Computer Standards & Interfaces 34 (2012) 355–366

Contents lists available at SciVerse ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

A standard language for service delivery: Enabling understanding among stakeholders

Sietse Overbeek [a,*], Marijn Janssen [a], Patrick van Bommel [b]

[a] Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands
[b] Institute for Computing and Information Sciences, Radboud University Nijmegen, Heijendaalseweg 135, 6525 AJ Nijmegen, The Netherlands

ARTICLE INFO

Article history:
Received 31 August 2010
Received in revised form 9 November 2011
Accepted 26 December 2011
Available online 10 January 2012

Keywords:
Integrated service delivery
Ontology
Organizational networks
ORM
SBVR
Service composition

ABSTRACT

In Integrated Service Delivery (ISD), multiple service providers have to collaborate in order to serve as a one-stop shop for their clients. Although technical standards have been met, collaboration is difficult. Service providers do not know what kind of information other providers need, are not aware of each other's processes or simply do not understand each other due to the use of ambiguous terms. In this paper, foundations for a language are developed to specify the requirements for ISD and enable the unambiguous understanding of these requirements. A combination of the standards Object-Role Modeling (ORM) and Semantics of Business Vocabulary and Business Rules (SBVR) is used to ensure human readability and to have the full expressive power of formal languages. Composed expressions are developed to express logical, temporal, and geographical requirements. This enables service providers to understand how, when, and where services need to be integrated. By utilizing these foundations to generate a standard language for ISD, service providers can collaborate and they can understand complex client requirements which lead to improved ISD.

1. Introduction

Organizations acting as service providers collaborate more and more in networks to meet customer requirements. Noteworthy examples are industrial networks, in which providers work together to provide an integrated service (e.g. computers with integrated software from various suppliers) [1], and public administrations, in which inter-organizational networks 'have become a common mechanism for delivery of public services' [2]. Collaboration is not easy as providers are often not familiar with each other's processes and the specific requirements that constrain collaboration. Contemporary service providers are shifting from purely supplying common, non-electronic services towards more demand-driven and personalized electronic service delivery (see e.g. [3]). Initially, service providers were operating in silos aiming to satisfy recurring rather than irregular client needs. As such, assessing and reacting to needs does not provide the flexibility to react to new needs or even changes in the environment. Integrated Service Delivery (ISD) concerns a bundle of services offered by more than one service provider that matches variable client needs and environmental changes [4–6]. With ISD, clients perceive a bundle of services provided by various service providers as a whole and do not have to deal with each single provider.

Currently, web services are used as a standard technology to enable ISD. Each service provider can make services accessible as web services and ISD can be created by composing and invoking a set of web services [7,8]. 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 [9]. 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. This enables dynamic service binding and increases cross-language and cross-platform interoperability [10,11]. Another advantage of webservices is that they can be composed. Composing services rather than accessing a single service provides service providers and application developers the opportunity to develop value-added services by combining existing web services [8].

Although there is technological support for creating ISD by means of web services, a standard language is lacking to describe the inter-organizational requirements that service providers need to satisfy for successful realization of ISD [12]. To allow service providers undergo a transition from delivering fragmented services towards delivering integrated services, it should be specified first which requirements these organizations should meet to realize ISD. There is also a need for a shared understanding of these requirements, because service providers use a variety of terms and definitions. Sometimes identical terms are defined differently by varying service providers or definitions of terms are ambiguously interpreted by different service providers. This denotes that there is a need for a shared vocabulary that is easy to understand by the providers involved. For example, an American bank may require that a client can only open a savings account if the client can deposit at least a thousand dollars. This implies that there is a requirement for a client wishing to open a savings account. Assume that this banking

* Corresponding author. Tel.: +31 15 2785526; fax: +31 15 2783741.
E-mail addresses: S.J.Overbeek@tudelft.nl (S. Overbeek), M.F.W.H.A.Janssen@tudelft.nl (M. Janssen), P.vanBommel@cs.ru.nl (P. van Bommel).

0920-5489/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2011.12.006

service is part of an integrated service. In this case, problems in collaboration occur if the results of dealing with the aforementioned example requirement are not clear for the other collaborating service providers. Assume that a work permit service is also part of the integrated service which requires a client to have a savings account not later than two weeks after the request for a work permit. The client's success or failure to fulfill the requirement to open a savings account should then also be known for the service provider of the work permit service.

Two requirements can be derived from this example to understand *how* the banking service and the work permit service can be delivered and integrated. The first one is the requirement to deposit thousand dollars in order to open a savings account. The second one is the requirement of having a savings account in order to obtain a work permit. The first requirement is related to the service delivery, while the second one is related to the integration of the two services. The requirement that a client should have a savings account not later than two weeks after the request for a work permit is an example of *when* services can be delivered or integrated. It is also known that one service is delivered by a bank and the other is delivered by the immigration and naturalization service. This is of importance to know *where* the services are delivered or integrated. This straightforward example shows that the risk of ISD is that service providers are unable to realize ISD because they do not understand each other or might not have enough understanding of the situation. The goal of this paper is to provide foundations for a standard language to specify requirements to enable ISD. The foundations describe logical, temporal, and geographical information to understand how, when, and where services need to be integrated and delivered. The empirical motivation that has led to the fulfillment of this research is discussed in Section 2, which is followed by a problem analysis in Section 3. The proposed properties of a standard language for ISD are presented in Section 4. In Section 5, we discuss how requirements for ISD can be retrieved in practice and in Section 6 the properties of the foundations are evaluated. In Section 7, we address related research. The conclusions are presented in Section 8.

2. Empirical motivation

The motivation for this research has been empirically founded by investigating a real-life *expat* case study that is used to increase our understanding of ISD [13]. Expats are people who want to migrate to another country to live and work there. To be able to do so, they need at least a (temporary) residence permit, a registration in the citizens' registry, a bank account, a job, health insurance, and housing. ISD can be realized when the involvement or participation of organizations, actors, services, resources, and events is investigated [5,14–16]. Resulting from the case study, an ISD ontology is presented describing the main elements of ISD. Recent research has already shown the usefulness of an e-government ontology for facilitation and organization of public service composition and provisioning [17]. Fig. 1 visualizes the situation where a bank, the immigration and naturalization service, the tax administration, and a health insurance company work together to provide a set of integrated services. The expat case is used for two reasons: (1) to illustrate that the requirements for ISD should be made clear to all the participating organizations in order to realize a shared understanding among these organizations, and (2) to gather properties for the proposed standard language to describe requirements for ISD. Without ISD, the client needs to open a bank account, acquire a residence permit and apply for a tax declaration and health insurance via the appropriate services provided by different service providers. These services can be bundled and offered to the client on a portal, which is a web-based application that enables organizations to provide clients with a single gateway to information that fulfills their personal information need [18].

The foundations for a standard ISD language that are specified in Section 4 form the core of this paper and are based on the ontology and case study. These foundations can be used to generate the requirements needed to achieve ISD. They have been formalized to offer precise syntax and semantics. The requirements that can be formed by using these foundations are verbalized by using the Semantics of Business Vocabulary and Business Rules (SBVR) specification that has been published by the Object Management Group (OMG) recently [19]. SBVR defines the meta model for documenting the semantics of business vocabulary and business rules such as ISD requirements. SBVR was selected as it improves readability and prevents ambiguous interpretations among service providers who need to interpret ISD requirements, and it is a human-readable language that at the same time has the full power of formal languages. A comparable approach aimed at realizing human-readable business process models without incorporating unnecessary technicalities can be found in [20]. This underlying thought can also be applied to create a standard language to specify requirements for integrated service delivery. Further analysis of the problems in integrated service delivery for which this standard language can be a solution is discussed next.

3. Problem analysis

It is observed that ontologies are becoming quintessential for organizations, that see them as vital machine-processable semantic resources for many application areas [21]. An ontology is an agreed understanding of a certain domain, formally represented as logical theory in the form of a computer-based resource. Complex software applications, such as web services, can communicate meaningfully to exchange data. Thus, these applications make such data transactions interoperate independently of their internal technologies via a shared ontology. The shared ontology, which includes semantics related to the domain in which the service providers involved operate, will create a starting point for realizing ISD. As a shared ontology is an agreed understanding of how services should be integrated, it makes it possible to identify essential concepts and relationships between concepts for ISD.

3.1. Viewpoints

The case study is a starting point to generate insights in the requirements to enable ISD. The expat case study shows that five viewpoints are of importance for the creation of an ontology, see Fig. 2. A viewpoint is that part of the expat case study that is of special interest for a service provider [22]. For example, a bank is especially interested in that part of the case related to their own banking services. Subsequently, the ontology can be used as a basis for providing support to an ISD. First, from an *organization-oriented view*, domain-specific knowledge has been acquired from the case study by applying a bottom-up approach, in which existing business processes have been examined that include information on the provisioning of services during process fulfillment. Several process models have been created on the basis of this study. Secondly, from an *actor-oriented view*, we have analyzed how actors working togetherin such processes would fulfill their part of a process in an organization. To realize ISD, the tasks of several types of actors need to be coordinated, because multiple actors are involved in delivering complex services. In this case, an actor can be defined as an entity able to perform a task, for instance a human or a computer. An example of an actor in the expat case is a citizen. As part of the case study, eleven interviews have been conducted with expats to understand how they participated in process fulfillment in their attempts to acquire a residence permit, a registration in the citizens' registry, etc. Thirdly, the *service-oriented view* describes organizations offering concrete services that are required for a successful process fulfillment. An example of a web service is the possibility to open a bank account on the web. Fourthly, the *resource-oriented view* describes resources belonging to an organization and the processes that use specific resources. Examples of resources are people, computers, and office supplies. Finally, the *event-oriented view* describes events that are transmitted between a set of integrated and interacting services. Such events are consumed or

produced by actors in organizations. An example of an event in the expat case is the event 'residence permit form received', when an expat receives a residence permit. From the event-oriented view, the concept of an Event-Driven Service-Oriented Architecture (EDSOA) can be identified [16,23].

This notion defines a methodology for designing and implementing computer-based applications and systems in which events (i.e., a change of state during service delivery) are transmitted between a set of integrated and interacting web services [16]. An actor consuming an event can subscribe to an event broker who manages such events, and an actor who produces an event publishes to this broker. When an event is broadcasted by an actor, the broker facilitates the forwarding of this event to a demanding actor. If a demanding actor is unavailable, the broker can store the event and try to forward it later. Building applications and systems based on an EDSOA allows these applications and systems to be more responsive, since such systems are geared more towards unpredictable and asynchronous environments. Eventually, because an EDSOA can enable ISD and coordination of events between services in practice, the concept of EDSOA is also part of the ISD ontology.

The expat case contains problems that are typical of ISD and involves the collaboration of various service providers. The main problem in this regard is the tendency of service providers to operate in silos in a time when they often have to work together with other partners in the public and private sector to provide a service. Also, existing business processes sometimes fail to match the exact needs of expats. After summarizing the insights from several viewpoints involving a case presenting problems that are typical of ISD, we can now visualize the ontology. This helps to distinguish the key concepts and relationships that form the basis of coordinating the activities that are needed for an ISD.

3.2. Ontology for describing concepts and relationships in ISD

As the basis for understanding the requirements for ISD, an ontology has been developed capturing the main concepts and relationships. This is necessary to know which elements need to be taken into account when modeling requirements for ISD. Fig. 3 represents the resulting Object-Role Modeling (ORM) model of the ontology for an ISD, which was introduced in Ref. [13]. ORM is a rich and formalized conceptual data modeling technique that can be used for the conceptual modeling of database models as well as for a variety of other modeling purposes, such as the modeling of ontologies. With ORM, the syntax and the semantics can be coherently formulated in a mathematical language. It has a long running affiliation with the field of conceptual modeling involving varied, often non-technical audiences [24], and it includes a stable attribute-free graphical notation [25]. Objects are treated as concepts in ORM, which 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. These relationships are called fact types. For more details on Object-Role Modeling, see Refs. [26–28]. The ontology contains eight central concepts that result from knowledge gained from the expat case summarized in Section 3.1: role, actor, service, process, resource, organization, event, and event-driven service-oriented architecture. Note that the concepts of actor, service, resource, organization, and event are directly derived from the five viewpoints. The remaining concepts are directly related to the derived concepts and these relations are made explicit in the ontology.

With regard to the ORM model of the ISD ontology shown in Fig. 3, three different constraints can be distinguished: the mandatory, uniqueness, and subset constraints. The *mandatory* constraint is sometimes referred to as the total role constraint, in which each instance of an object type has to play the role to which the total role constraint is related. Fig. 3 shows that every process is performed by an actor. *Uniqueness* constraints refer to the notion that instances of object types may play a certain combination of roles at most once. For example, Fig. 3 shows that the combination of a specific actor performing an instance of a process can occur once at the most. *Subset* constraints, finally, are 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. In Fig. 3, a subset constraint is used to indicate that each service being used by an actor should be required for some process. Next, we complement the ORM model with foundations for a language that can be used to specify requirements for service providers in order to achieve an ISD.

4. Expressing the properties of a language for an integrated service delivery

A specification that is directly associated with the conceptual modeling language ORM is the Semantics of Business Vocabulary and Business Rules specification [19]. SBVR makes it possible to describe properties of

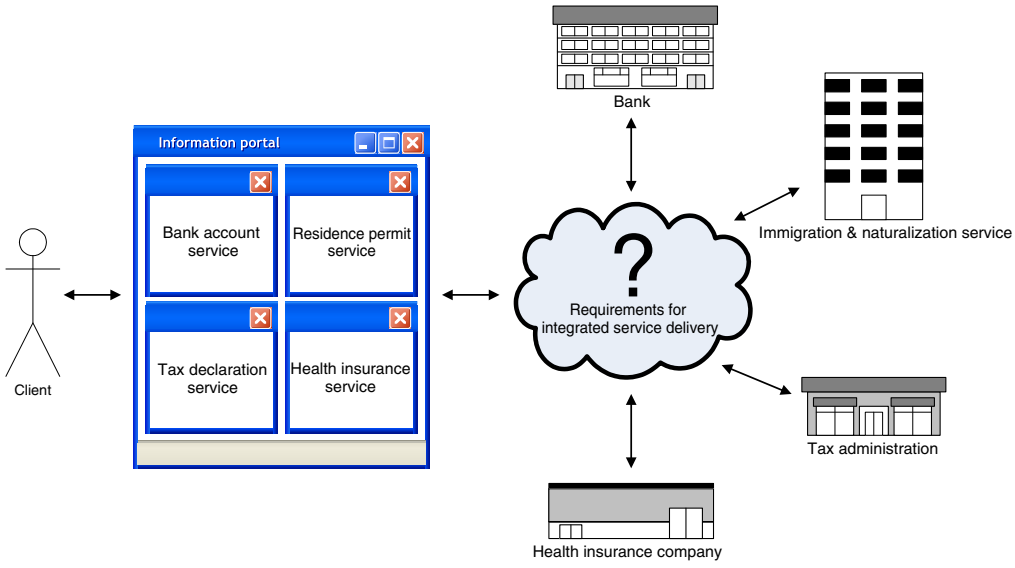

Fig. 1. The need for understanding and specifying requirements for ISD.

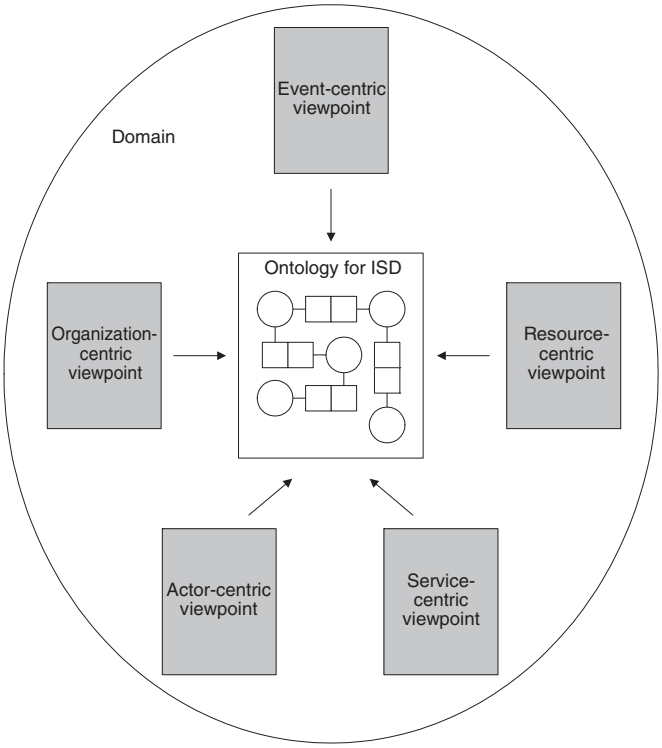

Fig. 2. Viewpoints of service providers.

the underlying domain in terms of the domain language, and at the same time has the full power of formal logical languages, such as first-order predicate calculus. Together with ORM, this enables to address the variety and ambiguity in terms and definitions used by different service providers. The names of the roles that instances of object types can play, such as 'IsConsumedBy', are shown in Fig. 3. These verbalizations make it possible to relate the concepts that are part of the ontology. By assigning names to the roles in SBVR, a language can be created that is suitable for specifying requirements for an ISD based on the ontology. Fig. 3 is, therefore, the basis for our language.

Other initiatives have been assessed on their suitability to specify ISD requirements, which include the Semantic Markup for Web Services (OWL-S) [29], the Process Specification Language (PSL) [30], the Semantic Web Services Ontology (SWSO) [31], the Web Service Modeling Ontology (WSMO) [32], the Business Process Execution Language for Web Services (BPEL) [33], the Language for Information Structure and Access Descriptions (LISA-D) [34], and the Object-Role Calculus (ORC). However, SBVR seems the most suitable language to specify ISD requirements as most of the other mentioned languages are more suitable for other needs, such as: the declaration, description and characterization of services (OWL-S, SWSO, WSMO), representation for manufacturing processes that support automated reasoning (PSL), and specification of cross-organizational processes in a web services context (BPEL). The languages LISA-D and ORC are comparable to SBVR, but these languages have never reached the level of an international standard language and are not widely adopted in practice. Because of these reasons, none of the mentioned other languages seem to be more suitable than SBVR to describe ISD requirements in a human-readable way while also bearing a formal foundation. We will now continue to discuss the verbalizations represented in Fig. 3.

The SBVR verbalizations are divided into basic expressions and composed expressions. *Basic expressions* are simple verbalizations that are be formed by navigating the ontology in such a way that a single sentence is formed. There are four text styles that can be used for SBVR verbalizations that have a formal meaning [19]. An underlined term 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 the object types in an ORM model. A double underlined name is used for a designation of an individual concept. Names in SBVR are verbalizations of the object instances. The *verb* style is used for designations of fact types in an ORM model. A normal font style 'keyword' is used for linguistic symbols that are used to construct statements, i.e. the words that can be combined with other designations to form statements and definitions. The verbalization 'actor … *uses* service …' is an example of a basic expression that can be derived from the ontology. The dots are placeholders for possible object instances. *Composed expressions* are more complex in the sense that basic expressions are combined (and which can be formed recursively by combining composed expressions) by construction operators (see Sections 4.2, 4.3, and 4.4). A construction operator is a mathematical operator that is used to assemble basic or existing expressions [35]. Construction operators can be found in various domains, such as the logical, temporal, and geographical domains.

Construction operators from the logical domain are used to display the logical structure of possible conclusions [36]. In ancient times, natural languages were already parsed to form in fixed-patterns, i.e., conclusions, that were drawn from sentences occurring in natural language. An example of such a conclusion is: 'All cats are carnivores. No cat dislikes fresh fish. Consequently, every cat eats fresh fish'. First, the conclusions can be made visible by using logical construction operators, which are quite often rather hidden in natural language. By using logical construction operators to describe parts of a language for ISD, we are able to draw conclusions in a valid way. This also makes it possible to identify fairly complex conclusions that often remain hidden in natural languages. Construction operators from the temporal domain can be used to express an *instant* of time, a *duration* or a *period* with regard to a requirement for an ISD [37]. An instant is a point in time (e.g., 2009 May 19, 1:00 p.m. MDT), a duration is a length of time (e.g., 2 weeks), and a period is an anchored duration of time (e.g., 2009 May 19–2009 May 21 PST). Construction operators from the geographical domain can be used to express geographical information with regard to an ISD requirement. This implies that, if *spatial* information (e.g., under, near, north of, etc.) is relevant to expressing a requirement, a geographical construction operator can be used [38].

Four properties of a standard language for ISD are determined such that it is suitable for specifying requirements for ISD [35]. First, the *expressiveness* of the language is important. Expressiveness is relevant for a language for ISD, where basic and composed expressions are assumed to form a correct representation of the requirements for integrating services. The basic and composed expressions should *exactly* and *precisely* reflect possible requirements for ISD in order to let the language for ISD be expressive. In general, however, expressiveness reduces *tractability*, which is the simplicity with which a computer is able to process a basic or composed expression [35]. Tractability is *inversely proportional* to the difficulty involved in solving a problem. Simply put, inverse proportion means that, as the absolute value or magnitude of one variable becomes bigger, the absolute value or magnitude of another variable becomes smaller, such that their product (the constant of proportionality) is always the same. For example, the time it takes to complete a journey is inversely proportional to the speed with which one travels. The tractability of a language for ISD is considered to be inversely proportional to the amount of time, space (computer memory), and auxiliary knowledge needed for a computer to process basic or composed expressions that comprise the language for ISD in order to understand which requirements should be met for the delivery of a certain composed set of services.

Subsequently, *comprehensibility* is important with regard to interaction with humans who want to understand requirements for ISD [39]. In the next sections, we show that humans can also understand the simple structure of the proposed standard language for ISD. Finally, ISD requirements can be represented in a *compact* way by nesting

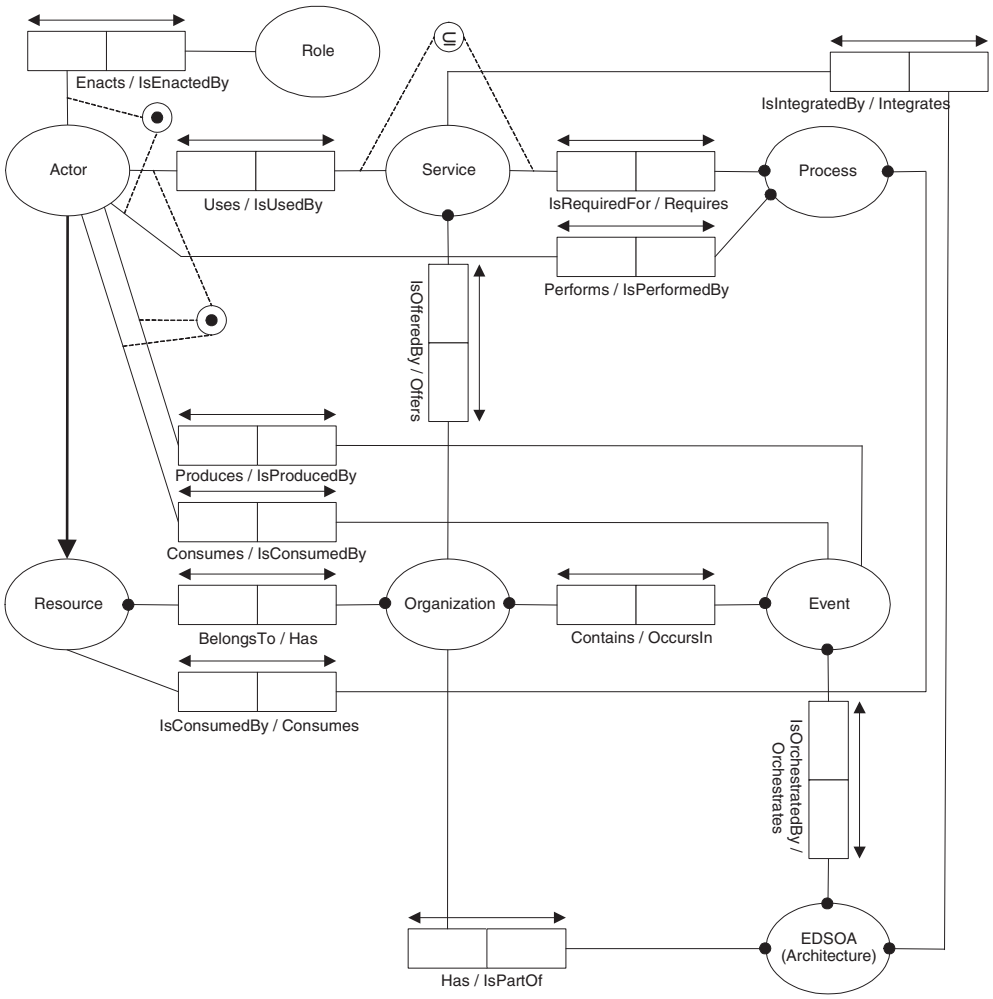

Fig. 3. A case-based ontology for integrated service delivery [13].

construction operators and basic expressions in a structure of composed expressions. Humans involved in the ISD process may benefit from compactness when formulating requirements for ISD, because compact expressions save mental space and time when reading or formulating requirements [35]. For reasons of clarity, the mentioned four properties of a standard language for ISD are briefly summarized in Table 1.

In Section 6 we will evaluate the proposed language for ISD based on the four properties as mentioned here. What is derived first from the ontology shown in Fig. 3 are basic expressions for developing basic requirements for ISD.

4.1. Basic expressions

The SBVR specification can be used to specify basic expressions for the eight concepts included in the ISD ontology shown in Fig. 3. A basic expression can be found by navigating the ontology and exactly augmenting two object instances with a relationship (the name of the fact type in the ontological ORM model) between them. A basic expression as part of a language for ISD can be defined as follows. Let $\mathcal{OT}$ be the set of object types, $\mathcal{IN}$ be the set of object instances, $\mathcal{RE}$ be the set of relations (or fact types in an ORM model), and $\mathcal{EX}$ be the set of all possible expressions. The set $\mathcal{BE}$ is the set of basic expressions which implies $\mathcal{BE} \subseteq \mathcal{EX}$. The formal signature of a function that can add these parts to form a basic expression can be described as follows:

$$Add : \mathcal{OT} \times \mathcal{IN} \times \mathcal{RE} \times \mathcal{OT} \times \mathcal{IN} \rightarrow \mathcal{EX} \quad (1)$$

Assume that $o_1, o_2 \in \mathcal{OT}, i_1, i_2 \in \mathcal{IN}, r \in \mathcal{RE}$, and $b \in \mathcal{BE}$. The expression $Add(o_1, i_1, r, o_2, i_2) = b$ shows that a basic expression b is formed by a sequence of an object type, an object instance, a relationship, another object type, and another instance. Basic expressions that can be found in Fig. 3 are listed in Table 2, together with expressions for instances originating from the above-mentioned case study. For instance, the first basic expression shown in Table 2 can be formed by using the add equation as follows: $Add(\underline{\text{Actor}}, \underline{\text{John}}, \textit{enacts}, \underline{\text{role}}, \underline{\underline{\text{expat}}})$. Note that construction operators are not used for basic expressions, but for combining basic expressions to form composed expressions.

The basic expressions that are listed in the table can be categorized according to the five viewpoints as introduced in Section 3.1. The basic expressions that start with the actor and role object types are expressions as seen from an actor-oriented perspective. The basic expressions that start with the service object type are expressions that can be categorized as expressions as seen from a service-oriented perspective. The expressions starting with the resource object type are resource-oriented expressions. Basic expressions starting with the object types organization and process can be viewed as organization-oriented expressions. Finally, expressions starting with the architecture and event object types are basic expressions as seen from an event-oriented perspective. This shows that expressions from all five viewpoints on ISD can be developed. The basic expressions listed in Table 2 can be assembled to formulate more complex expressions which are dubbed as composed expressions. The composed expressions are divided into composed expressions with logical operators, temporal operators, and geographical operators.

Table 1
Four properties of a standard language for ISD.

Property	Explanation
Expressiveness	Exact and precise reflection of requirements for ISD
Tractability	Simplicity with which a computer is able to process basic or composed expressions
Comprehensibility	Understandability of the language for ISD by humans
Compactness	The extent to which the language for ISD is densely structured

4.2. Composed expressions with logical operators

Two basic expressions can be assembled by means of a construction operator, resulting in a composed SBVR expression. This can be realized by using the compose equation:

$$Compose : \mathcal{E}\ \mathcal{X} \times \mathcal{E}\ \mathcal{X} \rightarrow \mathcal{C}\ \mathcal{E} \tag{2}$$

The set $\mathcal{C}\ \mathcal{E}$ is the set of composed expressions. The compose equation can assemble basic expressions, but composed expressions can also be assembled recursively. Because $\mathcal{BE}, \mathcal{CE} \subseteq \mathcal{EX}$, it is possible to assemble basic expressions and composed expressions, which always results in a new composed expression. When referring to the five viewpoints from Section 3.1, it becomes obvious that composed expressions can be used to form requirements for ISD that are based on a combination of those viewpoints. There are four possibilities to create composed expressions:

1. $\exists_{x,y \in \mathcal{B}\ \mathcal{E}} \exists_{z \in \mathcal{C}\ \mathcal{E}} [Compose(x,y) = z]$
2. $\exists_{x \in \mathcal{B}\ \mathcal{E}} \exists_{y,z \in \mathcal{C}\ \mathcal{E}} [Compose(x,y) = z]$
3. $\exists_{y \in \mathcal{B}\ \mathcal{E}} \exists_{x,z \in \mathcal{C}\ \mathcal{E}} [Compose(x,y) = z]$
4. $\exists_{x,y,z \in \mathcal{C}\ \mathcal{E}} [Compose(x,y) = z]$

This implies that:

1. A composed expression can be formed by combining two basic expressions.
2. A composed expression can be formed by combining a basic expression with a composed expression.
3. A composed expression can be formed by combining a composed expression with a basic expression.
4. A composed expression can be formed by recursively combining two composed expressions.

The simplest construction operators to form composed expressions can be the logical operators for conjunction (and), disjunction (or), and negation (not). These operators are also part of the SBVR specification [19] to compose statements. These operators can be used for Boolean composed expressions that result in true or false. For instance, it can be true that actor Jane *enacts* role tax officer and actor Jane *performs* process handle tax declaration . This is an example of a composed expression that combines both the actor-oriented viewpoint and the organization-oriented viewpoint on ISD. Subsequently, composed expressions may also be used in a set-theoretical context. For example, the following verbalization produces *all* the roles that are enacted by a certain actor Jane and *all* the processes that are performed by that actor: Actor Jane *enacts* a role and actor Jane *performs* process. The meaning of the logical operators should be clarified in order to use them in composed expressions. In fact, a logical operator combines two basic expressions to form a composed expression. Using the above-mentioned definition of the set $\mathcal{B}\ \mathcal{E}$, the formal signature of the logical operators can be provided as follows:

$$\_\text{and}\_, \_\text{or}\_, \_\text{not}\_ \subseteq \mathcal{BE} \times \mathcal{BE} \tag{3}$$

Thus, x and y means that two basic expressions $x, y \in \mathcal{B}\ \mathcal{E}$ are both true. The definitions of the and, or, and not construction operators are easy to provide by using the common mathematical operators for logical expressions (see e.g. Ref. [36]):

$$x \text{ and } y \triangleq x \wedge y \tag{4}$$

$$x \text{ or } y \triangleq x \vee y \tag{5}$$

$$x \text{ not } y \triangleq x \neq y \tag{6}$$

Next, composed expressions can be formed to verbalize time-related requirements for ISD.

4.3. Composed expressions with temporal operators

As mentioned earlier, construction operators from the temporal domain can be used to express an *instant* of time, a *duration*, or a *period* with regard to a requirement [37]. *Allen's operators* [40] provide a point of departure for adding temporal expressiveness to our language for ISD. Construction operators from the temporal domain are not included in the current SBVR specification yet [19]. Therefore, we will introduce such operators in this section to be able to generate SBVR statements that include temporal operators. For instance, the composed expression actor Jane *enacts* role tax officer after actor Jane *belongs to* organization tax administration shows that actor 'Jane' became a tax officer after becoming employed at the tax administration. Fig. 4 visually depicts Allen's operators as 13 mutually exclusive relationships between an ordered pair of closed, proper periods P_1 and P_2. Note that $P_1, P_2 \subseteq \mathcal{P}\ \mathcal{E}$. The set $\mathcal{P}\ \mathcal{E}$ contains possible proper periods. To use Allen's operators in composed SBVR expressions, we need to understand the meaning of the operators. In our case, we need to understand when a situation verbalized by a basic expression *starts* and when it *ends*. For example, it is only possible to know the meaning of operators like 'before', 'after', or 'equals' if the start and end times of a situation verbalized by a basic expression are known. This allows for clear differentiation between two time periods. The formal signature of the start and end equations can be represented as follows:

$$Start, End : \mathcal{B}\ \mathcal{E} \rightarrow \mathcal{P}\ \mathcal{E} \tag{7}$$

For example, $Start(b) = p$ implies that the start time of the occurred situation expressed by b is p. Next, formal definitions of start and end times are modeled to give meaning to these equations:

$$Start(b) \triangleq \forall_{b \in \mathcal{B}\ \mathcal{E}} \forall_{p \in \mathcal{P}\ \mathcal{E}} [Start(b) < p \wedge p \neq Start(b)] \tag{8}$$

The definition of the start time shows that there are no time instants that can be earlier than instant $Start(b)$, i.e. the start time.

$$End(b) \triangleq \forall_{b \in \mathcal{B}\ \mathcal{E}} \forall_{p \in \mathcal{P}\ \mathcal{E}} [End(b) > p \wedge p \neq End(b)] \tag{9}$$

The definition of the end time shows that there are no time instants that can be later than instant $End(b)$.

Using the start and end time equations, notations can be introduced for each of Allen's operators shown in Fig. 4. The 'before' operator can be formalized as follows:

$$\_\text{before}\_ \subseteq \mathcal{BE} \times \mathcal{BE} \tag{10}$$

Note that all temporal operators are proper subsets of the Cartesian product of two sets of basic expressions. The notation above is, therefore, not repeated for the remaining operators shown in Fig. 4. The definition of 'before' can be represented as follows, where $x, y \in \mathcal{B}\ \mathcal{E}$:

$$x \text{ before } y \triangleq End(x) < Start(y) \tag{11}$$

If x before y, then the end time of a situation expressed in x should always precede the start time of another situation y. Using the basic

Table 2
Basic expressions in SBVR for concepts related to integrated service delivery.

Basic expression	Example
Actor ... *enacts* role ...	Actor John *enacts* role expat
Actor ... *uses* service ...	Actor John *uses* service residence permit
Actor ... *performs* process ...	Actor John *performs* process request residence permit
Actor ... *produces* event ...	Actor John *produces* event residence permit form signed
Actor ... *consumes* event ...	Actor John *consumes* event residence permit form collected
Actor ... *belongs to* organization ...	Actor Jane *belongs to* organization tax admin
Actor ... *is consumed by* process ...	Actor Jane *is consumed by* process handle tax declaration
Role ... *is enacted by* actor ...	Role tax officer *is enacted by* actor Jane
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
Resource ... *belongs to* organization ...	Resource tax mgmt. system *belongs to* organization tax admin
Resource ... *is consumed by* process ...	Resource tax mgmt. system *is consumed by* process handle tax declaration
Organization ... *offers* service ...	Organization tax admin *offers* service tax declaration
Organization ... *has* resource ...	Organization tax admin *has* resource tax mgmt. system
Organization ... *contains* event ...	Organization tax admin *contains* event tax declaration received
Organization ... *has* architecture ...	Organization tax admin *has* architecture expat SOA
Architecture ... *is part of* organization ...	Architecture expat SOA *is part of* organization tax admin
Architecture ... *orchestrates* event ...	Architecture expat SOA *orchestrates* event res. permit form signed
Architecture ... *integrates* service ...	Architecture expat SOA *integrates* service residence permit
Event ... *is orchestrated by* architecture ...	Event permit form collected *is orchestrated by* architecture expat SOA
Event ... *occurs in* organization ...	Event tax declaration received *occurs in* organization tax admin
Event ... *is consumed by* actor ...	Event tax declaration received *is consumed by* actor Jane
Event ... *is produced by* actor ...	Event tax declaration sent *is produced by* actor John
Process ... *requires* service ...	Process request residence permit *requires* service residence permit
Process ... *is performed by* actor ...	Process handle tax declaration *is performed by* actor Jane
Process ... *consumes* resource ...	Process handle tax declaration *consumes* resource tax mgmt. system

expressions from Table 2, a meaningful composed expression can now be formed, such as: Event tax declaration sent *is produced by* actor John before process handle tax declaration *is performed by* actor Jane. The definition of the 'after' operator is comparable to that of the 'before' operator:

$$x \text{ after } y \triangleq Start(x) > End(y) \tag{12}$$

The meaning of the 'equals' operator shown in Fig. 4 is that the start time and end time instants of two situations verbalized by basic expressions must be exactly equal:

$$x \text{ equals } y \triangleq Start(x) = Start(y) \wedge End(x) = End(y) \tag{13}$$

When combining two basic expressions from Table 2 by means of this operator, a meaningful composed expression can be formed: Role tax officer *is enacted by* actor Jane equals actor Jane *belongs to* organization tax administration. This composed expression is true if Jane has indeed started working and has also ended working as a tax officer at the tax administration (without changing jobs at the same company in the meantime). Subsequently, the 'meets' operator can be defined as follows:

$$x \text{meets} y \triangleq End(x) = Start(y) \wedge Start(x) < Start(y) \wedge End(x) < End(y) \tag{14}$$

The end time instant of a situation verbalized by a basic expression must be equal to the start time instant of another situation if both situations 'meet' each other. Moreover, the start time of a situation that meets another situation must be earlier than the start time of the situation that is met. This also applies to the end times. Table 2 shows two basic expressions that can be assembled by means of the 'meets' operator as follows: Actor John *produces* event residence permit form signed meets architecture expat SOA *orchestrates* event residence permit form signed. The definition of the 'met by' operator is now trivial:

$$x \text{ met by} y \triangleq Start(x) = End(y) \wedge Start(x) > Start(y) \wedge End(x) > End(y) \tag{15}$$

The next operators are the 'overlaps' and 'overlapped by' operators. The start time and the end time of a situation that overlaps another situation must be earlier than those of the situation that is overlapped:

$$x \text{ overlap } y \triangleq Start(y) > Start(x) \wedge End(y) > End(x) \tag{16}$$

$$x \text{ overlapped by } y \triangleq Start(x) > Start(y) \wedge End(x) > End(y) \tag{17}$$

An example related to Table 2 can be verbalized as follows: Actor John *enacts* role expat overlaps role tax officer *is enacted by* actor Jane. The operators 'during' and 'contains' can be used to indicate that a situation takes place within the time span of another situation. This can be formalized by the following equations:

$$x \text{ during } y \triangleq Start(x) > Start(y) \wedge End(x) < End(y) \tag{18}$$

$$x \text{ contains } y \triangleq Start(y) > Start(x) \wedge End(y) < End(x) \tag{19}$$

A meaningful expression related to the contents of Table 2 is easy to find: Actor John *uses* service residence permit during actor John *enacts* role expat. Then, the 'starts' and 'started by' operators can be modeled:

$$x \text{ starts } y \triangleq Start(x) \Rightarrow Start(y) \wedge End(x) \neq End(y) \tag{20}$$

$$x \text{ started by } y \triangleq Start(y) \Rightarrow Start(x) \wedge End(y) \neq End(x) \tag{21}$$

The 'starts' operator can be used to assemble the following basic expressions, for instance: Event tax declaration received *occurs in* organization tax administration starts event tax declaration received *is consumed by* actor Jane. Finally, the last two temporal construction operators are the 'finishes' and 'finished by' operators:

$$x \text{ finishes } y \triangleq End(x) \Rightarrow End(y) \wedge Start(x) \neq Start(y) \tag{22}$$

$$x \text{ finished by } y \triangleq End(y) \Rightarrow End(x) \wedge Start(y) \neq Start(x) \tag{23}$$

Subsequently, geographical requirements can be formed by using *geographical operators*. Like temporal operators, these kinds of operators

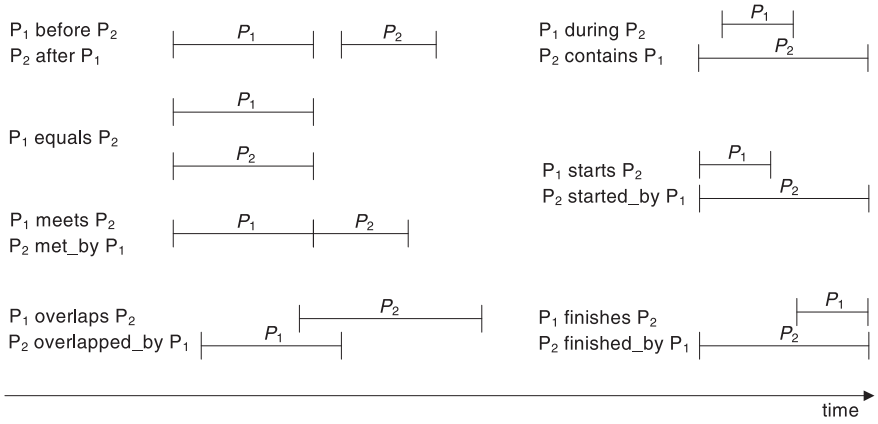

Fig. 4. Allen's operators for comparing closed proper periods, adapted from [37,40].

aren't part of the SBVR specification yet. Therefore, they are introduced in the next section such that they can be used in SBVR verbalizations to specify ISD requirements.

4.4. Composed expressions with geographical operators

Geographical expressions can be used to form requirements regarding the locations of concepts that are related to ISD. For instance, the expression actor John *produces* event residence permit form signed north of actor Jane *performs* process handle residence permit request shows that John is signing a residence permit form north of Jane, who performs a process to handle a request for a residence permit. To work with geographical operators, the *coordinates* of objects that are related to the geographical operator need to be determined. It is possible to reason about the spatial relationships between objects if such coordinates are known. The three-dimensional coordinates of a main concept mentioned in a basic expression can be found by means of the coordinates equation:

$$Coord : \mathcal{B}\,\mathcal{E} \rightarrow \mathbb{R} \times \mathbb{R} \times \mathbb{R} \tag{24}$$

The coordinates can be plotted on a three-dimensional Cartesian coordinate system, with an origin and three axis lines *X*, *Y*, and *Z*. For example, $Coord(b) = (3, 1, 4)$ indicates what the coordinates are for a main concept as part of an arbitrary basic expression *b*. Fig. 5 shows how this can be represented visually. Several geographical operators can be presented and used to form composed expressions based on a spatial relationship between two basic expressions. Some useful geographical operators are 'above', 'under', 'ahead', 'behind', 'north of', 'south of', 'east of', and 'west of'. The signature of these operators can also be formalized in the following way:

$$\_\text{above}\_ \subseteq \mathcal{BE} \times \mathcal{BE} \tag{25}$$

The signatures of the other geographical operators discussed in this section are identical. The 'above' and 'under' operators can be defined as follows:

$$b_1 \text{ above } b_2 \triangleq \exists_{b_1,b_2 \in \mathcal{B}\,\mathcal{E}} \exists_{x_1,x_2,y_1,y_2,z_1,z_2 \in R}[Coord(b_1) = (x_1,y_1,z_1) \wedge Coord(b_2) = (x_2,y_2,z_2) \wedge z_1 > z_2] \tag{26}$$

$$b_1 \text{ under } b_2 \triangleq \exists_{b_1,b_2 \in \mathcal{B}\,\mathcal{E}} \exists_{x_1,x_2,y_1,y_2,z_1,z_2 \in R}[Coord(b_1) = (x_1,y_1,z_1) \wedge Coord(b_2) = (x_2,y_2,z_2) \wedge z_1 < z_2] \tag{27}$$

A meaningful composed expression that is created by using the 'above' operator can be described as follows: Process handle tax declaration *is performed by* actor Jane above process assess housing benefit request *is performed by* actor George. Expressions such as these may be useful when, for instance, Jane is searching for additional information on housing benefits while performing her job as a tax officer. Knowing that her colleague George is working on a task related to this topic in close proximity (above her in the same building probably) can be helpful.

The intention of the 'ahead' and 'behind' operators is to express whether an object is exactly in front of (ahead) or to the rear of (behind) another object. In terms of the three-dimensional coordinates, this implies that the objects have equal coordinates on the *Y* and *Z* axes, but a different coordinate on the *X* axis, or formally:

$$b_1 \text{ ahead } b_2 \triangleq \exists_{b_1,b_2 \in \mathcal{B}\,\mathcal{E}} \exists_{x_1,x_2,y_1,y_2,z_1,z_2 \in R}[Coord(b_1) = (x_1,y_1,z_1) \wedge Coord(b_2) = (x_2,y_2,z_2) \wedge x_1 > x_2] \tag{28}$$

$$b_1 \text{ behind } b_2 \triangleq \exists_{b_1,b_2 \in \mathcal{B}\,\mathcal{E}} \exists_{x_1,x_2,y_1,y_2,z_1,z_2 \in R}[Coord(b_1) = (x_1,y_1,z_1) \wedge Coord(b_2) = (x_2,y_2,z_2) \wedge x_1 < x_2] \tag{29}$$

When an object is said to be north of another object, it is not important what the coordinates of the objects on the *Z* axis are, as long as the coordinates on the *Y* axis are the same and the coordinates on the *X* axis differ to let one object lie north of another object. This can be formalized as follows:

$$b_1 \text{ north of } b_2 \triangleq \exists_{b_1,b_2 \in \mathcal{B}\,\mathcal{E}} \exists_{x_1,x_2,y_1,y_2,z_1,z_2 \in R}[Coord(b_1) = (x_1,y_1,z_1) \wedge Coord(b_2) = (x_2,y_2,z_2) \wedge y_1 = y_2 \wedge x_1 > x_2] \tag{30}$$

Because the remaining operators related to the other quarters of the compass can be formalized in similar ways, they are not repeated here.

To close this section, Table 3 summarizes the operators that have been introduced to form logical, temporal, and geographical composed expressions. By adding this expressive power to the foundations, service providers can understand how, when, and where services need to be integrated and delivered.

5. Conceptual model of construction operators

To understand possible construction operators to form composed expressions, a cohesive model that brings these formalisms together is needed. Such a cohesive model of construction operations can be used to retrieve all requirements for a domain in which the model is used to provide an ISD. These requirements are formulated by means of the basic and composed expressions. When developing formalisms, textual or graphical techniques (or both) can be used to display such formalisms. Formal specifications use mathematical notations that offer precise syntax and semantics. Textual formalisms, however, may be

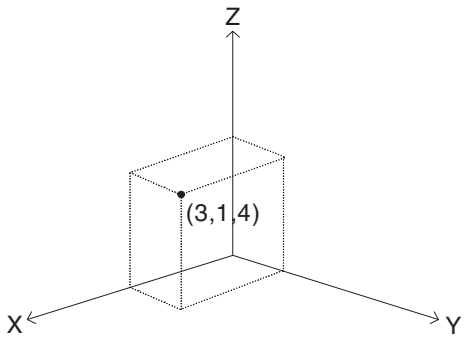

Fig. 5. Three-dimensional Cartesian coordinate system.

complex in nature and not acceptable to many stakeholders [24]. Visual formalisms offer graphical notations with semantics and also offer the possibility to model intuitive and well-organized formalisms. Unfortunately, hand-made diagrams are often hard to read when the complexity of the formalisms increases. Conceptual modeling languages incorporating a (semi-)formal modeling language are, therefore, more suitable for visualizing formal theory. In such a modeling language, the syntax and (in case of fully formal languages) semantics can be formulated coherently in a mathematical language. The Object-Role Modeling (ORM) language is useful for visualizing formalisms because of its formal foundations [26], its demonstrable applications in visualizing formalisms [41] and its long running affiliation with the field of conceptual modeling involving different, often non-technical stakeholders [24]. Fig. 6 depicts a *graphical* representation of the formalisms introduced in Sections 4.1, 4.2, 4.3, and 4.4. This conceptual model has been modeled in ORM and shows the object types, their relationships (fact types), and the constraints on these relationships. The basics of this graphical formal model can be explained as follows. As mentioned in Section 4.2, the start function can be depicted as follows:

$$Start : \mathcal{B}\,\mathcal{E} \rightarrow \mathcal{P}\,\mathcal{E}$$

For example, the expression $Start(b) = p$ states that the start time of a situation expressed by a basic expression $b \in \mathcal{B}\,\mathcal{E}$ is $p \in \mathcal{P}\,\mathcal{E}$. The sets as part of a function are visualized as object types in ORM. If dictated by the nature of the mathematical function, constraints, such as total role constraints or uniqueness constraints, should also be added to a possible ORM model. As such, a uniqueness constraint should be added to the role of object type $\mathcal{B}\,\mathcal{E}$. This ensures that every instance of object type $\mathcal{B}\,\mathcal{E}$ that plays a role in the corresponding fact type is unique. A total role constraint should also be added to object type $\mathcal{B}\,\mathcal{E}$, because the start function prescribes that every instance of object type $\mathcal{B}\,\mathcal{E}$ should play a role in the fact type 'Start'.

Table 3
Construction operators to form composed expressions in SBVR.

Logical	Temporal	Geographical
And	Before	Above
Or	After	Under
Not	Equals	Ahead
	Meets	Behind
	Met by	North of
	Overlaps	South of
	Overlapped by	East of
	During	West of
	Contains	
	Starts	
	Started by	
	Finishes	
	Finished by	

Fig. 6 also shows that two arrows are drawn from the object type 'Expression' to the object types 'BasicExpression' and 'ComposedExpression'. This denotes that there exists a *specialization relationship* between a subtype and a supertype [34], which implies that the instances of the subtype are also instances of the supertype. Each basic expression and composed expression is also an expression. To allow for proper specialization, subtypes have to be defined in terms of one or more of their supertypes. Such a decision criterion is referred to as a *Subtype Defining Rule* [42]. The subtype defining rule for basic expression can be verbalized in SBVR:

> Basic expression = expression *formed by* object type and object instance and object type and relation and object type and object instance

The subtype defining rule for composed expression is:

> Composed expression = expression *composed with* expression

Not every individual textual formal signature of the discussed construction operators discussed here is depicted separately in Fig. 6, because that would make the model too complex and unreadable. Instead, a fact type has been added for each type of construction operator, i.e. a fact type *logical operator*, *temporal operator*, and *geographical operator*.

5.1. Retrieving requirements for integrated service delivery

The ORM model of construction operators presented in Fig. 6 can be used to retrieve requirements to realize ISD. The requirements are formulated by means of the basic and composed expressions that can be retrieved from a populated ORM model. The population of elements in an ORM model, such as instances of object types and fact types can be found by applying the population function introduced in [42]. This function, which can be referred to as *Pop*, can be modeled as follows:

$$Pop : \mathcal{O}\,\mathcal{T} \rightarrow \Omega \tag{31}$$

The set $\mathcal{O}\,\mathcal{T}$ contains object types. Note that fact types are subsets of object types. The set Ω can be referred to as the *Universe of Instances*, abbreviated to UoI. The Universe of Instances contains all possible instances of types found in an ORM model. Assume that c is a composed expression that assembles two basic expressions by means of the 'meets' operator. The composed expression c can be verbalized as follows: Actor John *produces* event residence permit form signed meets architecture expat SOA *orchestrates* event residence permit form signed. This implies that the composed expression c is part of the population of the fact type *temporal operator*, because 'meets' is a temporal operator. Using the population function, this can be translated as: $c \in Pop$(temporal operator).

There are several possibilities to retrieve basic and composed expressions from the ORM model. First, to retrieve all basic expressions, the expression *Pop*(basic expression) should be used. In terms of the expatcase of Section 3, this results in the list of basic expressions shown in Table 2. To retrieve all composed expressions, the expression *Pop*(composed expression) can be used. This results in all composed expressions, no matter what type of construction operator is used to form the composed expression. If a distinction between logical, temporal, or geographical operators is desirable, the expressions *Pop*(logical operator), *Pop*(temporal operator), and *Pop*(geographical operator) can be used to retrieve composed expressions from the ORM model that have been formed using a specific type of operator.

At this point, it can be seen that the five viewpoints on the expat case study as mentioned in Section 3.1 provided the basis to develop an ontology for ISD and subsequently this ontology has been used to make explicit how requirements for ISD can be formulated by making use of basic and composed expressions. The composed expressions on their turn enable to formulate requirements that are based on a combination

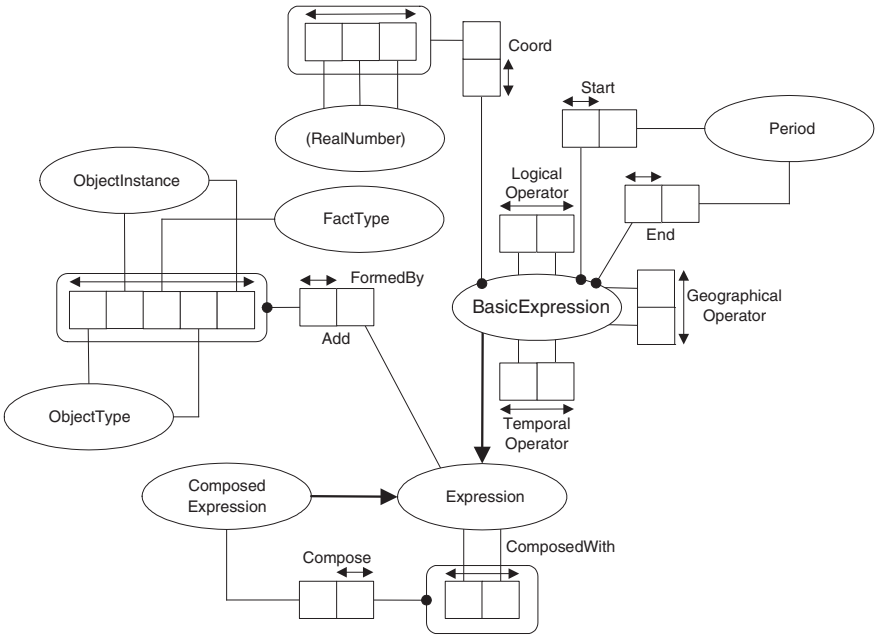

Fig. 6. ORM model of construction operators.

of viewpoints on ISD. Before proceeding to a comparison of our research on providing support for ISD with related research of others, an evaluation of the properties of our ISD language is discussed.

6. Evaluation of integrated service delivery language foundations

Four properties for an ISD language are of importance: expressiveness, tractability, comprehensibility, and compactness. As part of the expat case study it has been evaluated to what extent the proposed foundations for an ISD language possess these properties. The *expressiveness* of the language foundations is determined by the basic expressions and the construction operators that are used to form composed expressions. Basic expressions are based on the case-based ontology for an ISD shown in Fig. 3. In terms of expressiveness, relationships between the concepts shown in this ontology can be expressed. Thus, changes to this ontology affect the expressiveness of the language for ISD. A trade-off has to be made between expressiveness on the one hand, and flexibility, reusability, and understandability on the other hand. Although introducing more concepts and relationships between concepts in the ontology increases its expressiveness, it reduces its flexibility, reusability, and clarity. Furthermore, the introduction of the three different types of construction operators has increased expressiveness, making it possible to verbalize more complex requirements for ISD by composing basic expressions with logical, temporal, and geographical operators.

To further increase the expressiveness, the 'add' function to form basic expressions shown in Section 4.1 and the construction operators to form composed expressions shown in Sections 4.2, 4.3, and 4.4 have been formalized by using first-order predicate logic [36,43] combined with set theory [44,41]. We have chosen for a first-order predicate calculus, because a second-order or a higher order theory admits already a part of the set theory in using its higher order variables [44]. Second-order variables are essentially set variables. Predicate logic is an extension of propositional logic. By means of predicate logic, propositions and predicates can be expressed [43]. A proposition is an assertion or a statement, expressed in a sentence. Propositions can be connected with each other by means of logical symbols called connectives. A predicate designates that a certain object has a specific property. Furthermore, logical operations such as quantification can be performed by means of predicate logic, expressed by a word such as 'all'. Predicate logic has been used to realize an exact construction of the way to form basic expressions and the construction operators to form composed expressions. Syntax can be defined by the use of grammars, but also, typically, by the use of set theory [41]. Set theory has been used to define the way to form basic expressions and the construction operators as a many-sorted algebra with a number of sets, functions, and relations. By formally defining these parts we were able to prevent preliminary flaws during the process of constructing these building blocks of the language for ISD and to formulate in an exact and precise way [45].

The second property, *tractability*, is the ease with which a computer can process a basic or a composed expression [35]. The tractability of an ISD language is thought to be inversely proportional to the amount of time, space (computer memory), and auxiliary knowledge that is needed for a computer to process the language and to let it understand which requirements should be met for the delivery of a certain set of composed services. Expressiveness of the language for ISD reduces tractability, which means that tractability will be reduced when the ISD ontology becomes more complex and when more construction operators are introduced. By using a limited number of necessary concepts and relationships in the ISD ontology andno more than three types of construction operators we have aimed to produce a positive effect on the tractability of a language for ISD based on the foundations proposed in this paper.

Thirdly, the property of *comprehensibility* indicates whether requirements for ISD are understandable by humans when these requirements are formed using the proposed foundations for an ISD language. Comprehensibility is realized by using SBVR as a modeling language to describe requirements for ISD in a semi-natural language, while maintaining the full power of formal logical languages. This improves readability and prevents ambiguous interpretations. Finally, it allows requirements for ISD to be represented in a *compact* way by nesting construction operators and basic expressions in a structure of composed expressions. Service providers are supported in the ISD process by adapting the ontology and using the foundations which creates a shared and unambiguous understanding of the requirements involved in offering an integrated set of services.

7. Related research

In the last ten years, research into the (computer-based) facilitation and realization of ISD has emerged resulting in several studies which,

like this research, also relate to providing support for ISD. One recent study is the nine-stage e-government development model proposed in Ref. [46], in which the integration phase completes an e-government implementation and results in an ISD to citizens. This requires a thorough definition and promotion of concepts, techniques, and instrumentation of the e-government integration. Interactive models are crucial in analyzing the effect of the proposed changes and to reaching a shared agreement [46]. Because the proposed ontology for ISD and the foundations for an ISD language presented in this paper can be regarded as tools that can be used to reach such a common agreement between service providers, it is a response to the challenge issued in [46]. To facilitate the ISD process, ideas to generate trust are considered important [46]. This also applies to the proposed ISD language, because a lack of trust among service providers seriously limits their ability to provide an ISD.

There are many initiatives aimed at realizing ISD by creating interoperability in e-government, the idea being to create generic and reusable domain representations that can be used as a reference for modeling the organizations under study. One of these initiatives is the development of 'interoperability registries', which can contain services in a predefined format by imposing requirements on the infrastructure [47]. These registries can be used to create new electronic services [7]. A portal facilitating ISD by customizing e-services at the design stage of the services is presented in [4]. Run-time adaptation of the services by reacting to changes in the environment during service execution is also possible in the portal. Given a change condition, the portal system identifies migration rules that specify operations and tasks that need to be carried out to achieve run-time adaptation and to generate a new workflow. The use of an ontology is a crucial factor in achieving interoperability [48], because an ontology can give an overall meaning of business process knowledge in an enterprise. The Pi-Calculus theory is introduced as a solution to achieve interoperability between the needs of an enterprise and business process models. This theory can provide semantics between different ontological models that represent different domains [48]. A difference with our research is that the research described in [48] supports the reengineering of business processes instead of ISD.

With regard to modeling requirements for ISD, a variety of initiatives can be identified in the literature that intentionally focuses on the service buyer for which services should be appropriately delivered. In the case of public service providers, the service buyers are citizens. The *life events* approach models services from the citizen's point of view rather than from the perspective of the public service provider and its internal procedures [6]. Life events can be defined as 'any particular situation in the life of a citizen that must be dealt with and requires assistance, support or license from public service providers'. Examples of life events are: paying a fine, getting married, moving, losing a wallet, etc. Taking the life event approach as a starting point, several relevant parts may be incorporated into a language for ISD: the title of a life event, a description, relevant input and output documents when processing the life event, the scope of a life event, its version, and security conditions [6].

A methodology to deliver services to a citizen based on life events that is proposed in [6] consists of four parts. First of all, the problem with which a citizen is faced needs to be identified. Secondly, the input documents required to process an event and the scope need to be determined. Thirdly, all documents that are created when processing a life event need to be identified. Fourthly, all services and service providers that may be aware of the new service need to be updated when necessary. These four parts can be described using the proposed foundations of a language for ISD. Using the ISD ontology shown in Fig. 3 as a starting point, basic and composed expressions can be verbalized for all four parts of the methodology. These expressions then indicate whether a life event has been successfully processed by all the service providers involved, and what should be done if that is not yet the case.

In Ref. [49], user profiles of citizens rather than life events are used to make sure a service matches a citizen's needs and desires. User profiles are suitable data structures that store demographic data, preferences, needs and past behaviors of the corresponding users. A system proposed in [49] that is able to support the matching of supply and demand of public services consists of three main parts. Firstly, citizen handlers help citizens access e-government services. Secondly, service handlers detect the available services that are closest to a citizen's needs and profile. Thirdly, manager handlers support government agency managers in their decision-making activities. The proposed foundations for an ISD language can be used to specify which requirements need to be fulfilled by the specific *handlers* in the system proposed in [49]. Basic and composed expressions can be described that verbalize what requirements citizen handlers need to fulfill to allow citizens access e-government services successfully.

8. Conclusions

The presented research supports the realization of ISD by helping service providers in making the shift from individually supplying common, non-electronic services towards collaboratively supplying integrated, demand-driven, and personalized electronic services. Before service providers can be able to deliver integrated services they need to have an unambiguous and shared understanding as to which requirements should be met to achieve ISD. To meet these demands, foundations for a standard ISD language have been presented in this paper to specify requirements involved in enabling ISD. Based on an ontology and an expat case study we have been able to specify these foundations in the SBVR standard adopted by OMG [19], which consists of basic and composed expressions. A basic expression can be found by navigating the ontology and exactly augmenting two object instances with a relationship (the name of the fact type in the ontological ORM model) between the instances. Two basic expressions can be assembled by means of a construction operator, resulting in a composed expression. Three types of construction operators have been introduced that allow for the logical composition of basic expressions, as well as making it possible to assemble basic expressions that result in temporal and geographical expressions. By adding this expressive power to the foundations, service providers can understand how, when, and where services need to be integrated and delivered. Based on the conceptual ORM model of construction operators and its underlying formalisms, we have shown how service providers can retrieve their requirements in order to realize ISD in practice. Satisfaction of these requirements realizes successful collaboration between service providers, which is a necessity to meet complex customer requirements that demand service providers to work together to provide an integrated service.

References

[1] I. Bertolotti, L. Durante, P. Maggi, R. Sisto, A. Valenzano, Improving the security of industrial networks by means of formal verification, Computer Standards & Interfaces 29 (3) (2007) 387–397.
[2] H. Milward, K. Provan, Managing the hollow state: collaboration and contracting, Public Management Review 5 (1) (2003) 1–18.
[3] H. Chen, Digital government: technologies and practices, Decision Support Systems 34 (3) (2003) 223–227.
[4] S. Chun, V. Atluri, Ontology-based workflow change management for flexible eGovernment service delivery, Proceedings of the 2003 annual national conference on Digital government research, ACM International Conference Proceeding Series, Vol. 130, Digital Government Society of North America, Marina del Rey, CA, 2003, pp. 1–4.
[5] J. Kraaijenbrink, Centralization revisited? Problems on implementing integrated service delivery in The Netherlands, in: R. Traunmüller, K. Lenk (Eds.), Electronic Government, 1st International Conference, EGOV 2002, Aix-en-Provence, France, September 2–September 5, 2002, Proceedings, Springer, Berlin, 2002, pp. 10–17.
[6] L. Álvarez Sabucedo, L. Anido Rifón, R. Pérez, J. Santos Gago, Providing standard-oriented data models and interfaces to eGovernment services: a semantic-driven approach, Computer Standards & Interfaces 31 (5) (2009) 1014–1027.
[7] R. Feenstra, M. Janssen, R. Wagenaar, Evaluating web service composition methods: the need for including multi-actor elements, The Electronic Journal of e-Government 5 (2) (2007) 153–164.

[8] J. Yang, M. Papazoglou, Service components for managing the life-cycle of service compositions, Information Systems 29 (2) (2004) 97–125.
[9] C. Ferris, J. Farrell, What are web services? Communications of the ACM 46 (6) (2003) 31–34.
[10] J. Park, A high performance backoff protocol for fast execution of composite web services, Computers & Industrial Engineering 51 (1) (2006) 14–25.
[11] J. Ralyté, M. Jeusfeld, P. Backlund, H. Kühn, N. Arni-Bloch, A knowledge-based approach to manage information systems interoperability, Information Systems 33 (7–8) (2008) 754–784.
[12] F. Lampathaki, S. Mouzakitis, G. Gionis, Y. Charalabidis, D. Askounis, Business to business interoperability: a current review of XML data integration standards, Computer Standards & Interfaces 31 (6) (2009) 1045–1055.
[13] S. Overbeek, A. Klievink, M. Janssen, A flexible, event-driven, service-oriented architecture for orchestrating service delivery, IEEE Intelligent Systems 24 (5) (2009) 31–41.
[14] W. Ke, K. Wei, Successful e-government in Singapore, Communications of the ACM 47 (6) (2004) 95–99.
[15] J. Dang, A. Hedayati, K. Hampel, C. Toklu, An ontological framework for adaptive medical workflow, Journal of Biomedical Informatics 41 (5) (2008) 829–836.
[16] S.-T. Yuan, M.-R. Lu, An value-centric event driven model and architecture: a case study of adaptive complement of SOA for distributed care service delivery, Expert Systems with Applications 36 (2) (2009) 3671–3694.
[17] A.-M. Sourouni, G. Kourlimpinis, S. Mouzakitis, D. Askounis, Towards the government transformation: an ontology-based government knowledge repository, Computer Standards & Interfaces 32 (1–2) (2010) 44–53.
[18] C. Dias, Corporate portals: a literature review of a new concept in information management, International Journal of Information Management 21 (4) (2001) 269–287.
[19] OMG, Semantics of business vocabulary and business rules (SBVR), v1.0, OMG available specification formal/2008-01-02, Object Management Group, Needham, MA, USA, 2008.
[20] H. Fernández, E. Palacios-González, V. Garcia-Díaz, B. G-Bustelo, O. Martínez, J. Lovelle, SBPMN—an easier business process modeling notation for business users, Computer Standards & Interfaces 32 (1–2) (2010) 18–28.
[21] M. Jarrar, R. Meersman, Ontology engineering—the Dogma approach, in: T. Dillon, E. Chang, R. Meersman, K. Sycara (Eds.), Advances in Web Semantics I, Springer, Berlin, 2008, pp. 7–34.
[22] IEEE, IEEE recommended practice for architectural description of software-intensive systems, Tech. Rep. IEEE Std 1471-2000, IEEE Computer Society, New York, NY, USA, 2000.
[23] Q. Sheng, B. Benatallah, Z. Maamar, User-centric services provisioning in wireless environments, Communications of the ACM 51 (11) (2008) 130–135.
[24] E. Falkenberg, Concepts for modelling information, in: G. Nijssen (Ed.), Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems, Freudenstadt, North-Holland Publishing, Amsterdam, 1976, pp. 95–109.
[25] M. Jarrar, Mapping ORM into the SHOIN/OWL description logic—towards a methodological and expressive graphical notation for ontology, in: R. Meersman, Z. Tari, P. Herrero (Eds.), On the move to meaningful internet systems 2007: OTM 2007 Workshops, Vilamoura, Portugal, November 25–30, 2007, Proceedings, Part I, Lecture Notes in Computer Science, Vilamoura, Portugal, Vol. 4805, Springer, Berlin, 2007, pp. 729–741.
[26] T. Halpin, Information Modeling and Relational Databases, from Conceptual Analysis to Logical Design, Morgan Kaufmann, San Mateo, CA, 2001.
[27] A.T. Hofstede, T.V.D. Weide, Expressiveness in conceptual data modelling, Data & Knowledge Engineering 10 (1) (1993) 65–100.
[28] S. Overbeek, P.V. Bommel, H. Proper, D. Rijsenbrij, Visualizing formalisms with ORM models, in: R. Meersman, Z. Tari, P. Herrero (Eds.), On the move to meaningful internet systems 2007: OTM 2007 Workshops, Vilamoura, Portugal, November 25–30, 2007, Proceedings, Part I, Lecture Notes in Computer Science, Vol. 4805, Springer, Berlin, 2007, pp. 709–718.
[29] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara, OWL-S: semantic markup for web services, W3C member submission, Web-Ontology Working Group, 2004.
[30] ISO, Software engineering—product quality—part 1: quality model, International Standard ISO 18629-1, International Organization for Standardization, Geneva, Switzerland, 2004.
[31] S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuinness, J. Su, S. Tabet, Semantic web services ontology (SWSO): Version 1.0, W3C member submission, Semantic Web Services Initiative, 2005.
[32] T. Vitvar, J. Kopecky, J. Viskova, A. Mocan, M. Kerrigan, D. Fensel, Semantic web services architecture with lightweight descriptions of services, in: M. Zelkowitz (Ed.), Advances in Computers: Social Networking and The Web, Vol. 76, Academic Press, Burlington, MA, USA, 2009, pp. 177–224.
[33] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana, Business process execution language for web services, International Standard Version 1.1, BEA Systems, IBM, and Microsoft, 2003.
[34] A.T. Hofstede, H. Proper, T.V.D. Weide, Formal definition of a conceptual language for the description and manipulation of information models, Information Systems 18 (7) (1993) 489–523.
[35] B. Wondergem, P.V. Bommel, T.V.D. Weide, Combining Boolean logic and linguistic structure, Information and Software Technology 43 (1) (2001) 53–59.
[36] J. Benthem, H. Ditmarsch, J. Ketting, J. Lodder, W. Meyer-Viol, Logica voor Informatica, 3rd Edition Pearson Education Benelux, Amsterdam, 2003 in Dutch.
[37] T. Halpin, Temporal modeling and ORM, in: R. Meersman, Z. Tari, P. Herrero (Eds.), On the move to meaningful internet systems 2008: OTM 2008 Workshops, Monterrey, Mexico, November 9–14, 2008, Proceedings, Lecture Notes in Computer Science, Monterrey, Mexico, Vol. 5333, Springer, Berlin, 2008, pp. 688–698.
[38] D. Ferrés, A. Ageno, H. Rodríguez, The GeoTALP-IR system at GeoCLEF 2005: experiments using a QA-based IR system, linguistic analysis, and a geographical thesaurus, in: C. Peters (Ed.), Accessing Multilingual Information Repositories, Springer, Berlin, 2005, pp. 947–955.
[39] M. Thuering, J. Hannemann, J. Haake, Hypermedia and cognition: designing for comprehension, Communications of the ACM 38 (8) (1995) 57–66.
[40] J. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (11) (1983) 832–843.
[41] A.T. Hofstede, H. Proper, How to formalize it? Information and Software Technology 40 (10) (1998) 519–540.
[42] P.V. Bommel, A.T. Hofstede, T.V.D. Weide, Semantics and verification of object-role models, Information Systems 16 (5) (1991) 471–495.
[43] A. Hamilton, Logic for Mathematicians, Cambridge University Press, Cambridge, UK, EU, 1978.
[44] A. Levy, Basic Set Theory, Dover Publications, Mineola, NY, USA, 2002.
[45] A. Hall, Seven myths of formal methods, IEEE Software 7 (5) (1990) 11–19.
[46] B. Zarei, A. Ghapanchi, B. Sattary, Toward national e-government development models for developing countries: a nine-stage model, The International Information & Library Review 40 (3) (2008) 199–207.
[47] Y. Charalabidis, F. Lampathaki, J. Psarras, Combination of interoperability registries with process and data management tools for governmental services transformation, in: R. Sprague Jr. (Ed.), 42st Hawaii International Conference on Systems Science (HICSS-42 2009), Proceedings, 5–8 January 2009, Waikoloa, Big Island, HI, USA, Vol. 130, IEEE Computer Society, Los Alamitos, CA, 2009, pp. 1–10.
[48] V. Damjanović, Semantic reengineering of business processes, Information Systems 35 (4) (2010) 496–504.
[49] P. De Meo, G. Quattrone, D. Ursino, A decision support system for designing new services tailored to citizen profiles in a complex and distributed e-government scenario, Data & Knowledge Engineering 67 (1) (2008) 161–184.

Sietse Overbeek is an Assistant Professor in the field of (web-based) information systems and service engineering at Delft University of Technology. His main research interests include (web-based) information systems, conceptual modeling, ontology modeling, service-oriented and event-driven architectures, formal methods, and cognitive matchmaking. Sietse received his Master's degree from the Radboud University Nijmegen, the Netherlands in April 2005, and received his Ph.D. from the same university in April 2009. Sietse has (co-)authored numerous journal papers, conference publications, and books. He has co-chaired a number of conferences and is a member of several editorial boards and programme committees. For more information, please see: www.sietseoverbeek.nl.

Marijn Janssen is Director of the interdisciplinary Systems Engineering, Policy Analyses and Management Master programme, the Compliance Management Master, runs a MBA module on Business processes and IT, and is an Associate Professor within the Information and Communication Technology section of the Technology, Policy and Management Faculty of Delft University of Technology. His research interests are in the field of coordinating public-private service networks. He serves on several editorial boards and is involved in the organization of a number of conferences. He published over 200 refereed publications. For more information, please see: www.tbm.tudelft.nl/marijnj.

Patrick van Bommel is an Assistant Professor at the Radboud University Nijmegen. His main research interests include information modeling, databases and document systems. The focus of his research is: foundations of models and their transformations in order to establish theoretical properties and practical tools. He received his Master's degree in Computer Science in 1990, and the degree of Ph.D. in Mathematics and Computer Science, from the Radboud University Nijmegen, the Netherlands in 1995. For more information, please see: www.cs.ru.nl/~pvb.