Workshop-Bericht

EAI 2004 – Enterprise Application Integration

OFFIS, Oldenburg, 12. – 13. Februar 2004

Die heutige Flut von Informationsquellen im betrieblichen Umfeld bestimmt die immer weiter zunehmende Notwendigkeit der Integration von Anwendungssystemen und Datenbanken innerhalb der betrieblichen Datenverarbeitung und über Unternehmensgrenzen hinweg. Enterprise Application Integration (EAI) bezeichnet die Planung, die Methoden und die Software, um heterogene, autonome Anwendungssysteme unternehmensweit oder -übergreifend, unter Einhaltung bestimmter Protokolle und unter Vermeidung von Abhängigkeiten, zu integrieren.

Der Workshop EAI 2004 widmete sich diesem Themenspektrum. Er fand als gemeinsame Veranstaltung des GI-Arbeitskreises *Enterprise Architecture* und der GMDS-/GI-Arbeitsgruppe *KIS – Informationssysteme im Gesundheitswesen* vom 12.–13. Februar 2004 am OFFIS Institut in Oldenburg statt. Auf dem Workshop wurde die Thematik EAI aus übergreifender, ganzheitlicher Sicht betrachtet, und es wurden die besonderen Herausforderungen aufgegriffen, diskutiert und der State-of-the-Art bestimmt. Das Programm umfasste insgesamt 13 Beiträge aus Wissenschaft und Praxis, die sich mit Technologien sowie Methodik, Nutzen und Kosten des EAI und technischen Umsetzungsproblemen in der Praxis auseinandersetzten.

Neben Vorträgen zu EAI-Methoden und -Technologien gab es Beiträge, die das Thema EAI aus Anwendungssicht näher beleuchteten. Der Fokus lag dabei auf Anwendungen aus dem Gesundheitswesen. Im Zentrum einer integrierten Gesundheitsversorgung sollte eine kooperative, sektorübergreifende und qualitätskontrollierte Behandlung der Patienten stehen. Diese wiederum ist nur effizient möglich, wenn die Informationssysteme der involvierten Leistungserbringer im Gesundheitswesen zusammenarbeiten. Berichte aus anderen Anwendungsbereichen, etwa der digitalen Fotografie und dem Elektrik-/Elektronik-Bereich, rundeten diese Betrachtungen ab, und ermöglichten einen Vergleich domänenspezifischer Anforderungen.

Am Workshop nahmen mehr als 50 Personen aus Wissenschaft und Praxis teil, was das große Interesse an der Thematik EAI unterstreicht. Aufgrund dieser hohen Resonanz wird es im kommenden Jahr mit der EAI'05 in Marburg eine weitere Auflage des Workshops geben. Der Tagungsband ist in der CEUR Workshop Proceedings-Reihe (Vol-93) erschienen und kann Online wie folgt abgerufen werden:
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-93/

Vier ausgewählte Beiträge des Workshops sind in diesem Heft abgedruckt.

Manfred Reichert, Wilhelm Hasselbring

DÍGAME: A Vision of an Active Multidatabase with Push–based Schema and Data Propagation

Cristian Pérez de Laborda, Christopher Popfinger, and Stefan Conrad

Institute of Computer Science
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Germany
{perezdel, popfinger, conrad}@cs.uni-duesseldorf.de

Abstract. Sharing information in loosely coupled enterprises or virtual corporations demands a flexible and dynamic architecture, suitable for their individual data policies. The aim of this paper is to present the DÍGAME architecture, which balances both, local autonomy and a reasonable degree of information sharing. Therefore we combine the well known concept of loosely coupled multidatabases with more recent research in Peer–to–Peer or grid computing, satisfying the needs of modern intra- and inter-enterprise collaboration. In our architecture data and schema updates are propagated actively to subscribing component databases, without being managed by any central authority. This replication gives us the possibility to realize individual integration on each single peer.

1 Motivation

Since the first centralized databases found their way into the enterprises in the late 60s, the needs and requirements have changed towards a more distributed management of data. Today there are many corporations which possess a large amount of databases, often spread over different regions or countries and generally connected to a network. These local databases (components or component systems) typically raised in an autonomous and independent manner, fitting the special needs of the users at the local site. The design of the databases and the functionalities provided, intend to fulfil the aims of the departments. This leads to logical and physical differences in the databases concerning data formats, concurrency control, the data manipulation language or the data model [1]. An information system is required to integrate the information of these heterogeneous data sources to provide a global access.

One of the main challenges in the integration of data in such environments, is the autonomy of the participating data nodes (peers). This autonomy implies the ability to choose its own database design and operational behaviour [2]. Local autonomy is tightly attached to the data ownership, i.e. who is responsible for the correctness, availability and consistency of the shared data. Centralizing data means, to limit local autonomy and revoke the responsibility from the local administrator, which is not reasonable in many cases. The federated architecture

for decentralizing data has to balance both, the highest possible local autonomy and a reasonable degree of information sharing [3].

In this paper we introduce the vision of the DÍGAME architecture, a **D**ynamic **I**nformation **G**rid in an **A**ctive **M**ultidatabase **E**nvironment, which actively propagates data and schema updates over import/export-components between dynamically connectable data peers. This architecture offers a flexible and fail–safe information platform based on the data policies in organisations achieving a feasible trade–off between local autonomy and a reasonable degree of information sharing.

2 DÍGAME Architecture

2.1 Vision

The aim of this work is to introduce an architecture which allows the dynamic connection of data sources without restricting their local autonomy in order to share selected information. This union is based on Peer-to-Peer (P2P) concepts and operates without any central administrative instance.

The administrator of each peer makes a subset of its data accessible. Other peers are now able to integrate this data into their local databases, subscribing to a specific part of the data provided. Thereupon updates are propagated automatically to the subscribers by the data source, including both, data and schema modifications. Each data source of this dynamic information grid is herewith able to maintain an up-to-date replica of the required data and schema items. As there is no general rule for the integration of the replicated data, it has to be integrated individually by the administrator of each subscriber database.

Our architecture is especially designed to support dynamic intra– and inter–enterprise collaboration, by enabling each department involved to supply all relevant partners with the required information.

2.2 Components

We will now discuss the required components of the architecture using a case study, to draw up the benefits of our dynamic information grid. This example describes our approach to solve one of the multiple challenges concerning collaborative work: distributed information management.

Consider a worldwide operating company, planning the launch of a new product. To simplify our scenario, we assume that there are solely three departments involved in this business process, the executive board (management), the sales office and product engineering. Further departments may join this collaboration at any time. Each department manages its own database, to store the information for which it is responsible. The management produces basic data of the product. This includes deadlines, descriptions, workflows and additional objectives. This management information is substantial for the further product development and the work in the participating departments. The product engineering uses a predefined part of that management data as basic conditions for the concrete implementation and technical realization. Local applications like CAD or measurement programs create additional data which has to be stored separately. According to the product engineering the sales department enriches the authoritative management data with concrete concepts for the oncoming product launch. Furthermore concrete development plans of the product engineering are required to prepare sales strategies. Both, sales and product engineering departments, concretize the strategic guidelines of the management in their specific assignment. To keep track of the costs and the progress of the project, it is indispensable for the management to access the product engineering and sales departement's relevant information just mentioned.

Basically there are two different techniques for providing the peers (departments) with the required data. Contrary to the commonly used method querying the data sources actively, our approach uses replication of data and schema, initiated by the data source. Referring to our example, the executive board gets data updates whenever changes occur in the sales and/or product engineering databases rather than having to request for updated data items continuously.

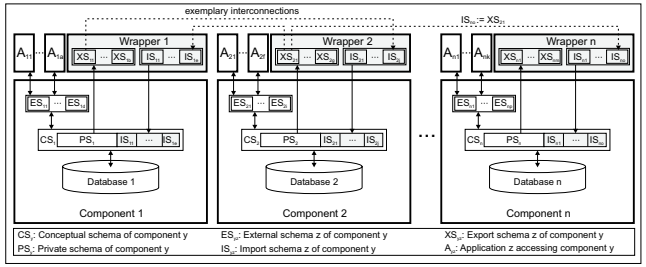

Fig. 1. DÍGAME Architecture

Our DÍGAME architecture (Figure 1) consists of the following components:

Autonomous Component Databases: As already mentioned our architecture is designed for integrating data from across autonomous data sources. The peers involved are linked to an information grid, retaining their local autonomy completely. According to the 3-level-architecture [4] and the architecture for loosely coupled multidatabases [3], each component database consists, besides the internal schema, of a conceptual and several external schemas. The conceptual schema (CS) comprises the locally maintained private schema (PS) and a couple of schemas imported from other peers (IS), managed by a wrapper component. Local applications (A) access the data

via external schemas (ES), which are derived from the whole conceptual schema, providing solely read-only access to all imported schemas. The grid infrastructure does not include a global view over the integrated data, but every peer maintains its own integrated schema. Due to the absence of a global view, we have ideal conditions for individual integrations on each peer.

Wrapper: The core of our dynamic information grid is the wrapper component. As part of the middleware, it is responsible for negotiating and establishing communication and exchanging data between the component databases. Therefore it maintains a *repository* containing all the meta data accumulated, particularly a copy of the import schemas mentioned above and export schemas (XS) based exclusively on the private schema. Of course corresponding schema information has to be stored only if data is imported or exported. Thus the participation level corresponds directly to the amount of schema information the wrapper has to manage. In fact each import schema matches an export schema, offered by one of the remaining peers.

In addition to the repository, two more modules have to be implemented inside the wrapper. A *publishing unit* is used for transmitting information. Earlier research proposes several mechanisms helping a wrapper to identify data modifications [5, 6]. If there are triggers of underlaying database systems available, they should be used. The counterpart of the publisher is a *subscribing unit*, which receives incoming information. Both units contain a *negotiator*, which sets up a communication channel, used by a *data handler*, to exchange data in a standardized format (e.g. RDF).

2.3 Characteristics

Our architecture combines the advantages of established concepts known not only in the database field, but also in related research areas, like grid or P2P computing. The combination of the established *Three Schema Architecture* [4] and that of loosely coupled multidatabases [3] with the achievements of the more recent field of P2P data management [7] provides a promising framework for an enterprise information platform. We are thus able to apply the flexible interconnectivity of P2P systems to multidatabases, including not only relational databases, but a loosely coupled federation of virtually any kind of data source. Although we focus in this paper on relational database systems, our architecture can be adapted to X.500 directory services or even file systems by adjusting the wrapper component.

In fact, the data replication on each subscriber database provides a couple of advantages, according to distributed databases systems [8]. As the peers serve all their local applications with the data required, there is no need for these to query remote data sources, leading to an increase of performance. Furthermore, a temporary network blackout can be bridged without being noticed by the applications. Comparable to distributed databases, in scenarios with more data queried than modified, even a significant reduction of network traffic can be obtained. Of course a replication entails the redundant storage of data items, but this disadvantage is neglectable due to the rapid decrease of storage costs.

To ensure a high level of data quality, the data can only be modified by the data owner. Information sharing between autonomous data sources is realized without loss of data ownership and autonomy on each peer, leading to a higher quality of data [9].

To guarantee both, the correctness and up-to-dateness of the data, each single modification can be propagated by pushing it to the subscriber databases. Hence each peer is able to provide, a running network environment supposed, up-to-date data to its applications at any time. Due to the push-based characteristics of DíGAME updates may be lost if a communication failure caused by a network or computer breakdown occurs. In this case there are basically two possibilities to re-synchronize the data. Either the publishing peer repeats the lost update propagations or the subscribing peer itself demands for these data and schema modifications. Since the first option enforces the data source to keep a complete track on the success of every propagation to each subscribers, we prefer to integrate a pull-based fallback mechanism into our architecture. This means that the subscribing peer has to search for lost data and schema updates by itself. The concrete definition of this functionality is part of future work.

3 Related Work

Simultaneously to the first generation of grid computing in the mid 1990s [10] some efforts arised to use distributed resources for information retrieval. Although the *Information Grid* of Rao et al. [11] is focused on giving an integrative user interface for distributed information, this approach can be seen as an early forerunner of the so called *Data Grid* [12], a specialization and extension of grid computing. Its intention is to create an architecture of integrated heterogeneous technologies in a coordinated fashion. Though we admit that a global metadata repository as proposed by Chervenak et al. would simplify many of the challenges, we abstain from that that effort of re-centralization, as it brings many difficulties about: every schema change has to be replicated to the global schema directory. The effect is a single point of failure, exactly the opposite of what we wanted to construct. We thus prefer to keep the databases as they are: autonomous, loosely coupled, and without a single point of failure.

With the raise of filesharing systems like Napster or Gnutella [13] the database community started to seriously adopt the idea of P2P systems to the formerly known loosely coupled database systems. Contrary to the data grid, P2P database systems do not have a global control in form of a global registry, global services, or a global resource management, but multiple databases with overlapping and inconsistent data. These P2P databases resemble heterogeneous and distributed databases, also known as *multidatabases* [14]. Currently the database community makes a great effort in investigating P2P databases. Particularly the *Piazza* [7] project is worth mentioning, where a P2P system is built up with the techniques of the *Semantic Web* [15] with local point–to–point

data translations, rather than mapping to common mediated schemas or ontologies. Contrary to Halevy et al., we deal mainly with relational data and do not have a global schema, as every peer may have its own import–/export–schema combination. For a more general glimpse on data mappings in P2P systems see [16].

Our strategy allows data to be exchanged among distributed databases connected through a lazy network. This means, that although a running network may not be guaranteed and thus some data broadcasts may be lost, the system heals itself. This challenge resembles the problems known from environments with mobile databases. Current research covers synchronous mobile client synchronization, i.e. data changes are propagated periodically and not just in time of the data change. In contrast to the broadcast disks, we ensure in our model, that data is only broadcasted to the clients when changes occur, unless the communication between both peers crashes. Hence our approach resembles a *push–based* system with a *pull-based* fallback, similar to [17] with the major difference that our approach is not based on broadcast disks, but on a push–based replication strategy also found in mobile clients like [18], resembling the software engineering's *Observer–Pattern* [19]. This pattern gives us a prototype of how to notify all interested databases about data updates [18]. As a result, communication is only started, if a data update has occurred and a database is interested. In consequence, data broadcasts are minimized.

Following the argumentation in [20] and [12] our model provides Single–Master Replication, the only guarantor for data stability and clear defined data flows.

Most of the research on active multidatabases has been done concerning global integrity. Chawathe et al. [21] propose a a toolkit for constraint management in loosely coupled systems. To mention is also the idea of Gupta and Widom to optimize the testing of global constraints by local verification [22]. Conrad and Türker [6] sketch a more general architecture for an active federated database system. They extend a multidatabase system by ECA-Rules to preserve consistency. A main challenge hereby is to detect local events, especially schema and data modifications, which is commonly done by a software module for each data source, i.e. a monitor or wrapper component. Basically two approaches are therefore proposed: Conrad and Türker use the event detection ability of the underlying subsystem, while Blanco et al. [5] use the operating system to signal schema modifications by directly observing changes to the data(base) files.

4 Conclusion and Future Work

We have presented in this paper an architecture for a dynamic information grid in an active multidatabase environment, suitable for sharing information across autonomous and heterogeneous data sources. Our loosely coupled federation enriched with P2P and grid computing concepts, enables collaborative work preserving local autonomy. Data and schema modifications are actively propagated to the clients after they have subscribed to the information offered by the data source.

For a complete implementation of the DÍGAME architecture, there are still some challenges to be taken. In the next step of the project we will focus on the detailed specification of the wrapper component, particularly on the negotiator and the data handler, for being able to establish communication between isolated peers. Therefore we have to specify a communication protocol and a data and schema exchange format.

Due to its characteristics DÍGAME provides a sophisticated infrastructure for a diversified application field, including e-business, e-science or e-health, initiating the next generation of collaborative work.

References

1. Litwin, W., Abdellatif, A.: Multidatabase Interoperability. Computer **19** (1986) 10–18
2. Mullen, J.G., Elmagarmid, A.K., Kim, W., Sharif-Askary, J.: On the Impossibility of Atomic Commitment in Multidatabase Systems. In: Proceedings of the 2nd International Conference on System Integration, Morristown, New Jersey, IEEE Computer Society Press (1992) 625–634
3. Heimbigner, D., McLeod, D.: A Federated Architecture for Information Management. ACM Transactions on Information Systems (TOIS) **3** (1985) 253–278
4. Burns, T., Fong, E.N., Jefferson, D., Knox, R., Mark, L., Reedy, C., Reich, L., Roussopoulos, N., Truszkowski, W.: Reference model for dbms standardization, database architecture framework task group (daftg) of the ansi/x3/sparc database system study group. SIGMOD Record **15** (1986) 19–58
5. Blanco, J.M., Illarramendi, A., Pérez, J.M., Goñi, A.: Making a Federated Database System Active. In: Database and Expert Systems Applications, A.M. Tjoa and I. Ramos (eds.), Springer Verlag, ISBN 3-211-82400-6. (1992) 345–351
6. Türker, C., Conrad, S.: Towards Maintaining Integrity of Federated Databases. In: Data Management Systems, Proc. of the 3rd Int. Workshop on Information Technology, BIWIT'97, July 2–4, 1997, Biarritz, France, Los Alamitos, CA, IEEE Computer Society Press (1997) 93–100
7. Halevy, A.Y., Ives, Z.G., Mork, P., Tatarinov, I.: Piazza: Data Management Infrastructure for Semantic Web Applications. In: Proceedings of the twelfth international conference on World Wide Web, Budapest, Hungary (2003) 556–567
8. Ceri, S., Pelagatti, G.: Distributed databases principles and systems. McGraw-Hill, Inc. (1984)
9. van Alstyne, M., Brynjolfsson, E., Madnick, S.: Why not one big database?: principles for data ownership. Decision Support Systems **15** (1995) 267–284
10. Roure, D.D., Baker, M.A., Jennings, N.R., Shadbolt, N.R.: The Evolution of the Grid. In Berman, F., Fox, G., Hey, T., eds.: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons Inc., New York (2003) 65–100
11. Rao, R., Card, S.K., Jellinek, H.D., Mackinlay, J.D., Robertson, G.G.: The Information Grid: A Framework for Information Retrieval and Retrieval-Centered Applications. In: Proc. of the 5th Annual Symposium on User Interface Software and Technology (UIST'92), Monterey, CA (1992) 23–32

12. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., Tuecke, S.: The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications **23** (2000) 187–200
13. Carlsson, B., Gustavsson, R.: The Rise and Fall of Napster - An Evolutionary Approach. In: AMT 2001, Proceedings of the 6th International Computer Science Conference - Active Media Technology. Volume 2252 of Lecture Notes in Computer Science., Hong Kong, China, Springer (2001) 347–354
14. Bernstein, P.A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu:, I.: Data management for peer-to-peer computing: A vision. In: Proc. of the Fifth International Workshop on the Web and Databases, WebDB 2002, Madison, WI (2002)
15. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)
16. Kementsietsidis, A., Arenas, M., Miller, R.J.: Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, San Diego, CA, ACM Press (2003) 325–336
17. Acharya, S., Franklin, M., Zdonik, S.: Balancing push and pull for data broadcast. In: Proceedings of the 1997 ACM SIGMOD international conference on Management of data, Tucson, Arizona, ACM Press (1997) 183–194
18. Hara, T.: Cooperative caching by mobile clients in push-based information systems. In: Proceedings of the eleventh international conference on Information and knowledge management, McLean, Virginia, USA (2002) 186–193
19. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements od Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series. Addison-Wesley Publishing Company, New York, NY (1995)
20. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The Dangers of Replication and a Solution. In: Proceedings of the 1996 ACM SIGMOD international conference on Management of data, Montreal, Canada, ACM Press (1996) 173–182
21. Chawathe, S., Garcia-Molina, H., Widom, J.: A Toolkit For Constraint Management In Heterogeneous Information Systems. In: Proceedings of the International Conference on Data Engineering, New Orleans, Louisiana (1996) 56–65
22. Gupta, A., Widom, J.: Local Verification of Global Integrity Constraints in Distributed Databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data SIGMOD'93, Washington, DC (1993) 49–58

Prozessorientierte B2B-P2P-Integration für Wissensmanagementanwendungen

Uwe Küssner

neofonie Technologieentwicklung und Informationsmanagement GmbH,
Robert-Koch-Platz 4 , D-10115 Berlin,
Uwe.Kuessner@neofonie.de

Zusammenfassung Workflowmanagementsysteme, die spezielle Unterstützung für interorganisationale Workflows bieten, sind ein vielversprechendes Integrationswerkzeug für B2B-Integrationen. Wir stellen Anforderungen an derartige Systeme vor und berücksichtigen dabei auch Anforderungen aus dem Bereich des Wissensmanagements. Wir präsentieren am Beispiel eines sich in der Entwicklung befindlichen Systems, wie die Anforderungen funktional realisiert und wie sie architektonisch auf Basis eines P2P-Protokolls umgesetzt werden und bewerten die eingesetzte Technologie JXTA.

1 Einführung

Die Integration von Unternehmensanwendungen(EAI) ist nicht nur eine informationstechnische Aufgabe. Aus Unternehmenssicht handelt es sich dabei in erster Linie um eine Maßnahme des Geschäftsprozessmanagement (GPM). Aus Sicht des GPMs geht es dabei darum, bestehende Prozesse zu optimieren oder zukünftige möglichst optimal zu gestalten, indem durch softwaretechnische Integration die Zusammenarbeit der Prozessschritte verbessert oder überhaupt erst ermöglicht wird. Insbesondere bei einer prozessorientierten Integration durch Workflowmanagementsysteme(WfMS) sollte vor der eigentlichen Integration eine Modellierung, Analyse und (Re)design der betreffenden Geschäftsprozesse(GP) durchgeführt werden. Auf der anderen Seite besteht ein enger Zusammenhang zwischen Wissensmanagement(WM) und GPM, siehe dazu z.B. [2]. Wissen und Prozesse sind eng miteinander verknüpft, da bei der Bearbeitung von Prozessschritten immer auch Wissen generiert, verteilt, bewahrt oder verwendet wird. Die Verarbeitung von Wissen ist immer kontextabhängig, der GP gibt diesen Kontext vor und ist somit gewissermaßen der „natürliche“ Ort zum Ansatz von WM. So gesehen ist es naheliegend, die Einführung eines prozessorientierten Integrationswerkzeugs (i.e. WfMS) nicht nur aus Sicht des GPMs, sondern integriert mit WM aufzusetzen. Bei der Analyse/Modellierung eines Workflows und Einführung eines WfMS sollten die Wissensaspekte dementsprechend mit berücksichtigt werden. Durch die zunehmende Virtualisierung von Unternehmen, zunehmendes Outsourcing von Prozessschritten und Durchdringung der Internettechnologie steigt der Bedarf an B2B-Integration. Eine technologische Antwort darauf sind B2B-WfMSe, die die speziellen Anforderungen an

Standort- oder Unternehmensgrenzen überschreitende Workflows(Wfs) berücksichtigen. Auch aus WM-Sicht ergeben sich weitere Anforderungen an derartige B2B-WfMS gegenüber traditionellen WfMSen.

Ein interessanter technologischer Ansatz, derartige Systeme zu konstruieren, liegt mit dem Peer-to-Peer(P2P) basierten Kommunikationsmodell vor. Im Folgenden stellen wir die speziellen Anforderungen an WfMSe im B2B-Bereich vor und berücksichtigen dabei auch die Anforderungen des WMs. Schließlich stellen wir an Hand des konkreten Systems „neofonie architect:flow“ vor, wie die Anforderungen architektonisch und funktional umgesetzt werden und zeigen welchen Stellenwert dabei die verwendete P2P-Technologie hat. Das System befindet sich zur Zeit noch in der Entwicklung. Die Entwicklung ist teilweise mit BMBF-Mitteln gefördert.

2 B2B-Workflow-Management-System (B2B-WfMS)

WfMSe als Teil einer Integrationsplattform überwinden einige der Schwächen von Hub-and-Spoke-Architekturen nach dem Publish/Subscribe(P/S)-Kommunikationsmodell wie: Multistep-Integration, Geschäftslogik, bidirektionale Kommunikation (siehe [4], Seite 12). WfMSe, als Werkzeuge des GPMs, dienen der Automatisierung von GPen. Voraussetzung zur Automatisierung ist die vorherige Erstellung eines formalen Modells (Wf-Schema), in dem der Workflow als Ablauf von einzelnen Aktivitäten beschrieben wird. Typische Modellelemente sind hier Sequenz, Splits (paralleler Ablauf), Joins (Synchronisationspunkte) und Bedingungen usw. Einen Vergleich der Ausdrucksstärke von verschiedenen Systemen ist in [1] zu finden. Bei den Aktivitäten können automatische, von solchen mit manuellen Anteil unterschieden werden. Aus Sicht des WfMSs bestehen die Aktivitäten aus Programmen/Prozessen, die unter der Kontrolle des WfMSs ausgeführt werden. Gibt es einen manuellen Anteil, fungieren die Programme als Werkzeuge zur Bearbeitung des Prozessschrittes, die natürlich nicht von beliebigen, sondern nur von dafür vorgesehenen Mitarbeitern bedient werden sollen. In diesem Fall ist es Aufgabe des WfMSs die entsprechenden Aufgaben in die Taskliste der dafür vorgesehenen Mitarbeiter einzuordnen. Diese Zuordnung findet nicht zur Designphase des Wf-Schemas, sondern zur Laufzeit einer konkreten Wf-Instanz statt. Eine Wf-Instanz ist eine Repräsentation eines konkreten Wfs zu einem gegeben Schema. Die Erzeugung von Wf-Instanzen gehört ebenfalls zu den Aufgaben des WfMSs. Herkömmliche WfMSe unterstützen nur Wfs, die innerhalb eines Unternehmens ablaufen, aber keine Standort übergreifenden oder gar Unternehmensgrenzen überschreitenden Wfs (B2B-Workflows). In diesem Bereich haben wir es mit besonderen Anforderungen zu tun:

Sicherheit. Im Vergleich zu intraorganisationalen Workflows gibt es bei interorganisationalen Workflows erhöhte Sicherheitsanforderungen:

Die *Authentizität* ist auf verschiedenen Granularitäten sicherzustellen. So muss verhindert werden, dass sich ein Unternehmen für ein anderes ausgeben kann. Das gleiche gilt auf feinerer Granularität für Mitarbeiter, die einen Prozessschritt bearbeiten. Die Ergebnisse von einigen Prozessschritten müssen *vertraulich* behandelt werden und erfordern deswegen Verschlüsselung. Ein Sachbearbeiter, der eine Aktivität ausgeführt hat, darf dieses hinterher nicht erfolgreich abstreiten können (*Verbindlichkeit*). Unternehmensinterne Workflows sollen von außerhalb nicht einsehbar sein (*Verborgenheit*). Eine feingranulare Zugriffskontrolle muss sicherstellen, dass nur solche Mitarbeiter einzelne Prozessschritte und die damit verbunden Dokumente bearbeiten können, die dafür zugeteilt sind.

Autonomie, Architektur und Transaktionen. Eine Reihe von Anforderungen leiten sich aus der Autonomie der partizipierenden Organisationen ab. Partizipanten gehen bezüglich der Teilnahme an einem B2-Wfs naturgemäß eine Verpflichtung ein, wollen aber darüberhinaus weitgehend autonom agieren können. Unternehmen wollen sich beispielsweise nicht von zentralen Komponenten außerhalb ihres Standorts abhängig machen, die wesendliche Teile ihrer Prozesse kontrollieren. Das gilt beispielsweise für Transaktionsmonitore, die Ressourcen sogar blockieren können. Bezüglich Transaktionen möchte man daher bei B2B-Workflows Alternativen zu ressourcenblockierenden Systemen haben. Daher sollen sog. Business-Transaktionen unterstützt werden, d.h. langandauernde Transaktionen unter Aufweichung der ACID-Bedingungen. Statt Isolation sollen z.B. Wf-spezifische Kompensationsaktivitäten möglich sein. Autonomie bedeutet auch, dass das Wf-Schema keine Annahmen bezüglich spezifischer Mitarbeiter, Anwendungen oder der Organisationsstruktur machen darf. Weiterhin sollte die konkrete Realisierung eines Workflows innerhalb der Organisation festgelegt werden können. Aus Sicht des Changemanagement bedeutet dies, das die partizipierenden Organisationen ihre Struktur, Mitarbeiter und Anwendungen ändern können, ohne dass das für die anderen Partizipanten transparent wird. Aus WM-Sicht bedeutet dies, dass man nur soviel Wissen über Interna weitergibt, wie es die Zusammenarbeit erfordert.

Wissensmanagement Das System soll sowohl eine Personifizierungs- als auch eine Kodifizierungstrategie [5] unterstützen. Bei der *Kodifizierungsstrategie* geht es vorrangig darum, Wissen zu externalisieren ([11]) und in kodifizierter Form (Dokumente etc) dem Anwender zur Verfügung zu stellen. Bei der *Personifizierungsstrategie* ist nicht das Generieren, Sammeln und Speichern von Wissen von vorrangiger Bedeutung, sondern die Identifikation von geeigneten Wissensträgern. Das System sollte den Aufbau und die Durchführung einer Kommunikationsbeziehung zu den Wissensträgern unterstützen. Für organisationsübergreifende Workflows bedeutet dies, dass man gezielt Wissensträger ansprechen können soll, von denen man weder Namen noch Abteilung weiß. Das System soll sowohl Prozesswissen als auch Funktionswissen managen. Als Prozesswissen bezeichnen wir das Wissen über den Prozessablauf, beteiligte Rollen, Personen, Organisationseinheiten, notwendige Daten und Ressourcen. Als Funktionswissen bezeichnen wir Wissen, welches für die Durchführung einzelner Prozessschritte notwendig ist. WfMS unterstützen normalerweise nur das Prozess-

und vernachlässigen das Funktionswissen. In einem B2B-Kontext verstehen wir unter Prozesswissensmanagement auch, dass der verteilte Wf-Entstehungsprozess unterstützt wird. Funktionswissen soll explizit gemacht werden können, um es entweder beim Scheduling oder zum Zwecke des Skillmanagement zu nutzen. Durch die erweitete Funktionalität gegenüber traditionellen WfMS gibt es weitere Auswertungs- und Darstellungsmöglichkeiten. Hier ist eine offene Schnittstelle gefordert, so das beispielsweise Skillmanagement eingefügt werden kann.

3 Peer-to-Peer-Technologie

Die Peer-To-Peer-Technologie ist ein Spezialfall der traditionellen Client/Server Technologie, wobei im Idealfall jeder Knoten im Netz sowohl Client als auch Serverfunktionalität hat. Im Unterschied zum üblichen Client/Server-Ansatz sind somit alle Knoten prinzipiell gleichberechtigt. Um eine besondere Robustheit zu erreichen, agieren die Knoten(Peers) weitgehend autonom, d.h. sie können selbst entscheiden, sich z.B. vom Netz abzutrennen oder welche Dienste sie zur Verfügung stellen. Bei Kommunikation zwischen Peers gibt es keine zentrale Instanz, die die Nachrichten filtert und verteilt etc. Stattdessen kommunizieren die Peers „direkt“, was nicht ausschließt, dass andere Peers vermittelnd tätig sind, d.h. die Nachricht annehmen und weiterleiten. Entscheidend ist, dass es keine *zentrale* Vermittlungsstelle gibt. Typische Anwendungen im P2P-Bereich sind: File-Sharing, Instant-Messaging(IM), und kollaborative Anwendungen. Beim P2P-Filesharing werden die Daten direkt zwischen den Peers ausgetauscht und nicht in einem zentralen Server zwischengehalten. IM ermöglicht eine „getippte“ Kommunikation in „Echtzeit“, beim P2P-Messaging werden die Nachrichten wiederum direkt zwischen den Peers ausgetauscht. File-Sharing und IM sind beides Funktionen, die sinnvoll in einem kollaborativen Kontext eingesetzt werden. Andere Funktionen, die mit P2P-Technologie realisiert werden können, sind Chatrooms, elektronische Foren, Co-Authoring etc. Als spezifische Ausprägung des P2P-Ansatzes haben wir die JAVA-Implementierung des JXTA-Protokolls gewählt, welche im Rahmen eines von SUN initiierten Projekts entstanden ist. JXTA erlaubt es unter anderem, an einem Peer Dienste (Services) anzubieten und diese durch sog. Advertisements bekannt zu geben. Andere Peers können diese Dienste suchen und mit dem Diensterbringer über eine virtuelle Verbindung (Pipe) kommunizieren. Die Verbindungen werden virtuell genannt, weil sie auf konkrete Protokolle wie TCP oder HTTP abgebildet werden. Pipes werden mit Endpunkten verbunden, die wiederum von konkreten IP-Adressen abstrahieren. Auf diese Weise wird ein unabhängiger Adressierungsschema etabliert, welches sicherstellt, dass die Kommunikation auch dann reibungslos funktioniert, wenn die IP-Adressen dynamisch zugeteilt werden. Gateway-Peers erlauben die Überwindung von Firewalls, indem sie TCP- mit HTTP-Pipes verbinden. Firewalls sind in den meisten Fällen so konfiguriert, das ausgehende HTTP-Anfragen wie sie von üblichen Webbrowsern erzeugt werden, zugelassen sind. Eingebunden in die JAVA-Implementierung von JXTA ist eine Security-Suite, die Verschlüsselung, Peer-Authentifizierung etc anbietet.

4 Architektur des neofonie B2B-WfMS *:flow*

Die grobe Architektur der Wf-Komponente „neofonie architect:flow“ besteht aus zwei P2P-Ebenen. Organisationen, die an Wfs partizipieren, werden durch Peers (Typ A) repräsentiert, die bei der Bearbeitung von Wfs direkt miteinander kommunizieren. Jeder dieser Peers ist auf tieferer Ebene ein Typ-B-Peer in dem unternehmensinternen P2P-Netz. Unternehmensinterne Peers schließlich bieten Dienste an, die bei der Bearbeitung eines Wfs in Anspruch genommen werden. Siehe dazu Abbildung 1 linke Seite.

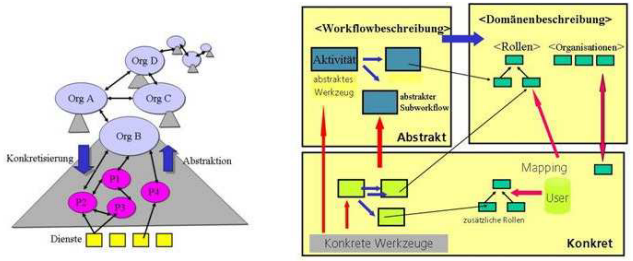

Abbildung 1.

Das System ist XML-basiert, d.h. sowohl das Wf-Schema, als auch die Wf-Instanz wird als XML-Dokument repräsentiert. Jede Wf-Instanz enthält die zu bearbeitenden Objekte als Payload (virtuelle Umlaufmappe) und wandert im Zuge der Bearbeitung von einem Peer zum nächsten. Dabei kann die Instanz bei einem Split auf verschieden Peers verteilt und bei synchronisierenden Joins wieder an einem Peer zusammengeführt werden. Die Auswahl der Peers wird entsprechend des Wf-Schemas von der Wf-Engine vorgenommen. Dabei werden gegebenenfalls die Ergebnisse der bereits vollzogenen Aktivitäten berücksichtigt, z.B. bei Fallunterscheidungen. Die Menge der geeigneten Peers wird durch Ausnutzung des JXTA-Protokolls ermittelt: Aktivitäten entsprechen auf JXTA-Ebene Diensten und werden daher per Advertisement bekanntgegeben und können gesucht werden. Entscheidend zur Umsetzung vieler der genannten Anforderungen ist das zugrundeliegende *Domänenkonzept*, welches sehr ähnlich zu dem in [7] beschriebenen Konzept ausgelegt ist. Jeder Wf wird einer Domäne zugeordnet; innerhalb einer Domäne können verschiedene Wfs definiert sein. In der Domänenbeschreibung werden die partizipierenden Organisationen genannt und die vorkommenden Rollen deklariert. In dem Wf-Schema wird die Domäne benannt und für jede Aktivität angegeben, unter welcher Rolle sie ausgeführt

wird. Weiterhin wird dort für jede Aktivität ihre Signatur als XML-Schema beschrieben. Damit werden die Ein- und Ausgabeparameter festgelegt, sowie der Teil des Payloads identifiziert, auf den zugegriffen wird. Ergänzend wird die genaue Zugriffsart (z.B. erzeugend, nur lesend) pro Aktivität festgelegt. Die feingranulare Zugriffskontrolle stellt sicher, dass ein Benutzer nur dann die spezifizierten Zugriffsrechte erhält, wenn er in der Rolle aktiv ist und die Aktivität auch gerade ausführt.(vergl. [7]) Der Standard XMLDSIG wird verwendet, um die jeweils bearbeitenden Teile der Wf-Instanz userspezifisch zu signieren. Einige der Sicherheitsanforderungen können mithilfe der JXTA-Security-Suite erfüllt werden, so ist beispielsweise TLS als Transportverschlüsselung eingebunden, die Authentifikation von Peers wird durch Peer-Zertifikate sichergestellt. Eine ausführlichere Darstellung zu den Sicherheitsanforderungen und ihre Realisierung findet sich in [8]. Transaktionale Wf werden durch Einbindung des Business Transaction Protocols unterstützt. Um die Autonomieanforderungen zu erfüllen sind Wf-Schemas in mehrfacher Weise unterspezifiziert. Beim Übergang von Typ-A auf TyP-B-Peers, also beim Eintritt in ein Unternehmen wird bezüglich der unterspezifizierten Anteile eine Konkretisierung vorgenommen. Beim Austritt aus dem Unternehmen wird umgekehrt durch einen Abstraktionsschritt die Konkretisierung wieder rückgängig gemacht. Der Zusammenhang ist in der Abbildung 1 auf der rechten Seite dargestellt. Im Einzelnen betrifft das folgende Bestandteile:

- **Rollen abstrahieren von konkreten Benutzern und Organisationsstruktur.** In dem Wf-Schema wird für jede Aktivität die Rolle angegeben, unter der sie ausgeführt wird. Rollen werden in der Domänenbeschreibung abstrakt definiert. Die Rollenbeschreibung besteht dort nur aus einem Namen, einer natürlichsprachlichen Beschreibung sowie der Angabe von Superrollen. Dadurch, dass die Rollen domänenspezifisch definiert sind, haben sie keinen Bezug zur Organisationsstruktur der beteiligten Organisationen. Die Anbindung von Usern geschieht unternehmensintern durch eine Zuordnung von Usern zu Rollen. An dieser Stelle kann die Benutzerverwaltung durch LDAP, NT oder UNIX-Systeme eingebunden werden. Diese Systeme fassen Benutzter zu Gruppen zusammen. Häufig reflektieren die Gruppen auch die Organisationsstruktur. Durch Ausnutzung des Gruppenkonzepts können bei der Zuordnung nicht nur einzelne Benutzer, sondern Gruppen, und damit gegebenenfalls organisatorische Einheiten, berücksichtigt werden.
- **abstrakte Werkzeuge.** Ähnlich wie MIME-Typen erlaubt das System das verwendete Werkzeug im Schema zunächst abstrakt festzulegen, z.B. „Textverarbeitung“. Die Konkretisierung (z.B. „MS-Word“) findet in diesem Fall für die einzelnen Peers statt. Aus Sicherheitsgründen können für bestimmte Aktivitäten auch spezifische Programme fest vorgeschrieben werden, die nicht mehr verändert werden können.
- **unterspezifizierte Sub-Workflows.** Unternehmensinterne Teilworkflows können in dem Wf-Schema unterspezifiziert bleiben, die konkrete Realisierung wird in einem extra Dokument modelliert und ist nach außen nicht sichtbar. Auf diese Weise können verschiedene Organisationen mit unterschiedlichen Realisierungen arbeiten, ohne das Einzelheiten der Realisierung bekannt gegeben werden müssen.

5 Wissensmanagementunterstützung

In diesem Abschnitt soll auf spezielle Systemfunktionen eingegangen werden, die :flow besonderes für Wissensmanagement geeignet machen, insbesondere im Kontext von den Anforderungen bei interorganisationalen Workflows.

5.1 Management von Prozesswissen

Jeder Wf ist einer Domäne zugeordnet. Für jede Domäne gibt es eine Metadomäne und einen Meta-Wf, die die kollaborative Entwicklung eines Wfs ermöglichen. Der Payload des Meta-Wfs besteht aus dem sich in der Entwicklung befindlichen Wf-Schema des Objekt-Wfs. Dadurch, das der Entstehungsprozess selbst als Wf beschrieben ist, ist er äußerst flexibel und kann leicht an die jeweiligen spezifischen Anforderungen angepasst werden. Dieser Ansatz ist wiederum für die geografisch verteilte Entwicklung sinnvoll, wie sie bei B2B-Workflows auftritt. Weiterhin werden Wf-Schemas in einem Repository abgespeichert, auf dem fallbasierter oder kategorienbasierter Zugriff besteht. Damit ist bei der Entwicklung von neuen Wfs das gesamte explizite Prozesswissen verfügbar (Kodifizierungsstrategie). Zusätzlich können die jeweils Verantwortlichen kontaktiert werden (Personifizierungsstrategie).

5.2 Management von Funktionswissen

Ähnlich wie in [9] dargestellt, erlauben wir für jede Rolle/Aktivität/Anwendung die explizite Annotierung von benötigten Fähigkeiten/Fertigkeiten/Kenntnissen (Skills). Zwischen Anwendung, Aktivität, Rolle besteht dabei eine Vererbungsrelation. Diese Modellierung kann benutzt werden um Wissenslandkarten über Wfs-Rolle-Skills zu generieren oder einen Erfahrungsindex der Mitarbeiter aufzubauen oder andersrum bei der Auswahl des Sachbearbeiters vom System berücksichtigt werden. Der GP liefert den Kontext für Wissensprozesse. Dieses kann im System ausgenutzt werden, wenn man die Wissensprozesse explizit mitmodelliert, und den Kontext zugänglich macht. Z.B. können automatisch Suchmasken entsprechend der aktuellen Instanz ausgefüllt werden. Ein ähnliches Vorgehen ist in [6] dargestellt. Ein Wf-Schema hat möglicherweise viele Instanzen. Diese werden in einer von neofonie entwickelten XML-Datenbank abgelegt. Fallbasiertes Retrieval liefert zu der aktuellen Instanz die ähnlichsten Fälle, die in der Vergangenheit bearbeitet wurden.

5.3 Kommunikationsmanagement

Das System unterstützt eine Vielzahl von Kommunikationsmöglichkeiten. Der Payload kann zur Laufzeit um Kommentare,Fragen und Anregungen erweitert

werden, die dann weiteren Bearbeitern zugänglich sind. Überhaupt es möglich derartige Fragen, Anmerkungen zu formulieren und über unterschiedliche Kanäle zu verteilen. Die P2P-Technlogie macht es leicht, Anwendungen wie IM, Gruppen-Chat und Diskussionsforen zu realisieren. Natürlich ist auch eMail verfügbar. Durch das (Meta)Domänen- und Rollenkonzept sind vielfältige Adressatenkreise verfügbar. **Instanzspezifisch:** Es können gezielt Bearbeiter kontaktiert werden, die einen bestimmten Arbeitsschritt in der vorliegenden Instanz bearbeitet haben. Es kann eine Nachricht an die Gruppe aller, die bisher die Instanz bearbeitet hat, gesendet werden. Schließlich kann eine Nachricht an alle geschickt werden, die diese Instanz bereits bearbeitet haben oder zukünftig bearbeiten werden. Man beachte, dass sich die Adressierung nicht nur auf die aktuelle Instanz bezieht, sondern für beliebige Instanzen gilt. Wenn das oben genannte fallbasierte Retrieval beispielsweise eine ähnliche Instanz geliefert hat, kann der Sachbearbeiter den Kollegen kontaktieren, der den entsprechenden Arbeitsschritt bei dem ähnlichen Fall bearbeitet hat. **Schemaspezifisch:** Es kann eine Nachricht an alle geschickt werden, die potentiell Instanzen des Schemas bearbeiten. Der Empfängerkreis kann durch Auswahl von bestimmten Rollen eingeschränkt werden. Auf diese Weise kann z.B. eine Diskussion über Probleme bei der Bearbeitung einer bestimmten Aktivität geführt werden, wobei alle Personen adressiert werden, die auf Grund ihrer Rollenzuordnung die Aktivität durchführen können. **Meta-Wf-Spezifisch:** Es kann eine Nachricht an alle geschickt werden, die das Wf-Schema entwickelt haben. Hier können Fragen oder Verbesserungsvorschläge für den Workflow diskutiert werden. **Domänenspezifisch:** Es kann eine Nachricht an alle gesendet werden, die eine bestimmte Rolle innerhalb der Domäne innehaben. Das ist eine workflowübergreifende Kommunikation, da sich innerhalb einer Domäne mehrere Workflows befinden können. **Metadomänenspezifisch:** Es kann eine Nachricht an die Metadomäne geschickt werden, d.h. an diejenigen, die die vorkommenden Rollen festgelegt haben.

Für alle oben genannten Adressierungsvarianten kann durch Festlegung auf bestimmte Organisationen der Empfängerkreis weiter eingeschränkt werden. Damit ist es beispielsweise möglich eine Nachricht an die unternehmensinternen Bearbeiter eines Workflowschemas zu schicken. Abschließend wollen wir die Kommunikationsmöglichkeiten an Hand eines Szenarios erläutern: Ein international tätiges Kreditunternehmen verfasst in der Zentrale Kreditrichtlinien, die dann zu Umsetzung an die einzelnen Länderniederlassungen gehen. Die Kreditrichtlinien lassen noch einen gewissen Spielraum bei der Umsetzung zu. Der italienische Sachbearbeiter einer bestimmten Kreditrichtlinie kann jetzt seinen Kollegen in Deutschland kontaktieren, der genau dieselbe Richtlinie umsetzten muss, ohne den Namen des Sachbearbeiters, die Abteilung etc wissen zu müssen. Die Flexibilität der Kommunikation kann nochmals gesteigert werden, wenn die Kommunikationsbeziehungen wiederum als Wf modelliert werden, damit ist es z.B. möglich Folgendes auszudrücken: Sende eine Frage an eine bestimmte Gruppe, wenn drei Antworten eingegangen oder Zeitschranke überschritten ist, dann lösche die Frage aus der Liste der Empfänger, die die Nachricht noch nicht gelesen haben.

6 Bewertung der P2P-Technologie

Die Bewertung des Einsatzes von P2P-Technlogie ist immer ein kritischer Punkt. Natürlich kann man ein System mit gleicher Funktionalität auch auf eher traditionellem Wege (Client-Server-Architektur) realisieren. In der beschrieben Architektur kommt das Dokument zu dem Benutzer, der es zu bearbeiten hat, in einer Client-Server Architektur ist es normalerweise andersherum: der Nutzer wendet sich mit Hilfe eines Clientprogramms an den Server und geht damit zum Dokument. Das in dem Zusammenhang der Bewertung von P2P-Systemen häufig vorgetragene Argument der strukturellen Analogie ist aus unserer Sicht bestenfalls motivierend für den Einsatz von P2P. Eine Bewertung müsste entlang vorher definierter Kriterien vollzogen werden, und zwar vergleichend zwischen Systemen mit verschiedenen Architekturen. Abgesehen von dem großen Aufwand der mehrfachen Realisierung ist ein solcher Vergleich immer noch nur begrenzt aussagefähig: Schließlich werden konkrete Systeme auf Basis unterschiedlicher Architekturmodelle verglichen und nicht die Architekturmodelle an sich. Letztendlich wird sich eine Art empirische Bewertung ergeben, die sich auf Grund von Erfahrungen beim Design von Systemen entwickelt. Eigentlich sehen wir in P2P auch nur eine Spielart des Client-Server-Ansatzes und Systeme werden in vielen Fällen hybrid sein, also Aspekte haben, die man eher einer Client-Server-Architektur zuordnen würde. Das ist im vorliegenden Fall nicht anders. Einige Aufgaben bei einer WfMS erfordern eher eine zentrale Komponente, und widersprechen damit dem P2P-Gedanken. Dazu gehören eine Überwachungskomponente, die es erlaubt zu jeder Wf-Instanz Informationen zur Verfügung zu stellen, welche Aktivitäten gerade ausgeführt werden. Andere Beispiele sind die Repository-Komponente sowie Unterstützung von Transaktionen. In unserem System unterscheiden wir deshalb als Hostrechner für Peers Server von Arbeitsplatzstationen von Mitarbeitern. Einige Dienste werden auf „immer verfügbaren“ Servern lokalisiert. Es gibt einige Bereiche, wo sich P2P in der Ausprägung von JXTA als günstig erweist. Als Vorteil kann verbucht werden, dass P2P-Architekturen dem Autonomiebedürfnis der Organisationen stärker gerecht werden: Es gibt keinen zentralen Server außerhalb der Organisation, von dessen Ausfallsicherheit die Organisation abhängig ist. Ist auf der anderen Seite gewünscht, die Administration des WfMS outzusourcen, ist möglicherweise aber doch ein zentraler Server erwünscht. Ein Vorteil der verwendeten P2P-Technlogie ist die Unterstützung von mobilen Anwendern und Geräten. Das betrifft den Aspekt der Offlinebearbeitung sowie der Kommunikation. Ein User kann sein Notebook unter beliebiger IP-Adresse mit dem Netz verbinden und einen Task annehmen. Die Bearbeitung des Tasks kann offline geschehen, sofern das benötigte Werkzeug lokal vorhanden ist. Sobald das Gerät wieder IP-Kontakt hat, wird das Ergebnis übermittelt. JXTA überlagert bestehende Netzwerkarchitekturen mit einem virtuellen Overlaynetzwerk. Auf dieses Weise ist es prinzipiell möglich auf Protokolle für Ad-Hoc-Vernetzung wie Bluetooth aufzusetzen. Weiterhin erscheint P2P-Technologe gut geeignet für die Realisierung von ergänzender kollaborativer Kommunikationsunterstützung, wie sie aus Wissensmanagementsicht erwünscht

ist. Die Beurteilung der Fähigkeit von JXTA, durch Firewalls durch Ausnutzung von http zu tunneln, wird je nach Security-Policy unterschiedlich ausfallen.

7 Ähnliche Ansätze und Ausblick

Wir haben ein B2B-WfMS vorgestellt, welches auf Basis von P2P-Technologie implementiert ist und besondere Unterstützung im Bereich WM bietet. Es gibt eine Reihe von Arbeiten im Bereich prozessorientiertes WM [2]. Für System :flow kennzeichnend ist die Unterstützung in der Designphase sowie das umfangreiche Kommunikationsmanagement. Informationen über P2P-basierte WfMS findet man bisher nur wenige, z.B. in [10]. Unter der URL s2s.neofonie.de [12] findet man unter dem Stichwort „S2S Interaktiv“ eine Beispielanwendung von :flow, bei der eine Kommunikationsbeziehung explizit modelliert ist. Eine naheliegende Weiterentwicklung des Systems liegt darin, neben dem Workflow auch die Softwarewerkzeuge über das JXTA-Netzwerk zu verschicken.

Literatur

1. Aalst, W.M.P., Hofstede,A.H.M. , Kiepuszewski,B., Barros,A.P.: :Workflow Pattern (2002)
2. Abdecker, Andreas, Hinkelmann, Knut, Maus, Heiko, Müller, Hans Jürgen *Geschäftsprozessorientiertes Wissensmanagement* Springer Verlag, ISBN 3-540-42970-0, (2002)
3. Böhm, Markus: *Entwicklung von Worklfow-Typen* Springer Verlag, ISBN 3-540-66394-0 (2000)
4. Bussler, Christoph: *B2B Integration. Concepts and Architecture.* Springer Verlag, ISBN 3-540-43487-9 (2003)
5. Hansen, M.T., Nohria,N., Tierney,T.: *Whats's your Stragegy for Knowledge Management* In: Harward Business Review, March-April 1999,pp.106-116
6. Hinkelmann, Knut, Kraragiannis Dimitris, Telesko Rainer: *PROMOTE-Methodologie und Werzeuge für geschäftsproessorientiertes Wissensmanagement* in [2]
7. Kang, Myong ,H., Park, Joon s., Froscher, Judith N.: *Access Control Mechanism for Inter-organisation Workflow*
8. Küssner,U.: *Sicherheitsanforderungen bei B2B-Workflows: Realisierung mit JXTA-Security und XML-Security-Standards.* JAVA-Spektrum Ausgabe 1, Januar/Februar '04 (2004)
9. Nägele,Rainer, Schreiner, Peter: *Potenziale und Grenzen von Business Process Management Tools für geschäftsprozessorientiertes Wissensmanagement* in [2]
10. Noll,J.: *Process Enactment in Virtual Software Organisations*
11. Nonaka, Ikujiro, Takeuchi, Hirotaka; *Die Organsisation des Wissens* Campus Verlag, ISBN 3-593-53643-0 (1997)
12. Wertlen, Ron: DFN Science-to-Science: Peer-to-Peer Scientific Research Terena Networking Conference 2003 (2003), Zagreb, Coratia

Typische Integrationsszenarien und deren Unterstützung durch Web Services und andere Technologien

Cornelia Boles, Jörg Friebe, Till Luhmann

OSC – IM Systems AG
Industriestr. 11
26121 Oldenburg
cornelia.boles@osc-ims.de, joerg.friebe@osc-ims.de, till.luhmann@osc-ims.de

Abstract. Die Integration existierender und neu einzuführender Anwendungen ist heute eine Kernaufgabe im Bereich der IT. Um für das jeweilige Integrationsszenario die geeignete Integrationstechnologie auswählen zu können, müssen die Integrationsszenarien und die daraus resultierenden Anforderungen an die Integrationstechnologien genauer untersucht werden. Der vorliegende Artikel beschreibt nach einer einführenden Betrachtung der möglichen Integrationsebenen die charakteristischen Merkmale von Integrationsszenarien. Im Anschluss daran werden die wichtigsten Integrationstechnologien kurz vorgestellt und bezüglich ihrer Eignung für die einzelnen Merkmalsausprägungen der Integrationsszenarien bewertet. Der Artikel schließt mit einer Zusammenfassung und einem Ausblick.

1 Einleitung

Das Thema Web Services wird derzeit in vielen Publikationen und auf großen Konferenzen behandelt. Doch Web Services sind längst kein reines Forschungsthema mehr. Vielmehr werden sie vermehrt in Unternehmen eingesetzt, in erster Linie zur Integration von Anwendungen. Die Vielzahl der im Umfeld von Web Services entstehenden Standards (z.B. zu den Themen Sicherheit, Transaktionen oder Web Services Choreographien) zeigt, dass die Technologie heute einerseits noch nicht ganz ausgereift ist, sie aber andererseits als Basis angesehen werden kann, auf der aufbauend weitere Technologien entstehen werden. Die Kernaspekte von Web Services sind jedoch soweit standardisiert, dass sie sich auch in Produktivumgebungen einsetzen lassen.

Im Gebiet der Anwendungsintegration stellt sich die Frage, wie sich Web Services im Vergleich zu etablierten Integrationstechnologien und Werkzeugen (z.B. aus dem Enterprise Application Integration Umfeld) positionieren. Um diese Frage beantworten zu können, werden im vorliegenden Artikel Integrationsszenarien genauer untersucht und anhand ihrer charakteristischen Merkmale klassifiziert.

Der Artikel gliedert sich dabei wie folgt: Nach einer grundlegenden Betrachtung der Ebenen, auf denen Softwaresysteme integriert werden können, werden in die charakteristischen Merkmale von Integrationsszenarien diskutiert. Der letzte Abschnitt stellt bestehende Integrationstechnologien vor und geht darauf ein, für welche

Integrationsmerkmale die verschiedenen Technologien geeignet sind. Der Artikel schließt mit einer Zusammenfassung und einem Ausblick.

2 Integrationsebenen

Die Integration von Softwaresystemen in der Informationsverarbeitung kann mindestens auf den Ebenen der Daten, Anwendungen und Prozesse betrachtet werden [BFGH02]. Die dabei relevanten Aspekte werden im Folgenden dargelegt.

Datenintegration

Grundlegende Techniken zur Integration von Informationssystemen sind auf der Ebene der Datenbestände bzw. auf Basis der Schemata zugrunde liegender Datenbanken angesiedelt. Ziel ist fast immer die Sicherstellung anwendungsübergreifender Konsistenz der Daten. Eine erste naheliegende Lösung sieht die Einführung eines unternehmensweit integrierten Datenschemas vor. Die Idee ist hier, nur eine Datenhaltungskomponente und nur ein Datenschema zu verwenden und Anwendungen so zu entwickeln, dass sie auf dieses Datenschema aufsetzen. Somit können die Konsistenzsicherungsmechanismen des DBMS genutzt werden. Dieser Ansatz ist jedoch nur sinnvoll verwendbar, wenn Anwendungen neu entwickelt bzw. reimplementiert werden sollen oder können. Die oft in Unternehmen vorzufindende heterogene Systemlandschaft macht andere Ansätze erforderlich.

Zur Datenintegration wird in heterogenen Systemlandschaften häufig Datenzugriffsmiddleware eingesetzt. Sie ermöglicht neben einer für Anwendungen transparenten Einbindung externer Daten den Erhalt vorhandener Zuständigkeiten für Datenpflege und -verwaltung sowie die Wahrung der Autonomie existierender Anwendungen. Verwendete Technologien sind hier Open Database Connectivity (ODBC) und deren Nachfolgertechnologien wie OLEDB und ADO.net sowie die Java Database Connectivity (JDBC) und deren weiterführende Entwicklungen wie Entity Beans im J2EE-Umfeld. Die Integration der Schemata erfolgt hier somit lediglich auf logischer Ebene. Als Voraussetzung für die Integration externer Datenbestände müssen Systemhersteller die Semantik der Datenschemata offen legen bzw. den direkten Zugriff auf den Datenbestand freischalten. Ist dieser Zugriff nur über spezielle Application Programming Interfaces (APIs) möglich, so sind Ansätze, die nur auf Mechanismen beteiligter (relationaler) Datenbankmanagementsysteme (DBMS) basieren, nicht anwendbar.

Anwendungsintegration

Neben der Integration von Schemata und Daten verschiedener Anwendungen ist jedoch auch die Integration der Anwendungen selbst wichtig. Ziel ist dabei einerseits die oben genannte anwendungsübergreifende Konsistenz der Daten. Andererseits wird die möglichst weitgehende Wiederverwendung und Nutzung von vorhandener Funktionalität aus den integrierten Systemen (funktionale Integration) sowie eine einheitliche Benutzungsoberfläche (GUI-Integration) angestrebt. Als Basis für die Systemkopplung dienen Integrationsplattformen, welche die erforderlichen Dienste bereitstellen. Derzeit bieten sich mit .net [MSOFT03] und J2EE [Mon02] zwei Integrationsplattformen an, die die notwendige Infrastruktur bereitstellen. Mit Einschränkungen kann auch CORBA [OMG02] hier genannt werden, doch insbesondere in Bezug auf GUI-Integration (Nutzung von Web-Frontends) sowie Verfügbarkeit und Unterstützung der Schnittstellen durch Anwendungen und Komponenten (Einbettung von Visual Basic bzw. zahlreicher Java-Pakete und APIs) bieten J2EE und .net hier Vorteile.

Oft kommt bei der Anwendungsintegration Message-Oriented Middleware [Bern96] zum Einsatz. J2EE bietet hier das Konzept der Message Queues, Publish/Subscribe-Mechanismen und Message-Driven Beans. Die damit realisierbaren Architekturen verbinden funktionale Integration mit den Vorteilen asynchroner Kommunikation (Send and Forget) und erlauben eine lose Kopplung der beteiligten Anwendungen, denn oft besteht die Anforderung, dass Systeme durch Downzeiten angekoppelter Systeme nicht in ihrer eigenen Verfügbarkeit beeinträchtigt werden dürfen. Das Nachrichtenformat ist typischerweise XML-basiert, um die Einbindung weiterer Systeme zu erleichtern. Die funktionale Integration allein reicht jedoch nicht aus, da hierdurch zwar Funktionalität wiederverwendet werden kann, nicht aber Dialogelemente, die zur Parametrisierung bzw. Präsentation der Funktionalität nötig sind. Aus diesem Grund wird die Integration der Anwendungen auch auf der Ebene der Benutzungsoberflächen durchgeführt. Hierzu sind mehrere Wege denkbar. Der klassische Weg führt zur Integration von GUI-Controls, wie sie als Microsoft Foundation Classes, Java AWT oder Swing bekannt sind [Sun02]. Teilweise lassen sie sich auch im Java-Applet-Umfeld nutzen. Ist jedoch der Einsatz im Internet erforderlich (z.B. im B2B-Umfeld oder im Bereich Customer Self Service), so sind schlanke, nahezu überall lauffähige Frontends erforderlich, die auf den Client-Rechnern keine Arbeiten (Installation, Administration) erfordern. Die Wahl fällt hier auf HTML-Clients, die z.B. mit den Technologien ASP.net bzw. JSP realisiert werden können.

Prozessintegration

Daten- und Anwendungsintegration reichen nicht aus, wenn komplexe Geschäftsprozesse unterstützt werden müssen, die zudem einem ständigen Wandel unterliegen. In diesem Fall muss es mit einfachen Mitteln möglich sein, die Geschäftsprozesse um Aktivitäten zu erweitern oder anzupassen. Hierzu bedarf es jedoch einer übergeordneten Steuerungsebene, mit der anwendungsübergreifende Geschäftsprozesse realisiert werden können. Um dies umzusetzen, werden Konzepte aus den Bereichen Workflowmanagement (WFM) und Enterprise Application Integration (EAI) bemüht. Hierbei werden die Workflow-Schemadaten, Workflow-Instanzdaten sowie Akteure und Aktionen zentral in einem Prozesssteuerungssystem verwaltet. Je nach Art der Prozesse kann dies ein Workflowmanagementsystem (WfMS) oder eine EAI-Engine sein. Im Mittelpunkt der Prozessintegration stehen die betriebswirtschaftlichen und auf den Kunden ausgerichteten Geschäftsprozesse. Ziel der Prozessintegration ist die vollständige elektronische Bearbeitung aller kundenorientierten Prozesse innerhalb des Unternehmens. Zentrale Aspekte sind hier die automatischen Aktivitäten und die

Arbeitskorb-Metapher. Während erstere völlig ohne Eingriff des Anwenders durch das System gesteuert werden (durch asynchrone Nachrichtenkommunikation, in Einzelfällen auch durch synchronen Aufruf von z.B. Session Beans), stellt der Arbeitskorb die zentrale Schnittstelle zwischen Prozess und Benutzer dar. Auch hier steht die asynchrone Bearbeitung im Vordergrund. Wird z.B. in einem Geschäftsprozess nach Ausführung von automatischen Aktivitäten eine Bearbeitung durch einen Anwender notwendig, so erhält dieser (oder seine Rolle) im Arbeitskorb eine entsprechende manuelle Aktivität zugeordnet. Die Aktivität ist z.B. anhand von Web-Formularen durchzuführen, die vom System dem Anwender prozessgesteuert vorgelegt werden. Nach der Bearbeitung übernimmt das System die weitere Steuerung des Prozesses.

3 Charakteristika eines Integrationsszenarios

Betrachtet man die in konkreten Projekten auftretenden Integrationsszenarien genauer, so stellt man schnell fest, dass sie sich oftmals nicht genau einer Integrationsebene zuordnen lassen. Vielmehr wachsen die Integrationsebenen zusammen, so dass die Integration auf mehreren Ebenen erfolgt. Damit reicht die Klassifikation nach Integrationsebenen nicht aus, um Integrationsszenarien zu charakterisieren und die für das jeweilige Szenario geeignete Integrationstechnologie auszuwählen.

Um dennoch die Auswahl der geeigneten Technologie unterstützen zu können, haben die Autoren die charakteristischen Merkmale von Integrationsszenarien analysiert. Dabei haben wurden sechs Merkmale identifiziert: Automatisierungsgrad und Benutzerinteraktion, Strukturiertheit der Daten, Dynamik der Daten, Anzahl der zu integrierenden Systeme, Kommunikationsparadigma und Status und Zustandshaltung, die im Folgenden genauer erläutert werden.

Automatisierungsgrad und Benutzerinteraktion

Zwei fast immer in Integrationsszenarien relevante, zudem komplementäre Aspekte sind der Grad der Automatisierbarkeit von Prozessschritten und der Grad der benötigten Benutzerinteraktion. Ist der Grad der Automatisierbarkeit hoch, so können die einzelnen Prozessschritte bzw. Aktivitäten ohne Interaktion durch einen Bearbeiter durchgeführt werden. Ziel der Geschäftsprozessoptimierung ist es typischerweise, diesen Grad zu erhöhen. Ein hoher Automatisierungsgrad bedeutet zudem, dass Aspekte wie Benutzerverwaltung, Rollen, Authentifizierung weniger relevant sind. Einzelne Aktivitäten können vom System direkt ohne Rückkopplung durch den Anwender durchgeführt werden. Dies bedeutet auch, dass die Lokalisation der Aktivität unkritisch ist, und z.B. von Loadbalancing-Strategien bestimmt werden kann (serverbasierte Durchführung automatischer Aktivitäten) und dass synchrone Kopplung nicht notwendig ist (da kein Benutzer unmittelbar auf Rückmeldung wartet). Ein hoher Automatisierungsgrad erfordert effiziente Konzepte für Interprozesskommunikation, Multithreading und Schnittstellen zu zu steuernden Komponenten (APIs).

Dem entgegen steht der Grad der notwendigen Benutzerinteraktion bei der Bearbeitung von Geschäftsprozessen. Notwendige Benutzerinteraktionen reduzieren den Grad der Automatisierbarkeit und sollten minimiert werden. Derartige Interaktionen können jedoch vielfältig sein, z.B. Synchronisationspunkte innerhalb weitgehend automatisierter Prozesse, an denen der Anwender automatisch erstellte Berechnungsergebnisse prüfen muss, bevor die nächste automatische Aktivität gestartet wird, Erfassung von zusätzlichen Daten über Bildschirmformulare oder manuelle Bearbeitung von Dokumenten (z.B. Textdokumente oder Grafiken). Benutzerinteraktionen erfordern einen Mechanismus, über den Aktivitäten einzelnen Anwendern oder Rollen zugeordnet werden können (Arbeitskorb-Metapher). Dieser Mechanismus wird insbesondere zur asynchronen Kopplung genutzt. Zur Einbindung von Formularen ist ein entsprechendes Framework nötig. Weiterhin sind clientseitige Schnittstellen zu Anwendungen notwendig, die zur Unterstützung bestimmter Aktivitäten (z.B. Dokumentbearbeitung) notwendig sind.

Der Grad der Automatisierbarkeit und Benutzerinteraktion variiert und ist vom Geschäftsprozess abhängig. In den seltensten Fällen liegen in Unternehmen die Extremsituationen (sehr hohe Automatisierung, sehr geringe Benutzerinteraktion bzw. sehr geringe Automatisierung, sehr hohe Benutzerinteraktion) vor.

Strukturiertheit der Daten

Ein wichtiger Aspekt bei der Integration ist die Struktur bzw. der Strukturierungsgrad der zu integrierenden Daten. In Unternehmen ist dabei typischerweise eine große Bandbreite an Daten anzutreffen, die in Geschäftsprozessen relevant sind. Zur Optimierung der Geschäftsprozesse ist es sinnvoll, den Anteil strukturierter Daten zu erhöhen und den Anteil an unstrukturierten Daten zu verringern, denn strukturierte Daten erlauben einen höheren Automatisierungsgrad, während unstrukturierte Daten fast immer erweiterte Benutzerinteraktionen, z.B. Interpretation, erfordern.

Beim Austausch auf Dateiebene sind die Daten häufig rudimentär formatiert, z.B. in Form kommaseparierter Werte. Diese lassen sich innerhalb einer Integrationsplattform durch File-Adapter einlesen und weiterverarbeiten. Ist die Struktur der Daten selbst komplex, wie z.B. bei EDI-Formaten, so gibt es heute zu einer XML-Darstellung praktisch keine Alternative, weil der individuelle Entwicklungsaufwand für das Einlesen und Parsen der Daten dadurch entfällt. Handelt es sich hingegen um Dokumente in proprietären Formaten, wie z.B. Word-Dateien, so werden die Daten in der Regel nur als Ganzes weitergeleitet oder archiviert. Zwar existieren inhaltliche Zugriffsmöglichkeiten über Objektmodelle, doch die individuellen Entwicklungsaufwände für eine inhaltliche Interpretation der Dokumente sind meist beträchtlich, so dass hierfür einfachere Mechanismen eingesetzt werden. Die letzte Gruppe der zu verarbeitenden Daten bilden die unstrukturierten Dokumente, wie z.B. Bilddateien, Faxe oder eingescannte Schriftstücke.

Dynamik der Daten

Als Datenbasis einer Integrationslösung kann im Prinzip die gemeinsame Datenbasis aller angeschlossenen Anwendungen angesehen werden. Ziel beim Betrieb einer Integrationsplattform sollte es sein, die Gesamtheit der Daten in einem konsistenten

Zustand zu halten. Da sich bei einer Vielzahl von angeschlossenen Systemen Datenredundanzen nicht vermeiden lassen, sind Abgleichsmechanismen zu implementieren. Hierbei muss sorgfältig zwischen der Häufigkeit von Datenänderungen, der Notwendigkeit einer Verfügbarkeit aktueller Daten in einzelnen Systemen und dem beim Datenabgleich entstehenden Kommunikationsaufwand abgewägt werden.

Bei bestimmten hochdynamischen Daten ist die Aktualität das entscheidende Wertmerkmal. Hierzu zählten Börsendaten, aber auch Wetterdaten. Entscheidend ist hier, dass die Daten nur in dem Moment der Messung bzw. Aufnahme wirklich aktuell sind. Daher müssen die Daten möglichst zeitnah beim Interessenten zur Wahrnehmung gelangen. Zum Beispiel werden hereinkommende Daten als Nachrichten entgegengenommen und sofort auf die Zielsysteme zum Abruf verteilt.

In anderen Fällen können zum automatisierten Datenabgleich mit mehreren Zielsystemen Publish-Subscribe-Mechanismen eingesetzt werden. Sofern die Änderungen plattformseitig gesammelt werden und der Abgleich nur selten stattfindet, ist es sinnvoll, die in einem System verfügbaren Daten mit einem Aktualitätszeitstempel zu versehen. Betriebsintern kann dem Bearbeiter die Möglichkeit eingeräumt werden, den Datenabgleich bei Bedarf manuell anzustoßen. Gängige EAI-Plattformen bieten hierfür entsprechende GUIs an.

Anzahl der zu integrierenden Systeme

Soll eine zentrale Integrationsplattform eingesetzt werden, um die Anzahl individueller Systemschnittstellen (Punkt-zu-Punkt-Verbindungen) zu reduzieren, so fällt auf, dass bei wenigen Systemen häufig mehr Schnittstellen von und zu der Integrationsplattform zu implementieren sind als ursprünglich Punkt-zu-Punkt-Verbindungen existieren. In der Regel ist nämlich nur ein Bruchteil aller möglichen Punkt-zu-Punkt-Schnittstellen sinnvoll und daher auch realisiert, wohl aber müssen als Voraussetzung für die Integration Kopplungen aller Systeme zur zentralen Plattform erstellt werden. Das Argument der Aufwandsreduktion ist somit erst ab einer größeren Anzahl von Systemen wirksam. Wird hingegen auf die höhere Qualität der Systemanbindungen und die Mehrwertdienste einer Integrationsplattform Wert gelegt, so tritt das Aufwandsargument in den Hintergrund. Daher kann eine plattformgestützte Integrationslösung bereits bei nur zwei Systemen sinnvoll sein.

Werden sehr viele Systeme integriert, so muss beachtet werden, dass mit der Anzahl der Systeme sich in der Regel auch die Technologien der Adapter immer stärker diversifizieren. Setzt die Plattform schwerpunktmäßig auf individuelle Adapterentwicklung, so können hier erhebliche Aufwände schlummern. Im Vorteil sind hier Plattformen, die bereits einen großen Satz von konfigurierbaren technologiespezifischen Adaptern mitbringen. Alternativ dazu bietet es sich an, Adapter auf Basis von Web Services zu realisieren. Hierdurch schafft man eine Anwendungskonnektivität, die unabhängig von der gewählten Integrationsplattform ist.

Sind die Systeme auf verschiedene, durch Firewalls getrennte Netzwerke verteilt, so kann die Integrationsplattform in jedem Netzwerk Kommunikationsserver platzieren, die über freigeschaltete Ports kommunizieren und damit eine Art Backbone für die Prozesskommunikation darstellen. Dies erlaubt es der Plattform theoretisch, Prozesse zentral administrierbar und überwachbar zu halten. Eine Alternative hierzu stellt ein zentraler Server dar, der über ein VPN direkt mit den verschiedenen Systemen kommuniziert.

Kommunikationsparadigma

Eine besonders lose Form der Kommunikation ist der Export und Import von Austauschdateien: Die Quellanwendung schreibt ihre Daten in ein File in einem Austauschverzeichnis, welches von der Zielanwendung oder von einem Importskript im Batchbetrieb regelmäßig geprüft wird („push and poll“). Gerade Altsysteme bieten häufig nur diese Art des Datenaustausches an. Auch beim Einsatz einer Integrationsplattform muss die Ankopplung dieser Systeme asynchron über individuell konfigurierte File-Adapter erfolgen. Insbesondere die koordinierte Verteilung der Daten auf mehrere Zielsysteme (z.B. beim Stammdatenabgleich) wird durch die Plattform besser unterstützt als z. B. auf der Ebene der Shellskripte. Als weitere Form der asynchronen Kommunikation z.B. mit externen Systemen werden EMails eingesetzt. Eingehende EMails werden über spezielle Adapter interpretiert und weiterverarbeitet, ebenso werden E-Mails plattformseitig inhaltlich generiert und versandt.

Verfügt eine Anwendung über ein Remote API oder handelt es sich um eine datenbankgestützte Anwendung, so stehen verschiedene Mechanismen der synchronen Kommunikation offen: Zunächst kann die Anwendung über einen individuell zu entwickelnden bzw. proprietären Systemadapter der Integrationsplattform angesprochen werden. Alternativ dazu bietet sich eine Systemanbindung mittels Web Services an: Für verschiedene Programmiersprachen (z.B. Java und .net-Sprachen) stehen heute mächtige Tools zur Verfügung, mit denen aus einem Remote API anwendungsspezifische Web Service Schnittstellen inkl. WSDL-Beschreibungen generiert werden können. Bei einer datenbankgestützten Anwendung ohne eigenes API ergibt sich als weitere synchrone Kommunikationsmöglichkeit der pragmatische Weg des direkten Datenbankzugriffs. Der zugehörige individuell konfigurierte Datenbankadapter kann seine Funktionsschnittstelle plattformseitig z.B. über ein Web Service „Facade“ anbieten.

Mit der Anzahl der integrierten Systeme steigt auch der Kommunikationsaufwand innerhalb der Plattform an. Eine Methode, um die Kommunikation zu reduzieren, ist der von Java her bekannte Publish-Subscribe-Mechanismus, der auch von heutigen Integrationsplattformen unterstützt wird. Dabei melden sich nur „interessierte“ Anwendungen beim Verteiler bestimmter, von der Plattform definierter Nachrichten an. Nützlich ist dieses Verfahren z.B. bei komplexeren Koordinierungsaufgaben, wie dem oben erwähnten Stammdatenabgleich.

Status und Zustandshaltung

Der Bearbeitungsstand von Geschäftsprozessen wird klassischerweise manuell über Auftrags- oder Laufzettel erfasst, die von Bearbeitungsinstanz zu Bearbeitungsinstanz weitergereicht werden. Werden solche Vorgänge automatisiert und mittels einer Integrationsplattform gesteuert, so sollten auch die Zustände der einzelnen Prozessinstanzen über diese Plattform fortgeschrieben werden. Hierbei können drei Arten von

Zustandsinformation unterschieden werden: Zunächst ist für jede Prozessinstanz ein globaler Prozesszustand (z.B. „aktiv“) definiert. Ergänzt wird dieser durch ein Protokoll der im Prozessablauf durchgeführten Aktivitäten und der beteiligten Akteure (Systeme, Bearbeiter), d.h. durch eine Beschreibung des Weges, auf dem der aktuelle Prozesszustand erreicht wurde (Historienfortschreibung). Hinzu kommen prozessspezifische Informationen, für welche das System die Möglichkeit einer individuellen Festlegung anbieten muss. Sämtliche Zustandsinformationen sollten durch das System automatisch instanziiert, initialisiert und aktualisiert werden.

Ein weiterer Faktor ist die Persistierung der Statusinformation, ohne die z.B. keine langlaufenden Geschäftsprozesse abgewickelt werden könnten (im J2EE-Umfeld wird diese Aufgabe standardmäßig von Stateful Session-Beans erfüllt). Sinnvoll ist ferner, dass ein Integrationssystem bei Prozessänderungen noch aktive alte und bereits gestartete neue Prozessinstanzen parallel betreiben kann. Hierfür muss das System die gleichzeitige Verwaltung der individuellen Zustandsinformationen beider Prozessversionen unterstützen.

4 Integrationstechnologien

Die Entscheidung für eine konkrete Integrationstechnologie ist abhängig von den charakteristischen Merkmalen des Integrationsszenarios. Prinzipiell lassen sich die Integrationstechnologien in die Gruppen „klassische“ EAI-Werkzeuge, Middleware-Technologien, Web Services und Web Services in Kombination mit Business Process Modeling Languages unterteilen. Tabelle 1 gibt einen Überblick über die verschiedenen Integrationstechnologien und zeigt, welche Integrationscharakteristika die einzelnen Technologien unterstützen. Zum Vergleich wird in der Tabelle zusätzlich die Punkt-zu-Punkt-Integration den übrigen Integrationstechnologien gegenübergestellt.

	EAI-Werkzeuge	**WfMS**	**Middleware**	**Web Services**	**Web Services + BPML**	**Punkt-zu-Punkt**
Hoher Automatisierungsgrad	++	+	–	–	++	+[1]
Viele Benutzerinteraktionen	o	++	–	–	+	+[1]
Unterstützung für strukturierte Daten	++	o	++	++	++	++
Unterstützung für unstrukturierte Daten	o	++	o	o	+	o
Eignung für Real Time Szenarien	+	+	++	o	o	++
Eignung für Batch-Prozesse	++	o	+	o	o	+[1]
Integration vieler Anwendungen	++	+	++	++	++	– –
Synchrone Kommunikation	++	+	++	++	++	+
Asynchrone Kommunikation	++	++	++	o	+	+
Unterstützung von Zustandshaltung	++	++	o	–	+	+[1]

[1] Bestandteil der beteiligten Anwendungen

Tabelle 1. Eignung der Integrationstechnologien für die verschiedenen Merkmalsausprägungen der Integrationsszenarien

Klassische EAI-Werkzeuge

EAI-Werkzeuge ermöglichen die Integration existierender Anwendungen auf Daten-, Anwendungs- und Prozessebene. Bei der Integration mittels eines EAI-Werkzeugs ist typischerweise für jede Anwendung, die integriert werden soll, ein Adapter zu erstellen, der die Transformation der Daten und der entfernten Systemaufrufe realisiert. Diese Adapter basieren in der Regel auf proprietären Klassenbibliotheken.

Neben der reinen Transformation der Daten und Systemaufrufe stellen die meisten EAI-Werkzeuge Funktionalität zur Lastverteilung, Ausfallsicherheit und zum Monitoring der EAI-Aktivitäten zur Verfügung. EAI-Werkzeuge eignen sich in erster Linie für vollautomatisierte Arbeitsabläufe, die auf strukturierten Daten arbeiten. Dabei wird der Bearbeitungsstand (Status) der einzelnen Arbeitsabläufe in der Regel vom EAI-Werkzeug registriert und dann über das EAI-Werkzeug überwacht.

Mit der wachsenden Popularität der Web Services haben auch die EAI-Anbieter das Potential dieser Technologie erkannt. So bieten heute viele EAI-Werkzeuganbieter die Integration nicht mehr nur mittels proprietärer Adapter an, sondern ermöglichen es den Integratoren, Anwendungen über Web Services an das EAI-Werkzeug anzubinden.

Workflow Management Systeme

Ebenso wie klassische EAI-Systeme dienen auch WfMS der Automatisierung von Arbeitsabläufen. Der wesentliche Unterschied zwischen diesen beiden Systemklassen liegt jedoch in der Art der Anwendungen, die automatisiert werden sollen. Während bei EAI-Systemen vollautomatisierte Prozesse im Vordergrund stehen, konzentrieren sich WfMS auf Arbeitabläufe, die Benutzerinteraktionen erfordern. Der Bedarf an Benutzerinteraktionen ist auch dadurch begründet, dass WfMS mit unstrukturierter Daten arbeiten. Um diese Daten geeignet interpretieren und verarbeiten zu können, ist meist der Eingriff eines Menschen erforderlich.

Middleware-Technologien

Middleware-Technologien für sich genommen sind nicht ausreichend für die Integration existierender Anwendungen. Sie bieten jedoch eine Infrastruktur, mittels derer Anwendungen miteinander kommunizieren, Daten ausgetauscht und entfernte Methoden einer anderen Anwendung aufgerufen werden können. Damit eignen sich Middleware-Technologien für eine Integration auf Daten- und/oder Anwendungsebene. Die Prozessintegration wird durch die Middleware-Technologien nur insoweit unterstützt, dass entfernte Systemaufrufe erfolgen können. Wann, wie und in welcher Reihenfolge diese Aufrufe erfolgen, wird von den beteiligten Anwendungen bestimmt. Die Middleware selbst hat von Abhängigkeiten zwischen den einzelnen Systemaufrufen und damit vom Status der Arbeitsabläufe keinerlei Kenntnis.

Web Services

Die Web Services Standards bieten standardisierte Protokolle (SOAP) und Schnittstellenbeschreibungen (WSDL) für das Auffinden (UDDI) und die Nutzung entfernter Dienste. So kann ein Anbieter Anwendungsfunktionalität als Dienst über einen Web Service zur Verfügung stellen, der von beliebigen Nutzern (Personen oder Softwaresystemen) genutzt werden kann. Web Services eignen sich damit für eine lose Kopplung von Softwaresystemen.

Die Web-Services-Technologie erlaubt es nicht, verschiedene aufeinanderfolgende Dienstaufrufe zueinander in Beziehung zu setzen (ausser die beteiligten Anwendungen regeln dies), oder eine Abfolge von verschiedenen Dienstaufrufen festzulegen. Sie befasst sich immer nur mit genau einer Nutzung eines Dienstes.

Web Services in Kombination mit Business Process Modeling Languages

Zur Spezifikation von Geschäftsprozessen sind in den letzten Jahren eine Reihe von Standards wie BPEL4WS, BMPL, XPDL, UML entstanden, wobei BPEL4WS, BMPL und XPDL XML-basierte Sprachen sind [BPML03, BPEL03]. BPEL4WS (Business Process Execution Language for Web Services) ist speziell auf die Koordination von Web Services ausgerichtet. Gesteuert und überwacht wird die Ausführung eines in BPEL4WS spezifizierten Geschäftsprozesses von einer sogenannten „BPEL Engine“. Derzeit ist die Entwicklung der Web Services noch nicht ausgereift. So gibt es noch einige offene Fragen bzgl. der Sicherheit und Transaktionen im Zusammenhang mit Web Services. Diese Probleme wurden jedoch erkannt, und es sind Standards im Entstehen, die sich mit den genannten Fragestellungen befassen. Für die Zukunft bietet die Kombination aus BPEL4WS und Web Services damit eine echte Alternative zu den klassischen EAI-Werkzeugen. Andererseits ist es auch denkbar, dass EAI-Werkzeuge um eine BPEL Engine erweitert werden.

Prinzipiell lassen sich mit Web Services und Business Process Modeling Languages Integrationsszenarien jeglicher Art realisieren, da sie alle oben genannten Integrationsmerkmale unterstützen.

5 Zusammenfassung und Ausblick

Dass Integration heute ein wichtiges Thema in fast allen Firmen ist, zeigen aktuelle Statistiken (vgl. [Born03]). Als Reaktion auf diesen hohen Bedarf nach Integrationslösungen haben Hersteller aus verschiedenen Bereichen ihre Werkzeuge um (standardisierte) Integrationsschnittstellen erweitert bzw. eigenständige Integrationswerkzeuge auf den Markt gebracht. Neben diesen reinen Integrationswerkzeugen haben sich in den letzten Jahren Basistechnologien für die Integration (z.B. CORBA, Web Services) etabliert. Die damit entstandene Vielfalt an Technologien und Werkzeugen stellt mit der Integration beauftragte Personen vor eine schwierige Aufgabe.

Um das geeignete Integrationswerkzeug bzw. die geeignete Integrationstechnologie auswählen zu können, muss das Integrationsszenario genauer untersucht warden, und es sollte eine Vision entwickelt werden, welche weiteren Integrationen in den nächsten Monaten und Jahren angestrebt werden. Auf der Basis dieser Vision kann dann die derzeit für diese Vision geeignete Integrationslösung ausgewählt werden.

Insgesamt lässt sich feststellen, dass die verschiedenen Integrationstechnologien immer weiter zusammen wachsen. So unterstützen heute bereits eine Reihe von EAI-Werkzeugen Web Services, und Workflow Management Systeme setzen auf standardisierte Business Process Modeling Languages. Solange der Prozess des Zusammenwachsens jedoch noch nicht abgeschlossen ist, steht der Integrationsverantwortliche vor der Qual der Wahl.

Literatur:

[Bern96] Bernstein, Philip A. "Middleware: A Model for Distributed System Services," Communications of the ACM, February 1996, Vol. 39, No.2, 86-98.

[BFGH02] Bunjes, B., Friebe, J., Götze, R. und Harren, A., Integration von Daten, Anwendungen und Prozessen am Beispiel des Telekommunikationsunternehmens EWE TEL, Wirtschaftsinformatik, 44, 2002, 5, 415-423

[Born03] Born, A. und Diercks, J.: „Anwendungsintegration: grenzenlose Zusammenarbeit“, iX 11 (November 2003): 87-98

[OMG02] Object Management Group: „Common Object Request Broker Architecture (CORBA/IIOP)”, http://www.omg.org/technology/documents/corba_spec_catalog.htm

[BPML03] Business Process Modeling Language, http://www.bpmi.org

[BPEL03] BPEL4WS, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[Sun02] Sun Microsystems, Inc: „The Java Platform: Five Years in Review.” http://java.sun.com/features/2000/06/time-line.html

[MSOFT03] Microsoft, Inc: „Enterprise Solution Patterns Using Microsoft .NET“, http://msdn.microsoft.com/practices/type/Patterns/Enterprise

[Mon02] Monson-Haefel, Richard: „Enterprise Java Beans“, O’Reilly, 2001

[Nus00] R. Nussdorfer, Das EAI-Buch, E-Business und EAI, Integration von Anwendungen, Trends, Technologie und Lösungen, CSA Consulting, 2000

Funktionsgetriebene Integration von Legacy-Systemen mit Web Services

T. Teschke[1], H. Jaekel[1],
S. Krieghoff[2], M. Langnickel[2],
W. Hasselbring[3] und R. Reussner[3]

[1] OFFIS, Bereich „Betriebliches Informations- und Wissensmanagement“
{teschke|jaekel}@offis.de
[2] Kommunale Datenverarbeitung Oldenburg (KDO)
{krieghoff|langnickel}@kdo.de
[3] Universität Oldenburg, Department für Informatik,
Abteilung Software Engineering
{hasselbring|reussner}@informatik.uni-oldenburg.de

Zusammenfassung Dieser Beitrag beschreibt die Nutzung von Web Services bei der Migration eines Legacy-Systems. Es wird eine Architektur vorgestellt, auf deren Grundlage die *Kommunale Datenverarbeitung Oldenburg (KDO)* ihre umfangreichen Legacy-Anwendungssysteme in eine moderne Softwarearchitektur integriert. Grundlage dieser Architektur ist das Dublo-Architekturmuster, das die sanfte Integration und Migration von Legacy-Anwendungssystemen unterstützt. Das betrachtete kommunale Anwendungssystem wird über Web Services in eine J2EE-Architektur integriert, wobei die zustandsabhängige Verfügbarkeit einzelner Funktionen anhand von Protokollautomaten kontrolliert wird.

1 Einleitung

Die *Kommunale Datenverarbeitung Oldenburg (KDO)* ist ein Softwarehaus und IT-Dienstleister für die kommunale Verwaltung. Angesichts des enormen Kostendrucks im kommunalen Bereich und wachsender bzw. sich änderndender Aufgaben der Verwaltungen müssen die von der KDO entwickelten kommunalen Informationssysteme flexibel und kostengünstig an neue, kundenspezifische Anforderungen anpassbar und somit langfristig nutzbar sein.

Bislang sind die Client/Server-Informationssysteme der KDO auf der Grundlage von Informix-Datenbanksystemen und der 4Js-Implementierung [Fou03] der Informix 4GL [IBM03] entwickelt worden, wobei Präsentations- und Anwendungscode in den 4GL-Programmen eng verzahnt sind. 4GL ist eine so genannte „fourth generation language“ [WW90]. Maßnahmen zur Sicherung der Konsistenz der in den Datenbankrelationen abgelegten Daten sind nicht auf Datenbankebene, sondern durch die Fachverfahren realisiert. Dies ist durch die Notwendigkeit begründet, den Betrieb auch mit alten Datenbankversionen sicherzustellen, da Kommunen aus Kostengründen nicht regelmäßig in neue Datenbanktechnologie investieren bzw. Vollwartung abschließen können.

Die KDO hat sich dazu entschlossen, diese (monolithische) 2-Schichten-Architektur mittelfristig durch eine flexiblere Mehrschichtenarchitektur zur ersetzen, die auf aktuellen Standards aufsetzt und moderne Techniken des Software Engineering berücksichtigt. Da die aktuelle Implementierung der Fachverfahren umfangreiches Domänenwissen beinhaltet und daher nicht einfach ersetzt werden kann, sollen diese Altverfahren zunächst über Web Services in die neue Architektur eingebunden und später schrittweise abgelöst werden. Das Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und -Systeme (OFFIS), ein An-Institut der Carl von Ossietzky Universität Oldenburg, begleitet die KDO bei dieser Umstellung ihrer Softwarearchitektur. Die in diesem Beitrag vorgestellte Arbeit ist das Ergebnis einer Kooperation des OFFIS-Bereichs „Betriebliches Informations- und Wissensmanagement“, der Abteilung „Software Engineering“ des Departments für Informatik und der KDO.

Dieser Beitrag ist wie folgt aufgebaut: Ausgehend von einer kurzen Vorstellung des Dublo-Architekturmusters in Abschnitt 2 skizzieren wir in Abschnitt 3 drei grundsätzliche Alternativen zur Identifikation von Web Services in Legacy-Systemen. In Abschnitt 4 beschreiben wir den bei der KDO verfolgten Ansatz zur Identifikation von Web Services in den Altverfahren und deren Integration in die neue Architektur, bevor wir den Beitrag in Abschnitt 5 mit einer Zusammenfassung und einem Ausblick beschließen.

2 Sanfte Legacy-Integration und -Migration nach dem Dublo-Muster

Legacy-Anwendungssysteme unterscheiden sich von modernen Softwarearchitekturen oft darin, dass sie monolithisch aufgebaut sind und nicht zwischen unterschiedlichen Schichten für Präsentation, Anwendungslogik und Datenhaltung unterscheiden. Stand der Softwaretechnik sind heute jedoch mehrschichtige Architekturen mit klarer Trennung von Präsentation, Anwendungslogik und Datenhaltung, mit denen u. a. vereinfachte Austauschbarkeit und Anpassbarkeit von Komponenten, verbesserte Nachvollziehbarkeit fachlicher Anforderungen in der Schicht der Geschäftslogik oder erhöhte Transparenz von Datenbankaspekten angestrebt werden.

Eine sanfte Migration von monolithischen Legacy-Systemen zu modernen, mehrschichtigen Softwarearchitekturen gestattet das *Dublo-Muster* [HRJ+04]. Dublo steht für „Dual Business Logic“ und basiert im Kern auf dem Gedanken, Geschäftslogik an zwei Stellen zu halten: im Legacy-System sowie in neu eingeführten Mittelschichten. Dabei wird (neue) Geschäftslogik in einer neuen Geschäftslogikschicht implementiert und mittels eines Legacy-Adapters [GHJV96] mit der alten Geschäftslogik verbunden. Neben dem Zugriff auf alte Geschäftslogik wird dieser Adapter insbesondere auch für den Datenbankzugriff genutzt, d. h. Zugriffe der neu implementierten Geschäftslogik auf die Datenbank erfolgen ausschließlich über den existierenden Legacy-Code. Dieser Zugriff auf die Datenbank durch bestehenden Legacy-Code führt zwar zu einer Verteilung von Geschäftslogik auf zwei Schichten, hat aber den Vorteil, dass alte Geschäftslogik

zunächst weiter verwendet werden kann (um schließlich schrittweise ersetzt zu werden). Insbesondere kann so aufwändige Geschäftslogik des Legacy-Systems, die (wie bei 4GL Systemen häufig) intrinsisch mit dem Datenbenkzugriff verwoben ist, weiter genutzt werden und mehr oder weniger unverändert durch die mittleren Schichten zu einer neuen Präsentationsschicht weitergereicht werden. Im Laufe der Migration wird dann immer mehr Geschäftslogik portiert, bis der Legacy-Adapter schließlich vollständig durch einen Datenbank-Adapter ersetzt werden kann. Abbildung 1 zeigt die Struktur des Dublo-Architekturmusters.

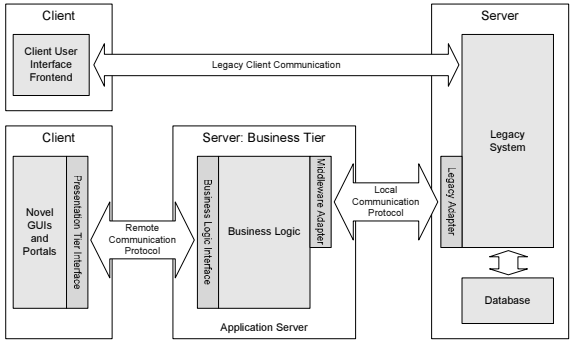

Abbildung 1. Strukturelle Sicht auf das Dublo-Architekturmuster

Im Kontext der KDO wird das Dublo-Architekturmuster eingesetzt, um einen J2EE-Server [Sha01] an das bestehende Informix-4GL-System anzubinden. Dabei werden die Legacy-Adapter über Web Services realisiert. Von der Nutzung einer standardisierten, plattformunabhängigen Technologie wie Web Services verspricht man sich insbesondere Flexibilität bei Technologiewechseln in der Datenbank- und Geschäftslogikschicht [PG03]. Alternativen zur Identifikation geeigneter Web Services werden im folgenden Abschnitt diskutiert. Die KDO wird die durch das Dublo-Muster eingeführte neue Präsentationsschicht ergänzend zum Zugriff auf neue Geschäftslogik auch für den Zugriff auf alte Geschäftslogik entwickeln, um die alte Präsentationsschicht abzulösen.

Die Verbindung zwischen Präsentationsschicht und Geschäftslogikschicht muss aus Gründen der Datensicherheit gesondert gesichert werden. Dazu kann in einer Java-Umgebung beispielsweise RMI und SSL eingesetzt werden. Im Kontext der KDO wurde diese Verbindung mittels OSCI (Online Services Computer Interface) [OSC] gemäß gesetzlichen Vorgaben gesichert. Durch Sicherung der Verbindung zwischen Präsentations- und Geschäftslogikschicht laufen Server, die diese Schichten realisieren, in einer gesicherten Umgebung, d. h. insbesondere kann die Verbindung zwischen Geschäftslogik- und Datenbankschicht als gesichert angesehen werden.

3 Alternativen zur Identifikation von Web Services

Bei der Identifikation von Web Services (oder allgemeiner: Klassen) in Legacy-Systemen lassen sich nach [WBF97] drei alternative Ansätze unterscheiden:

1. *Datengetrieben*: Identifikation von Web Services zur Repräsentation des Objektmodells auf der Grundlage der im Legacy-System implementierten Datenstrukturen.
2. *Funktionsgetrieben*: Identifikation von Web Services zur Bündelung der für die Durchführung zentraler Anwendungsfälle relevanten Dienste auf der Grundlage der im Legacy-System realisierten Funktionalität.
3. *Objektgetrieben*: Erstellung eines Objektmodells der Anwendungsdomäne und Analyse des Legacy-Systems, um Code-Elemente zu identifizieren, die die Datenstrukturen und Funktionalität entsprechender Klassen implementieren.

Im Kontext der KDO erschien die objektgetriebene Identifikation von Web Services, die z. B. auf einem durch Entity Beans definierten Ziel-Objektmodell aufsetzen könnte, grundsätzlich nicht gangbar. Da die den Altverfahren zugrunde liegenden Relationen im Allgemeinen nicht normalisiert sind und sich daraus erhöhte Differenzen zwischen Ist- und (normalisiertem) Ziel-Objektmodell ableiten ließen, waren bei diesem Ansatz Schwierigkeiten bei der Zuordnung von Funktionen der Fachverfahren zu den Entity Beans des Ziel-Objektmodells zu erwarten.

Ähnlich wie der objektgetriebene Ansatz zielt auch die datengetriebene Identifikation von Web Services auf die Implementierung eines Objektmodells durch Legacy-Funktionen ab. Der wesentliche Unterschied zwischen beiden Ansätzen ist darin zu sehen, dass der objektgetriebene Ansatz von einem Ziel-Objektmodell ausgeht, während der datengetriebene Ansatz auf dem aktuell implementierten Objektmodell aufbaut. Beim datengetriebenen Ansatz werden geeignete Web Services im Sinne abstrakter Datentypen (ADTs) ausgehend von den im Legacy-System implementierten Datenstrukturen identifiziert. Dies können zum einen (komplexe) Datentypen, die im Quellcode der Anwendung genutzt werden, und zum anderen (relationale) Datenbankschemata sein. In der Forschung sind verschiedenartige Ansätze auf Basis von Techniken wie Clustering und Konzeptanalyse (vgl. z. B. [vDK99]) oder Datenfluss-, Kontrollfluss- und Syntaxanalyse (vgl. z. B. [KP99]) untersucht worden.

Der funktionsgetriebene Ansatz zielt im Kern auf die „Entdeckung“ von Anwendungsfällen in Legacy-Systemen ab, für die durch geeignet zusammengestellte Web Services eine Schnittstelle definiert werden soll. Dabei kann sich diese Ermittlung von Anwendungsfällen an den einzelnen Legacy-Programmen, die

die betriebliche Funktionalität des Legacy-Systems implementieren, oder an den Bildschirmmasken des Systems orientieren. Erstgenannter Ansatz wird z. B. von Erlikh [Erl02] verfolgt, der zunächst Web Services durch iterative Verfeinerung von Mengen funktional zusammengehöriger Programme identifiziert. Die APIs dieser Web Services werden anschließend durch Einsatz verschiedener Techniken des *Knowledge Mining* ermittelt. Bei dem maskenorientierten Ansatz, den Stroulia et al. in [SERS02] vorgestellen, werden durch die Analyse der Benutzerinteraktionen mit dem Legacy-System Anwendungsfälle extrahiert, für die eine neue Schnittstelle (in unserem Fall in Form von Web Services) anzubieten ist. Durch die Bestimmung der für die Durchführung dieser Anwendungsfälle genutzten Funktionen lassen sich die APIs entsprechender Web Services definieren.

4 Identifikation und Nutzung von Web Services bei der KDO

Da sich die Struktur der Implementierung der einzelnen kommunalen Fachverfahren und ihrer Funktionen primär an den unterstützten Anwendungsfällen und Geschäftsprozessen und weniger an den zugrunde liegenden Datenbankstrukturen orientiert, hat sich die KDO für einen funktionsgetriebenen Ansatz zur Identifikation und Nutzung von Web Services entschieden. Dabei erschien das von Erlikh [Erl02] vorgeschlagene Verfahren aufgrund des Einsatzes von Techniken des Knowledge Mining zu komplex. Wesentliche Gedanken des von Stroulia et al. [SERS02] untersuchten Ansatzes ließen sich dagegen auf die KDO übertragen, da in den Fachverfahren eine enge Kopplung zwischen den Bildschirmmasken und den Funktionen besteht. Sein Einsatz setzt allerdings voraus, dass die Struktur der aktuellen Masken in der J2EE-Implementierung im Wesentlichen beibehalten bleibt.

Die Identifikation der Web Services, die zur Nutzung der Altverfahren zu exportieren sind, orientiert sich an den bestehenden Fachverfahren und deren Bildschirmmasken und wird manuell durchgeführt. Für jede Maske werden dabei die Benutzerinteraktionen ermittelt, mit denen datenverarbeitende Vorgänge ausgelöst werden, die jeweils auszuführenden Funktionen analysiert und (für jede dieser Interaktionen) entsprechende Operationen eines Web Services definiert. Ein Web Service kann dabei die Operationen einer Maske oder einer funktional zusammenhängenden Sequenz von Masken umfassen. Die Parameter der Operationen eines Web Service werden aus den Feldern der jeweiligen Eingabemaske abgeleitet. Die für einen Web Service zulässigen Interaktionsfolgen (Sequenzen von Operationsaufrufen) werden durch Analyse der Abhängigkeiten zwischen Bildschirmmasken bestimmt und in Form eines endlichen (Protokoll-) Automaten spezifiziert, um die korrekte Nutzung der Dienste der Altverfahren sicherzustellen. Sind Bildschirmmasken nicht vorhanden, oder sind ihre Abhängigkeiten nicht (implizit oder explizit) spezifiziert, so können auch Verfahren zur Analyse von Geschäftsprozessen („Process Mining“) genutzt werden, um die Menge zulässiger Interaktionsfolgen zu bestimmen [Sch02]. Die explizite Angabe von zulässigen Sequenzen ermöglicht die Erkennung von unzulässigen Aufrufsequenzen zur Laufzeit. Der Vorteil dieser Erkennung liegt in der exakten Identifikation des ersten unzulässigen Aufrufs. Würde man diese Überprüfung unzulässiger Aufrufreihenfolgen nicht durchführen, würde das Fehlverhalten, welches durch eine unzulässige Aufrufreihenfolge verursacht wird, entweder gar nicht oder aber möglicherweise so spät entdeckt, dass eine Identifikation des unzulässigen Aufrufs nur schwer oder gar nicht möglich ist. Beschreibt man nicht nur die zulässigen Aufrufsequenzen des Adapters, sondern zusätzlich auch die von der Geschäftslogikschicht benötigten Dienste, ist eine statische Interoperabilitsprüfung der Aufrufsequenzen (Protokolle) möglich. Wird diese Prüfung schon zur Entwicklungszeit durchgeführt, können Überprüfungen der Aufrufsequenzen zur Laufzeit wegfallen, da eine statische Prüfung alle inkompatiblen Interaktionsfolgen ausschließt. Da der Aufwand einer Modellierung der von der Geschäftslogik benötigten Aufrufsequenzen als zu hoch eingeschätzt wurde, verzichtete man im beschriebenen Projekt auf statische Überprüfungen.

Ein Ansatz, der mit unserer Automatennutzung vergleichbar ist, wird bei verschiedenen Beschreibungssprachen für Web Services verfolgt. So lässt sich im Web Service Choreography Interface (WSCI) [W3C02] das beobachtbare Verhalten eines Dienstes als temporale und logische Abhängigkeiten zwischen den ausgetauschten Nachrichten beschreiben. Die Business Process Execution Language for Web Services (BPEL4WS) [BEA03] definiert eine Notation, die die temporalen und logischen Bedingungen bei der Zusammensetzung von Geschäftsprozessen aus mehreren Web Services beschreibt. Formal lassen sich Sequenzen von Web Services mit den gängigen Notationen zur Protokollspezifikation, also z.B. endlichen Automaten oder Petri-Netzen, beschreiben [BRSM03, MPP02]. Da noch nicht absehbar ist, welcher der konkurrierenden Standards WSCI oder BPEL4WS sich zukünftig durchsetzen wird, werden bei der KDO endliche Automaten zur Beschreibung der zulässigen Interaktionsfolgen eingesetzt.

Eine Voraussetzung für den Zugang zur implementierten Funktionalität über Web Services ist die Trennung der GUI-Aspekte von dem eigentlichen Anwendungscode durch Auslagerung des jeweiligen Codes in eigenständige 4GL-Funktionen. Dabei werden Interaktionen, die bislang aufgrund der Verzahnung von GUI- und Anwendungsaspekten innerhalb einer Funktion abgefragt und verarbeitet wurden, durch die Aufspaltung der 4GL-Funktionen an den Interaktionspunkten berücksichtigt. Die Web-Service-Schnittstelle kann dann auf technischer Ebene als Adapter für die resultierenden 4GL-Anwendungsfunktionen bereitgestellt werden.

Der Zugriff auf die Altverfahren der KDO über Web Services kann gemäß dem in Abbildung 2 dargestellten Sequenzdiagramm erfolgen. Dabei werden die folgenden Schritte ausgeführt:

① Aufruf eines Dienstes durch einen (Präsentations-)Client: Ein Client ruft eine Methode einer zustandsbehafteten Session Bean [DYK01] auf, die durch einen J2EE-Server bereitgestellt wird.

② Initialisierung der Session Bean: Sofern die Session Bean noch nicht initialisiert wurde, meldet diese sich zunächst bei einem „Fassaden-Server“ mit einer Kundenkennung sowie dem gewünschten Verfahren und der auszuführenden

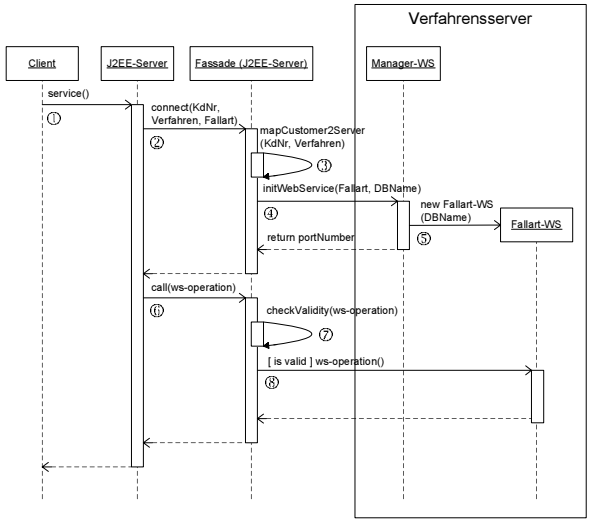

Abbildung 2. Integration von Altverfahren über Web Services

Fallart an (der Fassaden-Server bietet in Anlehnung an das gleichnamige Entwurfsmuster [GHJV96] eine abstrakte Schnittstelle zu den Altverfahren an). Ist die Session Bean bereits initialisiert, wird dieser Schritt übersprungen und direkt mit Schritt 6 fortgefahren.

③ Auswahl des Verfahrensservers: Der Fassaden-Server, dessen Dienste ebenfalls über (zustandsbehaftete) Session Beans implementiert sein können, wählt anhand der Kundenkennung und des Verfahrens einen zuständigen Verfahrensserver aus. Dabei erfolgt auch die Bestimmung der durch das Verfahren zu benutzenden Datenbank.

④ Beantragung einer Web-Service-Instanz: Der Fassaden-Server beantragt bei einem „Manager-Web-Service“, der auf dem zuvor ermittelten Verfahrensserver läuft und die Erzeugung von Web Services kontrolliert, eine Instanz des für die gewünschte Fallart zuständigen Web Services und übergibt dabei den Namen der zu benutzenden Datenbank.

⑤ Erzeugung der Web-Service-Instanz: Der Manager-Web-Service legt eine Instanz des gewünschten Web Services als neuen Prozess an (da die 4Js-Programme der Altverfahren globale Variablen nutzen, erfolgt hier eine Trennung zwischen unterschiedlichen Clients durch die Erzeugung unabhängiger Prozesse). Er gibt die Nummer des Ports zurück, über den der Prozess identifiziert und angesprochen werden kann. Mit der anschließenden Speicherung dieser Port-Nummer auf dem Fassaden-Server wird die Initialisierung der Session Bean (Schritt ②) abgeschlossen.

⑥ Anforderung des Dienstes: Die zustandsbehaftete Session Bean fordert die dem clientseitig angeforderten Dienst entsprechende Operation des Web Services beim Fassaden-Server an.

⑦ Prüfung der Gültigkeit des Aufrufs: Der Fassaden-Server prüft die Zulässigkeit des angeforderten Aufrufs anhand eines Protokollautomaten, der die zulässigen Interaktionsfolgen einer Web-Service-Session beschreibt.

⑧ Ausführung der Operation: Ist der angeforderte Aufruf zulässig, wird die Operation des Web Services ausgeführt und Rückgabewerte an den Client zurückgegeben, anderenfalls wird ein Fehler signalisiert.

Aus Platzgründen verzichten wir hier auf eine detaillierte Betrachtung des Abbaus von Verbindungen zwischen J2EE-Server und Web-Service-Instanz. Inwieweit die zusätzliche Serverlast, die durch die Erzeugung einer möglicherweise hohen Anzahl von Web-Service-Prozessen entsteht, kritisch ist, muss sich im praktischem Einsatz noch zeigen.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde ein Ansatz zur funktionsgetriebenen Integration von Legacy-Systemen mit Web Services im Kontext kommunaler Informationssysteme vorgestellt, den wir als Weiterentwicklung bestehender Diskussionen auf dem Gebiet der Anwendungsintegration einordnen.

Im Anschluss an eine kurze Einleitung, in der die Ausgangssituation der KDO skizziert wurde, folgte eine knappe Einführung in die sanfte Integration und Migration von Legacy-Anwendungssystemen nach dem Dublo-Architekturmuster. Eine Analyse grundsätzlicher Alternativen zur Identifikation von Web Services in Legacy-Systemen bildete die Grundlage für den nachfolgend vorgestellten Ansatz zur Identifikation und Nutzung von Web Services, über den die umfangreichen kommunalen Fachverfahren in die moderne Mehrschichtenarchitektur der KDO eingebunden werden. Um die Zulässigkeit der Reihenfolge von Web-Service-Aufrufen sicherzustellen, kommen in der vorgestellten Architektur Spezifikationen endlicher Automaten zum Einsatz, auf deren Grundlage zulässige Interaktionsfolgen mit dem Legacy-System beschrieben werden.

Da die Verwendung des Dublo-Musters aufgrund der auf zwei Schichten verteilten, z. T. duplizierten Geschäftslogik einen gewissen Mehraufwand für die Pflege des Anwendungssystems impliziert, beschränkt sich die Anwendbarkeit dieses Musters auf „stabile“ Legacy-Systeme, die keine großen funktionalen oder strukturellen Änderungen mehr erfahren. Als Schwachpunkt des in diesem Beitrag vorgestellten Ansatzes muss die Beibehaltung des alten, oftmals im Laufe

der Jahre degenerierten Datenbankschemas in neuen Fachanwendungen betrachtet werden. Künftige Arbeiten betreffen daher die Konzipierung eines Migrationspfades zur vollständigen Ablösung von Legacy-Systemen und die dabei durchzuführende Evolution des Datenbankschemas. Während des parallelen Betriebs alter und neuer Geschäftlogik ist zu erwarten, dass die semantische Heterogenität zwischen alter und neuer Geschäftslogik noch weitere Herausforderungen birgt.

Literatur

[BEA03] BEA, IBM, Microsoft, SAP AG und Siebel Systems, http://www.ibm.com/developerworks/library/ws-bpel/: *Business Process Execution Language for Web Services Version 1.1*, 2003. Besucht am 18.12.2003.

[BRSM03] Berardi, D., F. De Rosa, L. De Santis und M. Mecella: *Finite State Automata as Conceptual Model for E-Services*. In: *Integrated Design & and Process Technology*. Society for Process & Design Sciences, 2003.

[DYK01] DeMichiel, L. G., L. Ü. Yalçinalp und S. Krishnan: *Enterprise JavaBeans Specification, Version 2.0*. Sun Microsystems, 2001.

[Erl02] Erlikh, L.: *Integrating Legacy Systems Using Web Services*. eAI Journal, 2002.

[Fou03] Four J's, http://www.4js.com: *Four J's Development Tools*, 2003. Besucht am 25.08.2003.

[GHJV96] Gamma, E., R. Helm, R. Johnson und J. Vlissides: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 1. Auflage, 1996.

[HRJ+04] Hasselbring, W., R. Reussner, H. Jaekel, J. Schlegelmilch, T. Teschke und S. Krieghoff: *The Dublo Architecture Pattern for Smooth Migration of Business Information Systems*. In: *26th International Conference on Software Engineering (ICSE) 2004*, Edinburgh, Schottland, Großbritannien, Mai 2004.

[IBM03] IBM, http://www-3.ibm.com/software/data/informix/tools/4gl/: *Informix 4GL product family*, 2003. Besucht am 25.08.2003.

[KP99] Kontogiannis, K. und P. Patil: *Evidence Driven Object Identification in Procedural Code*. In: *Proceedings of IEEE Conference on Software Technology and Engineering Practice (STEP '99)*, Seiten 12–21, Pittsburgh, Pennsylvania, USA, August 1999.

[MPP02] Mecella, Massimo, Francesco Parisi Presicce und Barbara Pernici: *Modeling E-service Orchestration through Petri Nets*. Lecture Notes in Computer Science, 2444:38ff., 2002.

[OSC] OSCI, http://www.osci.de: *Online Services Computer Interface*. Besucht am 18.12.2003.

[PG03] Papzoglou, M.P. und D. Georgakopoulo: *Special Section on Servuce Oriented Computing*. Communications of the ACM, 46(10):24–61, Oktober 2003.

[Sch02] Schimm, Guido: *Process Miner — A Tool for Mining Process Schemes from Event-Based Data*. Lecture Notes in Computer Science, 2424:525ff., 2002.

[SERS02] Stroulia, E., M. El-Ramly und P. Sorenson: *From Legacy to Web through Interaction Modeling*. In: *Proceedings of the International Conference on Software Maintenance (ICSM '02)*, Seiten 320–329, Montreal, Kanada, Oktober 2002. IEEE Press.

[Sha01] Shannon, B.: *Java 2 Platform Enterprise Edition Specification, v1.3*. Sun Microsystems, 2001.

[vDK99] Deursen, A. van und T. Kuipers: *Identifying Objects Using Cluster and Concept Analysis*. In: *Proceedings of the 1999 International Conference on Software Engineering (ICSE '99)*, Seiten 246–255, Los Angeles, USA, Mai 1999. ACM.

[W3C02] W3C, http://www.w3.org/TR/wsci/: *Web Service Choreography Interface (WSCI) 1.0*, 2002. Besucht am 18.12.2003.

[WBF97] Wiggerts, T., H. Bosma und E. Fielt: *Scenarios for the Identification of Objects in Legacy Systems*. In: Baxter, I. D., A. Quilici und C. Verhoef (Herausgeber): *Proceedings of the 4th Working Conference on Reverse Engineering (WCRE '97)*, Seiten 24–32, Amsterdam, Niederlande, Oktober 1997. IEEE Computer Society Press.

[WW90] Wojtkowski, W.G. und W. Wojtkowski: *Applications Software Programming With Fourth-Generation Languages*. Wadsworth Publishing, 1990.