ACTA TECHNICA CORVINIENSIS
BULLETIN OF ENGINEERING

1.2. ANA DANIELA CRISTEA, 1. Ovidiu GELU TIRIAN

WEB SERVICES WITH APPLICATION SERVER ABAP

Abstract:
The Application Server ABAP (AS ABAP) is part of the application layer that belongs to the SAP NetWeaver platform. By using it, we have not only the possibility to create Web Services, but also to easily consume the Web Service created with other technologies. The purpose of the present paper is to present either the way we can create a Web Service by using AS ABAP, or the way we can consume a Web Service in the Web Dynpro ABAP. In this respect, we create a Web Service (inside-out type) that has a Function Module as end point. Then, we use the new SOA Manager application to manage, configure and monitor its definition, we test the created Web Service and we define a proxy and a logical port to consume it in the Web Dynpro ABAP.

Keywords:
Web Service, Application Server ABAP, SAP NetWeaver Platform, Web Dynpro ABAP

INTRODUCTION

The development environment of the Application Server ABAP used to create ABAP–based applications is the ABAP Workbench. This environment offers the possibility to publish, search for and call a Web Service (WS).
Fig. 1 shows the basic architecture of the WS Framework that belongs to the AS ABAP [1].

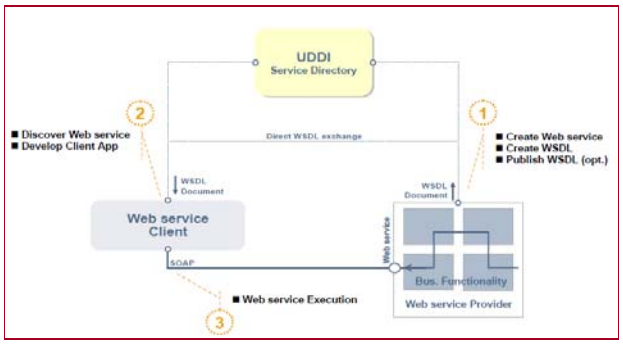


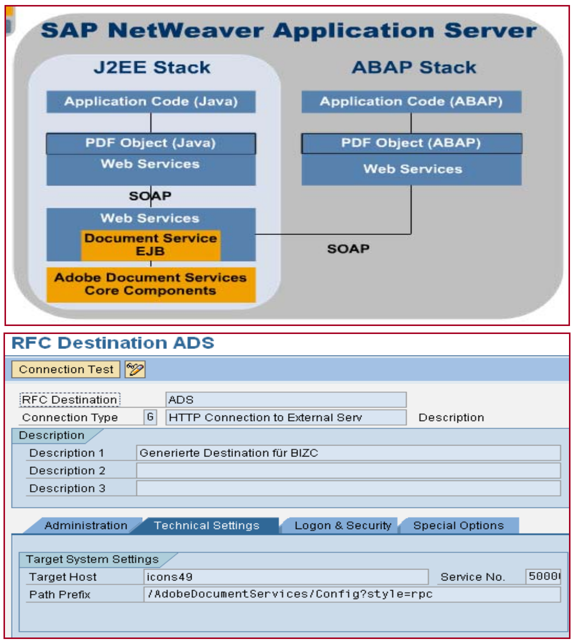
Fig. 1 Basic WS Framework architecture [1]

The WS fundamental technologies are:

- **SOAP** – Simple Object Access Protocol, XML based, extensible protocol that describes how to invoke a Web service;
- **UDDI** – Universal Description, Discovery and Integration, business registry that can be used to index WSDL documents; so, this is searchable;
- **WSDL** – Web Service Description Language, special form of XML that contains all the information that a client needs to invoke the WS;
- **WS-security** – security standard as X.509, Kerberos, Secure Socket Layer Protocol SSL, etc.

The service provider creates the implementation of the WS and provides the WSDL document. He is responsible for the execution of the functionality provided by the WS.
In ABAP, we can create a service provider for an ESR Service Interface, the so-called „outside-in" provider, or for an existing ABAP object, the so-called „inside-out". A service required can be Enterprise Service Repository, URL/HTTP destination or a local File.
A WS can be used in many situations, from email validation to automation. As an example of using a WS in SAP NetWeaver, we can mention the communication between AS ABAP and

Adobe Document Services (ADS) that run on Java stack, communication that is made via a WS. Fig. 2 schematically shows this communication and the HTTP connection to the External Server [2].



RFC Destination	ADS	
Connection Type	G	HTTP Connection to External Serv

Description

Description 1	Generierte Destination für BIZC
Description 2	
Description 3	

Target System Settings

Target Host	icons49	Service No.	5000
Path Prefix	/AdobeDocumentServices/Config?style=rpc		

Fig. 2 Web Service example [2]

Some advantages of the WS are presented below [3]:

- *There are defined independently of programming platforms and languages;*
- *The WS definitions are expressed in XML syntax;*
- *They can be developed in any programming language;*
- *They can be published in a common directory based on the UDDI standard;*
- *They can be easily executed over the internet.*

We can store released Web Services in a UDDI registry. For the purpose of this paper, we have used our service provider landscape.

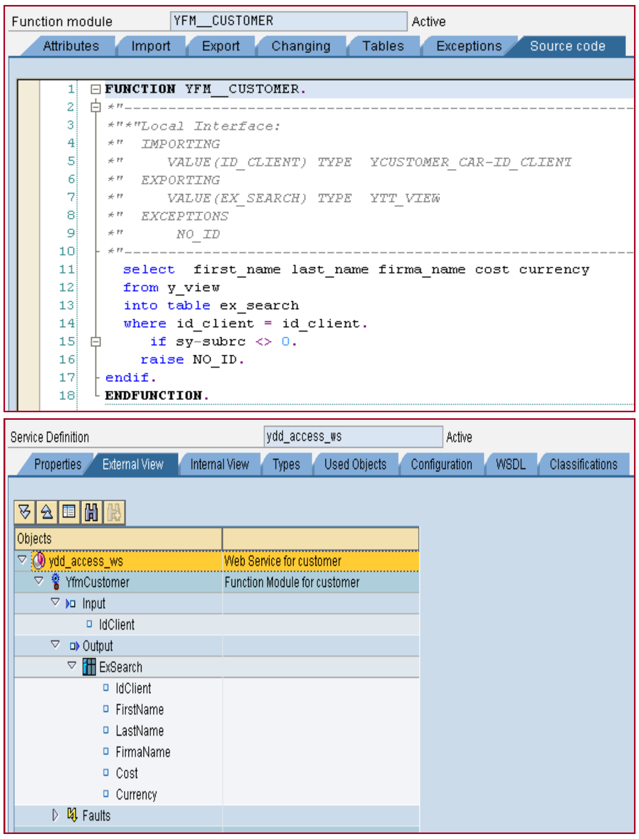
There are many organizations that offer web services for free (for example, the reference [4]). At the reference [5], we can find the web address of the public UDDI service directory that offers SAP.

CREATING AND TESTING A WEB SERVICE

With the ABAP Workbench, we have many options to create a WS. For example, we can use a BAPI, a Function Module, a Function Group or a Message Interface.

We want to provide selection access to a database table. A Function Module implementation will be used as the Web Service end point. After the implementation of the Function Module, we create the Web Service definition with only a few mouse clicks. In this case, we have created an inside-out service provider, because we have started with the existing functionality and interfaces from inside our system and used them as the basis of a new system.

In our case, the WS will require the customer ID and deliver the selected information. Fig. 3 shows the structure of our Function Module and the created Web Service, by using the Service Definition Wizard.



Function module YFM__CUSTOMER Active

```
FUNCTION YFM__CUSTOMER.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(ID_CLIENT) TYPE  YCUSTOMER_CAR-ID_CLIENT
*"  EXPORTING
*"     VALUE(EX_SEARCH) TYPE  YTT_VIEW
*"  EXCEPTIONS
*"      NO_ID
*"----------------------------------------------------------------------
  select  first_name last_name firma_name cost currency
  from y_view
  into table ex_search
  where id_client = id_client.
     if sy-subrc <> 0.
    raise NO_ID.
endif.
ENDFUNCTION.
```

Service Definition ydd_access_ws Active

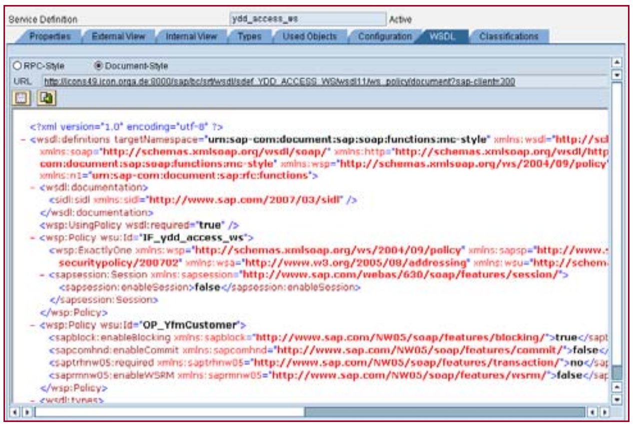
```
Objects
ydd_access_ws              Web Service for customer
  YfmCustomer              Function Module for customer
    Input
      IdClient
    Output
      ExSearch
        IdClient
        FirstName
        LastName
        FirmaName
        Cost
        Currency
    Faults
```

Fig. 3 Function Module and Web Service structure

In the WSDL Tab, we can find the XML representation of the WS definition. Fig. 4 shows the structure of this file.

From the SAP NetWeaver 7.0, the SP14 Web Services in the ABAP development environment are no longer managed with the transactions WSADMIN and WSCONFIG. We can use these transactions only for the old WS. To manage the new WS, we use the transaction SOAMANAGER. This transaction represents a new Web Dynpro

ABAP application that helps us to manage, configure and monitor the service definitions.
The Service-Oriented Architectures (SOA) enables the effective management of an SOA implementation, represents a concept that offers much more than a WS [6]. Fig. 5 shows our WS into the Web Service Administration option from the SOA Manager.



Service Definition ydd_access_ws Active

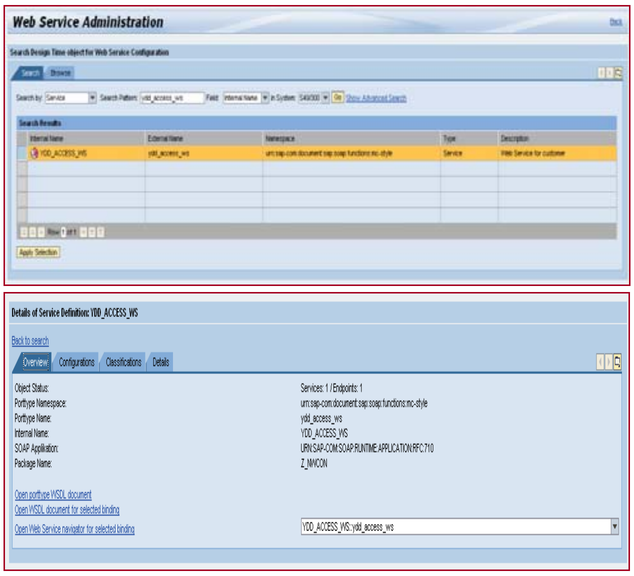
Properties | External View | Internal View | Types | Used Objects | Configuration | WSDL | Classifications

○ RPC-Style ◉ Document-Style

URL http://icons49.icon.orga.de:8000/sap/bc/srt/wsdl/sdef_YDD_ACCESS_WS/wsdl11/ws_policy/document?sap-client=200

```
  <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions targetNamespace="urn:sap-com:document:sap:soap:functions:mc-style" xmlns:wsdl="http://scl
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http
    com:document:sap:soap:functions:mc-style" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:n1="urn:sap-com:document:sap:rfc:functions">
  - <wsdl:documentation>
      <sidl:sidl xmlns:sidl="http://www.sap.com/2007/03/sidl" />
    </wsdl:documentation>
    <wsp:UsingPolicy wsdl:required="true" />
  - <wsp:Policy wsu:Id="IF_ydd_access_ws">
      <wsp:ExactlyOne xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:sapsp="http://www.
        securitypolicy/200702" xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsu="http://schem
    - <sapsession:Session xmlns:sapsession="http://www.sap.com/webas/630/soap/features/session/">
        <sapsession:enableSession>false</sapsession:enableSession>
      </sapsession:Session>
    </wsp:Policy>
  - <wsp:Policy wsu:Id="OP_YfmCustomer">
      <sapblock:enableBlocking xmlns:sapblock="http://www.sap.com/NW05/soap/features/blocking/">true</sapt
      <sapcomhnd:enableCommit xmlns:sapcomhnd="http://www.sap.com/NW05/soap/features/commit/">false</
      <saptrhnw05:required xmlns:saptrhnw05="http://www.sap.com/NW05/soap/features/transaction/">no</sap
      <saprmnw05:enableWSRM xmlns:saprmnw05="http://www.sap.com/NW05/soap/features/wsrm/">false</sap
    </wsp:Policy>
  - <wsdl:types>
```

Fig. 4 The WSDL structure


Web Service Administration
Search Design Time object for Web Service Configuration
Search
Browse
Search Results
Apply Selection

Details of Service Definition: YDD_ACCESS_WS

Back to search

Overview | Configurations | Classifications | Details

Object Status:	Services: 1 / Endpoints: 1
Porttype Namespace:	urn:sap-com:document:sap:soap:functions:mc-style
Porttype Name:	ydd_access_ws
Internal Name:	YDD_ACCESS_WS
SOAP Applikation:	URN:SAP-COM:SOAP:RUNTIME:APPLICATION:RFC:710
Package Name:	Z_NWCON

Open porttype WSDL document
Open WSDL document for selected binding
Open Web Service navigator for selected binding YDD_ACCESS_WS::ydd_access_ws

Fig. 5 Web Service administration

To configure a Web service, we must create an end point that contains a runtime configuration. We have created only an end point, but we have the possibility to create more than one if we want to provide the same service with different runtime configurations.
AS NetWeaver offers the possibility to create secure WS; we can speak about security at the transport layer and security at the message layer. Fig. 5 shows the Security Provider specially created for our service.

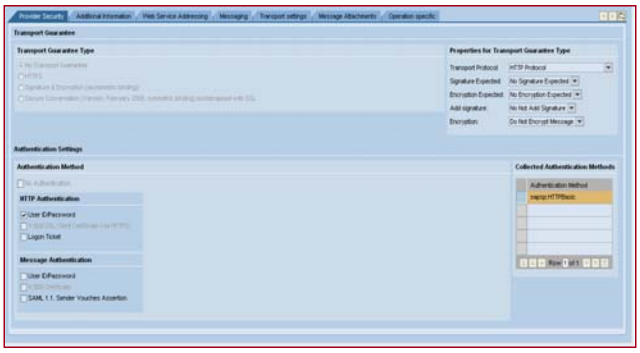
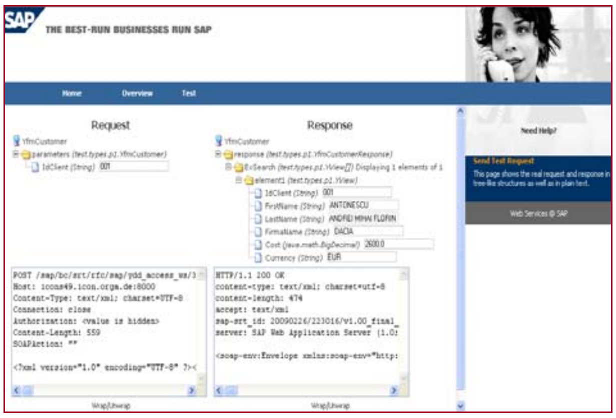


Fig. 5 Security Provider

As we have seen, our transport protocol is the HTTP protocol, authentication through user ID and password. For this kind of protocol, we can choose one of the following security functions [7]:

- *Server-side authentication;*
- *Client-side authentication;*
- *Mutual authentication;*
- *Encryption and integrity.*

Before developing the client application for our WS, we have to test it. With this test, we ensure that it works correctly and can be consumed in the Web Dynpro ABAP without any problems.



Request:
YfmCustomer
parameters (test.types.p1.YfmCustomer)
IdClient (String) 001

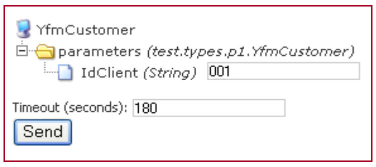
Response:
YfmCustomer
response (test.types.p1.YfmCustomerResponse)
ExSearch (test.types.p1.YView[]) Displaying 1 elements of 1
element1 (test.types.p1.YView)
IdClient (String) 001
FirstName (String) ANTONESCU
LastName (String) ANDREI MIHAI FLORIN
Firmaname (String) DACIA
Cost (java.math.BigDecimal) 2600.0
Currency (String) EUR

```
POST /sap/bc/srt/rfc/sap/ydd_access_ws/3
Host: icons49.icon.orga.de:8000
Content-Type: text/xml; charset=UTF-8
Connection: close
Authorization: <value is hidden>
Content-Length: 559
SOAPAction: ""

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

```
HTTP/1.1 200 OK
content-type: text/xml; charset=utf-8
content-length: 474
accept: text/xml
sap-srt_id: 20090226/223016/v1.00_final_
server: SAP Web Application Server (1.0;

<soap-env:Envelope xmlns:soap-env="http:
```



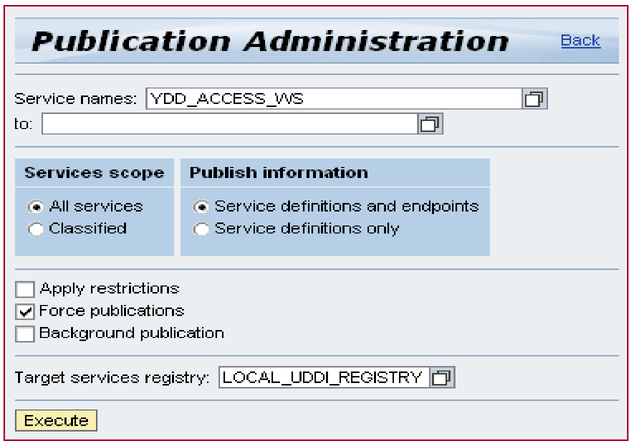
YfmCustomer
parameters (test.types.p1.YfmCustomer)
IdClient (String) 001

Timeout (seconds): 180

Send

Fig. 6 Testing the WS

To test our WS, we use "Open Web Service navigator for selected binding" from the SOA Manager, after we have set the address of the application server on which the J2EE is running. The Web Service Navigator is open. Then, we enter the user ID and the password, to be able to test it. Fig. 6 shows the way we can see if the WS works correctly. We enter our WS parameter idClient that will be passed as a request to the Application Server. The Web Service Navigator *sends us back the response, including all the records for the searched ID. Another possibility to test our WS is to use the WS navigator via URL http://<host>:<port>/wsnavigator.*



Publication Administration Back

Service names: YDD_ACCESS_WS
to:

Services scope	Publish information
(●) All services	(●) Service definitions and endpoints
( ) Classified	( ) Service definitions only

☐ Apply restrictions
☑ Force publications
☐ Background publication

Target services registry: LOCAL_UDDI_REGISTRY

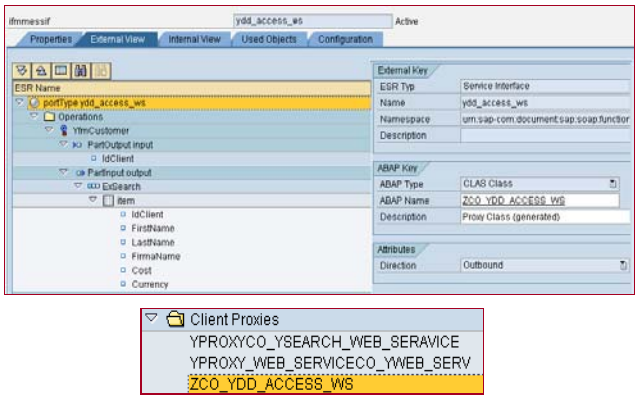
Execute

Fig. 7 Publishing the Web Service

The Service Registry is a central Register for the WS. Here, we can publish our WS by using the WSPUBLISH transaction, or the Publication Administration from the SOA Manager (Fig. 7). The Service Registry offers the possibility to search for a WS by using some Categories. To classify a WS, we can use the WSCLASS transaction.

CONSUMING A WEB SERVICE IN THE WEB DYNPRO ABAP

Web Dynpro ABAP is the SAP technology used to create web business applications in accordance with the Model View Controller (MVC) paradigm. According to this paradigm, the application data and their user interface are separated.



ifmmessif ydd_access_ws Active

Properties | External View | Internal View | Used Objects | Configuration

ESR Name
- portType ydd_access_ws
  - Operations
    - YfmCustomer
      - PartOutput input
        - IdClient
      - PartInput output
        - ExSearch
          - item
            - IdClient
            - FirstName
            - LastName
            - FirmaName
            - Cost
            - Currency

External Key

ESR Typ	Service Interface
Name	ydd_access_ws
Namespace	urn:sap-com:document:sap:soap:functior
Description	

ABAP Key

ABAP Type	CLAS Class
ABAP Name	ZCO_YDD_ACCESS_WS
Description	Proxy Class (generated)

Attributes

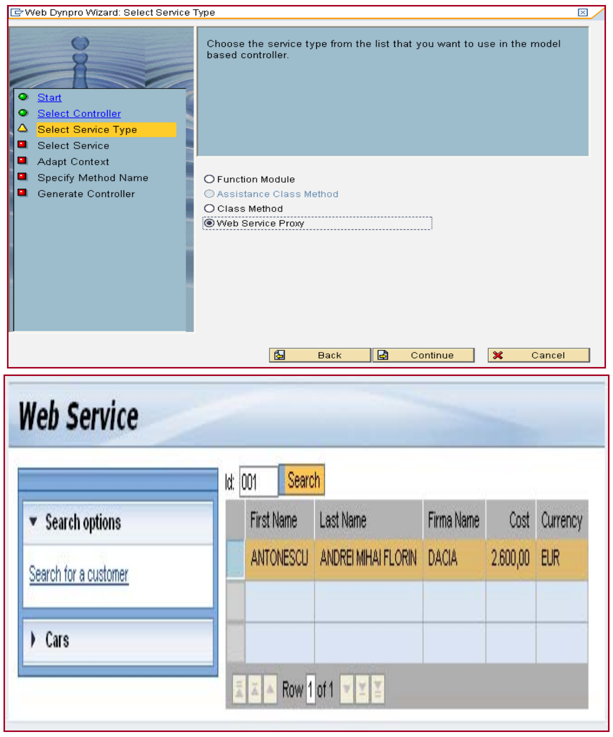
Direction	Outbound

Client Proxies
- YPROXYCO_YSEARCH_WEB_SERAVICE
- YPROXY_WEB_SERVICECO_YWEB_SERV
- ZCO_YDD_ACCESS_WS

Fig. 8 Proxy Object structure

The Model represents the business logic, the View represents the user interface and the controller has certain responsibilities, as the communication between the model and the view. More details about the Web Dynpro ABAP can be found at the references [8-10].

As we have seen, the WSL document describes our WS. To be able to consume this WS, we have to create a client proxy and a logical port for him. Fig. 8 shows the proxy structure.



Web Dynpro Wizard: Select Service Type

Start
Select Controller
Select Service Type
Select Service
Adapt Context
Specify Method Name
Generate Controller

Choose the service type from the list that you want to use in the model based controller.

( ) Function Module
( ) Assistance Class Method
( ) Class Method
[✓ Web Service Proxy]

Back | Continue | Cancel

Web Service

Search options
Search for a customer
Cars

Id: 001 Search

First Name	Last Name	Firma Name	Cost	Currency
ANTONESCU	ANDREI MIHAI FLORIN	DACIA	2.600,00	EUR

Row 1 of 1

Fig. 9 Consuming the WS Wizard and the User Interface

We consume our defined Web Service as a model in the Web Dynpro. In this way, we are not interested about the way to implement the business logic, but we use only its functionality. Fig. 9 shows the proper User Interface and how we can consume a WS in the Web Dynpro ABAP.

CONCLUSION

WS are modules in service-oriented software architectures; they are executable units that can be called in heterogeneous system landscapes.

In this paper, we have used some of the WS concepts with the Application Server ABAP. Through an example, it has been examined the inside-out approach for generating WS and consuming them by using the Web Dynpro ABAP. In the same time, we have used the Web Service navigator to test our WS. So, we have seen that a WS can be tested without being necessary to have a consuming application.

The new "trend" in SOA is the Enterprise Services, and that's why we have a new

SOAMANAGER transaction that incorporates the functionality of the old WSCONFIG and WSADMIN and adds new capabilities required to integrate a WS in this concept.

REFERENCES

[1.] https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/ 30f 1b585-0a01-0010-3d96-ad0ea291c4f9

[2.] https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/ 20029530-54ef-2910-1b93-c41608ae0c90

[3.] https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/ f65ecf90-0201-0010-94b0-c9983be54c67

[4.] http://www.xmethods.net/ve2/index.po

[5.] http://uddi.sap.com

[6.] Martin Huvar,Timm Falter,Thomas Fiedler, Alexander Zubev, Developing Applications with Entreprise SOA, Galileo Press 2008

[7.] Martin Raepple, The Developer's Guide to SAP NetWeaver Security, Galileo Press, 2007

[8.] U. Gellert, D. Cristea, Web Dynpro ABAP praxisbook, Springer, in press

[9.] Rich Heilman, Thomas Jung, Next Generation ABAP Development, Galileo Press 2007

[10.] The 14th international conference, The knowledge based organization, November 2008, Cristea Ana Daniela, Adela Diana Berdie, Osaci Mihaela, User Interfaces with Web Dynpro ABAP and Web Dynpro Java, "Nicolae Balcescu" land Forces Academy publishing Haus Sibiu, 2008.

[11.] http://help.sap.com

AUTHORS & AFFILIATION

[1.2.] ANA DANIELA CRISTEA,
[2.] Ovidiu GELU TIRIAN

[1.] UNIVERSITY "POLITEHNICA" TIMISOARA, FACULTY OF ENGINEERING HUNEDOARA, ROMANIA
[2.] NWCON TECHNOLOGY CONSULTING, GERMANY

ACTA TECHNICA CORVINIENSIS – BULLETIN of ENGINEERING
ISSN: 2067-3809 [CD-Rom, online]