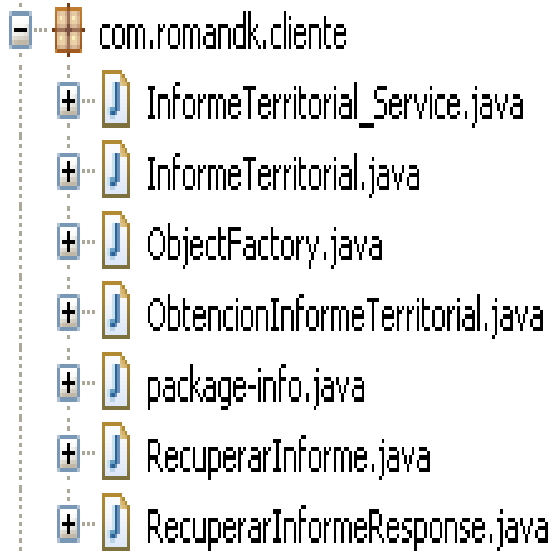
Por medio de la instrucción *wsimport* de la API JAX-WS se generarán las clases representantes del servicio y aquellas que llevaran a cabo la conversiones entre objeto Java y documento XML.

Servicios Web

Generación del proxy cliente(2)

JAX-WS: WSIMPORT

```
<target name="wsimport" depends="init">
   <wsdljava wsdl="http://localhost:8080/InformeTerritorial/?wsdl" package="com.romandk.cliente"
   destdir="c:/proxy" keep="true" verbose="true"/>
</target>
```



Servicios Web

Generación del proxy cliente(3)

Tags principales del documento XML en formato WSDL:

- ***<types>*** Define los tipos de datos utilizados por el Servicio Web.
- ***<message>*** Mensajes que conocerá el Servicio Web.
- ***<portType>*** Operaciones ejecutadas por el Servicio Web.
- ***<binding>*** Detalles sobre los protocolos de comunicación utilizados por cada *port*.
- ***<service>*** Define la ubicación en la cual invocar al servicio.

Temario

- Introducción
- Evolución de las arquitecturas distribuidas
- Arquitectura Orientada a Servicios
- Servicios Web
- BMP (*Business Process Management*)
- Sistema de Ventanilla Única
- Conclusión
- Líneas de trabajo futuras

BPM (Business Process Management)

Un proceso de negocios consta de un conjunto de tareas lógicamente relacionadas que cuando son ejecutadas en un orden apropiado y siguiendo alguna regla de negocio producen una salida de negocio.

Características BPM:

- Reducción de las posibles diferencias entre los requerimientos del negocio y los sistemas IT.
- Aumento en la productividad y disminución de los costos operacionales debido a la automatización de los procesos de negocio.
- Aumento en la flexibilidad y agilidad de la corporación gracias a la separación de la lógica de procesos de negocios.
- Reduce los costos de desarrollo y esfuerzo mediante el uso de herramientas de programación gráfica.

BPM (Business Process Management)

Ciclo de vida BPM

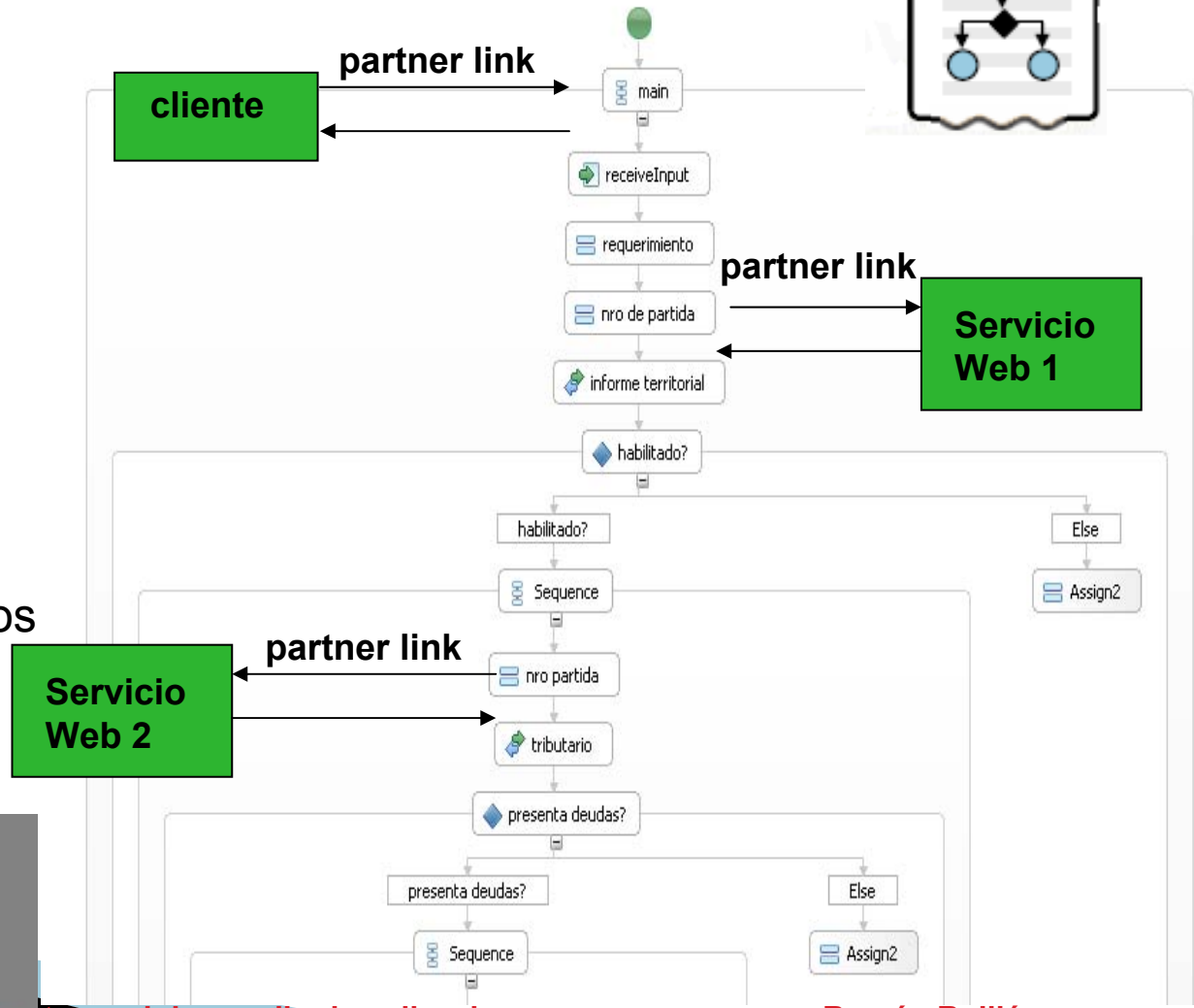
- **Modelado:** Un BPM consta de herramientas visuales que nos permiten diagramar el flujo de comunicación entre los procesos de negocios y tener una visión más clara de la lógica que lleva a cabo cada proceso.
- **Integración:** La integración de los procesos de negocios, se puede llevar a cabo por medio del lenguaje estándar para la ejecución de procesos de negocios, BPEL.
- **Ejecución:** Cada BPM consta de un motor de BPM el cual es capaz de ejecutar los procesos de negocios.
- **Control:** La ejecución de cada proceso de negocio puede ser monitorizada por lo que es posible obtener métricas de estos procesos para poder ser analizadas y comparadas.

BPM (Business Process Management)

Definición de un proceso BPEL

¿ Por qué utilizar BPEL ?

- Nos permite trabajar sobre la lógica de procesos.
- Invocaciones sincronas y asíncronas de Servicios Web.
- Los programas se definen sobre XML.
- Orquesta las invocaciones a los Servicios Web.



BPM (Business Process Management)

Tecnologías empleadas

- JBPM motor de ejecución de procesos BPM.
- Eclipse BPEL designer. Modelador gráfico de procesos de negocios.

BPM (Business Process Management)

Composición de un proceso BPEL

```
<bpel:import namespace="http://service.romandk.com/" location=
"http://127.0.0.1:8080/Servicio-Web-C?wsdl" importType="http://schemas.xmlsoap.org/wsdl/"
></bpel:import>
    <bpel:import namespace="http://service.romandk.com/" location=
"http://127.0.0.1:8080/StoredProcedureService?wsdl" importType="http://schemas.xmlsoap.org/wsdl/"
></bpel:import>
    <bpel:import namespace="http://servicio.romandk.com/" location=
"http://127.0.0.1:8080/InformeTributario?wsdl" importType="http://schemas.xmlsoap.org/wsdl/"
></bpel:import>
    <bpel:import namespace="http://servicio.romandk.com/" location=
"http://127.0.0.1:8080/InformeTerritorial?wsdl" importType="http://schemas.xmlsoap.org/wsdl/"
></bpel:import>
    <bpel:import location="orquestacionArtifacts.wsdl" namespace="http://eclipse.org/bpel/sample"
        importType="http://schemas.xmlsoap.org/wsdl/"/>

    <bpel:partnerLinks>
        <bpel:partnerLink name="client"partnerLinkType="tns:orquestacion" myRole=
"orquestacionProvider"/>
        <bpel:partnerLink name="territorial" partnerLinkType="tns:territorialPLT" myRole=
"territorialServer" partnerRole="territorialClient"></bpel:partnerLink>
        <bpel:partnerLink name="tributario" partnerLinkType="tns:tributarioPLT" myRole=
"tributarioServer" partnerRole="tributarioClient"></bpel:partnerLink>
        <bpel:partnerLink name="caratulacion" partnerLinkType="tns:caratularPLT" myRole=
"caratularServer" partnerRole="caratularClient"></bpel:partnerLink>
        <bpel:partnerLink name="categoria" partnerLinkType="tns:categoriaPLT" myRole=
"categoriaServer" partnerRole="categoriaClient"></bpel:partnerLink>
  </bpel:partnerLinks>
```

import

Lista todos los documentos *wsdl* de los servicios web utilizados en el proceso BPEL

PartnerLink

Lista los servicios participantes en el flujo del proceso de negocio, los roles que cumplen y partnerLinkType asociados

BPM (Business Process Management)

Composición de un proceso BPEL(2)

```
<bpel:variables>
  <bpel:variable name="input" messageType="tns:orquestacionRequestMessage"/>
  <bpel:variable name="output" messageType="tns:orquestacionResponseMessage"/>
  <bpel:variable name="territorialIn" messageType=
"ns1:ObtencionInformeTerritorial_getInformeTerritorial"/>
  <bpel:variable name="territorialOut" messageType=
"ns1:ObtencionInformeTerritorial_getInformeTerritorialResponse"/>
    <bpel:variable name="tributarioIn" messageType=
"ns1:informeTributarioService_obtenerInformeTributario"/>
    <bpel:variable name="tributarioOut" messageType=
"ns1:informeTributarioService_obtenerInformeTributarioResponse"/>
    <bpel:variable name="deudasize" type="ns2:int"/>
    <bpel:variable name="indexdeuda" type="ns2:int"/>
    <bpel:variable name="caratulaIn" messageType=
"ns3:StoredProcedureCaratular_caratularTramite"/>
    <bpel:variable name="caratulaOut" messageType=
"ns3:StoredProcedureCaratular_caratularTramiteResponse"/>
    <bpel:variable name="categoriaIn" messageType="ns3:InvocaCC_invocaservicioCC"/>
    <bpel:variable name="categoriaOut" messageType="ns3:InvocaCC_invocaservicioCCResponse"/
</bpel:variables>
```

Variables

Sirven para mantener el estado de los proceso de negocio. permiten almacenar mensajes que mantendrán los valores recibidos y enviados por los servicios. Los tipos de cada variable pueden ser mensajes WSDL, un tipo de dato simple definido en XML o un elemento XML.

BPM (Business Process Management)

Composición de un proceso BPEL(3)

```
<bpel:sequence name="main">
  <bpel:receive name="receiveInput" partnerLink="client" portType="tns:orquestacion"
            operation="process" variable="input" createInstance="yes"/>
   <bpel:assign validate="no" name="requerimiento">
        <bpel:copy>
            <bpel:from>
                <bpel:literal xml:space="preserve"><tns:PruebaTramiteResponse xmlns:tns="http://eclipse.org/bpel/tramite'
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <tns:result></tns:result>
</tns:PruebaTramiteResponse>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="output" part="payload"></bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from part="payload" variable="input">
                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                    <![CDATA[tns:input/tns:arg0]]>
                </bpel:query>
            </bpel:from>
            <bpel:to part="payload" variable="output">
                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                    <![CDATA[tns:result/tns:arg0]]>
                </bpel:query>
            </bpel:to>
        </bpel:copy>
    </bpel:assign>
    <bpel:if name="Estado viabilidad ?">
        <bpel:condition><![CDATA[$output.payload/tns:result/tns:arg0/tns:tramite/tns:estado = 2]]></bpel:condition>
        <bpel:sequence name="viabilidad legal"><bpel:assign validate="no" name="Assign">
        <bpel:else>
    </bpel:if>
     <bpel:reply name="replyOutput" partnerLink="client" portType="tns:orquestacion"
          operation="process" variable="output"/>
</bpel:sequence>
```

Lógica del proceso

Dentro del tag ***<bpel:sequence>*** se define la lógica de proceso de negocios, y la declaración de los receive, invoke, assign, ... de los procesos de negocios

Temario

- Introducción
- Evolución de las arquitecturas distribuidas
- Arquitectura Orientada a Servicios
- Servicios Web
- BMP (*Business Process Management*)
- Sistema de Ventanilla Única
- Conclusión
- Líneas de trabajo futuras

Sistema de ventanilla única

Planteo del problema

El Sistema de Ventanilla Única, consiste en la gestión de trámites de habilitación de actividades económicas comerciales.
El proceso de habilitación de cada uno de estos locales comerciales, consta de una serie de pasos en los cuales el trámite irá pasando por diferentes etapas y tomando diferentes estados definidos por el circuito de negocio.
Los cambios de estado que sufren los documentos, ocurren como resultado de la validación de la información asociada y la información requerida para el traspaso al siguiente estado.

Sistema de ventanilla única

Uso de Servicios Web y BPEL

Para resolver el problema planteado se utilizo una arquitectura orientada a servicios para:

- Integrar los diferentes sistemas *legacy* involucrados en el proceso de habilitación comercial.
- Administrar la lógica de negocios de manera centralizada.

Servicio Ventanilla Única



Fecha: 09/12/2009

Funciones

- tramites pendientes(2)
- Busqueda trámite

Opciones de Visualización

Ordenación por expediente

Mostrar pendientes

Mostrar terminados

tramites tributarios

Inicio Tramite

Nro expediente	Estado	Detalle
23	viabilidad paga	
14	categorizacion	
9	viabilidad paga	
97	categorizacion	
7	categorizacion	
99	categorizacion	

Lista los tramites pendientes de habilitación, de acuerdo al perfil del usuario.

Servicio Ventanilla Única

Fecha: 09/12/2009

Estando en el estado de Viabilidad legal paga, el trámite estará listo para comprobar si se encuentra en condiciones para su habilitación

Funciones

- tramites pendientes(2)
- Busqueda trámite

Opciones de Visualización

Ordenación por expediente

Mostrar pendientes

Mostrar terminados

tramites tributarios

Inicio Tramite

Nro de solicitud de prefactibilidad: 15

Tipo Tramite:

Habilitacion de local para uso comerial

Estado Tramite: viabilidad paga

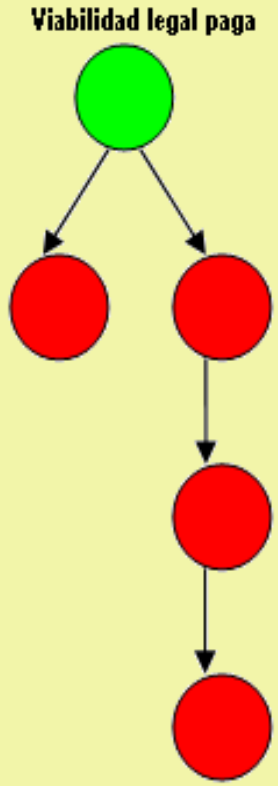
Expediente numero: 9

Solicitante: mauro bocelli

Nro de Partida: 017955

Caratula: sin especificar

enviar formulario



Servicio Ventanilla Única



Fecha: 09/12/2009

Funciones

- tramites pendientes(2)
- Busqueda trámite

Opciones de Visualización

Ordenación por expediente

Mostrar pendientes

Mostrar terminados

Invocación al proceso de negocio

tramites tributarios

Inicio Tramite

Nro de solicitud de prefact

Tipo Tramite: Habilitacion de local para

Estado Tramite: viabilidad pa

Expediente numero: 9

Solicitante: mauro bocelli

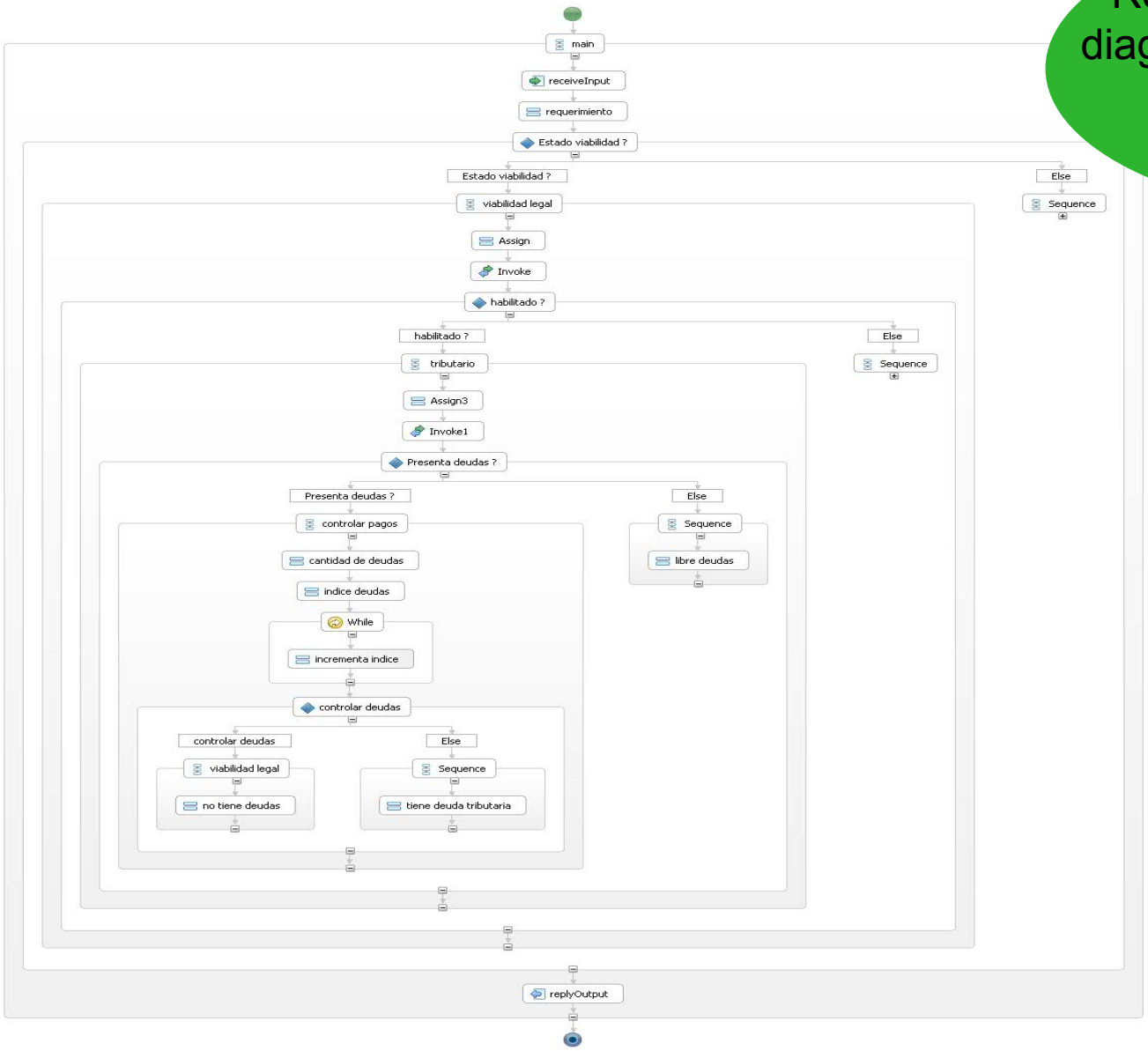
Nro de Partida: 017955

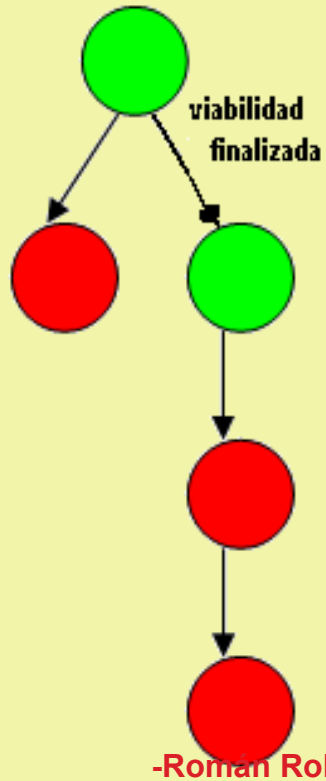
Caratula: sin especifica

enviar formulario

```
OrquestacionService ts = new OrquestacionService();
Orquestacion orquestacion = ts.getOrquestacionProviderPort();
/*   Armo el mensaje */
OrquestacionRequest payload = new OrquestacionRequest();
//Estado tramite
EstadoTramite estadotramite = new EstadoTramite();
//tipo tramite
TipoTramite tipoTramite = new TipoTramite();
//tramite
Tramite tramite = new Tramite();
tramite.setCodigo(cod);
tramite.setEstadoTramite(estadotramite);
tramite.setIdTramite(idtramite);
tramite.setTipoTramite(tipoTramite);
//Requerimiento
Requerimiento requerimiento = new Requerimiento();
requerimiento.setCodigo(nropartida);
requerimiento.setIdRequerimiento(Long.parseLong("1"));
requerimiento.setInforme("");
requerimiento.setTramite(tramite);
//IniciarTramite
IniciarTramite iniciartramite = new IniciarTramite();
iniciartramite.setArg0(requerimiento);
payload.setInput(iniciartramite);
OrquestacionResponse orquestacionResonse = orquestacion.process(payload);
```

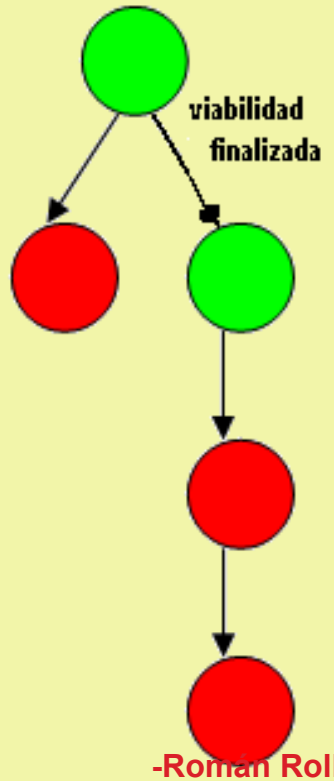
Representación grafica del diagrama BPEL para el estado De viabilidad.





Representación por medio del lenguaje XML

```
<bpel:sequence name="main">
  <bpel:receive name="receiveInput" partnerLink="client"
                portType="tns:orquestacion"
                operation="process" variable="input"
                createInstance="yes"/>
   <bpel:assign validate="no" name="requerimiento">
        <bpel:copy>
            <bpel:from>
                <bpel:literal xml:space="preserve"><tns:PruebaTramiteResponse xmlns:tns=
"http://eclipse.org/bpel/tramite" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <tns:result></tns:result>
</tns:PruebaTramiteResponse>
</bpel:literal>
            </bpel:from>
            <bpel:to variable="output" part="payload"></bpel:to>
        </bpel:copy>
        <bpel:copy>
            <bpel:from part="payload" variable="input">
                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                    <![CDATA[tns:input/tns:arg0]]>
                </bpel:query>
            </bpel:from>
            <bpel:to part="payload" variable="output">
                <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                    <![CDATA[tns:result/tns:arg0]]>
                </bpel:query>
            </bpel:to>
        </bpel:copy>
    </bpel:assign>
    <bpel:if name="Estado viabilidad ?">
        <bpel:condition><![CDATA[$output.payload/tns:result/tns:arg0/tns:tramite/tns:estado =
2]]></bpel:condition>
        <bpel:sequence name="viabilidad legal"><bpel:assign validate="no" name="Assign">
            <bpel:copy>
                <bpel:from part="payload" variable="output">
                    <bpel:query queryLanguage=
"urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0"><![CDATA[tns:result/tns:arg0/tns:codigo]]>
</bpel:query>
                </bpel:from>
                <bpel:to part="getInformeTerritorial" variable="territorialIn">
                    <bpel:query queryLanguage=
"urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
                        <![CDATA[arg0]]>
                    </bpel:query>
                </bpel:to>
            </bpel:copy>
        </bpel:assign>
        <bpel:invoke name="Invoke" partnerLink="territorial" operation=
"getInformeTerritorial" portType="ns1:ObtencionInformeTerritorial" inputVariable="territorialIn"
outputVariable="territorialOut"></bpel:invoke>
```



Temario

- Introducción
- Evolución de las arquitecturas distribuidas
- Arquitectura Orientada a Servicios
- Servicios Web
- BMP (*Business Process Management*)
- Sistema de Ventanilla Única
- Conclusión
- Líneas de trabajo futuras

Conclusión

El potencial alcanzado combinando tecnologías como Servicios Web y BPM en el desarrollo de una arquitectura orientada a servicios (SOA) conforma un aporte en cuanto a:

- Abstracción.
- Rápido desarrollo.
- Definición de procesos de negocios.

Conclusión (2)

- Se realizó un análisis que muestra de que manera el desarrollo de una aplicación SOA mediante Servicios Web y el lenguaje de orquestación BPEL permiten obtener una forma más organizada y eficiente para la integración de sistemas.
- El desarrollo práctico de un caso real, mostró la manera en la cual estas tecnologías interactúan.
- El desarrollo de los temas muestra la gran diversidad de aspectos y tecnologías que involucra una arquitectura orientada a servicios.

Temario

- Introducción
- Evolución de las arquitecturas distribuidas
- Arquitectura Orientada a Servicios
- Servicios Web
- BMP (*Business Process Management*)
- Sistema de Ventanilla Única
- Conclusión
- Líneas de trabajo futuras

Líneas de trabajo futuras

- Análisis de herramientas BPMN y su mapeo hacia un proceso de negocios sobre el lenguaje BPEL.
- Estándares de seguridad propuestos por W3C y OASIS.
- Transacciones distribuidas entre Servicios Web.