SOCA (2012) 6:167–188
DOI 10.1007/s11761-011-0100-0

ORIGINAL RESEARCH PAPER

Designing, formalizing, and evaluating a flexible architecture for integrated service delivery: combining event-driven and service-oriented architectures

Sietse Overbeek · Marijn Janssen · Patrick van Bommel

Received: 1 September 2010 / Revised: 4 November 2011 / Accepted: 29 December 2011 / Published online: 10 January 2012

Abstract An influx of service providers collaborate in networks to meet their clients' demands. Integrated service delivery (ISD) is a way to let networked service providers offer services to their clients by bundling selected services offered by each provider so that clients do not have to deal with each single provider anymore. Designing such a network is a complicated endeavor as independent organizations need to collaborate and should understand how their activities are dependent on each other. Communication of events is necessary to deal with unpredictable and complex processes in such a network. In contrast with conventional event-driven architecture and service-oriented architecture (SOA) approaches, the hybrid model of event-driven interactions and SOA offers the required flexibility to realize ISD. This flexibility is realized by integrating not only services but also the processes of the different service providers to supply such services. A design science approach has been applied resulting in a detailed and formalized design of an event-driven service-oriented architecture (EDSOA). The EDSOA has been illustrated to show how ISD is realized with support of the architecture in a scenario concerning an application for a temporary residence permit by an immigrant. An evaluative workshop has been conducted which reflected that the following criteria are most important for successful organizational adoption of the EDSOA: expected usefulness, fit with organizational standards, use of trusted technology, and ease of maintenance.

Keywords Design science · EDSOA · Integrated service delivery · Public/private service network · Web services

S. Overbeek (✉) · M. Janssen
Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands
e-mail: S.J.Overbeek@tudelft.nl

M. Janssen
e-mail: M.F.W.H.A.Janssen@tudelft.nl

P. van Bommel
Institute for Computing and Information Sciences, Radboud University Nijmegen, Heijendaalseweg 135, 6525 AJ Nijmegen, The Netherlands
e-mail: P.vanBommel@cs.ru.nl

1 Introduction

Service providers increasingly collaborate in networks to meet client demands. An example of such a network is an industrial network, which includes private service providers that offer an integrated service. This service can be exemplified by a computer with integrated software from various suppliers [1]. A public service network is another type of network which 'has become a common mechanism for delivery of public services' [2]. A combination of both network types is also possible leading to a collaboration between private and public service providers [3]. In such networks, services are delivered by forming temporary coalitions of service providers. These coalitions can be different depending on the integrated service that is provided to a client, which requires high levels of flexibility [4]. Furthermore, these networks might change over time as new service providers might enter the network and others might withdraw. Matching supply and demand of services is a difficult task given the existence of varying networks of service providers and complex client demands. This task is realized by integrated service delivery (ISD), which is a way to let service providers offer a coordinated bundle of services that match variable client needs [5]. With ISD, clients perceive a bundle of services provided by various service providers as a whole and they do not have to deal with each single provider.

Numerous research projects can be identified in literature that have resulted in service-oriented architectures (SOAs) as

a possible technology to provide support to realize ISD (see e.g., [6,7]). Each service provider can make services accessible as *Web services*, which can be integrated and made available by means of a SOA. A Web service can be defined as a software component identified by a URI, whose interfaces and bindings can be defined, described, and discovered as XML artifacts [8]. One of the advantages of Web services is that they allow for the decoupling of service interfaces from considerations related to service implementation and platform selection. According to Chung and Chao [9], "current SOA technologies focus on supporting service integration and collaboration in terms of interface and functionality but less attention has been paid to the issues of service content integration which could lead to service provision inconsistency" [9]. The EDSOA proposed in this paper contributes to this drawback of current SOAs by not only integrating the services but also integrating the relevant *processes* of the different service providers. Performing such a cross-organizational process decreases service provision inconsistency because of the dynamic assembly of the individual processes for service delivery. Such a hybrid EDSOA model has been mentioned in [10] as one of the possible future enterprise service-oriented architectural styles. However, fully elaborated designs, formalisms, and implementations that materialize this architecture style are still a rarity [10].

Next to SOA, event-driven architectures (EDAs) are available technologies to provide support for ISD [11]. In the context of integrated service delivery, an event can be viewed as a request for input from some actor in the cross-organizational process in order to let the process execution continue [12]. An actor is an entity (such as a human or a computer) that is able to perform a task. Events are helpful to keep track of process execution and to understand which actions are demanded from which actor. The identification of one or more tasks that are necessary to process the event and possible assignment of such tasks to available actors is done by EDAs. However, it is the combination of EDA with SOA that makes up a flexible infrastructure for integrated service delivery. Advanced structuring and dynamic adjustments of services by the EDSOA are the main principles for *flexibility* [13]. By appropriately structuring cross-organizational service compositions and interconnected processes, service providers can reduce the effort involved in adjusting to changing business environments. Through IT-supported learning and adaptation, service providers can effectively and quickly reconfigure a set of cross-organizational processes that are appropriate for a changed business environment [13].

Consequently, we have aimed to combine the best of both worlds to develop and evaluate an elaborated design of an event-driven service-oriented architecture (EDSOA) to allow service providers undergo a transition from delivering fragmented services toward flexible delivery of integrated services in which multiple service providers are involved. An EDSOA can be viewed as an architectural style that uses events to coordinate demand-driven services across a network of service providers. This EDSOA consists of a set of connected software components aiming to provide computer-based support for collaborating service providers to successfully achieve ISD. The used research approach is introduced in Sect. 2, while the proposed EDSOA is shown and explained in Sects. 3–6. Section 7 includes an illustration of how the EDSOA can be exploited in practice by means of a scenario in which an immigrant who has moved to the Netherlands applies for a temporary residence permit. An evaluative workshop has been conducted to analyze which criteria are most important for successful adoption of the EDSOA by service providers which is elaborated in Sect. 8. Section 9 compares our study with other approaches in the field and outlines the benefits of our approach compared to others. The conclusions of this paper and an overview of future research are presented in Sect. 10.

2 Research approach

The research approach for the design and evaluation of the EDSOA is based on *design science* [14,15]. This section consists of three parts. Firstly, the design science approach itself will be introduced briefly. Secondly, the five research steps that have been carried out during this research are presented. Thirdly, a list of the core concepts is presented which are the foundations of the EDSOA design.

The larger part of the research currently conducted in the areas of computer science (CS) and information systems (IS) is aimed at the desire to understand and to find new truths about why things work the way they do [16]. That kind of research is generally known as 'natural science' research. On the other hand, there is 'design science' research, which focuses on creation. Design science is aimed at determining how things ought to be in order to attain goals and to function [17]. The purpose of design is to change existing situations into preferred ones [17]. There are five types of theories that can be distinguished [18]: (1) a theory for analyzing, (2) explaining, (3) predicting, (4) explaining and predicting, and (5) design and action. Design science belongs to the latter category and is aimed at creating technological artifacts that serve human purposes, which is in contrast with natural science which is aimed at trying to understand reality. Its technology-oriented results are assessed against criteria of value or utility, and it is determined if they actually work or improve the old situation. The artifact in our study is the design of an EDSOA. Applying design science is aimed at creating effective artifacts instead of purely generating general theoretical knowledge. Design science consists of two basic activities: build and evaluate [15]. Building is the process of constructing an artifact for a specific purpose, and

evaluation is the process of determining how well the artifact is built. An artifact is built to perform a specific task; in this case, the EDSOA is built in order to support ISD and it has been specifically evaluated in a workshop to determine the extent to which service providers acknowledge the advance of the EDSOA design compared to already existing service-oriented architectures.

The construction of the EDSOA artifact is elaborated in Sects. 3–6. The EDSOA design is illustrated in section 7 by means of a scenario in which an immigrant who has moved to the Netherlands applies for a temporary residence permit. The evaluation of the EDSOA design is explained in Sect. 8. The design steps that have been conducted when executing this research are as follows:

1. The development of a detailed model to match supply and demand of services (Sect. 3).
2. The development of a model to depict how event-initiated interactions are realized between actors (Sect. 4).
3. The development of an ontology to generate information to let actors understand the semantics of a cross-organizational process resulting in integrated service delivery (Sects. 5, 6).
4. The illustration of the EDSOA model by means of a scenario in which an immigrant who has moved to the Netherlands applies for a temporary residence permit (Sect. 7).
5. The evaluation of the EDSOA model by means of a workshop to determine the willingness of service providers to adopt the EDSOA (Sect. 8).

When relating these steps to the two basic design science activities, the first three design steps can be viewed as *building* activities whereas the fourth and fifth steps are *evaluation* activities.

Design science results consist of constructs, models, methods, or implementations. For this research, the emphasis is on the development and evaluation of the *constructs* and *models*. Constructs are the defined core concepts in a research to describe the *Weltanschauung*, i.e., 'view of the world'. Core concepts that are defined in this research are centered around the EDSOA design to support ISD and they are listed below. They have been further categorized according to their purpose in the EDSOA design, resulting in concepts related to event-initiated interactions, concepts related to processes, and concepts related to services. The core concepts related to *event-initiated interactions* are:

Actor: An entity (such as a human or a computer) that is able to perform a task.

Broker: A role that can be enacted by an actor for distribution of published events to subscribers of that event type.

Event: A request for input from some actor in the cross-organizational process in order to let the process execution continue [12].

Publisher: A role that can be enacted by an actor for the broadcasting of events or reception of service calls.

Subscriber: A role that can be enacted by an actor for the reception of events and dispatch of calls for services.

The core concepts related to *processes* are:

Cross-organizational process: A set of tasks spanning multiple organizations that are directed from a beginning to an end through a number of operations [19].

Ontology: An agreed understanding of a certain domain, formally represented as logical theory in the form of a computer-based resource [20]. In the EDSOA design, an ontology is used to verbalize process information.

Process specification: A model of the mutually visible message exchange behavior of each of the actors involved in a process [19].

The core concepts related to *services* are:

Flexibility: The extent to which a service provider can adjust to changing business environments by structuring service compositions and processes [13].

Integrated service delivery: A bundle of services provided to clients by various service providers as a whole.

Service call: A message sent by an event subscriber to an event publisher indicating that the sender wishes to act on an event.

Service description: Additional semantics in order to tell the users of a service how the service looks like and how it can be invoked [21].

Service matchmaking: To meet a client's need for services by offering services that fulfill this need [22].

Service provider: Organization that offers services to its clients.

Web-based portal: A Web-based software application that enables service providers to offer clients a single gateway to information that fulfills their personal information need [23].

Web service: A software component identified by a URI, whose interfaces and bindings can be defined, described, and discovered as XML artifacts [8].

As in natural science, these core concepts form a basic language for this research and the concepts can be combined in higher-order constructions to describe and explain the EDSOA design to support ISD. These higher-order constructions are in fact the models that are shown and described in this paper. Concepts that are related with these core concepts and not yet mentioned will be further discussed in the

coming sections. The concepts are materialized in the EDSOA design, consisting of the detailed model of service matchmaking shown in Sect. 3, the detailed model of event-initiated interactions shown in Sect. 4, and a part of the EDSOA for realization of ontology-based process information shown in Sects. 5 and 6. With these models, a mechanism is realized to coordinate the enactment of a cross-organizational process flexibly and in a way that matches an integrated delivery of services and processes. These models which in fact are the results of the building activities are elaborated hereafter.

3 Designing an architecture for matching supply and demand of services

Answering a client's complex request demands the selection and integration of various services provided by multiple service providers. This requires that the interpreted services by the client at the demand side are matched with services provided by the service provider at the supply side. A detailed design of an EDSOA is shown in Fig. 1 and elaborated in this section. At the top-left of the figure, it is shown that a *client* interacts with a Web-based portal to make his service demands explicit. Important concepts as part of the EDSOA design are defined in an exact and precise manner by providing sound formalizations of those concepts. Thus, the service demands of a client can be formalized as follows:

$$\mathsf{Need} : \mathcal{CL} \rightarrow [0, 1] \tag{1}$$

The set $\mathcal{CL}$ is the set of clients. The service demand or service *need* in general can be expressed as a real number within the range [0, 1]. This need function simply expresses the need of a client for a possible service in terms of a real number between 0 and 1. The example expression $\mathsf{Need}(c) = 1$ shows that a client $c \in \mathcal{CL}$ has a very urgent need for some service. If the client is an *immigrant*, for example, the demands can be a request for a residence permit or a health insurance. The service need is equal to 0 if both the services are supplied by the provider and also used by the client. The number 0 expresses that a client's need for a service has been gratified because he has been provided with a matching service to satisfy him. *Which* service a client is specifically looking for in order to satisfy his demands is not incorporated in the need function, as this relation can be made explicit by means of the search function which is shown below. The search function shows which services a client is looking for, and combining this with the need function enables to specify how extreme the need is to acquire a certain service. The portal matches a client's demands and the available Web services in a *Web service repository*. This repository is maintained by the actors representing the service providers that collaborate to achieve ISD, and it is based on Universal Description, Discovery, and Integration (UDDI) of services (see: http://uddi.xml.org). UDDI enables browsing and searching of services based on several categorization schemes describing industry sectors, product catalogs, and geographic information (see e.g., [24]). Service providers are responsible for publishing a description of the services they provide. Examples of service-providing organizations are the Immigration and Naturalization Service, and an embassy such as is depicted in the top-right part of Fig. 1. Clients must be able to search for the services they require and must be able to use them. The process to match supply and demand of services consists of five steps [22]:

1. A service is published in a UDDI registry by a service provider.
2. The client searches the UDDI registry for a specific service. In this case, the client enters a search query in the portal to search for services.
3. Descriptions of candidate services are returned.
4. The client decides which service to use and invokes this service at the provider.
5. The service is executed and the result is sent back to the client.

These five steps can be formalized by means of the following functions. The publish function formalizes the publishing of a service in a UDDI registry by a service provider:

$$\mathsf{SPublish} : \mathcal{SE} \rightarrow \mathcal{SP} \tag{2}$$

The set $\mathcal{SE}$ is the set of services, while the set $\mathcal{SP}$ is the set of service providers. For example, an insurance company $p \in \mathcal{SP}$ that publishes a health insurance service $s \in \mathcal{SE}$ can be expressed as $\mathsf{SPublish}(s) = p$. The formalization of clients that search the UDDI registry can be made explicit as follows:

$$\mathsf{Search}, \mathsf{Select} \subseteq \mathcal{CL} \times \mathcal{SE} \tag{3}$$

The expression $(c, s) \in \mathsf{Search}$ shows that client c searches for service s in the repository. The descriptions of services are used in order to return candidate services to a client. A client may search the service repository for suitable services, and issuing a search command leads to the presentation of a list of services to the client from which the client can select a service. Service selection is part of the fourth step from the service matching process and is further explained in the next paragraph. The 'select' equation has an identical formalization as the 'search' equation, i.e., both equations are Cartesian products of the same sets. This is because a client as an element of the set of clients can *search* for a service as an element of the set of services. Subsequently, a client as an element of the set of clients can also *select* a service as an element of the set of services. The fact that each service has a unique description can be modeled as follows:

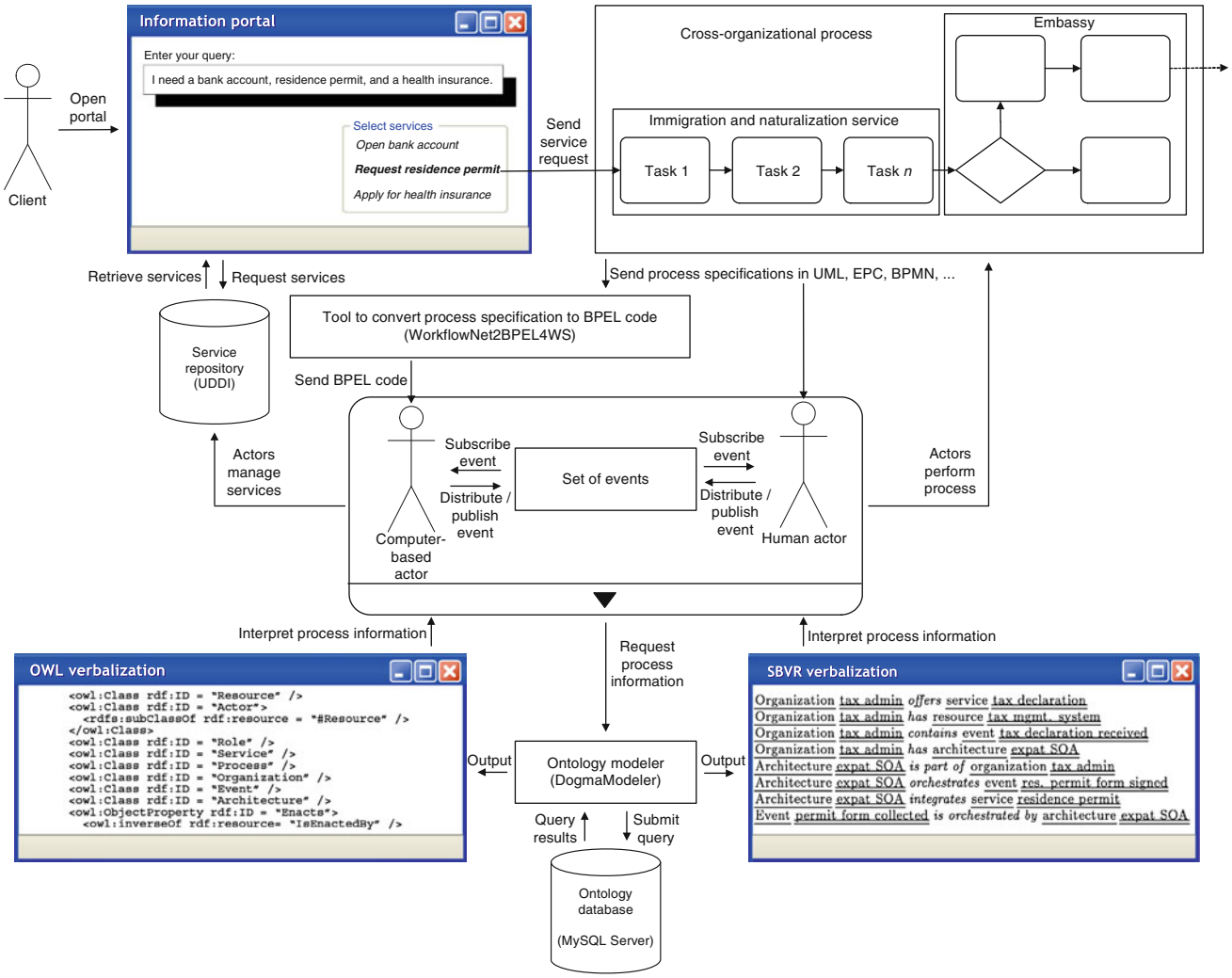

Fig. 1 Design of an event-driven service-oriented architecture

$$\mathsf{Describe} : \mathcal{SE} \leftrightarrow \mathcal{SD} \tag{4}$$

This *bijective* function enforces that the elements of the set $\mathcal{SE}$ should be related with exactly one element of the set $\mathcal{SD}$, and every relation between two members of these sets may not occur more than once. For example, the expression $\mathsf{Describe}(s) = d$ shows that service s has a unique description $d \in \mathcal{SD}$. All services have descriptions, and all descriptions are uniquely assigned to a service. Next, the return of the resulting descriptions of candidate services is formalized as follows:

$$\mathsf{Result} \subseteq \mathcal{CL} \times \wp(\mathcal{SD}) \tag{5}$$

This implies that the expression $(c, D) \in \mathsf{Result}$ shows that a client c receives a set of descriptions $D \subseteq \mathcal{SD}$. Example descriptions that are shown in the top-left of Fig. 1 are 'open bank account', 'request residence permit', and 'apply for health insurance'. A client can select a constituent service to make use of in the portal, such as the 'request for a residence permit' service shown in Fig. 1.

This is the fourth step from the service matching process. The formalization of this step results in the 'select' function which has an identical formalization as the search equation as explained above. Once a client has selected a service from the list of candidate services, a cross-organizational process is initiated by those organizations that take part in delivering the constituent service. An organizational process is a set of one or more linked procedures or tasks which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships [25]. Using this definition, a process can be formally modeled as follows:

$$\mathsf{Process} \subseteq \wp(\mathcal{TA}) \times \mathcal{GL} \tag{6}$$

The set $\mathcal{TA}$ is the set of tasks, while the set $\mathcal{GL}$ is the set of goals. A set of one or more linked tasks $T \subseteq \mathcal{TA}$ which

collectively realize some goal $g \in \mathcal{GL}$ and, therefore, can be considered as a process can be expressed as $(T, g) \in \mathsf{Process}$. Individual organizational processes that need to be executed to offer a service that is part of a set of integrated services need to be combined to realize a joint process for offering a constituent service or a set of integrated services.

In other words, how the fifth step to match supply and demand of services is realized. Delivery of one or more services by means of a cross-organizational process can be formalized as follows:

$$\mathsf{Delivery} : \wp(\wp(\mathcal{TA}) \times \mathcal{GL}) \rightarrow \mathcal{SE} \tag{7}$$

The expression $\mathsf{Delivery}(\{X, v\}, \{Y, w\}) = s$ shows that a cross-organizational process consists of the processes $\{X, v\}$ and $\{Y, w\}$ which result in the delivery of a service s. In the middle of Fig. 1, two types of possible *actors* are shown which may perform a cross-organizational process. These actors are either human actors or computer-based actors. They are part of a service-providing organization performing atomic tasks as part of an organizational process, or in the case of ISD a cross-organizational process. An example of a human actor is a tax officer that processes an immigrant's request for a tax declaration. An example of a computer-based actor is an intelligent software agent that partially completes an immigrant's (Web-based) tax declaration form based on the immigrant's personal data stored at the tax office or at connected agencies that share data with the tax office. Actors actually *perform* the cross-organizational process such as is shown in Fig. 1. In fact, actors perform the constituent *tasks* as part of a process which can be modeled in a formal way as follows:

$$\mathsf{Perform} : \mathcal{TA} \rightarrow \mathcal{AC} \tag{8}$$

The performance of a task $t \in \mathcal{TA}$ by an actor $a \in \mathcal{AC}$ can simply be expressed as $\mathsf{Perform}(t) = a$. The set of actors $\mathcal{AC}$ is the union of the sets of human actors and computer-based actors, because a distinction is made between human actors and computer-based actors. Formally, this implies $\mathcal{AC} = \mathcal{HA} \cup \mathcal{CA}$, where $\mathcal{HA}$ is the set of human actors and $\mathcal{CA}$ is the set of computer-based actors. Both types of actors require readable process specifications in order to contribute to process execution. The read function can be introduced for this purpose:

$$\mathsf{Read} \subseteq \mathcal{AC} \times \mathcal{PS} \tag{9}$$

The expression $(a, q) \in \mathsf{Read}$ shows that an actor a reads a process specification $q \in \mathcal{PS}$, where $\mathcal{PS}$ is the set of process specifications. More specifically, a computer-based actor is only able to read machine-readable process specifications, while a human actor requires human-readable specifications. The set of process specifications $\mathcal{PS}$ is the union of the sets of human-readable and machine-readable specifications. Formally, this implies $\mathcal{PS} = \mathcal{HS} \cup \mathcal{MS}$, where $\mathcal{HS}$ is the set of human-readable specifications and $\mathcal{MS}$ is the set of machine-readable specifications. Specification readability can now be formally expressed as:

$$\exists_{a_1 \in \mathcal{HA}} \exists_{a_2 \in \mathcal{CA}} \exists_{q_1 \in \mathcal{HS}} \exists_{q_2 \in \mathcal{MS}} [(a_1, q_1) \in \mathsf{Read} \wedge (a_2, q_2) \in \mathsf{Read} \wedge (a_1, q_2) \notin \mathsf{Read} \wedge (a_2, q_1) \notin \mathsf{Read}] \tag{10}$$

Organizations that take part in the execution of a cross-organizational process may specify their part of the process in a variety of human-readable process specification languages, such as UML activity diagrams, Event-driven Process Chains (EPCs), and the Business Process Modeling Notation (BPMN). The EDSOA design includes the *WorkflowNet2BPEL4WS* [19] component to convert human-readable process specifications to machine-readable ones. The Business Process Execution Language for Web Services (BPEL4WS or BPEL for short) [26] is the de facto standard for machine-readable process specifications and is specifically intended to support cross-organizational processes in a Web services context. The WorkflowNet2BPEL4WS component enables to map the aforementioned variety of source specification languages onto BPEL and to translate those mappings to BPEL code. This BPEL code is then sent to a *computer-based actor*, which is shown in the middle of the EDSOA figure. The formalization of the translation from a human-readable specification to a machine-readable specification can be modeled as follows:

$$\mathsf{Trans} : \mathcal{HS} \rightarrowtail \mathcal{MS} \tag{11}$$

A translation from a human-readable specification $q_1 \in \mathcal{HS}$ to a machine-readable version $q_2 \in \mathcal{MS}$ can be expressed as $\mathsf{Trans}(q_1) = q_2$. For example, q_1 can be a BPMN diagram which can be translated to its machine-readable BPEL equivalent q_2 by using WorkflowNet2BPEL4WS. The 'trans' function is a *partial* function, which implies that a human-readable specification can be translated to its machine-readable BPEL equivalent, and it implies that it is not mandatory for every human-readable specification to be translated to a machine-readable variant in case no computer-based actors are involved. The rounded rectangle with the black arrow at the bottom of the rectangle indicates that a more detailed design of the *event-initiated interactions* part of the architecture can be revealed, which is elaborated next.

The introduced formalisms can be visualized in a diagram to obtain a well-organized graphical representation of the formal model of matching supply and demand of services. In this case, the textual formalisms have been introduced first before generating a visualization in terms of a diagram. Another possible way to create a formal model is to first perform an information analysis to discover relevant objects and relationships between these objects resulting in a conceptual model. Subsequently, this model can be used to derive textual formalisms on its turn. A suitable modeling language

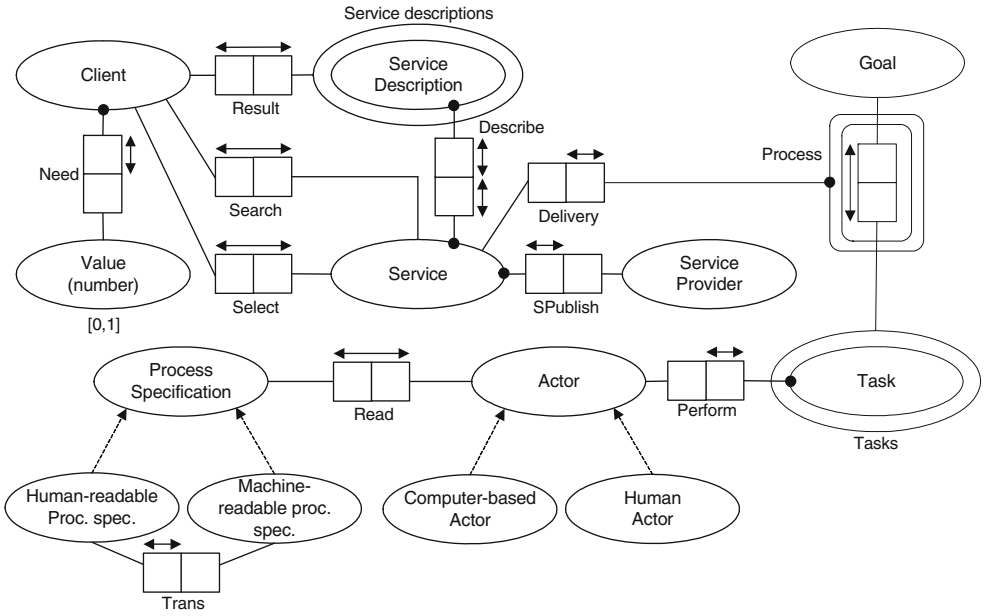

Fig. 2 Object-Role Modeling (ORM) model of service matchmaking

incorporating a formal syntax and semantics is the Object-Role Modeling (ORM) language. Figure 2 shows an ORM model of the presented formalisms that graphically represents the matching of services. ORM is, like UML or ER, a conceptual modeling language that can be used for a variety of modeling purposes, such as the modeling of databases or ontologies [27]. A specific advantage of using ORM is that it has a stable graphical notation, because it is attribute-free [20]. In other words, objects are treated as concepts. This makes ORM immune to changes in the model that cause attributes to be remodeled as objects or relationships. In an ORM model, ovals represent object types (which are the counterparts of classes), while boxes represent relationships between object types. The dashed arrows express the generalization of the specific process specification object types and actor object types. For example, the instances of the *computer-based actor* object type and the *human actor* object type together are equal to the instances of the generalized actor object type. In summary, the five steps that comprise the process of matching supply and demand of services can be related to the formalisms as follows:

1. The function 'SPublish' is the formalization of the publishing of a service by a service provider in a registry.
2. A client's need for a service and, subsequently, the search for a service to fulfill this need is comprised by the 'Need' and 'Search' functions.
3. The return of service descriptions to a client is materialized by the 'Describe' and 'Result' functions.
4. The invocation of a service is reflected by the 'Select' function.
5. Finally, the functions 'Process', 'Delivery', 'Perform', 'Read', and 'Trans' make up the final step, in which a service is executed and in which the result is returned to a client.

Now that the upper part of Fig. 1 has been elaborated, and the rounded rectangle in the middle of Fig. 1 that shows *event-initiated interactions* between actors that participate in the execution of a cross-organizational process is discussed next.

4 Event-initiated interactions

The core of the architecture consists of event-based interactions among loosely coupled service providers. An event can be viewed as a request for input from some actor in the cross-organizational process in order to let the process execution continue [12]. In the context of integrated service delivery, events are used to keep track of the execution of the cross-organizational process as a whole and to understand which actions are demanded from which actor in that process. An example of an event is 'awaiting actor's signature on tax declaration form' when a certain actor needs to take action in a tax declaration process. When an event occurs, actors must determine whether and how to deal with it. If they respond, they must identify one or more activities that are necessary to process the event. This identification is done by event-initiated interactions in the EDSOA. Human and computer-based actors are involved in event-initiated interactions.

Events are generated when an actor participates in a process, i.e., events can be derived from the fulfillment of tasks

by actors as task fulfillment results in changes ultimately leading to integrated service delivery. These changes in the process or 'status updates' are made explicit by the events. The 'request for services' is always the first event that is generated. During the execution of a cross-organizational process, events may arise to indicate that actors participating in the process need to follow up on that event if applicable [28]. This can occur if an actor requires another actor in the cross-organizational process to take action, e.g., if secured Web-based forms need to be filled in for which only a limited number of actors have matching credentials, or when expertise on tax law needs to be applied that is possessed by expert actors. These events are managed and monitored by the EDSOA based on runtime process information. This leads to the flexibility required to offer different varieties of integrated services in order to cope with changing client demands. Human and computer-based actors can act as an event *publisher*, event *subscriber*, or as an event *broker*. An actor can publish an event of a certain type directly to an actor in the process that has subscribed to that event type. For example, an event type can be *tax declaration*. An actor interested in tax declaration events can subscribe to this event type. An event type is instantiated by an event. This can be formalized as follows:

$$\mathsf{EType} : \mathcal{EV} \rightarrow \mathcal{ET} \tag{12}$$

The expression $\mathsf{EType}(e) = t$ shows that an event $e \in \mathcal{EV}$ is of the type $t \in \mathcal{ET}$, where $\mathcal{EV}$ is the set of events and $\mathcal{ET}$ is the set of event types. An actor that publishes an event to another actor that acts as a broker or a subscriber can be modeled as follows:

$$\mathsf{Publish} : \mathcal{AC} \times \mathcal{AC} \rightarrow \mathcal{EV} \tag{13}$$

The expression $\mathsf{Publish}(x, y) = e$ shows that actor $x \in \mathcal{AC}$ publishes an event e to some actor $y \in \mathcal{AC}$, where $\mathcal{AC}$ is the set of actors. It should be noted that actor x can publish this same event to other actors than only to actor y if necessary. For example, actor x can be an immigration officer that requires a digital signature from an employee y working at the embassy to create a residence permit for an immigrant in the cross-organizational process. When a broker is involved, events can be published to this broker that can distribute them to actors that have subscribed to these events. This is an advantage for the publishing actor, because in that case, he does not need to determine anymore to which subscribers an event needs to be sent. The distribution of events from a broker to a subscriber is formalized by means of the *distribute function*, which is equal to the publish function and is therefore not repeated here.

Assume that actor y is subscribed to the event type 'residence permit'. Actor y then receives the published event from actor x and can decide to act on that event. A service call is initiated to the publisher if the actor decides to act on an event. An actor can also decide to act on more than one event by sending more than one service calls. This is formalized by the service call function:

$$\mathsf{Call} : \mathcal{AC} \times \mathcal{AC} \rightarrow \wp(\mathcal{SC}) \tag{14}$$

The expression $\mathsf{Call}(x, y) = S$ shows that actor x initiates one or more service calls $S \subseteq \mathcal{SC}$ to actor y to indicate that actor x will follow up on one or more events, where $\mathcal{SC}$ is the set of service calls and S is a set containing one or more initiated service calls. A publisher reacts on a service call to determine which tasks can be fulfilled in a cross-organizational process by the actor that has sent the service call in order to settle the event. This is formalized by means of the following function:

$$\mathsf{Reacts} : \mathcal{PB} \rightarrow \mathcal{SC} \tag{15}$$

This function shows that a publisher $p \in \mathcal{PB}$ can react on a service call $s \in \mathcal{SC}$, where $\mathcal{PB}$ is the set of publishers. The subscribe function is used to indicate to which event types a subscriber is subscribed:

$$\mathsf{Subscribe} : \mathcal{SR} \rightarrow \wp(\mathcal{ET}) \tag{16}$$

A subscriber $r \in \mathcal{SR}$ that is subscribed to one or more event types can be expressed as follows: $\mathsf{Subscribe}(r) = T$, where $\mathcal{SR}$ is the set of subscribers and T is the set containing one or more events to which a subscriber is subscribed. If a broker is involved, he can *match* events that are published with event types to which an actor is subscribed. This can be formalized by the match function:

$$\mathsf{Match} : \mathcal{BR} \rightarrow \wp(\mathcal{EV} \times \wp(\mathcal{ET})) \tag{17}$$

Assume that $b \in \mathcal{BR}$, $e \in \mathcal{EV}$, and $T \subseteq \mathcal{ET}$, where $\mathcal{BR}$ is the set of brokers. The expression $\mathsf{Match}(b) = \langle e, T \rangle$ denotes that a published event e forms a pair with a set of event types T, because they are matched by broker b. The introduced formalisms related to event-initiated interactions can also be visualized in a diagram to obtain a graphical representation of the formal model. Figure 3 shows an ORM model of the presented formalisms that graphically represents the settlement of event-initiated interactions. Figure 3 shows a *subset* constraint that has been added to make sure that a broker can only use event types to determine a match if there are also actors that have subscribed to these types. A subset constraint is used to indicate that instances of an object type that play a certain role are also part of a set of instances that play another role. The figure also shows three arrows that are drawn from the object type 'Actor' to the object types 'Broker', 'Publisher', and 'Subscriber'. This denotes that there exists a *specialization relationship* between a subtype and a supertype [29], which implies that the instances of the subtype are also instances of the supertype. Each 'Broker', 'Publisher', and 'Subscriber' is also an 'Actor'. To allow for proper specialization, subtypes have to be defined in terms of one or more

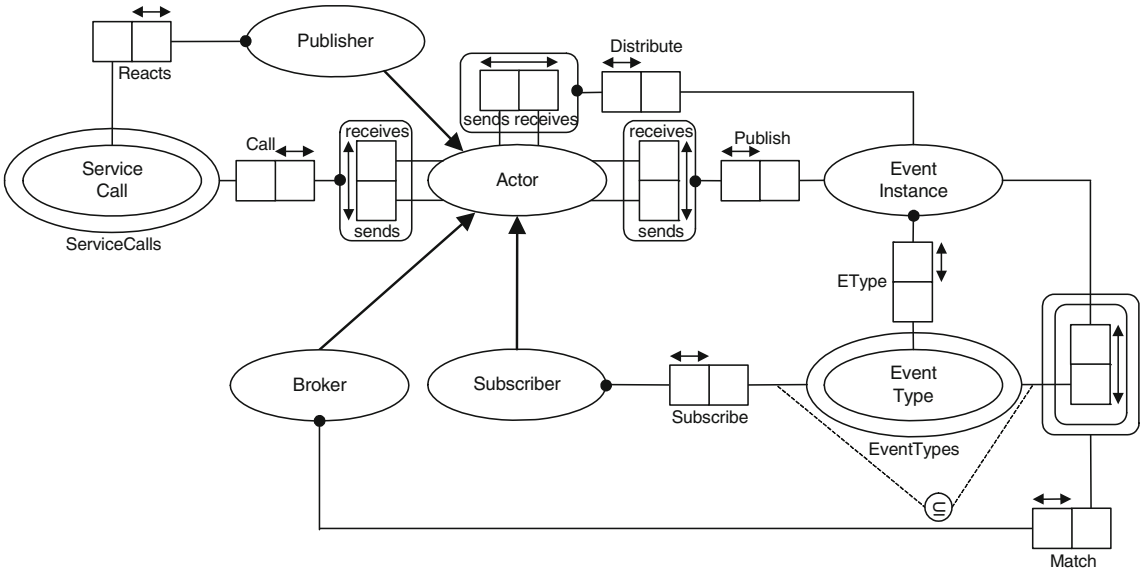

Fig. 3 Object-Role Modeling (ORM) model of event-initiated interactions

of their supertypes. Such a decision criterion is referred to as a *Subtype Defining Rule* [30].

The subtype defining rules can be derived from the ORM model by using the defined functions and sets, but they can also be described by using the Semantics of Business Vocabulary and Business Rules (SBVR) specification that has been published by the Object Management Group (OMG) recently [31]. SBVR defines the meta model for documenting the semantics of business vocabulary and business rules. SBVR is also a human-readable language that at the same time has the full power of formal languages. The use of SBVR in the EDSOA design is further explained in the next section in which the last part of the EDSOA design is explained that is related to human-readable and ontology-based process information. There are four font styles that can be used for SBVR verbalizations that have a formal meaning [31]. The term font is used for a designation for a noun concept (other than an individual concept), one that is part of a vocabulary being used or defined. Terms in SBVR are verbalizations of objects or classes. The name font is used for a designation of an individual concept. Names in SBVR are verbalizations of class instances. The *verb* font is used for designations of relationships between objects. The 'keyword' font is used for linguistic symbols used to construct statements, i.e., the words that can be combined with other designations to form statements and definitions. Thus, when using sets and functions, the subtype defining rule for 'Broker' is:

$$\forall_{b \in \mathcal{BR}} \exists_{x \in \mathcal{AC}} [\mathsf{Publish}(b, x) \wedge \mathsf{Publish}(x, b)] \tag{18}$$

This subtype defining rule can be explained as follows. The expression $\mathsf{Publish}(b, x)$ shows that broker $b \in \mathcal{BR}$ publishes an event instance to some other actor $x \in \mathcal{AC}$, where $\mathcal{BR}$ is the set of brokers and $\mathcal{AC}$ is the set of actors. A broker is also an actor as $\mathcal{BR} \subseteq \mathcal{AC}$. The expression $\mathsf{Publish}(x, b)$ shows that some actor x receives an event instance from broker b. When interpreting the quantifiers on the leftmost side of the subtype defining rule together with the logical expressions, it can be concluded that each broker sends at least one event instance but also receives at least one event instance from some actor. Its SBVR counterpart can then be described as follows:

Each broker is an actor that *receives* at least one event instance and *sends* at least one event instance

The subtype defining rule for 'Publisher' is:

$$\forall_{p \in \mathcal{PB}} \exists_{x \in \mathcal{AC}} [\mathsf{Call}(x, p) \wedge \mathsf{Publish}(p, x)] \tag{19}$$

The expression $\mathsf{Call}(x, p)$ shows that some actor $x \in \mathcal{AC}$ sends a service call to publisher $p \in \mathcal{PB}$, where $\mathcal{PB}$ is the set of publishers. A publisher is also an actor as $\mathcal{PB} \subseteq \mathcal{AC}$. The expression $\mathsf{Publish}(p, x)$ shows that publisher p sends an event instance to actor x. When interpreting the quantifiers on the leftmost side of the subtype defining rule together with the logical expressions, it can be concluded that each publisher receives at least one service call but also sends at least one event instance to some actor. Its SBVR counterpart can be described as follows:

Each publisher is an actor that *receives* at least one service call and *sends* at least one event instance

Finally, the subtype defining rule for 'Subscriber' is:

$$\forall_{r \in \mathcal{SR}} \exists_{x \in \mathcal{AC}} [\mathsf{Call}(r, x) \wedge \mathsf{Publish}(x, r)] \tag{20}$$

The expression $\mathsf{Call}(r, x)$ shows that some subscriber $r \in \mathcal{SR}$ sends a service call to some actor $x \in \mathcal{AC}$, where $\mathcal{SR}$ is the set of subscribers. A subscriber is also an actor as $\mathcal{SR} \subseteq \mathcal{AC}$. The expression $\mathsf{Publish}(x, r)$ shows that actor x sends an event instance to subscriber r. When interpreting the quantifiers on the leftmost side of the subtype defining rule together with the logical expressions, it can be concluded that each subscriber sends at least one service call but also receives at least one event instance from some actor. Its SBVR counterpart can be described as follows:

Each subscriber is an actor that *sends* at least one service call and *receives* at least one event instance

The ORM model and the textual formalisms in this section comprise a detailed design for event-initiated interactions as part of the architecture, which is coupled to the remaining components of the EDSOA that are shown at the bottom of Fig. 1. These components indicate the involvement of an ontology that forms a basis for the delivery of process information to actors and are discussed next.

5 Ontology-based process information

In order to act on an event, an actor needs to interpret process-related information in order to successfully fulfill his part of the cross-organizational process. Successful integrated service delivery can be realized if all actors who participate in a cross-organizational process understand the semantics of the cross-organizational process leading to successful process enactment. An *ontology* can be used to realize an agreed understanding among actors of such process information, and it is able to capture this information highly independently of any particular event or activity [20]. In this case, an ontology makes it possible to identify essential concepts, relationships between concepts, and process constraints in cross-organizational process fulfillment. An ORM model of such an ontology is shown in [28] containing eight central concepts, constraints, and relationships between these concepts that resulted from knowledge gained from a case study in which integrated services were offered to immigrants. Immigrants are people who want to migrate to another country to live and work there.

The ontological concepts of *role*, *actor*, *service*, *process*, *resource*, *service provider*, *event*, and *EDSOA* are included [28]. The ontology's *role* concept denotes a specification of an actor enactment. An actor enacts a role during process performance or, at a more granular level, task performance. An employee of the Immigration and Naturalization Service playing the role of registrar is an example of such an actor in a public organization. The immigrant is another example of an actor in the public domain. An actor can perform an organizational *process* and uses *services* that are required for organizational processes. For example, a registrar who registers a new residence permit for an immigrant who comes to the Netherlands might perform a process called 'grant residence permit'. During this performance, the registrar uses a service to create a new residence permit registration. Organizations also have *resources* they use to perform processes. A human actor and a computer-based actor is a specialization of the resource concept, because an actor works at an organization and is required for process performance. The *event* and *EDSOA* concepts are also part of the ontology, because events occur during process fulfillment and are part of the EDSOA [28].

A 'request for process information' can be submitted by an actor to the *ontology modeler* component which is shown in Fig. 1. This component is implemented by the *DogmaModeler* ontology engineering software tool, which can be used to model, browse, and realize manageable ontologies based on ORM as the modeling language for the ontology. For more information on DogmaModeler, see [32]. Human actors and computer-based actors that participate in a cross-organizational process need to interpret this ontology to understand the process information when reacting on an event. DogmaModeler can translate the ORM ontology to human-interpretable respectively computer-interpretable process information. The translation of the ORM ontological model to human-understandable information is based on the aforementioned Semantics of Business Vocabulary and Business Rules (SBVR) specification [31]. SBVR improves readability and prevents ambiguous interpretations among human actors who need to interpret process information, because of its readability and expressiveness. The translation of the ORM model to computer-interpretable information is based on the Web Ontology Language (OWL) because this facilitates greater machine interpretability of Web content than that supported by, e.g., XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics [33].

This can be explained in the context of the Semantic Web. The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web. OWL is on top of a growing stack of W3C recommendations related to the Semantic Web, which are the eXtensible Markup Language (XML), the Resource Description Framework (RDF), and RDF Schema [33]. In more detail, it is stated that: "XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents. XML Schema is a language for restricting the structure of XML documents and also extends XML with data types. RDF is a data model for objects ('resources') and relations between them, provides a simple semantics for this data model, and these data models can be represented in an XML syntax. RDF Schema

is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization hierarchies of such properties and classes. OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g., disjointness), cardinality (e.g., 'exactly one'), equality, richer typing of properties, characteristics of properties (e.g., symmetry), and enumerated classes" [33].

Actual instances of the objects (or classes) that are part of the ontology and the actual relations between objects for some cross-organizational process can be retrieved from the *ontology database*. Based on MySQL Server, this database has the ORM ontology as its relational database model, enabling a query-based retrieval of the object instances (or class members) that are created during execution of the cross-organizational process. These data can be used for successful execution of some cross-organizational process. Examples of SBVR descriptions and outputs produced by the DogmaModeler are shown and explained in the next section to make clear how process information is represented for human actors and computer-based actors, respectively.

6 Verbalizations of process information

The DogmaModeler produces output reflecting process information in the OWL specification or the SBVR specification. Figure 1 already shows a glimpse of an example OWL specification that is related to the delivery of services to immigrants. Figure 4 verbalizes possible concepts, relationships, and constraints in OWL. These concepts, relationships, and constraints have been derived from the aforementioned ontology for cross-organizational process fulfillment. The OWL representation shows that there are several *classes* prescribed by the ontology that are important to take into account for computer-based actors participating in a cross-organizational process for integrated service delivery. These classes are actually the eight ontological concepts mentioned in Sect. 5. The relationships that these classes have with each other are expressed by means of *object property* tags. For instance, the relationship 'enacts' denotes that an actor can *enact* a role. Process constraints are represented by the *restriction* tags. For example, the process constraint shown in Fig. 4 expresses that at least one event should be coordinated by the EDSOA. Example *class members* of the classes that are part of the OWL representation of cross-organizational requirements are shown in Fig. 5. Class members such as these are stored in the ontology database of the EDSOA. Computer-based actors that need to act on an event can gather additional process information which they can use to successfully fulfill their part of the process. The examples of Fig. 5 are related to a cross-organizational process for the delivery of services to immigrants. Thus, these OWL representations will assist a computer-based actor to understand the concepts, relationships, and constraints that are important for cross-organizational process fulfillment. Because actors are autonomous in the sense that they are able to act on events based on their own decisions, it is no guarantee that actors will put this process information to good use. However, communicating this process information to actors enables to steer and guide the way they act on events and, in the end, will positively influence service delivery to the client.

```
<owl:Class rdf:ID = "Resource" />
<owl:Class rdf:ID = "Actor">
  <rdfs:subClassOf rdf:resource = "#Resource" />
</owl:Class>
<owl:Class rdf:ID = "Role" />
<owl:Class rdf:ID = "Service" />
<owl:Class rdf:ID = "Process" />
<owl:Class rdf:ID = "Organization" />
<owl:Class rdf:ID = "Event" />
<owl:Class rdf:ID = "Architecture" />
<owl:ObjectProperty rdf:ID = "Enacts">
  <owl:inverseOf rdf:resource= "IsEnactedBy" />
  <rdfs:domain rdf:resource = "#Actor" />
  <rdfs:range rdf:resource = "#Role" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID = "Uses">
  <owl:inverseOf rdf:resource= "IsUsedBy" />
  <rdfs:domain rdf:resource = "#Actor" />
  <rdfs:range rdf:resource = "#Service" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID = "Produces">
  <owl:inverseOf rdf:resource= "IsProducedBy" />
  <rdfs:domain rdf:resource = "#Actor" />
  <rdfs:range rdf:resource = "#Event" />
</owl:ObjectProperty>
....
<owl:Restriction>
  <owl:onProperty rdf:resource= "#Coordinates.Event" />
  <owl:minCardinality rdf:datatype=
  "&xsd;nonNegativeInteger">1</owl:minCardinality>
</owl:Restriction>
```

Fig. 4 A partial OWL representation of cross-organizational process information

```
<Actor rdf:ID="John" />
<Role rdf:ID="MigrationOfficer" />
<Service rdf:ID="ResidencePermit" />
<Process rdf:ID="RequestResidencePermit" />
<Organization rdf:ID="ImmigrationService" />
<Event rdf:ID="ResidencePermitSent" />
<Architecture rdf:ID="ImmigrantSOA" />
```

Fig. 5 Examples of class members in OWL for some cross-organizational process

Specifying the ontology of the process requirements in SBVR enables human actors to read process information. Table 1 provides a list of items and examples of how this information can materialize in some cross-organizational process. From the examples shown in Table 1 can be derived which services are used by which actors, which services are integrated, which services are required for which processes, and which services are eventually delivered. This process information can be useful for those human actors that wish

Table 1 Human-readable process information in SBVR

Process information in SBVR	Example
Service …*is used by* actor …	Service residence permit *is used by* actor John
Service …*is integrated by* architecture …	Service residence permit *is integrated by* architecture expat SOA
Service …*is required for* process …	Service residence permit *is required for* process request residence permit
Service …*is offered by* organization …	Service tax declaration *is offered by* organization tax admin

to know these details of services that are currently delivered to clients.

After having elaborated the distinct parts of the EDSOA design as shown in Fig. 1, being the service matchmaking part, the event-initiated interactions part, and the ontology-based process information part, an illustration of the EDSOA design in a practical scenario is shown hereafter to understand how the EDSOA can be exploited in practice.

7 Illustrating the EDSOA design

The presented models and formalisms that comprise the EDSOA design can be materialized in a scenario to illustrate how ISD is realized with the support of the EDSOA. The presented scenario concerns an application for a temporary residence permit by an immigrant who has moved to the Netherlands and who has just acquired a labor contract from his employee. Recall from Sect. 3 that the first step in the service matchmaking part concerns the need for services of a client. Assume that an immigrant c that is in the set of clients $c \in \mathcal{CL}$ has a very urgent need for a service. Using the service need function from Sect. 3, this can be expressed as $\mathsf{Need}(c) = 1$. This need for a service by the immigrant relates to a service to supply a temporary residence permit. The service to provide a temporary residence permit involves multiple organizations that need to collaborate in a cross-organizational process to provide the permit to the immigrant.

The five-step process to match supply and demand of such a service as mentioned in Sect. 3 can be illustrated in this context. The service s to apply for a temporary residence permit online that is part of the set of services $\mathcal{SE}$ should be published by a service provider $p \in \mathcal{SP}$, which is the Immigration and Naturalization Service (INS) in this case and which is part of the set of service providers. The publishing of this service is expressed as $\mathsf{SPublish}(s) = p$. Then, the immigrant c searches for this service s by using the Web-based portal that forms the interface between the immigrant and the service provider. This is expressed as $(c, s) \in \mathsf{Search}$. The temporary residence application service s is described as $\mathsf{Describe}(s) = d$, and the expression $(c, D) \in \mathsf{Result}$ shows that service descriptions $D \subseteq \mathcal{SD}$ are returned as a result. If the search query entered by the immigrant is, for example, identical to 'I need a temporary residence permit' then the service description 'request temporary residence permit' is contained in the list D. The immigrant wants to make use of this service and selects it which is expressed as $(c, s) \in \mathsf{Select}$.

A cross-organizational process is initiated to supply the service to request a temporary residence permit now that the immigrant has selected it. Figure 6 shows the BPMN diagram of this process for requesting a temporary residence permit in the Netherlands. The expression $(T, g) \in \mathsf{Process}$ shows that the tasks $T \subseteq \mathcal{TA}$ shown in the BPMN diagram collectively realize the goal $g \in \mathcal{GL}$ to supply a temporary residence permit. The figure shows that four entities participate in the cross-organizational process to deliver a temporary residence permit. These are the embassy, the immigrant, the immigrant's employer, and the INS. The processes carried out by these organizations altogether form the cross-organizational process. The process that is performed by the embassy and that finalizes a temporary residence permit is expressed as $\{X, x\}$, where X stands for the two tasks shown in the embassy swimlane and x is the goal 'finalize temporary residence permit'. Both tasks require human involvement, as expert knowledge is still needed to determine whether or not someone is eligible to receive a temporary residence permit.

The immigrant wants to collect his temporary residence permit, which is his goal y. The five tasks shown in the immigrant swimlane make up his part of the cross-organizational process which is expressed as $\{Y, y\}$. Each of these tasks, except for the task 'collect temporary residence permit', involves an automated part. The receipt of the labor contract, the sending of the employee data, the receipt of the positive advice, and the request for a temporary residence permit can all be done digitally. The temporary residence permit itself still needs to be collected at the embassy in person.

The employer mediates between the immigrant and the INS, which has as the goal to send a positive advice from the INS to the immigrant. The expression $\{V, v\}$ reflects the

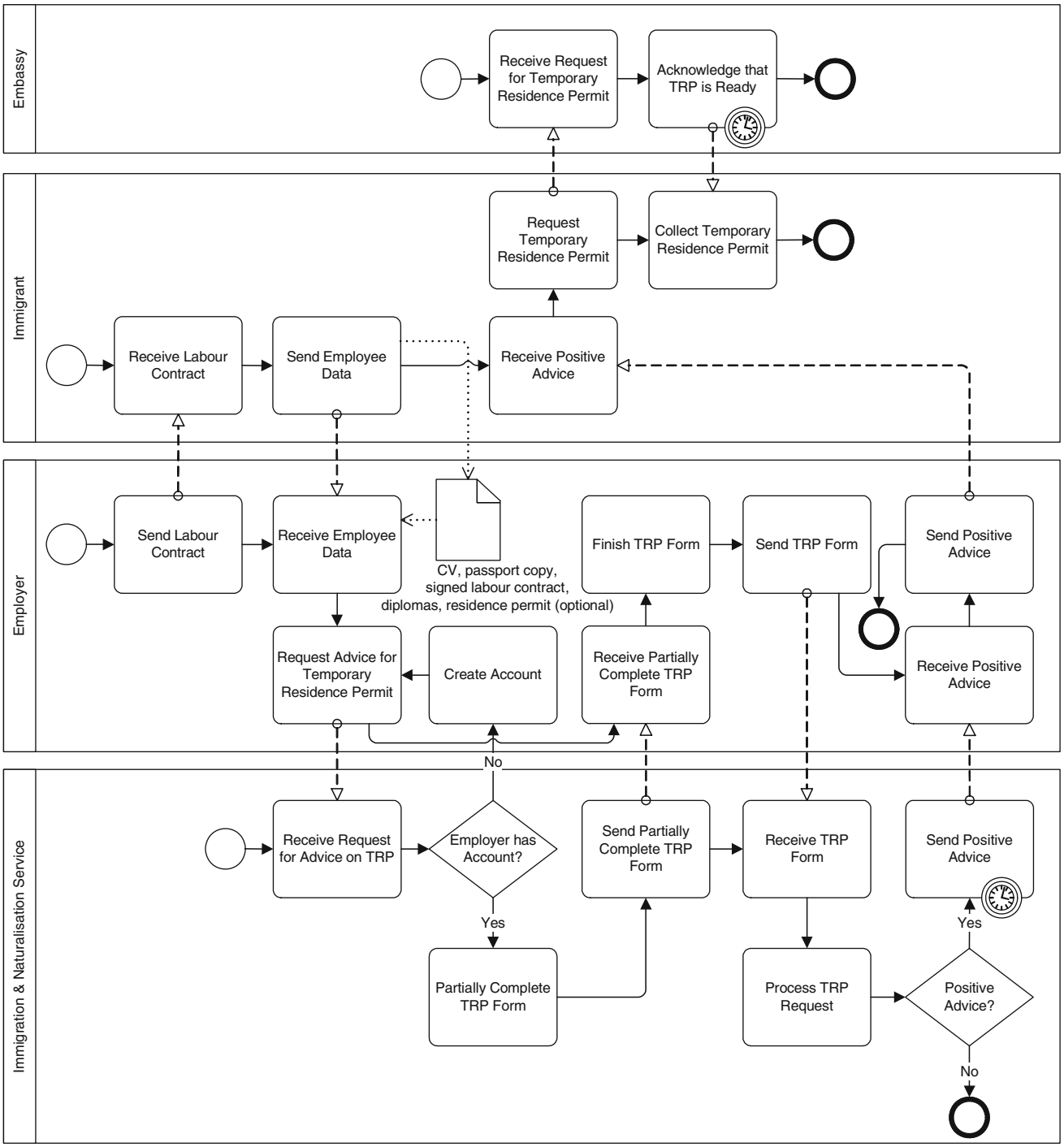

Fig. 6 Process to request a temporary residence permit in the Netherlands

nine tasks that are carried out by the employer which have as the goal to send a positive advice from the INS to the immigrant. All information that is collected and sent as part of the execution of these nine tasks involve automation as the information can be exchanged digitally. This includes digital copies of the documents related to the employee data. The creation of an employer account at the INS can also be done on the Web site of the INS by the employer. The employer needs to be involved specifically to complete the temporary residence permit form.

The tasks to process the request for a temporary residence permit that is done by the INS can be depicted as $\{W, w\}$. The tasks that still need specific expert human involvement are the tasks to partially complete the temporary residence permit

Table 2 Example events in the process to supply a temporary residence permit

Event instance	Event type	Actor
Request labor contract	Labor contract	Immigrant
Labor contract sent	Labor contract	Employer
Labor contract received	Labor contract	Immigrant
Request employee data	Employee data	Employer
Employee data sent	Employee data	Immigrant
Employee data received	Employee data	Employer
Request advice for TRP	Temporary residence permit	Employer
Advisory request received	Temporary residence permit	INS employee
Employer account check sent	Employer account	INS employee
Partially completed TRP form sent	Temporary residence permit	INS employee
…	…	…

form and the decision process to come up with a positive or negative advice on providing a permit. The delivery of the temporary residence permit service s can now be expressed as $\mathsf{Delivery}(\{X, x\}, \{Y, y\}, \{V, v\}, \{W, w\}) = s$. The immigrant and the actors that are employed at one of the three organizations perform the tasks of the cross-organizational process. For example, a human resource officer $a \in \mathcal{AC}$ working at the immigrant's employer performing the task $t \in \mathcal{TA}$ to send a labor contract is expressed as $\mathsf{Perform}(t) = a$. As the HR officer is a human actor, he will read a process specification $q_1 \in \mathcal{PS}$ just like the one in Fig. 6. Using the WorkflowNet2BPEL4WS component of the EDSOA, the BPMN specification q_1 can be translated to, for example, a machine-readable BPEL version q_2 which is expressed as $\mathsf{Trans}(q_1) = q_2$. This illustrates that the inclusion of this translation component in the EDSOA provides the possibility to deliver readable process specifications for human and computer-based actors involved in the cross-organizational process.

Events are generated during execution of the cross-organizational process to supply the temporary residence permit to the immigrant. Table 2 shows a partial list of example events during execution of this process including the type of event and which actor publishes an event. The event type function shown in Sect. 4 can be applied to express which event instance is of which type. For instance, if $e \in \mathcal{EV}$ is the event 'request labor contract', it is of the type $t \in \mathcal{ET}$ where t is the type 'labor contract'. Using the event type function, this can be expressed as $\mathsf{EType}(e) = t$. The expression $\mathsf{Publish}(x, y) = e$ shows that an immigrant $x \in \mathcal{AC}$ publishes the event 'request labor contract' to his employer $y \in \mathcal{AC}$. When a broker is involved to which an immigrant can publish events, they can be distributed to actors that have subscribed to the event type 'labor contract'. This subscriber can be the employer that is able to process the request for a labor contract. The expression $\mathsf{Distribute}(z, y) = e$ shows that a broker $z \in \mathcal{AC}$ distributes the event 'request labor contract' to the employer y. The employer can be subscribed to those event types in the process that require his involvement. For example, the employer can be subscribed to the event types 'labor contract', 'employee data', 'employer account', and 'temporary residence permit'. This can be expressed as $\mathsf{Subscribe}(y) = T$, where these event types are part of the set T.

The employer can initiate service calls $S \subseteq \mathcal{SC}$ to indicate on which event instances he will follow up, which is expressed by $\mathsf{Call}(y, z) = S$. For example, the employer can sent a service call $s \in \mathcal{SC}$ to indicate that he will follow up on the event instance 'request labor contract' that was sent by the immigrant. As the employer is subscribed to event instances of the type 'labor contract', he receives these kind of event instances. The publisher of the event instance 'request labor contract' can make clear what is required to do by the employer once the employer has indicated that he would like to follow up on that event instance. This reaction of the immigrant x as the event publisher can be expressed as $\mathsf{Reacts}(x) = s$. The matching of event instances that are published with event types to which an actor is subscribed can be done by a broker. The expression $\mathsf{Match}(z) = \langle e, T \rangle$ shows that broker z matches the event instance 'request labor contract' with the event types in the set T to which the employer is subscribed. There is a match as the employer is subscribed to the event type 'labor contract', which means that the event instance is distributed to the employer. These event-initiated interactions as part of the EDSOA design support the fulfillment of the cross-organizational process to deliver the requested temporary residence permit by allowing insight into the interactions between the involved actors in the process.

To assist an actor in following up on an event by performing a part of the cross-organizational process, the actor can submit a 'request for process information' to the ontology modeler component shown in Fig. 1 in order to retrieve process information. Once a service request has been made by a client and a cross-organizational process has been started to deliver that service data are stored in the ontology database about the process enactment, this adds an additional possibility to share process information among the actors involved in the cross-organizational process supported by the EDSOA. For example, the embassy that is involved might want to know more about the service the immigrant 'John Doe' is requesting, where it is possibly integrated with other services, for which process the service is required and what actors have already produced which events in order to track the status of the process and service delivery as a whole.

Table 3 Example of requested human-readable process information in SBVR

Process information in SBVR	Example
Service …*is used by* actor …	Service temporary residence permit *is used by* actor John Doe
Service …*is integrated by* architecture …	Service temporary residence permit *is integrated by* architecture EDSOA@INS
Service …*is required for* process …	Service temporary residence permit *is required for* process request temporary residence permit
Actor …*produces* event …	Actor INS employee *produces* event Partially completed TRP form sent

Table 3 shows this example output of human-readable process information in SBVR. Now that the EDSOA has been illustrated as a whole in a practical scenario and it will now be determined which criteria of the proposed EDSOA design are most important for successful adoption by service providers in the following evaluative part.

8 Evaluation of the EDSOA design

In design science, the artifact needs to be evaluated based on the tasks it should perform, in our situation the creation of ISD. These tasks are decomposed in various criteria and evaluated in a workshop. These criteria are selected from literature [34] with in the back of the mind a participant's predisposition toward adopting the architecture for the organization he represents. A discussion of the possible criteria and reaching consensus about the criteria is explicitly part of the workshop setup in order to: (1) find out their interpretations of the criteria, (2) determine to what extent the participants agree with the criteria, and (3) collect their own criteria in addition to the criteria selected from the literature as well. The EDSOA evaluation has been carried out by conducting a one-day workshop, during which a total of eight employees from five different organizations judged the design. The five different Dutch organizations comprised the Social Insurance Bank, the Tax and Customs Administration, the Institute for Employee Insurances, the ICT Administration Organization, and the Municipality of Enschede, the Netherlands.

A Group Support System (GSS) [35] has been used to facilitate the evaluation process. A GSS is a computer-based meeting system that supports collaboration between the workshop participants. The system offers the benefits that participants can provide content in parallel and that they remain anonymous. A GSS also structures and guides the workshop and minutes are made available automatically after the workshop for data analysis purposes. The eight participating employees were two IT architects, an IT consultant specialized in integrated service delivery, two senior advisors in information provisioning, a project leader of a project on computer-supported service delivery, a requirements engineer, and an advisor in service provisioning. The evaluation of the EDSOA is in fact an application of the *multiple-attribute value theory* (MAVT) [36]. MAVT can be used to address problems that involve a set of criteria that have to be evaluated on the basis of conflicting objectives. This makes MAVT a suitable theory to determine which criteria of the EDSOA are seen as most important such that service providers will adopt it.

8.1 Workshop design

Based on the MAVT approach, the workshop comprised five phases: (1) a presentation of the EDSOA design, (2) presentation and discussion of possible evaluation criteria, (3) reaching consensus about the criteria to be used, (4) scoring of criteria, and (5) ranking of criteria. These phases can be further explained as follows. In the first phase, the detailed EDSOA design discussed in Sect. 3 was shown and explained to the participants. Second, a first evaluation round was conducted based on 15 criteria that should be taken into account when implementing the EDSOA to support service providers. The criteria are designed to indicate a participant's predisposition toward adopting the architecture for the organization he represents, and they indicate the willingness of an organization to employ the architecture [37]. The criteria are based on criteria for evaluation of adaptive and adaptable systems found in [34] and customized for evaluation of the EDSOA. They are listed below together with their explanations.

Architecture acceptance: The extent to which an organization requires an architecture that supports the realization of ISD.

Compatibility: The extent to which the architecture fits the organizational infrastructure.

Expected usefulness: The extent to which it is expected that the architecture poses advantages for the organization.

Expected ease of implementation: The extent to which it is expected that an organization can easily implement the architecture.

Effectiveness: The extent to which an organization expects that the architecture will provide support for ISD.

Willingness to use: The willingness of an organization to use the architecture.

Trust in technology: The extent to which an organization trusts the technology to support ISD based on the architecture.

Trust in partners: The extent to which an organization trusts other parties to collaborate with to realize ISD.

Appreciation by management: The extent to which an organization's management agrees with deployment of the architecture.

Experience with changes: The extent to which an organization has experience with adoption of computer-based support for ISD.

Time availability: The available time of an organization to implement the architecture.

Emotional involvement: The attitude of an organization toward adoption of the architecture.

Willingness to change: The willingness of an organization to cooperate with changes needed to adopt the architecture.

Employee satisfaction: The extent to which employees of an organization will be satisfied with the architecture that supports them in their work to realize ISD for clients.

Employee participation: The extent to which employees are involved during the process of implementing the architecture in an organization.

The criteria were shown on laptops that were handed out to the workshop participants, and the GSS software enabled them to provide comments for each criterion that were shown to all participants. The comments were used as input for the third phase of the workshop to achieve consensus among the participants about the list of criteria. The participants determined that three more criteria were needed: coherence with related initiatives that support ISD, ease of maintenance after implementing the architecture, and the extent to which the architecture fits organizational standards. The first additional criterion was introduced because the participants reasoned that it is important to identify related initiatives within the organization in order to prevent from reinventing the wheel and in order to profit from lessons learned in other initiatives. The second additional criterion was brought up because the participants mentioned that IT maintenance is always a costly phenomenon for organizations and, therefore, maintenance should be as easy as possible. The third criterion was brought up because implementation efforts are reduced if the standards used as part of the EDSOA will fit with standards that are already used in organizations. Using their laptops with the GSS software, the participants *scored* the criteria during phase four on a five-point Likert scale. Afterward, the GSS computed the total average score for each criterion.

This reflected that the highest total average score of 4.5 out of 5 was awarded to the criterion *expected usefulness*. This means that the participants expect that the usefulness of the presented EDSOA should be made clear whether the organizations they represent are going to adopt it. A total average score of 4.38 was awarded to the criterion *organizational standards*. This implies that the participants stress the importance that the EDSOA fits existing organizational standards. The criteria *ease of maintenance* and *coherence with related initiatives* ended up ex aequo on a third place, i.e., once implemented, the participants think that the architecture has to be easy to maintain and it should have coherence with other organizational initiatives that support service delivery. The fifth and final phase of the workshop consisted of *ranking* the criteria. The most important criterion is placed on position one, while the least important criterion is placed on the last position. The results for the ranked criteria have been calculated by using several functions, which are discussed in the following part. In our case, there are 18 criteria, comprising of the original 15 shown in the list above plus the 3 additional ones introduced by the participants in phase three. Assume that the set $C \subseteq \mathcal{CR}$ reflects the total number of criteria identified during this workshop, where $\mathcal{CR}$ is the set of evaluation criteria for EDSOA designs in general. The expression $|C| = 18$ shows that C contains 18 criteria. It should be noted that the pipe symbols denote the cardinality. In other words, they are used for counting the number of elements in a set. Next, the frequency function is needed to determine how many times a criterion has been ranked at which position:

$$\mathsf{Freq} : \mathcal{CR} \times \mathbb{N} \to \mathbb{N} \tag{21}$$

The results showed that the criterion *expected usefulness* has been placed on position one by five participants. Using the frequency function, this can be expressed as: $\mathsf{Freq}(c, 1) = 5$, where $c \in C$ represents the criterion *expected usefulness*. Subsequently, the total number of participants needs to be determined. In our case, there are 8 participants. Assume that the set $P \subseteq \mathcal{PR}$ reflects the total number of workshop participants, where $\mathcal{PR}$ is the set of workshop participants in general. The expression $|P| = 8$ shows that P contains 8 participants. Finally, the rank function can calculate the end result for each ranked criterion:

$$\mathsf{Rank} : \mathcal{CR} \times \wp(\mathcal{CR}) \to \mathbb{N} \tag{22}$$

The expression $\mathsf{Rank}(c, C)$ is the computation of the final ranking result for a ranked criterion c. To understand how this works, the definition of the rank criterion can be given as follows:

$$\mathsf{Rank}(c, C) \triangleq \sum_{i=1}^{|C|} \frac{\mathsf{Freq}(c, i) \cdot (1 + \mathsf{Total}(C) - i)}{|P|} \tag{23}$$

This definition can be explained by calculating the final ranking result for the criterion *expected usefulness*. This criterion

has been placed five times on position one, one time on position two, one time on position three, and one time on position fourteen. The higher a criterion is ranked by a participant, the higher the score for that criterion. Ranking a criterion on position one results in 18 points in this case (note that this is equal to $\mathsf{Total}(C)$). Ranking it on the last position results in the awarding of only 1 point. The score is multiplied by the number of participants. For example, if five participants rank a criterion on position one, this means that the score of 18 is multiplied by 5 in that case. Subsequently, the total score is divided by the total number of participants. The rank function can now be used to calculate the final ranking score for the *expected usefulness* criterion:

$$\mathsf{Rank}(c, C) = \frac{5 \cdot 18 + 1 \cdot 17 + 1 \cdot 16 + 1 \cdot 5}{8} = 16$$

This implies that a final ranking score of 16 is awarded to this criterion, which makes it the criterion that is ranked highest by the participants. Again, the expected usefulness of the EDSOA seems very important to demonstrate in order to successfully employ the architecture at the participants' organizations. The criterion *trust in technology* ended up second with a score of 13.38 and the criterion *ease of maintenance* ended up third with a score of 12.63 points. This shows that, when compared with the other criteria, the participants believe that trust in the technology to support integrated service delivery based on the EDSOA and ease of maintenance is considered very important.

8.2 Workshop results

Several findings can be noticed when studying the workshop results. The participants decided to introduce three more criteria during the third phase of the workshop, which they thought were omitted in the list. These were coherence with other initiatives, ease of maintenance, and fit with organizational standards. It turned out that these criteria were regarded as very important by the participants when all criteria were scored in the next phase, because the newly introduced criteria ended up in the top three, with *coherence with other initiatives* and *ease of maintenance* sharing third place. However, the criterion of *expected usefulness*, that already existed in the original list of criteria presented in phase two, received the highest score. This criterion also ended up in first place when all criteria were *ranked* instead of *scored* in the final phase. Table 4 shows that the top four of the lists of ranked criteria and scored criteria contain the same criteria, except for *coherence with other initiatives*. This criterion is only ninth in the list of ranked criteria, whereas expected ease of implementation is ninth in the list of scored criteria. Thus, the four criteria that consequently represent the top four of both lists of criteria are expected usefulness, organizational standards, trust in technology, and ease of maintenance. This implies that when implementing the EDSOA to support service providers it should be useful, fit with organizational standards, be based on trusted technology, and be easy to maintain. We found that the participants were mostly interested in whether the EDSOA would do what it had been designed to do, regardless of the implementation. From this, it can be concluded that the participants viewed the EDSOA design from a non-technical angle, namely an angle that is concerned with the suitability of the EDSOA for use in the organizations where the participants work. Subsequently, a comparison of this research with related work will be presented now that the distinct parts of the EDSOA design have been presented, illustrated, and evaluated as well.

Table 4 List of scored and ranked criteria

#	Scored criteria	Ranked criteria
1	Expected usefulness	Expected usefulness
2	Organizational standards	Trust in technology
3	Ease of maintenance and coherence with other initiatives	Ease of maintenance
4	Trust in technology	Organizational standards
5	Trust in partners	Architecture acceptance
6	Architecture acceptance	Appreciation by management
7	Willingness to use	Expected ease of implementation
8	Willingness to change	Trust in partners
9	Expected ease of implementation	Coherence with other initiatives
10	Appreciation by management	Time availability
11	Time availability	Willingness to change
12	Compatibility	Compatibility
13	Experience with changes	Experience with changes
14	Emotional involvement	Emotional involvement
15	Employee satisfaction	Willingness to use
16	Employee participation	Employee participation
17	Effectiveness	Effectiveness
18	–	Employee satisfaction

9 Related work

The research presented in this paper relates to other work that has to do with three architecture styles: (1) the traditional service-oriented architecture (SOA), (2) the event-driven architectural style (EDA), and (3) the hybrid event-driven service-oriented architectural model (EDSOA). Related work to integrated service delivery is also specifically mentioned in this section.

9.1 Service-oriented architecture

According to the 6 guidelines in the OASIS Reference Model for SOA [6], a SOA (1) is expected to have entities that can be

identified as services, (2) is able to identify how visibility is established between service providers and clients, (3) is able to identify how interaction is mediated, (4) is able to identify how the effect of using services is understood, (5) has descriptions associated with services and is able to identify the execution context required to support interaction, and (6) should identify how policies are handled and how contracts may be modeled and enforced. These 6 guidelines can be successfully applied to our EDSOA model as follows. (1) Entities that can be identified as services are included in the UDDI registry included in our design. (2) Visibility between service providers and clients is established by matching supply and demand of services in the Web-based portal. (3) Interactions between the different parties involved are mediated by the matchmaking procedure explained in Sect. 3 and the event-driven interactions explained in Sect. 4. (4) The effects of using services are understood by applying the *service need* function introduced in Sect. 3. The service need is equal to 0 if both the services are supplied by the provider and also used by the client. The service need is equal to 0 if both the services are supplied by the provider and also used by the client. The service need is equal to 1 if relevant services have been offered nor used by the client. (5) Service descriptions are also stored in the UDDI registry of the EDSOA design. (6) Lastly, the handling of policies and contracts is realized by communicating process information to actors in the cross-organizational process.

A very recent project to realize a SOA that offers Web services to city officials, officers, citizens, and tourists is discussed in [7]. Specifically, a SOA for city portals is proposed to design, integrate, and streamline city systems and applications. When interpreting the SOA design, however, it is not made clear what the role is of computer-based actors and human actors that perform processes leading to service provisioning. An advantage of the largely user-oriented approach used to design the architecture shown in [7] is that *user preferences* can be taken into account when interacting with the user via the portal. This can lead to an improvement of service delivery when interacting with returning users. Events that are generated during the process to deliver services are not considered in traditional SOA styles but have been introduced in the EDA style. Related work in the field of EDA is discussed next.

9.2 Event-driven architecture

Current research on the EDA style shows that a lot of insights into the functioning of an event-initiated mechanism have been developed, but the combination of such a mechanism with other core parts that make up an infrastructure for integrated service delivery is less discussed. These core parts are the service matchmaking part, the ontology for providing process information for computer-based actors and human actors, and providing descriptions of cross-organizational processes to actors that generate and react on events. For example, relevant work describing the functioning of an EDA is provided in [38, 39]. By means of our EDSOA design, however, we have made an attempt to also centralize the actors that have to deal with the mechanisms that an EDA (and a SOA) offer. In [11], three mechanisms for event-driven interactions are discerned: simple event processing, stream event processing, and complex event processing. In simple event processing, a 'notable' event happens, initiating actions that need to be performed by actors. In stream event processing, the real-time flow of information in and around the organization is broadcasted to information subscribers. This information can be used by actors to react on events. Complex event processing deals with evaluating a confluence of events and then taking action. The event-driven mechanism described in Sect. 4 can be best regarded as a complex event processing mechanism, as events are used to understand who needs to do what in the cross-organizational process. Moreover, intelligent computer-based actors and human actors can generate events after interpreting the relevant process descriptions. Actors can derive process information from the DogmaModeler to successfully perform the actions that are initiated by events.

9.3 Event-driven service-oriented architecture

Niblett and Graham [40] have illustrated how event-based interactions can be introduced in SOA in a standardized way. The combination of EDA and SOA into a single EDSOA infrastructure brings many advantages, because it is quite common for a single service to combine both request/response and event-oriented message exchanges [40]. Opportunities for crossover emerge when combining both EDA and SOA patterns in the same infrastructure. For example, the more straightforward determination of which services are needed for a client such as described in Sect. 3 can be realized without event-driven interactions, but the resulting integrated set of services is realized by means of event-driven interactions between the actors that deliver the result. The realization of these mechanisms is more straightforward if the service matchmaking and the event-handling fabrics are all a single integrated whole [40].

The research of Yuan and Lu [41] shows an EDSOA based on a novel concept of value-centric processing and communication of events. Incorporating this value-centric way of thinking in an EDSOA can offer a service provider to learn more information about their clients by generating client profiles. In contrast with the approach of Yuan and Lu [41], it is noticeable that our approach considers the difference between providing support for human actors and computer-based actors that perform cross-organizational processes. This distinction has two advantages. First, process

specifications can be interpreted by both humans and computers in their own language without having to lose expressive power. Second, data retrieved from the ontology database as part of the architecture are verbalized in a human-readable as well as a computer-readable language by using DogmaModeler. As such, the process information requested by actors in order to achieve ISD can be verbalized in their own languages. These same differences are found when comparing our design with that of Laliwala and Chaudhary [42]. Moreover, the coupling between the EDSOA and existing organizational processes is not made within the model of [42], while the EDSOA design in this paper aims at not only deploying the EDSOA to realize integrated services for clients, but also to realize integrated processes for service providers (resulting in a cross-organizational process).

A Disaster Notification and Resource Allocation System (DNRAS) based on an Alert Management System (AMS) implemented through Web services has been presented in [4]. This unified platform can be viewed as an EDSOA that has been specifically tailor-made for the disaster management domain. Alerts that are sent to different parties involved in disaster management can be seen as events, while those service providers are searched which can play the role required by the alert (e.g., pharmacies and large hospitals with ample storage of the required medicine). The DNRAS shows similarities with our EDSOA that have already proved useful. Some of these characteristics of the EDSOA-like architecture presented in [4] concern the support of timely interactions among various parties, the focus on notification and monitoring, resource inquiry and allocation, as well as the mobility of information.

The research described in [43] describes a solution for extending SOA with EDA concepts. The presented solution enables services to act as event producers and event consumers, but at the same time retain the interfaces and their operations. It also enables event-driven service orchestrations in business processes. The SOA and EDA concepts are combined by means of extensions to the Web Service Description Language (WSDL) and BPEL, and an XML representation of events and their payloads is also presented. A difference with our work is that the focus in [43] is on combining both approaches in a technical fashion to provide designers and developers the best of both worlds. In our work, we took the view of the service requester as a starting point and subsequently the service need of this requester could then be matched with the available services of the providers. In the case of providing multiple services by different organizations, these services are offered as a whole by executing a cross-organizational process resulting in integrated service delivery. In other words, we have not specifically looked at how the SOA and EDA concepts could be connected in a technical way but we have also aimed at understanding how the combination of EDA and SOA could realize integrated service delivery by jointly offering multiple services by multiple organizations based on complex service requests.

9.4 Integrated service delivery

Generally speaking, two options exist for realizing ISD: centrally concentrating all intelligence in a single entity and harnessing decentralized intelligence to accomplish integration. Often, top-down analyzes are based on static processes, optimization of processes, and strict process control. Decentralized approaches are adaptive, with decentralized responsibilities of autonomous and networked parties. Advanced structuring and dynamic adjustment are the main principles for flexibility as proposed in [13]. By appropriately structuring inter-organizational information flows and interconnected processes, organizations can reduce the effort involved in adjusting to changing business environments. Through IT-supported learning and adaptation, organizations can effectively and quickly reconfigure a set of inter-organizational processes that are appropriate for a changed business environment [13]. Furthermore, many workflows focus on advanced structuring, which hinders easy modification and changes during workflow executions. This can complicate a workflow's implementation or a process-aware information system's configuration [44]. Adaptive decentralized workflows are more flexible, and organizations can modify or update them. As long as ontological constraints are not violated, it does not matter how you compose an organizational workflow to integrate and deliver services [45]. We can distinguish several approaches that contribute to achieving flexible workflows, and which we can modify during workflow execution [46]:

Dynamic model evolution: Changes in dynamic workflows also require that changing process models are well managed. To support this, we can specify process model changes via a taxonomy of change modalities and a language for the unambiguous specification of procedural change.

Emergent process modeling: A common approach to managing dynamic workflow tailoring is based on partially specified process models and depends on flexible workflow systems to refine and execute them at runtime.

Exception handling: If workflows must be modified because processes change, workflow reliability can be maintained by using an exception-handling technique similar to exception handling in programming languages.

Flexibility by user selection: Providing users a workflow system with some freedom, offering them multiple workflow execution paths, results in a more flexible workflow enactment.

Finally, we can use different ontologies on the Web to describe different services. In Sect. 5, it has been mentioned that the conceptual modeling language ORM has been used to describe an ontology for ISD. However, different modeling languages can describe the same ontology. When this is the case, inter-organizational translations of the ontology are necessary. These translations, however, should also be organized in a decentralized way—that is, they should be made by actors who have the background knowledge and processing goals to perform these translations.

10 Conclusions and future research

This paper describes a detailed design of an event-driven service-oriented architecture to offer computer-based support for the realization of integrated service delivery. The main contribution of this research is the merging of aspects originating from SOA and EDA architectural styles and providing: (1) an extensive formalization, (2) an illustration showing how the EDSOA can be exploited in practice, and (3) an evaluation of the architecture. A design science approach has been applied when performing the research, leading to a list of core constructs to describe the context of the research, the actual EDSOA design, an illustration to apply the EDSOA in a practical scenario, and an evaluation consisting of an evaluative workshop. The design of the architecture shows components that can interact to realize support for ISD once the EDSOA is implemented. Among these components is a Web-based portal, which matches a client's demand for services and available services that can be supplied. The EDSOA that has been proposed in this paper contributes to the integration of services and processes of the different service providers. Executing such a cross-organizational process decreases service provision inconsistency because of the dynamic assembly of the individual processes for service delivery. The evaluation has been conducted to analyze which criteria the implemented architecture should satisfy leading to the adoption of the architecture by service providers and to show how the EDSOA can be practically exploited.

A five-step formalized approach has been introduced to match supply and demand of services. Available services can be stored in a UDDI-based service repository, which has to be managed by actors that are brought into action by the service providers. The supply of an integrated set of services to a client is realized by executing a cross-organizational process. Such a process is executed by human actors and computer-based actors. Process specifications that are represented as human-readable UML, EPC, and BPMN models can therefore also be translated to machine-readable BPEL code by incorporating the 'WorkflowNet2BPEL4WS' tool into the design. The participation of actors in process execution is supported by means of event-initiated interactions. An elaborated formal model of event-initiated interactions is described in this paper, showing how event-driven process execution works. Actors can enact the role of subscriber, publisher, and broker when dealing with events. As a subscriber, an actor can indicate in which event types he is interested. A publisher presents arising events, which can be communicated to a subscriber by means of a broker that matches supply and demand of event types. Actors that participate in the execution of a cross-organizational process need to know process information to successfully realize ISD. This information can be retrieved from an ontology database, which has an ontology for ISD as its relational database model. The DogmaModeler tool has been included in the design so that the output from the ontology database can be converted to human-readable and machine-readable verbalizations.

The outcomes of the evaluation reflected which criteria of the proposed EDSOA are most important for successful adoption by service providers. The four criteria that *consequently* represented the top four of the lists of scored and ranked criteria were expected usefulness, organizational standards, trust in technology, and ease of maintenance. This implied that a service provider will adopt the proposed EDSOA if its expected usefulness is made clear, if it fits with organizational standards, if it is based on trusted technology, and if it is easy to maintain. The evaluation also included a scenario to illustrate how ISD is realized with the support of the EDSOA. The presented scenario concerned an application for a temporary residence permit by an immigrant who had moved to the Netherlands and who had just acquired a labor contract from his employee.

In future research, the EDSOA design and the adoption criteria will be used to operationalize the EDSOA in a research project that is concerned with supply chain logistics. In this project, the EDSOA will be used to realize integrated service delivery for regulating organizations and trading organizations. A regulating organization such as the Tax and Customs Administration will be able to use services for inspection of goods and monitoring services to inspect tax payment by trading organizations. For short, regulating organizations will be able to use services to support them in their regulating activities. Trading organizations will be able to use services that support them in efficient provisioning of required data about the goods they are trading. These data are used by regulating organizations for controlling purposes. Currently, all sorts of trading organizations offer these data in different formats, which hampers regulating organizations to extract the required information from the data. Therefore, regulating organizations will also be provided with services that automatically translate different data formats in a readable format for each regulating organization for improved information extraction.

References

1. Bertolotti I, Durante L, Maggi P, Sisto R, Valenzano A (2007) Improving the security of industrial networks by means of formal verification. Comput Stand Interfaces 29(3):387–397
2. Milward H, Provan K (2003) Managing the hollow state: collaboration and contracting. Public Manag Rev 5(1):1–18
3. Lee J, Chae H, Kim CH, Kim K (2009) Design of product ontology architecture for collaborative enterprises. Expert Syst Appl 36(2):2300–2309
4. Chiu D, Lin D, Kafeza E, Wang M, Hu H, Hu H, Zhuang Y (2010) Alert based disaster notification and resource allocation. Inf Syst Front 12(1):29–47
5. Álvarez Sabucedo L, Anido Rifón L, Pérez R, Santos Gago J (2009) Providing standard-oriented data models and interfaces to eGovernment services: a semantic-driven approach. Comput Stand Interfaces 31(5):1014–1027
6. MacKenzie C, Laskey K, McCabe F, Brown P, Metz R (2006) Oasis reference model for service oriented architecture 1.0. OASIS Standard 12 October 2006, Organization for the Advancement of Structured Information Standards, Billerica
7. Zhu D, Li Y, Shi J, Xu Y, Shen W (2009) A service-oriented city portal framework and collaborative development platform. Inf Sci 179(15):2606–2617
8. Ferris C, Farrell J (2003) What are web services? Commun ACM 46(6):31–34
9. Chung JY, Chao KM (2007) A view on service-oriented architecture. Serv Oriented Comput Appl 1(2):93–95
10. Tang L, Dong J, Peng T, Tsai WT (2010) Modeling enterprise service-oriented architectural styles. Serv Oriented Comput Appl 4(2):81–107
11. Michelson B (2006) Event-driven architecture overview: event-driven SOA is just part of the EDA story. Tech. Rep. 2 Feb 2006, Patricia Seybold Group, Boston
12. Scheer A (2000) ARIS: business process modelling. Springer, Berlin
13. Gosain S, Malhotra A, El-Sawy O (2005) Coordinating for flexibility in ebusiness supply chains. J Manag Inf Syst 21(3):7–45
14. Hevner A, March S, Park J, Ram S (2004) Design science in information systems research. MIS Q 28(1):75–105
15. March S, Smith G (1995) Design and natural science research on information technology. Decis Support Syst 15(4):251–266
16. Geerts G (2011) A design science research methodology and its application to accounting information systems research. Int J Account Inf Syst 12(2):142–151
17. Simon H (1996) The sciences of the artificial, 3rd edn. MIT Press, Cambridge
18. Gregor S (2006) The nature of theory in information systems. MIS Q 30(3):611–642
19. van der Aalst W, Lassen K (2008) Translating unstructured workflow processes to readable BPEL: theory and implementation. Inf Softw Technol 50(3):131–159
20. Jarrar M, Meersman R (2008) . In: Dillon T, Chang E, Meersman R, Sycara K (eds) Advances in web semantics I. Springer, Berlin pp 7–34
21. Casteleyn S, Daniel F, Dolog P, Matera M (2009) Engineering web applications. Data-centric systems and applications. Springer, Berlin
22. Feenstra R, Janssen M, Wagenaar R (2007) Evaluating web service composition methods: the need for including multi-actor elements. Electron J e-Government 5(2):153–164
23. Dias C (2001) Corporate portals: a literature review of a new concept in information management. Int J Inf Manag 21(4):269–287
24. Liu M, Gao W, Shen Q, Hao Q, Yan J (2009) A semantic-augmented multilevel matching model of web services. Serv Oriented Comput Appl 3(3):205–215
25. WFMC (1999) Terminology & glossary. Tech. Rep. WFMC-TC-1011 issue 3.0, Workflow Management Coalition, Hampshire
26. Andrews T, Curbera F, Dholakia H, Goland Y, Klein J, Leymann F, Liu K, Roller D, Smith D, Thatte S, Trickovic I, Weerawarana S (2003) Business process execution language for web services. International Standard Version 1.1, BEA Systems, IBM, and Microsoft
27. Halpin T (2001) Information modeling and relational databases: from conceptual analysis to logical design. Morgan Kaufmann, San Mateo
28. Overbeek S, Klievink A, Janssen M (2009) A flexible, event-driven, service-oriented architecture for orchestrating service delivery. IEEE Intell Syst 24(5):31–41
29. Hofstede At, Proper HE, van der Weide T (1993) Formal definition of a conceptual language for the description and manipulation of information models. Inf Syst 18(7):489–523
30. van Bommel P, ter Hofstede A, van der Weide T (1991) Semantics and verification of object-role models. Inf Syst 16(5):471–495
31. OMG (2008) Semantics of business vocabulary and business rules (SBVR), v1.0. OMG available specification formal/2008-01-02, Object Management Group, Needham
32. Jarrar M, Demey J, Meersman R (2003) On using conceptual data modeling for ontology engineering. In: Spaccapietra S, March S, Aberer K (eds) Journal on data semantics, Lecture Notes in Computer Science, vol 2800. Springer, Berlin pp 185–207
33. McGuinness D, van Harmelen F (2004) OWL web ontology language overview. W3C recommendation, The OWL Working Group
34. van Velsen L, van der Geest T, Klaassen R, Steehouder M (2008) User-centered evaluation of adaptive and adaptable systems: a literature review. Knowl Eng Rev 23(3):261–281
35. de Vreede GJ, Dickson G (2000) Using GSS to support designing organizational processes and information systems: an action research study on collaborative business engineering. Group Decis Negot 9(2):161–183
36. Duarte B, Reis A (2006) Developing a projects evaluation system based on multiple attribute value theory. Comput Oper Res 33(5):1488–1504
37. Dillon A, Morris M (1996) User acceptance of new information technology: theories and models. In: Williams M (ed) Annual review of information science and technology. Information Today, Medford, pp 3–32
38. McGovern J, Sims O, Jain A, Little M (2006) Enterprise service oriented architectures: concepts, challenges, recommendations. The Enterprise Series. Springer, Berlin
39. Chen W, Wei J, Wu G, Qiao X (2008) In: Meersman R, Tari Z (eds) On the move to meaningful internet systems 2008: OTM 2008, Monterrey, Mexico, 9–14 Nov, 2008, Proceedings, Part I, Lecture Notes in Computer Science, vol 5331, Monterrey, Mexico (Springer, Berlin, 2008), Lecture Notes in Computer Science, vol 5331, pp 675–690
40. Niblett P, Graham S (2005) Events and service-oriented architecture: the OASIS web services notification specifications. IBM Syst J 44(4):869–886
41. Yuan ST, Lu MR (2009) An value-centric event driven model and architecture: a case study of adaptive complement of SOA for distributed care service delivery. Expert Syst Appl 36(2):3671–3694

42. Laliwala Z, Chaudhary S (2008) Event-driven service-oriented architecture. In: Lee V, Chen J, Ng WK, Ong KL, Tan T (eds) International conference on service systems and service management, 2008, Melbourne, Australia. IEEE Computer Society, Washinghton pp 1–6
43. Juric M (2010) WSDL and BPEL extensions for event driven architecture. Inf Softw Technol 52(10):1023–1043
44. Wynn M (2009) Reduction rules for YAWL workflows with cancellation regions and OR-joins. Inf Softw Technol 51(6):1010–1020
45. Burstein M (2004) Dynamic invocation of semantic web services that use unfamiliar technologies. IEEE Intell Syst 19(4):67–73
46. van der Aalst W, Benatallah B, Casati F, Curbera F, Verbeek E (2007) Business process management: where business processes and web services meet. Data Knowl Eng 61(1):1–5