Distribution of Human-Machine Interfaces in System-of-Systems Engineering

Sandro Leuchter and Dirk Mühlenberg

Fraunhofer Institute for Information and Data Processing (IITB)
Fraunhoferstr. 1, 76131 Karlsruhe, Germany
{sandro.leuchter,dirk.muehlenberg}@iitb.fraunhofer.de

Abstract. System-of-systems integration requires sharing of data, algorithms, user authorization/authentication, and user interfaces between independent systems. While SOA promises to solve the first issues the latter is still open. Within an experimental prototype for a distributed information system we have tested different methods to share not only the algorithmics and data of services but also their user interface. The experimental prototype consists of nodes providing services within process portals and nodes realizing services with software agents. Some of the services were extended with WSRP (web service remote portlet) to provide their own user interface components that can be transmitted between separated containers and application servers. Interoperability tests were conducted on JBoss and BEA Portal Workshop. Open questions remain on how the layout of one component should influence the internal layout of other GUI-components displayed concurrently. Former work on user interface management systems could improve todays tools in that respect.

Keywords: HMI, portlet, wsrp, web clipping, interoperability.

1 Preface and Problem Statement

Generally integration of software systems means to use functions or data from one system in the other. Enterprise application integration (EAI) deals with frameworks and middleware to integrate business data sources of different software systems. Another important field of data integration is how to deal with different user login identities and access rights. The problem is how to identify users from one system for the other (authentication) and how to map their access rights (authorization). Solutions have to address not only the technical side but also organizational issues. The third important field of system integration is how to distribute functionality between integrated systems. To achieve this functions have to be identified that are to be shared between systems. The system that provides such a function has to be extended with a new interface that can be accessed by function consumers over a network. The current technological approach is using a communication middleware like an enterprise service bus with different adaptors as interfaces to the services providers (e.g. via web services). A further area in integration is discovery of such services.

M.J. Smith and G. Salvendy (Eds.): Human Interface, Part I, HCII 2009, LNCS 5617, pp. 271–278, 2009.

It remains open how to interactively use services by consumers: Normally a service provides a specified functionality that is accessed via a certain interface over which parameters are given to the underlying process. The nature and semantics of the parameters depend on the service. But how to specify the parameters interactively remains to the consumer that has to build input fields in it own user interface. From an engineering point of view it is not useful to share functionality but not user interface elements. Thus there are some different methods emerging how to share user interface elements besides business data, user data, and functionality. In the following section web service remote portlets, web clipping and smart clients are presented.

2 Methods for Sharing Human-Machine-Interfaces

2.1 Technological Approach

In a context of a network of SOA nodes (like in a System of systems ad-hoc approach) services may be exchanged on a simple syntactic contract base i.e. by exchanging W3C web service signatures via WSDL (web service description language) files for integration in a business process coded in BPEL4WS (business process execution language for web services). This approach lacks any (semantic) information on how to use the selected Web service, the responsibility for the correct usage of the service falls to the consumer of the remote service. If the service is simply an endpoint for a deployed BPEL process, the advertisement via WSDL is enough for a correct operation. But in an end user scenario for a real business process in a big enterprise portal, the user needs the semantics of the service usage to avoid erroneous or useless operation. The requirement for usage semantics covers the range of provision of simple constraints on input parameters like valid numerical ranges over the more elaborated networked dependencies between parameters - in case of input space of the service is not normalized – to the necessity of user guidance through a wizard – in case of input parameter sets, which need a deep understanding of the sequence of parameter input and the appropriate expert knowledge for parameterising the underlying process like an automatic target recognition.

The lack of usage semantics may be easily remedied by not simply supplying only input / output data, but annotating the method's signature by usage handles in the form of declarative or real user graphical elements. This approach of UI (user interface) surfacing not only improves end user operation, but also avoids error prone code duplication of UI element generation in each portal, and makes the integration problem of portal fragments not a programming but a real management task performed by the administrator of the consuming portal. An additional essential advantage of this responsibility shift is, that the administrator is the one role which manages user authorization and authentication information, thus the management of the portal fragments goes hand in hand with SSO and access right configurations and management. Three roles are identified in the context of surfacing:

- the producer owns the service of interest and hosts it via a network-enabled protocol
- the consumer accesses the web service and provides the UI to its registered client

– the client accesses the consumer, which is a proxy to the UI of the service on the producer, displays the graphical elements and relays user interaction to the proxy.

The concept of UI surfacing is technically realized in different approaches, from which the most common are presented here in the following three sub-chapters.

2.2 WSRP

WSRP is the acronym for Web services for remote portlets, which summarizes the technology blend realizing this approach. In fact, reviewing the conceptual needs for UI surfacing, this solution implements the idea of consolidation of service data and the required GUI elements in a smart and direct fashion. The main idea is, to bring existing industry standards together to realize a new standard which fits seamlessly into existing portal and SOA technologies. In 2003, the OASIS [1] organization published their version 1.0 of the WSRP standard, which is adopted by all relevant portal vendors. One of the crucial factors was, that the main technical contributors (participants like Microsoft, IBM, ORACLE, …) tested their implementations from the beginning against each other, to assure the WSRP interoperability between portals based on different portal vendors (see Interoperability SC on OASIS website). Two technology base concepts, namely portlets and common W3C web service stack were brought together to constitute a new web service interface which published not only the data, but the complete graphical user interface in the context of portal fragments (named parts or portlets). WSRP service descriptions are published by standard WSRP files, but instead of coding the functional signature of the underlying process, the WSRP WSDL contains a series of well-defined technical service endpoints which realizes the WSRP framework. From these services two a mandatory:

– Self-description: this web-service allows the consumer to reflect the producer's capabilities and the portlets hosted on producer site inclusive their meta data,
– Access to HTML markup: this service allows the consumer to access the markup of the portlet running on a selected producer.

There are two optional service ports which expand the functionality of the consumed portlet:

– Registration: instantiates a binding between producer and consumer for accounting purposes or auditing, this binding allows the consumer to parameterise attributes of a portlet,
– Portlet management: this service gives access to the life cycle management of a portlet and some persistent state saving.

Each vendor has the possibility to publish and implement additional services, which are only significant between portals of this vendor.

The portlet specifications (JSR-168, 286) are understandable as an extension of the Java servlet specification (JSR-154), which in fact realizes rectangular non-overlapping areas in a standard web page. They are visualized as discrete windows of independent mini applications, this technique is preliminary intended for visualizing a bulk of diverse data in a compact and dense manner (like graphical charts or tables for a stock exchange page). A portal page based on portlets aggregates the data by a compact view through many small windows. Nevertheless, portlets may exchange

data or events by vendor-specific extensions, if this is necessary. Portlets may be implemented in any language, which is supported and understandable by the portlet container and the underlying web (or application) server, the UI elements may be coded in simple HTML up to elaborated usage of JSF (java server faces) or dynamic features like AJAX (asynchronous Javascript and XML).

With portlets as the main structuring building blocks of a portal page, one is able to aggregate an application by simply arranging the desired mini applications on the page and configuring the attributes of these portlets. This task may be performed by an administrator for main portal pages and their core applications, but also by registered users on their own pages (so called dashboards or community pages with accompanying user spaces). This is a dynamic approach, each addition, removing, rearranging and configuring of such pages is done during normal server uptime. To achieve such a dynamics, it is vital to provide a framework by the portal software, which allows these different modifications during runtime. These frameworks are vendor-specific with proprietary APIs, which are generally hidden by comfortable tools in the management workspace of the administrator or the community tools for the dashboard handling by normal portal community members.

2.3 Web Clipping

The web clipping concept stems from the need to display filtered Web content on small mobile devices like PDA (first implemented in Palm OS 3.5) to avoid overloading these crippled devices by the rich internet content designed mainly for the more powerful desktop computers. The main idea is to filter heavy static data from the overburdened internet pages (images, banner, videos, big audio streams), then fit these information blocks to the capabilities of the device, to cache the adjusted data only once for a page and to update only dynamic data during online time. The major difference to programmable web filter is, that the filter intelligence is generally hosted on a proxy server or at least a separate process, with which the client (some browser) interacts over an eventually proprietary (i.e. in the case of the Palm) protocol. This technique is adopted to some portal implementations and stand-alone frameworks (like Kapow RoboSuite Web Integration Platform [2]), thus enabling standard desktop web application to clip from existing remote portal pages. The main motivation in the context of desktop applications is the same as in the case of WSRP, to easily aggregate own portal pages by leaning some clipped content (and thus the underlying applications) from remote portals (known under the keyword “enterprise mashup via presentation level integration”).

Summarizing the existing implementations in the enterprise context, one finds three solutions to integrate web clipping in existing portal software:

- through specialized (web clip enabled) portlets (like portlet bridge [3])
- through browser extensions (like google notebook [4])
- as a remote service (like openkapow with a web clip robot [5])

A similar technique to web clipping is web scraping or harvesting (through web crawler) with the main motivation to focus, filter or sample information into a new often more condensed form.

One noteworthy aspect is the notice of legal issues which results in access or copy restrictions due to (often only printed) copyrights of the source pages.

2.4 Smart Clients

The third variant of sharing human-machine interfaces (HMI) presented here is a technique, which is not restricted to Web content. The metaphor of smart clients results from a very new technology blend of the frameworks Spring (J2EE abstraction [6]) and OSGI (open services gateway initiative [7]). These technologies brought together allow for a completely new client-server concept named as smart or rich client in contrast to thin and fat client known from the web respective swing fraction. The main technical advantage stems from the symmetrical conceptualization of using the same interfaces and data structures on client and server side; this symmetry allows a more dynamic assignment of responsibilities of the functional logic building blocks between server and client along the actual needs or constraints (like network availability) without changing one source code line. One possible realisation of this approach is available in the context of the Eclipse Server-Side framework (Rich Server Platform – RSP [8], see Fig. 1). Each client type mentioned above has its own pros and contras, but the smart client resembles the "copy & run" paradigm plus the rich responsiveness and comfort of a fat client as its best. One highlight is a smart client may run from a simple memory stick or any other removable media, allowing a sales representative to carry its own secure and actual version of his catalogue browser on a DVD ready for use on the client's site.

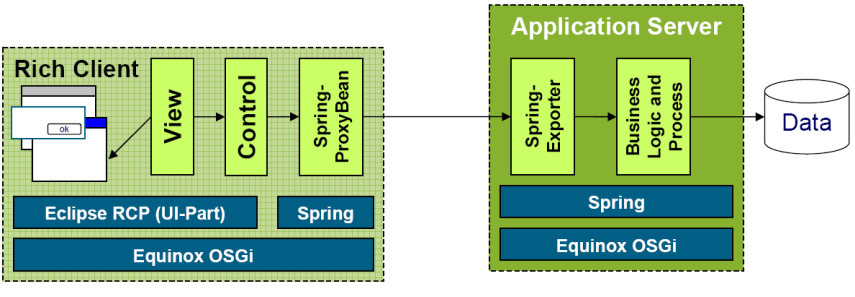

Fig. 1. Possible implementation of RSP [8]

The main disadvantage of this new technology is its unavailability in commercial or open source portal implementations, so actual implementations of a smart client concept have to coexist aside existing enterprise portals on their own platforms using server-side eclipse or the software from compeople [9].

3 Interoperability Tests

3.1 Experimental Setup and Procedure

For the experimental prototype we chose the WSRP framework due to the excellent integration and support into the portal software existing from earlier phases of the project. The OASIS Interoperability sub committee (SC) is the primary source for interoperability questions referring to the portal implementations of the participants in

this SC. In the experimental setup regarding distributed HMI two nodes built on different portal software were tested against each other. The used implementations were the BEA Weblogic Portal 9.2 and the JBoss Portal 2.6.

On the BEA side, there is a desktop with one book containing different pages. The desktop and its content is built solely with the existing management tools of BEA portal and workshop. BEA portal allows for the installation of a community site with accompanying tools, so registered users in this group may manage their own pages. In addition to the vendor provided tools, the prototype contains a special portlet, which allows dynamic adding and removing of portlets on consumer site. This portlet allows the existing prototype portal to aquire new portlets hosted by remote portal nodes.

The setup included a login portlet, the whole desktop is SSO enabled via SAML (Security Assertion Markup Language). The application portlet that was used in the tests is a content based image retrieval (CBIR) application running as a software agent on an agent platform (JADE). The portlet has access to the user's filesystem (user space), which may reside on producer, consumer or client site. The access to the client's data is done via up- or download from respective to the client's computer, the representation of the user space in a portal is implemented by an enterprise bean, which is a façade to a virtual filesystem residing on the bean's host node. The user space allows the delivery of data from one step in a workflow to the next one.

The application portlet is implemented as a pageflow portlet, which realizes a strict separation between control (Java) and view (JSP) in a web adopted model-view-controller manner. As a prototype for stress testing the WSRP concept and the different implementations, this portlet is overloaded with many web artifacts, which are candidates for producing problems on the remote consumer site. Some of these features are Javascript, URLs referencing local resources on a portal, different AJAX implementations, a tree representation of the user space and a HTTP file upload, which forces form submission more demanding. The portlet was tested on its producer site in all aspects regarding the functional behaviour and the portlet specific features like personalisation and management to assure correct functionality. The tests on the different consumer sites exposed many obstacles in programming real interoperable portlets.

On the JBoss site, we used the standard unmodified JBoss portal downloaded from jboss.org. This portal comes with a ready to use portal and community site, the only management tasks are to add users and to enable SSO via SAML. Portlets from a remote producer site are easily managed and arranged by the tools of the admin portal or the community customization tools.

4 Results

4.1 JBoss

The CBIR portlet can be displayed in the JBoss portal. The actual site displays the dashboard of user "admin". The portlet has a different visual style due to other style sheets delivered by this portlet container. Beside these visual differences the handling of the portlet content and the portlet controls is totally alike. One aspect which can be a show stopper is the fact that the registration of a consumer with a producer on the

BEA side has to provide a special attribute which is BEA specific. The consumer has to know it per se, or to avoid erroneous registration cycles, one can enforce a so called out-band registration on producer side with a protocol which handles in its first step an agreement about the next registration step. This agreement is achieved in any communication context, which may be a phone call or a simple letter.

4.2 BEA Portal Workshop

The first tests with the application and management portlet were done on two BEA Weblogic portal implementations to discover implementation issues regarding the WSRP concept in a homogeneous environment related to vendor specific incompatibilities.

5 Discussion

Open questions remain on how the layout of one component should influence the internal layout of other GUI-components displayed concurrently. A working mechanism for notification of a layout rearrangement of another portlet to accommodate the own data layout is the event messaging for portlets. The WSRP V1.0 specification makes no proposition, how to implement such a event mechanism for portlets, so this feature is vendor-specific, a standardisation is scheduled for the WSRP 2.0 paper, which is not yet released, not even as a draft. The work on standardisation of inter-portlet communication is done by the WSRP Cross portlet coordination SC. A generalisation of the event paradigm between arbitrary components of one portlet with one of another portlet is not available, the set of event types to react on is restricted to portlet mode or window state changes, the actions then raised are restricted to mode or state changes, page activation or a generic user event action. Events may be accompanied by a payload, which is simply user defined data. Thus the influence of a state change of an arbitrary element in one portlet is not easily communicable to another portlet, not with the supplied event framework. The proprietary implementations of events permit the usage of such a feature in the WSRP context.

One of the next steps in the project workflow is the inclusion of the Web clipping technique. This approach allows a reuse of nearly every part of an existing web page in a remote portal, not only the reuse or dissemination of portlets. First of all we have to test the possible integration types, portlets as a vehicle for web clipping seem to have much restrictions regarding the clipping functionality.

One major question in using distributed HMI components or even a simple web service is where to find the appropriate service for the user's problem. The retrieval of a service matching the user's requirements is the precondition for the overall system acceptance by the user. To achieve a successful matching, the system must provide a registry for services (directory service, yellow pages) and a convenient information model for the service and its capabilities. The information model is the publish-find-bind abstract model of WSRP. It states in its own data structure named "businessEntity" how to publish portlets and producers as own services in the registry. In W3C web service context a UDDI (Universal Description, Discovery and Integration) is responsible for managing the registrations and responses to search requests in the

context of the installed service model (the so called tModel). Version 1.1 of WSRP includes the concept of publishing the WSRP WSDL files to UDDI to manage the advertised WSRP service descriptions in a network-enabled repository. The information model of producers and portlets in the service model is very sparse and restricted to direct properties of the modelled components. Meta data is only accessible indirect via the service description web service. There is a potential need to enhance the model with semantic annotations to achieve an appropriate level for quality of service analogous to normal W3C web services lacking any semantic information model.

6 Conclusion

In this paper we have presented a study on distributed human-machine interfaces. We used a portlet approach to integrate not only functionality but also parts of graphical user interfaces over a SOA. Interoperability tests suggest that this standard is a promising way to integrate interactive software systems over a SOA. Depending on the nature of the applications to be integrated web clipping is also a relevant standard.

References

1. Internet portal of the Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org
2. Internet page Kapow RoboSuite Web Integration Platform, http://www.kapowtech.com
3. Internet page Portlet Bridge, http://www.protletbridge.org
4. Internet page Google Notebook, http://www.google.com/notebook
5. Internet page Openkapow, http://www.openkapow.com
6. Internet page Spring Framework, http://springframework.org
7. Internet page of OSGi Alliance, http://www.osgi.org
8. Internet page Rich Server Platform, http://www.infonoia.com/en/rich_server_platform.jsp
9. Internet page Compeople, http://www.compeople.de