Semantic Integration of Identity Data Repositories

Christian Emig[1], Kim Langer[1], Jürgen Biermann[2], Sebastian Abeck[1]

[1] Cooperation & Management, Universität Karlsruhe (TH), 76128 Karlsruhe
[2] iC Consult GmbH, Keltenring 14, 82041 Oberhaching

{ emig | langer | abeck } @cm-tm.uka.de, biermann@ic-consult.de

Abstract. With the continuously growing number of distributed and heterogeneous IT systems there is the need for structured and efficient identity management (IdM) processes. This implies that new users are created once and then the information is distributed to all applicable software systems same as if changes on existing user objects occur. The central issue is that there is no generally accepted standard for handling this information distribution because each system has its own internal representation of this data. Our approach is to give a semantic definition of the digital user objects' attributes to ease the mapping process of an abstract user object to the concrete instantiation of each software system. Therefore we created an ontology to define the mapping of users' attributes and an architecture which enables the semantic integration of identity data repositories. Our solution has been tested and tried in an implementation case study.

1 Introduction

The desire of enterprises to automate their business processes and to integrate existing IT solutions to enhance business performance spreads, among others, to the field of identity management (IdM). IdM can be defined as a set of processes and a supporting infrastructure for the creation, maintenance and use of digital identities (human users or IT systems) to enable efficient authentication, authorization and access control [1]. Processes in IdM include user provisioning, decommissioning and auditing [2]. To increase the automation of these processes, there is the need to provide an integrated view on the data which is being administered (cf. Figure 1), especially the user-specific data (i.e. the digital identities). This data is stored either stand-alone or can be directly attached to the business applications where it is employed to enforce access control. It is held in directories though other data storage solutions such as relational databases or XML-based files are conceivable as well. The repository may be distributed over different systems, and in case of a directory-based approach, the information in the repository is quite often accessible via the Lightweight Directory Access Protocol (LDAP). A digital identity is the representation of a subject that includes an identifier (e.g. a unique number), credentials and attributes. To enable efficient identity management it is important to keep the different identity repositories synchronized which implies an integration effort. The integration of heterogeneous data from different data repositories raises several questions that have not been fully answered yet.

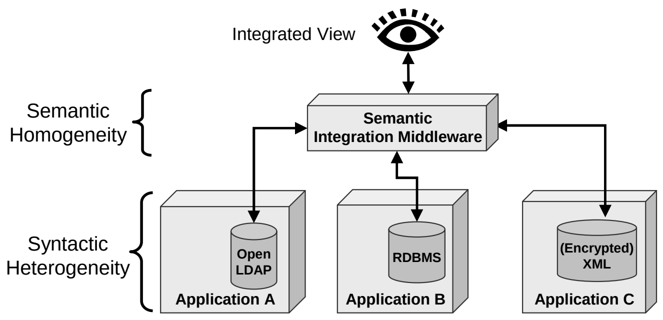

Figure 1. Semantic Integration of Heterogeneous Data

The integration of data can be performed on different tiers of data representation. The common integration approach is done on a syntactic level, merely pushing bytes from one integration end to the other with the meaning of data and so the mapping rules being hard-coded into the synchronization middleware. This approach though is rather shortsighted since the semantics of data are not being focused and changes in semantics entail direct changes within the synchronization middleware. Thus a more mature approach is needed which takes into account the semantics of the data objects processed. The term semantics in such a context is strongly correlated with ontologies. They play a key role in sharing a collective understanding of the semantics of the domain being described. Therefore, ontologies have to be considered when there is the need to elevate data repository integration to a semantic level and to provide an integrated view on data.

The two contributions of this paper are:

(1) The definition of an extensible **ontology** to enable semantic integration of syntactically heterogeneous identity data repositories (cf. chapter 3, "Person Ontology"). When addressing heterogeneity in this context it is important to point out that this heterogeneity is encountered on a syntactic and structural tier whereas the semantic tier appears quite homogenous, since all information on the user object in the domain of identity management is quite similar.

(2) An **architecture** supporting the semantic integration of identity data repositories (cf. chapter 4, "Data Repository Container"). Our approach wraps the application-specific user directories to a *data repository container* (DRC) by adding additional components to the integration architecture.

The paper is organized as follows: Chapter 2 treats the related work about and the state-of-the-art of semantics in digital identities and how ontologies can ease the information integration in this context. Furthermore, architectural approaches are introduced that try to enhance directories to actively propagate local data changes to other directories. Chapter 3 and 4 describe our conceptual contributions which are applied in a case study illustrated in chapter 5. A conclusion and an outlook on future work in this area close the body of the paper.

2 Related Work and State-of-the-Art

2.1 Semantic Aspects

When dealing with the integration of different data sources it is for certain that data heterogeneity between the various data sources will be encountered. Problems referring to heterogeneity of data are already widely known within the distributed database systems community. In [3] it is distinguished between structural and syntactic heterogeneity (i.e. schematic heterogeneity) on the one hand and semantic heterogeneity (i.e. data heterogeneity) on the other hand. Structural heterogeneity means that different information systems store their data in different structures (e.g. schema). Semantic heterogeneity considers the contents of an information item and its intended meaning. In order to achieve semantic interoperability in a heterogeneous information system, the meaning of the information that is interchanged has to be understood across the systems. It is common knowledge that ensuring the semantic interoperability is much easier when staying in the same semantic domain [4, 5]. Though we are dealing with different IT systems that are not restricted to a specific domain, we can take advantage of the fact that it is not their business-related part that we are investigating. We look at very specific data objects: the users along with their identity and access management related properties.

There have been frequent discussions on how to handle heterogeneity when considering the semantic level. In [4], how to use ontologies in the context of information integration is discussed. The authors suggest reducing the hard-coding which does the translation between the terminologies of pairs of systems by applying ontologies to the formal specification of the meaning of terms. According to [6], three different directions can be identified when applying ontologies: single ontology approaches, multiple ontologies approaches and hybrid approaches being a mixture of both. Single ontology approaches use one global ontology providing a shared vocabulary for the specification of the semantics. All information sources are related to this global ontology. Single ontology approaches can be applied to integration problems where all information sources to be integrated provide nearly the same view of a domain. Using multiple ontology approaches, each information source is described by its own ontology. Quite often this moves the complexity to an intermediary ontology which is needed to enable the matching between the different domains. Hybrid approaches work with local ontologies as well but try to use a globally shared vocabulary or glossary, which is to be used by the human developer when designing the local ontologies. For our approach of enabling semantic integration of identity data repositories we can take advantage of the fact that the semantics of user-centric identity data have a common core which can be instantiated at almost all of the existing IT systems. This facilitates the creation of a single ontology approach describing a person and his/her attributes which can then be mapped to the different IT systems.

The main causes for semantic heterogeneity are classified in [7]. In our case the central focus at the semantic level is to address so-called “naming conflicts” that occur when naming schemes of information differ. “Scaling conflicts” and “confounding conflicts” are not to be expected in our scenario. Naming conflicts can occur if the attribute names of digital identities differ at the various systems. For example, think of *givenName='John'* and *firstName='John'*. This problem can be addressed by introducing synonyms which can be handled using ontologies. The granularity of information is a further point to be solved. Problems occur if in the one repository the attributes *firstName* and *lastName* exist whilst in the other repository only the concatenation of both, called *commonName* can be found. This can also be solved using ontologies.

The application of ontologies in integration scenarios is described in [5], which concentrates at the process for the definition of ontologies. The ontology design process is regarded as a bottom-up approach taking the schemas of multiple databases as input and producing as output a single unified database schema combined with a mapping from the individual databases to the unified database. As the number of IT systems that apply identity management and access control and therefore need user objects is enormous, a starting point has to be found where the parts of users' data are defined which are not application-specific. This is where the state-of-the-art concerning existing proposals about how to describe users has to be investigated.

Most of the present suggestions concerning how to describe a user are rather simple – they only deal with the basic properties a person can have. As a starting point the nearest idea is to examine the directories where the users' digital identity information is stored. The leading standard in this context is LDAP [8]. Though being expandable by creating new schemata or schema extensions the focus is clearly on the syntactic and structural layer. Attributes have "meaningful names" and can be described using plain text. For the integration of user data objects over more than one LDAP-based directory, the schema has to be interchanged and implemented at all participating directories, which reduces flexibility in the integration process. To set up on the syntactically defined data, simple ontologies have been defined [9, 10]. Further ontologies such as the "Enterprise Ontology" developed by the University of Edinburgh (available at the ontology library at [11]) also include rather primitive person classes. There are some other suggestions such as the HR-XML initiative [12]. The HR-XML consortium is a non-commercial and independent organization responsible for the release of a standard human resource vocabulary. Though no knowledge representation language such as OWL (Ontology Web Language, [13]) is used to describe the included items, we do use it as a starting point to set-up our *person ontology* as described in chapter 3.

2.2 Architectural Aspects

It is not enough to address only semantics when looking at identity data repository integration. There must be a way to handle the actual synchronization process over the different repositories. As repositories usually act passively with respect to event propagation, it must be possible to determine if a change has occurred in the repository. Architecturally different approaches are available which solve this problem. There is the retro change log (RCL) [14] featured by Sun which is implemented as an additional sub-tree in the directory server. Using RCL, a record of each change made to the directory server is stored by duplicating changed objects to a specific sub-tree. The idea is that external synchronization software checks this sub-tree, takes the information and finally deletes the entries there. Due to the tight alignment with the LDAP basis standards this is a very interoperable approach. A major deficiency is the need to regularly poll the directory there is nothing like a publish/subscribe-based event propagation. Problems occur if there is more than one remote directory trying to synchronize, because it is not defined when the last party is informed about the changed object and when the object can be removed.

The need for a proactive notification in case of changes inside the repository is described in [15]. There it is argued that an exclusively inactive (i.e. passive) interface to directory services can hinder server scalability and indirectly restrict the behavior of potential applications. So the authors propose to extend directory services' interfaces with a proactive mode with which clients can express their interest in changes in the environment according to a publish/subscribe paradigm. The problem of this approach is that only the passiveness is overcome but the semantic heterogeneity is not addressed which is still to be handled individually on a 1:1 basis by the synchronization application. Another solution for enhancing data repositories to proactively propagate data changes is described in [16] but again not on a semantic basis. The authors implemented an effect similar to a trigger in databases by adding an intercepting gateway which is capable of starting the propagation of data to different directories based on the content of the invoked action in front of their LDAP-based directory. This approach moves directories out of their role as a passive data source but it is strongly bound to LDAP-based directories. There is nothing like an aid in semantic matching as the point-to-point specific rules have to be hard-coded at the gateway.

3 Person Ontology

An ontology helps to separate the meaning of data from its representation. Thus the meaning of data is extracted from applications, databases and directories, and can be altered independently without having to adjust the data's representation itself. This approach offers different advantages. First of all, the use of ontologies can provide a unified nomenclature for the entities of the domain of interest. It also yields semantic uniqueness, which implies that entities in the ontology have a distinguishable and semantically well-defined meaning. Therefore it can be prevented that items that are syntactically (e.g. their names) but not semantically equal are believed to have the same meaning. Furthermore the use of an ontology leads to a more flexible integration since hard-coding of the meaning of data is prevented and changes to that meaning can be made without having to cope with the data itself. Another benefit of ontologies is that once a conceptualization of the specific domain has been accomplished, it is possible to share this knowledge in a well-defined and formal manner so that other parties are able to use that knowledge directly. Moreover, if an ontology is at hand it is possible to conduct consistency checks on its extension. Beyond that it is feasible to extract implicit knowledge from the ontology's extension. These facts lead to the conclusion that sophisticated identity management should be accomplished using ontologies. In the following an ontology representing the meaning of the person object will be elaborated in order to overcome the syntactic heterogeneity of user data within different data repositories.

The development of the *person ontology* has been tightly aligned to the approach described in [17]. The first issue that has been addressed in this context is the description of the ontology's domain. Basically that is the domain of IdM or more precisely persons' or users' identity data. After defining the domain, the next step is to investigate how the ontology will be used and what audience it appeals to. The answer to the first of these concerns is that it is designed for the integration of syntactically heterogeneous data sources holding digital identities. This implies that the main audience to make use of and augment the ontology will be essentially system integrators and identity managers responsible for the establishment of processes such as provisioning, synchronization and decommissioning.

With these prerequisites made explicit we started to enumerate the most important attributes associated with persons. The heuristics to do so is considering the importance of the attributes, or in other words the frequency they are encountered, from existing data representations such as LDAP or HR-XML. Some of the elements that we have distinguished this way are *commonName*, *address*, *credentials*, *title*, *email*, *telephone*, *fax* and *birthday*. As the next step we split up the elements into (simple) datatype properties (i.e. attributes) and into complex elements, which we defined as classes and put into a hierarchy using OWL. Examples for complex types are *commonName*, *address* and *credentials*. Additionally, the relationships between these classes were formalized using object properties. Thereby classes, such as *commonName* and *cn* have been characterised to be semantically equivalent. However classes that have divergent meanings such as *userPassword* from LDAP and *password* from HR-XML were marked to be semantically different. Finally the primitive attributes of each class had to be defined. This was done by using the datatype property construct provided by OWL. Thereby a special focus was set on the definition of semantically alike properties to automate the mapping process. To give a short example of the possible connections between classes, a part of the person class is described: A person has the *hasAddress* object properties that associates him with the *address* class. This *address* class in turn is a super-class to further classes such as *postalAddress*. A *postalAddress* again is connected to a *commonName* via the *hasCN* object property and holds datatype properties such as *street*, *zipCode*, *country*. A *street* is assembled from its *streetName* along with a *houseNumber* with and an optional *suffix*. This is just a small excerpt from the complete ontology. The major part of the ontology is reserved for the definition of syntactically divergent terms that have semantically the same meaning such as the attributes *mobile*, *mobilePhone* and *mobilePhoneNumber*.

We have developed the *person ontology* using Protégé [18] Version 3.1. The tool we used to verify OWL-DL conformity is the OWL Validator of WonderWeb [19]. To assure the correctness and consistency of the *person ontology* we deployed the Racer DIG (Description Logic Implementation Group) Reasoner [20]. In Figure 2 we depict an extract of our *person ontology*, graphically modelled using DLG² [21]. DLG² is a graphically-based language that can be used to simplify the presentation of RDF (Resource Description Framework) and therefore OWL models. The idea is to exemplary show the relevant constructs that we have applied in the ontology in a human readable manner. DLG² enables for a flexible modelling of datatype properties. They can either be defined inside a class or externally in an ellipse to allow modelling of equivalent properties.

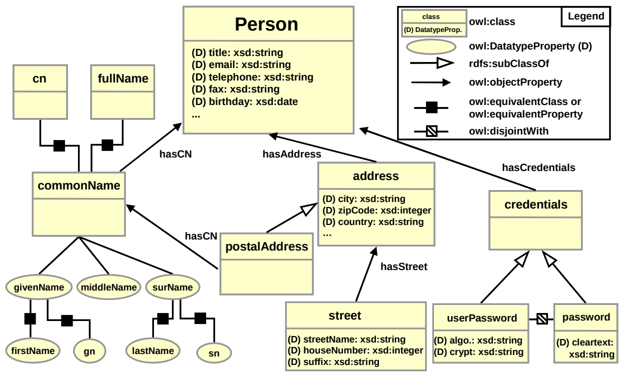

Figure 2. Person Ontology (Extract in DLG²)

4 Data Repository Container

Defining the semantics of users' digital identities using a global ontology builds the first element of the integration's core. The second is a suitable architecture building on this ontology and enabling the repository integration as traditional data sources are not capable of this. At a glance our approach is to extend the traditional, passive identity data sources by adding further components. We call an enhanced data source *data repository container* (DRC). Though a DRC still can act locally and autonomously, it is to be attached to a specific kind of message broker to which each DRC subscribes in order to both publish and receive information on changed user objects. In the following sub-chapters we introduce the architecture that we have developed.

4.1 Functional Requirements and Approach

The following functional requirements led to the development of the *data repository container* (DRC):

1. The DRC should be capable of handling any kind of common data storage technology, such as directories, relational database management systems (RDBMS), XML files or any similar kind of technology. This should be achieved by introducing an abstraction layer which we called *data source wrapper* (DSW).

2. To be able to align an incoming user object with the representation of the local repository, we employ a *semantic engine* (SE) which does the mapping and filtering of the incoming attributes to the local ones with the help of the *person ontology*.

3. Changes in the local source repository must be propagated proactively by the DRC instead of making the destinations poll the source on a regularly basis. As a means of interaction, a message-oriented approach should be pursued to enable a flexible, reliable and loosely-coupled mechanism for the information interchange. This functionality is put into a component that we call *semantic event dispatcher* (SED) that is located inside each DRC and takes care of both sending out and receiving these event messages containing the changed user objects.

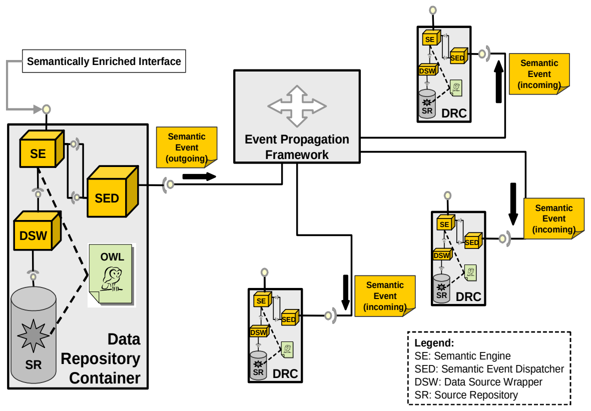
Semantically Enriched Interface
SE
SED
DSW
OWL
SR
Data Repository Container
Semantic Event (outgoing)
Event Propagation Framework
Semantic Event (incoming)
Semantic Event (incoming)
Semantic Event (incoming)
DRC
DRC
DRC
Legend:
SE: Semantic Engine
SED: Semantic Event Dispatcher
DSW: Data Source Wrapper
SR: Source Repository

Figure 3. Architecture for Semantic Directory Integration

4. With a growing number of DRCs, a point-to-point connection between all the containers results in high integration costs due to the n-squared problem when new *data repository containers* are being added. This can be avoided by setting up a message broker which we call *event propagation framework* (EPF) and to which every DRC is connected. Unlike traditional solutions, the basic setup of this EPF can be rather simple. By offering a standardized interface it is fed by its corresponding SED if a user object is created or changed (outgoing semantic event from the view of a DRC) and distributes this information to all DRCs participating (incoming semantic event).

4.2 Data Source Wrapper

The *data source wrapper* (DSW) acts as a converter between the underlying technology dependent protocol, e.g. LDAP, and a common protocol used to access it. Its design is based on the wrapper and adapter design pattern described in [22]. The fundamental idea behind a wrapper or adapter is to map the interface of a class, in this case the interface of the source repository, to an interface expected by a client. Thus it solves the problem of interoperability between incompatible interfaces. This is especially important to reuse functionality; in our case it enables the reuse of the components SE and SED. Compared with the SE that is described in the next section and which accomplishes semantic data integration, the DSW's job is to ensure interoperability between the SE and the elements on the data layer. Thus it accomplishes a syntactic integration of the source repositories. In conclusion, it becomes possible to disengage from the underlying protocol and present a uniform interface to the SE.

4.3 Semantic Engine

The *semantic engine* (SE) constitutes the central element within the DRC's architecture. Its responsibility is to semantically integrate person data from different sources based on the *person ontology*. Thus it performs the mapping of users' properties. This means that the SE basically does a semantic transformation from an incoming person object, either coming from its regular service interface or via notification from other

DRCs via the EPF, to a person object that is expected by the underlying, local repository. The SE is based on the proxy design pattern as described by [22]. Therefore it functions as an interface to the data repository. Accordingly all interaction with the data repository has to pass the SE. This enables the SE to detect all relevant changes in the repository. Moreover, the SE can raise events based on the actions performed on the repository and dispatch these events via the *semantic event dispatcher* (SED).

4.4 Semantic Event Dispatcher

The *data repository containers* should be able to exchange events in a flexible, reliable and standardized way. To uphold the principle of separation of concerns the SE should not be responsible for communication issues. Therefore another component is needed that hides the complexity of message exchange to the SE. This is precisely the task of the *semantic event dispatcher* (SED). Its purpose is to expose an interface to the SE, thus making the interaction logic transparent for the SE. If an event has been detected by the SE, it is passed to the SED which takes care of the communication with the *event propagation framework*. The SED is designed according to the design pattern façade as introduced by [22]. All logic involving the distribution of events is delegated to the SED. Events are published on so-called topics. A conceivable example of topics in such a context could be <hostname>.update or <hostname>.create. To get the full information, the subscription of *.* is recommended, but DRCs that are interested only in updates from a specific DRC could subscribe to the topic <hostname>.* at the EPF. Parallel to the distribution of events, the SED is also accountable for the reception of events by subscribing to the relevant topics at the EPF.

4.5 Event Propagation Framework

A central *event propagation framework* (EPF) is a connector between the DRCs that reduces point-to-point communication between different DRCs. It acts as the authority responsible for the distribution of the changes in user objects. The DRCs subscribe at the EPF and there is the possibility to define different topics. The tagging of the events to specific topics is done by the sending SED, so the EPF acts as a message broker with all the features as asynchronicity and loose-coupling.

4.6 Collaboration Aspects

Figure 4 illustrates how the components forming the *data repository container* work together. This is exemplary shown by a DRC receiving an (incoming) event from the EPF.

An event sent by the EPF is received by the SED. Each event is associated with a certain topic it is published on. In order to receive events the SED must have previously subscribed to the appropriate messaging topic at the EPF. After the SED has received an event it is passed on to the SE using the *processEvent* method. The SE takes on the semantic processing of the event by aligning its syntax to the syntax of the local DSW based on the *person ontology*. After this has been accomplished the appropriate action on the source repository is executed. In this case this action is a *delete* operation.

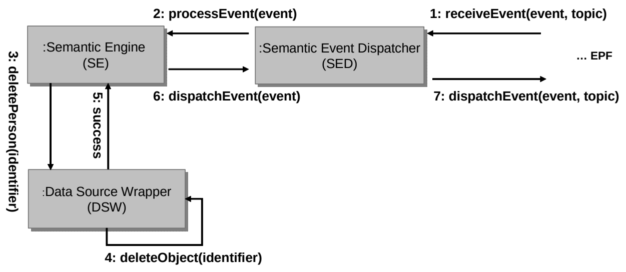
2: processEvent(event)
1: receiveEvent(event, topic)
:Semantic Engine (SE)
:Semantic Event Dispatcher (SED)
... EPF
6: dispatchEvent(event)
7: dispatchEvent(event, topic)
3: deletePerson(identifier)
5: success
:Data Source Wrapper (DSW)
4: deleteObject(identifier)

Figure 4. Data Repository Container – Collaboration Diagram

After the action has successfully been performed by the DSW the SE dispatches a new event since a deletion has occurred. This message is again passed to the SED. The SED in turn posts the event on a certain topic (e.g. <host-name>.acknowledgement) so that further DRCs are accurately notified and data consistency is assured. This enables other DRCs or a centralized auditing system to check if all DRCs have processed a specific event.

5 Implementation Case Study

We have developed our solution to fit into a project context at a major automotive company. Though our design is mostly technology independent, the preference was to use Java technology for the actual implementation. With the blue-print of a future service-oriented architecture (SOA) that is planned, the outer interface of the DRC was to be implemented as a web service which implies a WSDL-style interface description as well as SOAP communication. JBoss was the choice considering the necessary application server which a DRC is deployed to. The wrapping to web service interfaces is done by the JBoss internal component WS4EE. The three components SE, SED and DSW are implemented as Enterprise Java Beans (EJB). The *semantic engine* is a stateless session bean and its interface is exposed as a web service as an enhancement to the simple and proprietary interface of the source repository. The *semantic event dispatcher* is implemented as a message driven bean and utilizes the concepts of Java Messaging Services (JMS) to communicate with the EPF and the SE. The *data source wrapper* is implemented as an entity bean representing a standardized storage for the user objects. The components and their coupling are depicted in the architectural overview in Figure 5. The *semantic engine* is supported by a helper class, the s*emantic mapper* class. This class does the semantic mapping of and creation of person objects based on the *person ontology* elaborated in chapter 3. To access the ontology we use the JENA API [23]. Mapping and transformation are done based on the knowledge given by the ontology. This means that if further mappings have to be defined only the ontology must be altered but not the code of the EJBs.

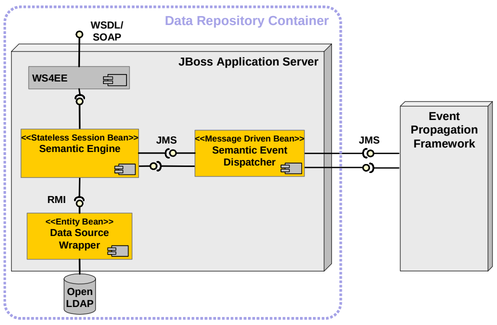

Figure 5. Case Study – Implementation of a Data Repository Container

In this scenario, various directories had to be synchronized. There was a distinction between offline and online synchronization. The latter means that change events are propagated short time after they occurred. We focused on the online synchronization. The synchronization application in use was proprietarily developed with individual logic for point-to-point synchronization. We set up the EPF as a central message broker and attached the corporate meta directory (Sun Directory Server) as well as the Microsoft Active Directory, PeopleSoft ePeople and Lotus Notes. The overall amount of classes that have been defined in the ontology is 9. It further contains 71 datatype and object properties. The OWL file has 800 lines and is about 40 kilobytes in size.

6 Conclusion and Further Work

In this paper a solution for the semantic integration of person objects has been presented. We have introduced a core set of a *person ontology* that can be flexibly extended as well as an architecture enhancing traditional identity repositories to active and semantic-enabled *data repository containers*. These enable the integration process for user provisioning, decommissioning and synchronization. For the loose coupling of the different *data repository containers* we developed the *event propagation framework* as a message broker. It allows the distribution of events between the different DRCs and centralizes the point-to-point connections.

Currently the development of the ability to dispatch events without the need of an active change in the source directory is a main issue which must be solved. For example the date of a person leaving the company is quite often recorded in advance, so the SED should be able to dispatch the event automatically at the point of time it is needed for the overall decommissioning process. Another issue is the implementation of natural language processing for the *semantic engine* in order to allow a declarative data source access. With upcoming service-oriented architecture (SOA) in mind, we have already applied SOA paradigms like loose coupling and web service interfaces achieving better interoperability. For a further SOA alignment, the embedding of the *event propagation framework* to the enterprise service bus (ESB) of an SOA is to be tightened.

7 References

1. Burton Group: Concepts and Definitions (Glossary), Version 2.0, September 2005.
2. Phillip J Windley: Digital Identity, O'Reilly Media; 1st edition, August 2005.
3. V. Kashyap and A. Sheth: Schematic and semantic semilarities between database objects: A context-based approach. The International Journal on Very Large Data Bases, 5(4):276–304, 1996.
4. Zhan Cui, Dean Jones and Paul O'Brien: Issues in Ontology-based Information Integration, IJCAI Seattle / USA, 2002.
5. Chris Partridge: The Role of Ontology in Semantic Integration, OOPSLA 2002, Seattle.
6. H.Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann and S. Hübner: Ontology-Based Integration of Information - A Survey of Existing Approaches, Intelligent Systems Group, Center for Computing Technologies, University of Bremen, 2001.
7. Cheng Hian Goh. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources, MIT, 1997.
8. Lightweight Directory Access Protocol (v3). URL: http://www.ietf.org/rfc/rfc2251.txt
9. Li Ding, Harry Chen, Lalana Kagal, Tim Finin: DAML Person Ontology, 2002.
   URL: http://daml.umbc.edu/ontologies/ittalks/person ´
10. UMBC Ebiquity Research Group: Person Ontology.
    URL: http://ebiquity.umbc.edu/ontology/person.owl
11. Standord University: Knowledge Systems Laboratory Ontology Editor, June 2006.
    URL: http://www-ksl-svc.stanford.edu:5915/
12. Homepage of the HR-XML Consortium.
    URL: http://www.hr-xml.org/
13. World Wide Web Consortium (W3C): OWL Web Ontology Language Overview, W3C Recommendation, February 2004.
    URL: http://www.w3.org/TR/owl-features/
14. Sun: Retro Change Log Plug-In.
    URL: http://docs.sun.com/source/816-6698-10/replicat.html#15790
15. Fabian E. Bustamante, Patrick Widener and Karsten Schwan: A Case for Proactivity in Directory Services, Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC), 2002.
16. Robert Arlein, Juliana Freire, Narain Gehani, Daniel Lieuwen, and Joann Ordille: Making LDAP Active with the LTAP Gateway, in Proceedings Workshop on Databases in Telecommunication, September 1999.
17. Natalya F. Noy and Deborah L. McGuinness: Ontology Development 101: A Guide to Creating Your First Ontology, Stanford University, Stanford, CA, 94305, 2001.
18. Stanford Medical Informatics: Protégé Ontology Editor, 2005.
    URL: http://protege.stanford.edu
19. Sean Bechhofer and Raphael Volz: WonderWeb OWL Ontology Validator, 2003.
    URL: http://phoebus.cs.man.ac.uk:9999/OWL/Validator
20. Racer DIG Reasoner, July 2006.
    URL: http://www.racer-systems.com/de/index.phtml, http://dig.sourceforge.net/
21. Xiaoshu Wang, Jonas S. Almeida: DLG2 - A Graphical Presentation Language for RDF and OWL, 2005.
    URL: http://charlestoncore.musc.edu/docs/dlg2.html
22. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns, Addison Wesley, 1998.
23. Hewlett-Packard Development Company: JENA – A Semantic Web Framework for Java, 2005. URL: http://jena.sourceforge.net