A portal-based approach for user-centric legacy application integration in collaborative environments

Oliver Gmelch
Dept. of Information Systems
University of Regensburg
Germany
oliver.gmelch@wiwi.uni-regensburg.de

Günther Pernul
Dept. of Information Systems
University of Regensburg
Germany
guenther.pernul@wiwi.uni-regensburg.de

ABSTRACT

"Networked enterprises" are characterized by distributed teams of partner organizations, humans, computer applications, autonomous robots, and devices collaborating with each other in order to achieve higher productivity and to collaborate in joint projects or produce joint products that would have been impossible to develop without the contributions of multiple collaborators. In networked enterprises, special consideration must be paid to the IT systems which are in the position to integrate different applications across company boundaries as known from enterprise application integration. At the same time, high requirements are imposed on the employees within such alliances. The contribution of this paper is an architecture for legacy application integration in web-based portal systems, specifically tailored to the requirements of networked enterprises and focusing on a user-centric approach, allowing the user to customize his workspace to his own needs. Following its presentation, the proposed architecture is validated by a prototypical implementation.

1. INTRODUCTION

Globalization, shorter innovation cycles or increased competition are just a few of the challenges today's organizations are confronted with. The aggregation into virtual partnerships in order to create flexible and agile business networks between a number of partner organizations having complementary competencies appears as possible remedy to overcome this situation. As diagnosed by recent surveys performed for instance by AT&T [4] or the Gartner group [25], a significant increase in the number of business alliances can be expected in the near future.

Of major importance in virtual enterprises is the support by dedicated information and communication technology (ICT). Identified as a crucial characeristic of virtual partnerships by several authors (for instance, Katzy [21]), ICT systems are expected to provide support for different success factors such as integration on process level across company boundaries or information integration, implying provisioning the right information to the responsible user at the right time. As proposed by Shilakes et al. [28], one way of achieving this information integration is via the introduction of specifically-tailored enterprise portals, supporting the integration of different individual applications (even from different alliance partners or from dedicated application providers) in one common user interface consisting of a number of specifically-crafted portlets. It is the intention of these specific portlets to bridge the gap between application boundaries, which create the relations between individual applications, respective process steps and the users in a statical assignment.

Networked enterprises impose high demands on the parties involved in such collaborations, especially riquiring highly skilled personnel with special focus on their flexibility. This leads to the situation that companies are confronted with a "war for talent" as introduced by McKinsey employees Chambers et al. in 1998 [9]. The "war for talent" describes three qualitative challenges companies are confronted with, beginning with the statement that "[a] complex economy demands more sophisticated talent with global acumen, multicultural fluency, technological literacy, [and] entrepreneurial skills". Furthermore, according to Chambers et al., the emergence of small- to medium-sized companies has lead to a shortage of skilled personnel, as well as an increase in job mobility and thus an increase in job fluctuation.

As a consequence of the war for talent, we consider the employee in form of an IT user to be the origin of all value creation within an enterprise. Hence, this paper proposes a software architecture for a user-centric workplace specifically in the area of application integration with web-based portal systems. This architecture poses the chance to flexibly integrate distinct applications from both in- and outside a company's boundaries into a common web-based user interface and thus overcome the formerly static assignments between applications, tasks, and users.

The introduction of a user-centric workplace also bears the advantage that the user can adjust his workspace to his special needs himself. This is of special concern when talking about different application portfolios in use by different companies, which can make a difference in employees' productivity and the amount of training necessary to bring recently

10^{th} International Conference on Wirtschaftsinformatik,
$16^{th} - 18^{th}$ February 2011, Zurich, Switzerland

employed people up to par. Likewise, the creation of a user-centric working environment can make a significant point to distinguish an employer from the competition merely by its IT systems and the resulting attitude towards their employees. Furthermore, flexibility with regard to the application portfolio in use can help to avoid the phenomenon of vendor lock-in, which is characterized by strong dependency of a company to a software vendor. Not limited to applications within a specific company, the proposed approach can also be applied across company boundaries as long as the goals for all individual tasks dealt with in a networked enterprise are well defined, especially in terms of data formats expected. This bears the advantage that individual competencies can be brought into networked enterprises even better.

One way of bringing this user-centric attitude to life in an enterprise environment could be via the introduction of an application store, where a user could select from a set of preconfigured applications per category. Even though this may not be appropriate nor desirable for all types of applications, it may make a big difference when used carefully in situations where interoperability across various application boundaries can be assured due to reliable import/-export filters or standards-compliant software components. For these settings, the user's preferences in the application store may be remembered and taken into account for his everyday working environment.

This paper is intended to present a software architecture for a user-centric application integration approach alongside a brief overview of a prototypical implementation as carried out within the SPIKE project[1], which is funded under the FP7 programme of the European Union and targets the creation of a secure collaboration platform for process integration of external application systems, specifically tailored for the usage in networked enterprise settings by small- to medium-sized enterprises. As such, it follows two main organizational objectives: (1) Outsourcing parts of the value chain to business partners (and vice versa, offering such parts in form of services) and (2) enabling collaboration between members of participating organizations through ad-hoc created as well as predefined business processes.

The remainder of this paper is structured as follows: following this introduction, section 2 gives an overview on related work. Section 3 then provides an introduction on different means of application integration in order to distinguish this approach from other pre-existing work. Section 4 describes the user-centric approach by providing an architecture for user-centric application integration in portal systems, which is then further explained regarding its prototypical implementation as part of the SPIKE project in section 5; section 6 finally concludes this paper and gives an outlook on future work.

2. RELATED WORK

Collaborative software as introduced in section 1 in most cases is built around a portal system. A rather technical definition of the term portal is given in the Java Portlet Specification JSR 168 [1]. According to this specification, "a portal is a web-based application that – commonly – provides personalization, single-sign-on, content aggregation from different sources and hosts on the presentation layer of information systems. Aggregation is the action of integrating content from different sources within a webpage. A portal may have sophisticated personalization features to provide customized content to users. Portal pages may have different sets of portlets creating content for different users."

From an application developer's perspective, portals consist of several independent web applications, called portlets, which are combined together into one uniform user interface, running under a Java application server. The Java portlet specification JSR 168 defines a standard for individual portlets, thus enabling platform independence of portlets, aiding usage across different application servers and thereby guaranteeing a high degree of interoperability. The nature of portals consisting of independent web applications has paved the way for integrating external applications into web-based portal systems. Its successor, JSR 286 [19], has extended the portlet standard amongst others with the ability to establish communication between individual portlets, which had been requested many times before.

Related to the portlet standards is the concept of web services for remote portlets (WSRP), which has been established by OASIS and initially released in 2003, with a second revision in 2008. WSRP defines ways for portlets to integrate external applications adhering to this standard into a compliant portal [30]. The WSRP standard follows the paradigm of service orientation, defining a SOAP-based interface for data exchange between an external application and the portal system. By returning the intended user interface in HTML format to the portal server where the request originated, a WSRP-capable application can itself define the expected user interface without the need for major transformations and hence be included easily into IT systems of external parties associated within a networked enterprise. Furthermore, the external application can employ the portal's event distribution mechanism to exchange information with other applications in the portal.

Of special interest in this area of portal-based application integration is the PADEM portal reference architecture as originally published by Gurzki and Hinderer [18]. Likewise, issues related to application integration via portals have been identified by Diaz et al. [13, 12], fostering the idea of a switch from traditional content syndication towards portal syndication and proposing an integrated framework for transformation of existing web applications into portal-aware applications. In their proposal, they introduce a model-driven approach to achieving this switch towards portal-aware applications. Bellas et al. [6, 5] have proposed approaches to displaying web applications as portlets, introducing a chain of user-configurable "transformers" in order to perform customized adaption regarding individual applications. They propose an annotation-based approach to allow for automatic transformations. In the same subject, Paz et al. [24] have extended the idea of portlet integration by the introduction of semantic integration based on annotations. In order to achieve this, an approach using annotator portlets, wrapping remote portlets, has been presented by the authors. With regard to security aspects introduced

[1] http://www.spike-project.eu

by inter-portlet communication, recent work has been published on integration of the XACML architecture into portal systems [17].

The idea of user-centric approaches has been examined in a variety of ways including but not limited to software engineering, grid computing or identity management, of which the subject of user-centric privacy considerations is one of the most noteworthy in recent years. User-centric privacy follows the idea to gain the users' trust by enabling the user to control which personal information to pass to which party. Even though the idea of user-centric approaches has been examined in a variety of subjects, little research has been carried out so far on the area of user-centric approaches in the area of enterprise application integration mechanisms, where a broad range of user capabilities can be employed by empowering the user to tailor his workspace to his special needs so that he feels most comfortable and hence productive with.

3. MEANS OF APPLICATION INTEGRATION

Ever since the introduction of data processing in the 1960s, an integrated view on the processed data has been requested. In the decades since, the number of applications holding and processing their single share of the data has increased massively, and so has the need for integration between these individual application silos. The introduction and widespread adoption of the personal computer in the 1980s and 1990s has put additional pressure on an integrated and consistent view on data and the applications processing this data, respectively. Due to this situation, a magnitude of different approaches to cope with this challenge has arisen.

On the level of enterprise IT architecture design, Winter [32] has proposed an application architecture with recommendations for the design and implementation of applications (i.e. by bundling of functionality or by introduction of responsibilities and data usage), ultimately resulting in a reduction of complexity and the introduction of well-defined interfaces and thus an improved ratio of costs for application systems and the respective interfaces. Also on the level of application infrastructure, a pattern-based integration approach has been initiated [2], which is, however, limited to IBM products and hence only applicable to limited extent.

Also on the level of IT architectures, the idea of enterprise application integration (EAI) has been examined for a long time since the 1990s, resulting in a plethora of different definitions and differentiations between various types of integration identified [22]. In a nutshell, the idea of EAI is to introduce a central middleware component, governing all communication between individual applications and hence further reducing the number of interfaces required for all applications to communicate with each other properly. Linthicum [23] and Ruh [27] argue that EAI should allow for unrestricted usage of information and application functionality between all applications within an enterprise. Linthicum, for instance, has identified the following types of EAI approaches [23]:

- *Data-Level EAI*, resulting in integration on the level of different databases,
- *Application-Level EAI*, containing integration on the level of individual application programming interfaces,
- *Method-Level EAI*, meaning the sharing of business logic or methods, and
- *User Interface-Level EAI*, resulting in the integration of various applications on the level of their corresponding user interfaces.

As can be seen, the level of abstraction varies greatly between individual approaches in the context of EAI; a majority of methods, however, is focused on the technical side of EAI. Another method to enable enterprise integration on the technical level is the pattern-based approach as presented by Hohpe and Woolf [20], which provides a system of integration relations in the context of EAI. Including different design patterns and the respective areas of applicability, their approach shows potential solutions to individual challenges on a rather technical yet not product-oriented level. Dealing with the subject on a technical level and limited to individual patterns, however, it remains rather vague when it comes to an overall architecture, supporting networked enterprises and the introduction of a user-centric approach.

Also coming from a technical perspective with special focus on the area of portal systems, Daniel et al. have performed extensive work on the aspects of application integration [10, 11], resulting in the discovery and definition of problems associated with certain technologies and opportunities to overcome their respective weaknesses. Of major interest for Daniel et al. were desktop UI components, components realized via browser plug-ins, web mashups as well as web portals and portlets. One of the major findings in their research is the awareness about the diversity of different UI presentation techniques. For this reason, the authors suggest a standardization of service interfaces for user interface integration to succeed. Furthermore, the authors mention a lack of abstraction in order to establish composition in the context of user interface integration. Even though their works have shown the need for unification on the level of user interface representation, it does not provide new strong points on an architectural integration.

As stated by Lee et al., "enterprise integration means both technical and behavioral integration" [22]. In the field of behavioral integration, we consider the user as one crucial part. None of these approaches, however, have evaluated the user as the center of their respective application integration approaches. With their focus on the technical perspective of application integration, all these concepts fall short in considering the user-implied aspects of EAI, which may enable the user to customize his workplace up to his own needs. For this reason, this paper presents a user-centric approach towards portal-based application integration on user interface level, which is demonstrated in a software architecture overview in the next section. As such, the presented software architecture can be seen in line with the ideas as introduced by EAI, most notably the introduction of a middleware component for inclusion of and brokerage between external applications. Even though the concepts and techniques as followed by EAI are considered by some as a hyped topic which has exceeded its climax [3], the results and general require-

ments presented in this paper can also be adopted to other integrative techniques such as service-oriented architectures.

4. AN ARCHITECTURE FOR APPLICATION INTEGRATION IN PORTAL SYSTEMS

Based on the requirements of networked enterprises as evaluated by the SPIKE project in [7], this paper presents the approach of user-centric application integration. This section focuses on the description of the related architecture. The presentation is split up into two parts: section 4.1 gives an overview on the overall architecture and its elements, whereas section 4.2 presents the followed approach with regard to individual protocol types in more detail. The proposed architecture and its support for a broad range of protocols is considered fundamental for the introduction of a user-centric integration approach in collaborative environments, which allows users to customize application usage to their own needs. Due to the sheer amount of different applications and involved protocols available in the market, the proposed architecture offers support for a multitude of different protocols typically found in enterprise use as outlined in the next sections without, however, making any application-specific adjustments with regard to integration. The presentation of the architecture is implementation-agnostic; for implementation details, please see section 5.

4.1 Overall architecture

As the architecture is intended as collaborative platform for networked enterprises, it is embedded into a surrounding system architecture providing supporting components for networked enterprises, especially in the field of process orientation and user management (which are not presented in detail within this paper due to space limitations). In the following paragraph, however, a brief overview is given on the respective components and their interaction with outside components of the architecture. This architecture can serve as a blueprint for application integration in a collaboration platform such as the SPIKE platform as presented in [15].

The core of the portal architecture as depicted in figure 1 is represented by three components, namely *Intra Portlet Communication Management*, *Session Management*, and *Portal Display Management*. *Inter Portlet Communication Management* is responsible for collecting events from available portlet sources and delivering them to other portlet destinations, including Notification Management for external delivery in other user sessions. Within the portal session, this is performed using the established method of Inter Portlet Communication. *Session Management*, on the other hand, is in charge of user session context handling as well as storage and retrieval of user sessions. To do so, it makes use of the central storage repository as maintained by Content Management. The third component in the portal architecture in figure 1 is *Portal Display Management*. It is the task of this component to provide users with a visual representation of applications, thus carrying out the actual work of user interface integration of external applications. All three components inside the Portal Instance are in contact with *Interface Management*, which forms the interface to the external applications as connected via the integrative platform.

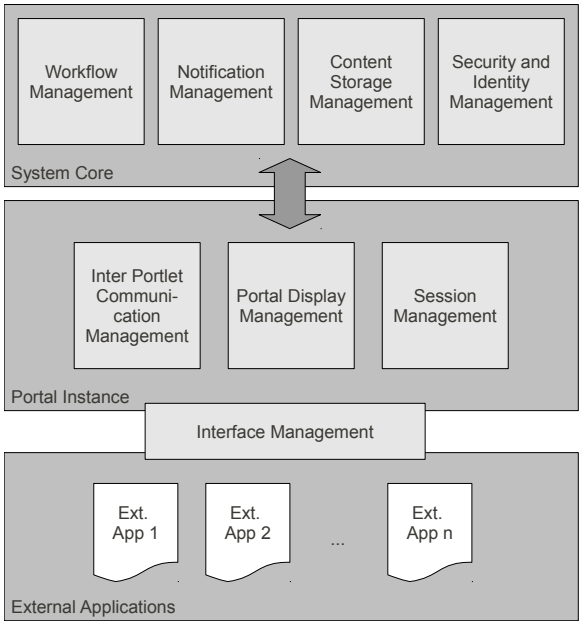

Figure 1: Portal Architecture

Surrounding this portal architecture, a number of other components can be found, which are aggregated in the System Core. First of all, the aforementioned *Content Management* provides a central storage repository, allowing for retrieval, update, and storage of all data present and brokered via the platform.

It is the duty of the *Security and Identity Management* component to provide access decisions based on the user's identity, containing information about the user's home company as well as personal information (i.e., department, email address, etc.) about a specific user. This information is retrieved from the external identity management system associated with a specific user, i.e. provided by his employer and selected and queried during login. Likewise, it has to be assured that only trustworthy users may enter the platform who have been granted the necessary privileges by their employer beforehand. Even more important, it also has to be guaranteed that only associated members of a networked enterprise may enter the system and perform actions with it as this platform is working on potentially highly sensitive information.

Notification Management, as briefly introduced before, is necessary for sending event notifications to the rest of the architecture and to receive events from other parts of the architecture, which are then further processed within the portal. These events can consist of various types, ranging from inter portlet communication between different application instances in the portal to exchange of status messages, containing i.e. information about a new user session or a logout event.

Finally, *Workflow Management* keeps track of all workflows, their respective instances and all associated tasks deployed within an alliance and hence has to maintain communication with the individual user sessions, represented via Session

Management.

Likewise, the correlation between Workflow Management and Session Management allows for inclusion of a user-centric application integration approach, where a user for certain tasks can customize the set of applications being used for task execution up to his own preferences. This can be performed following the widespread idea of an application store, where users can select from a range of applications, differentiated by categories. This way, the user can stick to the type of application he is used to and feels most comfortable with, thus reducing the need for his employer to provide further training in one specific application. On the level of process definition, this requires a concise definition of the goals to achieve for the individual tasks associated in a workflow as well as definition of the applications or data formats that these results can be produced with.

Even though this approach may not be desirable or feasible in all circumstances, enterprises can gain a broader range of options to provide their users with. Likewise, it can be expected that training costs can be reduced when users have to make little or no changes at all to their accustomed working environment whilst ensuring user satisfaction and creating a feeling of affiliation of the employees with their respective employer. This in turn can help to keep personnel fluctuation low and thus ensure customer satisfaction, which successively can help to strengthen the company's position in the market.

4.2 Detailed UI architecture

The major component for UI integration of external applications – as was already described in the previous section – is the Portal Display Management component, which is intended to provide support for a broad range of different application types. As different applications usually are provided by different protocols, the Portal Display Management component is designed to offer a flexible plugin mechanism to allow for integration of different protocol handlers. For an overview on the different types of applications currently supported by the architecture, see table 1. Differentiated by their nature as request- or communication-oriented protocols, the support ranges from HTTP and the SOAP-based WSRP protocol for the case of requested-oriented protocols to support for connection-oriented protocols, whereas further differentiation between text-based (currently, Telnet and SSH are considered here) and GUI-based protocols (where the current state considers RDP and VNC protocols) is made. At this point, it is important to note that this listing is neither complete nor exhaustive, meaning that protocol support is not limited to these protocols on an architectural level, but these protocols have been introduced by the SPIKE project based on the user requirements analysis performed by the project. Moreover, the architecture strives to provide a general, application-independent view at the supported protocols. For the case of HTML content presented later on, due to the large amount of different usage scenarios, three approaches were introduced to support these different scenarios appropriately.

The architecture regarding those individual protocol handlers was designed to follow the paradigm of service-oriented architectures. The services are generally stateless and atomic

Table 1: Currently Considered Protocol Types

	Connection-Oriented Protocols		Request-Oriented Protocols
	Text-based	GUI-based	
Protocols	SSH, Telnet	RDP, VNC	HTTP, WSRP

and grouped by functionality. Common to all protocol handlers is the idea of a proxy mechanism, which allows the protocol handlers to act as an intermediary between the target application and the targeted end user. This proxy mechanism can be implemented in two distinct manners: One approach is that the portal merely is responsible for application connection intialization and hands over a session token representing this connection to the service consumer, which then performs all interaction with the application independently from the portal instance.

Secondly, the portal can be acting as a proxy for the whole user session, meaning that all communication between service provider and -consumer will be routed through the portal. This bears a number of advantages: First of all, firewall policies will have to be adapted by the service provider only once. As all requests from the service provider's point of view will stem from the portal system, access on IP level will have to be granted to a target application only once for an unlimited number of users for this application type. Secondly, acting as a proxy bears the capability to perform transformations on the presentation layer, for instance on CSS/JavaScript level in case of HTML content. Moreover, based on the type of application, the proxy approach can allow for flexible and dynamic extraction of content elements adhering to previously defined rules as implemented during workflow specification phase. This can for instance be performed for the case of Telnet- or HTML-based user interfaces. Furthermore, as the platform provider is aware about all parties associated with its platform, it can make more educated access decisions about whether or not certain kinds of access attempts adhere to the access policy for the specific user or the user's employee.

The major drawback of this approach, however, is that the platform provider acting as the intermediary between service requester and service provider can potentially carry out man-in-the-middle attacks, thus putting all contents exchanged via said platform at risk. This risk therefore requires the platform provider to be a trustworthy third party, relied upon by all participants and entrenched respectively, i.e. by legal means. Furthermore, the aspect of runtime performance must be paid attention to. For the case of full-featured application proxying, a significant increase in the workload of the IT infrastructure is to expected, potentially leading to an increase in costs for the platform provisioning. Likewise, since all accesses are carried out via the portal platform, this imposes higher requirements on the availability of said platform.

As can be seen from figure 2, both types of communication are supported by the architecture. In the case of connection-

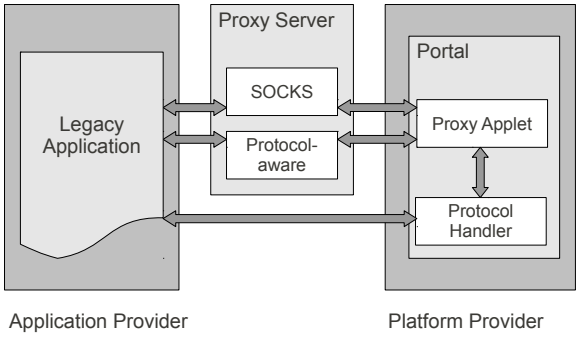

Figure 2: Asynchronous Applications: Architecture Overview

oriented protocols, communication is performed via a tailored applet running inside the portlet of this specific application instance, which can either communicate directly with the connected external application or create the connection via a dedicated proxy server. The latter case can – for future developments – be used to allow for session reliability as the proxy server could be used to maintain connectivity to the respective application even in the case that the browser session crashes or the applet is reloaded unintentionally by the user's browser, allowing the user to seamlessly continue his session at the point it was interrupted. With protocol-agnostic support being propagated by a SOCKS-compliant proxy module, further functionality can be gained by protocol-aware proxy modules. Some details about possible enhancements by such protocol-aware proxy modules are given in section 4.2.2. Also, figure 2 shows that the situation for request-oriented protocols is similar to the approach taken for connection-oriented protocols except for one difference: as markup-based content currently is the only content type supported here, the dedicated applet can be omitted since all web browsers can render markup natively.

The design of the architecture follows a number of software design patterns as originally presented by [16]. First and foremost, the model-2 pattern has widely influenced the design as it presents itself as the foundation for merely all web-based UI interaction, especially in the context of Java applications as is the case for SPIKE. Adopted from the well-known MVC pattern which is considered to be a set of strongly related patterns itself [14], model-2 is specifically tailored for web applications. The MVC pattern separates an application into three distinct parts: (1) the *controller*, connecting the latter two parts of the application with each other, (2) the *model*, responsible for the actual application logic, and (3) the *view*, which is in charge of rendering a visual representation of the application, the so-called user interface which can therefore be considered a visual representation of the model.

Moreover, the *facade pattern* has found major consideration in the design as it enables the support for a pluggable modules system as the architecture defines an interface on an abstract level, the facade, which is then implemented by every protocol plugin. All protocol-specific functionality is hidden by this service facade afterwards, thus making all accesses transparent to the portal system as the portal system merely hands over control to the corresponding plugin, which helps to keep the portal code clean and flexible.

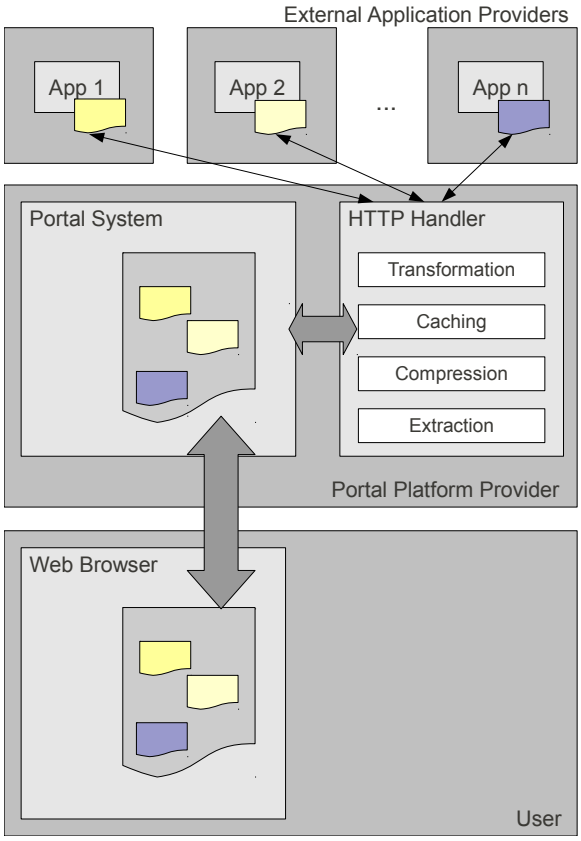

Figure 3: HTTP Handler: Architecture

4.2.1 Support for request-oriented protocols

Based on user requirements analysis, the highest ranked type of protocol to support in the presented architecture is the hypertext transfer protocol (HTTP), which forms the basis for the world wide web in its current form. HTTP Communication generally follows the request-/response paradigm and is not connection-oriented and stateless. Another important protocol taken into account by the software architecture is the WSRP protocol, enabling the integration of markup-based remote applications into a portal system. Due to the flexible nature of the architecture, however, support could be extended for other protocol types appropriately.

HTTP support in the presented software architecture focuses on integration of content provided in the hypertext markup language (HTML) and considers three distinct ways of integration in order to establish flexible yet efficient ways for application integration which are briefly outlined below. In this enumeration, the level of complexity and hence the possibilities of the respective methods increase significantly from top to bottom. Due to the wide nature of predominant HTML-based applications, the best-fitting integration mechanism must be determined on a case-by-case basis.

First of all, integration of HTML content can be carried out in a straight-forward manner using the HTML tag <*iframe*>,

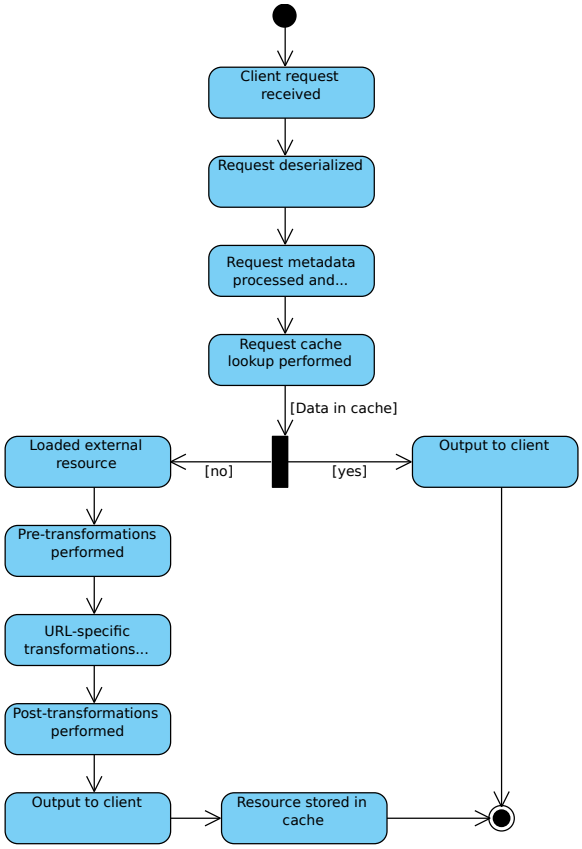

Figure 4: HTTP Handler: State Diagram

which is the most non-intrusive way of integrating an application as it merely passes all content through to the web browser, which renders the application as part of the portal website. Since HTML frames exhibit some limitations in terms of user experience (for instance, the risk of losing the portal session due to external links), this is also the least powerful integration mechanism and should be considered ultima ratio.

Secondly, integration can be carried out by a dedicated applet, which renders content described by markup in a separate applet independent from the end user's web browser and thus can achieve a significantly higher level of unity in application appearance and platform independence. This, however, comes at the price of increased system requirements for the associated client systems due to the separate rendering process and should preferably be considered for applications under direct influence of the application provider.

The third option is to perform integration via a dedicated HTTP handler acting as a reverse proxy and as such performing content rewriting, which allows for direct integration of HTML contents into the portal whilst posing higher browser compatibility requirements on the target application to serve all end users reliably. Due to the heterogeneous nature of HTML documents, this approach presents itself the most complex. Figure 3 shows the underlying architecture: Content from external applications is integrated into the platform using the HTTP handler, altering the content in such way that it properly fits into the portal, where all contents from individual applications are merged and presented to the user in a unified user interface. The HTTP handler acting as a reverse proxy, all content is modified as part of the transformation phase so that all requests in return to content delivered by the HTTP handler module will again invoke the HTTP handler, avoiding unintended side channel data flows. The execution of content rewriting also aids to avoid situations where users of the portal could be leaving the platform unintentionally due to popup windows or other external links. Moreover, the HTTP handler is to provide caching mechanisms in order to speed up connections by storing temporary copies of frequently accessed files as well as image compression capabilities in order to reduce the footprint of images and other large resources, which may be desirable for mobile devices with low-bandwidth network connectivity. Finally, the extraction stage serves to extract information retrieved from an external application based on predefined goals. Figure 4 shows a state diagram of HTTP handler, giving an overview on its individual states when processing a HTTP request. Beginning with a client request, the resource is first looked up in the cache, maintained by the HTTP handler for performance reasons. In case a resource is found, it is directly sent to the client; otherwise, the external resource is retrieved and applied a number of transformations in order to perform the tasks of information extraction and compression as outlined above and finally stored in the cache and sent to the client.

Another type of request-based application integration is gained by supporting the service-oriented paradigm, and more specifically the WSRP standard, allowing portals to include external applications into portal systems both on the data level as well as on the level of the user interface representation. This is achieved by providing WSRP consumer support in the architecture, which allows external application providers to integrate with the proposed solution without any changes to their existing WSRP-aware application.

Figure 5 shows the general flow of information for the case of WSRP application integration. An external application provider makes available an arbitrary number of WSRP-capable applications, each running in a portlet container. The WSRP producer is responsible for embedding the individual markup generated by every application into corresponding SOAP messages, which is then extracted by the respective consumer on the portal side, where the overall portal is rendered, consisting of the combined markup from the external applications plus any portal-specific markup, i.e. for user management, status message display, etc.

4.2.2 Support for connection-oriented protocols

As is the case for request-oriented protocols, the support for connection-oriented protocols in the presented architecture is currently limited to four protocols to demonstrate the feasibility of the approach. These protocols can be grouped into protocols supporting graphical user interfaces (GUI) such as RDP or VNC and text-oriented protocols such as SSH or Telnet. However, the extensibility gained by grouping functionality into separate plugins allows to fulfill later requirements in case of need.

Generally, in the application integration architecture, two

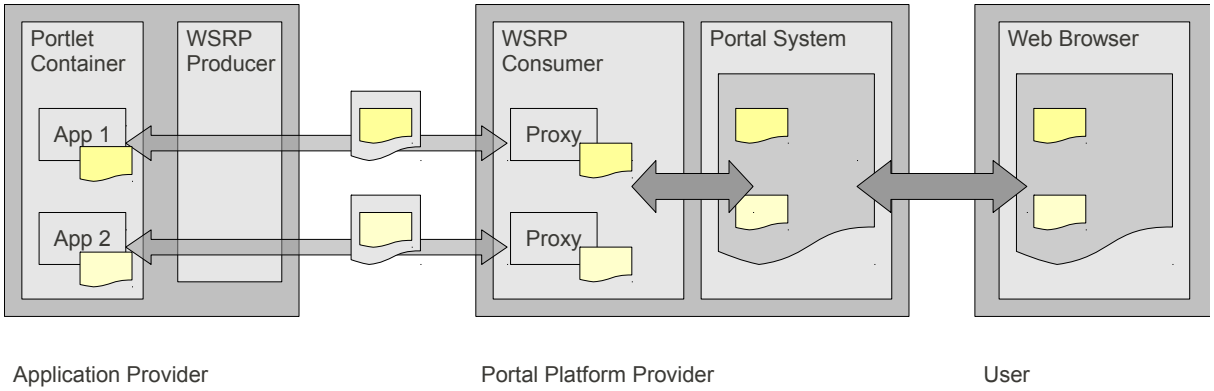

Figure 5: WSRP Support

modes of operation are supported as previously shown in figure 2: with and without intermediary proxy server, whereas operation with a proxy server between application and user can again be differentiated into usage of a SOCKS-compliant proxy server and an application- or protocol-specific proxy server, which can be thought to provide protocol-specific functionality for later requirements. Whilst additional functionality such as information extraction from the user interface can be implemented easily for the cluster of text-based protocols, the situation becomes more difficult when it comes to graphical user interfaces as is the case for instance with RDP and VNC. Depending on the protocol specification, however, additional features of the protocol can be employed. For instance, the RDP protocol foresees capabilities to not only transfer information about the GUI but also offers features such as file system redirection, enabling the integration platform to provide a file system which the external application can store task results on. The level of integration which can be achieved by the architecture is therefore limited by the specifics of the protocol in use.

5. IMPLEMENTATION AND EVALUATION

The software architecture as presented in the previous section has been implemented in order to prove its feasibility. The implementation has demonstrated the general applicability of the proposed solution in the environment of user-centric legacy application integration. While a number of issues has shown up during the implementation, these issues are not related to the overall architecture and hence do not diminish its value.

One major pillar during the implementation was the route of embracing and extending existing open source components to achieve the goals. The rationale behind using the source code of publicly available open source projects for the implementation is manifold: First of all, open source projects have – at least for major players on the market – reached a satisfactory degree of maturity. For instance, analysis of different operating system kernels from both open and closed source products has shown that despite greatly verying processes by which the respective products were developed, the metrics applied to the corresponding source code showed a comparable performance [29]. Likewise, commercial support is available for most major open source projects, meaning that no special knowledge has to be created inside an enterprise prior to using and potentially extending a specific open source project. Beyond that, a number of soft facts have been identified, ranging from customer relationship enhancement to earlier feedback from the community and voluntary actions by the community like localization activities [8]. Finally, available code provided by various open source projects presents a huge wealth of knowledge which has been tested in numerous installations around the world and which has been of great use for the prototypical implementation. For instance, the Liferay portal server, which forms the basis for all portal-related software development within the implementation, has been evaluated to consist of more than 1.7 million lines of code, which has been estimated to result in 502 person years on effort[2]. With the open source movement existing for a few years only, little expertise is available on the long-term performance and development of the open source paradigm. However, recent research seems to indicate that a commercial open source business model, where commercial open source firms build their business around an open source project that they fully control, can prove successful [26]. Even though this approach seems to constrain customers to the vendor of a specific product, there is still the availability of the product's source code that can allow them to continue usage of that product even after the case that the original vendor should no longer exist.

Although implementation of the proposed software architecture has been performed employing open source components, this does not limit the applicability of the underlying software architecture to open source components. Instead, the general architecture is implementation-agnostic, whereas the implementation only serves to prove its validity and has been carried out employing as many existing components as possible for complexity reasons.

For the reasons given above, for all portal developments during the prototypical implementation, the Liferay portal server[3] in its community edition has been chosen to form the basis. Liferay Portal is freely available under an open source license and has been selected mainly for its widespread acceptance, its user community and its extensibility. It is especially the extensibility provided by a so-called extension environment which allowed to create extensions to the por-

[2] `http://www.ohloh.net/p/liferay`
[3] `http://www.liferay.org`

tal's core while not touching the portal server's source code directly and which has therefore proven very useful for future version updates of the portal server underneath.

During the implementation, special emphasis was put on support for existing HTML-based web applications. For the integration of web applications, the three approaches as laid out in section 4.2 were implemented, of which the reverse proxy method has received the most attention due to its complexity and its possibilities. Evaluation has shown afterwards that while it works well for basic standards-compliant websites, the situation changes when non-conforming HTML documents come into play. As the reformatting and information extraction methods require the HTML code to be transformed into XHTML, standards conformance plays a crucial role. Despite not limiting the proposed architecture's value as a prototype, work is ongoing with regard to a more robust parsing of HTML pages.

During the implementation of WSRP support, the separation of duties in producer and consumer has shown to provide valuable support for application developers. Implementation itself has been carried out using the WSRP extension for Liferay portal and has shown that existing portlets can be easily provided as external applications, thus enabling the introduction of external application providers. This also undermindes the idea of networked enterprises where all partners can focus on their key competencies and hence improve the overall competitiveness and outcome of such alliances.

Besides the aforementioned adversities, the implementation has proven the proposed architecture's general applicability in the context of user interface integration, especially following the presented user-centric approach, which allows a user to employ functionalities commonly known from the nowadays prevalent application stores, where the user can choose from a set of available applications to adopt his workspace to his special needs.

6. CONCLUSION AND FUTURE WORK

This paper has presented reasons to perform application integration on a user interface level and has demonstrated an architecture for the inclusion of applications into portal systems. The architecture is designed to allow for a high level of flexibility in order to support future enhancements and to achieve a high degree of maintainability. At the same time, this paper has briefly introduced the prototypical implementation of this architecture, which has demonstrated the feasibility of the approach in an application prototype.

The combination of this architecture with the special requirements of networked enterprises can pose a significant advantage over traditional companies. Networked enterprises generally are characterized by a high degree of flexibility, and so are the requirements on their employees. The introduction of the user-centric attitude into the context of application integration can provide a way for companies to get advantage over their competition in the war for talent by providing their users with the ability to specifically tailor their work environment to their own needs whilst also providing ways to ensure or improve their competitiveness.

The implementation has proved the general applicability of the proposed architecture for the problem setting of interoperability in networked enterprises. At the same time, a number of weak points could be identified by the implementation, first and foremost the lack of standards compliance of many existing web applications. Even though a significant increase in the percentage of standards compliant websites has been detected recently [31], their share still remains at mere 4.1%. Therefore, implementations will have to be made more robust to also support non-standards compliant websites as well. Adding to that the sheer number of different techniques for web content presentation currently prevalent on the market besides HTML-based content like Adobe Flash, Microsoft Silverlight, or Oracle JavaFX-based content, the situation gets even more difficult. This also supports the point of Daniel et al. [11] who pledge for standardization on the UI level. A first step towards a more homogeneous presentation layer may occur due to an increasing acceptance of the HTML standard in its fifth release, which is expected to introduce – amongst others – capabilities for video playback and drag-and-drop facilities but has not been officially released by W3C and WHATWG yet. Also, the HTML 5 standard is likely to include recommendations regarding a user agent's behaviour in case of erroneous documents, which may help to increase the number of standards compliant websites. At the same time, the advent of HTML 5 will further ease the introduction of specific so-called widgets, small applications targeting at a limited set of functionality. In combination with the presented user-centric approach, this is expected to provide valuable extensions whilst reducing complexity by breaking up formerly strict application boundaries at the same time.

Besides improving their implementation to cope with the identified lack of standards compliance, future work of the authors is going to focus primarily on carrying out detailed analysis of the proposed architecture. The architecture will be evaluated based on the prototypical implementation alongside a number of axes. These axes will first of all comprise reliability aspects of individual types of applications in order to assess the degree of maturity of the proposed solution. Furthermore, implementation evaluation will also include detailed analysis of the runtime performance of the proposed solution as well as assessment of further protocol candidates for future implementation. Another field for future work is measuring the appropriateness of the proposed architecture in terms of the currently prevalent topic of cloud computing. With this architecture and networked enterprises as potential users of said applications, company boundaries are converging. As a result, provided that legal issues are considered, the location of an application provider no longer is of importance and hence opens the door for the proposed architecture which allows to include distinct applications into a uniform user interface for the networked enterprise. Consequently, aspects like availability, access times and provisioning times for individual applications become more and more important. Also, this paper primarily focuses on the technical implications, whilst business dimension considerations are subject to future research.

Acknowledgements

The research leading to these results is receiving funding from the European Community's Seventh Framework Programme under grant agreement no. 217098 and from the

European Regional Development Funds (ERDF). The content of this publication is the sole responsibility of the authors and in no way represents the view of the European Commission or its services.

7. REFERENCES

[1] A. Abramski and T. Schaeck. *Java Portlet Specification Version 1.0*. Java Community Process, 2004.

[2] J. Adams and S. Koushik. *Patterns for E-Business – A Strategy for Reuse*. IBM Press, 2001.

[3] S. Aier and J. Schelp. EAI und SOA–Was bleibt nach dem Hype? In *Proceedings der Multikonferenz Wirtschaftsinformatik*, pages 1469–1480, 2008.

[4] AT&T Corp. Collaboration across borders. `http://www.corp.att.com/emea/docs/s5_collaboration_eng.pdf`, 2008.

[5] F. Bellas, I. Paz, A. Pan, and O. Díaz. *Handbook of Research on Web Information Systems Quality*, chapter New Approaches to Portletization of Web Applications, pages 270–285. Idea Group Inc, 2008.

[6] F. Bellas, I. Paz, A. Pan, Óscar Díaz, V. Carneiro, and F. Cacheda. An automatic approach to displaying web applications as portlets. In *Distributed Computing and Internet Technology*, volume 4317/2006, pages 264–277. Springer Berlin / Heidelberg, 2006.

[7] C. Broser, C. Fritsch, O. Gmelch, G. Pernul, R. Schillinger, and S. Wiesbeck. Analysing Requirements for Virtual Business Alliances – The Case of SPIKE. In *Digital Business*, volume 21 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 35–44. Springer Berlin Heidelberg, 2010.

[8] E. Capra, C. Francalanci, and F. Merlo. An empirical study on the relationship between software design quality, development effort and governance in open source projects. *IEEE Transactions on Software Engineering*, 34:765–782, 2008.

[9] E. Chambers, M. Foulon, H. Handfield-Jones, S. Hankin, and E. Michaels III. The war for talent. *The McKinsey Quarterly*, 1(3), 1998.

[10] E. Daniel and J. Ward. Enterprise portals: Adressing the organisational and individual perspectives of information systems. In *Proceedings of the Thirteenth European Conference on Information Systems*, 2005.

[11] F. Daniel, J. Yu, B. Benatallah, F. Casati, M. Matera, and R. Saint-Paul. Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing*, 11:59–66, 2007.

[12] O. Díaz, A. Irastorza, J. S. Cuadrado, and L. M. Alonso. From page-centric to portlet-centric Web development: Easing the transition using MDD. *Information and Software Technology*, 50(12):1210 – 1231, 2008.

[13] O. Díaz and I. Paz. Turning web applications into portlets: Raising the issues. *Symposium on Applications and the Internet (SAINT'05)*, pages 31–37, 2005.

[14] E. Freeman, E. Freeman, B. Bates, K. Sierra, and M. Loukides. *Head First Design Patterns*. O'Reilly Media, Inc., 2004.

[15] K. Furdík. Secure Process-oriented Infrastructure for Networked Enterprises. In *Workshop on Data Analysis WDA'2009*, volume 2, pages 98–105. 2009.

[16] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series, 1995.

[17] O. Gmelch and G. Pernul. Preventing malicious Portlets from Communicating and Intercepting in Collaboration Portals. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*, 2010.

[18] T. Gurzki and H. Hinderer. Eine Referenzarchitektur für Software zur Realisierung von Unternehmensportalen. In U. Reimer, A. Abecker, S. Staab, and G. Stumme, editors, *Professionelles Wissensmanagement - Erfahrungen und Visionen*, GI-Edition - Lecture Notes in Informations (LNI). Bonner Köllen Verlag, 2003.

[19] S. Hepper. *JSR 286: Java Portlet Specification Version 2.0*. Java Community Process, 2008.

[20] G. Hohpe and B. Woolf. *Enterprise Integration Patterns*. Pearson Education, 2004.

[21] B. R. Katzy. Design and implementation of virtual organizations. In *Proceedings of the Thirty-First Hawaii International Conference on System Sciences (HICSS)*, volume 4, pages 142–151, Los Alamitos, CA, USA, 1998. IEEE Computer Society.

[22] J. Lee, K. Siau, and S. Hong. Enterprise Integration with ERP and EAI. *Communications of the ACM*, 46(2):54–60, 2003.

[23] D. S. Linthicum. *Enterprise Application Integration*. Addison-Wesley Information Technology Series. Addison-Wesley, 2000.

[24] I. Paz, O. Díaz, R. Baumgartner, and S. F. Anzuola. Semantically integrating portlets in portals through annotation. In *Web Information Systems - WISE 2006*, pages 436–447. Springer Berlin / Heidelberg, 2006.

[25] C. Pettey. Gartner Says Worldwide Web Conference and Team Collaboration Software Markets Will Reach $2.8 Billion in 2010. `https://www.gartner.com/it/page.jsp?id=507717`, June 2007.

[26] D. Riehle. The commercial open source business model. In *AMCIS 2009 Proceedings*, 2009.

[27] W. A. Ruh, F. X. Maginnis, and W. J. Brown. *Enterprise Application Integration*. John Wiley & Sons, Inc. New York, 2001.

[28] C. C. Shilakes and J. Tylman. Enterprise information portals. *Merril Lynch*, November 1998.

[29] D. Spinellis, G. Gousios, V. Karakoidas, P. Louridas, P. J. Adams, I. Samoladas, and I. Stamelos. Evaluating the quality of open source software. *Electronic Notes in Theoretical Computer Science*, 233:5 – 28, 2009.

[30] R. Thompson. *Web Services for Remote Portlets Specification v2.0*. OASIS, 2008.

[31] B. Wilson. MAMA: What is the Web made of? online publication at `http://dev.opera.com/articles/view/mama/`,

October 2008.

[32] R. Winter. Ein Modell zur Visualisierung der Anwendungslandschaft als Grundlage der Informationssystem-Architekturplanung. In J. Schelp and R. Winter, editors, *Integrationsmanagement*, pages 1–29. Springer Berlin / Heidelberg, 2006.