

Semantic Discovery and Composition of Web Services for RFID Applications in Border Control¹

Aabhas V Paliwal¹, Nabil Adam¹, Christof Bornhövd², Joachim Schaper²

¹RUTGERS UNIVERSITY, CIMIC, Ackerson Hall, Newark, NJ 07102, USA

²SAP Labs, LLC, Palo Alto Research Center., 3475 Deer Creek Road, Palo Alto, CA
94304, USA

Contact email: adam@adam.rutgers.edu

Abstract. Recognition is growing for the need of richer semantic specifications of Web Services to support automated service selection and invocation, automated or semi-automated service composition, as well as more comprehensive approaches to service monitoring. In this paper, we combine [12] an approach for composing Web Services on the semantic web and [10] using OWL-S for explicitly describing the semantics to Web Services. We propose an OWL-S based approach for the automatic composition of Semantic Web Services and present a technique to generate composite services from high-level declarative descriptions. We discuss its implementation for the shipment monitoring application, an open, multimodal end-to-end tracking and tracing system in the Border Control domain based on Semantic Web Services as part of the SAP Auto-ID infrastructure.

1. Introduction

With existing Web Service (WS) technology, it is difficult to map a complete real world operation to a particular set of Web Services due to the plethora of Web Services being provided. The problem is compounded by varying consumer (service user) needs and existence of multiple combinations of service that can be used to achieve the task. This renders the composition of services, with direct human intervention, more difficult, time consuming, and an error prone process. Semantically sufficient, machine interpretable Web Service description could facilitate the automated composition of a composite service [16]. The semantics of the WS are crucial to achieve automated composition offering relevant service characteristics. These characteristics are syntactic and semantic in nature [11]. The syntactic features include the operational parameters and service bindings. Application domain, functional categories, and offered business functionality are some of the semantic features. The semantic features of a WS are described utilizing ontologies, to extend WS interoperability to Semantic Web Service (SWS)

¹ This work is supported in part by the National Science Foundation under grant IIS-0306838 and SAP Labs, LLC.

interoperability. SWS integrate the evolving technologies dealing with ontologies and WS. This forms the basic foundation of our approach.

Ontologies, a pre-requisite for this paradigm, are a key component of this integration between the domains of Semantic Web and WS. They are used for the conceptualization of the application domain in a human understandable and machine-readable form. Ontologies and description languages based on ontologies such as OWL form the foundations on which automated WS composition can be based [4,11]. OWL-S [13] supports automated discovery and composition of Web Services by providing a set of ontologies that describe their properties and capabilities. The Process Ontology, part of the OWL-S specification, declares the semantic markup as it considers the Web Services as a set of simple and complex actions with pre-conditions and effects [16]. Simple services are independent, self-reliant services implementing the task functionality. Composite services on the other hand are a combination of services providing the task functionality. In this work we focus on issues related to automated semantic Web Service composition, in particular the way operations are interconnected, services are invoked, and messages are mapped to one another. We outline our approach for building semantically rich software services and illustrate its application of a tracking and tracing service for the Border Control domain. These tracking and tracing service is based on the deployment of RFID readers and environmental sensors within SAP's Auto-ID Infrastructure.

Radio Frequency Identification (RFID) represents a giant leap in the field of automated data collection. RFID devices can be used to read through most non-metallic substances and under other visually and environmentally challenging conditions. A basic RFID system consists of an antenna or coil, a transceiver (with decoder), a transponder (commonly called an RF tag) that is electronically programmed with unique information [3].

The antenna emits radio signals to activate the tag and read and write data from/to it. Antennas are the conduits between the tag and the transceiver, which controls the system's data acquisition and communication. RFID tags can be placed outside or inside a container, inside a pallet or even a packaged box. RFID tags are categorized as either active or passive. Active RFID tags are powered by an internal battery and are typically read/write [3]. The range depends on the power of the RF component. The tag can store any type of information about the container, e.g., ID number, position, manifest, and seal number. Passive RFID tags operate without a separate external power source and obtain operating power generated from the reader. Passive tags are consequently much lighter than active tags, less expensive, and offer a virtually unlimited operational lifetime. The trade off is that they have shorter read ranges than active tags and require a higher-powered reader. RFID data can be combined with data coming from environmental sensors which can also indicate observations like opening of container doors or cases, temperature alarms, or humidity conditions [3].

Through automatic, real-time object tracking, smart items technology can provide more accurate data about business operations in a more timely fashion, as well as help streamlining and automating the operations themselves. This leads to cost reduction and additional business benefits such as increased asset visibility, improved responsiveness, and even extended business opportunities. However, bridging the gap

between the physical and the digital world requires flexible and scalable system architecture to integrate automatic data acquisition with existing business processes. SAP's Auto-ID Infrastructure (AII) integrates data from RFID and sensor devices with enterprise applications. The AII converts RFID or sensor data into business process information by associating it with specified mapping rules and metadata. These mapping rules can feed incoming observation data directly to business processes running on backend systems, execute predefined business logic, or simply record the data in a persistent store for later analysis [3].

1.1 Problem Statement

Widespread availability of resources and services enables the interaction with a number of potential components. WS Discovery and Composition should seek and receive process descriptions by the advertising services and, as dynamically new requirements or functionalities are requested, find the most appropriate matches and incorporate them. This matchmaking is based on common, extensible, ontologies describing both services and their functionality. The initial step consists of WS discovery wherein potential components constituting the overall service have to be identified. The process of searching for possible matches between the advertised services and components of the composed service is called the Matchmaking phase, which includes finding all those services that can to some extent fulfill a requirement, and eventually recommend the best ones.

Our problem environment focuses on gathering real-time shipment monitoring data. This constitutes complex condition-effect operations wherein failure may occur at various stages of the operation. To meet the requirements of the environment there is the need for (1) Dynamic service composition relying on system feedback and conditions-effects associated with service operations. (2) Real-time mapping of the component services to services constituting the composite service (3) Service Monitoring capability, overseeing the service operations along with individual services, to achieve goal-reachability. (4) Mechanisms to facilitate seamless recovery from failure. This raises several research challenges for automation of Semantic Web Services in this environment. In this paper we limit our focus on addressing: (1) efficient discovery of relevant processes accurately (2) Service composition and selection based on the pre-conditions and post-effects.

The remainder of this paper is organized as follows. In Section 2 we will outline the motivating scenario together with a typical example. A discussion of our approach is presented in Section 3. The application of our approach to the Border Control domain is discussed in Sections 4, and our conclusion and future work are discussed in Section 5.

2. Motivating Scenario

2.1 Background

US Customs deals with a huge number of cargo trucks and shipments crossing borders by air, water and land. One third of the total number of trucks entering the United States per year come through just four international bridges between the province of Ontario, and the states of Michigan and New York [5]. Typically, a thorough physical inspection of a loaded 40-foot container or an 18-wheel truck at the border takes five inspectors three hours. While recent events call for heightened security measures at border control, it is not practical to inspect all cargo crossing the border. Therefore, it is essential to minimize the required human effort in the inspection and monitoring process without compromising on the quality [5,7]. Currently the Manifest information about cargo arriving in the US is entered into the AMS System (a new consolidated system called ACE is proposed to be implemented which interfaces with various agencies and other databases to check for compliance). To distribute processing and inspection of in-bond or in transit cargo (i.e., headed to a foreign location via the US), the “*point of arrival*” of the cargo may be different from the official “*point of entry*”. For example, cargo might arrive at Chicago but not officially be entered into the US until it reaches a customs checkpoint in Newark, NJ. The example used in this paper is based on US Customs regulations for importing [6], which implies that the potential gap between the point of entry and point of arrival and the lapse of 15 days or more leaves room for cargo tracking between the two points during that time period.

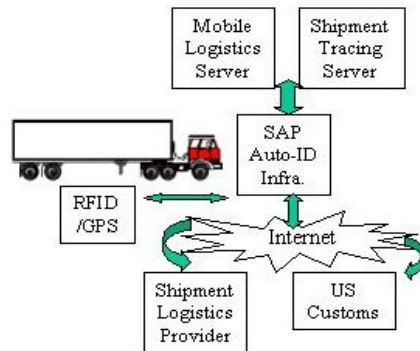


Fig. 1. Border Control Motivation Scenario.

Instead of using traditional identification and tracking methods, placing wireless electronic tags on equipment provides an accurate tracking methodology when the equipment is within the tracking range of an RFID device. RFID does not need a line of sight to track equipment. The track and trace data collected from RFID and GPS devices into the track and trace system can be seamlessly integrated into the US Customs’ own computer system and updated in real time as soon as the container changes position or status. A web portal can be deployed between the track and trace users and the US Customs’ own system. Using an XML interface the necessary data from the parent system, US Customs’ enterprise system, can be transferred into the Web-based system periodically. By using the web the data can be viewed from any Internet compatible device, from PC to a mobile phone to a PDA. A wide range of applications can also be generated to satisfy the needs of the businesses who wish to utilize their equipment to the full extent by

combining GPS and RFID technologies for an effective track and trace service, as illustrated by the Border Control scenario in Figure 1.

2.2 Border Control Example

As a running example we consider here the tracking and tracing application within the Border Control domain. Assume that a company provides a Monitor Shipment (MS) composite service that offers monitoring of shipments. The company's customers submit their request to MS. MS subcontracts from other Web Services to handle each request. Examples of subcontracted services include Tracking service (TK), Tracing service (TC), Shipment Description service (SD), Shipment Tag Identifier (STI), and Consignee Information (CI). A typical example scenario would be of a customer, i.e., the person receiving the shipment, using the MS service to track a shipment. The customer would start by invoking MS's, submitTrackingNumber operation to get the tracking information. MS interacts with STI via getShipmentTagID to obtain the shipment identifier tag. To get the tracking information, MS would then transparently interact with a tracking service via TK's getTrackingInfo operation. If the customer wishes, the customer could also be able to check the route history and obtain a report by invoking MS's displayRouteInformation operation. Subcontracting from TC's getShipmentTrace operation provides this service. The customer also has the option to obtain the details of the shipment by invoking showShipmentDetail provided by MS. However, before displaying the shipment details, MS would check the customer's identity by invoking CI's checkCustomerRegistration operation. If the check is positive MS would invoke the displayShipmentInformation operation offered by the shipment description service.

One of the (pre) conditions for the getShipmentTrace operation of TC is validShipmentManifest and service constraints are the access control privileges accorded to the client to view generateShipmentTraceReport. Preconditions are logical formulae that need to be satisfied by a service requestor prior to the execution of the service, e.g., check for the validity of client registration (validCustomerRegistration) to view the details of the shipment, SD's displayShipmentInformation operation. Effects are logical formulae that state what will be true upon the successful execution of the service, e.g., displayShipmentTag, for the retrieval of a shipment tag identifier by STI's getShipmentTagID operation. OWL-S effects are the side effects of the execution of the service [10].

Many information-providing services have no condition-effects. Complex conditional services in the Border Control domain, however, do have condition effects, such as checking client registration (condition) before the display of shipment information (effect). The description of these condition effects is, hence, critical to certain aspects of Web Service automation. In the Border Control domain, amalgamation of RFID technology with SWS provides enhanced shipment monitoring processes in agreement with the semantics of the US Customs surveillance procedures. This new technology as compared to the existing technology offers a higher degree of accuracy to the US Customs for efficiently identifying, managing and tracking individual cargo shipments.

3. Proposed Approach

We propose an OWL-S based approach for the automatic composition of Semantic Web Services from high-level declarative descriptions. The Customs process ontology describes and classifies major customs' surveillance procedures using process dependencies and specialization. Each procedure is modeled as a composite process, which is a collection of activities that can be decomposed into sub-activities, which may themselves be processes. In turn, synchronization is modeled as the management of dependencies that represent flows of control or data, between activities. The sub-processes are arranged into generalization-specialization taxonomy, with generic processes at the top and specialized processes underneath. The specialized process inherits the properties and functionality of the generic process. The specialized process extends the generic process with specific properties and functionality. This taxonomy supports multiple inheritances, as it is possible for a process to have multiple parents. With our approach we combine an approach for composing Web Services on the semantic web [12] with using OWL-S for explicitly describing the semantics to Web Services [10]. We discuss its implementation for the shipment monitoring application, an open, multimodal end-to-end tracking and tracing system in the Border Control domain based on Semantic Web Services as part of the SAP Auto-ID Infrastructure. An overview of the approach is presented in the next sub-section followed by a detailed discussion of the various components in the following sub-sections.

3.1 Overview

WS Composition involves the process of selecting, combining, and executing WS to achieve the objective of a user request. This involves resolving constraints between Web Service inputs, outputs, preconditions and effects (IOPEs) along with the outputs and effects (OEs) of the user request. In addition to matching IOPEs, the automated WS Composition problem also can involve selecting from alternative Web Services that match the IOPE constraints of the composition problem [10]. In order to select from alternative services, the service composer also requires some form of service selection. This also requires a representation of the properties, capabilities and functioning of a Web Service.

We first require an OWL-S description of that service that more fully represents the inputs and outputs of the service, i.e., constructing a composite process model that links the various operations provided by the Web Service into semantically meaningful message patterns, e.g., checking client registration before displaying shipment details. We refer to this first phase as the *service specification phase*. We make use of capability matching as described in [10] that compares the capabilities provided by any of the advertised services with the capabilities needed by the requester. The goal is to find the service provider that produces the results required for the requester. In general, it is unrealistic to expect that the capabilities offered by a service will exactly match the request. For example, the request may be for location information, and the task of the matching engine is to decide whether it can be accomplished by a service that provides tracking information [10,1]. The matchmaker

should determine how likely it is that each capability advertisement indicates that the service will accomplish the particular function specified in the request. A number of capability matching algorithms have been proposed for OWL-S such as [8,9,14]. They use the service descriptions in the Service Profiles and the ontologies that are available to decide whether there is a match between service requests and the advertisements of the services provided. We refer to this second phase as the *matchmaking phase*. Together, these two phases lay the foundation for automatic service discovery, service composition and service execution in the Border Control application.

WSDL, in essence, allows for the specification of the syntax of the input and output messages of a basic service, as well as other details needed for the invocation of the service. WSDL does not, however, support the specification of workflows composed of basic services. OWL-S and other related work may be viewed as efforts to lay the foundations for the most effective evolution of Web Service-related capabilities that can be supported with current and maturing technologies [10]. Therefore, we utilize mechanisms by which OWL-S can be used along with dominant Web Services standards, such as WSDL and UDDI.

3.2 Semantic Service Discovery

UDDI (Universal Description Discovery and Integration) is an industrial initiative to create an Internet wide network of registries for Web Services [15]. UDDI allows businesses to register their presence on the Web by specifying their points of contact both in terms of the ports used by the service to process requests and in terms of the physical contacts with people that can answer questions about the service. In addition, UDDI provides a language to specify an unbounded set of features of services that can help the process of service location and selection as well as service invocation.

[10] states that despite its role, UDDI provides a very weak discovery mechanism, which makes it impossible to locate a Web Service solely based on the business functionality it provides. The main problem with UDDI is that it does not provide a capability representation language such as the OWL-S Service Profile. As a consequence, UDDI does not provide capability-based search as identified by [10]. The result is that UDDI supports the location of information about the Web Service; only once it is known which Web Service to use. OWL-S and UDDI complement each other. UDDI provides a worldwide-distributed registry that is virtually an industry standard. On the other hand, OWL-S provides the information required for capability matching. The OWL-S/UDDI matchmaker integrates OWL-S capability matching into the UDDI registry. This integration is based on the mapping of OWL-S service profiles to UDDI Web Service representations. We make use of the mapping function provided in the approach mentioned in [10] that defines a set of specialized UDDI TModels that store OWL-S information that cannot be represented in the standard. The integrated OWL-S/UDDI provides the same functionality provided by UDDI using exactly the same API, so that any UDDI server can interact with it to retrieve information about available Web Services. In addition, our OWL-S/UDDI matchmaker supports capability matching by taking advantage of OWL-S capability representation and the matching process proposed in [10]. The result is a UDDI

repository, which allows searching for and finding Web Services by their capabilities. Although the approach presented in [10] forms the initial step of our combined approach, there exist other approaches to extend UDDI with semantic constructs. [1] propose a method for semantically enhancing the service discovery capabilities of UDDI. Their approach consists of an extension to the UDDI inquiry API specification to enable requesters to specify the required capabilities of a service. They also enhance the service discovery through UDDI by performing semantic matching and automatic service composition using planning algorithms.

3.3 Semantic Service Composability

A major issue when defining a composite service is the composability of its component services [2]. The mapping between the parameters requested by the calling operation, e.g., data types, number of parameters of the compound service, and number of parameters of the client service enables operation invocation. [12] identify two sets of *composability rules* to compare *syntactic* and *semantic* properties of Web Services. The syntactic composability rules include: (1) *mode composability*, which compares operation modes, for example a type *notification* operation at one service must be connected to a *one-way* operation type at a partner service and similarly, a *solicit-response* operation type maps to a type *request-response* operation at a partner service and (2) *binding composability*, which compares the binding protocols of interacting services, i.e., at least one of the protocols expected by a Web Service must be supported by the other. The semantic composability rules include: (1) *message composability*, which compares the number of message parameters, their data types, business roles, and units. Interactions between Web Services involve the exchange of messages, which consist of one or more parameters, each having a certain data type. It is hence important to check that the data types of the parameters sent by a service are compatible with the data types of the receiving service or can be converted accordingly. However not all parameters of the invoked operation need to be mapped to the parameters of the calling operation. The rationale is that an input message of a service operation may use only a subset of the parameters sent through an output message. For example, the input message of the service operation may exclude the weight parameter from the showShipmentDetail operation. The *message composability* rule thus compares the input and output messages of every pair of operations. We make use of the idea proposed by [12] to check that each input of an operation is data type-compatible with the output of the other operation, implying that the parameters of each input message map to all or some of the parameters contained in the output message of the other operation, as part of our approach. (2) *Operation semantics composability* compares the semantics of service operations. This rule ensures that interconnected operations have *compatible* purposes and categories. Based on the notion of *compatibility* between categories and purposes, [12] define *operation semantics composability* rules stating that calling service operation is *operation semantics composable* with the invoked service operation if the purpose of the former is compatible with the purpose of latter and the category of the calling operation is compatible with the category of the invoked operation. (3) *Qualitative composability* Web Service rules check the qualitative properties of interacting

operations such as the *Security* composability, which guarantees that if the calling operation uses security mechanisms, e.g., encryption and nonrepudiation to exchange messages, then the invoked operation uses them also. These rules also consider the *Privacy* composability of two services whereat the privacy preferences of the calling service should be subsumed by the privacy features exposed by the invoked service. We extend this composability model by combining it with the semantic UDDI approach described in [10,1].

3.4 Semantic Discovery and Composition

We propose an extended approach for the automatic composition of Web Services that consists of a specification, matchmaking, a selection, and a generation phase as illustrated in Figure 3. The specification phase enables high-level descriptions of the desired compositions using semantically extended WSDL. The matchmaking phase uses composability rules to generate *layouts* that conform to service composers' specifications [12]. Layouts include the list of component services and their interactions with each other required to form the composite service. The matchmaking algorithm uses as input the service composer's specification and UDDI of preexisting service interfaces, which is described in WSDL extended by semantic constructs [10]. Using the selected layout, a detailed description of the composite service can be automatically generated. This description includes the list of subcontracted services, mappings between the composite and subcontracted services operations and messages, and the control flow of subcontracted operations. The control flow refers to the execution order of the operations subcontracted by the composite service. Specification is the phase that requires the service composer's intervention to describe the desired composition. The matchmaking phase automatically generates composition plans based on the service composer's specification. The selection phase uses composability thresholds to select the best layout. Threshold parameters are given by service composers based on the profiles defined in the specification process. The generation phase automatically provides a description of the generated composite service in a specified language such as WSFL [12].

Specification Phase. Specification of operations forms the basic building blocks of Web Service in WSDL. Input/output message syntax and patterns are specified within the organizational structures of these operations. The *atomic process* in OWL-S is characterized primarily in terms of its inputs, outputs, preconditions, and effects. Here, the description being interpreted is the OWL-S process model published by the service provider along with the WSDL specs to which it is grounded. The class-hierarchical, description logic-based typing system of OWL gives types of inputs and outputs of an atomic process, which allows for the use of concepts defined and shared as part of the Semantic Web [10]. For example, the accompanying code sample gives a simplified OWL-S declaration of an atomic process with its IO specifications. It is assumed that TrackingID, TagID and ManifestID are classes defined in the application domain ontologies that have specific namespaces. The grounding for this atomic process would establish its correspondence to a particular WSDL operation, and the correspondence of each IO element to a particular WSDL message part element. Also, if needed, the grounding could specify an XSLT script to transform

```

<AtomicProcess ID="ShipmentDescription">
  <hasInput>
    <Input ID="TrackingNumber">
      <parameterType resource="#TrackingID">
      </Input>
    </hasInput>
    <hasInput>
      <Input ID="ShipmentTagID">
        <parameterType resource="#TagID">
        </Input>
      </hasInput>
    <hasOutput>
      <Output ID="ShipmentManifestID">
        <parameterType resource="#ManifestID">
        </Output>
      </hasOutput>
    </AtomicProcess>

```

Fig. 2. Input/Output Specification of Atomic Process

each OWL-expressed input to the precise syntactic form specified by WSDL and vice versa for outputs.

Matchmaking Phase. The general premise of the matchmaking algorithm is to initially map the composite Web Service operations to a set of specialized UDDI TModels that store the corresponding OWL-S information. The integrated OWL-S/UDDI provides the same functionality provided by

UDDI offering the same API, so that any UDDI repository can interact with it to retrieve information about available Web Services. Next, each calling operation of the composite service is mapped to one or more operations of the existing service. The algorithm looks for the composite Web Services description so that the purpose and category are compatible with that of the available services. Then the algorithm verifies that interacting services are binding composable. For each pair of operations, it also iteratively checks mode composability, operation semantics composability, and message composability.

Matchmaking Algorithm. The following steps describe the mapping of the `submitTrackingNumber` operation of the *Monitor Shipment* composite service according to our matchmaking algorithm.

- ◆ Map the composite Web Service operations to a set of specialized UDDI TModels that store the corresponding OWL-S information of the Tracking Service.
- ◆ Identify the component services (e.g., Tracking Service) supporting the SOAP protocol so that `submitTrackingNumber`'s purpose and category are compatible with the service purpose and category.
- ◆ Determine operations of the *Tracking Service* that are mode composable with `submitTrackingNumber`. Since `submitTrackingNumber` is a solicit-response type of operation it shall map to a corresponding request-response operation `getTrackingInfo`.
- ◆ Test the operations for message composability. The input of `submitTrackingNumber` is compared with the output of `getTrackingInfo`. All of `getTrackingInfo` output's parameters are mapped to the corresponding parameters of `submitTrackingNumber`'s. Since we can determine such a mapping, the two operations are message composable.
- ◆ Insert a "plug-in" between the operations in the layout, since both operations are syntactically and semantically composable.
- ◆ Perform iterations for all other service operations required by the composite service.

Selection and Generation Phase. At the end of the matchmaking phase, several composition layouts may have been generated. Composition layouts are sorted and returned according to their ranking. Plans with the highest ranking, based on the quality parameters specified within the service specifications, are returned first. Developing a comprehensive ranking algorithm forms part of our future work. The last phase in our approach aims at generating a detailed description of a composite service. This description includes the list of outsourced services, mappings between

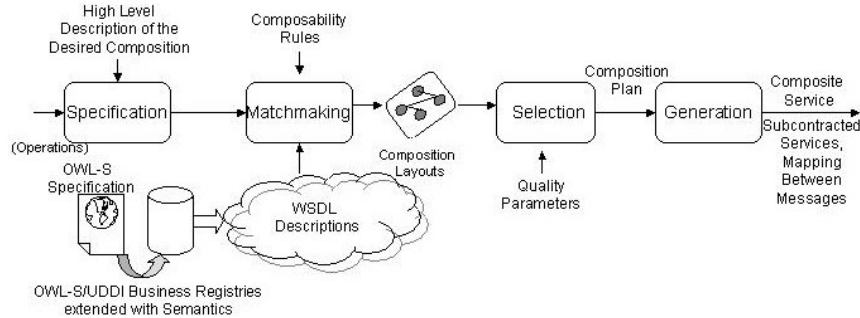


Fig. 3. Approach for Semantic Service Discovery and Semantic Composition composite service and component service operations, mappings between messages and parameters, and flow of control and data between component services.

4. Application to Border Control

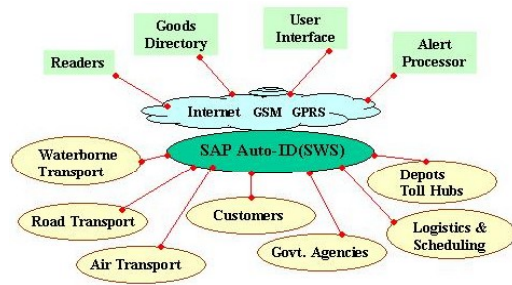


Fig. 4. Border Control Monitoring System Overview

A SWS-enabled Border Control Monitoring System is shown in Figure 4. The system is based on SAP's Auto-ID Infrastructure (AII) [3] to provide RFID support. The AII integrates RFID readers and environmental sensors with business applications. It converts RFID or sensor data to business process information by associating it with specified mapping rules

and metadata. These mapping rules can feed incoming observation data directly into business processes running on backend systems, execute predefined business logic, or record the data in a persistent store where it can be made available to a variety of Semantic Web Services. SWSs support allows the (semi-) automatic identification, composition, and execution of AII services (like Item Tracking and Tracing) in highly distributed, and rapidly changing environments where manual identification,

configuration, and composition of functionality (e.g., by a programmer) is not always feasible.

The remaining section is organized as follows. Sub-section 4.1 describes the various components of the SAP Auto-ID Infrastructure. The structure and operations of the tracking and tracing services are discussed in 4.2. Sub-section 4.3 presents the underlying application ontology.

4.1 Auto-ID Infrastructure Architecture

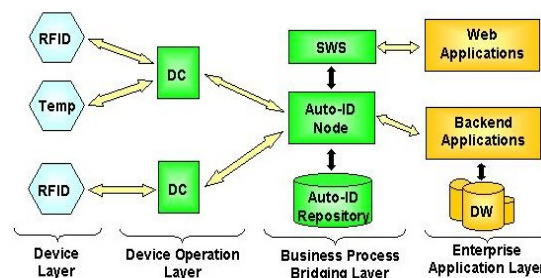


Fig. 5. Auto-ID Infrastructure Architecture

functionality to filter, condense, aggregate, and adjust received sensor data before passing it on to the next layer. The Business Process Bridging Layer associates incoming observation messages with existing business processes. At this layer status and history information, including object location, packaging information, and information about the environment of tagged objects is maintained. This functionality is realized by a so-called Auto-ID node. Finally, the Enterprise Application Layer supports business applications such as Online Shipment Tracking and Tracing, or Supply Chain Management (SCM), or, as in our case, different Semantic Web Services.

The architecture of SAP's Auto-ID Infrastructure is shown in Figure 5. Different types of sensor devices are supported via a hardware-independent low-level interface at the Device Layer. One or more Device Controllers forming the Device Operation Layer by coordinating multiple devices. This layer provides

4.2 Tracking and Tracing Services

Tracking and tracing services address the two main types of requirements efficient logistics management and enhanced shipment monitoring. Tracking is assumed to be a forward moving service (projects the expected location and time) and tracing a backward moving service (processes the past location at a given point in time).

Tracking is defined as a service that provides information on the current status of the tagged item, where the shipment or item identification is provided as input. Output of the service is all information available on that particular item.

Tracing is a more expansive service. Input to the tracing service includes a plan that includes the expected time and location of the shipment along the route of delivery. Output includes the output of the tracking service plus an indication of whether the item is in accordance with the submitted plan.

The tracking and tracing services are part of the Monitor Shipment composite service. Based in the monitoring information gathered by the SAP Auto-ID Infrastructure, through a graphical interface, the shipment to be monitored is specified to the system before a shipment enters the point of entry. The Monitor Shipment service consists of operations that, apart from processing the manifest information, process and analyze a planned timetable, a combination of locations and times, and an allowed delay time-

```
<owl:Class rdf:ID="Monitoring Service">
<owl:Class rdf:ID="Tracing Service">
  <rdfs:subClassOf rdf:resource="Monitoring Service"/>
</owl:Class>
<owl:Class rdf:ID="Tracking Service">
  <rdfs:subClassOf rdf:resource="Monitoring Service"/>
</owl:Class>
<owl:Class rdf:ID="TrackShipment">
  <rdfs:subClassOf
    rdf:resource="Tracing Service"/>
</owl:Class>
<owl:Class rdf:ID="TraceShipment">
  <rdfs:subClassOf
    rdf:resource="Tracking Service"/>
</owl:Class>
```

Fig. 6. Tracking and Tracing Service Structures

window. The real time observations on the movements of the shipment are based on the data gathered with the attached RFID tags and the installed readers along the path of travel. Deviations from the planned timetable cause an alert to be raised. The alert may be an internal event in the system raised at one of the AINs of the system. This alert is then transferred to the alert generation module interacting with the Monitor Shipment service. The system also provides the customers with the possibility to send in bookings and status report requests. The criteria for when to send the specific notifications to which customer are specified in the service class of the component services. The basic service structures for the Tracking and Tracing service are described as in Figure 6.

The tracking and the tracing services are implemented as SWS, as the system must operate reliably under dynamic changes of its topology: new readers may be added. Processes can cope with this role. Additionally the effect of a failing system component should be encapsulated as much as possible, limiting the reduction of the functionality still available at other system components - graceful degradation. Moreover, the organizational structure of the distributed participants on a global scale prohibits the adoption of a centralized model.

4.3 Border Control Ontology Fragment

The cargo manifest has 16 data fields to be filled. In addition to this, there are over 75 different documents that need to be sorted for every shipment. Complex representation along with the large volume of data associated with a single shipment creates the need for ontological representation of the data being submitted by the shipper. Ontologies such as our Border Control ontology aim to support the interpretation of this data. Figure 7 shows a overall fragment of the Border Control Ontology. Within our Border Control ontology fragment the Cargo Manifest, for example, has been represented by 16 fields, which represent the fields that need to be

filled within that form. The representation of these fields in this ontology is in the form of “owl:Datatypeproperty”. The field “WaterWayBillType” can take only three

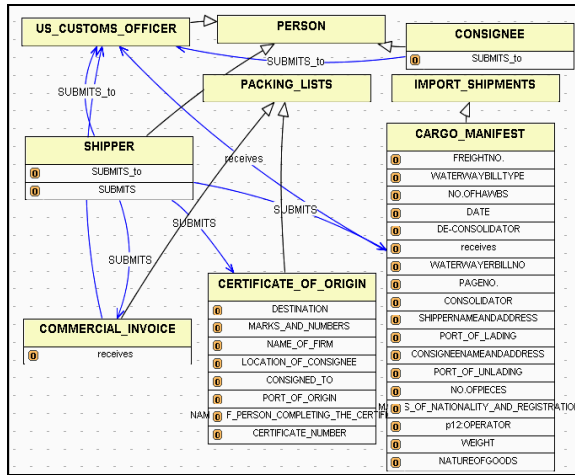


Fig. 7. Border Control Ontology Fragment

Control ontology contains several primitive concepts and the opportunity to expand the various terms is virtually unlimited.

possible values MASTER, HOUSE and SUB, which poses a restriction on this property.

Our Border Control ontology encompasses the terms, which represent the fields of the forms that are required to be submitted by a shipper. The goal is to provide a common vocabulary for the representation of the terms used in the forms to automate the search for anomalous data. Currently, our Border

5. Conclusion and Future Work

In this paper, we have expressed the potential benefits of richer semantics in specifying Web Services. We have provided an initial guide for deploying of Semantic Web Services in the Border Control domain using OWL-S in conjunction with WSDL, UDDI and related standards.² As part of our future work, we plan to extend the semantics to support greater automation of service selection and invocation and more comprehensive approaches to service monitoring and recovery from failure. Richer semantics can also help to provide fuller automation of such activities as verification, simulation, configuration, and negotiation of services. Mapping input/output parameters and specifying the pre/post conditions are found to be sufficient for proving properties of service compositions, however, these properties are assertions only on the initial and final states of the service, respectively. They do not help in specifying/verifying ongoing behavior of an individual service or a composed system. There is therefore a need to focus on research issues dealing with the monitoring of composite Web Services. For the specification of process’ preconditions and effects at a finer granularity, OWL-S allows for the use of a more expressive language than OWL, such as RuleML, DRS, or the recently proposed

² We would like to thank Shantaram Iyer for his valuable contributions to the CIMIC – SAP RFID project. We also benefited from fruitful discussions with other colleagues in CIMIC including Vandana Janeja and Vandana Chopra.

OWL Rules Language. SWRL is a preliminary proposal for a rule language designed to express rules and constraints in the framework of the OWL language. Our future work also involves exploring an approach to specifying rules on top of the process ontologies. The evolution of the RFID and sensor technology towards increasingly “smart” devices present additional privacy and security challenges that need to be addressed.

References:

1. Akkiraju, R., Goodwin, R., Doshi, P., and Roeder, S.: A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI. In Proc of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico, August 9 - 10, 2003
2. Berners-Lee, T.: Services and semantics: Web architecture, 2001.
3. Bornhövd, C., Lin, T., Haller, S., and Schaper, J.: Integrating Automatic Data Acquisition with Business Processes, Experiences with SAP's Auto-ID Infrastructure. In Proc of the 30th VLDB Conference, Toronto, Canada, August 29 – September 3, 2004
4. Bussler, C., Fensel, D., and Maedche, A.: A conceptual architecture for SemanticWeb enabled Web Services. SIGMOD Rec 31(4): 24–29, 2002.
5. Flynn, S.: America the Vulnerable. Foreign Affairs Magazine, January/February 2002.
6. Importing into the United States -- A Guide for Commercial Importers. At <http://www.cbp.gov/nafta/cgov/pdf/iius.pdf>
7. Janeja, V., Atluri, V., and Adam, N.R.: OUTLAW: Using Geo-Spatial Associations for Outlier Detection and Visual Analysis of Cargo Routes. In Proc of The Second National Conference on Digital Government (dg.o 2002), LA, May 1 9-22, 2002
8. Lei, L., and Horrocks, I.: A Software Framework for Matchmaking Based on Semantic Web Technology. In Proc. of the Twelfth International World Wide Web Conference (WWW 2003), pages 331-339, ACM, 2003.
9. Mandell, D. J., and McIlraith, S.A.: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. In Proc of the Second International Semantic Web Conference (ISWC2003), pp. 227-241, 2003.
10. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., and Sycara, K.: Bringing Semantics to Web Services: The OWL-S Approach. In Proc. of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004), San Diego, California, USA, July 6-9, 2004.
11. McIlraith, S., Son, T.C., and Zeng, H.: Semantic Web Services. IEEE Intelligent Systems, Special Issue on the Semantic Web, 16(2): 46--53, March/April, 2001.
12. Medjahed, B., Bouguettaya, A., and Elmagarmid, A.K.: Composing Web Services on the Semantic Web. The VLDB Journal — The International Journal on Very Large Data Bases, Volume 12 Issue 4, November 2003.
13. OWL-S 1.0 Release. At <http://www.daml.org/services/owl-s/1.0/>
14. Paolucci, M., Kawamura, T., Payne, T.R., and Sycara, K.: Semantic Matching of Web Services Capabilities. In Proc. of the First International Semantic Web Conference (ISWC2002), 2002.
15. Universal Description, Discovery and Integration (UDDI). Version 3, 2003. At <http://www.uddi.org/>
16. Wu, D., Parsia, B., Sirin, E., Hendler, J., and Nau, D.: Automating DAML-S Web Services Composition using SHOP2. In Proc of 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, October 2003.