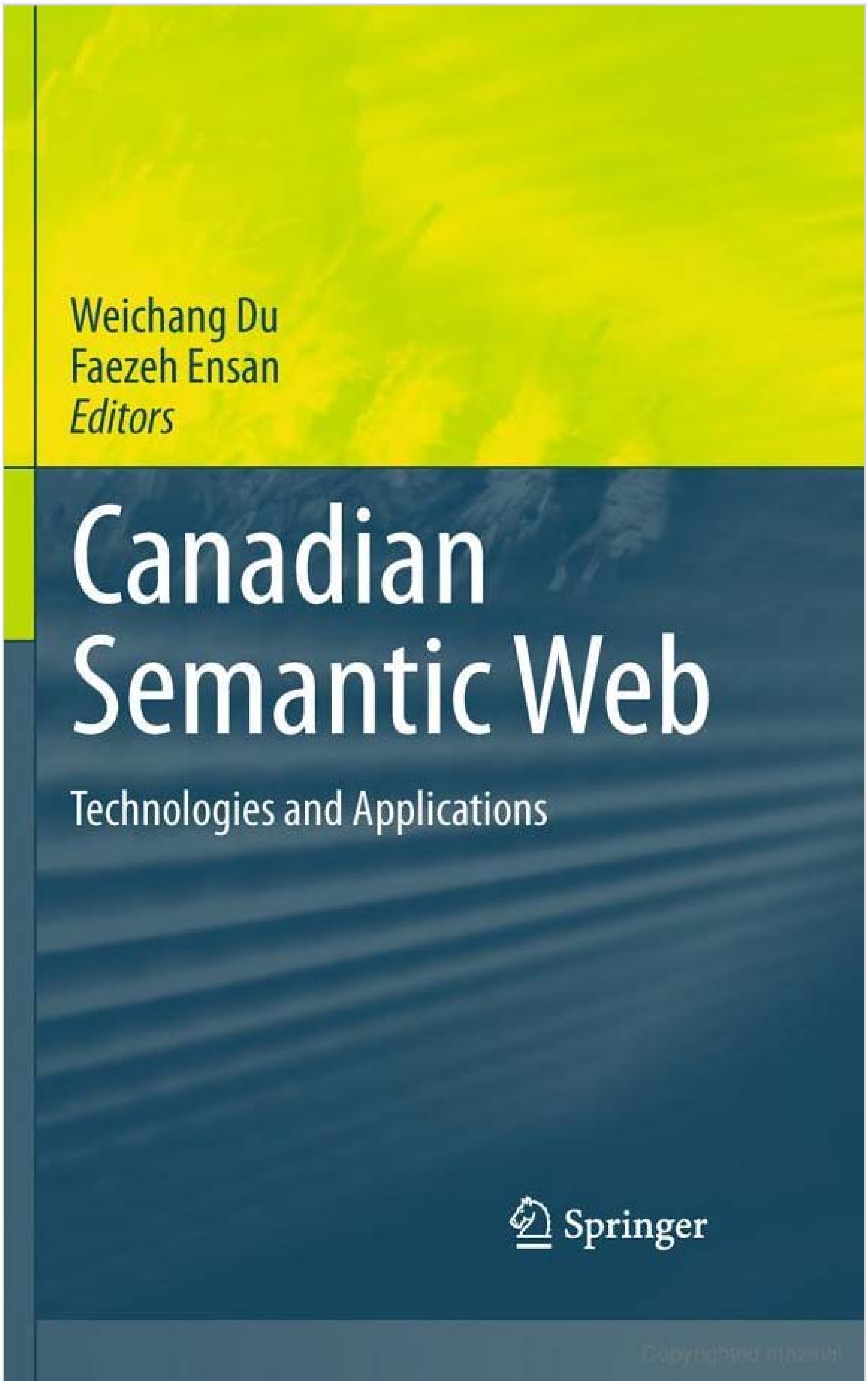


Weichang Du
Faezeh Ensan
Editors

Canadian Semantic Web

Technologies and Applications

Springer

Editors

Weichang Du
Faculty of Computer Science
University of New Brunswick
E3B 5A3 Fredericton
Canada
wdu@unb.ca

Faezeh Ensan
Faculty of Computer Science
University of New Brunswick
E3B 5A3 Fredericton
Canada
faezeh_ensan@yahoo.com

ISBN 978-1-4419-7334-4 e-ISBN 978-1-4419-7335-1
DOI 10.1007/978-1-4419-7335-1
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010933844

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Contents

Chapter 6
Semantic Service Matchmaking in the ATM Domain Considering Infrastructure Capability Constraints

Thomas Moser, Richard Mordinyi, Wikan Danar Sunindyo, and Stefan Biffl

Abstract In a service-oriented environment business processes flexibly build on software services provided by systems in a network. A key design challenge is the semantic matchmaking of business processes and software services in two steps: 1. Find for one business process the software services that meet or exceed the BP requirements; 2. Find for all business processes the software services that can be implemented within the capability constraints of the underlying network, which poses a major problem since even for small scenarios the solution space is typically very large. In this chapter we analyze requirements from mission-critical business processes in the Air Traffic Management (ATM) domain and introduce an approach for semi-automatic semantic matchmaking for software services, the "System-Wide Information Sharing" (SWIS) business process integration framework. A tool-supported semantic matchmaking process like SWIS can provide system designers and integrators with a set of promising software service candidates and therefore strongly reduces the human matching effort by focusing on a much smaller space of matchmaking candidates. We evaluate the feasibility of the SWIS approach in an industry use case from the ATM domain.

6.1 Introduction

Safety-critical systems and business processes, e.g., in the Air Traffic Management (ATM) domain, have to become more flexible to implement changes due to new business environments (e.g., mergers and acquisitions), new standards and regulations. Typical examples for such business processes are the life-cycle support of flights, consisting of a variety of completely different acitivities such as air surveil-

Thomas Moser
Christian Doppler Laboratory, Software Engineering Integration for Flexible Automation Systems, Vienna University of Technology, Austria
e-mail: thomas.moser@tuwien.ac.at

lance while the flight is airborne as well as the supply of catering goods. A promising approach follows the service-oriented architecture (SOA) paradigm that builds flexible new systems for business processes based on a set of software services provided by system nodes in a network. A key design challenge is the matchmaking of business processes and software services, i.e., finding the software services that a) best meet the requirements of the business processes under consideration and b) can be implemented with the available network capabilities. The solution space is typically large even for small problems and a general semantic solution to enable comprehensive tool support seems infeasible.

To provide a SOA solution for a set of business processes, meaning to identify suitable software services for business processes, designers and system integrators need to overcome 3 integration challenges that build on each other:

1. Technical integration connects networked systems that use heterogeneous technologies, i.e., different protocols, operational platforms, etc. Current technical integration approaches like Enterprise Service Bus (ESB) [6] or Service Oriented Architecture (SOA) [31] need manual configuration on the technical detail level and tool support is typically focused on a single technology or vendor.
2. Semantic integration translates data content and format between systems that use heterogeneous semantics, i.e., different terminologies for service names, data formats, etc. For semantic integration, there is no standard or framework available, making the semantic transformations between multiple services inefficient and expensive.
3. Business process support builds on technically and semantically integrated systems that provide software services the business process needs to fulfil its goal. The system integrator has to select software services that really match the requirements of the business process, and check whether the infrastructure capabilities can support the communication requirements of the chosen solution.

Large business process and software service integration networks consist of hundreds of integration nodes; changes of software service properties and network capabilities make the correct and efficient identification of feasible business process and software service pairs a recurring complex and error-prone task. Current service matchmaking approaches focus on either technical or semantic integration issues [48], while business process support is, to our knowledge, missing. Tool support for matchmaking of business processes and software services need to make the requirements of business processes and software services as well as the capabilities of software services and the underlying infrastructure understandable for machines.

In previous work, we introduced a systems integration approach, the ”system-wide information sharing” (SWIS) approach (see Figure 6.1). The SWIS framework explicitly models the semantics of integration requirements and capabilities using a machine-understandable notation (semantic integration) [35]; and the connectors and transformations between heterogeneous legacy systems (technical integration) to simplify systems integration (business process support) [34].

In this chapter, we describe the semantic matchmaking of business processes and software services and the optimization of the integration solution with re-

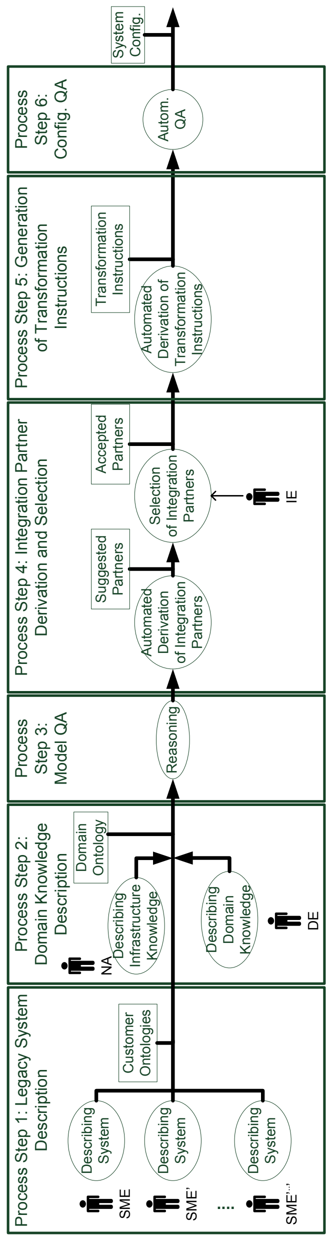

Process Step 1: Legacy System Description
SME
SME'
....
SME''
Describing System
Describing System
Describing System
Customer Ontologies
Process Step 2: Domain Knowledge Description
NA
Describing Infrastructure Knowledge
Describing Domain Knowledge
DE
Domain Ontology
Process Step 3: Model QA
Reasoning
Process Step 4: Integration Partner Derivation and Selection
Automated Derivation of Integration Partners
Suggested Partners
Selection of Integration Partners
IE
Accepted Partners
Process Step 5: Generation of Transformation Instructions
Automated Derivation of Transformation Instructions
Transformation Instructions
Process Step 6: Config. QA
Autom. QA
System Config.

Fig. 6.1 Overview SWIS systems integration approach.

spect to available network capabilities. Semantic matchmaking uses the machine-understandable SWIS models to describe business process and software service requirements and software service and network capabilities to derive 2 results: 1. Provide sets of possible software services for each business process; 2. Optimize the set of selected software services with multiple objectives (e.g., costs, delay) while observing the capabilities of the underlying network infrastructure, a variation of the knapsack problem [27]. We evaluate the feasibility of the SWIS approach in a use case from the ATM domain.

The remainder of this chapter is structured as follows: Section 6.2 summarizes related work, Section 6.3 motivates the research issues, while Section 6.4 describes the use case. Section 6.5 elaborates the semantic service matchmaking approach and the optimization of the integration solution. Section 6.6 evaluates the approach using a case study and discusses the results with regard to the research issues. Finally, Section 6.7 concludes and identifies further work.

6.2 Related Work

This section summarizes related work on technical integration, semantic integration with semantic web services, and service matchmaking with multi-objective optimization.

6.2.1 Technical Integration

Technical system integration is the task to combine networked systems that use heterogeneous technologies to appear as one big system. There are several levels at which system integration could be performed [2], but there is so far no standardized integration process that explains how to integrate systems in general.

System integration can require changes [21] in the actual business policy of a company not only due to the emerging communication needs between multiple computer systems but also due to the communication requirements which have to be established between business units. Therefore, integration can have strong implications on the company as improper integration solutions can lead to considerable inefficiency. Another integration challenge is to keep sufficient control over the involved applications as in most cases integration developers have only limited control over these applications, e.g., legacy systems.

Typical integration solutions focus only on either the semantic heterogeneity (i.e., heterogeneity on service level, e.g., heterogeneous service descriptions, message data fields or service policies) or technical heterogeneity (i.e., heterogeneity on network level, e.g., different technologies or protocols). There is, to our knowledge, no existing approach or concept that takes into consideration both technical and semantic heterogeneities at the same time. In order to cope with technological heterogeneity on service level a homogeneous middleware technology approach [16] could be used for syntactical transformation between services, while the semantic heterogeneity of services could be addressed by means of a common data schema [19]. Heterogeneity on network level may be addressed by using so called adapters transforming messages between each used combination of middleware technologies. However, in order to provide an effective continuous integration solution in this environment, both integration levels (i.e. service and network level) need to be addressed in a mutual way.

The derived limitations for such kinds of integration approaches are on the one hand the need for a common data schema [19], which is often a hard and time consuming procedure, if not even impossible in integration scenarios with several different stakeholders. On the other hand, the need for 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 [6] into: Hub and spoke (EAI brokers) vs. distributed integration, and coupled vs. separated application and integration logic. In the following, using current technology concepts for each category a brief discussion about their advantages and disadvantages with respect to the described scenario is given.

Application servers [16] are capable of interoperating through standardized protocols, however tind to tightly couple together integration logic and application logic. Additionally, as the name suggests a server based architecture style is used for integration and as such has proven to be inconvenient for the scenario. Traditional EAI brokers [6], some of them built upon application servers, use a hub-and-spoke architecture. This approach on the one hand has the benefit of centralized functions such as the management of business rules or routing knowledge, but on the other hand does not scale well across business unit or departmental boundaries, although it offers clear separations between application, integration and routing logic. Message-oriented Middleware [41] is capable of connecting application in a loosely coupled manner but requires low-level application coding intertwining integration and application logic. The resulting effort and complexity of implementing an integration platform with the support for any kind of existing middleware technologies and protocols therefore is considerably high. To enable transparent service integration, the Enterprise Service Bus (ESB) provides the infrastructure services for message exchange and routing as the infrastructure for Service Oriented Architecture (SOA) [31]. 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. Routing may be [6] itinerary-based, content-based, conditional-based defined manually [45] or dynamic [51]. In both cases the drawback is the minimal support for considering all functional and non-functional requirements of all service connections in the system. Dynamic configuration focuses mainly on creating a route for a special business case. Using manual configuration, a system integrator has to rely on his expertise, thus the high number of service interactions may get complex and the configuration error-prone. This may lead to routes that are configured in a way in which their influence on other business interactions is not fully known. As a consequence, business interactions may mutually violate their non-functional business requirements, such as message delivery within a specific time frame otherwise the message may be still useful but not up-to-date any more. Additionally, dynamic configuration may not cope with e.g. node failures fast enough due to missing routing alternatives, therefore possibly violating the same type of non-functional business service requirements.

6.2.2 Semantic Integration with Semantic Web Services

Semantic Integration is defined as the solving of problems originating from the intent to share data across disparate and semantically heterogeneous data [19]. These problems include the matching of ontologies or schemas, the detection of duplicate entries, the reconciliation of inconsistencies, and the modelling of complex relations in different sources [38]. Over the last years, semantic integration became increasingly crucial to a variety of information-processing applications and has received much attention in the web, database, data-mining and AI communities. 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 [11].

Doan and Halevy [10] summarize the research on semantic integration in the database community. There, the matching of two database schemas requires deciding if any two elements of both schemas match, meaning that they refer to the same real-world concept. Typical challenges include the efficient extraction of semantic information, unreliable clues for matching schema elements (e.g., element names, types, data values, schema structures and integrity constraints), incomplete schema and data clues, and subjective matching depending on the application. Rule-based matching techniques use hand-crafted and/or probabilistic rules to exploit schema information for the identification of mappings. Rule-based matching techniques are relatively inexpensive and fairly fast since the typically operate only on schemas and not on data instances. But this is also their main drawback, as they cannot exploit data instances effectively, even though the instances can encode a wealth of information. Additionally, in many cases effective matching rules are simply too difficult to hand craft. Learning-based matching techniques consider a variety of machine learning techniques to exploit both schema and data information. There is also a growing realization that schema- and data-related evidence in two schemas being matched often is inadequate for the matching process, leading to the inclusion of external evidences beyond the two current schemas to the matching process. The key idea here is that a matching tool must be able to learn from past matches. Goh [18] identified 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. Wache et al. [49] reviewed a set of ontology-based approaches and architectures that have been proposed in the context of data integration and interoperability.

Noy [37] 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. Two of the ontologies that are built specifically with the purpose of being formal top-level ontologies are the Suggested Upper Merged Ontology (SUMO) [36] and DOLCE [17]. 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 [5], [9]. The declarative formal representation of mappings is facilitated by the higher expressive power of ontology languages which provide the opportunity to represent mappings themselves in more expressive terms. There exists a large spectrum of how mappings are represented. Bridging axioms relate classes and properties of the two source ontologies and can be seen as translation rules referring to the concepts of source ontologies and e.g., specifying how to express a class in one ontology by collecting information from classes in another ontology. Another mapping representation is the declarative representation of mappings as instances in an ontology. This ontology can then be used by tools to perform the needed transformations. Then a mapping between two ontologies constitutes a set of instances of classes in the mapping ontology and can be used by applications to translate data from the source ontology to the target. 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.

Rosenthal et al. [44] extend the concept of semantic integration to semantics management, which has the goals of easing data sharing for both new and old systems, of ensuring that needed data is actually collected, and of maximizing over time the business value of an enterprise's information systems. To reach these goals, new areas of useful semantic agreements need to be produced proactively, helping enterprises to satisfy new requirements and also reducing costs by reducing unneeded semantic and representation diversities. Additionally, not only the needs of technology-savvy system integrators need to be considered, but also other roles (e.g., enterprise owners, architects, end users and developers) need assistance to have a greater shared understanding of what the data means. Finally, the definition of ”semantics” need to be broadened, to describe what data instances are collected and desired (as in publish/subscribe systems), not just concept definitions and relationships.

Uschold and Gruninger [47] identified four main categories of ontology application to provide a shared and common understanding of a domain that can be communicated between people and application systems [14]: Given the vast number of non-interoperable tools and formats, a given company or organization can benefit greatly by developing their own neutral ontology for authoring, and then developing translators from this ontology to the terminology required by the various target systems. To ensure no loss in translation, the neutral ontology must include only those features that are supported in all of the target systems. The trade-off here is loss of functionality of some of the tools; since certain special features may not be usable. While it is safe to assume there will not be global ontologies and formats agreed by one and all, it is nevertheless possible to create an ontology to be used as a neutral interchange format for translating among various formats. This avoids the need to create and maintain O(N2) translators and it makes it easier for new systems and formats to be introduced into an existing environment. In practical terms, this can result in dramatic savings in maintenance costs - it has been estimated that 95% of the costs of enterprise integration projects is maintenance [42].

There is a growing interest in the idea of Ontology-Driven Software Engineering, in which an ontology of a given domain is created and used as a basis for specification and development of some software. The benefits of ontology-based specification are best seen when there is a formal link between the ontology and the software. This is the approach of Model-Driven Architecture (MDA) [32] created and promoted by the Object Modeling Group (OMG) as well as ontology software which automatically creates Java classes and Java Documents from an ontology. A large variety of applications may use the access functions of the ontology. Not only does this ensure greater interoperation, but it also offers significant cost reduction for software evolution and maintenance. A suite of software tools all based on a single core ontology are semantically integrated for free, eliminating the need to develop translators. To facilitate search, an ontology is used as a structuring device for an information repository (e.g., documents, web pages, names of experts); this supports the organization and classification of repositories of information at a higher level of abstraction than is commonly used today Using ontologies to structure information repositories also entails the use of semantic indexing techniques, or adding semantic annotations to the documents themselves. If different repositories are indexed to different ontologies, then a semantically integrated information access system could deploy mappings between different ontologies and retrieve answers from multiple repositories.

The promise of Web Services and the need for widely accepted standards enabling them are by now well recognized, and considerable efforts are underway to define and evolve such standards in the commercial realm. In particular, the Web Services Description Language (WSDL) [8] is already well established as an essential building block in the evolving stack of Web Service technologies, allowing the specification of the syntax of the input and output messages of a basic service, as well as of other details needed for the invocation of the service. WSDL does not, however, support the specification of workflows composed of basic services. In this area, the Business Process Execution Language for Web Services (BPEL4WS)

[22], has the most prominent status. With respect to registering Web services, for purposes of advertising and discovery, Universal Description, Discovery and Integration (UDDI) [4] has received the most attention to date.

At the same time, recognition is growing of the need for richer semantic specifications of Web Services, so as to enable fuller, more flexible automation of service provision and use, support the construction of more powerful tools and methodologies, and promote the use of semantically well-founded reasoning about services. Because a rich representation language permits a more comprehensive specification of so many different aspects of services, they can provide a better foundation for a broad range of activities, across the Web service lifecycle. Furthermore, richer semantics can help to provide fuller automation of activities as verification, simulation, configuration, supply chain management, contracting, and negotiation of services [29].

To meet this need, researchers have been developing languages, architectures and related approaches for so called Semantic Web services [30]. The Ontology Web Language for Services (OWL-S) [28], which seeks to provide the building blocks for encoding rich semantic service descriptions in a way that builds naturally upon OWL [3], the Semantic Web language, supplies Web Service providers with a core set of markup language constructs for describing the properties and capabilities of their Web Services in unambiguous, computer-interpretable form. OWL-S markup of Web Services facilitates the automation of Web Service tasks, including automated Web Service discovery, execution, composition and interoperation.

WSDL-S [33] is another approach for annotating current Web Service standards with semantic descriptions. In WSDL-S, the expressivity of WSDL is enriched with semantics by employing concepts similar to those in OWL-S while being agnostic to the semantic representation language. The advantage of this approach to adding semantics to WSDL is multi-fold. First, users can, in an upwardly compatible way, describe both the semantics and operation level details in WSDL- a language that the developer community is familiar with. Second, by externalizing the semantic domain models, a language-agnostic approach to ontology representation is taken. This allows Web service developers to annotate their Web services with their choice of modelling language (such as OWL, or legacy models developed in UML or other knowledge representation languages). This is significant because the ability to reuse existing domain models expressed in modelling languages like UML can greatly alleviate the need to separately model semantics. Finally, it is relatively easy to update the existing tooling around WSDL specification to accommodate our incremental approach. Moreover, the externalization of the semantic domain models still allows for richer representations of domain concepts and relationships in languages such as OWL, thereby bringing together the best of both worlds. Use of expressive mapping representation and techniques can further enable this approach to deal with significant types of syntactic, structural, representational and semantic heterogeneity [1].

The Web Service Modeling Ontology (WSMO) [25] is a framework for Semantic Web Services which refines and extends the Web Service Modeling Framework (WSMF) [15] to a meta-ontology for Semantic Web services. WSMF defines a rich conceptual model for the development and the description of Web Services based on two main requirements: maximal decoupling and strong mediation. WSMO is accompanied by a formal language, the Web Service Modeling Language (WSML) that allows annotating Web Services according to the conceptual model. Also an execution environment (WSMX) [20] for the dynamic discovery, selection, mediation, invocation, and inter-operation of Semantic Web services based on the WSMO specification is included [13].

All three approaches, OWL-S, WSDL-S and WSMO, provide mechanism for semantically describing Web Services, with the major goal of allowing generic description of service functionality as well adding semantics to general service descriptions like provided/consumed messages or service bindings. This ambitious goal seems very useful for generic service descriptions; however its usage is limited in specific domains like in the ATM domain, since too specific features would complicate a generic approach too much. Therefore, we defined our own ontology-based architecture for describing the properties and features of the ATM services [34].

6.2.3 Service Matchmaking Approaches

Semantic matchmaking can be seen as major feature of semantic integration which supports designers and system integrators by providing sets of possible integration partners regarding both structural and semantic attributes. However, the relevant semantic concepts are hard to define unambiguously for general domains, thus the focus on a well-defined domain like ATM provides semantic clarity. Software components discovery and Web Service discovery can be classified into two categories: signature matching and semantic matching.

Purtilo and Atlee [43] propose a signature-matching approach by specifying the invocation parameters. Zaremski and Wing [52] describe exact and relaxed signature matching as a means for retrieving functions and modules from a software library. Wang and Stroulia [50] provide a structure-matching-based signature matching for Web Service discovery. Signature matching is an efficient means for software components retrieval, but two software components with similar signatures may have completely different behaviors.

Semantic matching addresses this problem by comparing software components based on formal descriptions of the semantics of their behaviors. Zaremski and Wing [53] extend their signature-matching work with a specification-matching scheme. Cho et al. [7] use a protocol to specify interoperability of objects. Semantic matching identifies suitable services more precisely than signature-matching methods, but the cost of formally defining provided and required services is considerable. Paolucci et al. [40] propose a DAML-S based approach for a declarative description of web services outside the representation capabilities of UDDI and WSDL. They provide an upper-level ontology of service profiles consisting of service actors, functional service attributes, and function service descriptions. Trastour et al. [46] define a set of requirements needed for service matchmaking based on Semantic Web techniques and evaluate a set of standard approaches (e.g., UDDI, ebXML) using these requirements. The potential complexity of the service descriptions, like attribute-value pairs or nested tree/graph style structures, requires a flexible and expressive metadata model. In order to support under-specified data structures like incomplete service advertisements, an approach needs to be able to express semi-structured data. Additionally, support for types and subsumption is needed to be able to work at different levels of generality. Finally, constraints need to be expressed to define and check the acceptable instances for service invocation.

Li and Horrocks [26] investigate how Semantic and Web Services technologies can be used to support service advertisement and discovery in e-Commerce. They describe the design and implementation of a service matchmaking prototype which uses a DAML-S based ontology and a Description Logic reasoner to compare ontology based service descriptions. By representing the semantics of service descriptions, the matchmaker enables to locate suitable web services automatically. The approach is evaluated using a realistic agent based e-commerce scenario. Although the initial classification of large numbers of service descriptions could be quite time consuming, subsequent matching of queries could be performed very efficiently.

Kolovski et al. [23] provide a mapping of WS-Policy to OWL. WS-Policy provides a general purpose model and syntax to describe the policies of a Web service. It specifies a base set of constructs that can be used and extended by other Web service specifications to describe a broad range of service requirements and capabilities. The main advantage of representing Web Service policies using OWL is that OWL is much more expressive than WS-Policy and thus provides a framework for exploring richer policy languages. Verma et al. [48] present an approach for matching the non-functional properties of Web Services represented using WS-Policy. Oldham et al. [39] present a framework for the matching of providers and consumers based on WS-Agreements. The WS-Agreement specification defines a language and protocol for capturing relationships with agreements between two parties.

Both WS-Policy and WS-Agreement define a generic framework for the representation of standard Web Service policies, however both frameworks seem too generic to be effectively used in a concrete scenario from a specialized domain like the ATM domain is. Therefore, we used the concept of describing Service policies using a knowledge representation language like OWL, but defined our own extendable policy representation language which is better suitable for the ATM domain [34].

6.3 Research Issues

Recent projects with industry partners from the ATM domain raised the need for semi-automated business process integration support in technology-driven integration environments. Recently, we developed a data-driven approach [16] that explicitly models the semantics of the problem space, i.e., business process integration requirements and network infrastructure capabilities [17]; the solution space, i.e., the connectors, and data transformations between software services. Finally, in this chapter we provide a process to bridge problem and solution spaces, i.e., identify feasible business process and software services pairs while fulfilling business requirements and optimizing the chosen integration solution according to multiple objectives. Figure 6.2 provides an overview on the integration layers, data flows between the integration layers, and the steps of the semantic service matchmaking process:

- SM1: For each business process, identify the suitable software services sets, which fulfil all business process service and data requirements. From these possible business process and software services sets, the system integrators choose the most promising sets, the so-called collaboration sets.
- SM2: The selected collaboration sets are then optimized regarding the original infrastructure requirements of both the business processs and the software services, as well as the available limited capabilities of the infrastructure's nodes and links. The outcome of SM2 is an optimized configuration of the integration solution, consisting of the selected collaboration sets as well as their grounding to the underlying integration network infrastructure.

Based on this, we derive the following research issues:

RI-1: Semantic Matchmaking of software service candidates for one business process (SM1). Provide machine-understandable descriptions for business process and software services requirements as well as software service and network capabilities to provide tool support for SM1 to make the search space reduction effective (low number of false negatives and false positives) and efficient (less human effort required) compared to the current human-based approach.

RI-2: Resource Feasibility Check and Optimization for all Collaborations (SM2). Provide a framework to enable a) checking the validity of a set of business processs and software services with the infra-structure capability constraints and b) ranking valid solutions by multiple optimization criteria like network cost and service delay.

6.4 ATM Scenario Description

This section describes the integration scenario from the ATM domain used throughout this chapter. The ATM use case (see Figure 6.3) represents information that is typically extracted from customers/participants in workshops on requirements and capabilities elicitation for information systems in the aviation domain. In safety-critical domains like ATM business process integration solutions have to pass certifications before deployment, which typical dynamic SOA solutions [6][31] cannot fulfill regarding the current rigid integration network in the ATM domain designed to guarantee integration requirements even in case of partial failure.

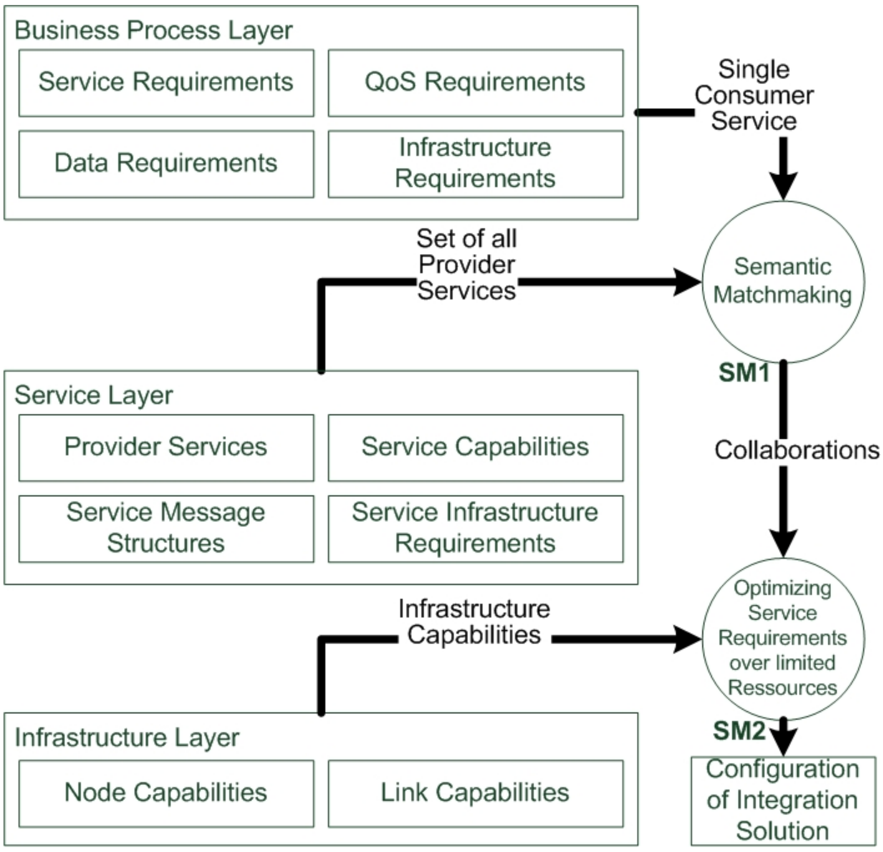


Fig. 6.2 Semantic Service Matchmaking Process Steps.

In the ATM domain semantic matchmaking is an effort for scarce human experts who have to cope with a huge search space and often miss better solutions due to their simple heuristic search strategies. Tool-supported semantic matchmaking provides designers and system integrators with a set of promising integration partner candidates and therefore strongly reduces the human matching effort by focusing on a much smaller space of feasible matchmaking candidates that can be rated according to relevant optimization criteria.

As shown in Figure 6.3, the integration network consists of business services connected to integration network nodes. Between these nodes, there may exist different kinds of network links using different transmission technologies (e.g., radio or wired transmission) as well as different middleware technologies for communication purposes. The capabilities of nodes and links, like throughput, availability, reliability, or security are explicitly modeled in order to be capable of selecting suitable communication paths for particular service requirements, e.g., the communication link

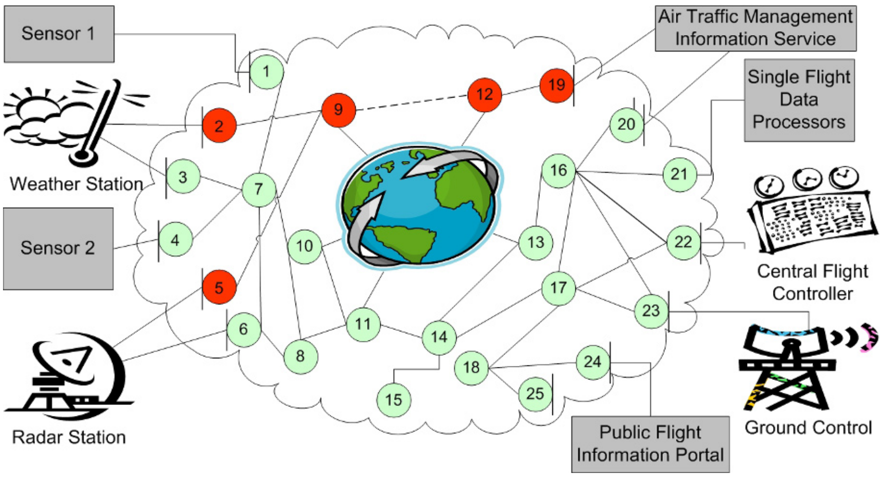

Sensor 1
Air Traffic Management Information Service
Single Flight Data Processors
Weather Station
Sensor 2
Central Flight Controller
Ground Control
Radar Station
Public Flight Information Portal
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Fig. 6.3 A Typical ATM Domain Integration Network.

between the red ATMIS Node and the red Node 12 represents a reliable and secured communication path which may be requested by e.g., the ATMIS business service.

In the Air Traffic Management domain complex information systems need to cooperate to provide data analysis and planning services, which consist in the core of safety-critical Air Traffic Management services and also added-value services for related businesses. Air Traffic Management is a relevant and dynamic business segment with changing business processes that need to be reflected in the integration of the underlying information and technical systems.

A major integration challenge is to explicitly model the knowledge embedded in systems and Air Traffic Management experts to provide a machine-understandable knowledge model for integration requirements between a set of complex information systems. Complex information systems consist of a large number of heterogeneous subsystems. Each of these subsystems may have different data types as well as heterogeneous system architectures. In addition, complex information systems typically have significant quality-of-service demands, e.g., regarding security, reliability, timing, and availability. Many of today's Air Traffic Management complex information systems were developed independently for targeted business needs, but when the business needs changed, these systems needed to be integrated into other parts of the organization [19]. Most of the system knowledge is still represented implicitly, either known by experts or described in human-only-readable sources, resulting in very limited tool support for systems integration. The process of adapting the cooperation the business system is traditionally a human-intensive approach of experts from the Air Traffic Management and technology domains).

Making the implicit expert knowledge explicit (see Figure 6.4) and understandable for machines can greatly facilitate tool support for systems integrators and engineers by providing automation for technical integration steps and automatic validation of integration solution candidates. Consequently, we employ the EKB framework as a data-driven approach that explicitly models the semantics of the problem space, i.e., integration requirements and capabilities; the solution space, i.e., the connectors, and data transformations between heterogeneous legacy systems; and finally provide a process to bridge problem and solution spaces, i.e., find out whether there are feasible solutions and minimize the cost of integration.

6.5 Semantic Service Matchmaking Approach

This section describes the semantic service matchmaking approach as well as the multi-objective optimization of the chosen integration services candidates.

6.5.1 Identification of Possible Collaboration Candidate Sets

The identification of possible collaboration candidate sets is implemented as a heuristic algorithm. Step by step, the possible collaboration candidate sets are reduced by applying the rules described to the possible collaboration candidate sets. The heuristic rules that are applied during the source/sink matching are described in the following paragraphs.

Message mapping. During the description of the software service messages, each software service message segment was mapped to a domain concept, which has been specified in the common domain ontology. Therefore, for all segments of the message required by a certain business process, it is searched for messages of the software services that contain segments, which are mapped to the same domain concept, and if possible, to the same message format.

Service Policies. In addition, software services can define requirements (policies) regarding preferred or unwanted software service partners, as well as other non-functional requirements, e.g., QoS requirements regarding the underlying integration network. A policy is a restriction or a condition for a single collaboration or a set of collaborations, in order to allow the communication via the underlying integration network. In SWIS-based applications, there are two kinds of policies. On the one hand, there are policies which are valid for all collaborations. They specify global conditions that need to be fulfilled by all collaborations, e.g., a maximum time for the delivery of messages. On the other hand, there are policies which are required only for a specific subset of collaborations. These policies specify conditions that need to be fulfilled by the collaborations containing particular software services, e.g., the communication has to use only secure links, or only a specified set of other software services is allowed to participate in the collaboration. The software service policies that regard other software services are evaluated by checking whether the attributes and tags of every software service of the particular collabora-

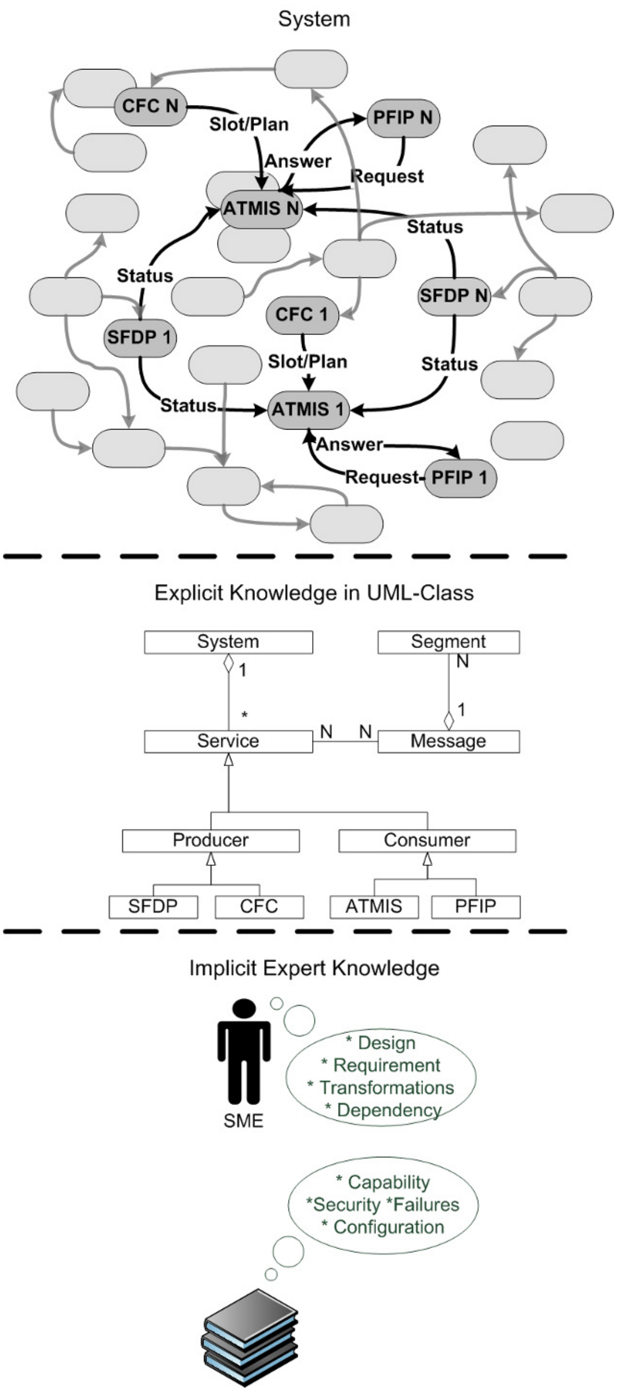


Fig. 6.4 Explicit and Implicit ATM Expert Knowledge [34].

tion candidate meet the service policies defined by the business process.

Format Translation. If a message segment is mapped to the same domain concept as the required message segment, but the formats of the two segments differ, check whether there is a converter defined for the two formats. A converter is used to convert the format of message segments from one basic data type to a different one. An explicit identifier is defined to allow the search for the converter at runtime (e.g., by using Java Reflection).

External Service Transformation. If the message segments differ in the domain concept they are mapped to, check if a service exists that consumes a segment mapped to the same domain concept as the segment of the message of the software service and provides a message with a segment mapped to the same domain concept of the segment of the message of the business process.

Route Deduction. As last rule it is checked whether there is an existing route between the nodes connecting the software services and the node connecting the business process.

If all the rules mentioned above are successfully applied to a set of one or more software services and a business process, then the particular set is accepted as collaboration candidate. If any of the rules cannot be met, the particular set is discarded as collaboration candidate.

6.5.2 Validity-Check and Optimization of Collaborations

Once all collaborations have been specified a Scenario is derived. A Scenario contains beside all collaborations a specification detailing how to configure the network infrastructure, so that the integration solution is optimized according to the given objectives. In the following the process steps needed to optimize the scenario is explained.

Preliminary Checks. The process step checks whether there is at least one single network route for each collaboration satisfying all global and collaboration specific policies. If this step cannot be completely satisfied the process raises an exception. The system integrator either updates or removes the collaborations which cannot be mapped to a network route, and restart the process step, or adapts the semantic infrastructure model, by adding additional nodes and links.

Route Derivation. Once it has been verified that each collaboration can be mapped to at least one route in the network, the process step derives every possible route for each collaboration. The only restrictions are that no node is allowed to appear twice within the same route and all policies have to be satisfied. The valid ones are retained; the ones violating the restrictions are removed. At the end of this process step, each collaboration will have either a single route or a set of valid routes to choose from.

Creating Scenarios. The processing step combines each route of each collaboration with each other. This means that a scenario consists of a set of collaborations where each collaboration represents exactly one route. The more scenarios are created, the higher the probability to find a scenario that is well suited for achieving the stated optimization objectives.

Evaluation. The process iterates through all scenarios and calculates their fitness according to the optimization objectives. The fitness of a scenario is the fitness of all its containing collaborations, and represents the real values (e.g. the time a message needs and the costs along the chosen route) of the objectives. The fitness represents the trade-off of the configuration, the routes of each collaboration predetermine. The set of fitness values is then analyzed according to the Pareto Front approach [12]. The Pareto Front contains either a single Scenario or a set of Scenarios. In the latter case there may be several ”nearly equivalent” configurations as integration solutions. Thus, the system integrator has to decide which one to pick for practical deployment.

Multi-Objective Optimization. We have accomplished the process of optimizing collaborations by implementing a Java version of the mPOEMS approach into the SWIS framework. mPoems is an evolutionary algorithm using the concept of dominance for multi-objective optimization. The results and explanations of the approach can be found at [24].

6.6 Case Study

In this section, we evaluate the SWIS framework using a clear and comprehensible example to show the principles of our approach. In addition, we discuss the results with regard to the identified research issues.

An example for semantic service matchmaking in the SWIS framework is shown in Figure 6.5. There are three services of provided by legacy systems, two provider services (ATMIS and SFDP) and one consumer service (PFIP). The consumer service needs information that can be obtained from the provider services, i.e. FlightID, Departure, Destination and FlightStatus. This needed information is provided separately by the two provider services, so the system has to find the suitable information that match with the consumer service's needs. Additionally, the service ATMIS_TransReqs defines a policy for the underlying integration network, stating that only secure network links may be used for the communication.

From the domain knowledge description, we know that Flight ID is a synonym for Flight Number, that Departure and Arrival are combinations of the airport code and country code of departure/arrival, and that the FlightStatus arrived or departed, can be derived by checking the occurrence of either TimeOfArrival or TimeOfDeparture.

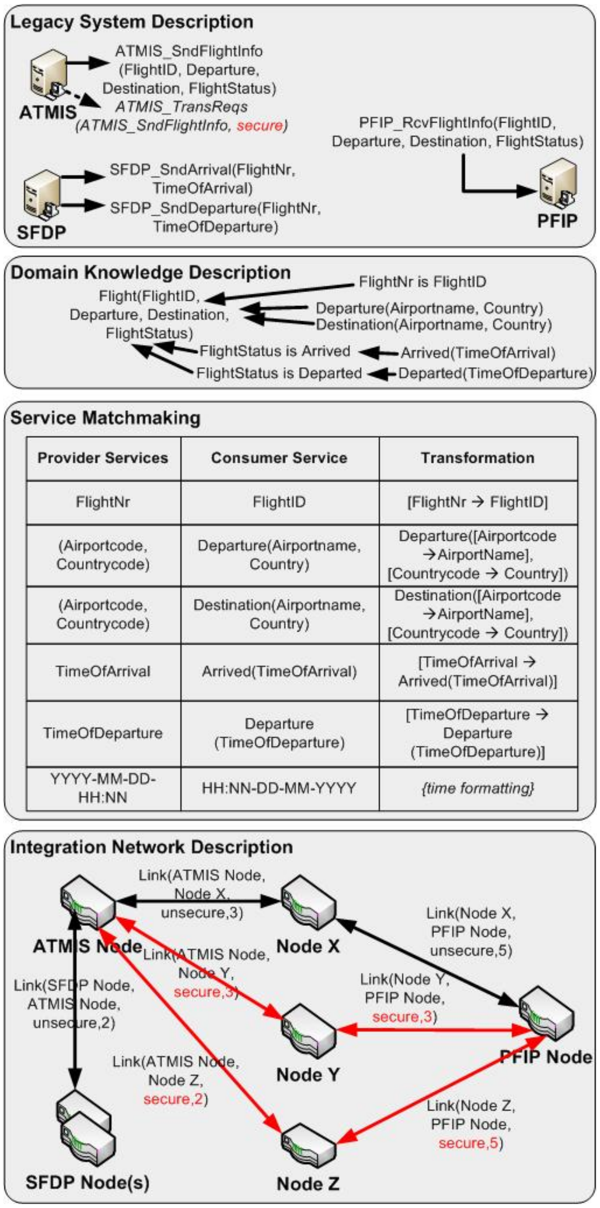
Next, we calculate the network resources needed for sending messages from the SFDP Node to the PFIP Node with less capacity. From the integration network description, we can see several nodes connected by links. Each link contains information regarding source node and target node, support for secure transmissions and the transmission delay. The communication between ATMIS to PFIP needs to be done using secure connections only. There are two possible connections, either via Node Y or via Node Z. The system will choose connection via Node Y because it has less delay (6) than connection via Node Z (7).

6.6.1 Discussion

The example shows that even for small problems the solution space is typically large. However, large business process and software service integration networks consist of hundreds of integration nodes; and changes of software service properties and network capabilities make the correct and efficient identification of feasible business process and software service pairs a recurring complex and error-prone task. By providing only sets of feasible/promising service provider and consumer candidates, semantic matchmaking supports designers and system integrators by providing sets of possible integration partners regarding both structural and semantic attributes. However, the relevant semantic concepts are hard to define unambiguously for general domains, thus the focus on a well-defined domain like ATM provides semantic clarity.

We used the concept of describing Service policies using a knowledge representation language like OWL, but defined our own extendable policy representation language which is better suitable for the ATM domain. We do not use standardized Web Service description frameworks because, since the strengths of Web Service description frameworks lies in the generality of the approach, however their weakness is that it may become complicated to describe domain-specific issues. For specific domains, it may be useful to use the principles of web service descriptions but tailor them to the domain. Additionally, we defined our own ontology-based architecture for describing the properties and features of the ATM services.

We have developed a data-driven approach [34] that explicitly models the semantics of the problem space, i.e., business process integration requirements and network infrastructure capabilities [35]; the solution space, i.e., the connectors, and data transformations between software services. In this chapter, we described a process to bridge problem and solution spaces, i.e., identify feasible business process and software services pairs while fulfilling business requirements and optimizing the chosen integration solution according to multiple objectives. In order to evaluate the proposed process, we have derived two major research issues that will be discussed in the following paragraphs.


Legacy System Description
ATMIS_SndFlightInfo (FlightID, Departure, Destination, FlightStatus)
ATMIS
ATMIS_TransReqs (ATMIS_SndFlightInfo, secure)
PFIP_RcvFlightInfo(FlightID, Departure, Destination, FlightStatus)
SFDP_SndArrival(FlightNr, TimeOfArrival)
SFDP_SndDeparture(FlightNr, TimeOfDeparture)
SFDP
PFIP
Domain Knowledge Description
FlightNr is FlightID
Flight(FlightID, Departure, Destination, FlightStatus)
Departure(Airportname, Country)
Destination(Airportname, Country)
FlightStatus is Arrived
Arrived(TimeOfArrival)
FlightStatus is Departed
Departed(TimeOfDeparture)
Service Matchmaking
Integration Network Description
Link(ATMIS Node, Node X, unsecure,3)
ATMIS Node
Node X
Link(ATMIS Node, Node Y, secure,3)
Link(Node X, PFIP Node, unsecure,5)
Link(SFDP Node, ATMIS Node, unsecure,2)
Link(Node Y, PFIP Node, secure,3)
Node Y
PFIP Node
Link(ATMIS Node, Node Z, secure,2)
Link(Node Z, PFIP Node, secure,5)
SFDP Node(s)
Node Z

Provider Services	Consumer Service	Transformation
FlightNr	FlightID	[FlightNr → FlightID]
(Airportcode, Countrycode)	Departure(Airportname, Country)	Departure([Airportcode →AirportName], [Countrycode → Country])
(Airportcode, Countrycode)	Destination(Airportname, Country)	Destination([Airportcode →AirportName], [Countrycode → Country])
TimeOfArrival	Arrived(TimeOfArrival)	[TimeOfArrival → Arrived(TimeOfArrival)]
TimeOfDeparture	Departure (TimeOfDeparture)	[TimeOfDeparture → Departure (TimeOfDeparture)]
YYYY-MM-DD-HH:NN	HH:NN-DD-MM-YYYY	*{time formatting}*

Fig. 6.5 Service Matchmaking Example.

Semantic Matchmaking of software service candidates for one business process. Current service matchmaking approaches focus on either technical or semantic integration issues [48], while business process support is, to our knowledge, missing. In the SWIS framework, we presented a combined service matchmaking approach that performs matching based on the data of the services and available service policies regarding other services. The SWIS framework's semantic service matchmaking enables an effective search space reduction and poses lower risk and effort compared to the current human-based approaches.

Resource Feasibility Check and Optimization for all Collaborations. The optimization process steps allow using existing resources efficiently. Out of all possible collaborations for a single business process which are creatable by means of the proposed semantic matchmaking approach, only those are desirable to be deployed in the integration solution which fulfills certain criteria. Those criteria are set up by the integration expert so that existing collaborations use the underlying integration network infrastructure with its limited resources as efficient as possible.

6.7 Conclusion

In this chapter we presented an approach for semi-automatic semantic matchmaking for software services, the ”System-Wide Information Sharing” (SWIS) Business Process integration framework. The SWIS Business Process integration frameworks uses the machine-understandable SWIS models to describe business process and software service requirements as well as software service and network capabilities to provide sets of possible software services for each business process. Out of these possible sets, the system integrators choose the wanted sets. These wanted sets are then optimized with multiple objectives (e.g., costs, delay) while observing the capabilities of the underlying network infrastructure.

We evaluated the feasibility of the SWIS approach in an industry use case from the Air Traffic Management (ATM) domain. The example shows that even for small problems the solution space is typically large, and even bigger for large business process and software service integration networks consisting of hundreds of integration nodes. A tool-supported semantic matchmaking process like SWIS can provide system designers and integrators with a set of promising Software Service candidates and therefore strongly reduces the human matching effort by focusing on a much smaller space of matchmaking candidates.

Major contributions of this chapter are

- Large Business Process and Software Service integration networks consist of hundreds of integration nodes; and changes of Software Service properties and network capabilities make the correct and efficient identification of feasible Business Process and Software Service pairs a recurring complex and error-prone task. By providing only sets of feasible/promising service provider and consumer candidates, semantic matchmaking supports designers and system integrators by providing sets of possible integration partners regarding both structural and semantic attributes. However, the relevant semantic concepts are hard to define unambiguously for general domains, thus the focus on a well-defined domain like ATM provides semantic clarity.

- We used the concept of describing Service policies using a knowledge representation language like OWL, but defined our own extendable policy representation language which is better suitable for the ATM domain. We do not use standardized Web Service description frameworks because, since the strengths of Web Service description frameworks lies in the generality of the approach, however their weakness is that it may become complicated to describe domain-specific issues. For specific domains, it may be useful to use the principles of web service descriptions but tailor them to the domain. Additionally, we defined our own ontology-based architecture for describing the properties and features of the ATM services.
- Current service matchmaking approaches focus on either technical or semantic integration issues, while business process support is, to our knowledge, missing. In this chapter, we presented a combined service matchmaking approach that performs matching based on the data of the services and available service policies regarding other services. Semantic service matchmaking enables an effective search space reduction and poses lower risk and effort compared to the current human-based approaches.
- The optimization process steps allow using existing resources efficiently. Out of all possible collaborations for a single Business Process which are creatable by means of the proposed semantic matchmaking approach, only those are desirable to be deployed in the integration solution which fulfills certain criteria. Those criteria are set up by the integration expert so that existing collaborations use the underlying integration network infrastructure with its limited resources as efficient as possible

Further Work. Further work will include a detailed description of the semantic design to translate between matched services and an evaluation measuring the effectiveness and efficiency of deriving the semantic transformation with tool-support compared to a manual approach.

Acknowledgements The authors would like to acknowledge all project members of the SWIS (System-Wide Information Sharing) project performed from 2006-2008 at Vienna University of Technology together with Frequentis AG and Austro Control GmbH. This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. In addition, this work has been partially funded by the Vienna University of Technology, in the Complex Systems Design & Engineering Lab.

References

1. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.T., Sheth, A., Verma, K.: Web service semantics-wsdl-s. W3C Member Submission **7** (2005)
2. Balasubramanian, K., Gokhale, A., Karsai, G., Sztipanovits, J., Neema, S.: Developing applications using model-driven design environments. COMPUTER pp. 33–40 (2006)
3. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: Owl web ontology language reference. W3C Recommendation **10** (2004)
4. Bellwood, T., Clement, L., Ehnebuske, D., Hately, A., Hondo, M., Husband, Y.L., Januszewski, K., Lee, S., McKee, B., Munter, J.: Uddi version 3.0. Published specification, Oasis (2002)
5. Bergamaschi, S., Castano, S., Vincini, M.: Semantic integration of semistructured and structured data sources. SIGMOD Rec. **28**(1), 54–59 (1999). 309897
6. Chappel, D.A.: Enterprise Service Bus. O'Reilly Media, Sebastopol, CA (2004)
7. Cho, I.H., McGregor, J.D., Krause, L.: A protocol based approach to specifying interoperability between objects. In: 26th International Conference on Technology of Object-Oriented Languages (TOOLS 26), pp. 84–96 (1998)
8. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web services description language (wsdl) 1.1 (2001)
9. Cruz, I.R., Huiyong, X., Feihong, H.: An ontology-based framework for xml semantic integration. In: International Database Engineering and Applications Symposium (IDEAS '04), pp. 217–226. IEEE (2004)
10. Doan, A., Halevy, A.: Semantic integration research in the database community: A brief survey. AI Magazine **26**(1), 83–94 (2005)
11. Doan, A., Noy, N.F., Halevy, A.Y.: Introduction to the special issue on semantic integration. SIGMOD Rec. **33**(4), 11–13 (2004). 1041412
12. Ehrgott, M.: Multicriteria Optimization. Springer (2005)
13. Feier, C., Roman, D., Polleres, A., Domingue, J., Stollberg, M., Fensel, D.: Towards intelligent web services: The web service modeling ontology (wsmo). In: International Conference on Intelligent Computing (ICIC) (2005). International Conference on Intelligent Computing (ICIC)
14. Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer (2003)
15. Fensel, D., Bussler, C.: The web service modeling framework wsmf. Electronic Commerce Research and Applications **1**(2), 113–137 (2002)
16. Gail, E.H., David, L., Jeromy, C., re, Fred, N., John, C., Martin, N.: Application servers: one size fits all ... not? In: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. ACM, Anaheim, CA, USA (2003). 949414 284-285
17. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A.: Sweetening wordnet with dolce. AI Magazine **24**(4), 13–24 (2003)
18. Goh, C.H.: Representing and reasoning about semantic conflicts in heterogeneous information systems. Ph.D. thesis, MIT (1996)
19. Halevy, A.: Why your data won't mix. Queue **3**(8), 50–58 (2005). 1103836
20. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: Wsmx-a semantic service-oriented architecture. In: International Conference on Web Services (ICWS 2005), pp. 321–328. IEEE (2005). Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on
21. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional (2004)
22. Juric, M.B.: Business Process Execution Language for Web Services BPEL and BPEL4WS 2nd Edition. Packt Publishing (2006)
23. Kolovski, V., Parsia, B., Katz, Y., Hendler, J.: Representing web service policies in owl-dl. In: 4th International Semantic Web Conference (ISWC 2005), pp. 461–475. Springer (2005)

24. Kubalk, J., Mordinyi, R., Biffl, S.: Multiobjective prototype optimization with evolved improvement steps. In: Evolutionary Computation in Combinatorial Optimization (2008)
25. Lausen, H., Polleres, A., Roman, D.: Web service modeling ontology (wsmo). W3C Member Submission **3** (2005)
26. Li, L., Horrocks, I.: A software framework for matchmaking based on semantic web technology. International Journal of Electronic Commerce **8**(4), 39–60 (2004)
27. Martello, S., Toth, P.: Knapsack problems: algorithms and computer implementations. John Wiley & Sons (1990)
28. Martin, D., Ankolekar, A., Burstein, M., Denker, G., Elenius, D., Hobb, J., Kagal, L., Lassila, O., McDermott, D., McGuinness, D., McIlraith, S., Paolucci, M., Parsia, B., Payne, T., Sabou, M., Schlenoff, C., Sirin, E., Solanki, M., Srinivasan, N., Sycara, K., Washington, R.: Owl-s 1.1 release (2004)
29. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M.: Bringing semantics to web services: The owl-s approach. In: First International Workshop on Semantic Web Services and Web Process Composition, pp. 26–42. Springer (2005). Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)
30. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. IEEE Intelligent Systems **16**(2), 46–53 (2001)
31. Mike, P.P., Willem-Jan, H.: Service oriented architectures: approaches, technologies and research issues. The VLDB Journal **16**(3), 389–415 (2007). 1265298
32. Miller, J., Mukerji, J.: Model driven architecture (mda). Object Management Group, Draft Specification ormsc/2001-07-01, July **9** (2001)
33. Miller, J., Verma, K., Rajasekaran, P., Sheth, A., Aggarwal, R., Sivashanmugam, K.: Wsdl-s: Adding semantics to wsdl-white paper (2004)
34. Moser, T., Mordinyi, R., Mikula, A., Biffl, S.: Efficient system integration using semantic requirements and capability models: An approach for integrating heterogeneous business services. In: 11th International Conference on Enterprise Information Systems (ICEIS 2009), pp. 56–63. Milan, Italy (2009)
35. Moser, T., Mordinyi, R., Mikula, A., Biffl, S.: Making expert knowledge explicit to facilitate tool support for integrating complex information systems in the atm domain. In: International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2009), pp. 90–97. Fukuoka, Japan (2009)
36. Niles, I., Pease, A.: Towards a standard upper ontology. In: 2nd International Conference on Formal Ontology in Information Systems, pp. 2–9. ACM (2001). Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001
37. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. SIGMOD Rec. **33**(4), 65–70 (2004). 1041421
38. Noy, N.F., Doan, A.H., Halevy, A.Y.: Semantic integration. AI Magazine **26**(1), 7–10 (2005)
39. Oldham, N., Verma, K., Sheth, A., Hakimpour, F.: Semantic ws-agreement partner selection. In: 15th International World Wide Web Conference, pp. 697–706. ACM, Edinburgh, Scotland (2006). 1135879 697-706
40. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: First International Semantic Web Conference, pp. 333–347. Springer (2002)
41. Piyush, M., Michael, P.: Benchmarking message-oriented middleware: Tibco versus sonicmq: Research articles. Concurr. Comput. : Pract. Exper. **17**(12), 1507–1526 (2005). 1085000
42. Pollock, J.: Integrations dirty little secret: Its a matter of semantics. Whitepaper, Modulant: The Interoperability Company (2002)
43. Purtilo, J.M., Atlee, J.M.: Module reuse by interface adaptation. Software - Practice and Experience **21**(6), 539–556 (1991)
44. Rosenthal, A., Seligman, L., Renner, S.: From semantic integration to semantics management: case studies and a way forward. SIGMOD Rec. **33**(4), 44–50 (2004). 1041418
45. Satoh, F., Nakamura, Y., Mukhi, N.K., Tatsubori, M., Ono, K.: Methodology and tools for end-to-end soa security configurations. In: Services - Part I, 2008. IEEE Congress on, pp. 307–314 (2008)

46. Trastour, D., Bartolini, C., Gonzalez-Castillo, J.: A semantic web approach to service description for matchmaking of services. HP LABORATORIES TECHNICAL REPORT (2001)
47. Uschold, M., Gruninger, M.: Ontologies and semantics for seamless connectivity. SIGMOD Rec. **33**(4), 58–64 (2004). 1041420
48. Verma, K., Akkiraju, R., Goodwin, R.: Semantic matching of web service policies. In: 2nd International Workshop on Semantic and Dynamic Web Process (SDWP 2005) (2005). SDWP Workshop
49. Wache, H., Vgele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hbner, S.: Ontology-based integration of information-a survey of existing approaches. In: Workshop on Ontologies and Information Sharing (IJCAI-01), vol. 2001, pp. 108–117. Seattle, USA (2001)
50. Wang, Y., Stroulia, E.: Flexible interface matching for web-service discovery. In: Fourth International Conference on Web Information Systems Engineering, (WISE 2003), pp. 147–156 (2003)
51. Xiaoying, B., Jihui, X., Bin, C., Sinan, X.: Dresr: Dynamic routing in enterprise service bus. In: e-Business Engineering, 2007. ICEBE 2007. IEEE International Conference on, pp. 528–531 (2007)
52. Zaremski, A.M., Wing, J.M.: Signature matching: A tool for using software libraries. ACM Transactions on Software Engineering and Methodology (4), 146–170 (1995)
53. Zaremski, A.M., Wing, J.M.: Specification matching of software components. ACM Trans. Softw. Eng. Methodology (TOSEM) **6**(4), 333–369 (1997). 261641