A Generic Architecture for User-Centric Portlet Integration

Oliver Gmelch
Department of Information Systems
University of Regensburg
Regensburg, Germany
Email: oliver.gmelch@wiwi.uni-regensburg.de

Günther Pernul
Department of Information Systems
University of Regensburg
Regensburg, Germany
Email: guenther.pernul@wiwi.uni-regensburg.de

***Abstract*—Numerous attempts have been made towards achieving flexible integration of external applications into a coherent enterprise architecture. The topic of "networked enterprises" describes flexible organisations characterised by distributed teams of partner companies, humans, computer applications, autonomous robots, and devices collaborating with each other in order to achieve higher productivity and to cooperate in joint projects or produce joint products that would have been impossible to develop without the contributions of multiple collaborators. In this context, flexible and secure integration of external services that can be adjusted during runtime becomes of crucial importance in order to quickly respond to changing market needs. In this paper, an architecture for dynamic application integration in web-based portal systems is presented which focuses on the user as the centre of consideration. Following its presentation, the proposed architecture is validated as part of a large-scale prototype.**

I. Introduction

Today, the key to economic success is flexible and fast reaction to market opportunities and changes. Considering the increase in competition due to the world-wide availability of information and the resulting world-wide comparability between different service offerings, enterprises are confronted with major threats due to new players on the market which might even outperform established businesses by imitating their business models. Besides these globalisation challenges, enterprises today are further confronted with a constant change in their environments imposed by shorter innovation cycles or increased competition.

One possible solution to overcome these hindering factors is to collaborate with other companies in different networks such as proposed for example by Ebers [1]. Of the three distinct approaches identified by Ebers [1], this paper focuses on the aspect of inter-organisational networks that are characterised by short setup times and short time to market in order to achieve innovative products emerging from the cooperation between different actors. These inter-organisational networks can form so-called networked enterprises which require a significant amount of flexibility from all alliance members. Membership in networked enterprises appears especially tempting for small- to medium-sized enterprises (SME) since their participation empowers them to compete with larger organisations or become attractive for customers who seek to approach larger organisations primarily [2, p. 36].

Consider a network of associated enterprises partnering in a joint project where several employees from the different firms are assigned to work for the joint project. Within the cooperation, the partner firms share common resources such as knowledge or software applications. As all partners this way contribute to the cooperation with their respective key competencies, previously unthinkable business opportunities can be reached out for, hence strengthening the market position of all cooperation members.

Situated in the context of networked enterprises, the main research contribution of this work is to demonstrate how portal systems, acting as intermediary between providers and consumers of services, can be embedded into networked enterprises by providing seamless access to all relevant information. To achieve this, this work presents a generic architecture for comprehensive portlet integration as its main contribution. This architecture is based on extensive user requirements analysis which originally involved more than one hundred user requirements. To prove its applicability, this architecture has been realised in a software demonstrator after its definition to assess its feasibility with regard to the goals outlined beforehand. Furthermore, extensive evaluation has been performed on this prototype, comparing use cases and user requirements outlined prior to implementation with the original goals. Embedded in a large-scale research project, implementation and evaluation have been carried out as part of the EU-funded research project *SPIKE*[1] involving eight different partners from different backgrounds. Among these partners, major industry partners contributed to the presented architecture with manifold practical experience.

The remainder of this paper is structured as follows: While section 2 addresses relevant related work in the field, section 3 presents the methodology followed for architecture definition and gives an overview on the resulting architecture, detailed in section 4 via dedicated viewpoints on the architecture. Section 5 shows evaluation results. Section 6 concludes this paper.

II. Related Work

Reference models and their representation are a lively researched subject. Governor et al. [3] describe reference

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

architectures as "a generic and somewhat abstract blueprint-type view of a system that includes the system's major components, the relationship among them, and the externally visible properties of those components". The authors further state that reference architectures are not specifically tailored for a particular usage scenario but rather serve as a starting point that can be specialised to fit relevant requirements. Based on this definition, this paper presents a generic reference architecture for the demands of networked enterprises with a special focus on service integration employing the concepts of Enterprise Application Integration (EAI).

Irani et al. [4] have introduced a taxonomy for EAI, arguing that there is a lack of common terminology in the topic of information systems integration. According to Irani et al., Grimson et al. [5] have suggested that the term EAI is merely limited to the integration of ERP systems. At the other end of the scale, Irani et al. mention Zahavi [6] who suggests that EAI covers both enterprise and cross-enterprise application integration. This idea is also postulated by Hasselbring [7]. The notion of cross-enterprise application integration also corresponds to the idea of networked enterprises and heterogeneous systems which build the original motivation for the presented research effort. Due to their diverse nature and hence an increased risk of configuration errors, Schneider and Birman consider heterogeneous systems as less secure than homogeneous systems without additional efforts [8]. When set up properly, however, Williams et al. even see a gain in security when used in conjunction with virtualisation techniques [9] in order to decently separate different applications from each other.

In the domain of EAI, mainly two different approaches can be differentiated: While *architecture-focused integration approaches* such as proposed by Winter [10], IBM [11], Lutz [12] or the Object Management Group (OMG) [13] address integration more on an architectural level, more technically oriented approaches such as proposed by Linthicum [14], [15], [16] or Hohpe et al. [17] focus on the technical implementation details of EAI projects. In contrast to these approaches, this paper presents a dynamic and user-centric mechanism for integration of external services in order to support the needs of networked enterprises. The user-centric approach allows the users to customize their workspace to their own needs and finally to use the best suitable tools for a given task, ultimately ensuring user satisfaction and productivity.

III. Architecture Methodology and Overview

To gain a comprehensive picture of the architecture and to verify its completeness, a proven methodology for the gathering of stakeholders and their respective requirements needs to be followed. For the architecture definition phase, the proposal of Rozanski and Woods [18] was adopted appropriately. Based on their methodology, a four-step approach was followed as further depicted in figure 1 and consisting of four distinct phases: subsequent to definition of architecture scope and context, involved stakeholders have been defined and their respective concerns captured which have finally condensed into the actual architecture definition.

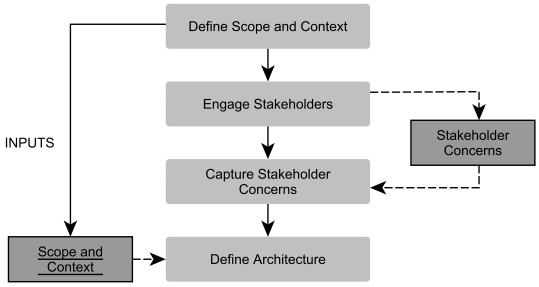

Fig. 1. Architecture Definition Methodology (adopted from [18, p. 78])

Intended as a collaborative platform for networked enterprises, it is embedded into a surrounding system architecture providing supporting components for collaboration members, especially in the field of process orientation and user management. As such, the architecture strives to achieve the following goals:

- Support for short-term cooperation, empowering team members to efficiently lookup proper network partners based on a set of requirements for the specific use case and to shorten time to market once a cooperation has been established.
- Support of user-centric application integration, putting the user into the centre of consideration.

Thus, the core of the platform builds on the following major functionalities as requested in the user requirements [19].

- Integration of external services into the platform, allowing for dynamic service consumption following the cloud computing paradigm.
- An integrative portal platform providing a single access point for all services operated on and integrated into the platform.
- Consistent user experience, achieved via single sign on and identity management facilities.
- Tight workflow integration via definition, execution, and management of workflows on the platform.

Operating as an intermediary between a wide range of different systems, the architecture is expected to support a broad set of different data to ensure proper collaboration between different team members. This data can consist of documents, generic unformatted content, parameters for service execution or the result of previous service usage(s). Moreover, all data processed on the platform will need to be kept separately from each other which is of high importance as the portal is the central point of access for all associated applications from a different trust background. Based on this consideration, communication of applications among each other needs to be taken into account and secured properly.

A. Context of the System

Clarifying the context of the proposed architecture, figure 2 shows the system boundaries and associated external stake-

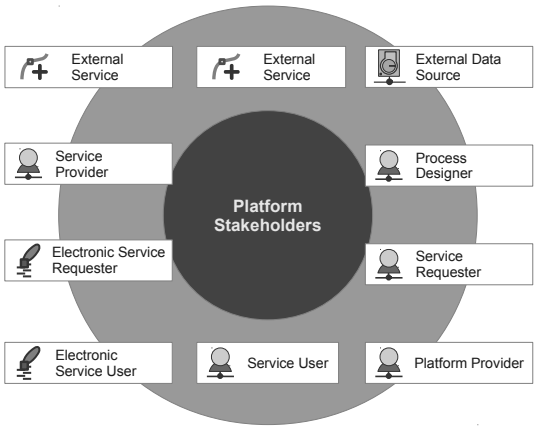

Fig. 2. Stakeholders of Virtual Enterprises

holders and services based on the user requirements as well as stakeholder groups. These actors are further outlined below.

- *Process designer*. A member of an institution in charge of a specific alliance and designing the processes to implement for alliance operation, including all information artefacts that are to be used in all individual processes.
- *Service provider*. An external entity providing a service (which can be either a complete dedicated application, an atomic service or a compound of several sub-services and processes) for usage in the presented platform, hence responsible for the actual registration of said service, its management and maintenance as well as later removal from the platform in case the application/service is no longer made available via the platform.
- *Service requester*. An actor performing the tasks of looking up relevant services, negotiations and platform integration. Can be either implemented as a human actor (human service requester) or a software component (electronic service requester) that automatically performs service discovery at runtime based on a set of predefined criteria.
- *Service user*. An alliance member consuming assets provided by a service provider. Following the idea of a user-centric approach, a service user is entitled to make a choice in case different applications are equally suited for a specific task. Similar to service requesters, service users can be either human actors or electronic counterparts building on the results of previous service execution(s).
- *External application/service*. Maintained by a dedicated service provider, external software components can be either monolithic applications providing a broad range of functionality or atomic services specifically tailored for one task. All applications and services are characterised by their particular protocol type used for integration into the platform.
- *Platform provider*: The central instance serving as an intermediary between all other parties associated via the platform. It is therefore under his responsibility to ensure availability of the system according to service level agreements with all involved parties. Also, the platform provider is responsible for guaranteeing proper payment processing so that financial compensation for service consumption can be assured in a reliable manner.

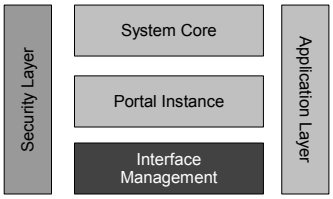

Fig. 3. Architecture Building Blocks

B. Architectural Building Blocks

In the following section, a brief overview is given on the individual components of the architecture and their interactions with outside components. As can be seen from figure 3, the system is characterised by individual components each responsible for specific aspects. Their respective functionality is briefly discussed below.

At the core of the proposed architecture, the *Portal Instance* represents the frontend to the end user which is responsible for providing users with a visual representation of applications and therefore carries out the actual work of user interface integration of external applications. Moreover, the *Portal Instance* is responsible for managing inter-portlet communication which implies the collection of events from portlet sources available in a given portal session and delivering them to other portlet destinations. Finally, the *Portal Instance* takes care of user session management by capturing the state of the current user session and the user's workspace as well as its storage and proper recovery during logout/login events.

Underneath the portal instance in figure 3 is the *Application Layer* component, responsible for managing the various applications available on the portal. This includes management of general information such as its name, protocol information and usage instructions as well as subscriptions management so that only subscribed companies can make use of an application and ensuring that application providers are granted proper compensation for service usages by the individual parties on the platform. Finally, the *Application Layer* deals with the question of constantly monitoring application availability as well as user satisfaction which can be additional factors during application selection.

The *Interface Management* component is in constant interaction with those two components and forms the interface to the external applications as connected via the integrative platform.

The *System Core* addresses general needs of the architecture. First of all, a centralised storage repository allows for retrieval, update, and storage of all data present and brokered via the platform. Secondly, global notification management across multiple portal instances is provided as well as workflow management facilities, keeping track of workflow instances

and all associated tasks deployed within an alliance. Likewise, a correlation between Workflow- and Session Management (as part of the *System Core*) helps to introduce a user-centric application integration approach where users can customise the set of applications used for task execution up to their own preferences. This way, users can stick to the type of application they are used to and feel most comfortable with, thus reducing the need for their employers to provide further training on a specific application. On the level of process definition, this requires a concise definition of the goals 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. [20]

The *Security Layer* component provides platform access decisions based on the user's identity, containing information about the user's home company as well as personal information (e.g. department or email address). This information is to be retrieved from the external identity management system associated with a specific user, i.e. provided by his employer and selected and queried during login. At the same time, it has to be assured that only trustworthy users who have been granted the necessary privileges by their employer beforehand may enter the platform and that only associated members of a networked enterprise may enter the system and perform actions with it.

IV. Architectural Views

Further extending the global view presented in the previous section, this section is devoted to describe the architecture in greater depth following the approach introduced by Rozanski and Woods [18]. The following sections focus on a selected aspect of the system each, presenting a subset of the overall architecture per architectural view. Due to space restrictions, presentation is limited to functional and information aspects as part of this paper, albeit further viewpoints covering concurrency, deployment, development or operational issues have been dealt with during the architecture definition.

A. Functional Viewpoint

The functional viewpoint is expected to provide an abstraction from the actual implementation and to depict the functionality of the system. To further illustrate the system, the following section focuses on the individual components of the system first and gives an overview on relevant interactions between those components and the data transmitted therein.

A global overview on the proposed architecture with general building blocks has already been given in figure 3. Based on these building blocks, the functionality as derived from the user requirements definition is fulfilled by the components illustrated in figure 4.

System Core: The heart of the architecture, consisting of modules for basic low-level functionality: workflow handling, notification management, storage facilities, as well as security and user identity services. It is therefore strongly affiliated with all surrounding architecture components as it aggregates common functionality shared by the various components from *Portal Instance*, *Security Layer*, and *Application Layer*.

Portal Instance: The graphical interface of the architecture to the user. For this reason, it aggregates information from the various sources available to the user, represented via distinct applications and/or services. At the same time, it is used for context capturing and communication among individual applications connected via the platform.

Internally, the *Portal Instance* can be separated between the core portal functionality and its interfaces offering integration capabilities with external applications. To achieve this, it is strongly aligned with the Interface Management component to establish data connections with outside applications. This allows for inter-portlet communication under special consideration of the user's current working context, which includes any other currently active applications, the user's login device, or other information from the user's session.

Application Layer: The logical counterpart to the Portal Instance from an architectural point of view. Whereas the Portal Instance reflects management of an overall portal session, the *Application Layer* focuses on one application integrated into the portal and provides a dedicated set of functionality for it. It therefore concentrates on application registration application, subscriptions management, accounting aspects as well as quality of service measurement and takes into account application availability as well as user satisfaction after usage. This can be seen as a prerequisite for a user-centric approach since potential users can be provided with manifold information about an external application and its provider prior to first usage.

Interface Management and External Applications: Responsible for functionality related to integration of applications such as user interface or data format conversions. Logically located between the Portal Instance and the respective external application, it serves as an intermediary to provide display, notification as well as other functionality to the portal session and needs to support a broad range of protocol types used by different application providers.

Security Layer: Orthogonal to the three layers introduced before, it is in charge of all security-related aspects of the system. As security needs to be dealt with in a wholistic manner, this layer spans across System Core, Portal Instance as well as Interface Management components.

Security inside the system is targeted on three different levels:

1) *Access Management* ensures that only authorised users are able to access the platform. This requires that the user's home company has been associated with the platform in a first step. Subsequently, the company needs to be affiliated with one or more alliances operating on the platform and the user needs to be assigned to one or more of these alliances.
2) *User Identity Management* retrieves information about a user's identity from the user's associated home company. Said information can be requested during login at the platform, but also during service execution in case further information about a user is needed to employ a special service.

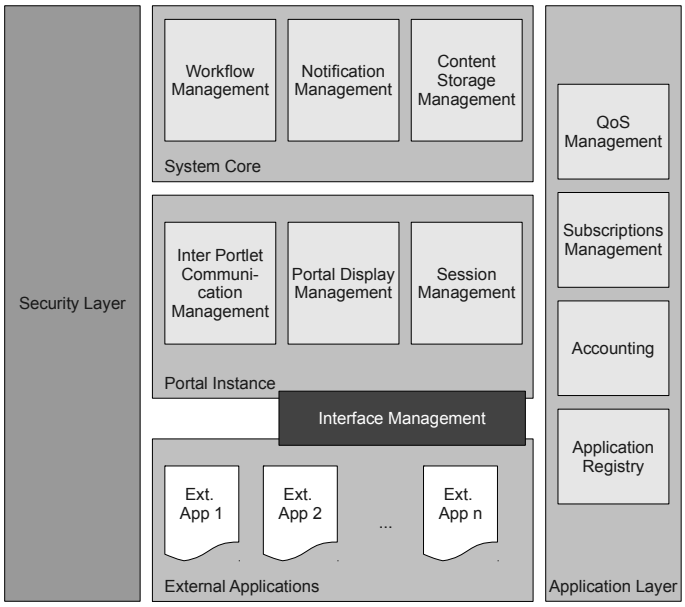

Fig. 4. Portal Architecture

3) The *Auditing* component takes over auditing-related tasks such as secure storage of service execution time as well as the respective actor performing this action. This information is later used for accounting aspects of the platform.

The architecture modules have been defined to maximise cohesion while still enabling loose coupling, two criteria with strong influence on software maintainability as pointed out by Yourdon and Constantine [21]. Figure 5 shows the major components and their respective interactions with each other.

As shown in figure 5, communication between the various building blocks of the architecture concentrates around a set of interfaces, described in the remainder of this section.

The *Storage* interface is relied upon by all other building blocks within the architecture. As such, it is expected to provide dedicated create/read/update/delete (CRUD) operations for the different entities employed within the architecture. To achieve this, it is in close contact with the security subsystem (described further below) to make educated access decisions for the individual objects maintained on the platform.

The *Security* component forms a sensitive part within the architecture. It is hence used by aforementioned storage facilities as well as workflow and application handling. Moreover, security aspects are taken into account by notification management and the definition of portlet communication policies, presented in more depth in [22] and represented by the *Notification* interface exposed by the System Core, which represents the notification mechanism.

Notifications are generated from external applications and help to organise the user's workspace for instance by informing about application availability, accomplished tasks, and user logins. They are maintained by the dedicated component in the System Core, taking into account security implications, i.e. ensuring association with the correct user, his session, and the proper security context when forwarding these notifications.

The *Search* interface as presented in figure 5 is in charge of retrieving results from both, the service repository as well as the workflow storage where all workflow results are maintained and is therefore used from system core as well as for application and application subscription management.

B. Information Viewpoint

Generally, information artefacts can be aggregated so that artefacts of one type all serve the same or a closely related purpose which can then be maintained in a consistent manner with respect to ownership or CRUD operations. Dealing with these artefacts, the Information Viewpoint presents information flows, data sources and their mutual relationships within the system architecture.

1) Data Elements: The architecture is intended to seamlessly integrate a series of different application types and to include the respective results into workflows performed in the context of different alliances involving different partners. During the design of the data elements, it was one major design goal to create rich entities in order to reduce the number of data items and their relations. Figure 6 presents an overview on all major data elements available in the system and their respective relationships. It needs to be pointed out, however, that this listing is not exhaustive and focuses on major data elements only. Moreover, all data elements have been modelled here only from the business logic point of view. For this reason, no data storage is shown in figure 6. Implementation of a proper storage mechanism as laid out in the functional viewpoint naturally needs to be performed prior to its usage, for instance by employing a persistence mechanism.

As can be seen from figure 6, the company object representing the various alliance members is considered at the core of

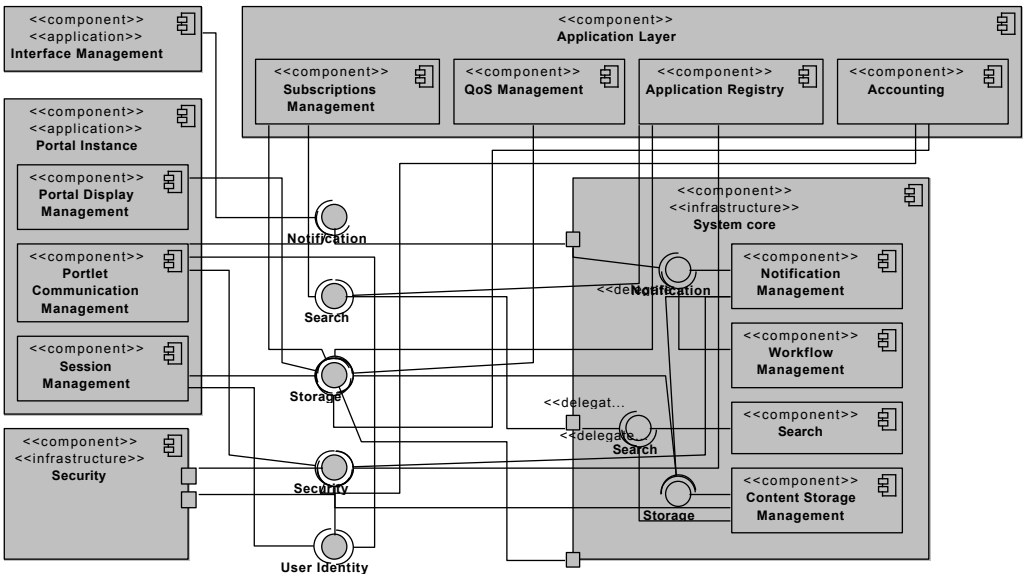

Fig. 5. Portal Architecture Relations

all data elements. A company can be member in an unlimited amount of alliances with an unlimited number of members. Alliances can define a series of workflows associated with them. Each of these workflows can contain a number of task results which can be used at a later time as part of the workflow.

Notifications can indicate different circumstances, ranging from (1) notifications about specific resources such as applications, where scheduled maintenance windows could be announced, over (2) portal-related notifications about user status change, indicating for instance login/logout events, and finally (3) workflow-related notifications, which could indicate status changes in single tasks or individual workflows.

Finally, the user-centric approach of the architecture is expressed by the fact that users can have multiple identities associated with them representing different contexts they are in, for instance when a user is accessing the portal from different locations with a different degree of trustworthiness or when partaking in different alliances at the same time. Furthermore, each of these identities can have their own set of application preferences.

2) Data Ownership: Due to the nature of the architecture as a workflow-oriented system, questions of data synchronisation and ownership arise; however, there is always one single source of truth: information is processed in one task step and then further transferred to the next task in the process chain. In this context, each of these tasks serves as a locking mechanism for all data available in the process, meaning that data can only be modified by the user associated with the current task step. During this time, no other access can be performed until the task step has been finished and the locking token is handed over to the subsequent task step.

3) Information Flow: With a flexible and dynamic architecture such as presented in this work, it is important to be aware about the possible flows of information. Based on the data available at this stage of architecture development, a high-level overview on relevant information and their respective flows is given below.

Since the platform targets integration of external users, data about user identities is exchanged with the appropriate external identity management systems retrieved on demand during login time or in case of service authorisation. Once evaluated, this data is no longer stored on the system and has to be retransmitted from the source system. For future implementations, authorisation at external web services could be performed using the service provisioning markup language (SPML) [23] which provides a standardised way to issue and deal with provisioning requests.

Another category of information covers all process-related information. On the one hand, this concerns workflow model information imported into the platform and interpreted by the associated workflow module. After successful import, no more modifications are performed during runtime. On the other hand, the individual workflows generate data in the form of performance metrics, audit logs, or task results. With the exception of task results, this data is stored in a read-only manner so that it cannot be tampered with later on. For the case of task results, these are subsequently transported to any successor tasks depending on this input where they are further dealt with. When stored, a new version of this data is saved so that all results can be rolled back transparently.

V. Evaluation

In order to assess the validity of the previously presented architecture, validation could be performed in the EU-funded research project *SPIKE*[2] in the form of a software-based prototype. This project involved eight distinct partners from

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

Fig. 6. UML Class Diagram of Major Architectural Data Elements

five different countries all across Europe from both, academic as well as from corporate background.

Originating from extensive user requirements analysis to serve actual market needs and resulting in the documentation of a variety of user requirements and use cases [19], these use cases could be aggregated into three different application cases each supervised by a dedicated user partner of the *SPIKE* project:

- *Information Hotel*, providing technical documentation services within and across organisations especially from a SME context. This application case was based on an existing collaborative business process involving a considerable number of companies.
- *Legacy Applications*, validating integration of external legacy applications. Basically, the functionality of this application case was related to the SPIKE service portal that enables service providers to offer their specific services to any service requester.
- *Identity Federations*, targeting an integrated view on user identities across multiple participants.

Each of these application cases was treated equally and evaluated independently. Since all three application cases were carried out in the particular domain of the responsible user partner, a high level of expertise and support for all respective developments and evaluation could be achieved.

Prior to the implementation phase, architecture validation has been performed by the project members to ensure that the envisioned architecture supports all use cases as condensed from the user requirements specification. During this process, it has been laid out that all user requirements and respective use cases could be supported by the architecture definition on a theoretical level prior to implementation.

Based on these results, the prototypical implementation based on open source components, predominantly the portal server Liferay, was subsequently developed and tested in two dedicated trial phases which followed an iterative approach

with an incremental set of functionality per iteration. While the first phase focused on standalone components, the second trial phase targeted an integrated system. This has allowed the project to incorporate feedback from the first testing attempt back into the system design before performing the second iteration of the testing phase. Each phase has been concluded with a pilot installation.

In total, more than twenty use cases were evaluated in these trial phases whereas each use case consisted of a series of related tasks testing dedicated parts of the architecture implementation for validity. Although not all requirements could be demonstrated in these two cycles, final evaluation has shown that the major goals as outlined in section III of this paper have been fulfilled to a satisfactory degree and that the underlying concepts can indeed pose additional value for the business settings that networked enterprises are typically confronted with.

VI. Conclusion and Future Work

This paper has presented a generic architecture that can serve as a reference to perform application integration in the domain of inter-organisational networks. Special emphasis has been put on a high level of customisability necessary to support consumption of cloud-based services in a flexible manner under the responsibility of the targeted user and to support future enhancements and hence achieve a high degree of maintainability. At the same time, this paper has shown the evaluation of the presented architecture as part of a large-scale research project which has demonstrated the feasibility of the approach in an application prototype.

Even though the approach for user-centric enterprise application integration as presented in this paper may not be desirable or feasible in all circumstances, enterprises this way are provided with a broader range of options to select from in order to flexibly react to changing conditions such as different market demands. This way, services can be consumed and exchanged dynamically. At the same time, users are empowered to choose from a set of fitting services for any given task and use exactly the kind of service they feel most comfortable with, hence reducing training costs while ensuring user satisfaction which can pose significant success factors to strengthen the company's position in the market.

Acknowledgements

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

This paper is part of a PhD thesis with the title "User-Centric Application Integration in Enterprise Portal Systems", to appear at Eul Verlag, Lohmar, Germany.

References

[1] M. Ebers, *The Formation of Inter-Organizational Networks*. Oxford University Press, 1997, ch. Explaining Inter-Organizational Network Formation, pp. 3–40.

[2] K. Thompson, *The Networked Enterprise: Competing for the Future Through Virtual Enterprise Networks*. Meghan-Kiffer Press, 2008.

[3] J. Governor, D. Hinchcliffe, and D. Nickull, *Web 2.0 Architectures*. O'Reilly Media, Inc., 2009.

[4] Z. Irani, M. Themistocleous, and P. E. Love, "The impact of enterprise application integration on information system lifecycles," *Information & Management*, vol. 41, no. 2, pp. 177–187, 2003.

[5] J. Grimson, W. Grimson, and W. Hasselbring, "The SI challenge in health care," *Communications of the ACM*, vol. 43, no. 6, pp. 48–55, 2000.

[6] R. Zahavi, *Application Integration with CORBA*. Wiley, 1999.

[7] W. Hasselbring, "Information system integration," *Communications of the ACM*, vol. 43, no. 6, pp. 32–38, 2000.

[8] F. B. Schneider and K. P. Birman, "The Monoculture Risk Put into Context," *IEEE Security & Privacy*, vol. 7, no. 1, pp. 14–17, January/February 2009.

[9] D. Williams, W. Hu, J. W. Davidson, J. D. Hiser, J. C. Knight, and A. Nguyen-Tuong, "Security through Diversity: Leveraging Virtual Machine Technology," *IEEE Security & Privacy*, vol. 7, no. 1, pp. 26–33, January/February 2009.

[10] R. Winter, "An Architecture Model for Supporting Application Integration Decisions," in *Proc. of the 11th European Conference on Information Systems (ECIS 2003)*, Naples, Italy, 2003.

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

[12] J. C. Lutz, "EAI architecture patterns," *EAI Journal*, pp. 64–73, March 2000.

[13] *UML Profile and Interchange Models for Enterprise Application Integration (EAI) Specification*, Object Management Group (OMG), 2004. [Online]. Available: http://www.omg.org/spec/EAI/

[14] D. S. Linthicum, *Enterprise Application Integrations*. Addison-Wesley, 2000.

[15] ——, *B2B application integration*. Addison-Wesley, 2001.

[16] ——, *Next Generation Application Integration: From Simple Information to Web Services*. Addison Wesley, 2003.

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

[18] N. Rozanski and E. Woods, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley, 2005.

[19] 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*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, O. e. a. Akan, Ed. Springer, Berlin, 2010, vol. 21, pp. 35–44.

[20] O. Gmelch and G. Pernul, "A portal-based approach for user-centric legacy application integration in collaborative environments," in *Wirtschaftinformatik Proceedings*, Zürich, CH, 2011, pp. 693–703.

[21] E. Yourdon and L. L. Constantine, *Structured Design: Fundamentals of a Discipline of Computer Program and System Design*. Prentice-Hall, 1979.

[22] O. Gmelch and G. Pernul, "Preventing malicious Portlets from Communicating and Intercepting in Collaboration Portals," in *Proc. of the International Conference on Security and Cryptography (SECRYPT '10)*, Athens, GR, 2010, pp. 177–182.

[23] G. Cole, J. Bohren, R. Boucher, D. Cohen, C. Collingham, R. Elron, M. Fanti, I. Glazer, J. Hu, R. Jacobsen, J. Larson, H. Lockhart, P. Mishra, M. Raepple, D. Rolls, K. Spaulding, G. Sadhi, C. Williams, and G. Woods, *Service Provisioning Markup Language (SPML) Version 2*, http://www.oasis-open.org/specs/, The Organization for the Advancement of Structured Information Standards (OASIS), 2006, retrieved 2012-01-08.