# RDF Containers – A Framework for the Integration of Distributed and Heterogeneous Applications

Richard Mordinyi[1,2], Thomas Moser[2], Martin Murth[1], Eva Kühn[1], and Stefan Biffl[2]

[1] Space-Based Computing Group, Vienna University of Technology, Austria
[2] Christian Doppler Laboratory "Software Engineering Integration for Flexible Automation Systems", Vienna University of Technology, Austria
{firstname.lastname}@tuwien.ac.at

**Abstract.** Current trends like globalization and virtual enterprises result in an increasing need for on-the-fly integration of distributed applications grown over the past decades and originally not intended for integration and cooperation. In order to enable the collaboration of such applications, aspects of distribution, such as heterogeneous data sources, heterogeneous network technologies and coordination requirements, have to be addressed in the integration process. In this position paper, we introduce a framework, the so-called RDF Containers, for technical and semantic integration of distributed and heterogeneous data sources considering integration requirements. We discuss the benefits and limitations of the proposed framework based on a real-world use case from the e-health domain. The major benefit is that both semantic and technical integration are supported by a single framework, while complexity aspects related to the integration process do not affect the integrated applications.

**Keywords:** semantic integration, space-based computing.

## 1 Introduction

Today's communication network technologies offer enough bandwidth and interconnection facilities for the development of applications based on combining well-established distributed applications, resulting in the increasing need for efficient and effective system integration methodologies [1]. Core questions are on the one hand side how to satisfy communication and coordination requirements between technologically different application platforms probably belonging to different autonomously acting enterprises. On the other hand side how to integrate data models across platform and domain boundaries, and finally how software engineers can be supported in designing and implementing distributed applications.

Hence, typical system integration challenges result from both technical heterogeneities, e.g., tools from different sources may use a range of technologies that become expensive and error-prone to integrate in traditional point-to-point ways; as well as from semantic heterogeneities, e.g., project participants may use different terms for common concepts in the application domain [2].

Modern technical integration approaches, such as the Enterprise Service Bus (ESB) concept [1], rely on message-based infrastructures and are capable of abstracting

R. Meersman et al. (Eds.): OTM 2010 Workshops, LNCS 6428, pp. 90–99, 2010.

complexity issues of distributed systems from the application. However, coordination logic (e.g., message ordering or coordination patterns) still remains in the application, hindering the application designers to focus entirely on the application itself [3]. Current alternative solutions for semantic integration like standards for data models [4], data-driven tool integration [5], or complete data transformation [6] work in principle, but pose their own challenges, such as inefficient and complex data access and query definitions, solutions which are not robust enough, or take considerable effort to develop and modify.

In this position paper, we introduce the RDF Containers approach as a platform for technical and semantic integration of distributed and heterogeneous data sources. RDF Containers combine the space based computing paradigm (SBC) for solving technical integration and coordination challenges [7], and the Engineering Knowledge Base (EKB) framework for supporting semantic integration [8]. SBC is a coordination middleware based on concepts of virtual shared memory that explicitly distinguishes between computation and coordination logic. This allows shifting coordination complexities into the middleware layer, thus minimizing the implementation effort for the application developers needed to control these coordination complexities and therefore allows focusing on application development entirely. Additionally, SBC provides mechanisms to abstract issues of technical heterogeneity in distributed systems, and therefore facilitates technical integration [7]. The EKB framework uses ontologies for explicitly modeling common and local concepts as well as mappings between these concepts, thus enabling semantic integration in multi-organizational scenarios. Standards are hard to apply for projects with experts from different organizations, who have independently invested efforts into the development of different kinds of local data standards or notations. The EKB framework allows these experts to use their established and well-known local tools and data models, while additionally providing access to data originating from tools of other organizations within their local tools using local data standards or notations [8, 9].

We discuss the benefits and limitations of RDF Containers based on a real-world use case from the e-Health domain, in which data regarding accident victims is dynamically consolidated from various sources (e.g., hospitals, doctors) and used for the coordination of their treatments. Major benefit of RDF Containers is that both semantic and technical integration are supported by a single framework, while complexity aspects related to the integration and coordination of processes do not affect the integrated applications.

The remainder of this paper is structured as follows: Section 2 summarizes related work on technical and semantic integration. Section 3 identifies the research issues, while section 4 introduces the use-case. Section 5 presents the proposed approach, section 6 discusses the findings, and finally section 7 concludes the paper and presents further work.

## 2 Related Work

This section summarizes related work on technical and semantic integration.

### 2.1 Technical Integration

System integration is the task to combine numerous different systems to appear as one big system. There are several levels at which system integration could be performed [10], but there is so far no standardized out-of-the-box solution for an integration process that explains how to integrate arbitrary systems. The limitations of integration over heterogeneous middleware technologies with different APIs, transportation capabilities, or network architecture styles implies the development of static and therefore inflexible wrappers between each combination of middleware technologies, and thus increases the complexity of communication. Traditional approaches for integration of business services can be categorized [1] into: Hub and spoke vs. distributed integration and coupled vs. separated application and integration logic. As an example, the Enterprise Service Bus (ESB) provides the infrastructure services for message exchange and routing as the infrastructure for Service Oriented Architecture (SOA) [11]. It provides a distributed integration platform and clear separation of business logic and integration logic. It offers routing services to navigate the requests to the relevant service provider based on a routing path specification. By relying on its basic functionality of routing messages between individual services, an ESB supports only a very basic type of coordination form. In case services need to coordinate each other in a more complex way in order to achieve a common goal, the complexity of changing coordination requirements cannot be realized in the middleware. Thus, the services themselves need to be adapted, such that by making use of the supported capabilities of an ESB only, the new business goal can be achieved. This implies increased implementation complexity [12], decreased efficiency [3], and higher service development time.

### 2.2 Semantic Integration

Semantic integration is defined as the solving of problems originating from the intent to share data across disparate and semantically heterogeneous data [2]. These problems include the matching of ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modeling of complex relations in different data sources. [13] One of the most important and most actively studied problems in semantic integration is establishing semantic correspondences (also called mappings) between vocabularies of different data sources [14]. There are three main categories of semantic conflicts in the context of data integration that can appear: confounding conflicts, scaling conflicts, and naming conflicts. The use of ontologies as a solution option to semantic integration and interoperability problems has been studied over the last 10 years. Noy [15] identified three major dimensions of the application of ontologies for supporting semantic integration: the task of finding mappings (semi-)automatically, the declarative formal representation of these mappings, and reasoning using these mappings.

There exist two major architectures for mapping discovery between ontologies. On the one hand, the vision is a general upper ontology which is agreed upon by developers of different applications. On the other hand, there are approaches comprising heuristics-based or machine learning techniques that use various characteristics of ontologies (e.g., structure, concepts, instances) to find mappings. These approaches are similar to approaches for mapping XML schemas or other structured data [16].

Naturally, defining the mappings between ontologies, either automatically, semi-automatically, or interactively, is not a goal in itself. The resulting mappings are used for various integration tasks: data transformation, query answering, or web-service composition, to name a few. Given that ontologies are often used for reasoning, it is only natural that many of these integration tasks involve reasoning over the source ontologies and the mappings [13].

## 3 Research Issues

The need for designing and implementing collaborating applications based on distributed and heterogeneous data sources is steadily increasing since today's communication network technologies offer enough bandwidth and interconnection facilities for such applications, as well as because of current trends towards globalization and virtual enterprises [1]. Because of technical and semantic gaps between different stakeholders and platforms, there are limitations to a comprehensive systems integration methodology. Core questions are on the one hand side how to enable communication and coordination between technologically different applications that typically originate from inter-enterprise collaborations, and on the other hand side how to integrate data models across platform and domain boundaries. In both cases, the complexity regarding the integration of such applications should not affect the integrated applications.

Traditional system integration approaches, such as the Enterprise Service Bus (ESB) concept [1], face challenges regarding technical integration and the efforts needed to cope with complexity issues (like cognitive complexity) of coordinating autonomous enterprises; as well as from semantic heterogeneities, e.g., project participants may use different terms for common concepts in the application domain [2].

Based on the limitations of traditional methodologies regarding technical and semantic integration of distributed applications and their heterogeneous data sources, we propose a unified system integration methodology capable of dealing with both technical and semantic heterogeneities. Therefore, we derive the following research issues.

**RI-1: Technical Integration taking into account coordination requirements.** To what extent do current technical integration frameworks support both the abstraction of heterogeneous network technologies and the flexible coordination of distributed applications?

**RI-2: Semantic Integration without the need for a common data schema.** Investigate the capabilities of current semantic integration solutions that do not imply the usage of a common data schema (which is hard or even impossible to achieve for well-established applications). How could semantic integration solutions support changes (e.g., new or updated data sources) of available data sources?

**RI-3: Effects on integrated distributed applications.** How could additional complexity resulting from the integration process itself be hidden from the integrated applications?

We discuss the proposed RDF Containers framework based on a real-world use case from the e-Health domain, in which patient summaries originating from various sources (e.g., hospitals, doctors) should be made available for healthcare authorities in EPS (European Patient Summary) style [17].

## 4 Use Case

The following use case from the e-Health area describes an extension of the use case defined in TripCom [18], where the development of a Europe-wide information system for the management and exchange of patient data was considered - also referred to as the European Patient Summary (EPS) [17] In the EPS use case, we employed RDF representations of patient data in shared spaces to achieve scalability for discovery of data published in the Internet. The extended use case introduces two additional requirements, describing frequently observed demands in real-world system implementations: (i) data about accident victims needs to be integrated in an ad-hoc manner, thus allowing to provide a consolidated view on the patients' health records within minutes or even seconds, and (ii) it must be possible to spontaneously change the treatments and treatment orders of patients, hence imposing high requirements on the coordination capabilities of the developed system.

**Story board:** During a trip through the Austrian Alps, a bus collides with a motor vehicle and many of the occupants are seriously injured. The first who arrive at the accident site are several ambulance men, providing first aid to the accident victims. Depending on type and severity of the injury, they request further ambulance cars and medical support. For this purpose, each ambulance man is equipped with a mobile device that can be used to record patient data, required medical support, and to request the transportation to a nearby hospital.

To ensure that the best possible medical treatment can be given to the accident victims, data from multiple and potentially heterogeneous data sources need to be consolidated: a) data from the accident site provided by the ambulance man, e.g., identity of the accident victim, type and severity of the injury, etc.; b) data from the accident victim's health care providers in their home countries, e.g., health records from the victim's general practitioners and specialists, x-ray images, information about allergies or intolerances, etc.; and c) data from the ambulances and the nearby hospitals, e.g., medical equipment and treatment options, availability of operating rooms, medications available in ambulance cars, etc.

The developed information system needs to integrate these data and to provide it to the physicians and clinical personnel of the hospitals. It further needs to determine which ambulance cars can take which injured persons and it has to coordinate the transportation of the victims to the surrounding hospitals. As there are typically not enough ambulance cars available right after an accident, it is also important that the ambulance cars are always used to maximum capacity. For example, slightly injured people should not be transported separately, and ambulance cars with emergency doctors should only transport seriously injured persons. At the hospitals, the accident victims are treated in the order in which they arrive. However, in case of critical injuries, the treatment of certain victims may need to be prioritized. It is therefore necessary that the single departments of a hospital can coordinate the treatment orders, that they can synchronize their most recent treatments, and that they can spontaneously adjust the treatment method when new information is provided.

# 5 RDF Containers

This section shortly introduces the Space-Based Computing (SBC) paradigm and the Engineering Knowledge Base (EKB) framework, which are combined by the RDF Containers approach described in the third subsection.

## 5.1 The Space-Based Computing (SBC) Paradigm

SBC explicitly distinguishes between computational and coordination logic, thus moving the complexities of coordination into the middleware layer allowing the application to focus on its business goals. It provides an extensible coordination logic with its main concepts being containers, coordinators, and aspects [3, 12]. Coordinators are Internet addressable resources that contain data (numbers, strings, records etc.) in a structured way. Each container possesses one or more coordinators (FIFO, key, random, vector etc.) that determine the semantic of accessing data. Pre and post aspects can be added to intercept operations on the containers. This allows supporting different architectural styles simultaneously and contributes to efficiently realize new business requirements without influencing the application. Especially, the coordinator concept represents the way how distributed applications coordinate themselves. By means of supporting coordination patterns within the middleware, a) the complexity of coordination can be decreased in the application, and b) allows the efficient coordination of distributed application, since the model explicitly represents business coordination requirements [19].

## 5.2 The Engineering Knowledge Base (EKB) Framework

The Engineering Knowledge Base (EKB) framework [8] is an ontology-based data modeling approach which supports explicit modeling of existing knowledge in machine-understandable syntax (e.g., knowledge of the EPS domain), with a focus on providing links between local data structures (e.g., English and Austrian National Patient Summaries) support the exchange of information between these local data structures and thus making systems engineering more efficient and flexible. The EKB framework stores the local knowledge in ontologies and provides semantic mapping services to access design-time and run-time concepts and data. The general mechanism of the EKB framework uses common concepts identified beforehand as basis for mappings between proprietary local knowledge and more generic domain-specific knowledge to support transformation between these tools and platforms [8, 9]. Due to the mappings between local ontologies and domain ontology data structures that are semantically equal can be identified, because they are either aligned to the same domain concept or belong to the same tree segment in the concept tree described in the domain ontology [20].

## 5.3 RDF Containers – Overview and Architecture

The basic architectural model consists of a set of data sources (e.g., triple stores) which conform to different heterogeneous data schemas. In our example, these data sources represent the different National Patient Summaries (NPS), e.g., English and

Austrian patient summaries, which use their own notations. The data sources are geographically distributed across Europe and thus need to be integrated both technically and semantically in order to meet the requirements of the European Patient Summary (EPS) scenario.

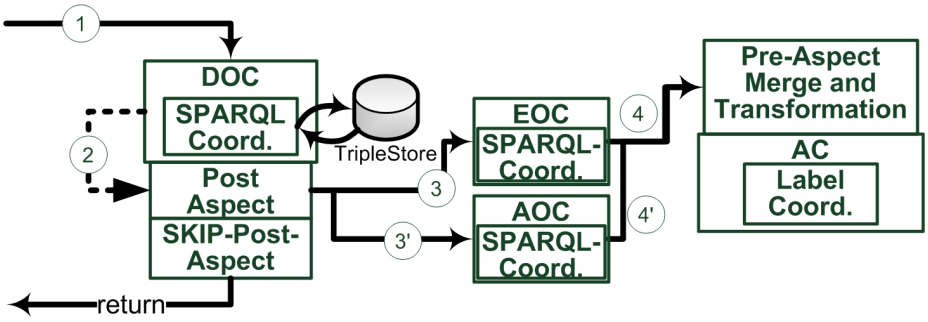


**Fig. 1.** Overview of the RDF Container Architecture for the EPS Use Case

Figure 1 depicts the architecture and an exemplary workflow of the RDF Containers for the EPS scenario. The architecture primarily consists of a container representing the common concepts of the EPS domain (*Domain Ontology Container-DOC*) and a set of containers representing the local concepts of the different NPSs (*English and Austrian Ontology Container-EOC, AOC*). In order to access the data stored in the data sources (e.g., triple stores), a special *SPARQL Coordinator* has been implemented and deployed. This coordinator abstracts various types of data sources that could be queried using SPARQL syntax. The exemplary workflow represents a read operation querying for information (e.g., regarding prior treatments or known allergies of a patient) from the available heterogeneous data sources.

As first step, an authorized role performs a query at the *DOC*. The operation provides two parameters: a unique ID identifying the operation itself, and a reference to the *Answer Container AC*. An AC is a container that stores the result of an operation in order to enable the execution of asynchronous operations. The coordinator maps the method call to an SPARQL query and executes it on the deployed TripleStore. The result of this operation indicates that two more additional operations (i.e., by querying the domain ontology the number of the local data sources to be queried, as well as their location identified through their namespace emerges) on national data sources have to be performed. In the second step a post-aspect intercepts the operation which by now also contains the results of its execution on the container. Based on the result set the post aspect realizes that two more operations, on national data sources, have to be performed. Therefore, the post-aspect performs automatically transformed versions of the original query on the English and Austrian NPSs asynchronously using the same parameter for the AC but changed IDs. In a next step the *SKIP-post-aspect* is called, which aborts the operation. However, as in the previous aspect the new operations have been executed in an asynchronous manner, only the operation on the *DOC* has been aborted without having any effects on the two new operations on *EOC* (step 3) and *AOC* (step 3'). Those two containers independently perform the

query operations, and based on the specified answer container, they write (steps 4 and 4') the result into the given *AC*. This container contains a pre-aspect that intercepts incoming operations in order to satisfy its two main functionalities. The first one is to wait for all sub-results belonging to the ID (the number of sub-results to wait for is determined by the identified number of local data sources to be queried) of the operation of step 1. The second task of that aspect is to merge the incoming results. The merge operation is performed by means of transformation instructions, so-called T-Maps [20] which describe how a concept can be mapped to another and how data can be manipulated (e.g., merged, transformed). Once the aspect finished manipulating incoming results based on the requirements of the operation of step 1, it writes the final results into the answer container *AC*.

## 6 Discussion and Conclusion

Current trends in IT like globalization and virtual enterprises result in an increasing need for on-the-fly integration of distributed applications grown over the past decades and originally not intended for integration [1]. Typical system integration challenges result from both technical heterogeneities, e.g., tools from different sources may use a range of technologies that become expensive and error-prone to integrate in traditional point-to-point ways; as well as from semantic heterogeneities, e.g., project participants may use different terms for common concepts in the application domain [2]. In addition, in most modern technical integration approaches, coordination logic (e.g., message ordering or coordination patterns) still remains in the application, hindering the application designers to focus entirely on the application itself [3].

In this position paper, we introduce the RDF Containers approach as a platform for technical and semantic integration of distributed and heterogeneous data sources. RDF Containers combine the space based computing paradigm (SBC) for solving technical integration and coordination challenges [7], and the Engineering Knowledge Base (EKB) framework for supporting semantic integration [8]. We discussed the benefits and limitations of RDF Containers based on a real-world use case from the e-Health domain, in which patient summaries originating from various sources (e.g., hospitals, doctors) should be made available for healthcare authorities in EPS (European Patient Summary) style [17, 18].

**RI-1: Technical Integration taking into account coordination requirements.** Regarding technical integration issues of the use case, the proposed RDF Containers offer advantages such as a) the efficient implementation of the used coordination patters (e.g., FIFO, LIFO, auction, marketplace); b) the support of both centralized (e.g., server-based) and distributed (e.g., peer-to-peer) integration approaches, which appear transparent to the application designer because of the abstraction capabilities of the SBC paradigm; and c) the support of several different architectural styles (e.g., dataflow, data-centered, implicit invocation), enabling the concurrent and/or exchangeable usage of different technical integration approaches. While discussing the use case scenario, we identified two possible limitations of using the SBC paradigm, namely on the one hand side the possible performance decrease because of the introduction of an additional abstraction layer, and on the other hand side the need for a change of mind regarding traditional application design and implementation.

**RI-2**: **Semantic Integration without the need for a common data schema.** Regarding semantic integration issues of the use case, the proposed RDF Containers offer advantages such as a) the fact that there is no common data schema needed which all project participants / data sources have to agree on; b) the possibility to define queries of heterogeneous data sources on a domain level without the need to stick to local and proprietary notations or syntax; and c) the automated derivation of transformation instructions of local and proprietary data models based on the identified common concepts and the mappings between these common concepts and the local, tool- and platform-specific concepts. While discussing the use case scenario, we identified two possible limitations of using the EKB framework, namely on the one hand side the initial effort for setting up the EKB framework properly, which may turn out to be too high for small applications, and on the other hand side the additional training effort for IT personnel regarding ontologies.

**RI-3: Effects on integrated distributed applications.** Additionally, the RDF Containers approach facilitates the architectural design and implementation of distributed applications. This allows shifting coordination complexities into the middleware layer, thus minimizing the implementation effort for the application developers needed to control these coordination complexities and therefore allows focusing on application development entirely. Furthermore, RDF Containers allow domain experts to use their established and well-known local tools and data models, while additionally providing access to data originating from tools of other organizations within their local tools using local data standards or notations [8, 9].

**Future work.** Future research will include exhaustive empirical evaluation in order to measure the efficiency and robustness of RDF Containers, and additionally the usability of the approach will be evaluated with industry practitioners. We will also investigate advanced data placement and data replication strategies based on extended semantic descriptions of the data (e.g., privacy, performance).

**Acknowledgments.** This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria, and by the Vienna University of Technology, Complex Systems Design & Engineering Lab.

## References

1. Chappel, D.A.: Enterprise Service Bus. O'Reilly Media, Sebastopol (2004)
2. Halevy, A.: Why your data won't mix. Queue 3, 50–58 (2005)
3. Kühn, E., Mordinyi, R., Keszthelyi, L., Schreiber, C.: Introducing the concept of customizable structured spaces for agent coordination in the production automation domain. In: 8th Int. Conf. on Autonomous Agents and Multi-Agent Systems, pp. 625–632 (2009)
4. Kruchten, P.: The rational unified process: an introduction. Addison-Wesley, Reading (2000)
5. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, Reading (2004)
6. Assmann, D., Dörr, J., Eisenbarth, M., Hefke, M., Soto, M., Szulman, P., Trifu, A.: Using Ontology-Based Reference Models in Digital Production Engineering Integration. In: 16th IFAC World Congress, Prague, Czech Republic (2005)

7. Mordinyi, R., Moser, T., Kühn, E., Biffl, S., Mikula, A.: Foundations for a Model-Driven Integration of Business Services in a Safety-Critical Application Domain. In: Software Engineering and Advanced Applications, Euromicro Conference, pp. 267–274 (2009)
8. Moser, T., Biffl, S., Sunindyo, W.D., Winkler, D.: Integrating Production Automation Expert Knowledge Across Engineering Stakeholder Domains. In: Int. Conf. Complex, Intelligent and Software Intensive Systems (CISIS 2010), pp. 352–359 (2010)
9. Biffl, S., Sunindyo, W.D., Moser, T.: Bridging Semantic Gaps Between Stakeholders in the Production Automation Domain with Ontology Areas. In: 21st Int. Conf. on Software Engineering and Knowledge Engineering, pp. 233–239 (2009)
10. Balasubramanian, K., Gokhale, A., Karsai, G., Sztipanovits, J., Neema, S.: Developing Applications Using Model-Driven Design Environments. Computer 39, 33 (2006)
11. Papazoglou, M.P., Heuvel, W.-J.: Service oriented architectures: approaches, technologies and research issues. The VLDB Journal 16, 389–415 (2007)
12. Mordinyi, R., Kühn, E., Schatten, A.: Space-based Architectures as Abstraction Layer for Distributed Business Applications. In: 4th Int. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS 2010), pp. 47–53 (2010)
13. Noy, N.F., Doan, A.H., Halevy, A.Y.: Semantic Integration. AI Magazine 26, 7–10 (2005)
14. Doan, A., Noy, N.F., Halevy, A.Y.: Introduction to the special issue on semantic integration. SIGMOD Rec. 33, 11–13 (2004)
15. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. SIGMOD Rec. 33, 65–70 (2004)
16. Bergamaschi, S., Castano, S., Vincini, M.: Semantic integration of semistructured and structured data sources. SIGMOD Rec. 28, 54–59 (1999)
17. Krummenacher, R., Simperl, E., Nixon, L., Cerizza, D., Valle, E.D., Della, E.: Enabling the European Patient Summary Through Triplespaces. In: 20th IEEE International Symposium on Computer-based medical systems, pp. 319–324 (2007)
18. Cerizza, D., Valle, E.D., Francisco, D.D., Krummenacher, R., Munoz, H., Murth, M., Simperl, E.P.-B.: State of the art and requirements analysis for sharing health data in the triplespace (2007), http://www.tripcom.org/docs/del/D8B.1.pdf
19. Mordinyi, R.: Managing Complex and Dynamic Software Systems with Space-Based Computing. Phd Thesis, Vienna University of Technology (2010)
20. Moser, T., Schimper, K., Mordinyi, R., Anjomshoaa, A.: SAMOA - A Semi-automated Ontology Alignment Method for Systems Integration in Safety-critical Environments. In: 2nd IEEE Int. Workshop on Ontology Alignment and Visualization, pp. 724–729 (2009)