Acta
Technica
Jaurinensis

Vol. 7., No.2., pp.207-223, 2014
DOI: 10.14513/actatechjaur.v7.n2.283
Available online at acta.sze.hu

Effective End-user Interfaces for Various Business Needs

A. Selmeci, T. Orosz

Óbuda University, Székesfehérvár, Hungary
e-mail:selmeci.attila@arek.uni-obuda.hu, orosz.tamas@arek.uni-obuda.hu

Abstract: SAP, the leading Enterprise Resources Planning System in the World, has been providing different effective solutions for business requirements for more than 40 years. Therefore SAP tried to follow the changes of the end-user requirements with different user interface solutions. The technology novelties drive the internal developments to support new protocols and offer more visual components. Our paper discovers the possibilities lying on the desk and gives directions to select the most effective technique for a given business need. In the last years the end-user want to have efficient UIs, which bring more potent to the daily work. Beyond offering effective and efficient options we uncovered the power of the different solutions in changing environments as well.

Keywords: user interface, SAP, front-end, efficient surface, sustainable UI

1. Introduction

SAP has a client-server architecture, which is enhanced according the market needs. This brings the changes and extension of user interface techniques as well [1-3]. This paper analyzes the different available approaches by showing the effectiveness of them. It describes the usability from end-user and project perspectives as well. To understand the mechanism of the different UI techniques first we explain the SAP architecture from UI point of view. In the paper we should distinguish between front-end based and Web-based UI surfaces also.

2. Basic Architecture of the SAP Systems

SAP solutions are based on the SAP Web Application Server technology. This technology provides open, scalable and robust infrastructure for running and developing dynamic applications. This is the core element of the SAP ERP (successor of R/3) system as well. The earlier R/2 releases had two layers, where the business logic and the storage as a monolith element took place on a mainframe, and the presentation layer appeared on a terminal. The communication capabilities were limited to LU 6.2 for other systems. R/3 systems inherit the real-time capability as we can see in the 'R' letter, but is redesigned for three-tier client-server architecture (see Fig. 1).

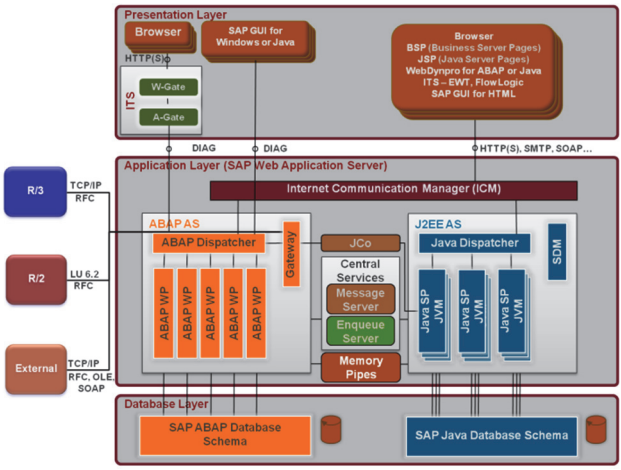

Figure 1. WebAS high-level technical architecture

This architecture means that the application is divided into three layers:

- Database layer: this is a single component (except some supported grid database systems), where the business data, and the whole repository are stored.
- Application layer: it is responsible for controlling and processing the applications. The business logic is running in this layer providing system-oriented services and ensures connectivity to presentation and other software components. For Web application this layer can be separated to more parts determining the connectivity, presentation, business logic itself, integration and persistent layers. (See details on Fig. 2. and later.)
- Presentation layer: it is the front-end, the Graphical User Interface, that runs on PCs, workstations, Web browsers or even on mobile devices.

Fig. 1. shows above the today's high-level technical architecture of the SAP Web Application Server.

The above figure refers some till now not detailed component and protocols used in communications. In the early R/3 system, before WebAS SAP does not provided direct HTTP protocol, but only DIAG (Dynamic Information and Action Gateway) and RFC (Remote Function Call) were provided. The DIAG is SAP's own protocol to communicate between the application layer and the SAP front-end software, called SAP GUI. For real data exchange SAP uses the CPI-C (Common Program Interface for Communication) based RFC protocol. This is a program-to-program communication protocol, which was used and implemented in the R/2 and R/3 world. The figure mentions the R/2 connection via LU 6.2 protocol. CPI-C and RFC communications could use TCP/IP and LU 6.2 communication as well. The IBM's Logical Unit (LU) 6.2 is part of the System Network Architecture (SNA) protocol. SAP provided by RFC a possible communication layer where other programs, system or even own developed front-end applications could use the system services.

SAP introduced the SAP NetWeaver Application and Integration platform around in 2003. This initiative was designed to give a comprehensive platform for many SAP applications and provide integration capabilities not only on technical level, but from business-oriented view as well. SAP NetWeaver integration platform is open and manages heterogeneous environments by the integration layers (containing several solutions, applications) from bottom up:

- Process integration (or the newer name: Business Process Integration): this integration layer guarantees the capability to integrate systems, technologies by interoperability using EAI (Enterprise Application Integration) tools as part of NetWeaver: Process Integration (PI), Business Process Management (BPM), Integration Broker.
- Information integration: As the process integration layer connected technically the implemented solutions in a company and in case of BPM on business level as well, higher abstraction level integration is also available. This integration manages data harmonization, consolidation through the company and beyond. Different data can be integrated in a company and different tools could solve the integration tasks. The documents and other contents should be organized around taxonomies and the unstructured objects should be accessed in a structured and role based manner. For this purpose SAP provides the Knowledge Management solution. The business data can be collected in data warehouse (BW, Business Warehouse) system for consolidation and business intelligence functions like reporting, dashboards and so on. SAP provides master data harmonization, mapping and full management capabilities within the information integration solution by Master Data Management (MDM).

People integration: as we integrated our best of bread (not only SAP solutions) through process and information integration it is useful to support user level integration as well. Main function of people integration is the collaboration. On the other hand the user interfaces from the different systems are integrated on the SAP NetWeaver Portal providing the right information to the right person on right time in right place. Beyond the portal the NetWeaver SOA (Service Oriented Architecture) capabilities offer a unified development and run-time tool for cross applications. The SAP Composite Environment (CE) supports the development of composite application providing web user interfaces for the new applications calling services from the different underlying application (even if they are beyond the company's boundaries).

The NetWeaver implements and applies open standards and can interoperate with other technologies like IBM WebSphere, J2EE or Microsoft .NET. As we mentioned, the NetWeaver is not only an integration platform, but an application platform as well. The application platform as it is shown on the above picture has two personalities: ABAP and J2EE. The ABAP is the original SAP owned application server, which is a process based server communicating with the database layer via process-to-process channel. The standard clients communicating via RFC or DIAG are connecting to the so-called dispatcher process, which dispatches the tasks to an available (empty) dialog process after queuing it for execution of the request [4-5]. The clients can log into an SAP ABAP system via server selection or group selection. In the first case the client is directly communicates with an SAP application instance immediately, but in the second

case the client asks the so-called message server to provide login information from a corresponding application instance. Corresponding mean in this case a good performing instance, so the message server helps in the logon load balancing as well. (The message server communicates short technical messages among the available application instances to provide an overall good performance from the response time point of view and collects data about the available services and load as well [6-8].) ABAP is the name of the internal SAP programming language referring to Advance Business Application Programming. Each of the SAP business applications are developed in ABAP, which is an open 4th generation event controlled language enhanced with object-orientation as well. The ABAP language has a so-called Open-SQL part for database communication as well. It provides (as described above) RFC protocol, which is an SAP implementation of RPC (Remote Process Call). SAP delivers and makes available to develop function modules (compilation units), which are remote enabled, and let other systems, programs, and applications to call services of the SAP systems. Main functionalities are implemented as function modules and many of them are RFC functions, so the SAP systems can be controlled and updated from outside via RFCs as well.

The other parts of the drown architecture will be discussed later to make easier the understanding of the expansion of the application server to Web Application Server.

As of NetWeaver releases the SAP Application Server is expanded with Web server functionality as well, and the whole SAP Web Application Server architecture can be separated as we mentioned above into five sub-layers according to the technical areas (see Fig. 2).

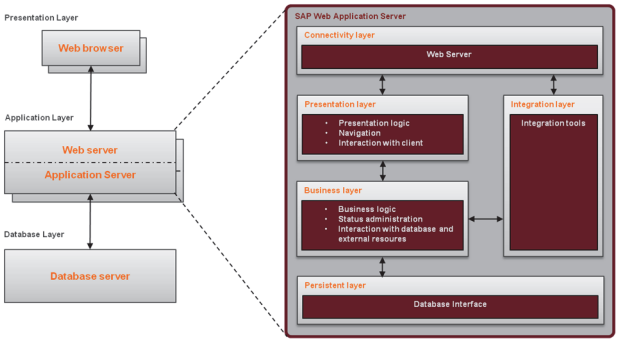

Figure 2. Sub-components of the Web Application Server

The 5 sub-components of the WebAS architecture:

- Presentation layer: presentation logic (view or screen definitions and switches), navigation, and the interaction with the client are defined here. The user interface can be developed on J2EE platform with Java Server Pages (JSP) or WebDynpro technology; on ABAP platform with Business Server Pages (BSP), WebDynpro for ABAP technology, or ITS (Internet Transaction Server) IAC (Internet Application Component, see later). For these kinds of surfaces the underlying business layer provides the business login and content.

- Business layer: the business logic is running here, so the status of the applications are administered as well on this layer, and of course the service call like database connection or external resource usage are defined and maintained here. This layer provides run-time environment for the business logic by processing the requests passed from the ICM (Internet Communication Manager, see below), and the responses are generated also here. The business logic (encapsulating the persistent layer) is implemented in J2EE environment by Enterprise JavaBeans (EJB), or in the ABAP environment they are developed in the ABAP Workbench using local objects and persistence.

- Integration layer: this layer opens the Web AS via integration engine to communicate via standard interfaces with other systems and solutions. Message service is provided for communication between the Web AS and through SAP XI (Exchange Infrastructure, basic EAI solution by SAP) connected systems. It enables other systems to integrate their functionalities and services to SAP Web AS for usage in Web applications.

- Connectivity layer: this is the first layer facing with client or connected systems, application. The main service, which provides the Web server functionality and other standard protocols, is the Internet Communication Manager (ICM). Through the Internet Communication Framework (ICF) many standard protocols are available, like Hypertext Transfer Protocol (HTTP), HTTPS (extension of HTTP running under the Secure Socket Layer (SSL)), Web Distributed Authoring and Versioning (WebDAV), Simple Object Access Protocol (SOAP), Fast Common Gateway Interface (FastCGI), and Simple Mail Transfer Protocol (SMTP), etc. ICM dispatches UI requests to the above mentioned presentation layer, and the ICF is used for connectivity using the listed, various communication protocols.

- Persistence layer: as we described above the Web AS ABAP environment guarantees database independent programming by Open SQL. The database interface (DBIF) handles the SQL requests and translates them to the underlying native database SQL language in an optimized way. SAP developed the Open SQL capabilities to the Java world as well and offers a variety of standard Application Programming Interfaces (APIs) to Java programmers, such as SQLJ and other Java technologies.

As we have seen the SAP Web AS with its two (ABAP and J2EE) personalities provides not only thick client capabilities (see it soon), but also many different Web UI options are offered out of the box.

3. Frontend Possibilities of SAP Systems

SAP provides a universal front-end solution called SAP GUI (SAP Graphical User Interface). SAP at the beginning provided stand-alone SAP GUI solutions for Mac, OS/2, Windows and so on, but later decided to support three flavors according the available platforms. The SAP GUI family contains

- SAP GUI for Windows supporting MS Windows operating systems based on OLE and ActiveX controls
- SAP GUI for Java supporting all environments running Java runtime environment. This GUI is generally called platin GUI as well, because it is a platform independent solution.
- SAP GUI for HTML is a web based GUI frequently called as WebGUI because of the name of the service, which provides it.

SAP communicates with the SAP GUI client via DIAG protocol and in some cases RFC is used as well. The SAP GUI is a thick client; it must be installed onto the front-end workstation. The standard GUI for SAP is the SAP GUI and it was designed for end-users working with numbers, calculations and so on (as the Fig. 3. shows on the right side).

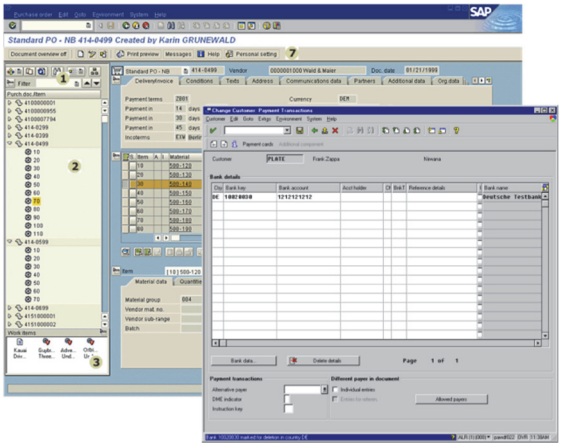

Figure 3. SAP GUI with and w/o Enjoy controls

The figure above contains two screen shots. The right one refers the original R/3 SAP GUI for Windows look. Each SAP application looks like the other, because the client software retrieves the screen element type, position, size, and of course the content and some other attributes, but the final design takes place on the front-end machine. Each of the screens has a standard tool bar containing standard, almost always available functions (like save, print, page up and down, etc.), a menu up to 8 items, but the last two items are always accessible having standard functions. If the application requires another, so-called customer tool bar can be defined holding pushbuttons with special functions. The surface itself can contain many different screen elements, like text field, input field, tab strip, table control, checkbox, radio button, frame, pushbutton, etc. The designs of these elements are predefined in the client software.

There are two main look and processing options of SAP programs: reports and screens. The two kinds of program are different because the reports are generally reading programs, which collect data and present them in a list. (In some cases report can be used for mass changes as well.) The screen or transaction programs are designed to maintain data in the SAP systems. An SAP transaction (Logical Unit of Work) leads

through more screens (or dynpro-s as the original name refer to dynamic program) collecting different data for a special purpose before booking it into the database. Behind the screens the transaction main program holds the status of the transaction and handles the dataflow and navigation between the screens. Each screen contains beyond the element list and attributes, the so-called flow logic, which has (at least) two main parts the Process Before Output (PBO) and Process After Input (PAI) event blocks. These event blocks are executed for each screen at the time as their names describe, so PBO can contain reading customizing, checking authorization, etc. and with these information we can set default values and modify the attributes (e.g. required or read-only) of the screen elements before it is sent to the client. When the client executes a function, which starts a request to the Application Server the client sends the information and the PAI event is raised. The transaction executes the PAI event block, where the entered data is checked against conditions and if everything is correct navigation to another screen or database updatecould take place. The above figure shows a screenshot of a transaction dynpro.

As we described the SAP application look similar. It is because SAP recommends to use a standard style guide not only for standard SAP delivered programs and applications, but for customer developed once as well. For that SAP owns a Web site http://www.sapdesignguild.org/ containing the relevant information for programmers or designers.

SAP realized that the users want to have a bit easier, colorful, manageable surface, so as of release R/3 4.6 SAP implemented the so-called Enjoy controls to make the end-user of SAP's software a V.I.P. meaning new visual design, new interaction design and personalization options. As it is shown on the left side of Fig. 3. SAP implemented many new screen elements and the whole design was changed a bit as well. The Enjoy controls are implemented as ActiveX controls in case of SAP GUI for Windows, but for JavaGUI SAP used JavaBeans instead.

SAP implemented Text Editor, HTML and picture viewer, tool bar, hierarchical tree controls and the so-called ALV (ABAP List Viewer) control, among others. The programming of controls is a bit different, because in case of controls a huge data amount is sent from the Application Server to the client to store and provide to the corresponding control. Automation handling and OLE or Java means (depending on SAPGUI for Windows or for Java) do this data exchange. In the server side embedding proxy objects handle the request and manages the data exchange through the control framework. If we use Enjoy controls we should keep in mind the data amount and the performance and use the available optimization techniques (e.g. incremental tree construction) not to put too much load to the network or to the front-end PC at a time. As an example the Tree controls make it possible not to send the whole tree in one step but only some levels (e.g. the first only) of the tree. If the end-user wants to open a tree node defined as a folder without having data under it, the client sends back an event for the child node (`EXPAND_NO_CHILDREN`) to bring to corresponding data. If the node is not opened the data will not be collected and sent to the client, so application processing, network and client side storage resources are saved.

SAP GUI gives enhanced opportunities implemented on client side as well. In this case depending on the client PC setting the same SAP transaction, screen can look different. Two options are available currently:

- GuiXT
- GUI Scripting.

For the GuiXT implementation on the front-end PC the GuiXT component should be installed as well (as part of the standard SAP GUI product CD). For each screen referred by transaction main program (module pool) name and the screen number a separate text file can be created for each language containing the script for the GuiXT. With the GuiXT screen elements can be hidden, moved, and new one pasted on the screen. This program runs with the SAP GUI process and modifies the look according to the script definition if available for the current screen. As an example an image can be placed to the screen be the following script row:

```
Image (20,30) "C:\Images\maci.jpg"
```

The local image file**C:\Images\maci.jpg** is displayed in row **20**, column**30**. This capability was delivered with the SAP GUI for earlier releases as well, than the Enjoy controls handle any pictures. The solution was designed by Synactive GmbH, and the whole documentation and technology of GuiXT is written on the http://www.synactive.com site.

The GUI Scripting is a different scripting technology and it was introduced as of Web AS 6.20. SAP GUI Scripting do not change the screen layout as GuiXT does. An object model is delivered, where the screen is represented with its controls at runtime and an API can be used to modify the object itself. The API can be used from the VBScript and JavaScript languages. The main different is that the GUI scripting will not change the surface, but a script can be created manually or even by recording a SAPGUI execution and the script content can be change to execute again with other data. It is useful to automate such tasks, which should be executed repeatedly. It is a very good tool for testing a transaction.

SAP developed for non-standard display environments the SAPConsole client to support character-cell terminals, including radio frequency (RF) devices or even web-equipped devices. This is not a SAP GUI, but a different approach to handle special UI-s. Two possible architecture are available: telnet client with a telnet server running the console displayer, and a browser based solution where an IIS Web Server runs the Web Displayer program, which communicate with the device. Fig. 4. shows an example.

Figure 4. SAP Console example

4. External, Developed Frontend options

As a communication layer SAP provides RFC enabled function to the outside world. These functions can be called from own developed programs or applications. The architecture figure shows above that the SAP communication through the gateway process, which handles the program-to-program communication to partner applications. SAP offers for external developers connectors to make easier the communication with the SAP:

- RFC Library: this is the classic RFC connector, which offers with its RFC API C-routines to create external RFC capability for own developments. RFC library makes it possible to create server or client RFC capable programs as well is C/C++ language. This library will be replaced by the enhanced NetWeaver RFC Library, which provides more features and improvement, but is not compatible to the classic RFC Library.
- SAP Java Connector: with the SAP JCo Java application can communicate with SAP systems. The connector contains toolkit for providing and consuming RFC services as well. As on the Fig. 1. is drown, SAP uses this connector for inside communication as well. (The ABAP stack of the Web AS is process based, but the Java stack is thread based, so different connections are needed for communications.)

SAP Connector for Microsoft .NET (SAP NCo): It makes possible to call SAP RFCs or BAPIs directly from .NET applications even it is written in Visual Basic, C++ or C#. The provided Proxy Wizard makes possible the integration of business data and processes of SAP systems into .NET by generating proxy classes, which contain the required methods, attributes (properties) and some table structure definitions. The connector gives easy connection with pre-generated content (e.g. logon) in the Visual Studio .NET environment. SAP NCo manages SAP SOAP connections as well.

With these connectors it is possible to develop any required UI for SAP forgetting the standard SAP GUI and its possibilities. The basis of the developed UI is the RFC modules, which could provide data from and to the SAP system.

SAP introduced the Business Framework Architecture (BFA) to provide an object-oriented layer on the SAP application functionality. The BFA collects the Business Objects, like Employee, Sales Order in so-called Business Components. Business

Objects have attributes, methods, events, as any object can have in an object-oriented environment. These Business Object components can be used within an SAP system, but some of the methods are public for even external programs as well. These public methods are the BAPIs (Business Application Programming Interfaces). Technically the BAPIs are implemented as remote-enabled function modules in the SAP system, but some of the external codes (using corresponding connectors) can refer to them not only as RFC functions, but also as methods of objects. It means real object oriented object are created on the caller side and the methods (only the public ones, the BAPIs) can be invoked. For sustainability SAP freezes the interface definitions of the BAPIs. The older programs and external UIs can still use the old BAPIs. If interface modification needed, SAP implements a new method, new BAPI with the new signature.

SAP can be used as OLE (Object Linking and Embedding) server or as OLE client. For UI development we can use SAP as an OLE server. Visual Basic or even Excel can be used directly to call up SAP, log in and use according the login authorization the (remote) services of the system. With this feature simple Excel applications can be developed where behind an SAP system provides (or stores) the data, so Excel is a special UI for SAP.

SAP itself uses this as well for Business Warehouse (SAP BW) UI, because the SAP Business Explorer (BEx) is an Excel based solution (delivered as a front-end component with SAP GUI as well). The end-user can evaluate old and current data to varying degrees of detail and from different perspectives on the Web and also in MS Excel.

5. SAP Web Frontend Options

SAP implemented around middle 90's a special Web enablement for the systems. During that time SAP had no Web Application Server technology as Fig. 2. shows, but a separated component was introduce to convert the SAP DIAG protocol to the Web HTTP protocol. This converter component is the Internet Transaction Server (ITS). The ITS contains two separated sub-component: application gate (A-gate) for application server communication and a Web gate (W-gate) running on a Web server providing mime object and rendering the converted information with additional date to provide a usable html page for the end-user. SAP offers three programming models as shown on Fig. 5.

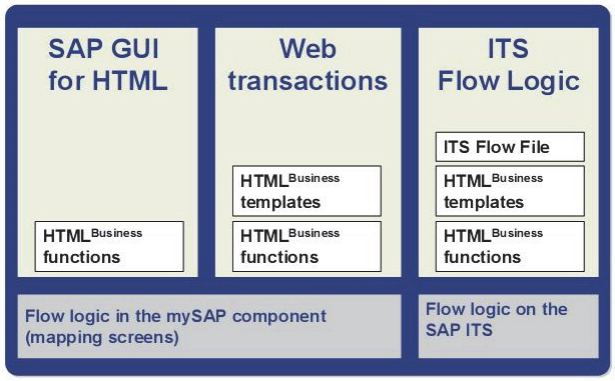

Figure 5. SAP ITS Programming Models

Earlier we mentioned that SAP GUI family has an SAP GUI for HTML, the so-called WebGUI. This is provided as a service by the ITS, which converts the standard SAP content to HTML Business functions. This UI is a standard front-end like each of the elements of the SAP GUI family.

The EWT (Easy Web Transaction) is a transaction, where the status of it takes place in the SAP system, because the basis of an EWT is a standard SAP transaction. Each of the screens (Dynpros) of the SAP transaction is translated in design time to so called HTML Business templates, which contain the screen element definitions using HTML Business functions. These templates are stored on the ITS server, and can be modified and enhanced according to the Web requirements (e.g. images can be placed instead of buttons, etc.). When the end-user executes the transaction on the Web, SAP starts it in the backend (using the PBI of the first screen) and sends the data (via DIAG protocol) to the ITS, which puts the data into the html page using the pre-generated HTML templates and the W-gate adds mime objects if needed. When the end-user executes a function (e.g. presses a pushbutton) the ITS receives the request and translates to the SAP "language" of the SAP protocol and sends forward to the SAP application server. The SAP executes the PAI processing block of the current screen and sets the status of the transaction. This UI serves as an easy way of converting existing SAP transaction to Web format.

SAP ITS offers a third programming model, where the status of the transaction is stored on the ITS side. The ITS Flo Logic manages event flows, where the ITS can call back to the SAP system to retrieve or store data if the process flow requires it, but the screen definition and the flow logic is totally defined on the ITS side. ITS can use only remote enabled function for this purpose, so only RFCs and BAPIs can be used for this task.

As the Fig. 2. shows the new SAP Web Application Server itself contains a built in Web server and with it offers beyond the old DIAG and RFC protocols many new a standard protocols like HTTP, SOAP, SMTP, etc. The main component of the Web enablement is the Internet Communication Manager (ICM) and the framework (ICF) around it. The services offered by a Web Application Server are listed in the ICF. As a special service SAP offers the so-called Integrated ITS as a service as well. So the Fig.

1. has the ITS offerings in the presentation layer because of the service. The `webgui` service is also offers as a standard service without installing any other components for it.

In the ABAP Web AS SAP introduced the page based server side scripting Web programming tool, the Business Server Pages (BSP). This is a real internal development tool and environment for Web UIs using local (e.g. database, file system) or remote (other SAP or non-SAP systems remote enabled functionalities) resources for data retrieval and storage. A BSP application is a complete functional web application containing everything like a classic SAP transaction. BSP defines a Web UI containing server-side scripting, arbitrary additional files (Images, background pictures, button GIFs,), Style-Sheets, etc. The technical structure of a BSP Application is shown on Fig. 6. Essentially a BSP application is divided to user interface and business logic. The user interface itself includes the static Web pages, dynamic, generated Web sites, which can be templates containing server-side scripting, where as scripting language JavaScript or ABAP can be used.

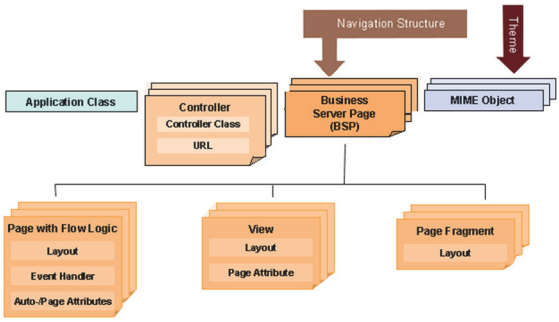

Figure 6. Structure of an SAP BSP Application

The scripts are executed dynamically during runtime. The generated static Web sites can be defined as pages with flow logic or even views. Depending on the requirements and the developing style pages can be defined with flow logic, simple views or only page fragments as the structure hierarchy of the BSP application shows on Fig. 6. The pages with flow logic are the original BSPs defining a page layout with scripts, attributes holding the state of the page or the application (depending on the definition), and event handlers to process end-user actions. The code snippet quoted below show a very simple example of page with flow logic using ABAP scripting. There are page attributes (`airlines, connections`) defined separately as global fields for the whole page, so the event handlers can use their content as well. The example creates a simple tree displaying connections by airlines. The code shows special and unusual tags. The `htmlb` tags as a library are defined according to the HTML for business definition, just like with ITS applications. There are many delivered libraries to use in BSP pages and the developer can define his own as well. Such library can be used as a rendering family, and makes it possible to use high-level programming constructs. The developer can achieve faster rendering than by hand. These tag libraries give one of the strengths of BSP applications.

```
<%@page language="abap" %>
<%@extension name="htmlb" prefix="htmlb" %>
<htmlb:content design="design2003" >
<htmlb:page title="Displaying a tree " >
<htmlb:form>
<htmlb:treeid    = "forest"
title = "This is a tree " >
<%
data: airline like line of airlines.
loop at airlines into airline.
%>
<htmlb:treeNode
id= "<%= airline-carrid %>"
text   = "<%= airline-carrname %>"
isOpen = "false" >
<%
data: connection
like line of connections.
read table connections
with key carrid= airline-carrid
transporting no fields.
ifsy-subrceq0.
loop at connections into connection
where carrid= airline-carrid.
%>
<htmlb:treeNode
id= "<%= connection-connid %>"
text = "<%= connection-connid %>"
isOpen = "false" />
<%
endloop.
endif.
%>
</htmlb:treeNode>
<%
endloop.
%>
</htmlb:tree>
<htmlb:textViewtext   = "Hello World!"
design = "EMPHASIZED" />
<htmlb:buttononClick= "myClickHandler"
text= "Press Me"/>
</htmlb:form>
</htmlb:page>
</htmlb:content>
```

As we referred above, not only special, but unusual tags are also in the html like code: `<%`, `%>`, `<%= airline-carrname %>`. These and some other (not mentioned) ones are the so-called BSP-directives. Using these directives we can use direct ABAP scripting in the code. E.g. in the code above a loop is executed on the `airlines` internal table to define a tree node for each airline. The internal table declarations are not part of the code, but they are defined among the page attributes (see Fig. 7. below).

The data of the internal tables are not read in the code, but separately in an event handler (`OnInitialization`) as the Fig. 7. shows below.

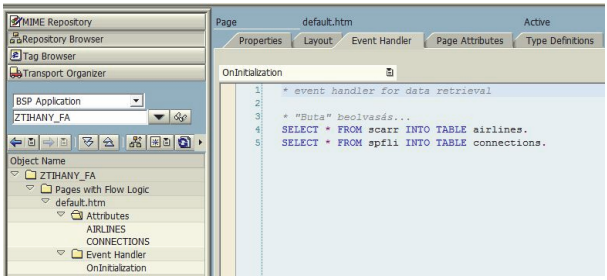

Figure 7. BSP development environment - events

There are more predefined events, which are triggered by the runtime environment except the input processing event, because this is triggered by the end-user action. As we have seen the BSP applications give much more capability, power and flexibility to the developer even with the standard pages with flow logic. In the later releases of BSP applications SAP introduced the model-view-controller design pattern. To define BSP application using the method SAP implemented views and controllers as shown on the Fig. 6. This kind of BSP applications are much more complex, but still gives the opportunity to define anything the end-user wants giving the capability for creating sustainable applications by separation of the user interface from the business logic.

Because of the ABAP scripting capability this Web tool has much higher interest than ITS ever (the old, experienced ABAP programmer can learn and program it easily) had. In the last releases SAP finished the development of standard Web transaction using BSP, but SAP still recommends using it for free-style Web programming for the customers.

SAP implemented first in J2EE Web AS and later in the ABAP Web AS as well the WebDynpro technology. The WebDynpro name come from the old, standard SAP screen name: dynpro. It uses the ModelView-Controller paradigm, and the definitions are stored as meta data and the system generates the corresponding code from this meta-data to create the real WebDynpro components. The WebDynpro components are reusable and according to the M-V-C paradigm they are sustainable because the look, the navigation, control and the business data management (model) are separated. The best way to keep the component unchanged during modification is to defin the model as methods of a class holding the state of the WebDynpro component. The methods are embedding tools for real services calls. A service call can be local real service, database extraction, managing files, or any remote service (RFC, BAPI from other SAP system, or Web service) consuming. WebDynpro cannot be used so freely as the BSP, because of the meta-data concept. Meta-data serves as the definition of the layout, screen elements, navigation, data exchange via hierarchical data store called context. The developer writes ABAP code only in special cases, like the real business logic takes place, or entry checks are needed. This makes easier to use different UI device for WebDynpro, because from the meta-data different, device dependent code can be generated by the SAP system. On the other hand it leads to restricted surface option, only pre-defined elements can be used and the outlook is also pre-defined (though if SAP NetWeaver Portal is used, the WebDynpro elements can take over the portal design for having same style).

The above Web application development techniques are code based ones. As the new requirements aroused SAP implemented a special, model-driven Web application technique as well. The WebDynpro applications give the opportunity to separate the service development from the real user interface design by using M-V-C design pattern, and meta data for element definitions. The SAP NetWeaver Visual Composer (NW VC) goes forward and gives a fully web based modeling tool for UI definition. By modeling we have an abstract layer of definition, so the result can be technology independent. With new technology only compilation must be done again. It increases the efficiency of the web application because the sustainability of it cost much less than with other techniques. NW VC implements WebDynpro and Adobe Flash capabilities as well. The modeling is lifted to a higher level here, because not only the developer can model, but also the business expert without development knowledge can create web base applications, like with ABAP Query any user can create a report in the classic ABAP environment. The design tool makes it possible to define data flow and event handling using business coherency. The service developer as in usual case defines the building blocks for NW VC and WebDynpros as well. In case of Visual Composer not only the reusable service blocks are predefined, but also the visual elements can be developed separately. This kind of programming model distinguishes between content admin, business expert, business application or service developer, and view developer (J2EE, ABAP or .NET). The main focus is on the business expert, because he is responsible for the business process, and he know it much better than any developer, so the developers just provide the view elements (like table, chart, etc.) and the data retrieval services, and the business experts puts these together by the modeling tool. It speeds up the development, makes the parts reusable and helps to modify the applications quickly according to the changing business needs.

6. How to Create Sustainable UIs

As we worked out the technical capabilities we can declare that the basis for creating sustainable backend functions for UI-s are the well defined RFCs and BAPIs, which can be called as WebServices as well. The main point of the sustainability in time when we have changes in the system is to have encapsulating, remote enable proxy services. These services can build the model level of the M-V-C paradigm to make a reusable and embedding layer for the UI-s. This makes possible not to change anything in the developed UI if any modification occurs behind (like upgrade, service modification), because the only the content of the embedding service should be modified if needed. This guarantees a more sustainable environment, because the end-user does note fill any changes on the UI, even if the back-end functionality is modified.

The opposite side also remains unmodified against the changes, if we consider a service or functionality on the back-end side stable, but we switch from a UI option to another one. As we learned above the almost all internal, standard SAP UIs can use any Workbench object, which provides functionality as model. But we can use remote-enabled function modules as well to implement model functionalities to guarantee the independence from the UIs (even external UIs). If we think about not RFCs, the standard Dynpro, ITS EWT, BSP or WebDynpro can be used as well. If we want to switch from one to the other the control and view layers should be redesigned, but the model can stay as it is. Using RFCs we have the opportunity to switch to ITS

FlowLogic or use separated Web surface, which can be an external one or another SAP Web AS based UI (like ABAP or Java WebDynpro). This changeability on UI side leads us beyond the UIs, because it tries to uncover the power of design and development methodologies advised for sustainable and maintainable solution in a changing environment [9-12]. These innovation for development strategies help us the generate better change management in a heterogeneous system landscape as well.

The Visual Composer would be a great tool but it needs NetWeaver Portal as basis element, which is not always offered at an implementation.

7. Conclusion

SAP and Microsoft ERP and other platform-based solution have many options to build a well-defined UI surface using difference data source and storage. Both solutions can provide services to consume by other application. With this property it is easy to create in a heterogeneous environment smoothly embedded new transactions, applications, which use services from other solutions for back-end functionalities. SAP can offer RFCs, BAPIs and Web Services for external use, so MS AX is a good candidate to profit from this offering for its developed UIs or UI extensions. The available RFCs and BAPIs services can be consumed only if the SAP NCo (SAP Connector for Microsoft .NET) is available for the developer. If no SAP connector is applied, Web Services (embedded RFCs and BAPIs as well) offered by SAP are consumable by AX.

SAP is not prepared for consuming services offered on any protocols. SAP can consume only provided Web Services and RFCs (and some other specialties not mentioned here). Definitely RFCs are provided mainly by SAP systems, but if a corresponding connector is in use, other applications, programs can also provide remotely callable functions for SAP. MS AX generally provides Web Services, which can be consumed by SAP applications based on SAP Web AS. But using the SAP NCo for Microsoft AX, not only Web Services, but also RFCs can be provided as well. When SAP consumes a Web Service or RFC provided by MS AX, different UIs, even standard ABAP Dnypro (screen based transaction) or WebDynpro can be based on them.

The interoperability and the service oriented new architectures and techniques make it easier to create application-wide transactions using UIs required. These UIs are maintainable and sustainable if the services behind are embedded to proxy classes or internal services.

References

[1] Sík-Lányi, C., Brown, J. D., Standen, P., Lewis, J., Butkute, V.: *Results of User Interface Evaluation of Serious Games for Students with Intellectual Disability*, Acta Polytechnica Hungarica vol. 9, no. 1, pp. 225-245, 2012

[2] Ősz, R., Pálmai, O.,: *A képernyő az interkulturális folyamatokban,* XI. Dunaújvárosi Nemzetközi Alkalmazott Nyelvészeti és Kommunikációs Konferencia, 2009

[3] Mátrai, R., Kosztyán, Zs.: *A New Method for the Characterization of the Perspicuity of User Interfaces,* Acta Polytechnica Hungarica vol. 9, no. 1, pp.139-156, 2012

[4] Barbaric, I.: *Design Patterns in Object-Oriented ABAP,* Galileo Press Bonn-Boston, 2010

[5] Heilman, R., Jung, T.: *Next Generation ABAP Development,* Galileo Press Bonn-Boston, 2007

[6] Muka; L., Lencse, G.: *Developing a meta-methodology for efficient simulation of infocommunication systems and related processes,* Infocommunications Journal, vol. LXIII, no. 7. pp. 9-14, 2008

[7] Muka, L., Lencse, G.: *Cooperating Modelling Methods for Performance Evaluation of Interconnected Infocommunication and Business Process Systems,* Proceedings of the 2008 European Simulation and Modelling Conference, 2008

[8] Muka, L., Benkő, B. K.: *Meta-level performance management of simulation: The problem context retrieval approach*, Periodica Polytechnica vol. 55, no. 1-2, pp. 53-64, 2011
DOI: 10.3311/pp.ee.2011-1-2.06

[9] Fodor, J., Ősz R.: *Possible applications of fuzzy methodology in the educational process,* IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 37-40, 2013
DOI: 10.1109/SAMI.2013.6480992

[10] Fodor, J., Ősz, R.: *Possible connecting areas of education and intelligent systems,* 2013 IEEE 9th International Conference on Computational Cybernetics (ICCC), pp. 51 – 56, 2013
DOI: 10.1109/ICCCyb.2013.6617560

[11] Ősz, R.: *Educational organization for new generation,* SAMI 2012, IEEE 10th Jubilee International Symposium on Applied Machine Intelligence and Informatics, Herl'any (Szlovákia) Conference Proceedings, pp. 373 – 375, 2012
DOI: 10.1109/SAMI.2012.6208992

[12] Ősz, R.*: New Technologies Mean New Methods of Learning?,* in Recent Advances in Modern Educational Technologies (editors: Hamido Fujita, Jun Sasaki), Iwate, WSEAS Press, ISBN:978-1-61804-180-7, pp. 59-64, 2013